AmebaDocs

Release 0.0.1

Ameba IoT

Feb 10, 2022

OPEN SOURCE SDKS

We have supported 3 unique developing platforms, Realtek **Arduino SDK**, **Standard SDK** and **MicroPython SDK**. All Getting-started guides, Examples, Tutorials, API reference and Datasheet can be found under the respective link below:

CHAPTER

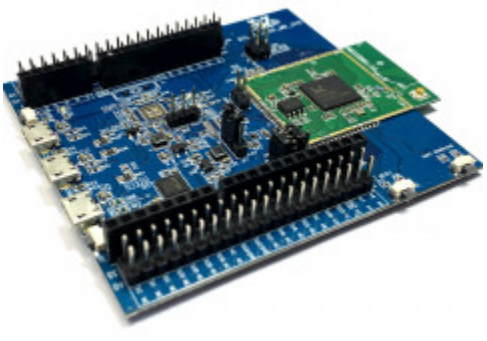
ONE

ARDUINO SDK

Arduino SDK is supported on the following 3 develpoment boards listed below:

1.1 AMB21 (RTL8722DM)

Welcome to AMB21 Arduino online documentation.



1.1.1 Getting Started

Ameba ARDUINO: Getting Started with AMB21

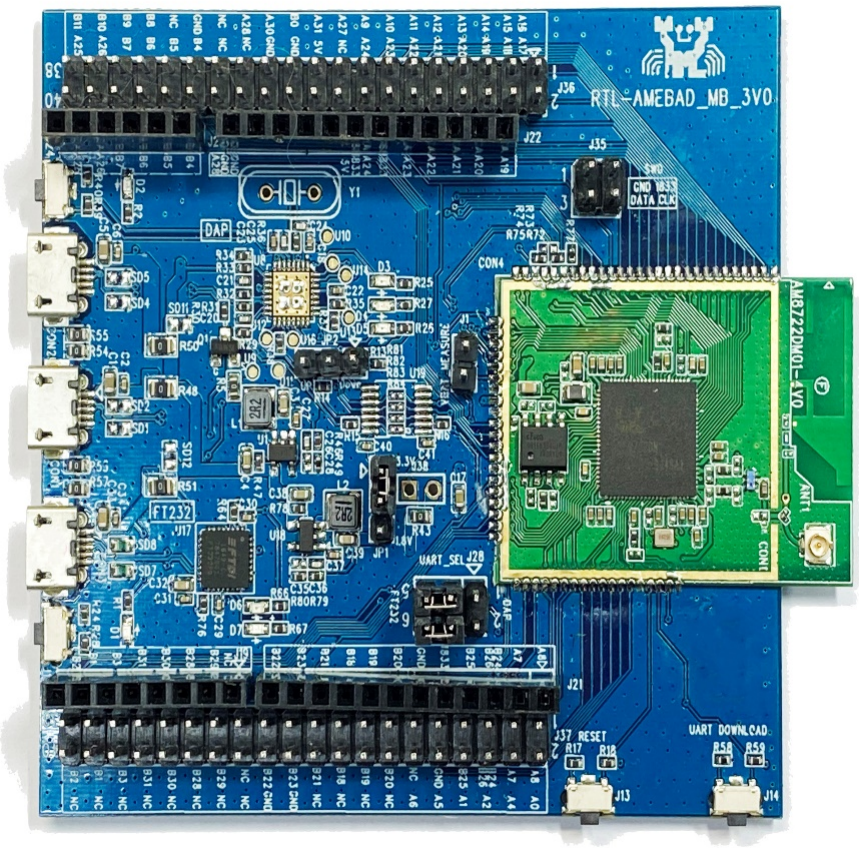
Required Environment

AMB21 board currently supports Windows OS 32-bits and 64-bits (WIN7/8/10), Linux OS (Ubuntu 18 LTS/20 LTS/latest) and macOS operating systems. Please use the latest OS version to have the best experiences. In this documentation, please use the latest version Arduino IDE (at least version 1.8.12).

Introduction to AmebaD[AMB21 / AMB22]

Ameba is an easy-to-program platform for developing all kind of IoT applications. AmebaD is equipped with various peripheral interfaces, including WiFi, GPIO INT, I2C, UART, SPI, PWM, ADC. Through these interfaces, AmebaD can connect with electronic components such as LED, switches, manometer, hygrometer, PM2.5 dust sensors, …etc.

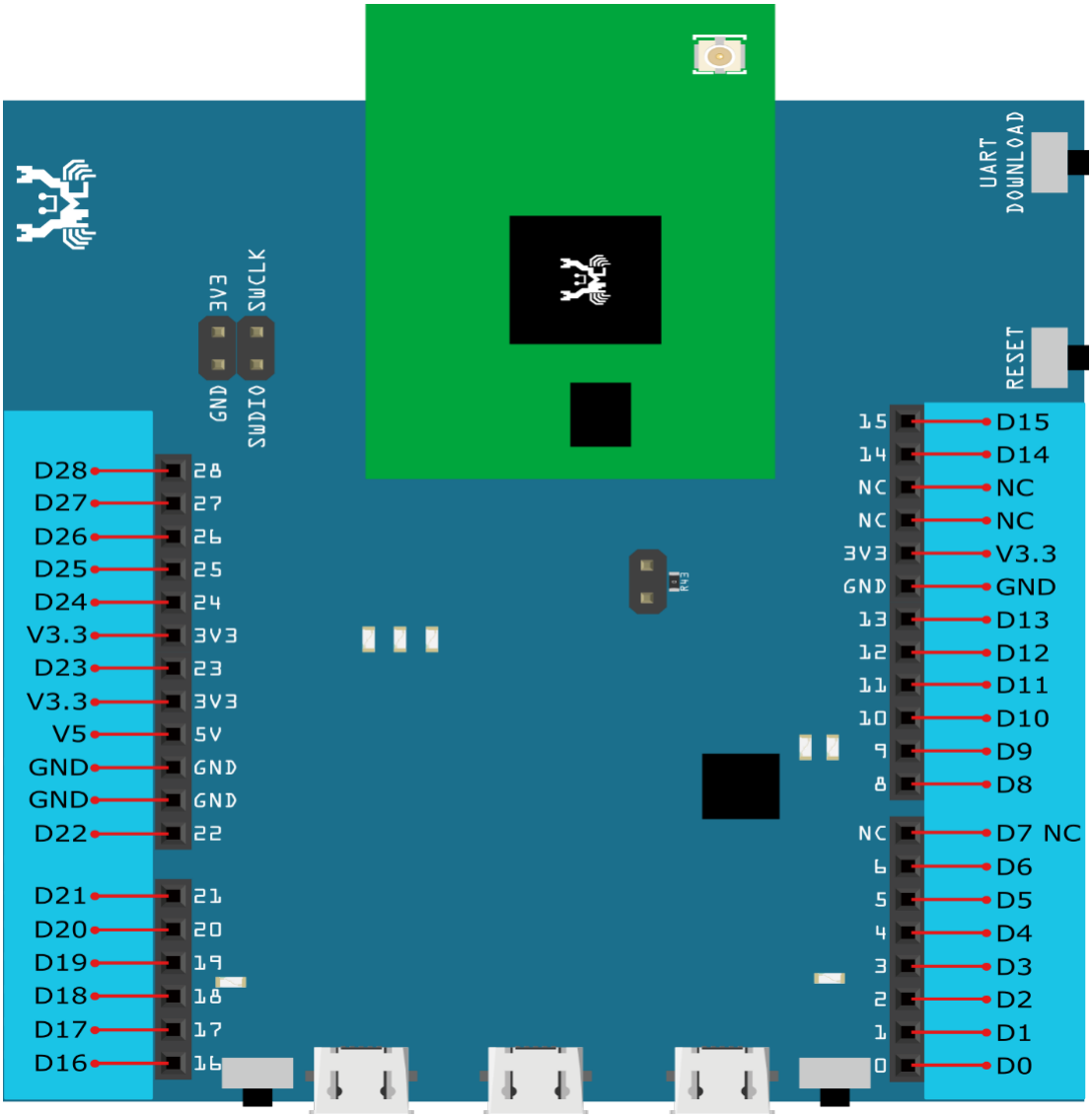
The collected data can be uploaded via WiFi and be utilized by applications on smart devices to realize IoT implementation.



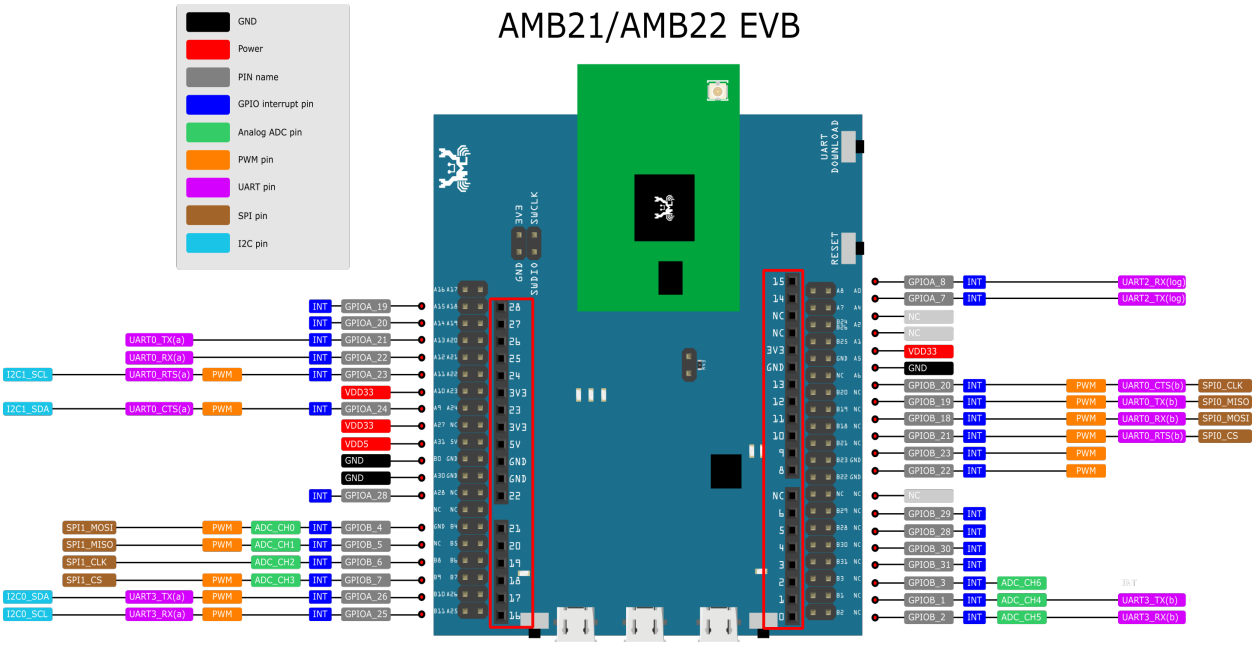
AMB21 and Arduino Uno have similar size, as shown in the above figure, and the pins on AMB21 are compatible with Arduino Uno.

AMB21 uses Micro USB to supply power, which is common in many smart devices.

Please refer to the following figure and table for the pin diagram and function of AMB21.


UART
DOWNLOAD
RESET
3V3
SWCLK
GND
SWDIO
D28 28
D27 27
D26 26
D25 25
D24 24
V3.3 3V3
D23 23
V3.3 3V3
V5 5V
GND GND
GND GND
D22 22
D21 21
D20 20
D19 19
D18 18
D17 17
D16 16
15 D15
14 D14
NC NC
NC NC
3V3 V3.3
GND GND
13 D13
12 D12
11 D11
10 D10
9 D9
8 D8
NC D7 NC
6 D6
5 D5
4 D4
3 D3
2 D2
1 D1
0 D0

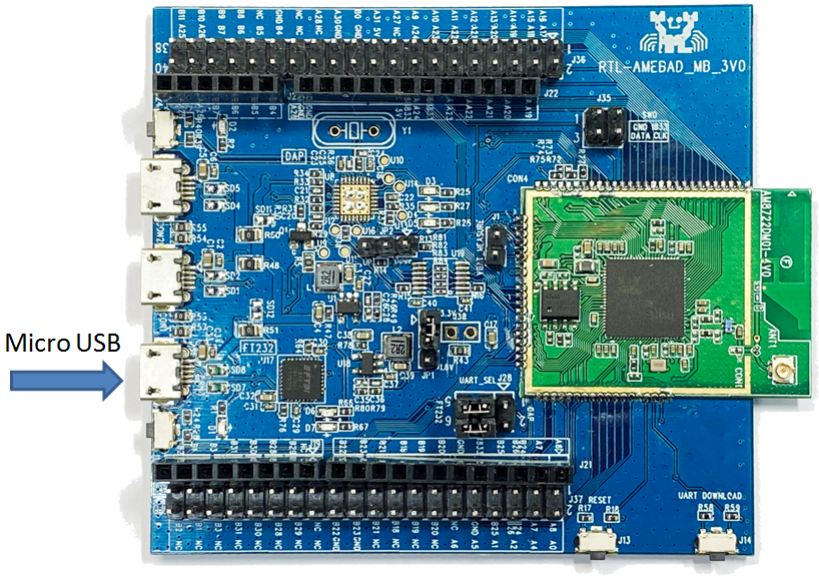
#	PIN name	GPIO	ADC	PWM	UART	SPI	I2C
D00	GPIOB_2	✓	ADC5		UART3_RX(b)		
D01	GPIOB_1	✓	ADC4		UART3_TX(b)		
D02	GPIOB_3	✓	ADC6				
D03	GPIOB_31	✓					
D04	GPIOB_30	✓					
D05	GPIOB_28	✓					
D06	GPIOB_29	✓					
D07	NC						
D08	GPIOB_22	✓		✓			
D09	GPIOB_23	✓		✓			
D10	GPIOB_21	✓		✓	UART0_RTS(b)	SPI0_CS	
D11	GPIOB_18	✓		✓	UART0_RX(b)	SPI0_MOSI	
D12	GPIOB_19	✓		✓	UART0_TX(b)	SPI0_MISO	
D13	GPIOB_20	✓		✓	UART0_CTS(b)	SPI0_CLK	
D14	GPIOA_7	✓			UART2_TX(log)		
D15	GPIOA_8	✓			UART2_RX(log)		
D16	GPIOA_25	✓		✓	UART3_RX(a)	I2C0_SCL	
D17	GPIOA_26	✓		✓	UART3_TX(a)	I2C0_SDA	
D18	GPIOB_7	✓	ADC3	✓		SPI1_CS	
D19	GPIOB_6	✓	ADC2			SPI1_CLK	
D20	GPIOB_5	✓	ADC1	✓		SPI1_MISO	
D21	GPIOB_4	✓	ADC0	✓		SPI1_MOSI	
D22	GPIOA_28	✓					
D23	GPIOA_24	✓		✓	UART0_CTS(a)	I2C1_SDA	
D24	GPIOA_23	✓		✓	UART0_RTS(a)	I2C1_SCL	
D25	GPIOA_22	✓			UART0_RX(a)		
D26	GPIOA_21	✓			UART0_TX(a)		
D27	GPIOA_20	✓					
D28	GPIOA_19	✓					



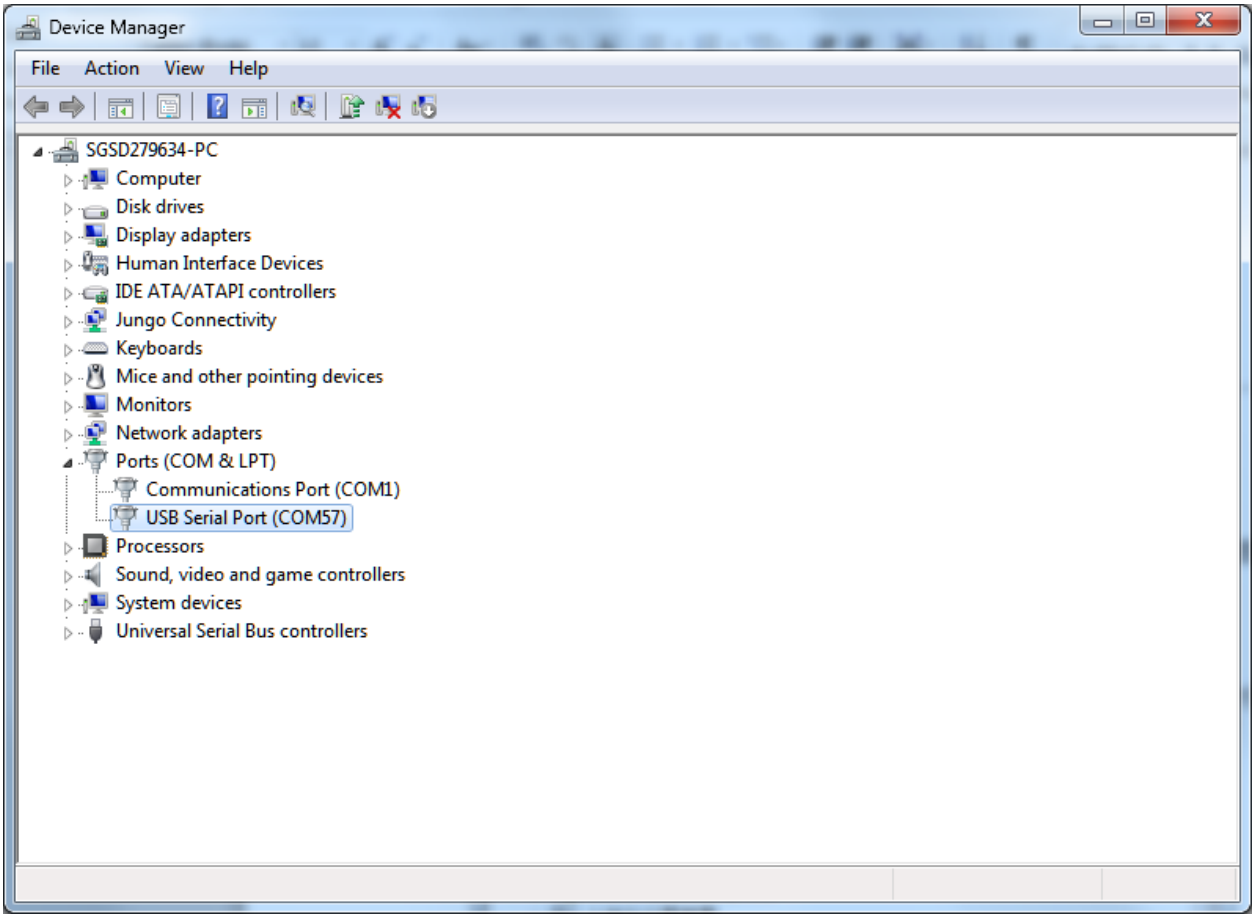
Setting up Development Environment

Step 1. Installing the Driver

First, connect AMB21 to the computer via Micro USB:



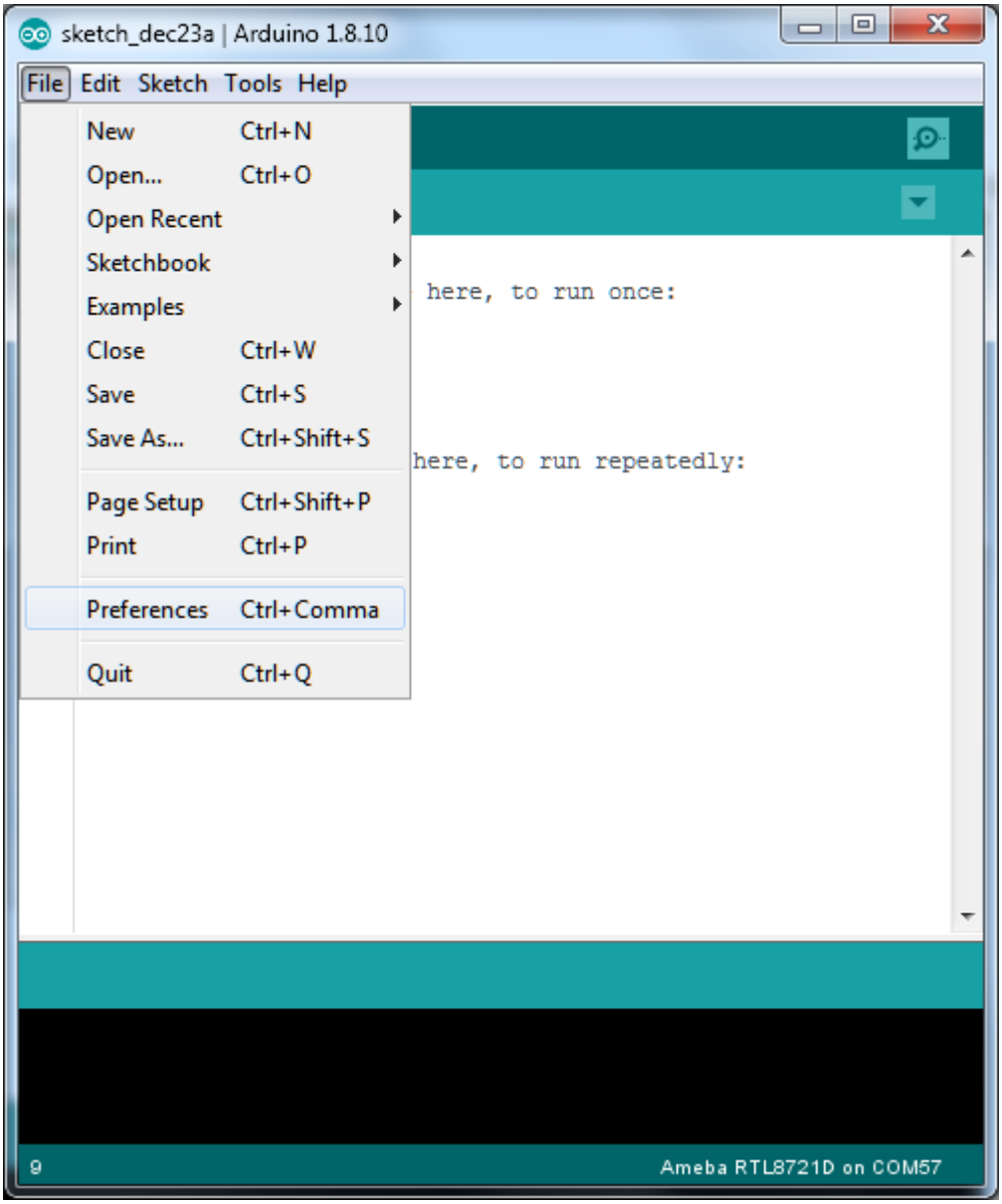
If this is the first time you connect AMB21 to your computer, the USB driver for AMB21 will be automatic installed.
If you have driver issue of connect board to your computer please go to here for USB driver.
You can check the COM port number in Device Manager of your computer:



Step 2. Set up Arduino IDE

From version 1.6.5, Arduino IDE supports third-party hardware. Therefore, we can use Arduino IDE to develop applications on AMB21, and the examples of Arduino can run on AMB21 too. Arduino IDE can be downloaded in the Arduino website.

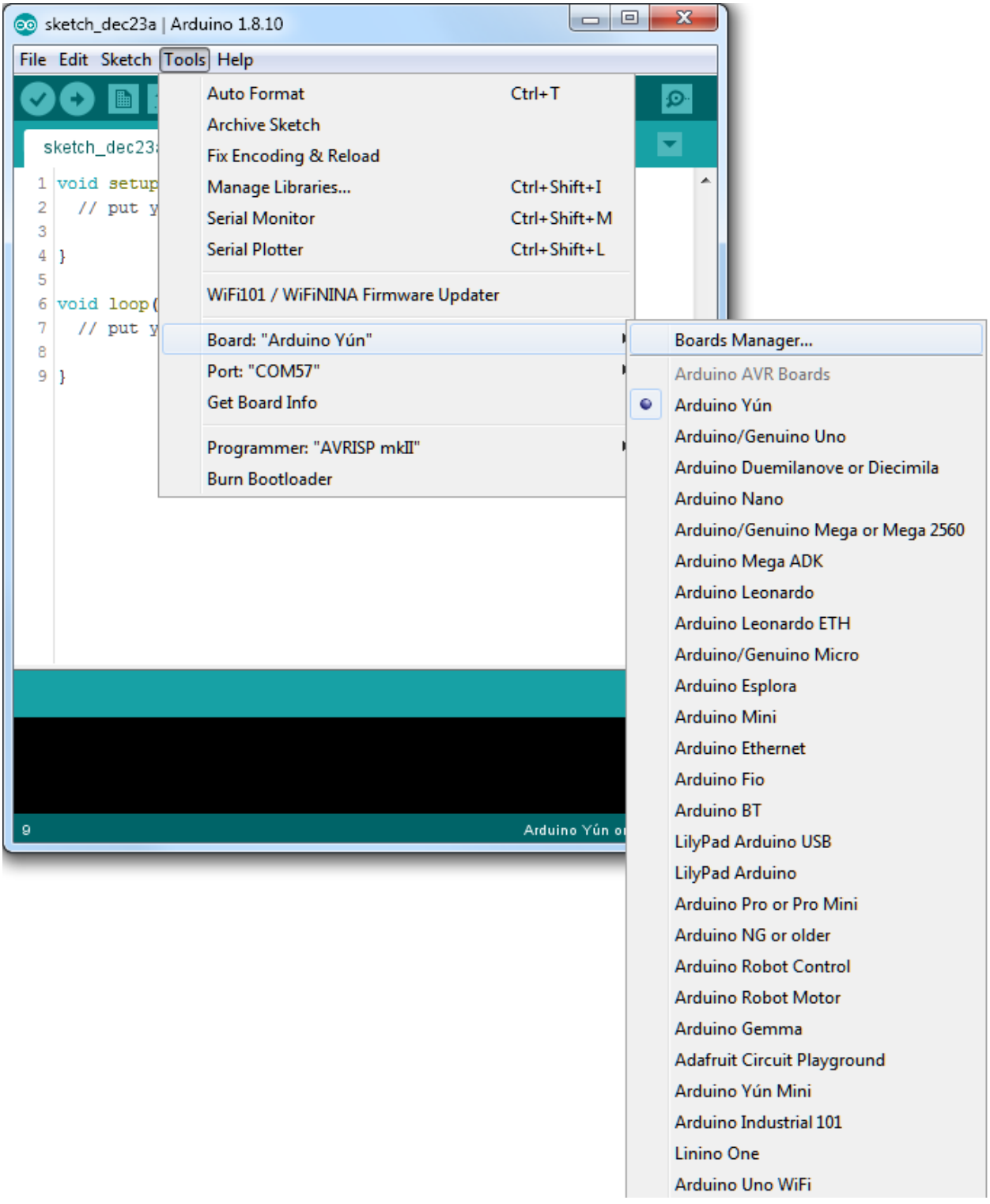
When the installation is finished, open Arduino IDE. To set up AMB21 correctly in Arduino IDE, go to *"File" -> "Preferences"*.



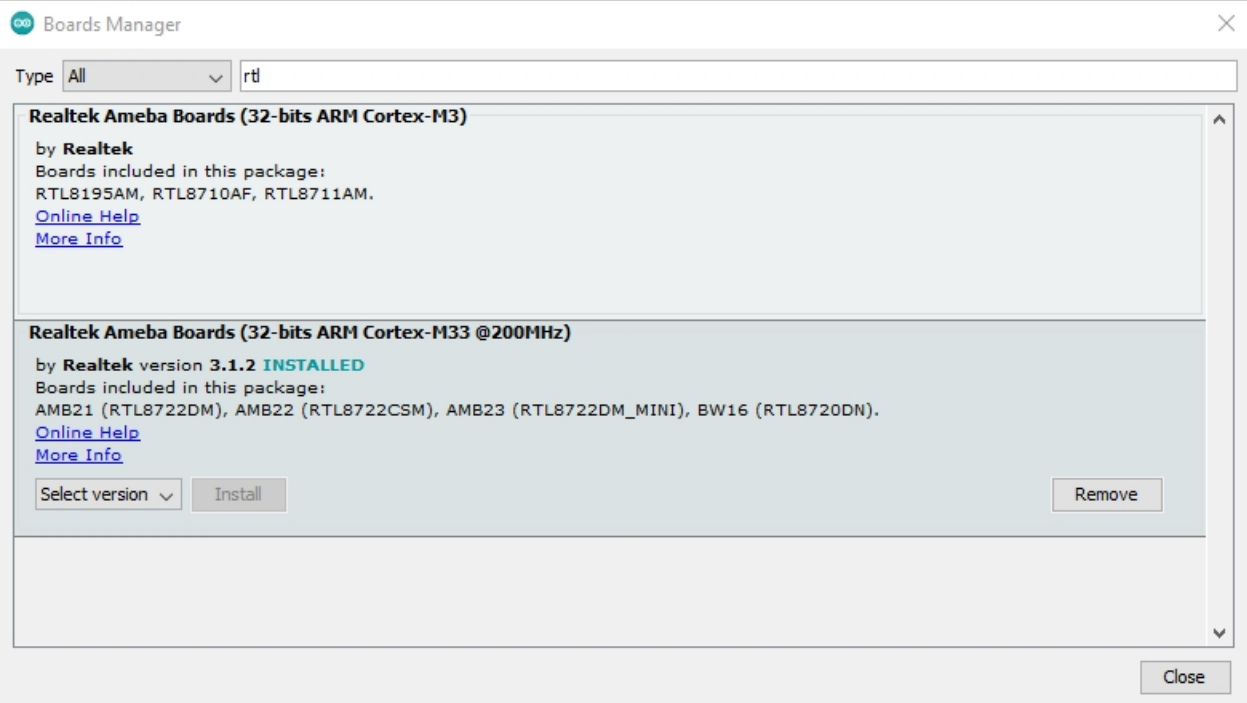
And paste the following URL into *"Additional Boards Manager URLs"* field:

```
https://github.com/ambiot/ambd_arduino/raw/master/Arduino_package/package_realtek.com_
↪amebad_index.json
```

Next, go to *"Tools"* -> *"Board"* -> *"Boards Manager"*:



The *"Boards Manager"* requires about 10~20 seconds to refresh all hardware files (if the network is in bad condition, it may take longer). Every time the new hardware is connected, we need to reopen the Board Manager. So, we close the *"Boards Manager"*, and then open it again. Find *"Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)"* in the list, click *"Install"*, then the Arduino IDE starts to download required files for AMB21.



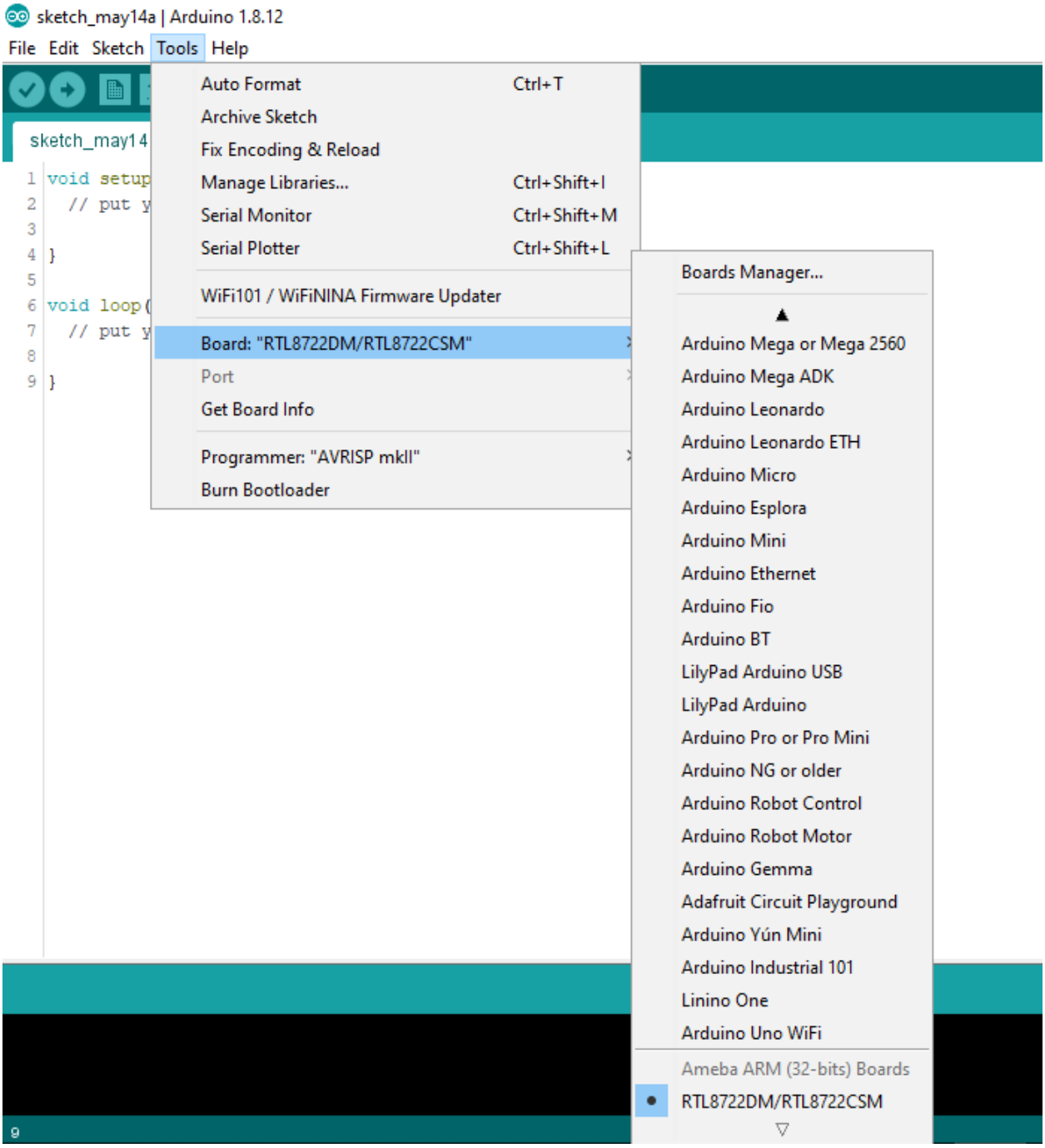
If you are facing GitHub downloading issue, please refer to the following link at Download/Software Development Kit. There are 3 sections:

1. "AmebaD_Arduino_patch1_SDK", please select at least 1 of the SDKs. There are 5 latest released SDK options.
2. "AmebaD_Arduino_patch2_Tools", please select according to your operation system. There are Windows, Linux and MacOS.
3. "AmebaD_Arduino_Source_Code", this section is optional download only wants to refer the latest source code.

   Download the files selected, then unzip (patch1 and patch2 are compulsory). There are "Install.doc"/"Install.pdf" for you to refer installation steps. According to your system, please run the installation tool in the "Offline_SDK_installation_tool" folder.

   After the installation tool running successfully, you may open Arduino IDE and proceed to *"Tools"* -> *"Board"* -> *"Boards Manager…"*. Try to find *"Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)"* in the list, click *"Install"*, then the Arduino IDE starts to download required files for AMB21.

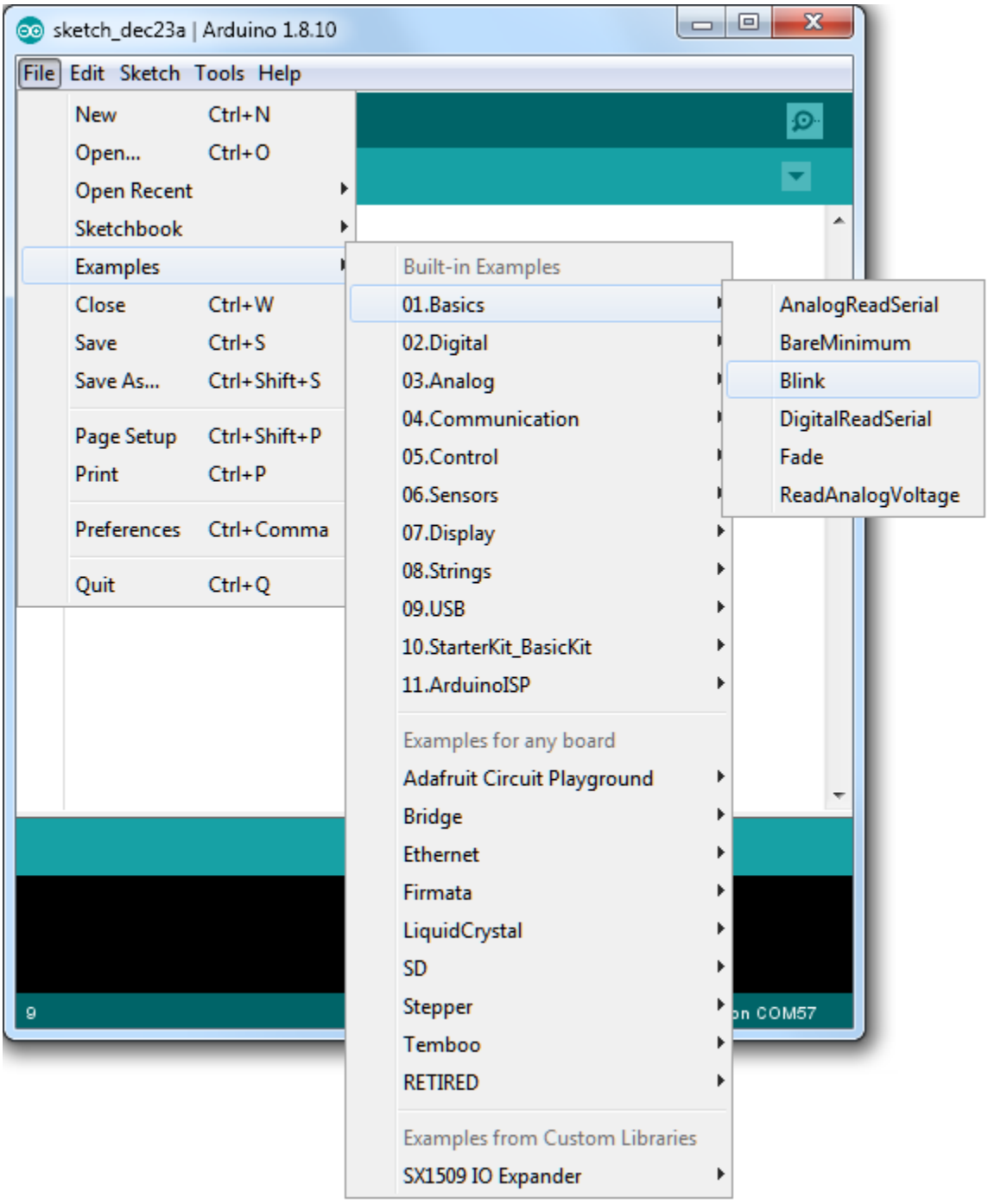
Finally, we select AMB21 as current connected board in *"Tools"* -> *"Board"* -> *"Ameba ARM (32-bits) Boards"* -> *"AMB21"*.


sketch_may14a | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_may14
1 void setup
2 // put y
3
4 }
5
6 void loop(
7 // put y
8
9 }
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "RTL8722DM/RTL8722CSM"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi
Ameba ARM (32-bits) Boards
RTL8722DM/RTL8722CSM
9

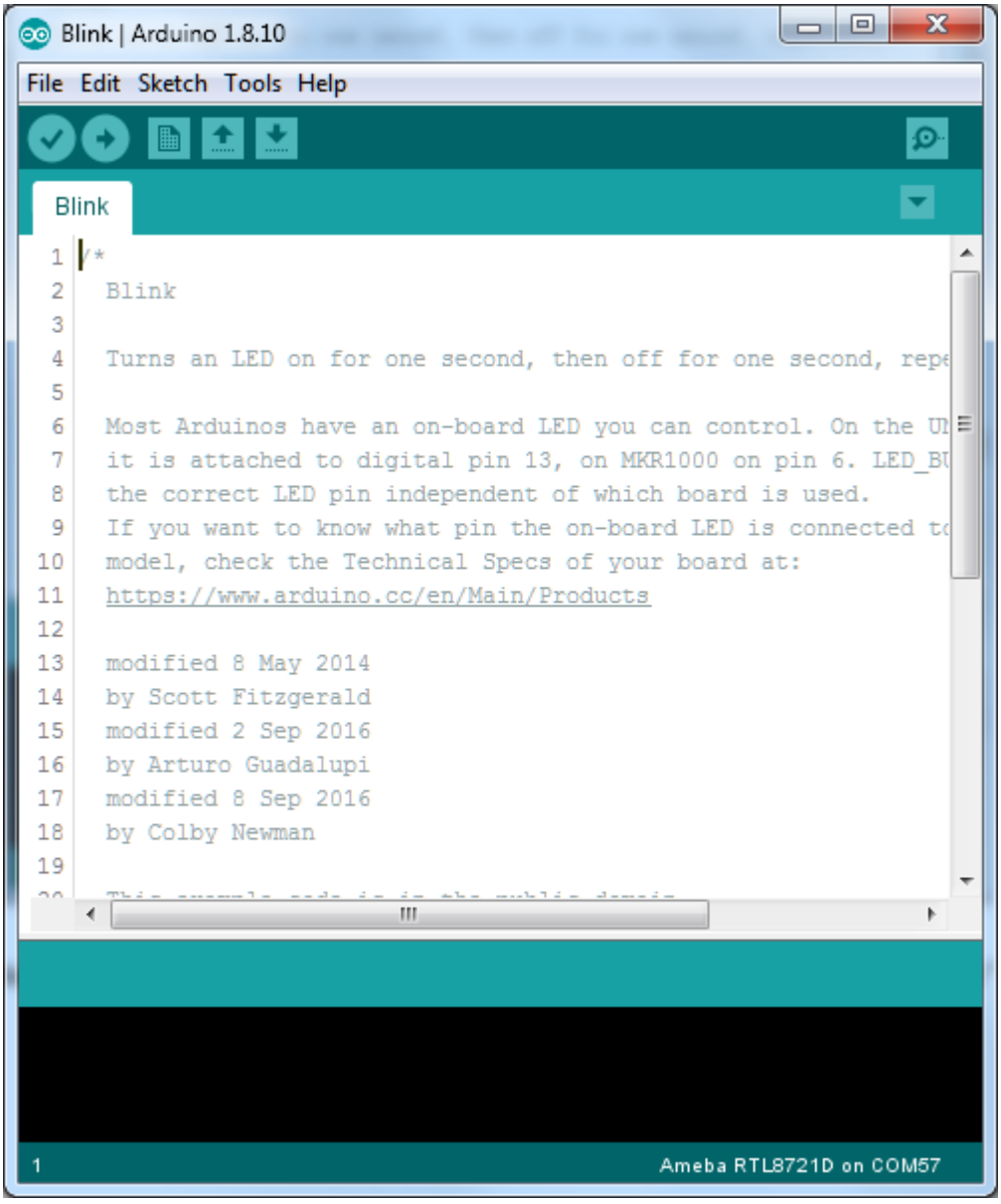
Try the First Example

Step 1. Compile & Upload

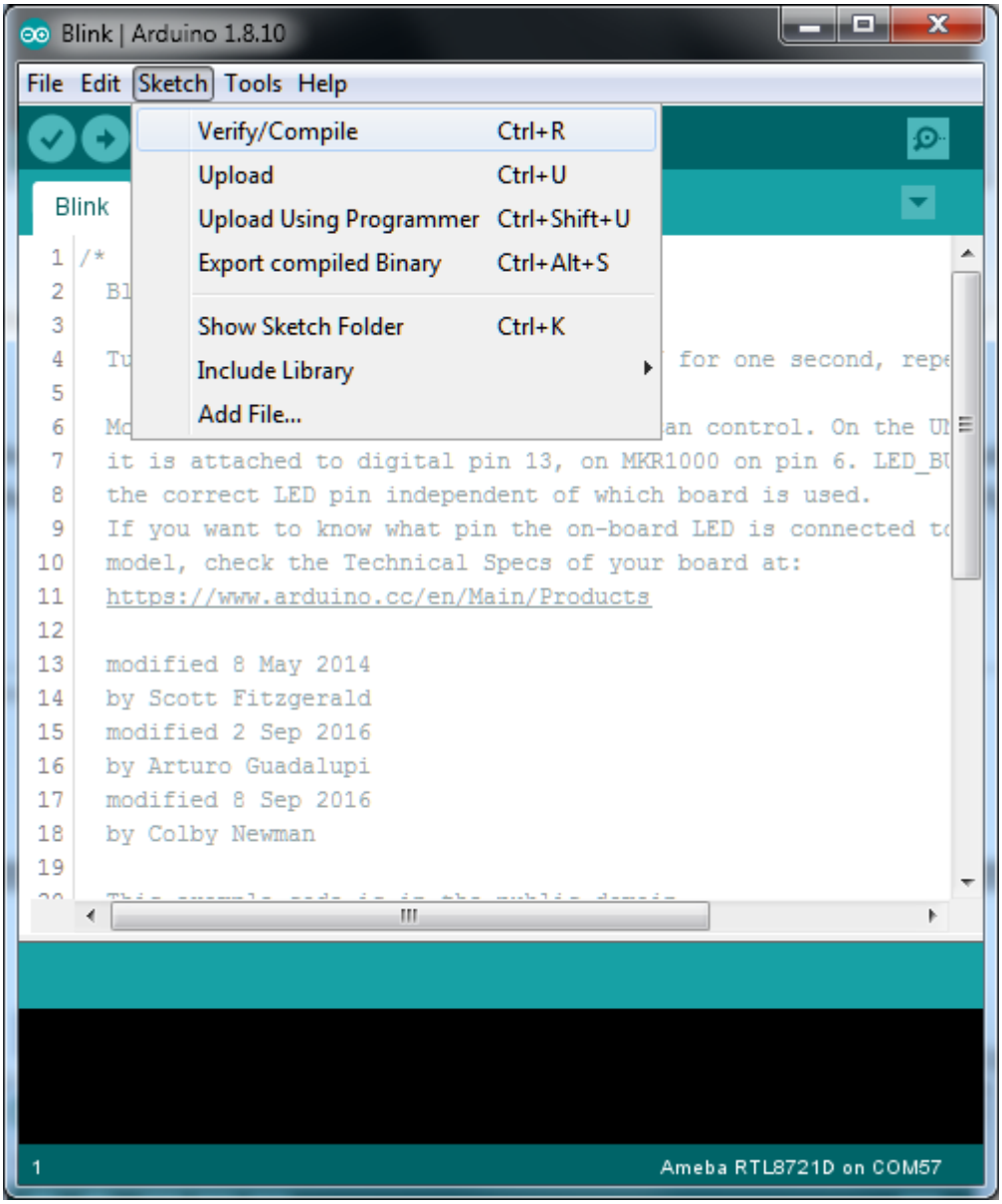
Arduino IDE provides many built-in examples, which can be compiled, uploaded and run directly on the boards. Here, we take the "Blink" example as the first try.
Open *"File" -> "Examples" -> "01.Basics" -> "Blink"*:



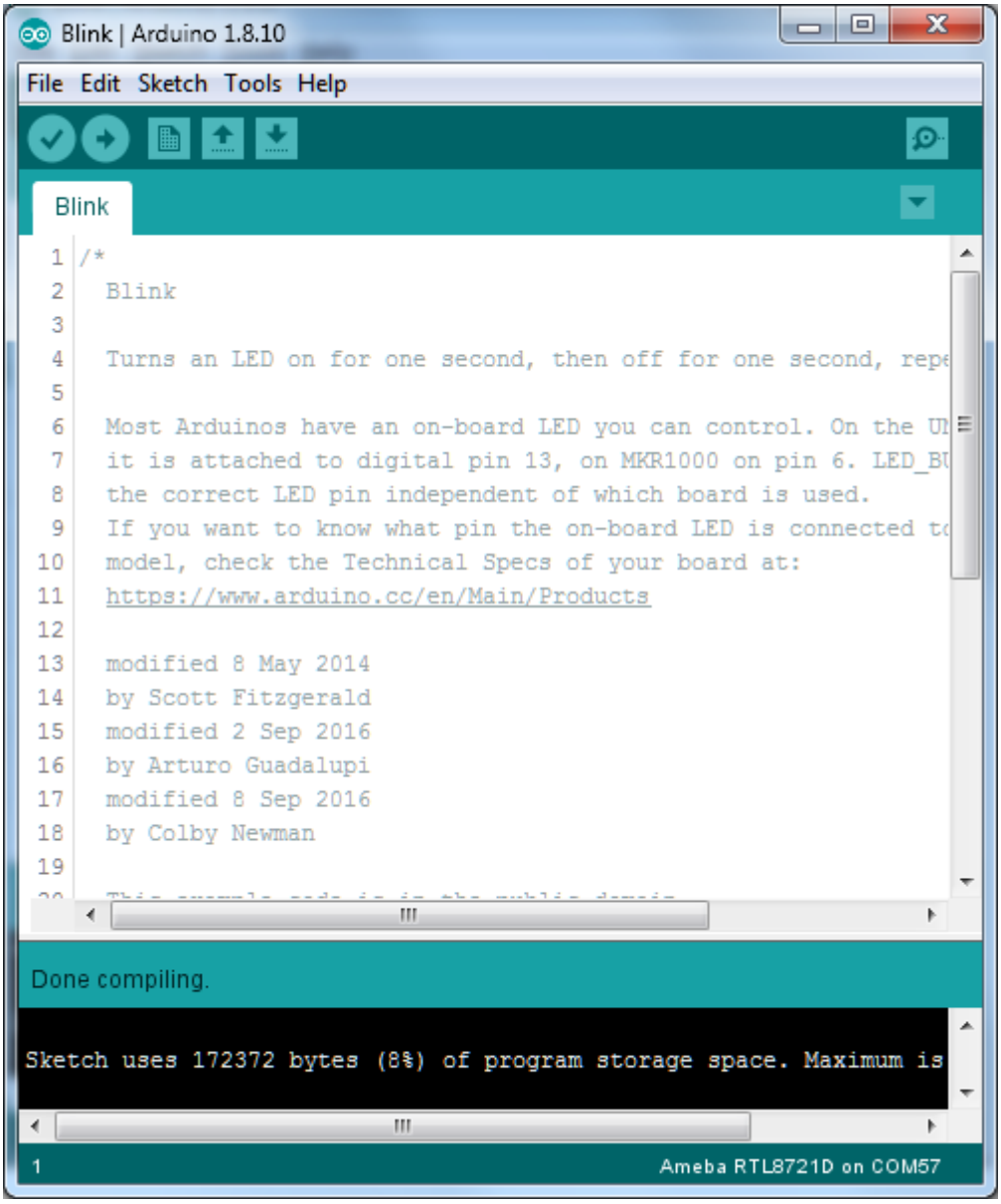
Arduino IDE opens a new window with the complete sample code.



Next, we compile the sample code directly; click *"Sketch" -> "Verify/Compile"*



Arduino IDE prints the compiling messages in the bottom area of the IDE window. When the compilation is finished, you will get the message similar to the following figure:

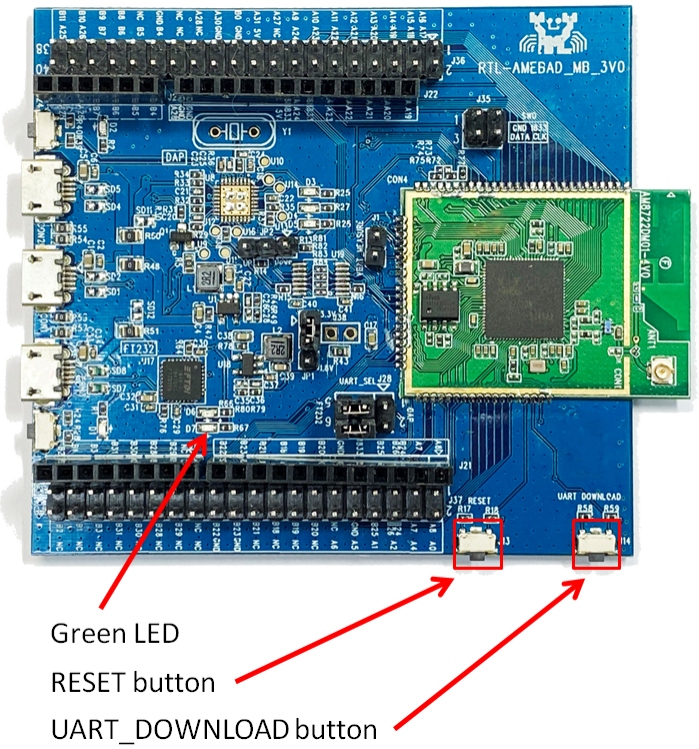


Afterwards, we will upload the compiled code to AMB21.
Please make sure AMB21 is connected to your computer, then click *"Sketch" -> "Upload"*.

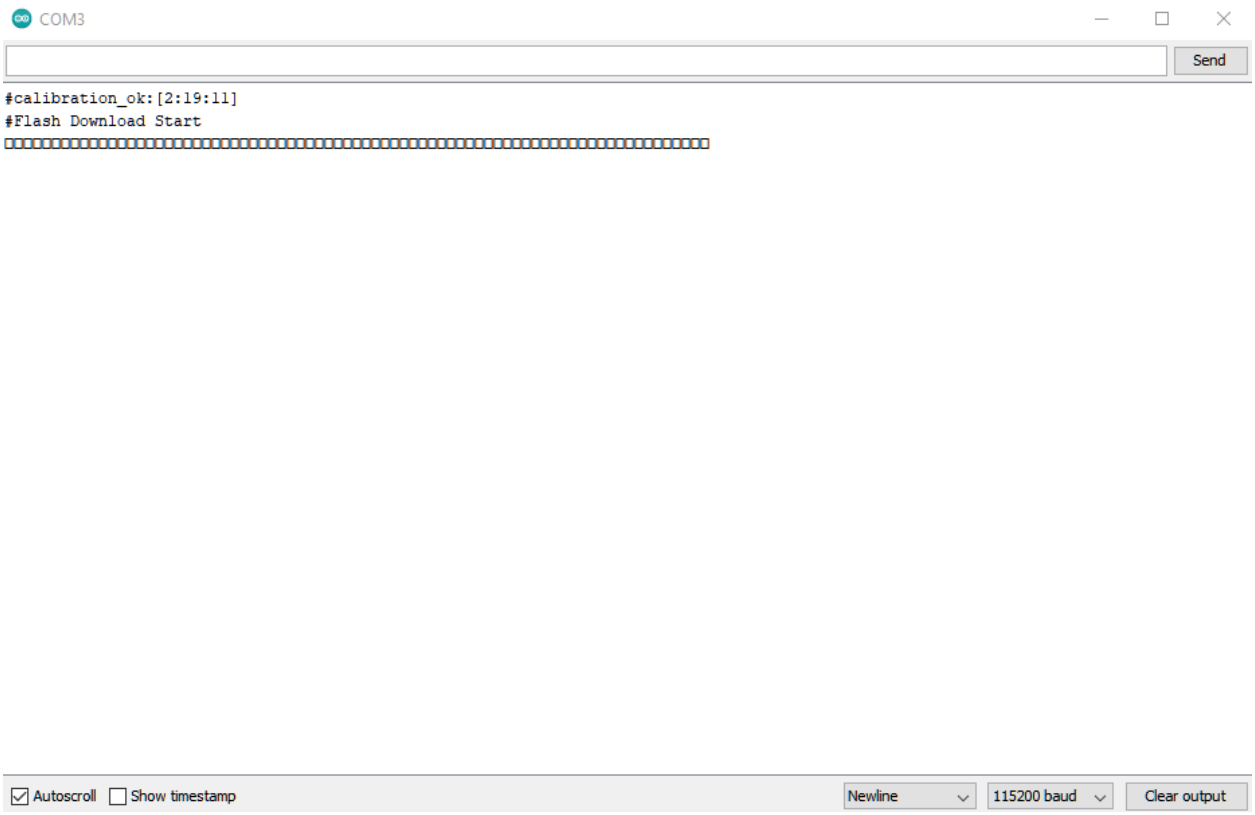
The Arduino IDE will compile first then upload. During the uploading process, users are required to enter the upload mode of the board. Arduino IDE will wait 5s for DEV board to enter the upload mode.

```
Uploading...
Sketch uses 172372 bytes (8%) of program storage space. Maximum is 2097152 bytes.
        1 file(s) copied.
Please enter the upload mode (wait 5s)
    05
    04
    03
    02
    01
```

To enter the upload mode, first press and hold the *UART_DOWNLOAD* button, then press the *RESET* button. If success, you should see the LED flashing on the DEV board.



It is optional for users to check if the board entered the upload mode. Open serial monitor/terminal and look for "#Flash Download Start". Note, it is normal that some serial terminals may show unknown characters as following picture.



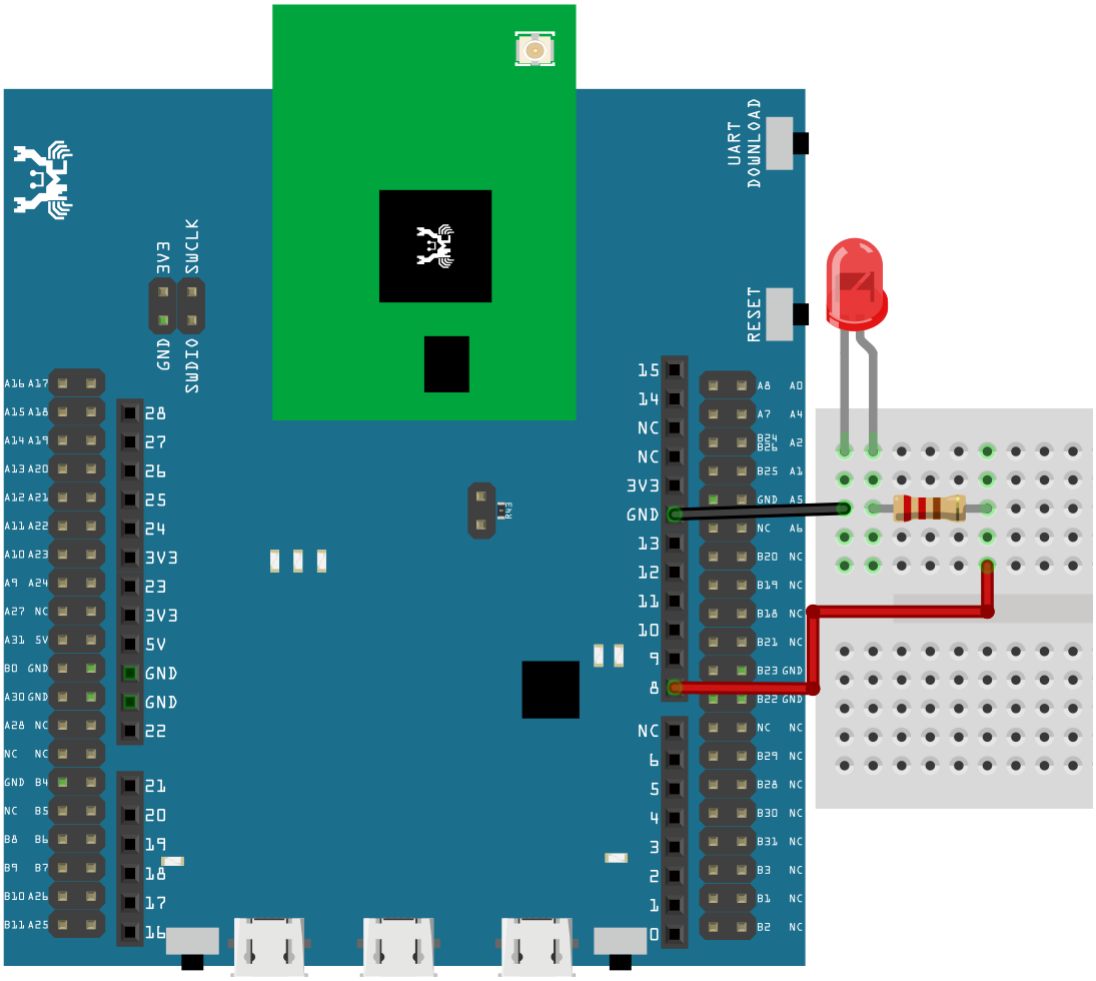
Again, during the uploading procedure the IDE prints messages. Uploading procedure takes considerably longer time (about 30 seconds to 1 minute). When upload completed, the "Done uploading" message is printed.

```
Done uploading.
Sketch uses 172372 bytes (8%) of program storage space. Maximum is 2097152 bytes.
        1 file(s) copied.
Please enter the upload mode (wait 5s)
    05
    04
    03
    02
    01
1       COM57       SUCCESS
    Upload km0_boot_all done.
1       COM57       SUCCESS
    Upload km4_boot_all done.
1       COM57       SUCCESS
    Upload km0_km4_image2 done.
```

Step 2.Run the Blink example

In each example, Arduino not only provides sample code, but also detailed documentation, including wiring diagram, sample code explanation, technical details, …etc. These examples can be directly used on AMB21.
So, we find the detailed information of the Blink example.

In short, this example makes LED blinks, and it uses GPIO pin 08 (refer to the pin diagram `D08`). Then we connect the LED and resistance as the following figure:
NOTE: In an LED, the longer pin is the positive pole, and shorter pin is the negative pole. So we connect the longer pin to `D08`, and connect the shorter pin to `GND`. In addition, please use a resister with suitable resistance in series between LED and GND to protect LED



Finally, press the *RESET* button, and you can see the *LED* blinking.

(End)

Note: If you face any issue, please refer to the FAQ and Trouble shooting sections on ../../support/index page.

1.1.2 Release History

Version 3.1.2 - 2021/12/28

- Feature:
  - Update SimpleWebServerWiFi example
  - Support BW16
- API Updates:
  - Update Wlan related naming from "AmebaWiFi" become "WiFi"
  - Update RTC library for minor bug fix
- Misc:
  - Update all Fritzing files for new name updates
  - AMB21, AMB22, AMB23, and BW16

Version 3.1.1 - 2021/12/25

- Feature:
  - Add BLE HID and examples
  - BLEHIDGamepad, BLEHIDKeyboard, and BLEHIDMouse
  - Update PowerSave examples
  - Support RTL8722DM MINI and RTL8720DN/BW16
  - Enable LwIP hostname edit
- API Updates:
  - Update API for PowerSave
  - Update ameba_d_tools 1.0.7 for all 3 platforms
  - Support RTL8720DN/BW16 and RTL8722DM MINI
  - Add more Aon wake up pins
  - Update API for IR
  - Removed requirement to define both IR TX and RX pins in IRDevice::begin
  - Removed previous limit on number of time durations IRDevice::send can accept
  - Update GPIO Int
  - Enable INPUT_IRQ_CHANGE
  - Add definition inside wiring_constants.h and wiring_digital.c, also complete the TODO part for attachInterrupt() as well
  - Update UART, for RTL8720DN/BW16 not showing log issue

  - Fix wrong attribute permissions for characteristic CCCD descriptor. Remove unused variable warnings
  - Update GTimer, for the internal timer ID validation test
  - Updated SPI connection for RTL8720DN/BW16
  - Update Google_Cloud_IoT example with new Google TLS cert
  - Update Analog Pin remove A0 and A1
  - Update Platform.txt for Windows OS with User Name having a space in between
  - Update all libs
- Misc:
  - Update AmebaEink.zip, SPI connection for RTL8720DN/BW16
  - Add Autoflash_patch folder
  - Update the Fritzing of RTL8720DN/BW16, remove A0 and A1

Version 3.1.0 - 2021/11/05

- Feature:
  - Support board RTL8720DN(BW16)
  - Add WiFiControlCar example
  - Add Arduboy zip library
  - Add WPA3 support
  - Add Amebad_HMI_MQTT zip library
  - Add support for IPV6 wiht 4 examples
  - WLAN lib update
  - Minor bug fix
- API Updates:
  - Support Microsoft Azure IoT cloud
  - Enable “strnlen” from rom
  - Add “#define yield” for compilation
  - Update PubSubClient lib
  - Update APIs for RTL8720DN(BW16) (SPI, I2C, Fatfs, Audiocodec and UART
  - Update jtag enable functions
  - Update wifi security option
  - Remove the unused libs lib_wifi_fw.a lib_wifi_ucps_fw.a
  - Update watchdog
  - Update AudioCodec
  - Pin mapping updates
  - Remove unused marcos
  - RTL8720DN(BW16) related naming update for all examples

  - Update PowerSave
- Misc
  - Add RTL8720DN_BW16 frizting folder
  - Move RTL8720DN_BW16 frizting files to correct folder
  - Rename folder name to short the length of path
  - Add Offline_SDK_installation_tool (Windows, Linux and MacOS)
  - Update linux tools for compatibility issue
  - Update RTL8722DM MINI and RTL8720DN(BW16) Fritzing and Pinmux
  - Update ameba_d_tools V1.0.6
  - Add Image_Releated folder
  - Correct the core from Cortex-M4 to Cortex-M33

Version 3.0.11 - 2021/10/26

- Feature:
  - Add example, FatfsSDIO - Read and open HTML file from SD card
- API Updates:
  - RTL8720DN/BW16 related compatibility update for all examples
- Misc
  - Update RTL8722DM MINI and RTL8720DN Fritzing and Pinmux

Version 3.0.10 - 2021/09/22

- Feature:
  - Add AudioCodec wav examples
- API Updates:
  - Pin mapping updates for RTL8722DM MINI
  - Remove unused marcos
  - Update platform.txt for bin files process
  - rollback for “wifi.h” update
  - Minor bug fix patch

Version 3.0.9 - 2021/09/13

- API Updates:
  - Pin mapping updates
  - Remove unused marcos
  - “wifi.h” related files change to “Amebawifi.h”

Version 3.0.8 - 2021/05/06

- Feature:
  - Add RTL8722DM_mini board
  - Add fatfs for SD card
  - Add AudioCodec
  - Add TensorFlow lite support with examples
  - Add zip libraries for TensorFlow lite support
  - Update SDK for supporting Arduino IDE 2.0
  - Update wlan lib
- API Updates:
  - Update zip libraries of Eink
  - ADC updates, Change calculation method to use EFUSE calibration parameters and SDK formula to improve accuracy
  - writing_analog updates, minor bug fix and support for mini board
  - SPI updates, minor bug fix and support for mini board
  - I2S updates, minor bug fix and support for mini board
  - IRDevice updates, minor bug fix

Version 3.0.7 - 2020/11/19

- Feature:
  - Add AmebaIRDevice example IRSendSONY
  - Update Ameba Arduino IRDevice API
  - Update Ameba Arduino SSL related API
  - Update Ameba Arduino Wlan API to support static IP function

---

Version 3.0.6 - 2020/10/28

- Feature:
  - Add Ameba RTC support
  - Add AmebaRTC example RTC and RTCAlarm
  - Add Ameba Watchdog support
  - Add AmebaWatchdog example WatchdogTimer
  - Update Ameba BLE support
  - Add AmebaBLE example BLEUartService, DHT_over_BLEUart
  - Update Ameba Wlan library
  - Update Ameba Wlan SDK structure, add AP mode hidden SSID support

---

Version 3.0.5 - 2020/09/09

- Feature:
  - Build in tool updates V1.0.4
  - Add zip lib AmebaEink
  - Add AmebaEink example EinkDisplayImage, EinkDisplayQR, and EinkDisplayText
  - Add google cloud examples
  - Update Amazon AWS related examples
  - Add power save support
  - Add AmebaPowerSave example TicklessMode, DeepSleepMode, DeepSleep_DHT_LCD_Example, and DeepSleep_DHT_Eink_Example

---

Version 3.0.4 - 2020/07/27

- Feature:
  - Update BLE library. Add example BLEBatteryClient and BLEWIfiConfig
  - Update from polarssl to mbedtls 2.4.0

Version 3.0.3 - 2020/07/03

- Feature:
  - Build in Image tool updates V1.0.3
  - Upload log clean up

Version 3.0.2 - 2020/06/30

- Feature:
  - Windows, Linux and macOS X support
  - Build in Image tool updates

Version 3.0.1 - 2020/05/15

- Feature:
  - Official release of AmebaD Arduino SDK
  - warning cleaning
  - I2C lib updates

Version 3.0.0 - 2020/05/01

- Feature:

- Support Boards Manager and Arduino IDE development
- WiFi scan AP, connect to AP, TCP Server/Client, including 5G
- Bluetooth, BLE
- GPIO digital in/out and interrupt
- ADC analog in/out (0 ~ 3.3V)
- PWM getting analog results with digital means
- SPI master and slave mode
- UART 1 for log, 2 for customize usage
- I2C master mode

1.1.3 Peripherals & Examples

Basic Examples

AMB21 (RTL8722DM/CSM) Supported ARDUINO built-in example table

There are many built-in examples in Arduino. In the table below, we list all examples that are compatible with Ameba.

Please refer to the following link to set up Ameba for Arduino IDE.
https://www.amebaiot.com/en/amebad-arduino-getting-started/
Please refer to the following link for Arduino built-in example details.
https://www.arduino.cc/en/Tutorial/BuiltInExamples/

Category	Name	Comment	Remarks
01. Basics	AnalogReadSerial	Connect potentiometer to 3.3V.	ADC pin options A0, A
	BareMinimum		
	Blink	Pin LED_BUILTIN sets to pin D8.	
	DigitalReadSerial		
	Fade		
	ReadAnalogVoltage		ADC pin options A0, A
02. Digital	BlinkWithoutDelay	Pin LED_BUILTIN sets to pin D8.	
	Button	Connect LED to pin D13.	
	Debounce	Connect LED to pin D13.	
	Digital InputPullup	Connect LED to pin D13.	
	StateChange Detection	Connect LED to pin D13.	
	toneKeyboard		
	toneMelody		
	toneMultiple		
	tonePitch Follower		
03. Analog	Analog InOutSerial		ADC pin options A0, A
	AnalogInput	Connect LED to pin D13.	ADC pin options A0, A

Table 1 – continued from previous page

Category	Name	Comment	Remarks
	Analog Write Mega		
	Calibration	Connect another LED to pin D13.	ADC pin options A0, A
	Fading		
	Smoothing		ADC pin options A0, A
04. Communication	ASCIITable		
	Dimmer		
	Graph	Connect potentiometer to 3.3V.	ADC pin options A0, A
	Midi	Use Serial1 and pin D26, or use Serial2 and pin D17.	
	MultiSerial		
	PhysicalPixel		
	Read ASCIIString		
	SerialCallResponse		ADC pin options A0, A
	SerialCallResponse ASCII		ADC pin options A0, A
	SerialEvent		
	Serial Passthrough		Serial options, Serial1 o
	Virtual ColorMixer		ADC pin options A0, A
05. Control	Arrays	Use pins D1, D2, D3, D4, D5, D6.	
	ForLoop Iteration	Use pins D1, D2, D3, D4, D5, D6.	
	IfStatement Conditional		ADC pin options A0, A
	switchCase		ADC pin options A0, A
	switchCase2	Use pins D1, D2, D3, D4, D5, D6.	
	While Statement Conditional	Connect another LED to pin D13.	ADC pin options A0, A
06. Display	barGraph	Use another pin to replace pin D7.	ADC pin options A0, A
	RowColumn Scanning		ADC pin options A0, A
07. Strings	Character Analysis		
	StringAddition Operator		
	StringAppend Operator		
	String CaseChanges		
	String Characters		
	StringComparision Operators		ADC pin options A0, A
	StringIndexOf		
	StringLength		
	StringLengthTrim		
	StringReplace		
	StringStartsWith EndsWith		
	StringSubstring		
	StringToInt		

Network Examples

BLE - BLE Battery Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery service is set up on the Ameba Bluetooth stack. A mobile phone is used to connect to the Ameba peripheral device and read the battery data.

Procedure

Ensure that the following Bluetooth apps are installed on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.

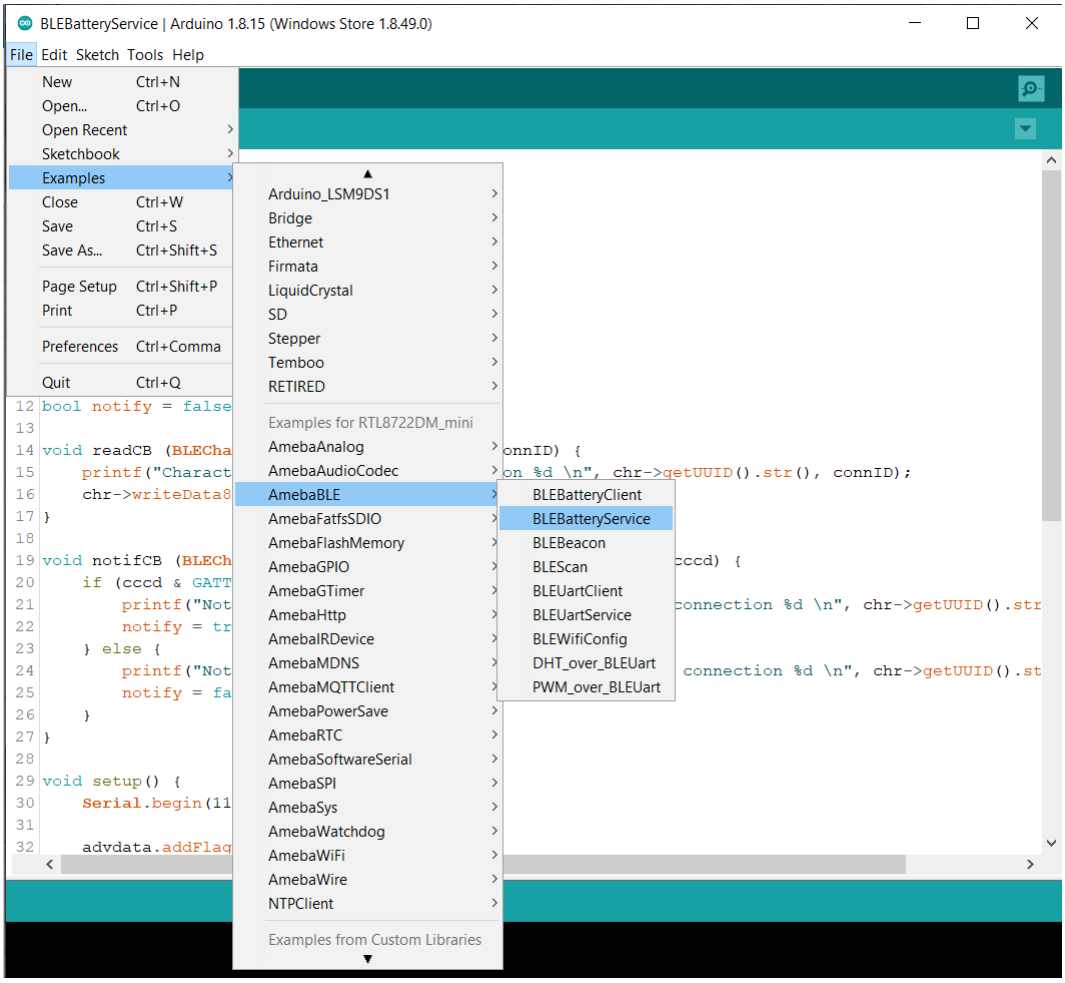
The recommended application is nRF connect, and is available at the links below:

- Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS : https://apps.apple.com/us/app/nrf-connect/id1054362403

LightBlue is an alternative application that can also be used, but has less features:

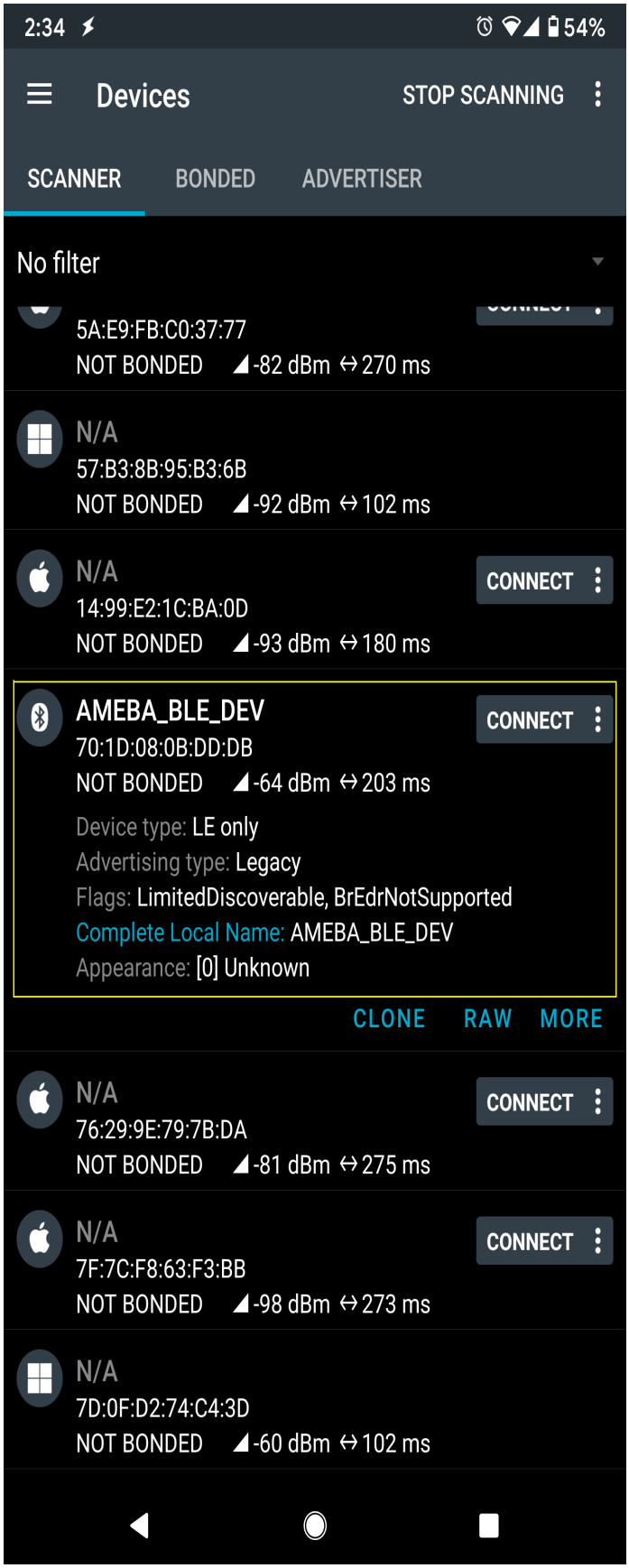
- Android: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS : https://apps.apple.com/us/app/lightblue/id557428110

Open the example, "`Files` -> "`Examples` -> "`AmebaBLE` -> "`BLEBatteryService`"

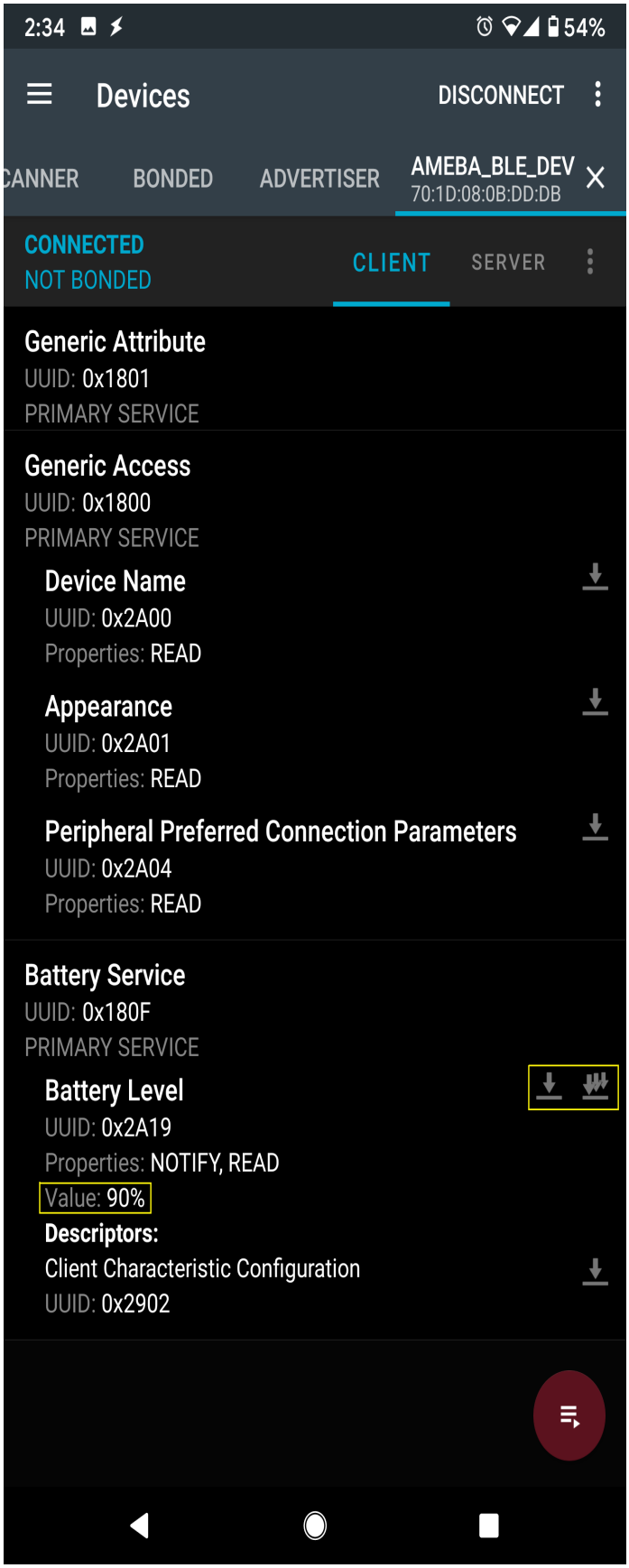


Upload the code and press the reset button on Ameba once the upload is finished.

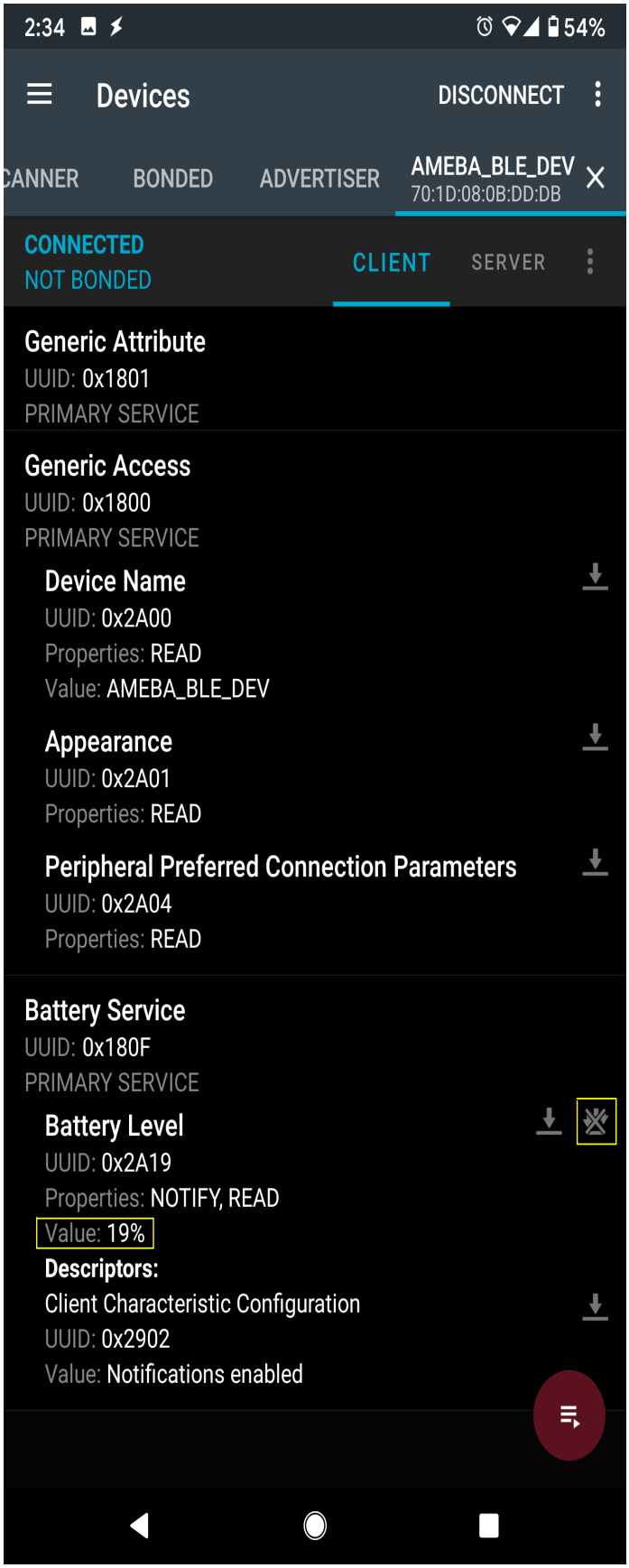
On your mobile phone, open the Bluetooth app and scan for the Bluetooth signal broadcast by Ameba, it should appear as a device named “AMEBA_BLE_DEV”.


2:34
54%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
No filter
5A:E9:FB:C0:37:77
NOT BONDED -82 dBm 270 ms
N/A
57:B3:8B:95:B3:6B
NOT BONDED -92 dBm 102 ms
N/A
CONNECT
14:99:E2:1C:BA:0D
NOT BONDED -93 dBm 180 ms
AMEBA_BLE_DEV
CONNECT
70:1D:08:0B:DD:DB
NOT BONDED -64 dBm 203 ms
Device type: LE only
Advertising type: Legacy
Flags: LimitedDiscoverable, BrEdrNotSupported
Complete Local Name: AMEBA_BLE_DEV
Appearance: [0] Unknown
CLONE
RAW
MORE
N/A
CONNECT
76:29:9E:79:7B:DA
NOT BONDED -81 dBm 275 ms
N/A
CONNECT
7F:7C:F8:63:F3:BB
NOT BONDED -98 dBm 273 ms
N/A
7D:0F:D2:74:C4:3D
NOT BONDED -60 dBm 102 ms

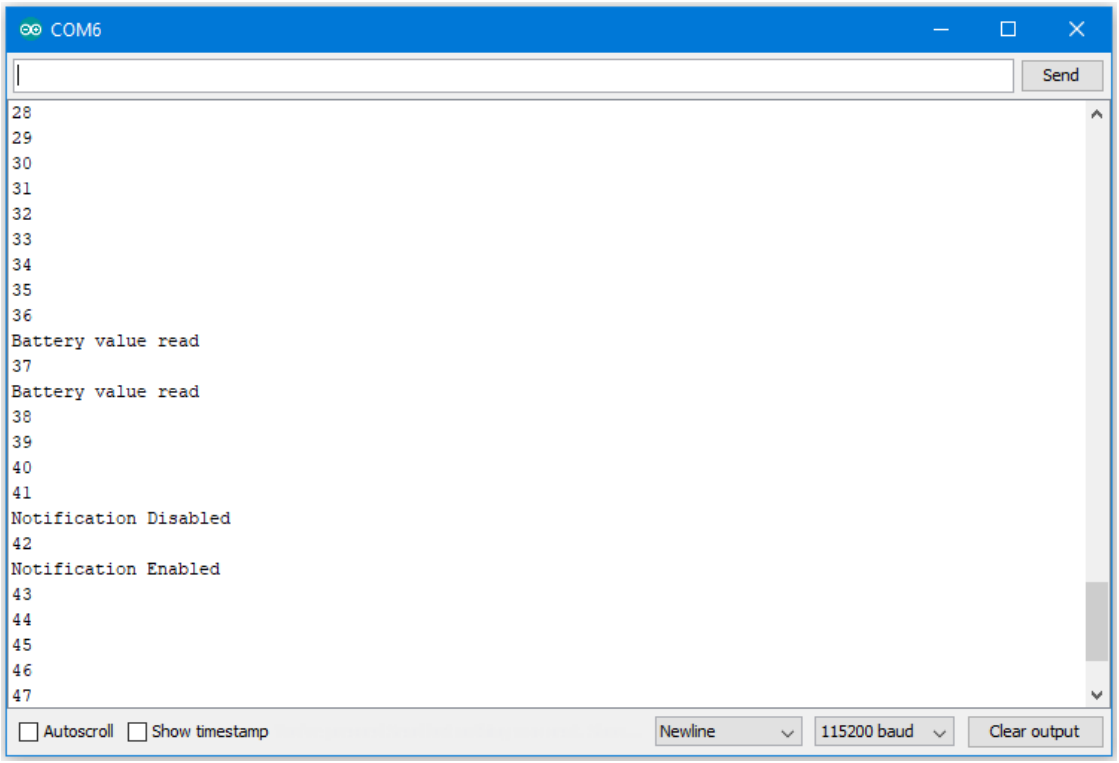
Connect to the Ameba Bluetooth device, and a list of available services should appear. Click on the battery service to expand it, and you can see the battery level data value. The arrows highlighted in the box on the right are used to read data and subscribe to notifications. Click on the single arrow to read the battery level value, and a 90% value will appear.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 90%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Click on the triple arrow to subscribe to updates on the battery level value, and the battery value will start updating by itself.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Value: AMEBA_BLE_DEV
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 19%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

The serial monitor will show the sketch increasing the battery level every second. When you click on either of the arrows, the sketch running on the Ameba will be notified, and will print out the action taken.



Code Reference

BLEService and BLECharacteristic classes are used to create and define the battery service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(GAP_ADTYPE_ADV_IND)` is used to set the advertisement type to a general undirected advertisement that allows for connections.

`setReadCallback()` and `setCCCDCallback()` is used to register functions that will be called when the battery level data is read, or notification is enabled by the user.

`BLE.configServer(1)` is used to tell the Bluetooth stack that there will be one service running.

`addService()` registers the battery service to the Bluetooth stack.

BLE – BLE Beacon

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

A BLE beacon broadcasts its identity to nearby Bluetooth devices, to enable the other devices to determine their location relative to the beacon, and to perform actions based on information broadcasted by the beacon.

Example applications of beacons include indoor positioning system, location-based advertising and more.

From the definition of its purpose as a broadcast device, a BLE beacon thus cannot be connected to, and can only send information in its Bluetooth advertisement packets.

There are several BLE beacon protocols. The Ameba BLEBeacon library supports the iBeacon and AltBeacon protocols.

Procedure

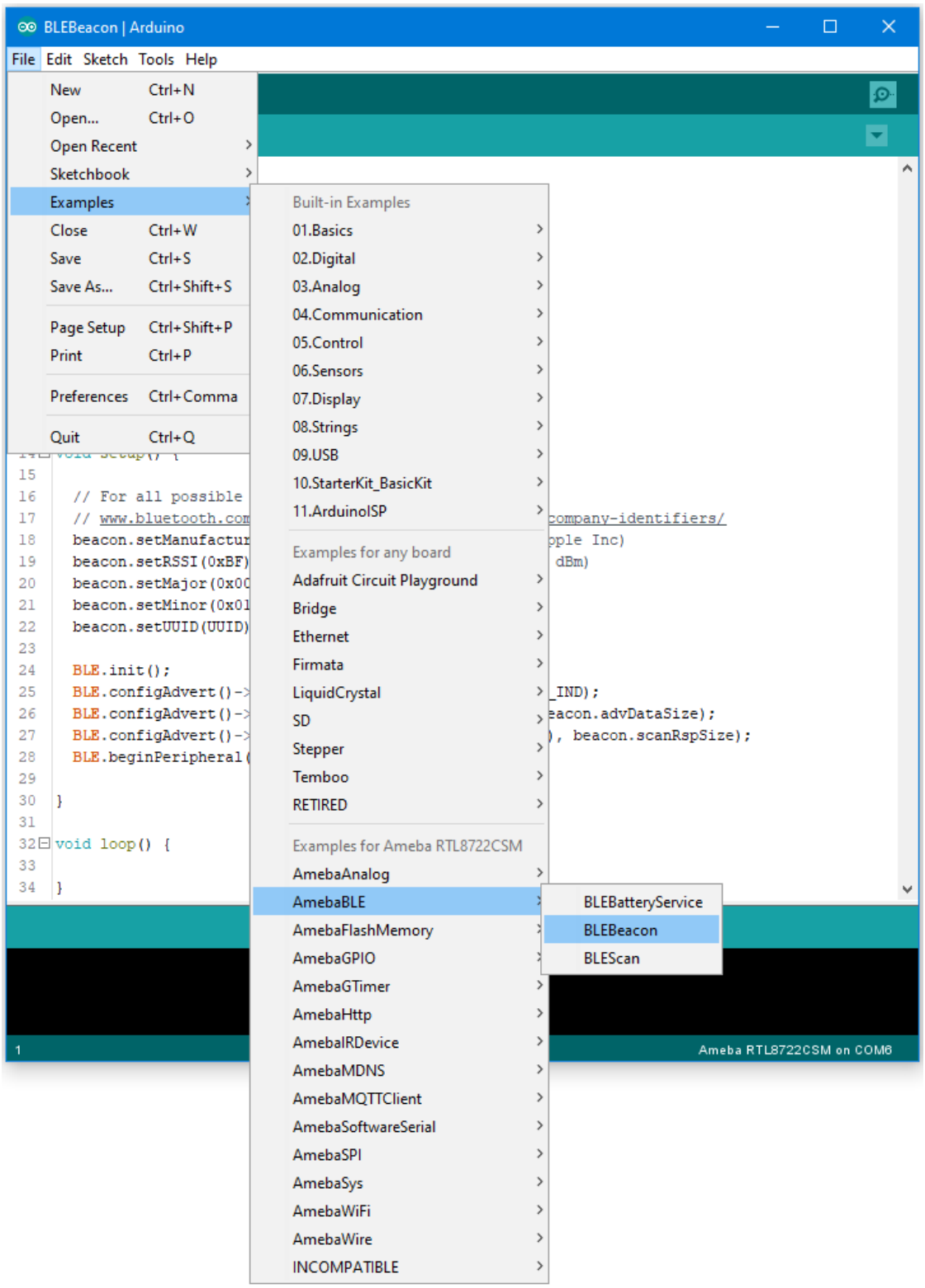
First, you need to install some Bluetooth apps on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.
The recommended application is nRF connect, and is available at the links below:

- **Android** : https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS :https://apps.apple.com/us/app/nrf-connect/id1054362403

LightBlue is an alternative application that can also be used, but has less features:

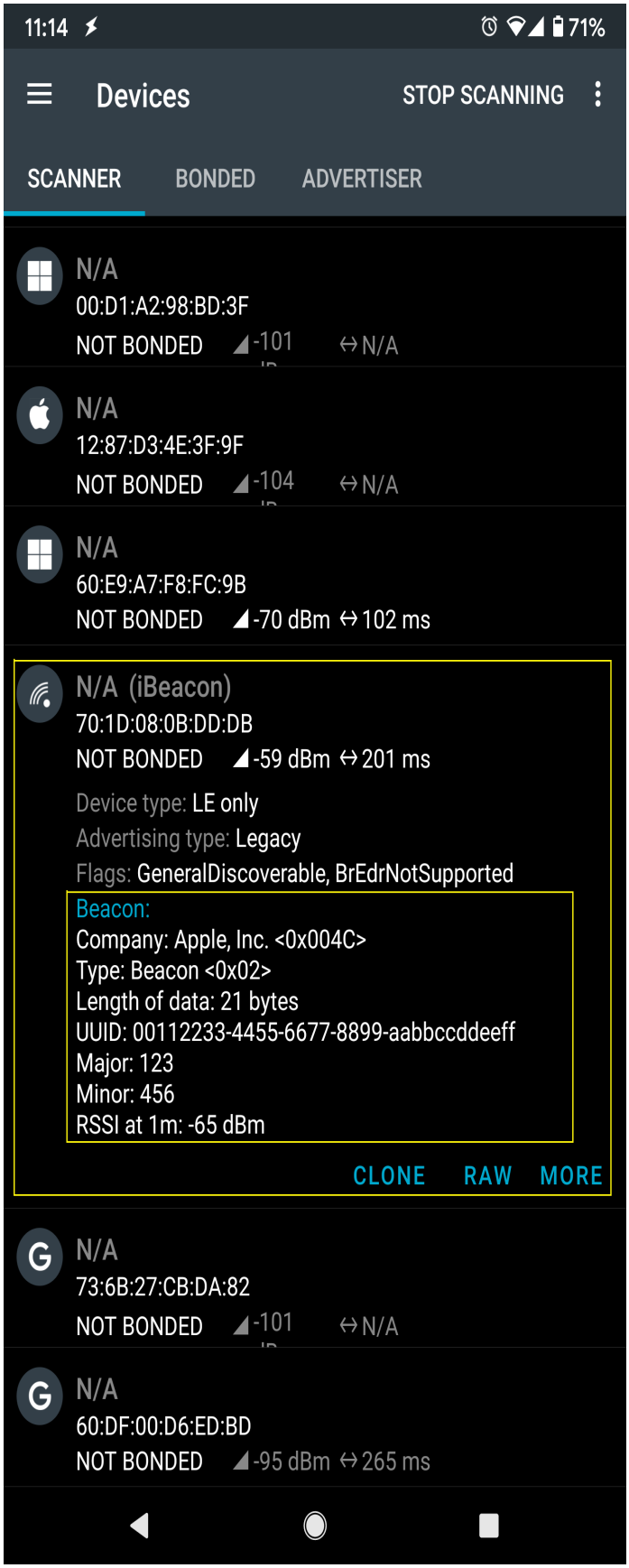
- **Android** : https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS :https://apps.apple.com/us/app/lightblue/id557428110

Open the example, “`Files` -> “`Examples` -> “`AmebaBLE` -> “`BLEBeacon`”



Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Bluetooth app and scan for the beacon signal broadcast by Ameba.


11:14
71%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
N/A
00:D1:A2:98:BD:3F
NOT BONDED
-101
N/A
N/A
12:87:D3:4E:3F:9F
NOT BONDED
-104
N/A
N/A
60:E9:A7:F8:FC:9B
NOT BONDED
-70 dBm
102 ms
N/A (iBeacon)
70:1D:08:0B:DD:DB
NOT BONDED
-59 dBm
201 ms
Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 00112233-4455-6677-8899-aabbccddeeff
Major: 123
Minor: 456
RSSI at 1m: -65 dBm
CLONE
RAW
MORE
N/A
73:6B:27:CB:DA:82
NOT BONDED
-101
N/A
N/A
60:DF:00:D6:ED:BD
NOT BONDED
-95 dBm
265 ms

If you happen to be in an environment with multiple BLE beacons, you can tap the entries to expand them, and verify that the beacon data is identical to the data in the sketch.

Code Reference

`setRssi()` is used to set the received signal strength indicator (rssi) data field for a beacon. The specification states that this should be the received signal strength from the beacon at a 1 meter distance. With no method to measure this, it is set to -65dBm as an estimate.

`setMajor()` and `setMinor()` are used to set the two data fields. The purpose of these data are left for the manufacturer of the beacon to define, and can be used in any way.

`setUUID()` is used to give the beacon a universally unique identifier (UUID). This is a 128-bit number usually expressed as a hexadecimal string. It is used to identify each unique beacon, and can be randomly generated for free online.

The BLEBeacon library includes both iBeacon and AltBeacon classes, replace line 6 iBeacon with altBeacon to create an AltBeacon instead. The data fields are mostly the same, with only minor changes, please look at the header files for more details.

`BLE.init()` is used to allocate memory and prepare Ameba for starting the Bluetooth stack.

`BLE.configAdvert()` is used to configure the Bluetooth advertisement settings, to which we pass the beacon data and set the device as non-connectable.

`BLE.beginPeripheral()` starts Ameba in Bluetooth peripheral mode, after which it will begin to advertise with the beacon data provided.

BLE – BLE Scan

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

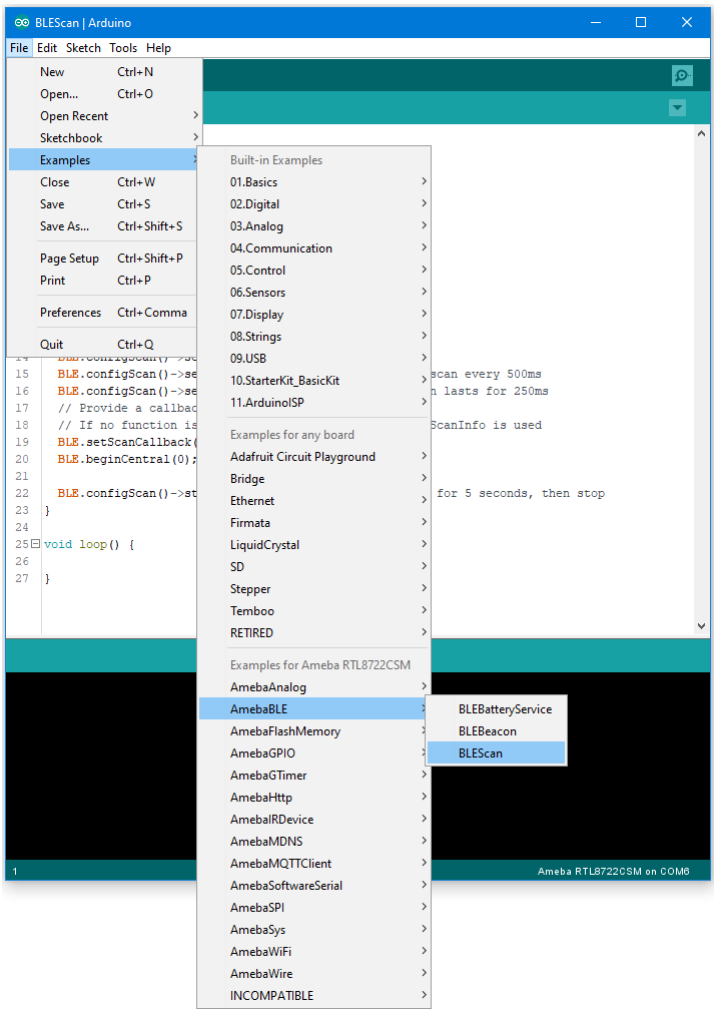
Example

Introduction

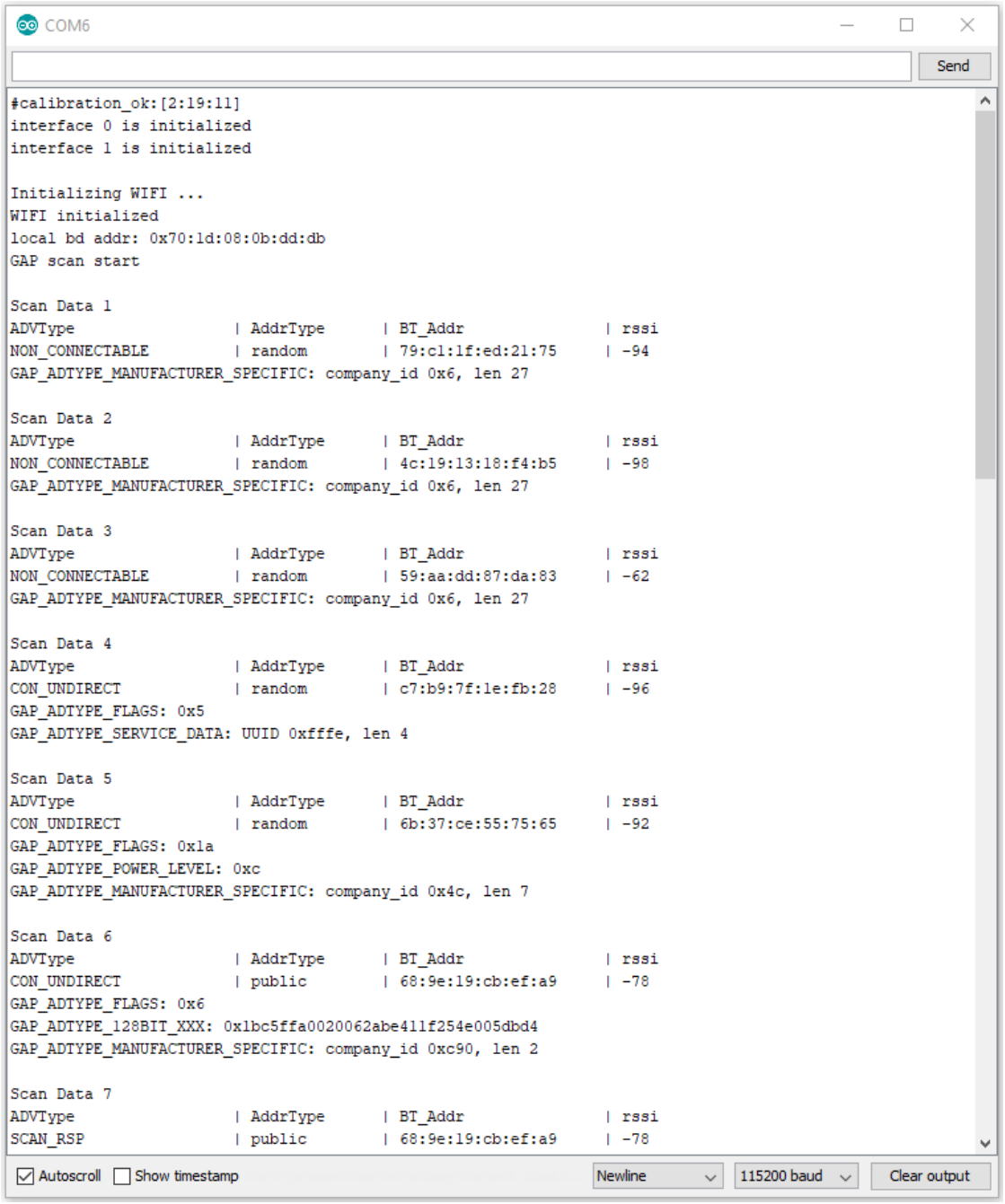
This example configures the Ameba as a Bluetooth central device, uses the scan functionality to scan for other Bluetooth devices, and prints out the results to the serial monitor.

Procedure

Open the example, "`Files`" -> "`Examples`" -> "`AmebaBLE`" -> "`BLEScan`"



Upload the code and press the reset button on Ameba once the upload is finished.
Open the Arduino serial monitor, and you should see the scan results of nearby Bluetooth devices formatted and printed out.



If you have the Bluetooth app nRF Connect installed, you can also use it to send out Bluetooth advertisements for the Ameba to pick up.

Code Reference

`setScanMode(GAP_SCAN_MODE_ACTIVE)` is used to set the scan mode. Active scanning will request for an additional scan response data packet from a device when it is found. Passive scanning will only look at the advertisement data, and not request for additional data.

`setScanInterval()` and `setScanWindow()` are used to set the frequency and duration of scans in milliseconds. A scan will start every interval duration, and each scan will last for the scan window duration. The scan window duration

should be lesser or equal to the scan interval. Set a short interval to discover devices rapidly, set a long interval to conserve power.

`setScanCallback(scanFunction)` is used to register a function to be called when scan results are received. This can be used to set a user function for additional processing of scan data, such as looking for a specific device. If no function is registered, the scan results are formatted and printed to the serial monitor by default.

`beginCentral(0)` is used to start the Bluetooth stack in Central mode. The argument 0 is used to indicate that no clients will be operating in central mode.

`startScan(5000)` is used to start the scanning process for a specified duration of 5000 milliseconds. The scan will repeat according to the set scan interval and scan window values. After 5000 milliseconds, the scan process will stop, and will be ready to be started again.

BLE – Battery Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Introduction

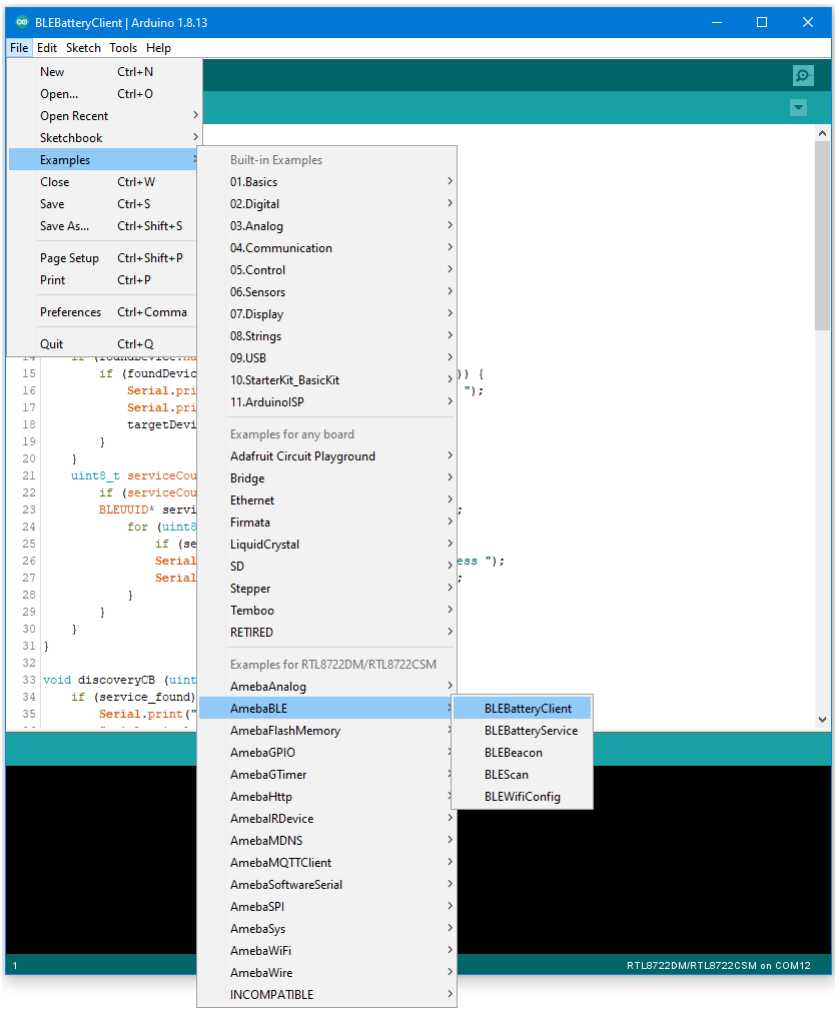
BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery client is set up on the Ameba Bluetooth stack. The client connects to another Ameba board running the corresponding BLE battery service to read the battery level data.

Procedure

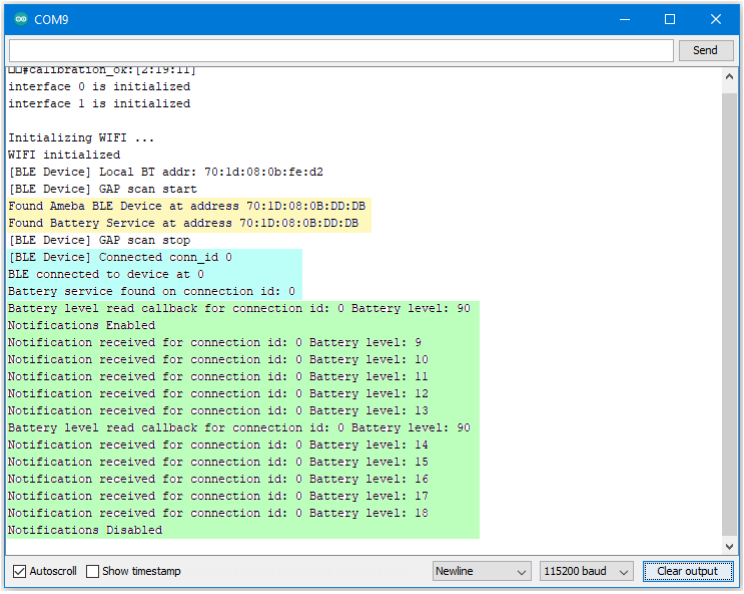
On the first Ameba board, upload the BLEBatteryService example code and let it run.

For the second Ameba board, open the example "`Files" -> "Examples" -> "AmebaBLE" -> "BLEBatteryClient`".



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor and observe the log messages as the Ameba board with the battery client scans, connects, and reads data from the Ameba board with the battery service.



Highlighted in yellow, the Ameba board with the battery client first scans for advertising BLE devices with the advertised device name “AMEBA_BLE_DEV” and the advertised service UUID of 0x180F representing the battery service.

After finding the target device, the Ameba board with the battery client forms a BLE connection and searches for a battery service on the connected device, highlighted in blue.

With the client connected to the service, the battery client begins to read data using both regular data reads and notifications, highlighted in green.

Code Reference

BLEClient is used to create a client object to discover services and characteristics on the connected device.

- `setNotifyCallback()` is used to register a function that will be called when a battery level notification is received.
- `BLE.configClient()` is used to configure the Bluetooth stack for client operation.
- `addClient(connID)` creates a new BLEClient object that corresponds to the connected device.

BLE – WiFi Configuration Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

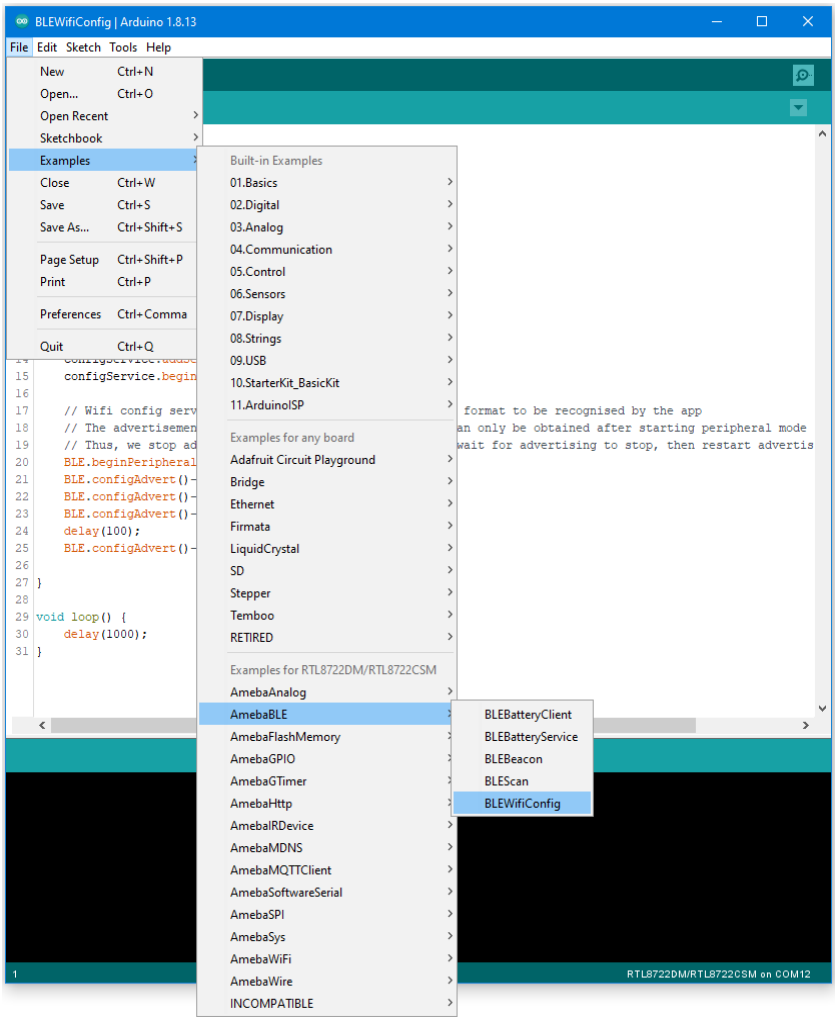
In this example, a WiFi configuration service is set up on the Ameba Bluetooth stack. A mobile phone with the configuration app connects to the Ameba device using BLE and configures the Ameba to connect to the correct WiFi access point.

Procedure

Ensure that the Realtek WiFi configuration app is installed on your mobile phone, it is available at:

– Google Play Store: https://play.google.com/store/apps/details?id=com.rtk.btconfig
– Apple App Store: https://apps.apple.com/sg/app/easy-wifi-config/id1194919510

Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEWifiConfigService`”.

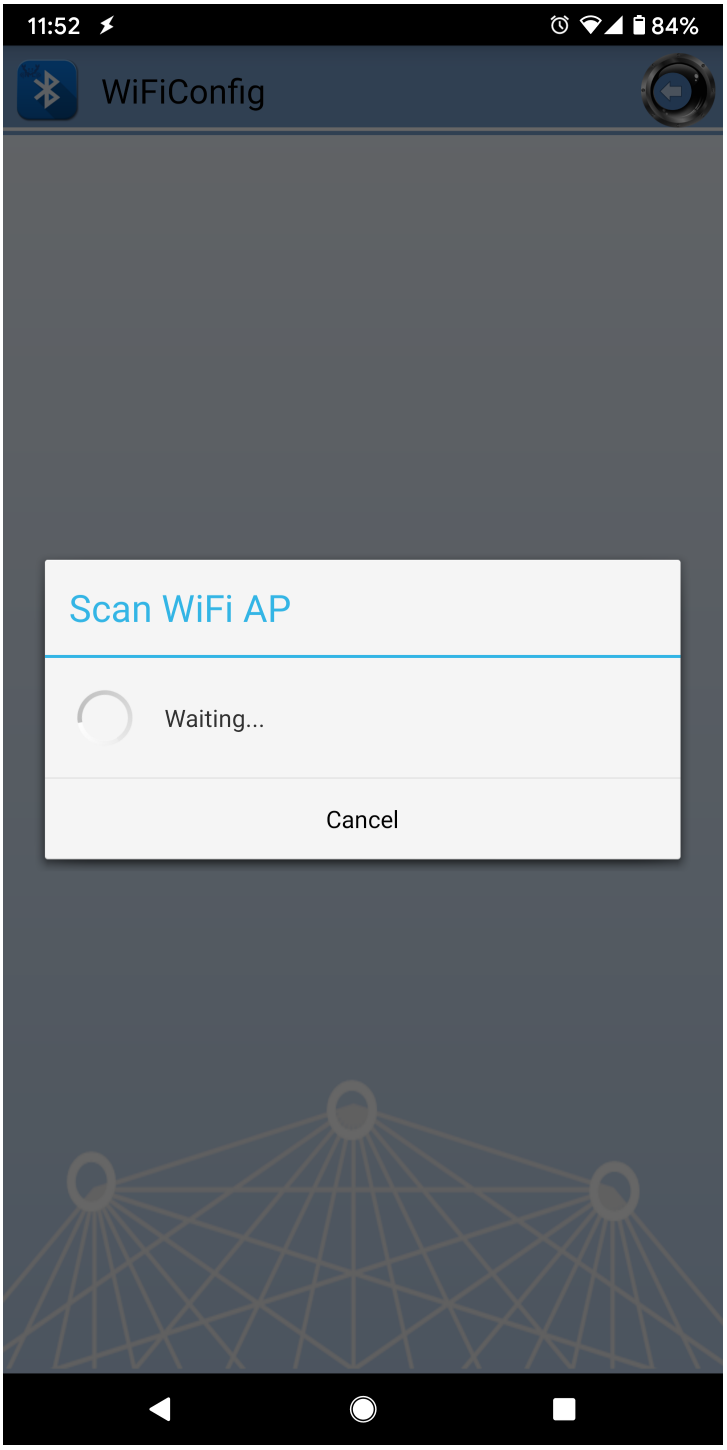


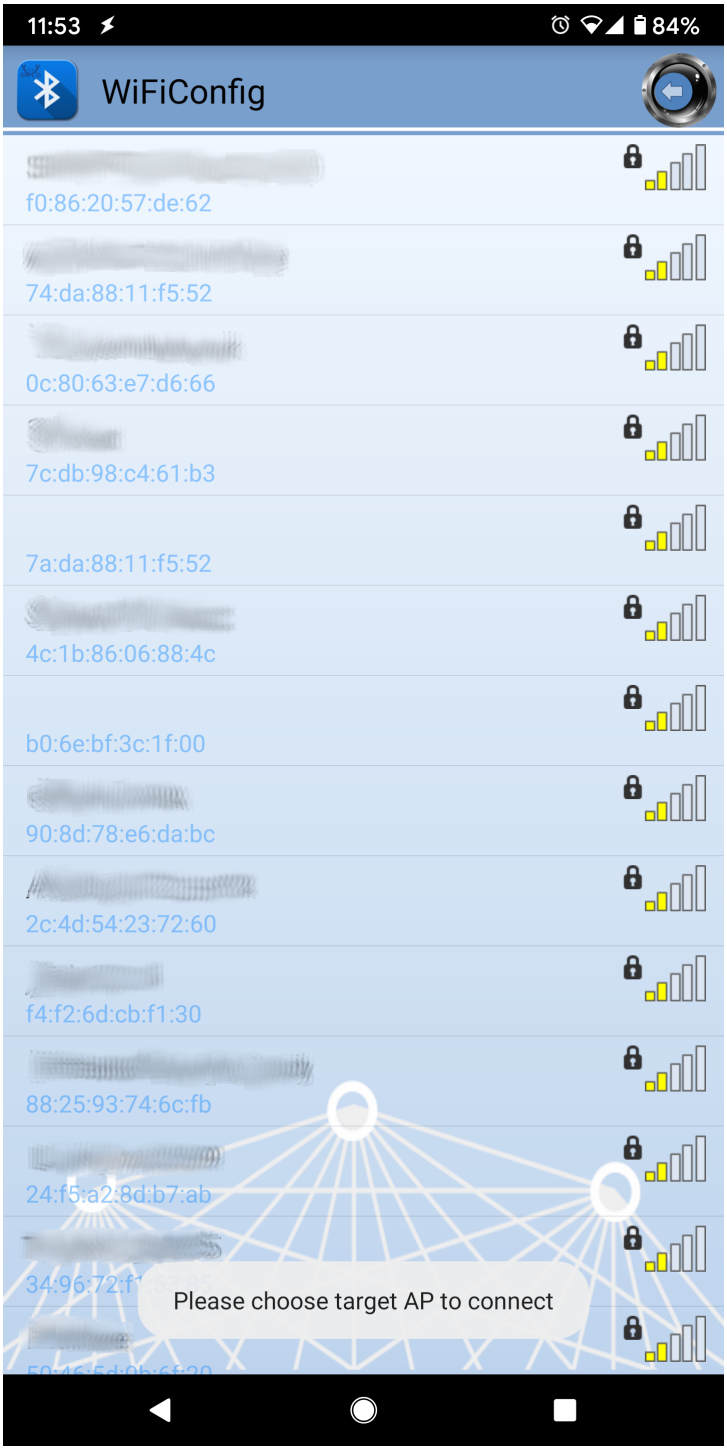
Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Realtek WiFiConfig app and tap the round button to scan for Ameba boards.

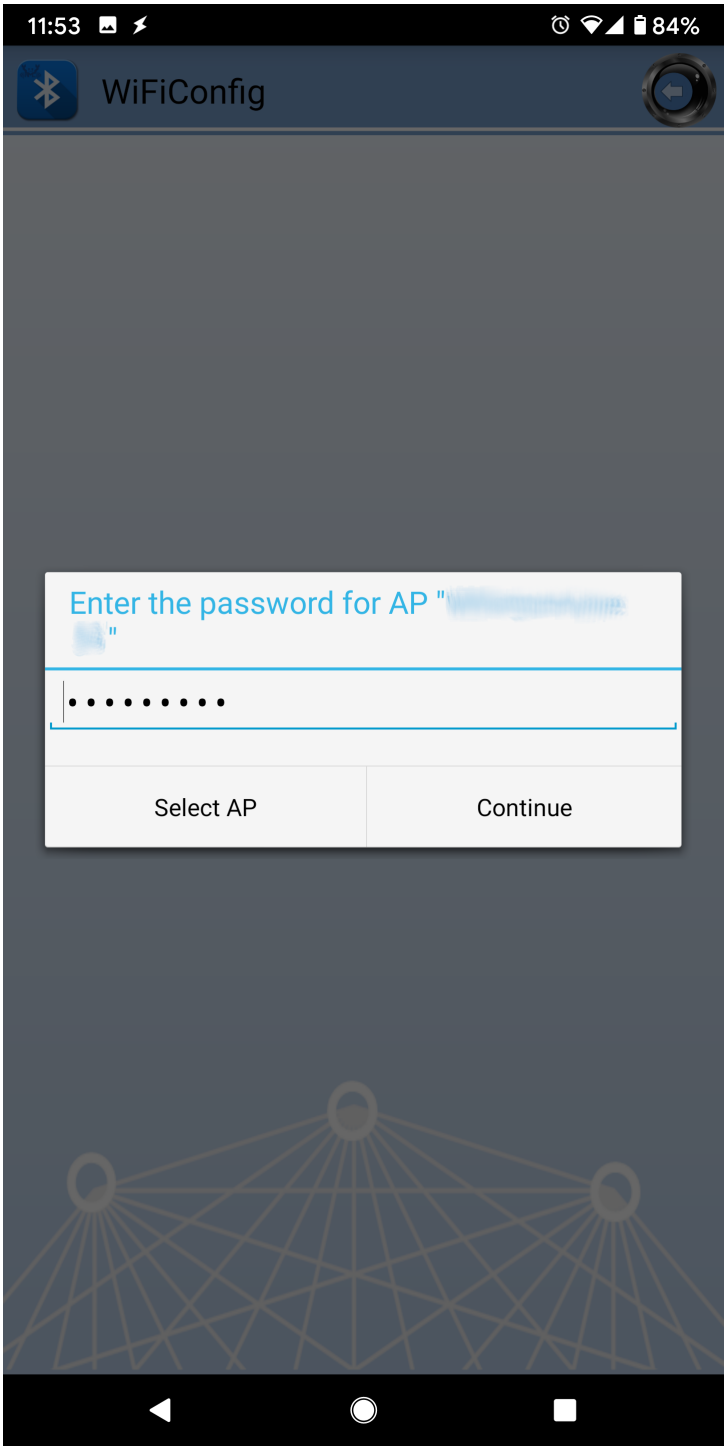
Select the correct Ameba board from the scan results. The app will connect to the Ameba board and ask the board to scan for WiFi networks and send the scan results back to the app using BLE.


4:16
82%
WiFiConfig
Choose device to configure
Ameba_0BFED2
70:1D:08:0B:FE:D2
Cancel

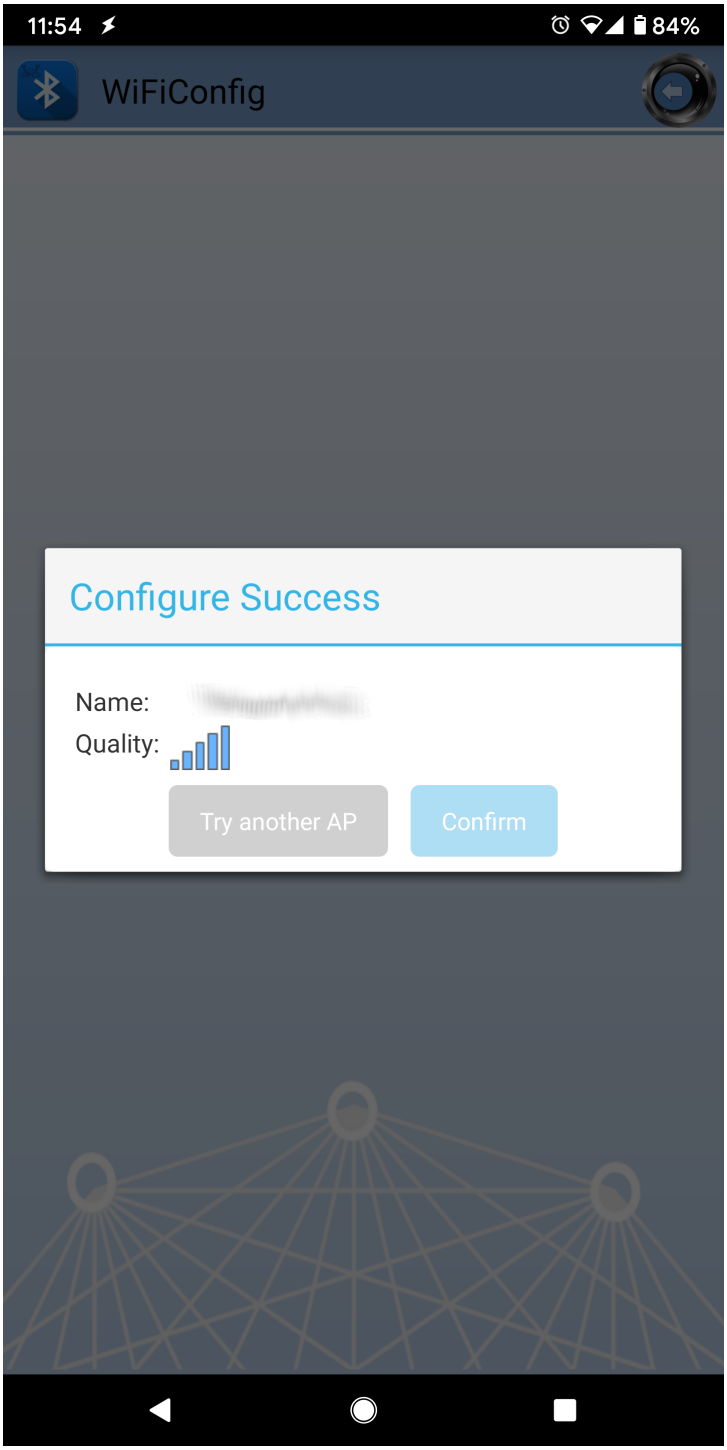

11:52
84%
WiFiConfig
Scan WiFi AP
Waiting...
Cancel



If your phone is currently connected to a WiFi network, the app will ask for the WiFi password to connect the Ameba board to the same WiFi network. Tap “Select AP” to choose another WiFi network, or enter the password and tap continue to connect Ameba to the selected WiFi network.



After the Ameba board connects to the WiFi network, the following message will be shown. Tap “Try another AP” to connect to another WiFi network or tap “Confirm” to keep the current WiFi network and disconnect BLE from the Ameba board.



Code Reference

BLEWifiConfigService is used to create an instance of the WiFi configuration service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(configService.advData())` is used to set the correct advertisement data necessary for the phone app to find the Ameba Bluetooth device.

BLE – BLE UART Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

Example

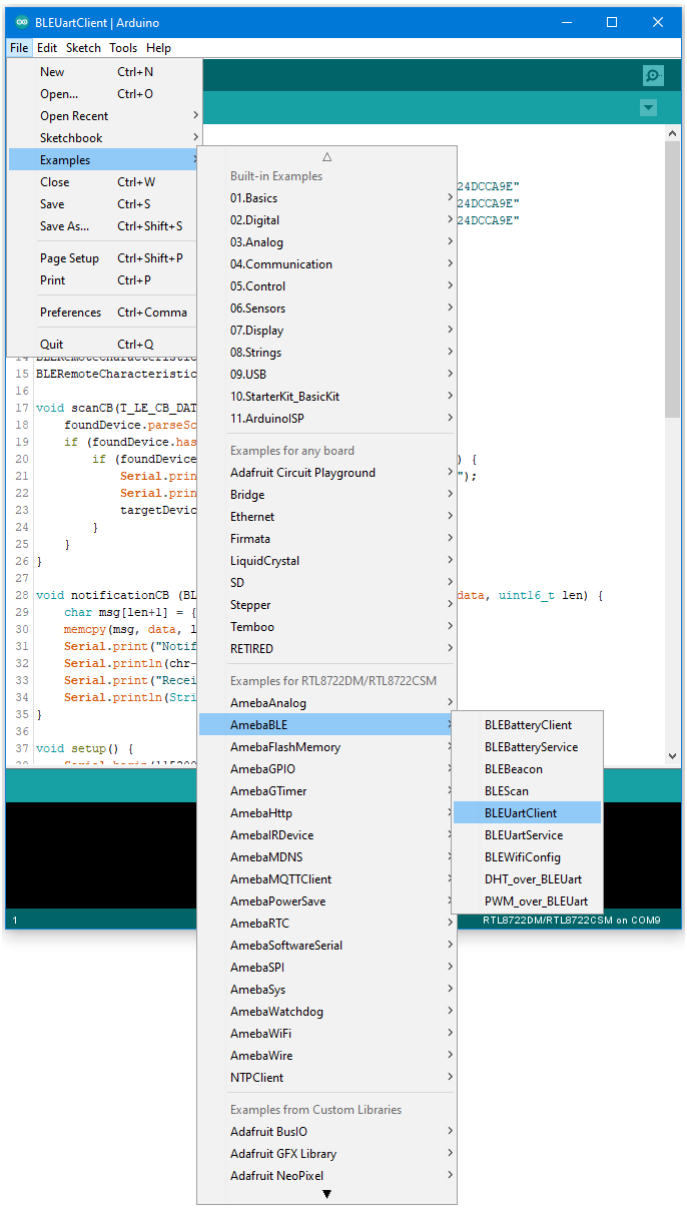
Introduction

In this example, two RTL8722 boards are connected using BLE. One board runs a BLE UART service, while the other connects to the service using a client and both boards are able to communicate with text messages over the UART service.

Procedure

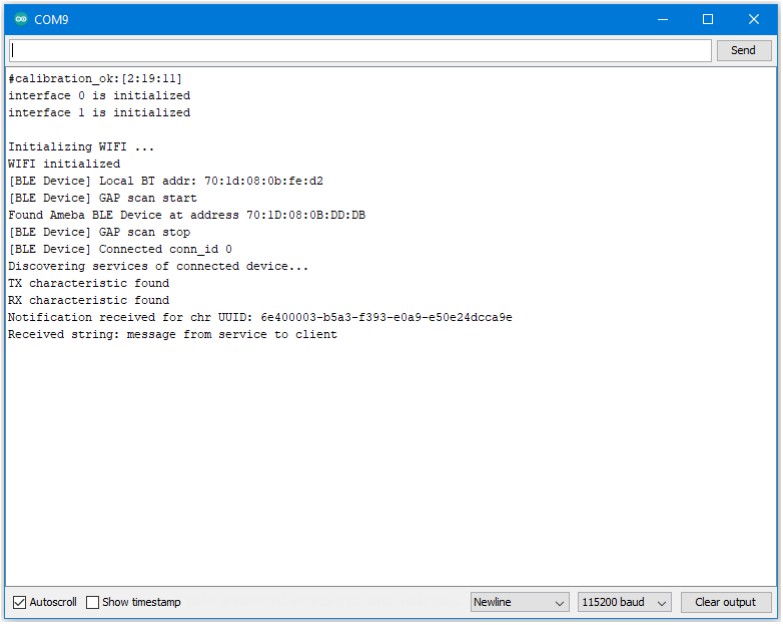
On the first board, upload the BLE UART service example code. Refer to the example guide for detailed instructions.

For the second board, open the example, "`Files" -> "Examples" -> "AmebaBLE" -> "BLEUart-Client`".



Upload the code and press the reset button on Ameba once the upload is finished.

Reset the UART service board first, wait for the BLE advertisement process to begin, and reset the UART client board. The client board should scan, discover, and connect to the service board. After connecting, the client board will verify that the correct UART service exists on the service board, before enabling notifications on the TX characteristic. Any message typed in the serial terminal will be sent to the other board using the UART service.



Code Reference

The BLEClient class is used to discover the services that exist on a connected BLE device. The discovery process will create BLERemoteService, BLERemoteCharacteristic and BLERemoteDescriptor objects corresponding to the services, characteristics and descriptors that exist on the connected device. These objects can then be used to read and write data to the connected device.

BLE – BLE UART Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS smartphone

Example

Introduction

With BLE, application data is sent and received using the GATT system. GATT uses services, characteristics, and attributes to organise data and control how the data can be read from and written to. The Bluetooth SIG specification for BLE includes several predefined services for common applications, but users are free to implement custom services and characteristics to best fit their data structure and application needs

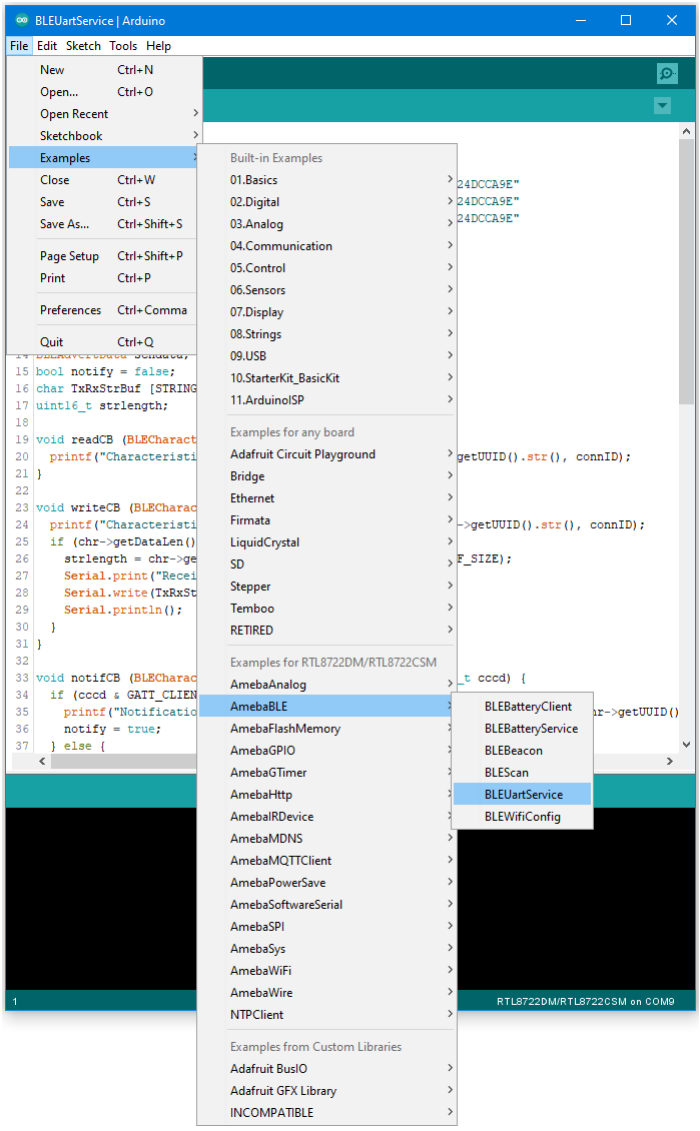
In this example, the BLEService and BLECharacteristic classes are used to implement a custom service for transmitting ASCII characters similar to regular UART. This custom service is the Nordic UART Service, which is supported in several smartphone apps.

Procedure

Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

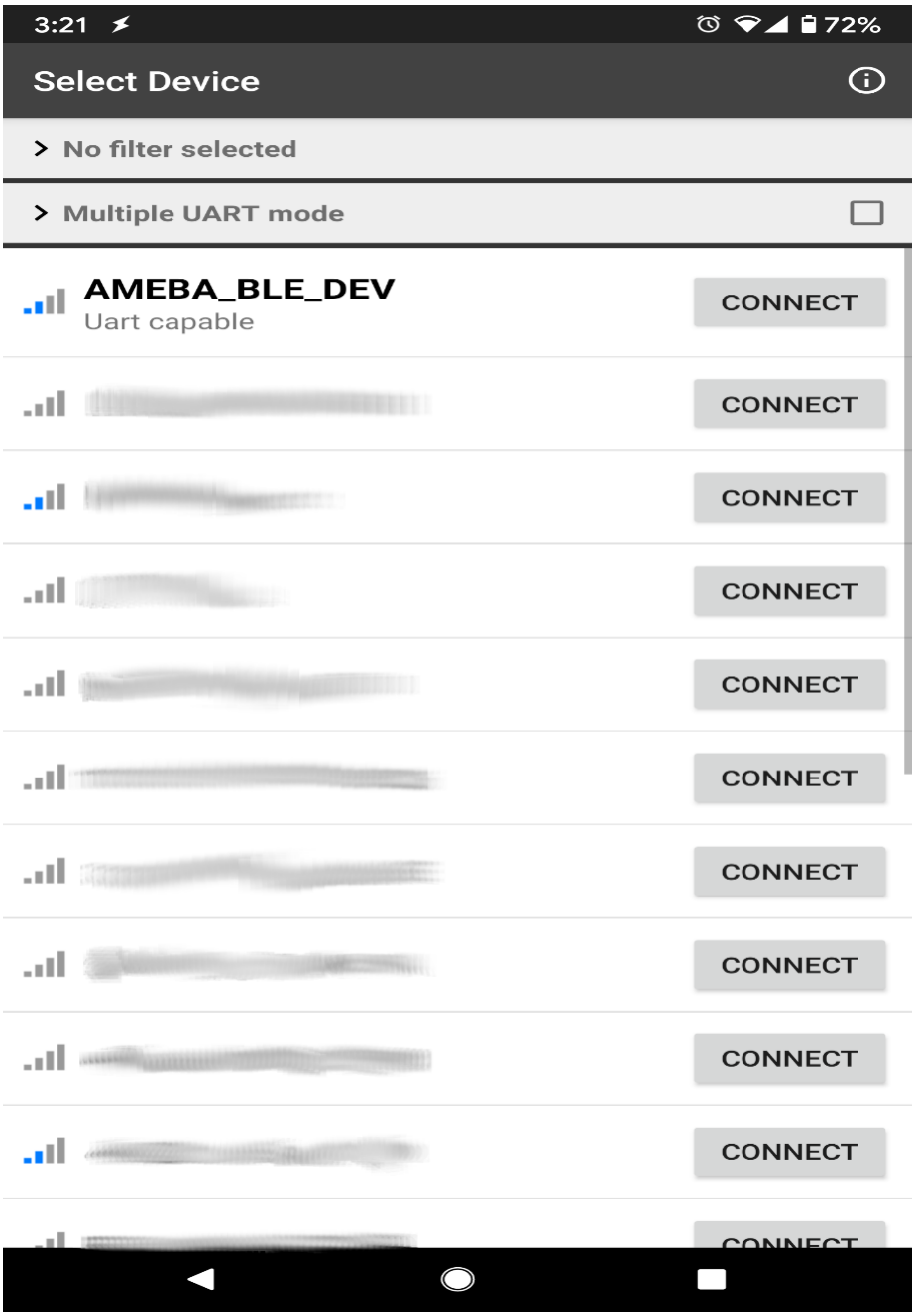
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

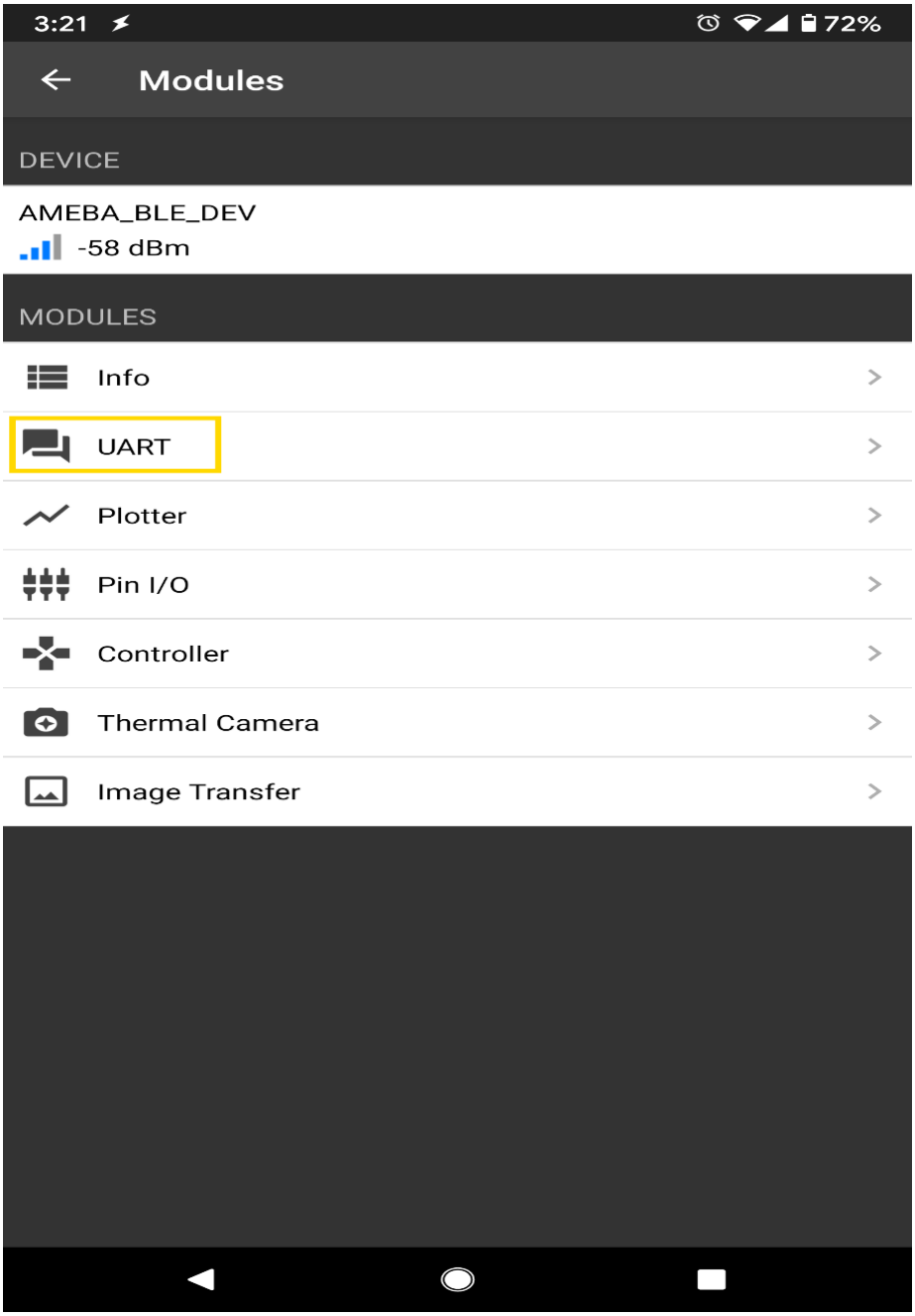
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEUartService`”.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app. Note that the BLE UART service on the Ameba board will only work with the UART and Plotter functions in the Bluefruit Connect app, other functions (Pin I/O, Image Transfer) require other BLE services

that are not included in this example.

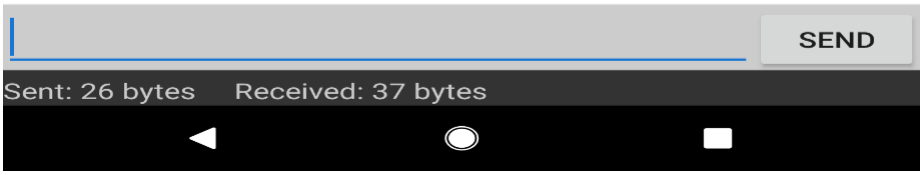


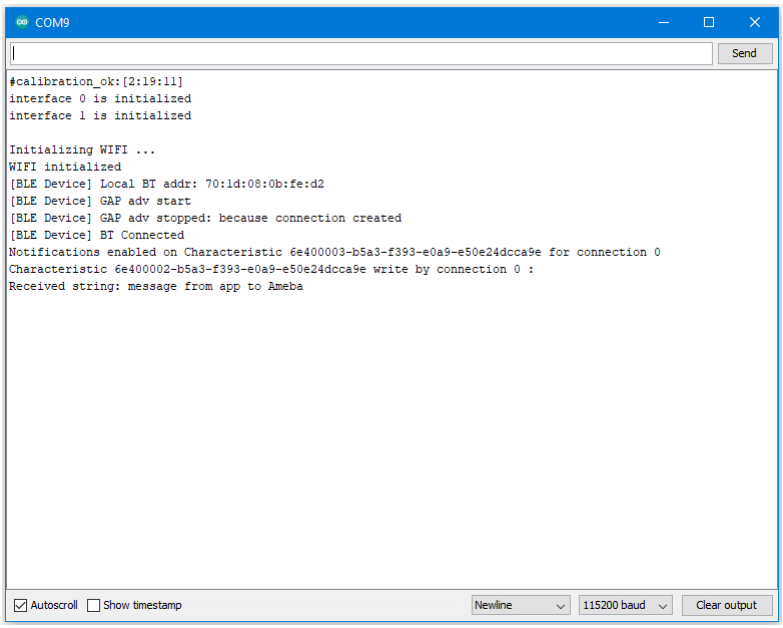


In the UART terminal section of the app, enter a message and click send. You should see the message appear in the Arduino serial monitor.

In the Arduino serial monitor, enter a message and click send. The message will appear in the smartphone app.


3:22
72%
UART
MQTT
message from app to Ameba
message from Ameba to smartphone app


SEND
Sent: 26 bytes
Received: 37 bytes



Code Reference

The BLECharacteristic class is used to create two characteristics, one for receive (Rx) and one for transmit (Tx), and added to a service created with the BLEService class.

The required read/write/notify properties are set for each characteristic using the `set__Property()` methods, and callback functions are registered using the `set__Callback()` methods. The required buffer size is also set for each characteristic so that it has enough memory to store a complete string.

When data is written to the receive characteristic, the registered callback function is called, which prints out the received data as a string to the serial monitor.

When data is received on the serial port, it is copied into the transmit characteristic buffer, and the `notify()` method is used to inform the connected device of the new data.

BLE – DHT over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21
- Android / iOS smartphone

Example

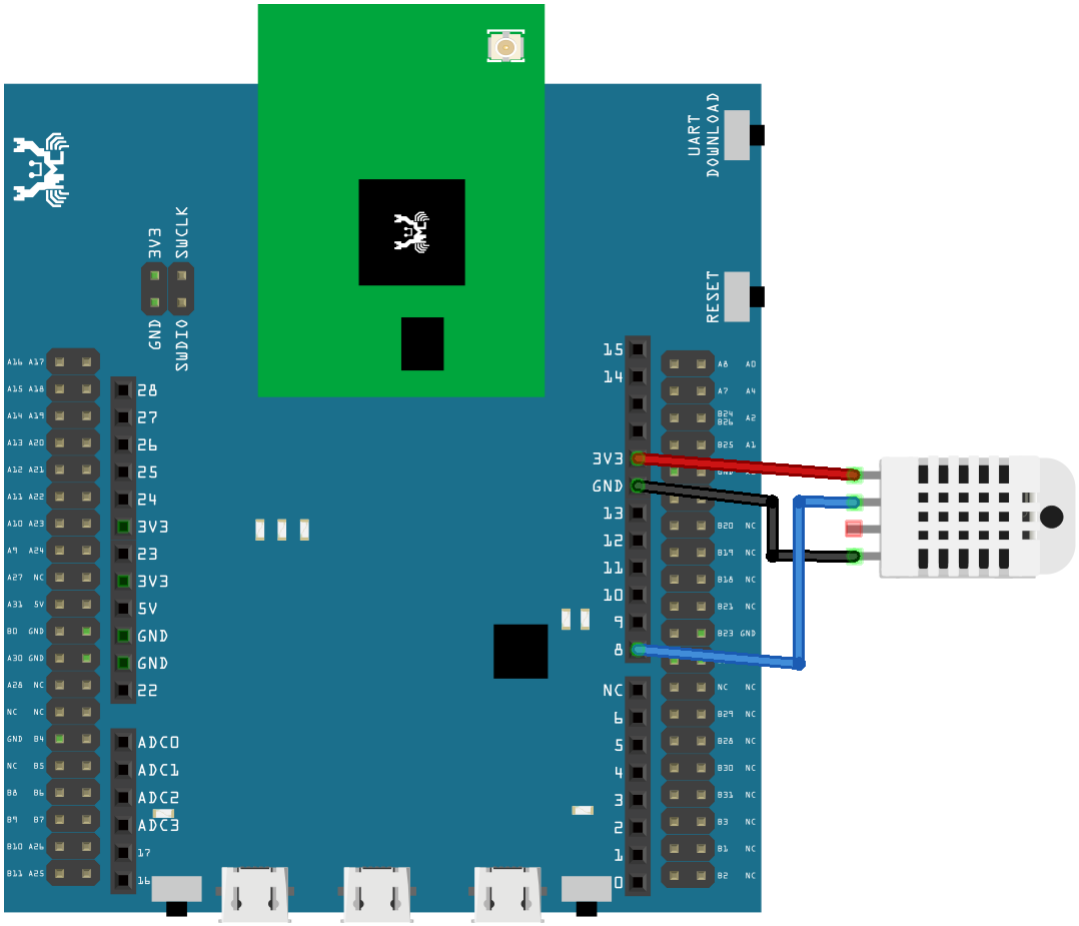
Introduction

In this example, the data obtained from a DHT temperature and humidity sensor are transmitted over a BLE UART service to a smartphone. Refer to the other examples for detailed explanations of using the DHT sensor and the BLE UART service.

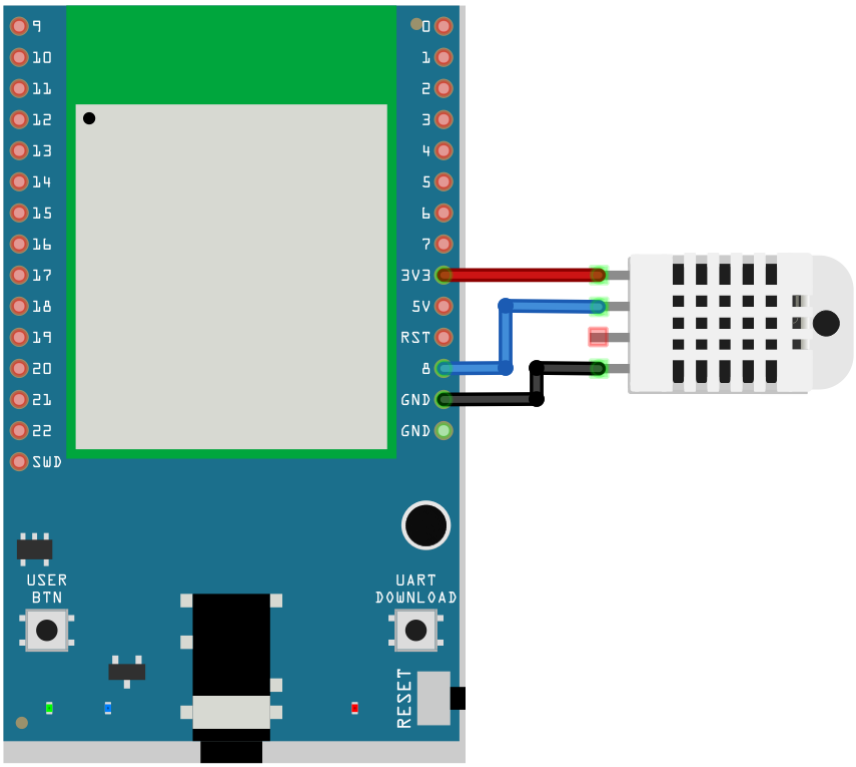
Procedure

Connect the DHT sensor to the Ameba board following the diagram.

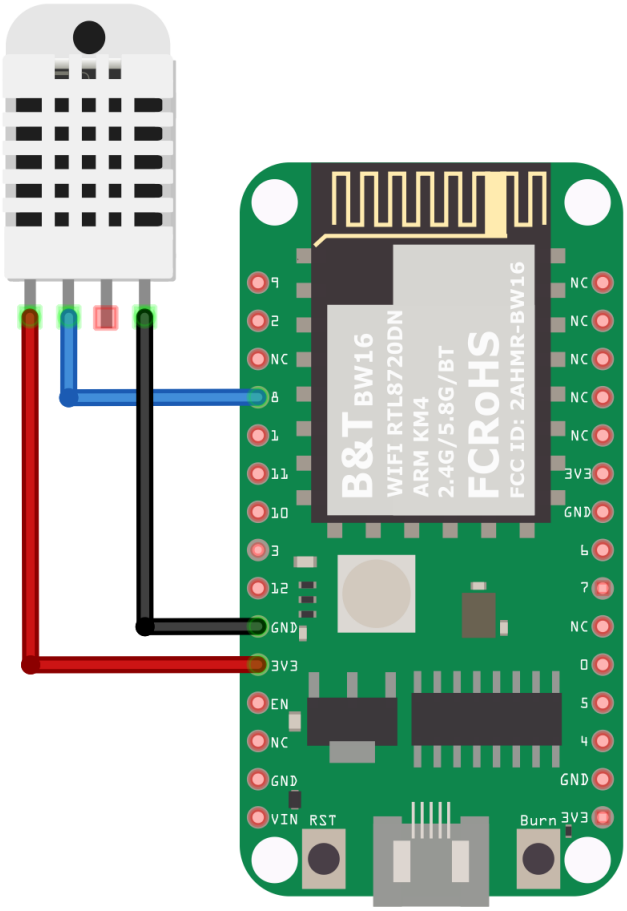
AMB21 / AMB22:



AMB23:


9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

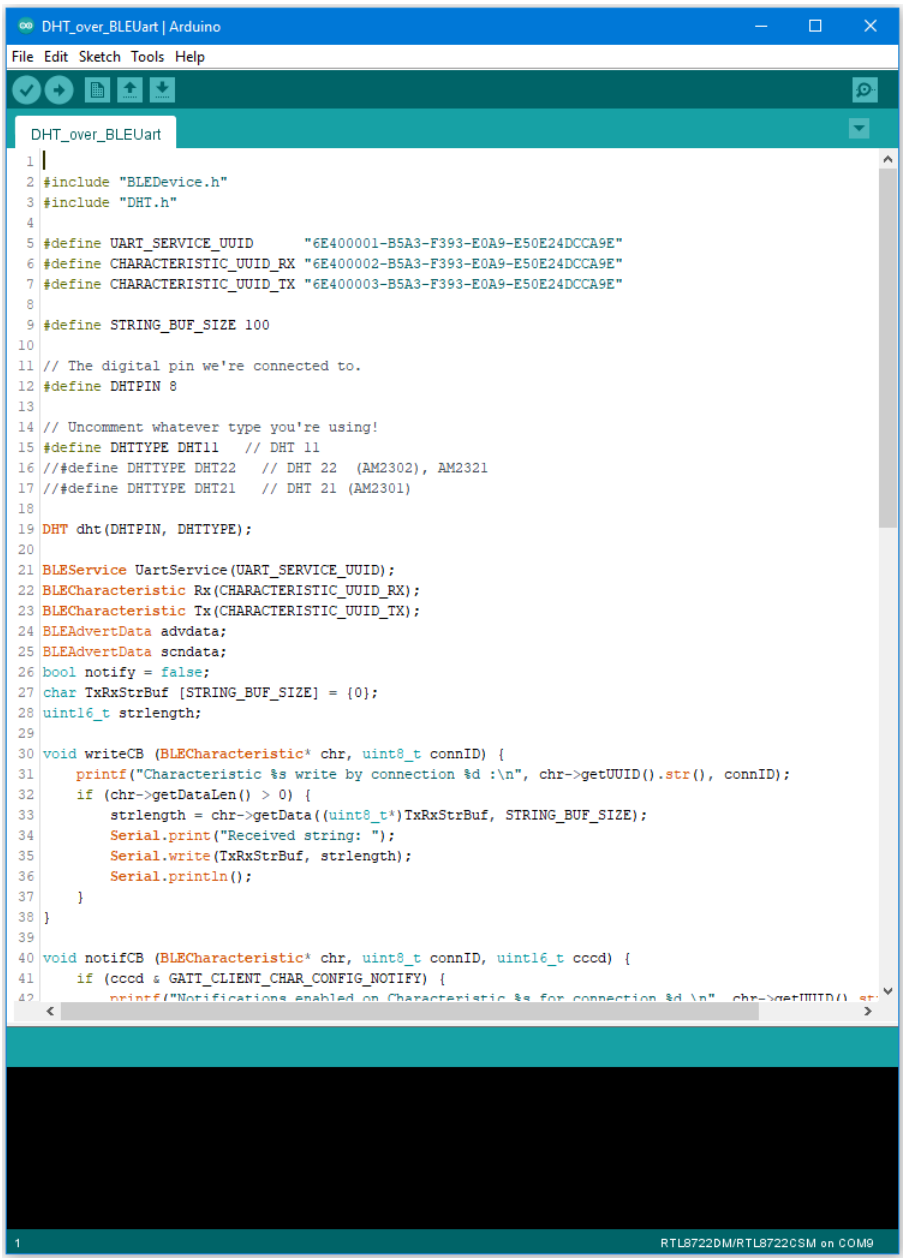
BW16:



Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
- Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connecta>https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

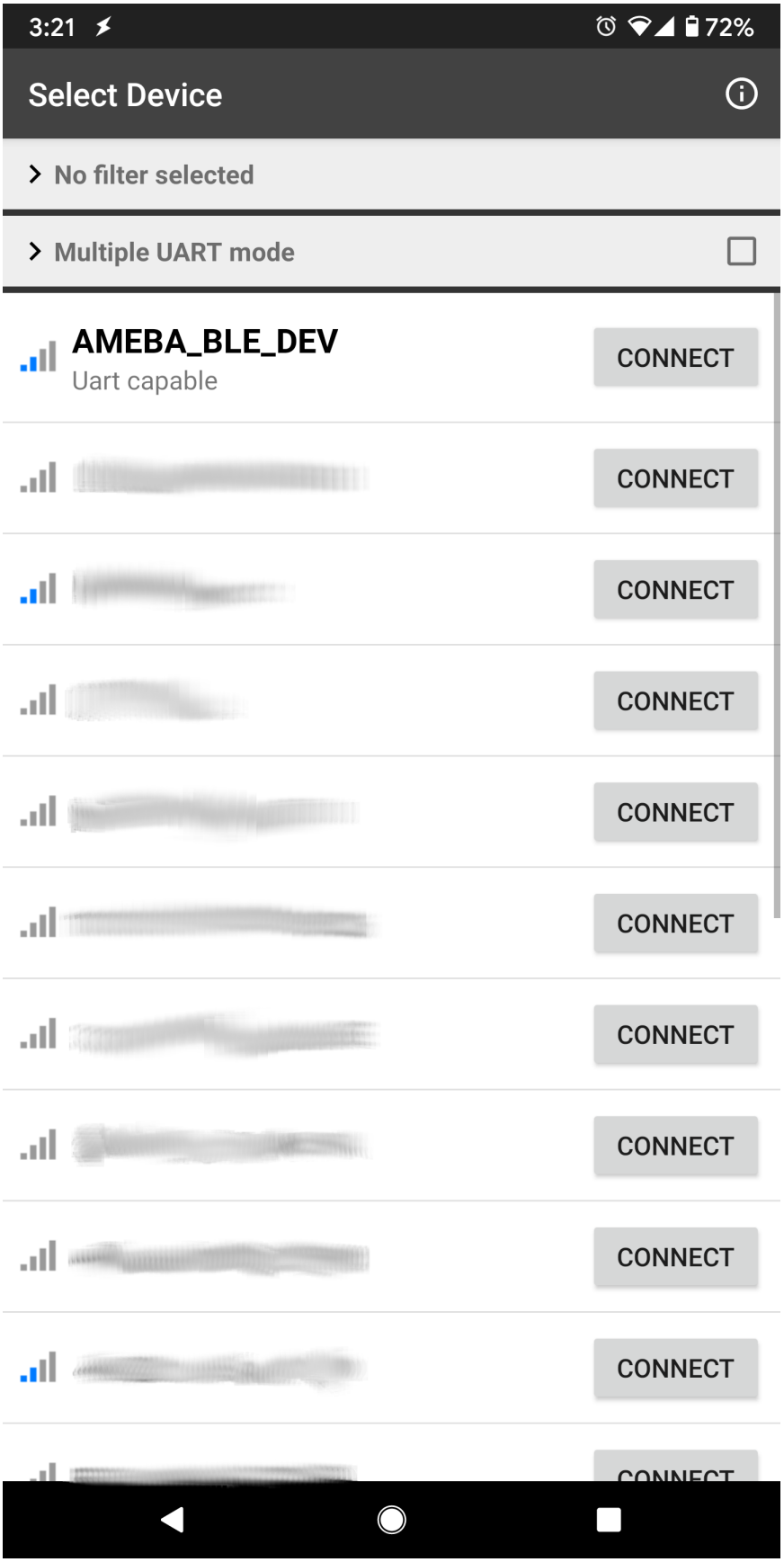
- Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

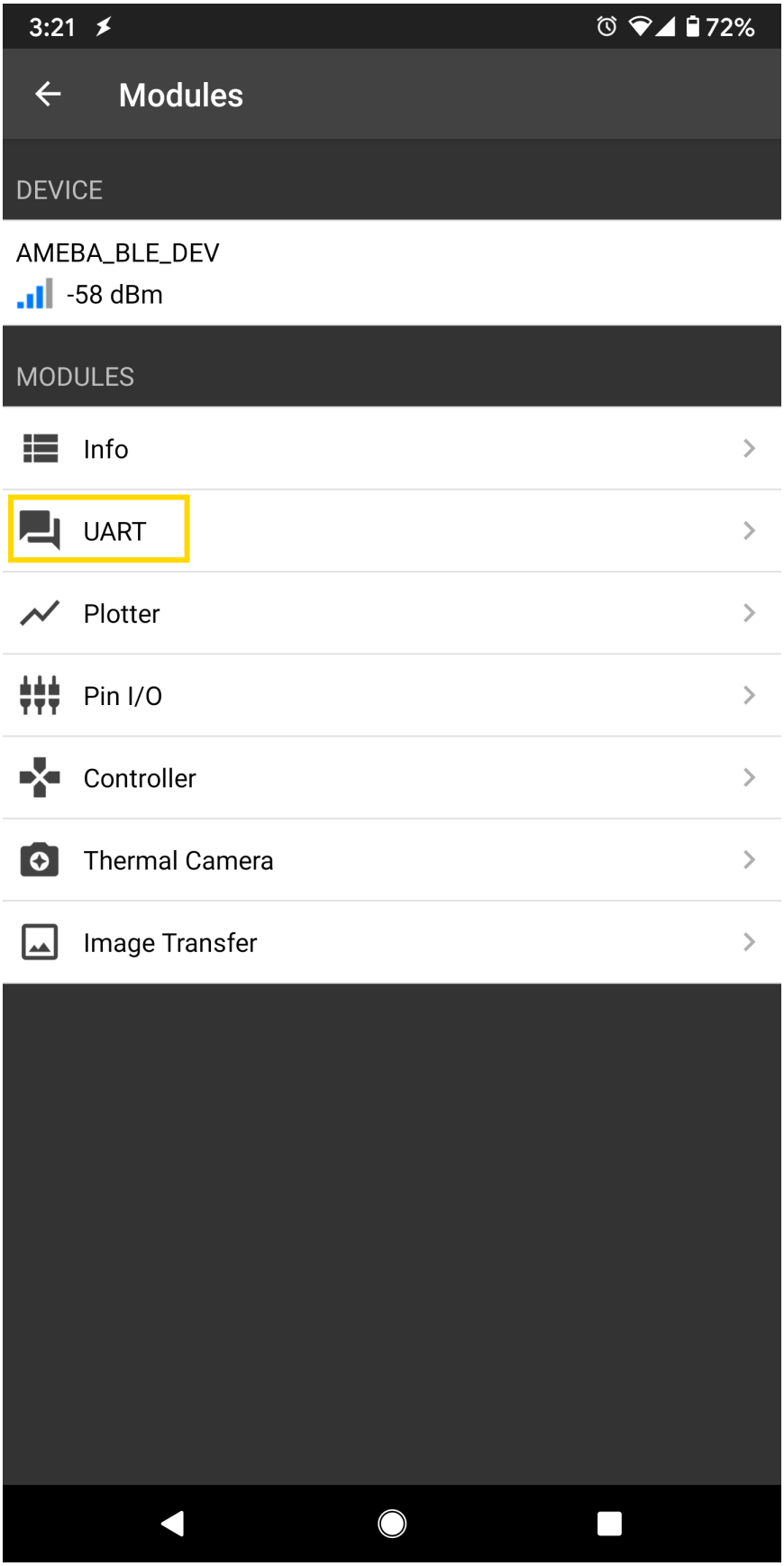
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`DHT_over_BLEUart`”.



Upload the code and press the reset button on Ameba once the upload is finished.

Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer

After starting the UART function, notifications should be received every 5 seconds containing the measured temperature and humidity.


4:20
81%
UART
MQTT
Humidity: 59% Temperature: 27°C
Humidity: 59% Temperature: 27°C
Humidity: 62% Temperature: 27°C
Humidity: 67% Temperature: 27°C
Humidity: 69% Temperature: 27°C
Humidity: 88% Temperature: 27°C
SEND
Sent: 0 bytes
Received: 198 bytes

BLE – PWM over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- RGB LED
- Android / iOS smartphone

Example

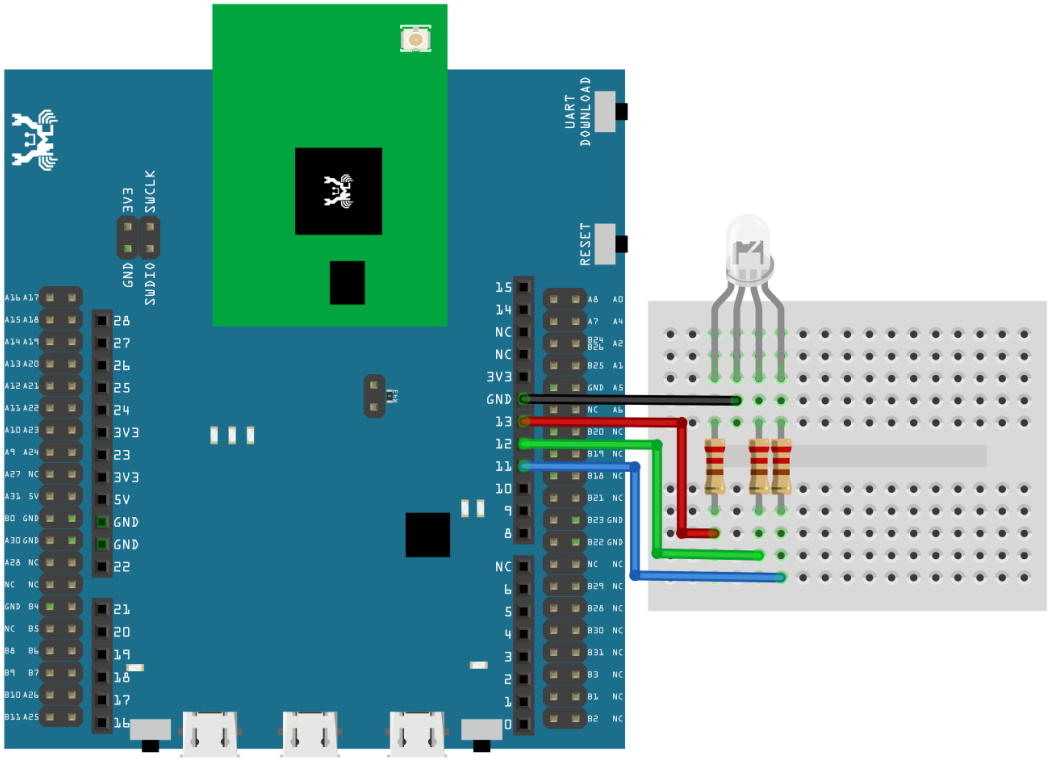
Introduction

In this example, a smartphone app is used to transmit commands over BLE UART to control the PWM outputs and change the color of a RGB LED. Refer to the other example guides for detailed explanations of the BLE UART service.

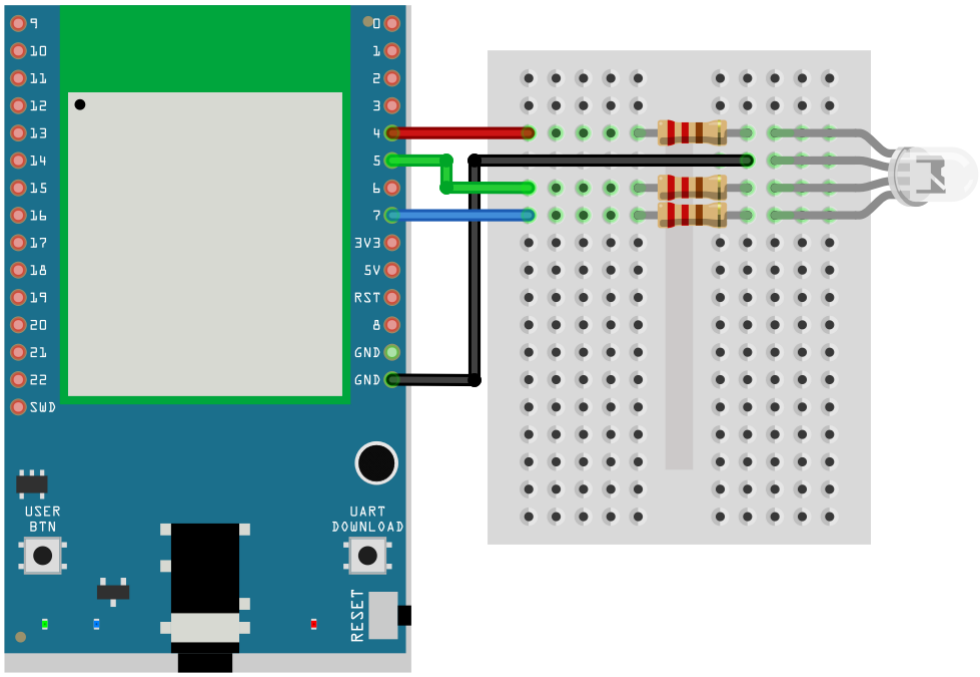
Procedure

Connect the RGB LED to the RTL8722 board following the diagram, the common LED pin may need to connect to 3.3V or GND depending on the type of LED (common anode / common cathode).

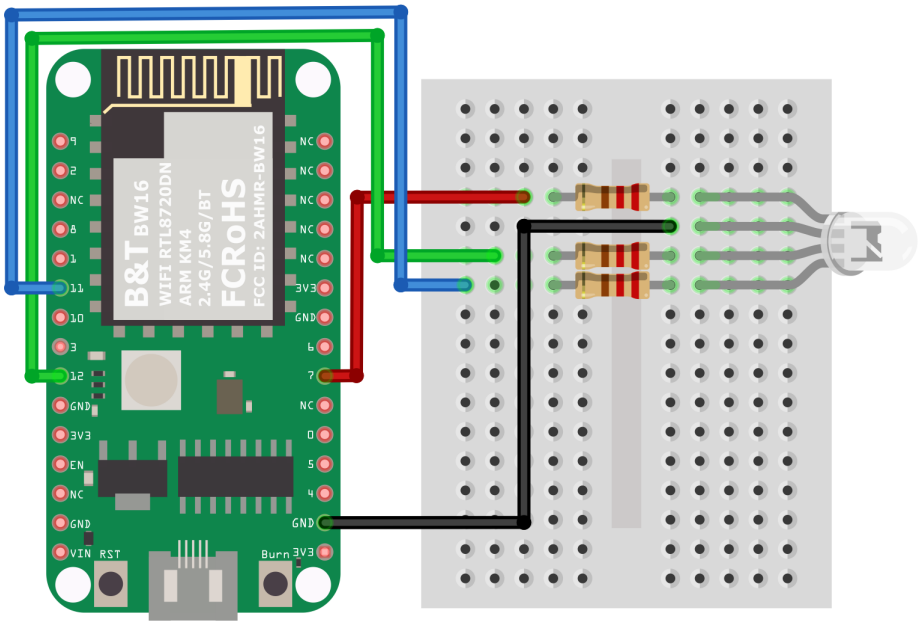
AMB21 /AMB22:



AMB23:



BW16:

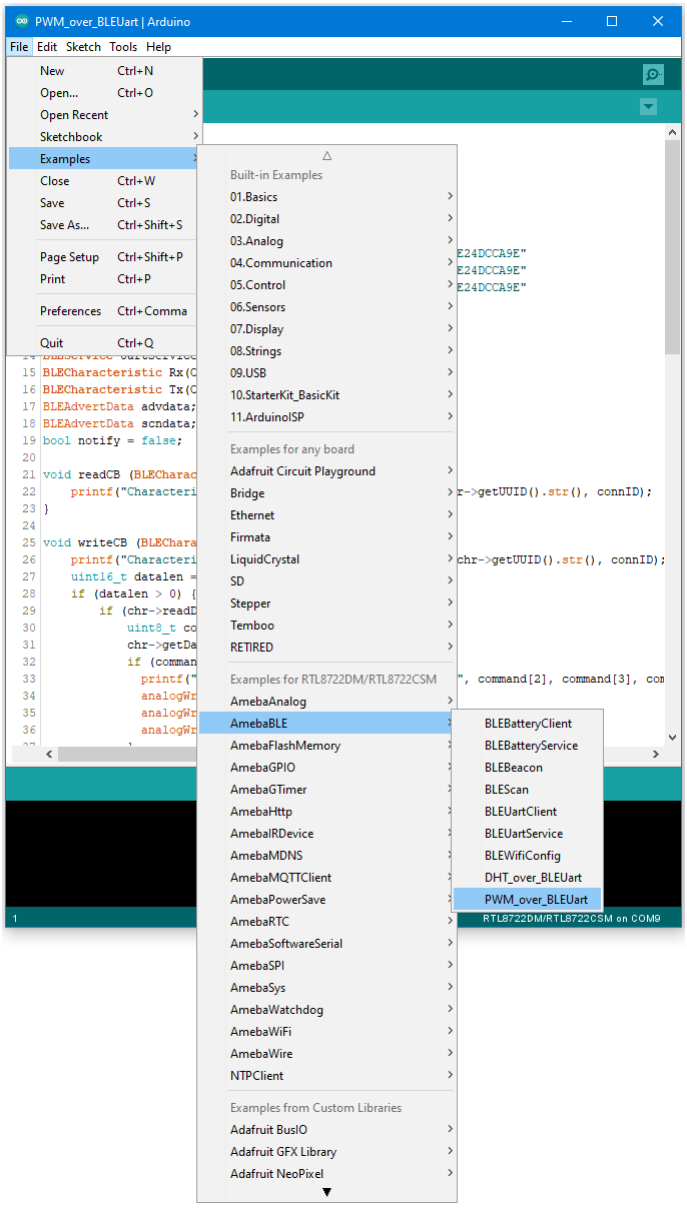


Ensure that the required app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect

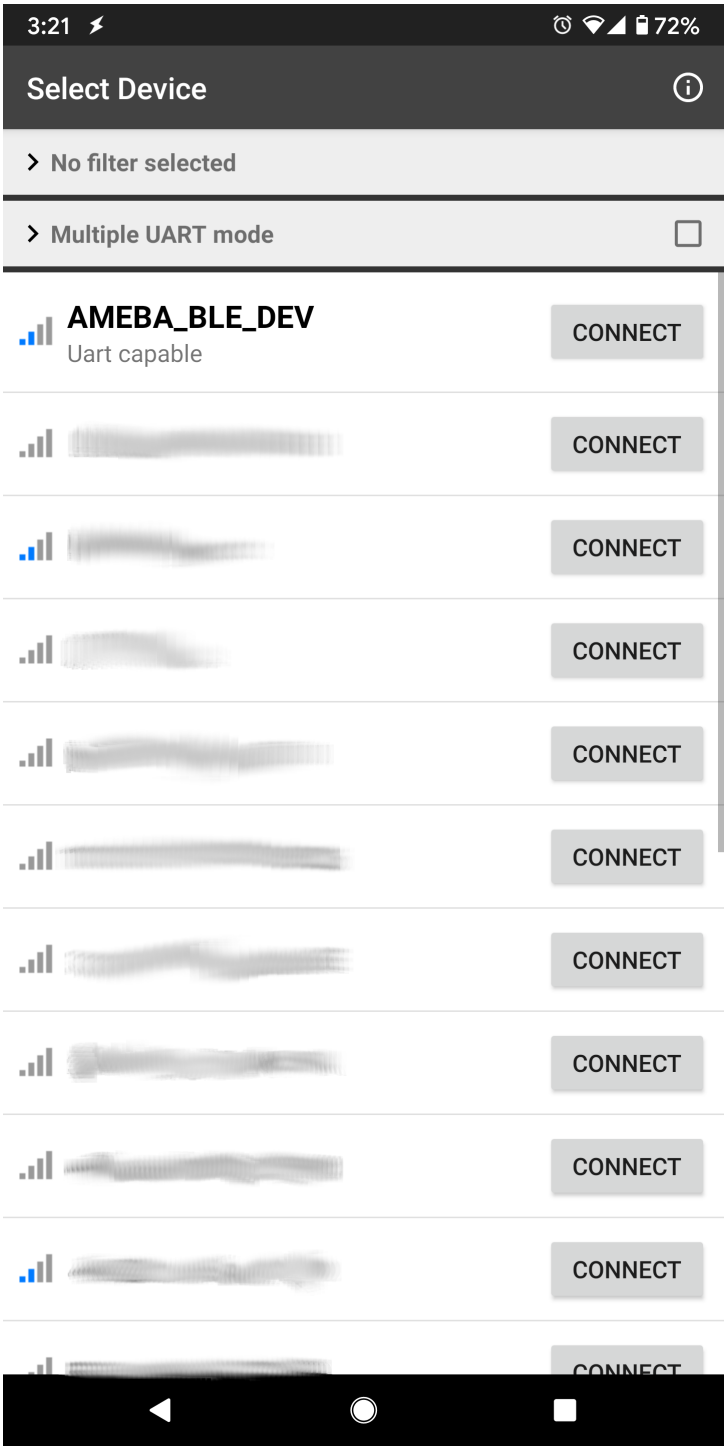
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

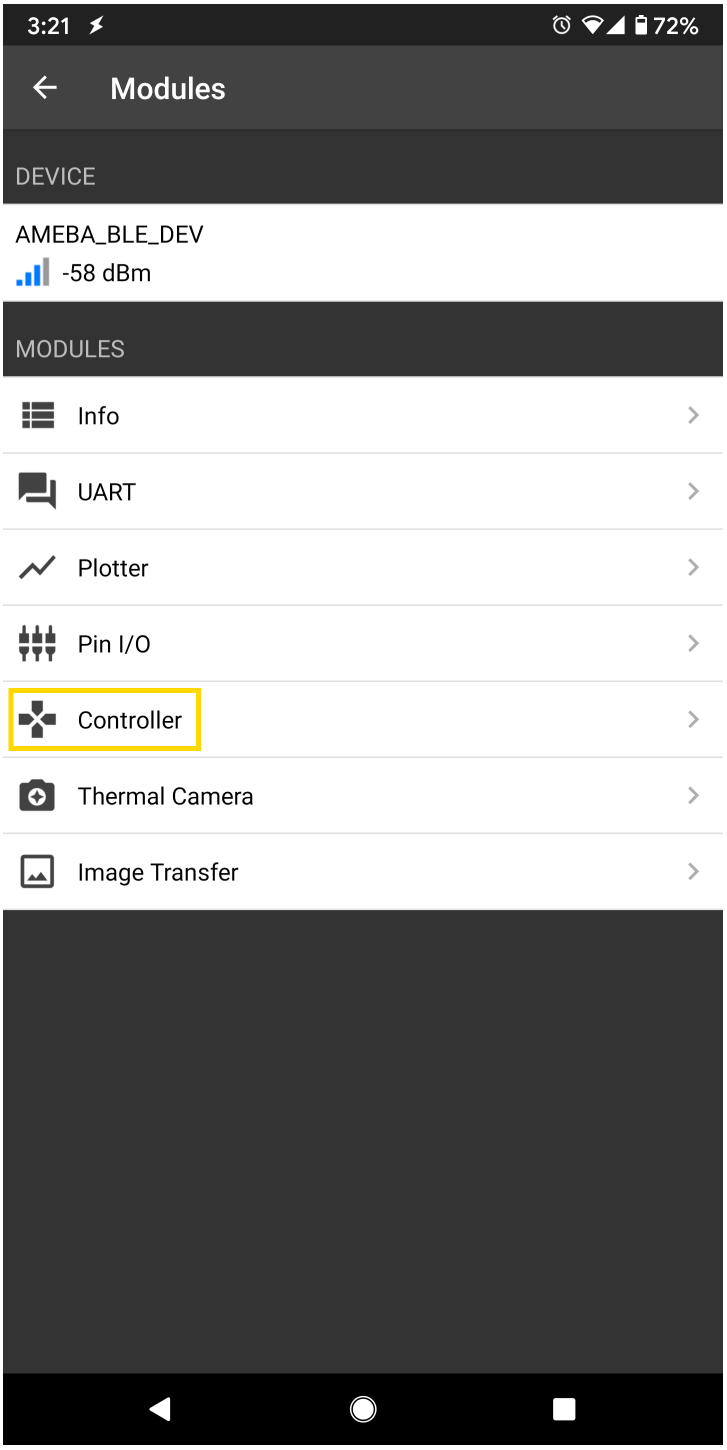
Open the example, “Files” -> “Examples” -> “AmebaBLE” -> “PWM_over_BLEUart”.

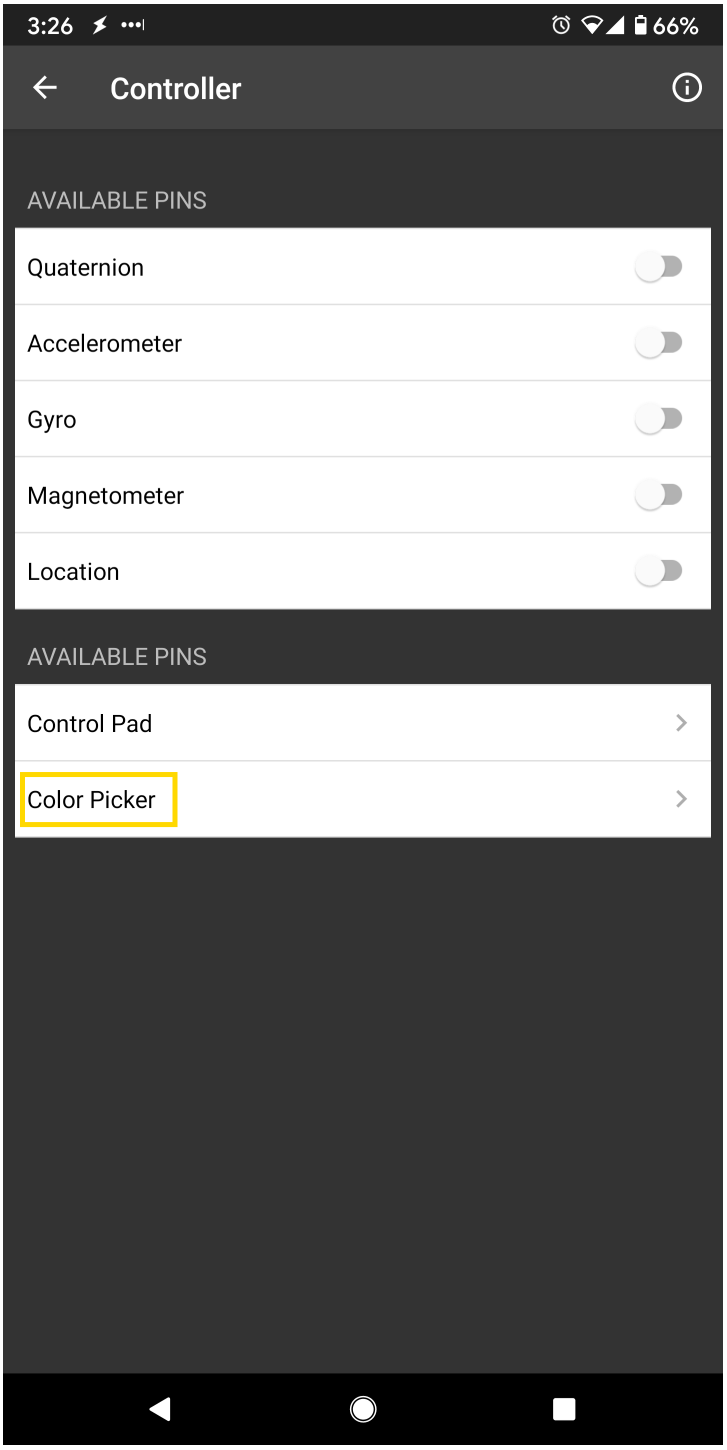
Upload the code and press the reset button on Ameba once the upload is finished.



Open the app on your smartphone, scan and connect to the board shown as “AMEBA_BLE_DEV” and choose the controller -> color picker function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer



Using the color selection wheel, saturation, and brightness sliders, choose a desired color and click select to send the RGB values to the board. You should see the RGB LED change to the matching color.

Code Reference

The RGB values are sent as three consecutive bytes prefixed by "!C" characters. The "!" exclamation mark is used to indicate that the following data is a command, and the "C" character is used to indicate that the data is RGB values. The received UART message is checked in the callback function for "!C" first, otherwise it is treated as a regular message and printed to the serial terminal.

BLE - HID Gamepad

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

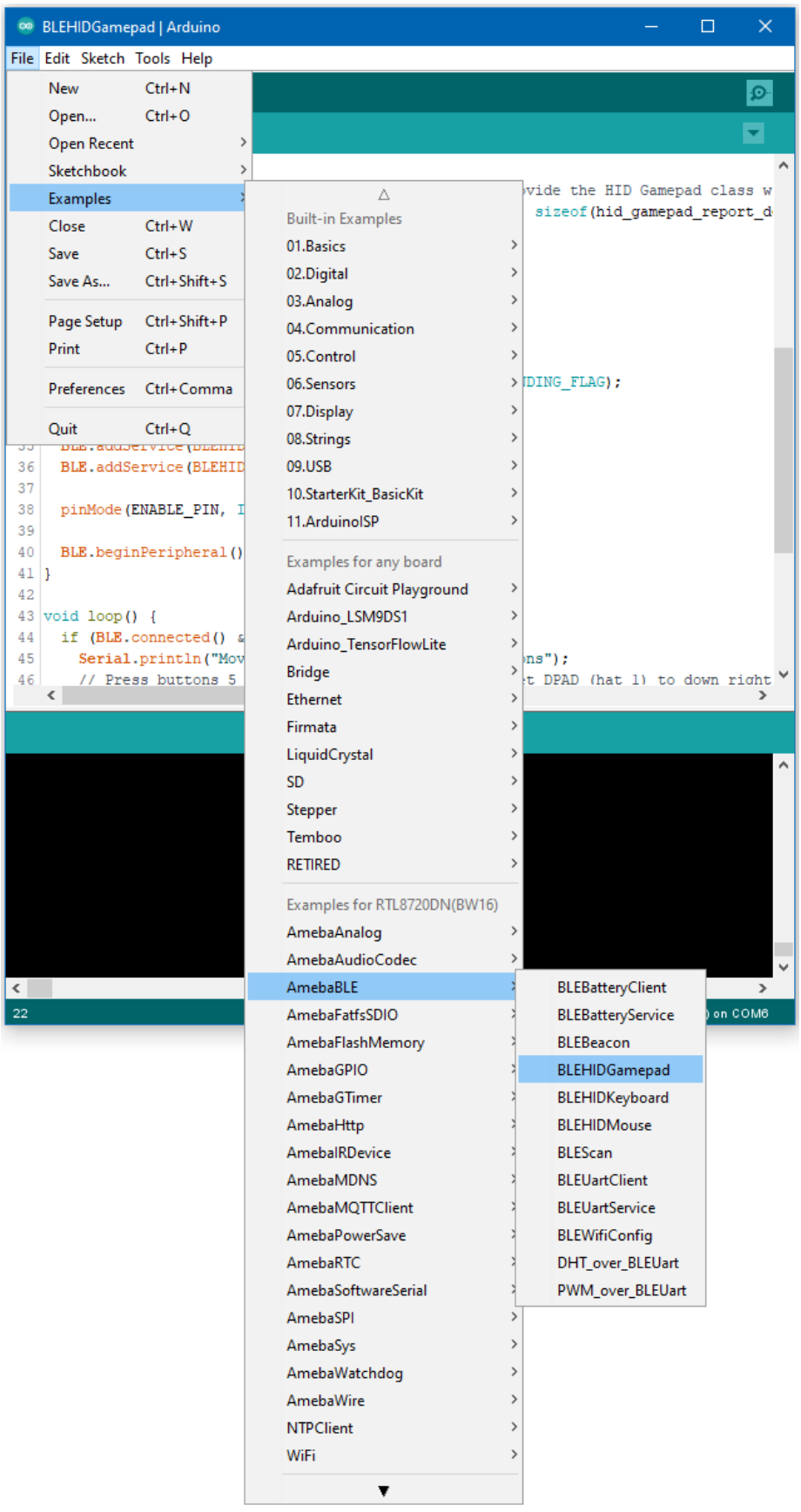
Example

Introduction

In this example, the RTL8722 board emulates a HID gamepad connected using BLE.

Procedure

Open the example, "`Files" -> "Examples" -> "AmebaBLE" -> BLEHIDGamepad`.

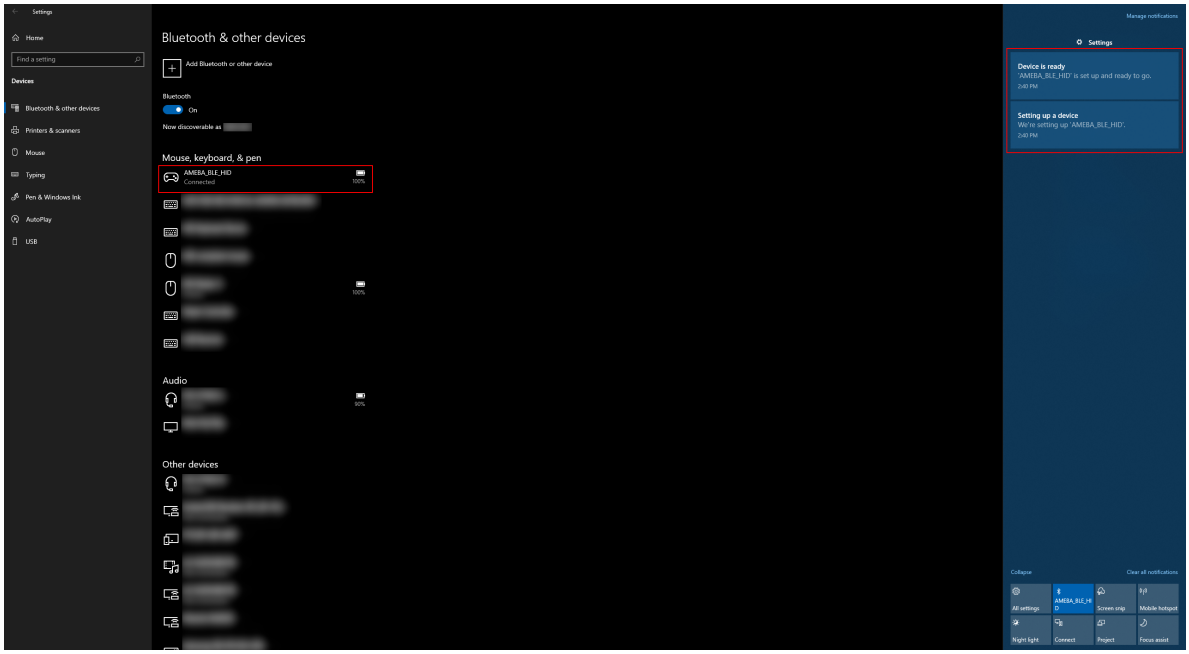

BLEHIDGamepad | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

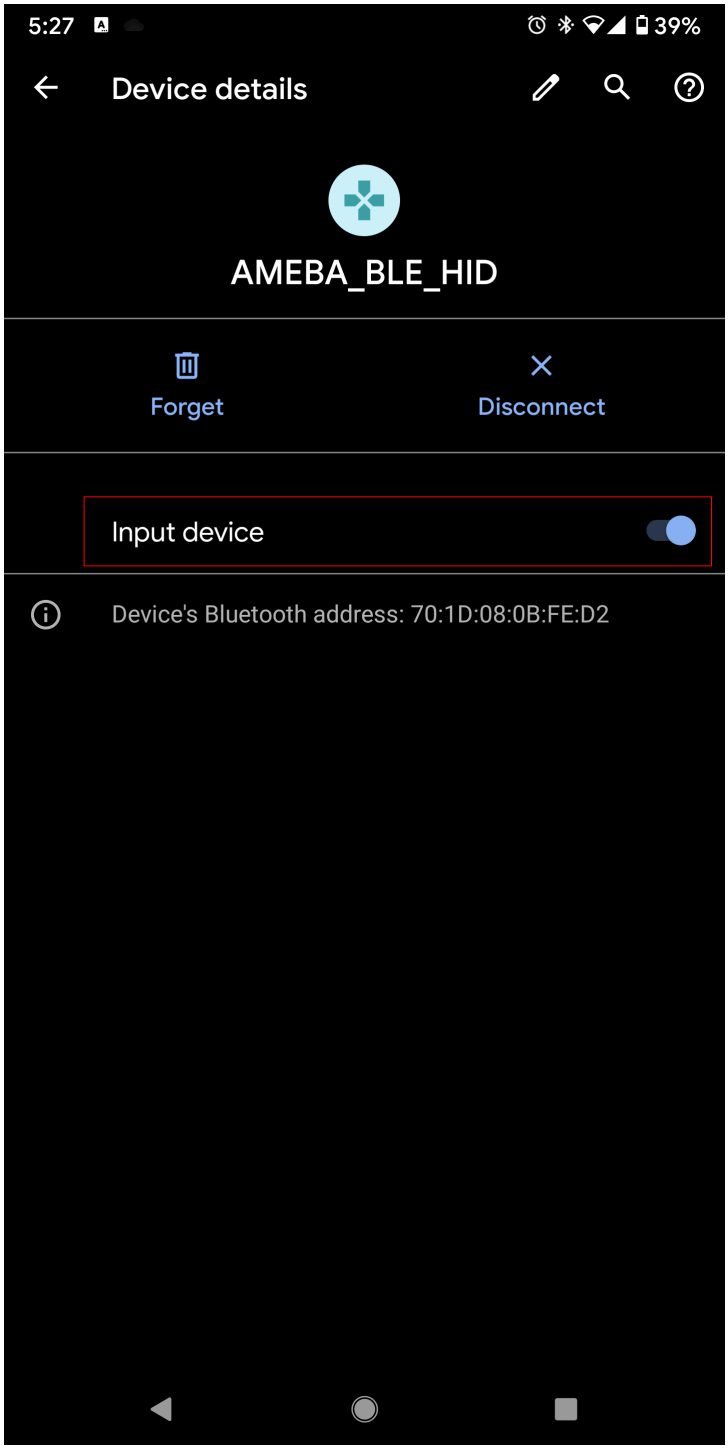
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.



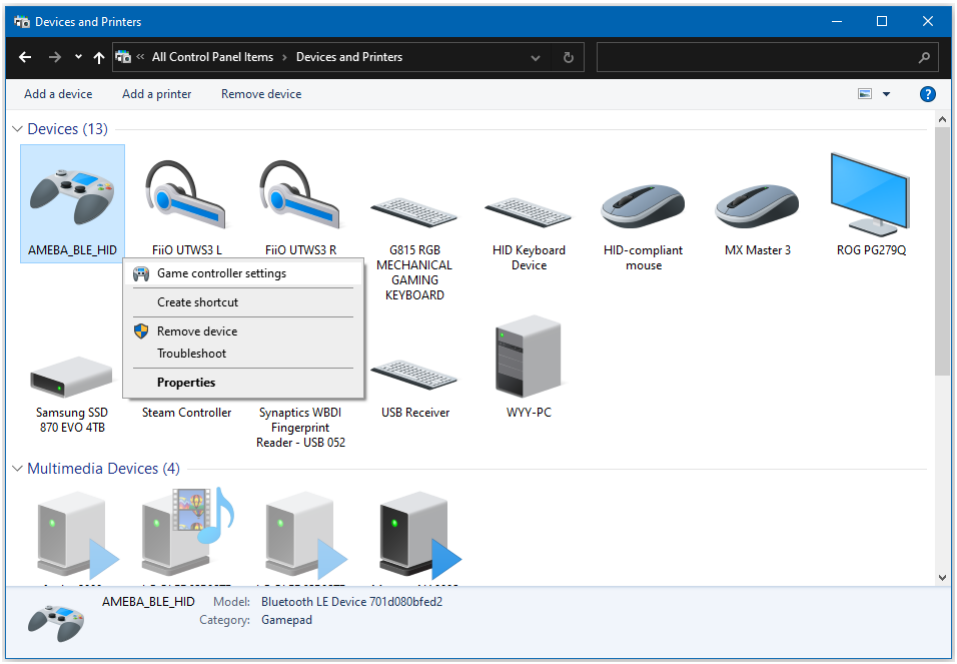
On Android, ensure that “Input device” is enabled for the board.

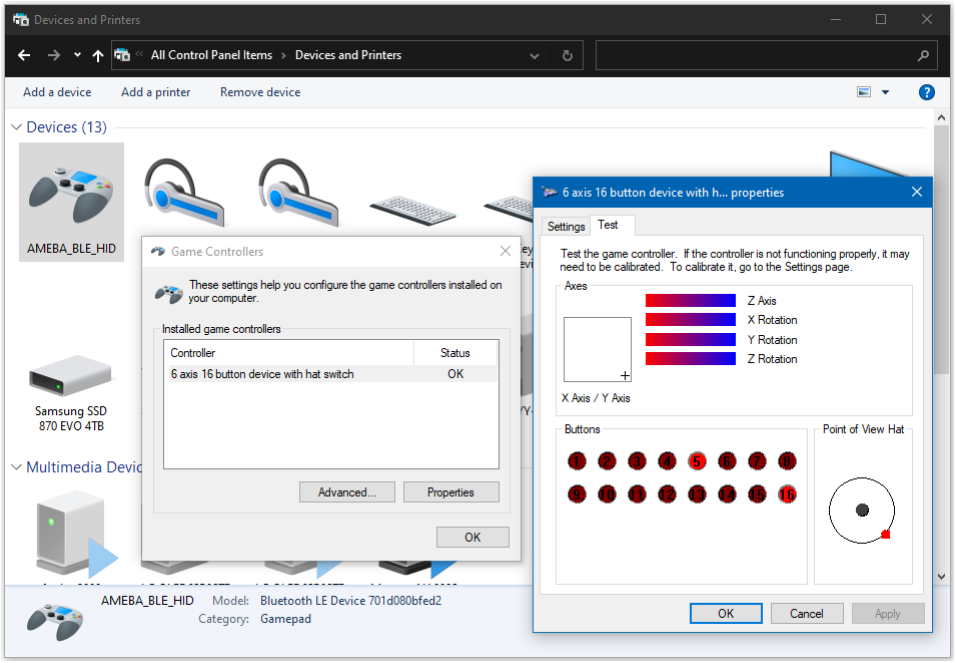


After the Bluetooth connection process is completed, the board is ready to send gamepad input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

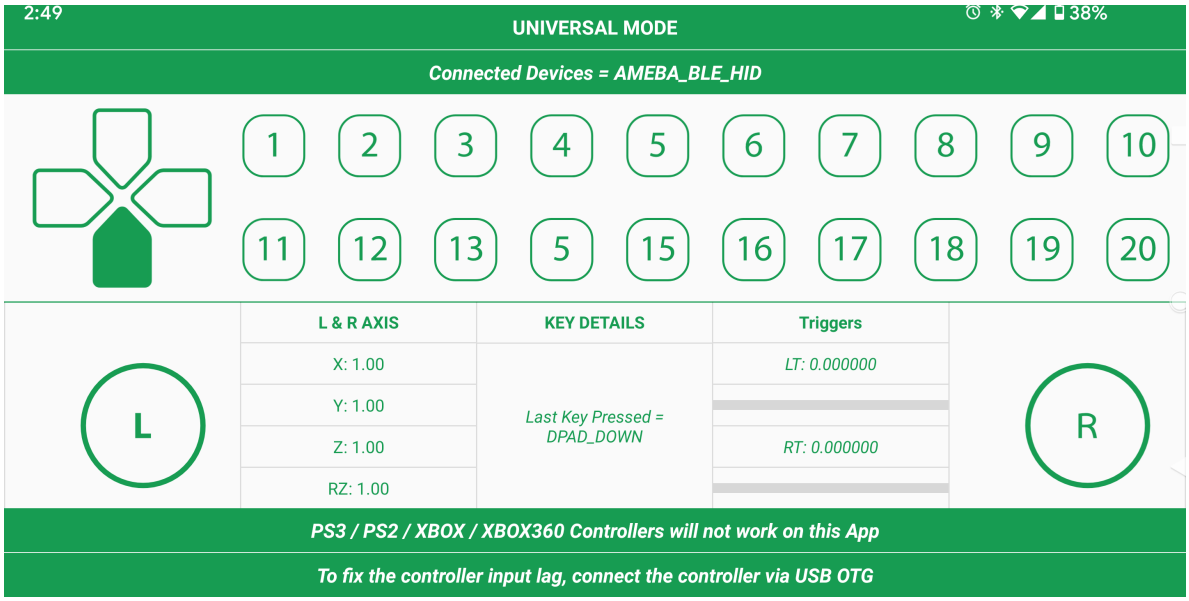
To view the input, open a browser window and go to Gamepad Tester. The connected gamepad device should show up here, and some of the buttons and axes should show changing values.

On Windows, gamepad input can also be viewed by going to “Control Panel” -> “Devices and Printers” -> “AMEBA_BLE_HID” -> “Game Controller Settings” -> “Properties”. The buttons and axes should also show changing values here.





On Android, gamepad testing apps such as Andriod Gamepad Tester can also be used to view the gamepad input.



Code Reference

By default, the board emulates a gamepad with an 8-direction hat switch (d-pad), 6 analog axes and 16 buttons. How the inputs are interpreted is dependent on the host device, and the button ordering may differ between devices. Also, some axes or buttons may be disabled or missing on certain host devices.

BLE - HID Keyboard

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

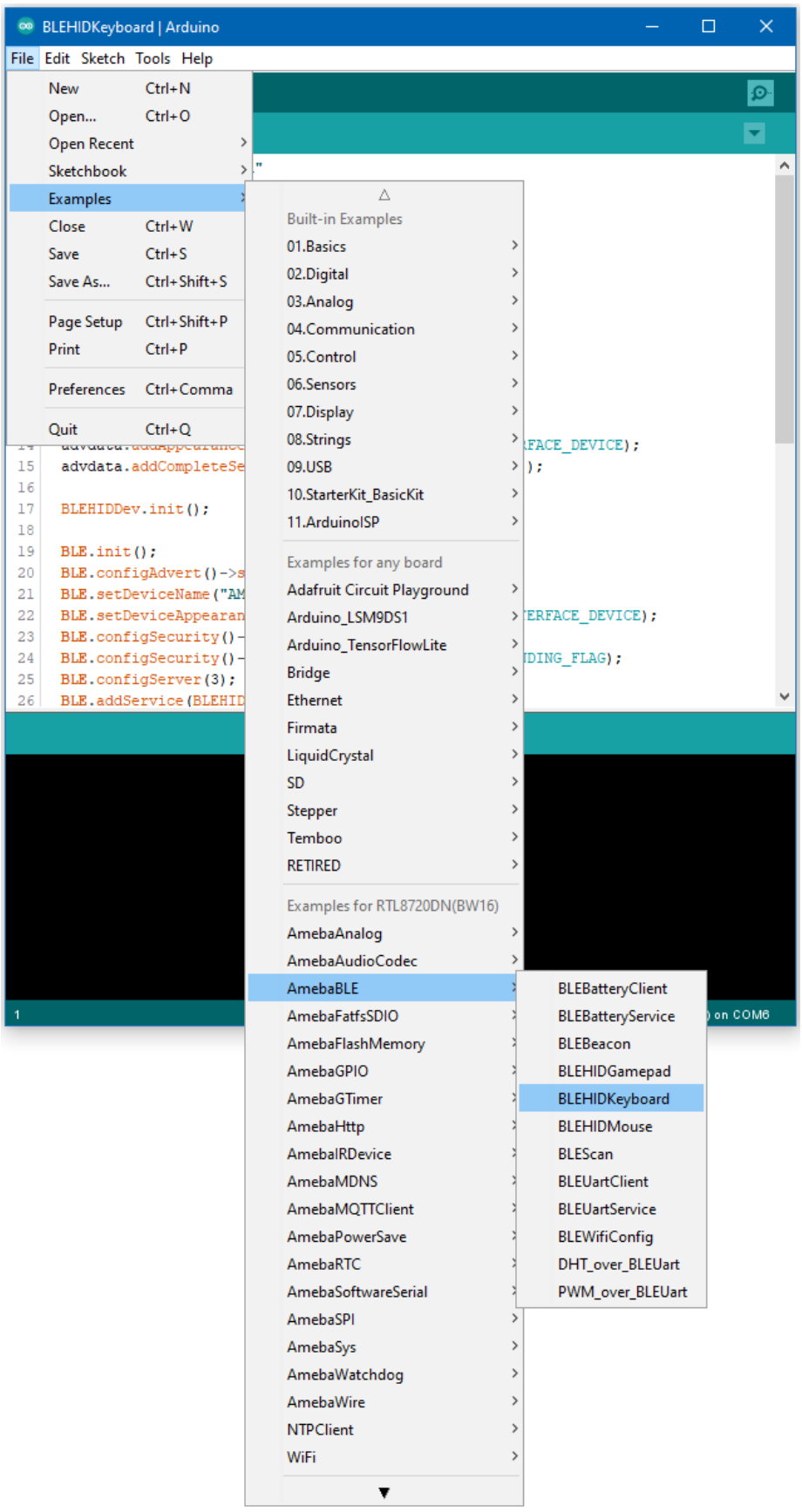
Example

Introduction

In this example, the RTL8722 board emulates a HID keyboard connected using BLE.

Procedure

Open the example, "`Files" -> "Examples" -> "AmebaBLE" -> BLEHIDKeyboard`.

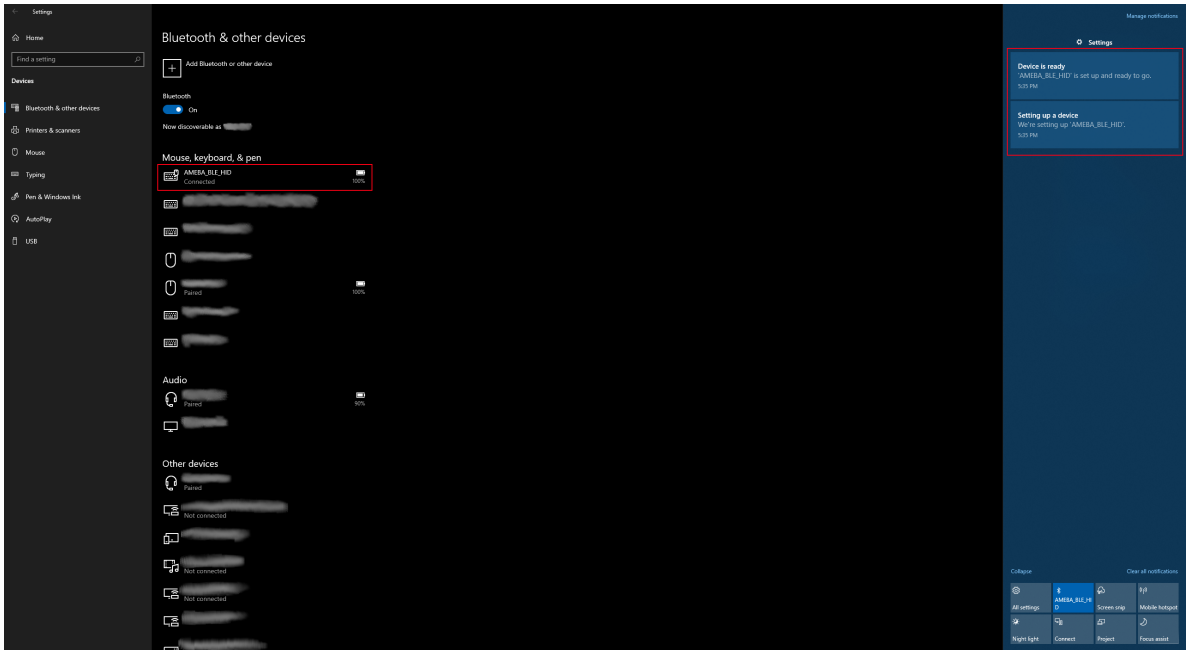

BLEHIDKeyboard | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

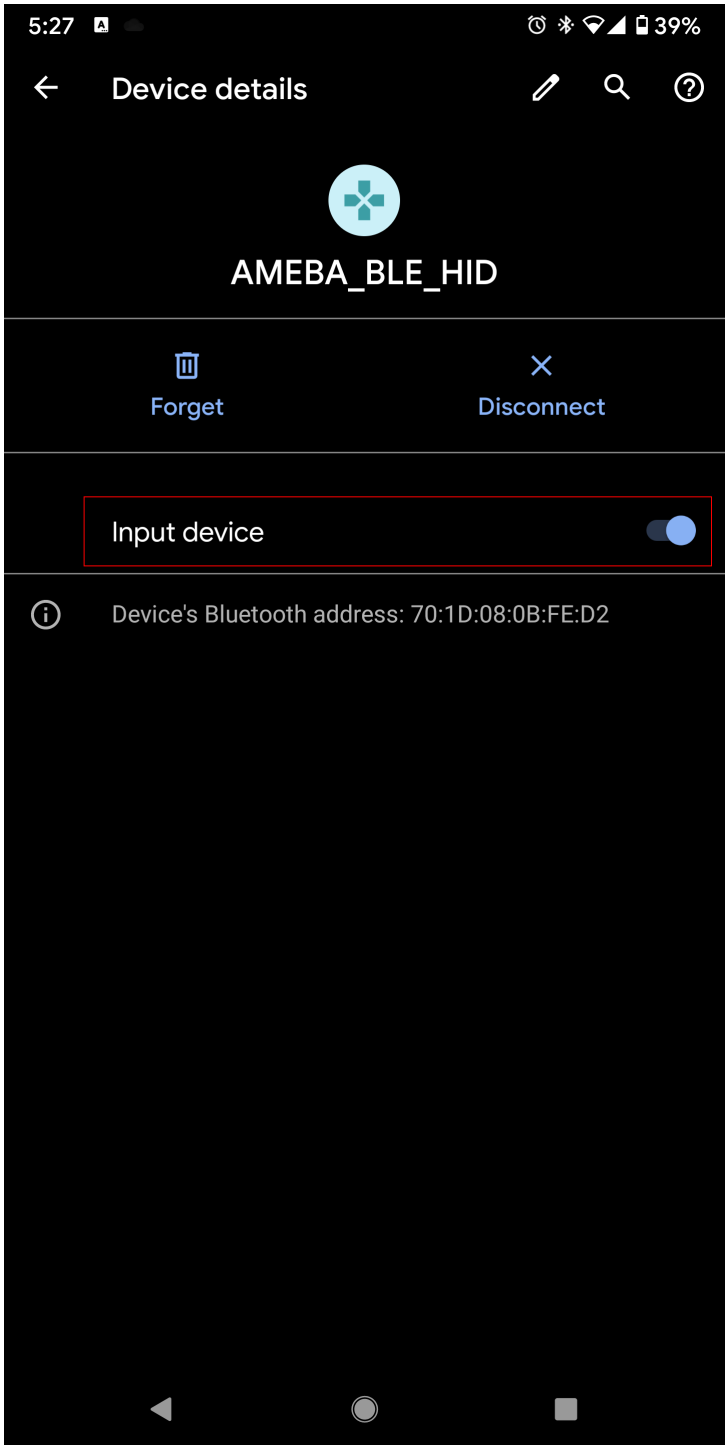
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.

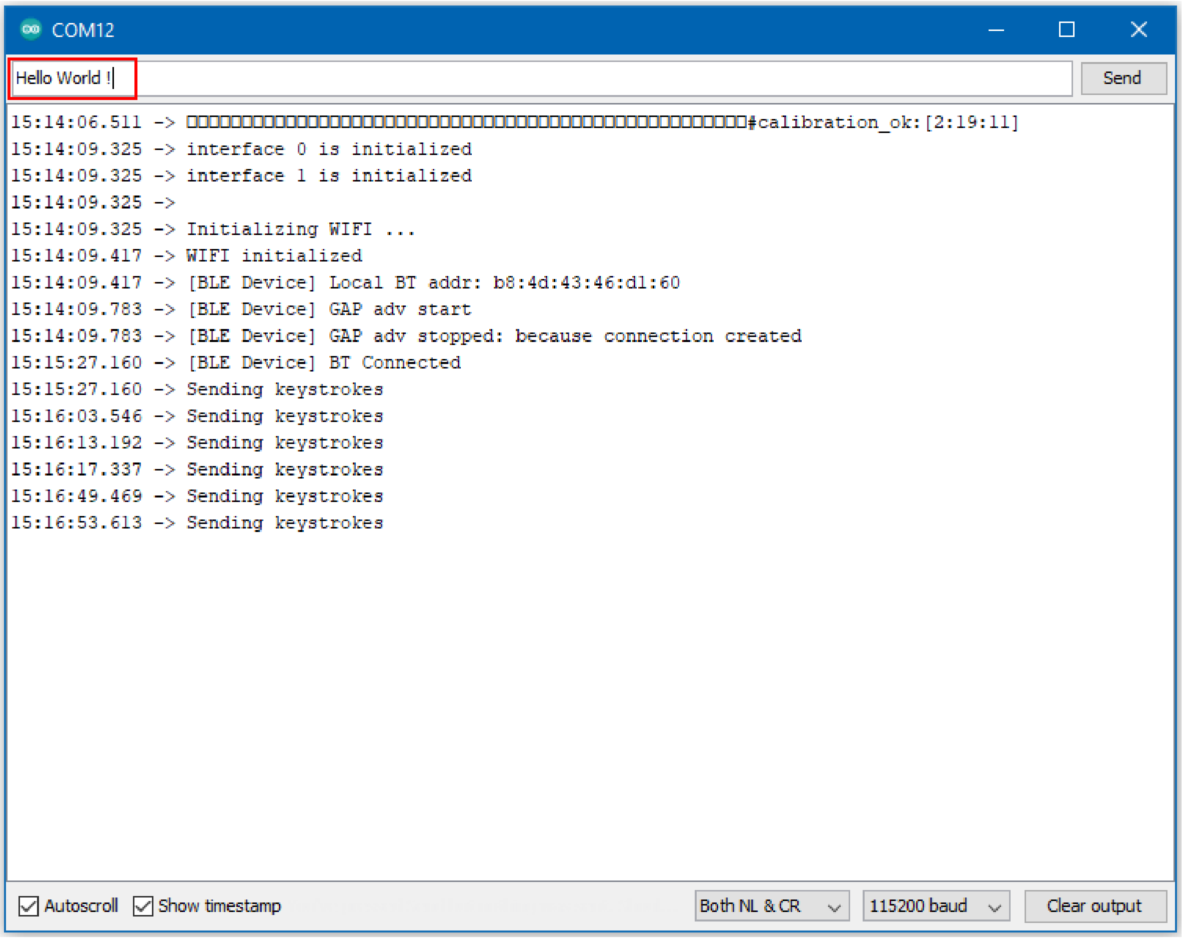


On Android, ensure that “Input device” is enabled for the board.



After the Bluetooth connection process is completed, the board is ready to send mouse input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

You should see the text “Hello World !” typed out and deleted repeatedly.



BLE - HID Mouse

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

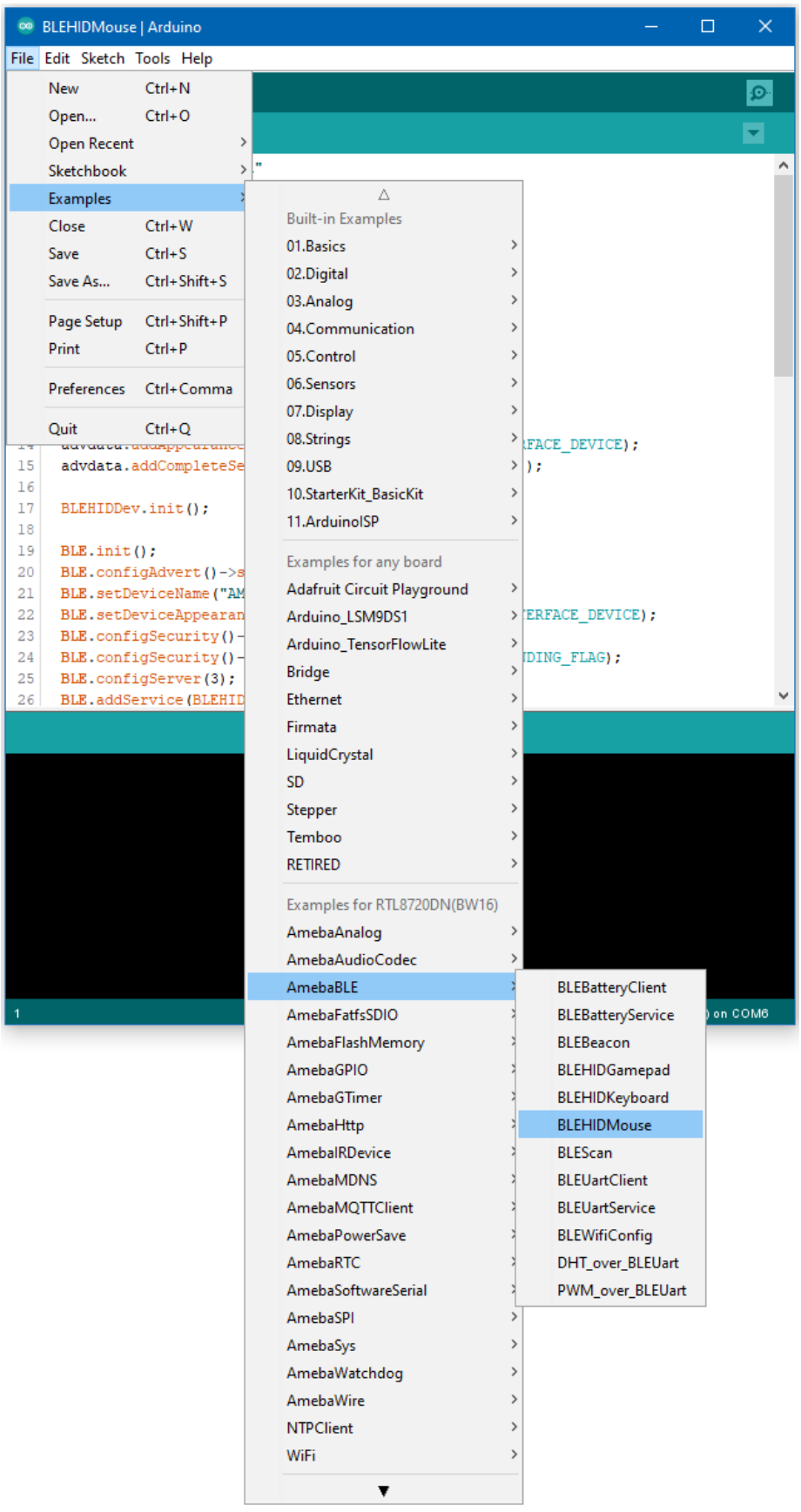
Example

Introduction

In this example, the RTL8722 board emulates a HID mouse connected using BLE.

Procedure

Open the example, “`Files`” `->` “`Examples`” `->` “`AmebaBLE`” `->` “`BLEHIDMouse`”.

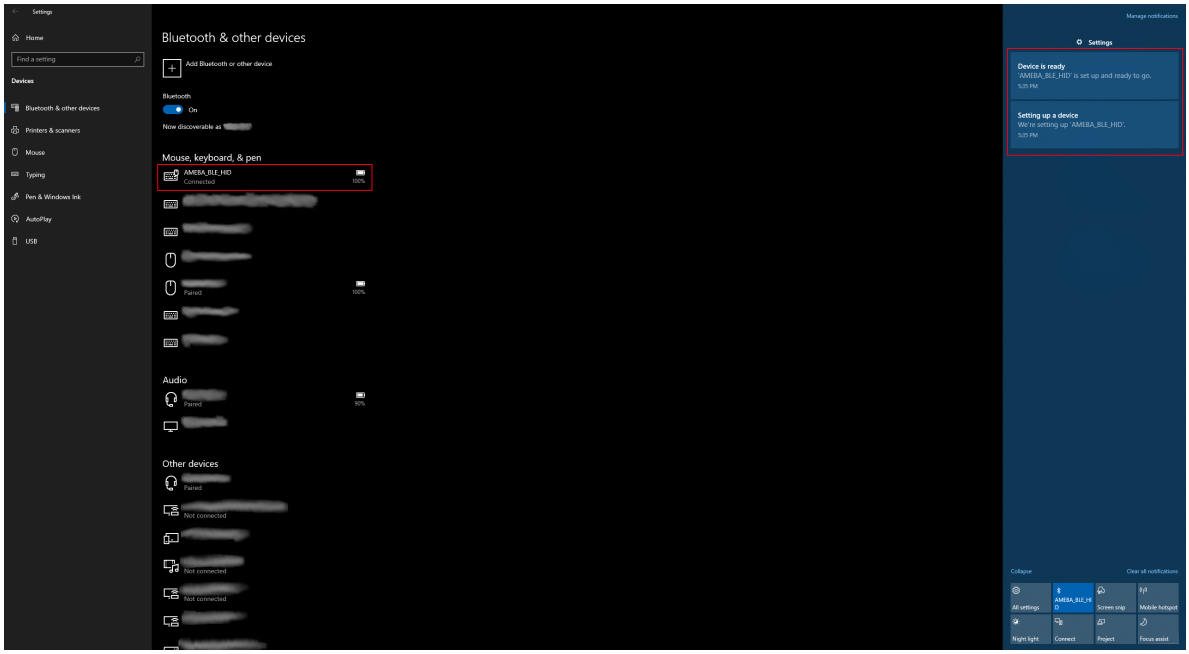

BLEHIDMouse | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

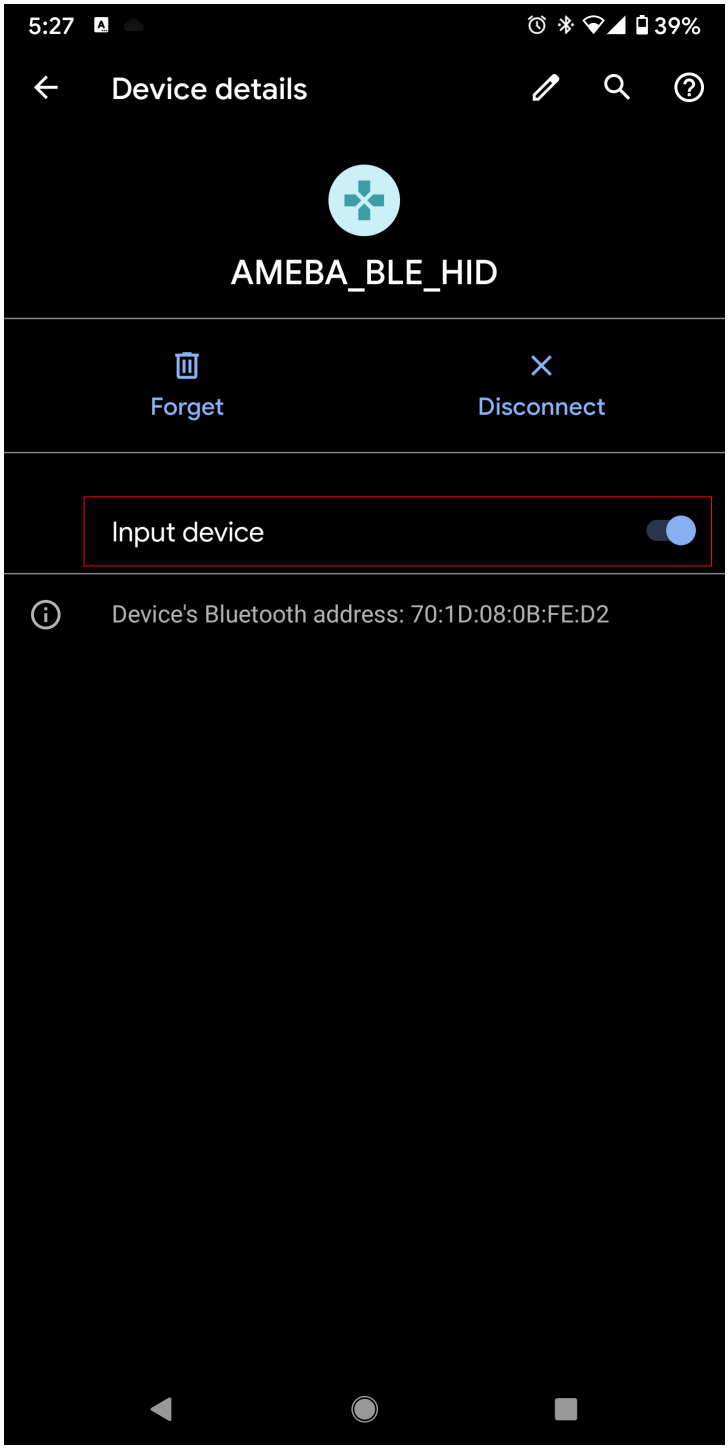
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.



On Android, ensure that “Input device” is enabled for the board.



After the Bluetooth connection process is completed, the board is ready to send mouse input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

You should see the mouse cursor move around four points in a square, performing right and left clicks, and scrolling up and down.

Code Reference

How the mouse input is interpreted is dependent on the host system. Some systems, such as mobile operating systems, may not support all mouse button input functions.

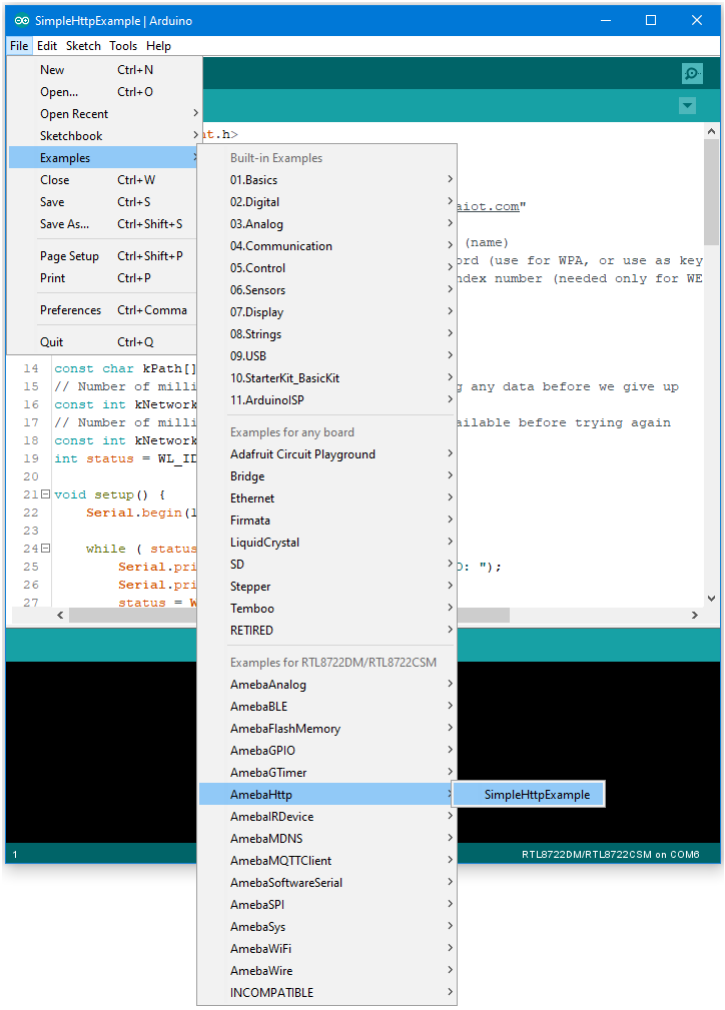
HTTP - Retrieve HTTP websites from the Internet

Materials

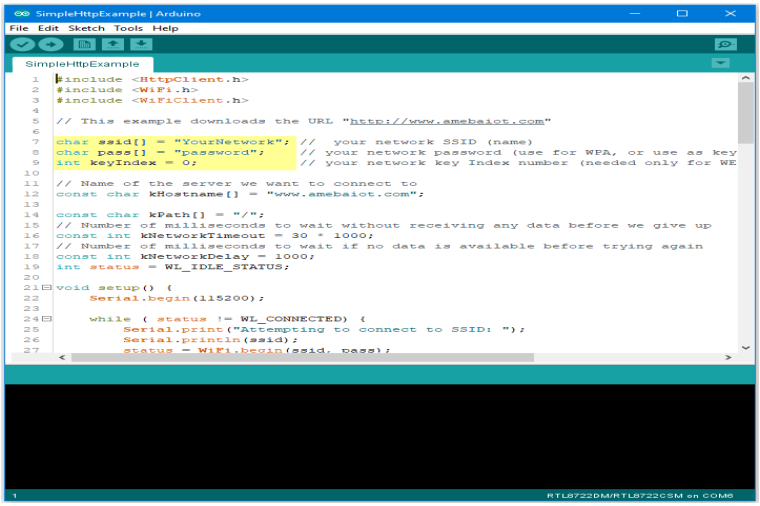
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

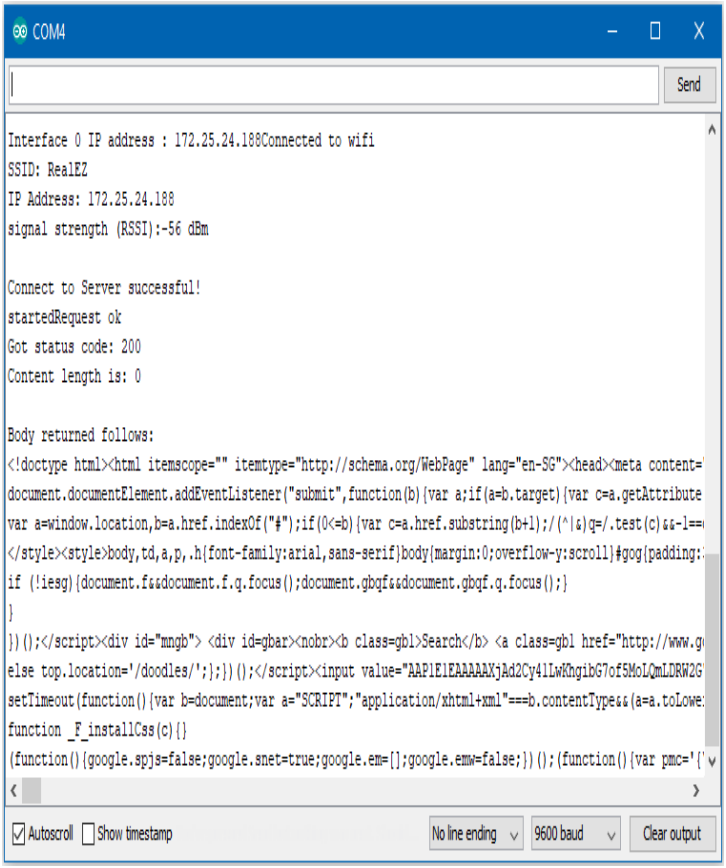
In this example, the HttpClient library is used to retrieve a webpage using the HTTP protocol.
First, make sure that the correct Ameba development board is selected in "Tools" -> "Board"
Then open "`File" -> "Examples" -> "AmebaHttp" -> "SimpleHttpExample`"



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in the Arduino IDE and you can see the information retrieved from the website.



Code Reference

Use `WiFi.begin()` to establish WiFi connection:
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use WiFiClient to create a client to handle the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiClient
Use HTTPClient to create a client to handle the HTTP connection.

Use `http.get()` to send a GET request to the website.

HTTP - Set up Server to Control LED

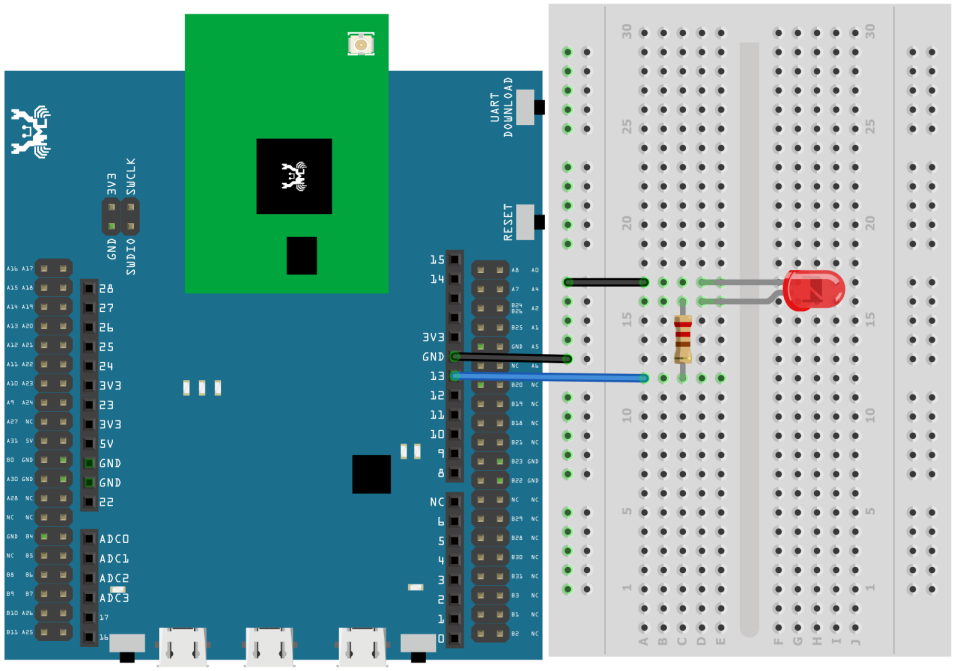
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Breadboard x 1
- LED x 1
- 1KΩ Resistor x 1

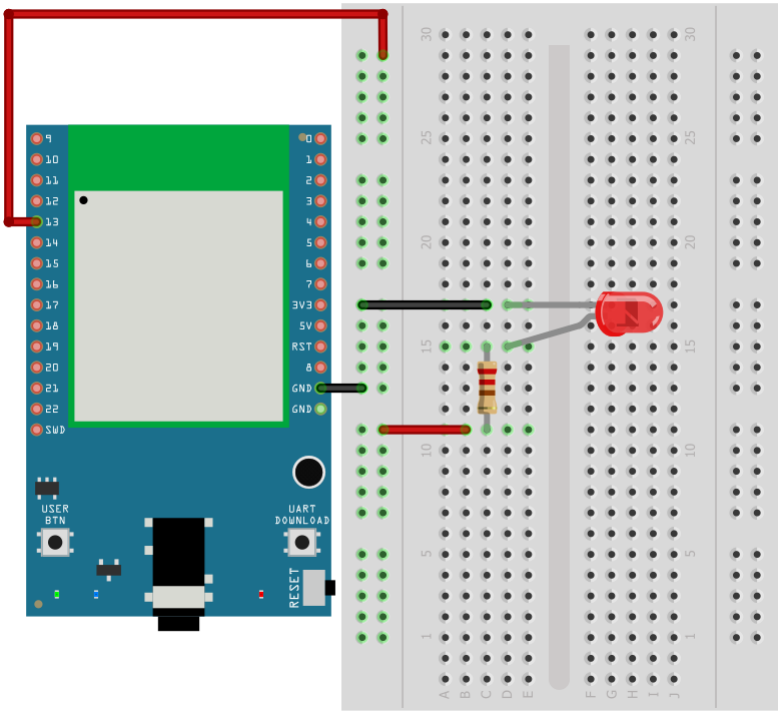
Procedure

In this example, we connect Ameba to WiFi and use Ameba as server, the user can control the LED on/off through a webpage.
First, connect Ameba with the LED.
In a LED, the longer pin is the positive pole, and the shorter pin is the negative pole. So, we connect the shorter pin to GND and connect the longer pin to D13. Additionally, to avoid the electric current exceeds the tolerance of the LED and causes damage, we connect a resistance on the positive pole.

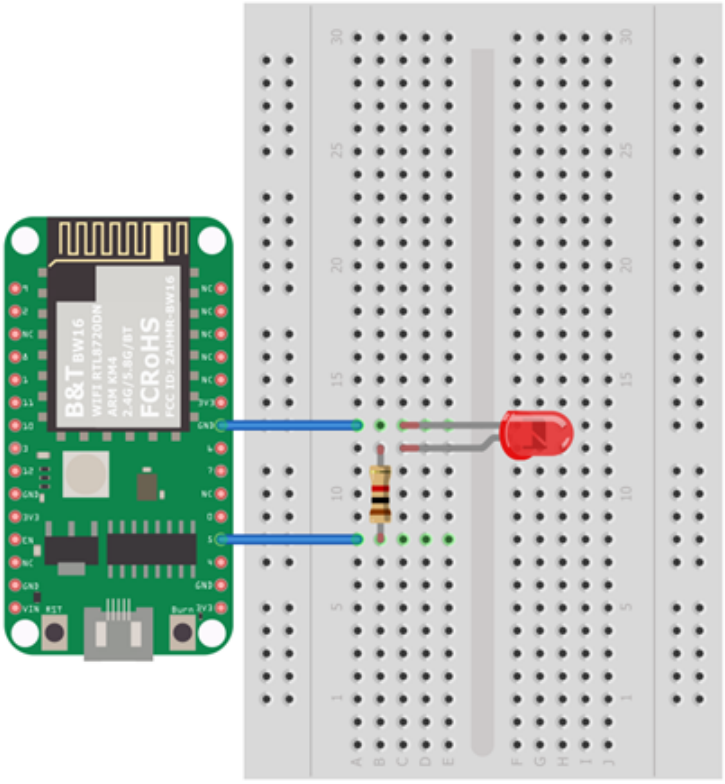
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

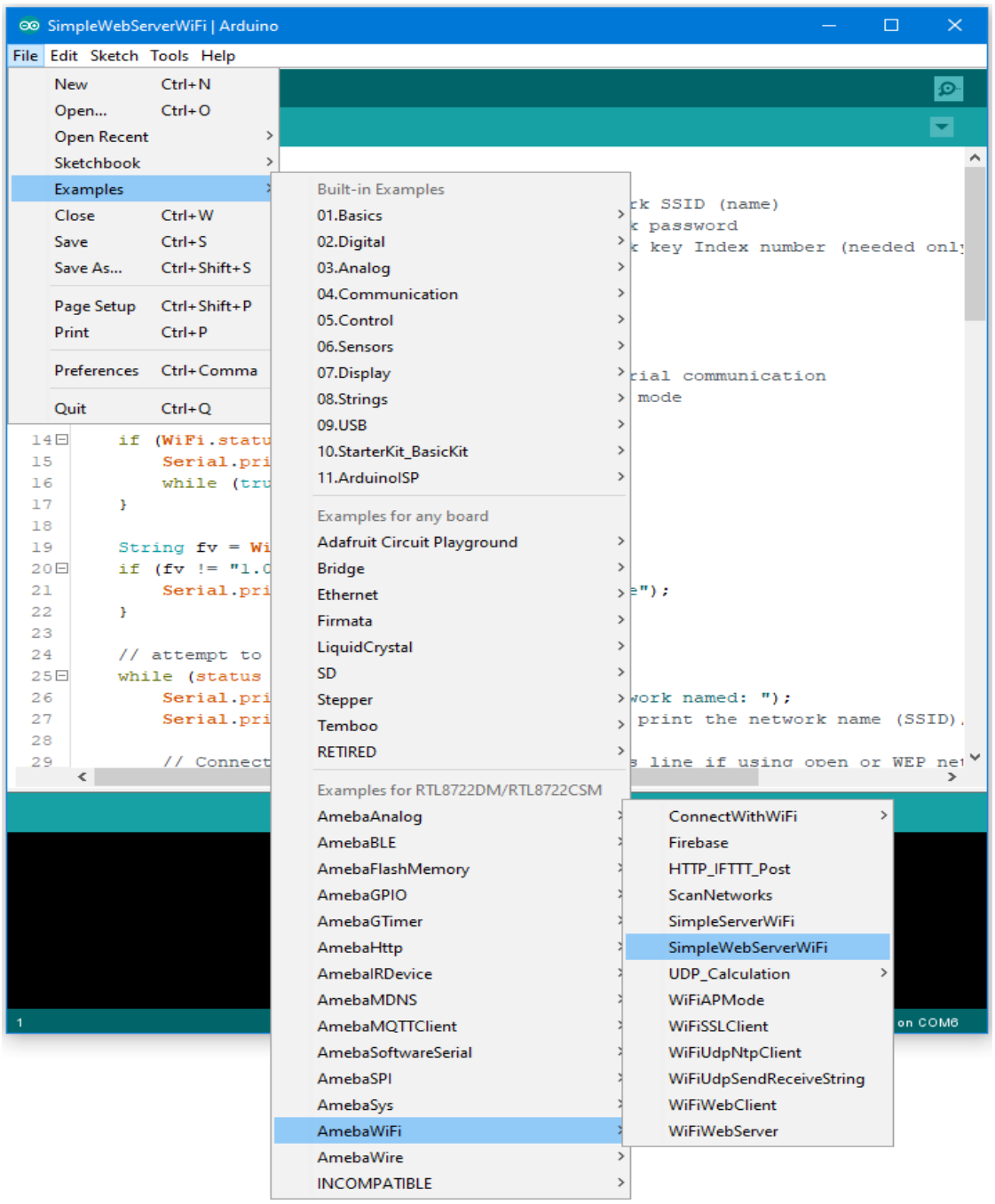


BW16 Wiring Diagram:

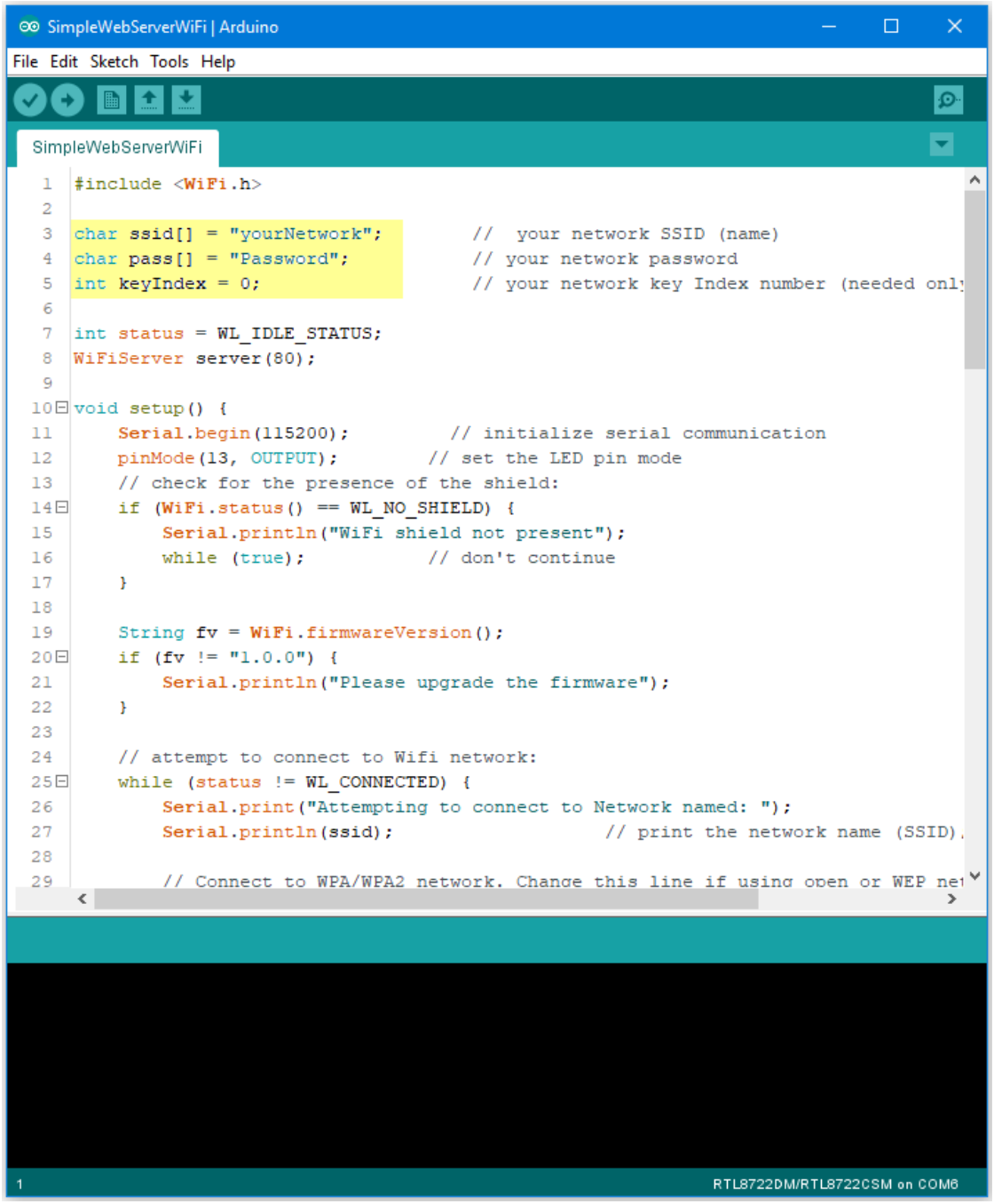


Note:
For BW16 board, you may consider to re-define "LED_PIN" macro to **10** for built-in green LED, or **11** for blue built-in LED, or **12** for red built-in LED to avoid using extra components.

Then open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`SimpleWebServerWiFi`"



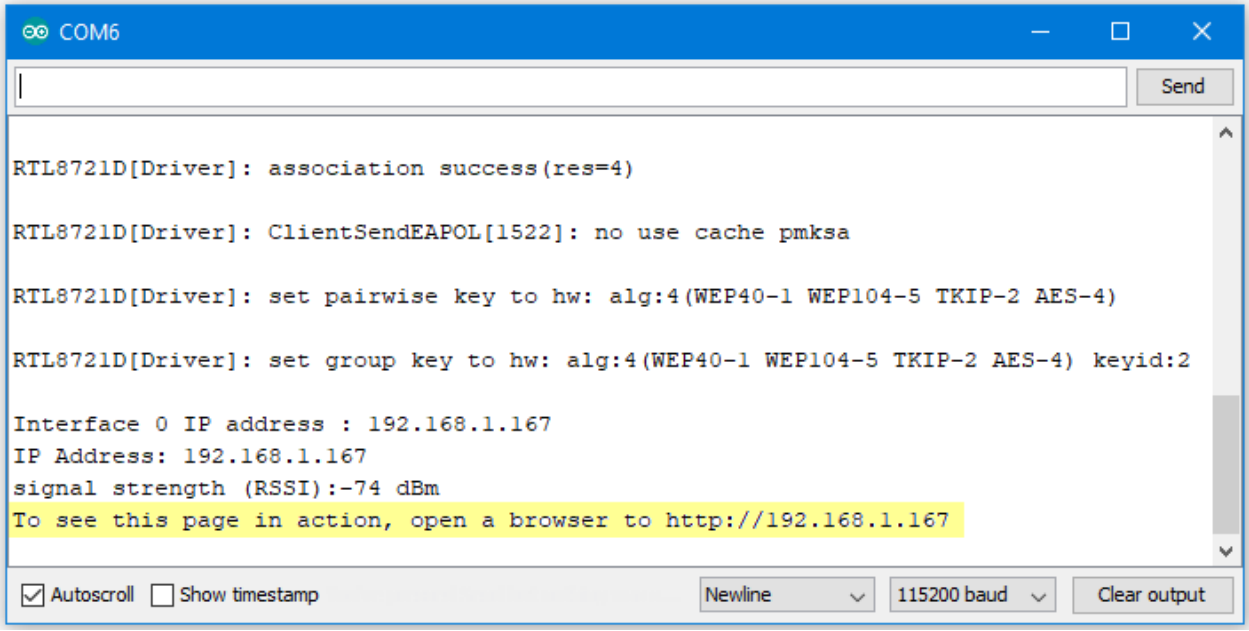
In the sample code, modify the highlighted snippet to corresponding information.



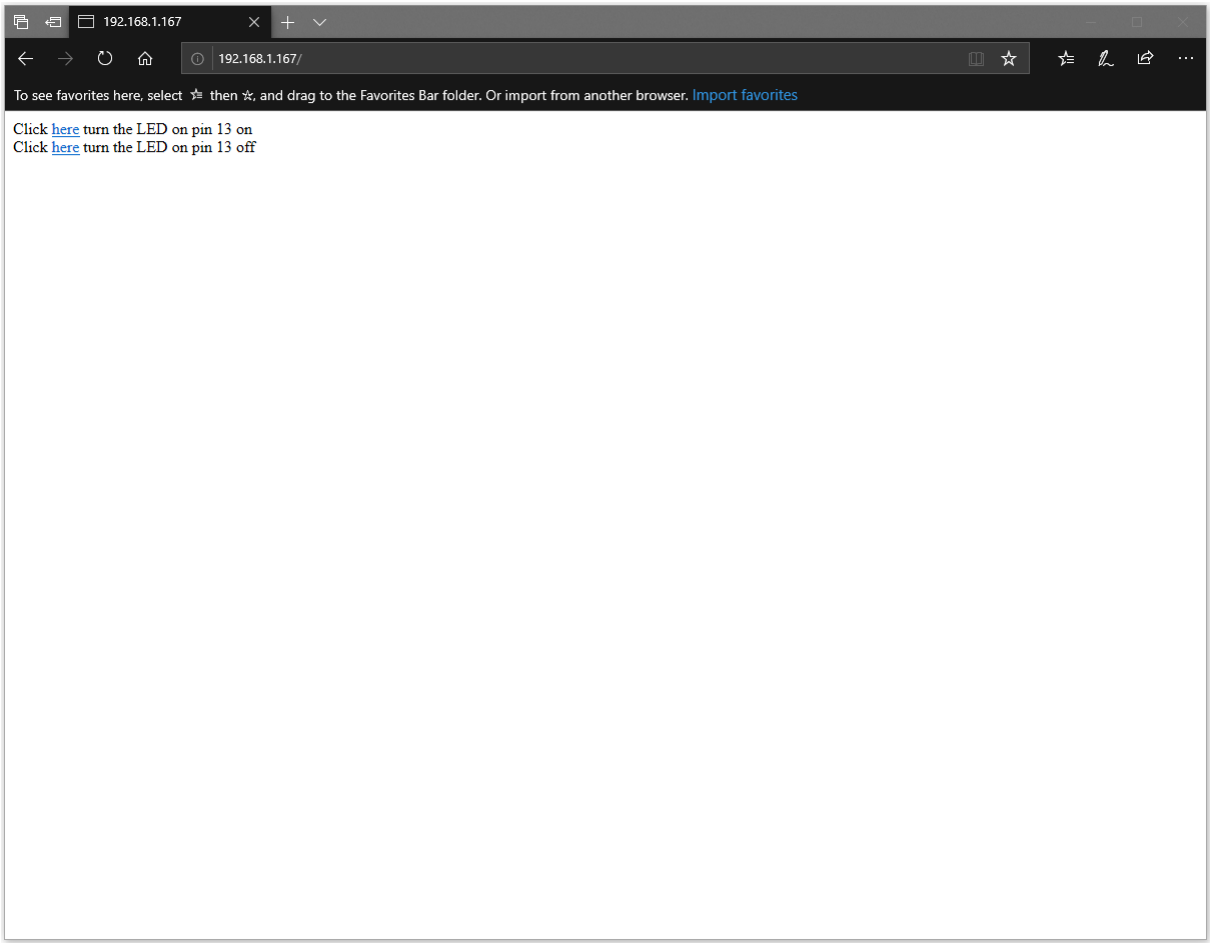
Upload the code and press the reset button on Ameba. When the connection is established, you will see the message:

```
"To see this page in action, open a browser to http://xxx.xxx.xxx.xxx"
```

in the Arduino IDE as shown in the figure:



Next, open the browser of a computer or a cell phone under the same WiFi domain, enter the address in the message.



In the webpage, you can turn on/off the LED.

Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use WiFi.RSSI() to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to get whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

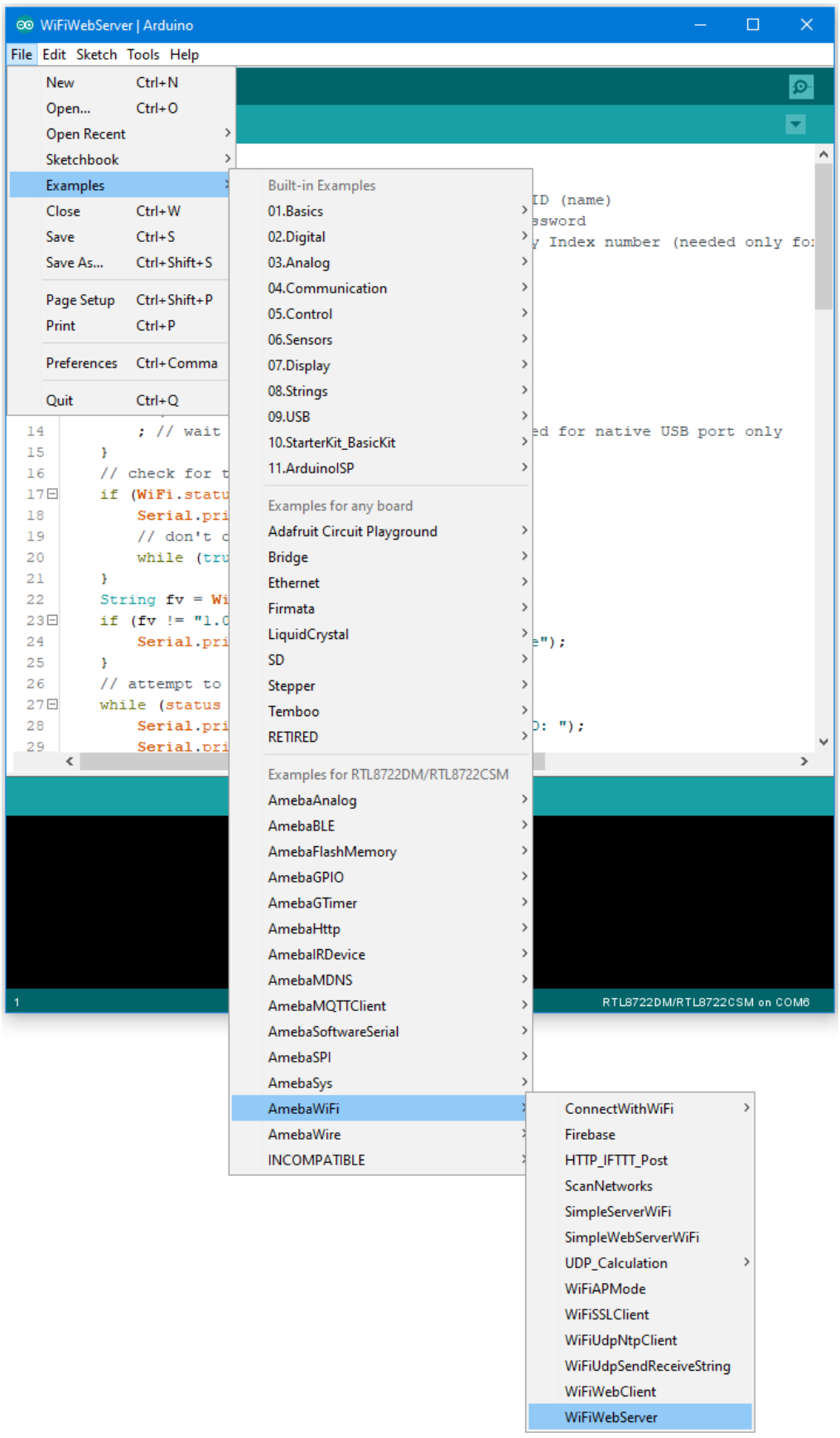
HTTP - Set up Server to Get the Ameba Status

Materials

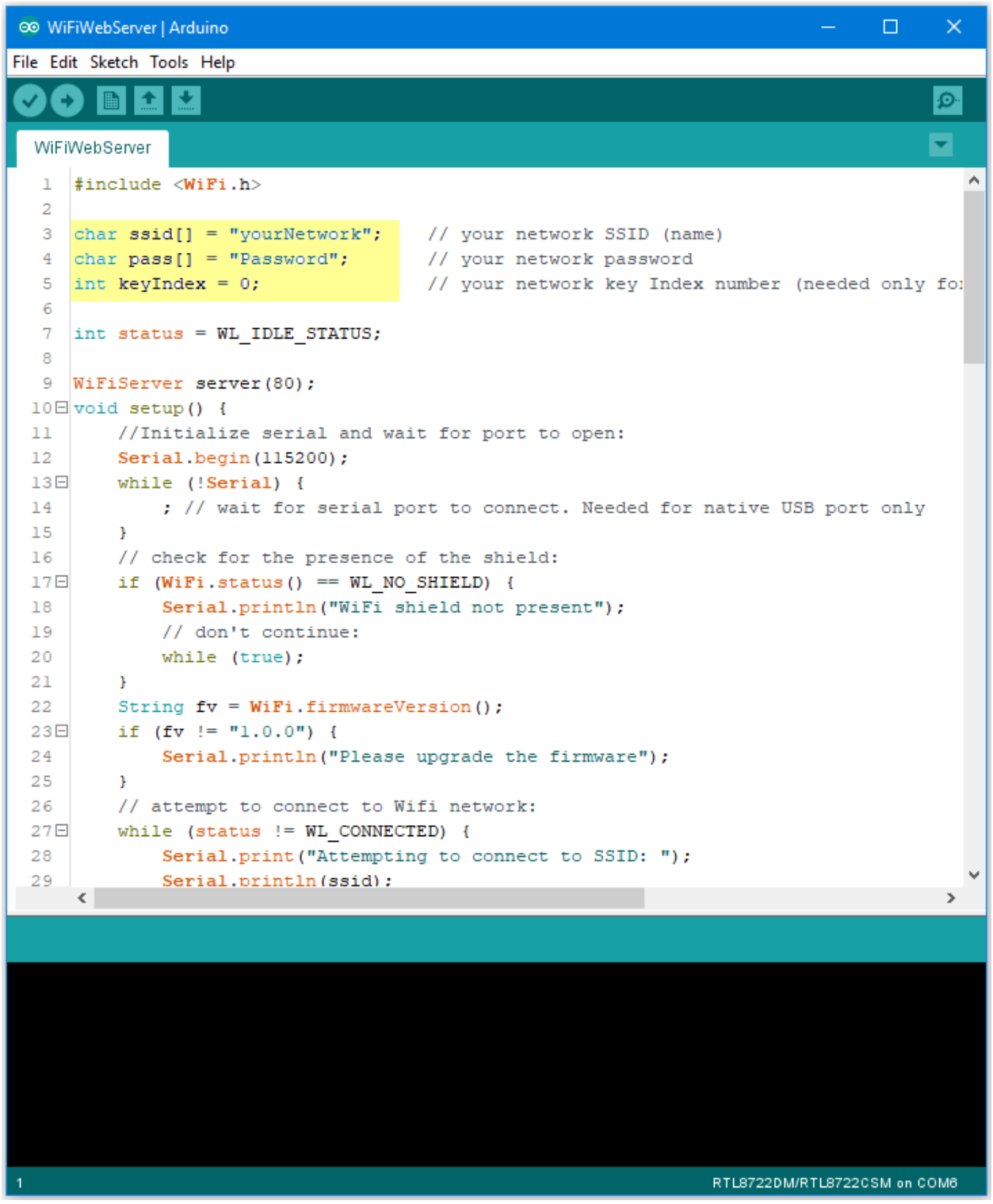
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

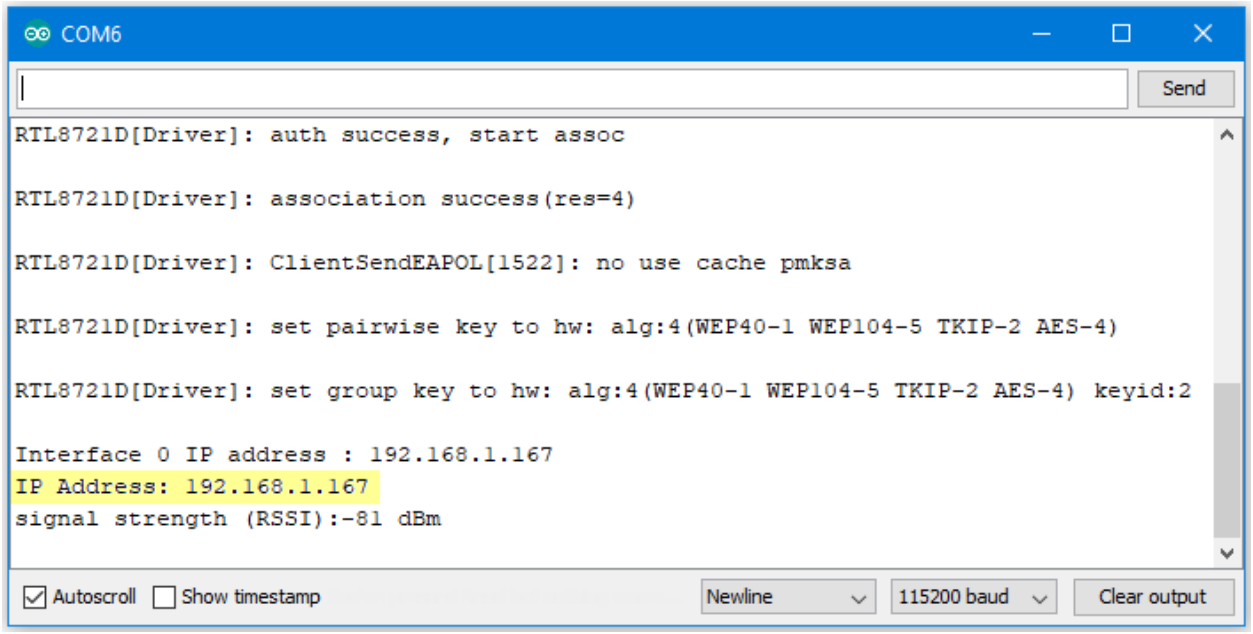
In this example, we connect Ameba to WiFi and use Ameba as server to send message to connected client.
First, open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiWebServer`"


WiFiWebServer | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
ConnectWithWiFi
Firebase
HTTP_IFTTT_Post
ScanNetworks
SimpleServerWiFi
SimpleWebServerWiFi
UDP_Calculation
WiFiAPMode
WiFiSSLClient
WiFiUdpNtpClient
WiFiUdpSendReceiveString
WiFiWebClient
WiFiWebServer
RTL8722DM/RTL8722CSM on COM6

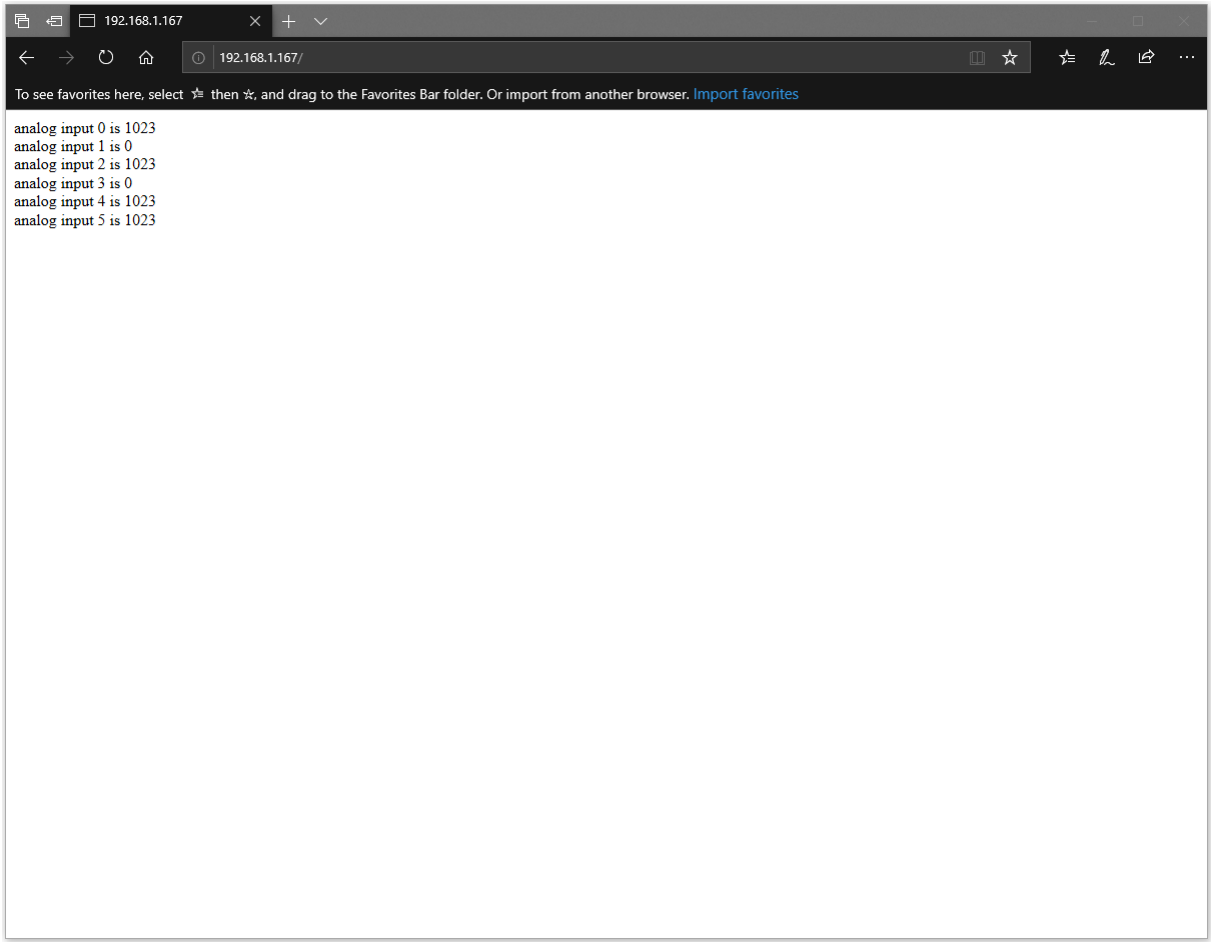
In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. After connecting to WiFi, Ameba starts to run as server. The IP of the server is shown in the serial monitor, and port is 80.



We connect to the server in a browser, and we can see the data sent from the server.



Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
se `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to check whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

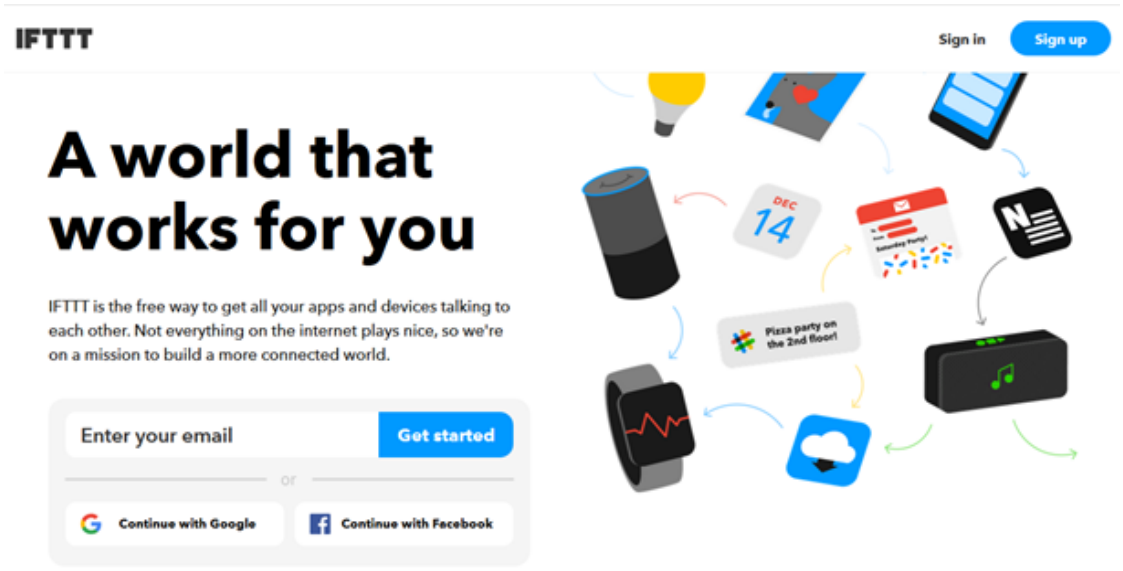
HTTP - Use IFTTT for Web Service

Introduction to IFTTT

IFTTT, known as If This Then That, is a website and mobile app and free web-based service to create the applets, or the chains of simple conditional statements. The applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, Pinterest etc.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- An account from https://ifttt.com/ , in order to access IFTTT service*



Note: Upon log in, there are several cloud and online services that are integrated with IFTTT platforms.

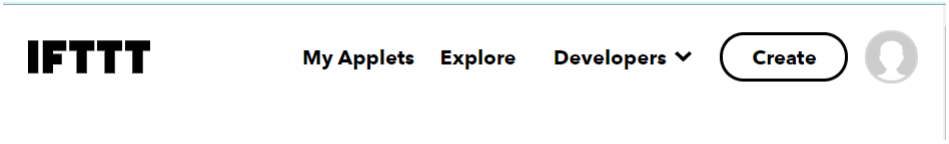
Example

- Generate Applet from IFTTT

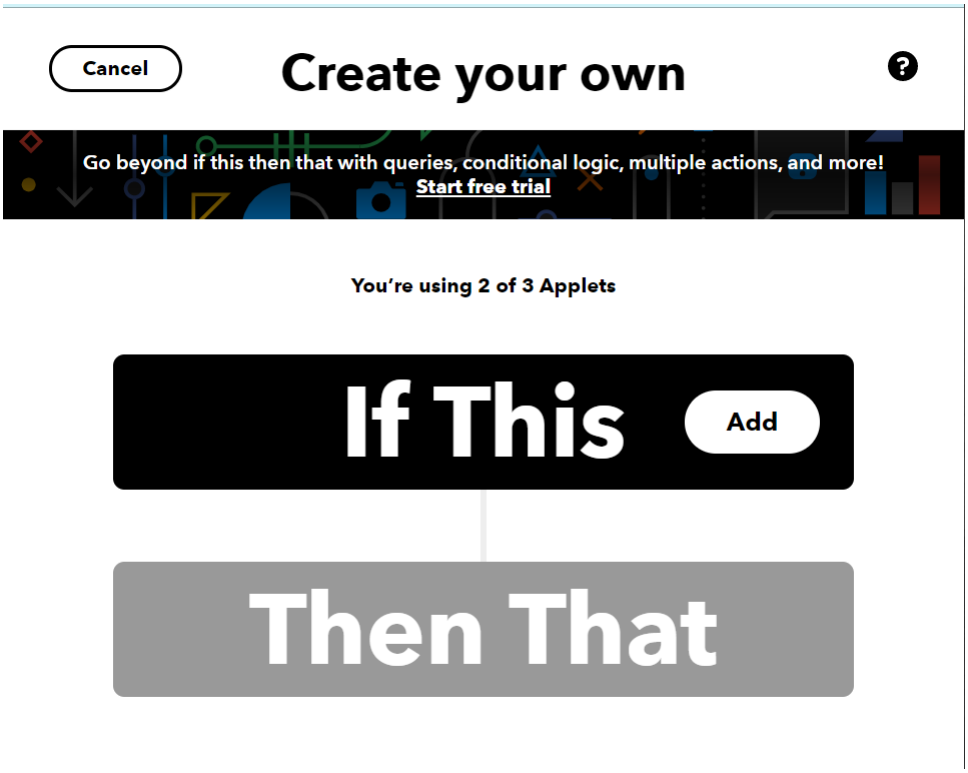
  In this example, we obtain an example of IFTTT Applet to send email to specified recipient.

  To run the example, HTTP POST feature of the Ameba is used to post a simple webhook service that is received by IFTTT platform and in turn be used to trigger a response (sending an email).

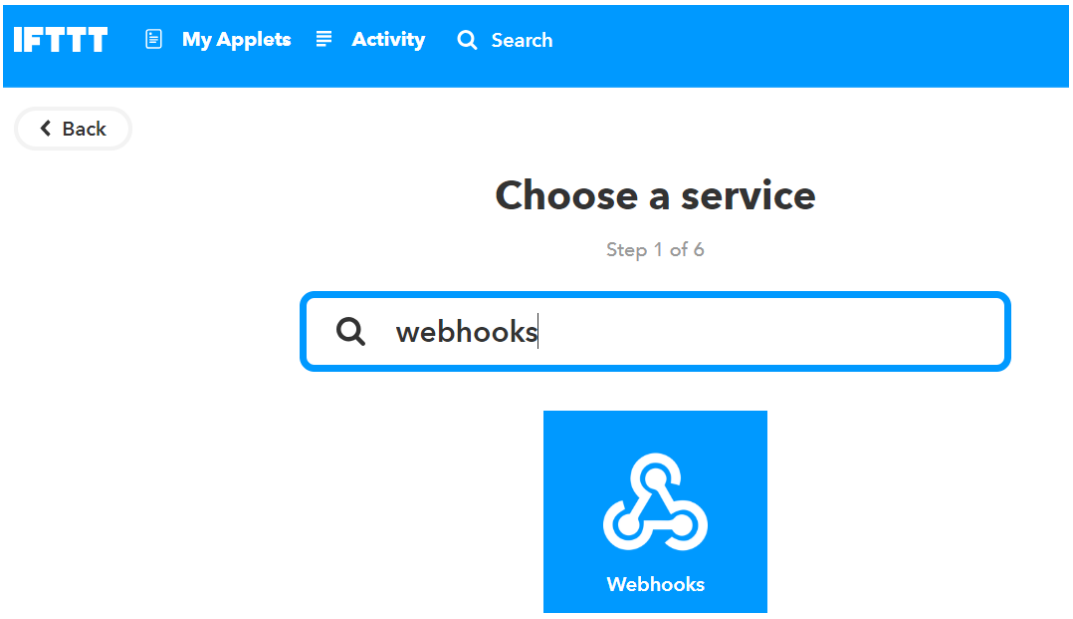
  After logging in https://ifttt.com/, click **Create** from the top bar.

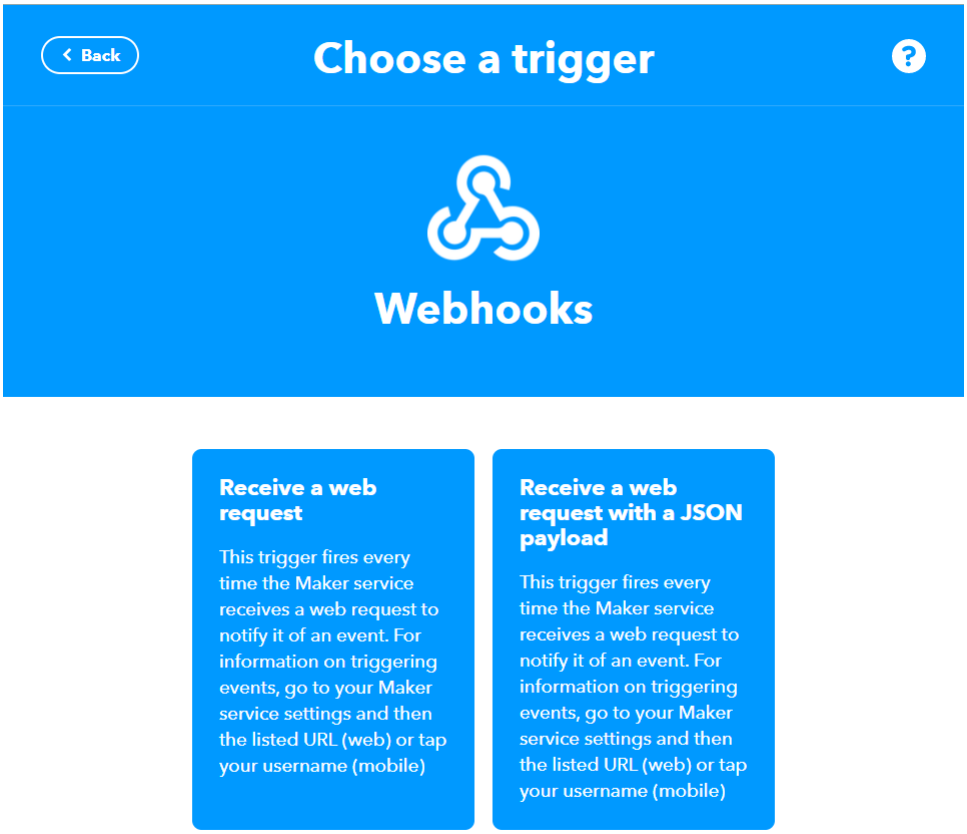
  Click **"Add"** to add the trigger.



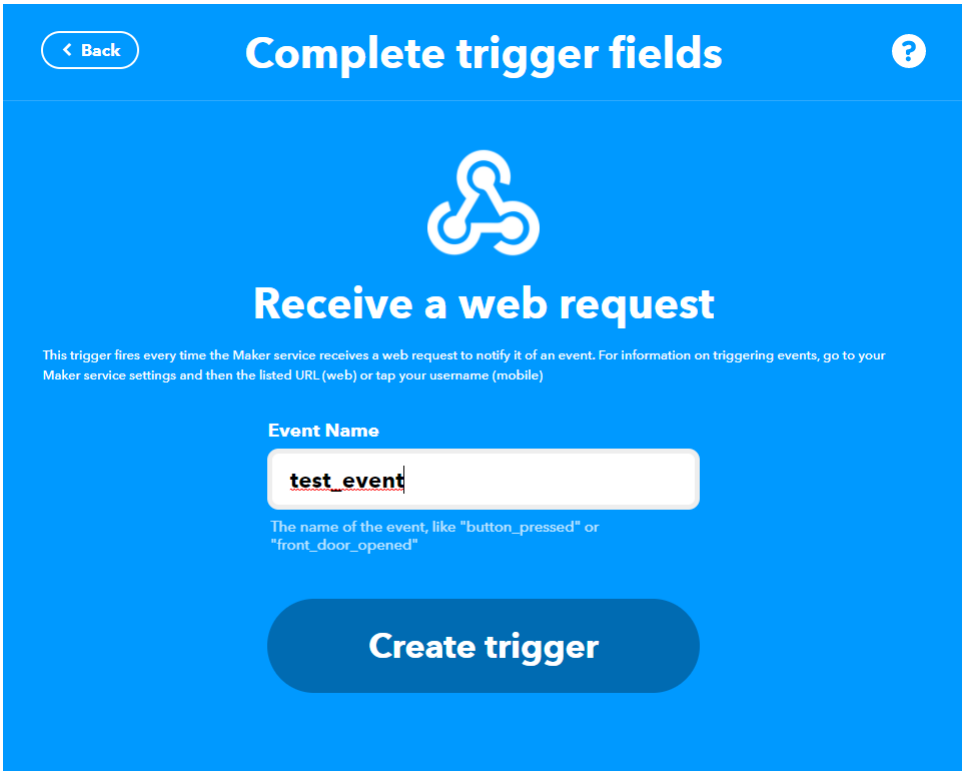
Choose Webhooks service as shown below. Alternatively, search the service by typing into the search bar.



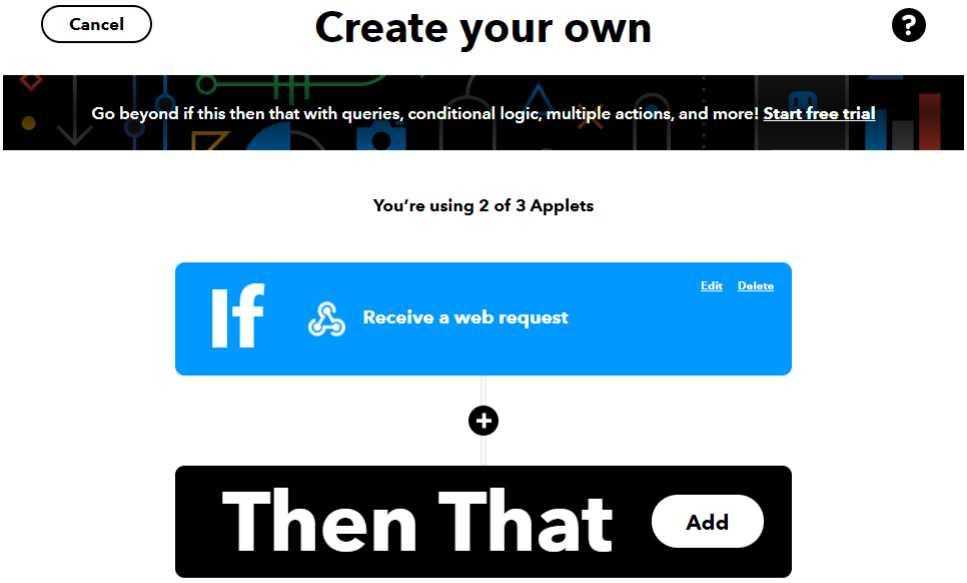
After that, the available triggers will appear. Choose Receive a Web request.



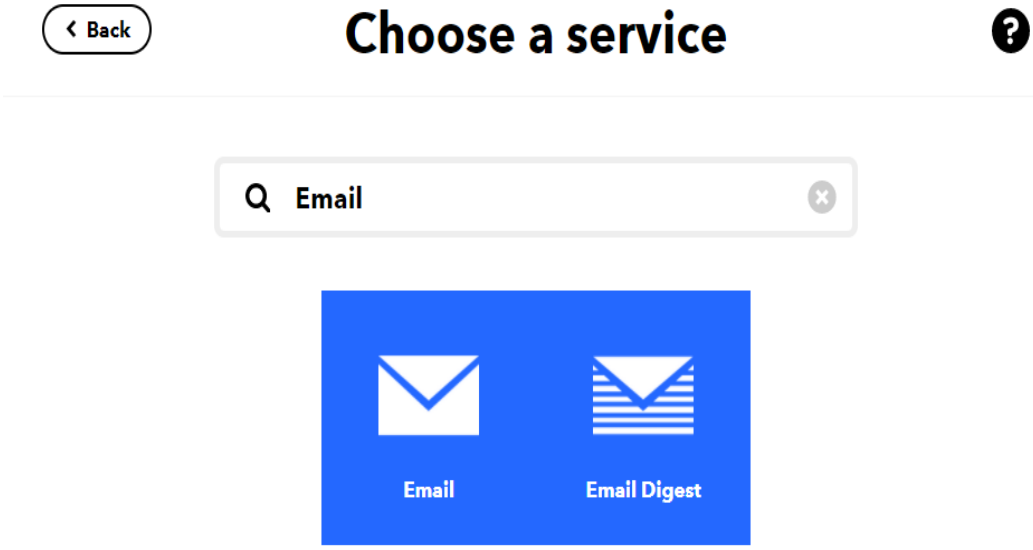
Next, an Event Name is required to identify the trigger successfully. In this example, set the Event name as “test_event”.



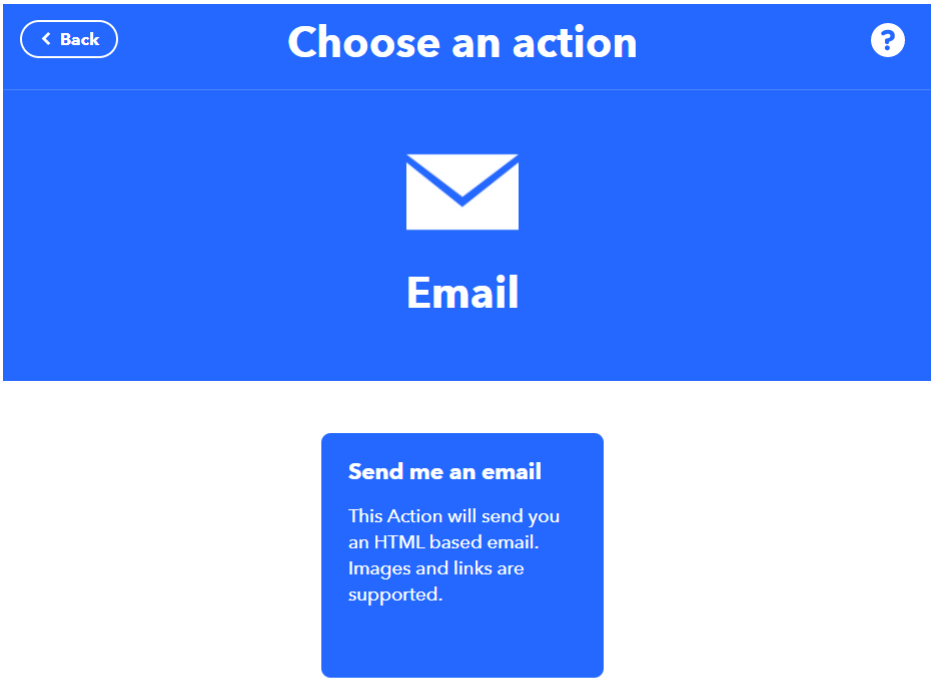
Next, click **Add** in Then That field to create the action service taken in response to the last trigger.



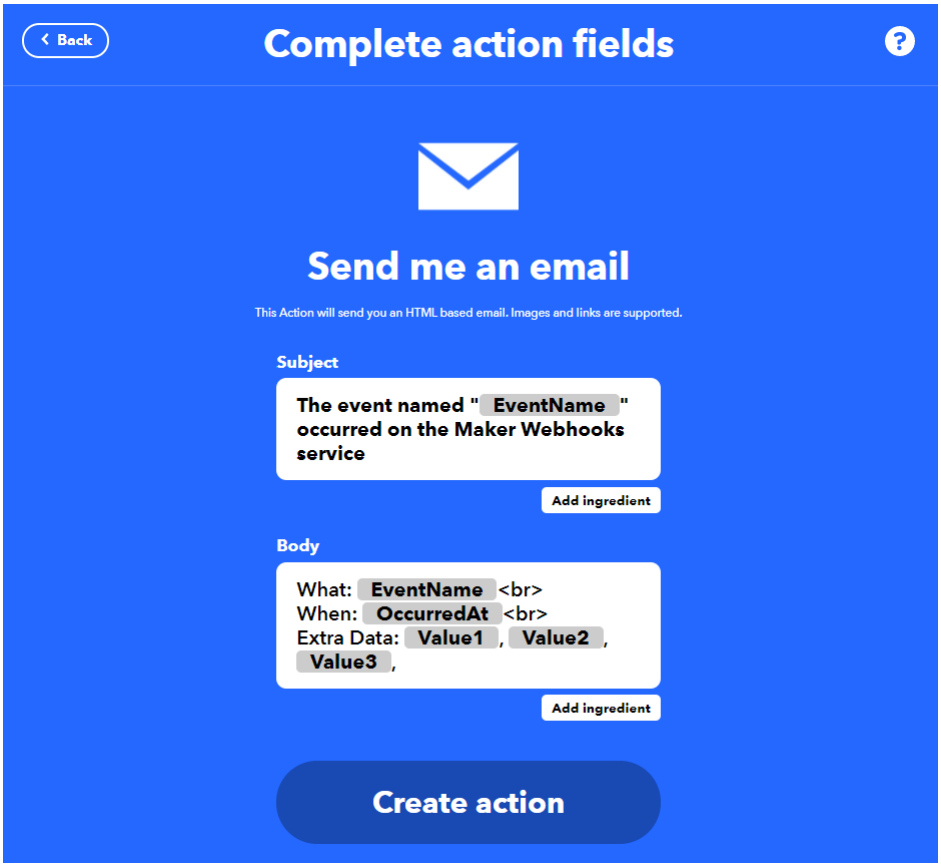
Choose Email as the action service.



Click on Send me an email.



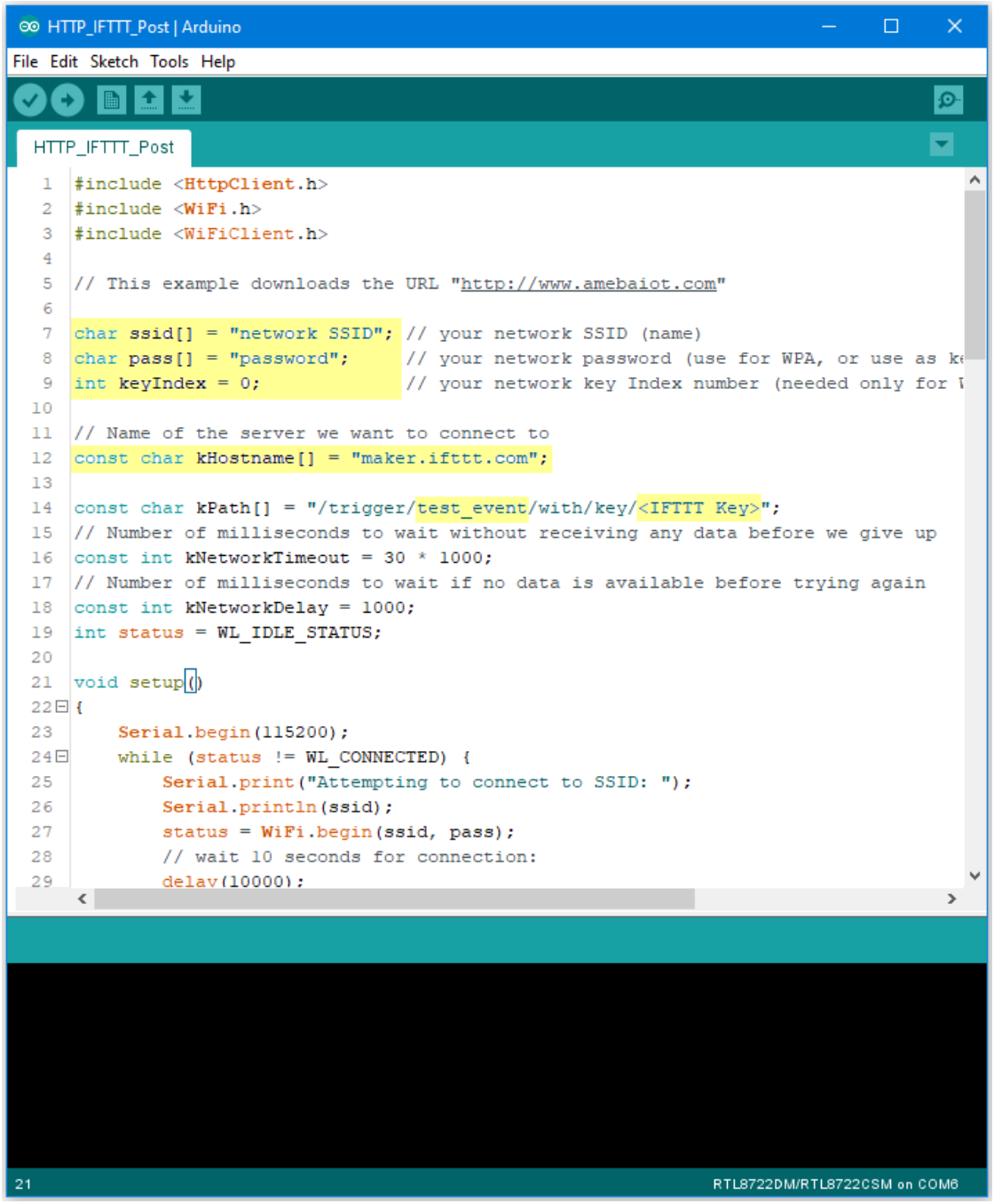
Under the template of **Send me an Email**, the contents of the email, such as subject and body is editable. Click **Create Action** to complete the action. Take note that **Email service** is offered to the email address registered under IFTTT account.



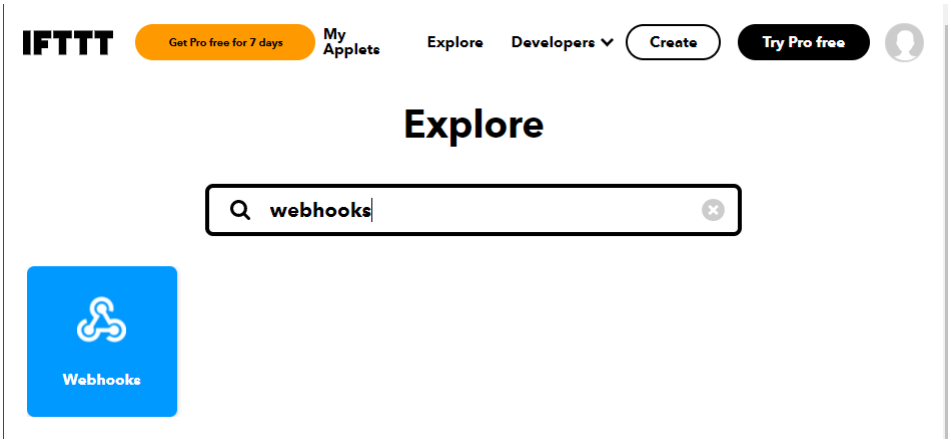
- Post the Trigger via Ameba

Once the Applet is ready in the IFTTT dashboard, the example program can be flashed onto the Ameba board to post the HTTP request.
Open the example code in "File" -> "Examples" -> "AmebaWiFi" -> "HTTP_IFTTT_Post"
In the example program, edit the following 3 items inside the code to make the program work.

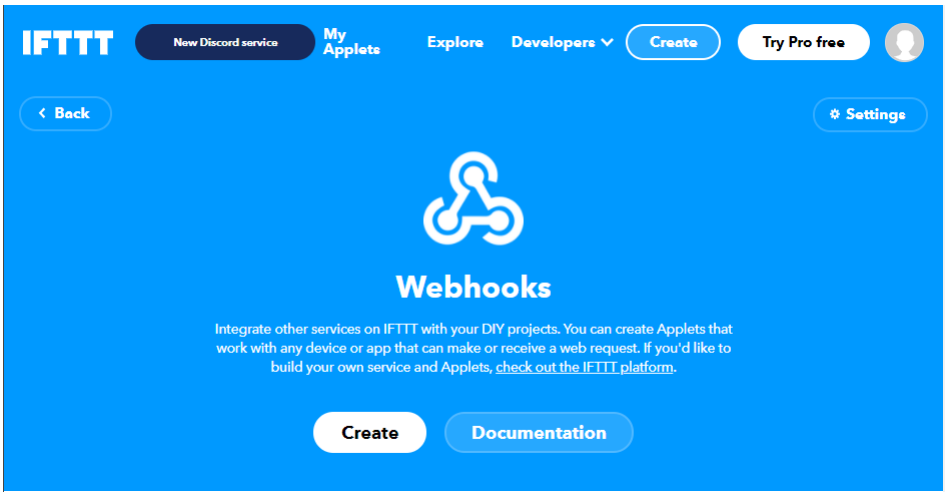
1. The WiFi credentials to connect to the Wi-Fi hotspot or access point of desirable choice.
2. Under the Host name field, enter the host name of the IFTTT service "maker.ifttt.com".
3. Under the Path name field, enter the Event name and key field "/trigger/Event name/with/key/Key Field"
   - Event name: The event name should be the same as the one specified in the IFTTT applet. In this example, the event name is "test_event".
   - Key Field: Available under webhook service in individual IFTTT account. See the next step for the steps to obtain the Key Field.



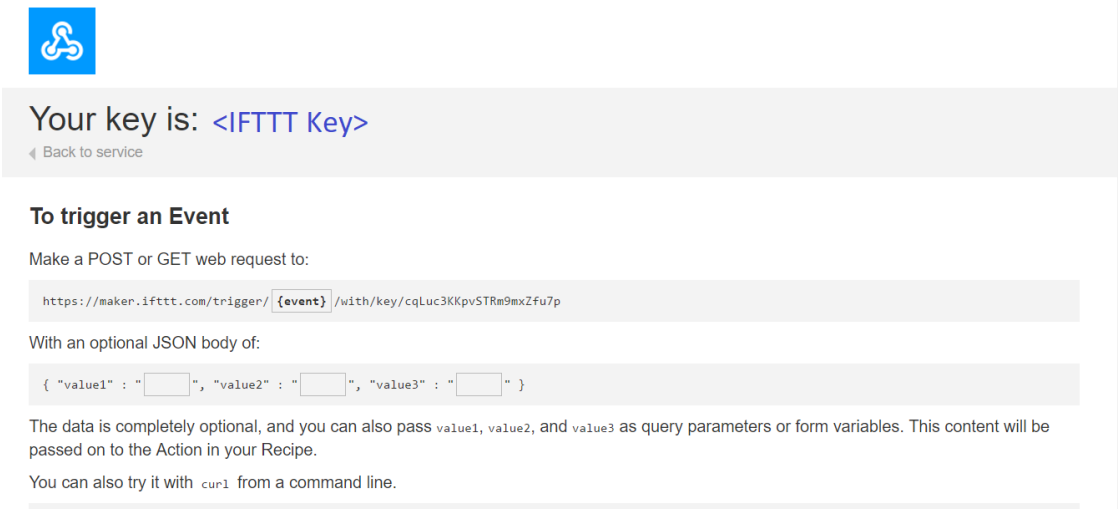
To obtain a key from documentation tab of the Webhooks, find the webhook service in the Explore tab.



On the Webhooks service page, click on the Documentation tab.

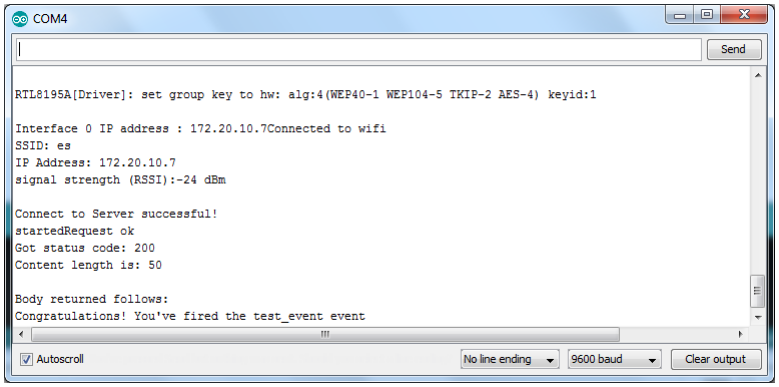


The key can be found in the documentation page. Also, information on how HTTP request can be used.

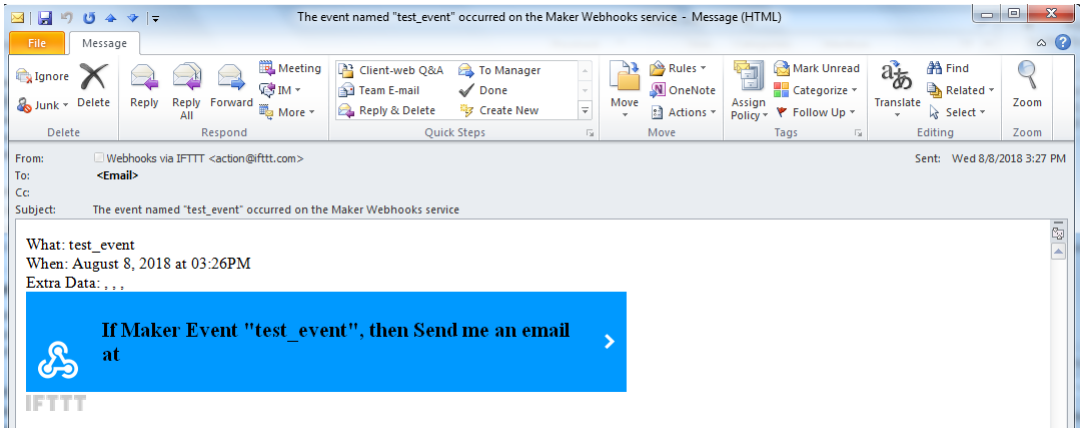


Once the example is ready, connect to Ameba board via USB Cable.

On the Arduino IDE, compile the code and upload the code onto Ameba and press the reset button. After the event has been successfully fired, “Congratulations! You have fired the test_event event” can be seen on the serial monitor and an email reminder for this event will be delivered.



Thereafter an email is sent to recipient email account registered at IFTTT Applet and email notification will be received.



IPv6 – Ameba as IPv6 Server/Client over TCP

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

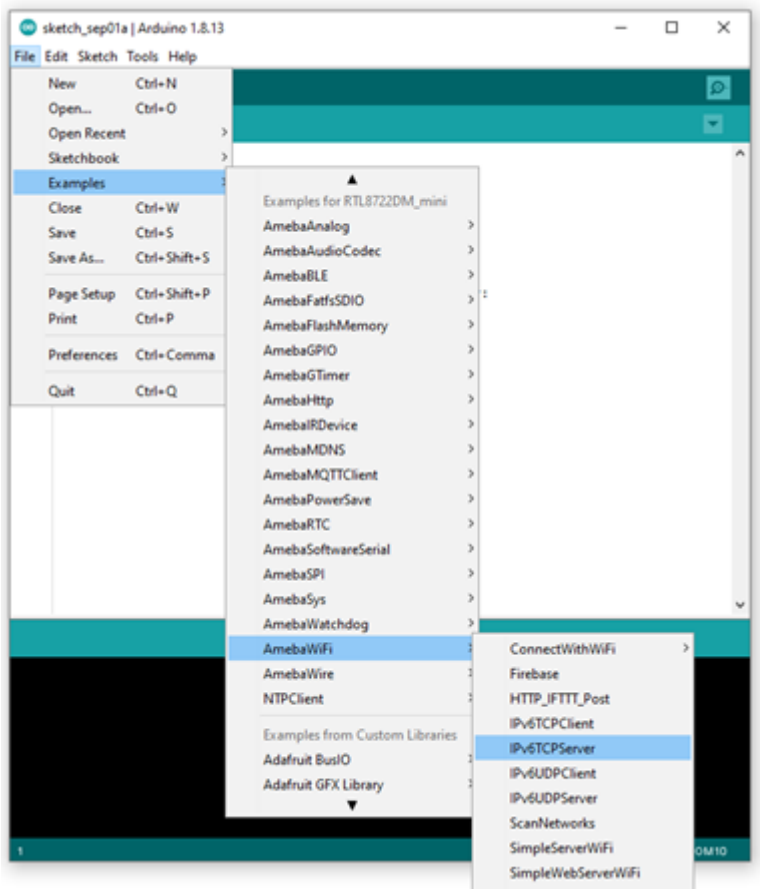
Example

Introduction

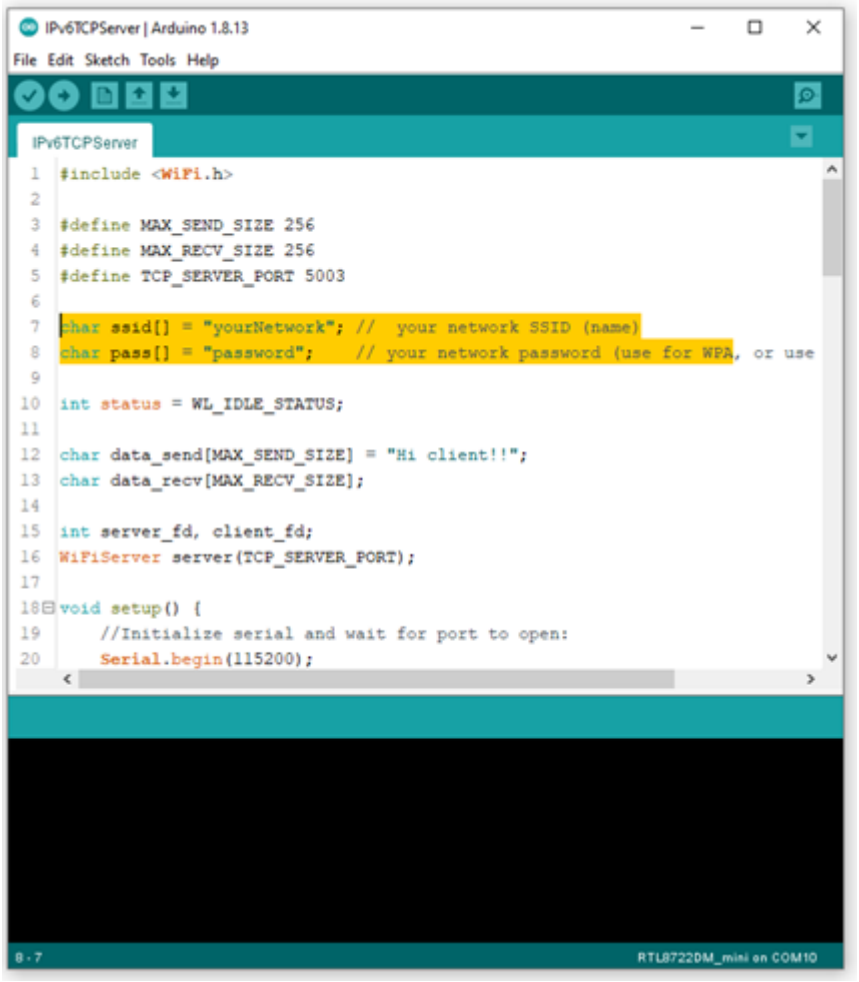
This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over TCP. Note that this example only works after you have set up the server and then configure the client accordingly.

Procedure

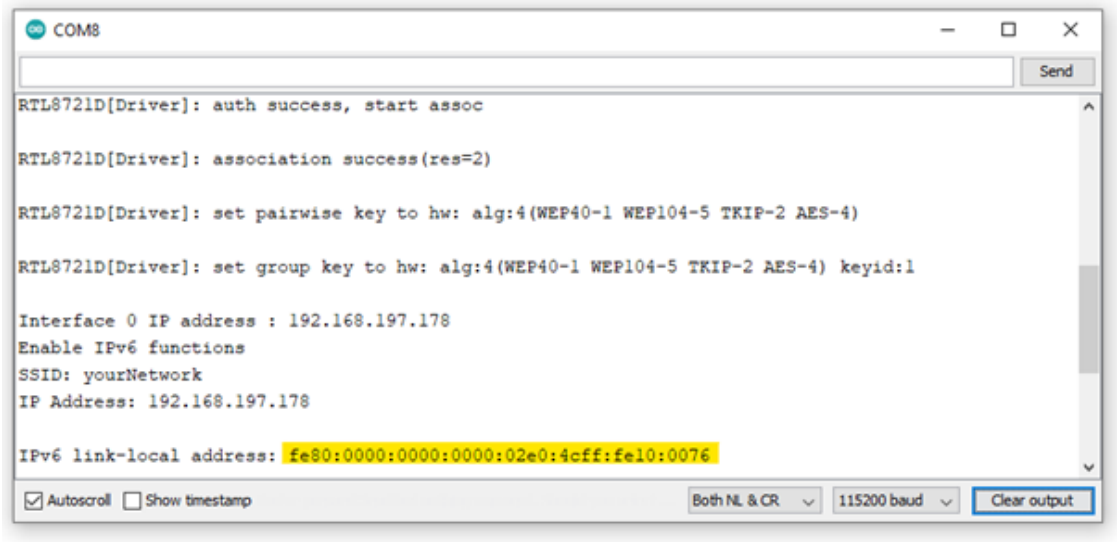
Step 1. IPv6TCPServer Open the example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6TCPServer`”.



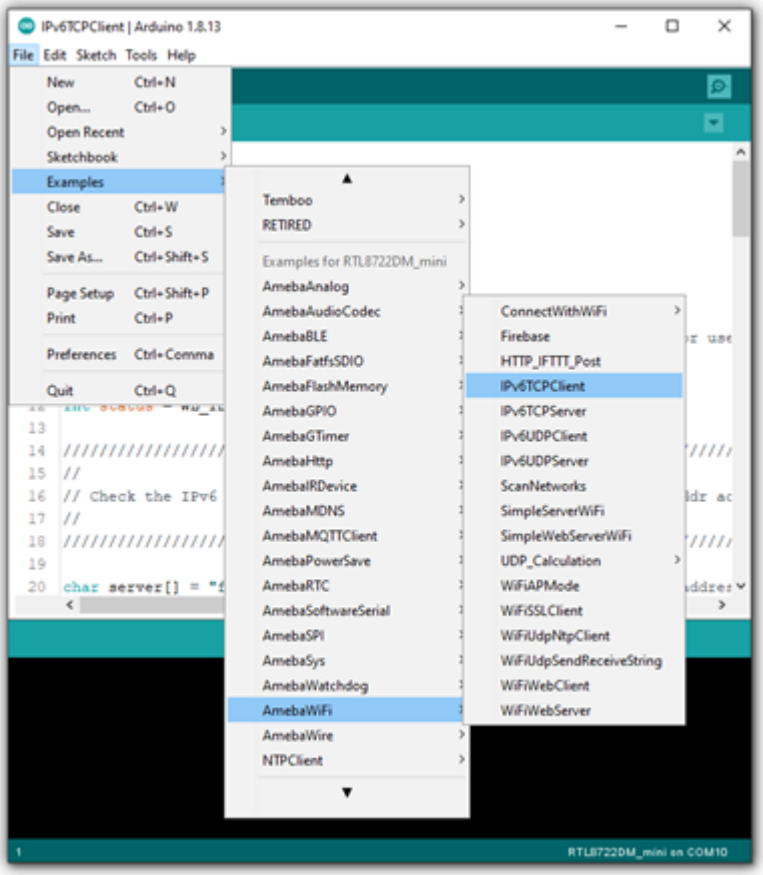
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.



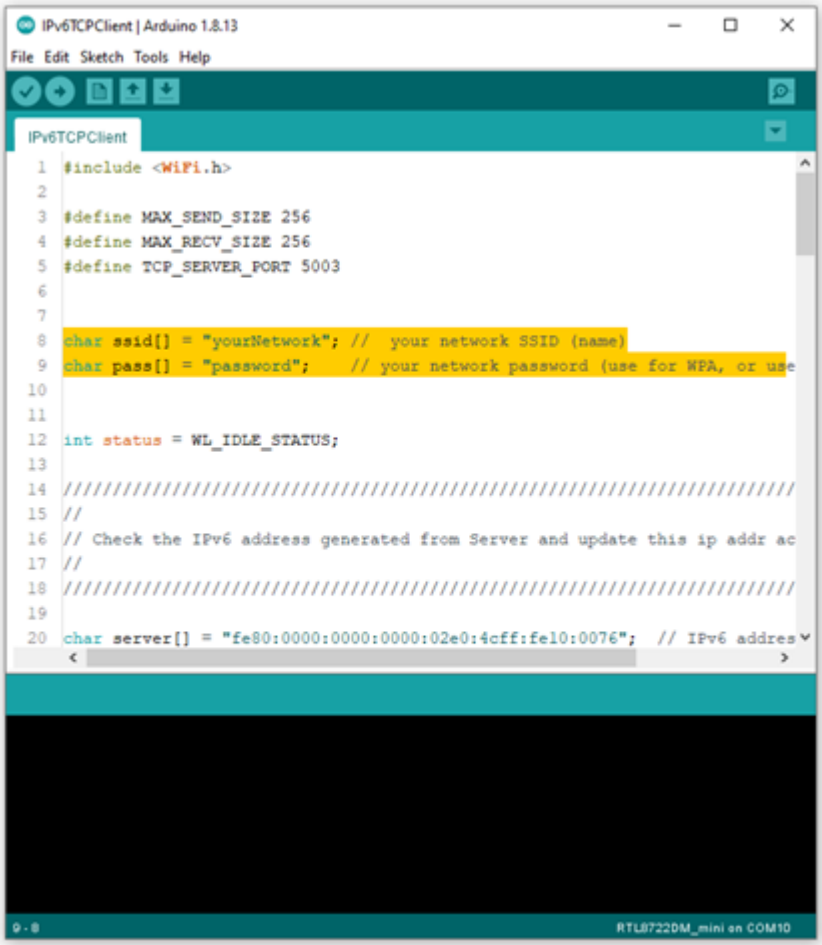
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



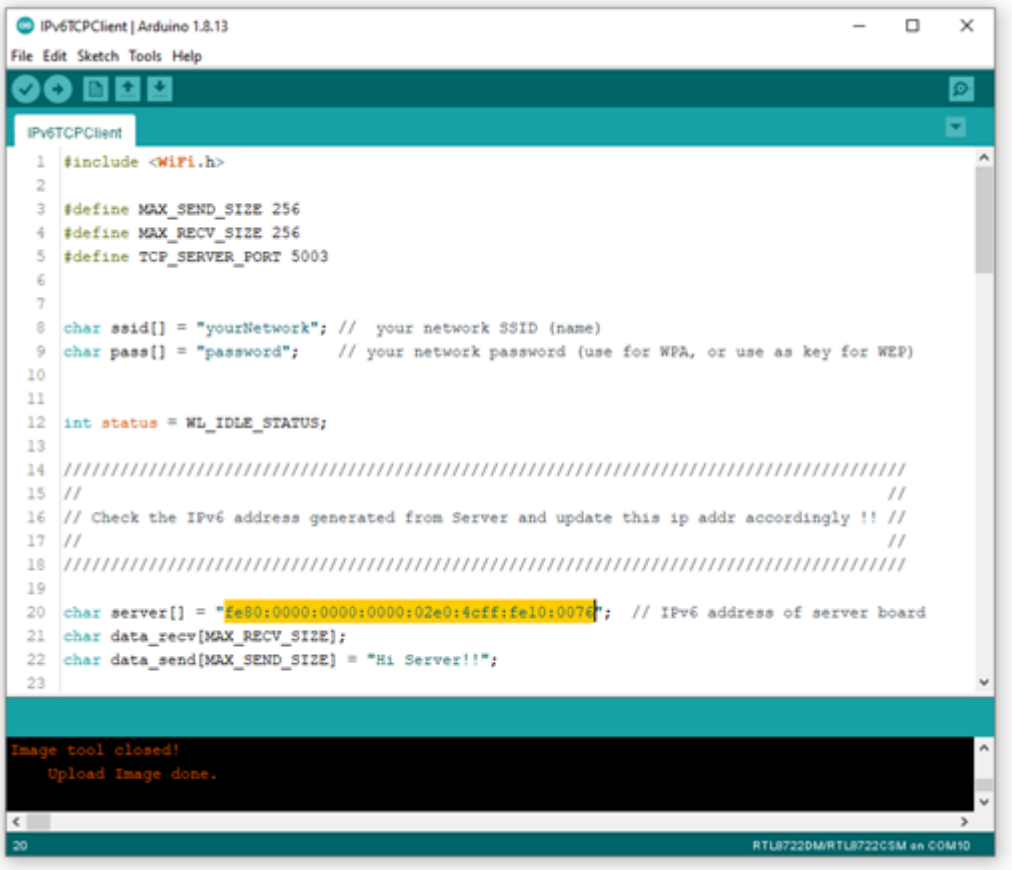
Step 2. IPv6TCPClient Now take the second Ameba D and open another example, "`Files` -> "`Examples` -> "`AmebaWiFi` -> "`IPv6TCPClient`".



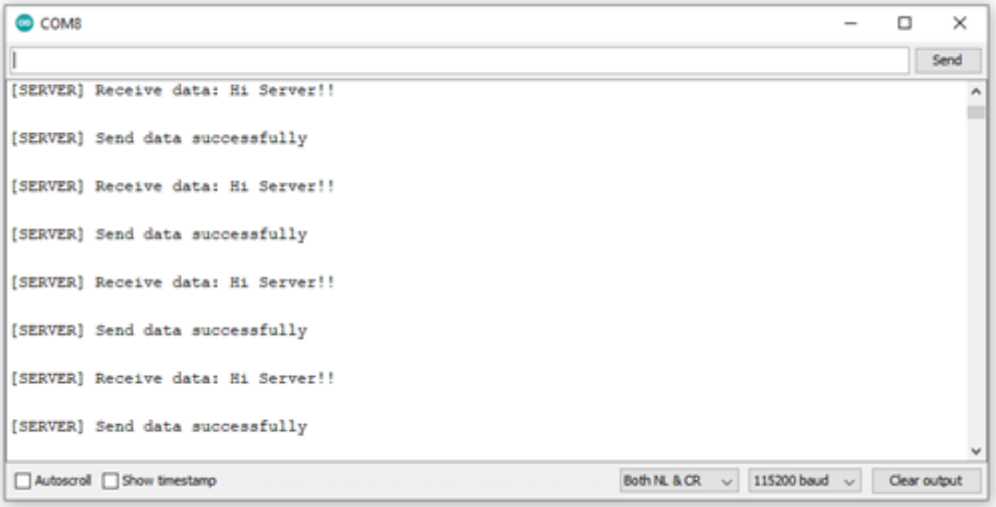
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

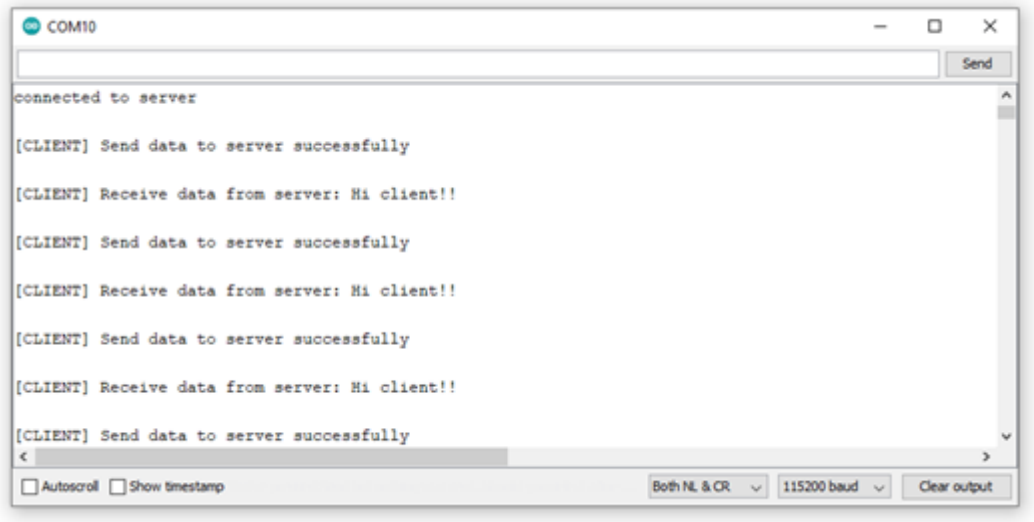


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6TCPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





IPv6 – Ameba as IPv6 Server/Client over UDP

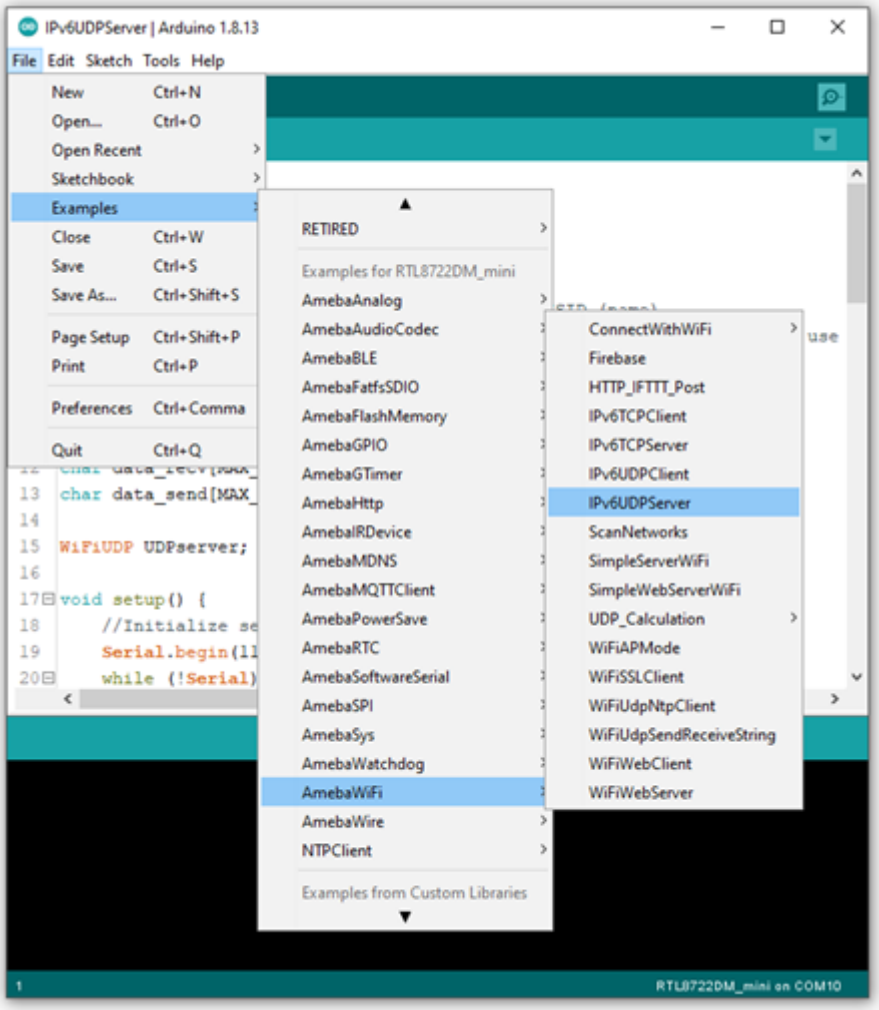
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

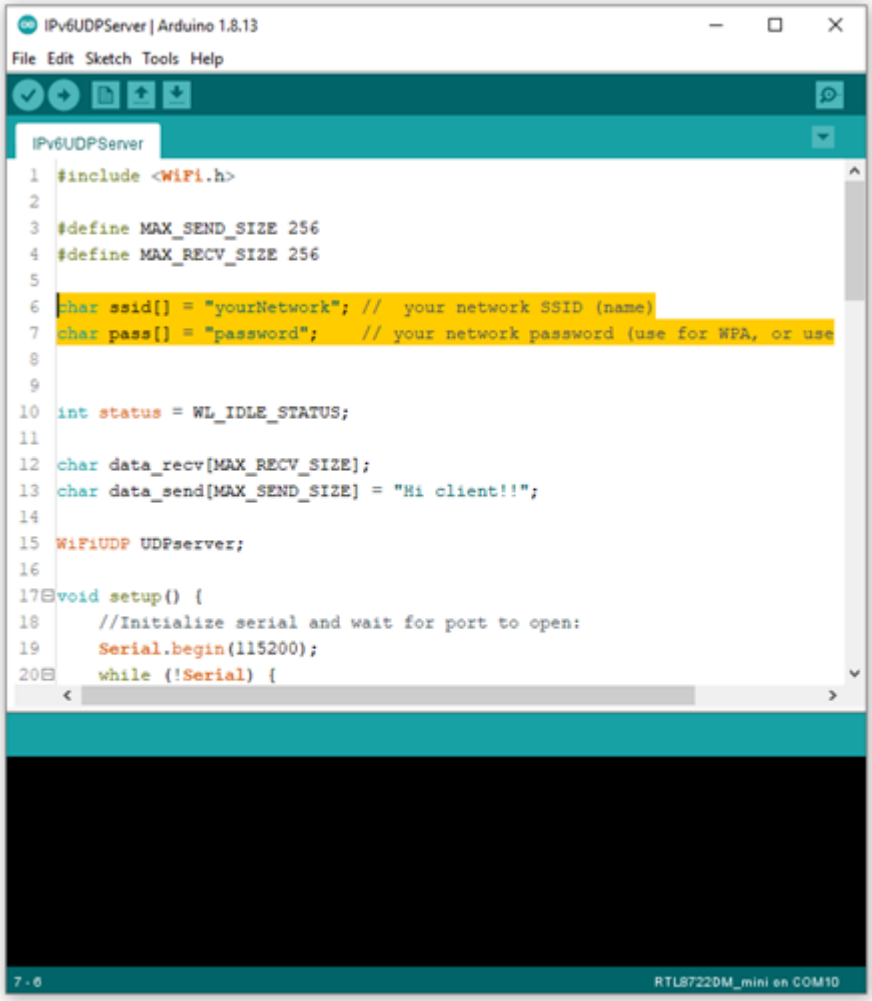
Example

Introduction

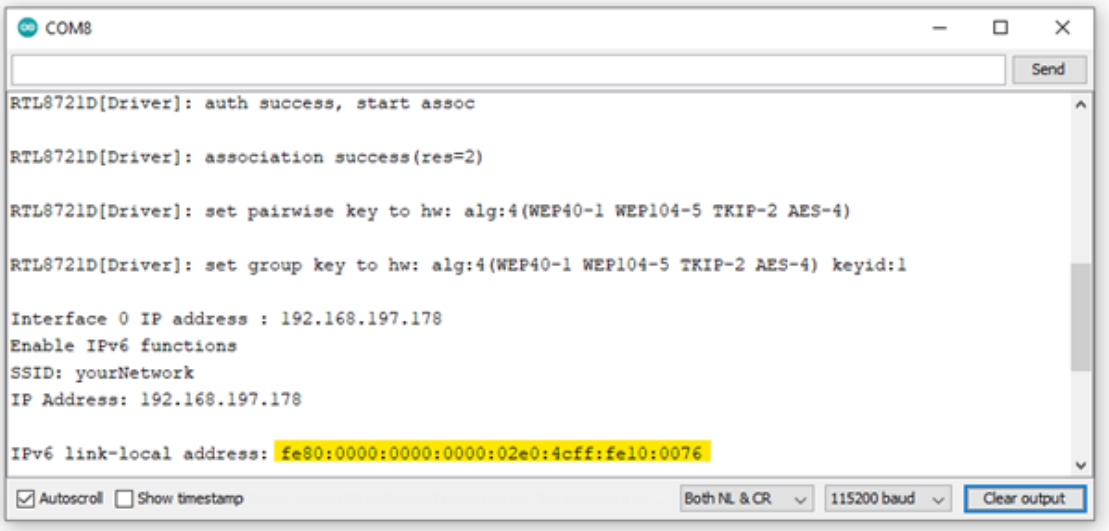
This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over UDP. Note that this example only works after you have set up the server and then configure the client accordingly.



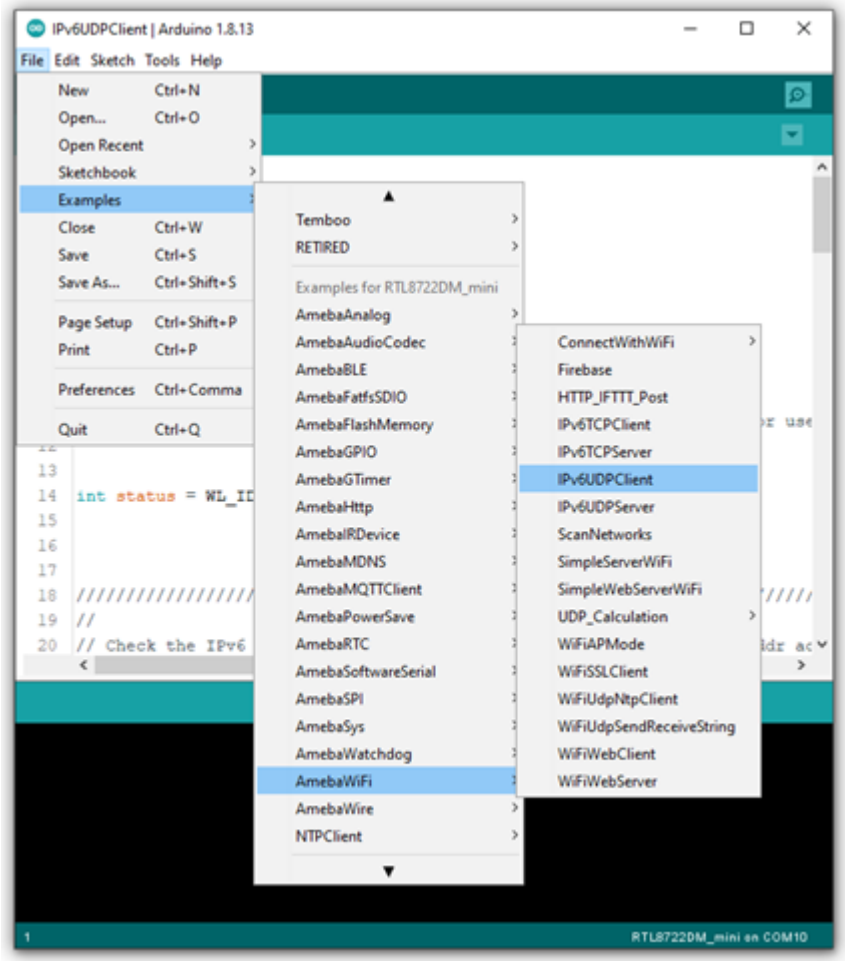
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.



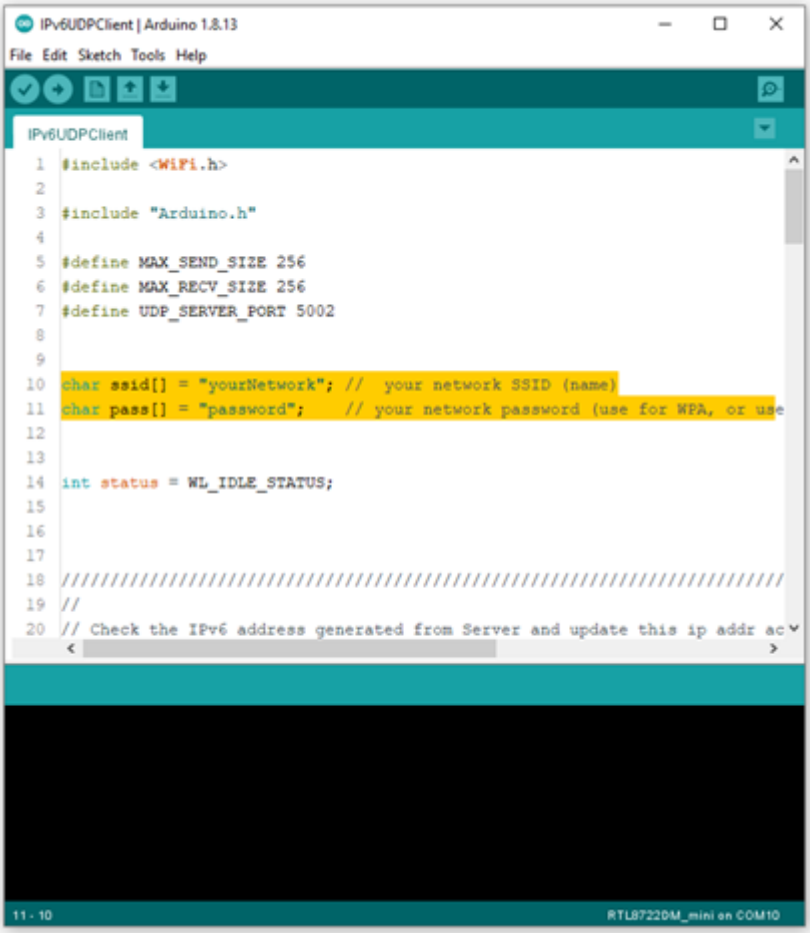
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



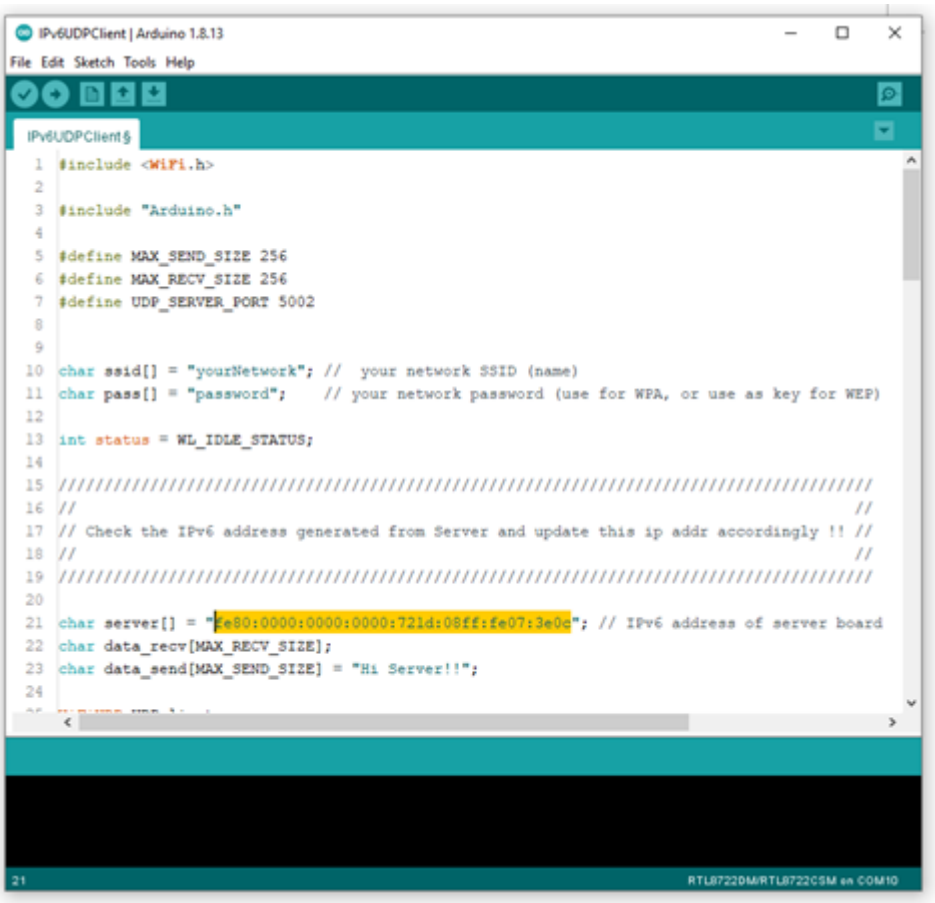
Step 2. IPv6UDPClient Now take the second Ameba D and open another example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6UDPClient`”.



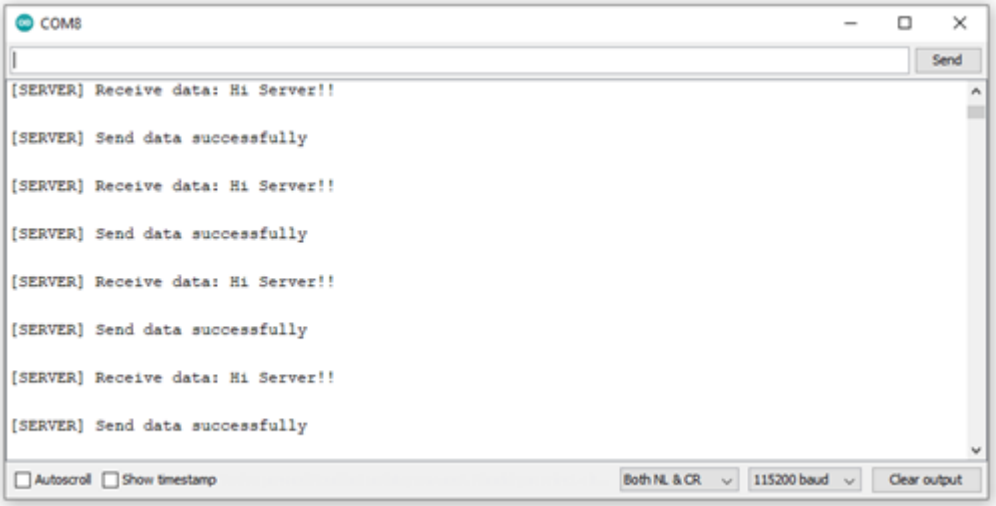
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

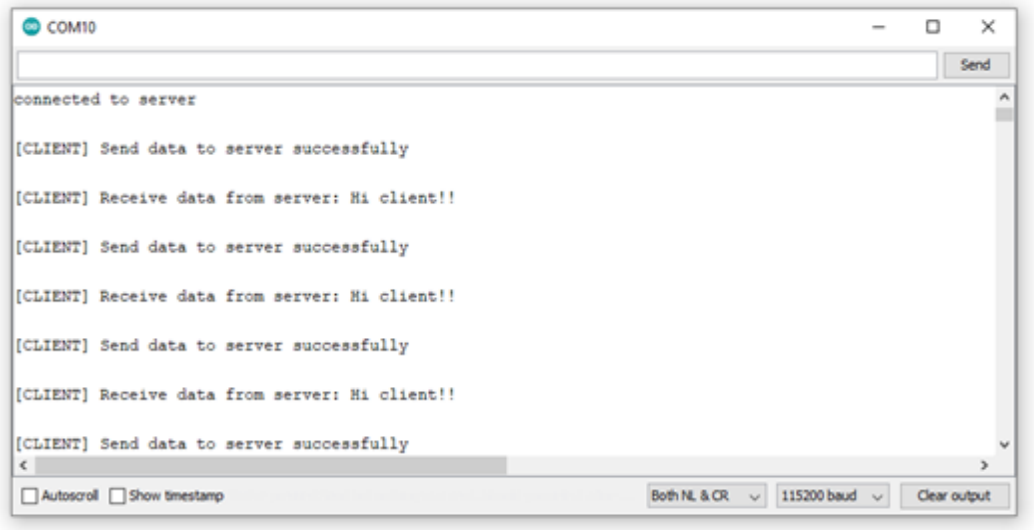


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6UDPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





MDNS - Set up mDNS Client on Arduino IDE

Preparation

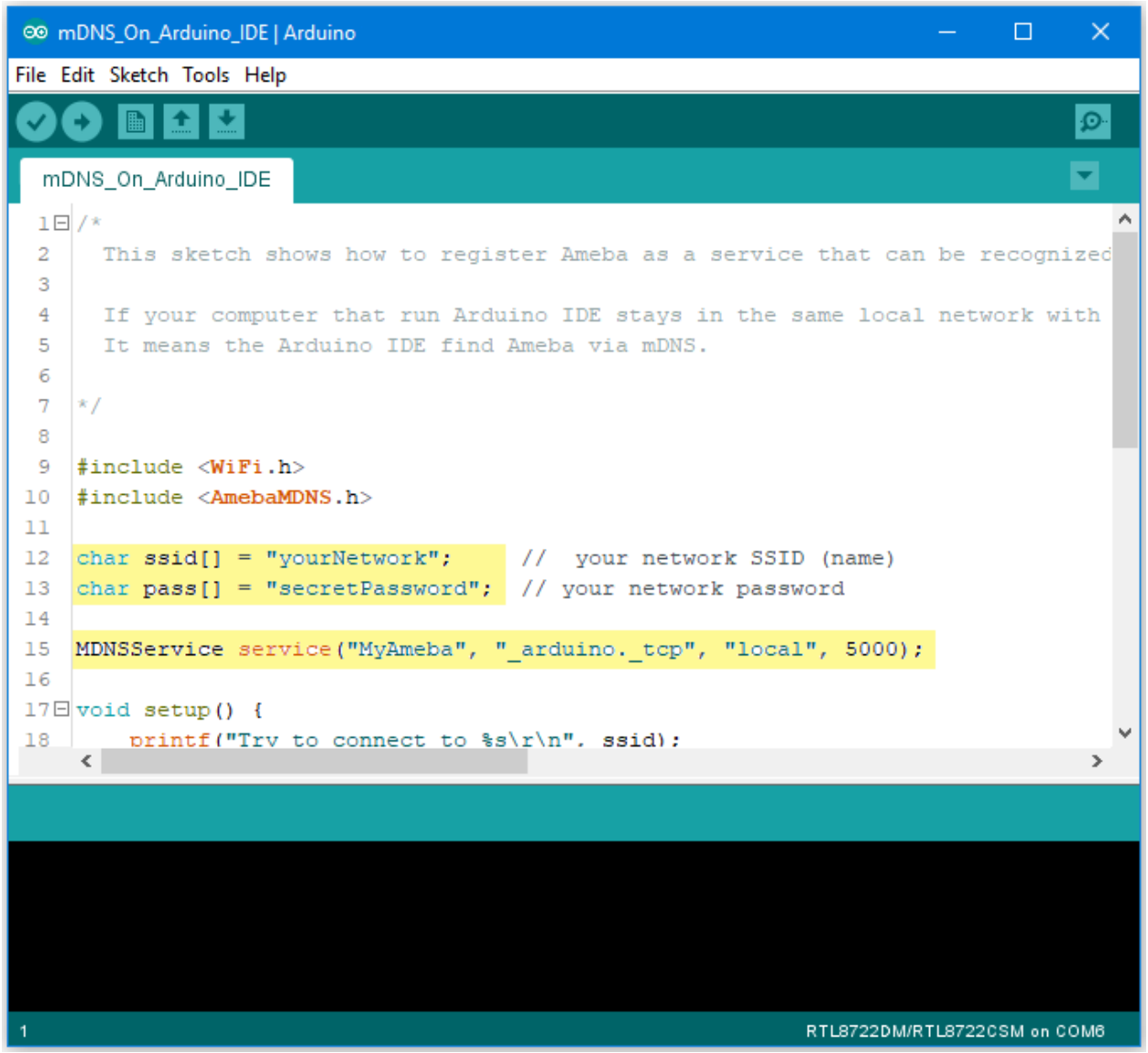
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

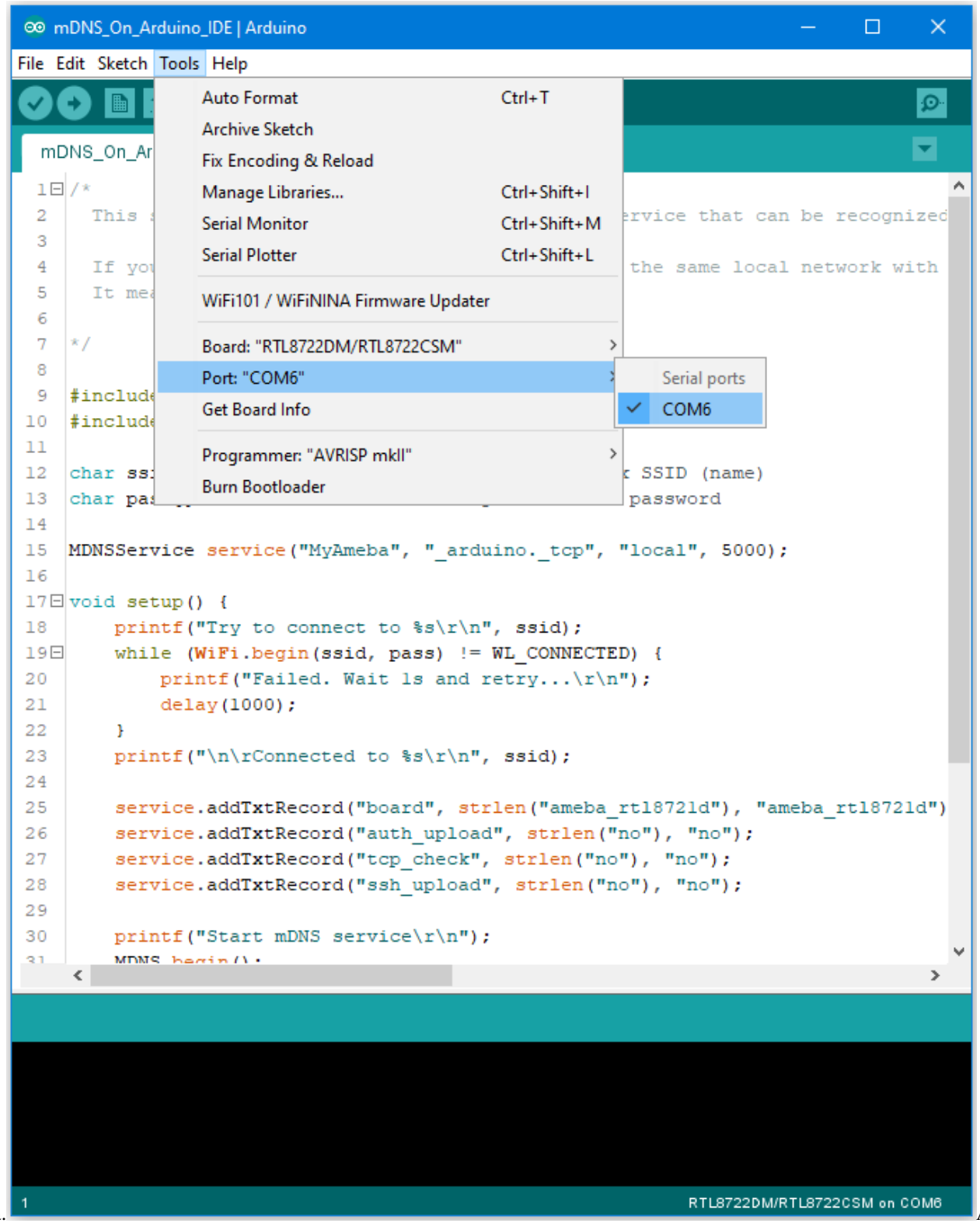
mDNS (Multicast DNS) is a protocol used in the local area network. It delivers the network information like IP address and provided services to others. mDNS is based on the UDP protocol, and it sends packets to 224.0.0.251 with port 5353 under IPv4 address. The naming style for the service follows the format: **{Instance Name}.{Protocol Name}.{Domain}**

- Instance Name: used to identify the name of the service
- Protocol Name: Divided into two parts, the front end is in regard to the name of the service, and it adds baseline as a prefix. The rear end is in regard to the transport protocol name it used, and it also adds baseline as a prefix
- Domain: Local area network in normal cases

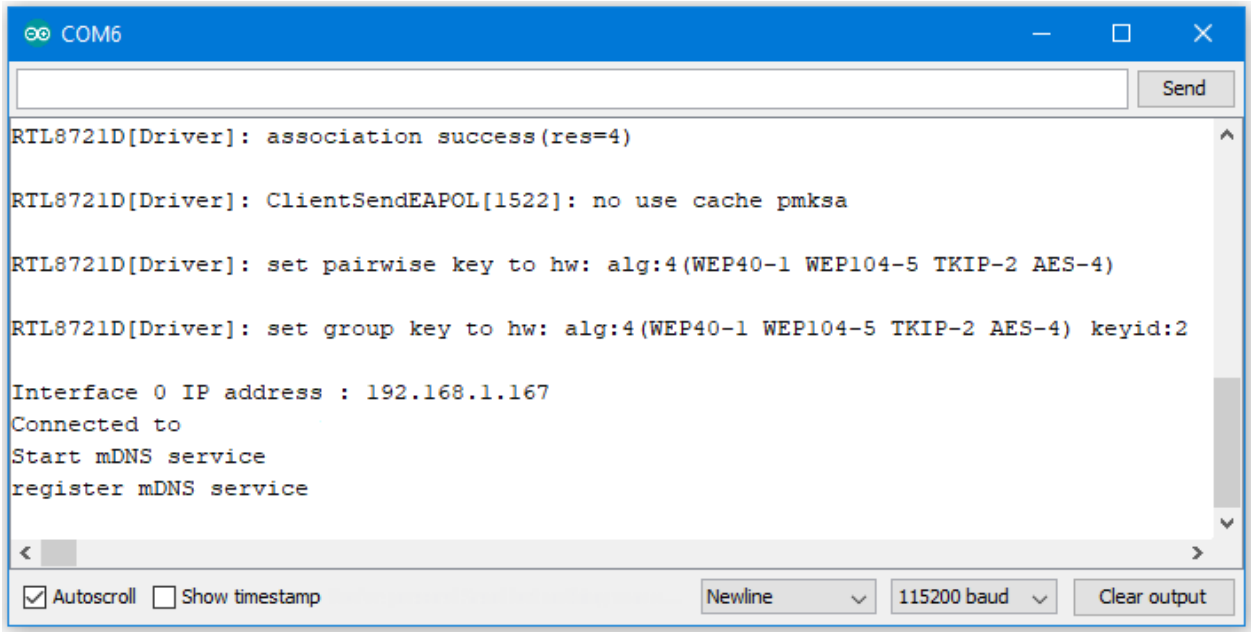
For example, Arduino IDE adopts the naming for the mDNS service which is used in OTA as following: **MyAmeba._arduino._tcp.local**
Among the naming example, “MyAmeba” can identify the Ameba device name and the name “MyAmeba” is changeable. “_arduino._tcp” is the protocol that Arduino IDE adopts, and the Domain is set as local in common.
Open the example, “`File`” -> “`Examples`” -> “`AmebaMDNS`” -> “`mdns_on_arduino_ide`”
You need to input ssid and password of the AP because the example will use WiFi connection.
And you can find out the naming of the service at the place where it declares MDNS Service. The example uses the default name “MyAmeba” and the name is changeable.



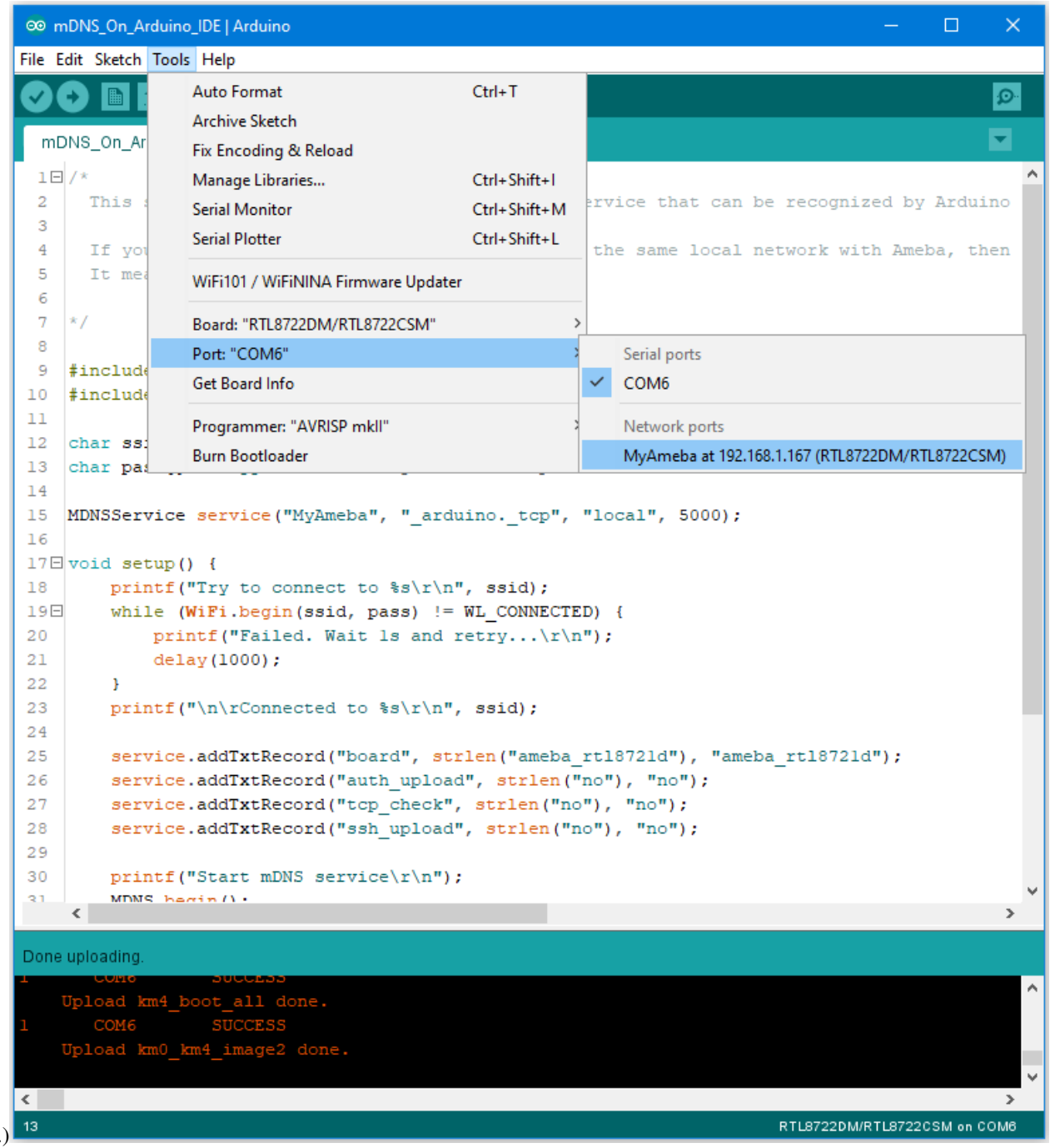
Next, go to ("Tools" -> "Port"), and you can find out at least one Serial Port. This port is simulated by Ameba board via USB. Choose this port and upload the compiled code to



Ameba. After uploading the code, press the reset button on Ameba and waiting for Ameba to connect with AP and activate the mDNS service after a while. You can see the Log at the bottom of the Serial Monitor.



Then you can find out the added item "Network Ports" **"MyAmeba at 192.168.1.167 (Ameba RTL8722DM/RTL8722CSM)"**, "MyAmeba" is the device name we set up, and "IP" is the IP address that AP assigned to Ameba, the IP address should be the same with the IP shown in the Serial Monitor. Last, "Ameba RTL8722DM/RTL8722CSM" is the type name of the board, and it means that Ameba can let Arduino IDE identify the mDNS service successfully.(We still can not use the Internet to upload the code, and we will explain this part in the OTA



example.)

If you cannot find the Network ports on your Arduino IDE, please check

- Does your computer in the same local area network with the Ameba?
- Restart the Arduino IDE, and it will find the mDNS service again
- Check the Log in Serial Monitor if the Ameba connects to the AP and activate mDNS service successfully

Code Reference

The program set up the mDNS service in the beginning, the first parameter is Instance Name, and it is changeable in this example. The second parameter is the protocol that the service used, and it would be "_arduino._tcp" for Arduino IDE. The third parameter is Domain, and it would be "local" in common. The fourth parameter is the port number for the service, it is 5000 here and we doesn't use it in the example.

```
MDNSService service("MyAmeba", "_arduino._tcp", "local", 5000);
```

After connected to the network, we set up some text fields for the service. For the following example, "board" is the name of the field, "ameba_rtl8721d" is the value of the field. "board" is used to let Arduino IDE check installed SDK to see if it exists known device or not. We will use the name of the device if there is known device, users can change "ameba_rtl8721d" to "yun" or other names to find out what's the difference if interested.

```
service.addTxtRecord("board", strlen("ameba_rtl8721d"),"ameba_rtl8721d");
```

Then we add three text fields "auth_upload", "tcp_check", and "ssh_upload", this example does not activate these services.

```
service.addTxtRecord("auth_upload", strlen("no"), "no");
service.addTxtRecord("tcp_check", strlen("no"), "no");
service.addTxtRecord("ssh_upload", strlen("no"), "no");
```

Next we activate MDNS

```
MDNS.begin();
```

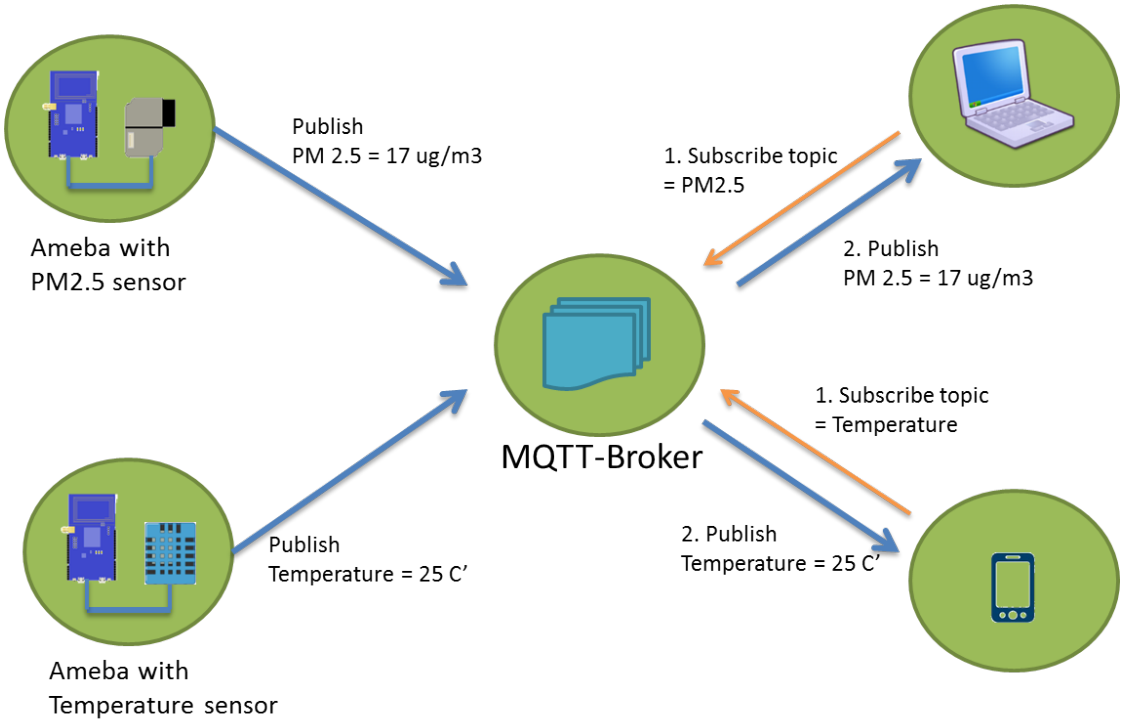
and register to the mDNS service.

```
MDNS.registerService(service);
```

MQTT - Set up MQTT Client to Communicate with Broker

Intro to MQTT

MQTT (Message Queuing Telemetry Transport) is a protocol proposed by IBM and Eurotech. The introduction in MQTT Official Website: MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. We can say MQTT is a protocol designed for IoT. MQTT is based on TCP/IP and transmits/receives data via publish/subscribe. Please refer to the figure below:



In the operation of MQTT, there are several roles:

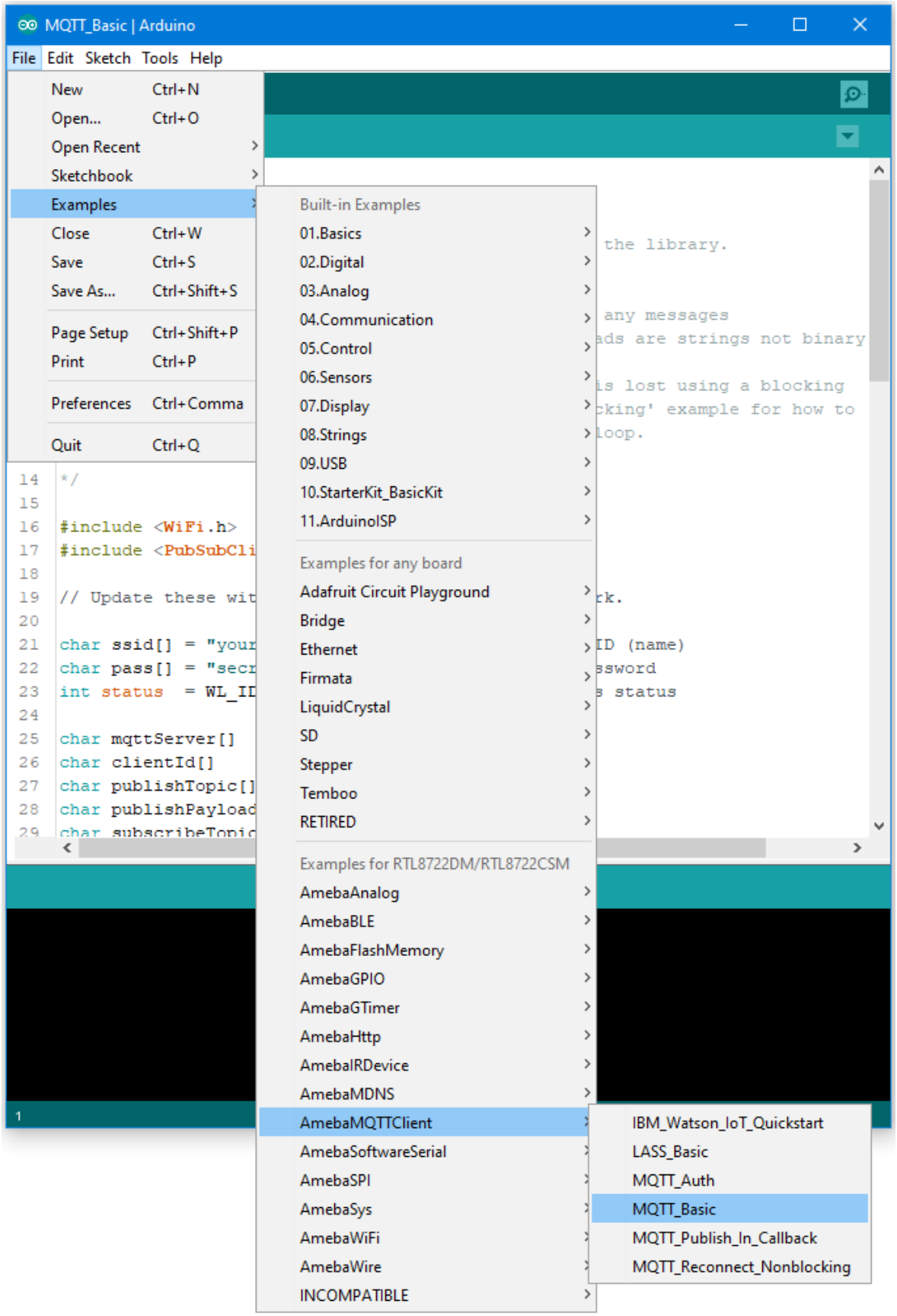
- **Publisher**: Usually publishers are the devices equipped with sensors (ex. Ameba). Publishers uploads the data of the sensors to MQTT-Broker, which serves as a database with MQTT service.
- **Subscriber**: Subscribers are referred to the devices which receive and observe messages, such as a laptop or a mobile phone.
- **Topic**: Topic is used to categorized the messages, for example the topic of a message can be "PM2.5" or "Temperature". Subscribers can choose messages of which topics they want to receive.

Preparation

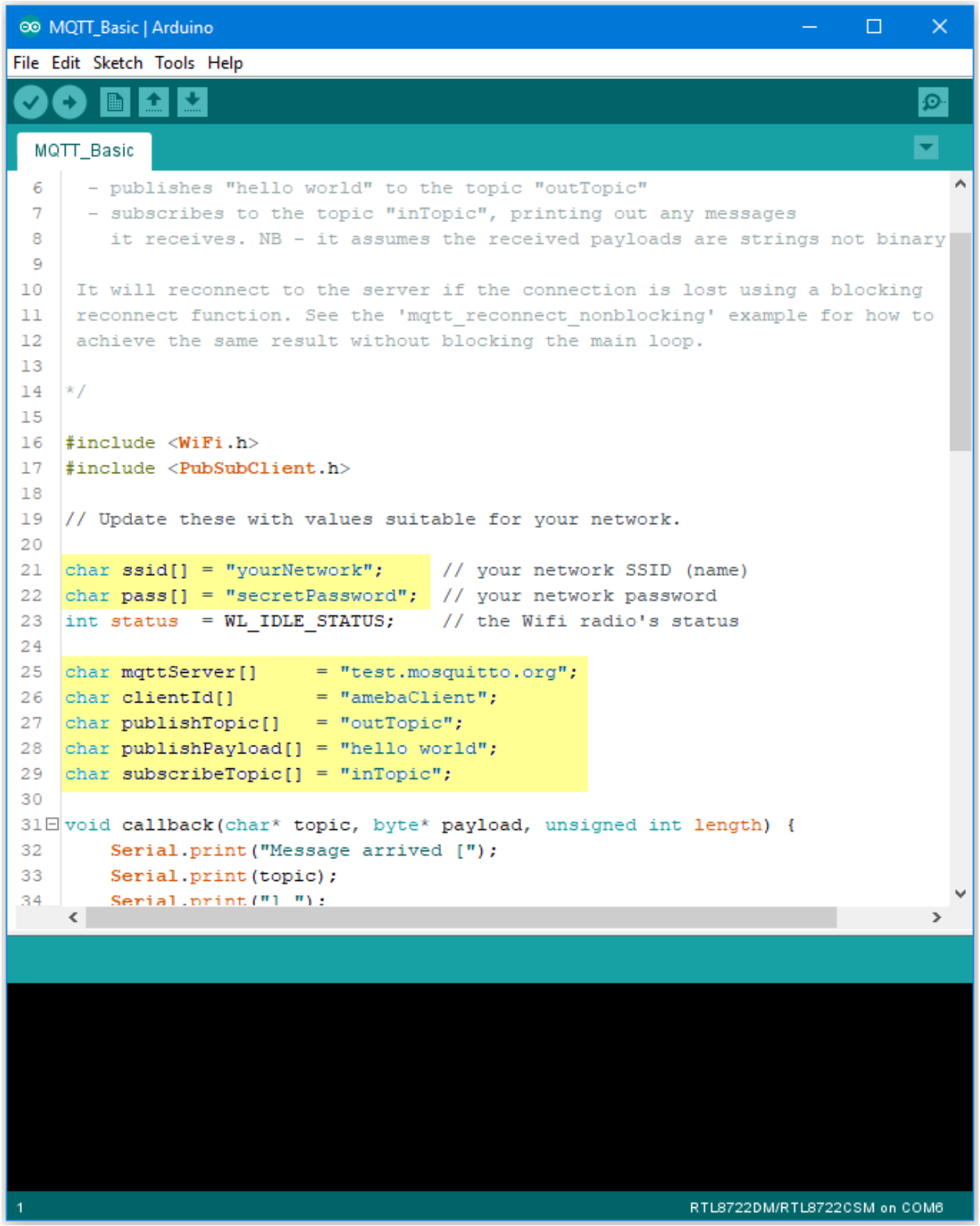
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to MQTT-Broker. Then send messages as publisher and receive messages from MQTT-Broker as subscriber.
Open the MQTT example "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`MQTT_Basic`"


MQTT_Basic | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking

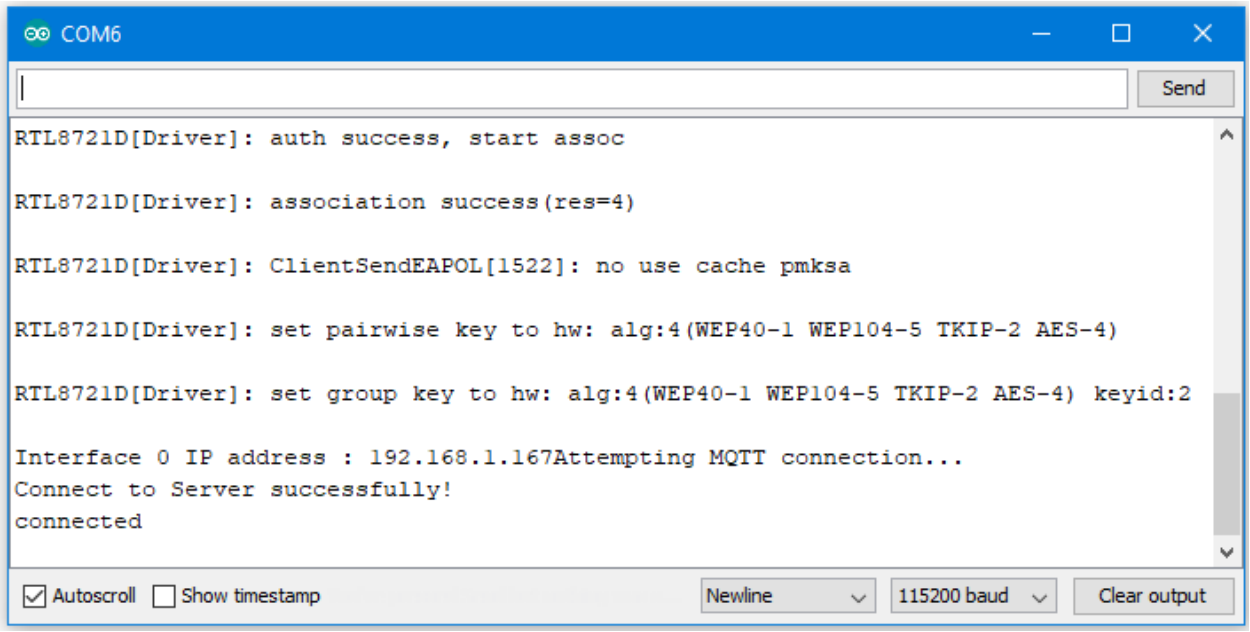
Please modify some WiFi-related parameters.
And some information related to MQTT:



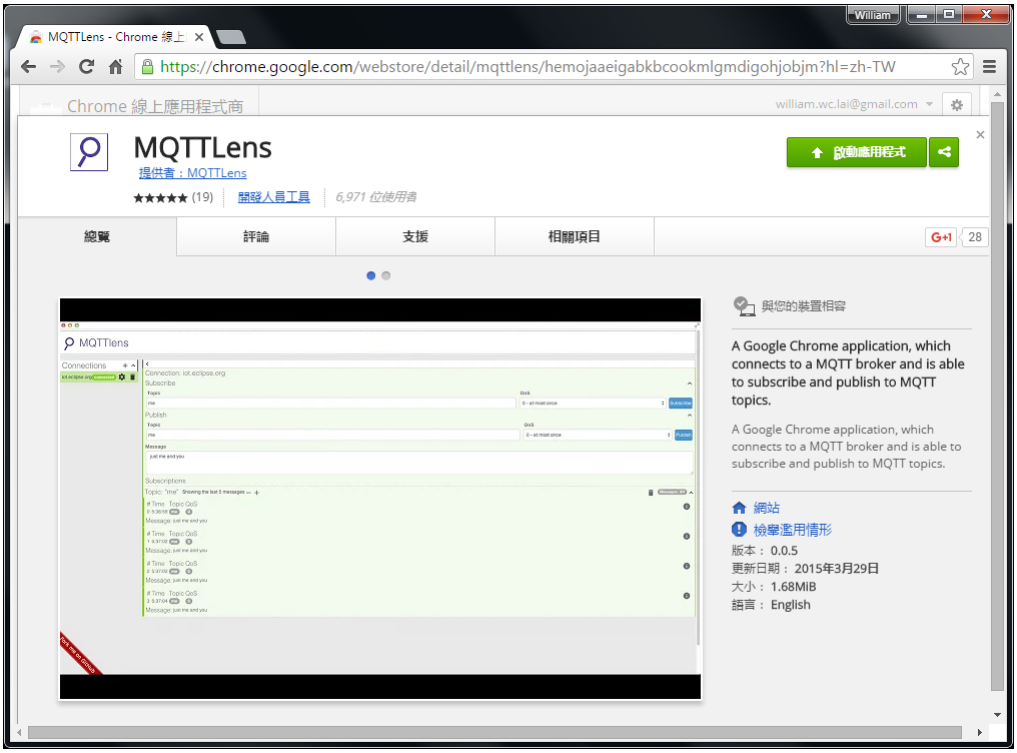
The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.

- “clientId” is an identifier for MQTT-Broker to identify the connected device.
- “publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
- “publishPayload” is the content to be published.
- “subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor



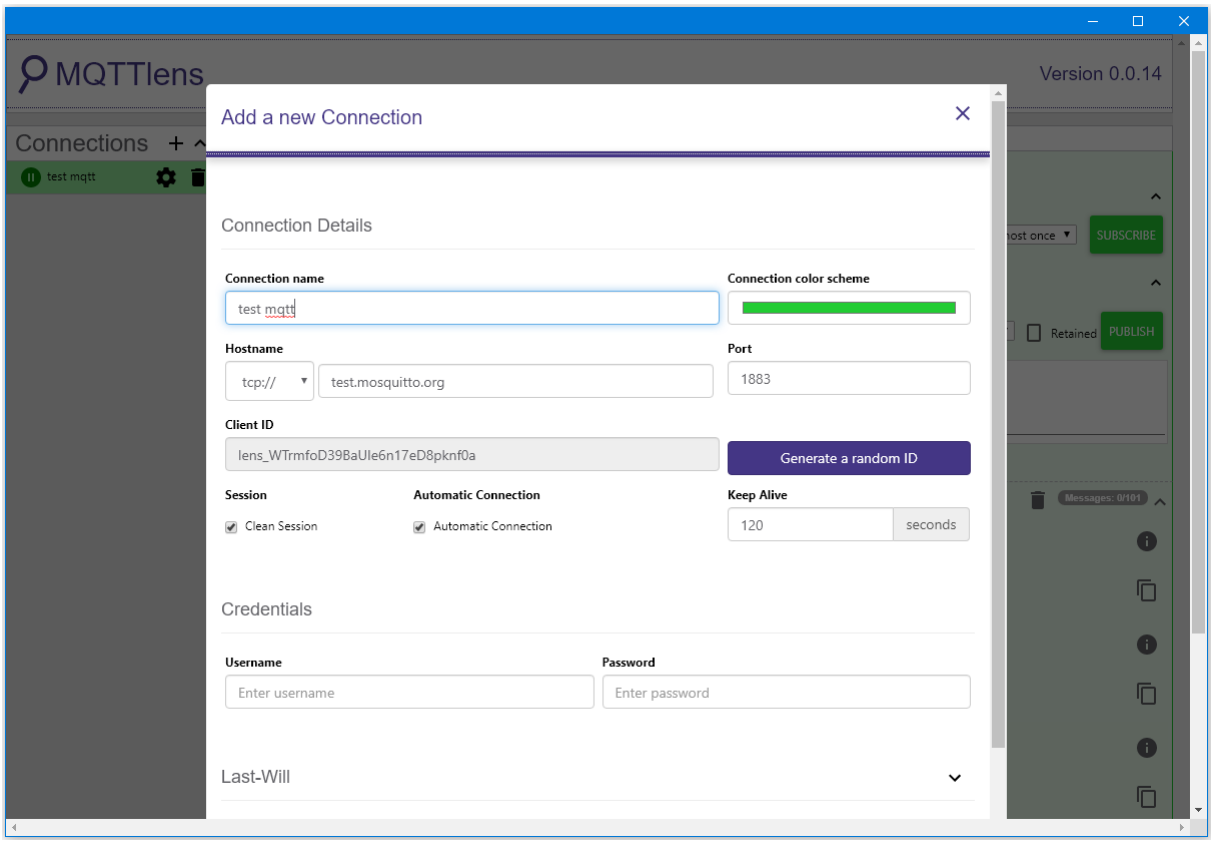
After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”.
To see the message, we need another MQTT client.
Here we use a chrome plugin “MQTTLens” to be the MQTT client. You can find it in google webstore.



Install and open the MQTTLens, click “+” next to “Connection” on the left, and fill in the required information

- **Connection Name**: Used to identify the connection, you can choose a name you like.
- **Hostname**: The MQTT-Broker server, here we use “iot.eclipse.org”
- **Client ID**: We use the default randomly generated ID.

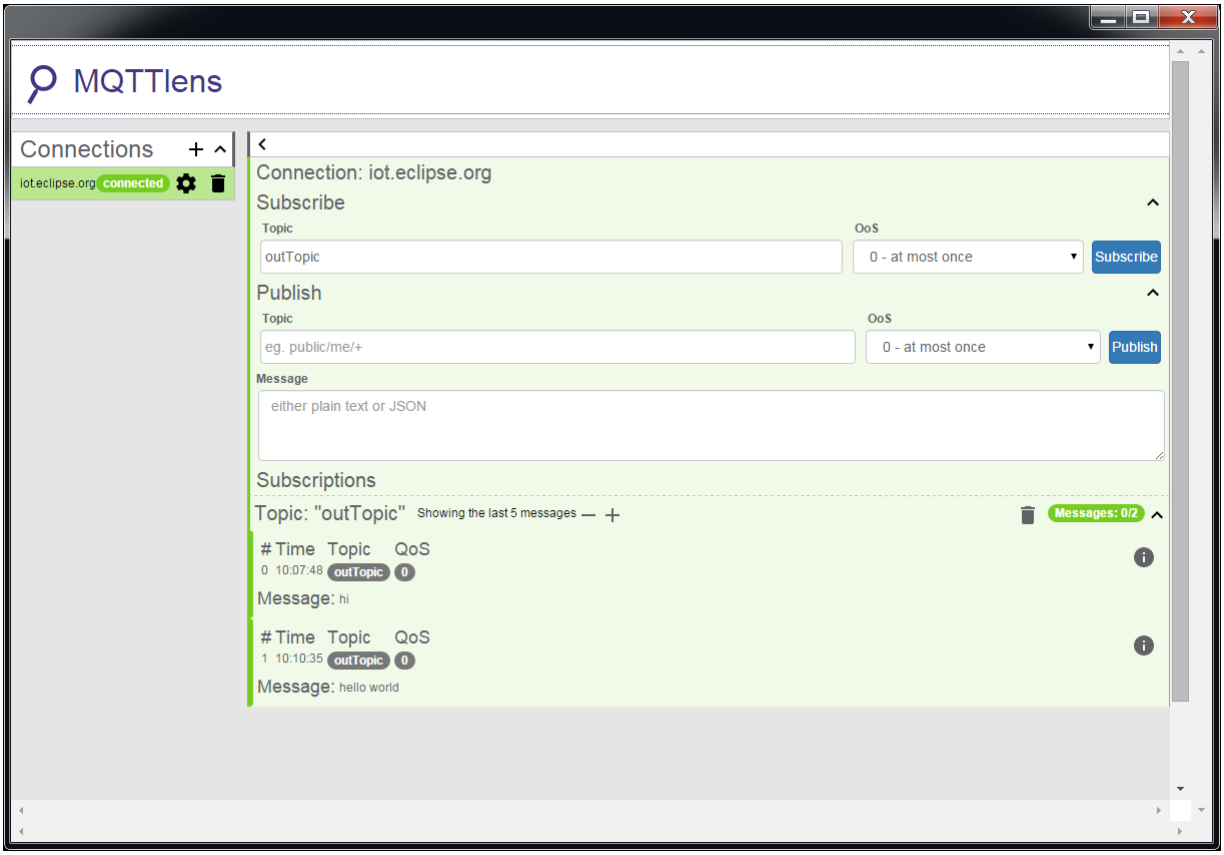
Then click “CREATE CONNECTION”.



Since we have not registered the topic we want to listen to, we would not receive any messages now.

Fill in “outTopic” in the “Topic” field and click “Subscribe”.

Wait for Ameba to send next message (or you can press the reset button). Then you can see the “hello world” message show up.



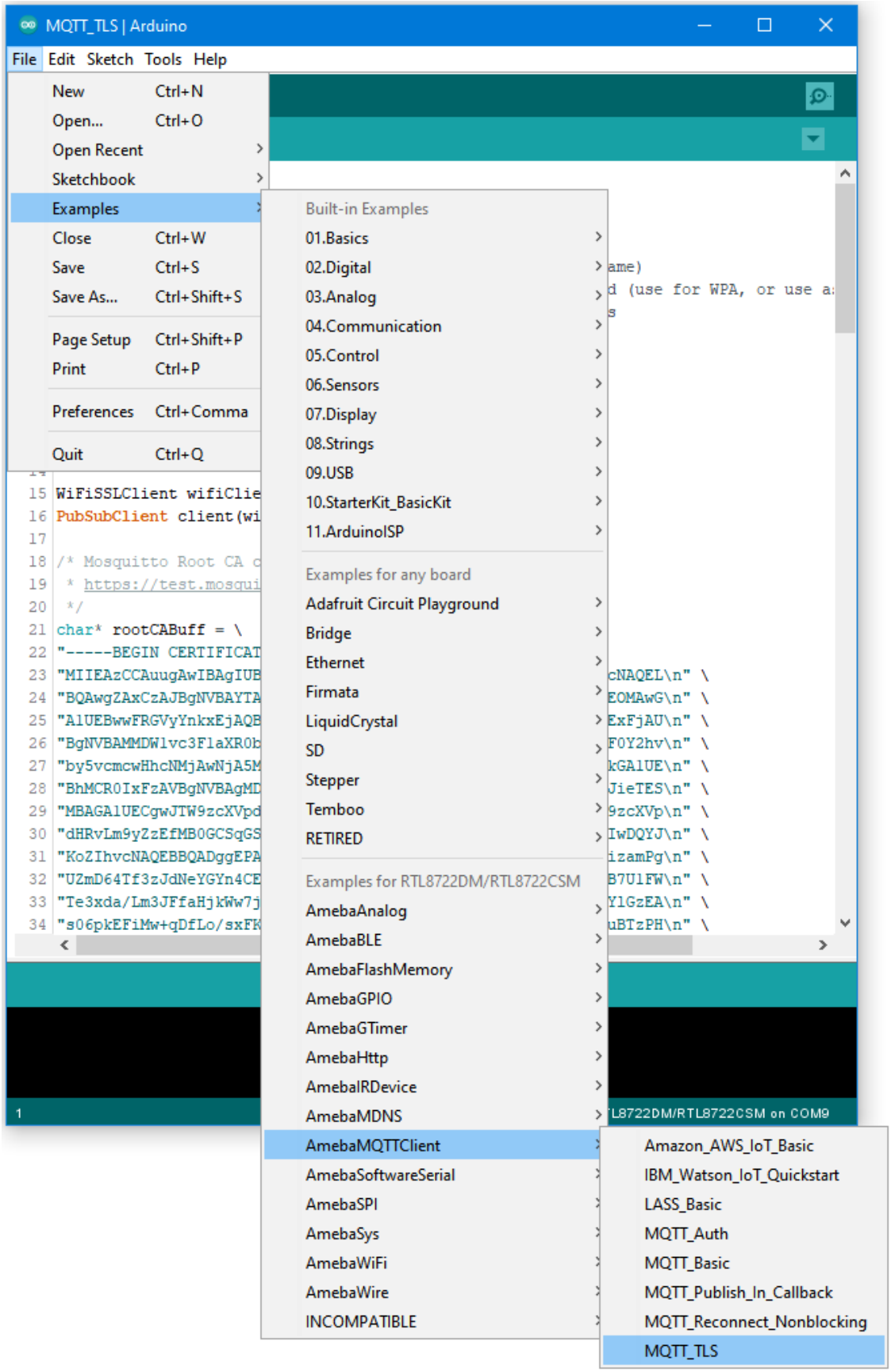
MQTT - Set up MQTT Client over TLS

Preparation

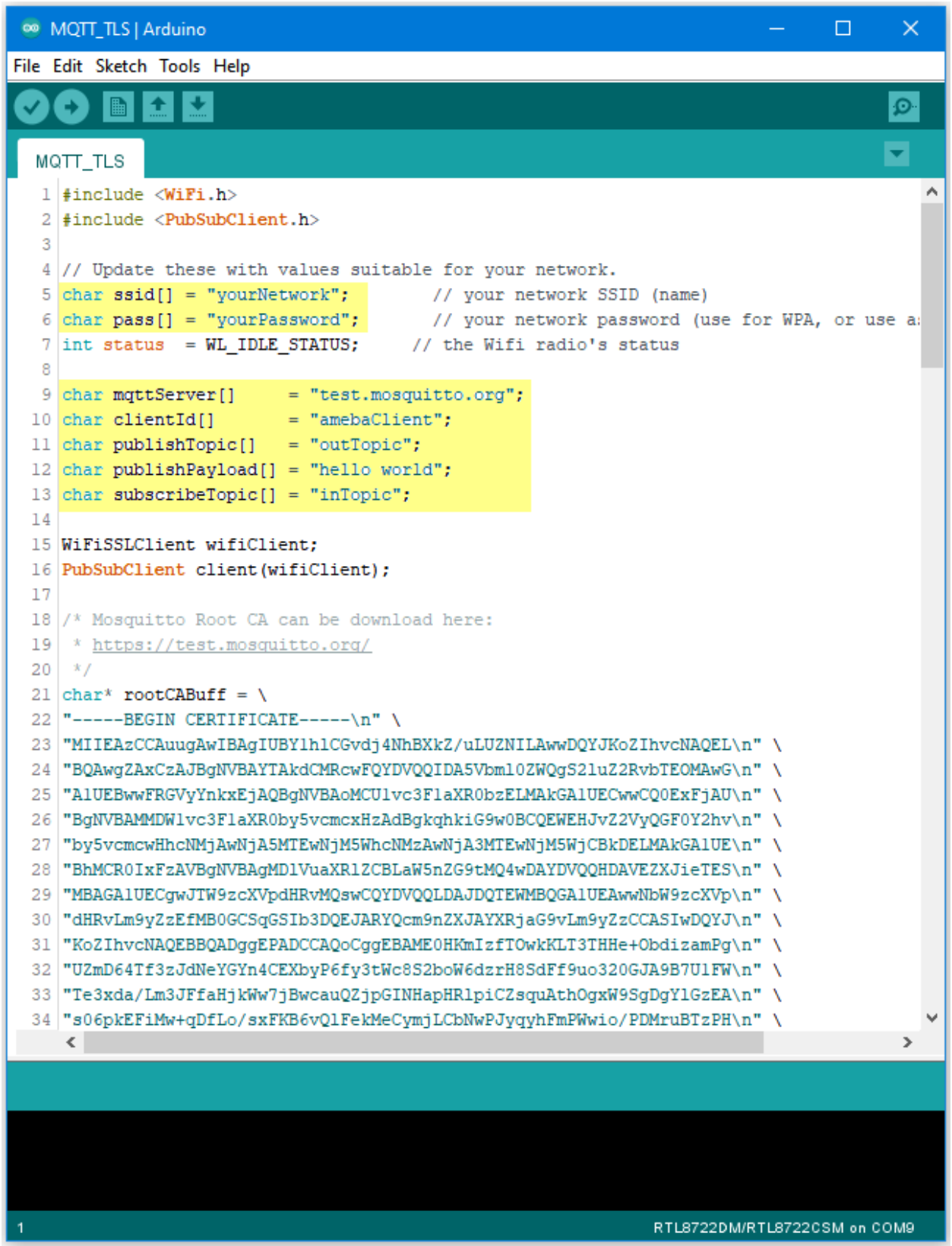
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to a MQTT broker using TLS authentication. Then send messages as a publisher and receive messages from as a subscriber. Open the MQTT example "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`MQTT_TLS`"

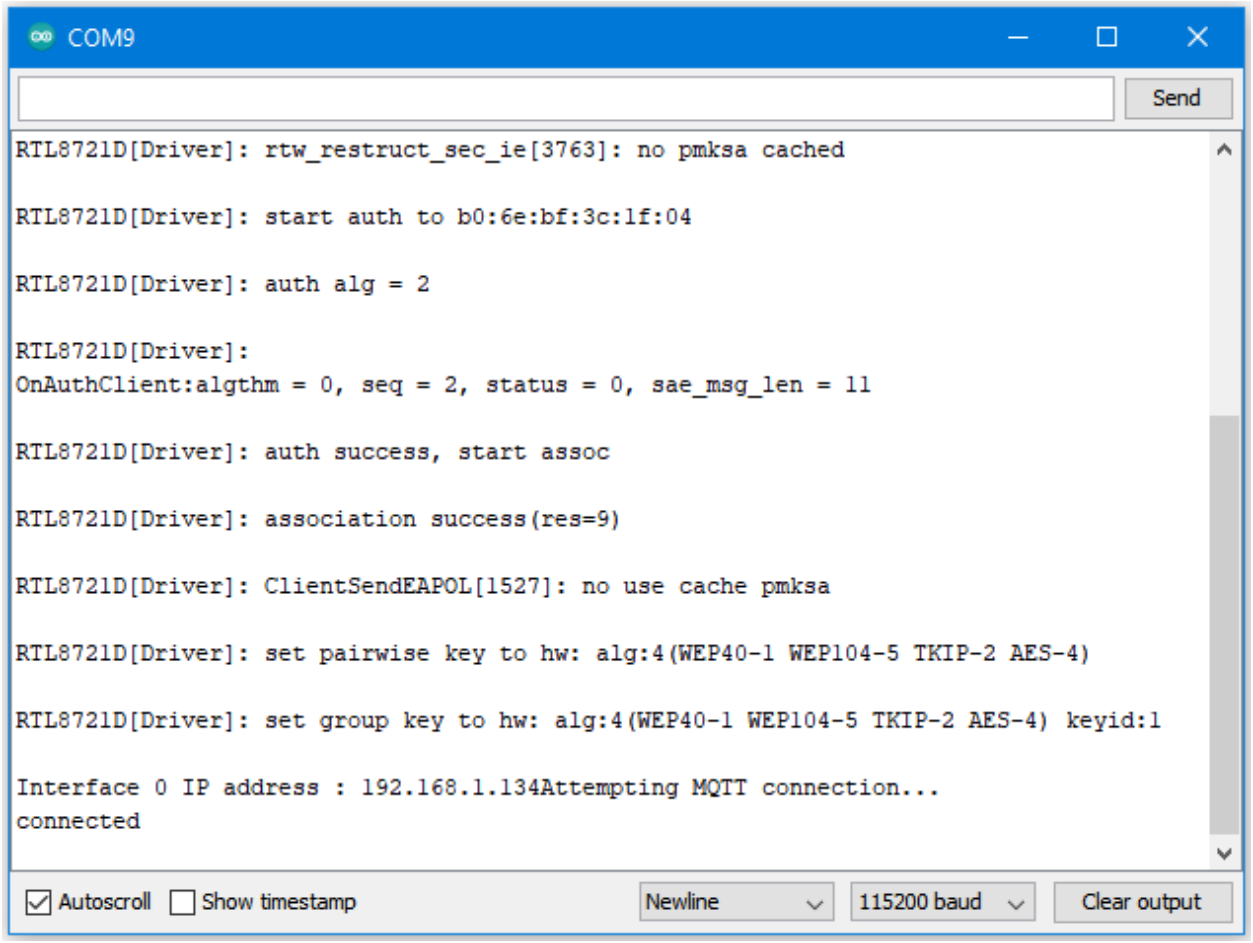

MQTT_TLS | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
Amazon_AWS_IoT_Basic
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking
MQTT_TLS

Please modify the WiFi-related parameters to connect to your WiFi network.
Modify the MQTT parameters to fit your application:



The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.
“clientId” is an identifier for MQTT-Broker to identify the connected device.
“publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
“publishPayload” is the content to be published.
“subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

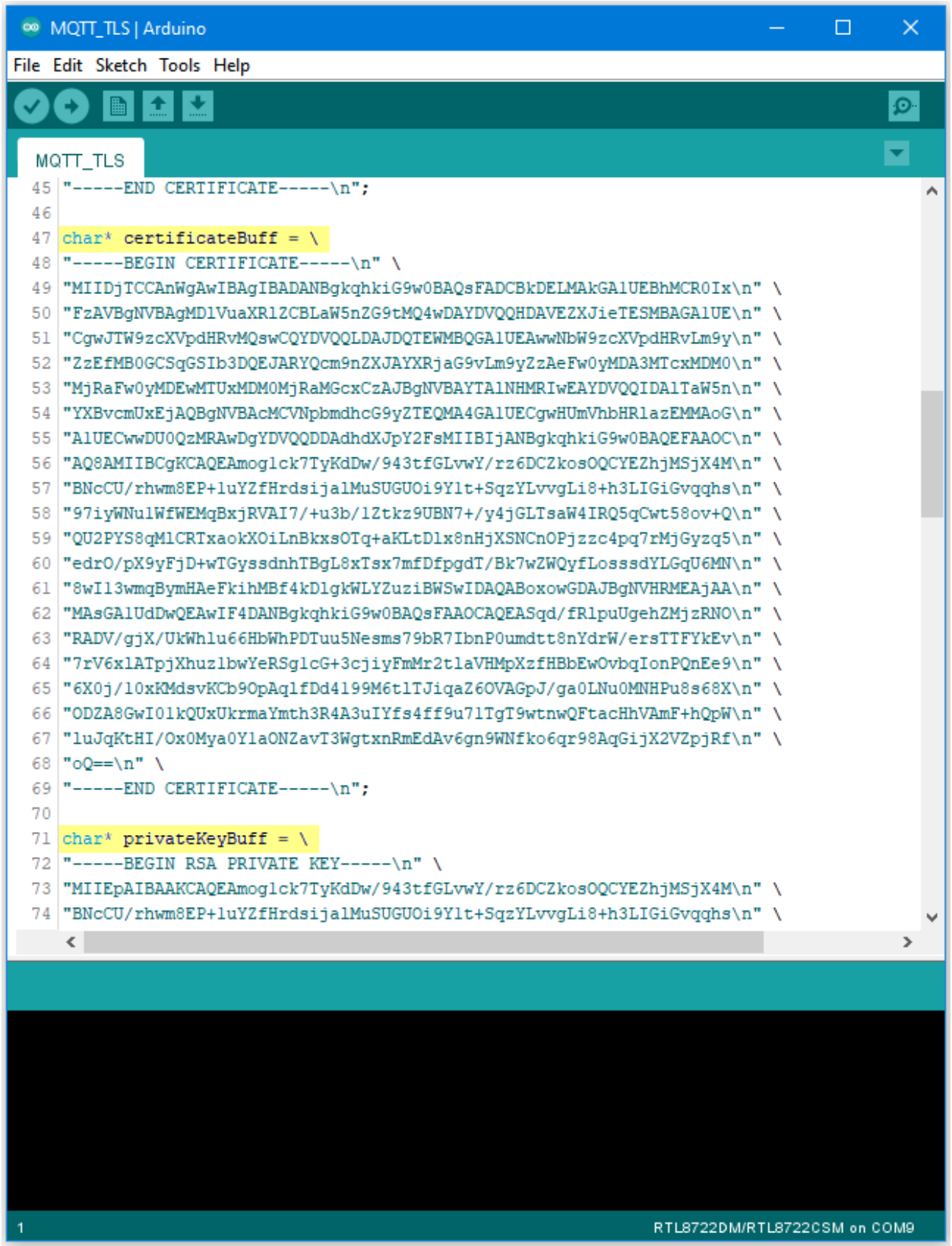
Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor

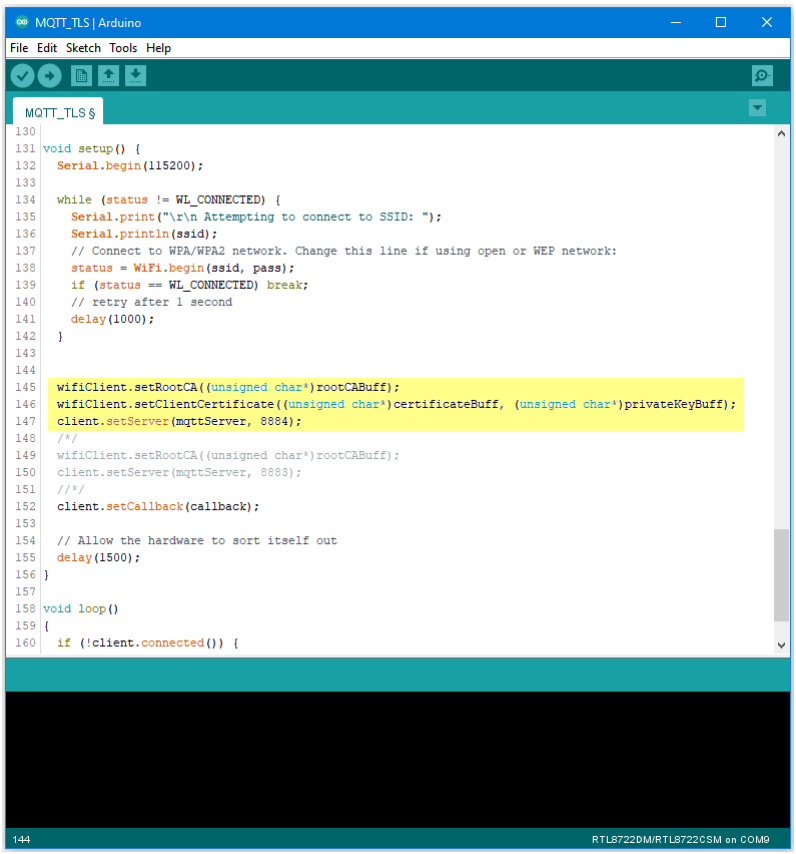


After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”. To see the message, use another MQTT client. Refer to the MQTT_Basic example guide on how to setup a PC-based MQTT client.

If you wish to use TLS client authentication in addition to server authentication, you will need to generate an OpenSSL private key and obtain a signed certificate from the server. For testing purposes, signed certificates can be obtained from test.mosquitto.org by following the guide at https://test.mosquitto.org/ssl/.

Replace the character strings “certificateBuff” and “privateKeyBuff” with your signed certificate and OpenSSL private key, ensuring that they are formatted the same way as the shown in the example code. Also uncomment the highlighted code to enable client authentication, and to change the MQTT port number.


MQTT_TLS | Arduino
File Edit Sketch Tools Help
MQTT_TLS
"-----END CERTIFICATE-----\n";
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDjTCCAnWgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix\n" \
"FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE\n" \
"CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y\n" \
"ZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzAeFw0yMDA3MTcxMDM0\n" \
"MjRaFw0yMDEwMTUxMDM0MjRaMGcxCzAJBgNVBAYTAlNHMRIwEAYDVQQIDAlTaW5n\n" \
"YXBvcmUxEjAQBgNVBAcMCVNpbmdhcG9yZTEQMA4GA1UECgwHUmVhbHRlazEMMAoG\n" \
"A1UECwwDU0QzMRAwDgYDVQQDDAdhdXJpY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOC\n" \
"AQ8AMIIBCgKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
"97iyWNulWfWEMqBxjRVAI7/+u3b/lZtkz9UBN7+/y4jGLTsaW4IRQ5qCwt58ov+Q\n" \
"QU2PYS8qMlCRTxaokXOiLnBkxsOTq+aKLtDlx8nHjXSNCnOPjzzc4pq7rMjGyzq5\n" \
"edrO/pX9yFjD+wTGyssdnhTBgL8xTsx7mfDfpgdT/Bk7wZWQyfLosssdYLGqU6MN\n" \
"8wIl3wmqBymHAeFkihMBf4kDlgkWLYZuziBWSwIDAQABoxowGDAJBgNVHRMEAjAA\n" \
"MAsGA1UdDwQEAwIF4DANBgkqhkiG9w0BAQsFAAOCAQEASqd/fRlpuUgehZMjzRNO\n" \
"RADV/gjX/UkWhlu66HbWhPDTuu5Nesms79bR7IbnP0umdtt8nYdrW/ersTTFYkEv\n" \
"7rV6xlATpjXhuzlbwYeRSglcG+3cjiyFmMr2tlaVHMpXzfHBbEwOvbqIonPQnEe9\n" \
"6X0j/l0xKMdsvKCb9OpAqlfDd4199M6tlTJiqaZ6OVAGpJ/ga0LNu0MNHPu8s68X\n" \
"ODZA8GwI0lkQUxUkrmaYmth3R4A3uIYfs4ff9u7lTgT9wtnwQFtacHhVAmF+hQpW\n" \
"luJqKtHI/Ox0Mya0YlaONZavT3WgtxnRmEdAv6gn9WNfko6qr98AqGijX2VZpjRf\n" \
"oQ==\n" \
"-----END CERTIFICATE-----\n";
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEpAIBAAKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
1
RTL8722DM/RTL8722CSM on COM9



MQTT - Use Amazon AWS IoT Shadow Service

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

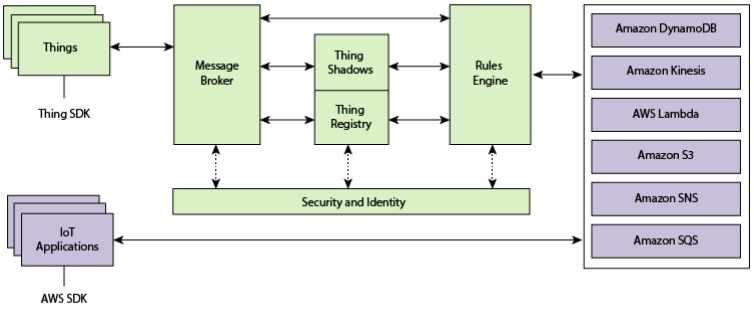
Example

Introduction

Amazon AWS IoT is a cloud IoT service platform:

Amazon AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline. (https://aws.amazon.com/iot/how-it-works/)

The service architecture of AWS IoT:

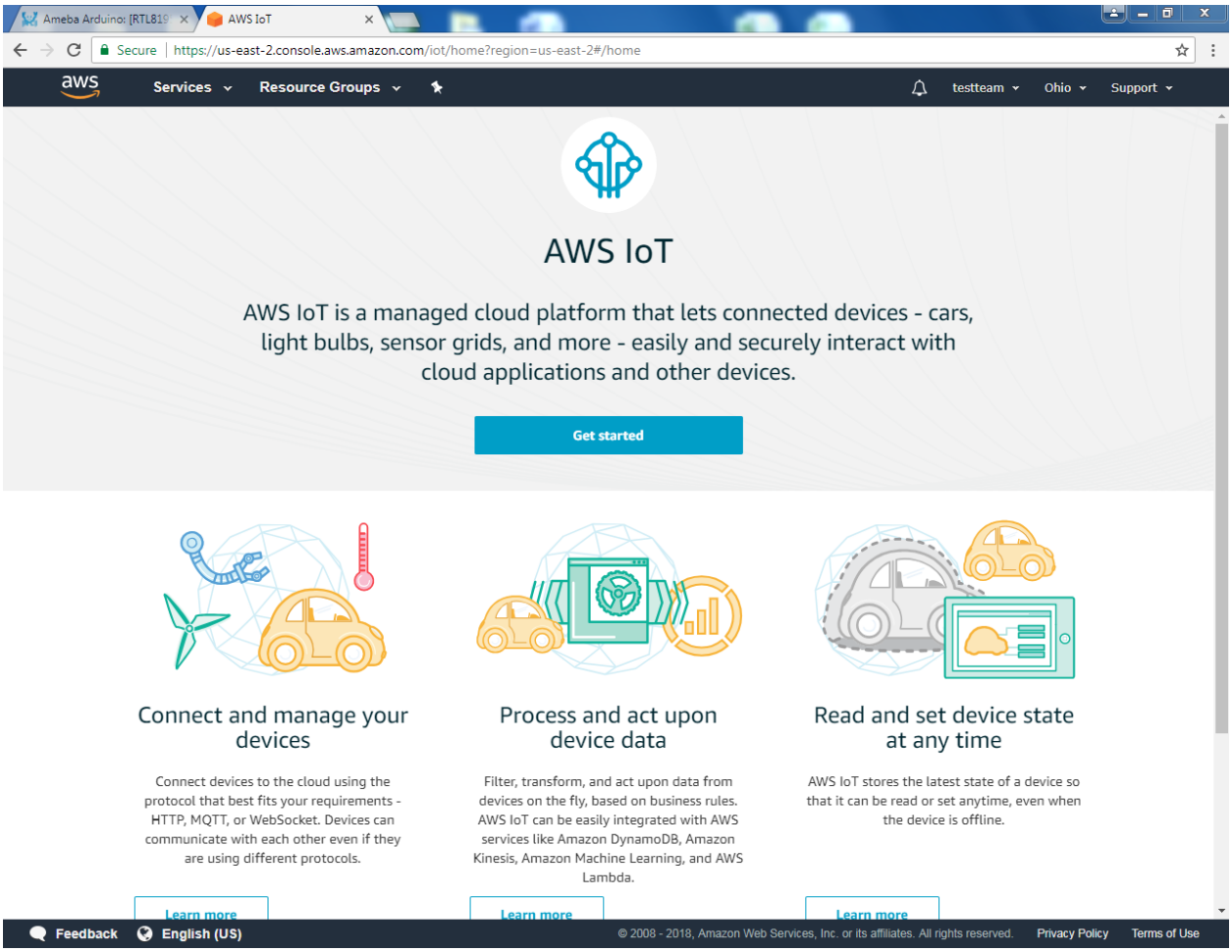


(Picture from http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html)
In the architecture, Ameba belongs to the upper-left “Things” block. A TLS secure channel will be established between “Things” and the MQTT Message Broker. Afterwards, “Things” and “Message Broker” communicate using MQTT Protocol via this secure channel. Behind the “Message Broker”, the “Thing Shadows” keeps messages temporarily when Ameba is offline, and sends the control message to Ameba next time it is connected. The “Rules Engine” allows you to place restrictions to the behavior of Things or to connect Things to other services of Amazon.

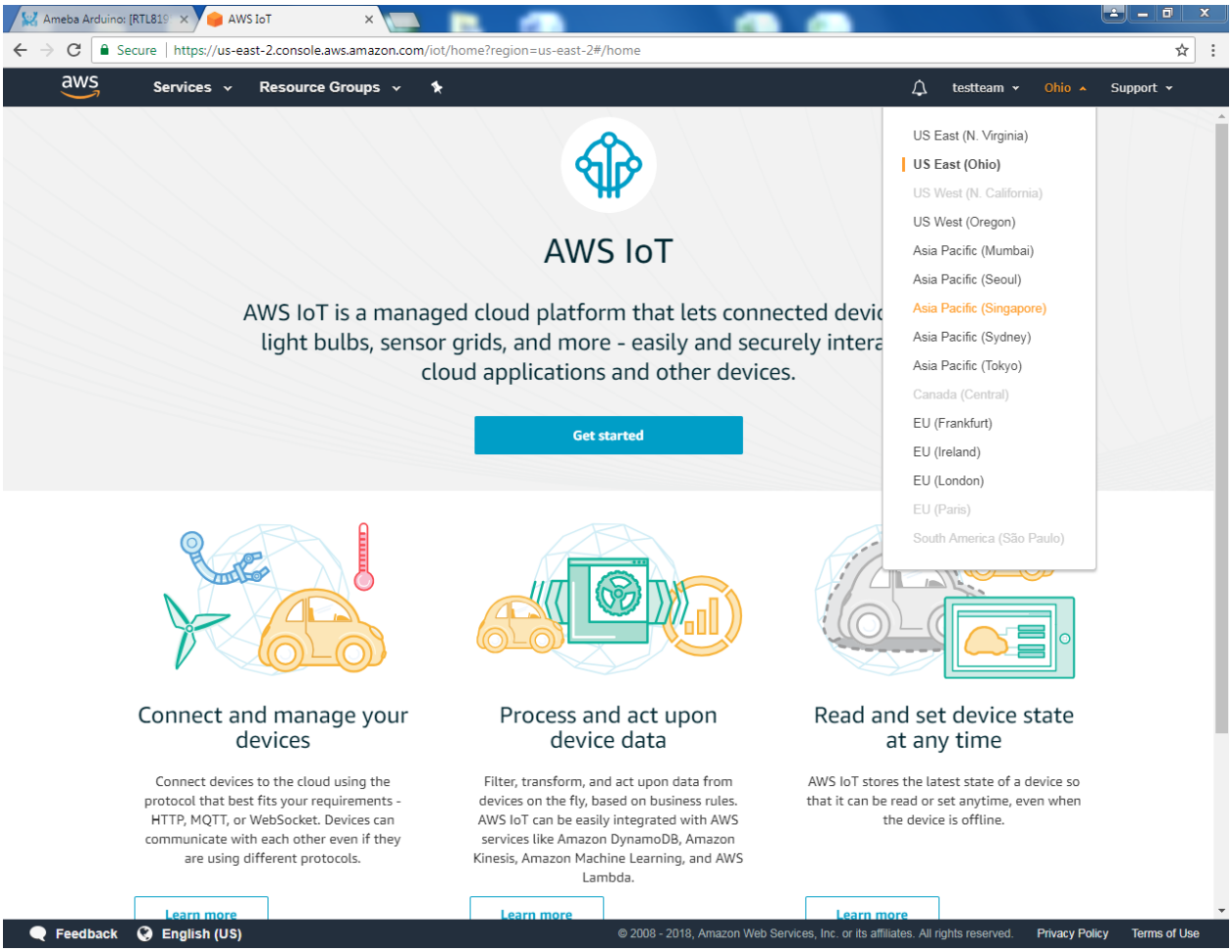
AWS Management Console

First, create an account and sign up for AWS IoT service:https://aws.amazon.com/
Afterwards, log in to the Amazon Management Console and click “IoT Core” found under services -> Internet of Things.

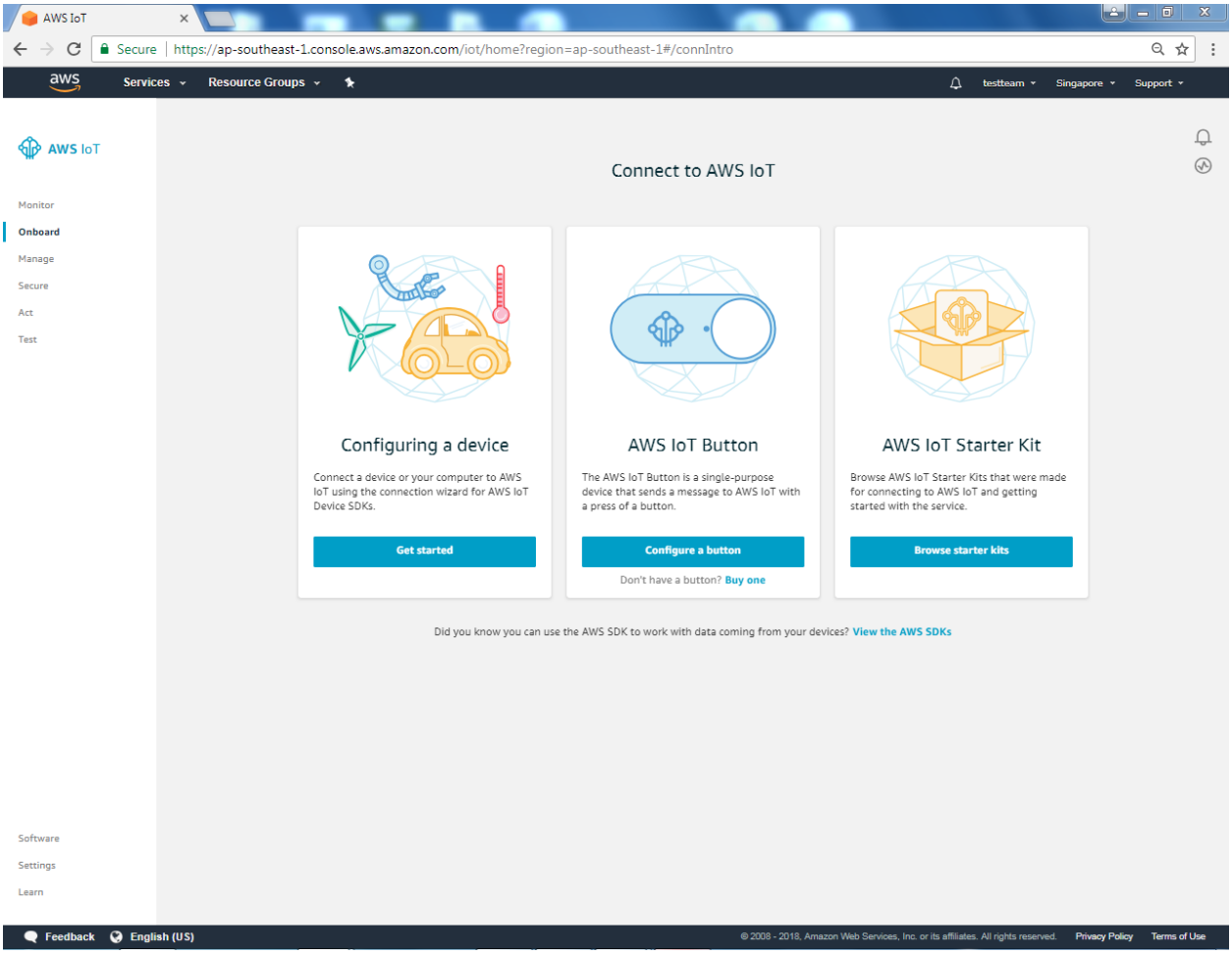
Then you will enter the home page of AWS IoT. To offer the best service quality, Amazon offers servers in different regions for users to choose from.
Click the region dropdown menu at the upper-right:



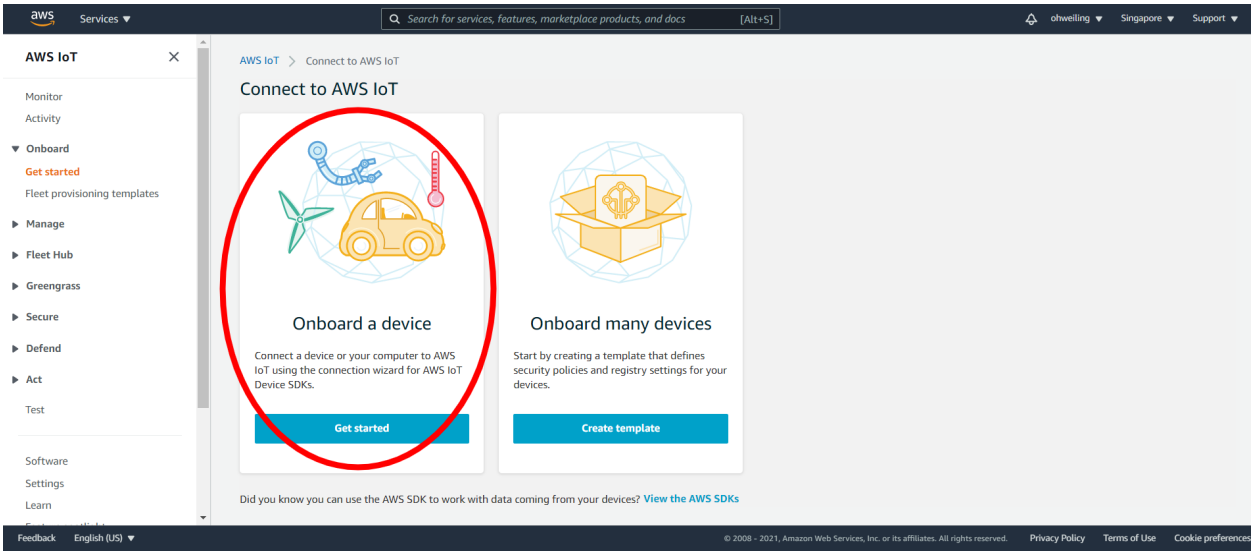
Choose a nearby region.



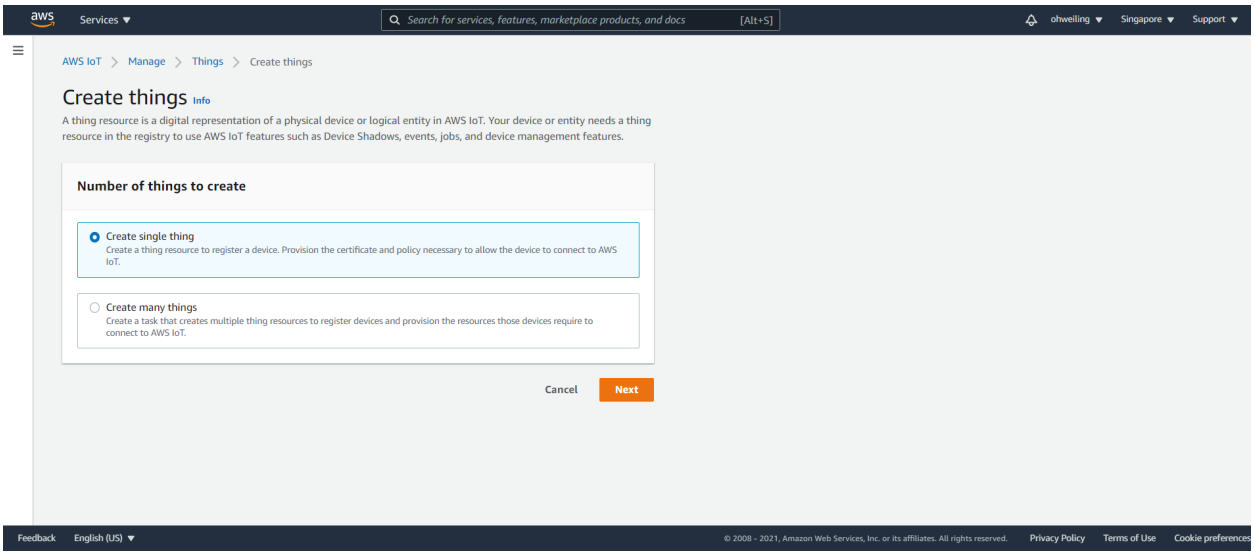
Then from Services, go to Onboard then Get Started.



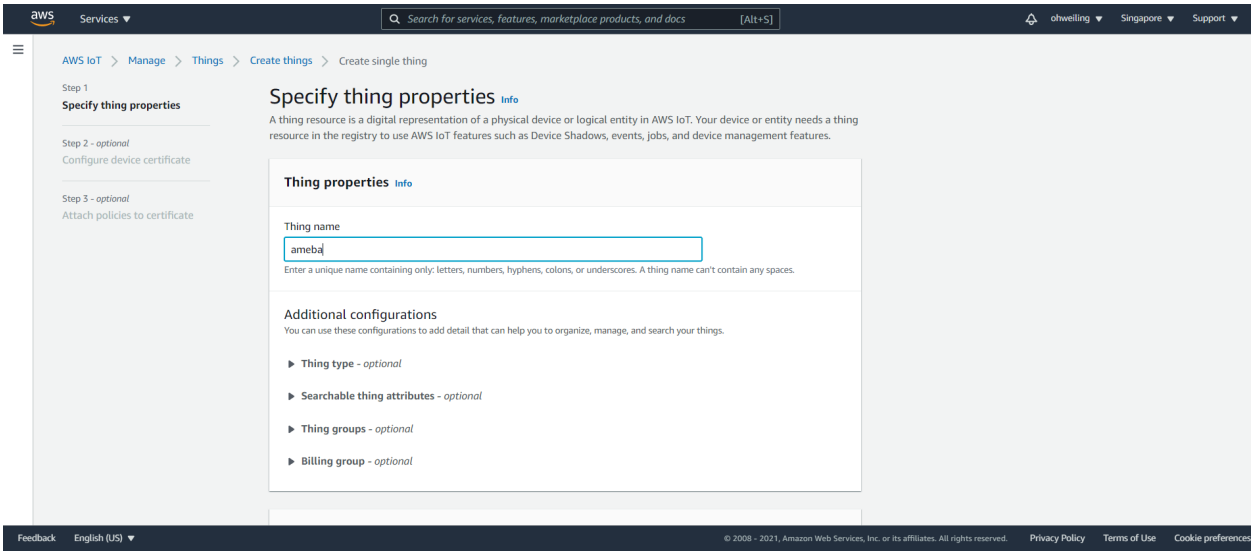
Enter the main page of AWS IoT. Under the Onboard a device, click Get started.



Click Create single thing

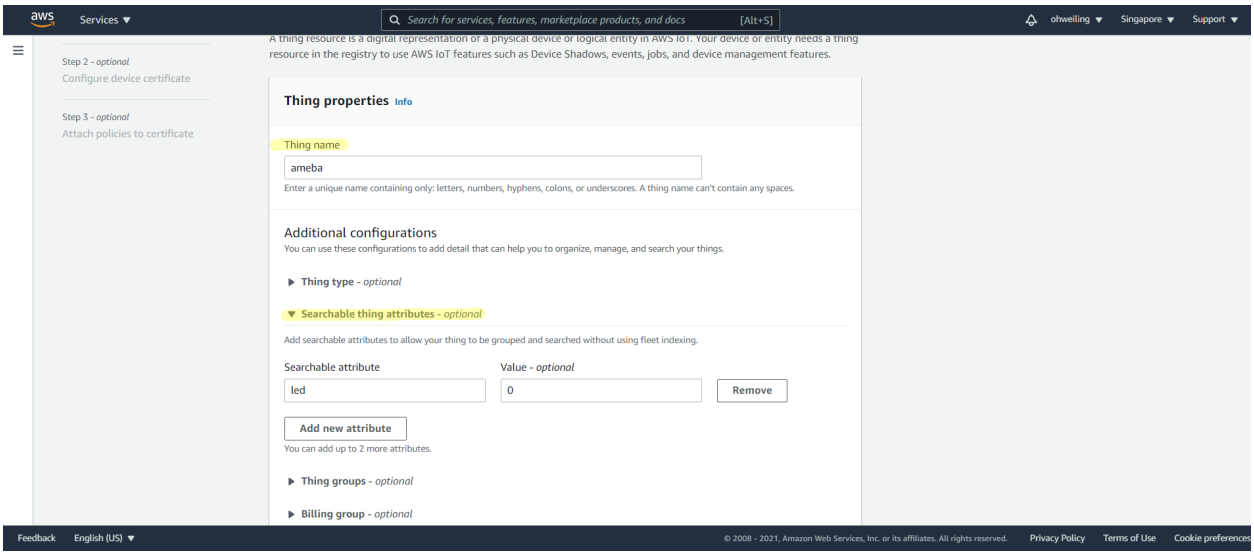


Fill in “ameba” on the name field. Attributes represent the status of Ameba.

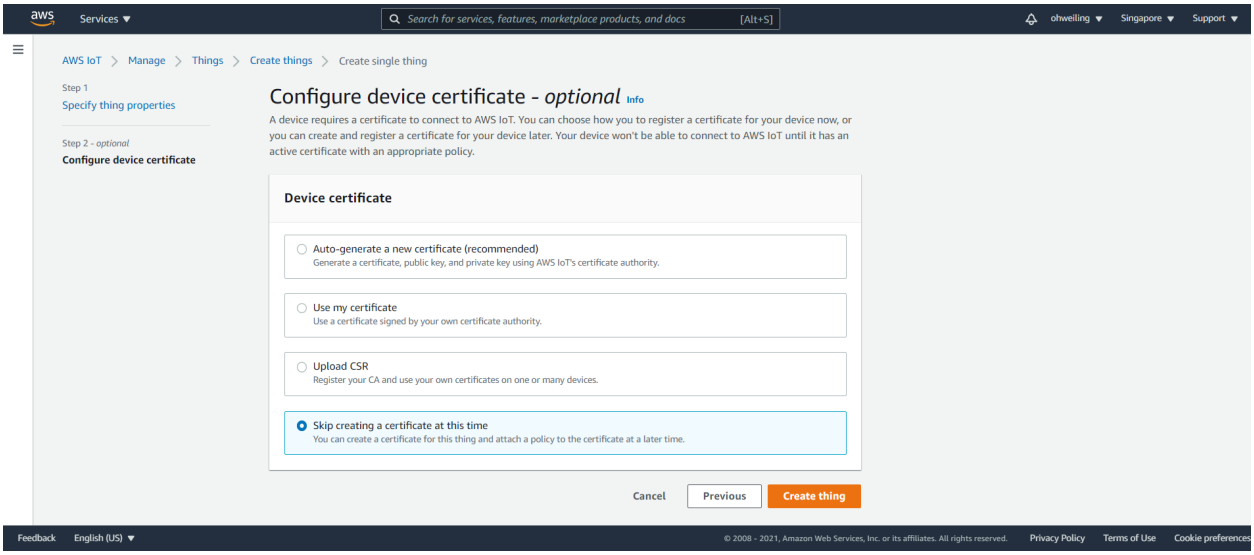


Under the searchable thing attributes. The value of the attributes can be updated directly by Ameba or by the control side and control side can request Ameba to set the attribute to desired value.

Here we add an attribute named “led” with value “0” and click “Next”.



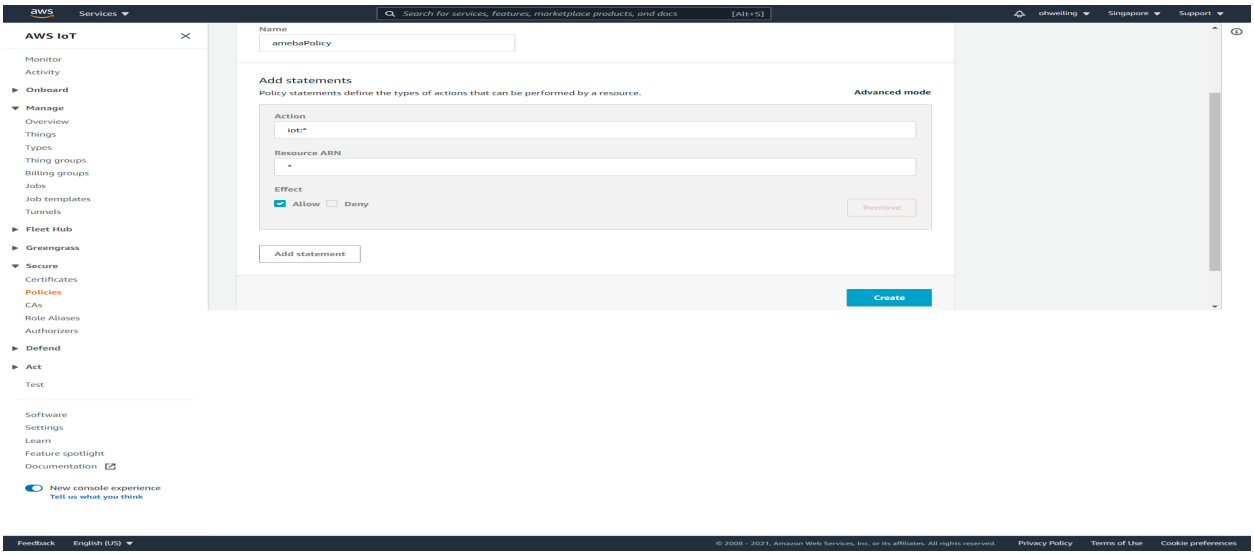
Click Skip creating a certificate at this time and then Create thing



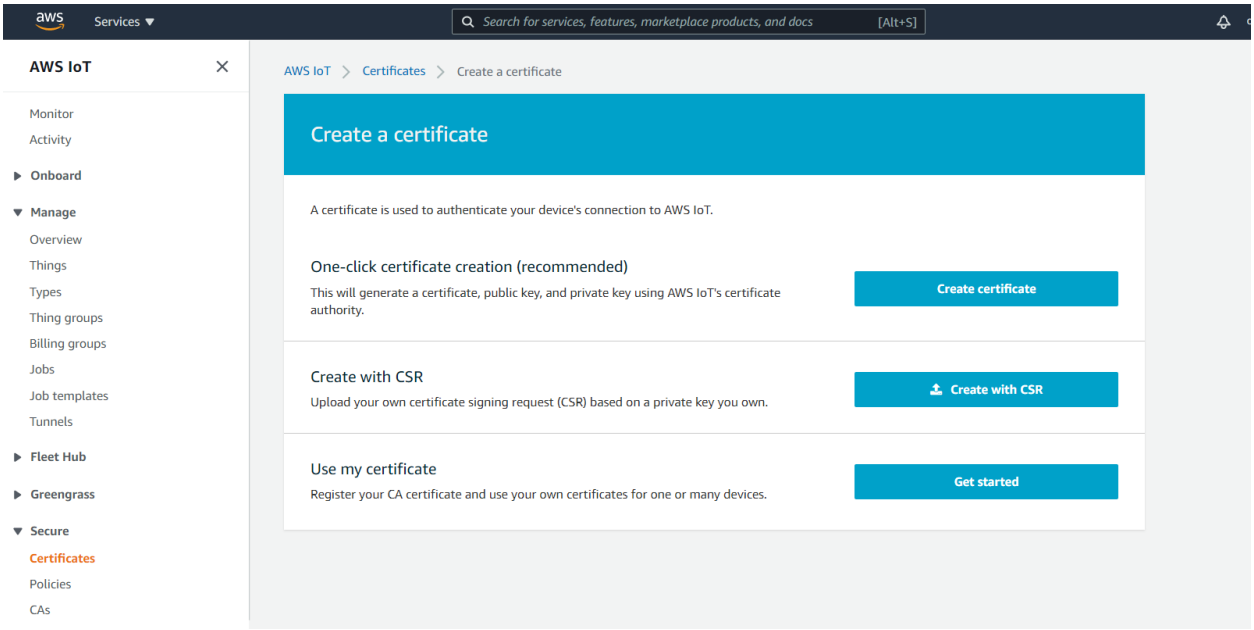
Next, click Policy¸ and create a policy. Policy is used to restrict the functions that a “thing” can do, it can limit the MQTT actions or specific topic that can be performed. Learn more about policy:

http://docs.aws.amazon.com/iot/latest/developerguide/authorization.html

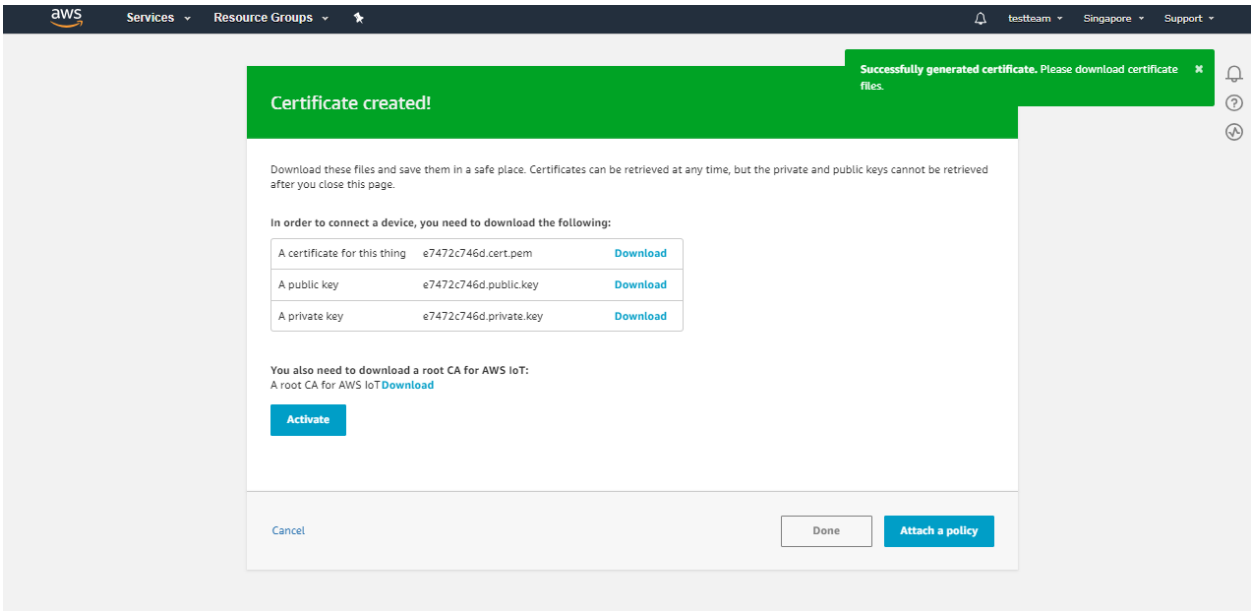
Here we do not place policy on Ameba. Fill in “amebaPolicy” in the Name field, “iot:” *in Action field and* “” in resources field. Then “Allow”. Finally, click “Create”.



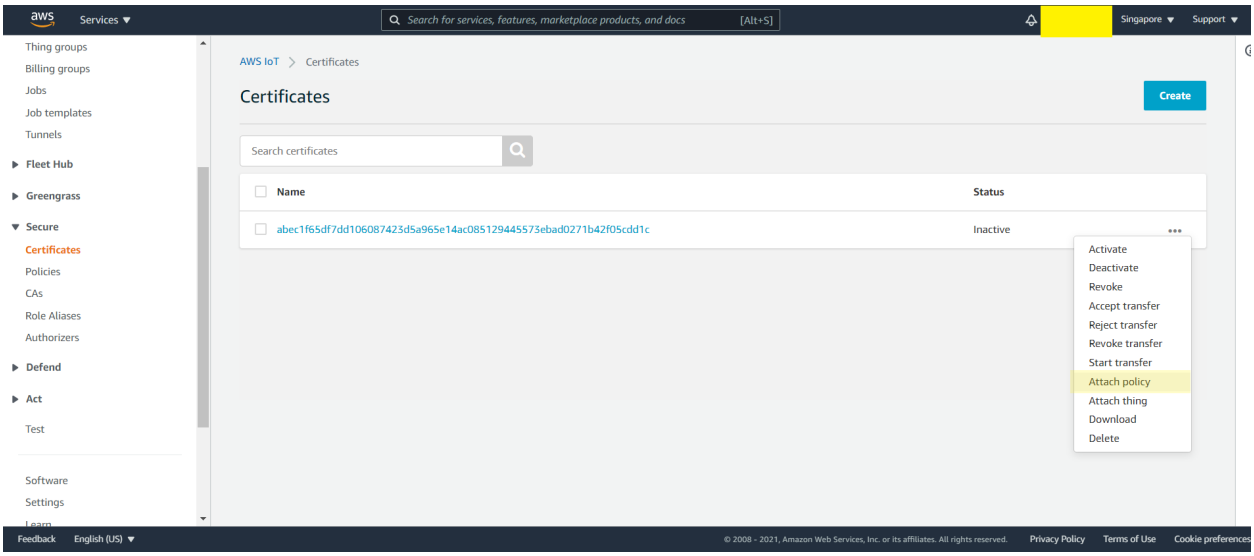
Next, we have to setup the TLS certificate. You can choose to user-defined or generate a certificate by AWS IoT. In this example we click Create Certificate to generate a TLS certificate.



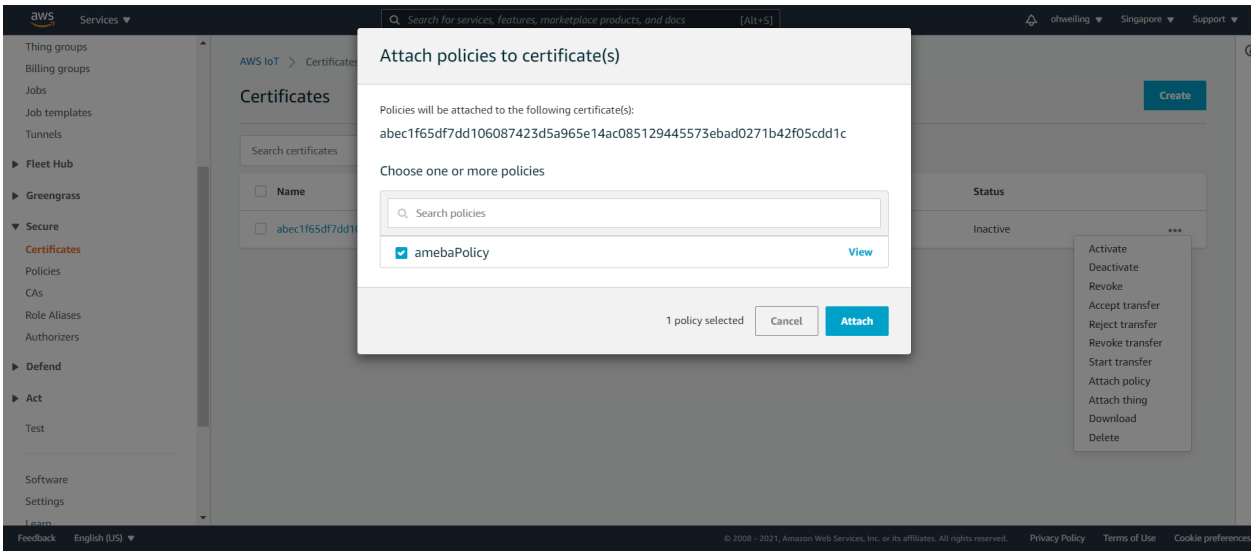
You can see 4 Links. Please download each of the link, “public key”, “private key”, “Certificate” and “rootCA”. After downloading the 4 files, click Done and go back to the certificate main page.



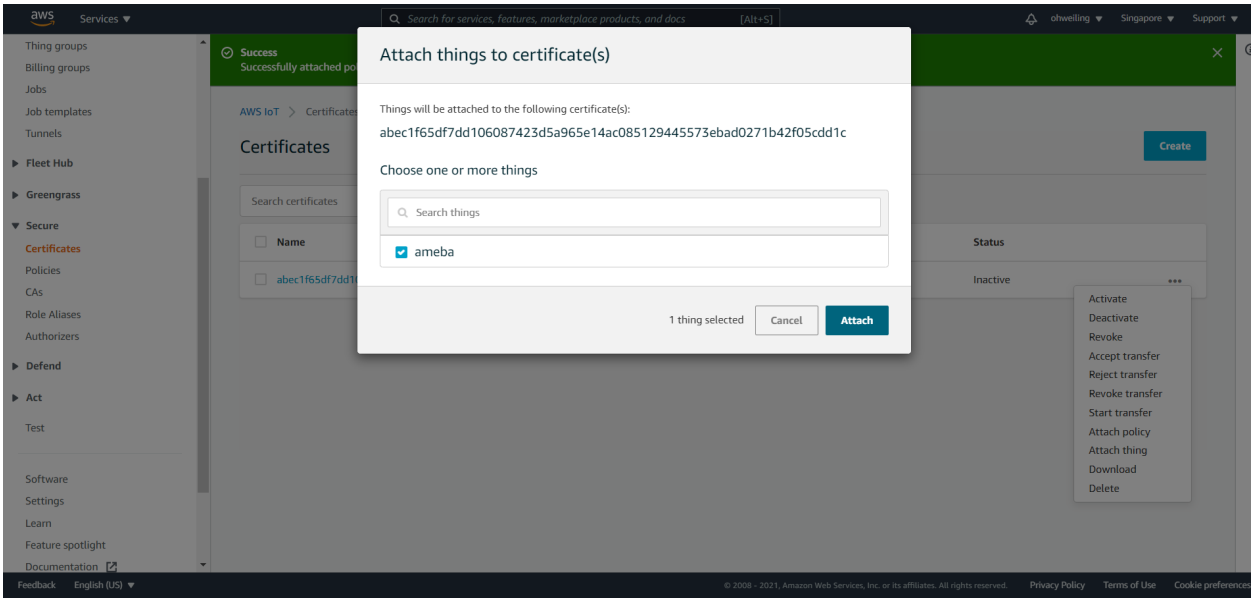
Click Attach a policy in the Actions dropdown menu.



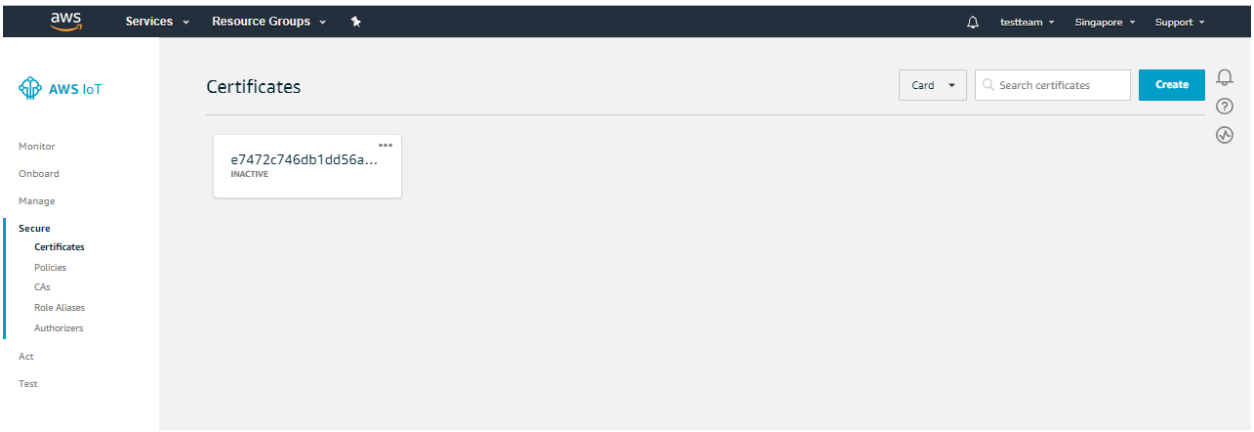
Choose amebaPolicy and click attach.



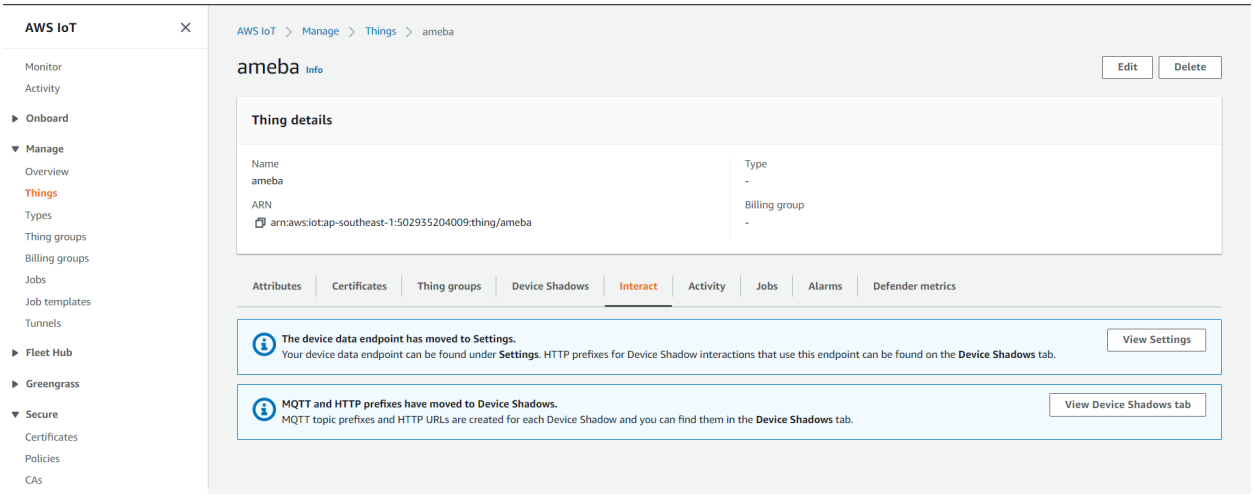
Then go back to the “Actions” drop-down menu at the top right of the certificates homepage, click on “Attach thing”, select the thing “ameba” you just created when the window below appears, then click on “Attach”



Go back to certificate main page and click Certificate and click Activate in the Actions drop down menu.



Next, click Manage, and click Things, then click “ameba” the thing we created just now.
Click on Interact and View settings.

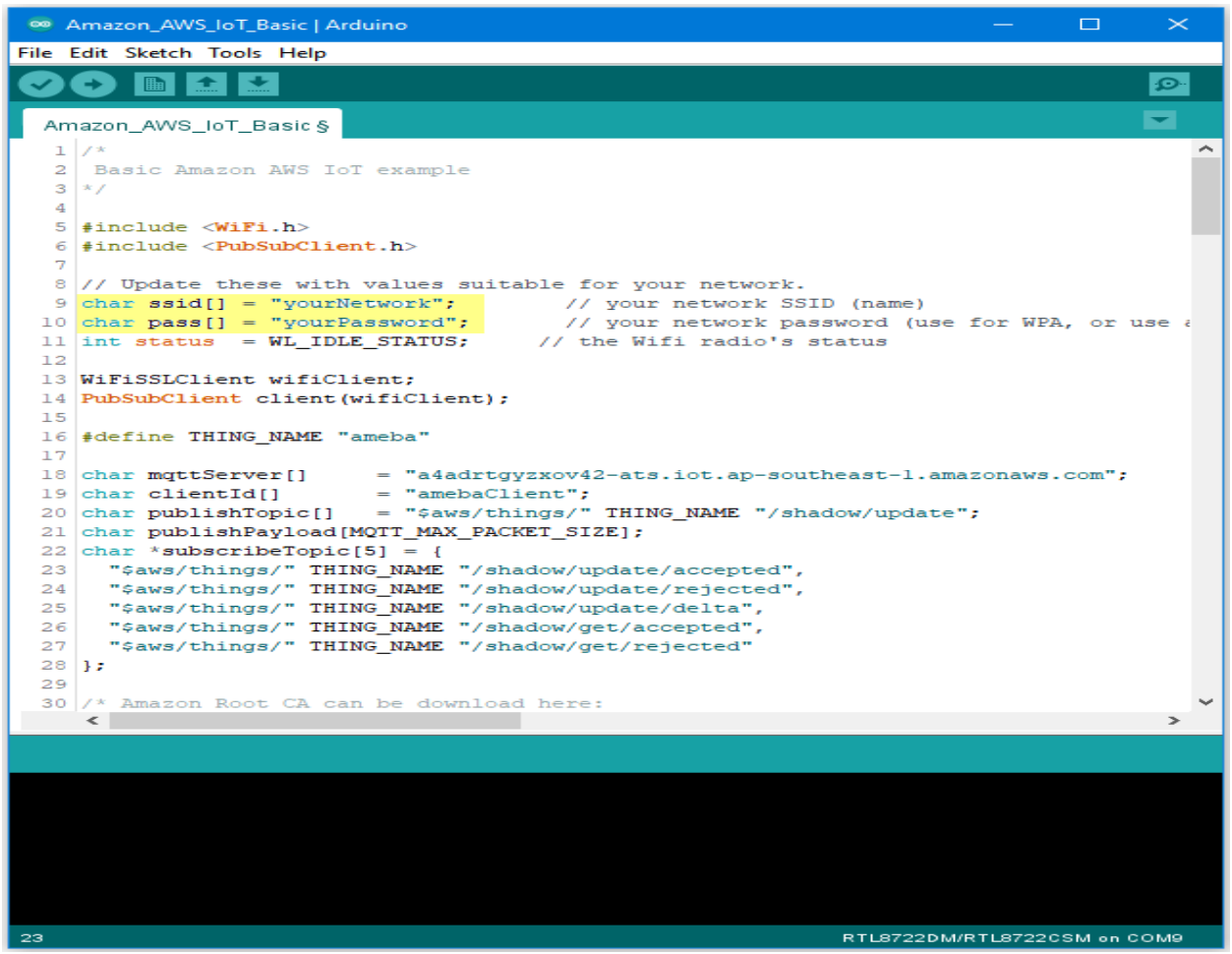


Find out the information of Rest API Endpoint to set Amazon Alexa:

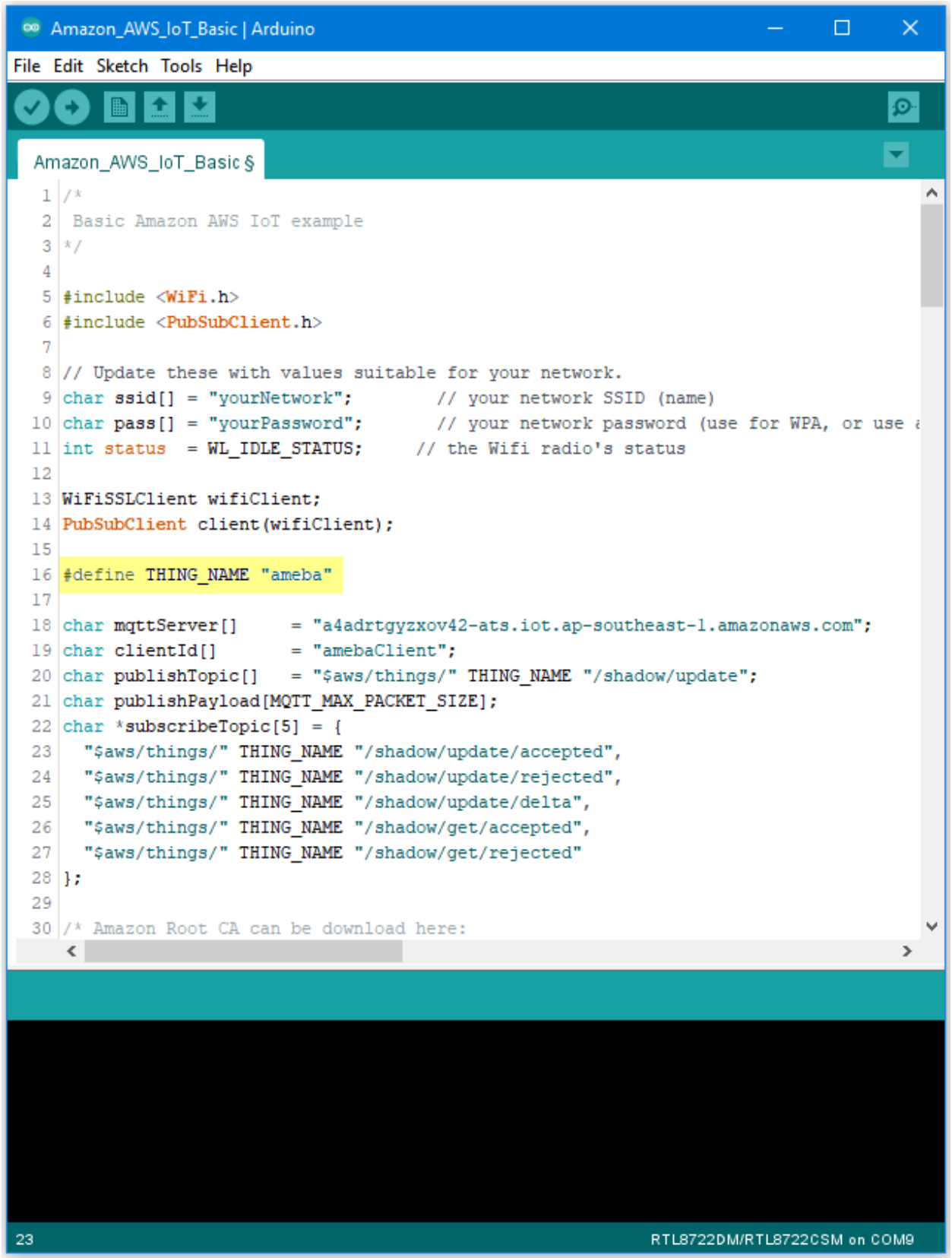
- REST API endpoint: In the value “https://a1a7oo4baosgyy.iot.us-east-1.amazonaws.com/things/ameba/shadow”, the part “a1a7oo4baosgyy.iot.us-east-1.amazonaws.com” is the MQTT Broker server address.
- MQTT topic The value “$aws/things/ameba/shadow/update” represents the MQTT topic we will use in the AWS IoT Shadow service (if we use MQTT only, without AWS IoT Shadow service, then we can specify other topic name). It is recommended to use “$aws/things/ameba/shadow/update” here.

Ameba setting

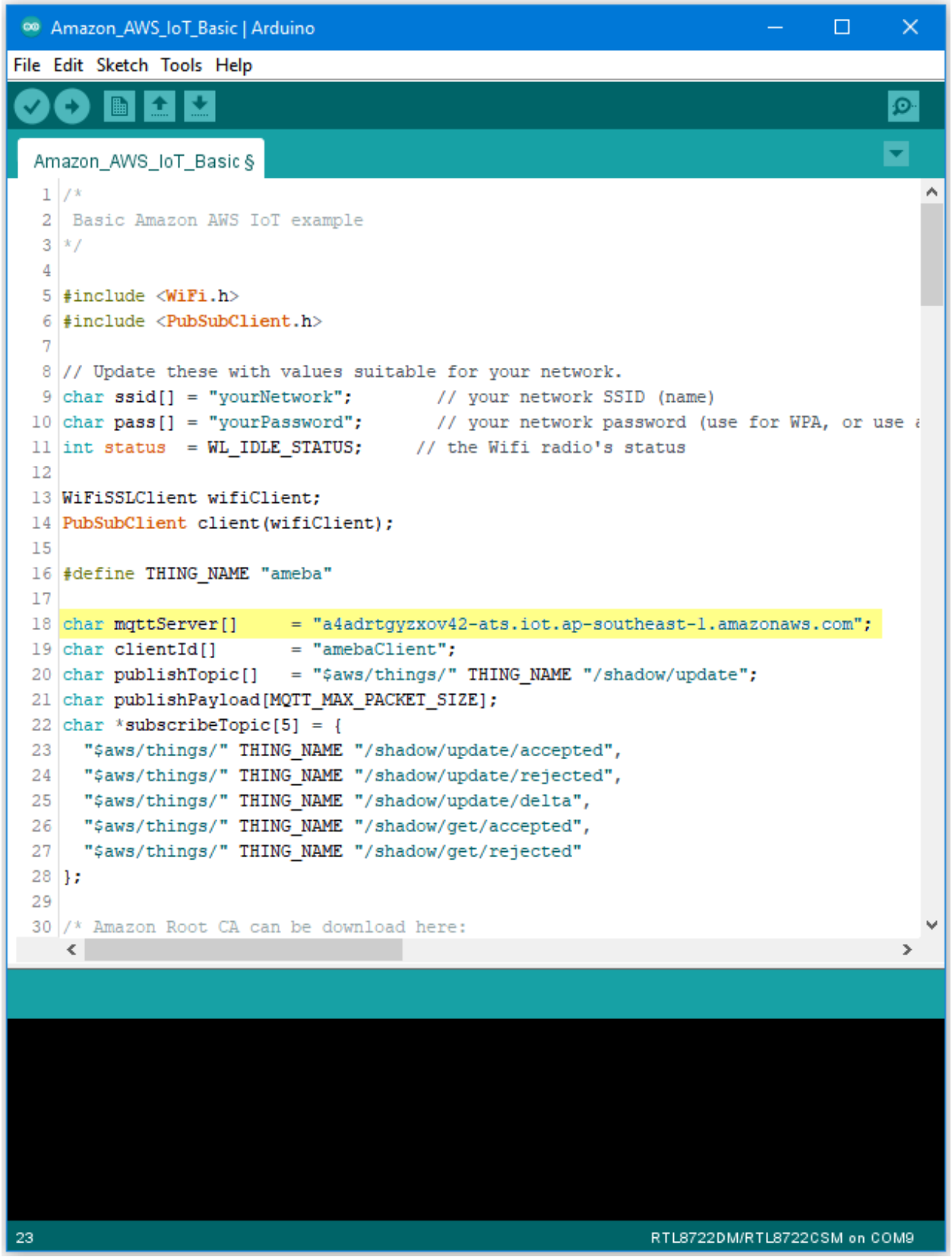
Open “`File` -> “`Examples`” -> “`AmebaMQTTClient`” -> “`Amazon_AWS_IoT_Basic`”
In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



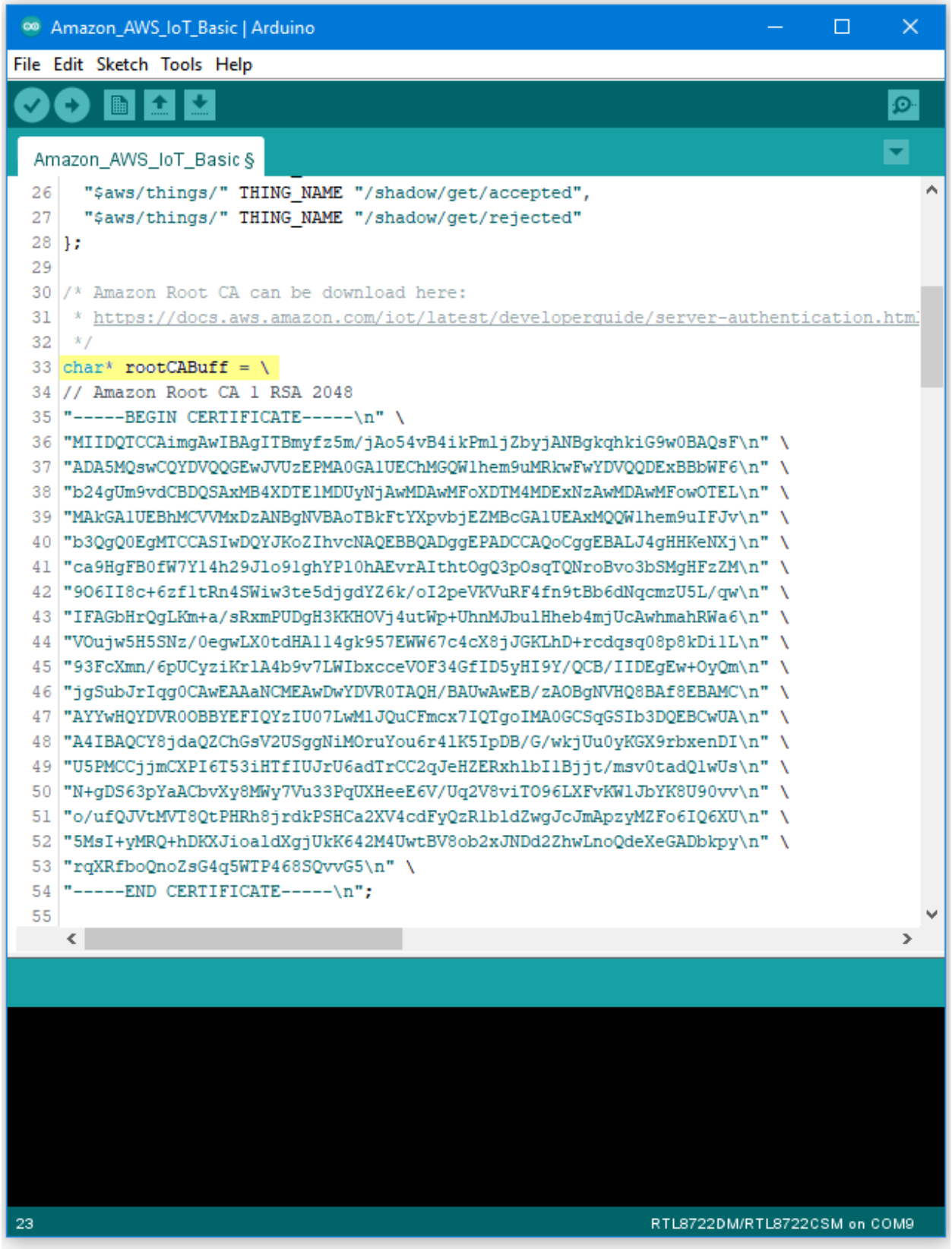
Then fill in the “thing” name “ameba”.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/*
Basic Amazon AWS IoT example
*/
#include <WiFi.h>
#include <PubSubClient.h>
// Update these with values suitable for your network.
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "yourPassword"; // your network password (use for WPA, or use
int status = WL_IDLE_STATUS; // the Wifi radio's status
WiFiSSLClient wifiClient;
PubSubClient client(wifiClient);
#define THING_NAME "ameba"
char mqttServer[] = "a4adrtgyzxov42-ats.iot.ap-southeast-1.amazonaws.com";
char clientId[] = "amebaClient";
char publishTopic[] = "$aws/things/" THING_NAME "/shadow/update";
char publishPayload[MQTT_MAX_PACKET_SIZE];
char *subscribeTopic[5] = {
"$aws/things/" THING_NAME "/shadow/update/accepted",
"$aws/things/" THING_NAME "/shadow/update/rejected",
"$aws/things/" THING_NAME "/shadow/update/delta",
"$aws/things/" THING_NAME "/shadow/get/accepted",
"$aws/things/" THING_NAME "/shadow/get/rejected"
};
/* Amazon Root CA can be download here:
23
RTL8722DM/RTL8722CSM on COM9

And the MQTT Broker server address we found earlier in AWS IoT.



Next, fill in the root CA used in TLS. Download and make sure the downloaded root CA contents conforms to the root CA used in the sketch.



Next, fill in the certificate we created in the AWS IoT Console (i.e., client certificate), usually its file name ends with "-certificate.pem.crt" (e.g., "efae24a533-certificate.pem.crt"). Open the certificate with a text editor, and adjust its format as follows to use in the sketch:

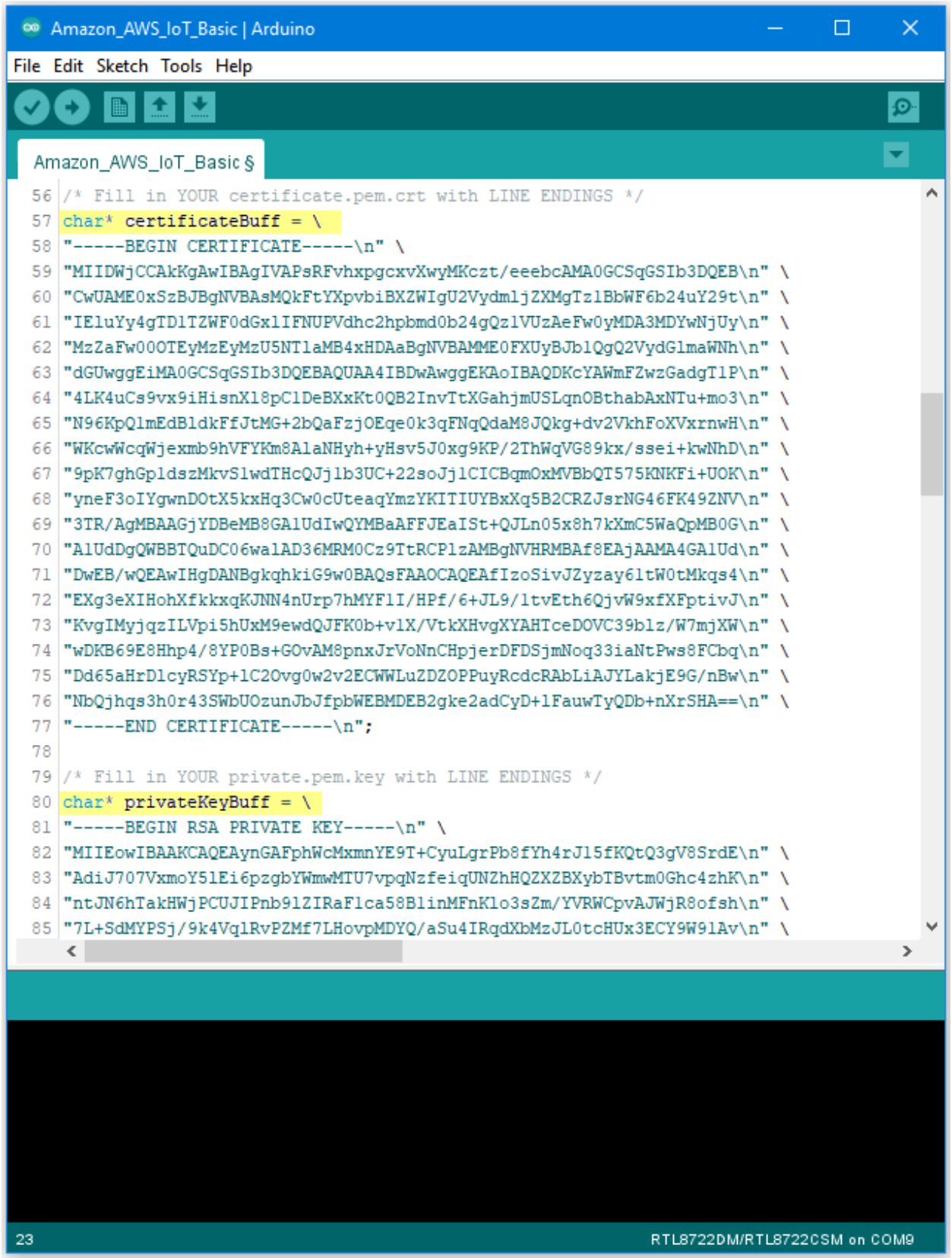
– Add the new line character "n" at the end of each line.

– Add double-quote at the beginning and the end of each line.

– To concatenate each line as a string, add "" at the end of each line.

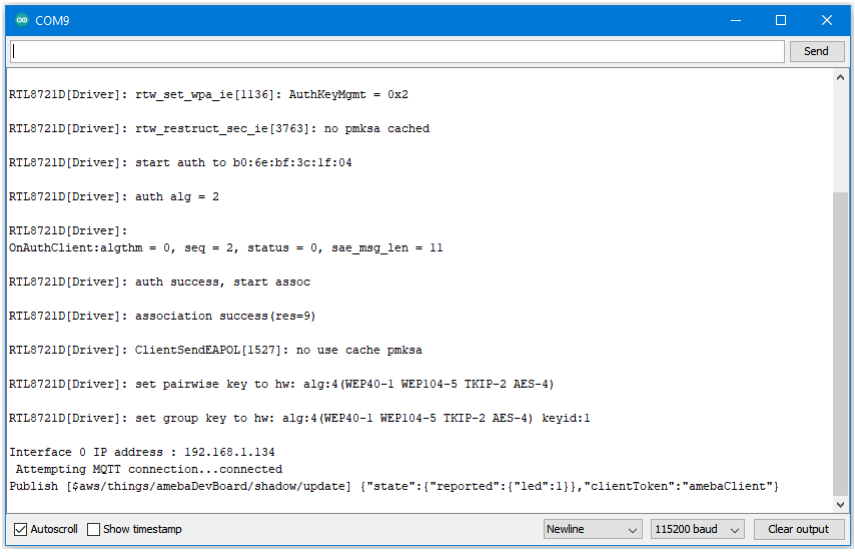
– The last line ends with semicolon.

Adjust the format of the private key in the same way and add it to privateKeyBuff.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/* Fill in YOUR certificate.pem.crt with LINE ENDINGS */
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDWjCCAkKgAwIBAgIVAPsRFvhxpgcxvXwyMKczt/eeebcAMA0GCSqGSIb3DQEB\n" \
"CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTzlBbWF6b24uY29t\n" \
"IEluYy4gTDlTZWF0dGxlIFNUPVdhc2hpbmd0b24gQzlVUzAeFw0yMDA3MDYwNjUy\n" \
"MzZaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJblQgQ2VydGlmaWNh\n" \
"dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDKcYAWmFZwzGadgTlP\n" \
"4LK4uCs9vx9iHisnXl8pClDeBXxKt0QB2InvTtXGahjmUSLqnOBthabAxNTu+mo3\n" \
"N96KpQlmEdBldkFfJtMG+2bQaFzjOEqe0k3qFNqQdaM8JQkg+dv2VkhFoXVxrnwH\n" \
"WKcwWcqWjexmb9hVFYKm8AlaNHyh+yHsv5J0xg9KP/2ThWqVG89kx/ssei+kwNhD\n" \
"9pK7ghGpldszMkvSlwdTHcQJjlb3UC+22soJjlCICBqmOxMVBbQT575KNKFi+UOK\n" \
"yneF3oIYgwnDOtX5kxHq3Cw0cUteaqYmzYKITIUYBxXq5B2CRZJsrNG46FK49ZNV\n" \
"3TR/AgMBAAGjYDBeMB8GA1UdIwQYMBaAFFJEaISt+QJLn05x8h7kXmC5WaQpMB0G\n" \
"A1UdDgQWBBTQuDC06walAD36MRM0Cz9TtRCPlzAMBgNVHRMBAf8EAjAAMA4GA1Ud\n" \
"DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAfIzoSivJZyzay6ltW0tMkqs4\n" \
"EXg3eXIHohXfkkxqKJNN4nUrp7hMYFlI/HPf/6+JL9/ltvEth6QjvW9xfXFptivJ\n" \
"KvgIMyjqzILVpi5hUxM9ewdQJFK0b+vlX/VtkXHvgXYAHTceDOVC39blz/W7mjXW\n" \
"wDKB69E8Hhp4/8YP0Bs+GOvAM8pnxJrVoNnCHpjerDFDSjmNoq33iaNtPws8FCbq\n" \
"Dd65aHrDlcyRSYp+lC2Ovg0w2v2ECWWLuZDZOPPuyRcdcRAbLiAJYLakjE9G/nBw\n" \
"NbQjhqs3h0r43SWbUOzunJbJfpbWEBMDEB2gke2adCyD+lFauwTyQDb+nXrSHA==\n" \
"-----END CERTIFICATE-----\n";
/* Fill in YOUR private.pem.key with LINE ENDINGS */
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEowIBAAKCAQEAynGAFphWcMxmnYE9T+CyuLgrPb8fYh4rJ15fKQtQ3gV8SrdE\n" \
"AdiJ707VxmoY5lEi6pzgbYWmwMTU7vpqNzfeiqUNZhHQZXZBXybTBvtm0Ghc4zhK\n" \
"ntJN6hTakHWjPCUJIPnb9lZIRaF1ca58BlinMFnKlo3sZm/YVRWCpvAJWjR8ofsh\n" \
"7L+SdMYPSj/9k4VqlRvPZMf7LHovpMDYQ/aSu4IRqdXbMzJL0tcHUx3ECY9W9lAv\n"
23
RTL8722DM/RTL8722CSM on COM9

Compile and run

Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor in the Arduino IDE and observe as Ameba connects to the AWS IoT server and sends updates on the LED state variable.



Alternatives

Ameba can also retrieve the current LED status variable from the AWS shadow. This is done by sending a message to the “shadow/get” topic. Refer to the Amazon_AWS_IoT_with_ACK example code for more information.

Code Reference

Change led state:
In this example, we use GPIO interface to control the led. We set led_pin to 10 and led_state to 1 by default in the sample code.

```
pinMode(led_pin, OUTPUT);
digitalWrite(led_pin, led_state);
```

Set up certificate:
Note that we use the WiFiSSLClient type of wifiClient.

```
WiFiSSLClient wifiClient;
```

WiFiSSLClient inherits Client, so it can be passed as the parameter of PubSubClient constructor.
Next, set up TLS certificate required in connection.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
wifiClient.setClientCertificate((unsigned char*)certificateBuff,(unsigned
↪char*)privateKeyBuff);
```

Configure MQTT Broker server
Then MQTT PubClient set MQTT Broker server to connect

```
client.setServer(mqttServer, 8883);
client.setCallback(callback);
```

Connect to MQTT Broker server:
In `loop()`, call `reconnect()` function and try to connect to MQTT Broker server and do the certificate verification.

```
while (!client.connected()) {
```

Subscribe & Publish
Next, subscribe to topics.

```
for (int i=0; i<5; i++) {
  client.subscribe(subscribeTopic[i]);
}
```

There are some common topics:
"$aws/things/ameba/shadow/update/accepted",
"$aws/things/ameba/shadow/update/rejected",
"$aws/things/ameba/shadow/update/delta",
"$aws/things/ameba/shadow/get/accepted",
"$aws/things/ameba/shadow/get/rejected"
Related documentation:
http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-data-flow.html

Then publish current status::

```
sprintf(publishPayload,
"{\"state\":{\"reported\":{\"led\":%d}},\"clientToken\":\"%s\"}",
led_state, clientId);
```

```
client.publish(publishTopic, publishPayload);
```

Listen to topic and make response:
In the callback function, we listen to the 5 subscribed topics and check if there are messages of "/shadow/get/accepted":

```
if (strstr(topic, "/shadow/get/accepted") != NULL) {
```

If there is, the message is from the control side. If the attribute state in the message is different from current state, publish the new state.

```
updateLedState(desired_led_state);
```

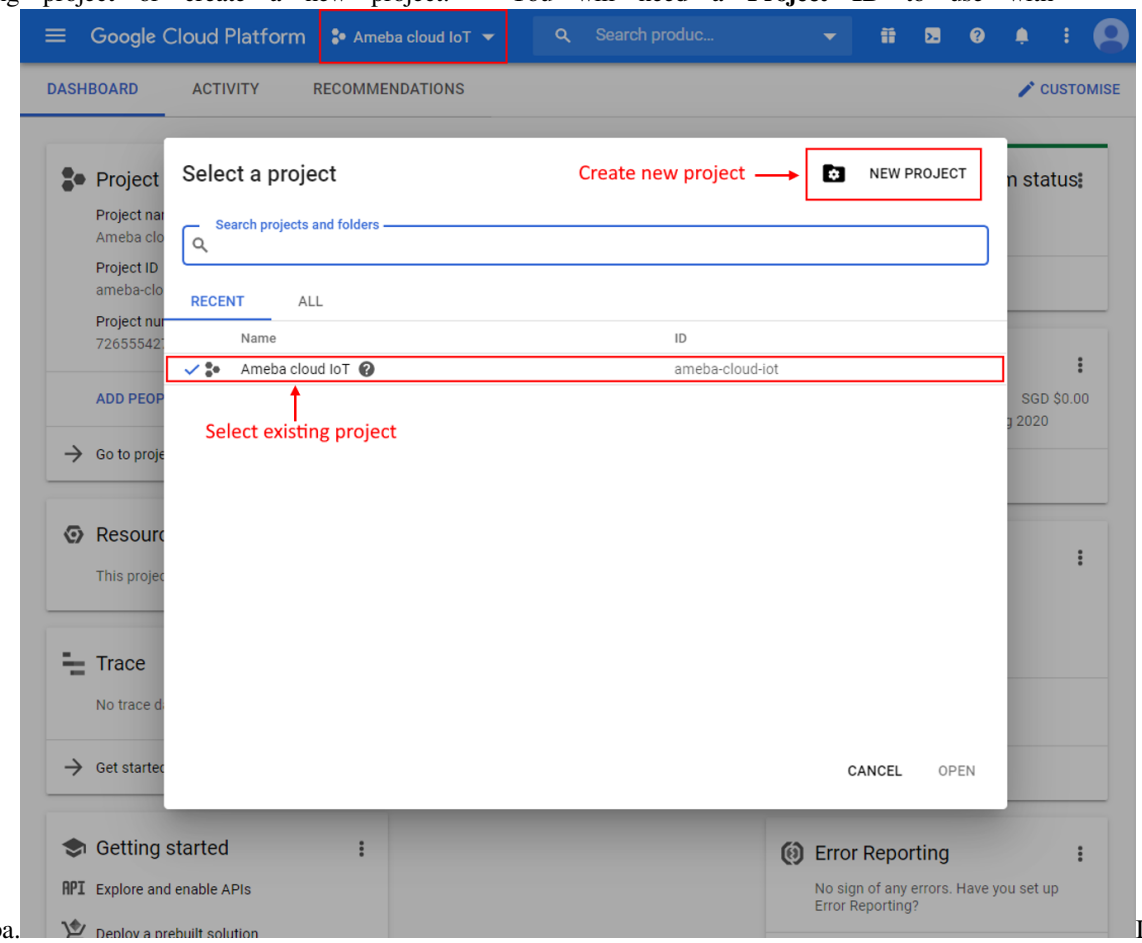
MQTT - Use Google Cloud IoT

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

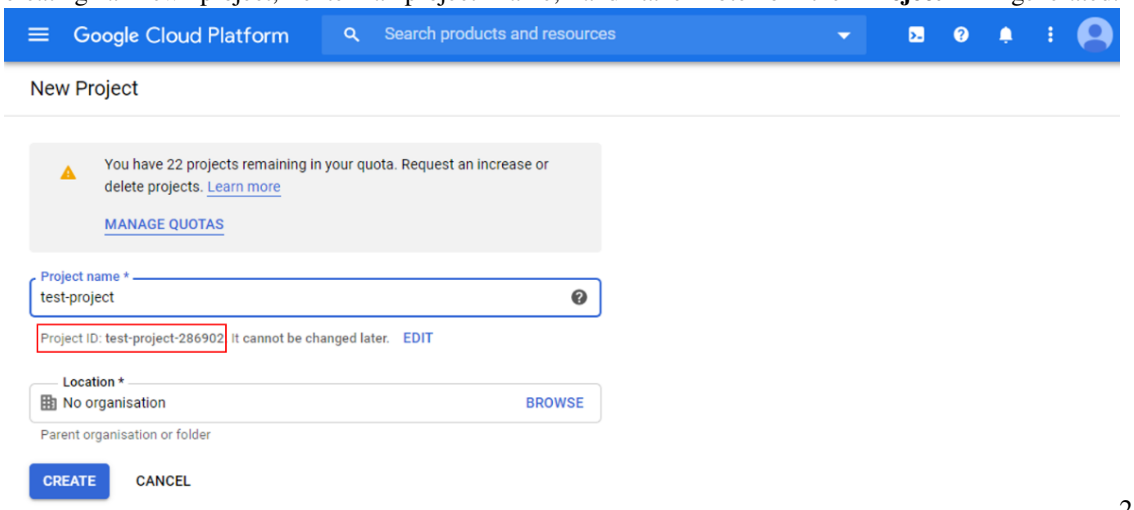
Google Cloud IoT Configuration

1. Select or create a Cloud Platform project In the Google Cloud Console, select an existing project or create a new project. You will need a **Project ID** to use with Ameba.

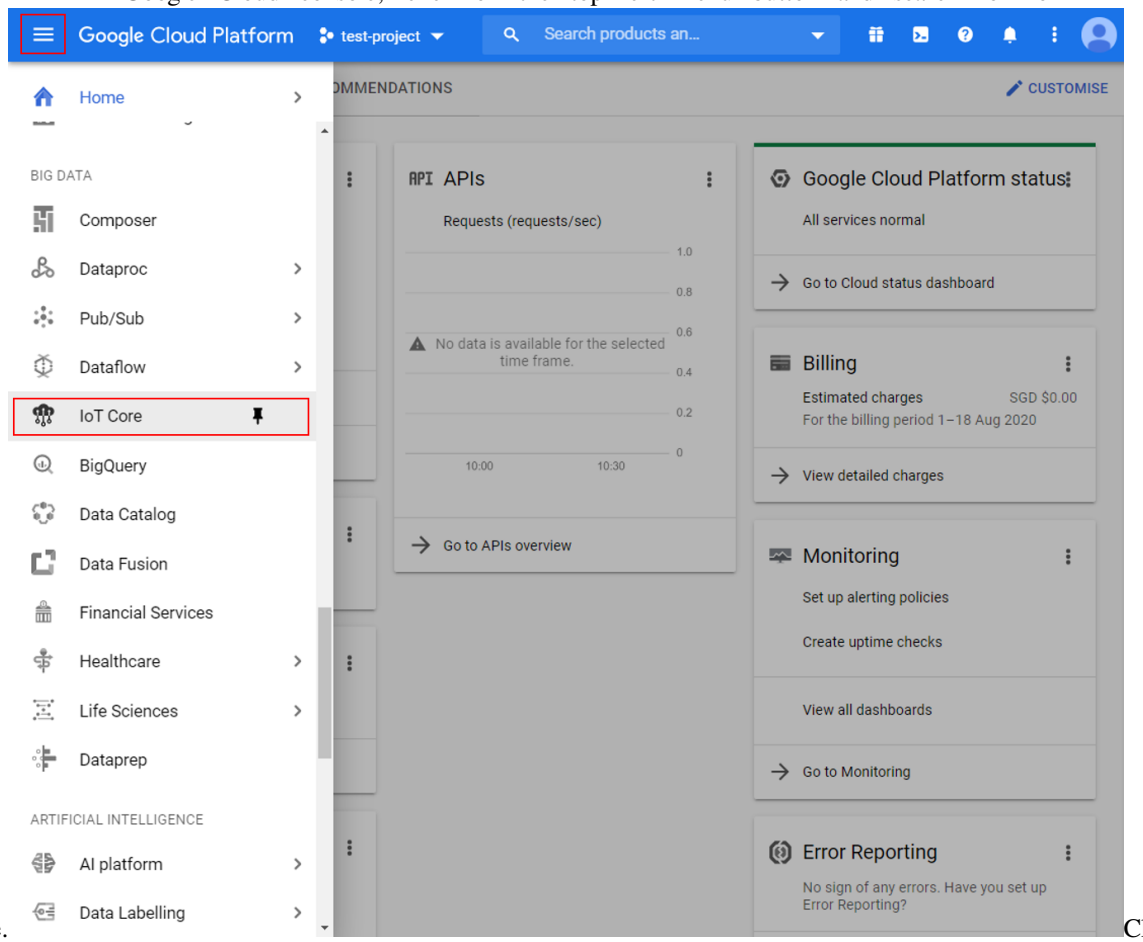


If

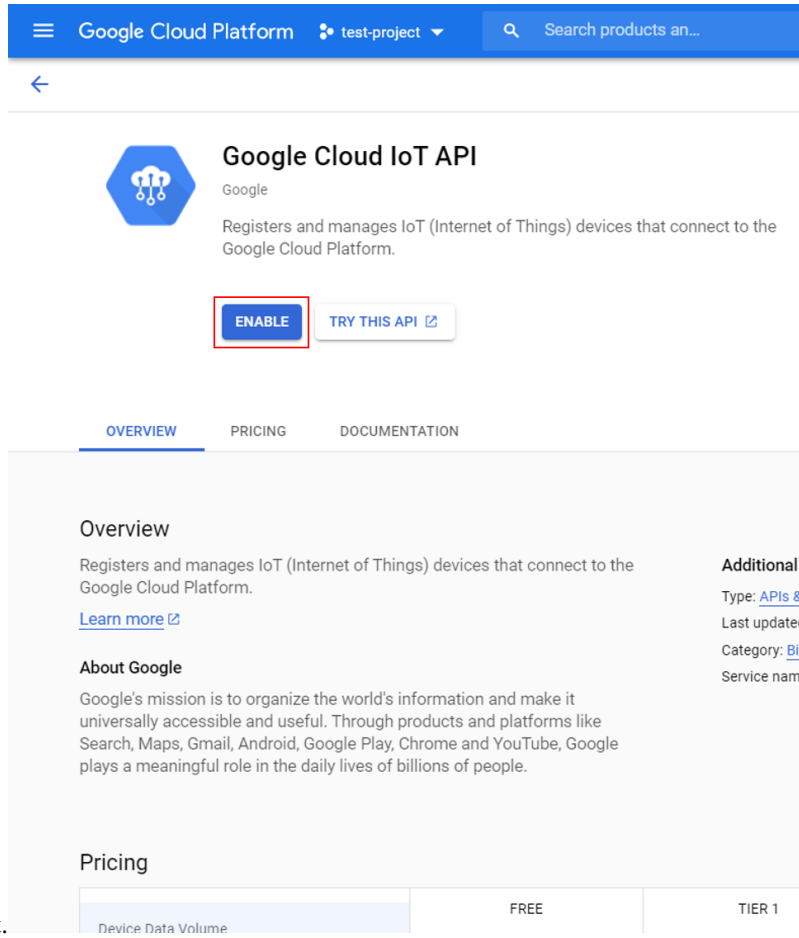
creating a new project, enter a project name, and take note of the **Project ID** generated.



2. Enable billing for your project Billing needs to be enabled for your project to use Google Cloud Platform features. Follow the guide in Google cloud documentation to enable billing. https://cloud.google.com/billing/docs/how-to/modify-project 3. Enable the Cloud IoT Core API In Google Cloud console, click on the top left menu button and search for IoT Core.

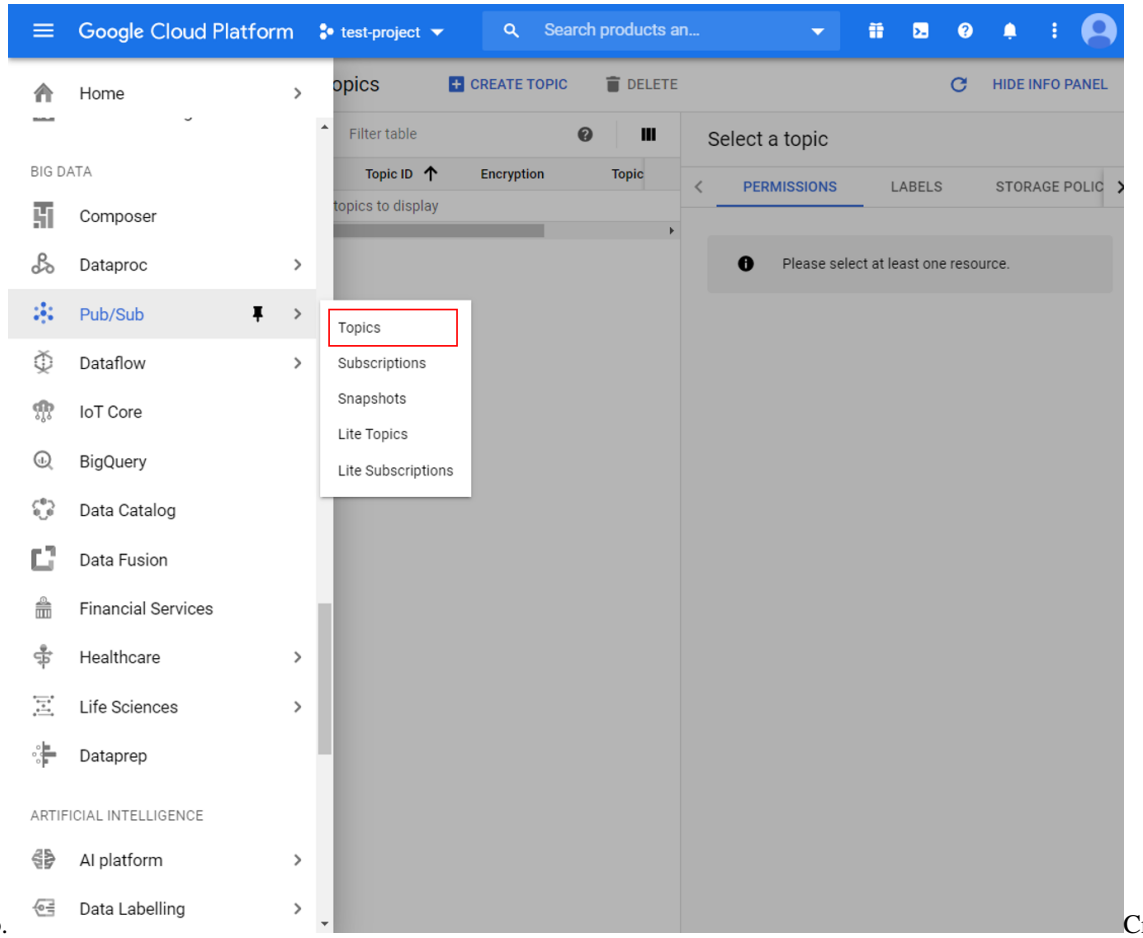


Click

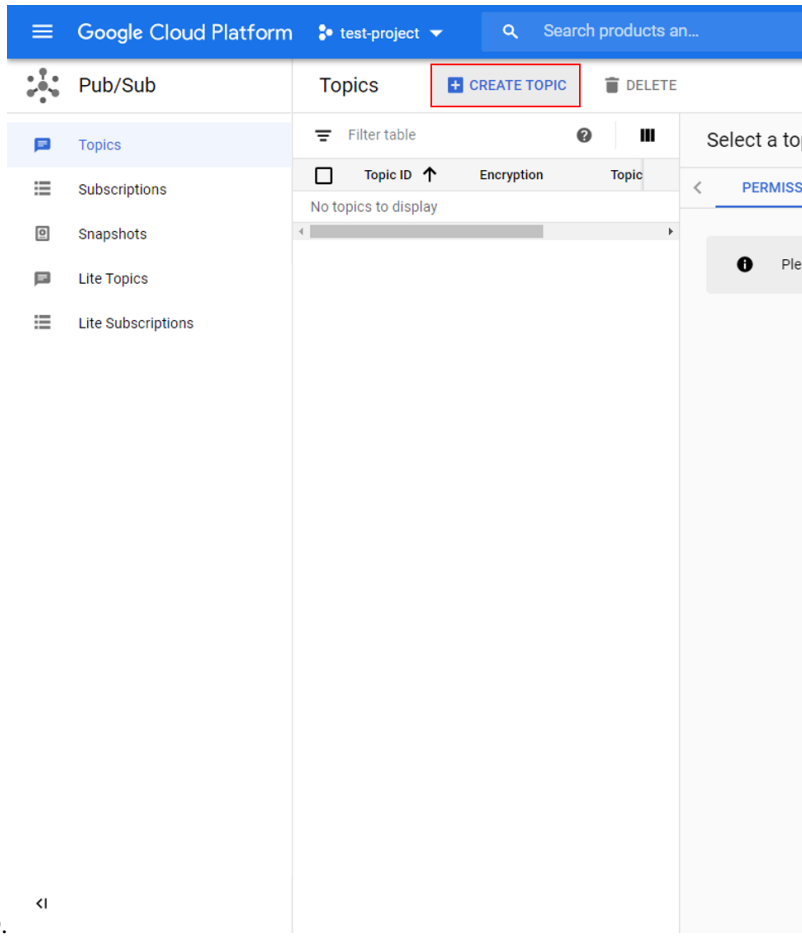


enable to activate Google Cloud IoT API for your project.

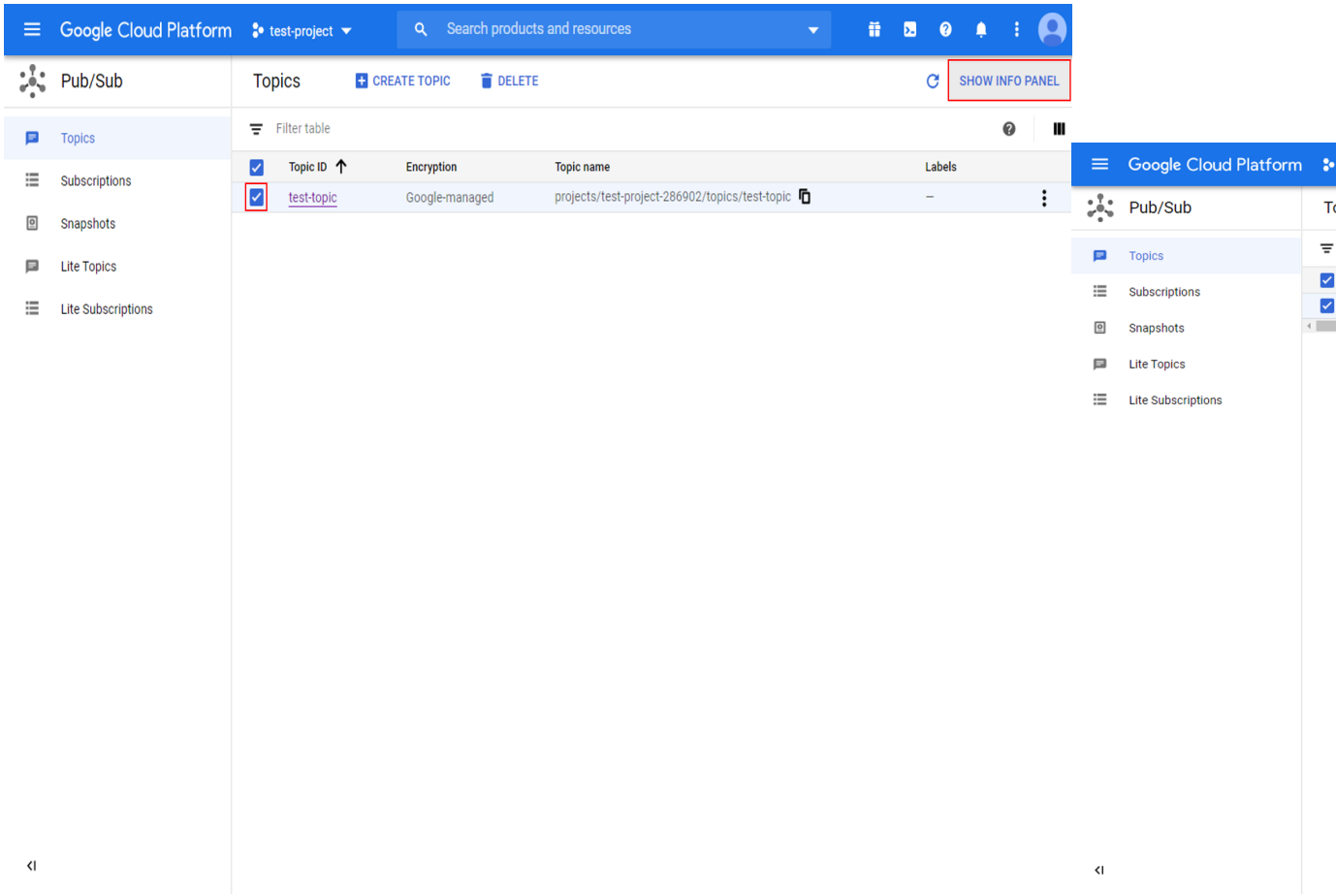
Create a Cloud Pub/Sub topic In Google Cloud console, click on the top left menu button and search for



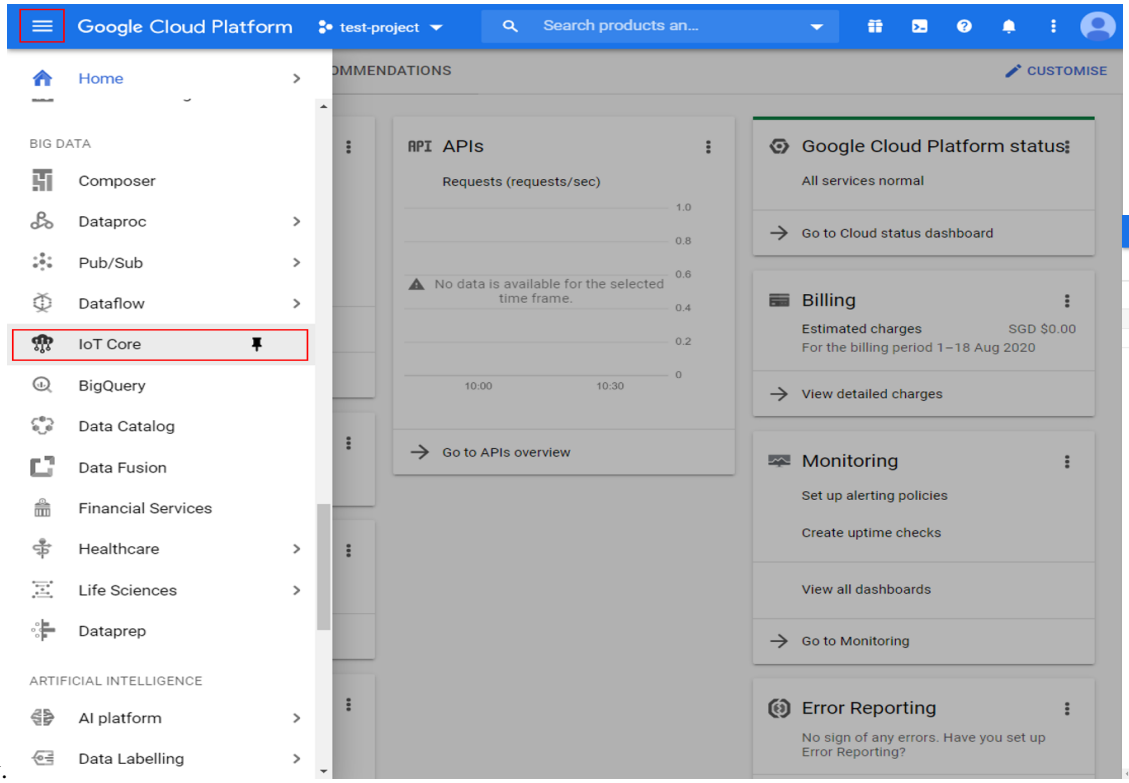
Pub/Sub. Create



a new topic for your project and give it a suitable topic ID. the topic is created, go to the permissions tab of the info panel, and add "cloud-iot@system.gserviceaccount.com" with the role of "Pub/Sub Publisher".

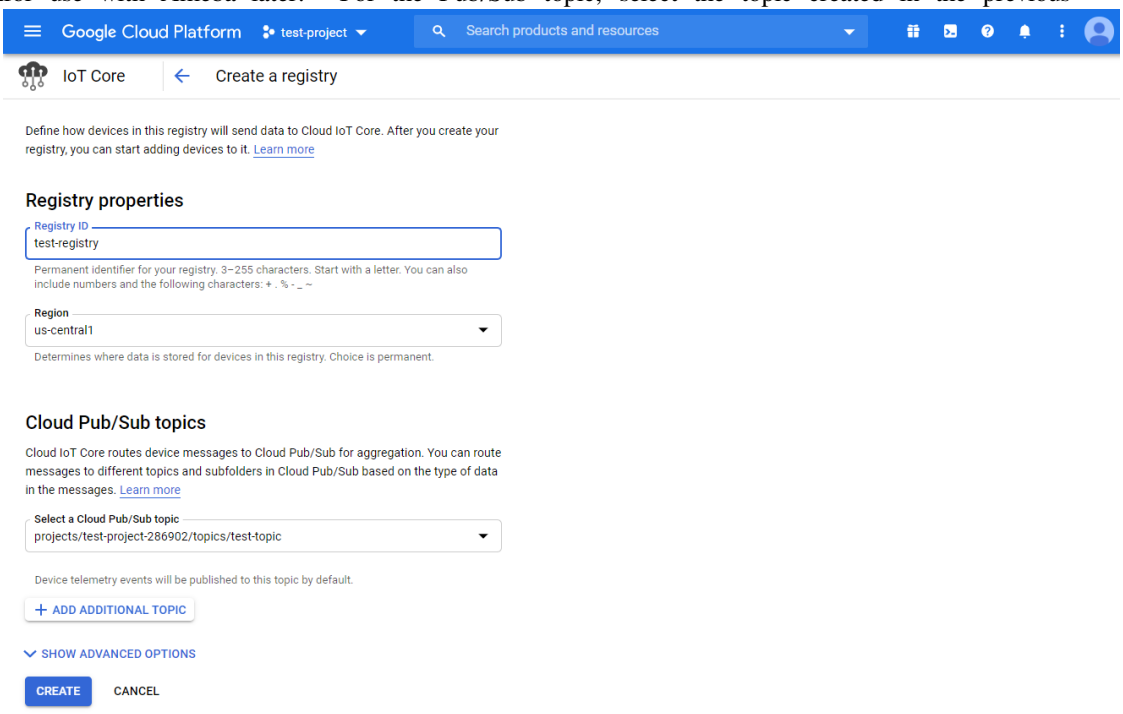


a device registry Go back to the IoT Core settings page and create a new



registry.

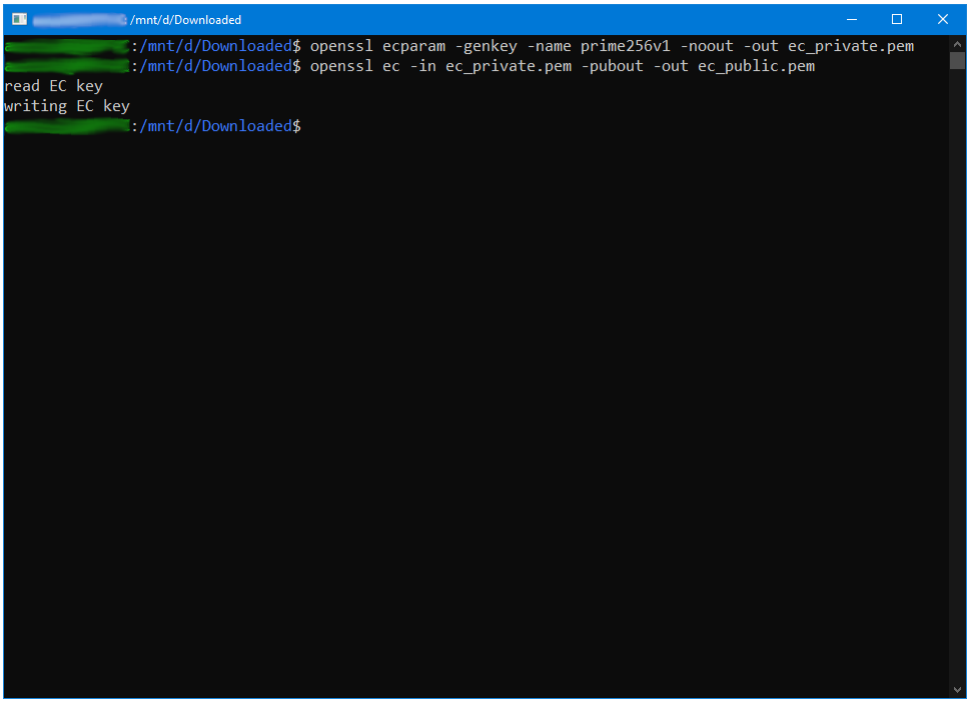
a suitable **Registry ID** and **in which to store data.** **Remember the **Registry ID** and **Region**for use with Ameba later. For the Pub/Sub topic, select the topic created in the previous



step.

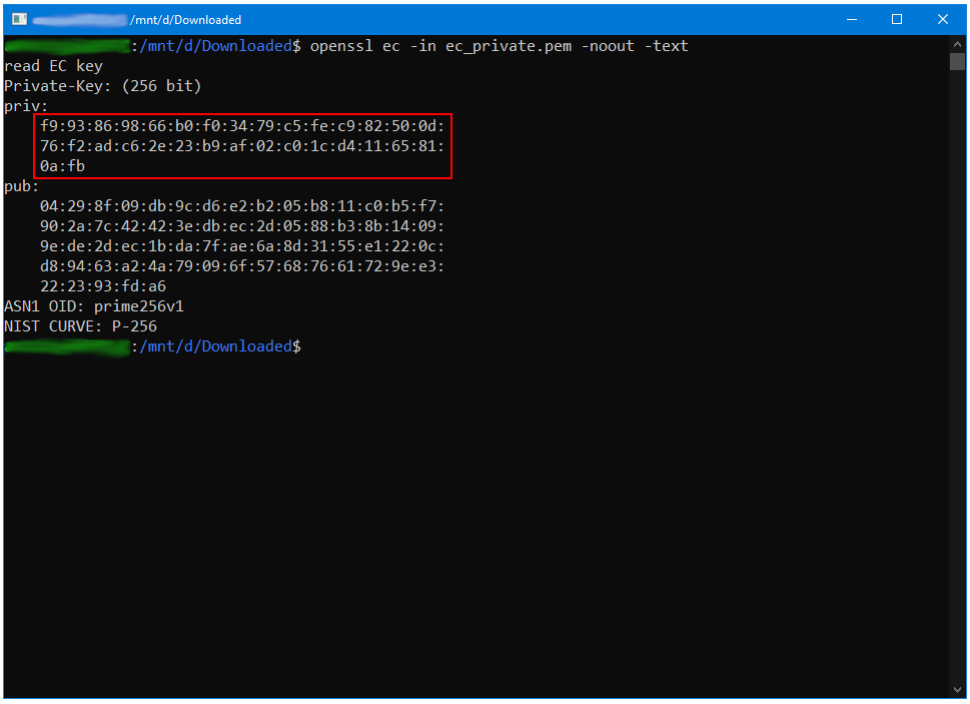
6. Create a public/private key pair Using Openssl in a terminal in Windows/Linux/MacOs, run the following commands to generate a private and public key pair. Two files will be created by these commands, “ec_private.pem” containing the private key, and “ec_public.pem” containing the public key.

```
$ openssl ecparam -genkey -name prime256v1 -noout -out ec_private.pem
$ openssl ec -in ec_private.pem -pubout -out ec_public.pem
```

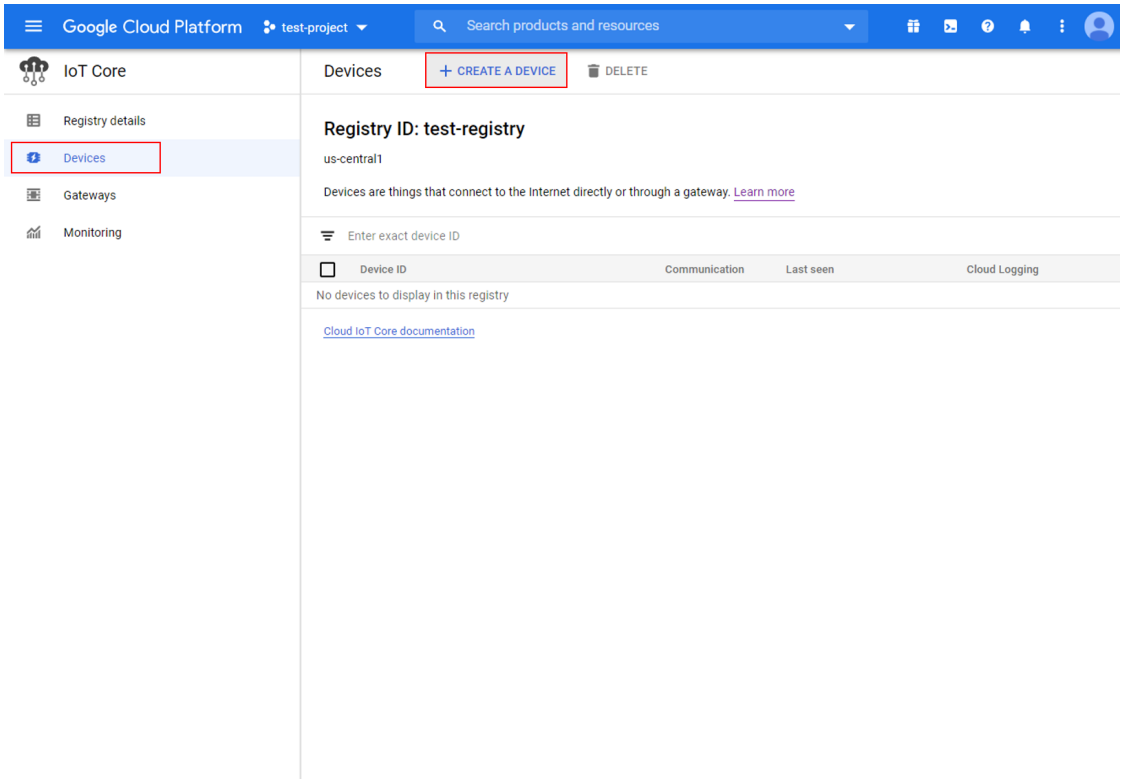


Run the next command to extract out the private key, and remember the highlighted string of hexadecimal numbers for use with Ameba later.

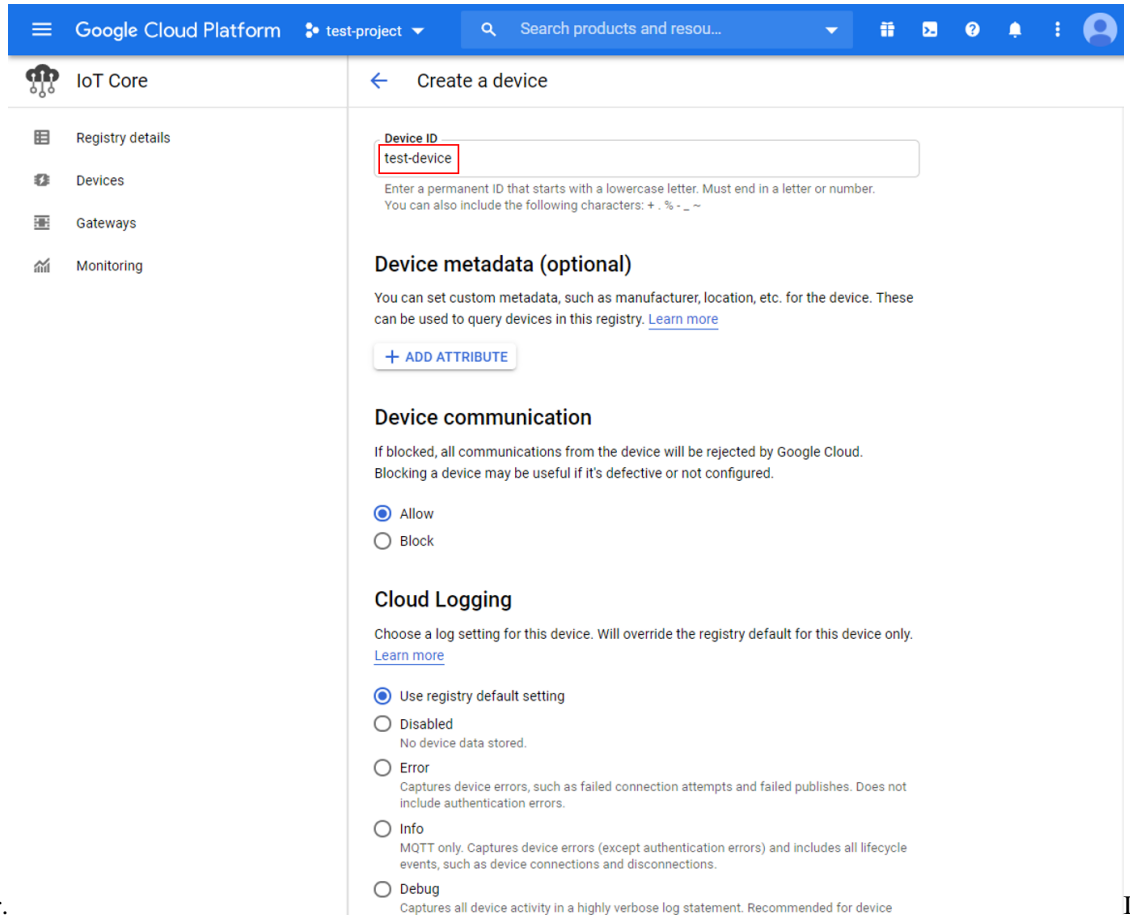
```
$ openssl ec -in ec_private.pem -noout -text
```



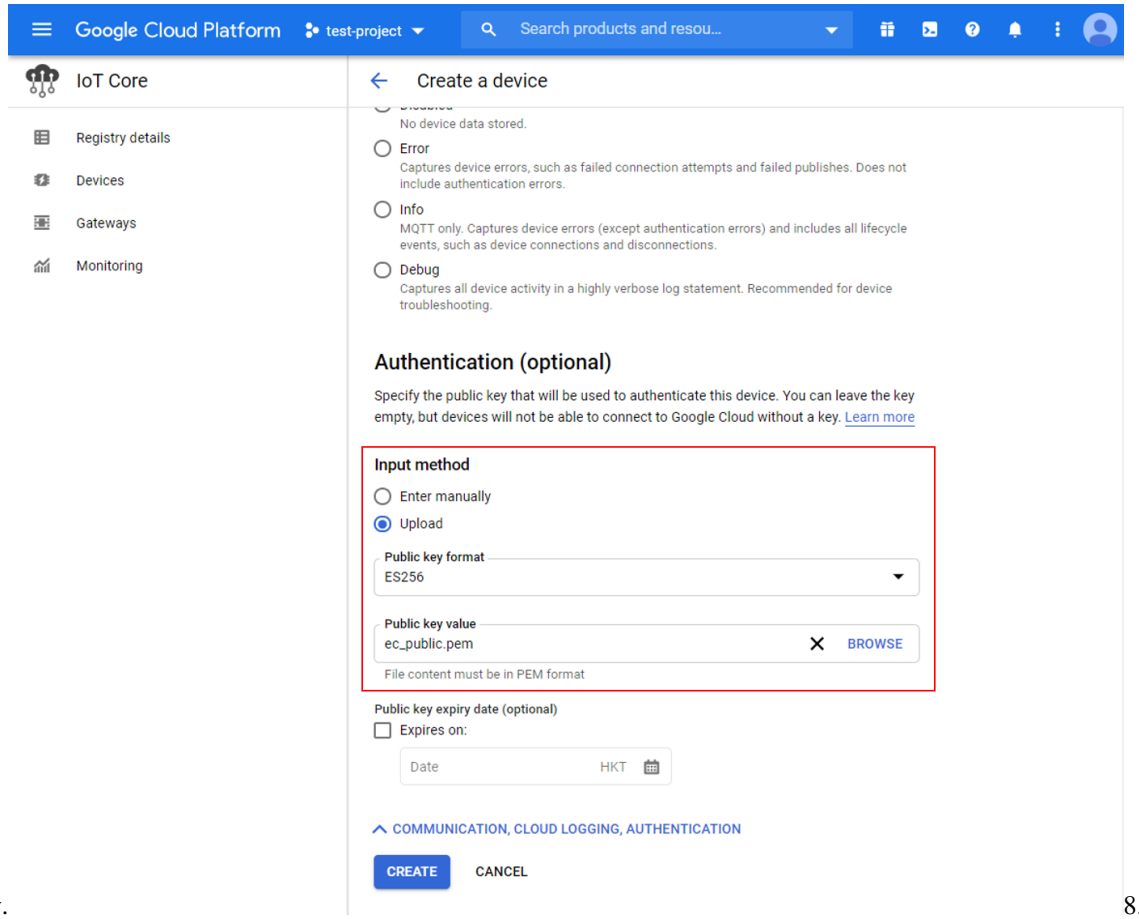
7. Create a device Go back to the IoT Core settings page and create a new device.



Give the device a suitable **Device ID** and remember it for use with Ameba

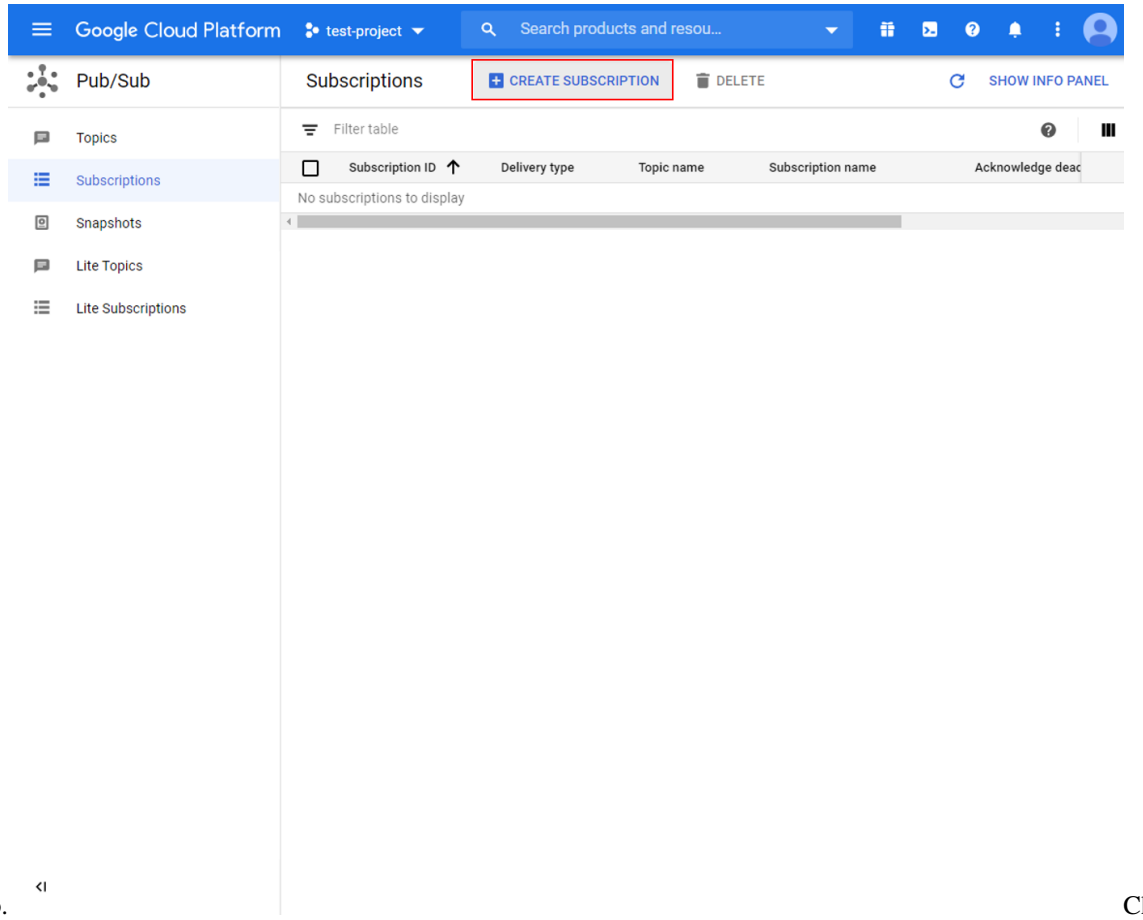


later. In the authentication section of the additional options, upload the previously generated “ec_public.pem” public

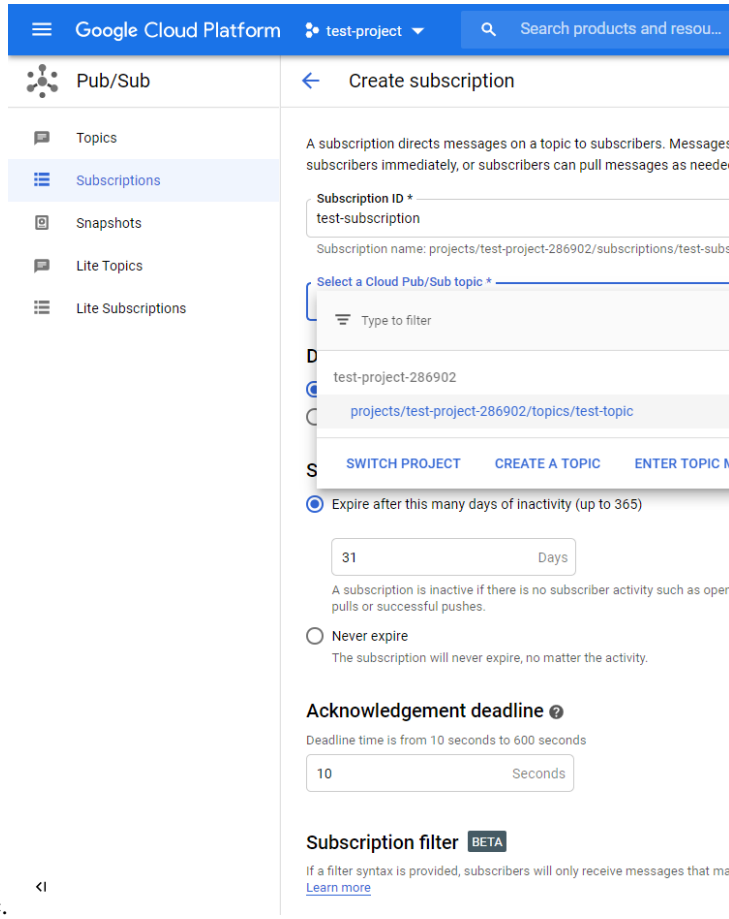


key.

8. Create a Cloud Pub/Sub subscription To observe messages sent by Ameba, create a subscription in



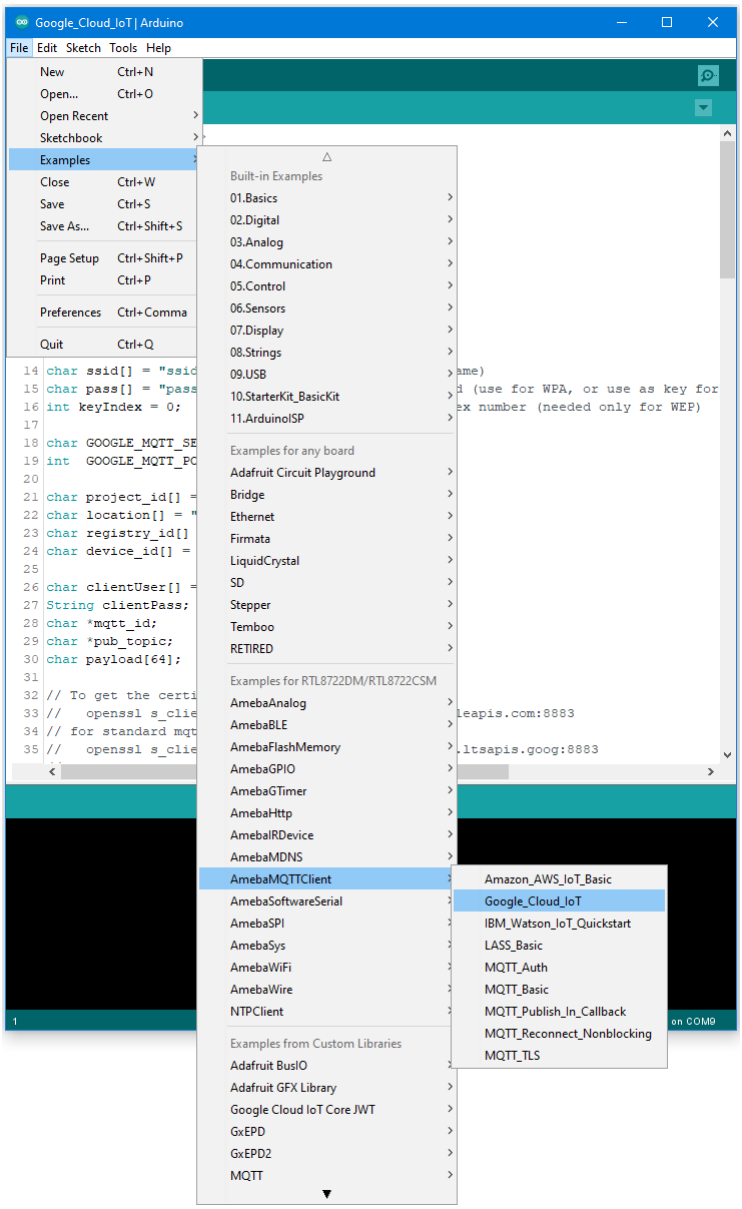
Pub/Sub. Choose



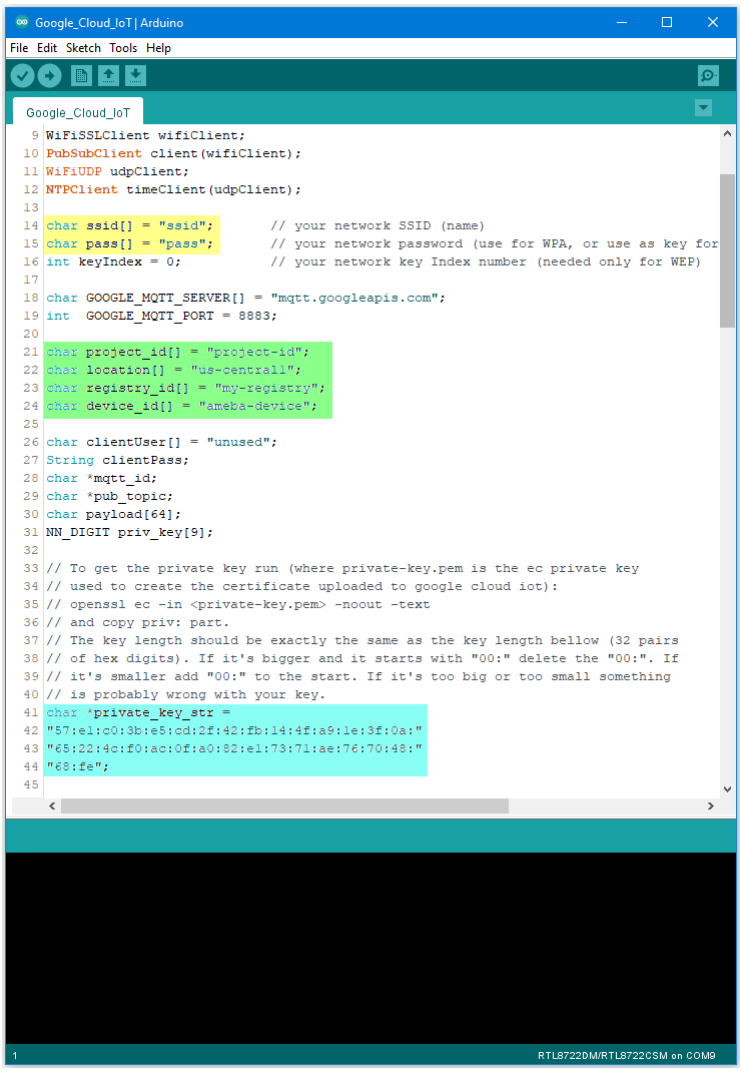
a suitable subscription ID and select the previously created topic.

Example

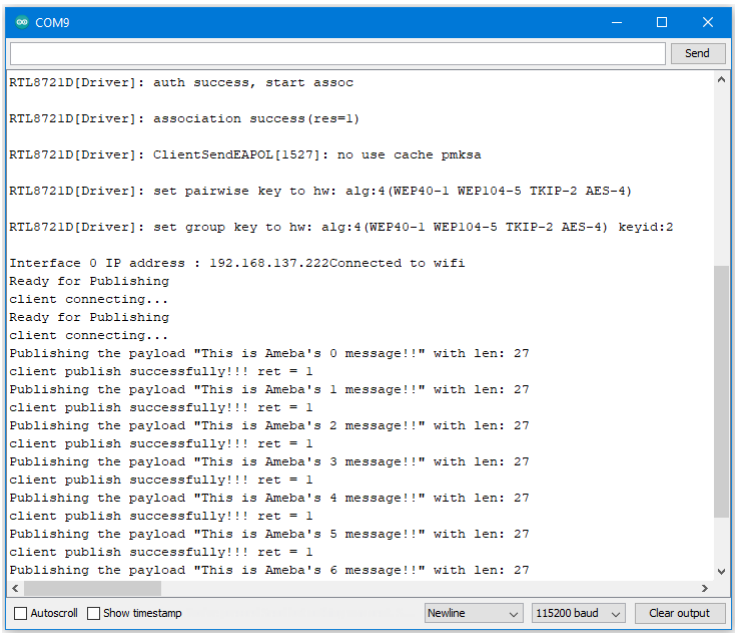
Open the example in "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`Google_Cloud_IoT`".



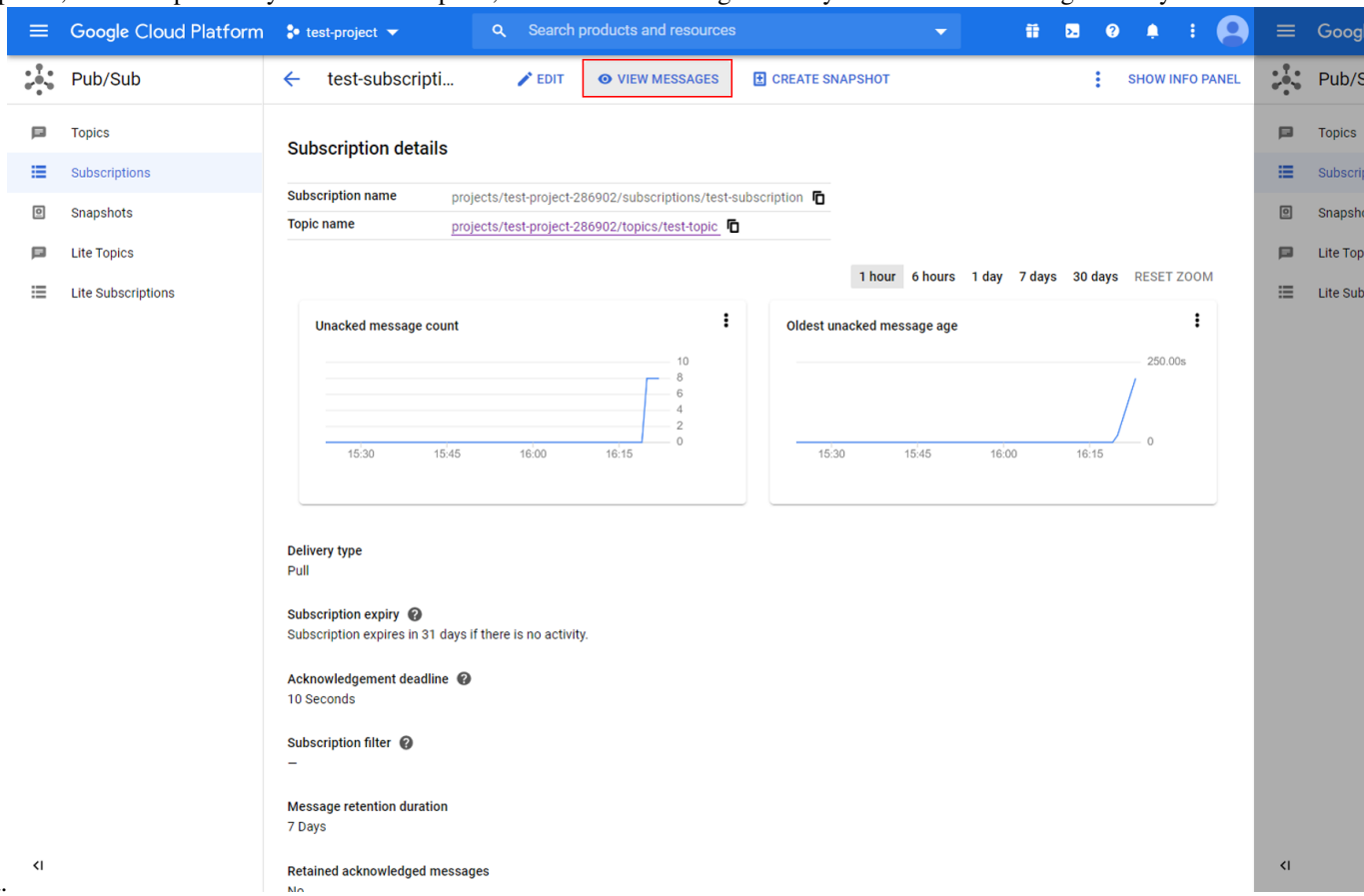
Enter the required information in the highlighted sections below.



In the yellow section, enter the SSID and password required to connect to your WiFi network. In the green section, enter the Project ID, server Region, Registry ID and Device ID previously configured in Google Cloud console. In the blue section, enter the hexadecimal string previously extracted from the private key. Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor and observe as Ameba connects and sends messages to Google Cloud IoT.



In Google Cloud console, go to Pub/Sub subscriptions, select the previously created subscription, and click view messages. Here you can view the messages sent by Ameba.



Code Reference

In `setup()`, we set up RootCA which is required to form a TLS connection with Google's servers.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
```

In `loop()`, each loop checks the Internet status and re-connect to it when the environment has a problem.

```
if (WiFi.status() != WL_CONNECTED) {
   while (WiFi.begin(ssid, pass) != WL_CONNECTED)
   {
      delay(1000);
   }
   Serial.println("Connected to wifi");
}
```

To publish messages, mqtt_id , clientPass and pub_topic are required. mqtt_id is generated by printing the project ID, server location, registry ID and device ID in the required format:

```
mqtt_id = (char *)malloc(strlen("projects/") + strlen(project_id) + strlen("/
↪locations/us-central1/registries/") + strlen(registry_id) + strlen("/devices/") +␣
↪strlen(device_id) + 1);
sprintf(mqtt_id, "projects/%s/locations/us-central1/registries/%s/devices/%s",␣
↪project_id, registry_id, device_id);
```

`clientPass` is generated using a JSON web token (JWT) generator function, which requires the project ID and current time, and signs it with the private key:

```
clientPass = CreateJwt(project_id, timeClient.getEpochTime(), priv_key);
```

`pub_topic` is generated by printing the project ID and topic in the required format:

```
pub_topic = (char *)malloc(strlen("/devices/") + strlen(device_id) + strlen("/events
↪") + 1);
sprintf(pub_topic, "/devices/%s/events", device_id);
```

MQTT Server setting:

```
client.setServer(GOOGLE_MQTT_SERVER, GOOGLE_MQTT_PORT);
client.setPublishQos(MQTTQOS1);
client.waitForAck(true);
```

Connect to google cloud and publish messages:

```
if (client.connect(mqtt_id, clientUser, clientPass.c_str())){
   // ...
        for(int i = 0; i < count; i++){
      // ...
      sprintf(payload, "This is Ameba's %d message!!", i);
      ret = client.publish(pub_topic, payload);
      // ...
   }
   // ...
   client.disconnect();
}
free(mqtt_id);
free(pub_topic);
```

MQTT - Upload PM2.5 Data to LASS System

Intro to LASS

The LASS stands for "Location Aware Sensor System". It is an open project and was started only for the interest of public welfare. Find detailed introduction here.
Practically, LASS is based on MQTT protocol to collect all kinds of uploaded data, and for those who need these data can subscribe top as well.
Find more LASS information at their official hackpad.

Preparation

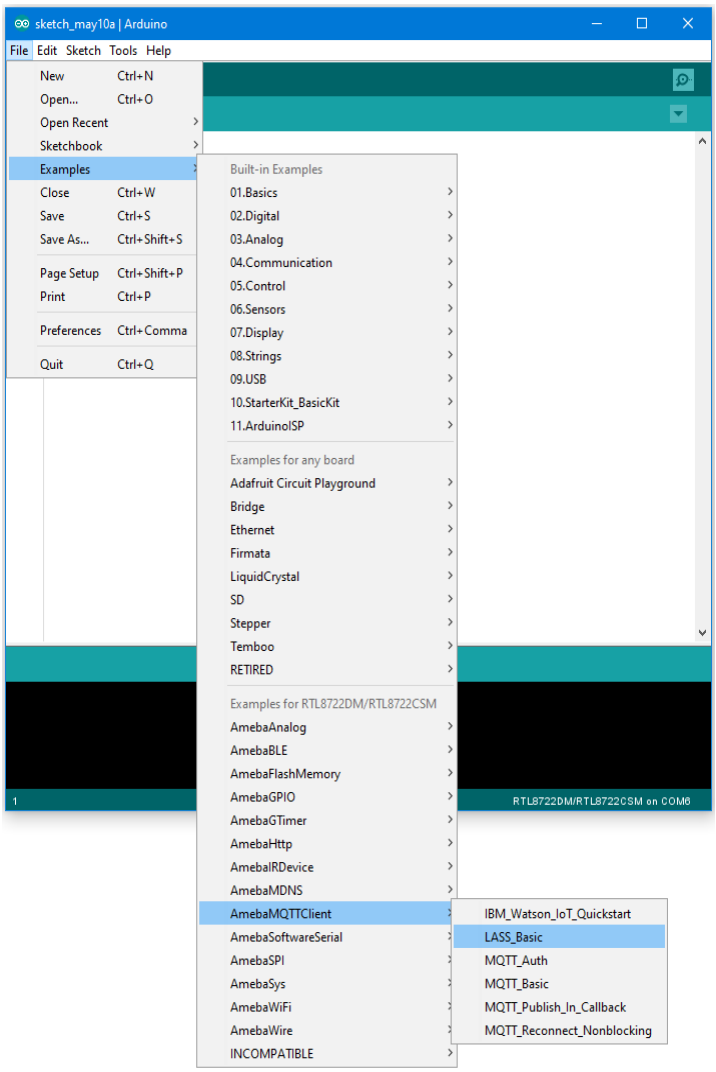
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x1

Example

In this example, we use applications mentioned at our website, including:

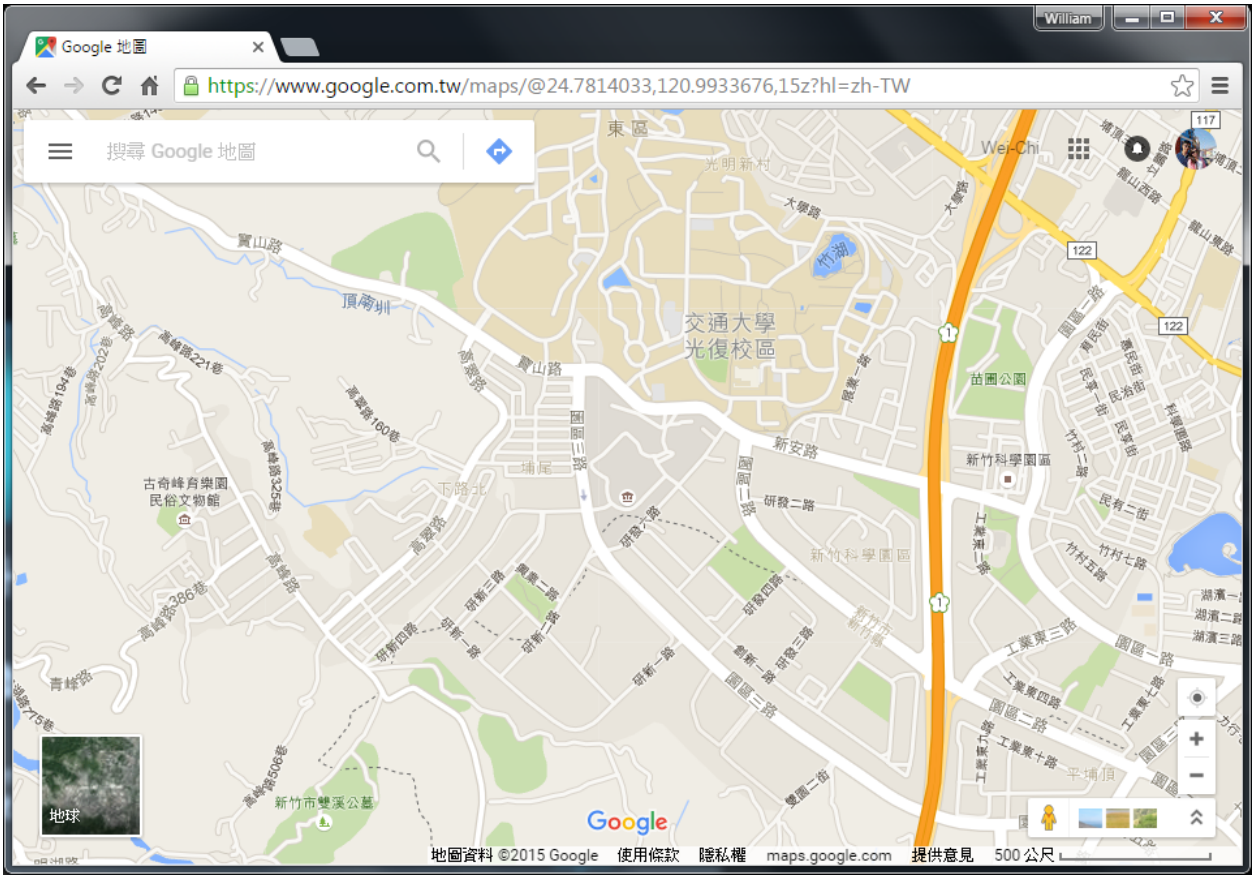
- MQTT: a MQTT-Broker to connect to LASS. The Client is "FT1_0XXXX", the XXXX are the four last digits of Ameba's Wi-Fi MAC, and the outTopic is "LASS/Test/Pm25Ameba/**clientID**", where clientID is the actual Ameba's MQTT client ID.
- NTP: uploaded data must have time notation
- PM2.5: uploaded data includes PM2.5 information

Open the example. "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`lass_basic`"

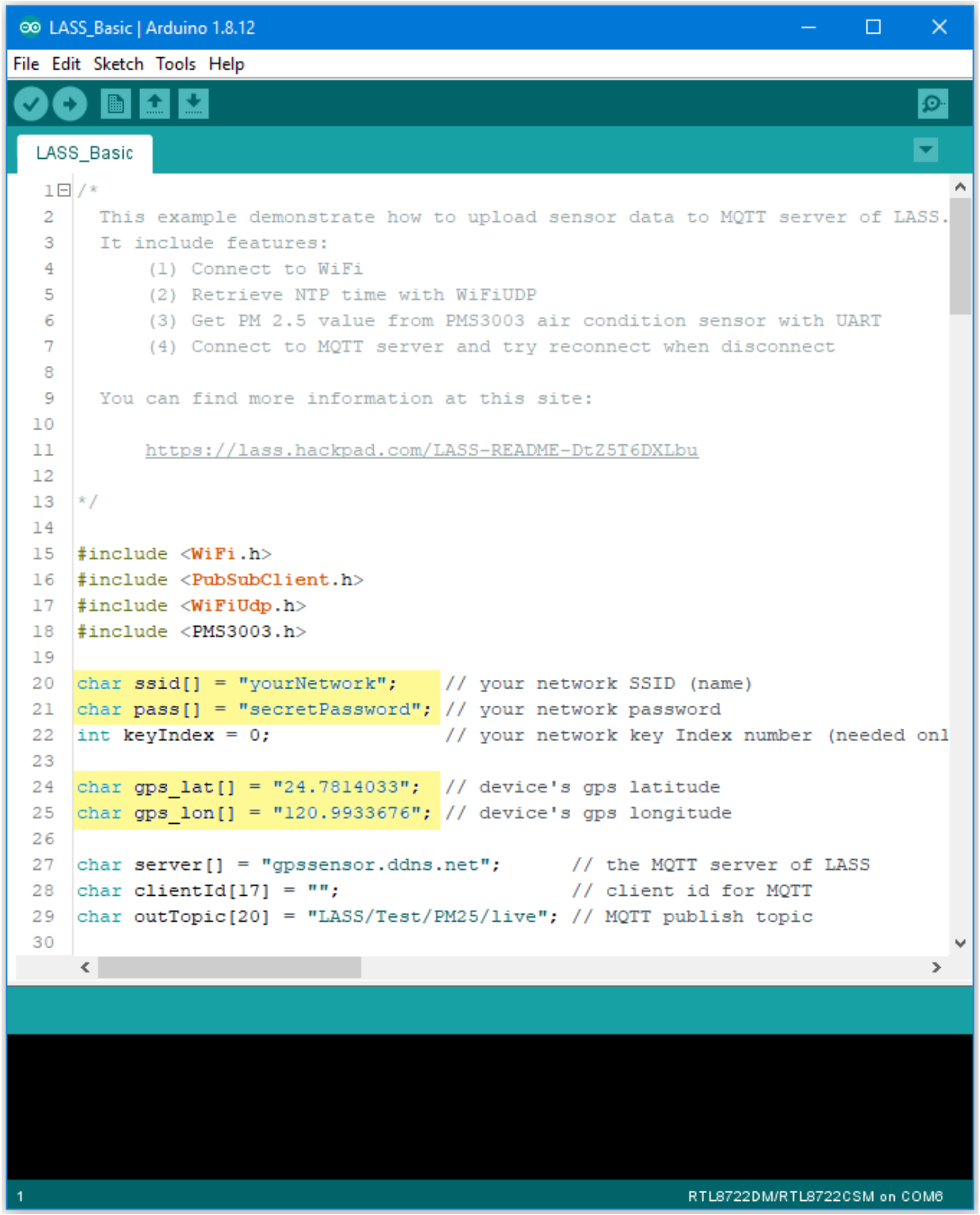


This example requires internet connection, so make sure you fill in SSID and PASS into AP information that you wish to connect.

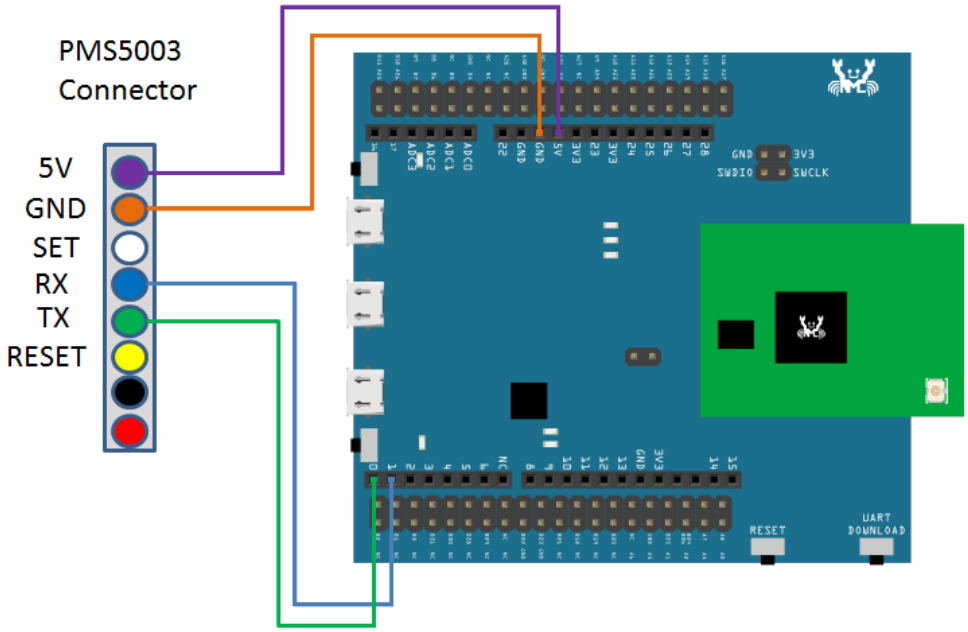
Also, LASS requires GPS information. There is no GPS sensor included in this example, so you must manually provide GPS information. Use Google Map to find the coordinates you plan to place your Ameba. You can see in this example that the latitude is 24.7814033, and the longitude is 120.9933676



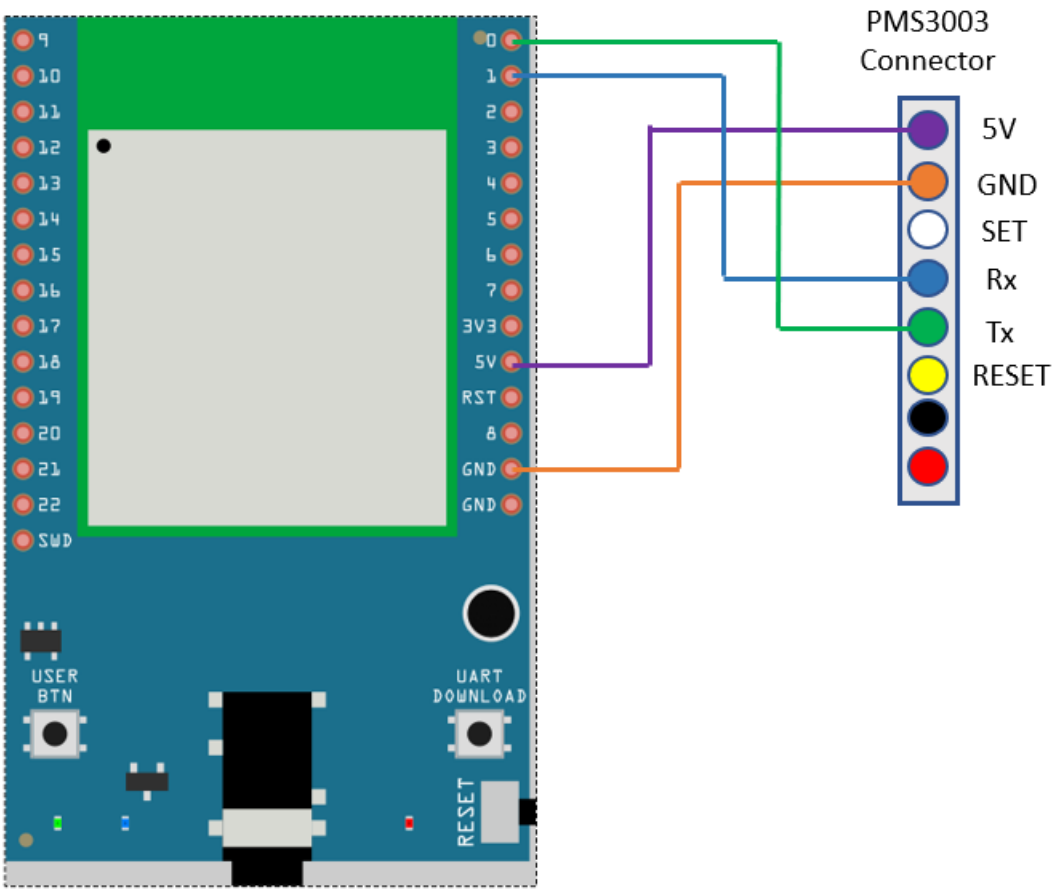
Fill in GPS info at `gps_lat` and `gps_lon`.



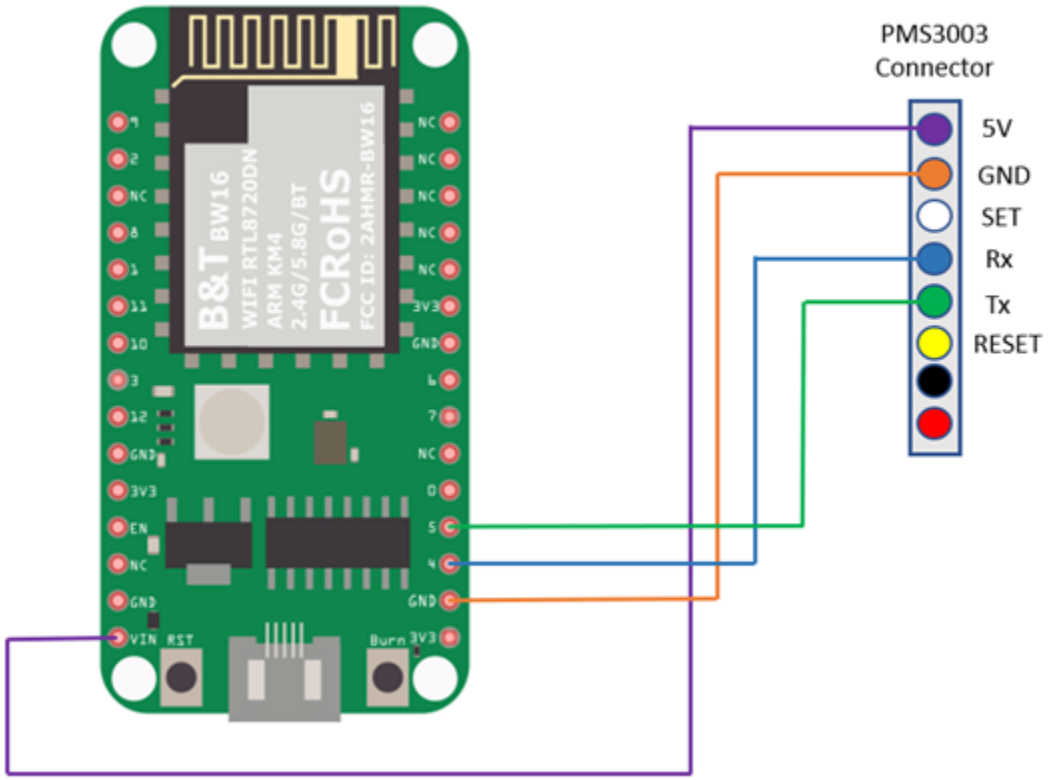
Then connect sensors according to UART-PlanTower PMS3003 wiring example.
AMB21 / AMB22:


PMS5003
Connector
5V
GND
SET
RX
TX
RESET
RESET
UART
DOWNLOAD

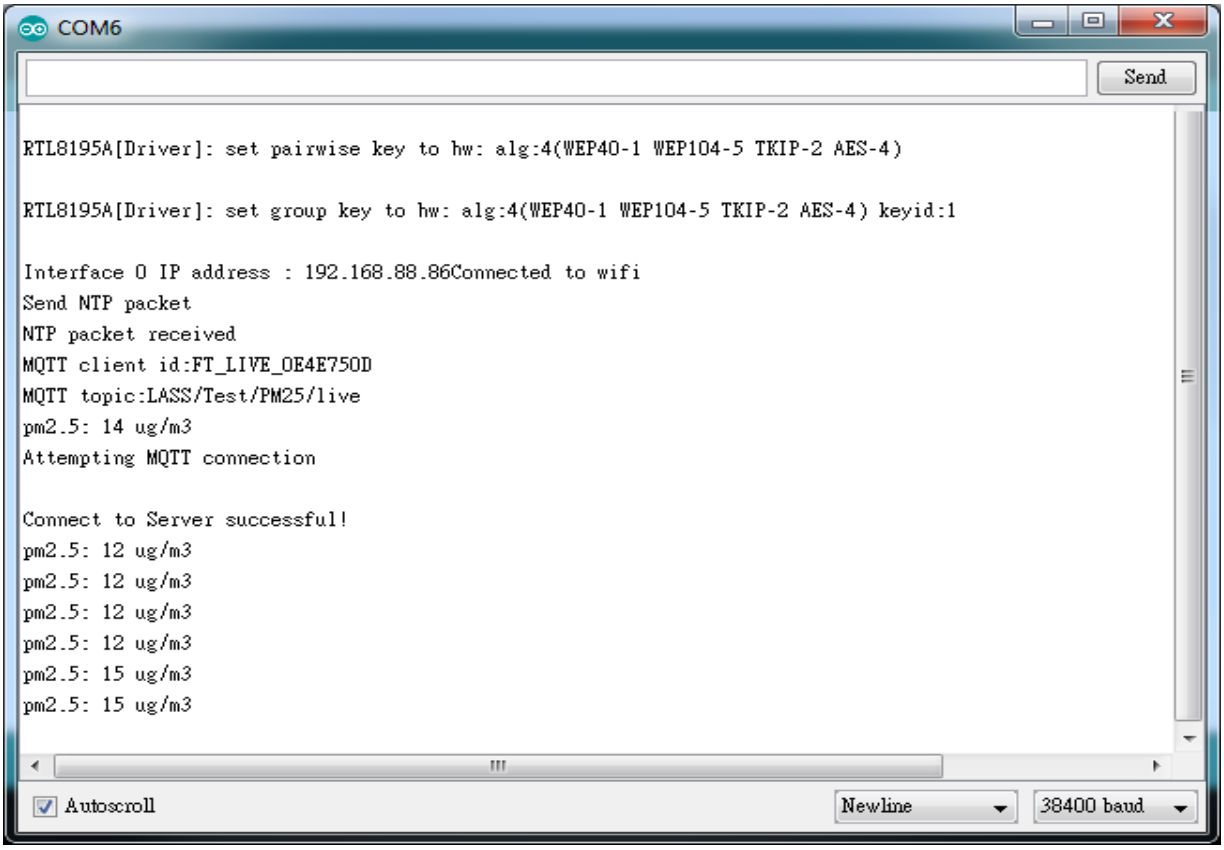
AMB23:


PMS3003
Connector
5V
GND
SET
Rx
Tx
RESET
9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

BW16:



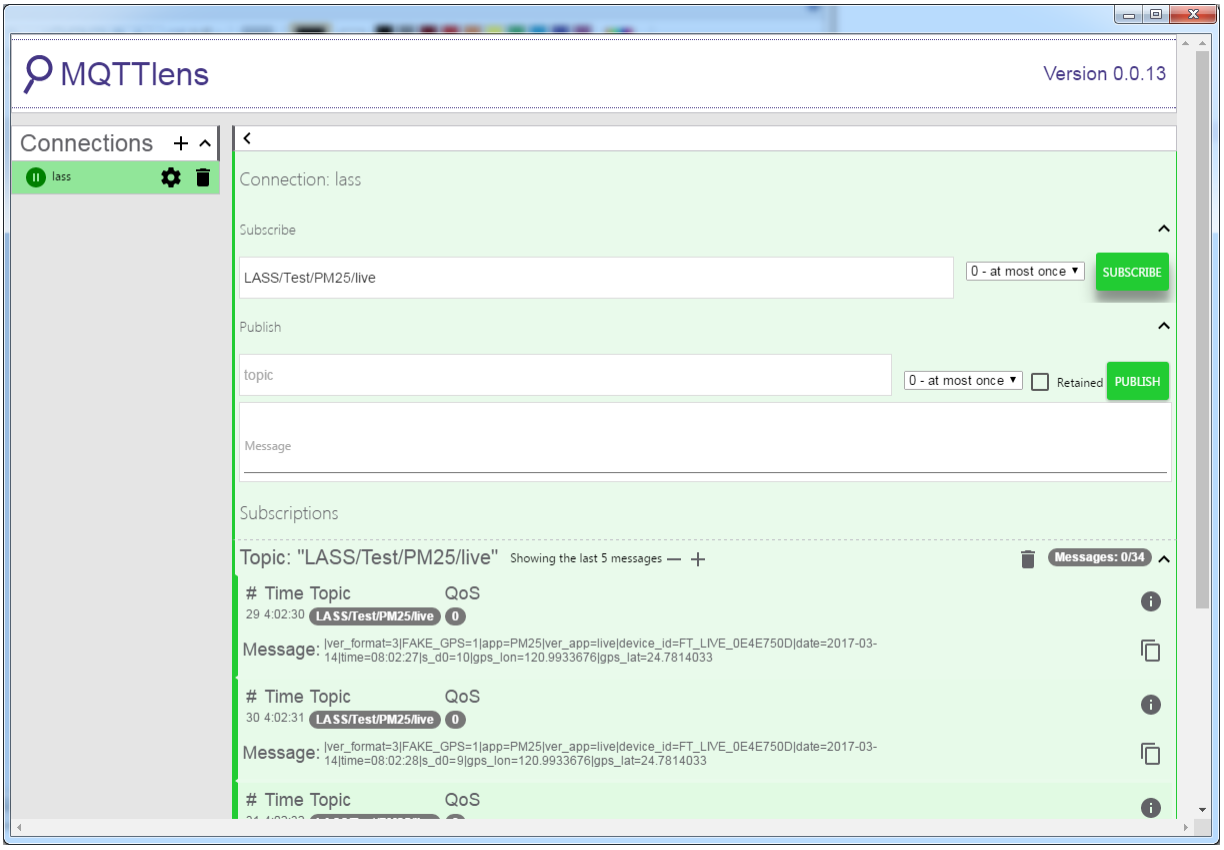
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba will attempt to read PM2.5 data every minute and upload it to LASS MQTT-Broker. Open Serial Monitor to see the uploaded data, including client id, topic, and current PM2.5 status.



We can also use MQTTlens to verify if the data is properly uploaded.

Enter “gpssensor.ddns.net” as the MQTT-Broker server and “LASS/Test/PM25/live” as the subscribe topic to receive data.

The time uses UTC format, and the PM2.5 data stores in s-d0. In the figure, s_d0 = 9 represents that the PM2.5 is 9, meaning that the entire publish/subscribe process is working successfully.



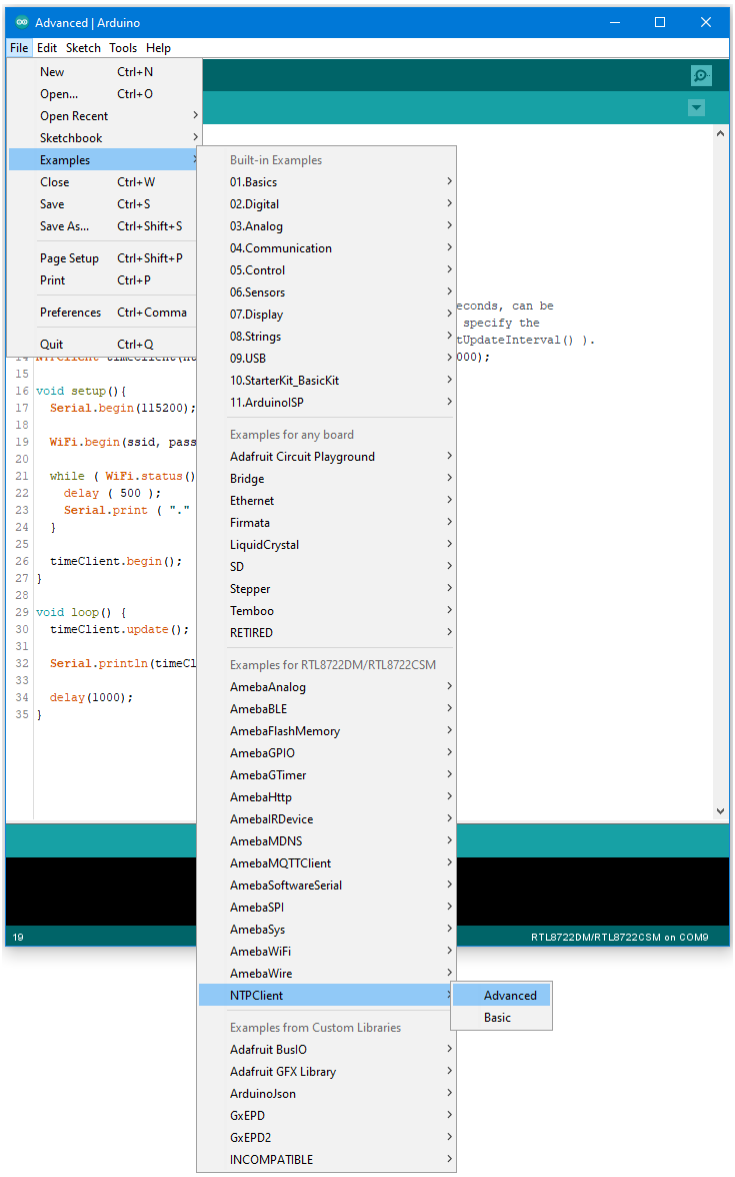
NTP - Retrieve Universal Time (UTC) by NTPClient library

Preparation

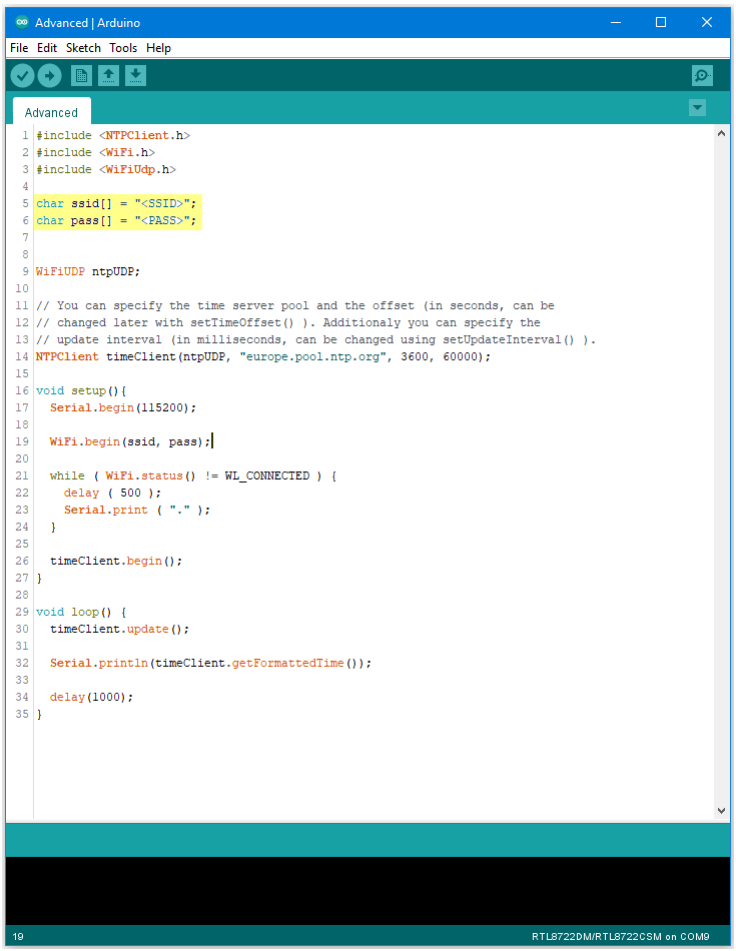
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

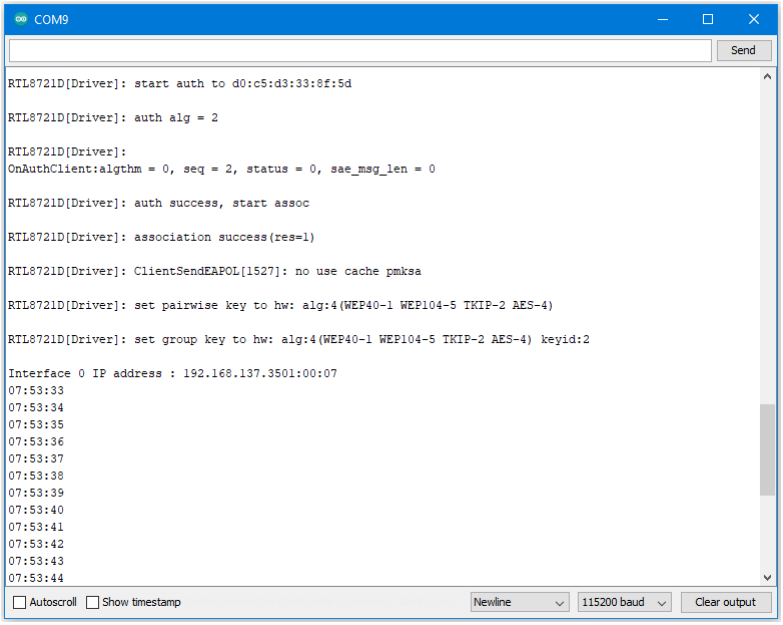
In this example, we use an NTP client to sync with NTP servers using UDP and keep track of time locally. Open the example. “`File” -> “Examples”-> “NTPClient” -> “Advanced`”



Modify the highlighted code section (ssid, password) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi, gets the UTC time from the NTP server, and prints out the current time with time zone offset to the serial monitor.



Code Reference

Configure NTP client:

The NTPClient needs to use a UDP client for communications. A WiFiUDP client is declared and passed to the NTPClient constructor, along with an NTP server address, time zone offset in seconds, and update interval in milliseconds. If detailed configuration is not needed, just passing in the UDP client is also sufficient, refer to the "NTPClient" -> "Basic" example.

```
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 3600, 60000);
```

Start NTP client:

After connecting to WiFi, the NTPClient is started using the `begin()` function, which causes the client to sync with the NTP server and get the UTC time.

```
WiFiUDP ntpUDP;
timeClient.begin();
```

Get local time:

`getFormattedTime()` is used to format the received UTC time into the local time zone. `update()` is called every loop so that the NTPClient will sync with the NTP server once every update interval.

```
timeClient.update();
timeClient.getFormattedTime();
```

WiFi - Approximate UDP Receive Delay

Materials

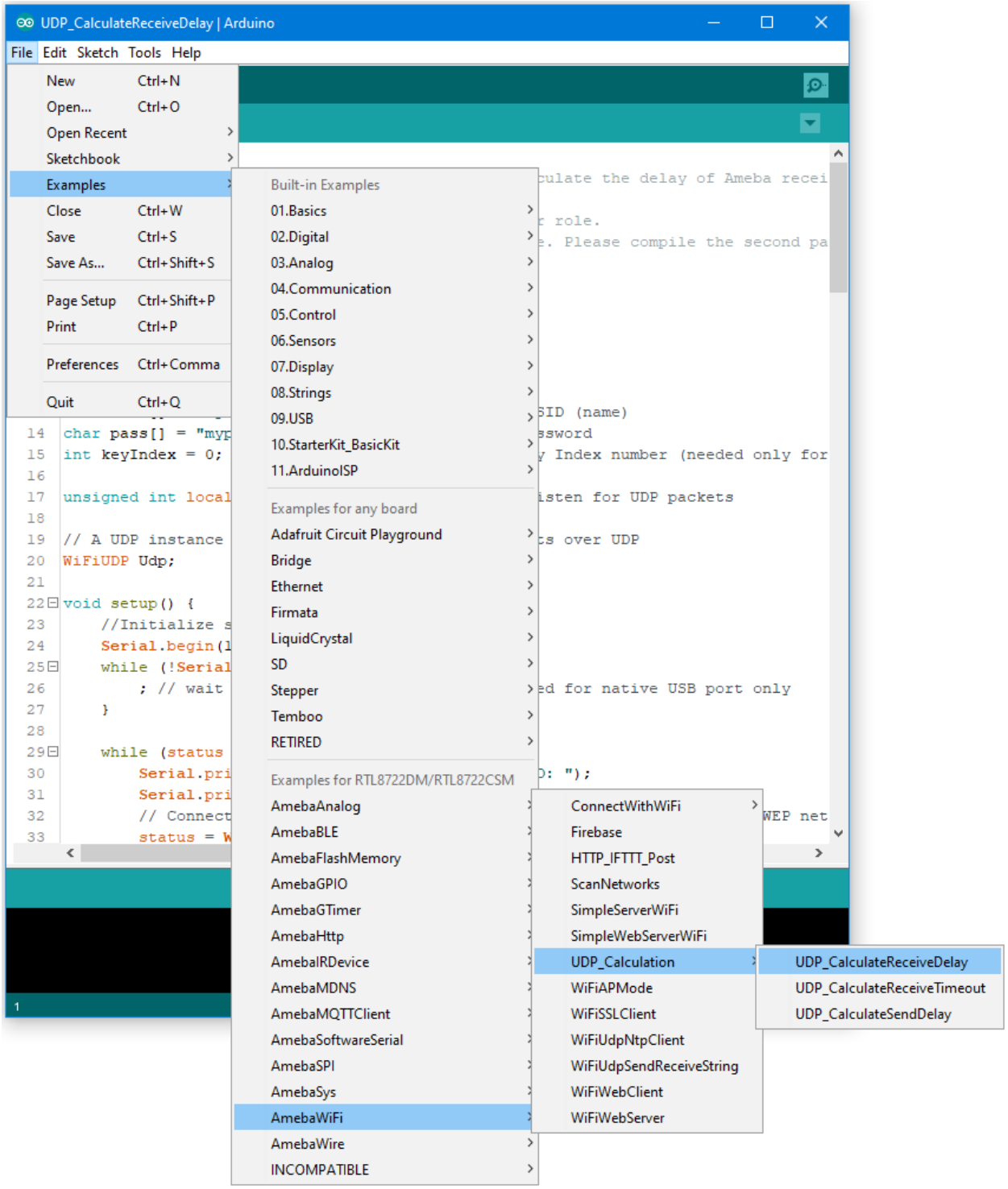
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

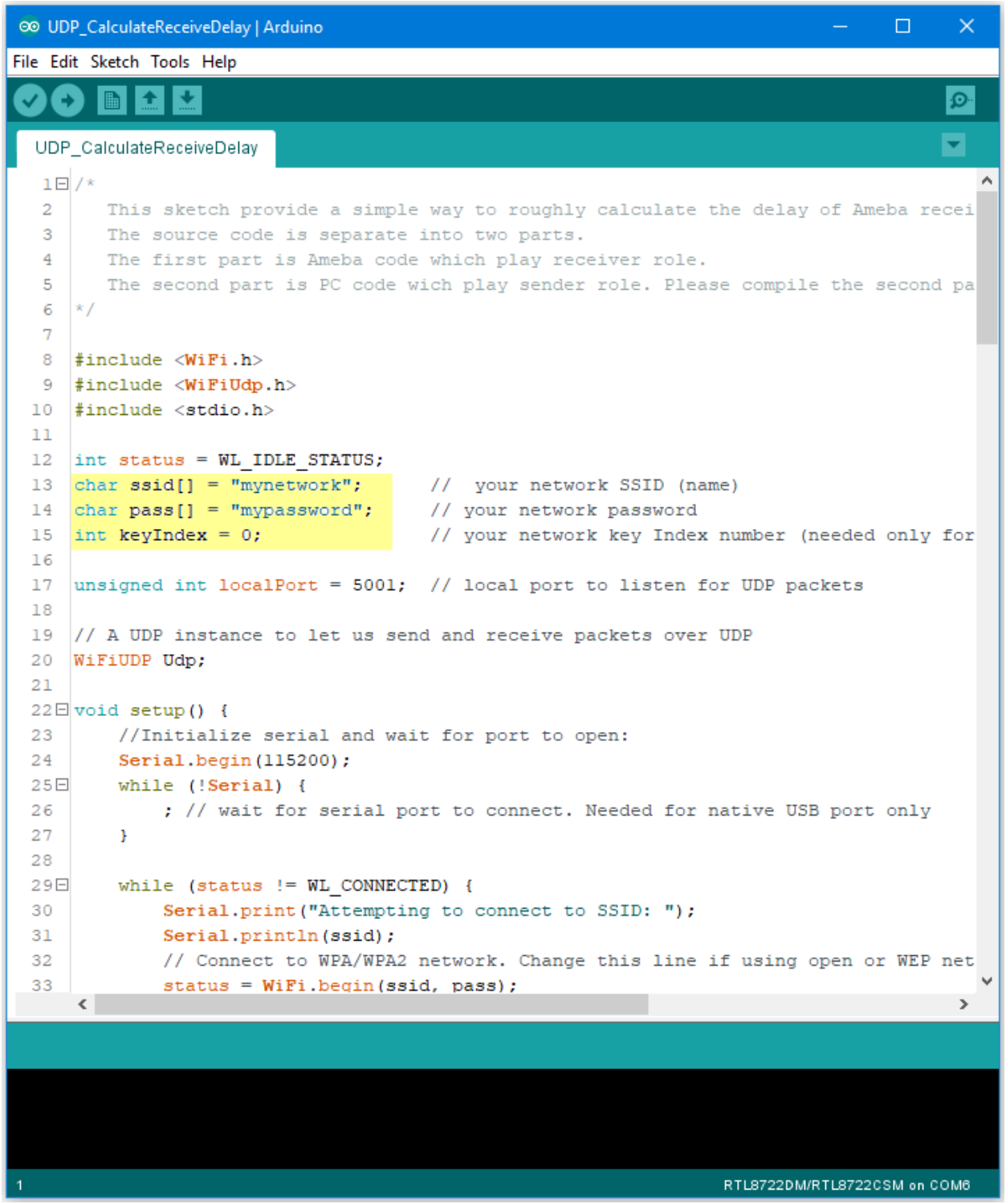
This example uses Ameba to receive UDP packets from a computer and calculates the UDP receive delay.

Ameba Preparation

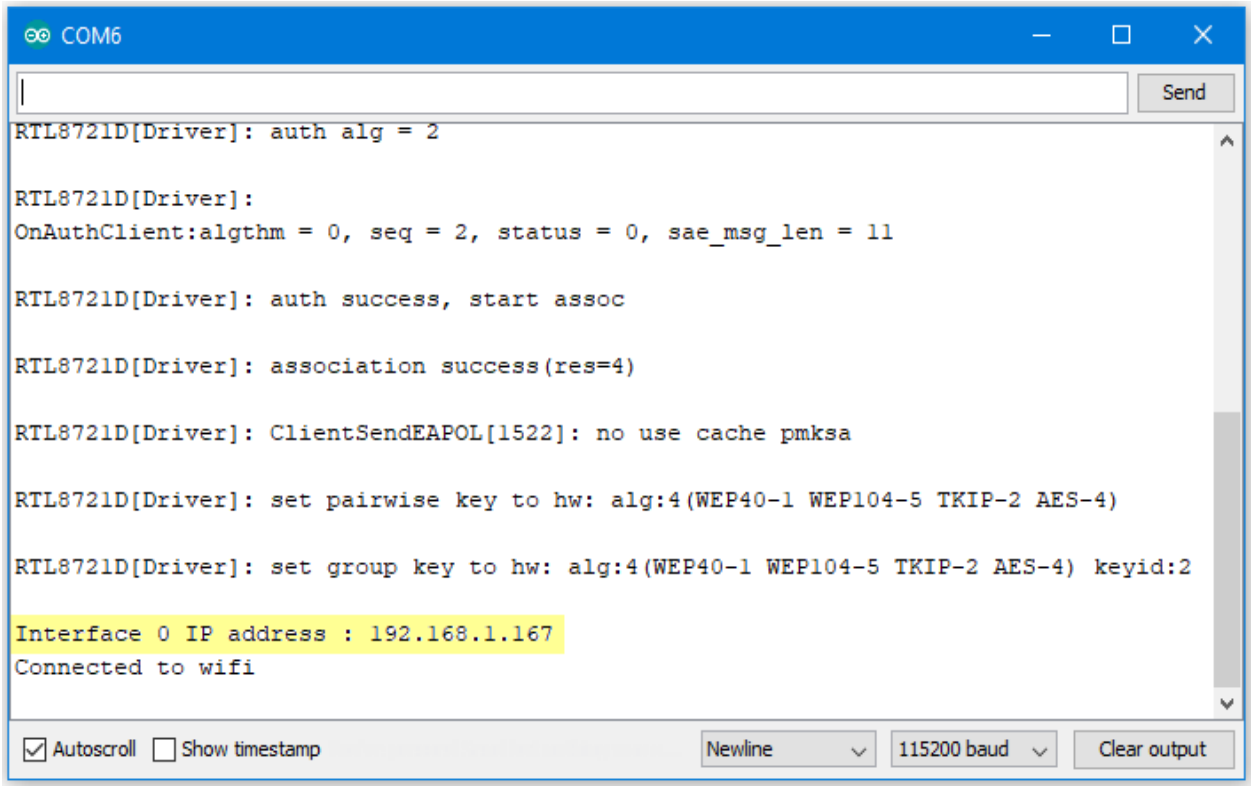
Open the "CalculateUdpReceiveDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation`" -> "`CalculateUdpReceiveDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



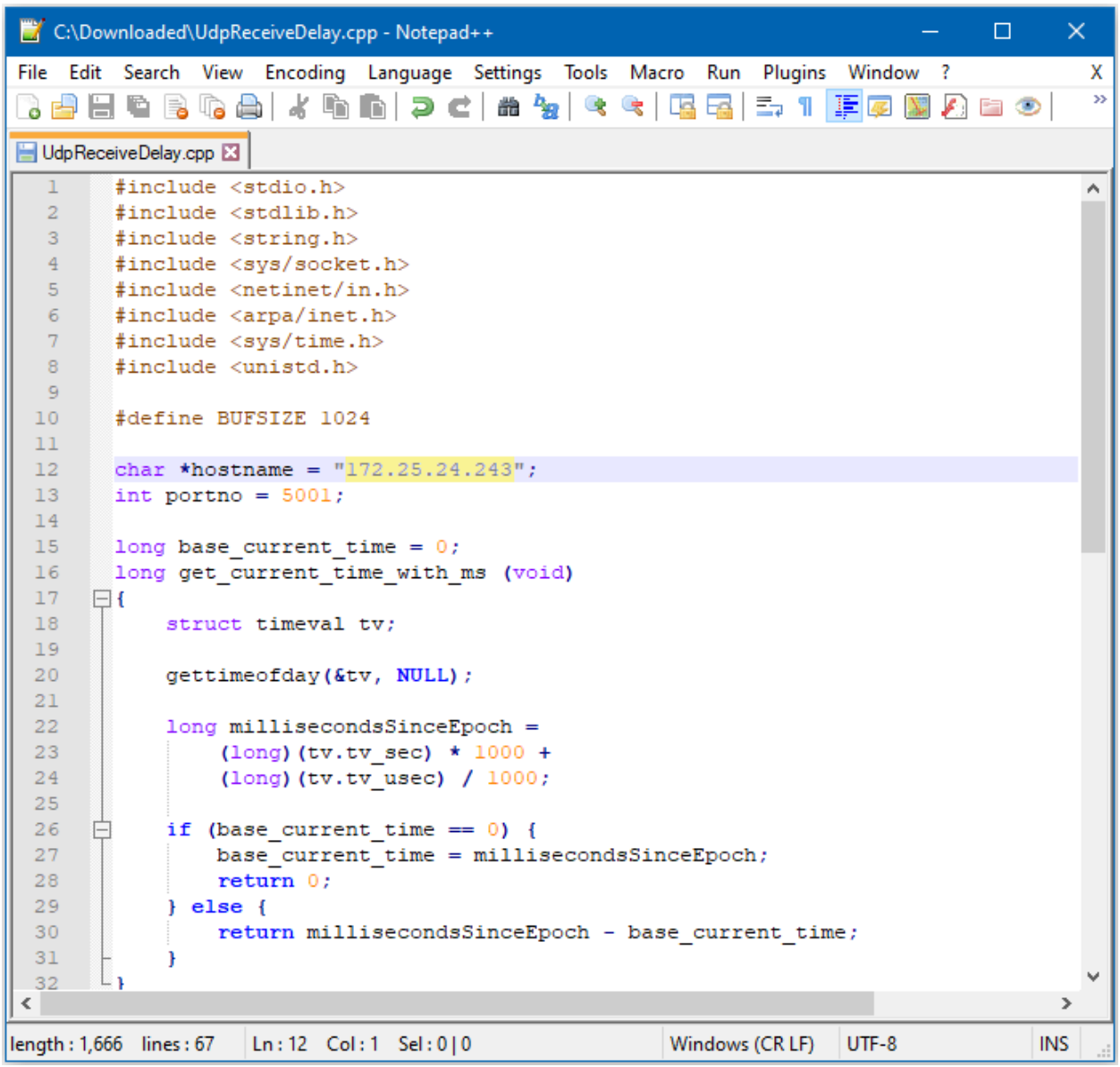
Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

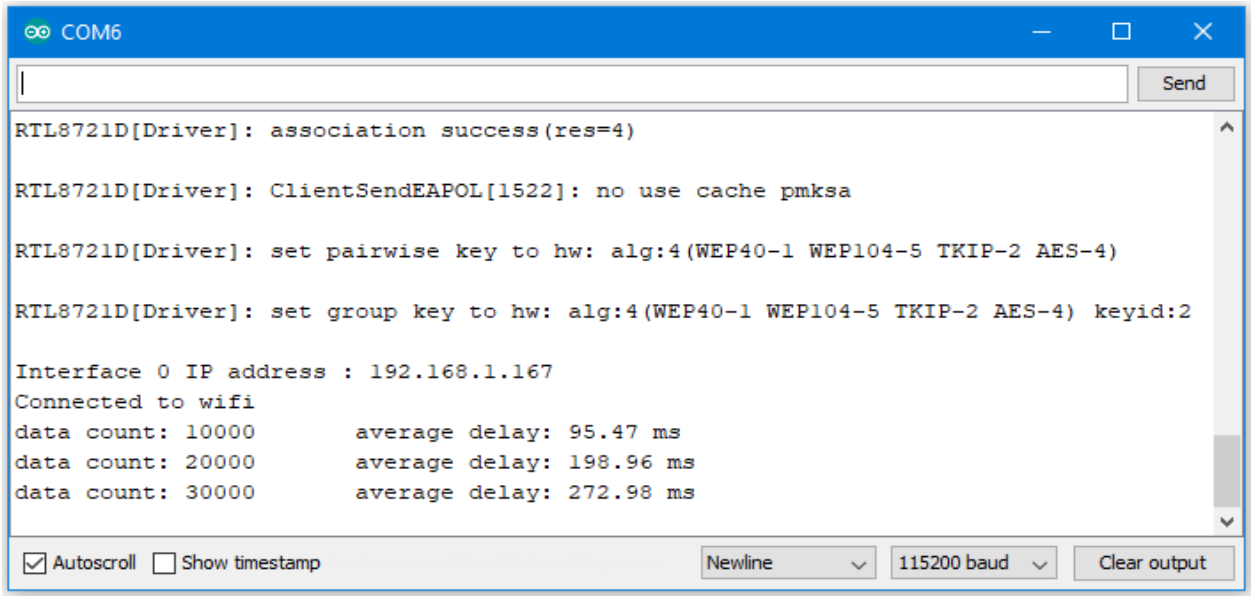
Follow the instructions there to install it. Next, from the "CalculateUdpReceiveDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveDelay.cpp", and use the command "g++ UdpReceiveDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpDelay.exe file, and the computer will begin to send packets to Ameba. Once 10000 packets have been received, Ameba will calculate the average delay and print out the result to the serial monitor. It may take up to a few minutes for 10000 packets to be sent.



WiFi - Approximate UDP Sending Delay

Materials

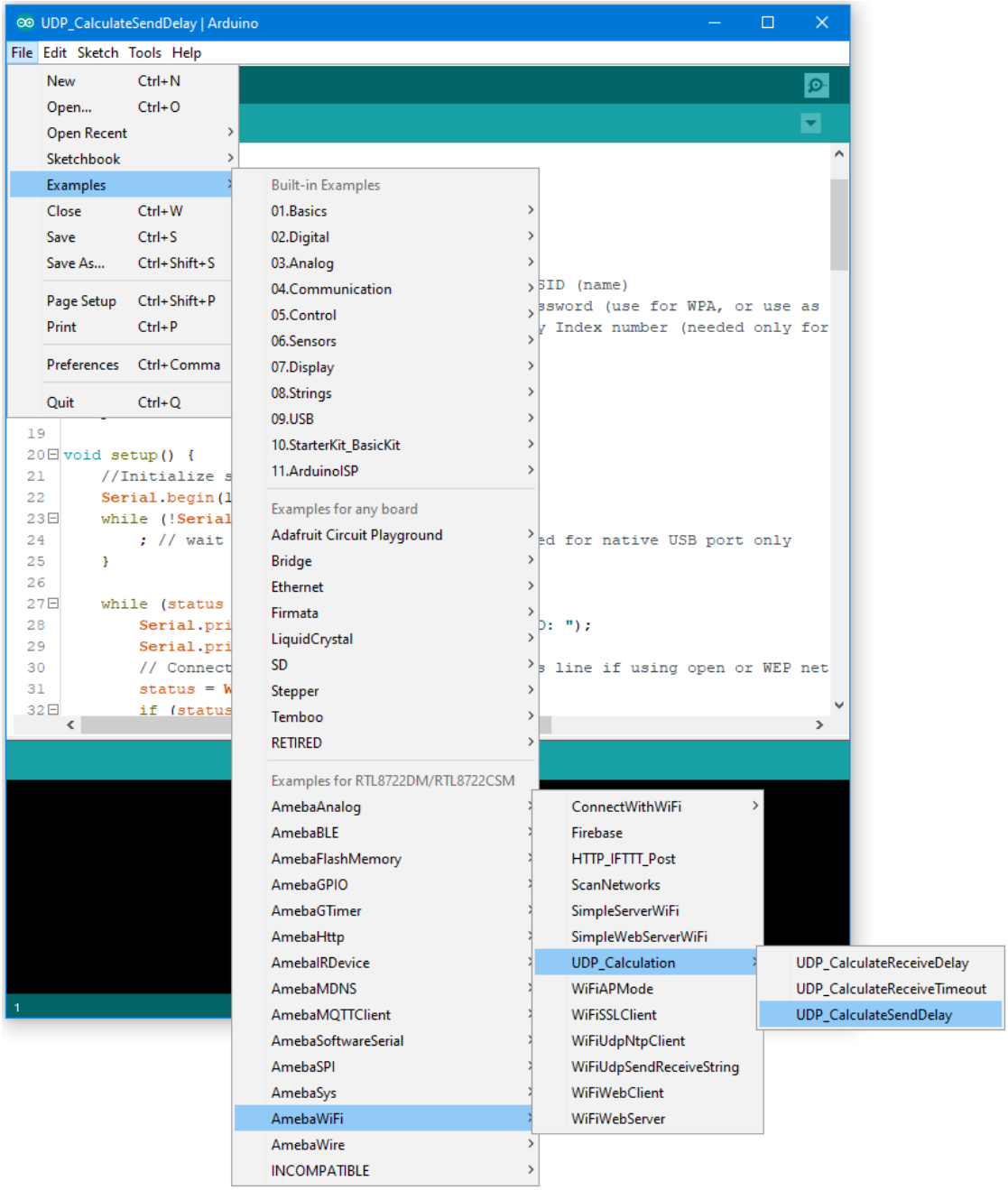
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

This example uses Ameba to send UDP packets to a computer and calculates the UDP sending delay.

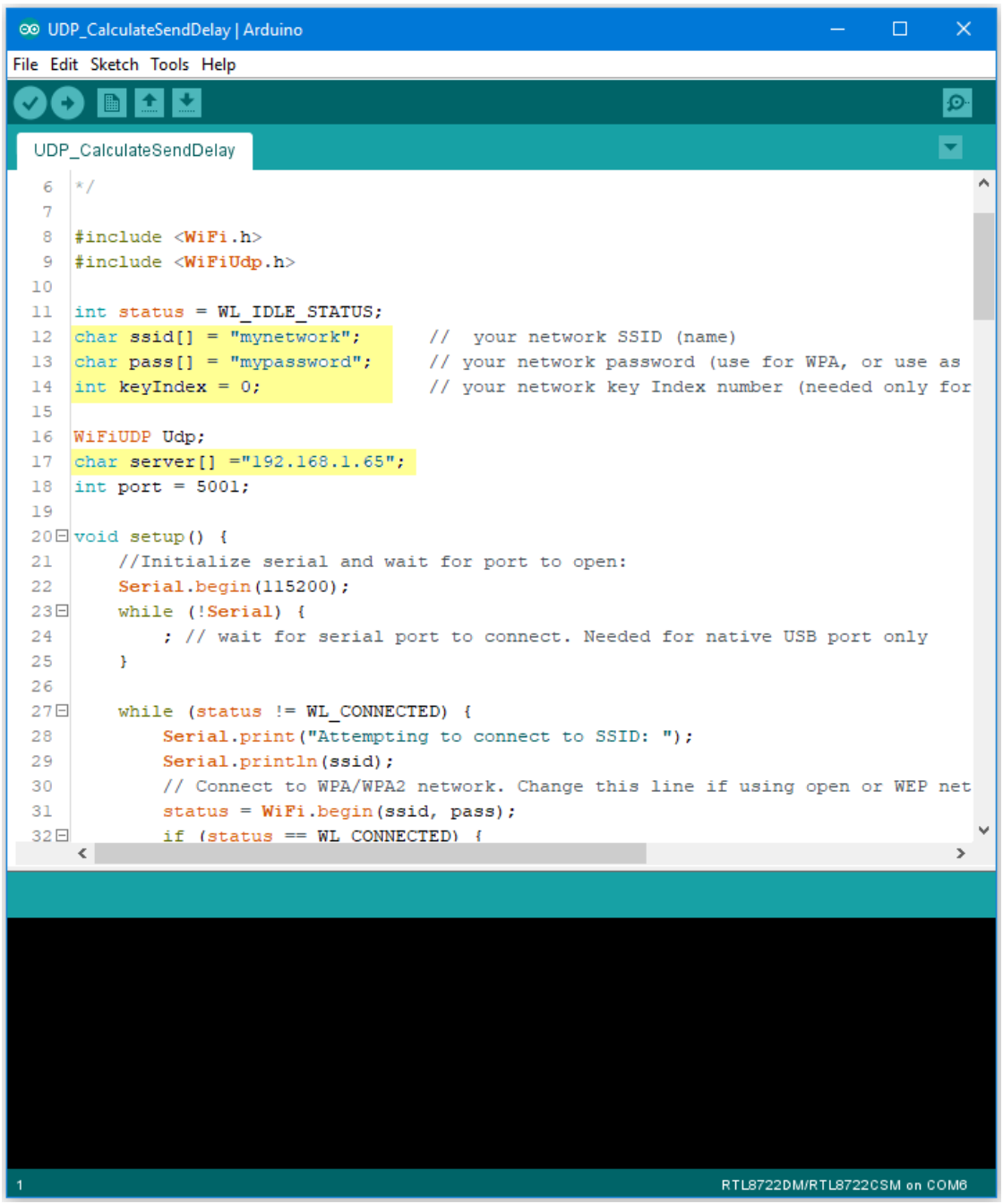
Ameba Preparation

Open the "CalculateUdpSendDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation `" -> "`CalculateUdpSendDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.

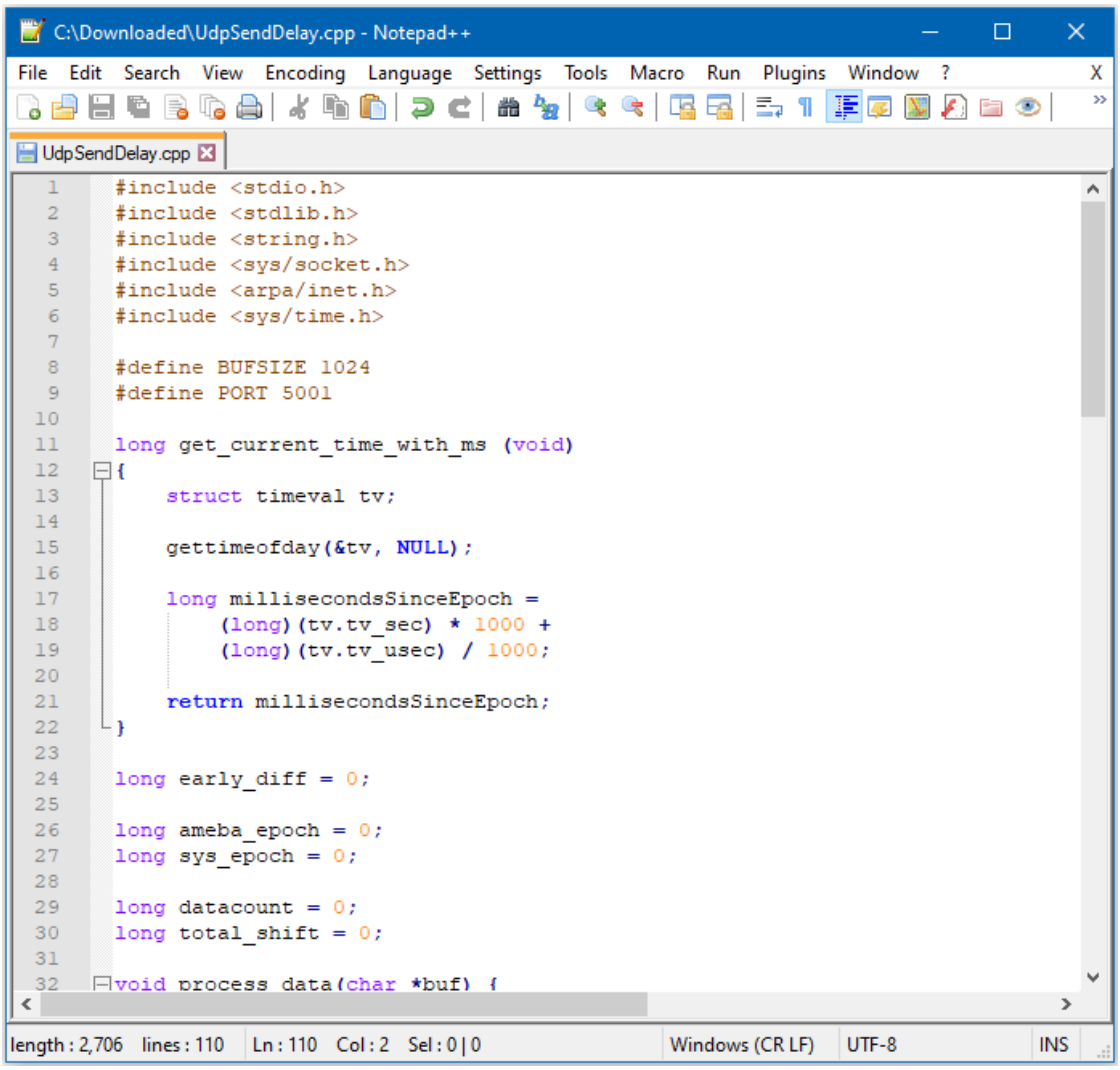
The server variable also needs to be changed to match the IP address of your computer. You can find the IP address using the “ipconfig” command in a terminal window.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpSendDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file and rename the file to "UdpSendDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpSendDelay.cpp", and use the command "g++ UdpSendDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

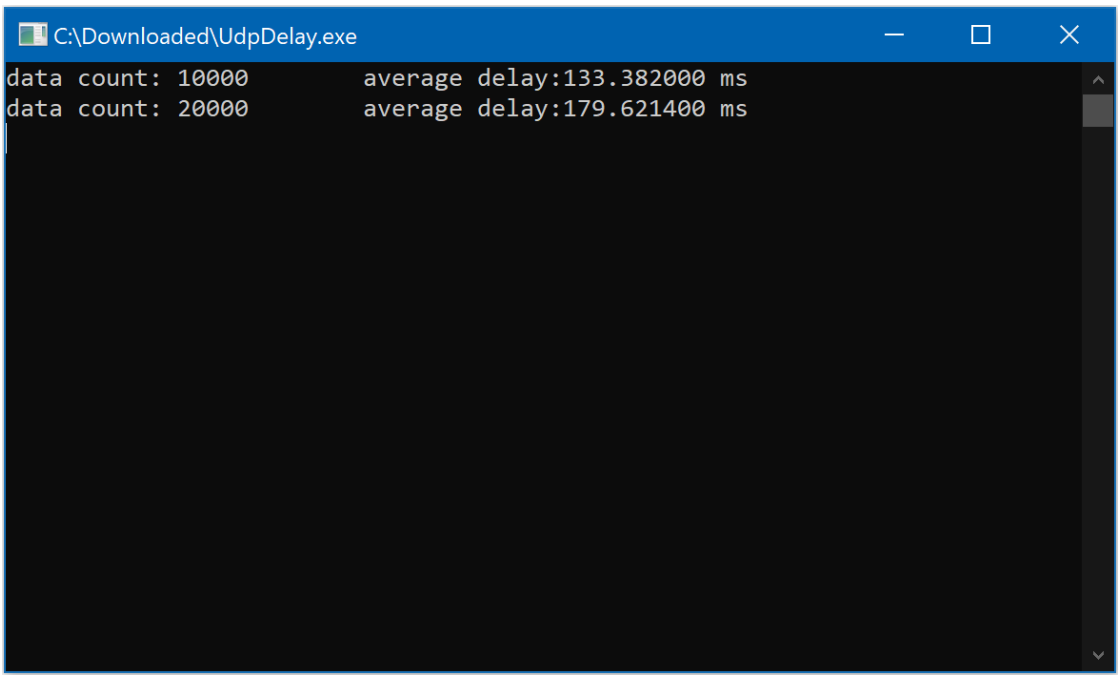
Running the Example

First, on the computer, run the UdpDelay.exe file, and the computer will begin to listen for packets from Ameba.

Next, compile and upload the code from the Arduino IDE to Ameba and press the reset button when the upload is complete.

The Ameba will begin to send UDP packets to the computer. Once 10000 packets have been received, the computer will calculate the average delay and print out the result.

It will take some time for 10000 packets to be sent.



WiFi - Approximate UDP Receive Timeout

Materials

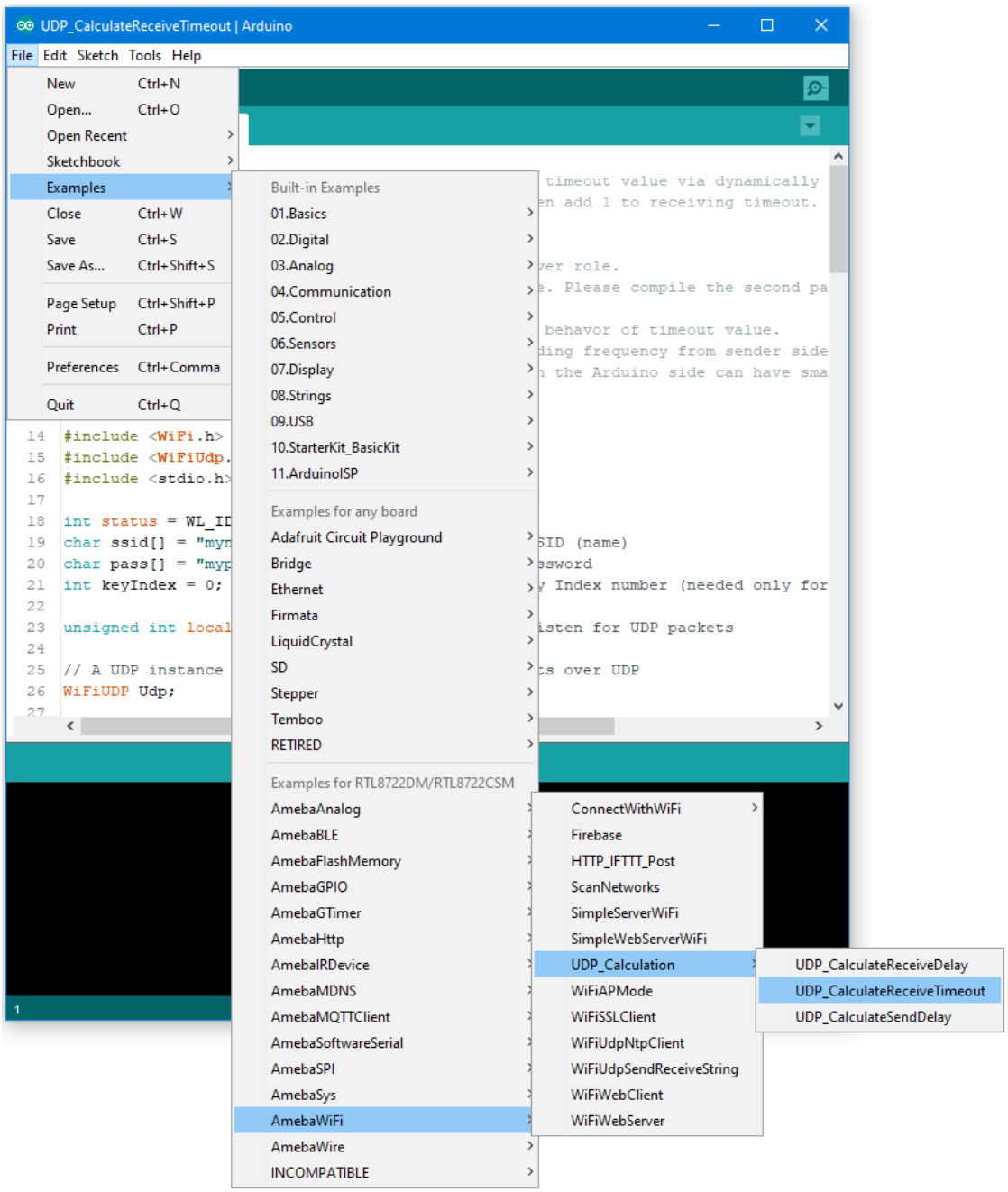
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

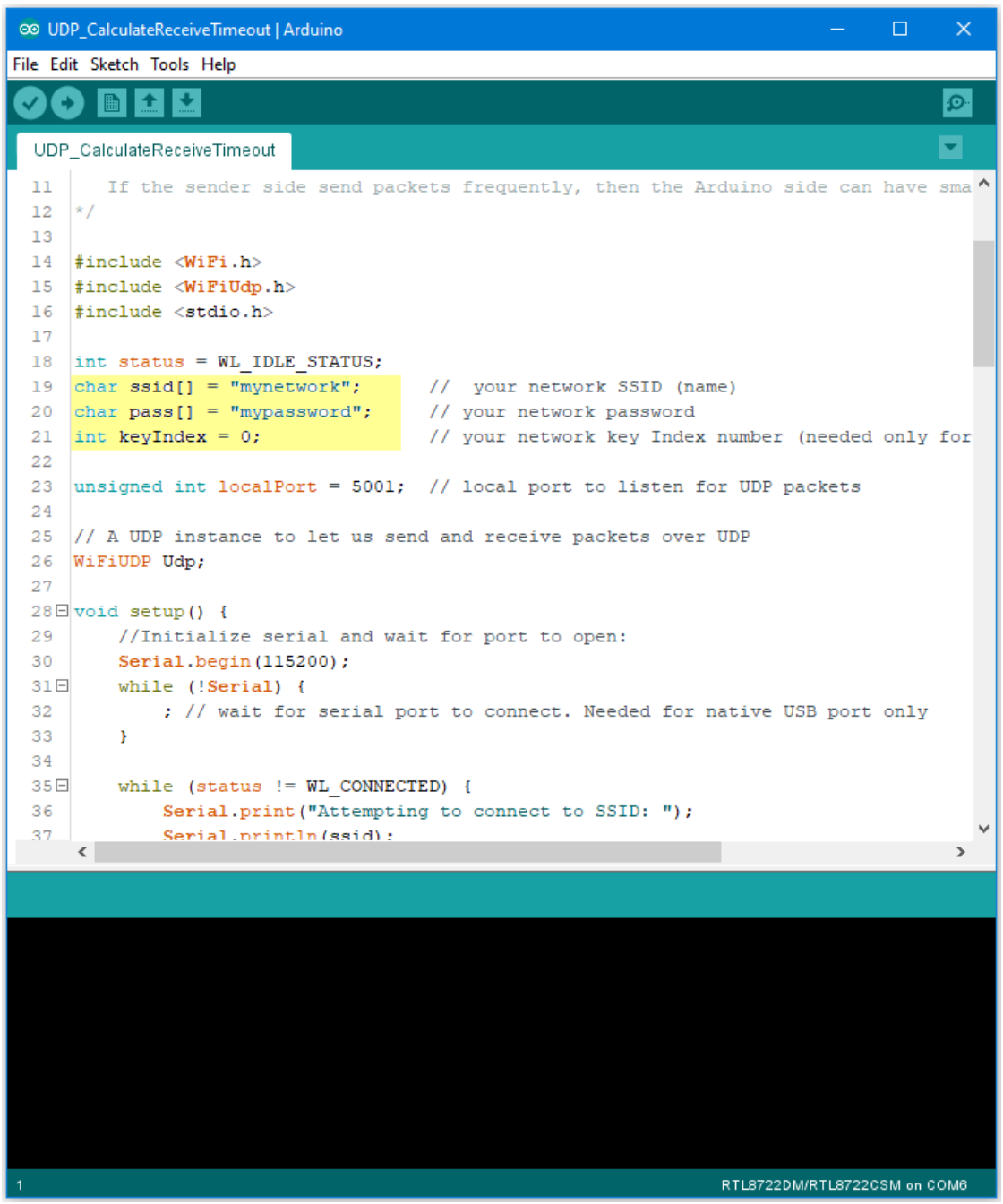
This example uses Ameba to receive UDP packets from a computer and calculates the allowed UDP receive timeout setting.

Ameba Preparation

Open the "CalculateUdpReceiveTimeout" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation `" -> "`CalculateUdpReceiveTimeout`".

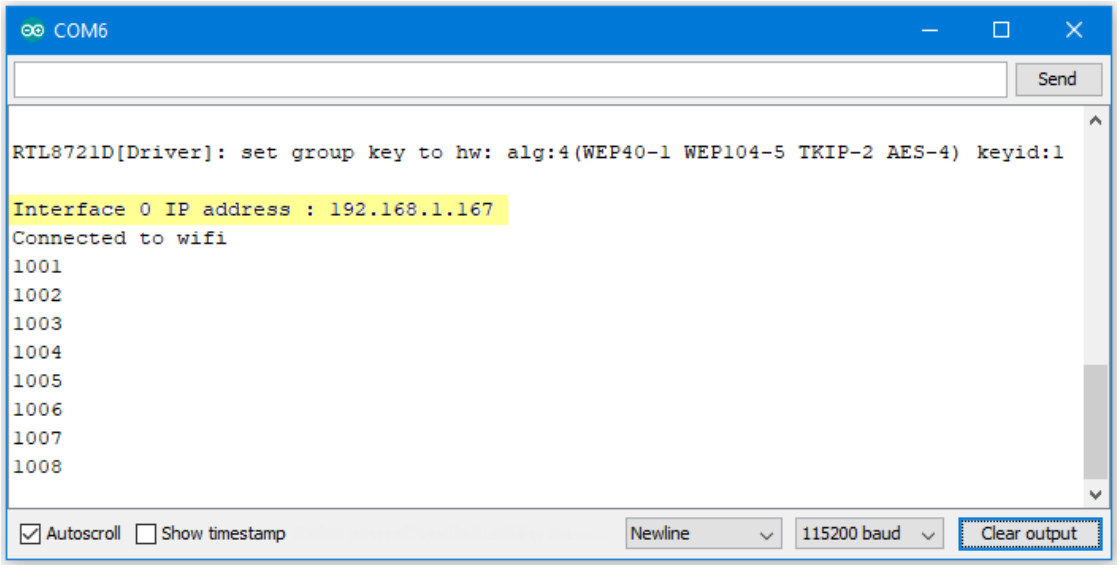


In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished.

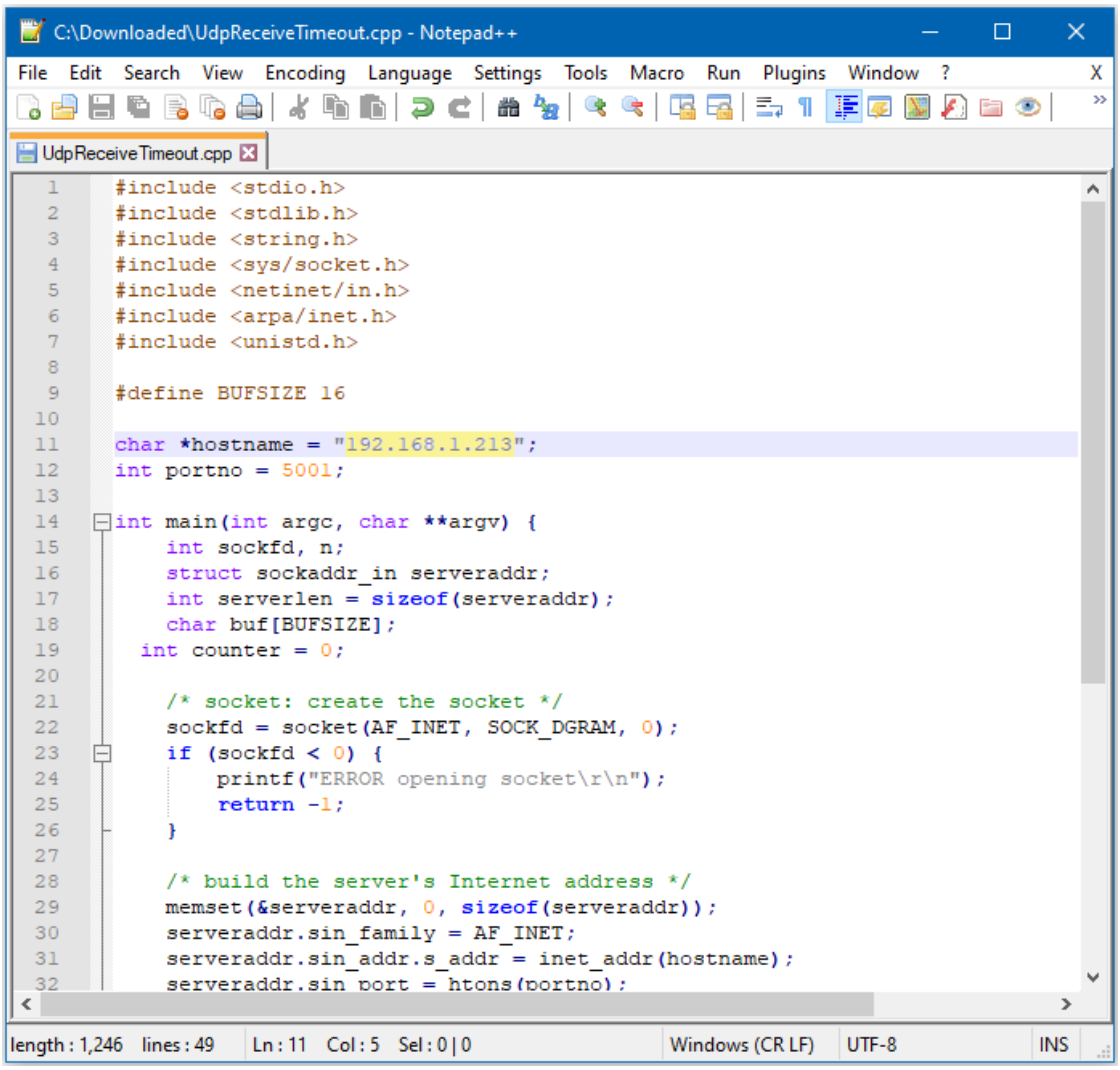
Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpReceiveTimeout" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveTimeout.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveTimeout.cpp", and use the command "g++ UdpReceiveTimeout.cpp -o UdpTimeout" to compile the code. A file named "UdpTimeout.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpTimeout.exe file, and the computer will begin to send packets continuously to Ameba.

The timeout value is set to 1000ms initially. For each packet received successfully, Ameba decreases the timeout value. The next packet must be received within the timeout period, otherwise Ameba registers a failed packet and increases the timeout value. Open the serial monitor and observe the timeout value converge to a minimum value.

WiFi - Connect to WiFi networks

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Procedure

There three common encryption type in WiFi connection. The first one is “OPEN”, which means there is no password needed to connect to this network. The second type of encryption is WPA, which requires the correct password to access. The third type is WEP, which requires a hexadecimal password and a keyindex.

In the following, we will give a brief introduction on how to establish WiFi connection with these three types of encryption on Ameba.

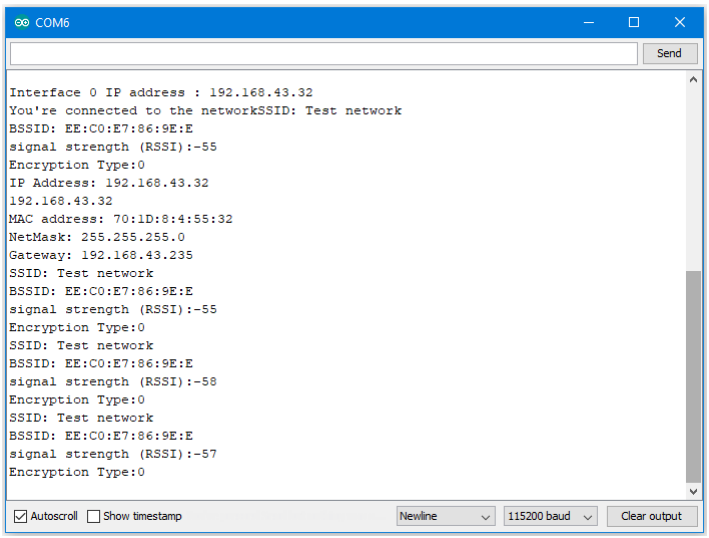
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”.

- Open (WiFi connection without password)

  Open the “ConnectNoEncryption” example in “File” -> “Examples” -> “AmebaWiFi” -> “ConnectWithWiFi” -> “ConnectNoEncryption”

  In the sample code, modify “ssid” to be the same as the WiFi SSID to be connected to.

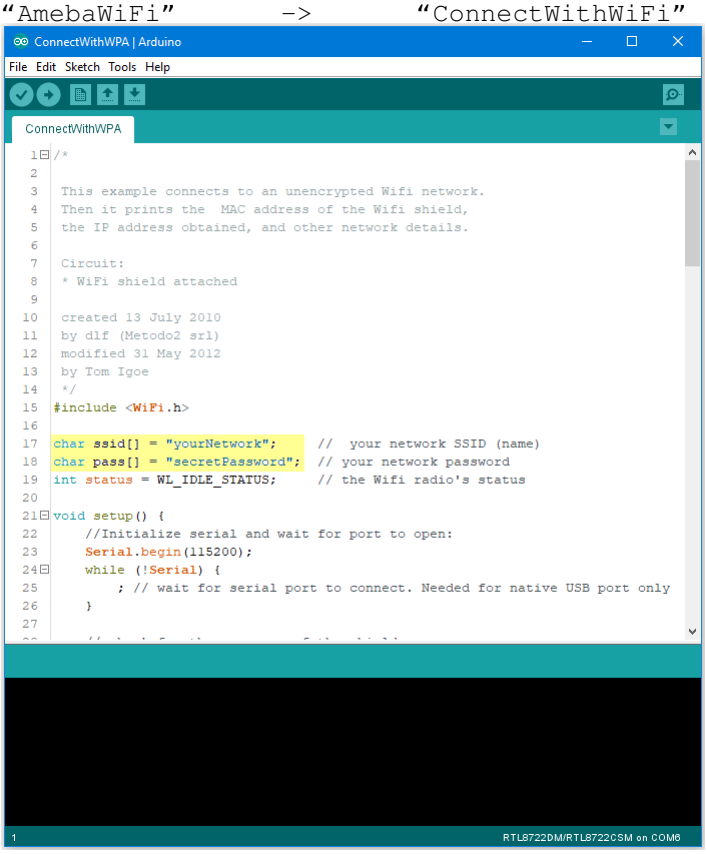
  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message “You’re connected to the networkSSID: XXXXX”, and the information of this WiFi connection is printed in the



serial monitor every 10 seconds.

- WiFi connection with WPA encryption

  Open the "ConnectWithWPA" example in "File" -> "Examples" -> "AmebaWiFi" -> "ConnectWithWiFi" -> "ConnectWithWPA"



  In the sample code, modify "ssid" to the WiFi SSID to be connected to and "pass" to the network password.

  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message "You're connected to the networkSSID: XXXXX", and the information of this WiFi connection is printed in the serial monitor every 10 seconds.

- WiFi connection with WEP encryption

Open the "ConnectWithWEP" example in "File" -> "Examples" -> "AmebaWiFi" -> "ConnectWithWiFi" -> "ConnectWithWEP"

In the sample code, modify "ssid" to the SSID to be connected, "key" to the hexadecimal password, "keyIndex" to your key index number.

Next, upload the sample code, and press the reset button on Ameba. Then you will see a message "You're connected to the networkSSID: XXXXX", and the information of this WiFi connection is printed in the IDE every 10 seconds.

Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.encryptionType()` to get the encryption type of the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiEncryptionType
Use `WiFi.BSSID()` to get the MAC address of the router you are connected to.
https://www.arduino.cc/en/Reference/WiFiBSSID
To get the information of Ameba:
Use `WiFi.macAddress()` to get the MAC address of Ameba.
https://www.arduino.cc/en/Reference/WiFiMACAddress

Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFi.subnetMask()` to get the subnet mask.
https://www.arduino.cc/en/Reference/WiFiSubnetMask
Use `WiFi.gatewayIP()` to get the WiFi shield's gateway IP address.
https://www.arduino.cc/en/Reference/WiFiGatewayIP

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

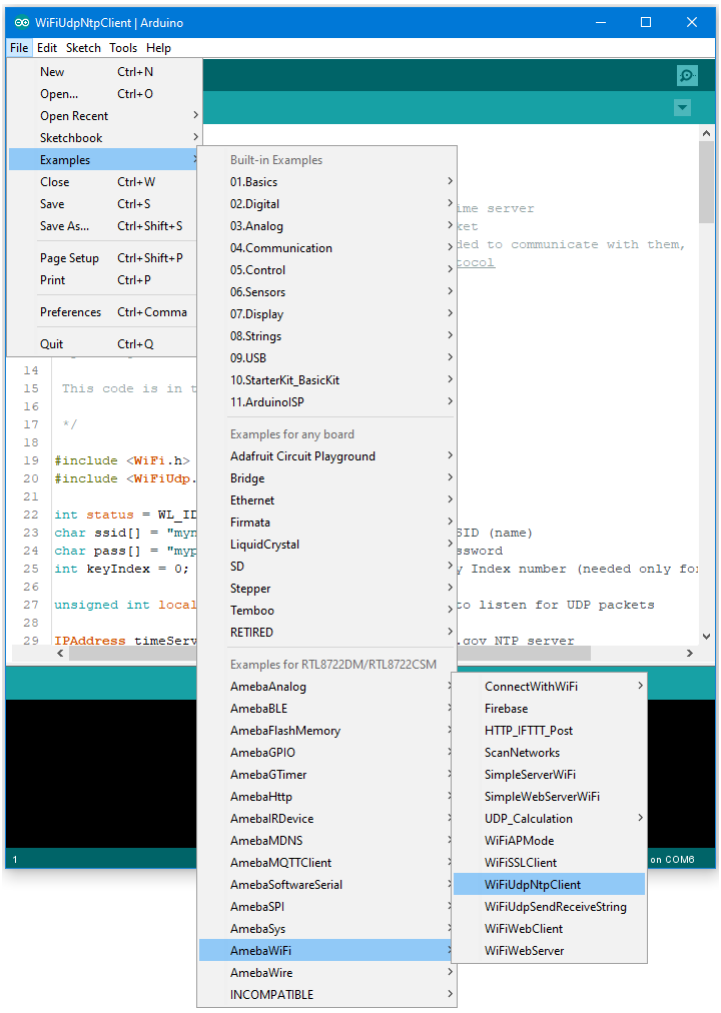
WiFi - Retrieve Universal Time (UTC) by UDP

Preparation

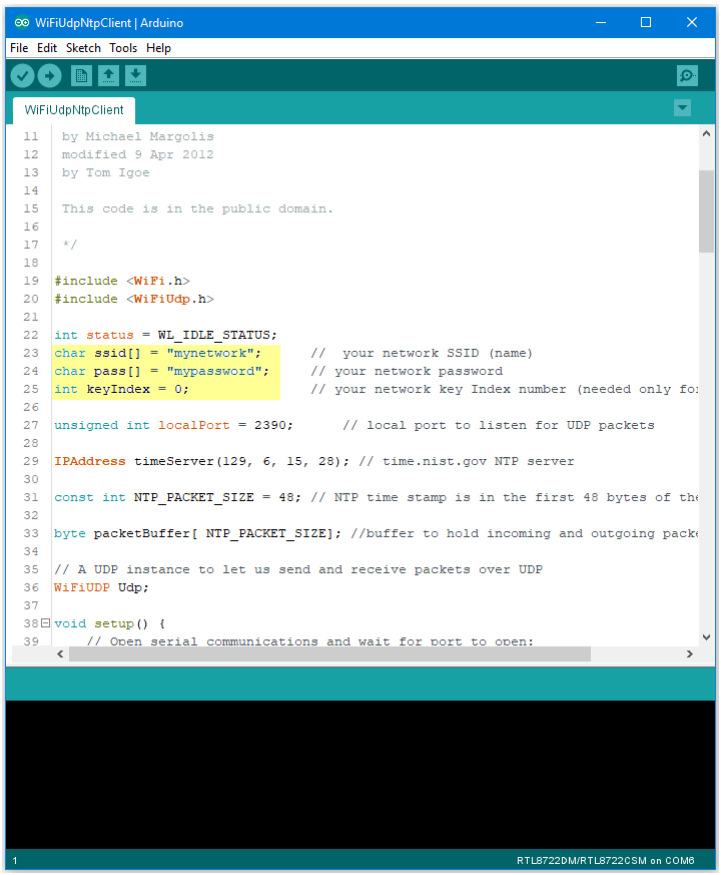
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to WiFi. Then send NTP (Network Time Protocol, RFC 1305) request to NTP server using UDP. After receiving the NTP request, the NTP server replies current UTC (Coordinated Universal Time) packet. We will parse the UTC packet to show current UTC time in the serial monitor.
Open the example: `"File" -> "Examples" -> "AmebaWiFi" -> "WiFiUdpNtpClient"`

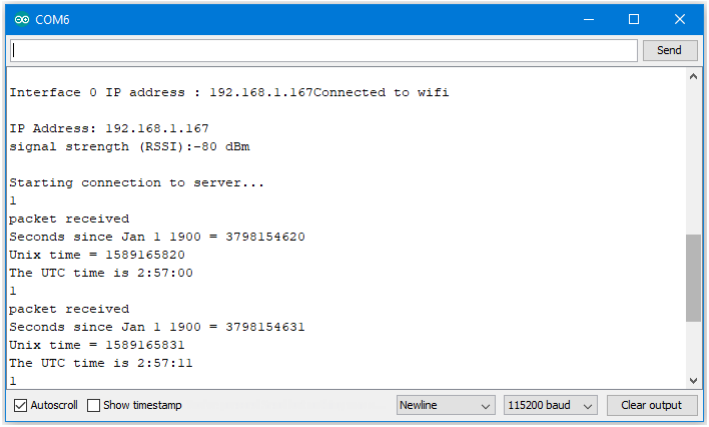


Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and sends NTP request packet to NTP server “129.6.15.28”.

We parse the replied packet and show UTC time in serial monitor:



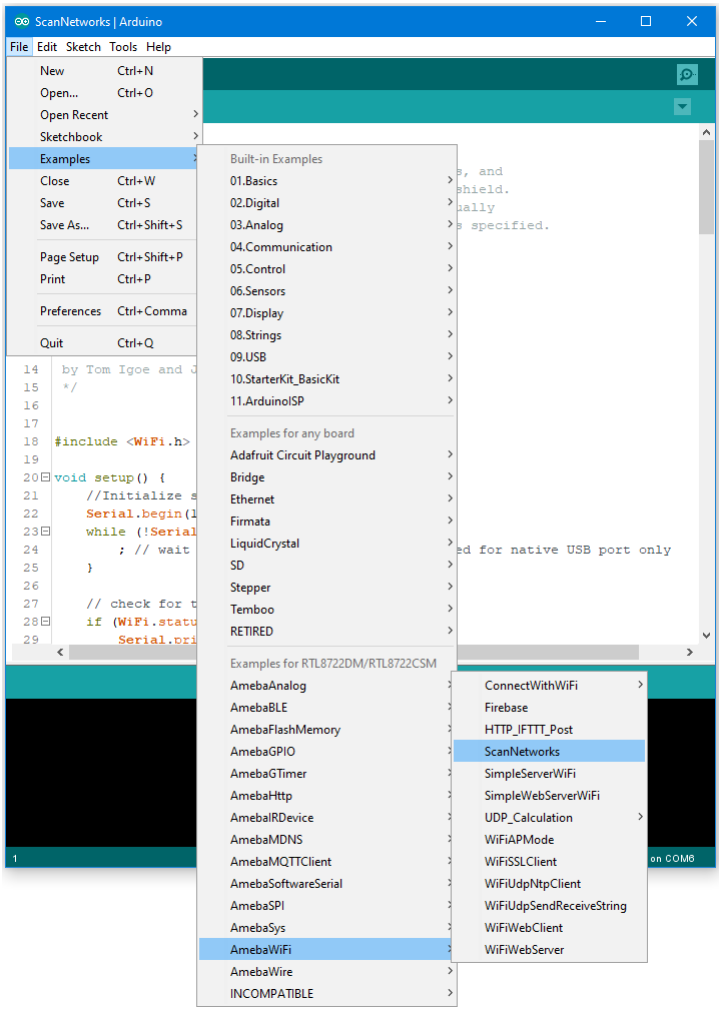
WiFi - Scan the surrounding WiFi networks

Materials

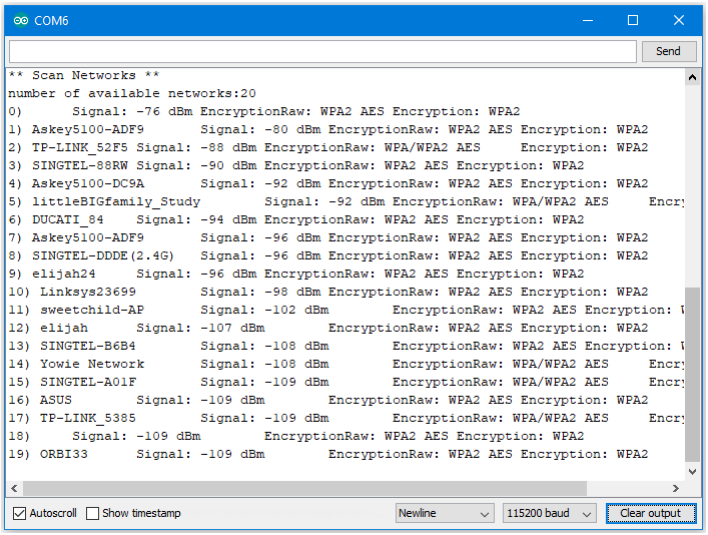
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Antenna x 1
- **AmebaD [:raw-html:`<p style="color:#1A76B4;">AMB21(RTL8722DM/CSM)</p>` / AMB23(RTL8722DM_MINI) / BW16(RTL8729DN)] x 1**

Example

In this example, we use Ameba to scan available WiFi hotspots in the surroundings, and prints the SSID, encryption type, signal strength information of each detected hotspot.
First, make sure the correct Ameba development board is selected in Arduino IDE: "`Tools`" -> "`Board`"
Open the "ScanNetworks" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`ScanNetworks`":



Then upload the sample code and press the reset button on Ameba. Afterwards, you can see "**Scan Networks**" message appears, with the detected WiFi hotspots and the information of each hotspot.



Code Reference

- First we use `WiFi.macAddress(mac)` to get the MAC address of Ameba: https://www.arduino.cc/en/Reference/WiFiMACAddress
- Then we use `WiFi.scanNetworks()` to detect WiFi hotspots: https://www.arduino.cc/en/Reference/WiFiScanNetworks
- To get information of detected WiFi hotspot: We use `WiFi.SSID(thisNet)` to retrieve SSID of a network: https://www.arduino.cc/en/Reference/WiFiSSID We use `WiFi.RSSI(thisNet)` to get the signal strength of the connection to the router: https://www.arduino.cc/en/Reference/WiFiRSSI
- We use `WiFi.encryptionType(thisNet)` to get the encryption type of the network: https://www.arduino.cc/en/Reference/WiFiEncryptionType

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

WiFi - Set up Server to communicate

Materials

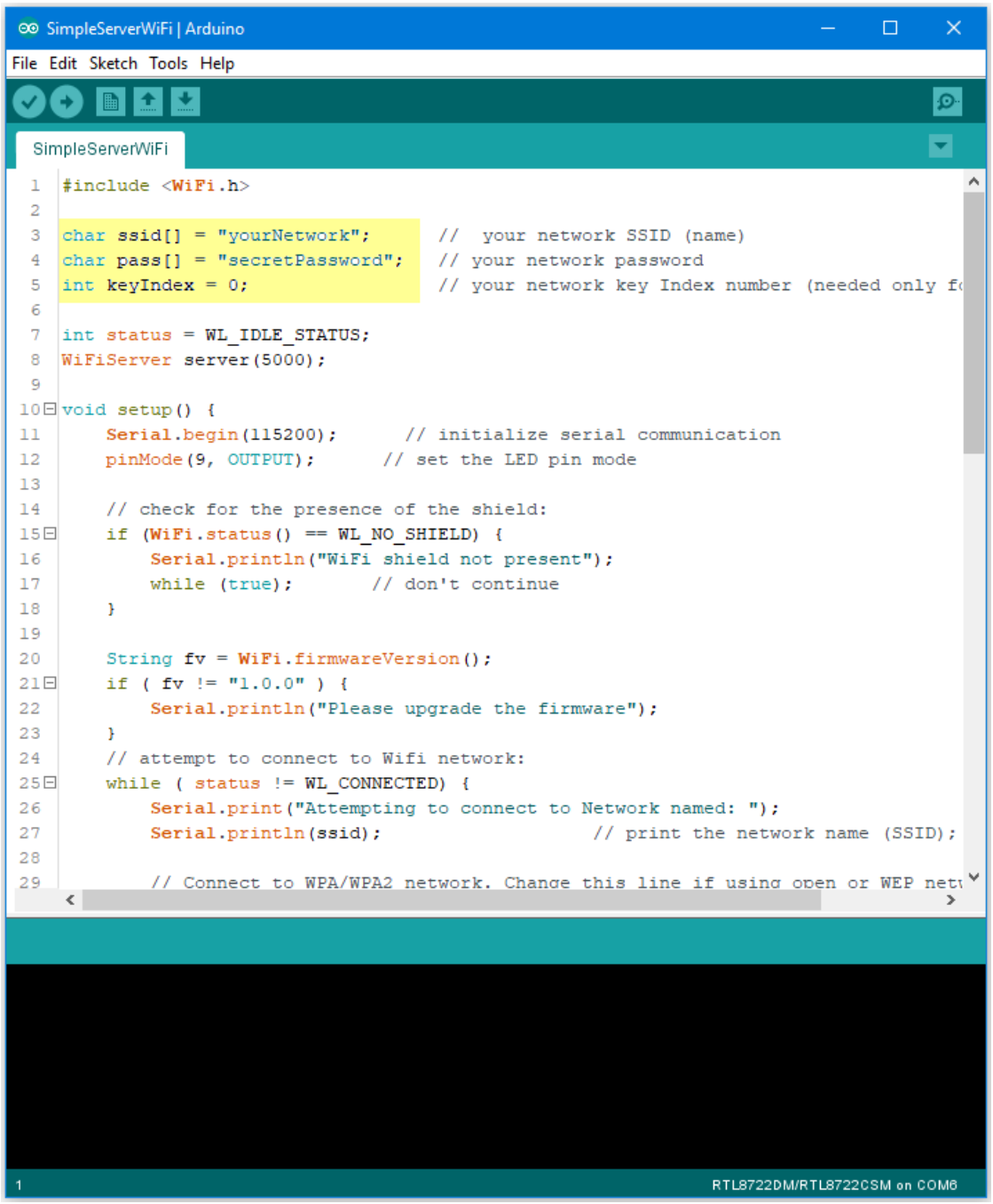
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Laptop Make sure it is connected to the same network domain as Ameba, and tcp tools are installed.

Example

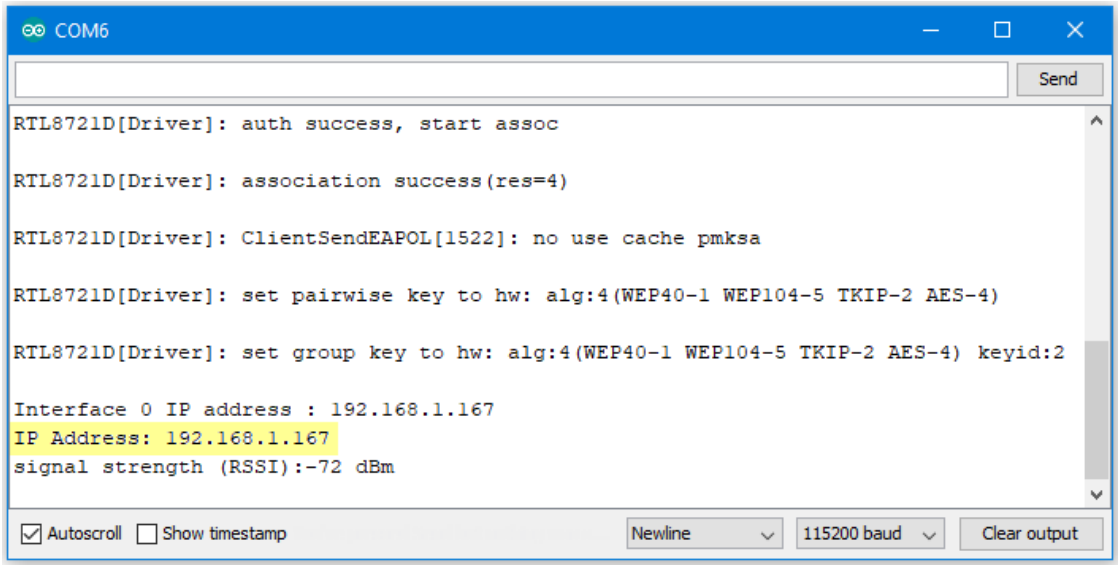
In this example, we first connect Ameba to WiFi, then we use Ameba as server to communicate with client.

First, we make sure the correct Ameba development board is set in “Tools” -> “Board”

Then, open the Simple WiFi Server example in “`File` -> “`Examples`” -> “`AmebaWiFi`” -> “`SimpleServerWiFi`”

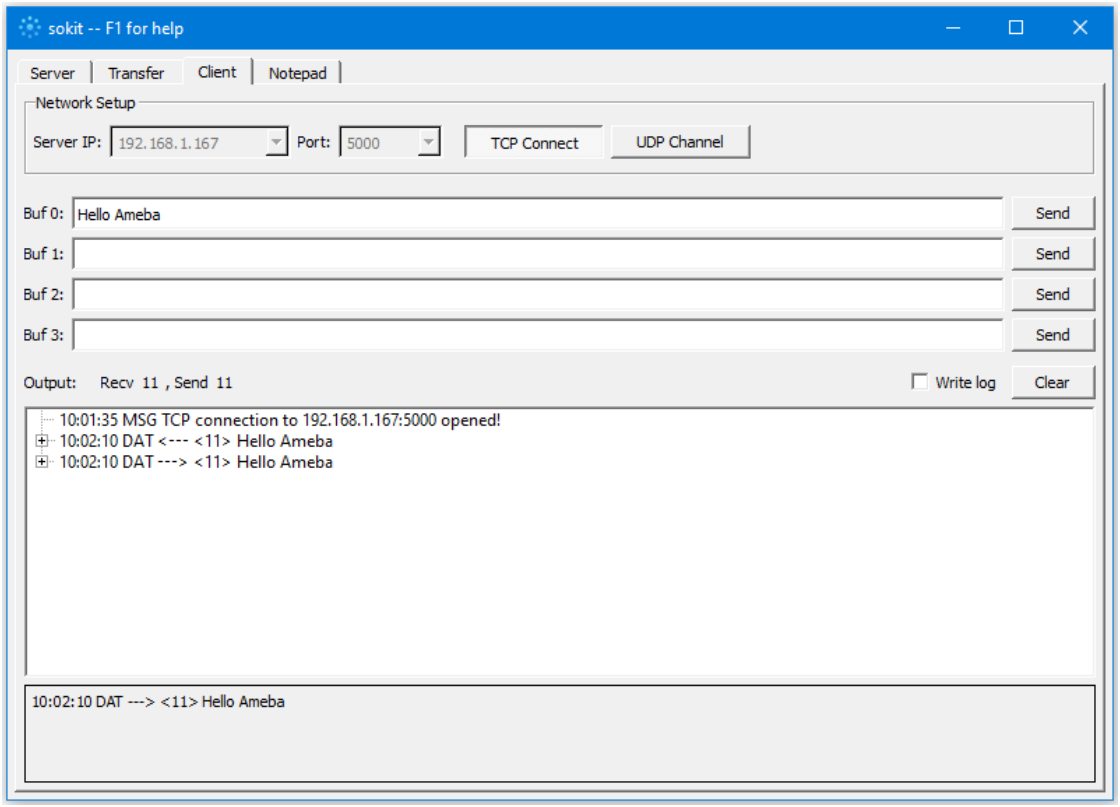


In the sample code, modify the highlighted parameters and enter the ssid and password for your WiFi connection.



Next, upload the code, then press the reset button on Ameba. At this moment, you will see the connection information is displayed in the console.

Next, we use the socket tool in the laptop to be the client and connect to the IP address of the Ameba board shown in the connection information at port 5000. (Note: The socket tool we used in this example is “sokit”)

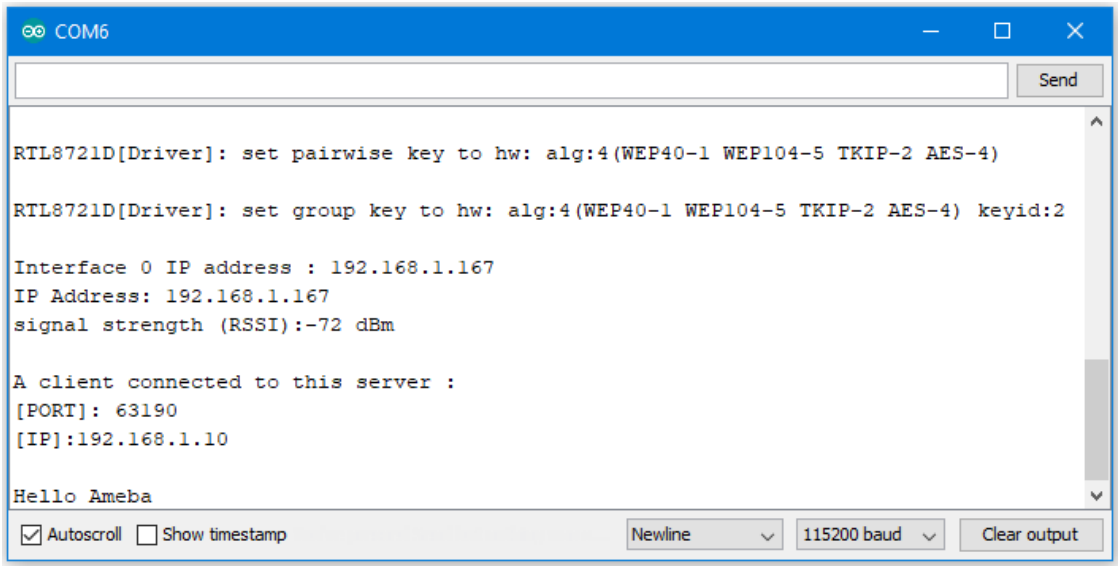


Click on the “Client” tab to choose the client mode, specify the IP and port of the server, then click “TCP Connect”.

If the connection is established successfully, the server shows a message: “A client connected to this Server”, and the IP and port of the connected client.

In this example, when the client and server are connected and the client sends a string to Ameba server, the Ameba server

returns the identical string back to the client.



The string sent to server is returned and showed at the client side.

Code Reference

Use `WiFi.begin()` to establish WiFi connection;
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the Ameba WiFi shield's IP address.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Create server and transmitting data:
Use `Server(port)` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.read()` to read the next byte received from the server.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.write()` to write data to the server.
https://www.arduino.cc/en/Reference/WiFiClientWrite
Use `client.stop()` to disconnect from the server.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up Client to Retrieve Google Search Information

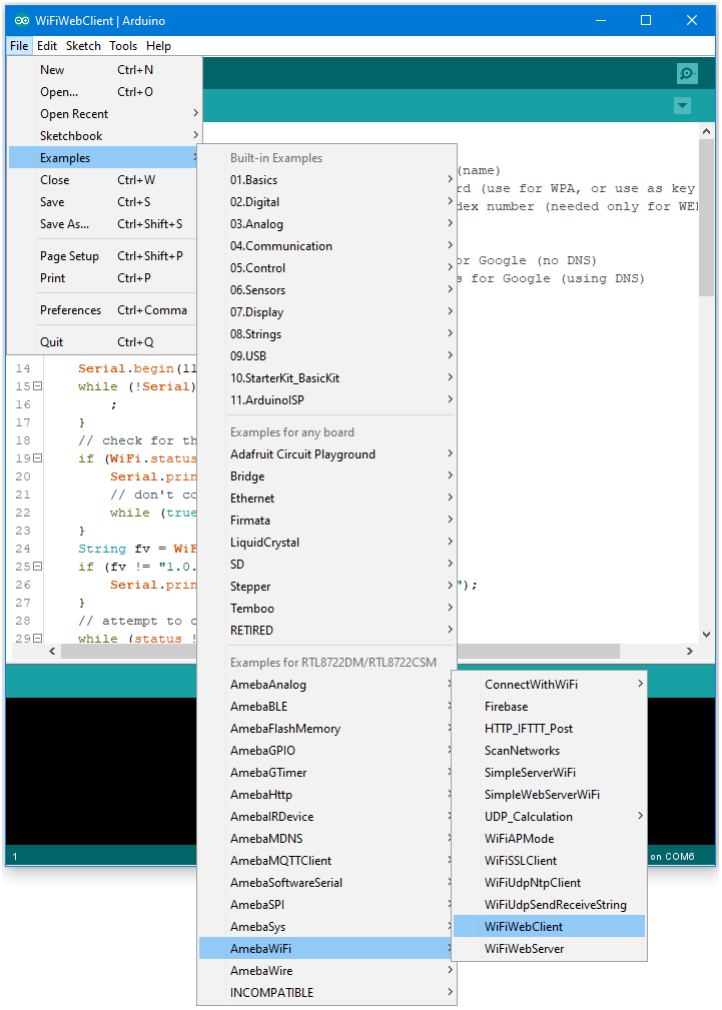
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

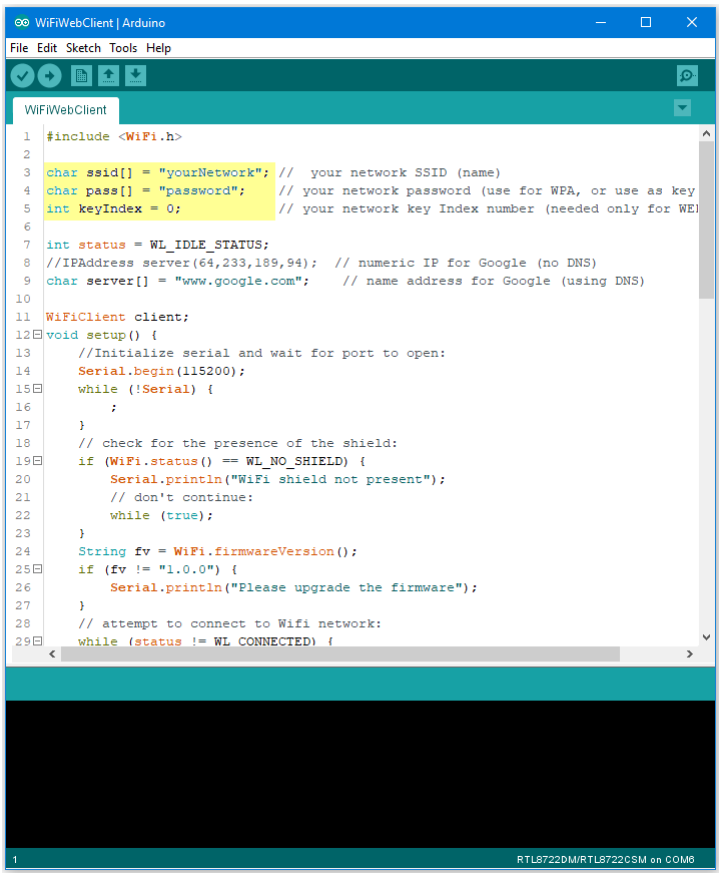
Example

In this example, we use Ameba to be a web client to retrieve information from the Internet.
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”

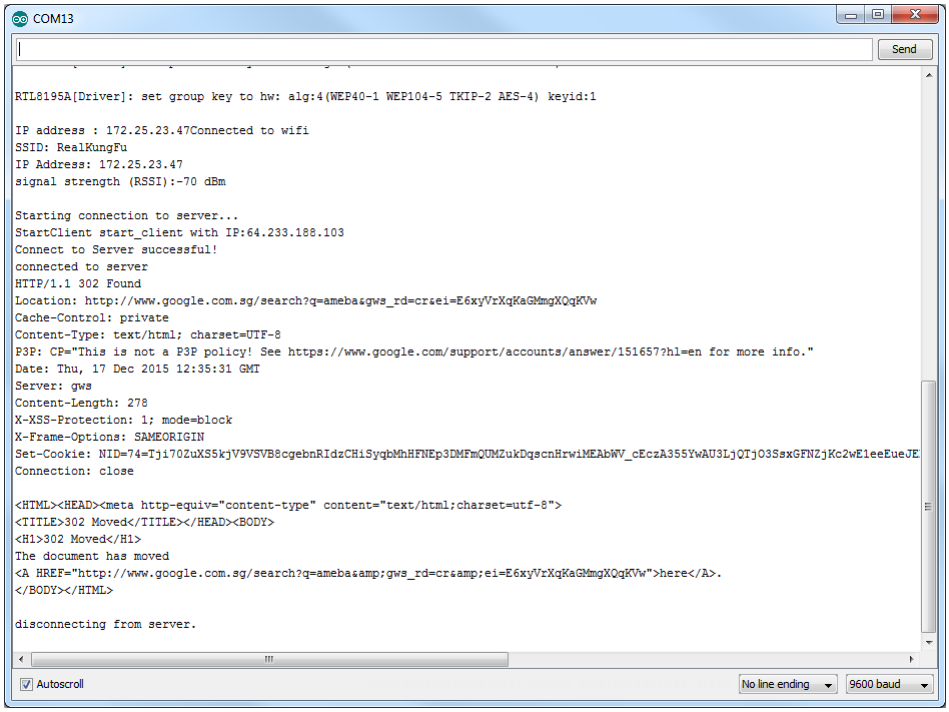
Then open “`File” -> “Examples” -> “AmebaWiFi” -> “WiFiWebClient`”



In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. Then you can see the information retrieved from Google is shown in the Arduino serial monitor.



Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection: Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiClient()` to create a client.
https://www.arduino.cc/en/Reference/WiFiClient
Use `client.connect()` to connect to the IP address and port specified.
https://www.arduino.cc/en/Reference/WiFiClientConnect
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up UDP Server to Communicate

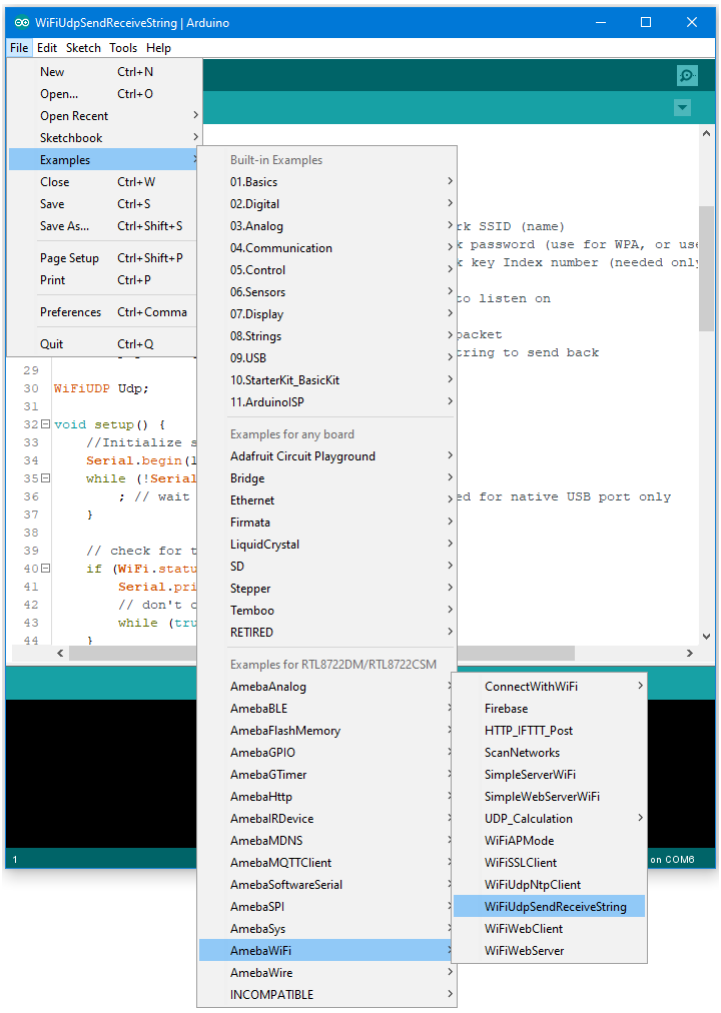
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

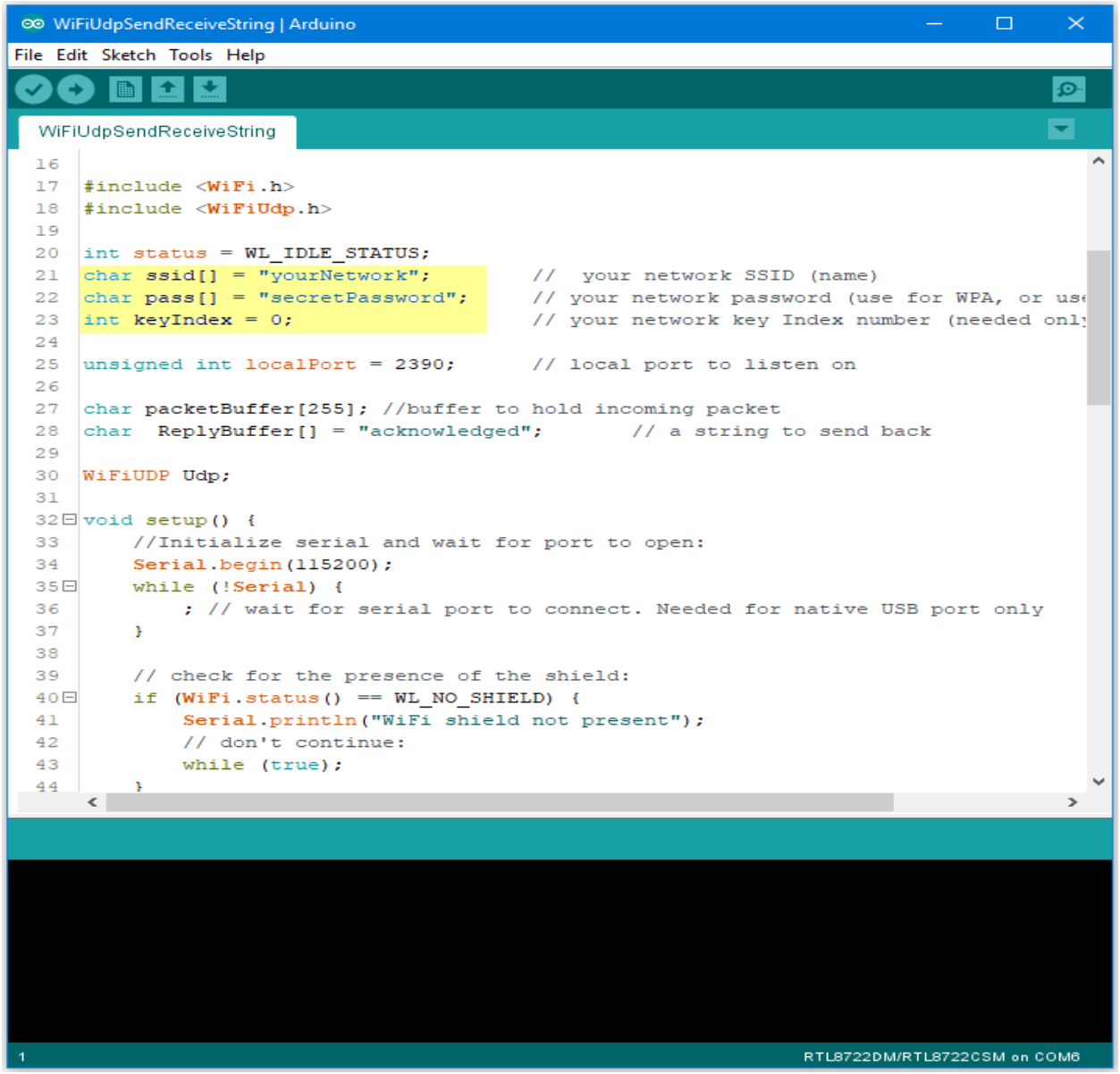
Example

In this example, we connect Ameba to WiFi and use Ameba to be an UDP server. When Ameba receives a message from UDP client, it replies "acknowledged" message to client.

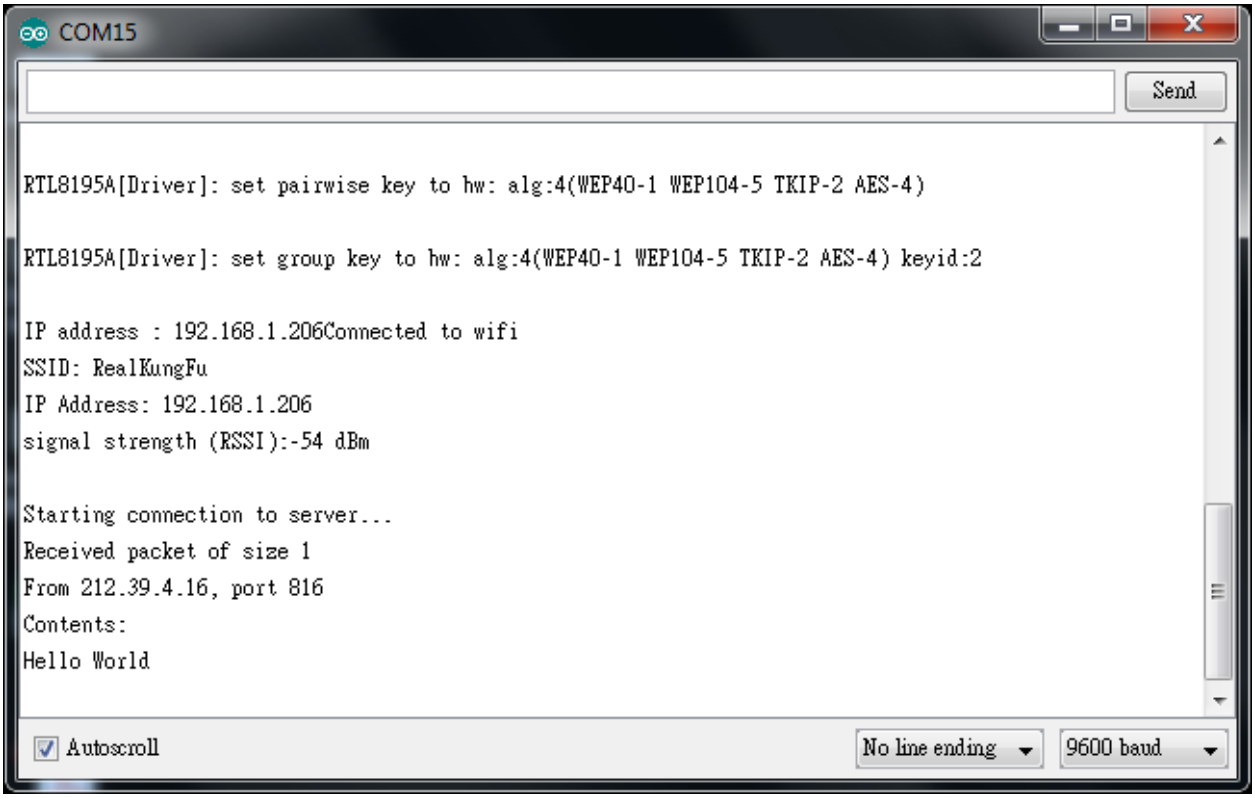
Open the WiFi Web Server example. "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiUdpSendReceiveString`"



Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



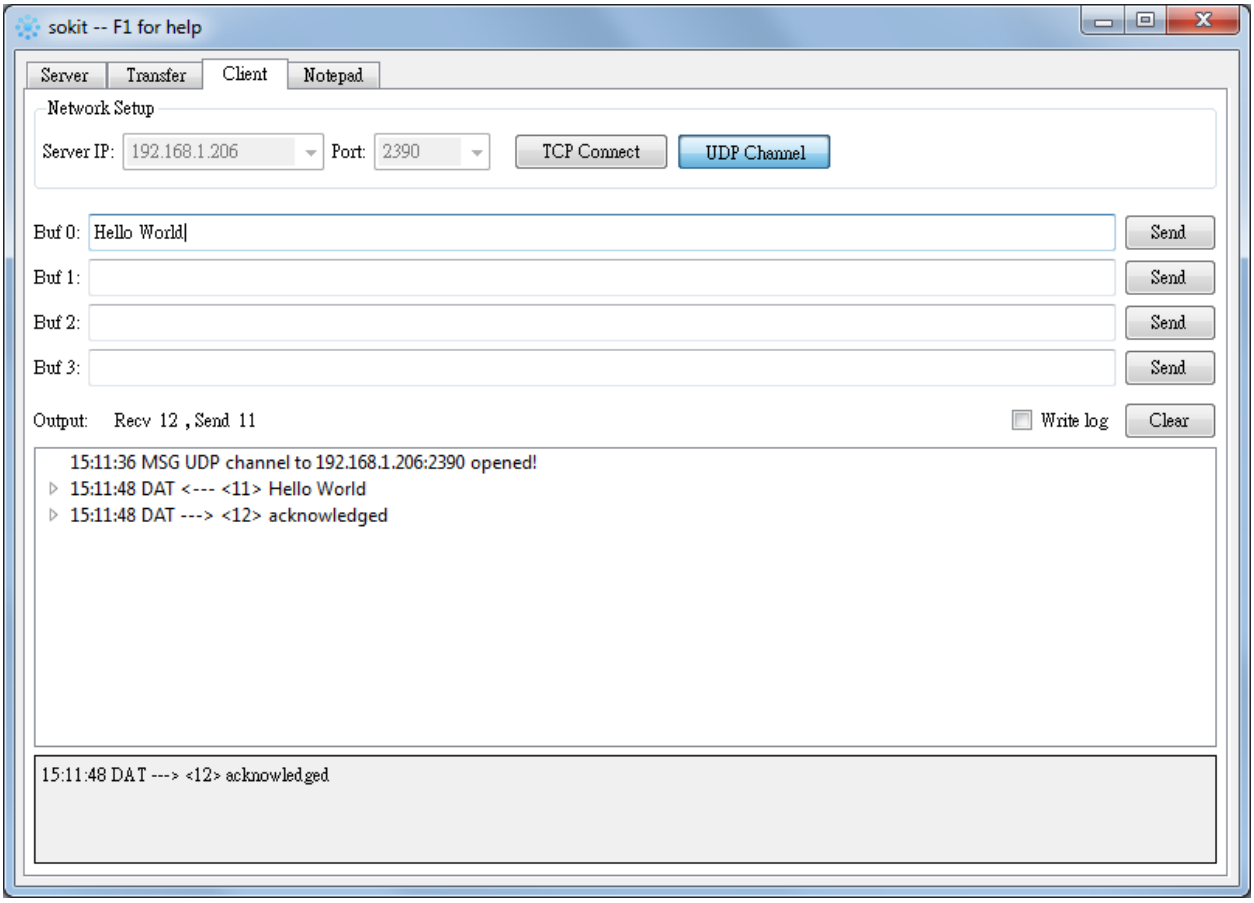
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and starts the UDP server with port 2390. After the UDP server starts service, Ameba prints the “Starting connection to server” message and waits for client connection.



As to the UDP client, we use “sokit” program in the computer to connect to UDP server.

Choose client mode and fill in the IP of UDP server (which is the IP of Ameba) and port 2390, then click “UDP Connect”.

After the connection is established, fill in “Hello World” in the Buf 0 field in sokit and click “Send”. Then you can see the Ameba UDP server replies “acknowledged”.



Code Reference

Refer to the Arduino tutorial for detailed information about this example.
https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString

First, use `begin()` to open an UDP port on Ameba.
https://www.arduino.cc/en/Reference/WiFiUDPBegin

Use `parsePacket()` to wait for data from client.
https://www.arduino.cc/en/Reference/WiFiUDPParsePacket

When a connection is established, use `remoteIP()` and `remotePort()` to get the IP and port of the client.
https://www.arduino.cc/en/Reference/WiFiUDPRemoteIP

Then use `read()` to read the data sent by client.
https://www.arduino.cc/en/Reference/WiFiUDPRead

To send reply, use `beginPacket()`, `write()`, `end()`.
https://www.arduino.cc/en/Reference/WiFiUDPBeginPacket
https://www.arduino.cc/en/Reference/WiFiUDPWrite
https://www.arduino.cc/en/Reference/WiFiUDPEndPacket

WiFi - Set up WiFi AP Mode

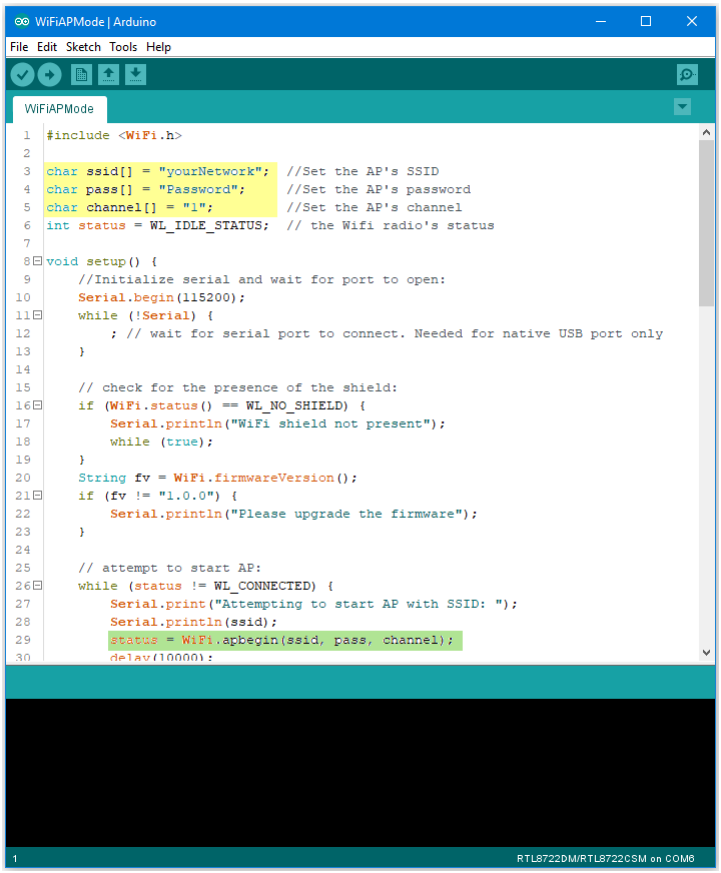
In AP mode, Ameba can accept at most 3 station connections, and can be set to open mode or WPA2 mode.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we turn on the AP mode of Ameba and connect station to Ameba.
Open the WiFi AP example, "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiAPMode`"



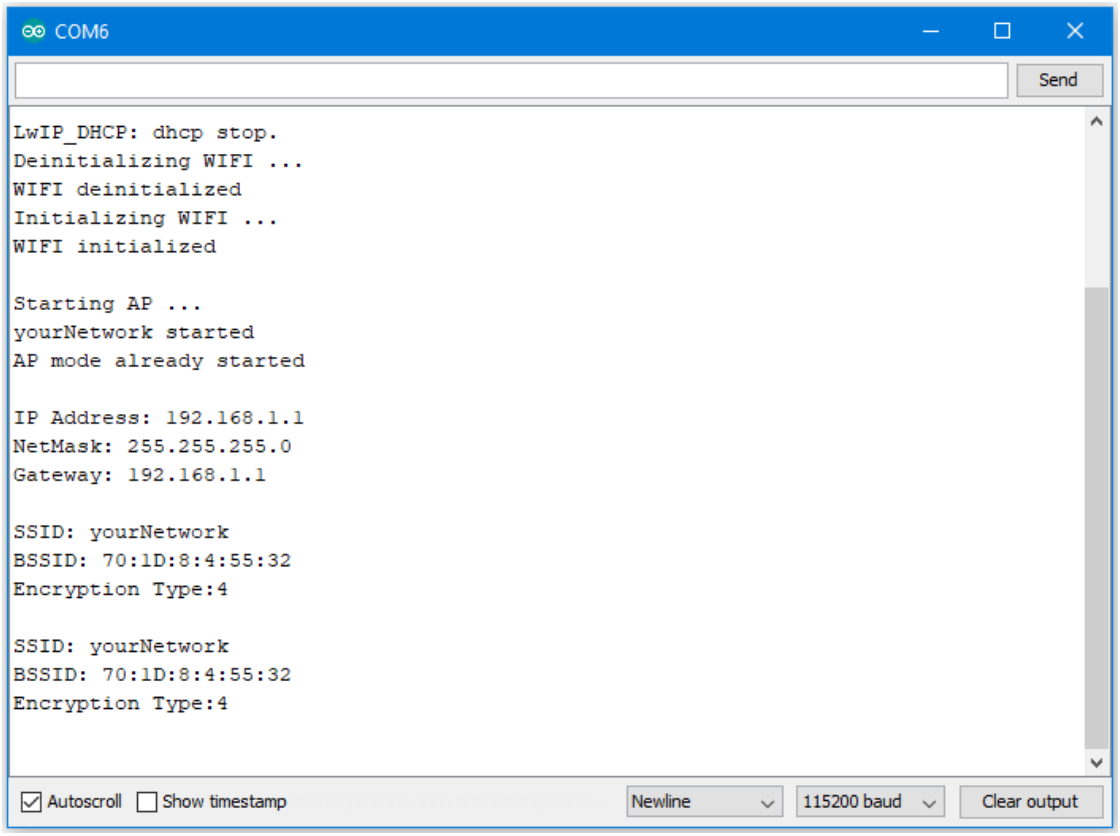
In the highlighted code snippet, fill in your SSID, PASSWORD and CHANNEL.

The code highlighted in green is the API we used to turn on the AP mode in security mode.

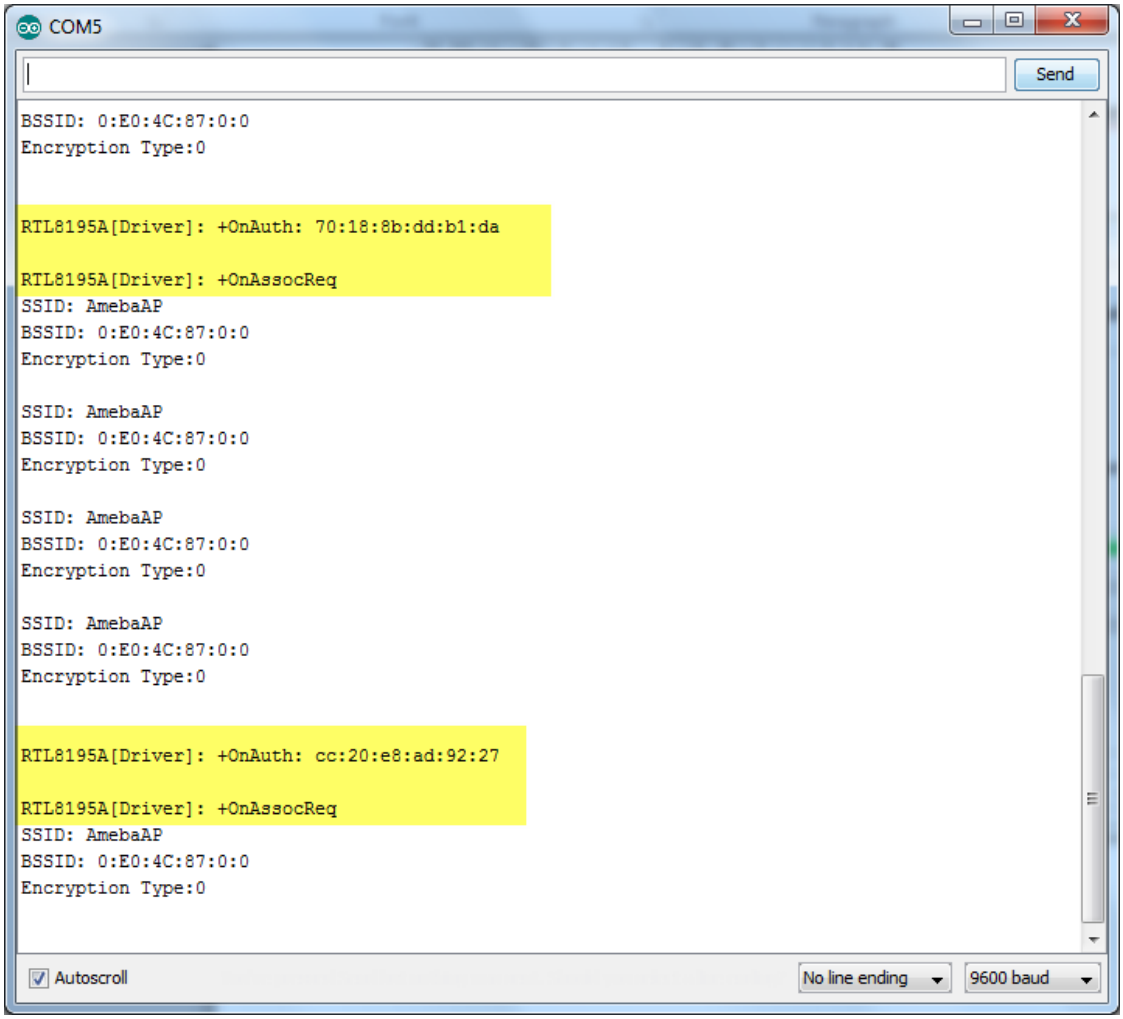
If you want to turn on the AP mode in open mode, please modify the code to

```
status = WiFi.apbegin(ssid, channel);
```

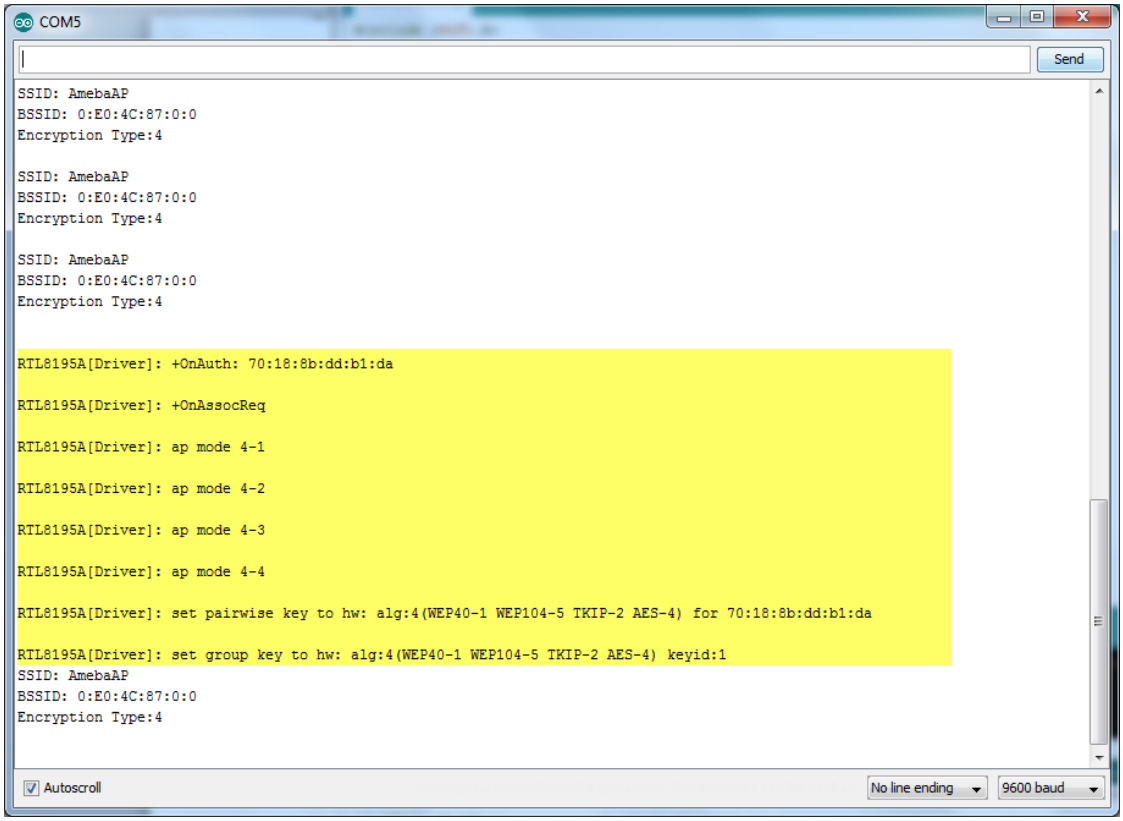
Then upload the sample code and press reset, and you can see related information shown in serial monitor.



In the figure below, we show the messages shown in serial monitor when two stations connect to Ameba AP in open mode:



In the figure below, we show the messages shown in serial monitor when a station connects to Ameba AP in security mode:



WiFi - Set up SSL Client for HTTPS Communication

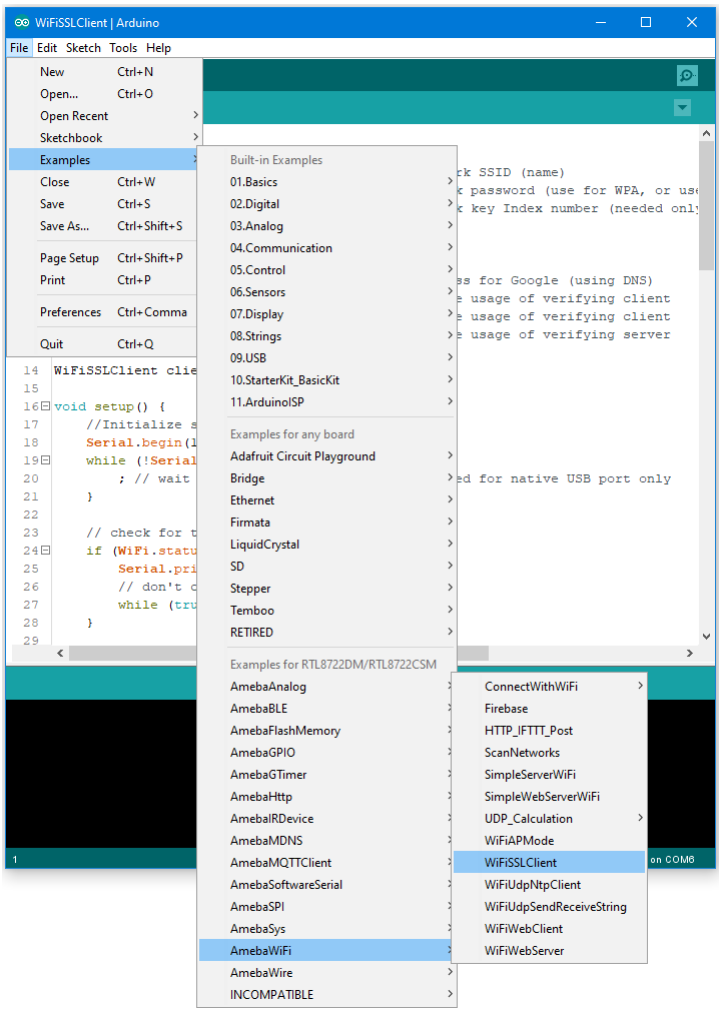
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

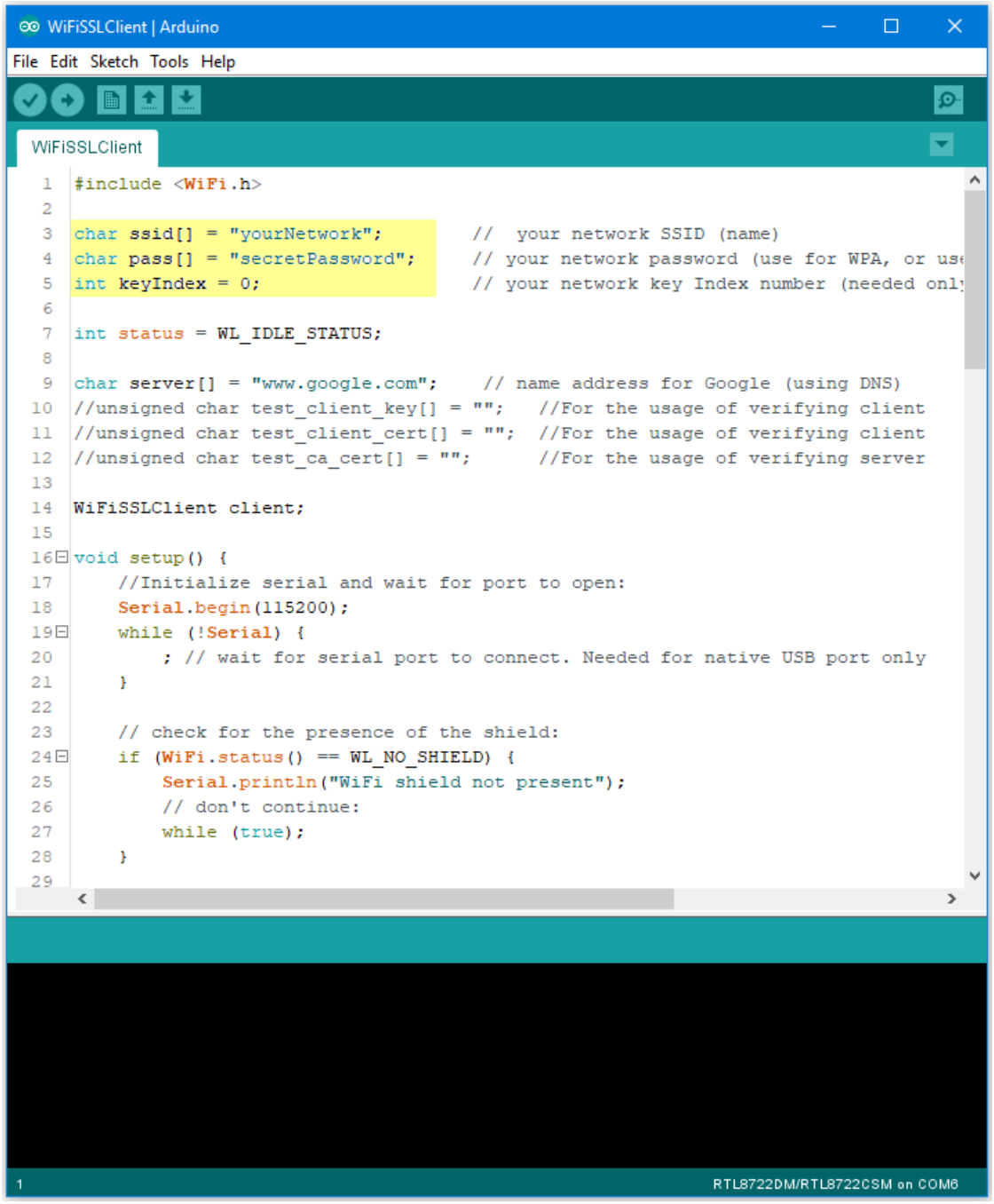
Example

This example uses Ameba to securely retrieve information from the internet using SSL. SSL is an acronym for Secure Sockets Layer. It is a cryptographic protocol designed to provide communications security over a computer network, by encrypting the messages passed between server and client.

Open the "WiFiSSLClient" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiSSLClient`".

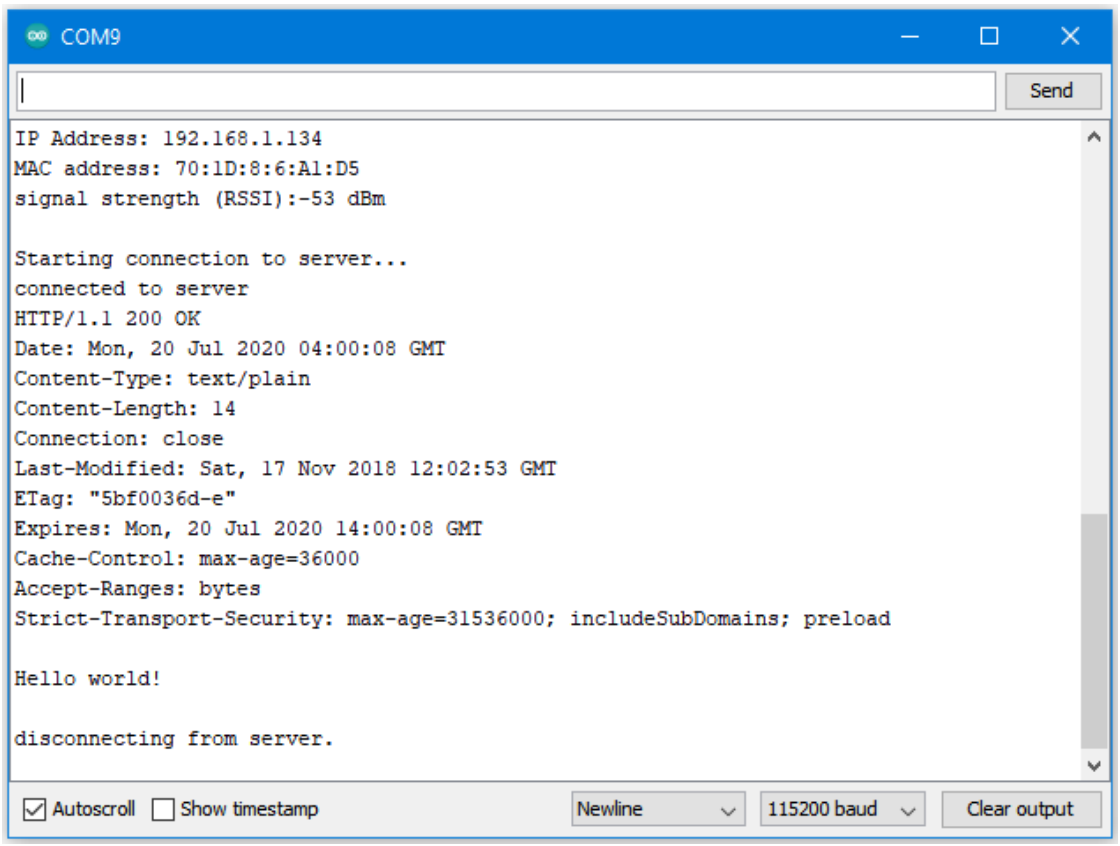


In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor in the Arduino IDE and observe as Ameba retrieves a text file from os.mbed.com.

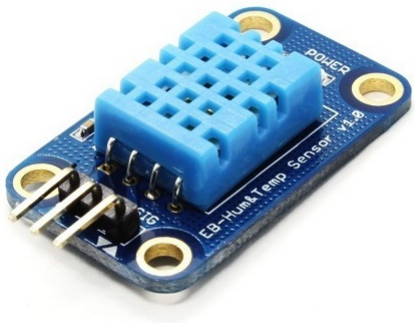
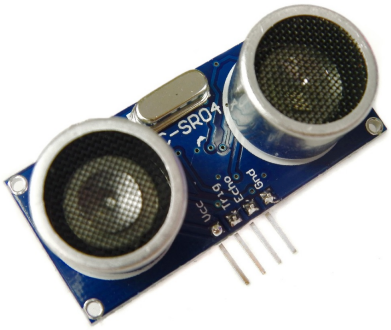
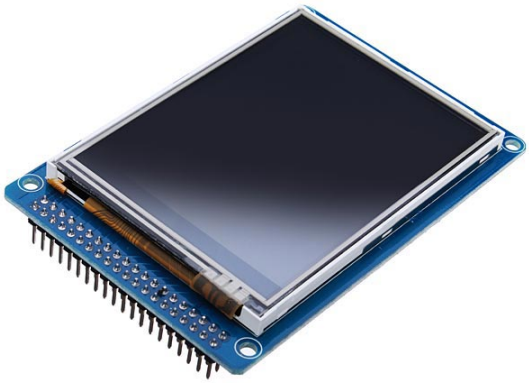
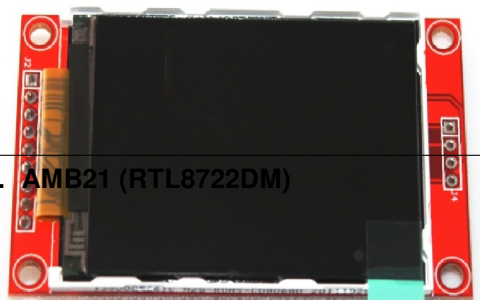
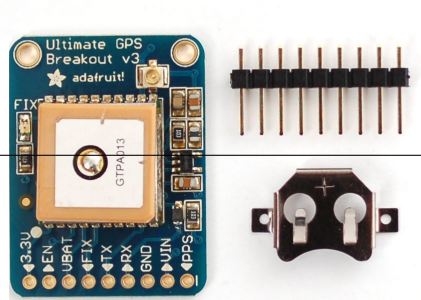


Code Reference

Use “WiFiSSLClient client;” to create a client that uses SSL. After creation, the client can be used in the same way as a regular client.

Components Used

Table 2: Components used in Examples

	
DHT11_DHT22 Humidity & temperature sensor	HC_SR04 Distance measurement function
	
ILI9341 TFT LCD TFT LCD display with SPI interface	PMS3003/5003 Air quality sensor that detects concentration of micro particulate matters
	

Peripheral Examples

AmebaMotors - Use Ameba as Server to Control Motors

Introduction to AmebaMotors

AmebaMotors is a library which provides API related to controlling motors. Please download the library: AmebaMotors
And add the library to Ameba: https://www.arduino.cc/en/Guide/Libraries#toc4

Materials

- AmebaD [AMB21 / AMB22] x 1
- L298N H-Bridge x 1
- 4-wheel motorcar or 2-wheel motorcar+Universal wheel

Example

Procedure

In this example, we connect Ameba to WiFi and use Ameba as server, the user can control a 4-wheel/2-wheel motorcar through a webpage.

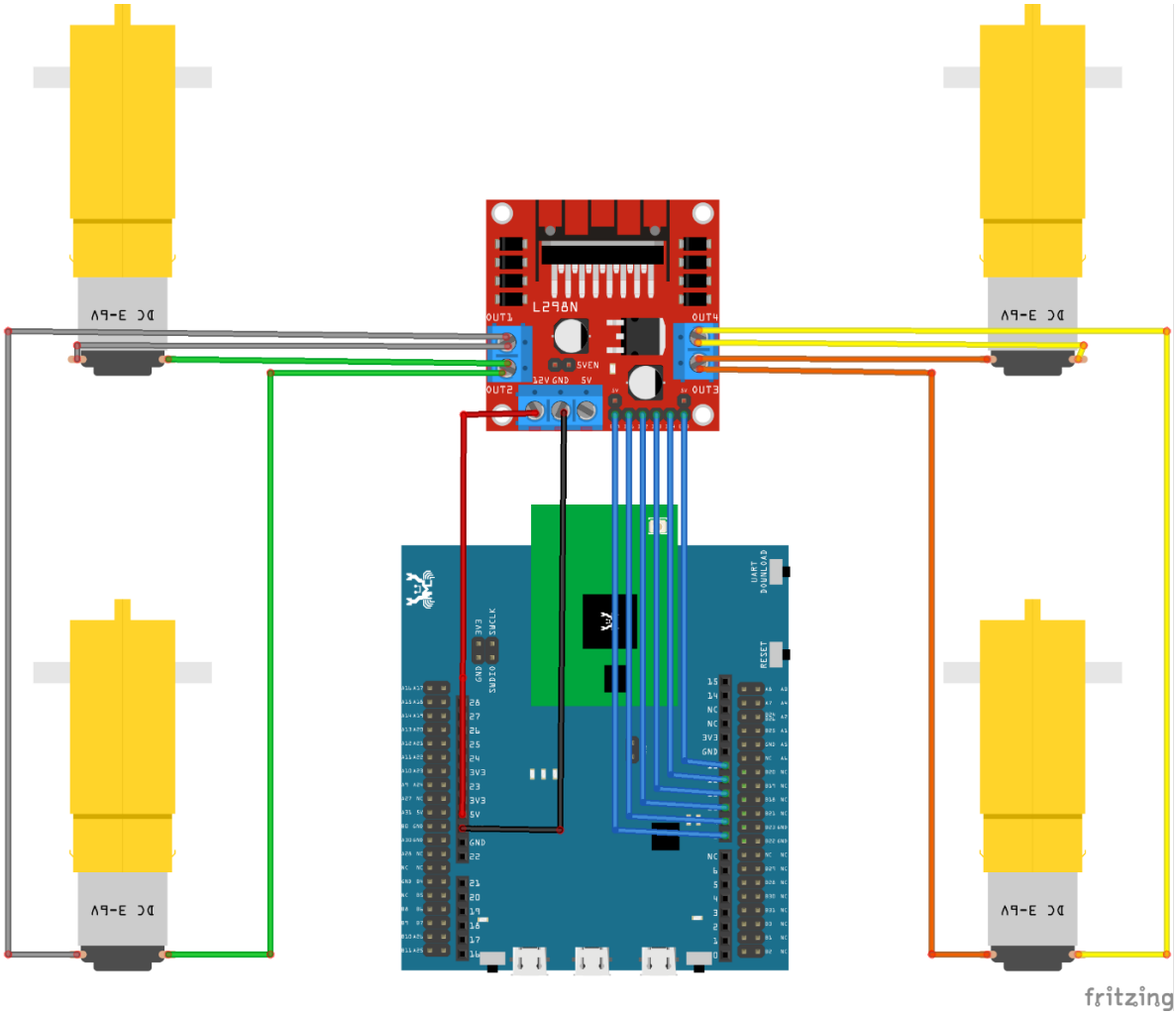
First, connect Ameba to the L298N H-Bridge and the motorcar.

To know more about motor movement and the technical details of the L298N H-Bridge, please check out this *link <https://www.amebaiot.com/en/ameba-arduino-amebamotors-basic/>*.
Open the example, "`Files` -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiControlCar`".

You will see we use the following pins in the example:

ENA	IN1	IN2	IN3	IN4	ENB
8	9	10	11	12	13

Wiring:



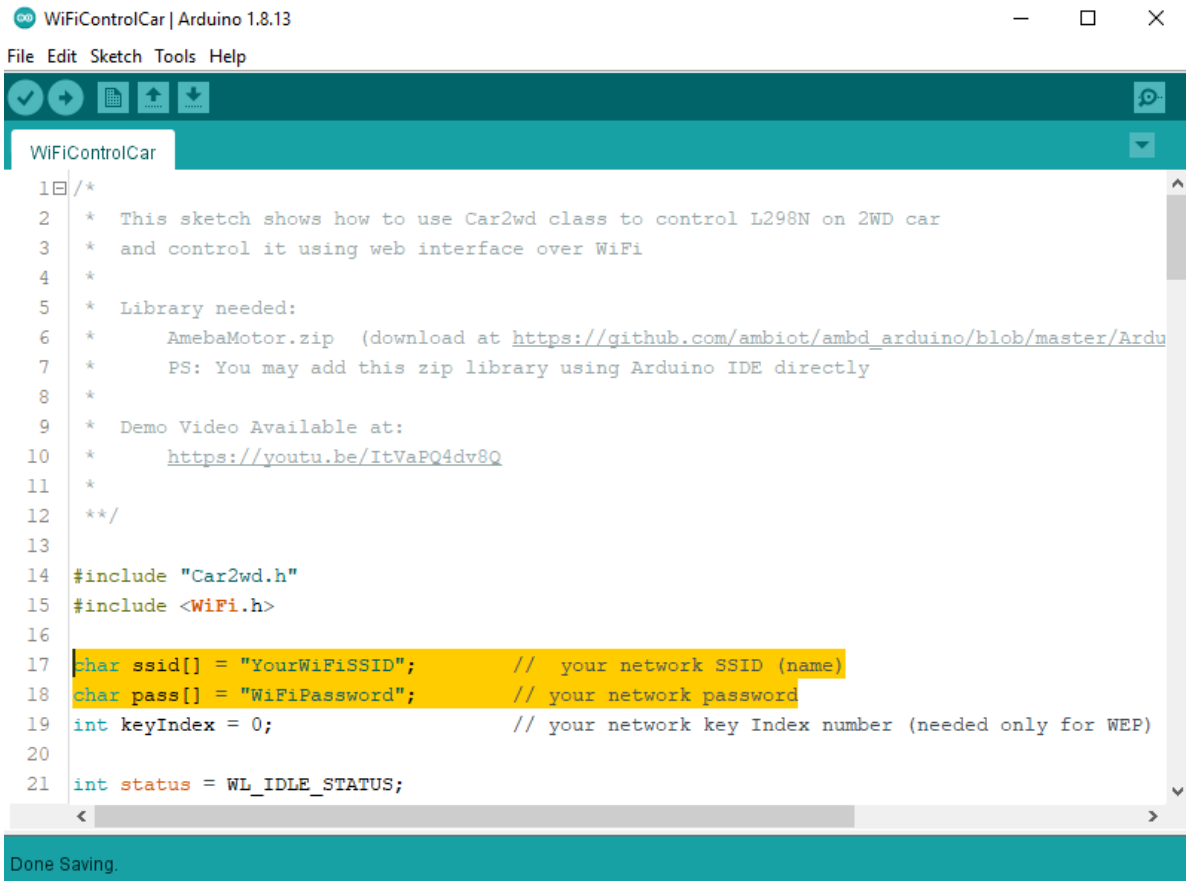
Note:

- We connect Ameba 5V to L298N +12V to supply power. However, not every L298N accepts 5V power supply, if this does not work, please connect L298N +12V to other power supply (e.g., +12V) and use L298N +5V to supply power to Ameba.
- The correct wiring of the motor depends on each model (may be opposite). Please run the test program first, make sure it runs correctly before assembling the motorcar.
- For convenience purposes, it's recommended to use Dupont line to organize the wiring of motors and L298N.

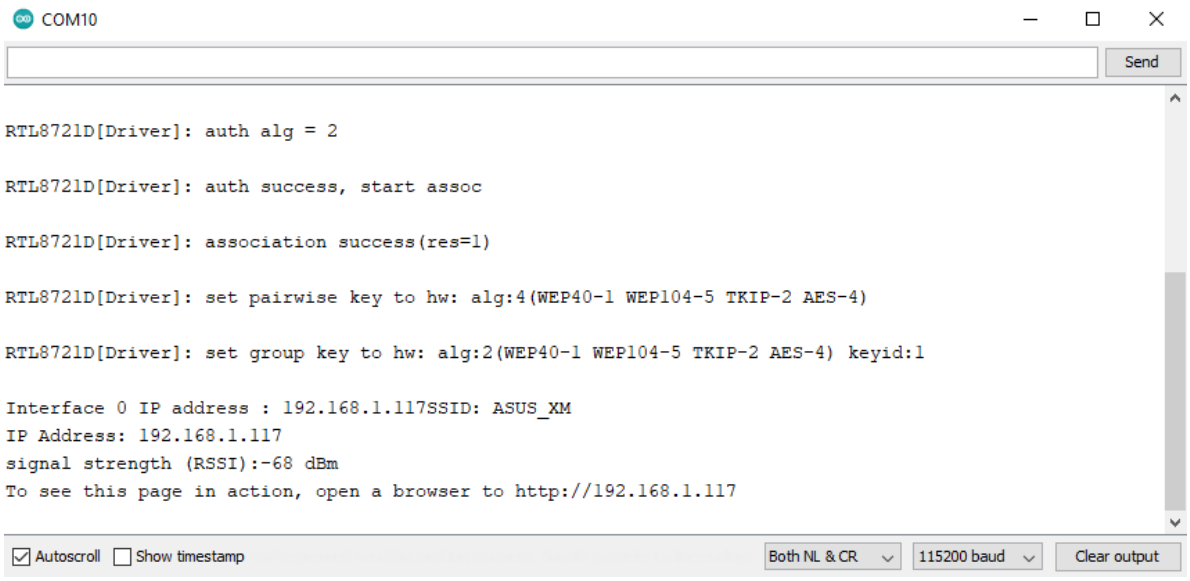
Every time you modify your program, please remember to unplug the power of L298N to avoid the motor running unexpectedly. Connect Ameba to power, upload the program, and then connect L298N to power when you are going to test the program.

Then, upload the code to Ameba

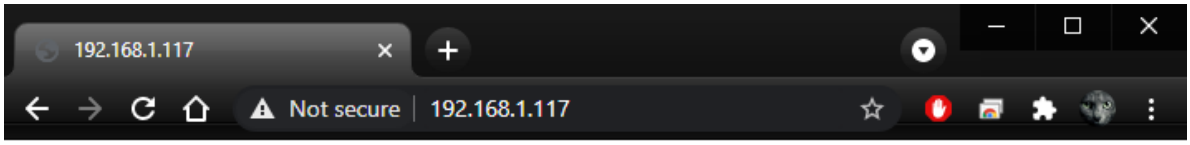
In the sample code, modify the highlighted snippet to corresponding information.



Upload the code and press the reset button on Ameba. When the connection is established, you will see the message "To see this page in action, open a browser to http://xxx.xxx.xxx.xxx" in the Arduino IDE, as shown in the figure:



Next, open the browser of a computer or a cell phone under the same WiFi domain, enter the address in the message.



Click here to move Forward
Click here to move Backward
Click here to turn Right
Click here to turn Left

In the webpage, you can press the corresponding button to control the motor car in any of the 4 directions.

Demo Video

Code Reference

Use `WiFi.begin()` to establish WiFi connection. https://www.arduino.cc/en/Reference/WiFiBegin

To get the information of a WiFi connection:

Use `WiFi.SSID()` to get SSID of the current connected network. https://www.arduino.cc/en/Reference/WiFiSSID

Use `WiFi.RSSI()` to get the signal strength of the connection. https://www.arduino.cc/en/Reference/WiFiRSSI

Use `WiFi.localIP()` to get the IP address of Ameba. https://www.arduino.cc/en/Reference/WiFiLocalIP

Use `WiFiServer server()` to create a server that listens on the specified port. https://www.arduino.cc/en/Reference/WiFiServer

Use `server.begin()` to tell the server to begin listening for incoming connections. https://www.arduino.cc/en/Reference/WiFiServerBegin

Use `server.available()` to get a client that is connected to the server and has data available for reading. https://www.arduino.cc/en/Reference/WiFiServerAvailable

Use `client.connected()` to get whether or not the client is connected. https://www.arduino.cc/en/Reference/WiFiClientConnected

Use `client.println()` to print data followed by a carriage return and newline. https://www.arduino.cc/en/Reference/WiFiClientPrintln

Use `client.print()` to print data to the server that a client is connected to. https://www.arduino.cc/en/Reference/WiFiClientPrint

Use `client.available()` to return the number of bytes available for reading. https://www.arduino.cc/en/Reference/WiFiClientAvailable

Use `client.read()` to read the next byte received from the server the client is connected to. https://www.arduino.cc/en/Reference/WiFiClientRead

Use `client.stop()` to disconnect from the server the client is connected to. https://www.arduino.cc/en/Reference/WiFIClientStop

Audio Codec – Basic Input Output

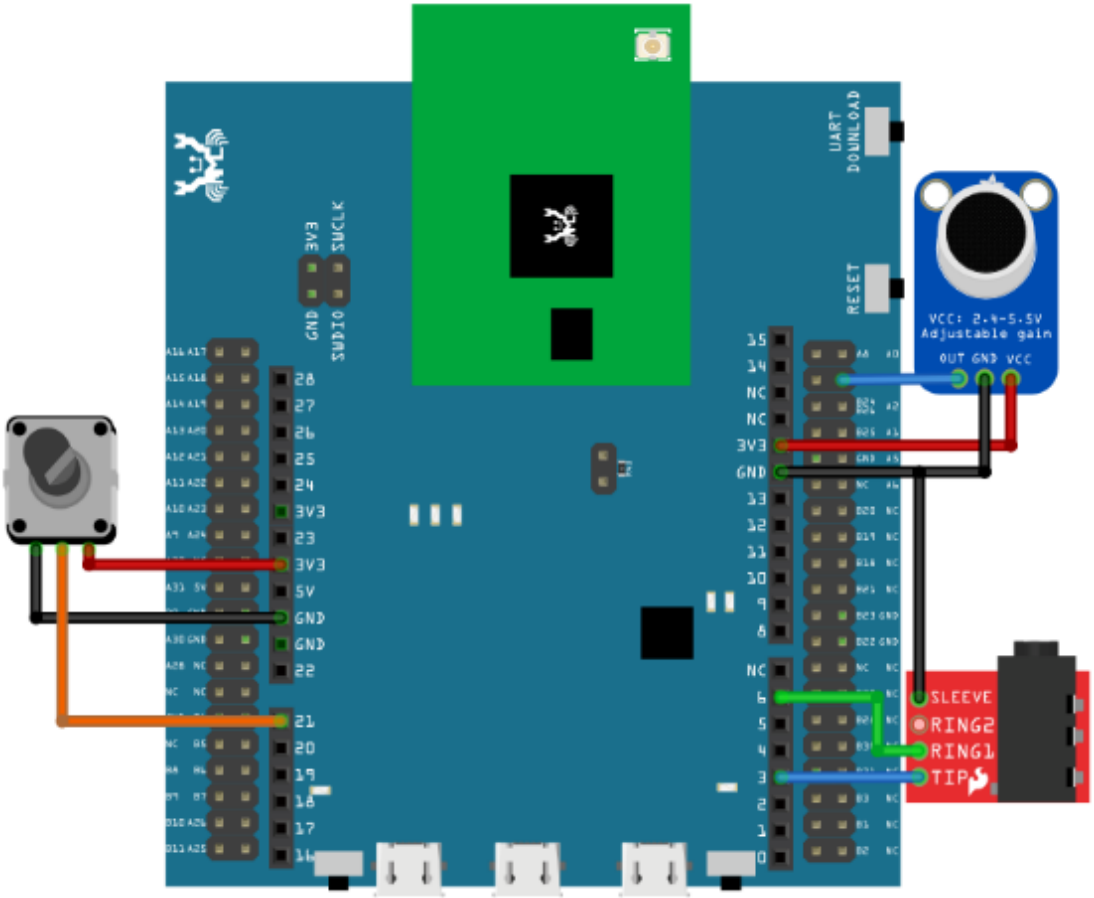
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Potentiometer x 1
- Analog microphone x 1 (e.g., Adafruit 1063 / 1064)
- 3.5mm TRS/TRRS breakout x 1 (e.g., Adafruit 2791 / Sparkfun 11570)

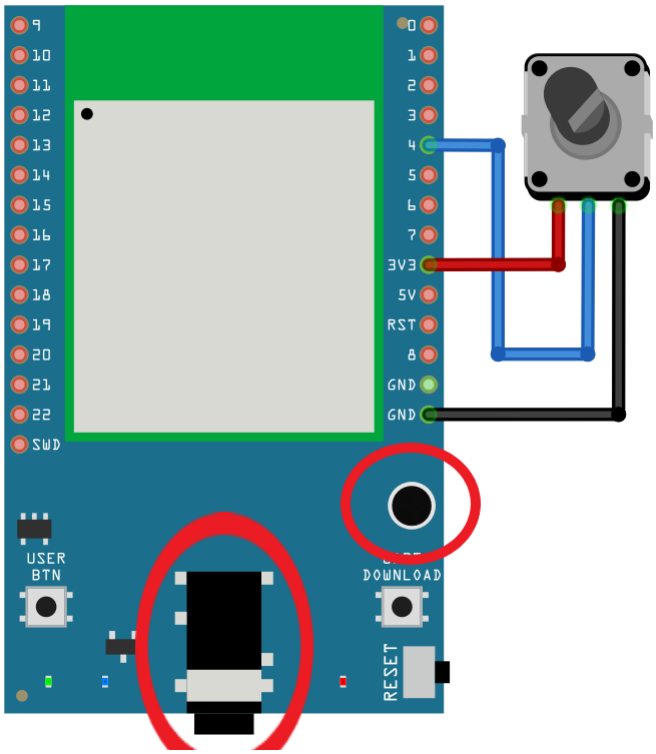
Example

Procedure
Connect the potentiometer, microphone and 3.5mm connector to the RTL8722 board following the diagram.

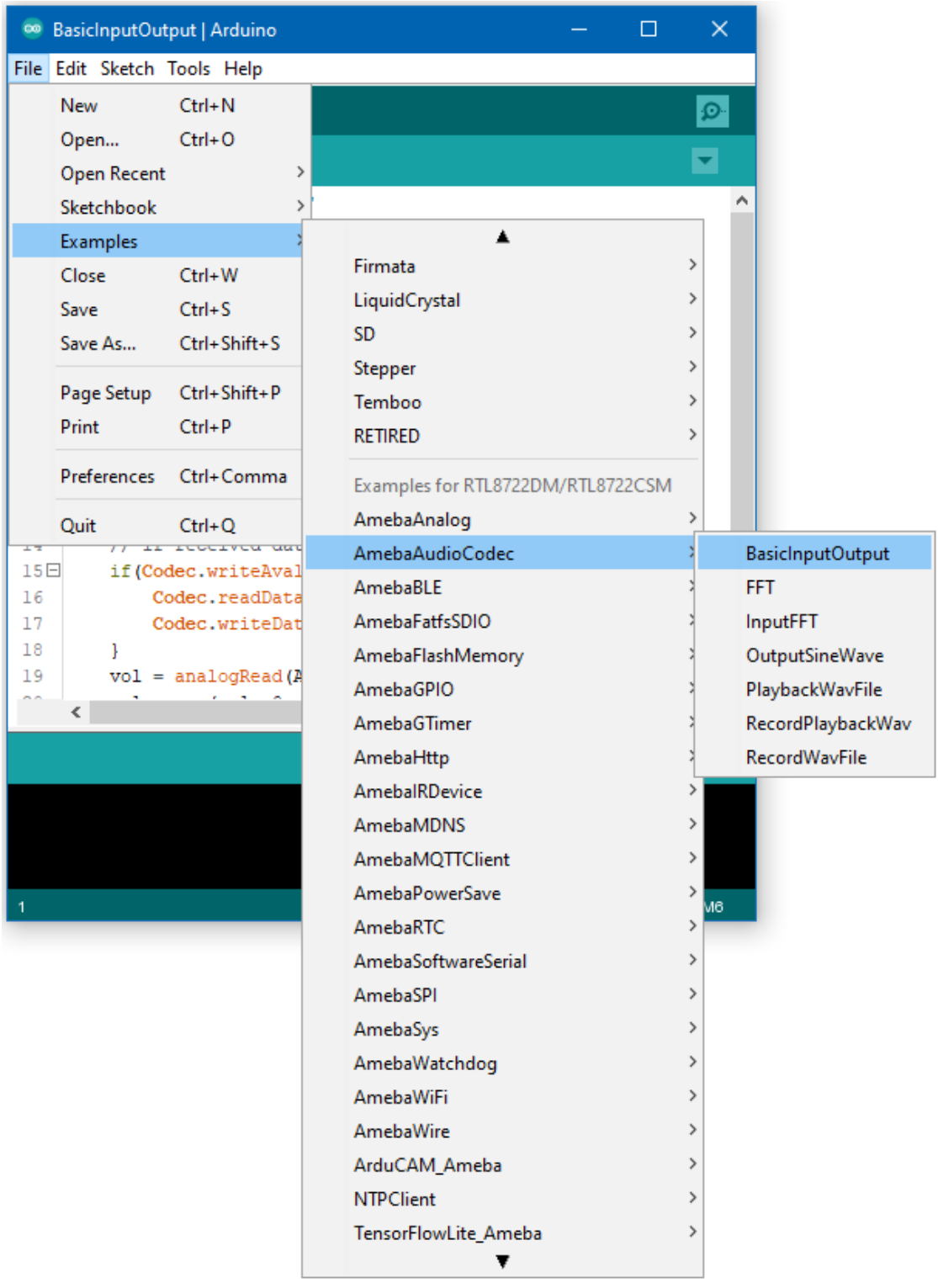
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



Open the example, `"Files" -> "Examples" -> “AmebaAudioCodec” -> “BasicInputOutput”`.



Upload the code and press the reset button on Ameba once the upload is finished.

Connect a pair of wired headphones to the 3.5mm audio jack, blow at the microphone, and you should hear the sounds picked-up by the microphone replayed in the headphones. Adjust the potentiometer and the output volume will change as well. Note: if you are using a microphone with an amplifier included, such as Adafruit 1063, the amplifier can lead to the microphone picking up more noise.

Audio Codec - FFT

Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1

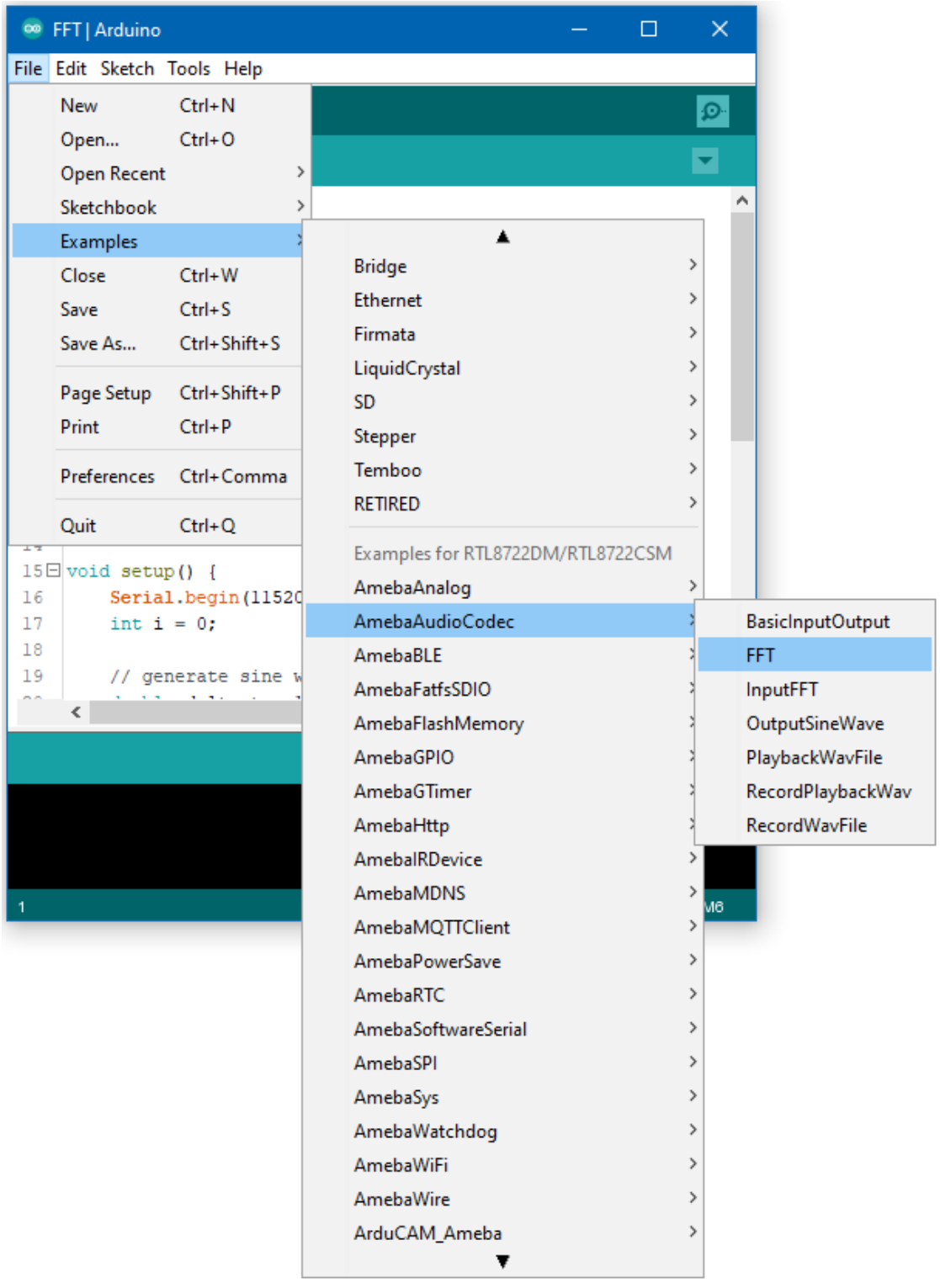
Example

Introduction

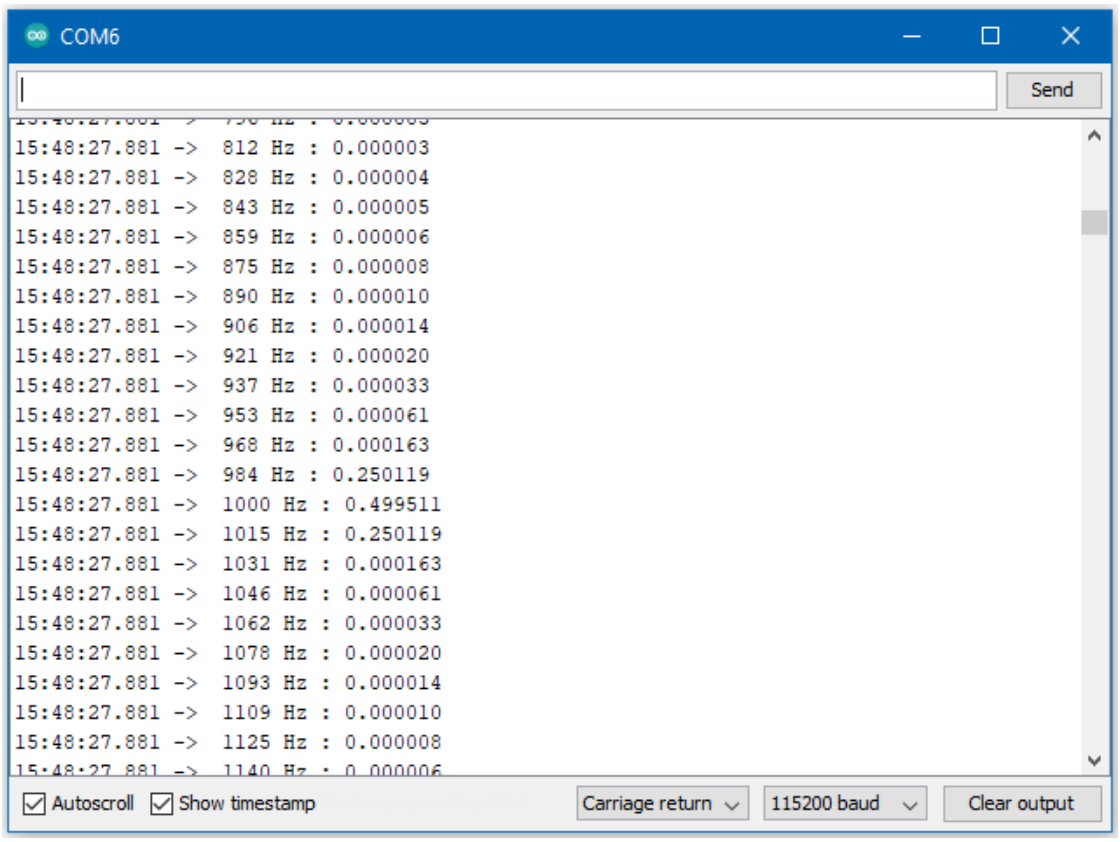
This example shows how to use the AudioCodec_FFT class to calculate the fast Fourier transform of a signal to extract the frequencies present in the signal.

Procedure

Open the example, `"Files" -> "Examples" -> “AmebaAudioCodec” -> “FFT”`.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor, and the output results of the AudioCodec_FFT calculation will be displayed.



Audio Codec - Input FFT

Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Analog microphone x 1 (e.g., Adafruit 1063 / 1064)

Example

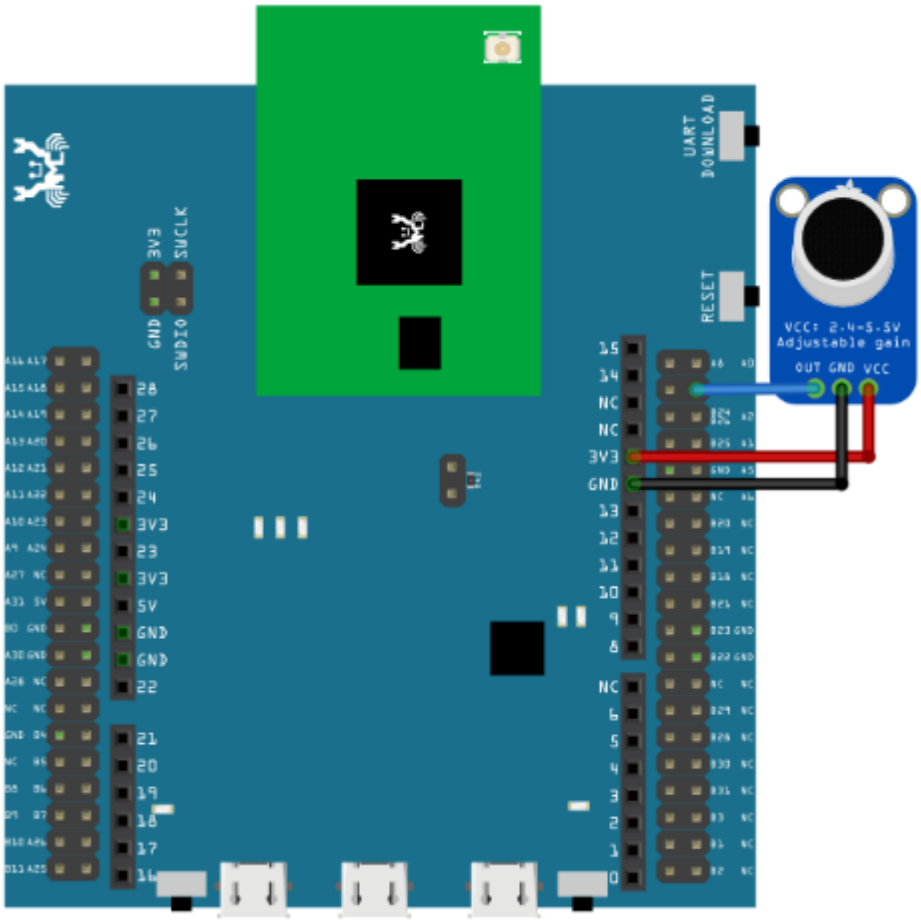
Introduction

This example shows how to use the FFT class to calculate the fast Fourier transform of the audio signal recorded by the microphone.

Procedure

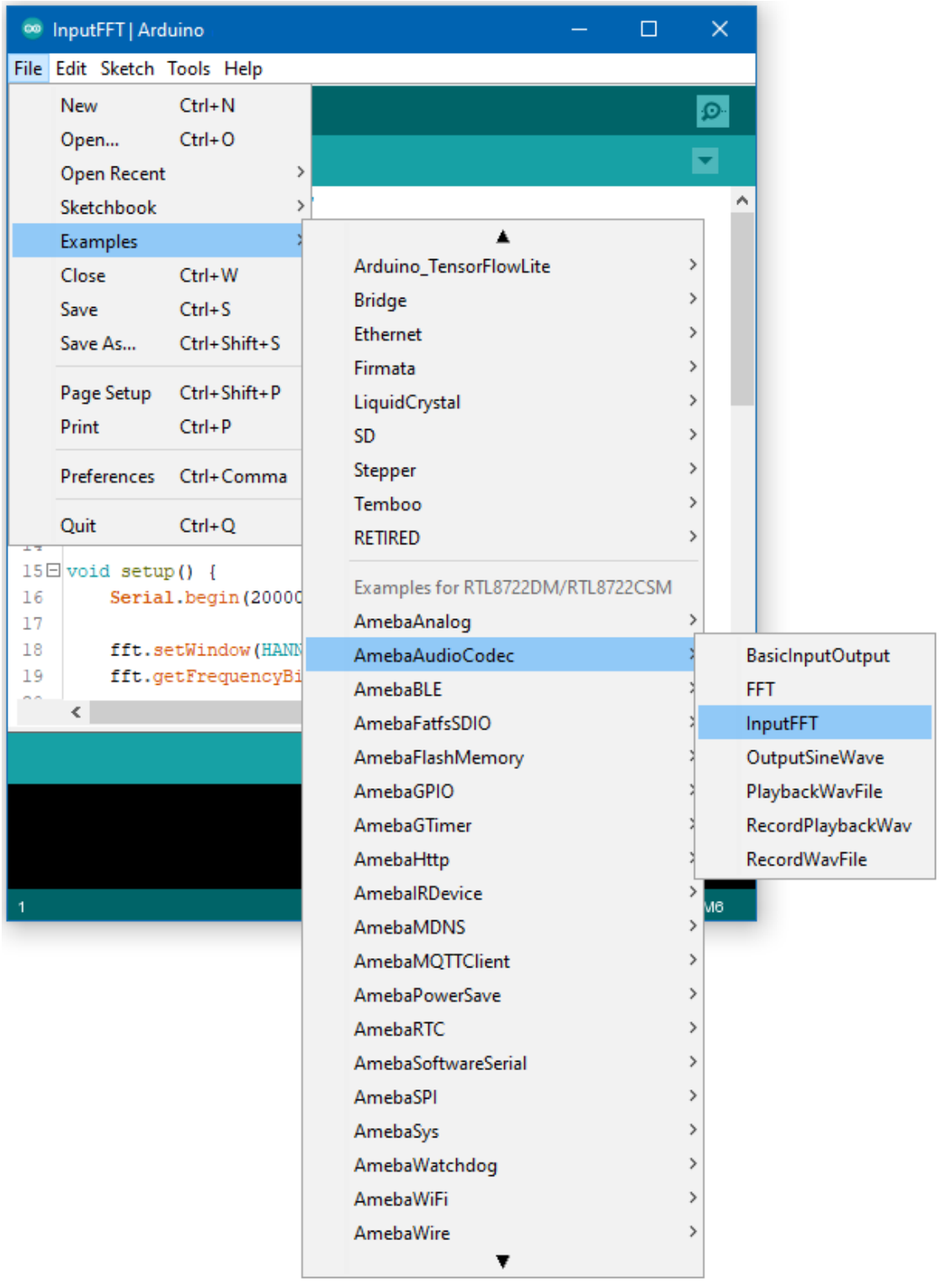
Connect the microphone to the RTL8722 board following the diagram.

AMB21 / AMB22 Wiring Diagram:

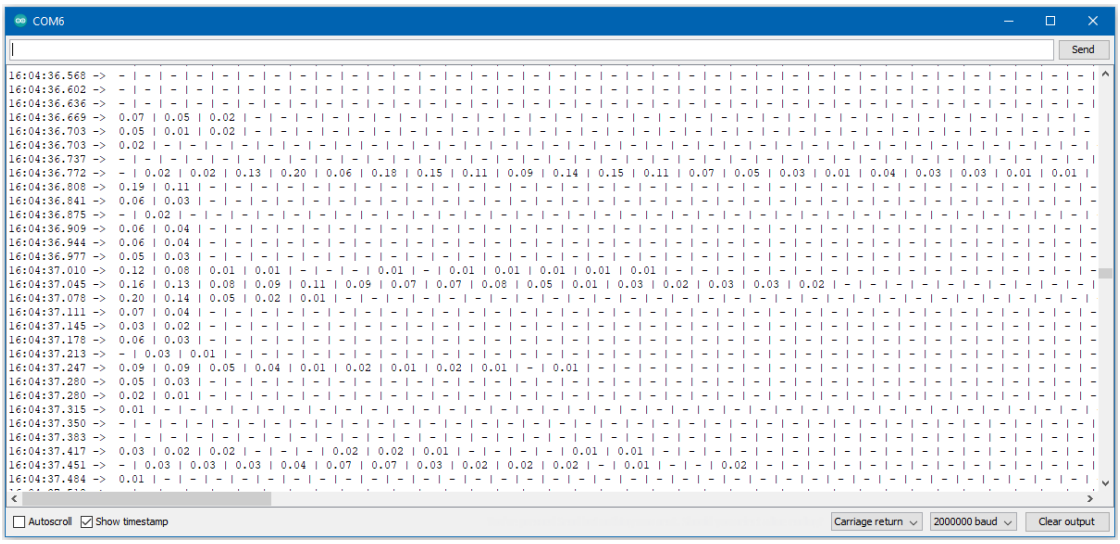


AMB23 Wiring Diagram:
As AMB23 have a built in microphone on the board, there is no need for any external microphone.

Next, open the example, `"Files" -> "Examples" -> “AmebaAudioCodec” -> “InputFFT”`.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor and change the baud rate to 2000000. A stream of FFT results of audio samples will be displayed. Try playing music or use a smartphone app to generate a sine wave into the microphone, and you should be able to see the FFT output change.



Audio Codec – Output Sine Wave

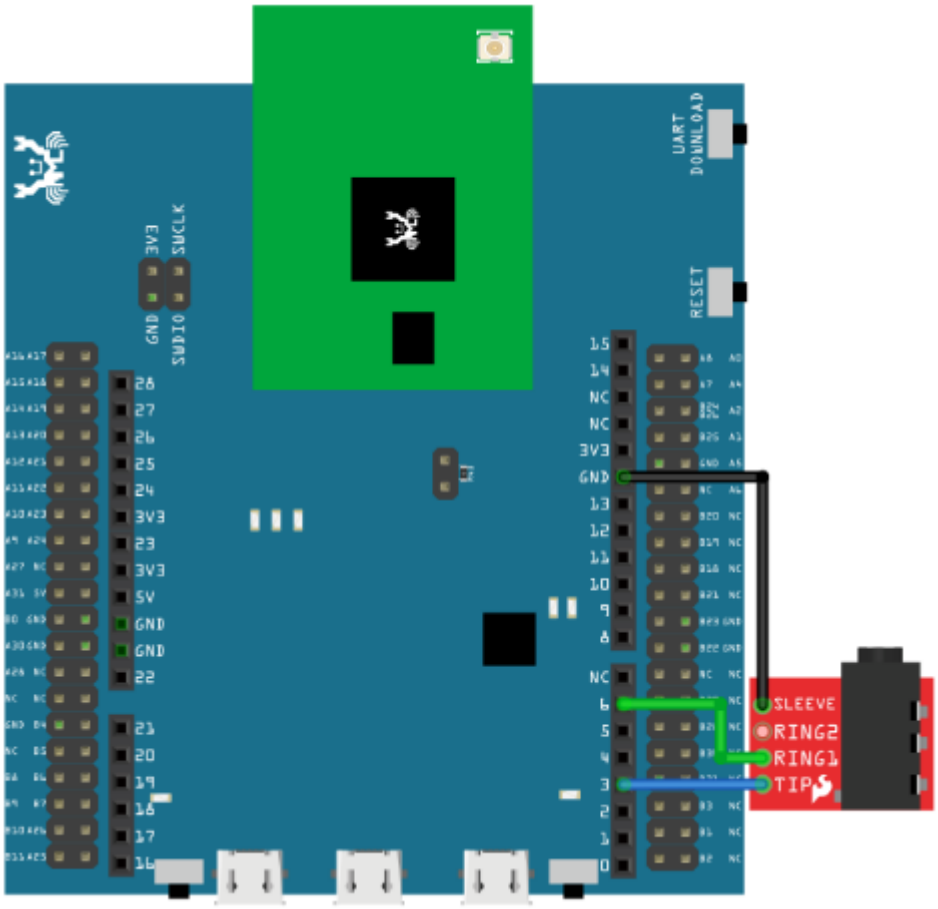
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- 3.5mm TRS/TRRS breakout x 1 (e.g., Adafruit 2791 / Sparkfun 11570)

Example

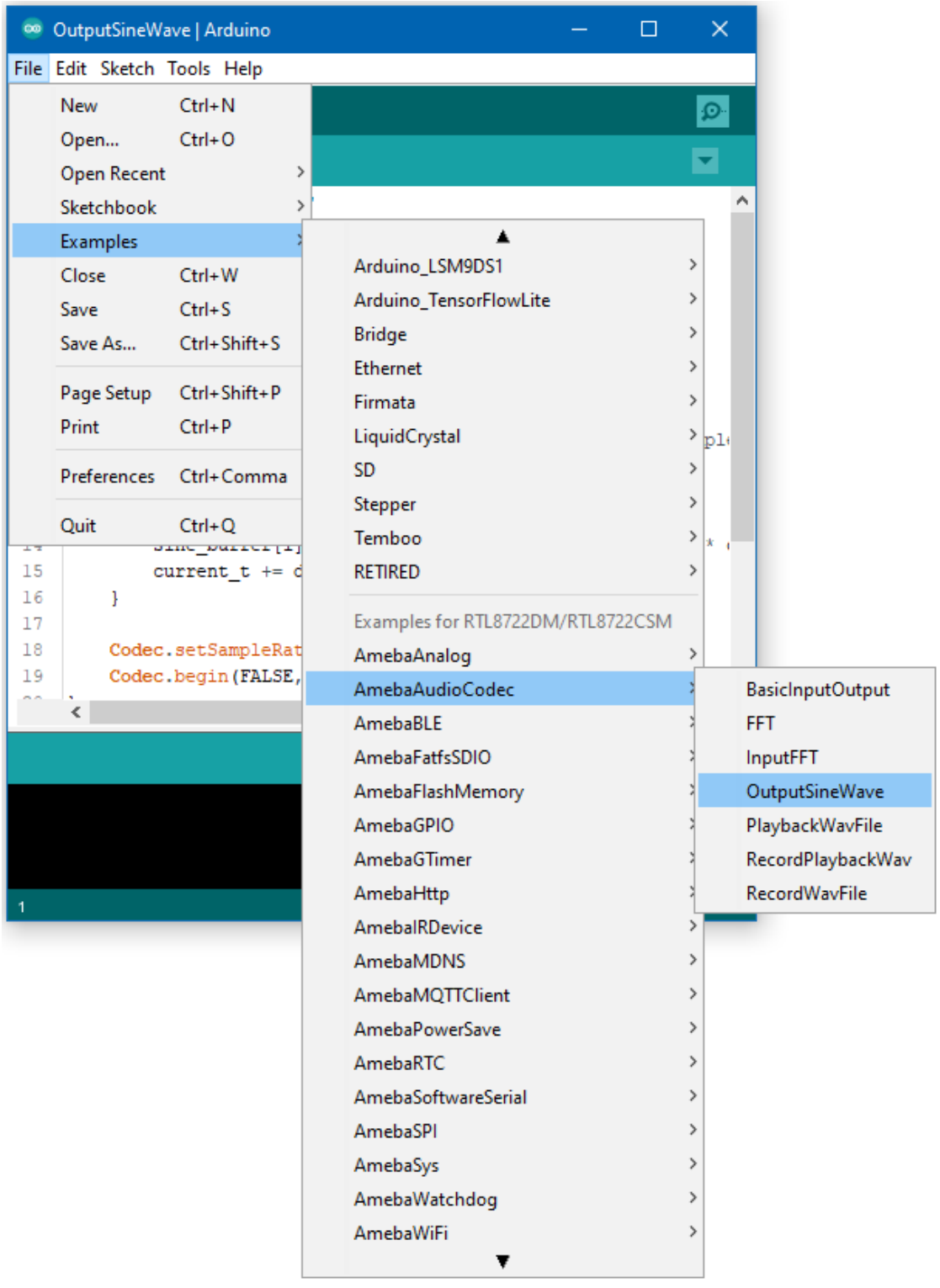
Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the 3.5mm connector to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
As AMB23 have a built in microphone on the board, there is no need for any external microphone.

Open the example, `"Files" -> "Examples" -> "AmebaAudioCodec" -> "OutputSineWave"`.



Upload the code and press the reset button on Ameba once the upload is finished.
Connect a pair of wired headphones to the 3.5mm audio jack and you should hear the generate single sinusoidal tone.

E-Paper - Display Images

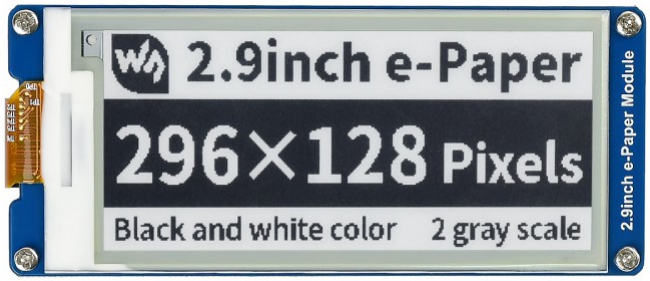
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

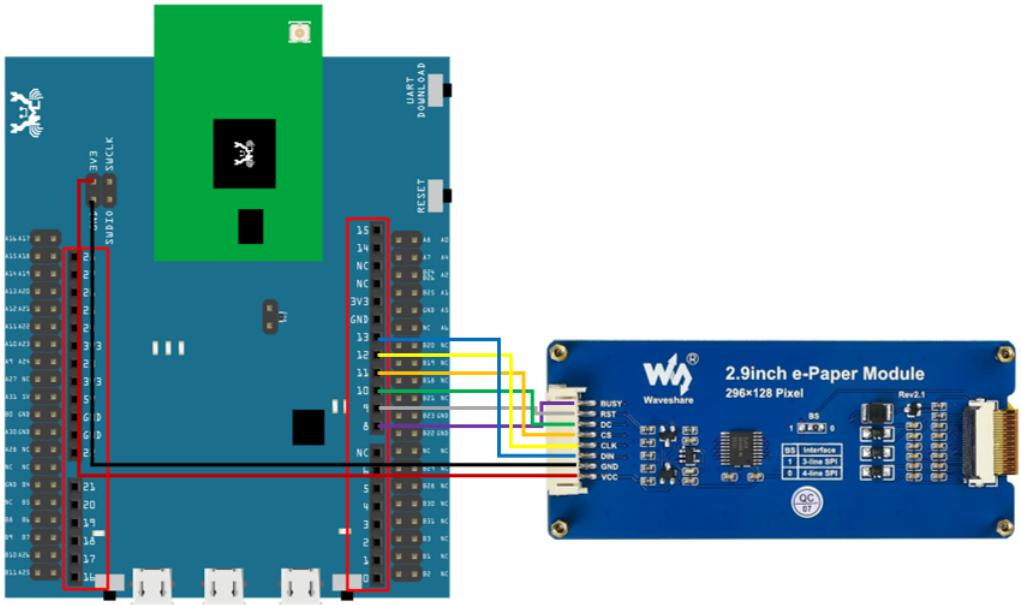
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

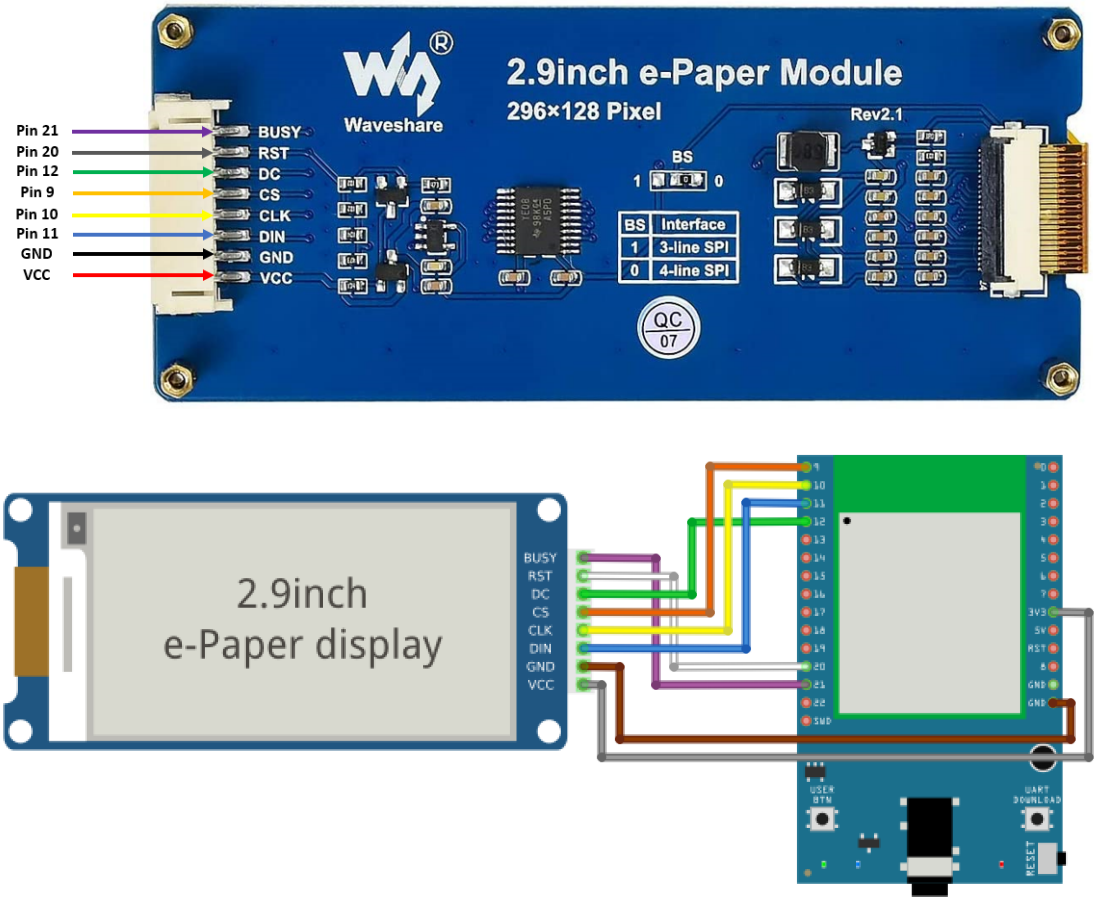
Front view of the e-Paper Module:



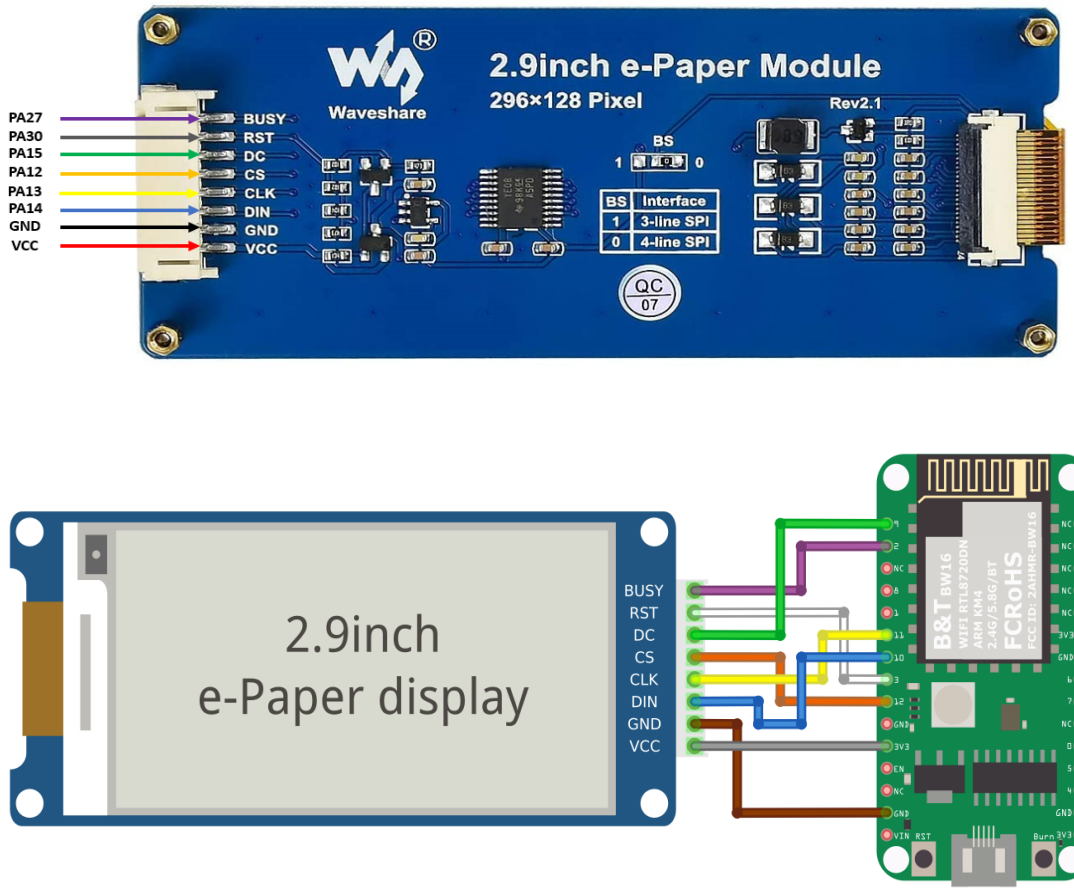
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:

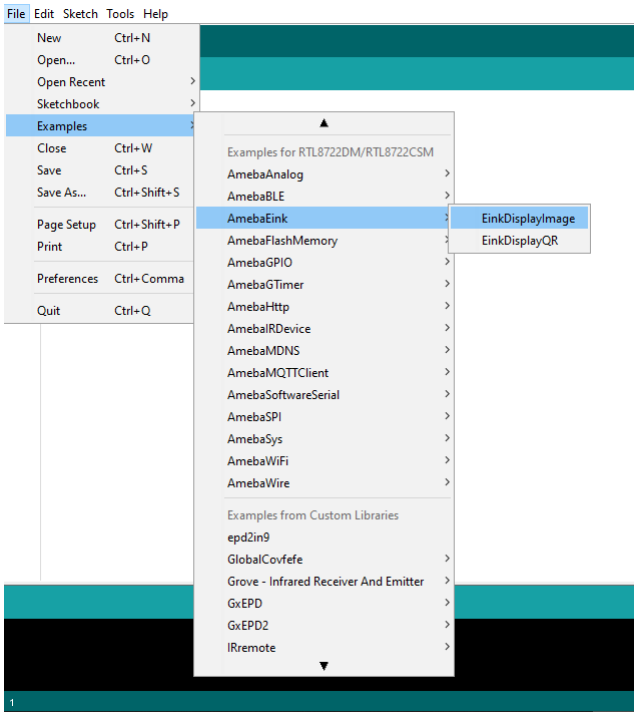


Firstly, you need to prepare a picture/photo in the format of 296×128 pixels. We can easily find a photo resizing tool online, for example, the Online Image Resizer.

Following the instructions on the website, then download the generated image in JPG format.

Secondly, we use the Image2LCD tool to transfer the downloaded 296×128 image into hexadecimal codes. You can visit this YouTube link to get detailed instructions.

Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`EinkDisplayImage`":



Press the reset button after uploading the sample code, you will need to wait for around 1-2 seconds for the e-Paper module to fresh its screen. Then the screen will start to display an image for 5 seconds first, then 3 different QR codes will be displayed every 5 seconds (showing in the screenshot below, you may scan the QR codes and find out more information if you wish to). Lastly, a gif which comes in form of 3 frames will be displayed for a few seconds.

Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html

E-Paper - Display Text

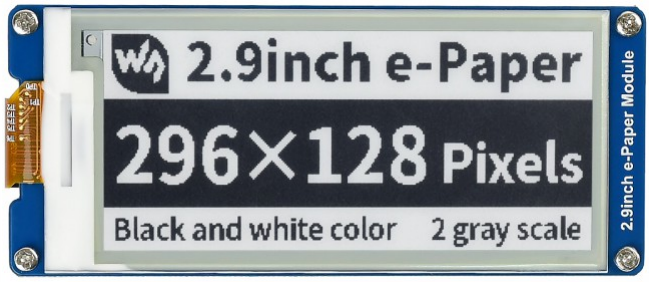
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

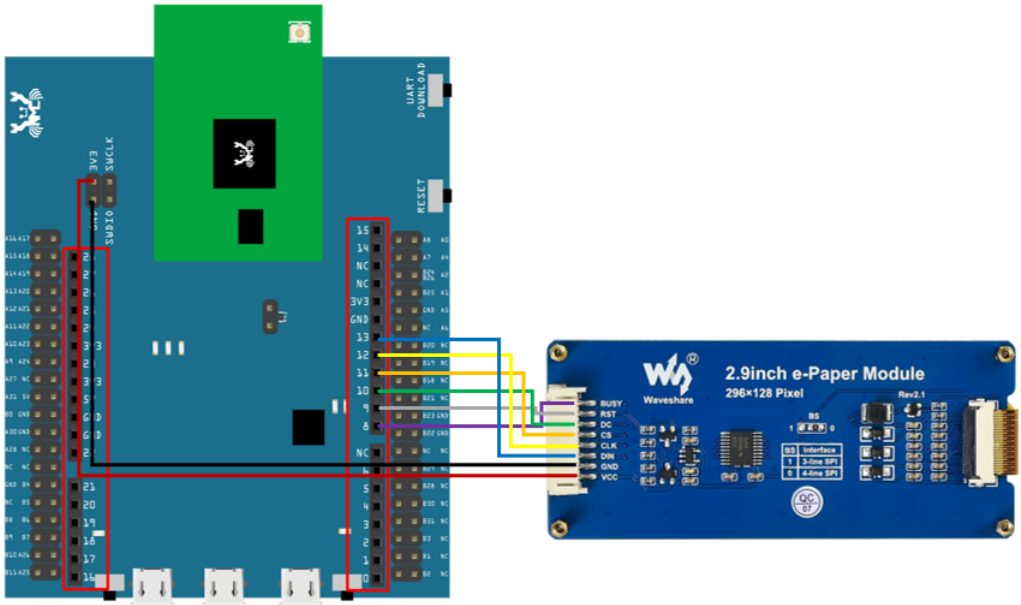
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

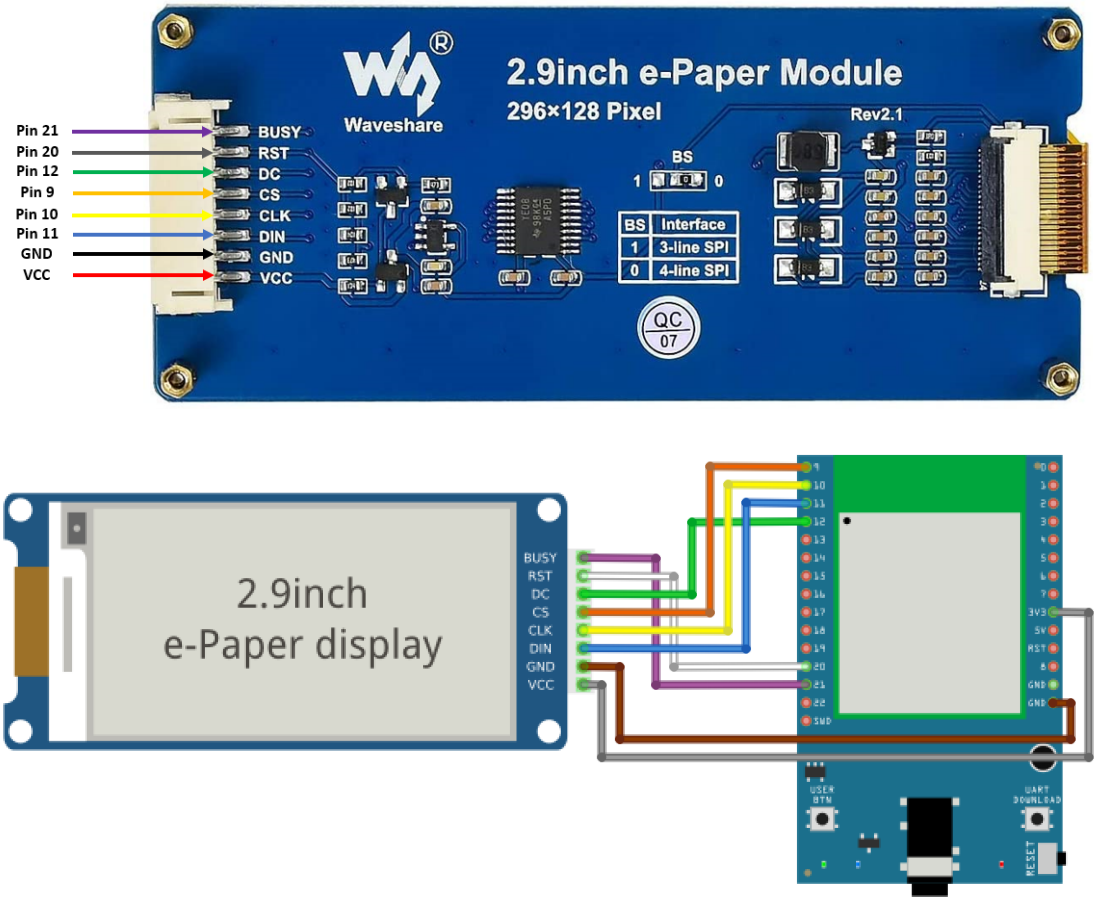
Front view of the e-Paper Module:



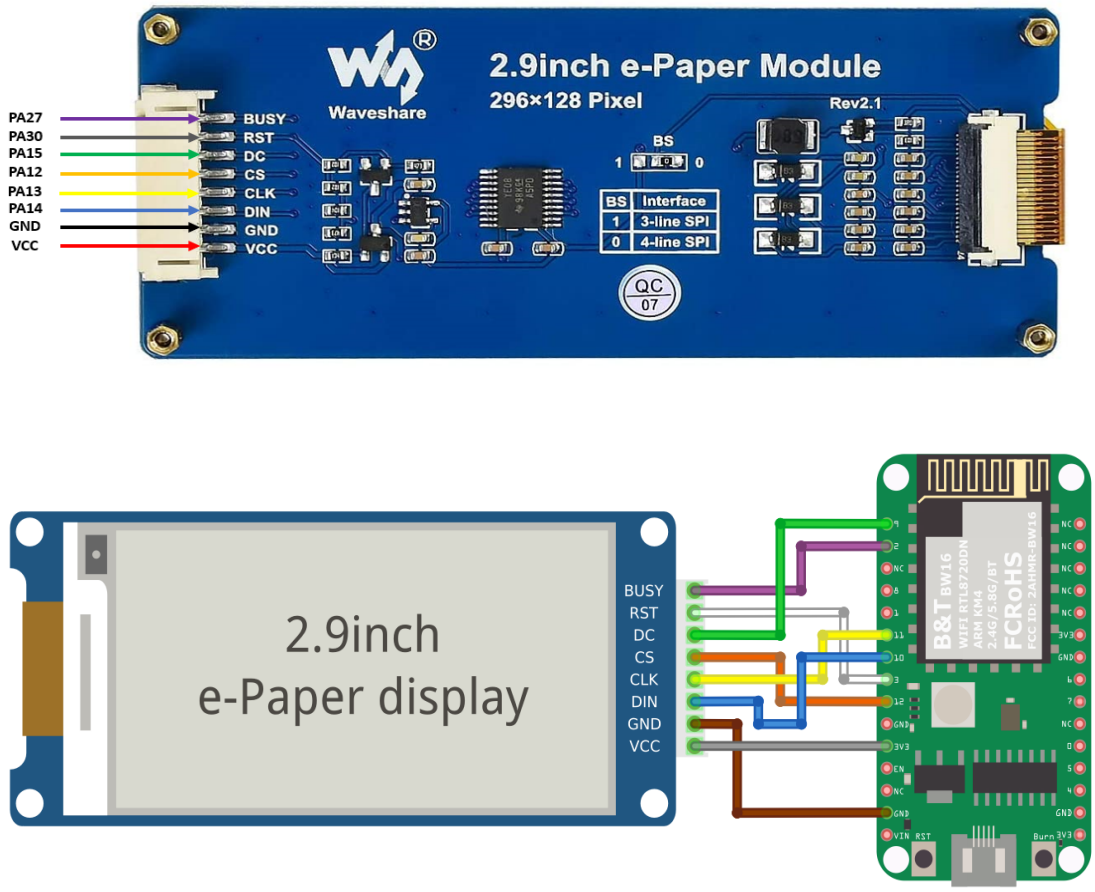
AMB21 / AMB22 Wiring Diagram:



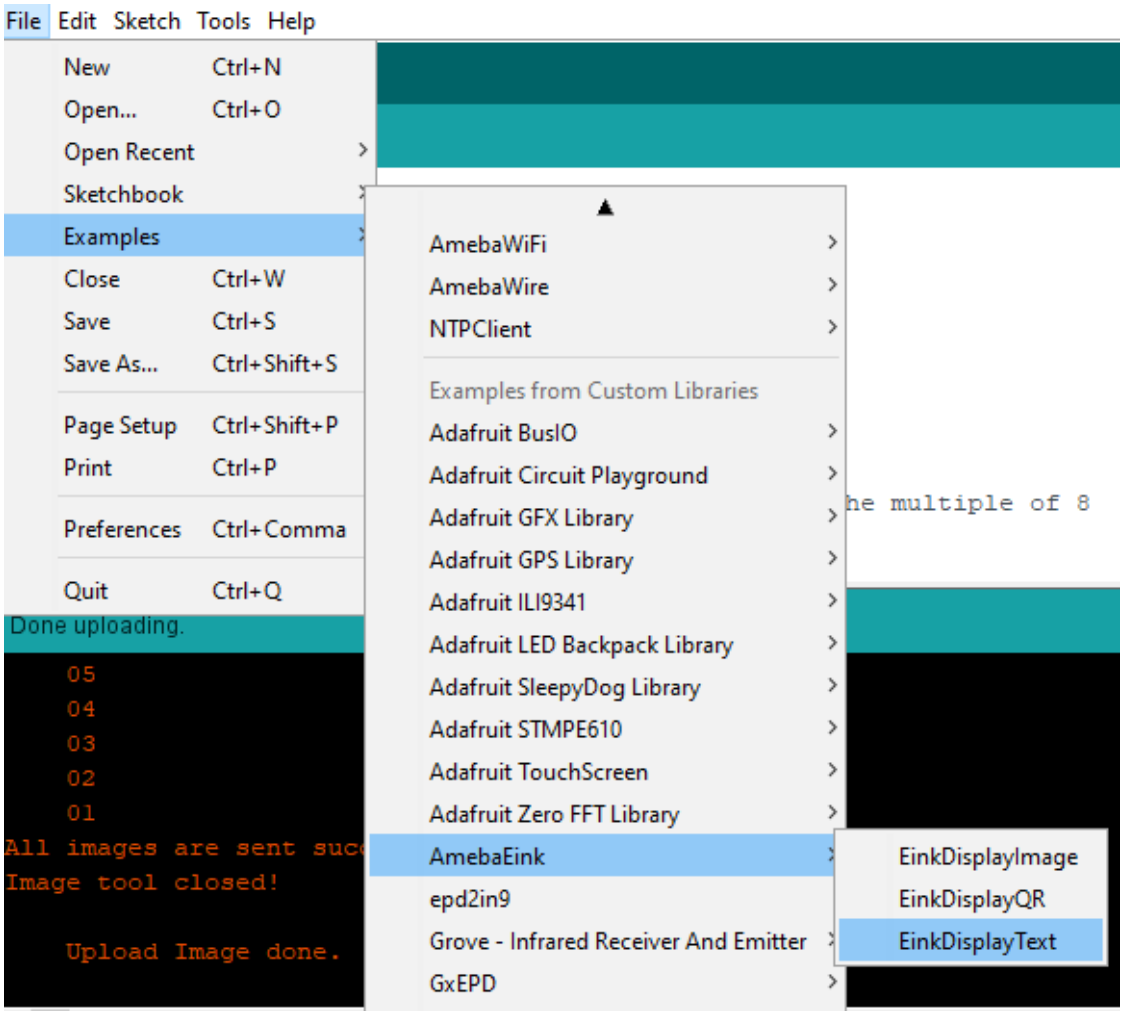
AMB23 Wiring Diagram:



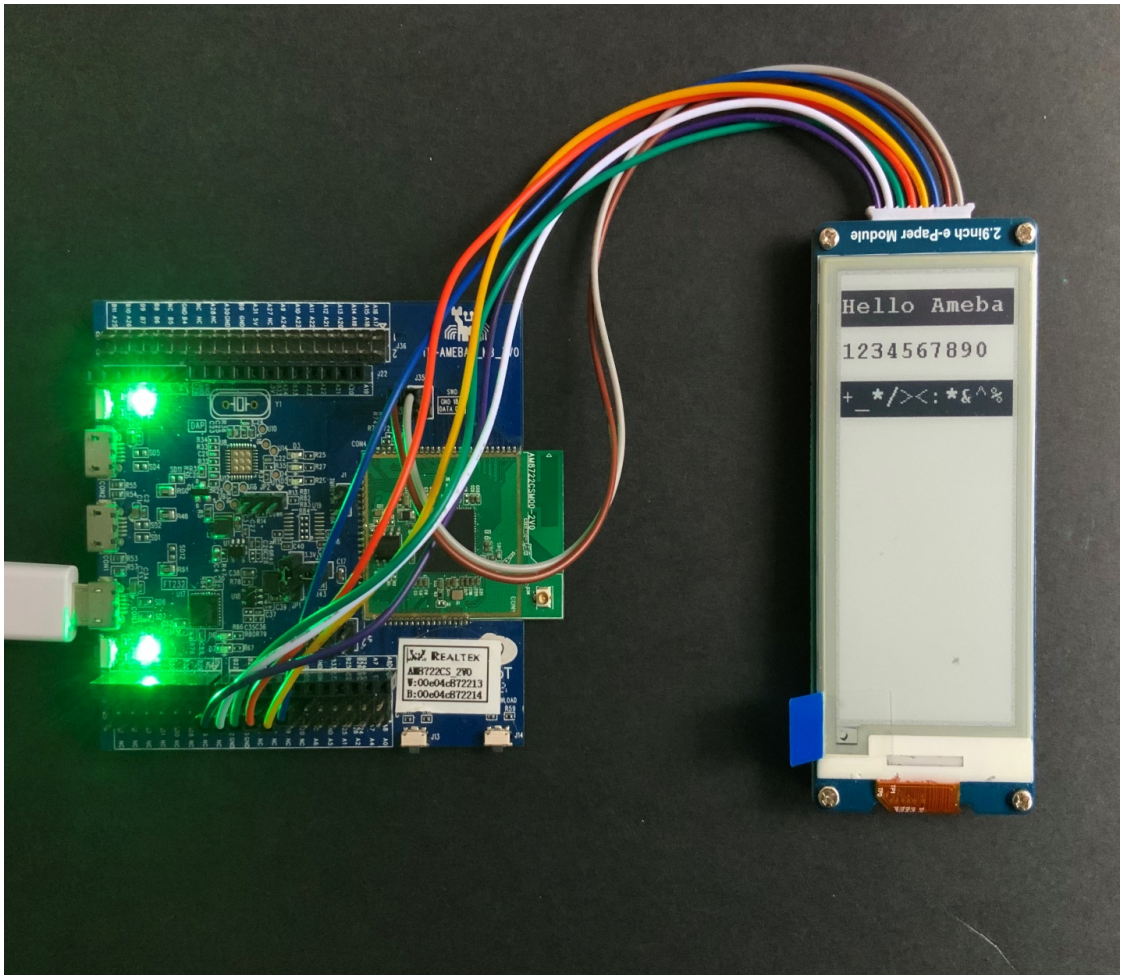
BW16 Wiring Diagram:



Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" -> "`Examples`" -> "`AmebaEink`" -> "`EinkDisplayText`":



Upload the code to the board and press the Reset button after the uploading is done. You will find these texts displayed on the board:



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html

E-Paper - Display User-generated QR Code

Materials

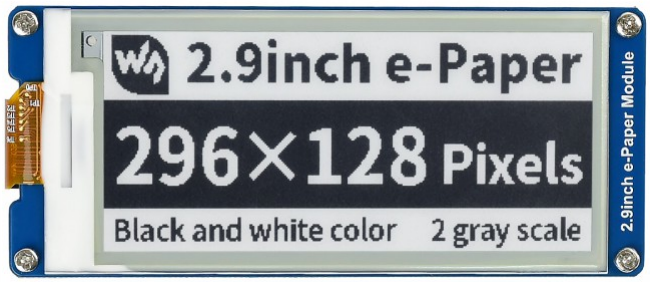
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

Example

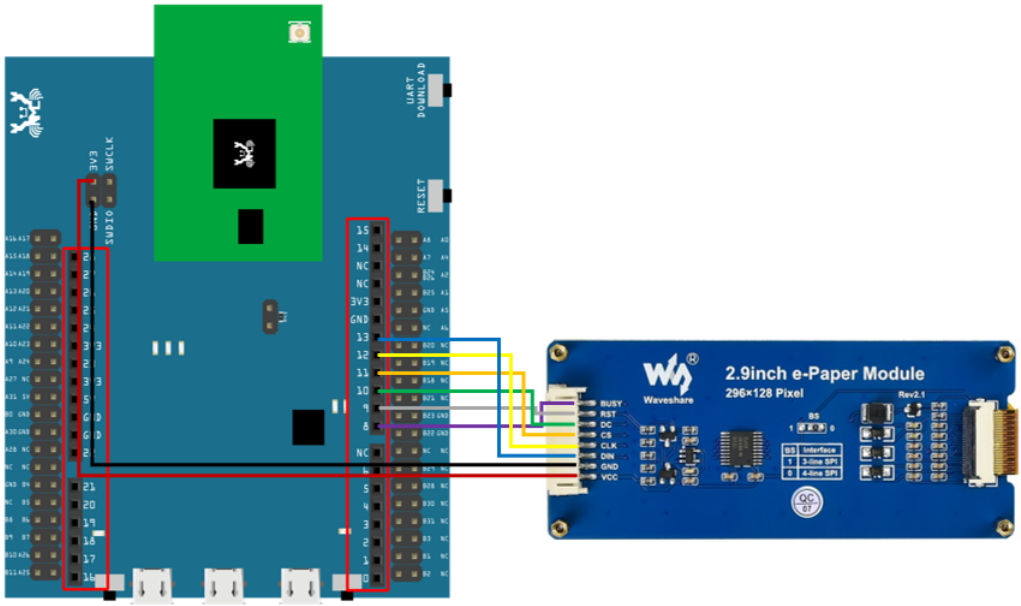
In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.

The 2.9” active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

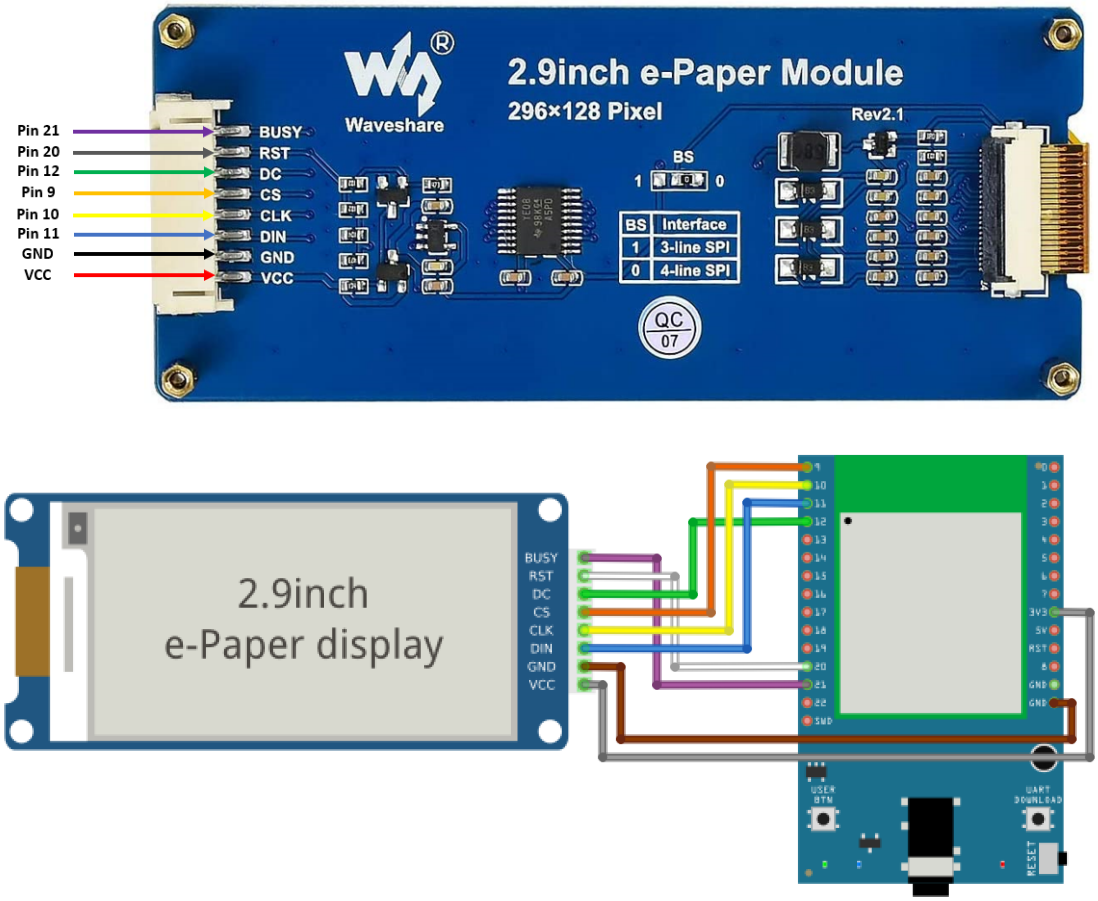
Front view of the e-Paper Module:



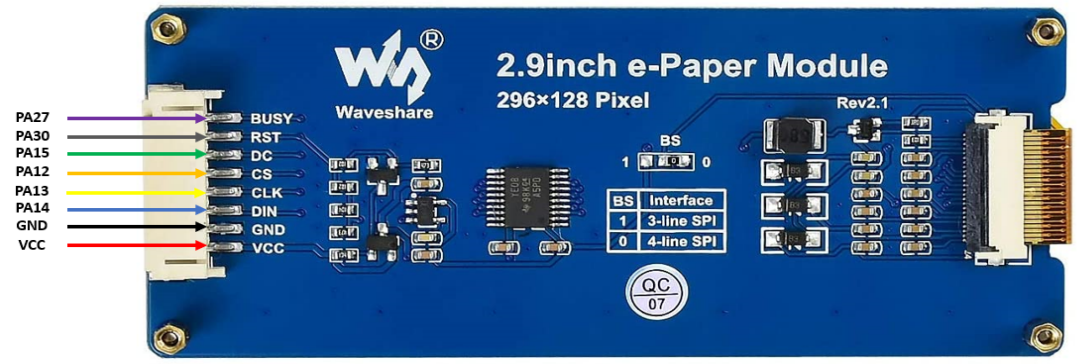
AMB21 / AMB22 Wiring Diagram:

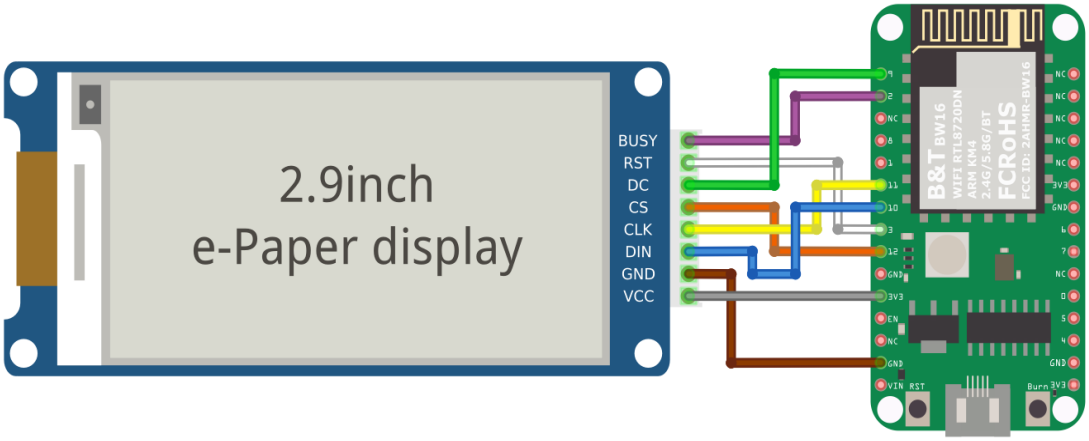


AMB23 Wiring Diagram:

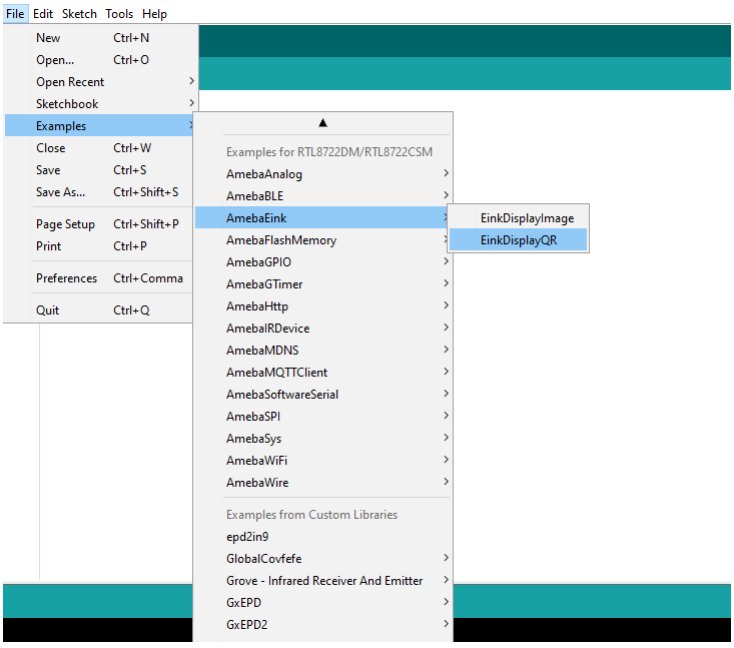


BW16 Wiring Diagram:

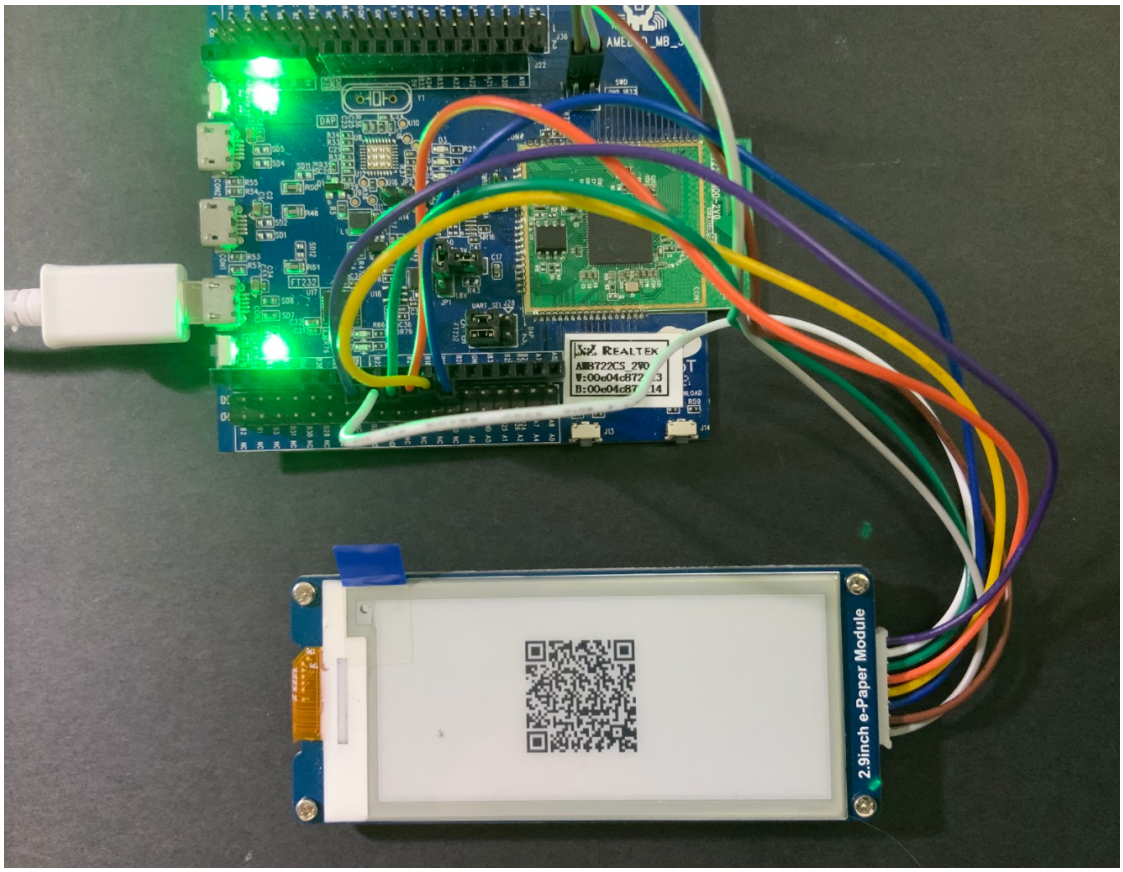




Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`DisplayQR`":



Modify the URL in the loop() section as your wish, after that, verify and upload the code to the Ameba board. Upon successfully upload the sample code and press the reset button, a QR code generated based on the URL of your input will be shown on the E-Paper module. The QR code showing below leads to our Ameba IoT official website: Ameba ARDUINO



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html
[3] A simple library for generating QR codes in C, optimized for processing and memory-constrained systems: https://github.com/ricmoo/QRCode#data-capacities

Flash Memory - Store data in FlashEEProm

Preparation

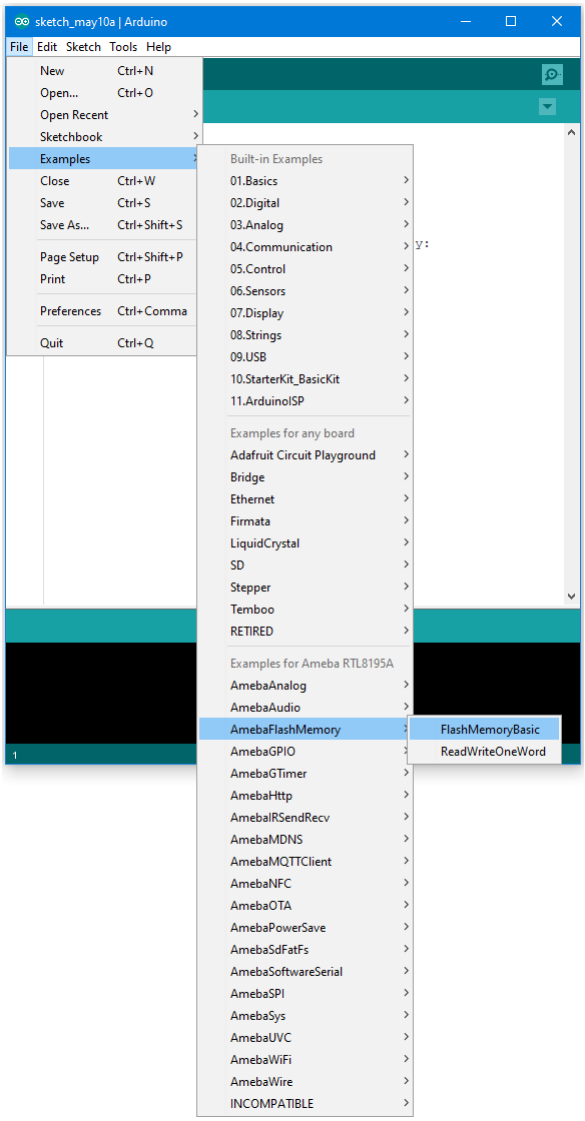
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

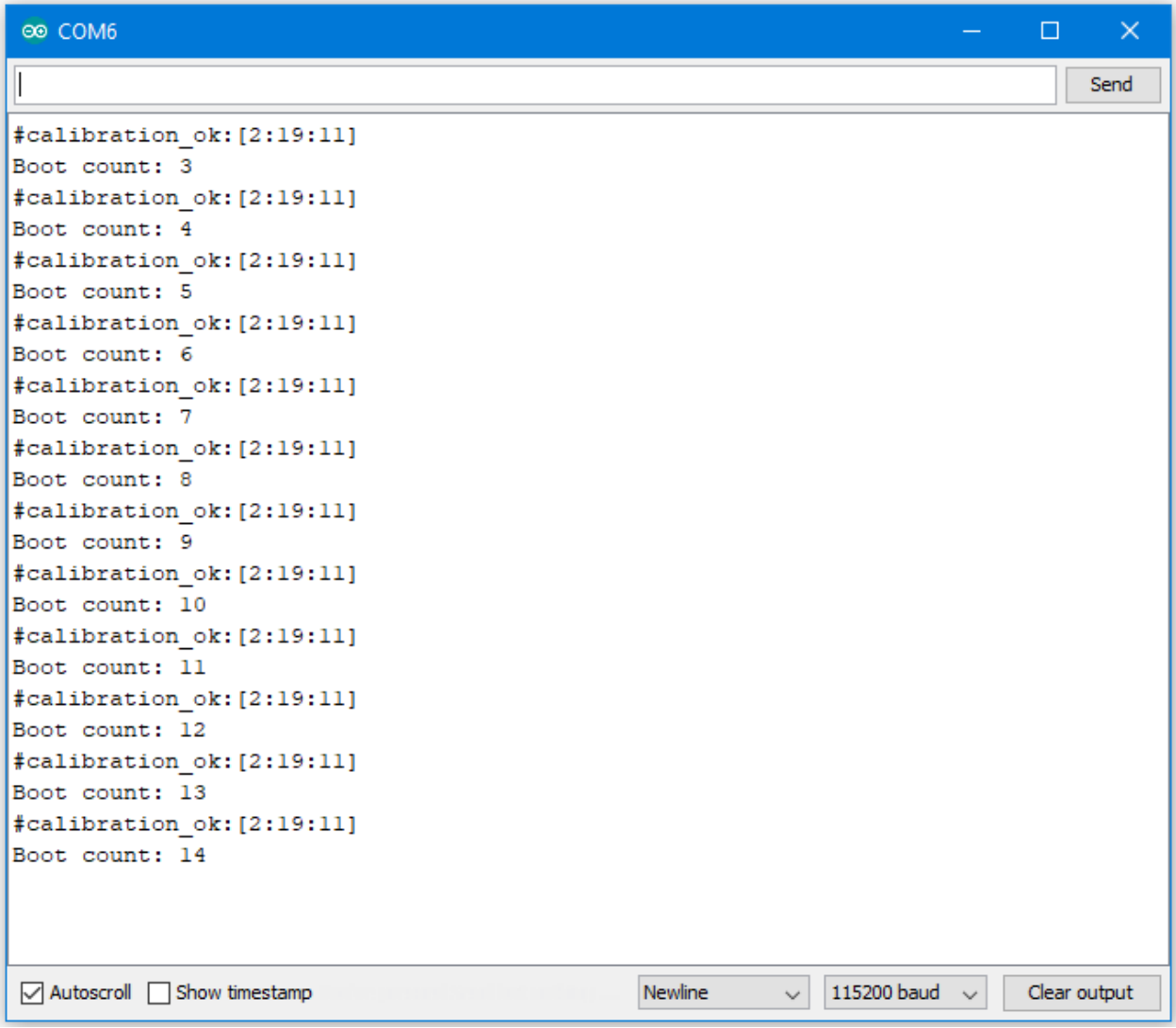
Ameba provides Flash Memory component for data storage and the data can be preserved when the power is off if necessary, e.g., compiled program. To avoid the memory space overlapped with the program on Ameba, the Flash API

uses the tail part of the address space, with sector size 4K.
In this example, we store the value of boot times in flash memory. Every time Ameba reboots, it reads the boot times from flash, increases the value by 1, and writes it back to flash memory.

First open the example, "`File` -> "`Example`" -> "`AmebaFlashMemory`" -> "`FlashMemoryBasic`"



Compile and upload to Ameba, then press the reset button.
Open the Serial Monitor, press the reset button for a few times. Then you can see the boot times value increases.



Code Reference

By default, the Flash Memory API uses address 0xFF000~0xFFFFF to store data.

There is limitation when writing to flash memory. That is, you can not directly write data to the same address you used in last write. To do that correctly, you need erase the sector first. The Flash API of Ameba uses a 4K SRAM to record the user modification and do the erase/write task together.

Use `FlashMemory.read()` to read from Flash memory.
Use `FlashMemory.buf[0] = 0x00;` to manipulate the 4K buf.
Use `FlashMemory.update();` to update the data in buf to Flash Memory.

Flash Memory - Use Flash Memory Larger Than 4K

Preparation

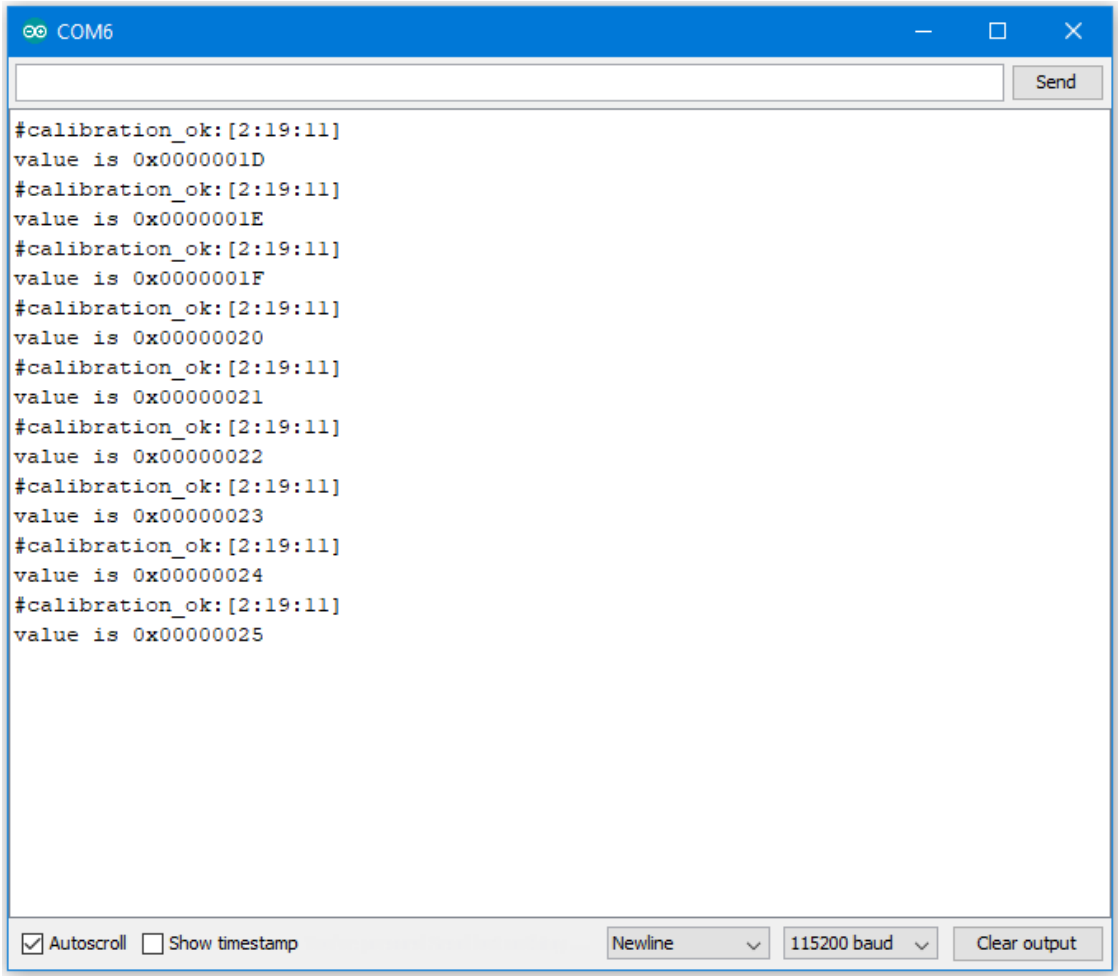
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Flash Memory API uses memory of 4K bytes, which is normally sufficient for most application. However, larger memory can be provided by specifying a specific memory address and required size.

First, open the sample code in "`File`" -> "`Examples`" -> "`AmebaFlashMemory`" -> "`ReadWriteOneWord`"

In this example, we specify the starting address of flash memory is 0xFC000 and size is 0x4000 (The default starting address is 0xFF000 and size is 0x1000).
Then calculate correct address according to the specified offset and perform read/write operation. In the sample code we use offset 0x3F00, that is, 0xFC000 + 0x3F00 = 0xFFF00 in flash. We set the value to 0 at first, then increase by 1 every time Ameba reboots.



Code Reference

We can use the flash api we used in previous flash memory example, but we need to use `begin()` function to specify the desired starting address and memory size.

```
FlashMemory.begin(0xFC000, 0x4000);
```

Use `readWord()` to read the value stored in a memory address. In the example, we read the value stored in memory offset 0x3F00, that is 0xFC000 + 0x3F00 = 0xFFF00. `readWord()` function reads a 32-bit value and returns it.

```
value = FlashMemory.readWord(0x3F00);
```

Use `writeWord()` to write to a memory address. The first argument is the memory offset, the second argument is the value to write to memory.

```
FlashMemory.writeWord(0x3F0C, value);
```

GPIO - Measure Distance By Ultrasound Module

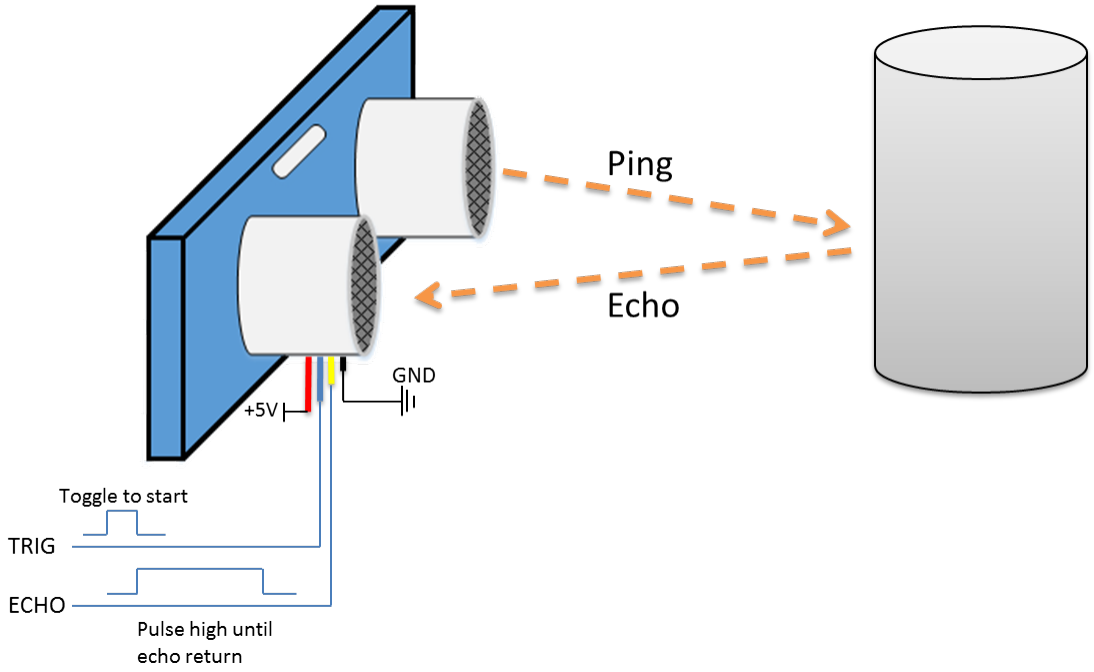
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- HC-SR04 Ultrasonic x 1
- Dropping resistor or Level converter

Example

HC-SR04 is a module that uses ultrasound to measure the distance. It looks like a pair of eyes in its appearance, therefore it's often installed onto robot-vehicle or mechanical bugs to be their eyes.

The way it works is that first we "toggle high" the TRIG pin (that is to pull high then pull low). The HC-SR04 would send eight 40kHz sound wave signal and pull high the ECHO pin. When the sound wave returns back, it pull low the ECHO pin.

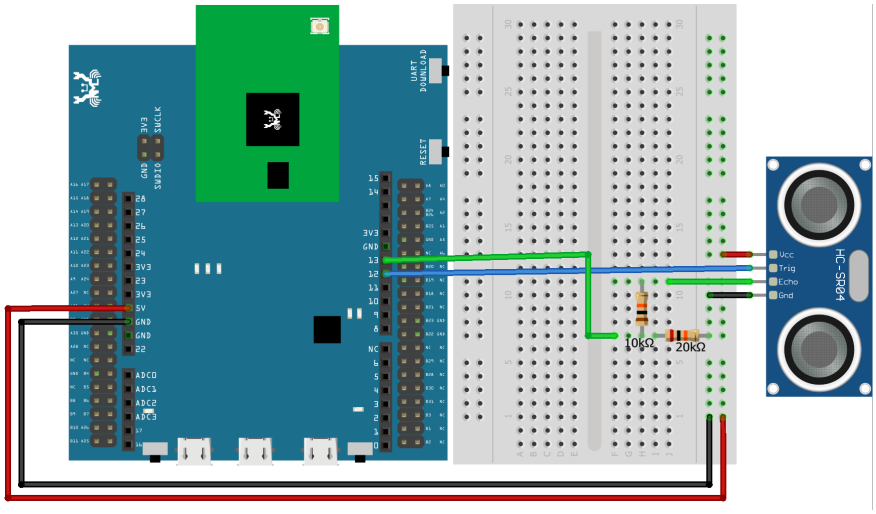


Assume the velocity of sound is 340 m/s, the time it takes for the sound to advance 1 cm in the air is 340*100*10^-6 = 29 us.

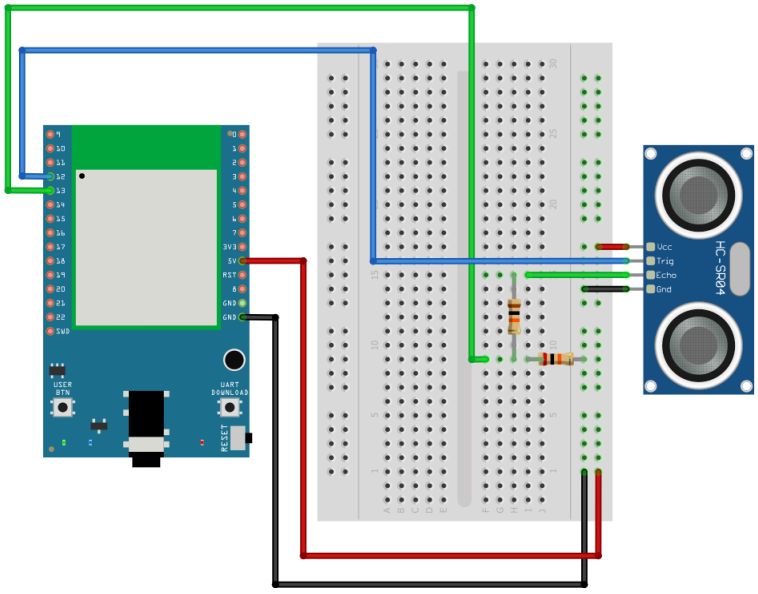
The sound wave actually travels twice the distance between HC-SR04 and the object, therefore the distance can be calculated by (time/29) / 2 = time / 58.

The working voltage of HC-SR04 is 5V. When we pull high the ECHO pin to 5V, the voltage might cause damage to the GPIO pin of Ameba. To avoid this situation, we need to drop the voltage as follows:

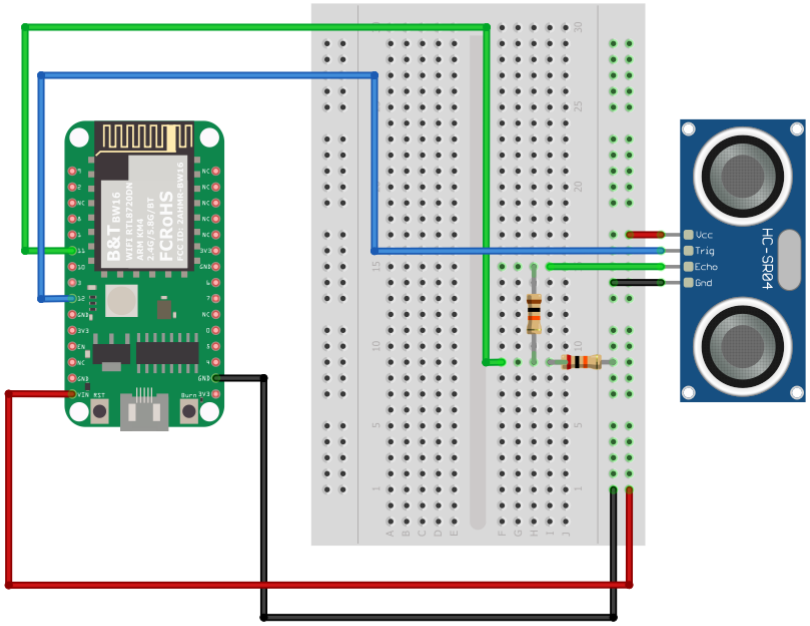
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

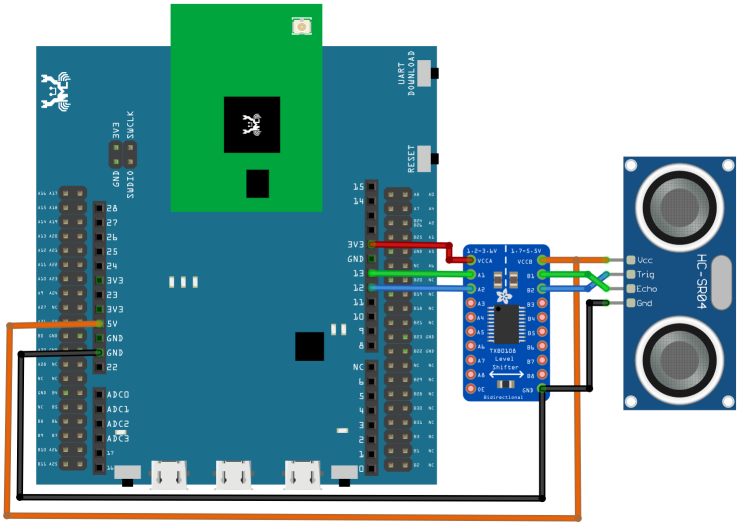


BW16 Wiring Diagram:

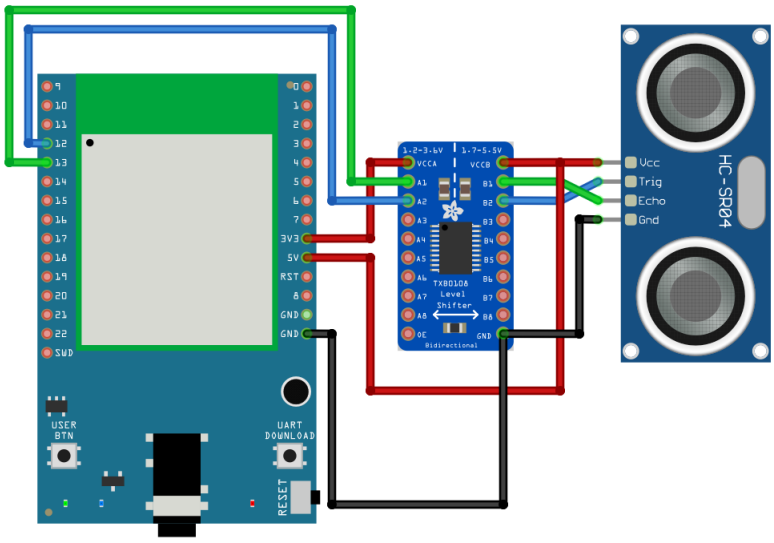


We pick the resistors with resistance 1:2, in the example we use 10kΩ and 20kΩ.
If you do not have resistors in hand, you can use level converter instead.The TXB0108 8 channel level converter is a suitable example:

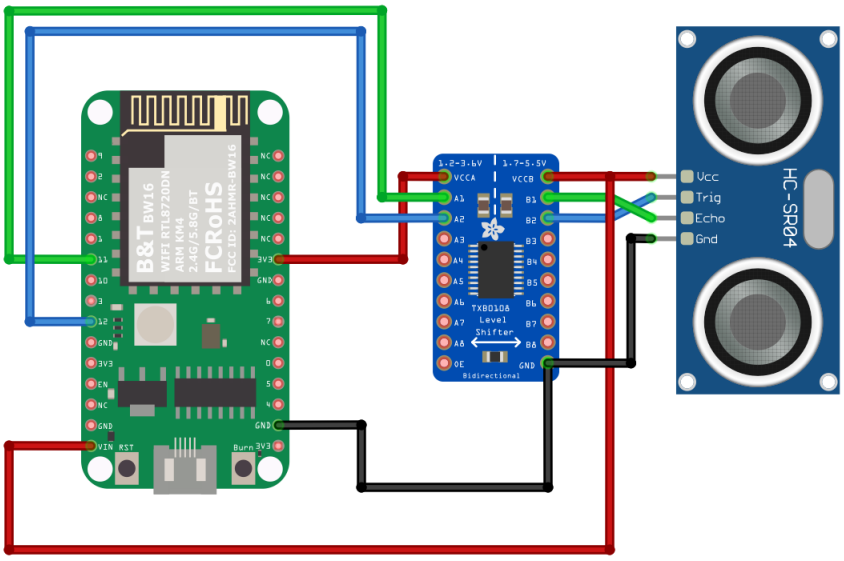
AMB21 / AMB22 Wiring Diagram:



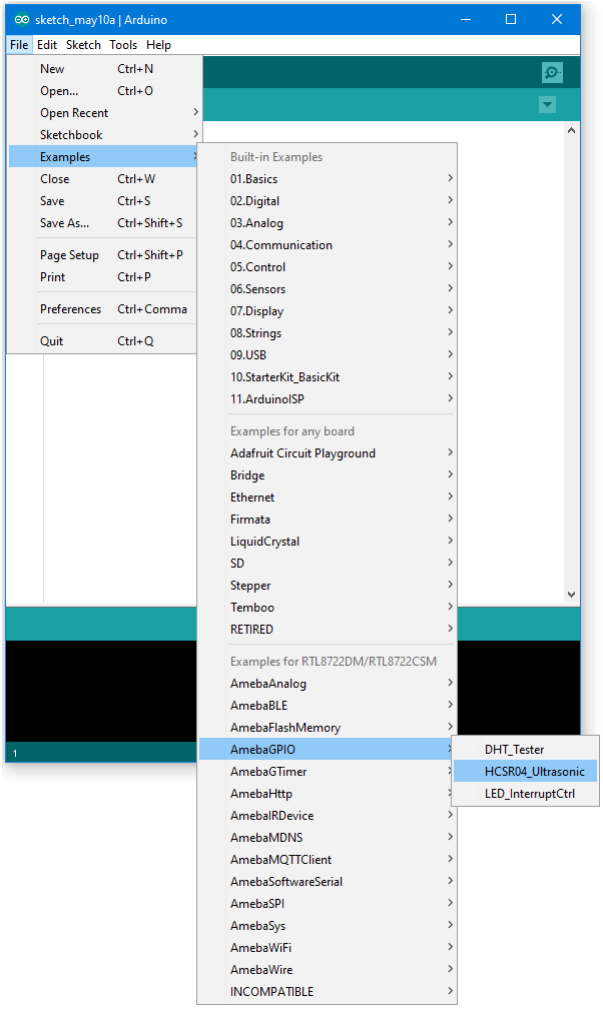
AMB23 Wiring Diagram:



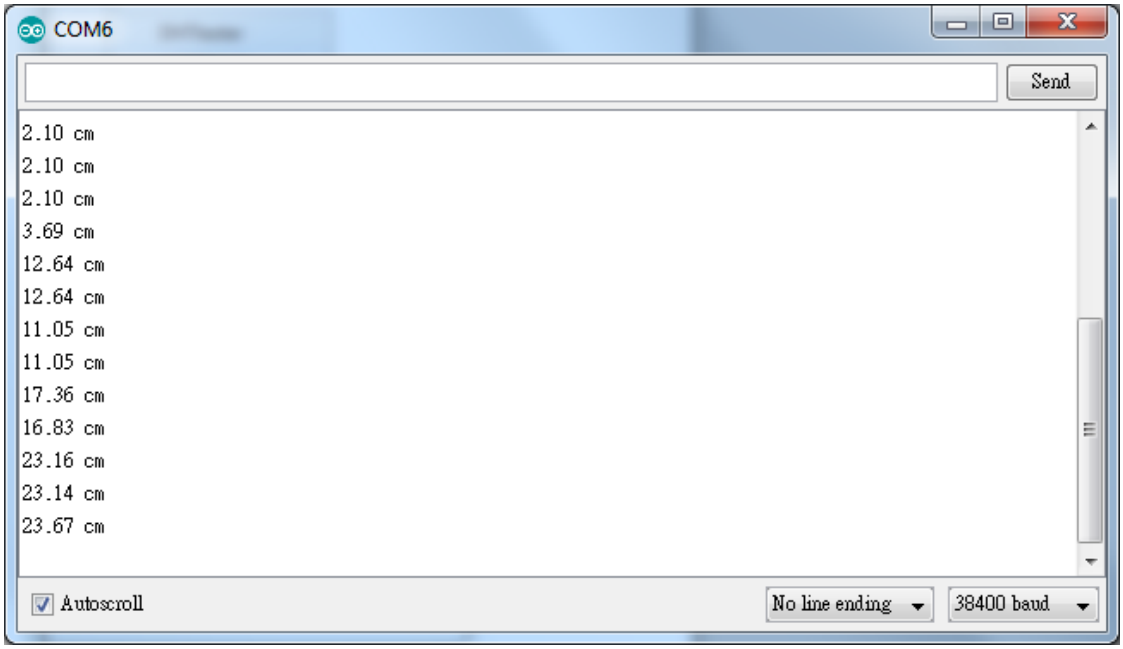
BW16 Wiring Diagram:



Next, open the sample code in "`File`" -> "`Examples`" -> "`AmebaGPIO`" -> "`HCSR04_Ultrasonic`"



Compile and upload to Ameba, then press the reset button. Open the Serial Monitor, the calculated result is output to serial monitor every 2 seconds.



Note that the HCSR04 module uses the reflection of sound wave to calculate the distance, thus the result can be affected by the surface material of the object (e.g., harsh surface tends to cause scattering of sound wave, and soft surface may cause the sound wave to be absorbed).

Code Reference

Before the measurement starts, we need to pull high the TRIG pin for 10us and then pull low. By doing this, we are telling the HC-SR04 that we are about to start the measurement:

```
digitalWrite(trigger_pin, HIGH);
delayMicroseconds(10);
digitalWrite(trigger_pin, LOW);
```

Next, use pulseIn to measure the time when the ECHO pin is pulled high.

```
duration = pulseIn (echo_pin, HIGH);
```

Finally, use the formula to calculate the distance.

```
distance = duration / 58;
```

GPIO - Measure Temperature and Humidity

Preparation

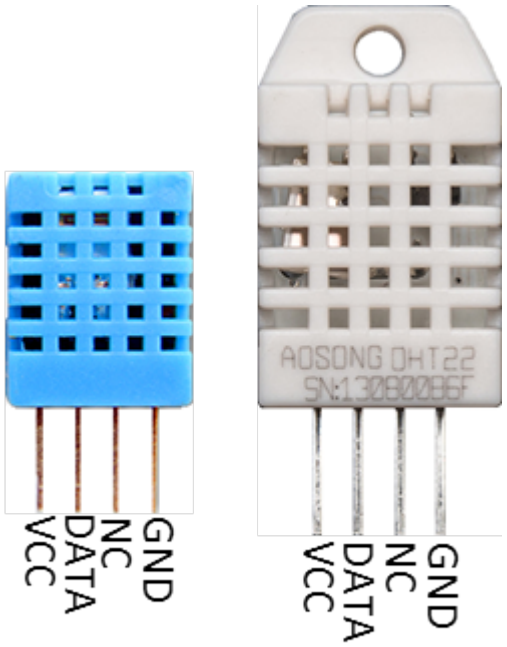
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21

Example

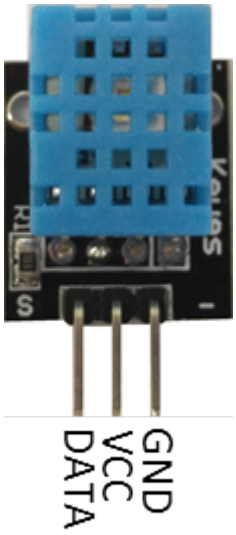
DHT11 is a temperature and humidity sensor which operates at voltage 3.3V~5V. At room temperature, the measurable range of the humidity is 20% ~ 90%RH with ±5%RH precision, the measurable range of the temperature is 0 ~ 50℃ with ±2℃ precision.

Another choice of temperature and humidity sensor is DHT22 sensor, which has better precision. Its measurable range of the humidity is 0%~100%RH with ±5%RH precision, the measurable range of the temperature is -40~125 ℃ with ±0.2℃ precision.

There are 4 pins on the sensor:



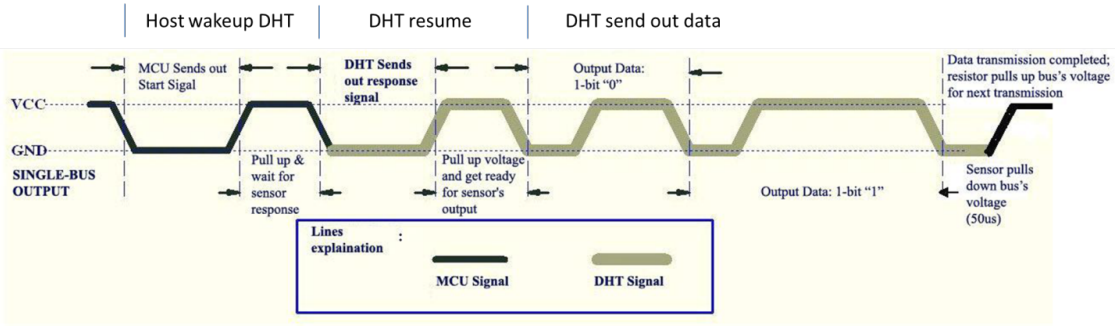
Since one of the 4 pins has no function, there are temperature/humidity sensors with only 3 pins on the market:



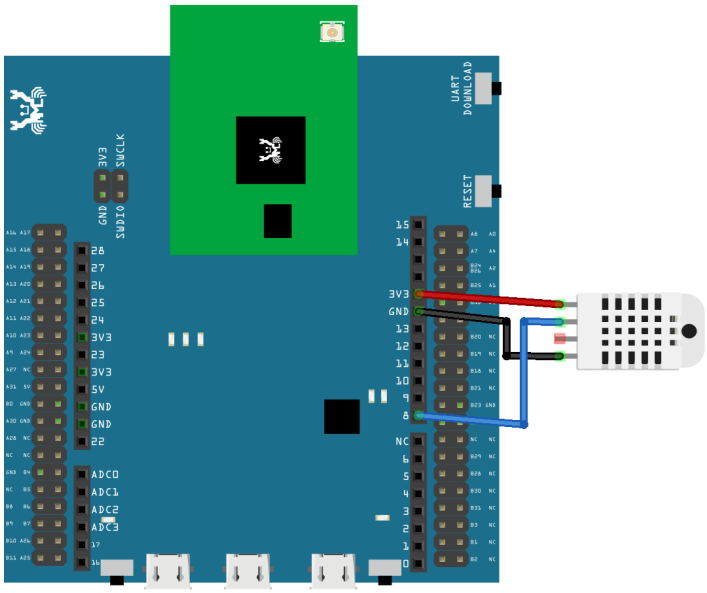
DHT is normally in the sleeping mode. To get the temperature/humidity data, please follow the steps:

1. Awake DHT: Ameba toggles low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital out to Ameba.

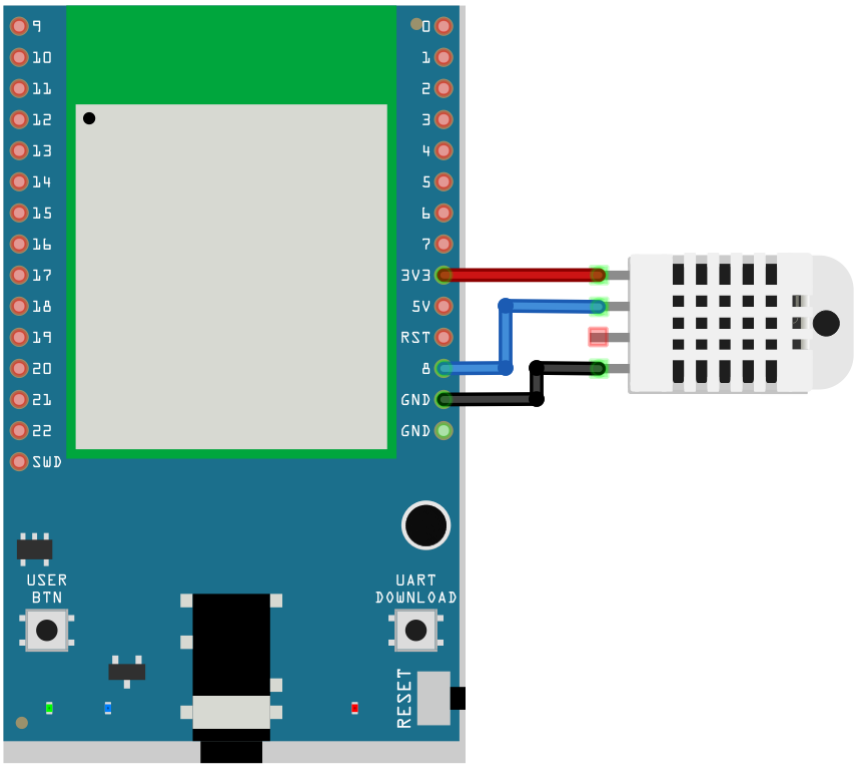
2. DHT response: DHT also toggle low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital in for Ameba.
3. DHT sends data: DHT sends out the temperature/humidity data (which has size 5 bytes) in a bit by bit manner. To represent each bit, DHT first pull low the DATA GPIO pin for a while and then pull high. If the duration of high is smaller than low, it stands for bit 0. Otherwise it stands for bit 1.



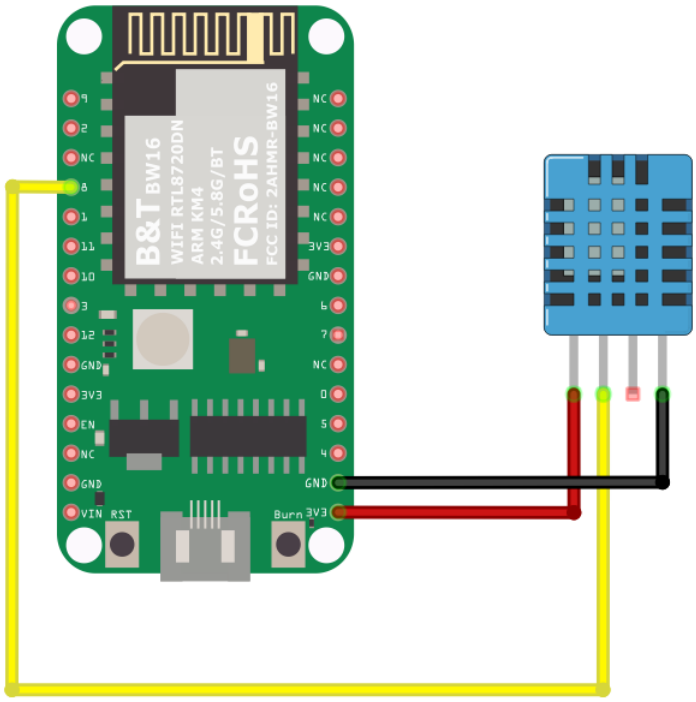
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

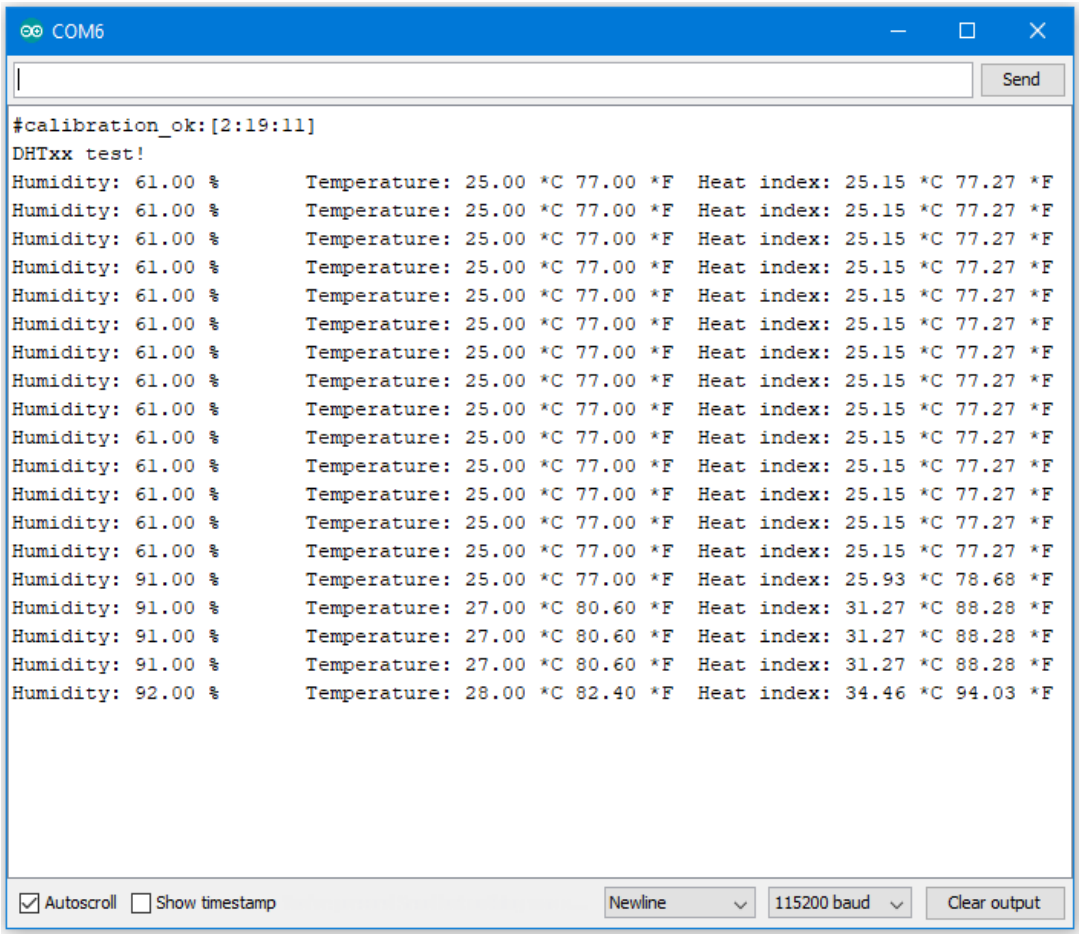


BW16 Wiring Diagram:



Open the sample code in "`Files`" -> "`Examples`" -> "`AmebaGPIO`" -> "`DHT_Tester`". Compile and

upload to Ameba, then press the reset button. The result would be shown on the Serial Monitor.



```
#calibration_ok:[2:19:11]
DHTxx test!
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 91.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.93 *C 78.68 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 92.00 %	Temperature: 28.00 *C 82.40 *F	Heat index: 34.46 *C 94.03 *F
```

Code Reference

Use `dht.readHumidity()` read the humidity value, and use `dht.readTemperature()` to read the temperature value.

Every time we read the temperature/humidity data, Ameba uses the buffered temperature/humidity data unless it found the data has expired (i.e., has not been updated for over 2 seconds). If the data is expired, Ameba issues a request to DHT to read the latest data.

GPIO - Use GPIO Interrupt To Control LED

Preparation

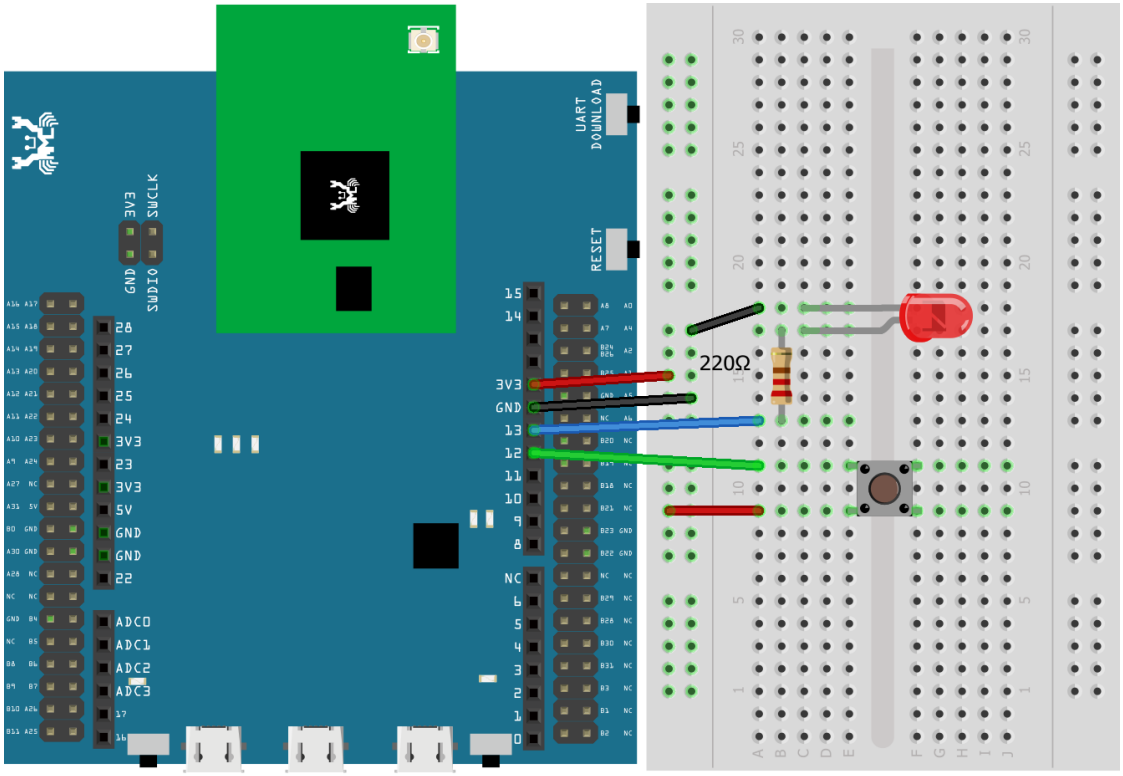
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- LED x 1
- Button x 1

Example

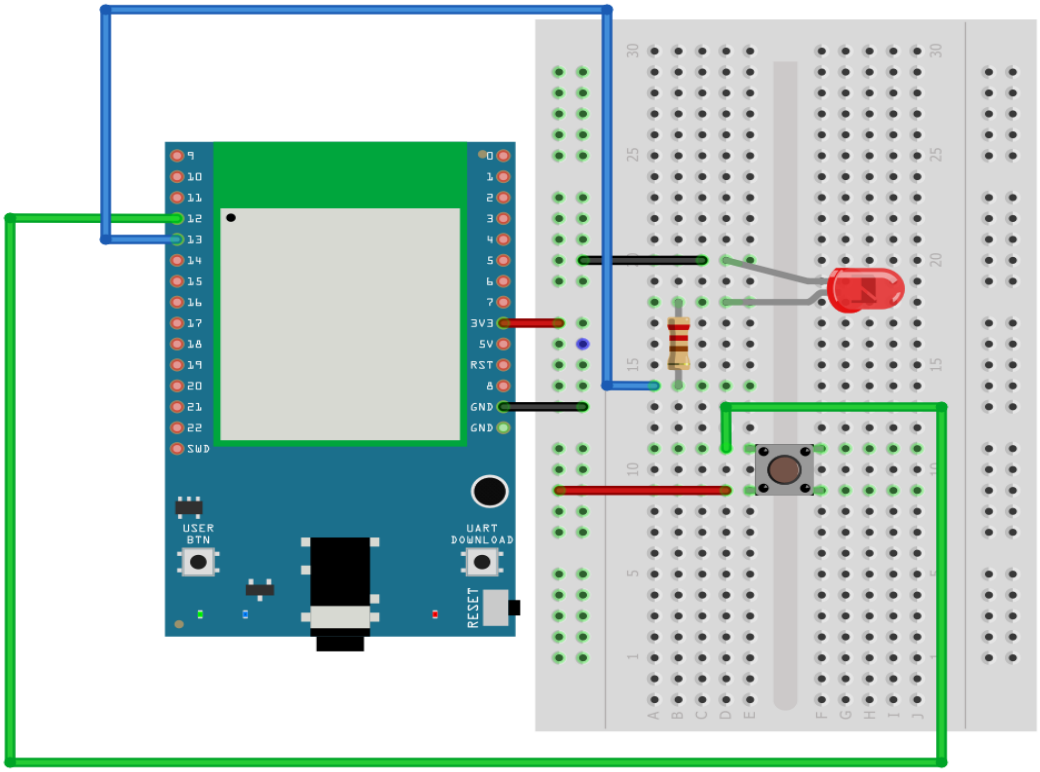
In this example, we use a button to trigger interrupt and control the LED. When we press and release the button, the LED dims, press and release the button again, and the LED lights.Note that in the Arduino example “Button and LED”, LED only lights when the button is pressed and hold, when we release the button, the LED dims.

Open the example, “`Files`” -> “`Examples`” -> “`AmebaGPIO`” -> “`LED_InterruptCtrl`”

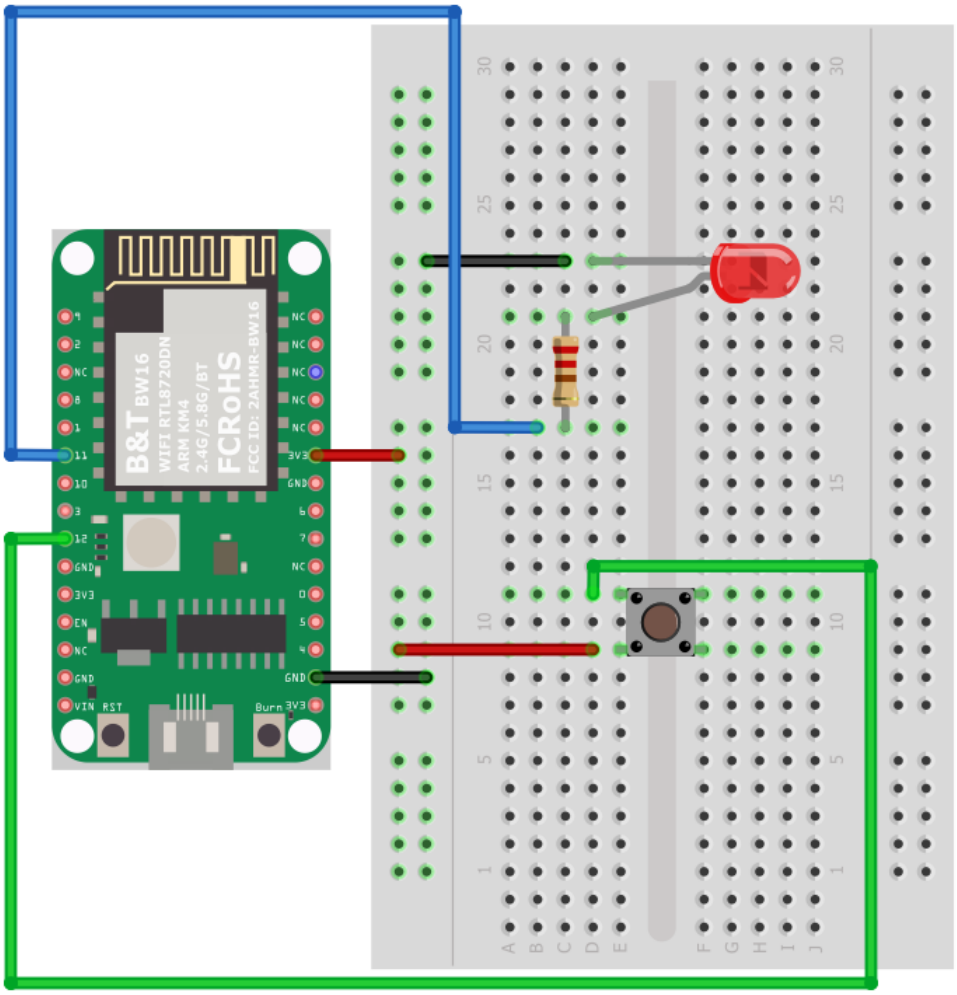
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Compile and upload the program, press reset.
The LED lights at first. Press and release the button, then the LED should dim. Press again, then the LED should light.

Code Reference

In

```
setup()
```

we set Pin 12 to

```
INPUT_IRQ_RISE
```

, this means that an interrupt occurs when the voltage of this pin changes from GND to 3V3. Therefore, we connect the other side of the button to 3V3, so as to trigger interrupt event when the button is pressed.

```
pinMode(button, INPUT_IRQ_RISE);
```

On the other hand, we can set pin 12 to

```
INPUT_IRQ_FALL
```

, this means that an interrupt occurs when the voltage of this pin changes from 3V3 to GND. In this case, the other side of the button is connected to GND.Next, we need to specify the funtion to be execute to handle the interrupt:

```
digitalSetIrqHandler(button, button_handler);
```

The second parameter is a function pointer, with prototype:

```
void button_handler(uint32_t id, uint32_t event)
```

In this handler, every time we press and release the button, we trigger an interrupt, and change the status of the LED.

GTimer - Using The Periodic GTimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Ameba provides 4 hardware GTimer for users to use. The timers' resolutions are at microseconds scale.
The timer can be set to be periodic or for single use. The periodic timers reset periodically, and the single-use timers do not.

Open the example, "`File" -> "Examples" -> "AmebaGTimer" -> "TimerPeriodical`". Compile and upload to Ameba, and press reset.
In the Serial Monitor, you can see the counter value is increased periodically.

Code Reference

The first argument of begin() is the timer id (0~3).
The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s.
The third argument specifies the function to call when the time is up. In the example, we call the "myhandler" function to increase the counter value by 1 and print the counter value to serial monitor.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler);
```

The GTimer is periodic by default, therefore "myhandler" function is called every second. When we want to stop the GTimer, use `stop()`:

```
GTimer.stop(0);
```

GTimer - Using the One-Time Gtimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we will use 4 One-Time GTimer, and pass user data to each timer.

Open the example "File" -> "Examples" -> "AmebaGTimer" -> "TimerOneshot". Compile and upload to Ameba, and press reset. Then you can see the 4 timer log printed to the serial monitor in series.

Code Reference

The first argument of begin() is the Timer ID (0~3). The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s. The third argument specifies the function to call when the time is up. The fourth argument is to set whether this timer is a periodic timer, we use "false" here to begin a single-use timer. The fifth argument is the user data, we give 0 here to represent that this is timer 0.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
```

Next, we set up the second timer, which has timer value 2 seconds, and user data 1. And other timers are set similarly.

```
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
```

In myhandler function, we print the user data to serial monitor. Since the 4 timers are separately set to count for 1, 2, 3, 4 seconds, from 1 second to 4 second, the user data of each timer are printed on the serial monitor in order. After 4 second, no log will be printed.

I2C - Send Data to Arduino UNO

Introduction of I2C

There are two roles in the operation of I2C, one is "master", the other is "slave". Only one master is allowed and can be connected to many slaves. Each slave has its unique address, which is used in the communication between master and the slave. I2C uses two pins, one is for data transmission (SDA), the other is for the clock (SCL). Master uses the SCL to inform slave of the upcoming data transmission, and the data is transmitted through SDA. The I2C example was named "Wire" in the Arduino example.

Materials

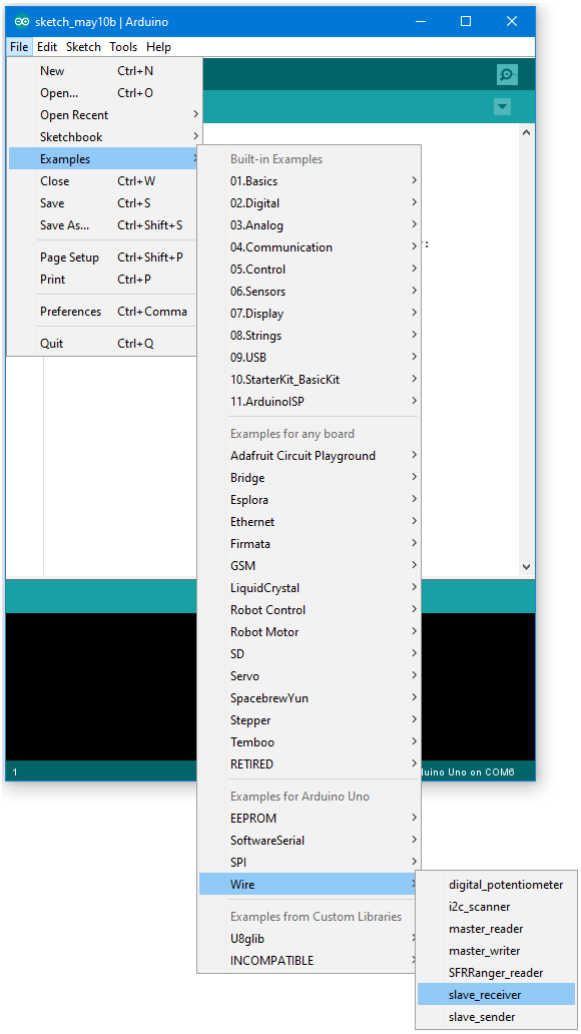
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In this example, we use Ameba as the I2C master writer, and use Arduino as the I2C slave receiver.
When the I2C slave receives string sent from I2C master, it prints the received string.

- Setting up Arduino Uno to be I2C Slave

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`"
Open the "Slave Receiver" example in "`Examples`" -> "`Wire`" -> "`slave_receiver`":

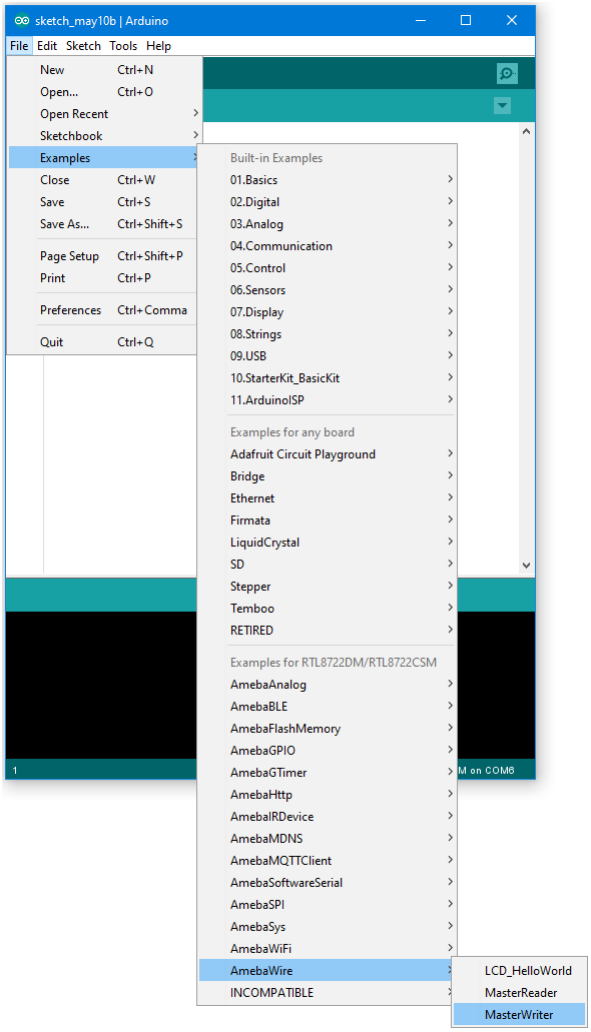


Then click "`Sketch`" -> "`Upload`" to compile and upload the example to Arduino Uno.

- Setting up Ameba to be I2C Master

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE:
"`Tools`" -> "`Board`"

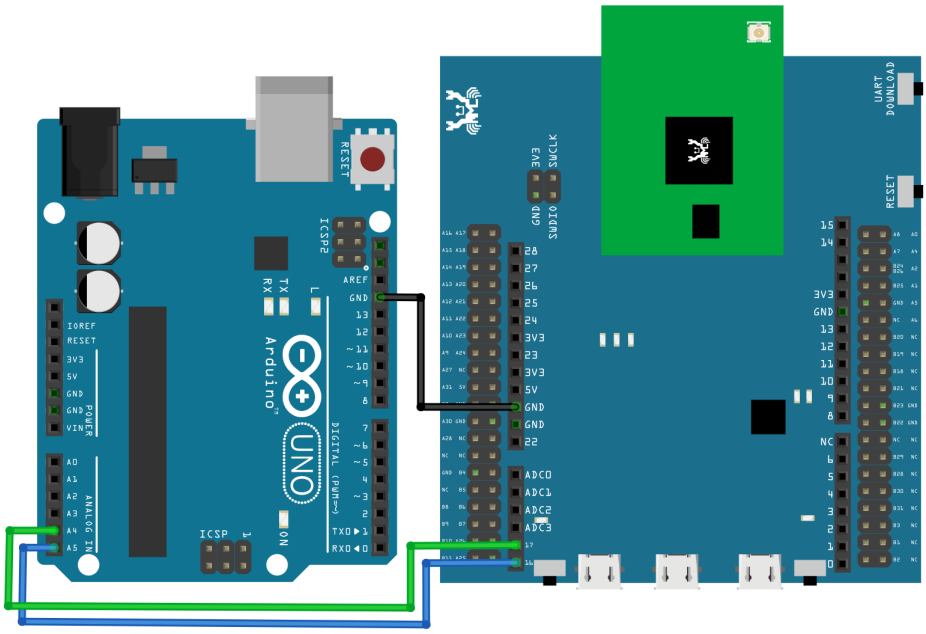
Then open the "Master Writer" example in "`File`" -> "`Examples`" -> "`AmebaWire`" -> "`MasterWriter`"



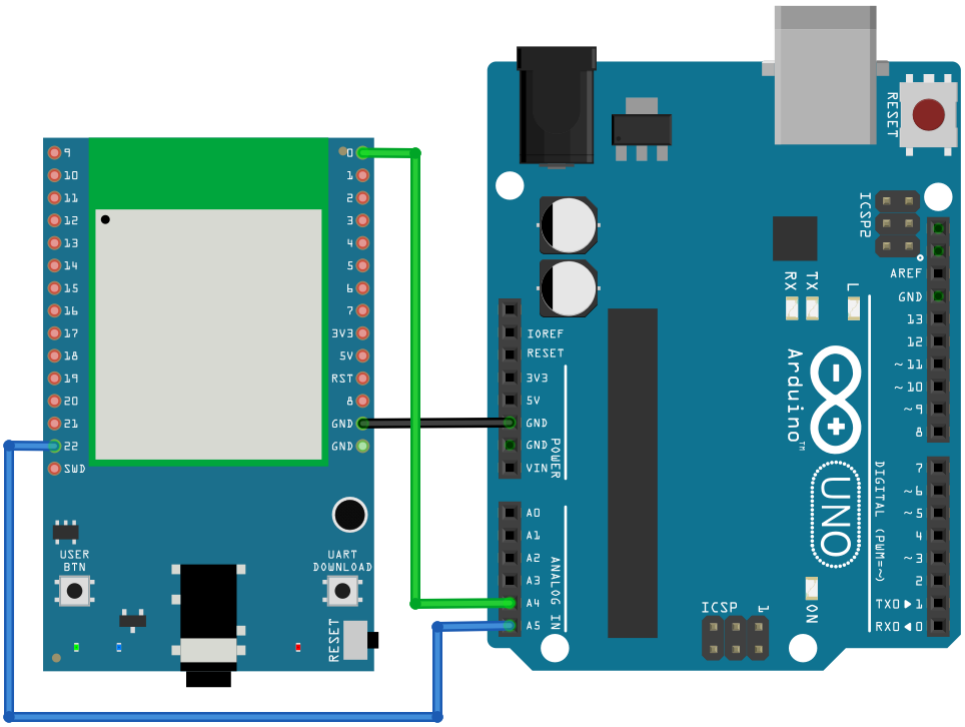
- Wiring

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

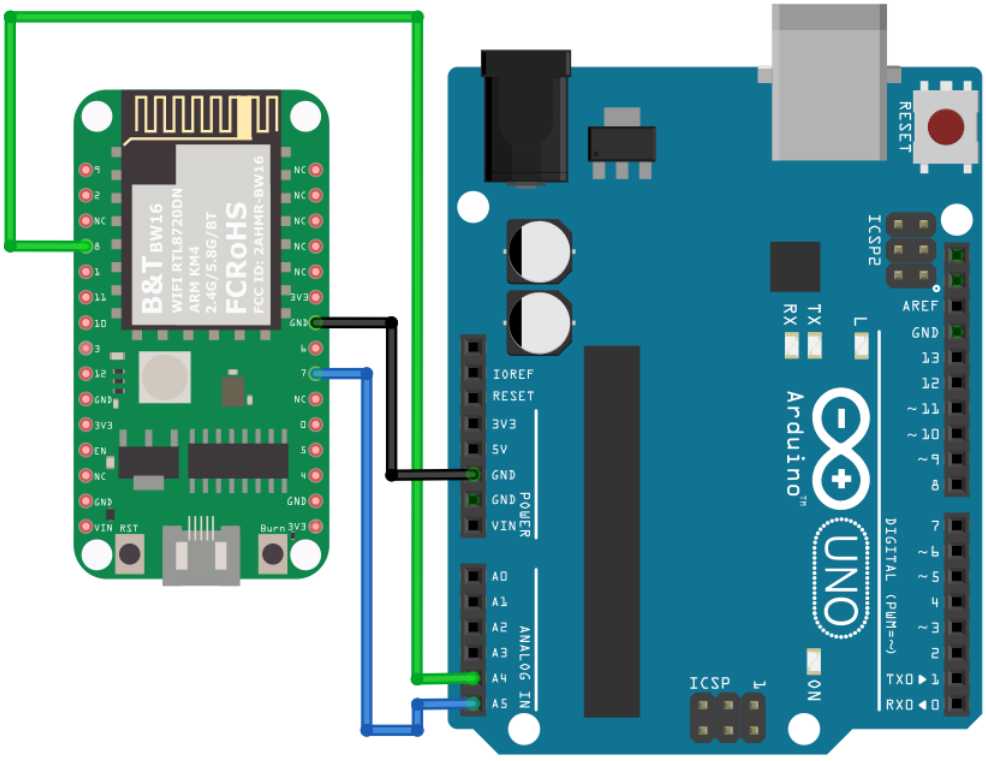
AMB21/ AMB22 Wiring Diagram:



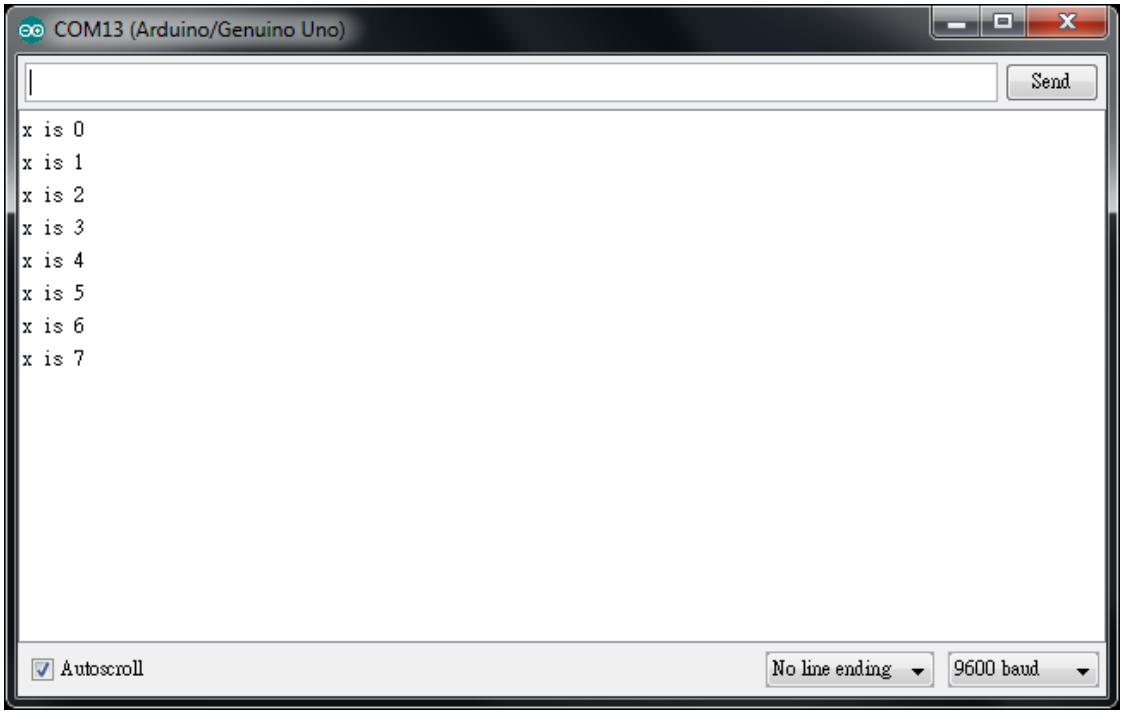
AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the Arduino IDE of the Arduino Uno and open the serial monitor ("Tools" -> "Serial Monitor").
In the Serial Monitor, you can see the messages printed from Arduino Uno.
Next, press the reset button on Arduino Uno. Now the Arduino Uno is waiting for the connection from I2C Master.
We press the reset button on Ameba to start to send messages. Then observe the serial monitor, you can see the messages show up every half second.



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterWriter

First use Wire.begin()/Wire.begin(address) to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses Wire.beginTransmission(address) to begin a transmission to the I2C slave with the given address:
https://www.arduino.cc/en/Reference/WireBeginTransmission

Uses Wire.write() to send data, and finally use Wire.endTransmission() to end a transmission to a Slave and transmits the bytes that were queued:
https://www.arduino.cc/en/Reference/WireEndTransmission

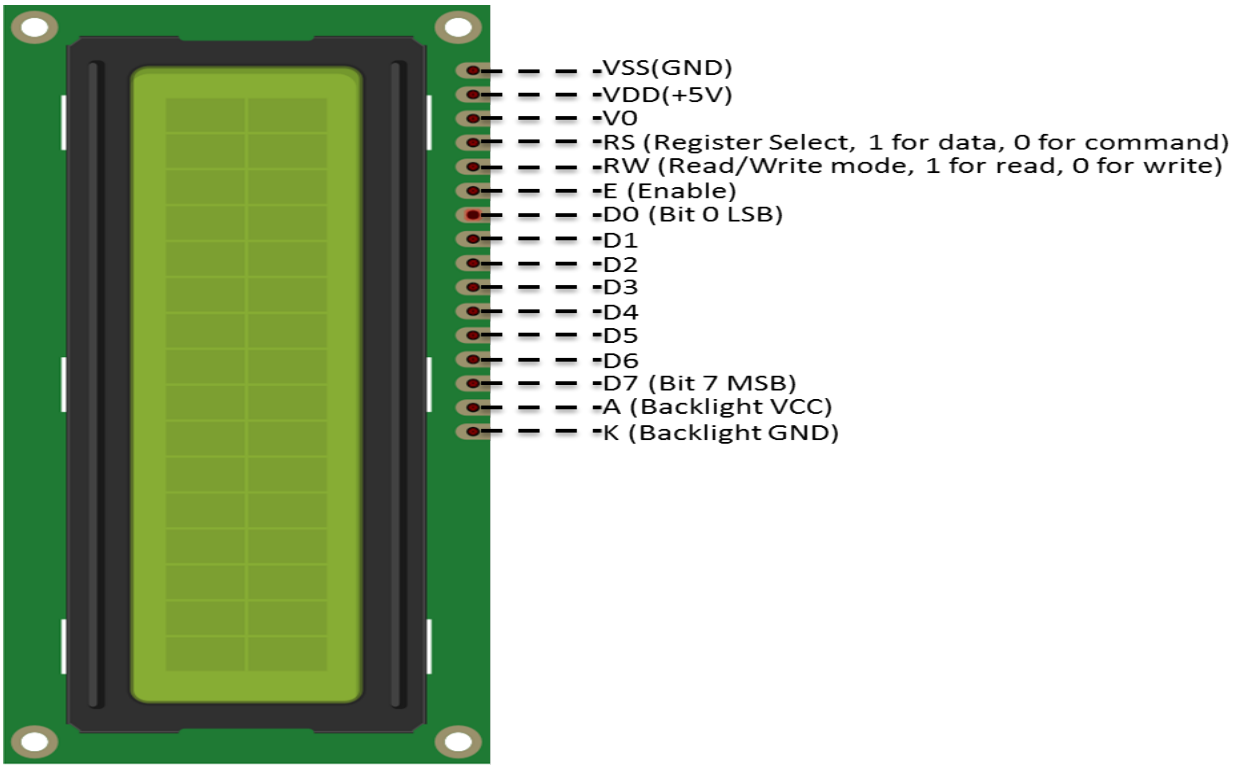
I2C - Display Data On LCD Screen

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- I2C 2×16 LCD

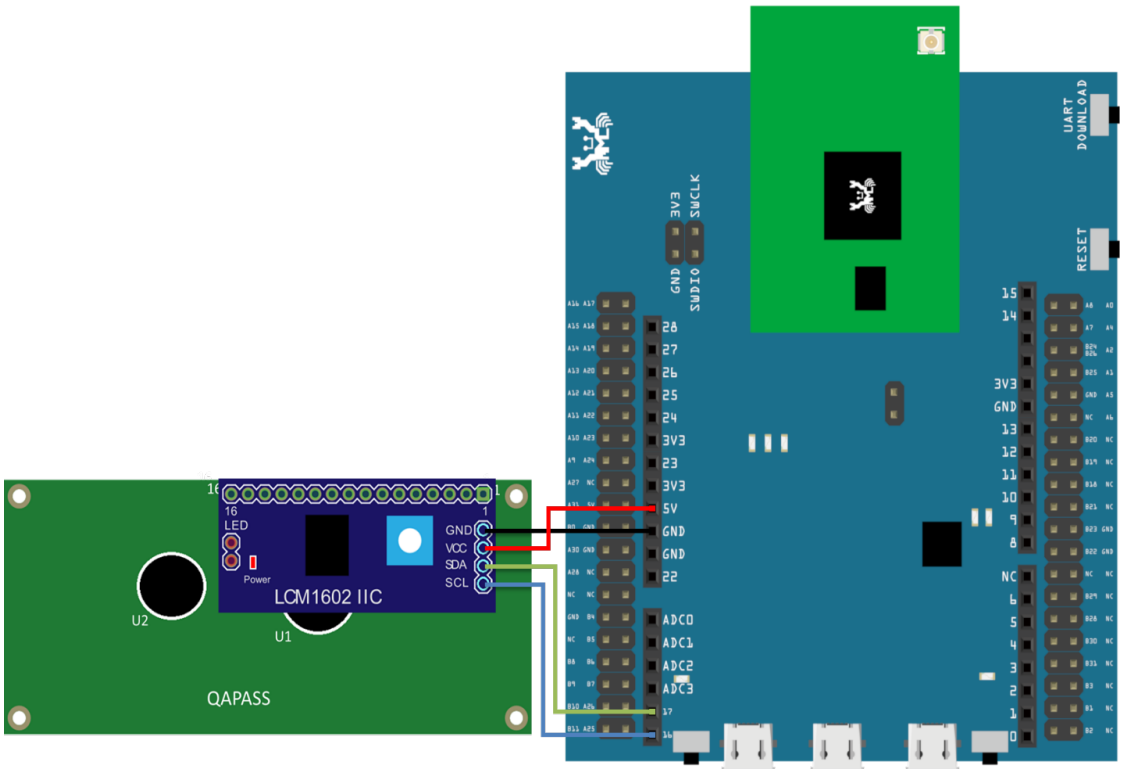
Example

Normally there are many pins on an LCD display, as shown in below figure.

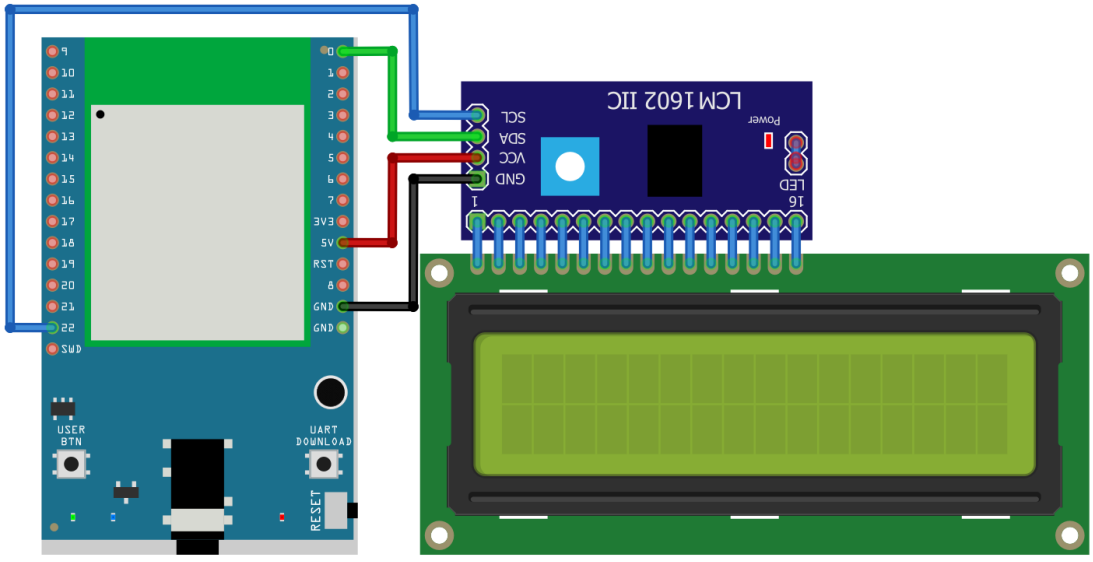


An LCD display can be equipped with an additional processing chip to process the data. The processing chip can connect to a microcontroller using the I2C interface.

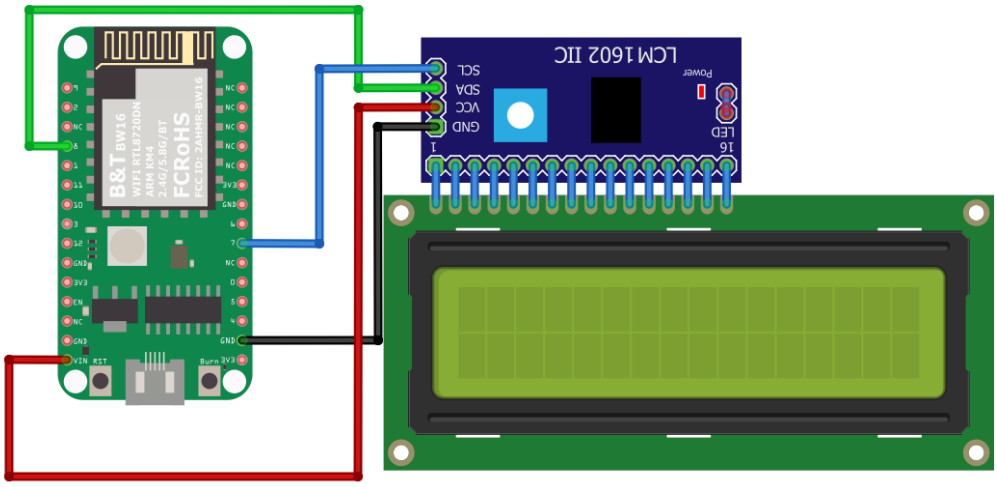
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

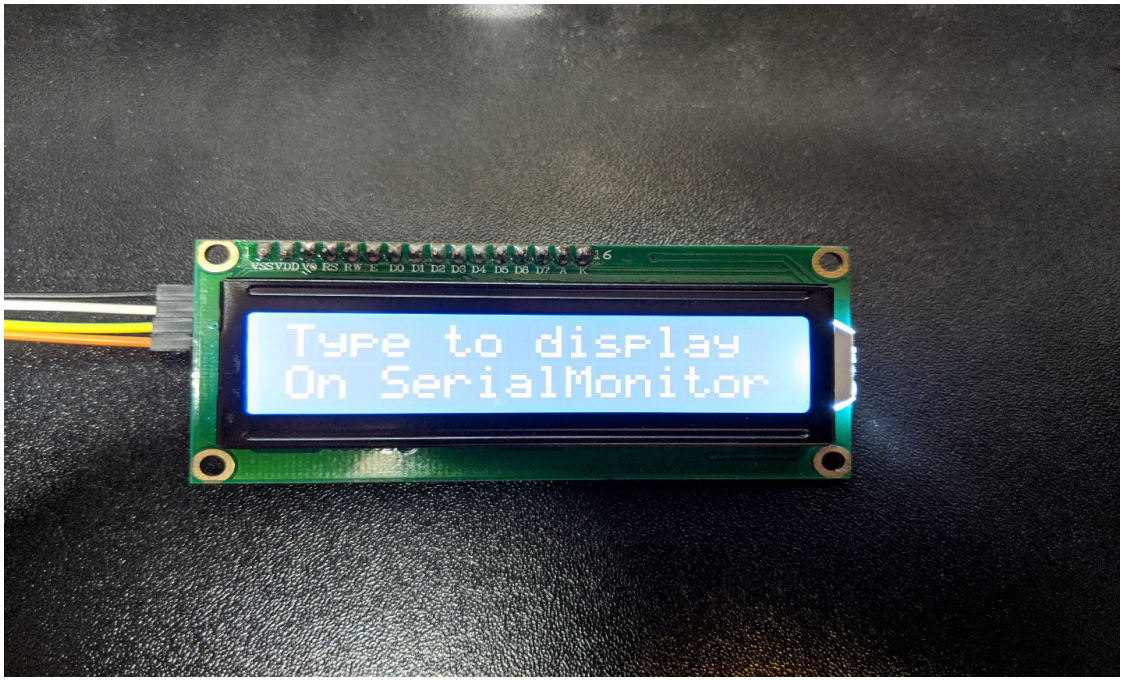


BW16 Wiring Diagram:



Open the example in "`File` -> "`Examples`" -> "`AmebaWire`" -> "`LCD_HelloWorld`".
Compile and upload to Ameba, then press the reset button.
Then you can see "Hello World" in the first line, and "Ameba" in the second line displayed on the LCD screen.

After 8 seconds, you can input to the Serial Monitor the string you would like to display on the LCD.



For example, we enter “123456789” and press “Send”:

Code Reference

The required settings of each model of LCD might be different, the constructor we use in this example is:

```
LiquidCrystal_I2C(uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs,
                  uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7,
                  uint8_t backlighPin, t_backlighPol pol);
```

And the setting parameters are as follows:

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C␣
↪address
```

The first parameter 0x27 is the address of I2C. Each of the following 8 parameters represents the meaning of each bit in a byte, i.e., En is bit 2, Rw is bit 1, Rs is bit 0, d4 is bit 4, and so forth.

Call `backlight()` to light the screen,
Call `setCursor(0, 0)` to set the position of the cursor.
LCD inherits the Print class, so we can use `lcd.print()` to output string on the screen.

I2C - Receive Data from Arduino UNO

Materials

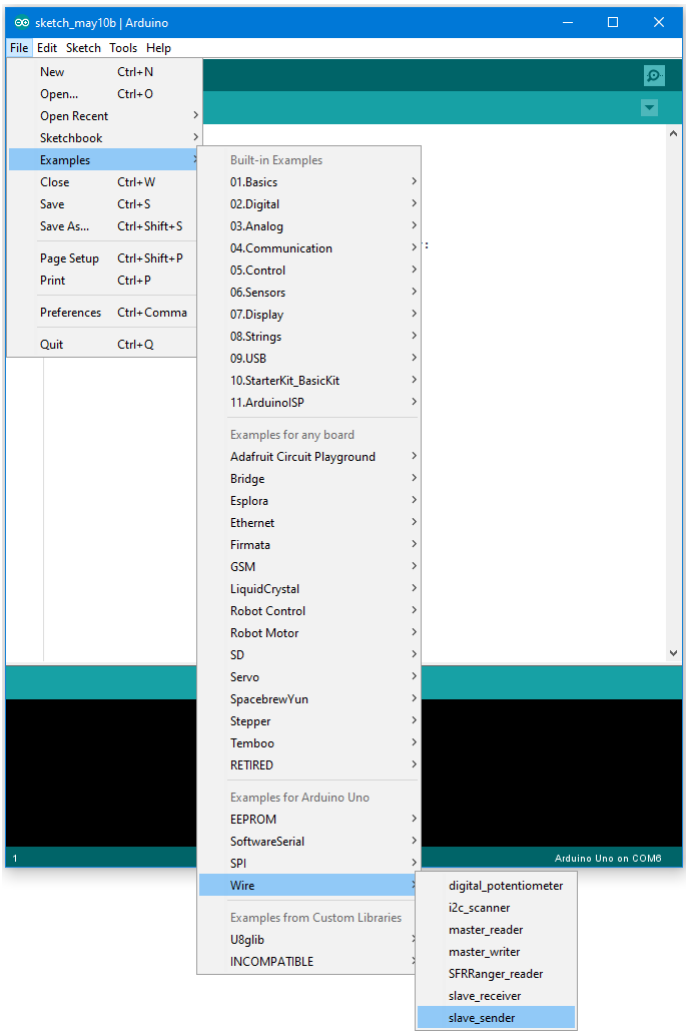
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In the previous example "I2C – Communicate with Arduino UNO via I2C" , Ameba, the I2C master, transmits data to the Arduino UNO, the I2C slave.
As to this example, Ameba is the I2C master, and receives data from the Arduino UNO, which is the I2C slave.

- **Setting up Arduino Uno to be I2C Slave**

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`":
Open "`Examples`" -> "`Wire`" -> "`slave_sender`"

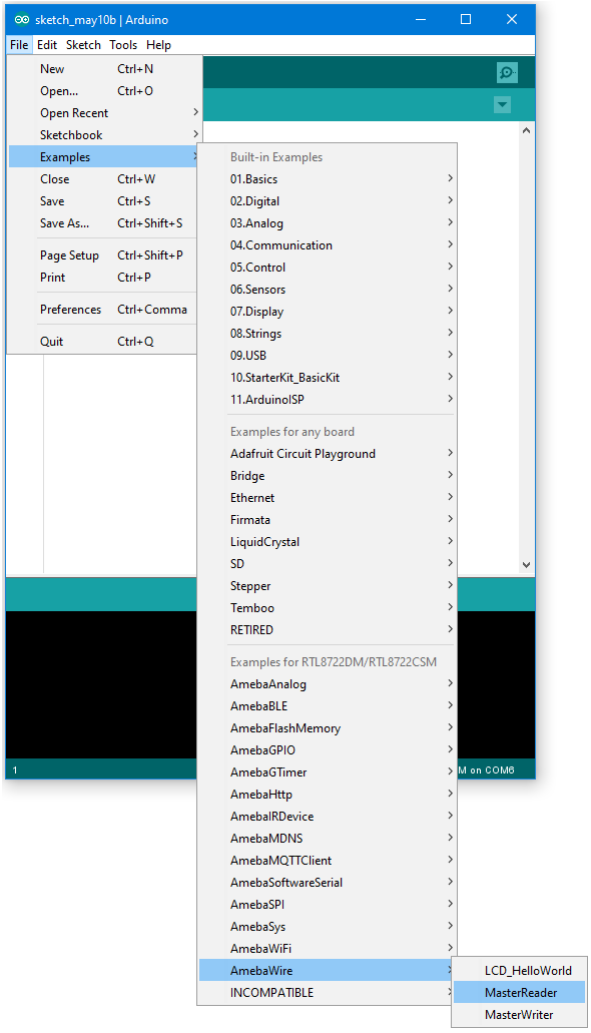


Then click “Sketch” -> “Upload” to compile and upload the example to Arduino Uno.

- **Setting up Ameba to be I2C Master**

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE: “Tools” -> “Board”

Open “`File`” -> “`Examples`” -> “`AmebaWire`” -> “`MasterReader`”

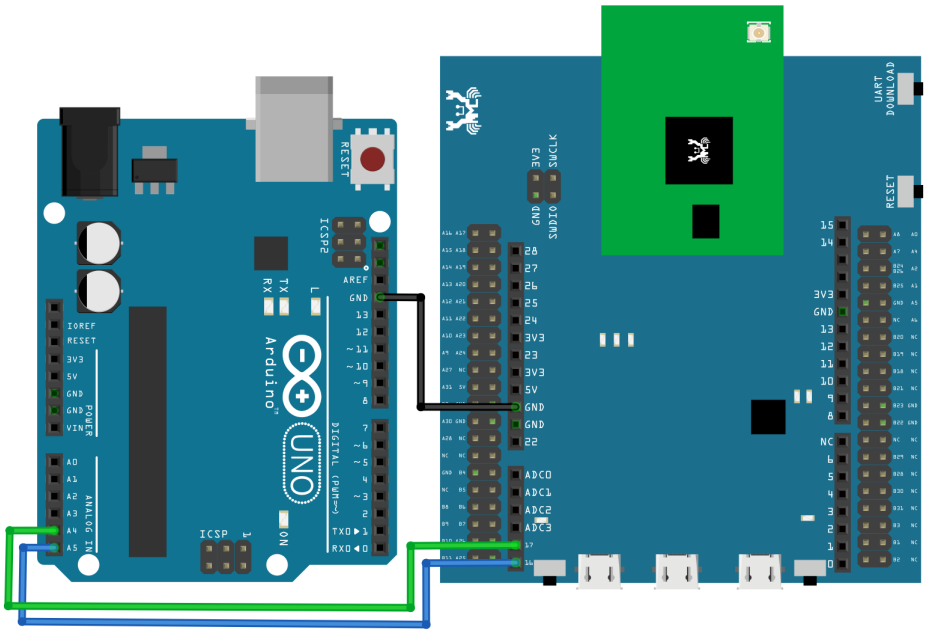


Click “Sketch” -> “Upload” to compile and upload the example to Ameba.

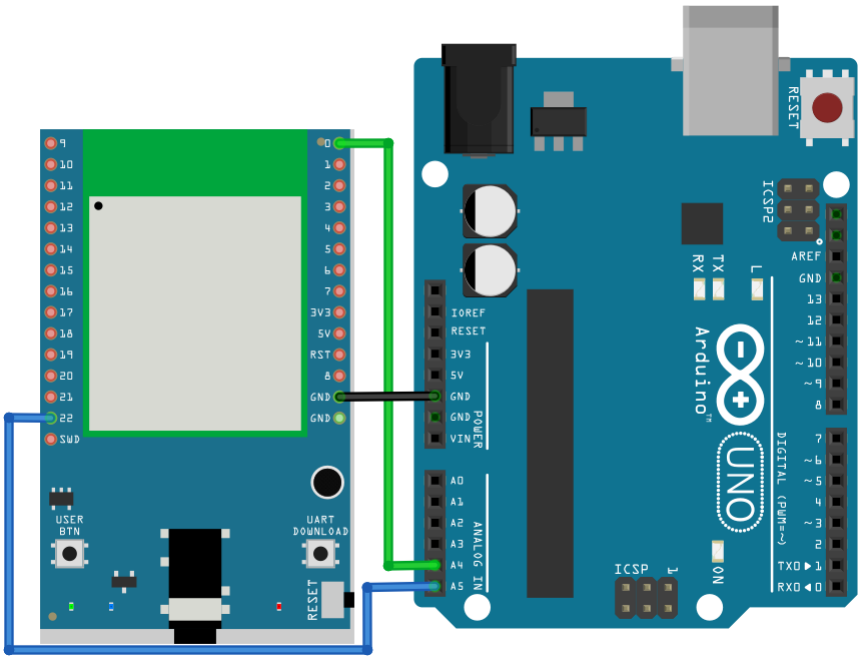
- **Wiring**

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

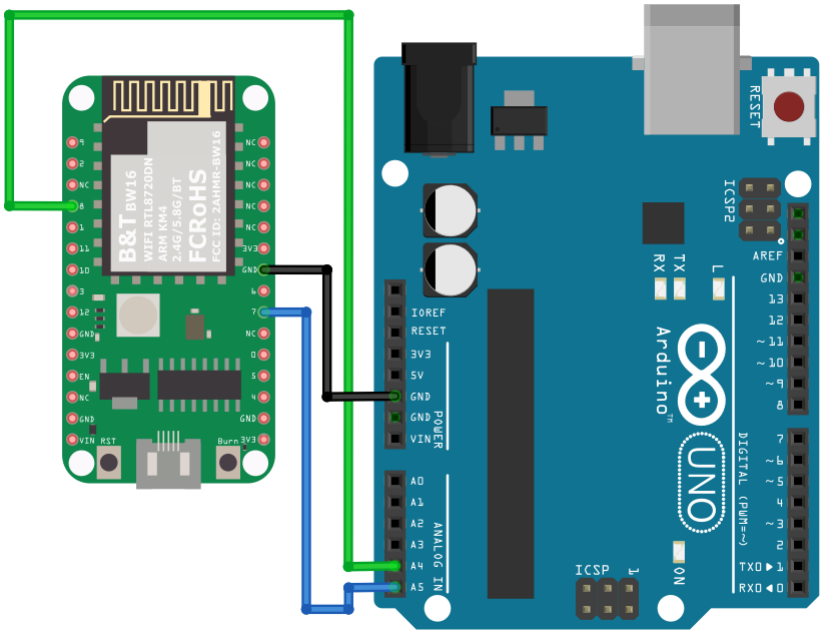
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

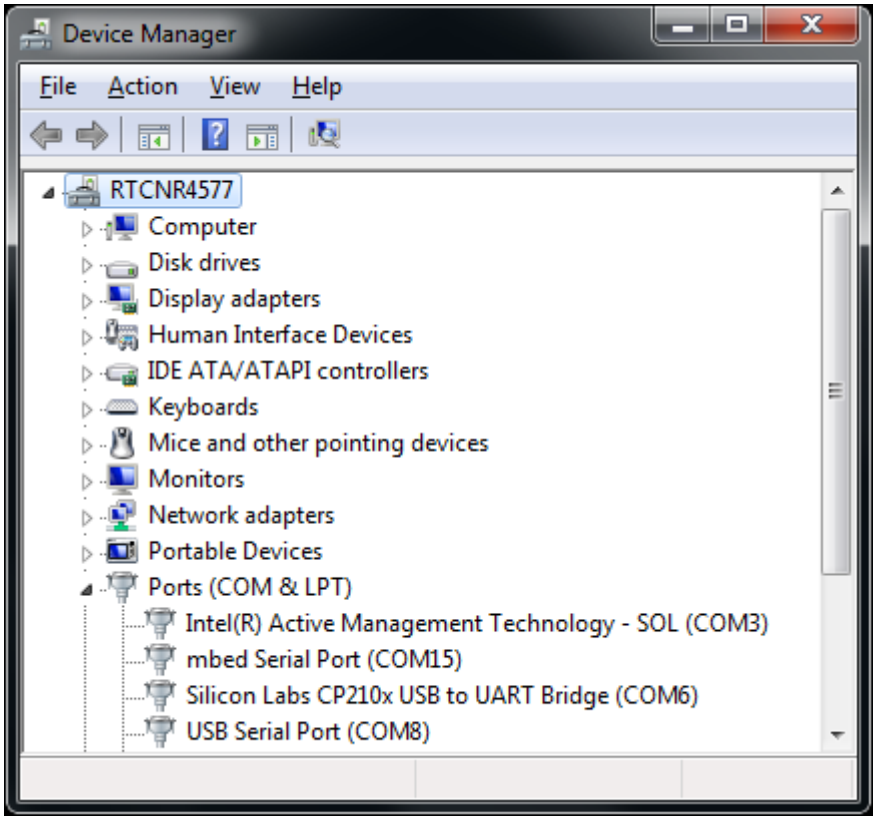


BW16 Wiring Diagram:



Next, we will observe the data receive by Ameba in the Serial Monitor.

(Note: If you do not know which port the Ameba development board is connected to, please find it in the Device Manager of Windows first. Ameba is connected as “mbed Serial Port”. For example, if you find mbed Serial Port (COM15) means Ameba is connected to port COM15.)



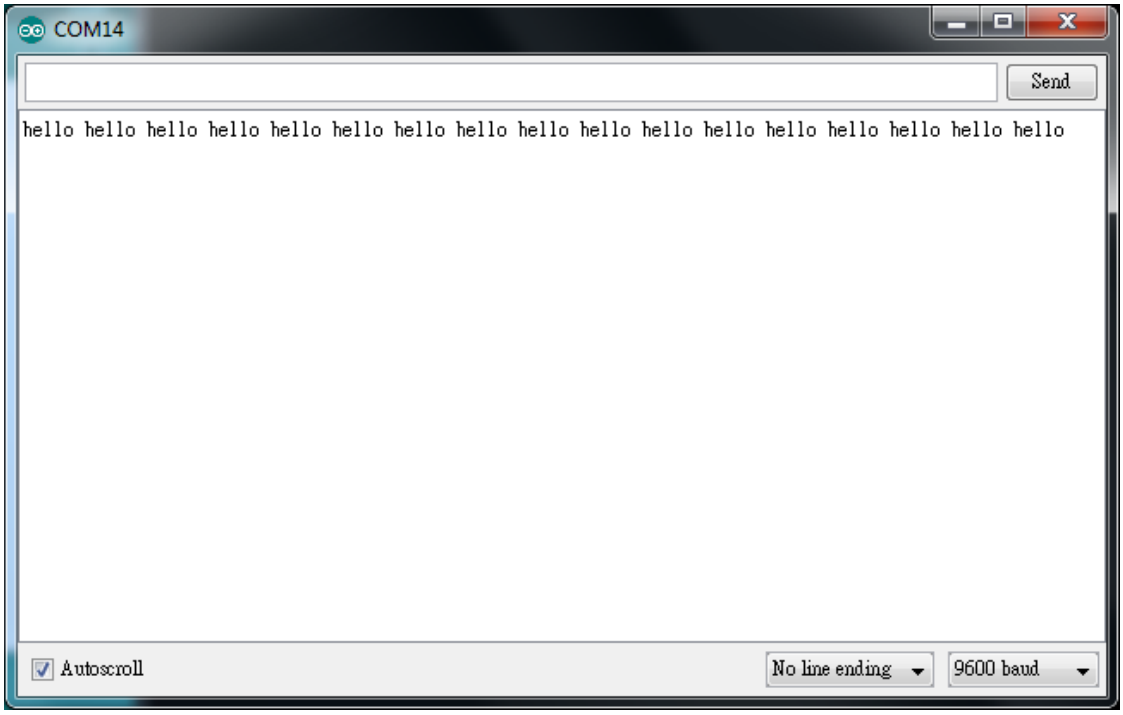
We select the port in “Tools” -> “Port” -> “COM15” (the port connected to Ameba)

Open the Arduino IDE window of the Ameba, go to “Tools” -> “Serial Monitor” to display the messages printed by Ameba.

Press the reset button on Arduino Uno, Arduino Uno now waits for connection from I2C master.

Then press the reset button on Ameba, Ameba will start to receive messages from Arduino Uno. And you can see the “hello ” message printed every half second in serial monitor.

(NOTE: If the message does not show in the Serial Monitor of Ameba, please close and open the serial monitor again.)



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterReader

First use `Wire.begin()` / `Wire.begin(address)` to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses `Wire.requestFrom()` to specify from which Slave to request data.
https://www.arduino.cc/en/Reference/WireRequestFrom

IR - Transmit IR NEC Raw Data And Decode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2
- Grove – Infrared Emitter x1 (Figure 1)
- Grove – Infrared Receiver x1 (Figure 2)

Example

In this example, we use two Ameba RTL8722 modules that connecting with an infrared (IR) Emitter and an IR Receiver separately to transmit and receive IR NEC Raw data.

Figure 1: Grove – Infrared Receiver

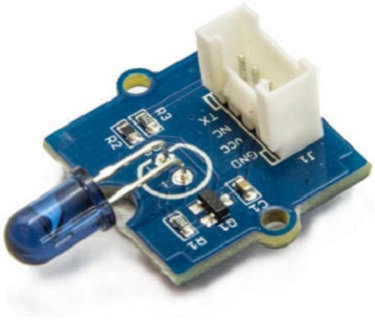


Figure 2: Grove – Infrared Emitter

On the transmission side, the transmitter will send IR NEC raw data. The raw data can be seen as consecutive durations of "marks" and "spaces" (Figure 3) in microseconds (us).

- Mark: a specific period of sending pulses
- Space: a specific period of sending nothing

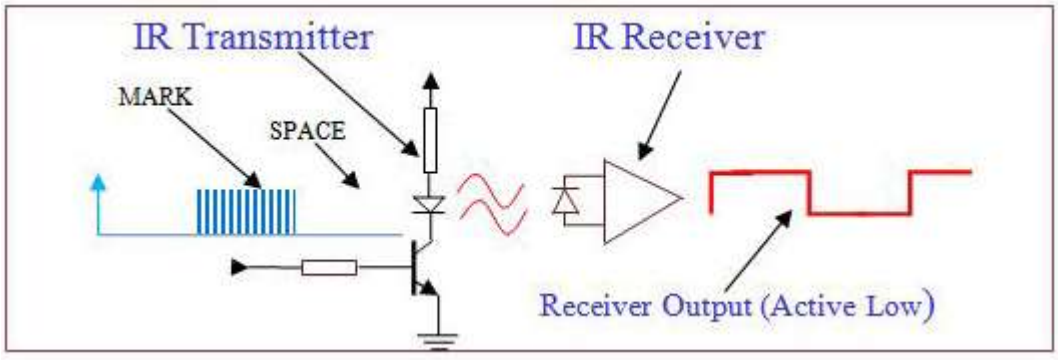


Figure 3: A typical IR transmission and reception setup implementation

For more details, please refer to SB-Projects' topic of IR Remote Control Theory to learn the theory of IR remote controls operation and a collection of IR protocol descriptions. In this example, we are going to use NEC (Now Renesas, also known as Japanese Format) as the transmission protocol.

NEC Features

- 8-bit address and 8-bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- The carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

Modulation

NEC protocol uses Pulse Distance Encoding of the bits for data communication (Figure 4). A logical “1” is represented by total duration of 2250us, with 560us of “marks” and (2250-560) us of “spaces”. While logical ”0” is represented by total duration of 1120us, with 560us “marks” and (1120-560) us of “spaces”.

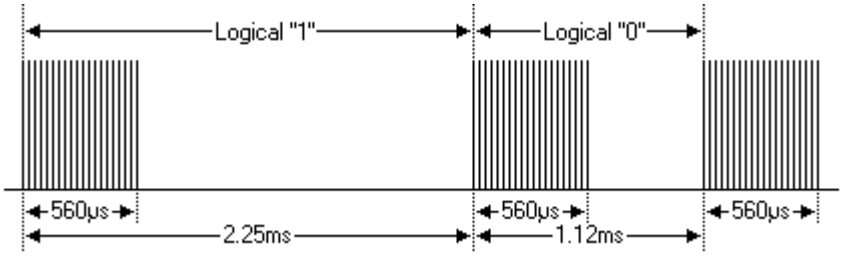


Figure 4: Modulation of NEC

Since a total number of 32-bit data together with the header and the end-bit will be transferred (Figure 5). If we separate the data in the time-frame (in us), there will be (2 + 32) x 2 + 1 = 69 “marks” / “spaces” to be transmitted (Figure 6), which forms the raw NEC data we would like to transmit in our Arduino “*.ino” file. This part of the code can be modified by users. Details of how to obtain raw data code for your remote devices, you may refer to Ken Shirriff’s blog, where it provides multiple libraries provided online.

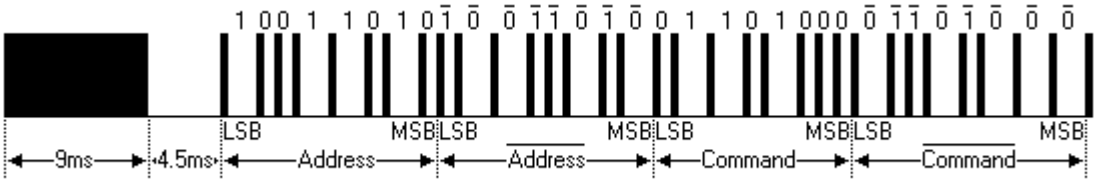


Figure 5: Sample of a Full NEC Data (in logic1 or 0)

```
9000, 4500,                                                                 // starting bit
560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560,       // address
560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~address
560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560,       // data
560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~data
560                                                                         // stoping bit
```

Figure 6: Sample of a Full NEC RAW Data (in us)

Figure 7 and 8 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722 board.

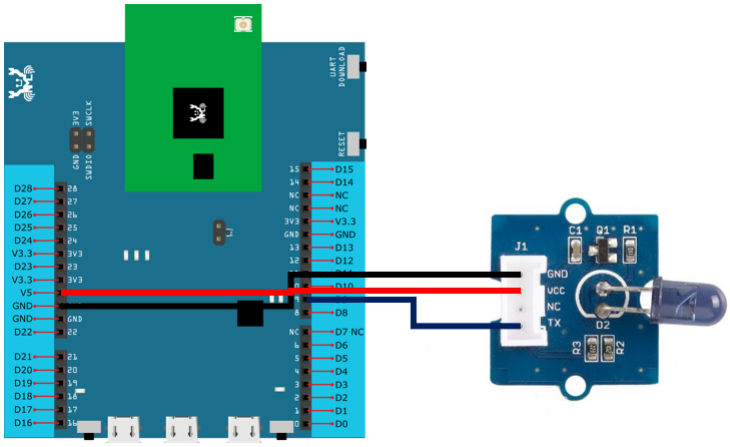


Figure 7: Pin configuration of IR Emitter and AMB21/AMB22

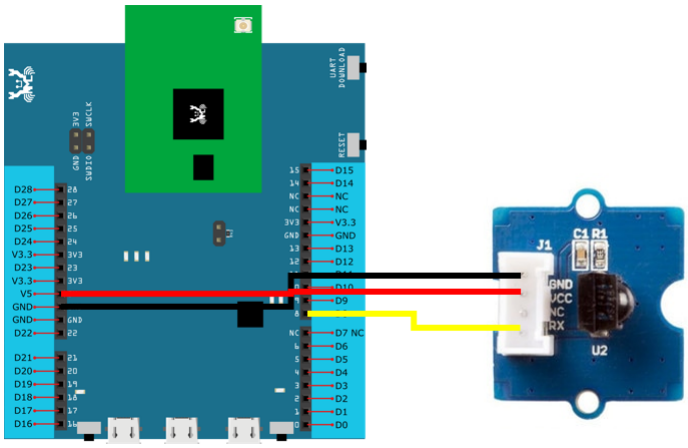


Figure 8: Pin configuration of the IR Receiver and Ameba RTL8722

Figure 9 and Figure 10 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722DM MINI.

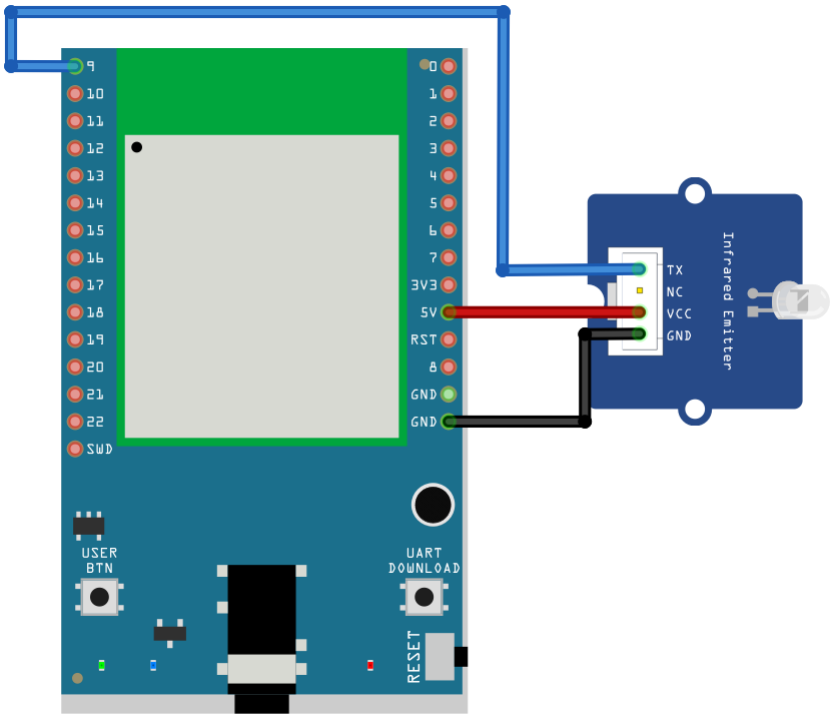


Figure 9: Pin configuration of IR Emitter and AMB23

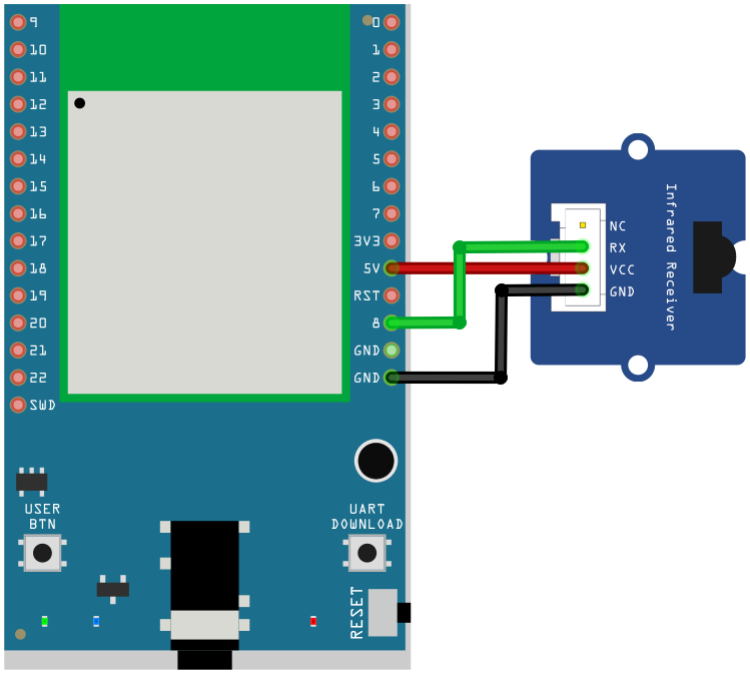


Figure 10: Pin configuration of the IR Receiver and AMB23

Figure 11 and Figure 12 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8720DN (BW16).

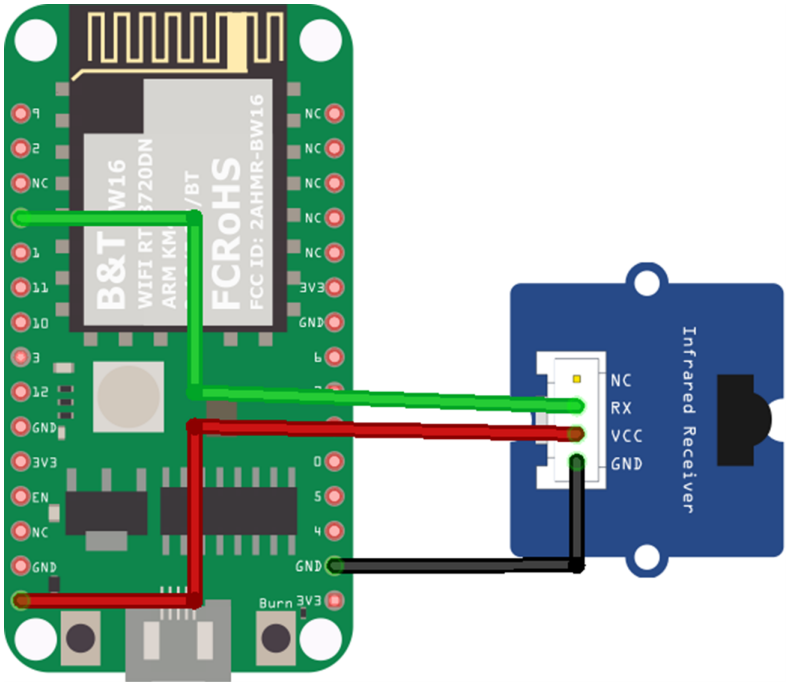


Figure 11: Pin configuration of IR Emitter and BW16

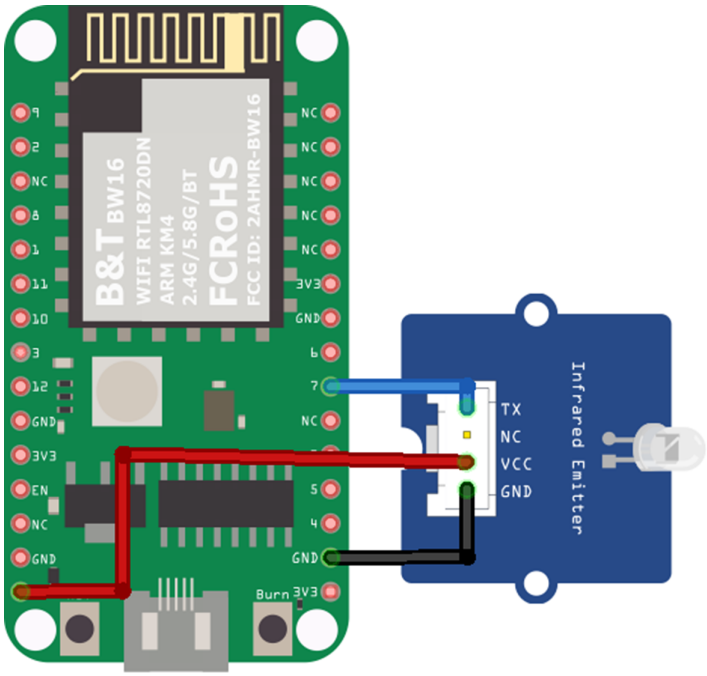


Figure 12: Pin configuration of IR Receiver and BW16

After the connection is being set up correctly, we will move to the coding part for this example. First, make sure the correct Ameba development board is selected in Arduino IDE: “Tools” -> “Board”.

Open the “IRSendRAW” example in “`File`” -> “`Examples`” -> “`AmebaIRDevice`” -> “`IRSendRAW`” (Figure 11) and upload to 1st board connected with IR Emitter:

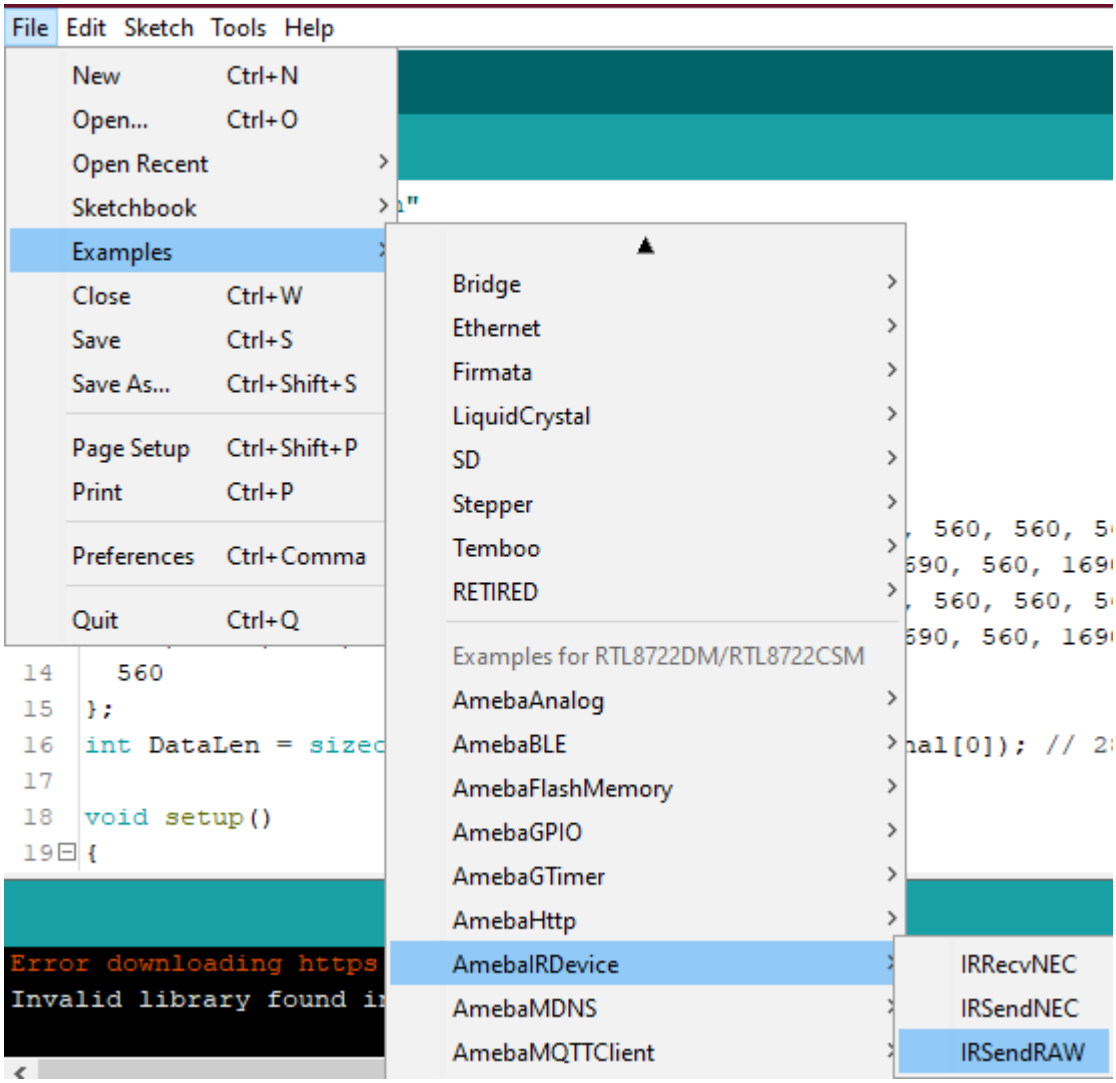


Figure 13: Example Location of IRSendRaw and IRRecvNEC

After successfully upload the sample code for IRSendRaw, you might need to upload the IRRecvNEC example for the 2nd board connected with IR Receiver from "`File`" -> "`Examples`" -> "`AmebaIRDevice`" -> "`IRRecvNEC`".

After opening the serial monitor on the IR Receiver side and press the reset buttons on two boards, the data "48" will be received every 3 seconds (due to the delays () function, not compulsory to wait). After decoding the signal from the receiving Pin D8 and transmitting Pin D9 with Logic Analyser and Pulse View (Figure 10), the result is also shown as "48" after decoding the receiving data with IR NEC Protocol.

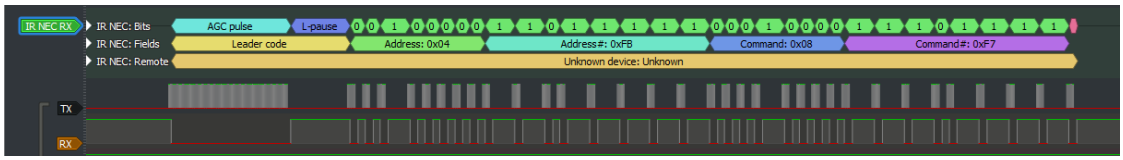


Figure 14: Pulse View results from sending and receiving pin

Code Reference

[1] Seeed Official website for Grove – Infrared Receiver
https://wiki.seeedstudio.com/Grove-Infrared_Receiver/

[2] Seed Official website for Grove – Infrared Emitter
https://wiki.seeedstudio.com/Grove-Infrared_Emitter/

[3] Ken SHirriff's blog on A Multi-Protocol Infrared Remote Library for the Arduino
http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

[4] SB-Projects: IR Remote Control Project
https://www.sbprojects.net/knowledge/ir/index.php

Power Save - Deep Sleep Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

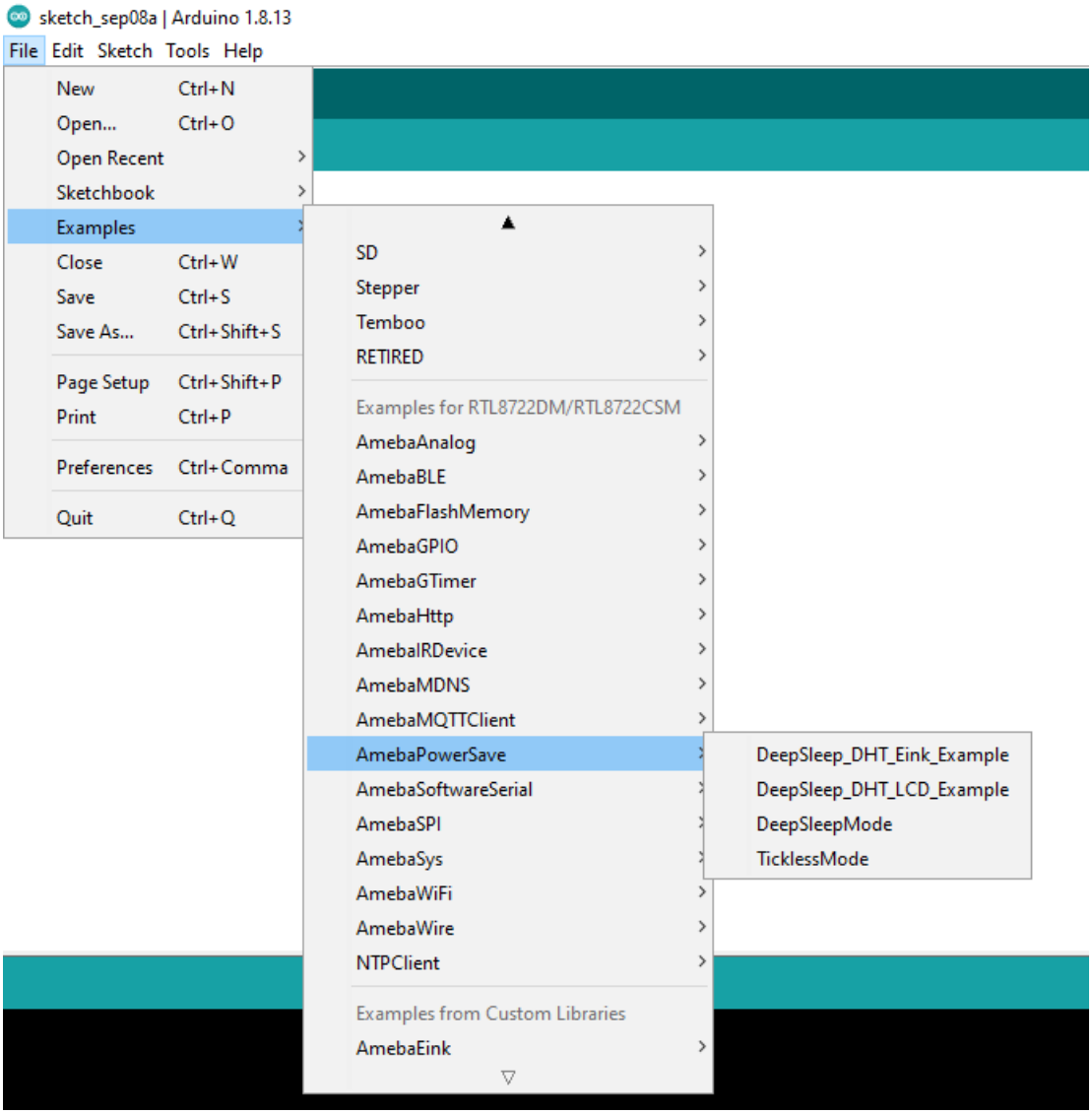
Example

Introduction

Ameba-D supports 2 low power modes which are deepsleep mode and sleep mode. Deep Sleep mode turns off more power domain than sleep mode. The power consumption of Deep Sleep mode is around 7μA to 8μA as compared to normal state which is around 22mA. This example describes how to enter Deep Sleep mode and configure the wakeup source

Procedure

Open "`File` -> "`Examples`" -> "`AmebaPowerSave`" -> "`DeepSleepMode`"



Set condition values as picture below.
"DS_WAKEUP_SOURCE" is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON Timer(SET_DS_AON_TIMER_WAKEUP);
AON GPIO pins(SET_AONWAKEPIN_WAKEUP);
RTC Timer(SET_DS_RTC_WAKEUP);
```

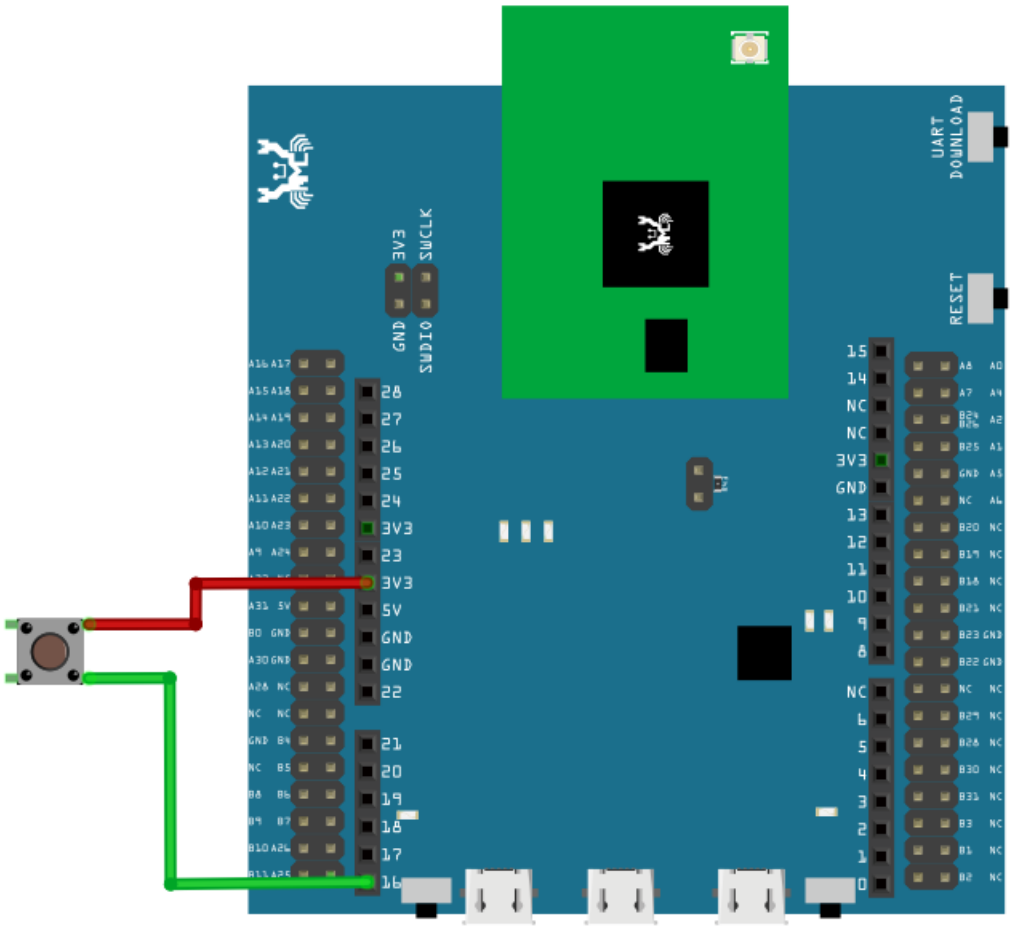
Using AON Timer as wakeup source

AON Timer can be set from 0 to 32760000ms range by `AON_TIMER_SLEEP_DURATION`

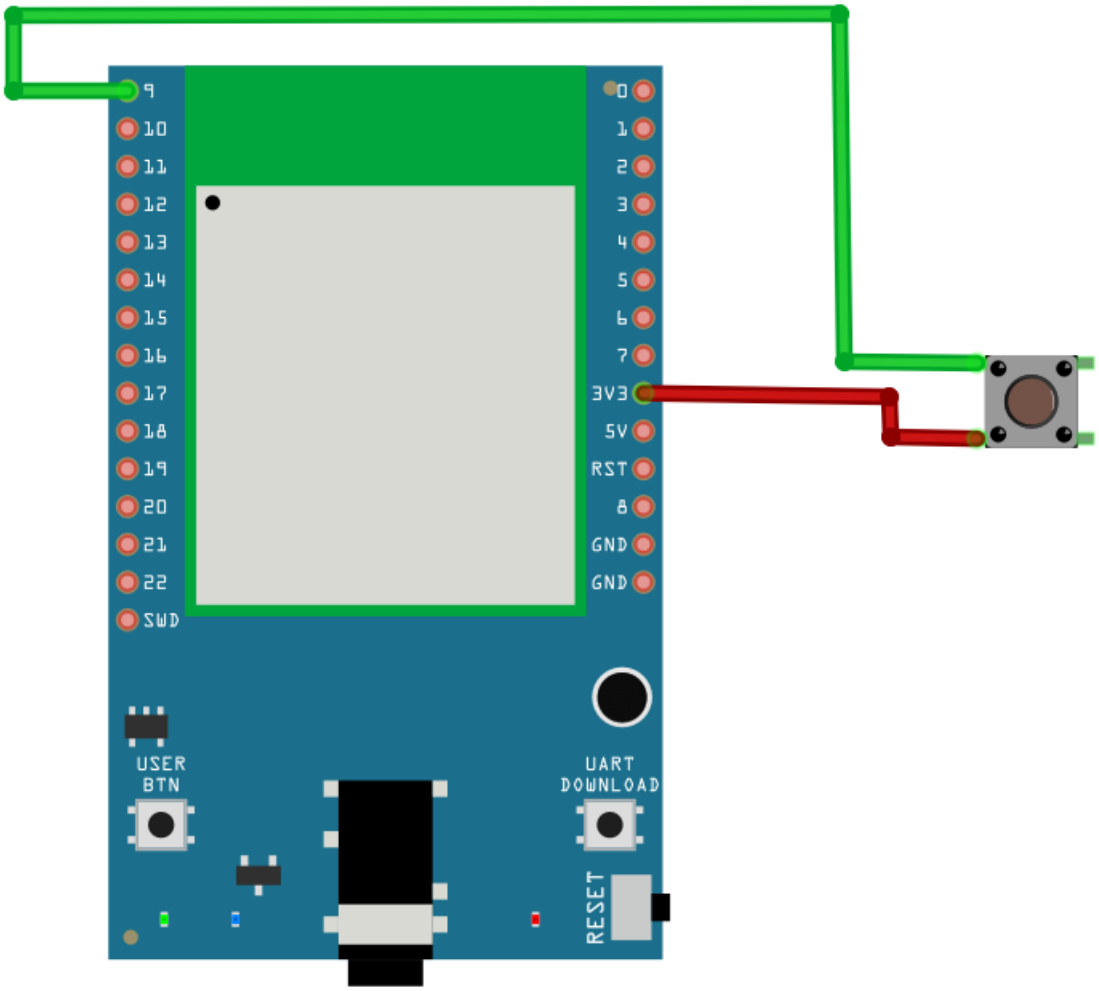
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16),

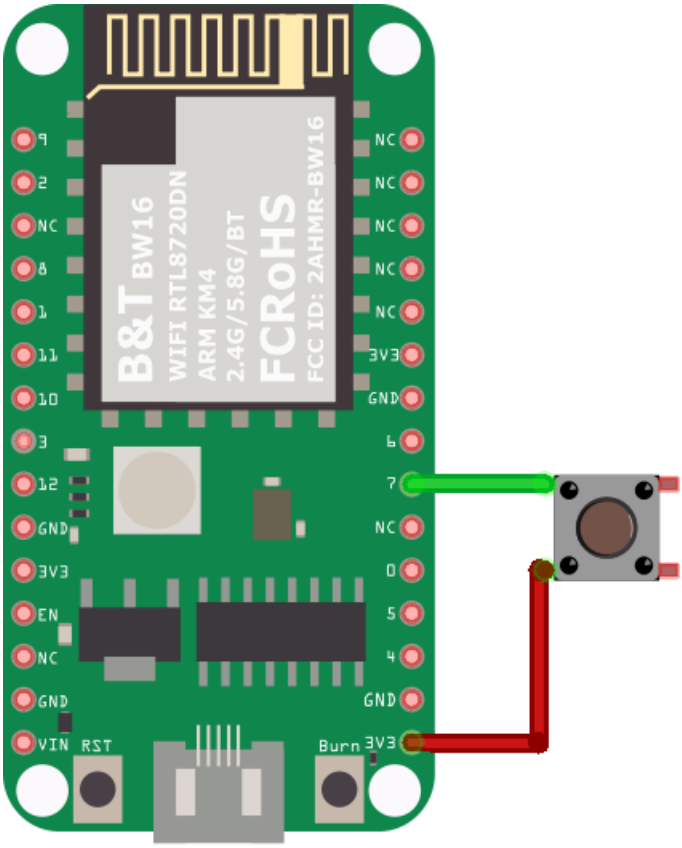
GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).



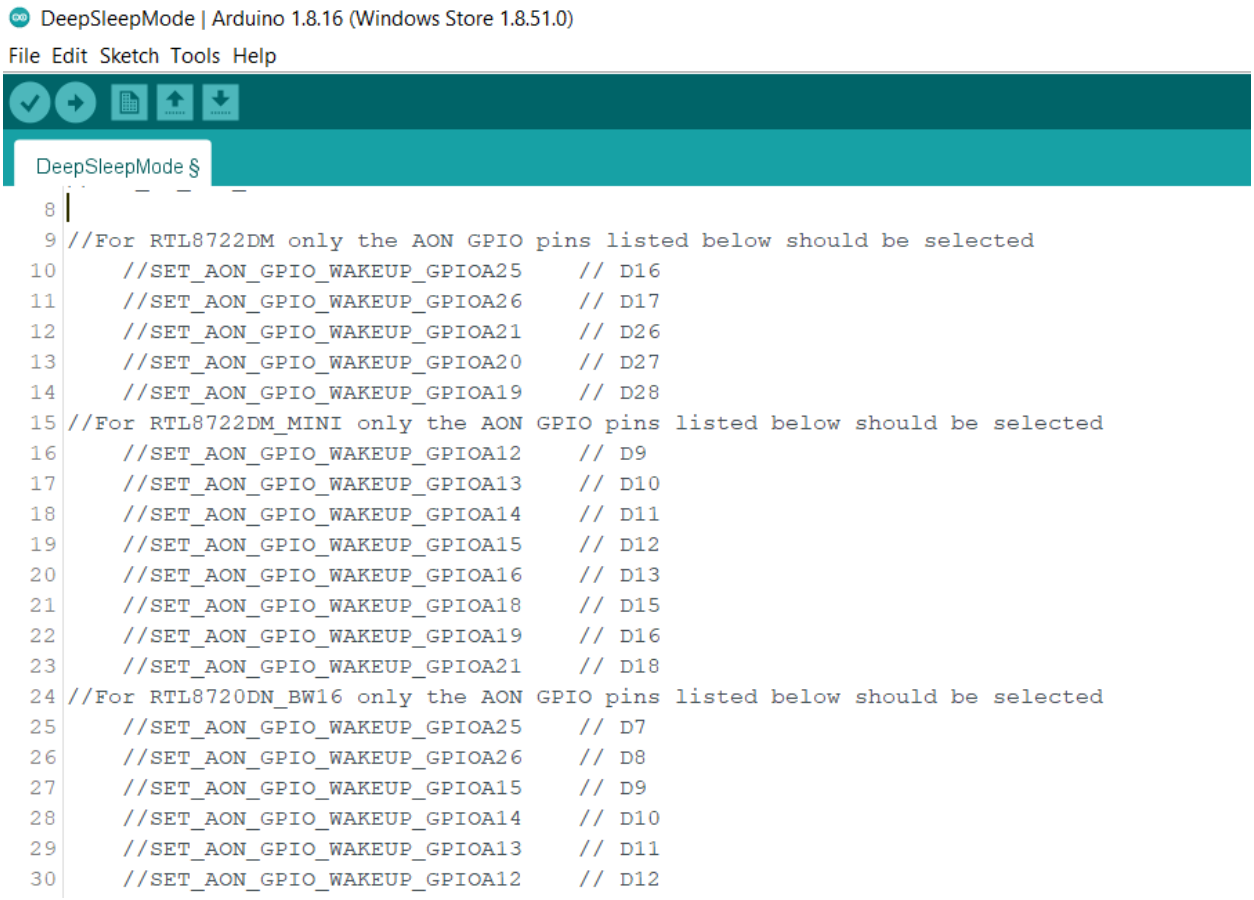
For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

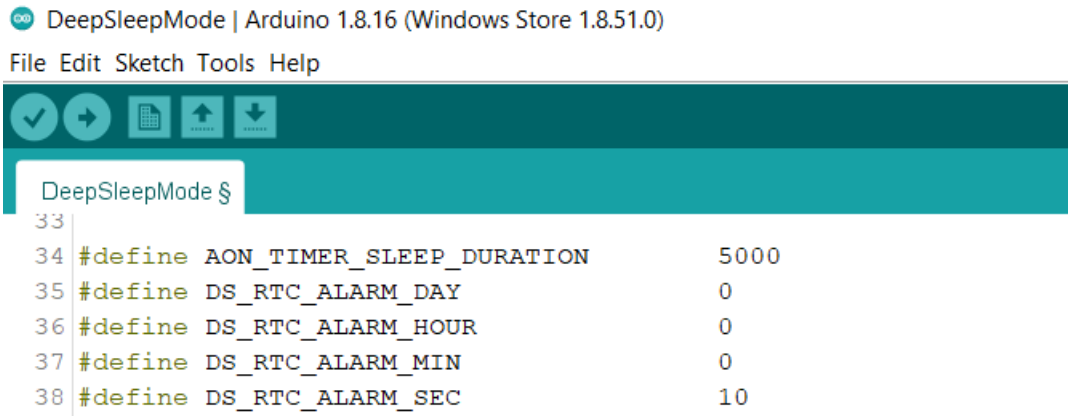


These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below



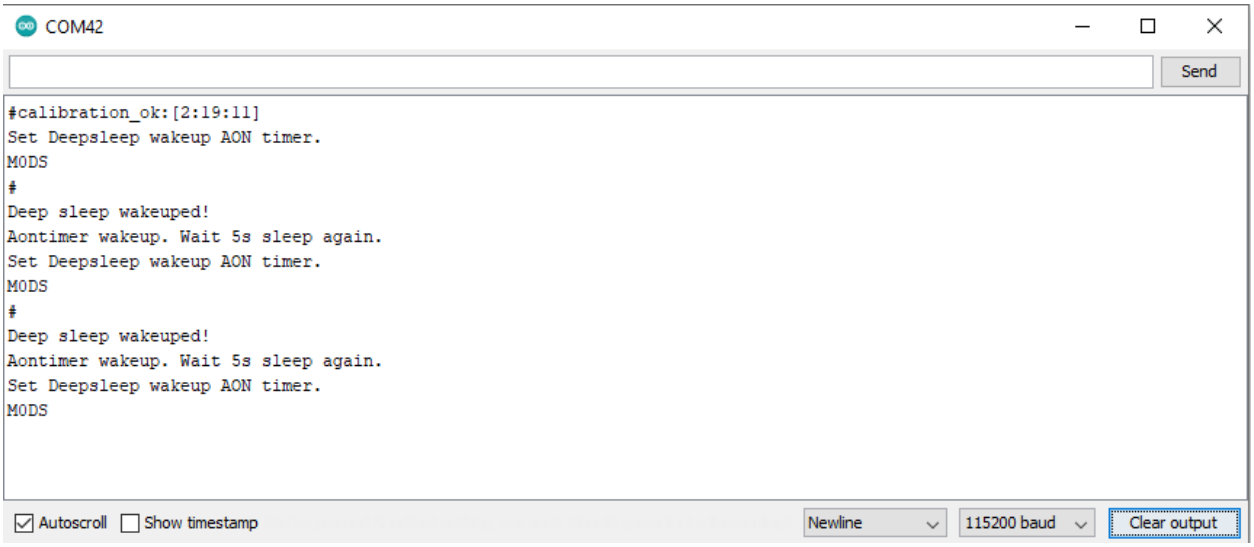
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

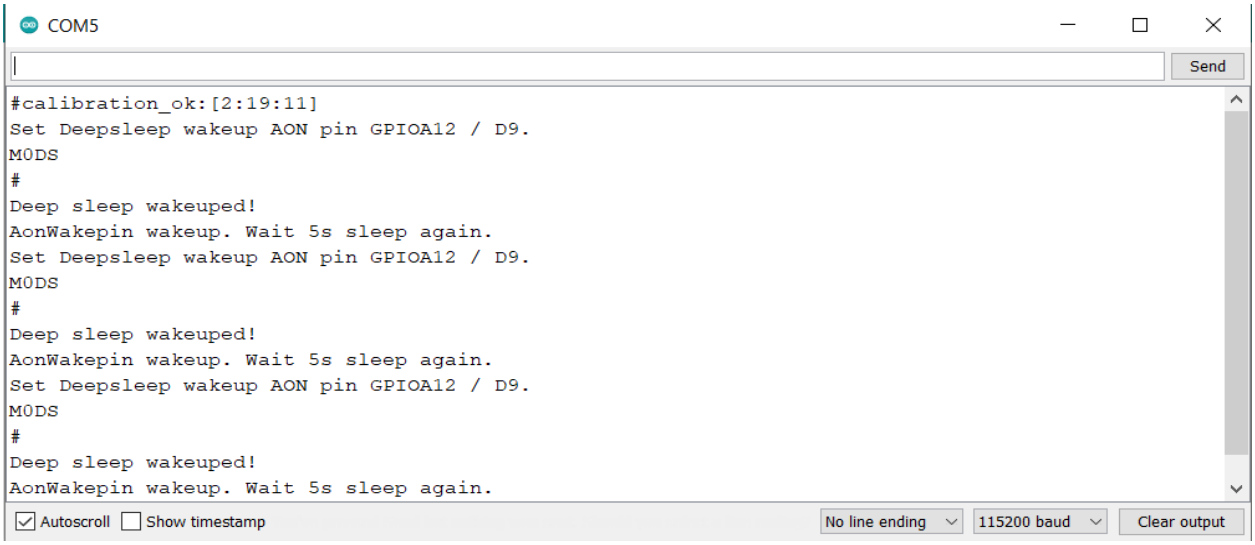


When all the condition values are set, the system will run and switch between normal and deep sleep mode which is controlled by the wakeup source. The serial monitor will display the switching log as shown below.

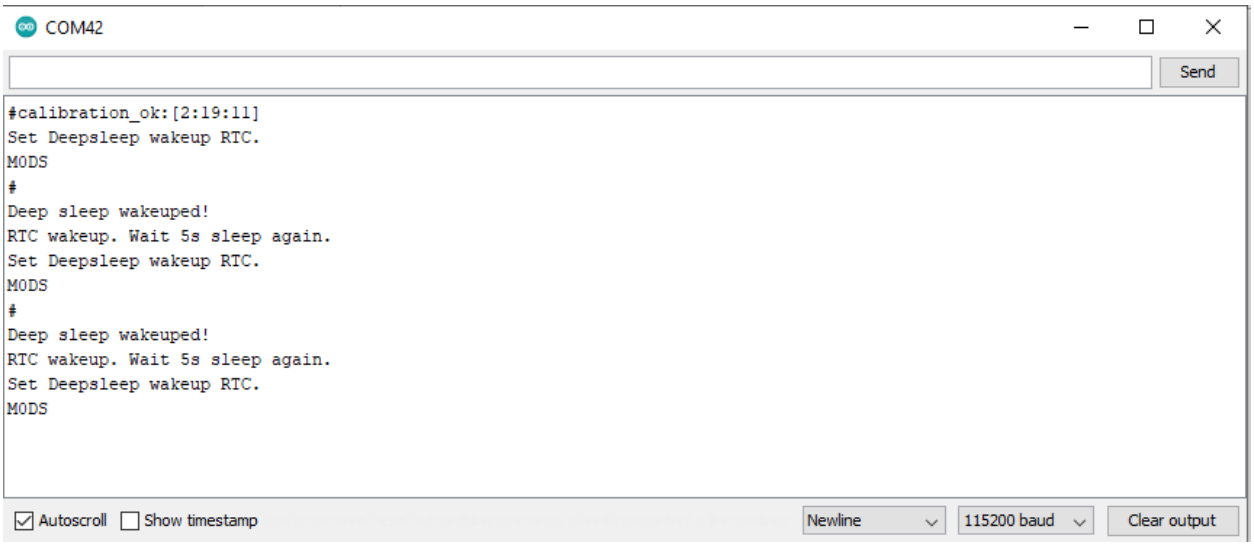
AON Timer



AON GPIO Pin



RTC Timer



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and E-paper

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

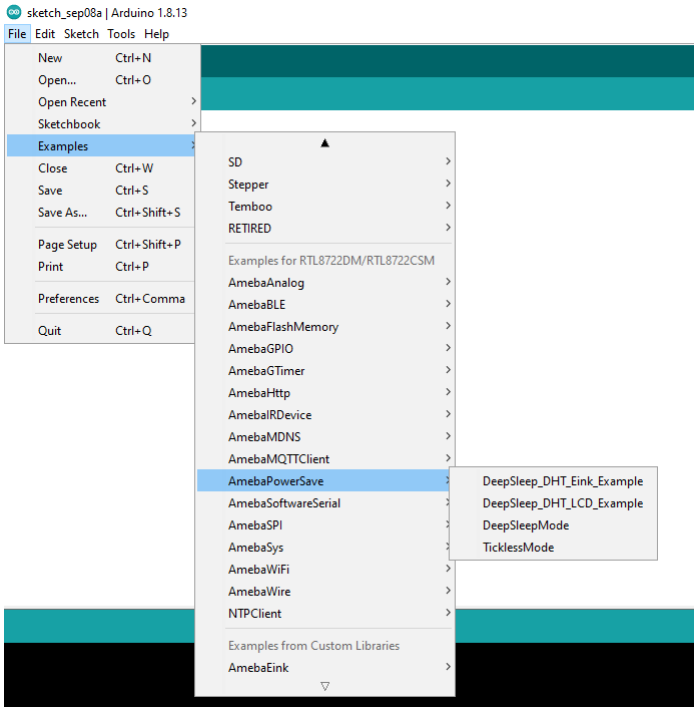
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between “working” and “sleep”(power save).Using DHT sensor to read data and display on Eink screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Download the Eink zip library, AmebaEink.zip, at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries. Then install the AmebaEink.zip.
Open “`File`” -> “`Examples`” -> “`AmebaPowerSave`” -> “`DeepSleep_DHT_Eink_Example`”



Set condition values as picture below.
`DS_WAKEUP_SOURCE` is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

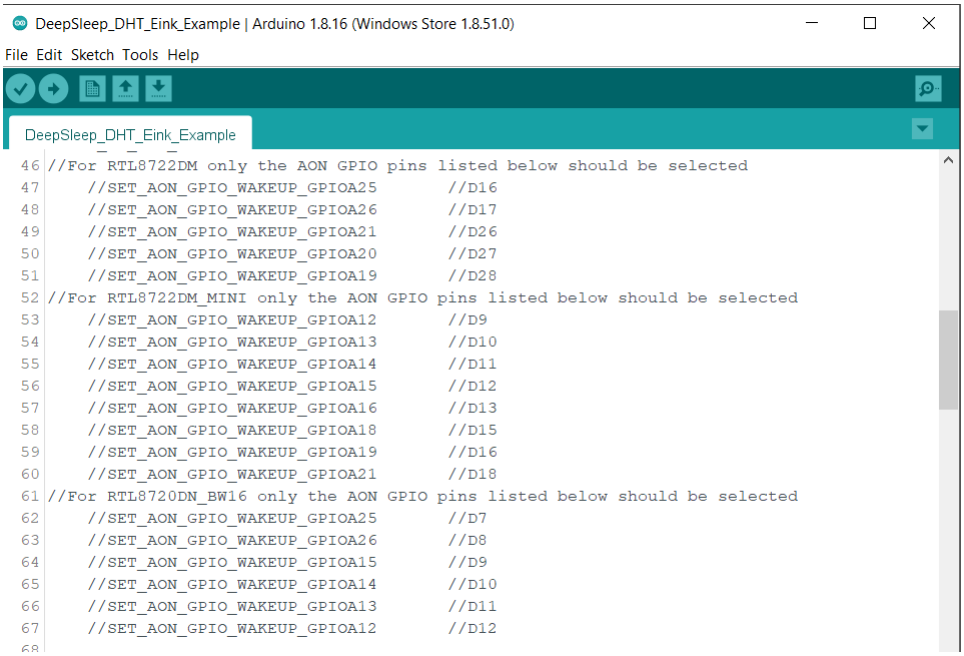
Using AON Timer as wakeup source
AON timer can be set from 0 to 32760000 range (unit ms) by `AON_TIMER_SLEEP_DURATION`

Using AON GPIO pins as wakeup source
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

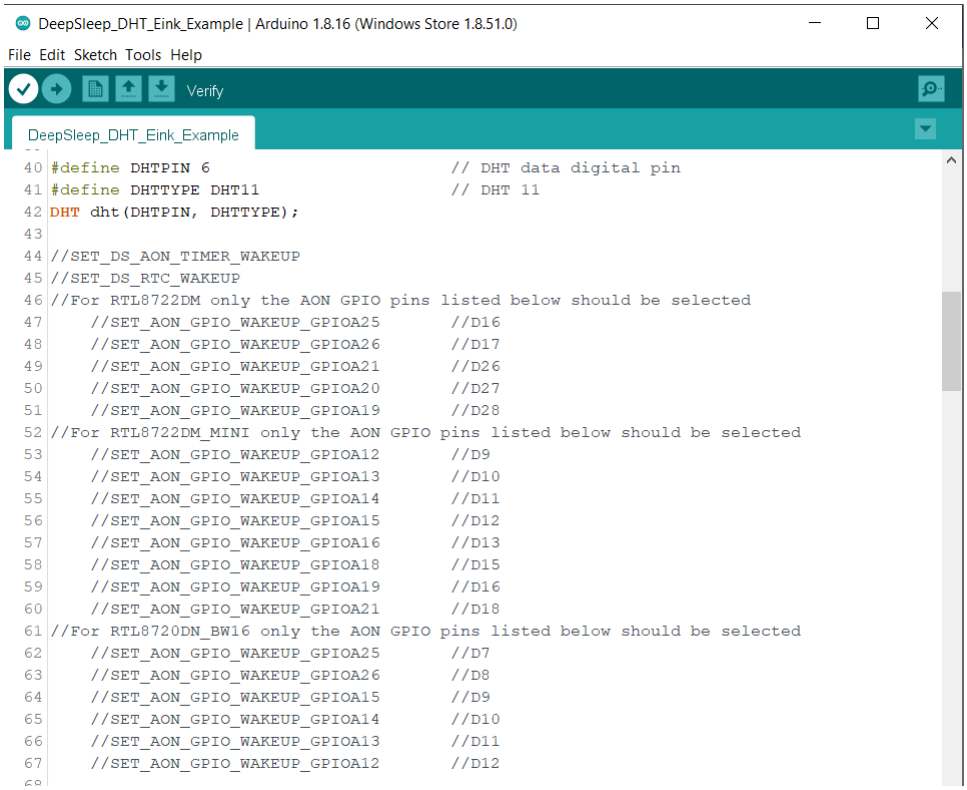
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. Eink screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and LCD

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

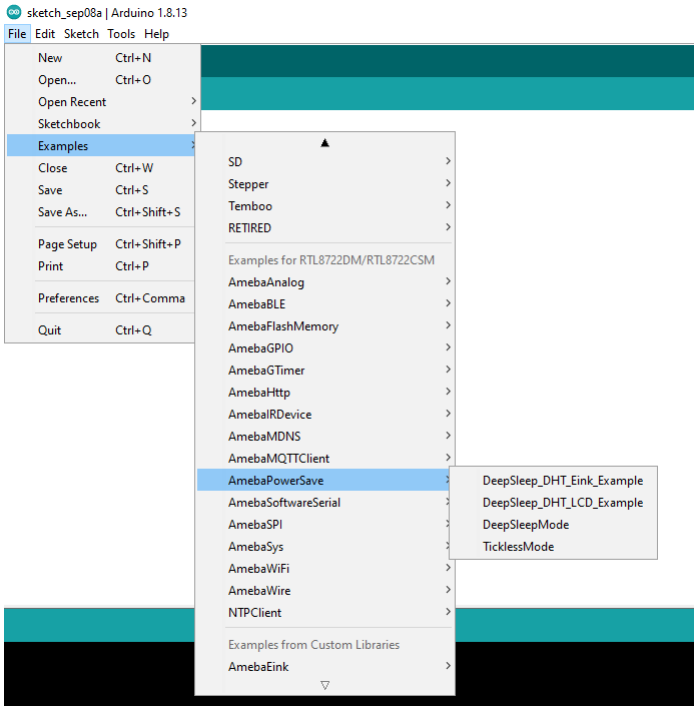
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between “working” and “sleep”(power save).Using DHT sensor to read data and display on LCD screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Open "File" -> "Examples" -> "AmebaPowerSave" -> "DeepSleep_DHT_LCD_Example"



Set condition values as picture below.

DS_WAKEUP_SOURCE is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

Using AON Timer as wakeup source

AON timer can be set from 0 to 32760000 range (unit ms) by AON_TIMER_SLEEP_DURATION

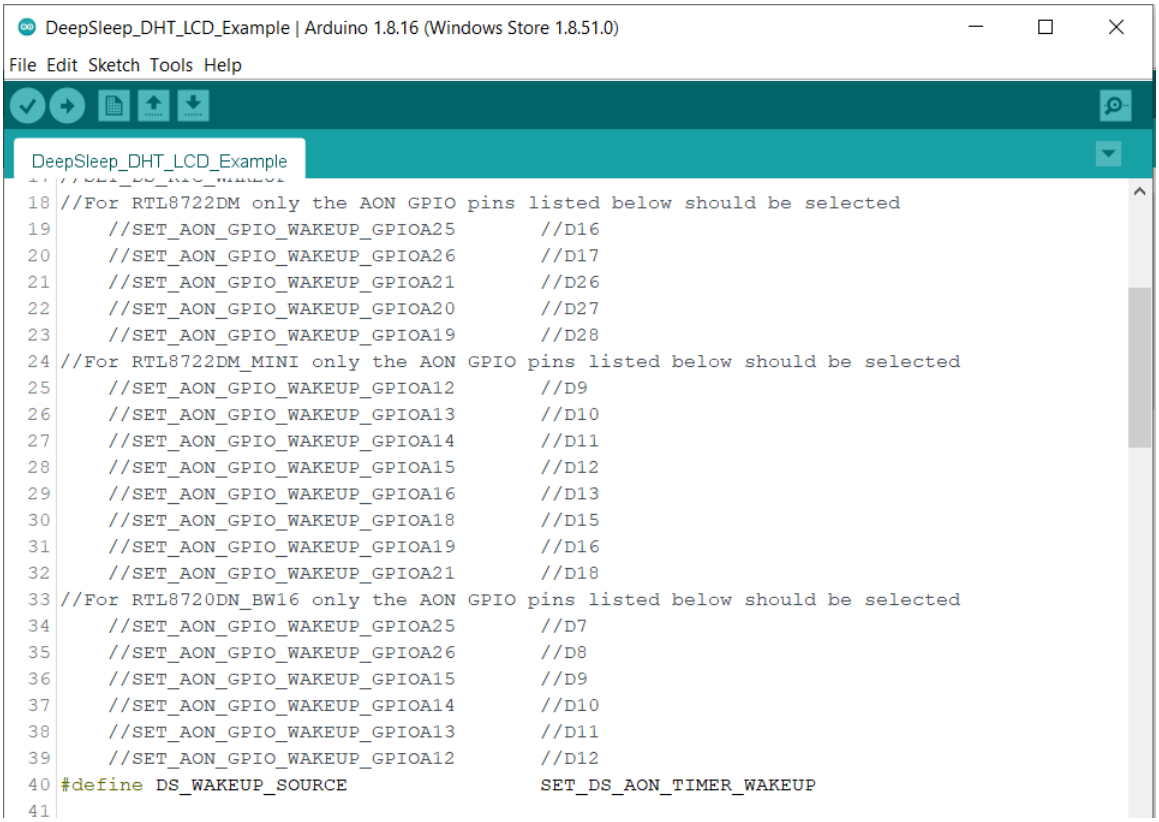
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

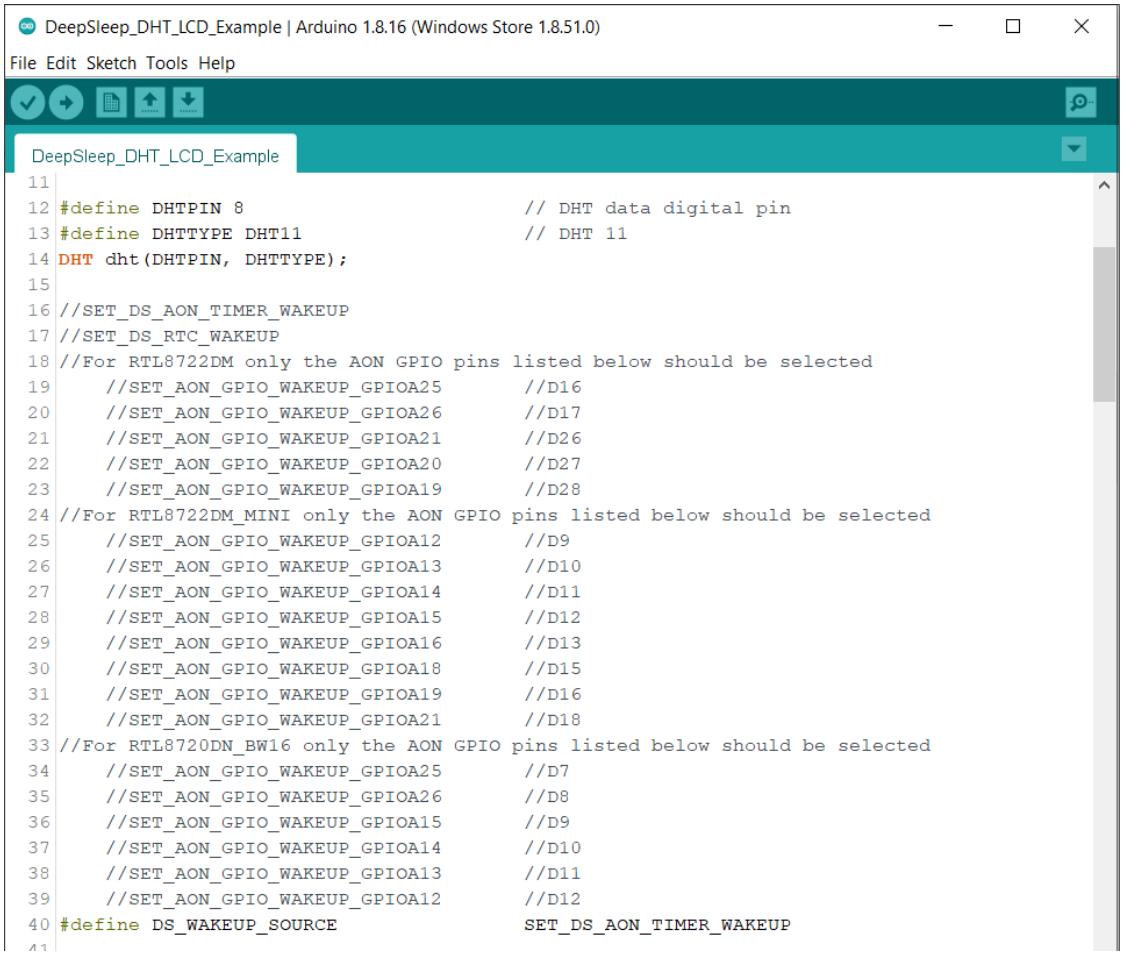
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. LCD screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Tickless Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

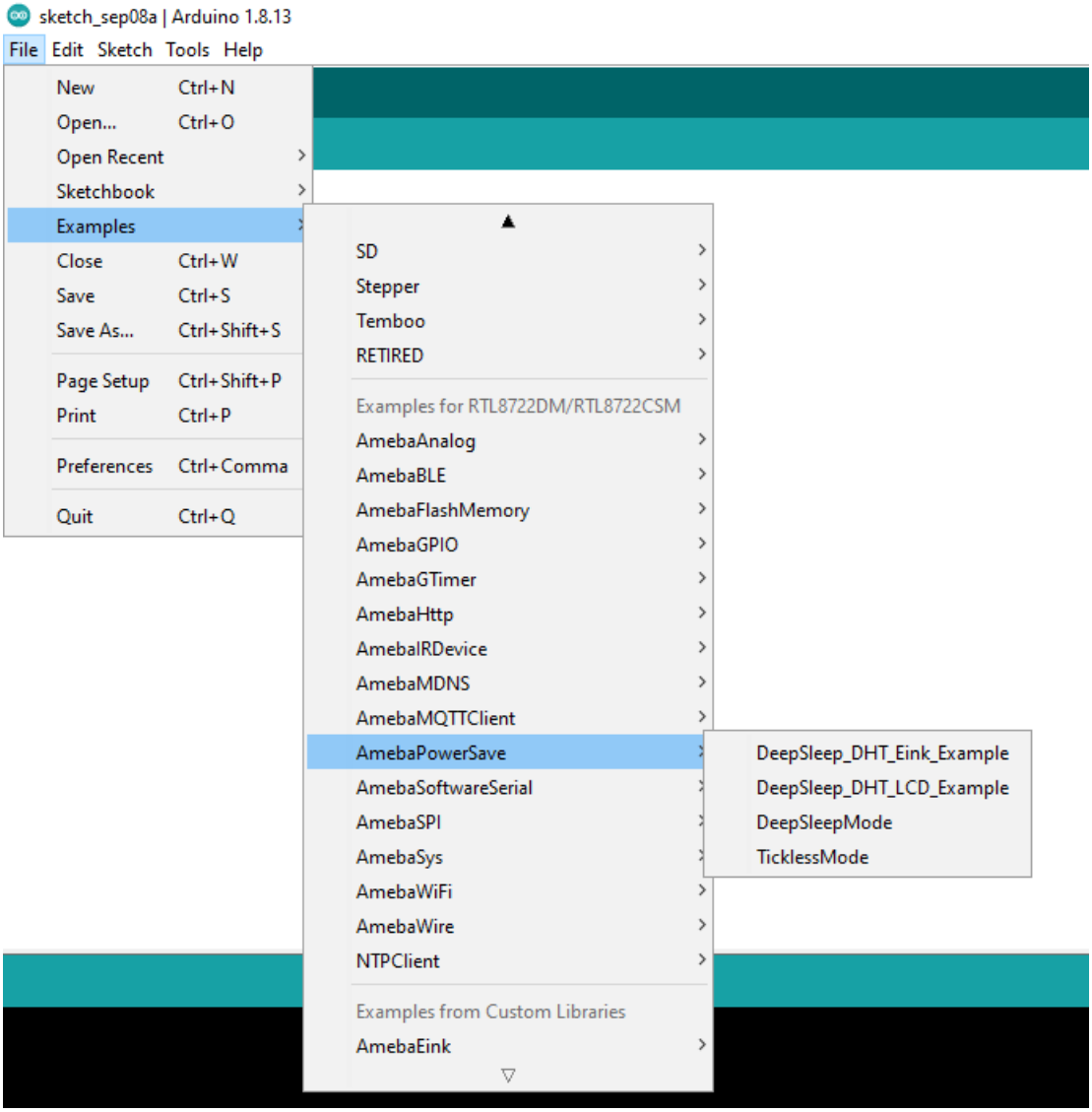
Example

Introduction

Ameba-D supports two low power modes which are deepsleep mode and sleep mode. The power consumptions of Tickless Sleep Mode is around 28uA to 30uA compare to normal state around 15mA. This example describes how to use freertos tickless with uart interruptable interface.

Procedure

Open “File” -> “Examples” -> “AmebaPowerSave” -> “TicklessMode”



Set condition values as picture below.

“TL_WAKEUP_SOURCE” is used to set the wake-up source, user can chose 3 wake up sources now,

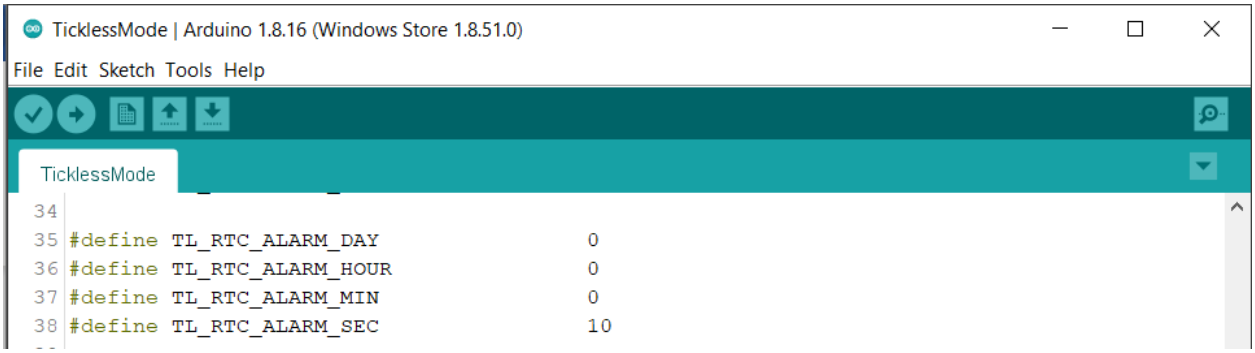
```
LOGUART(SET_TL_UART_WAKEUP);
RTC Timer(SET_TL_RTC_WAKEUP);
AON pins(SET_AON_WAKEPIN_WAKEUP);
```

Using LOGUART as wakeup source

When the LOGUART is selected as the wakeup source, the “TL_Suspend_function” will enter sleep mode. And then it is kept alive for 13s then enter sleep mode. To wakeup, press enter.

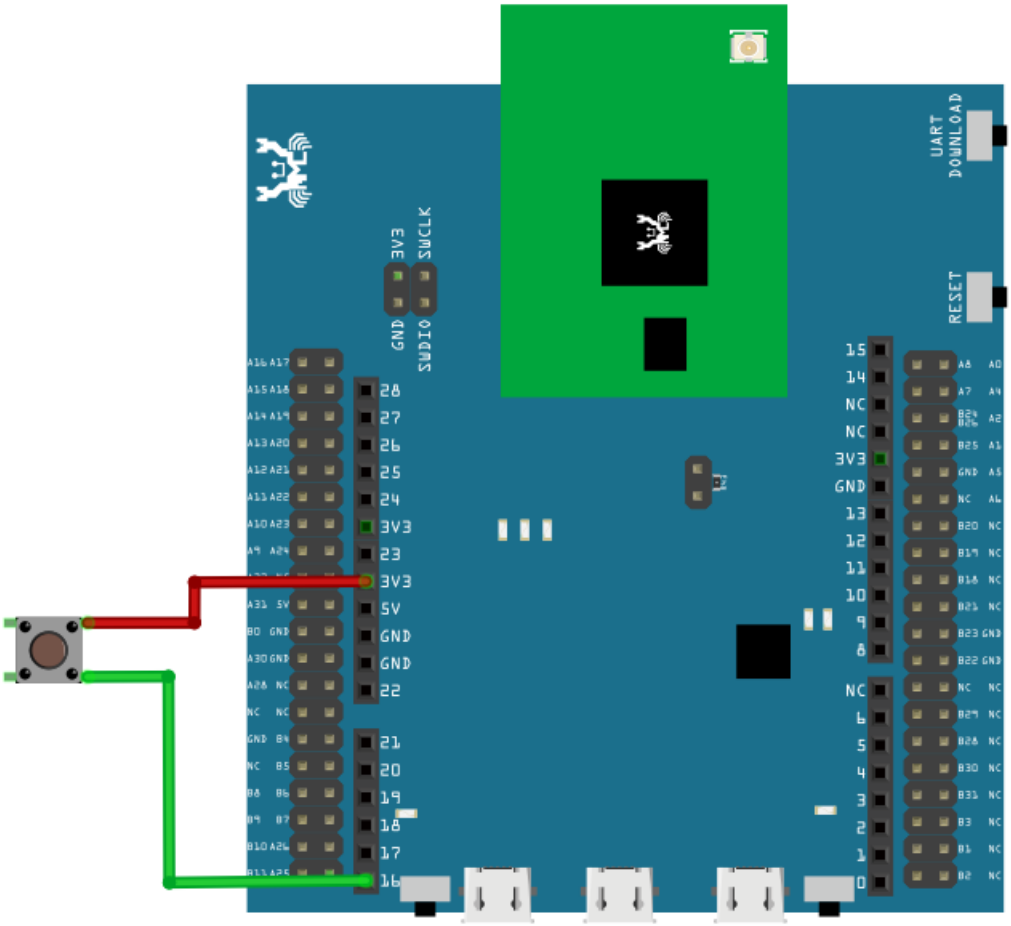
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values to be set, day, hour, min and sec. All 4 values can be set by DS_RTC_ALARM_DAY, DS_RTC_ALARM_HOUR, DS_RTC_ALARM_MIN, and DS_RTC_ALARM_SEC.

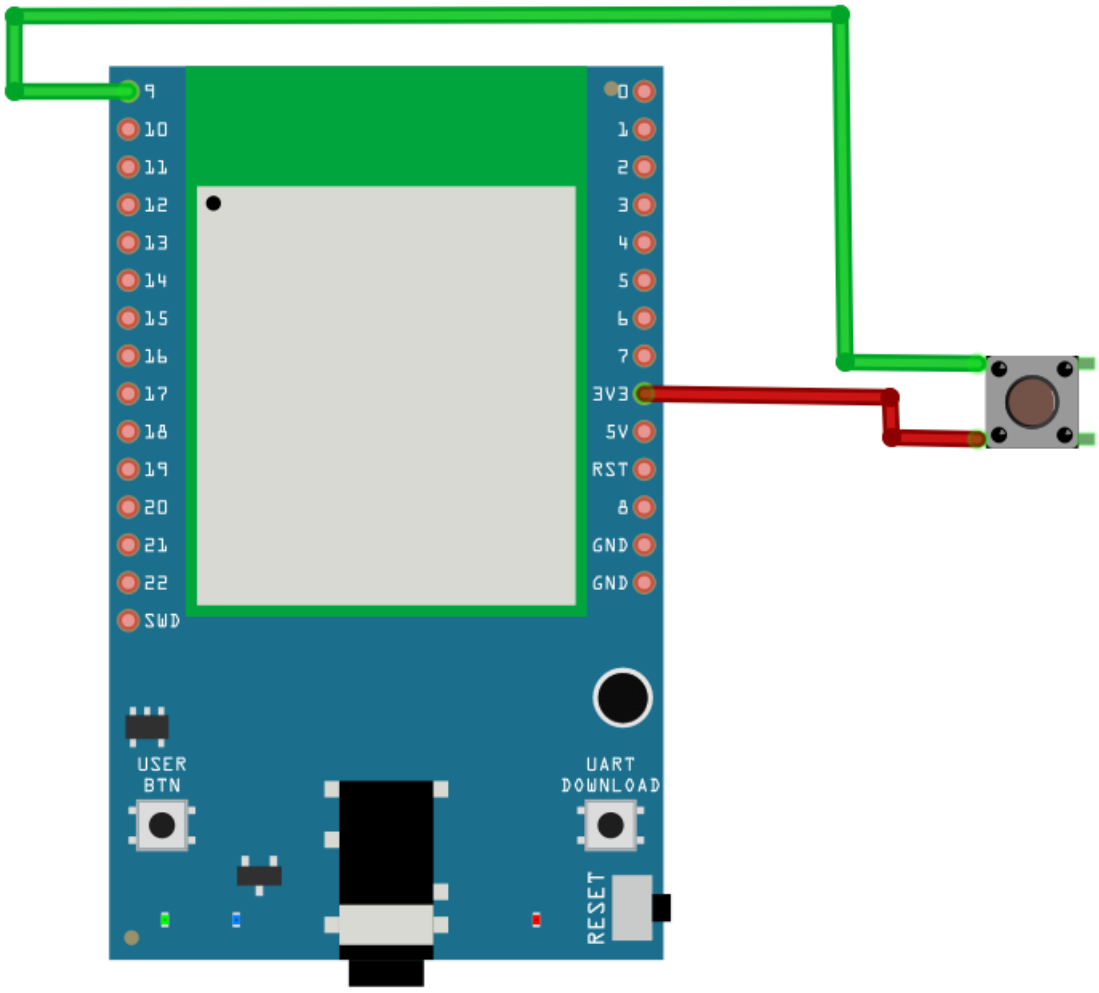


Using AON GPIO pins as wakeup source

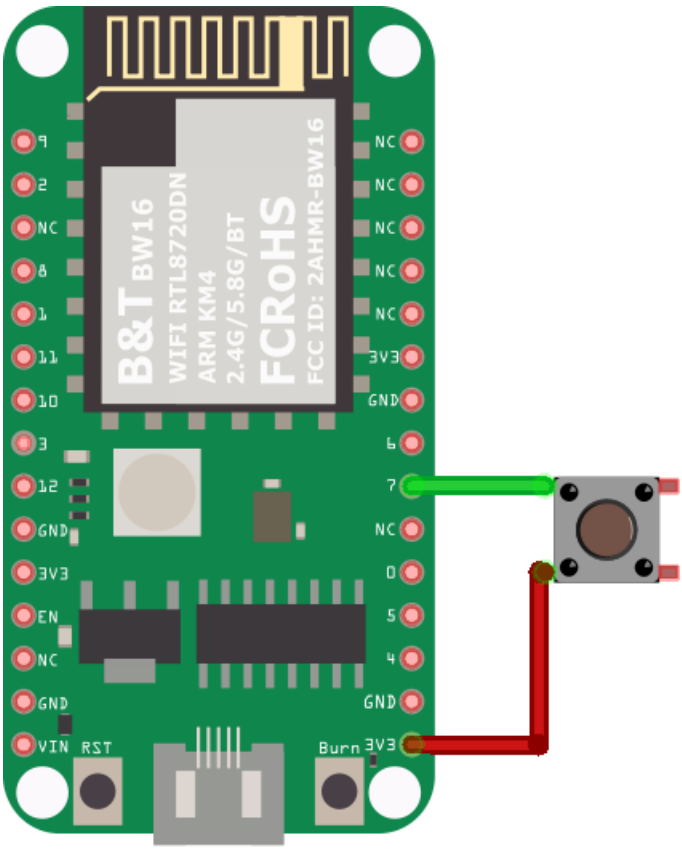
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

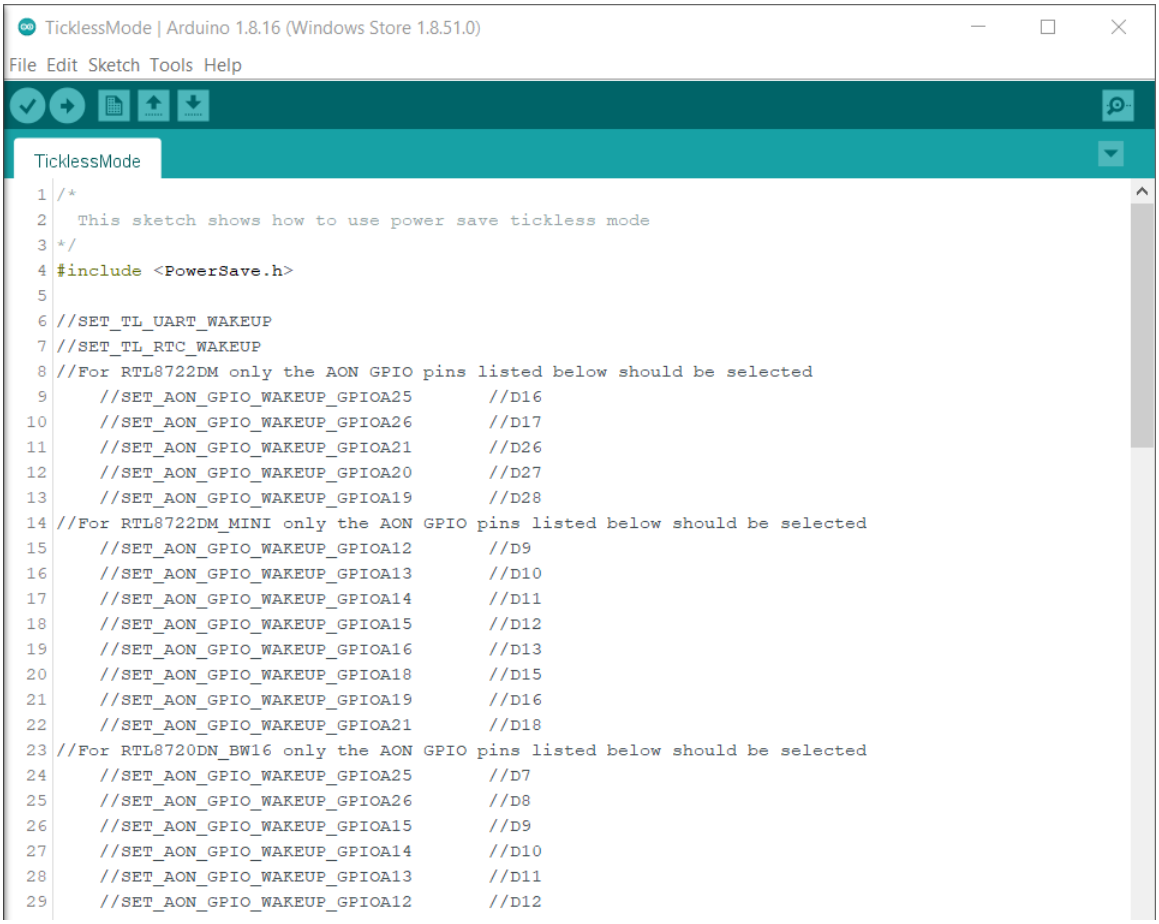


For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



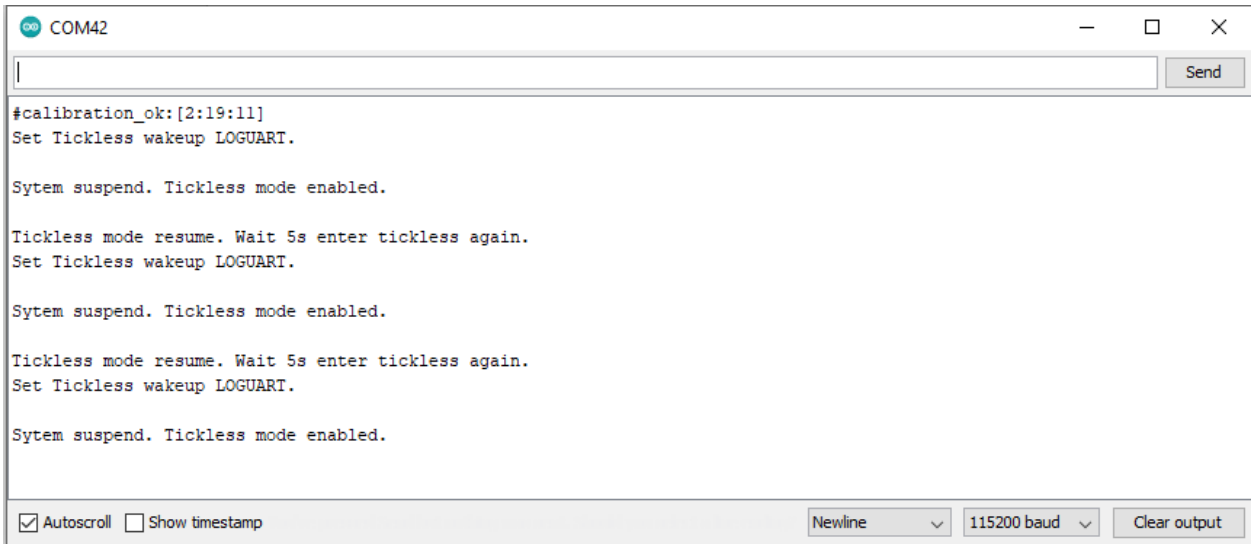
For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).


B&T BW16
WIFI RTL8720DN
ARM KM4
2.4G/5.8G/BT
FCC ID: 2AHMR-BW16
NC
NC
NC
NC
NC
3V3
GND
6
7
NC
0
5
4
GND
3V3
Burn
9
2
NC
8
1
11
10
3
12
GND
3V3
EN
NC
GND
VIN
RST

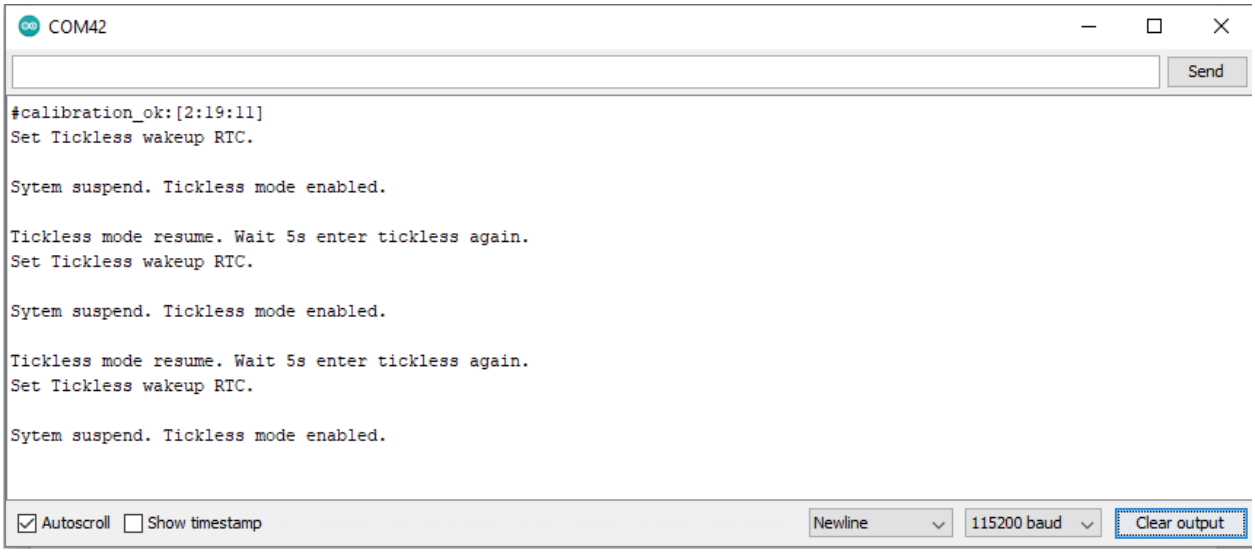


TL_SYSACTIVE_TIME is for setting time duration of the system to keep alive. (Unit ms)

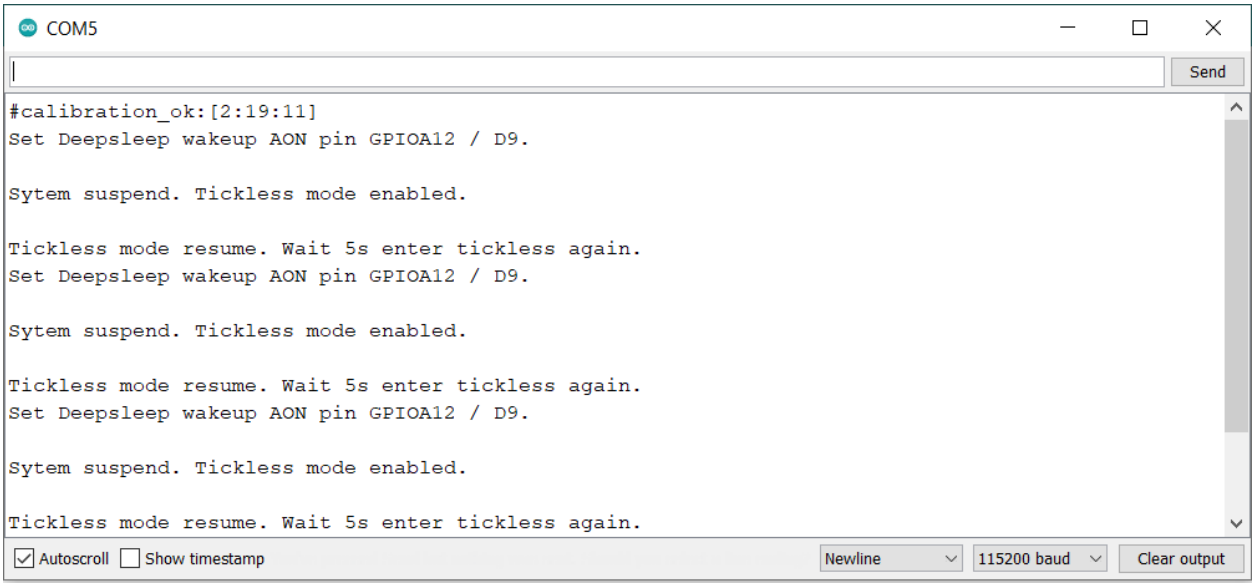
LOGUART



RTC Timer



AON GPIO Pins



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

PWM - Play Music by Buzzer

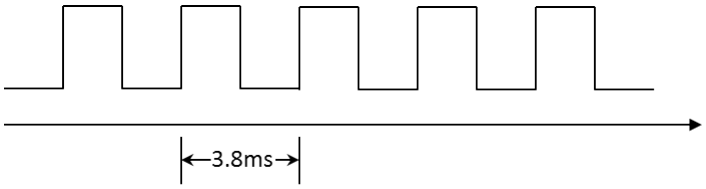
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Buzzer x 1

Example

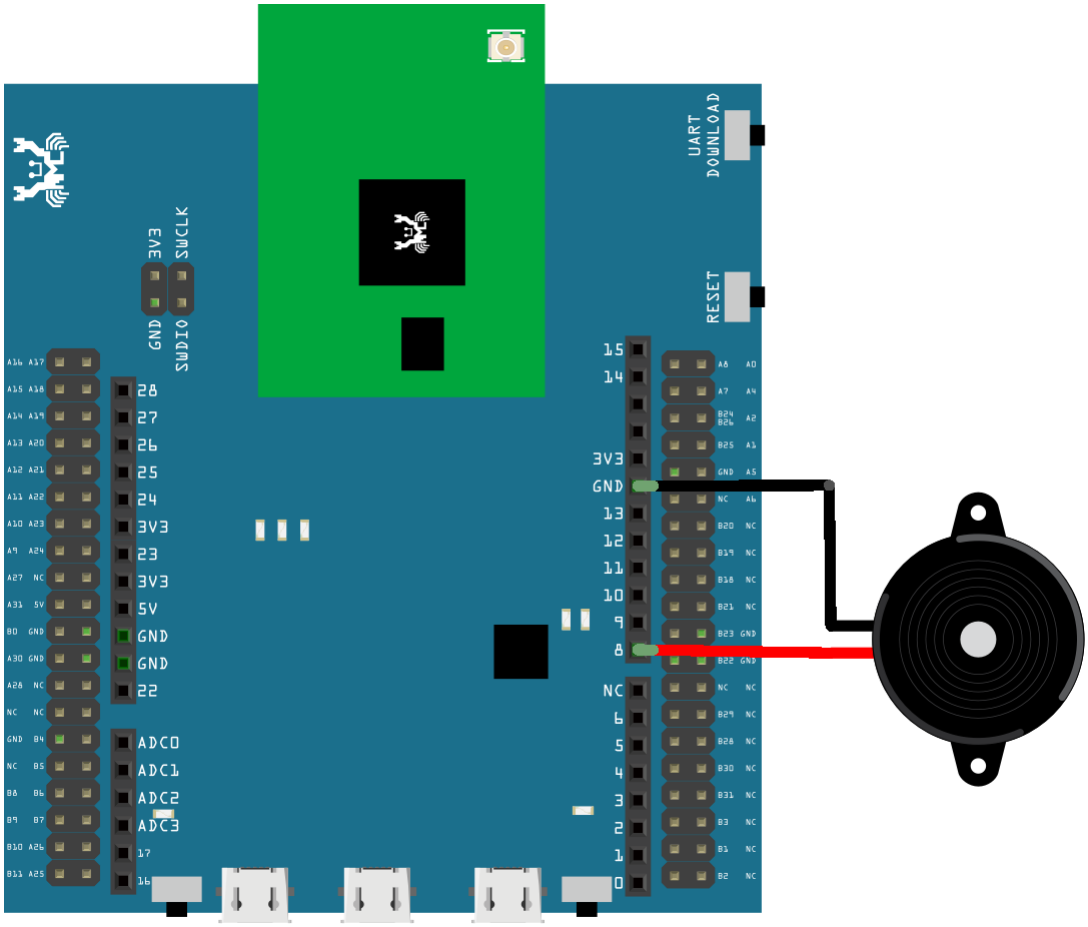
A sound is composed of volume, tone and timbre. Volume is determined by the amplitude of the sound wave. Tone is determined by the frequency of the sound wave. Timbre is determined by the waveform of the sound wave.

In this example, we use PWM to control the buzzer to emit sound with desired tone. As PWM outputs square wave, if we wish to emit tone C4 (frequency=262Hz), we have to make PWM to output square wave with wavelength 1/262 = 3.8ms:

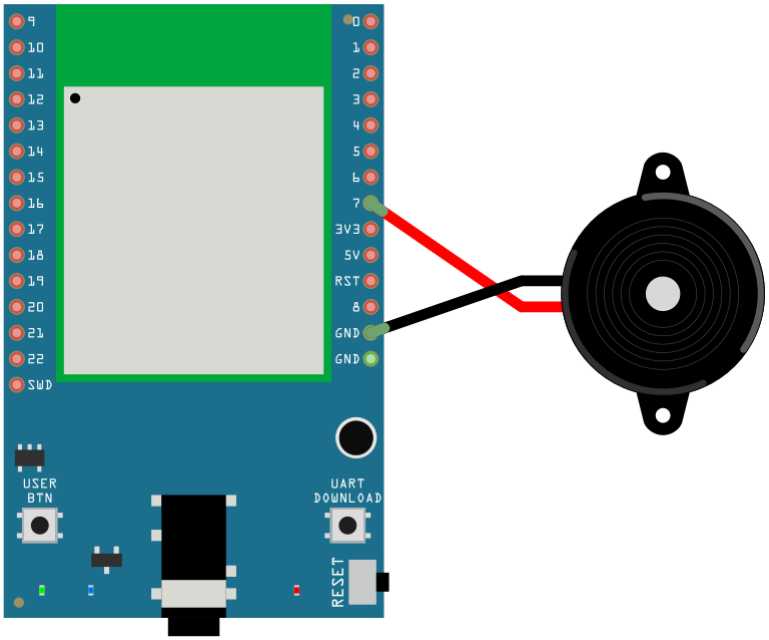


We use PWM to output sound wave with different frequency, so as to play music by the buzzer.
Connect the buzzer to the PWM output pin shown in the following diagrams.

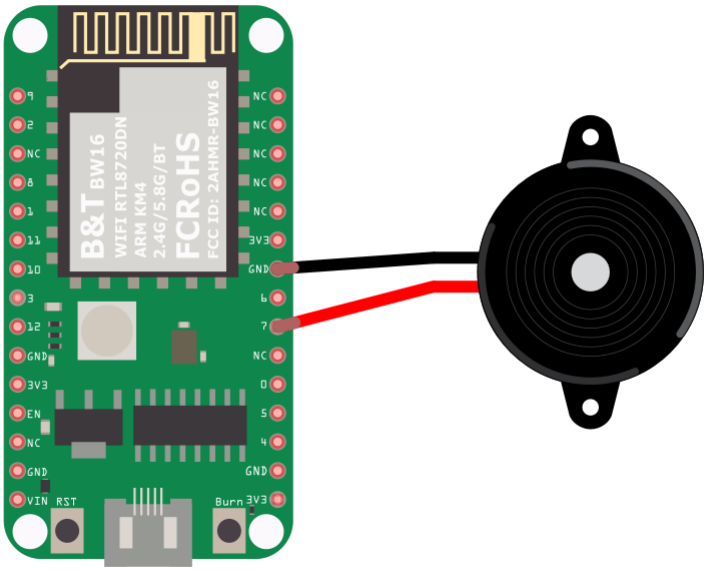
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example code in "`Examples`" -> "`AmebaAnalog`" -> "`TonePlayMelody`"
Compile and upload to Ameba, press the reset button. Then you can hear the buzzer playing music.

Code Reference

Ameba implement the tone() and noTone() API of Arduino:
https://www.arduino.cc/en/Reference/Tone
https://www.arduino.cc/en/Reference/NoTone

In the sample code, we initiate a melody array, which stores the tones to make. Another array, noteDurations, contains the length of each tone, 4 represents quarter note (equals to 3000ms/4 = 750ms, and plus an extra 30% time pause), 8 represents eighth note.

PWM - Servo Control

Preparation

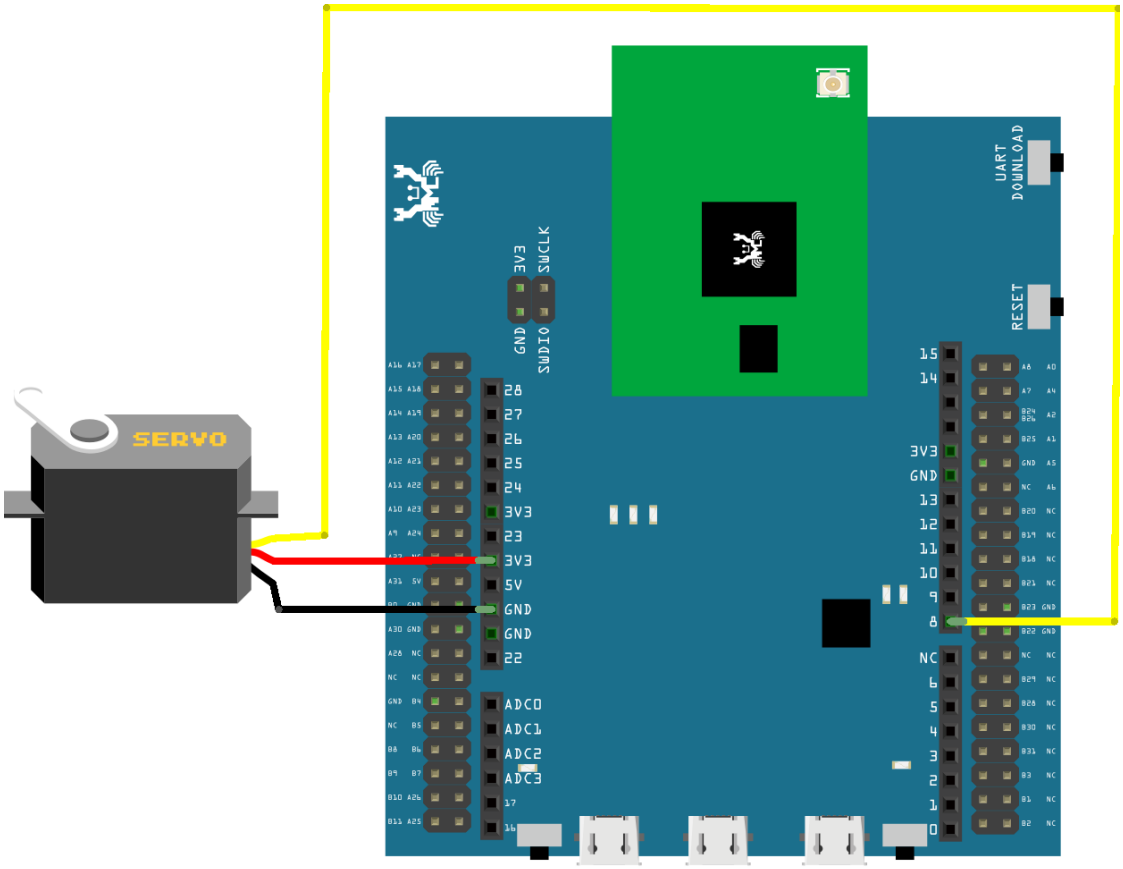
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Servo x 1 (Ex. Tower Pro SG90)

Example

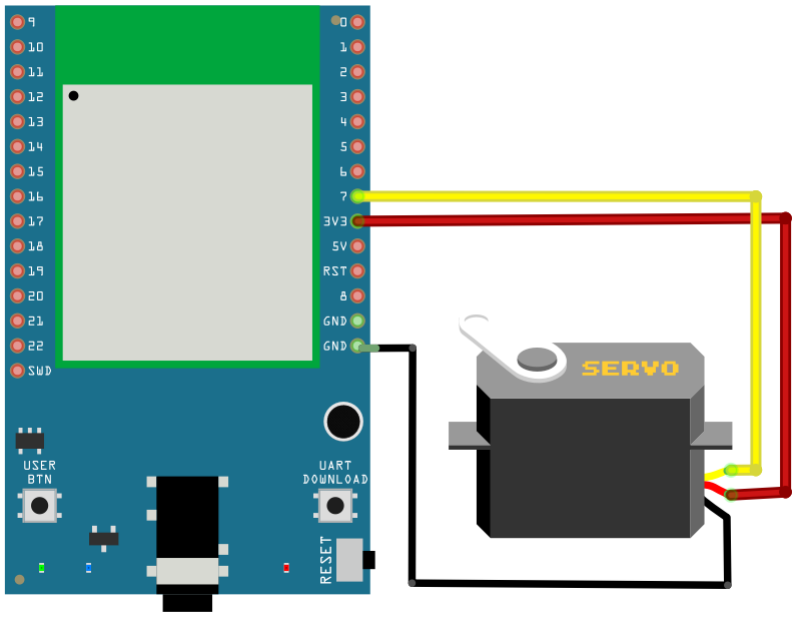
A typical servo has 3 wires, the red wire is for power, black or brown one should be connected to GND, and the other one is for signal data. We use PWM signal to control the rotation angle of the axis of the servo. The frequency of the signal is 50Hz, that is length 20ms. Each servo defines its pulse bandwidth, which is usually 1ms~2ms.

To control the rotation angle, for example if 1ms-length pulse rotates the axis to degree 0, then 1.5 ms pulse rotates the axis to 90 degrees, and 2 ms pulse rotates the axis to 180 degrees. Furthermore, a servo defines the “dead bandwidth”, which stands for the required minimum difference of the length of two consecutive pulse for the servo to work.

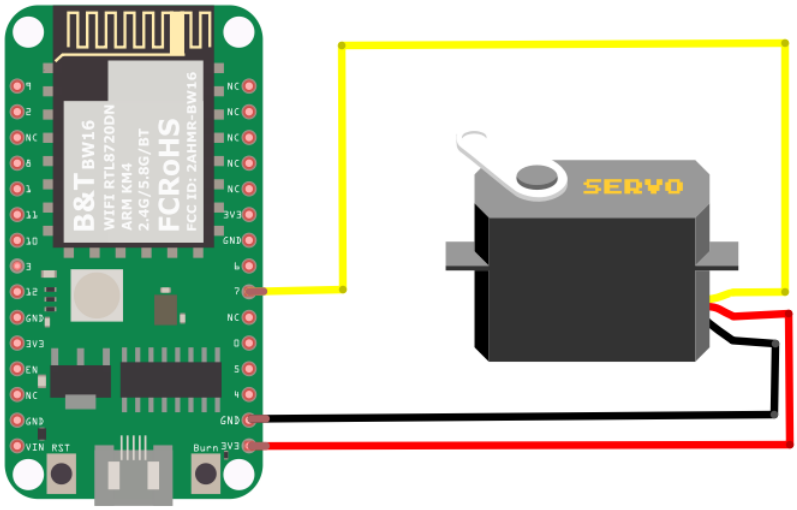
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example, "`File" -> "Examples" -> "AmebaAnalog" -> "ServoSweep`"
This example makes the servo to rotate from degree 0 to 180, and then rotate back to degree 0.

Code Reference

The Servo API of Ameba is similar to the API of Arduino. To distinguish from the original API of Arduino, we name the header file "AmebaServo.h" and the Class "AmebaServo", the usage is identical to the Arduino API.

The default pulse bandwidth of Arduino Servo is 0.5ms~2.4ms, which is the same as Tower Pro SG90. Therefore, we set the attached pin directly:

```
myservo.attach(9);
```

Next, rotate the axis to desired position:

```
myservo.write(pos);
```

RTC - Simple RTC

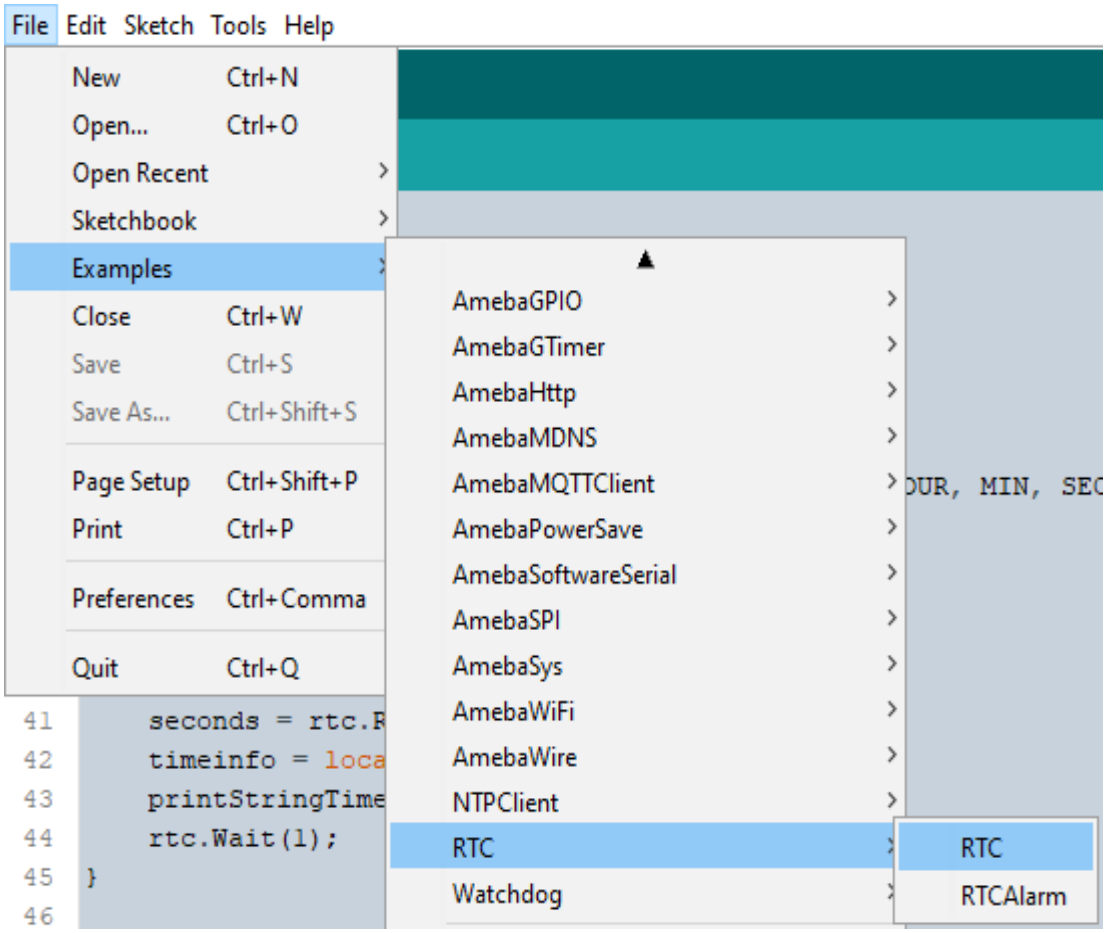
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

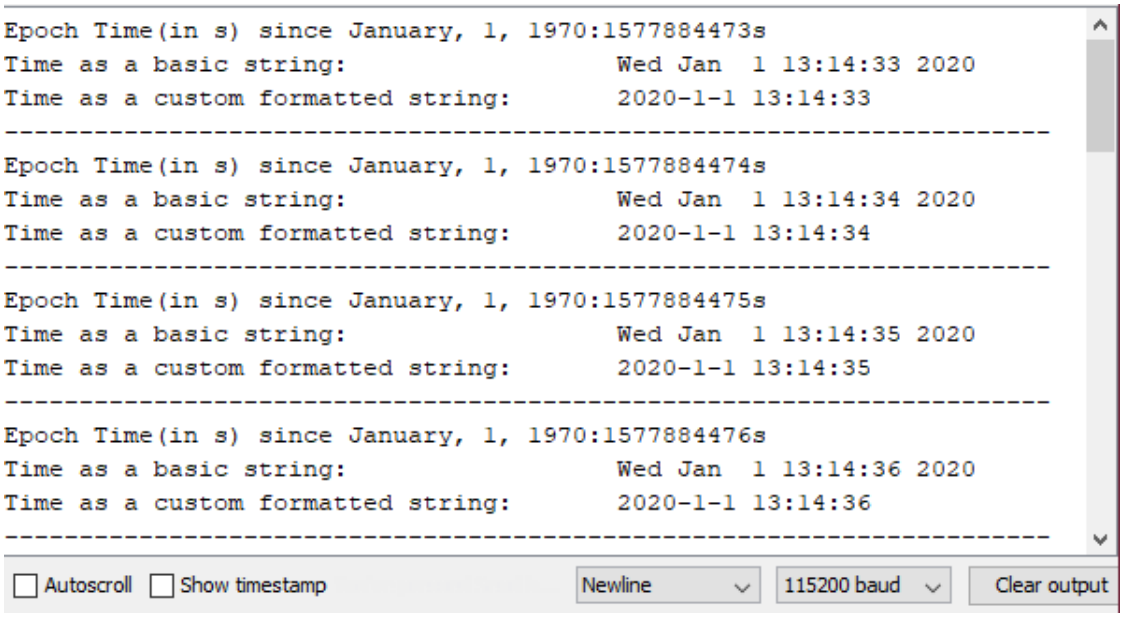
Example

This example demonstrates how to use the RTC library methods. This function describes how to use the RTC API. The RTC function is implemented by an independent BCD timer/counter.

Select the correct Ameba development board from the Arduino IDE: “Tools” -> “Board”.
Then open the “RTC” example from:
`"File" -> "Examples" -> "AmebaRTC" -> "RTC":`



Upon successfully upload the sample code and press the reset button, this example will print out time information since the user initialized time every second in the Serial Monitor.



Code Reference

[1] Simple RTC example from Arduino Tutorials:
https://www.arduino.cc/en/Tutorial/SimpleRTC

RTC - Simple RTC Alarm

Materials

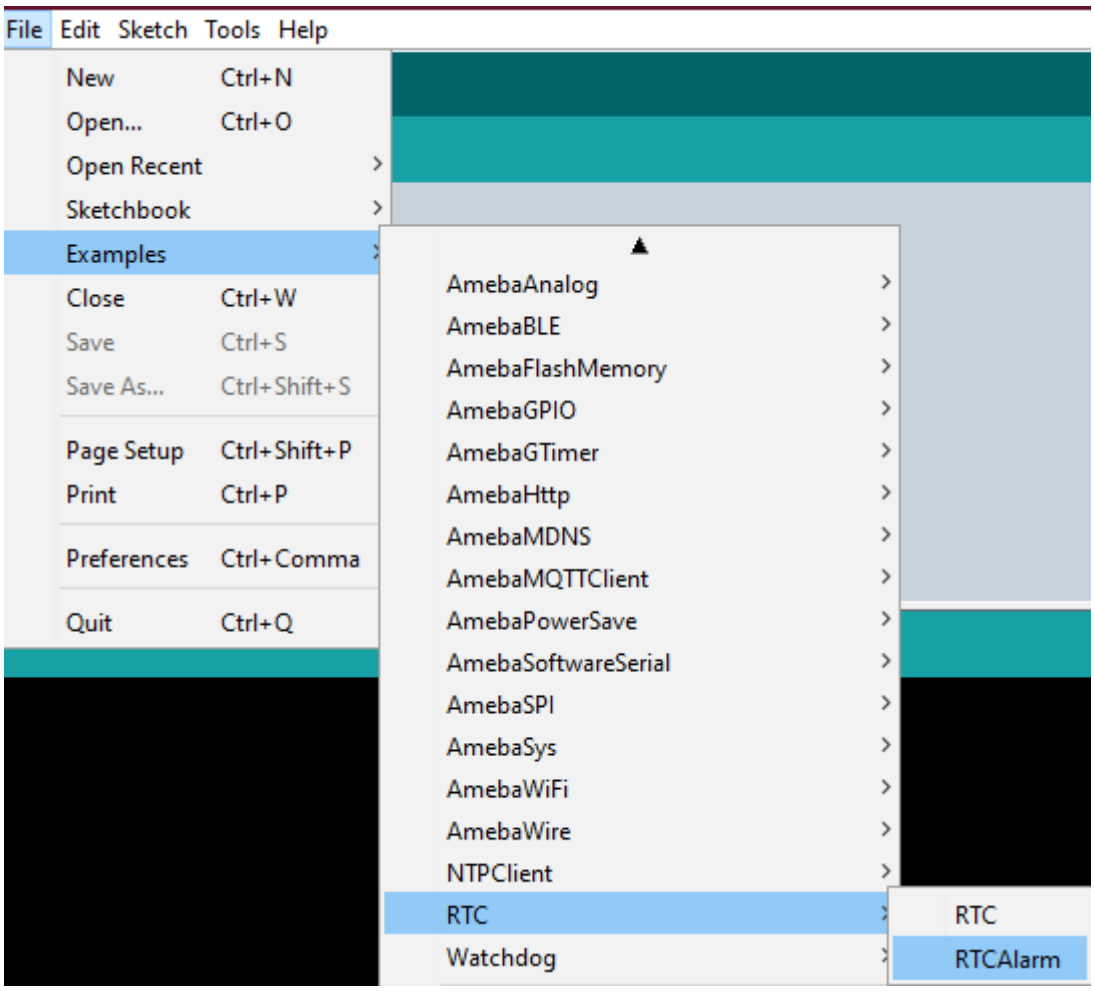
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

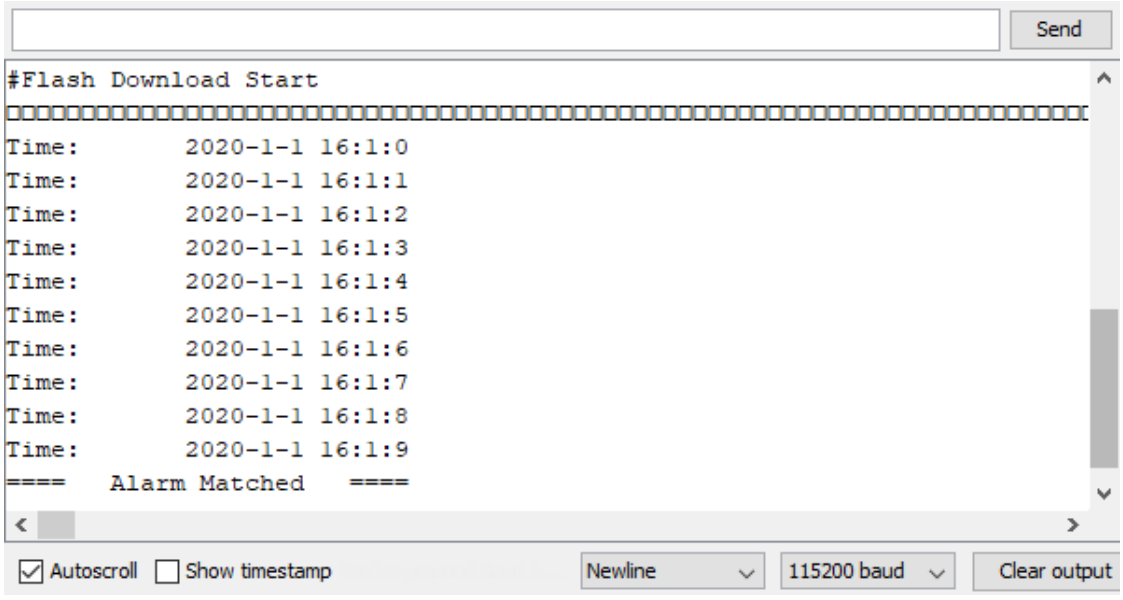
This example demonstrates how to use the RTC library methods to create a RTC Alarm, so that to do some tasks when an alarm is matched. In particular, the RTC time is set at 16:00:00 and an alarm at 16:00:10. When the time matches, "Alarm Match" information will be printed on the serial monitor.

First, select the correct Ameba development board from the Arduino IDE: "Tools" -> "Board".

Then open the "RTCAlarm" example from: `"File" -> "Examples" -> "RTC" -> "RTCAlarm"`:



In the example, the RTC time is set at 16:00:00 and an alarm is set at 16:00:10. Upon successfully upload the sample code and press the reset button. When the alarm time (10 seconds) is reached the attached interrupt function will print the following information: "Alarm Matched!" showing in this figure below.



SPI – Print Image And Text On LCD Screen

If you are not familiar with SPI, please read Introduction to SPI first.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1

Example

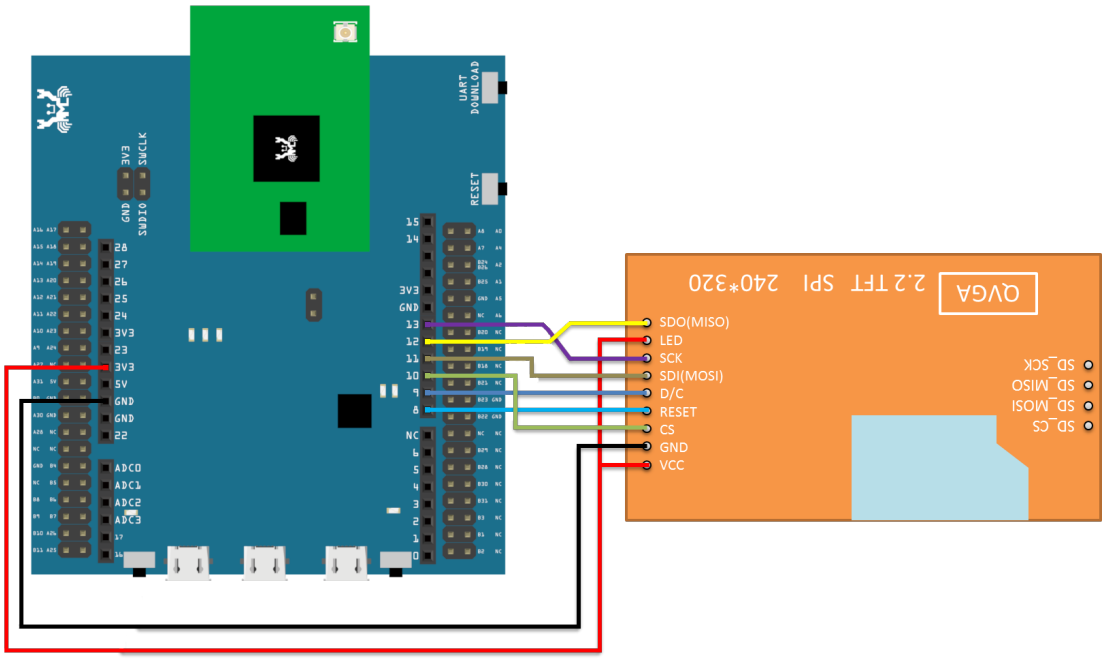
We have tested the following two models of ILI9341 TFT LCD with SPI interface:

- Adafruit 2.8″ TFT LCD (with touch screen)
  - https://www.adafruit.com/products/1651
  - https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2?view=all
- QVGA 2.2″ TFT LCD
  - http://www.lcdwiki.com/2.2inch_SPI_Module_ILI9341_SKU:MSP2202

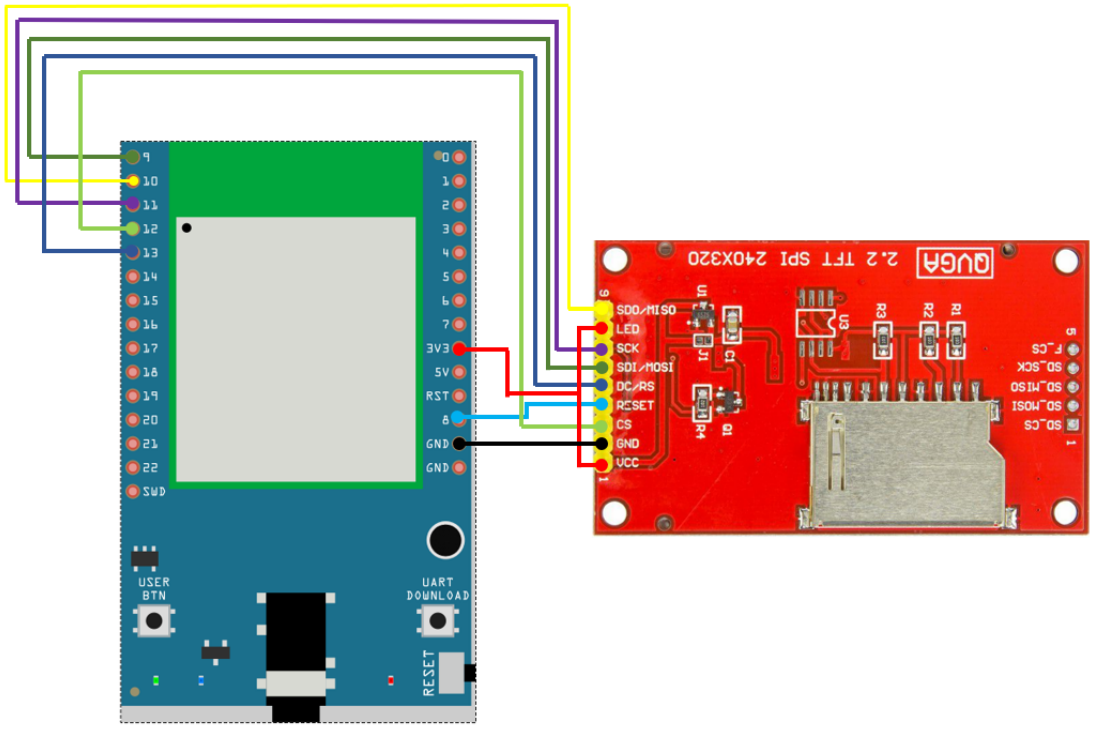
Common pins in ILI9341 TFT LCD with SPI interface:

- MOSI: Standard SPI Pin
- MISO: Standard SPI Pin
- SLK: Standard SPI Pin
- CS: Standard SPI Pin
- RESET: Used to reboot LCD.
- D/C: Data/Command. When it is at Low, the signal transmitted are commands, otherwise the data transmitted are data.
- LED (or BL): Adapt the screen backlight. Can be controlled by PWM or connected to VCC for 100% backlight.
- VCC: Connected to 3V or 5V, depends on its spec.
- GND: Connected to GND.

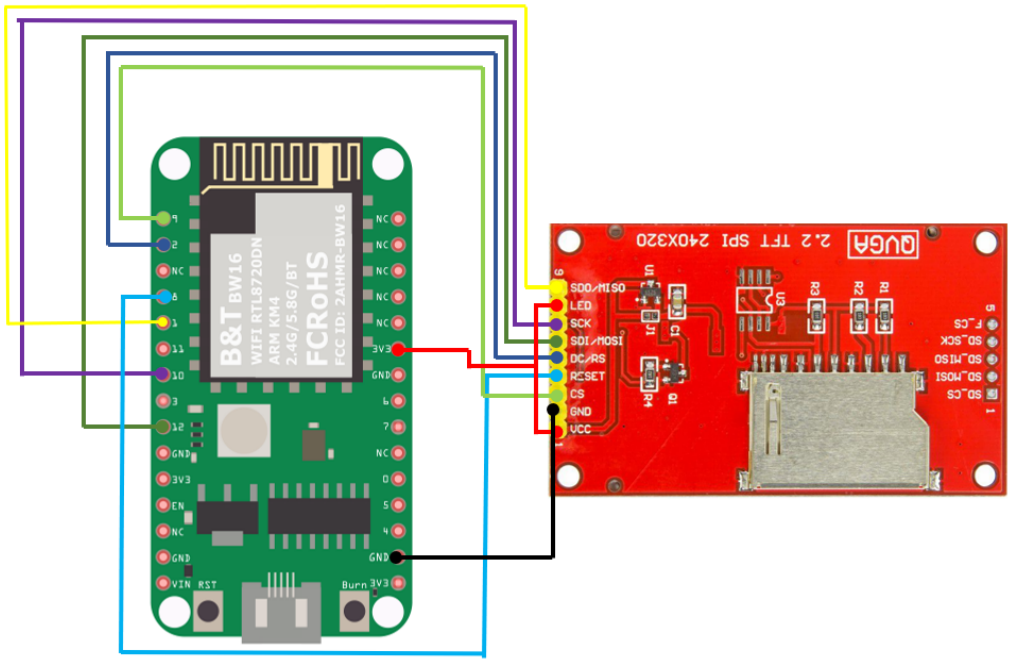
AMB21/ AMB22 and QVGA TFT LCD Wiring Diagram:



AMB23 and QVGA TFT LCD Wiring Diagram:



BW16 and QVGA TFT LCD Wiring Diagram:

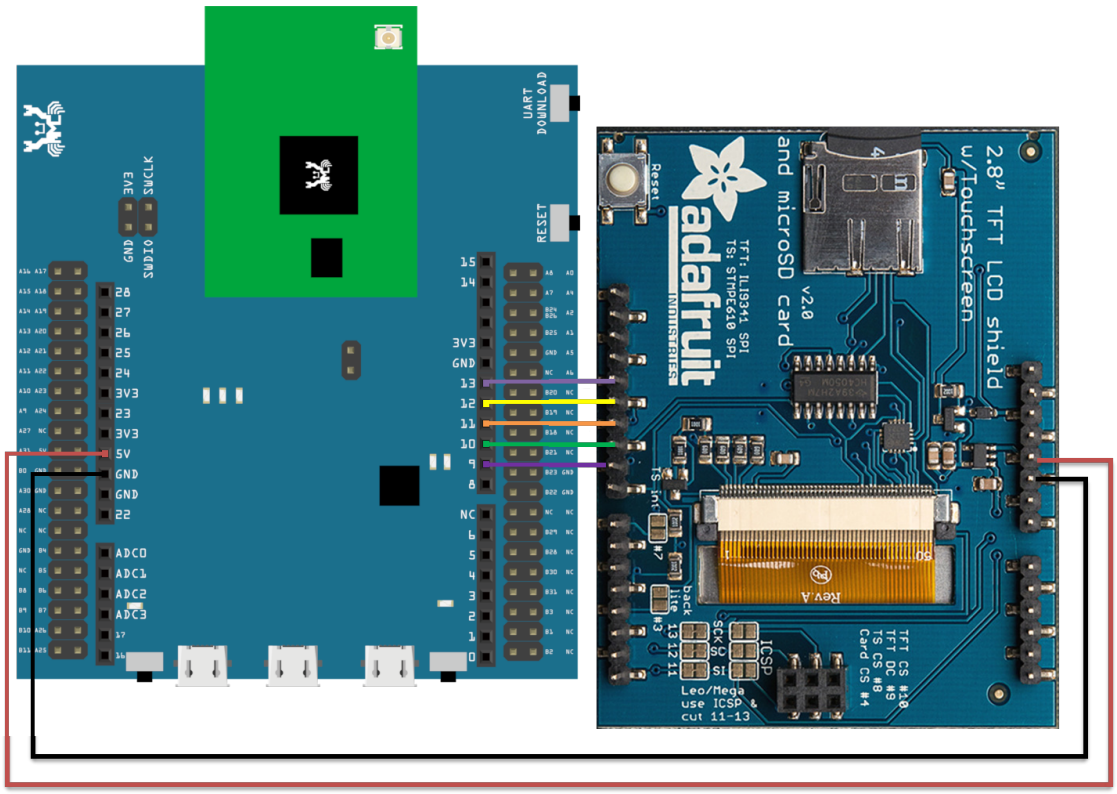


Wiring example of Adafruit 2.8” TFT LCD touch shield:

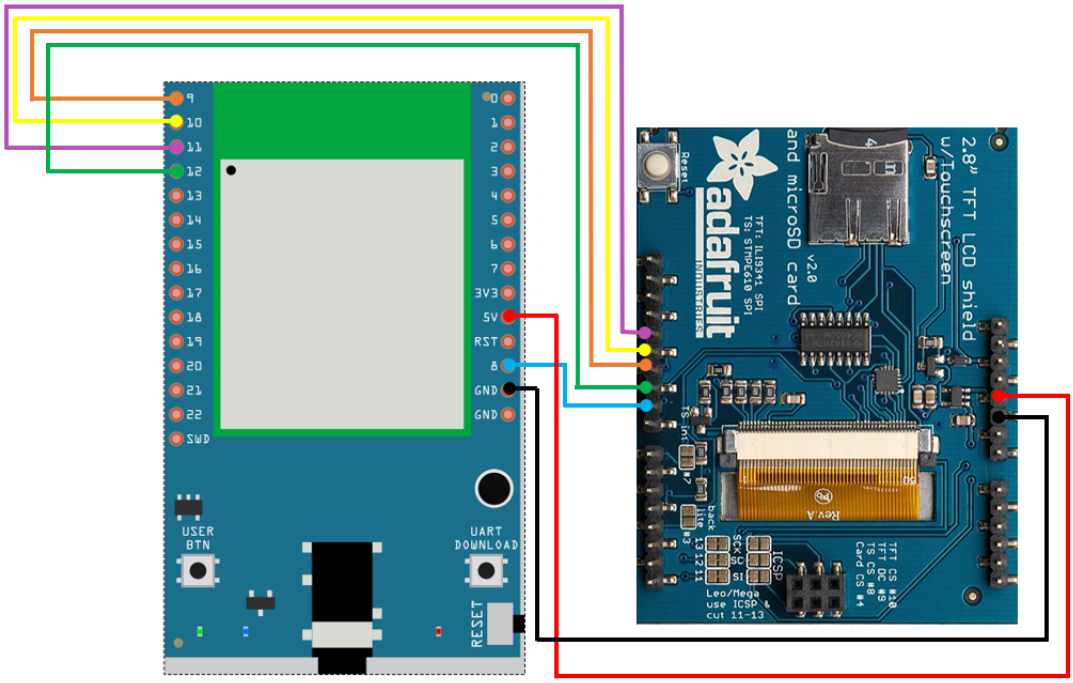
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.

AMB21 / AMB22 and Adafruit 2.8” TFT LCD touch shield Wiring Diagram:

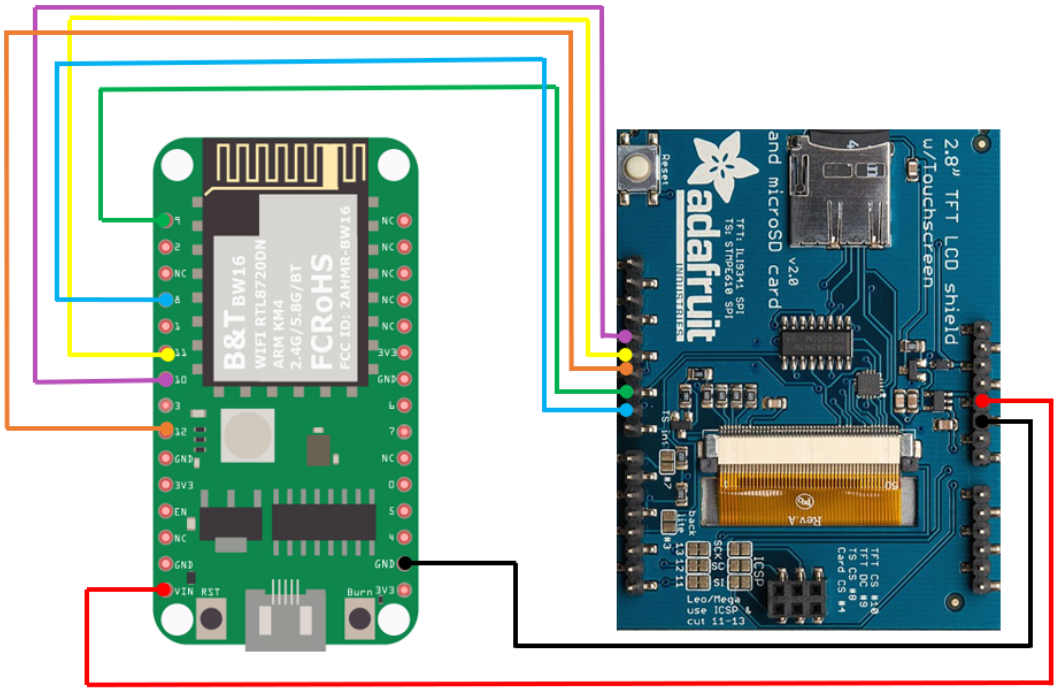
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.



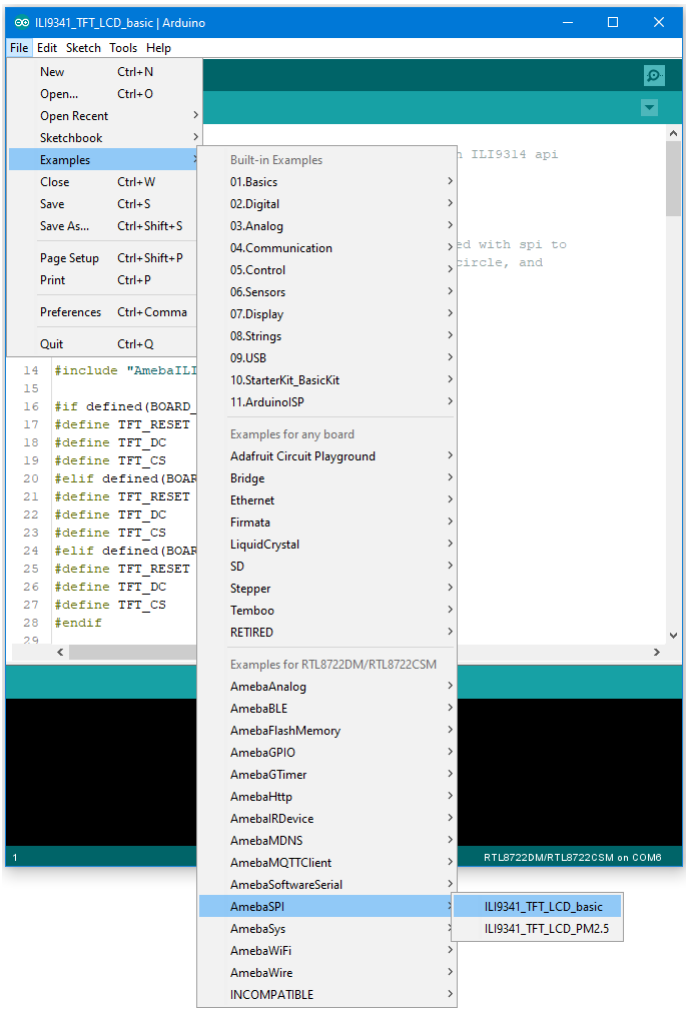
AMB23 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:



BW16 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:

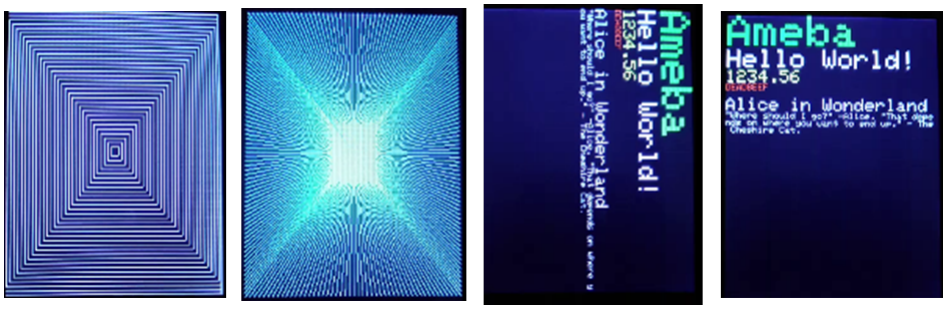


Open the example, “Files” -> “Examples” -> “AmebaSPI” -> “ILI9341_TFT_LCD_basic”



Compile and upload to Ameba, then press the reset button.

Then you can see some display tests appear on the LCD screen, such as displaying different colors, drawing vertical and horizontal lines, drawing circles, etc.…



Code Reference

- **RGB 16-bit**

  ILI9341 uses RGB 16-bit to display colors. Different from RGB 24-bit, it uses 5 bits for red, 6 bits for green, 5

bits for blue. For example, the RGB 24-bit representation of sky blue is 0x87CEFF, that is in binary:

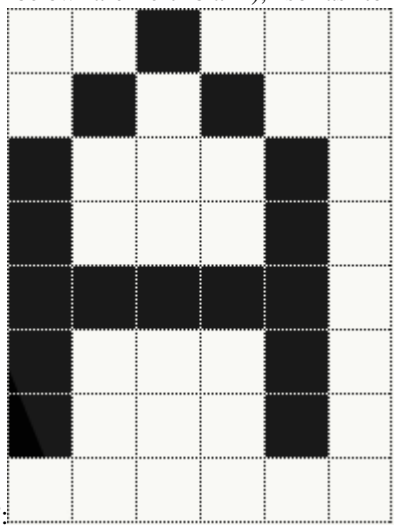
  - Red: 0x87 = B10000111
  - Green: 0xCE = B11001110
  - Blue: 0xFF = B11111111

and converted to RGB 16-bit:

  - Red: B10000
  - Green: B110011
  - Blue: B11111

Then concatenate them, which forms B1000011001111111 = 0x867F

- **Drawing of ILI9341**
  - First you must specify the range of the rectangle to draw, then pass the 2-byte RGB 16-bit color to ILI9341 corresponding to each pixel one by one, in this way ILI9341 fills each color to each pixel.
  - You still must specify the drawing range even though the range covers only one pixel.
  - From the rules we mentioned above, we can conclude that drawing vertical or horizontal lines are faster than diagonal lines.
- **Printing text on ILI9341**
  - In our API, each character is 5×7 but each character is printed to size 6×8 (its right side and below are left blank), so as to separate from next character. For example, the character "A":



  - The font size represents the dot size. For example, if the font size is 2, each dot in the character is a 2×2 rectangle
- **Screen rotation**
  - ILI9341 provides 0, 90, 180, 270 degrees screen rotation.
  - If the original width is 240 and original height is 320, when the screen rotates 90 degrees, the width becomes 320 and the height becomes 240.

SPI – Show PM2.5 Concentration On ILI9341 TFT LCD

If you are not familiar with SPI, please read Introduction to SPI first.

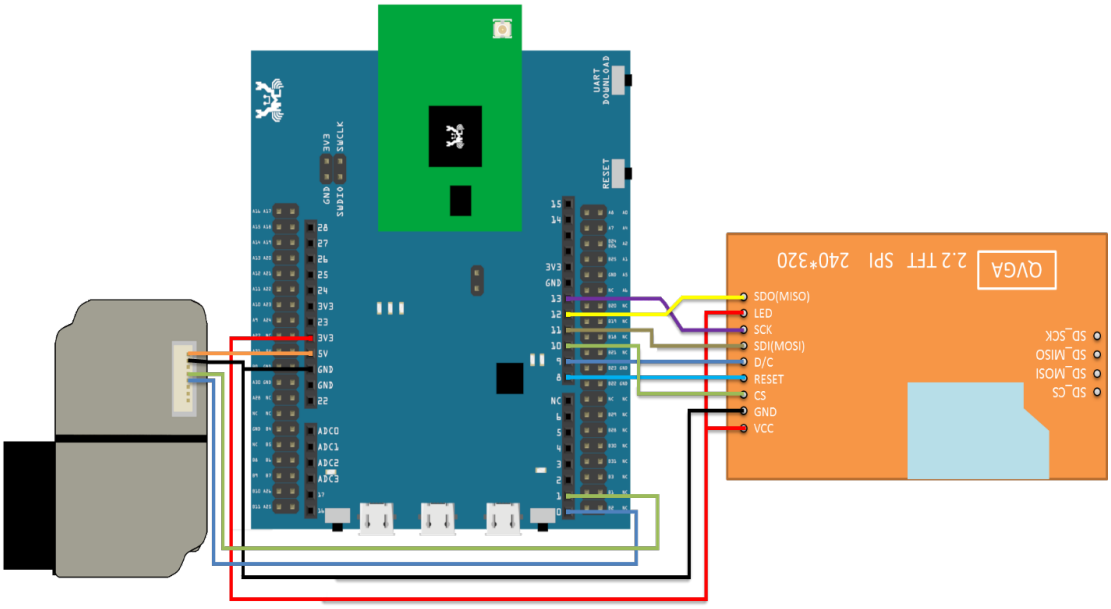
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1
- Plantower PMS3003 or PMS5003 x 1

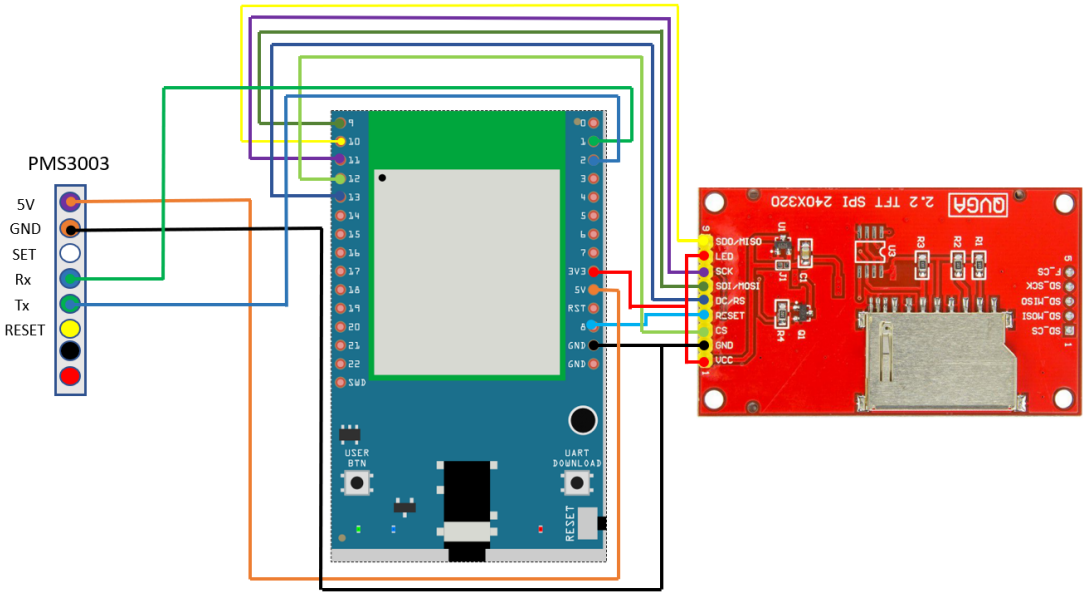
Example

This example extends previous PM2.5 example to show the PM2.5 concentration on the LCD.

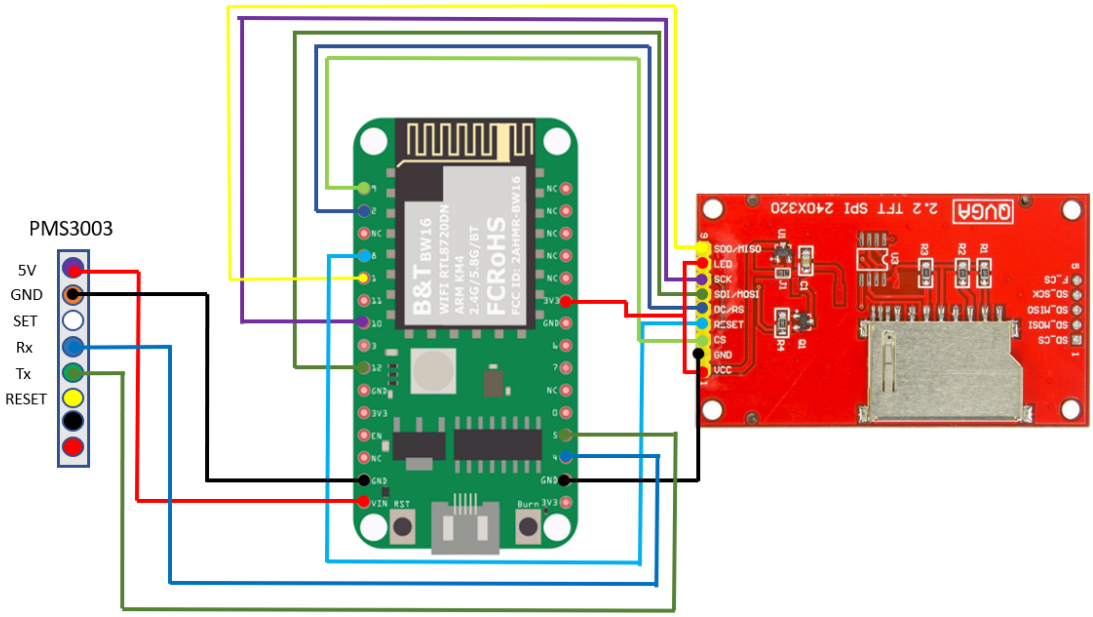
AMB21 / AMB22 and QVGA TFT LCD Wiring Diagram:
(Note: PMS3003/PMS5003 sensor requires 5V voltage)



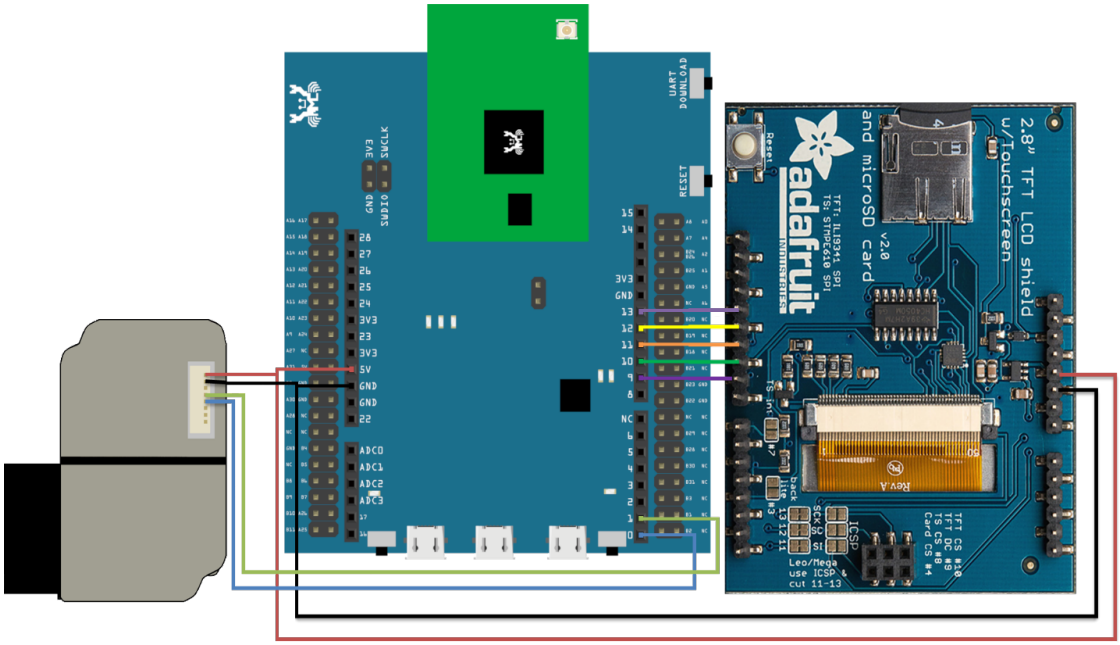
AMB23 and QVGA TFT LCD Wiring Diagram:



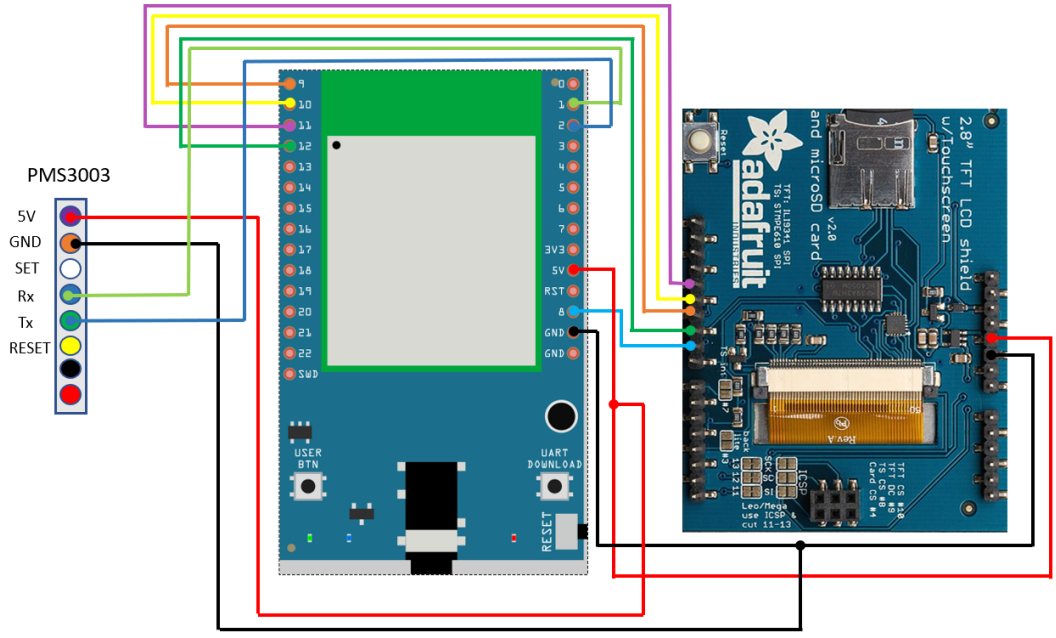
BW16 and QVGA TFT LCD Wiring Diagram:



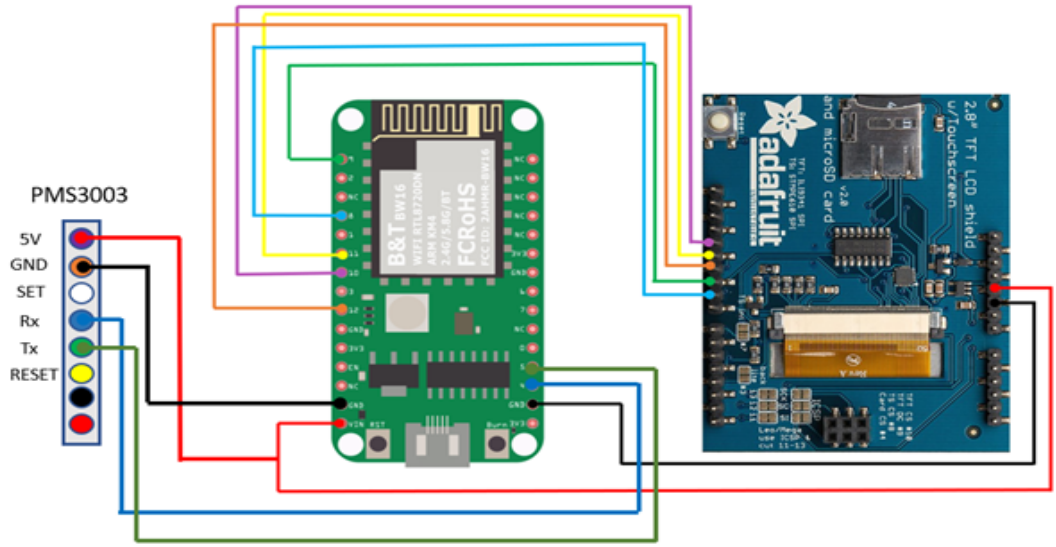
AMB21 / AMB22 and Adafruit 2.8" TFT LCD Wiring Diagram:



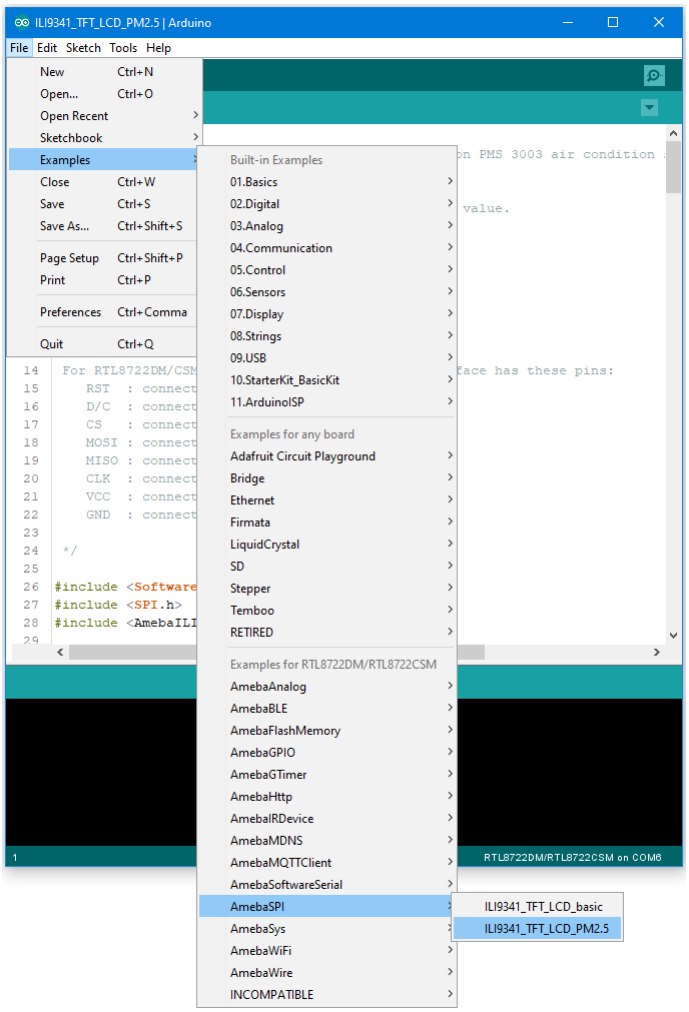
AMB23 and and Adafruit 2.8" TFT LCD Wiring Diagram:



BW16 and and Adafruit 2.8" TFT LCD Wiring Diagram:

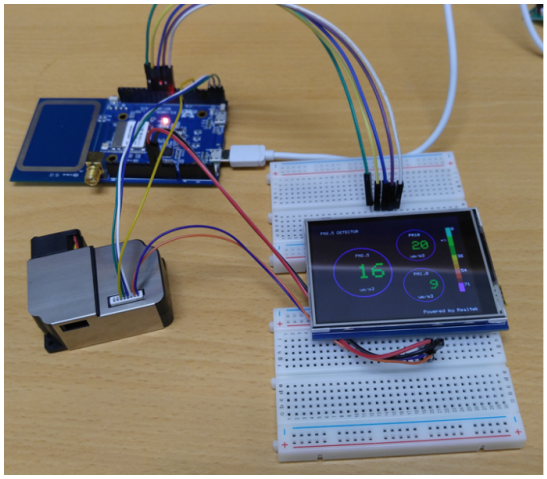
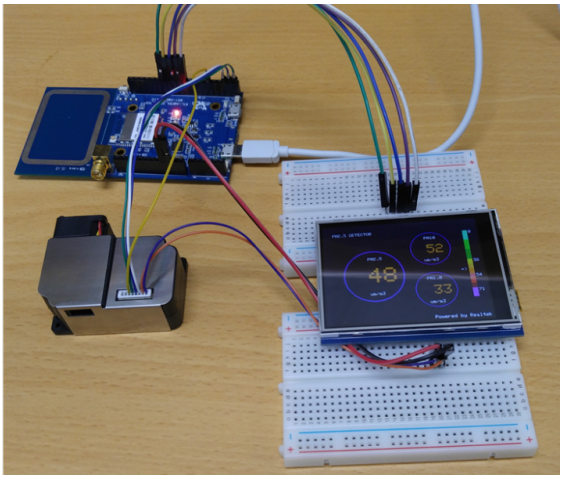


Open the example, "`Files` -> "`Examples`" -> "`AmebaSPI`" -> "`PM25_on_ILI9341_TFT_LCD`"



Compile and upload to Ameba, then press the reset button.

Then you can see the concentration value of PM1.0, PM2.5 and PM10 on the LCD.

Code Reference

In this example, first rotate the screen by 90 degrees, and draw the static components such as the circles, the measuring scale, and the title text. After the concentration value is detected, it is printed inside the circle.

TensorFlow Lite - Hello World

Materials

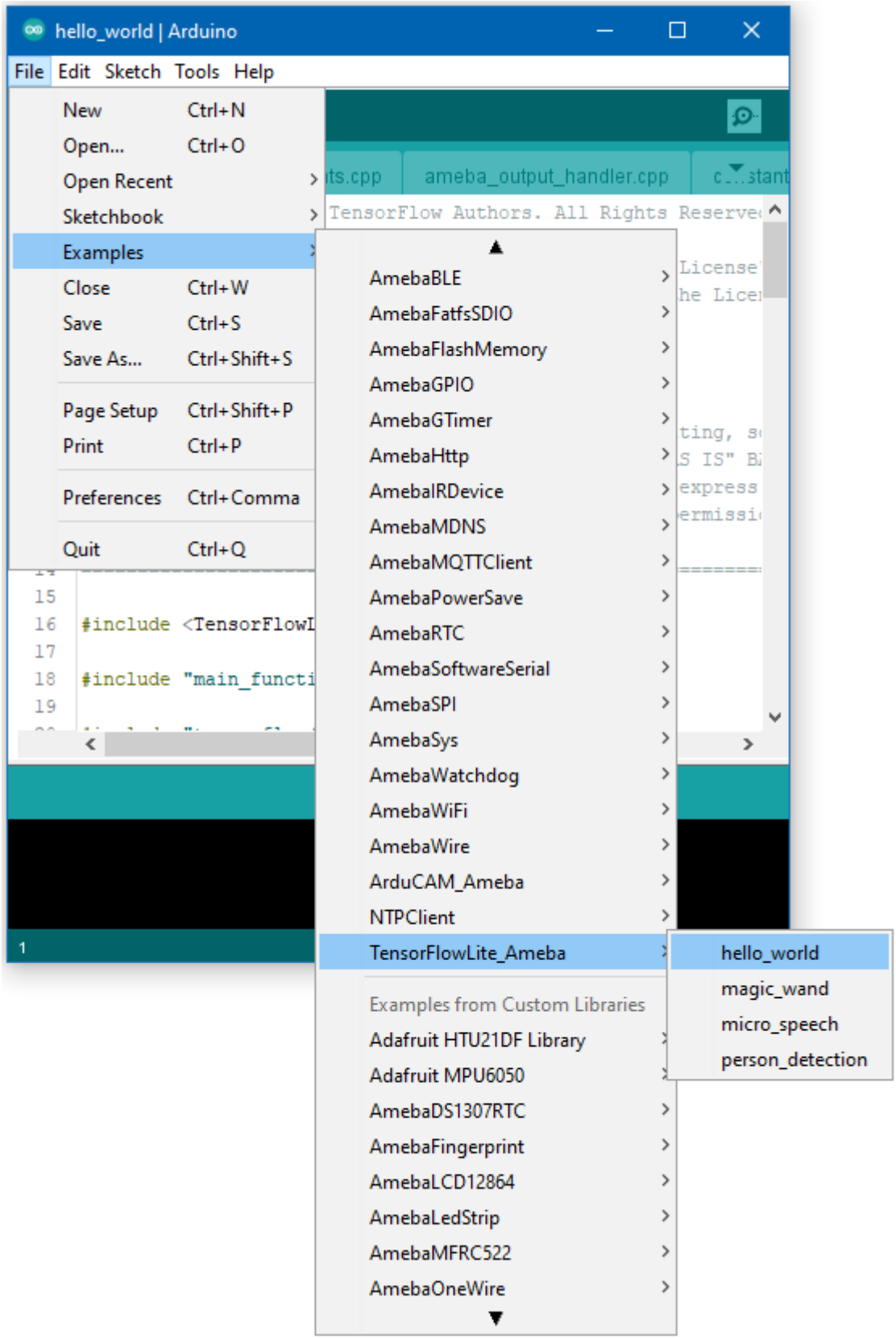
- AmebaD [AMB21 / AMB22 / AMB23] x 1
- LED x 1

Example

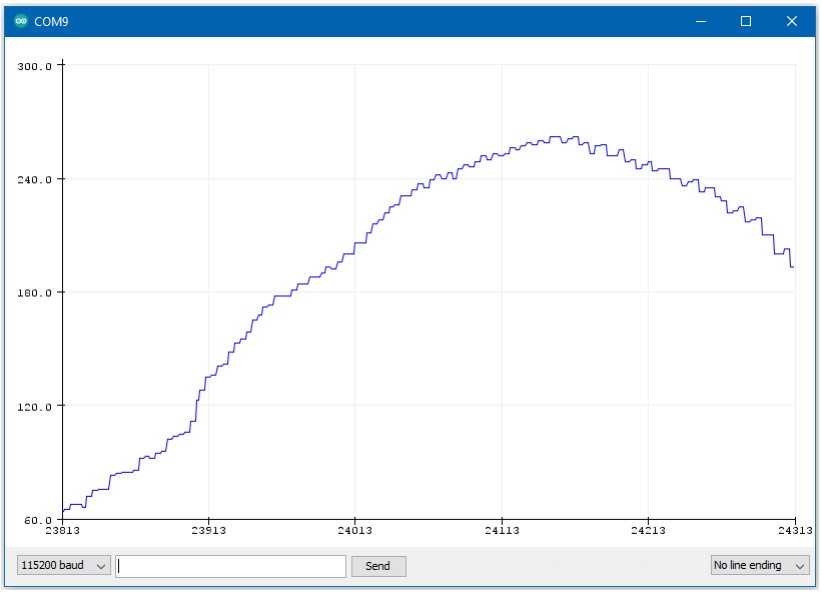
Procedure

Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries. Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it. Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

Open the example, `"Files" -> "Examples" -> “TensorFlowLite_Ameba” -> “hello_world”`.


hello_world | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
ameba_output_handler.cpp
TensorFlow Authors. All Rights Reserve
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
hello_world
magic_wand
micro_speech
person_detection
#include <TensorFlowL
#include "main_functi

Upload the code and press the reset button on Ameba once the upload is finished.
Connect the LED to digital pin 10 and ground, ensuring that the polarity is correct. You should see the LED fade in and out rapidly.
In the Arduino serial plotter, you can see the output value of the Tensorflow model plotted as a graph, it should resemble a sine wave.



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Magic Wand

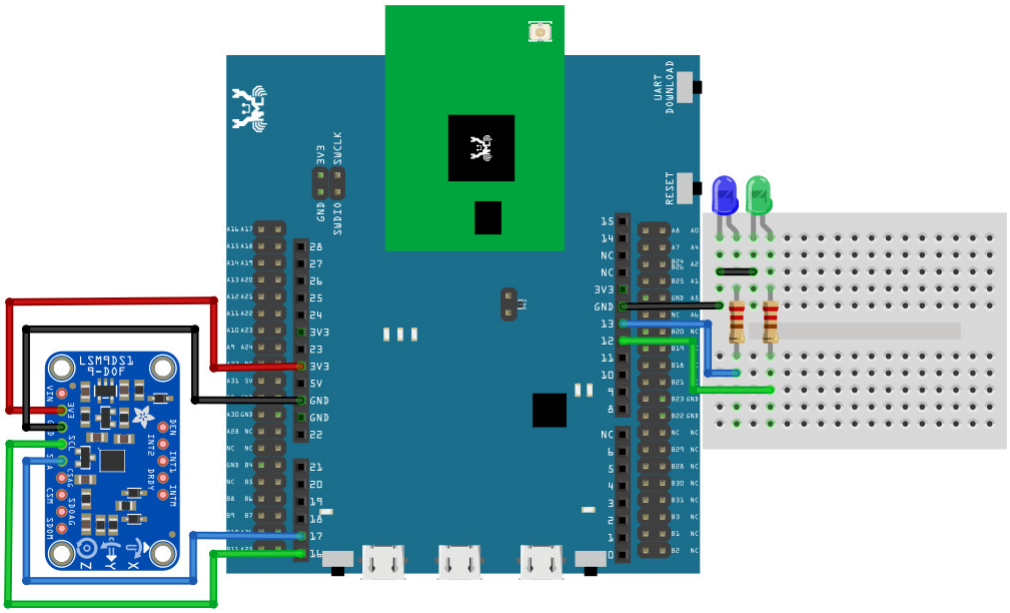
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Adafruit LSM9DS1 accelerometer
- LED x 2

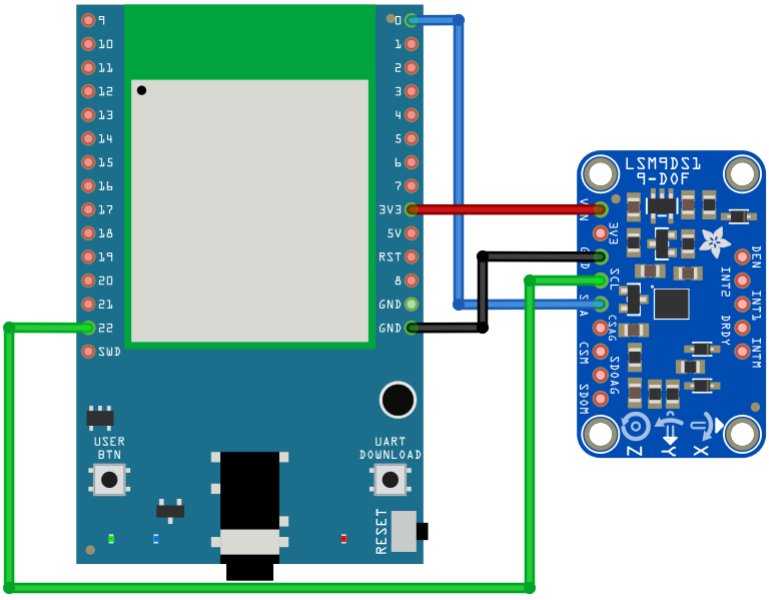
Example

Procedure

AMB21* / AMB22 Wiring Diagram:
Connect the accelerometer and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
For RTL8722DM MINI, we will use the onboard LEDs on the board itself.



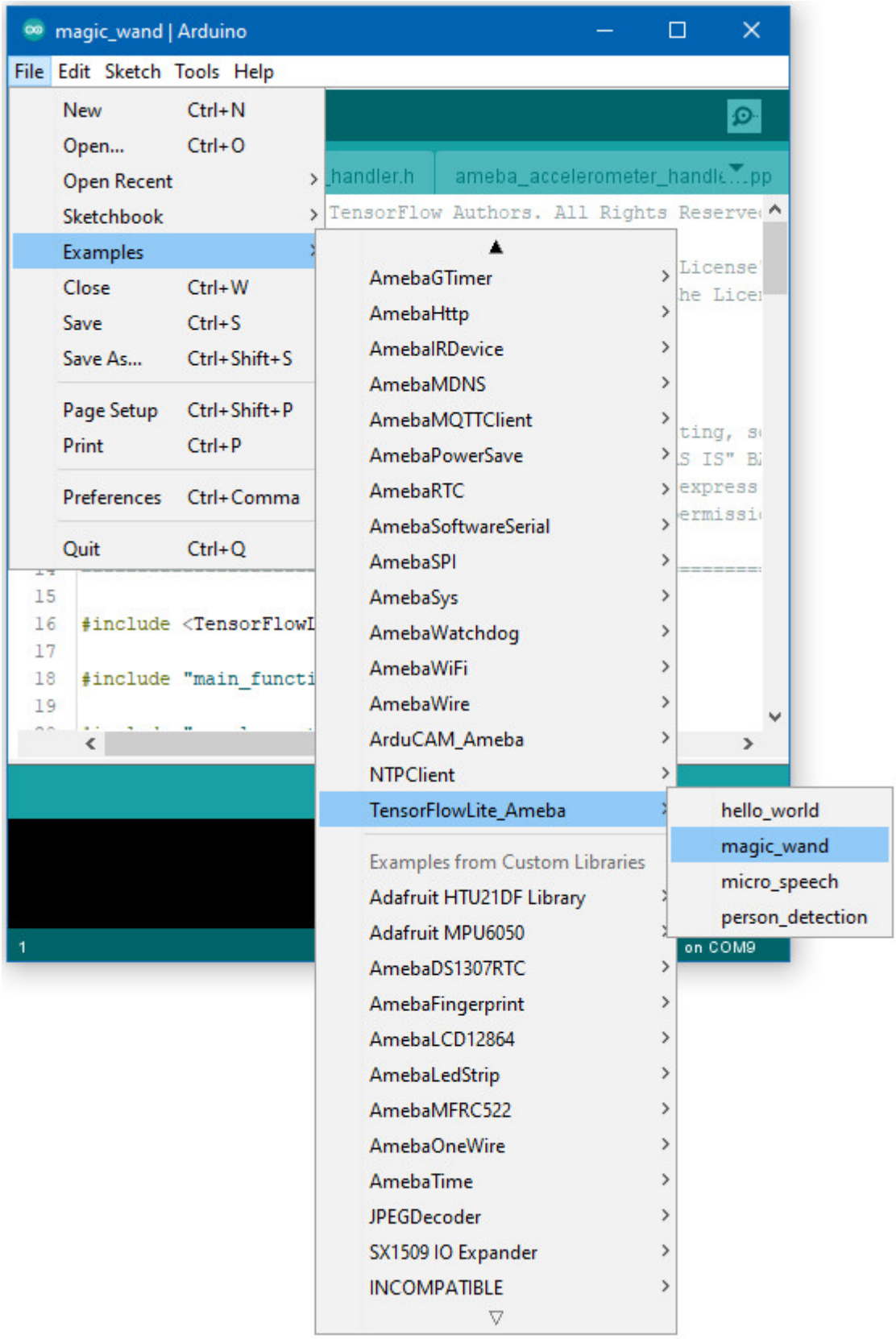
Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it.

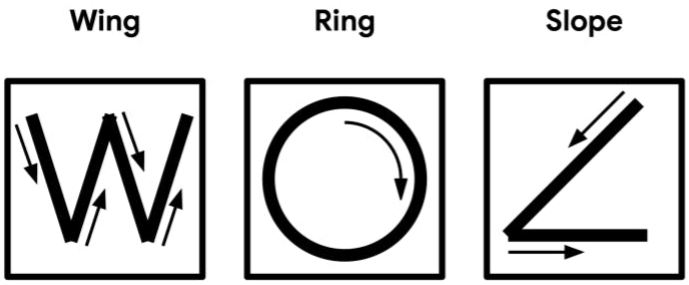
Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

In the Arduino IDE library manager, install the Arduino_LSM9DS1 library. This example has been tested with version 1.1.0 of the LSM9DS1 library.

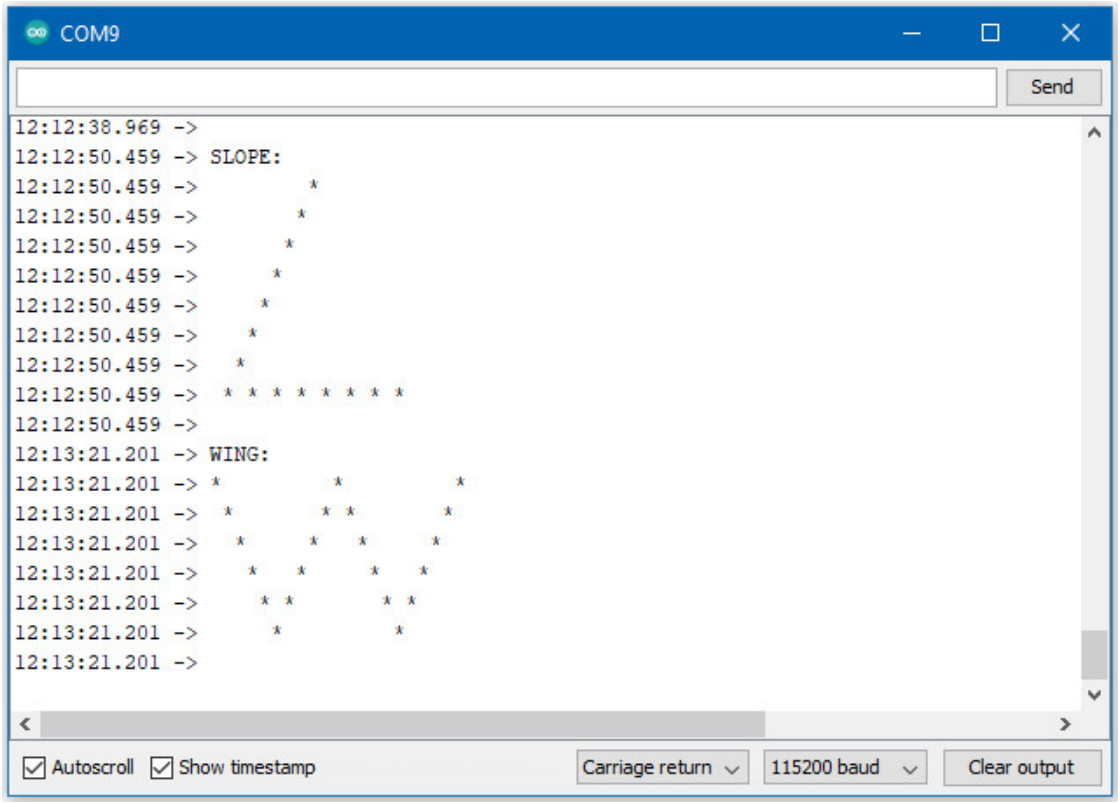
Open the example, `"Files" -> "Examples" -> “TensorFlowLite_Ameba” -> “magic_wand”`.


magic_wand | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
AmebaTime
JPEGDecoder
SX1509 IO Expander
INCOMPATIBLE
hello_world
magic_wand
micro_speech
person_detection
#include <TensorFlowL
#include "main_functi
on COM9

Upload the code and press the reset button on Ameba once the upload is finished.
Holding the accelerometer steady, with the positive x-axis pointing to the right and the positive z-axis pointing upwards, move it following the shapes as shown, moving it in a smooth motion over 1 to 2 seconds, avoiding any sharp movements.



If the movement is recognised by the Tensorflow Lite model, you should see the same shape output to the Arduino serial monitor. Different LEDs will light up corresponding to different recognized gestures.
Note that the wing shape is easy to achieve, while the slope and ring shapes tend to be harder to get right.



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Micro Speech

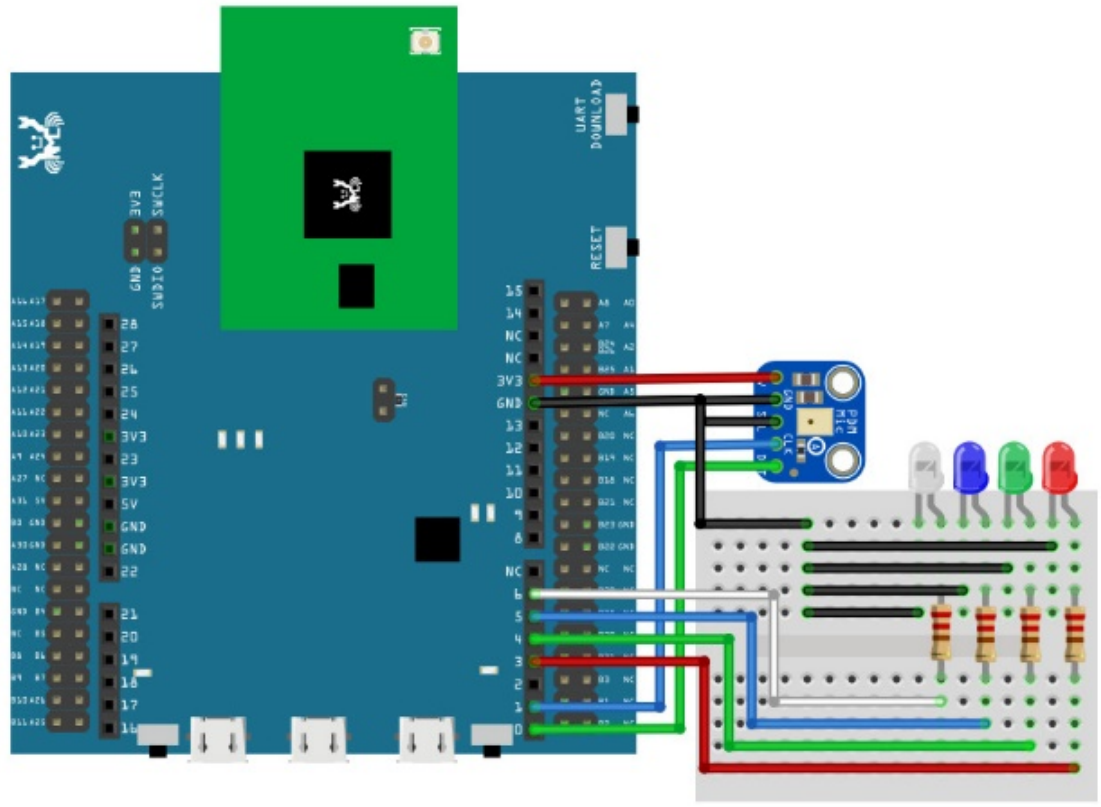
Preparation

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Adafruit PDM MEMS microphone
- LED x 4

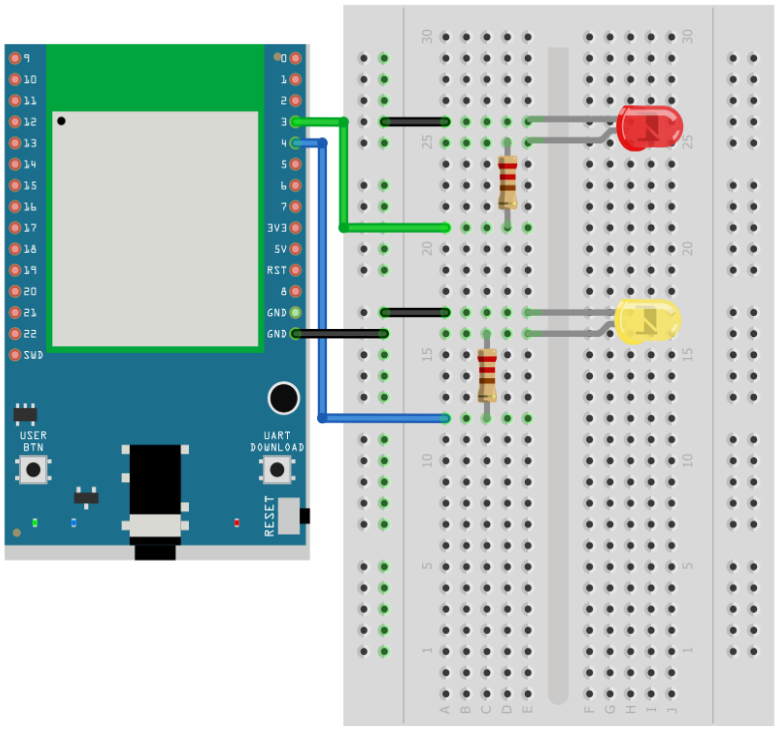
Example

Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the microphone and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
As RTL8722DM MINI have a built in microphone on the board, there is no need for any external microphone. For the LEDs, we will only connect two LEDs and then use the two onboard LEDs (Blue and Green).

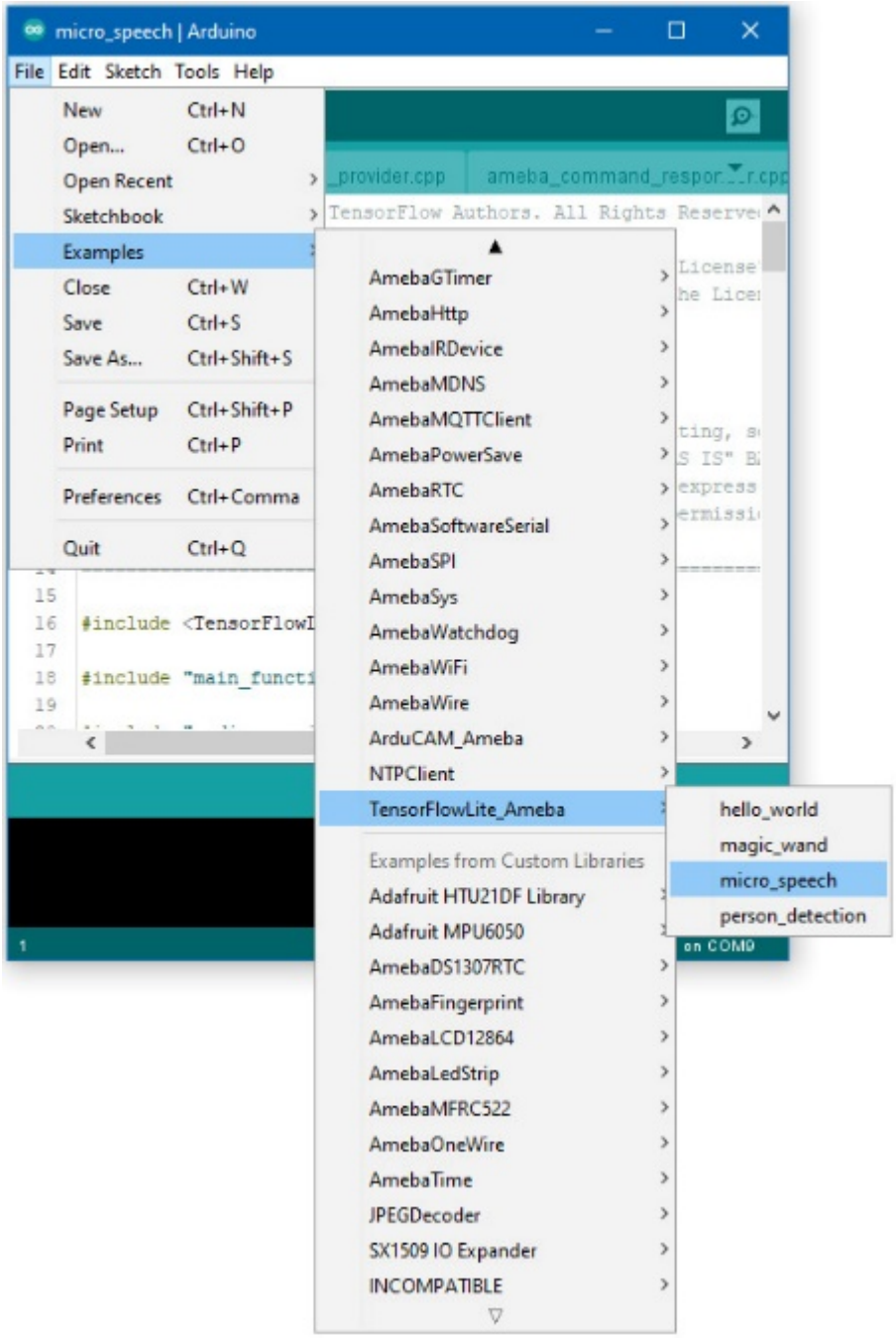


Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it.

Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

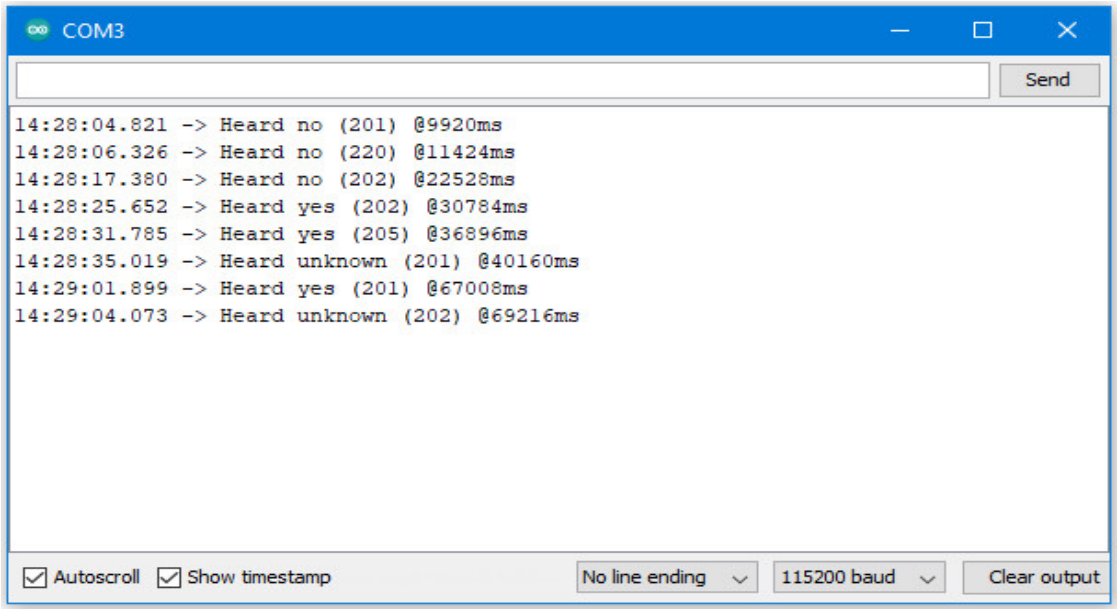
Open the example, `"Files" -> "Examples" -> “TensorFlowLite_Ameba” -> “micro_speech”`.



Upload the code and press the reset button on Ameba once the upload is finished.

Once it is running, you should see one of the LEDs flashing, indicating that it is processing audio. Saying the word “yes” will cause the green LED to light up. Saying the word “no” will cause the red LED to light up. If the word is not recognized, the blue LED will to light up.

The inference results are also output to the Arduino serial monitor, which appear as follows:



If you are having trouble in getting the words recognized, here are some tips:

- Ensure that your surroundings are quiet with minimal noise.
- Experiment with varying the distance of the microphone, starting with it at an arm's length.
- Experiment with different tones and volume when saying the words.
- Depending on how you pronounce the words, the characteristics of the microphone used, getting one keyword recognized may be easier than the other.

Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Person Detection

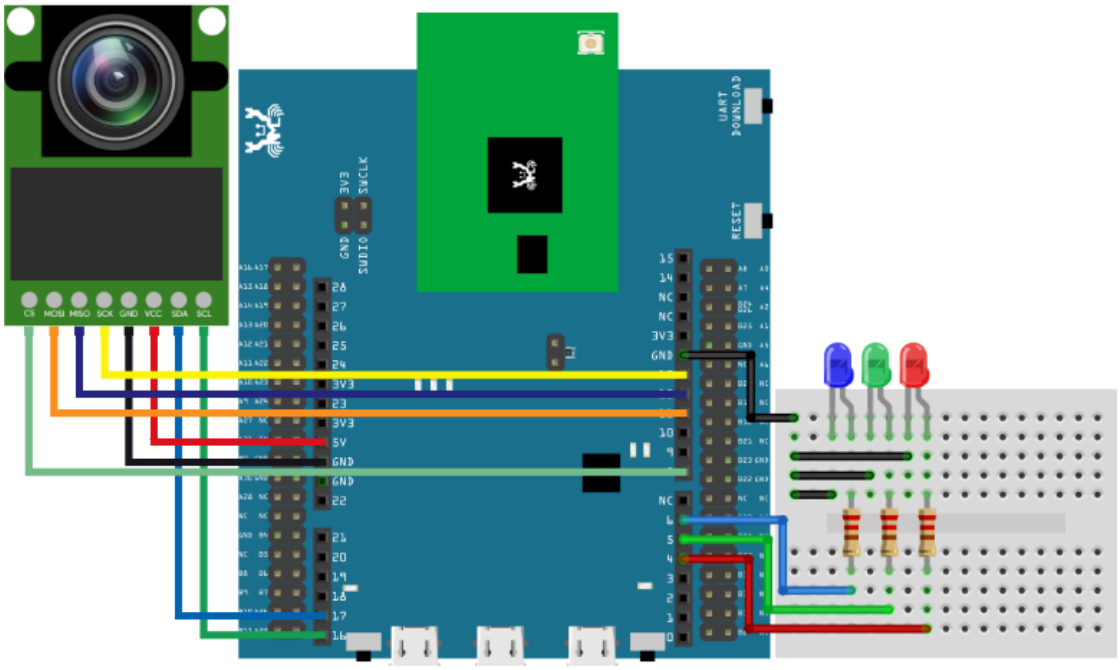
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Arducam Mini 2MP Plus OV2640 SPI Camera Module x 1
- LED x 3

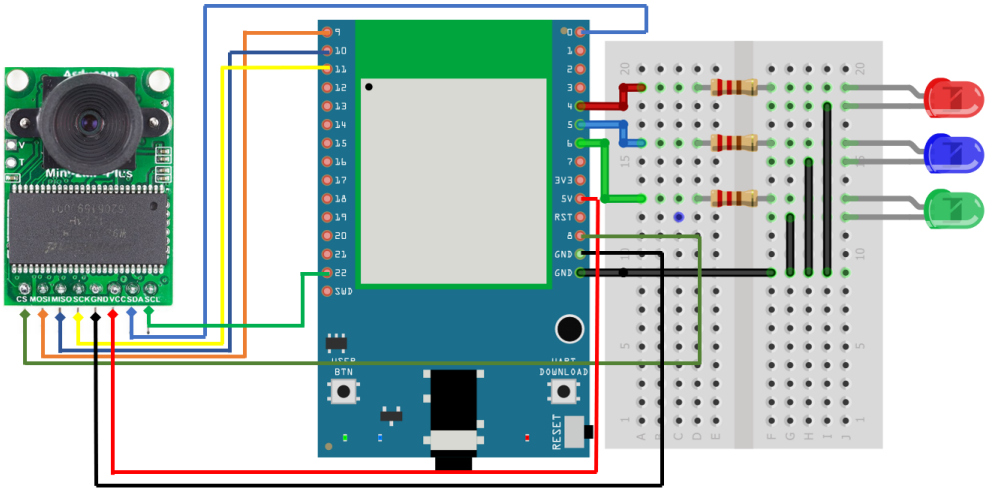
Example

Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the camera and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:



Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it. Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

You will also need to install the Ameba_ArduCAM library, found together with the TensorFlow Lite library.

In the Arduino IDE library manager, install the JPEGDecoder library. This example has been tested with version 1.8.0 of the JPEGDecoder library.

Once the library has installed, you will need to configure it to disable some optional components that are not compatible

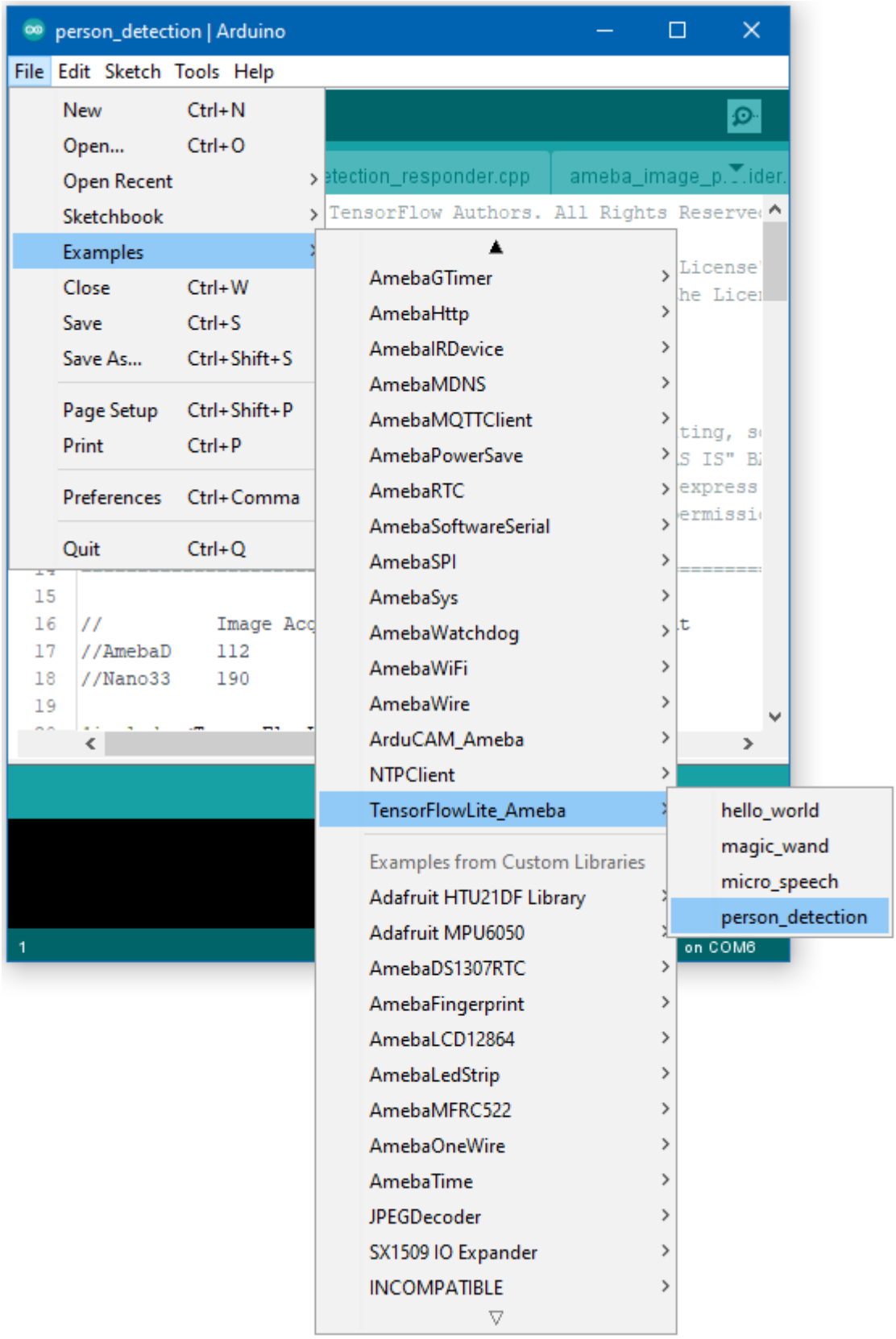
with the RTL8722DM. Open the following file:

```
Arduino/libraries/JPEGDecoder/src/User_Config.h
```

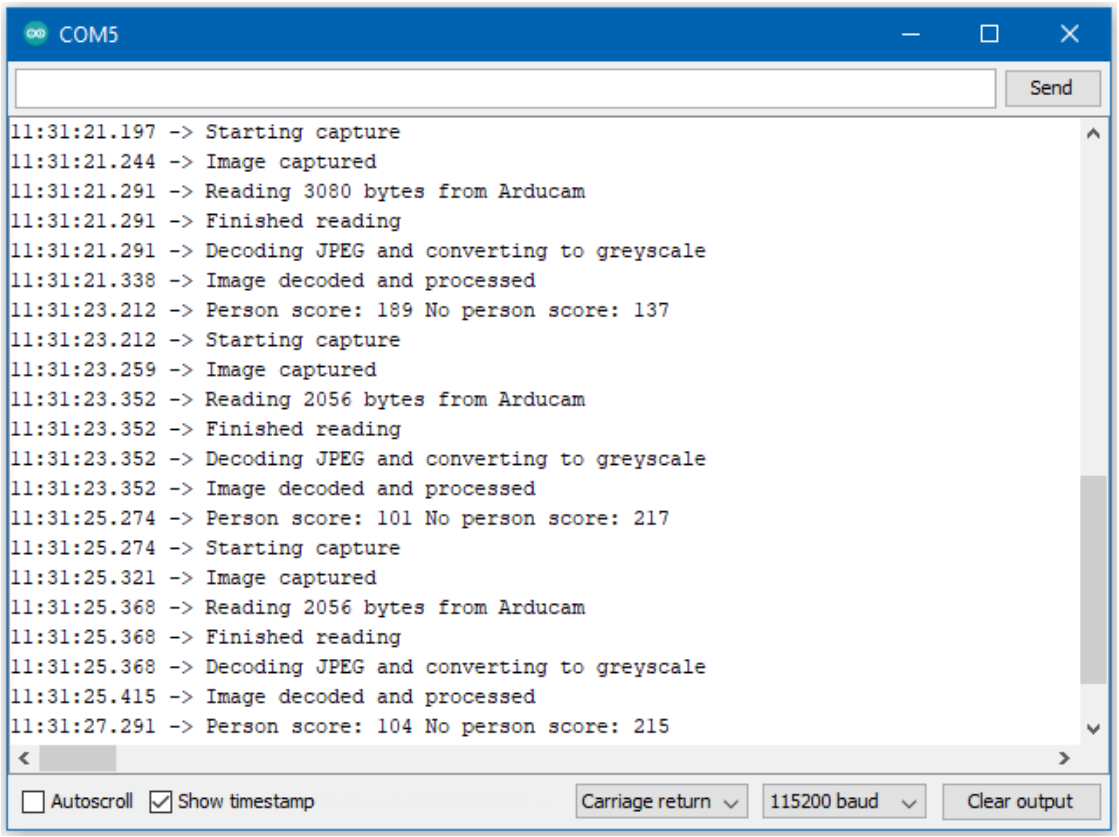
Make sure that both `#define LOAD_SD_LIBRARY` and `#define LOAD_SDFAT_LIBRARY` are commented out, as shown in this excerpt from the file:

```
//#define LOAD_SD_LIBRARY // Default SD Card library
//#define LOAD_SDFAT_LIBRARY // Use SdFat library instead, so SD Card SPI can be bit␣
↪bashed
```

Open the example, `"Files" -> "Examples" -> "TensorFlowLite_Ameba" -> "person_detection"`.


person_detection | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
AmebaTime
JPEGDecoder
SX1509 IO Expander
INCOMPATIBLE
hello_world
magic_wand
micro_speech
person_detection

Upload the code and press the reset button on Ameba once the upload is finished.
Once it is running, you should see the blue LED flashing once every few seconds, indicating that it has finished processing an image. The red LED will light up if it determines that there is no person in the previous image captured, and the green LED will light up if it determines that there is a person.
The inference results are also output to the Arduino serial monitor, which appear as follows:



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

UART - Communicate with PC over USB to Serial Module

Introduction of UART

UART uses two wire, one for transmitting and the other one for receiving, so the data transmission is bidirectional. The communication uses a predefined frequency (baud rate) to transmit data. In Arduino, UART is called “Serial”. There is only one hardware UART on Arduino Uno and it is primarily used to read the log and messages printed by Arduino (so it is also called “Log UART”). If we use the hardware UART for other purposes, the Log UART does not have resources to function. To provide more UART connections, it is possible to use a GPIO pin to simulate the behavior of UART with a software approach, this is called Software Serial. Ameba is equipped with several hardware UART ports, but it is also compatible with the Software Serial library.

Materials

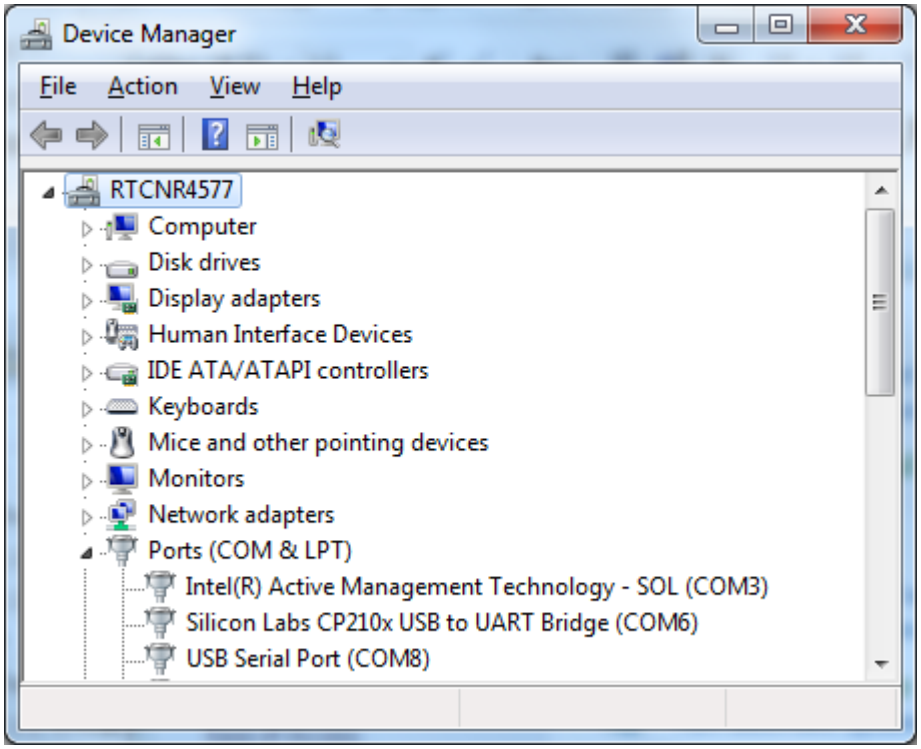
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

Example

In this example, we use UART to connect USB to TTL adapter to Ameba.
USB to TTL adapter sends data to Ameba, the data would be returned by Ameba, and showed on the screen.

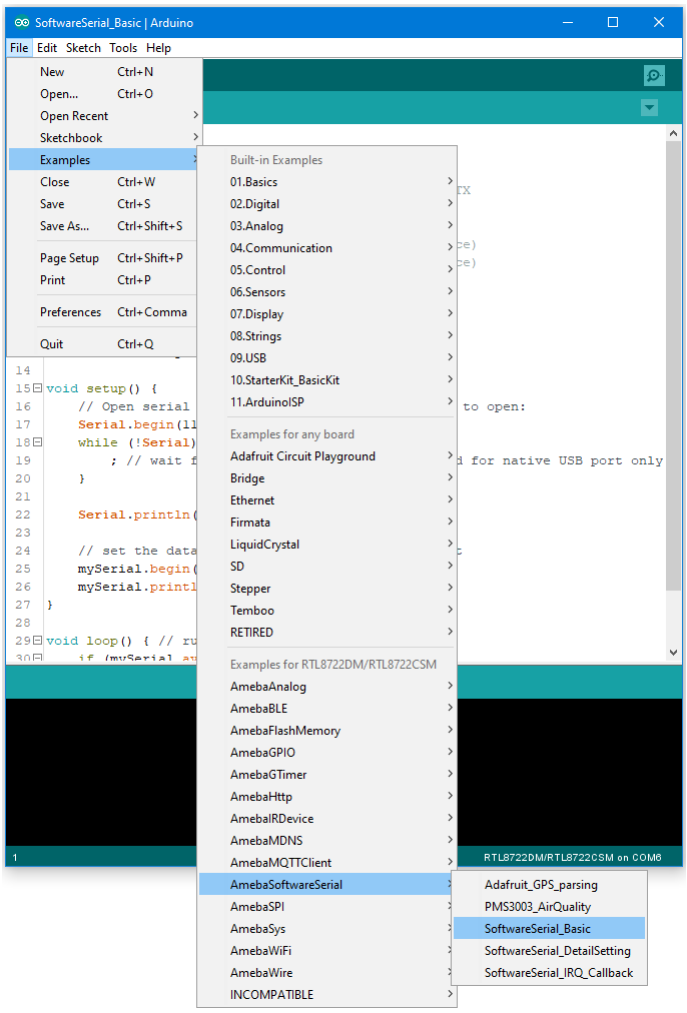
- **Install USB to TTL Adapter**

USB to TTL adapter converts USB to serial interface. Normally, there are at least 4 pins on the adapter, that is 3V3 (or 5V), GND, TX and RX. Generally, installing the driver for the USB to TTL adapter would be required before using it. If the adapter uses the chip of FTDI, Windows will search and install the driver automatically, otherwise, you may need to install corresponding driver yourself.
Afterwards, open device manager. You can find corresponding serial port number of the USB to TTL adapter:



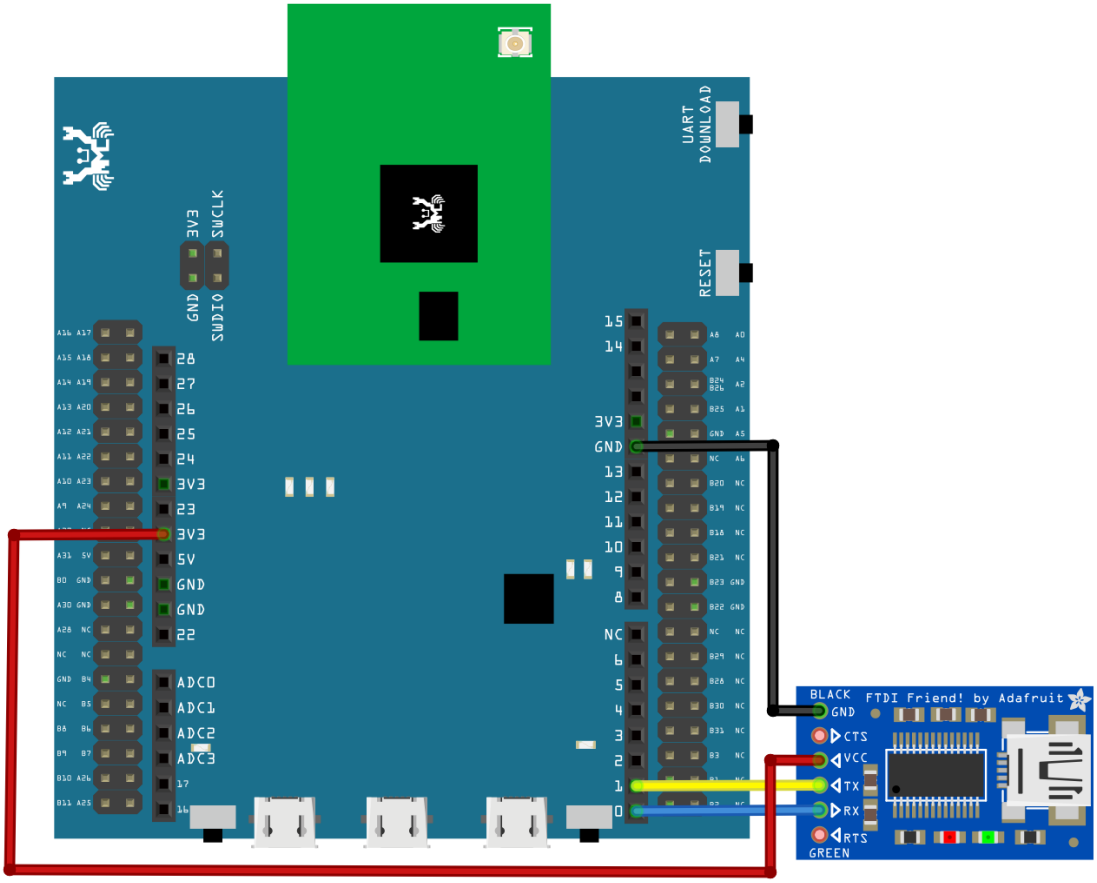
- Executing the Example

Open the “SoftwareSerialExample” example in “`File`” -> “`Examples`” -> “`AmebaSoftwareSerial`” -> “`SoftwareSerial_Basic`”:

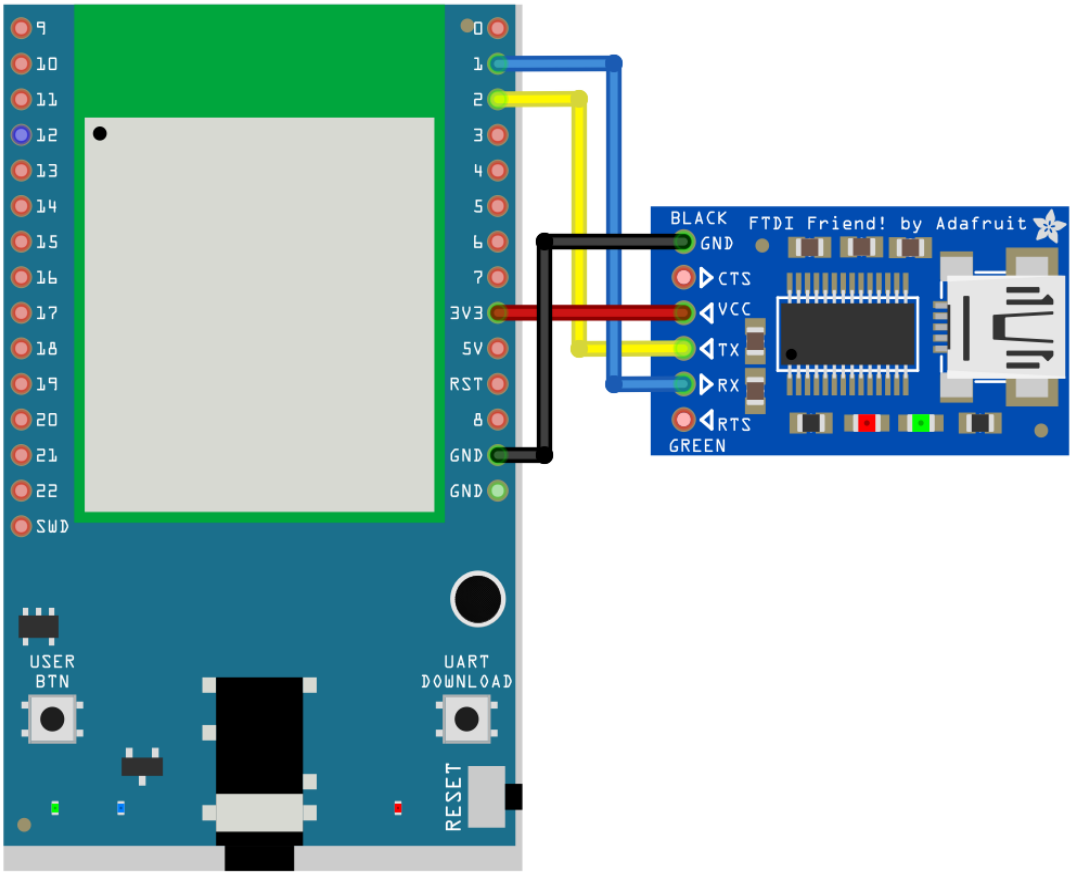


Connect the wire as the following diagrams show. The TX pin of USB to TTL adapter is connected to the RX of Ameba, and the RX pin of USB to TTL adapter is connected to the TX of Ameba.

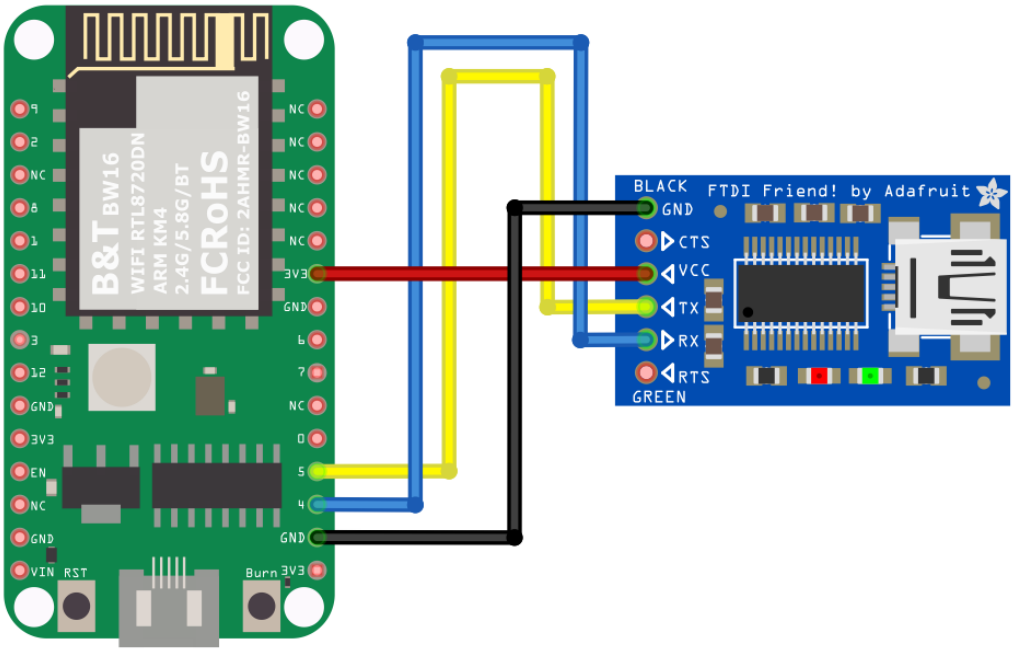
AMB21 / AMB22 Wiring Diagram:



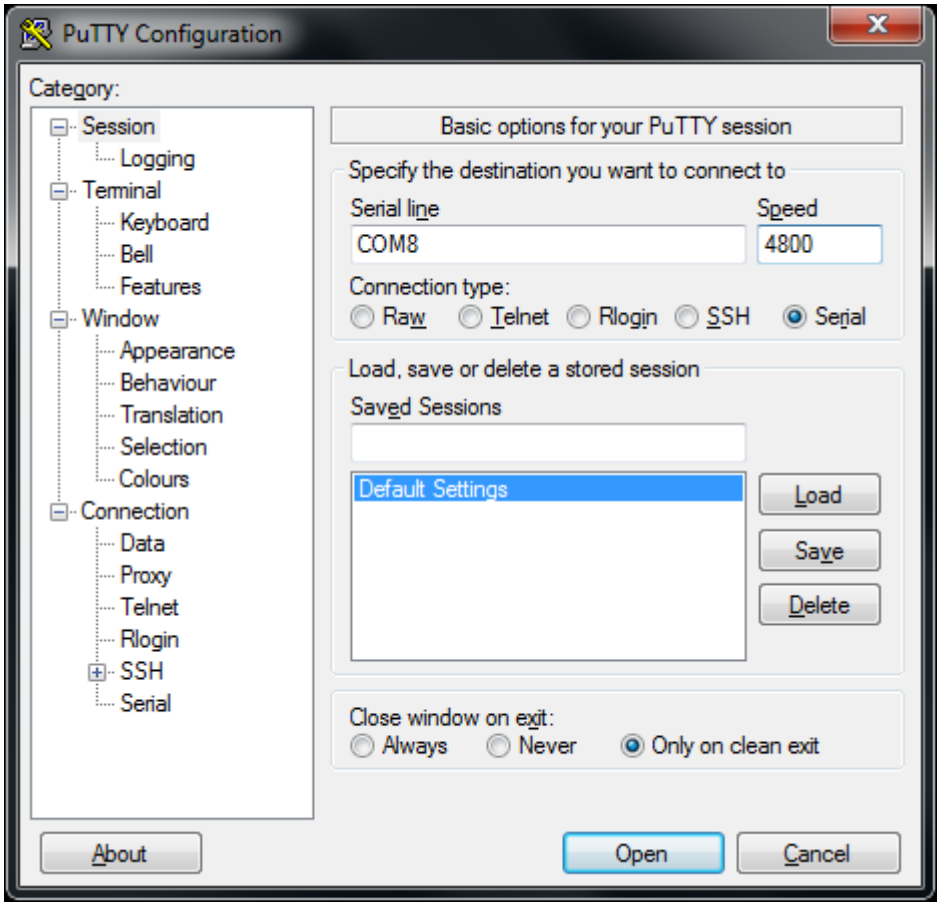
AMB23 Wiring Diagram:



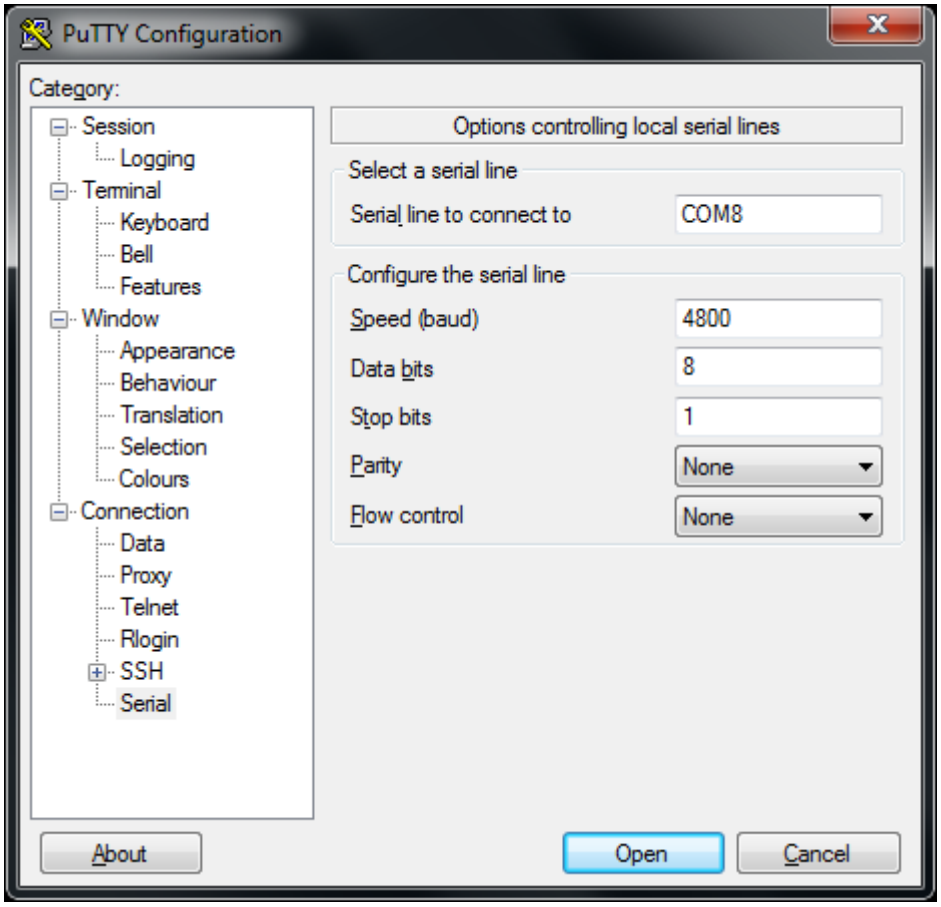
BW16 Wiring Diagram:



Next, open a serial port terminal, such as Putty or Tera Term. (Putty is used in this example). Open the Putty window, choose "Serial" in connection type, and specify the port number of the USB to TTL adapter (e.g. COM8). In the speed field, fill in the baud rate of this connection. Note that both sides of the connection should use the same baud rate. In this example we set baud rate 4800.



Next, select “Serial” on the left side. Set data bits to 8, stop bits to 1, parity to none, and flow control to none.



Then click Open and press the reset button on Ameba. You can see the "Hello, world?" message appears in Putty. If characters are typed into Putty, the input characters would be sent to Serial RX of Ameba by TX of USB to TTL Adapter, and returned by Serial TX of Ameba. Finally, RX of USB to TTL Adapter receives the returned characters and prints them in Putty. Therefore, if you insert "I am fine", you will get something like this:

Code Reference

First, use `SoftwareSerial:begin(speed)` to set the baud rate for the serial communication:
https://www.arduino.cc/en/Reference/SoftwareSerialBegin

Use `write()` to send data, and use `SoftwareSerial:available()` to get the number of bytes available for reading from a software serial port:

https://www.arduino.cc/en/Reference/SoftwareSerialAvailable
If there are data available to read, use read() to read from serial port.

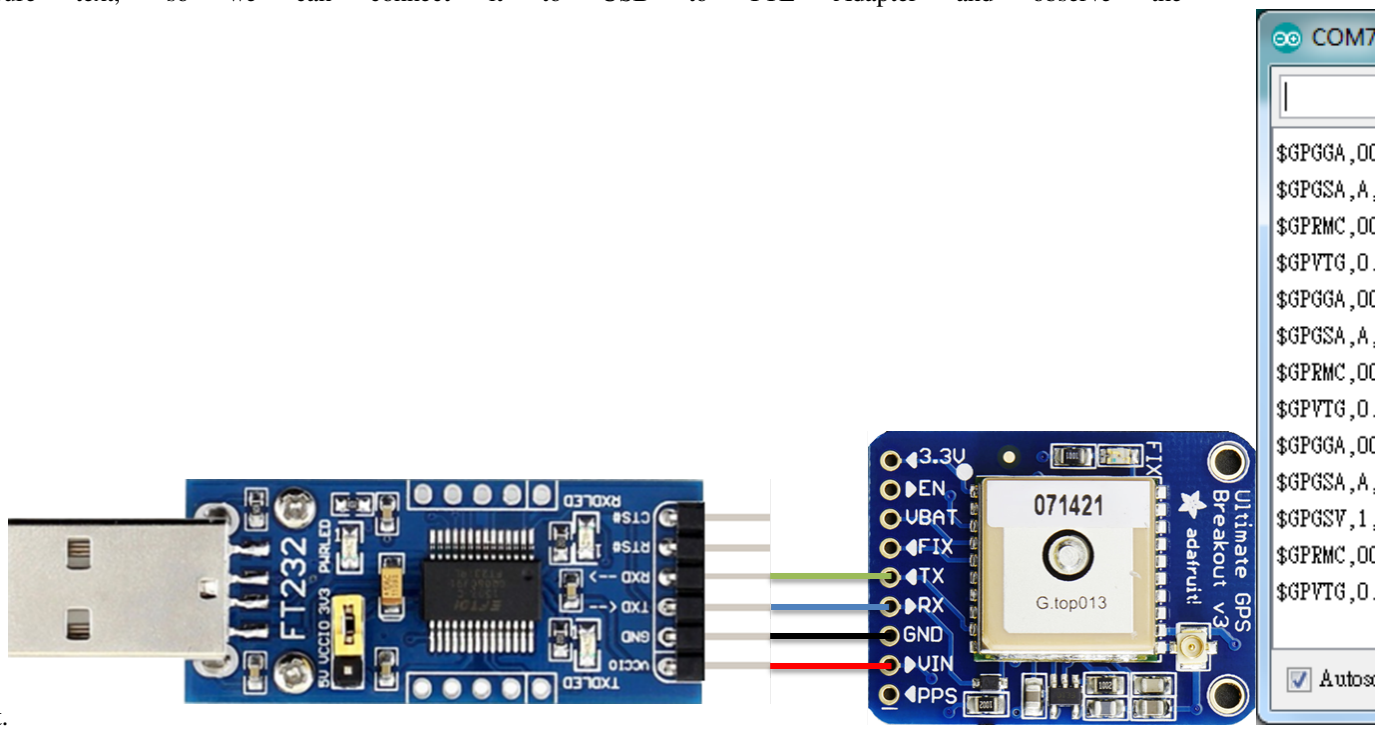
UART - Retrieve GPS Position

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Adafruit Ultimate GPS Breakout x 1 (Refer to official document)

Example

In this example, we use Adafruit Ultimate GPS Breakout. Its data format is pure text, so we can connect it to USB to TTL Adapter and observe the output.



It follows the NMEA sentence format (refer to http://aprs.gids.nl/nmea/) The GPS signal is weak in indoor environment. The status that the GPS signal is not received is called “not fix”. Bring the GPS module outdoors, when the GPS signal is “fix”, you would get message similar to the figure

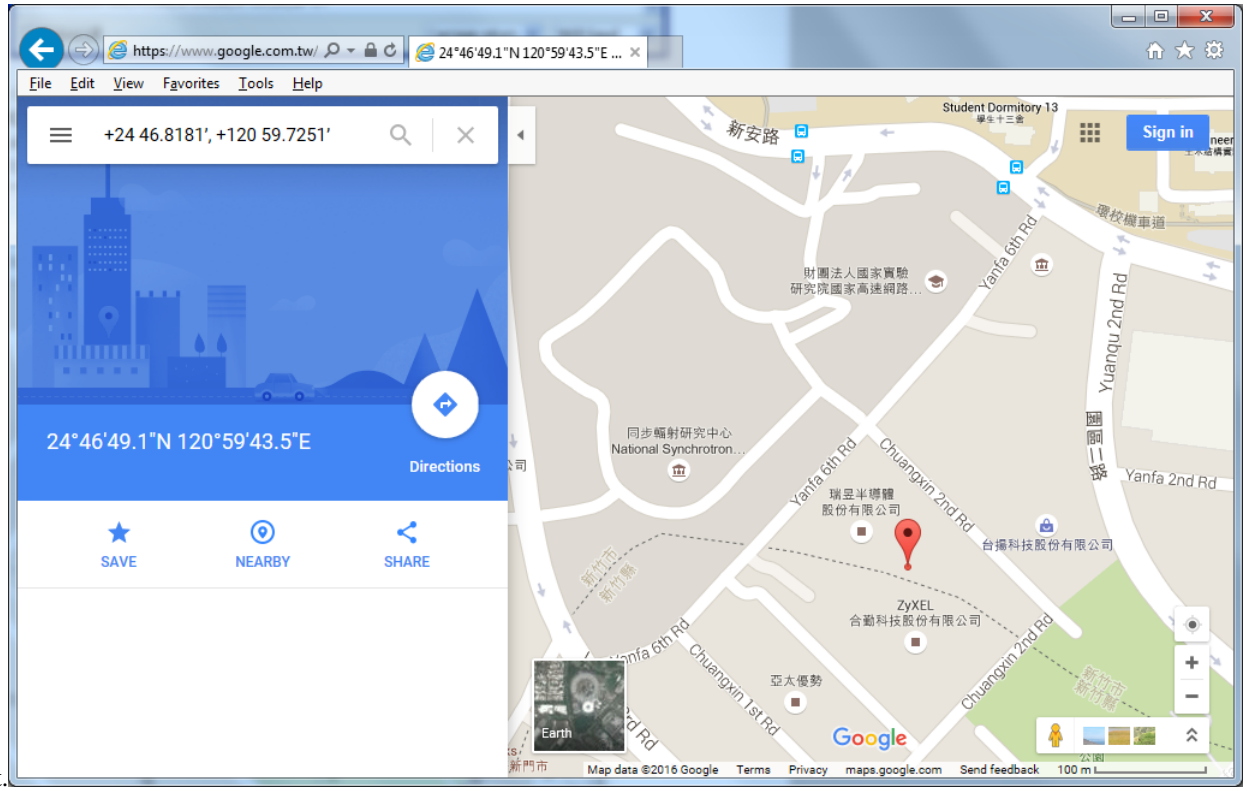
below.

In this example we are only interested in the "$GPRMC (Global Positioning Recommended Minimum Coordinates)":
$GPRMC,032122.000,A,2446.8181,N,12059.7251,E,0.39,78.89,270116,,,A*53
Each field is separated by a comma.

- First field is the GMT time (Greenwich Mean Time), that is 032122.000 in this example. The time format is HH:MM:SS.SSS, i.e., 03:21:22.000. Note that the time zone and the daylight-saving time adjustment should be handled on your own.
- Second field represents the status code
  - V: Void (Invalid)
  - A: Active, meaning the GPS signal is fix.
- The third to sixth fields represent the geolocation

In this example, 2446.8181,N represents 24 degrees 46.8181 minutes north latitude, and 12059.7251,E represents 120 degrees 59.7251 minutes east longitude.

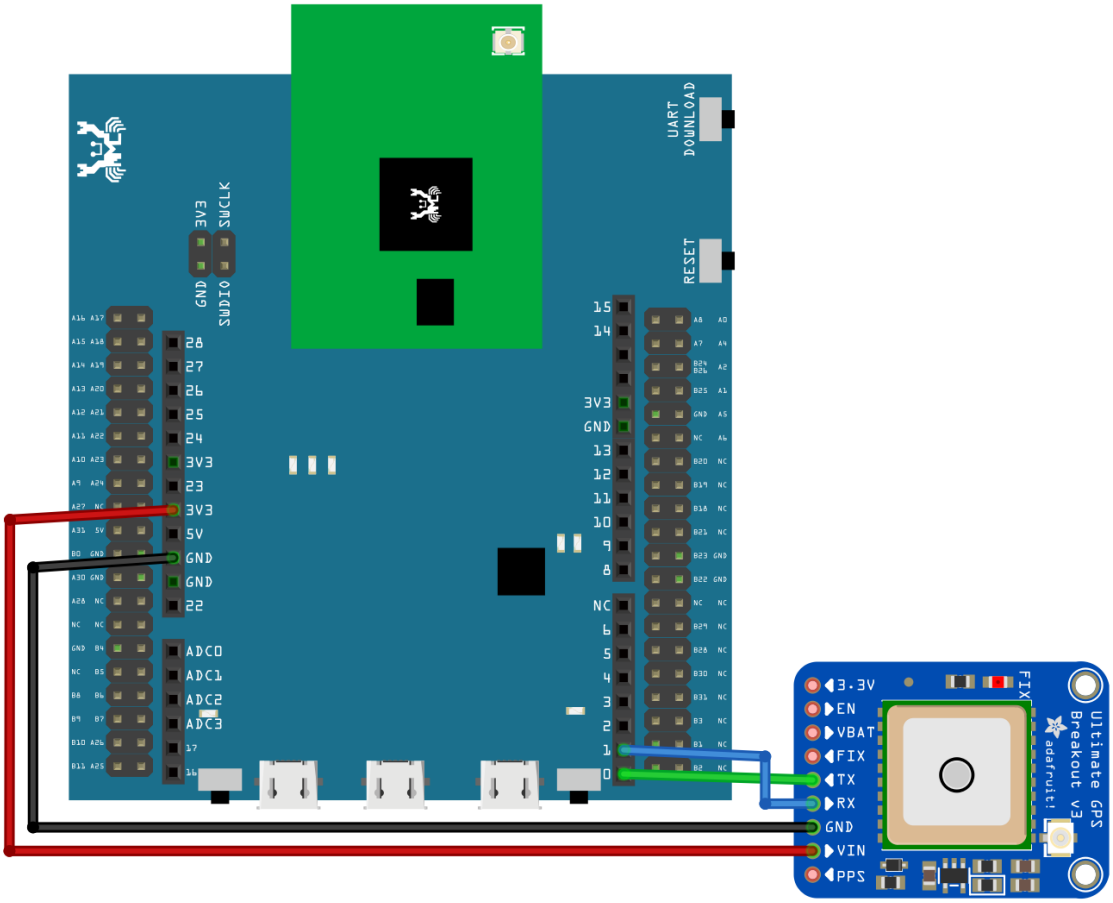
We can search **+24 46.8181', +120 59.7251'** in Google map to check whether the position is



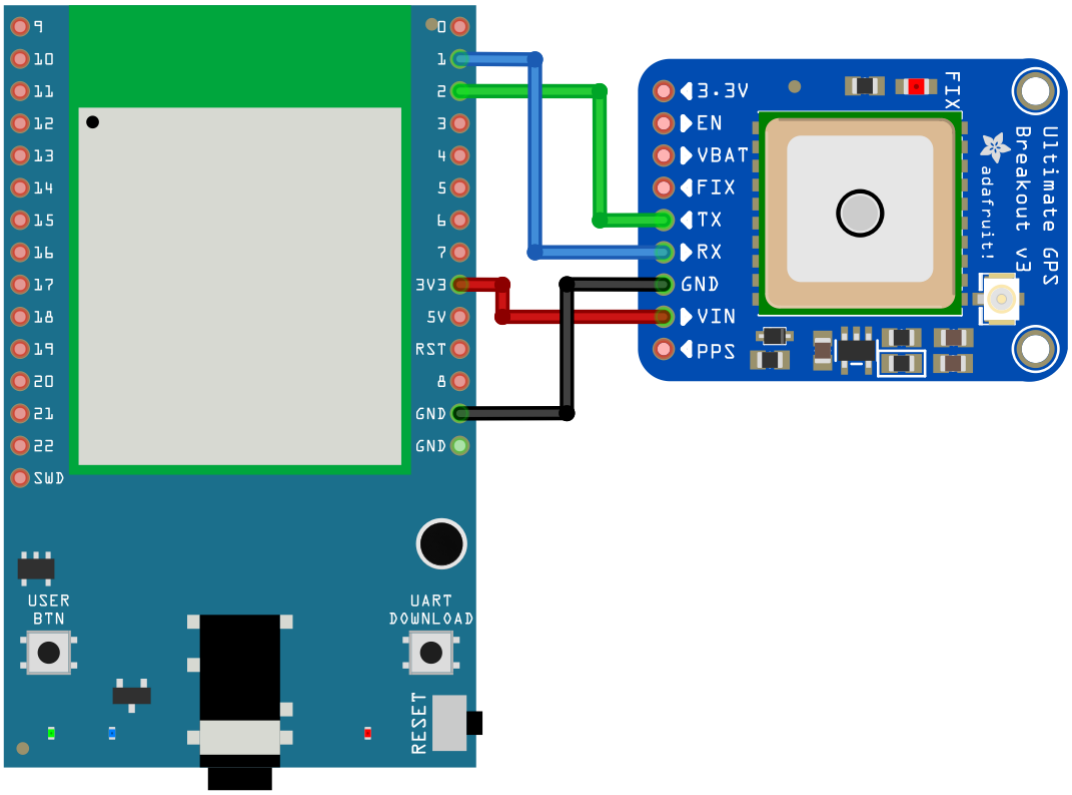
correct.

- The seventh field is relative speed(knot). 1 knot = 1.852km/hr, in this example the relative speed is 0.39 knot.
- The eighth field is the moving angle, which is calculated by its moving orbit.
- The ninth field is the date with format ddMMyy. In this example, "270116" stands for day 27, January, year 2016.
- The last field is checksum. In the example we have *53 as checksum.

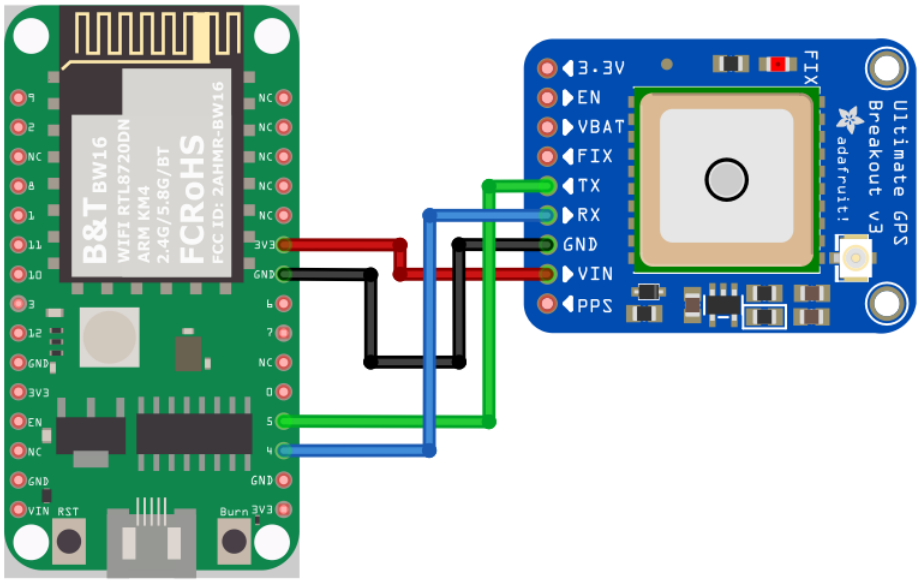
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

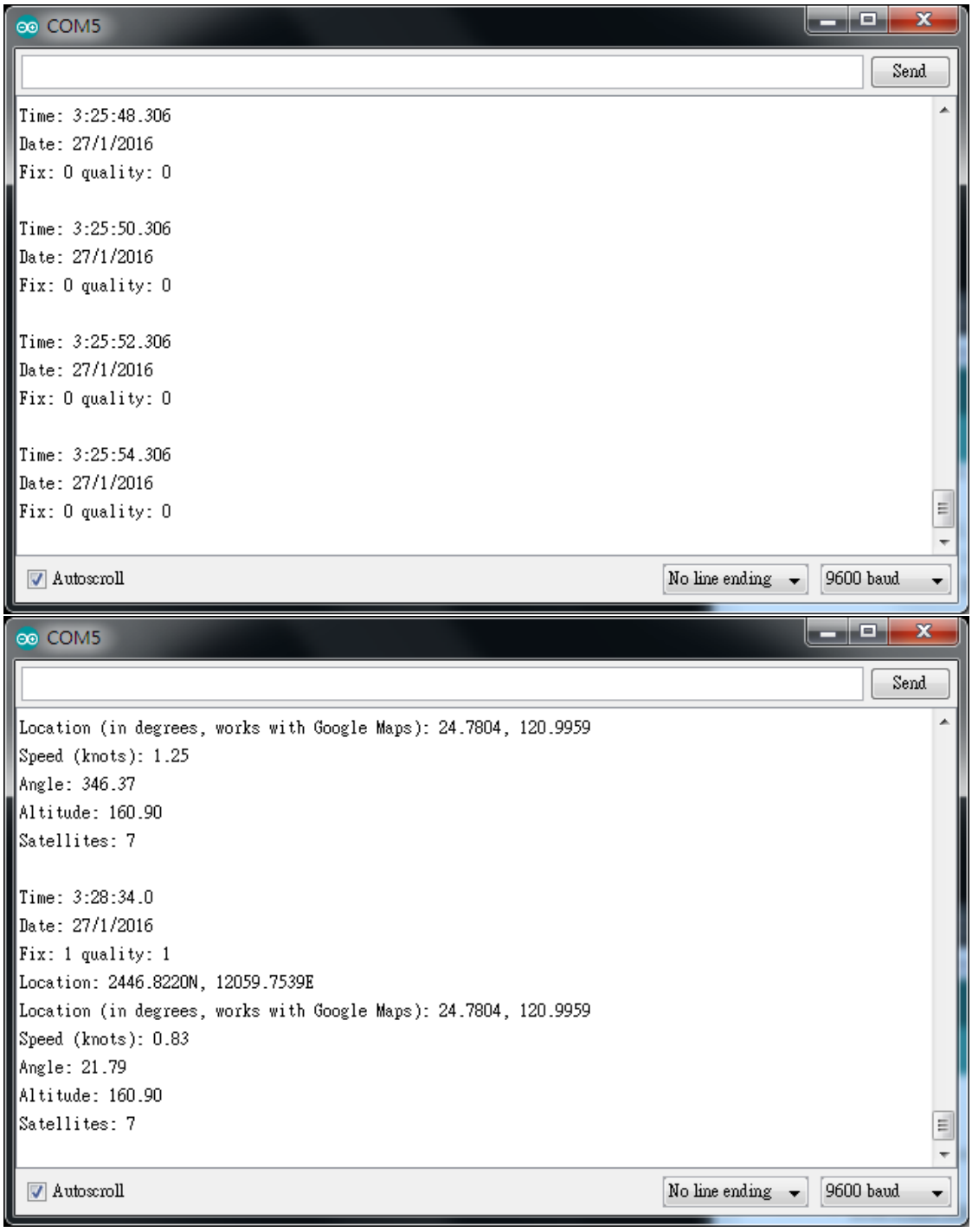


AMB23 Wiring Diagram:



Open the example in "Files" -> "Examples" -> "AmebaSoftwareSerial" -> "Adafruit_GPS_parsing".

Compile and upload to Ameba, then press the reset button.
The result will be output to Serial Monitor:



UART – Set Callback Function For UART Communications

Materials

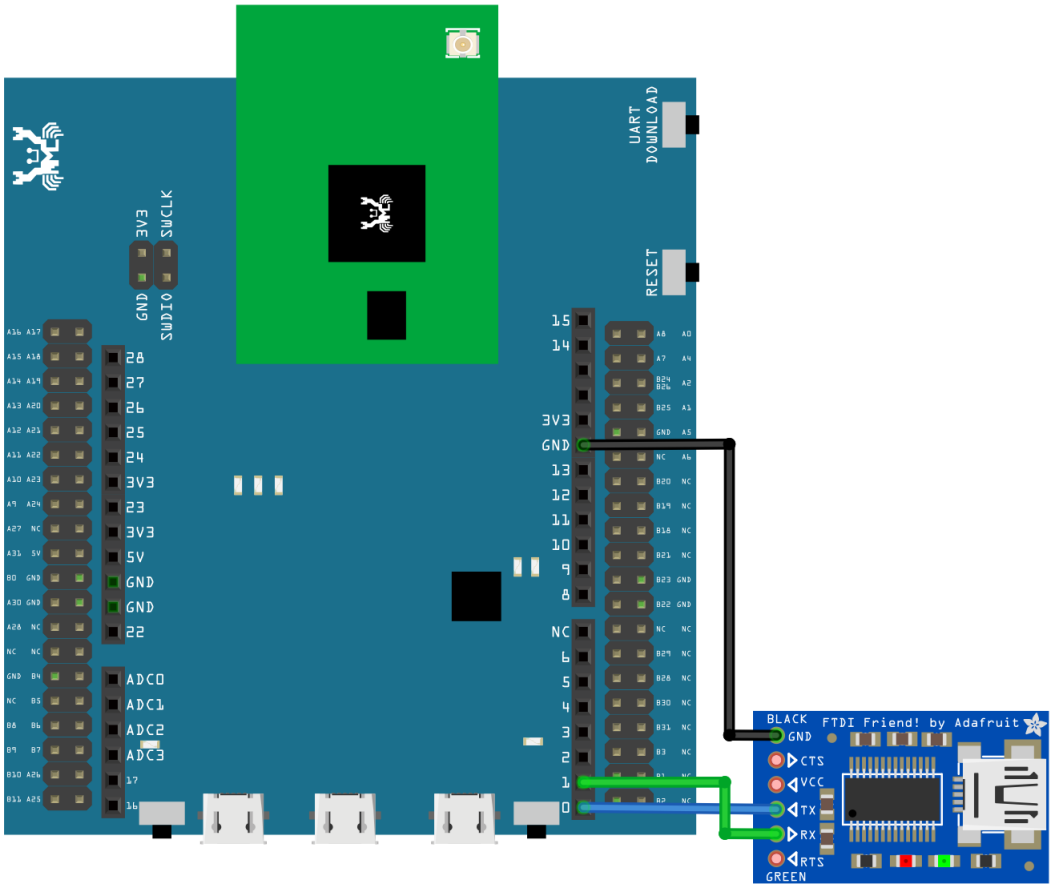
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

Example

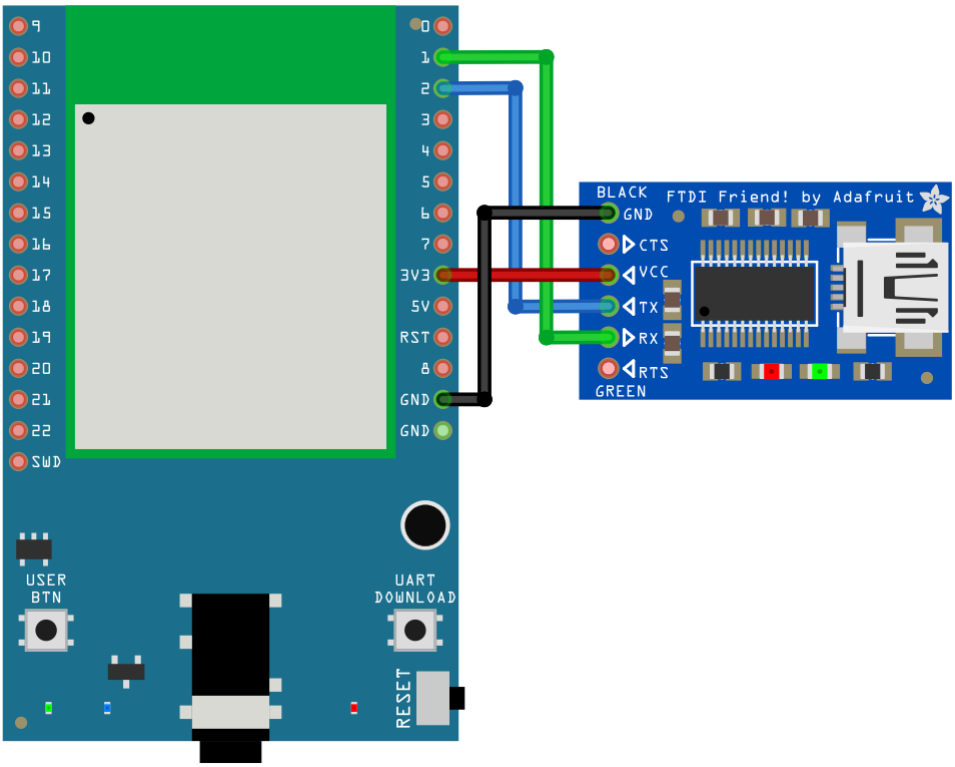
This example shows how to set a callback function for UART communication to process the UART data.

A USB to TTL adapter is required for this example. Ensure that you have the driver installed and connect it to the Ameba board as shown.

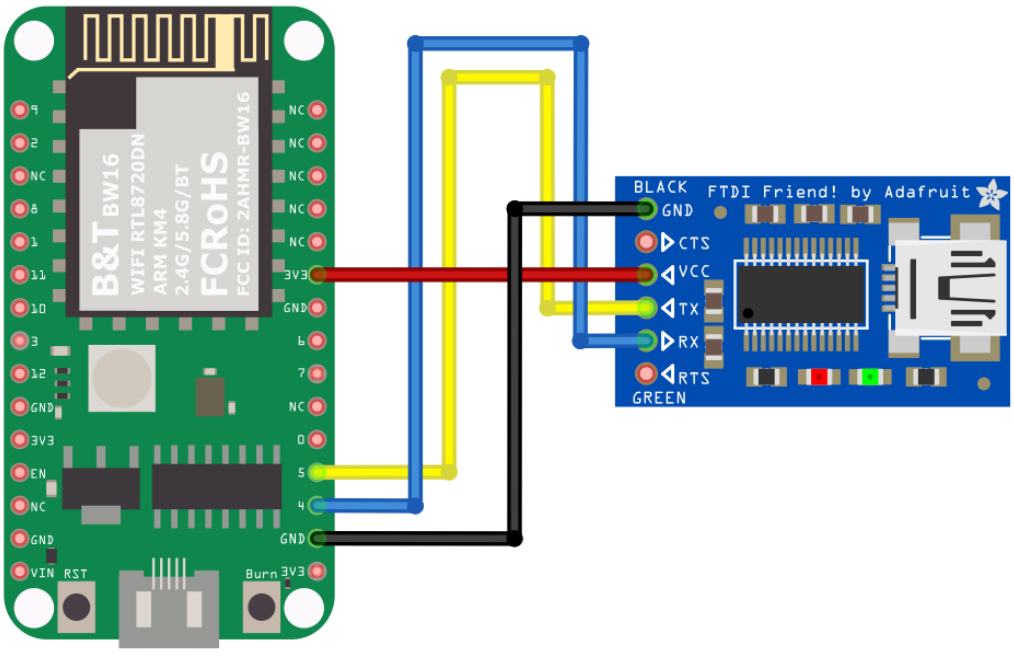
AMB21 / AMB22 Wiring Diagram:



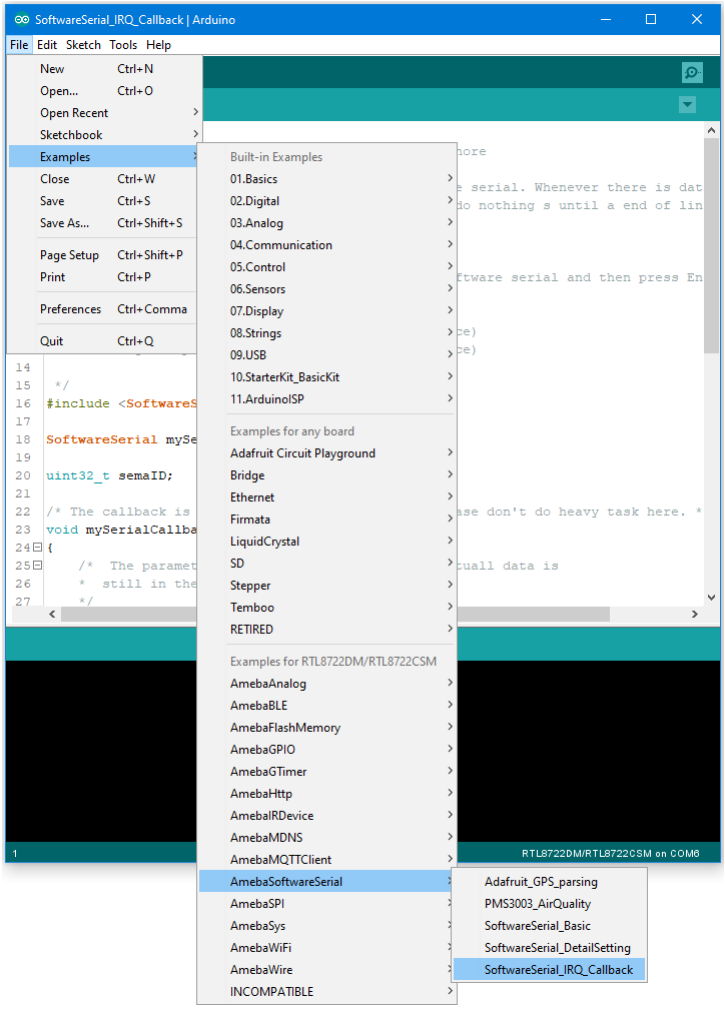
AMB23 Wiring Diagram:



BW16 Wiring Diagram:

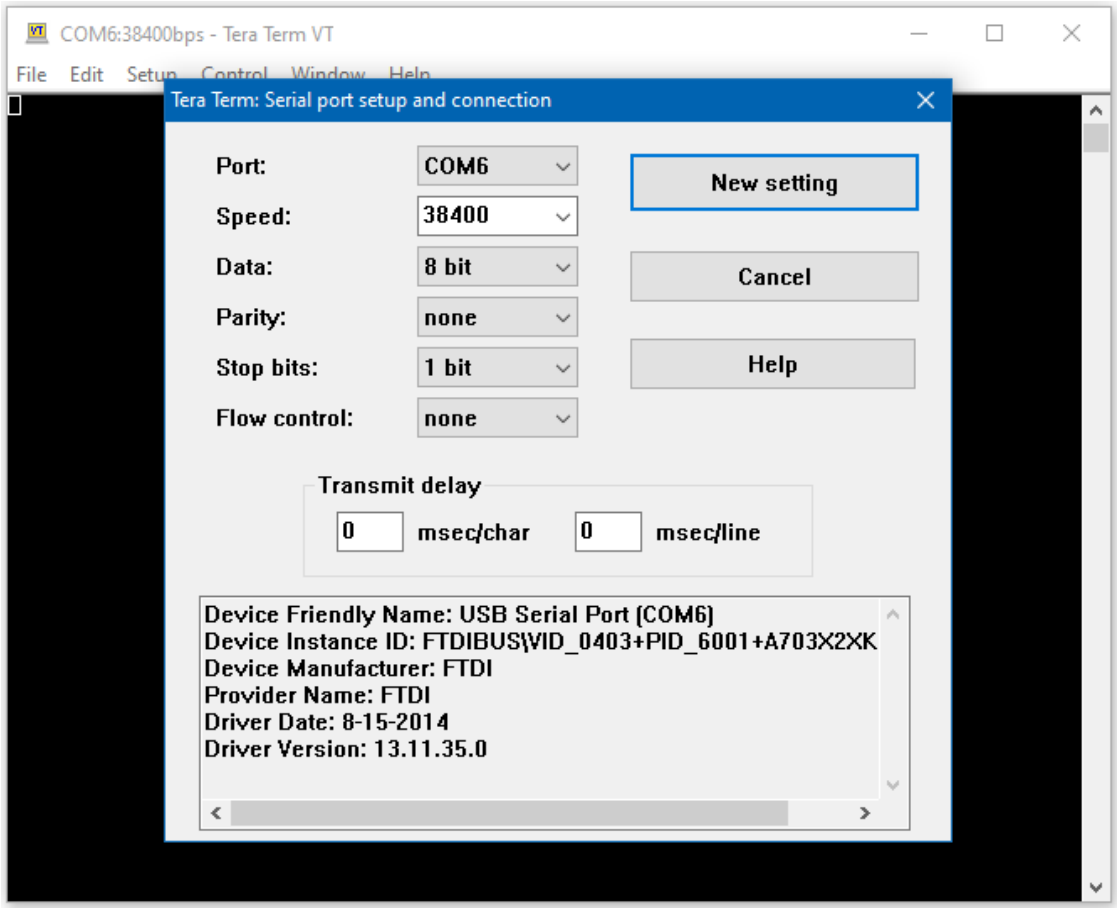


Open the example in "`File`" -> "`Examples`" -> "`AmebaSoftwareSerial`" -> "`SoftwareSerial_Irq_Callback`"

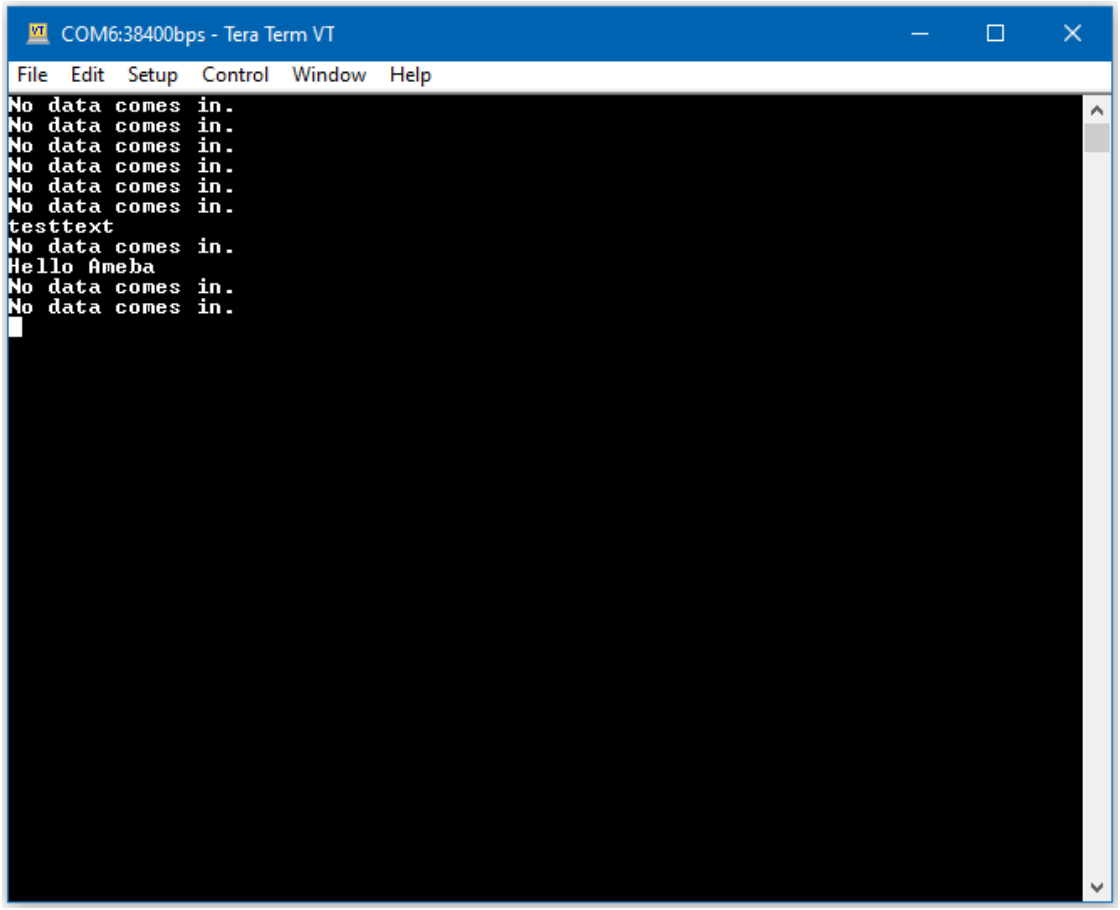


Upload the code and press the reset button on Ameba once the upload is finished.
Next, using a terminal program, such as TeraTerm or PuTTY, open a serial port and configure it according to the settings. Make sure the serial port number corresponds to the USB to TTL adapter.

- Speed: 38400
- Data: 8 bit
- Parity: none
- Stop bits: 1 bit
- Flow control: none



Once the serial port is open, type in the terminal and press the enter key, and you will see the corresponding output.



Code Reference

`mySerial.setAvailableCallback(mySerialCallback);` is used to set the function mySerialCallback as a callback function for software serial. When a new character is received, the callback function checks if the character corresponds to the enter key, and releases the semaphore if it is true, which in turn allows the main loop to print out all the previously received characters.

UART - PM2.5 Concentration in The Air

Preparation

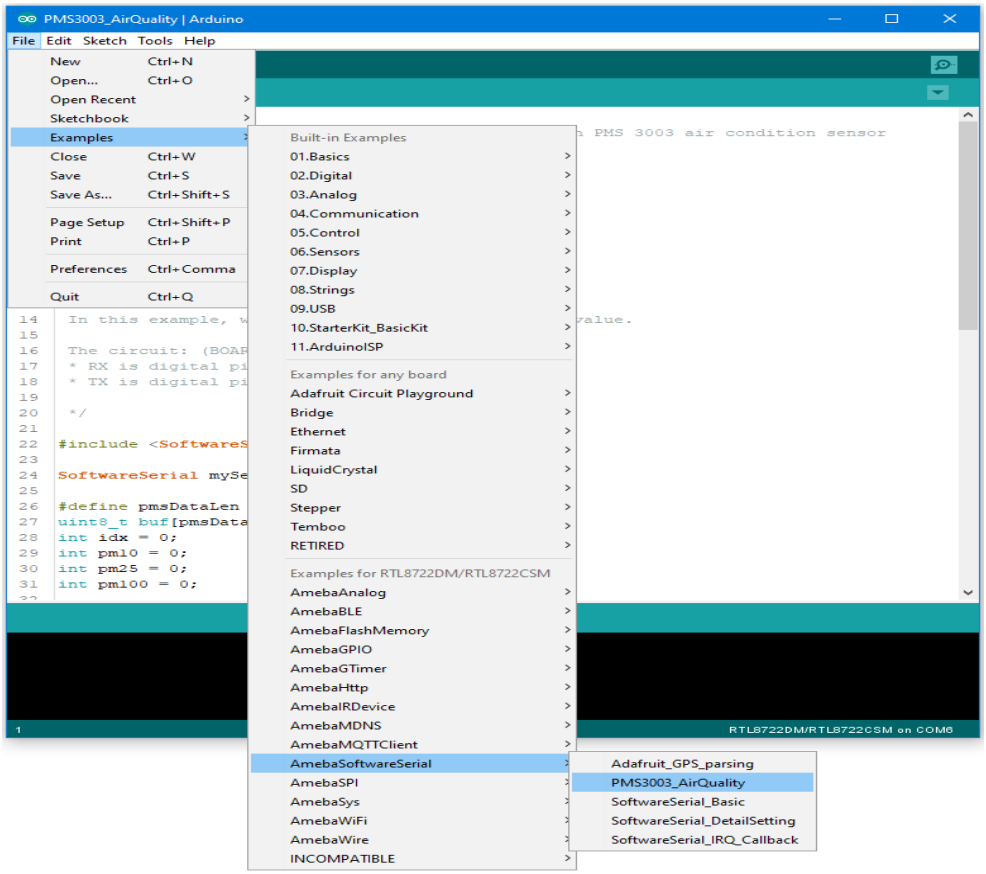
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x 1

Example

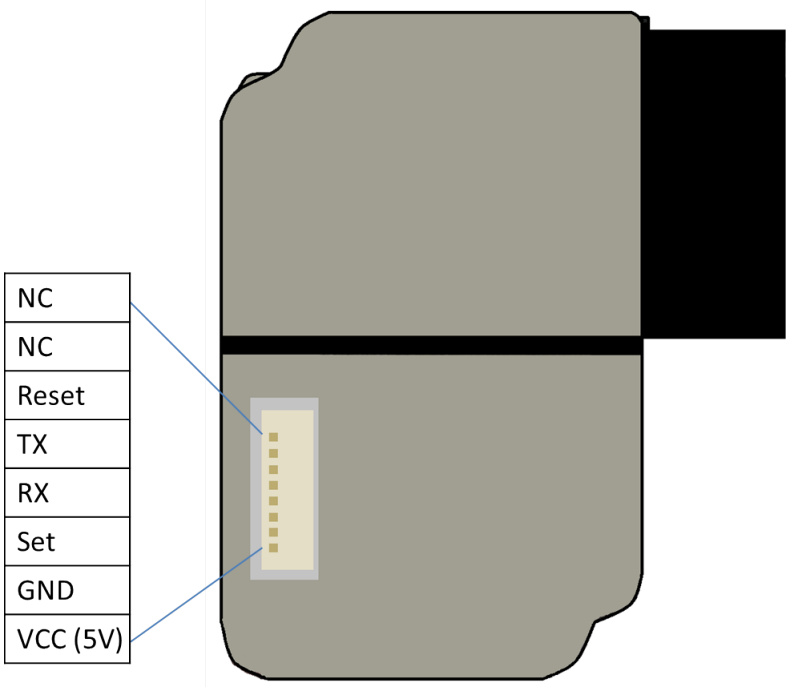
PMS3003 (or PMS5003) is a sensor of air quality, it can detect the concentration of those 0.3 to 10 micrometer particulate matters in the air. The sensor output its data via UART.

The PMS3003 (or PMS5003) sensor detects the concentration value of PM 1.0, PM 2.5, PM 10. Take PM 2.5 for example, it stands for the fine particles with a diameter of 2.5 micrometers or less.

Open the example in “File” -> “Examples” -> “AmebaSoftwareSerial” -> “PMS3003_AirQuality”



There are 8 pins in PMS3003:

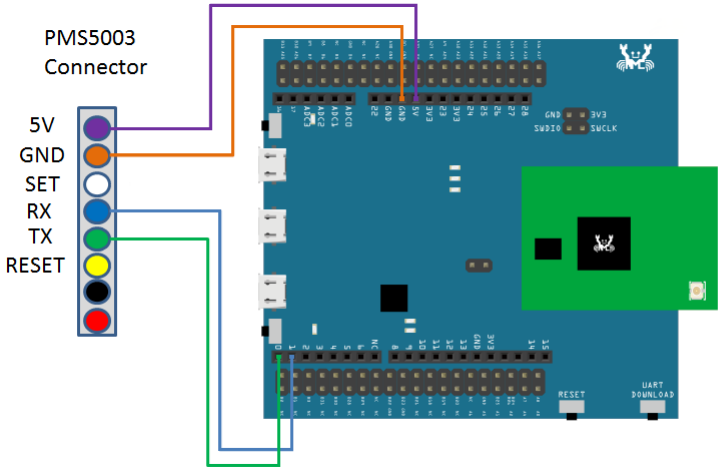


PMS3003 requires 5V power, but the working voltage of its IC is 3.3C. Therefore, the working voltage of Reset, TX, RX, Set are 3.3 as well. If the "Set" pin is pulled to high, the PMS3003 is put to operating mode. If the "Set" pin is pulled low, the PMS3003 is put to standby mode.

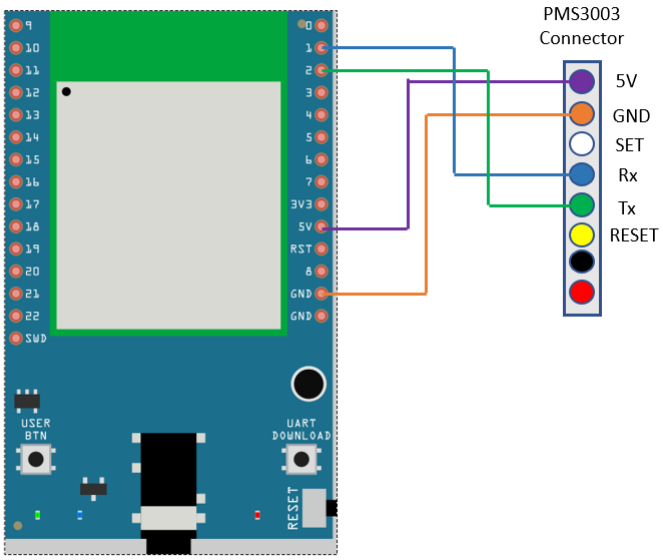
TX/RX pins are for UART connection. Under operating mode, PMS3003 output the data it reads continuously. Each data is of 32 byte, please refer to the following article for detailed data format information:

https://www.dfrobot.com/wiki/index.php?title=PM2.5_laser_dust_sensor_SKU:SEN0177 RTL8722

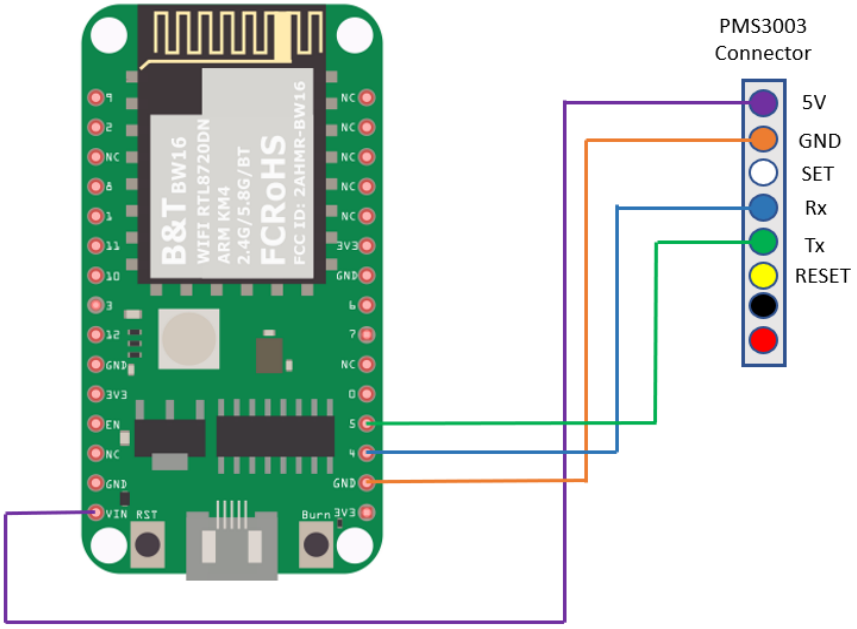
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

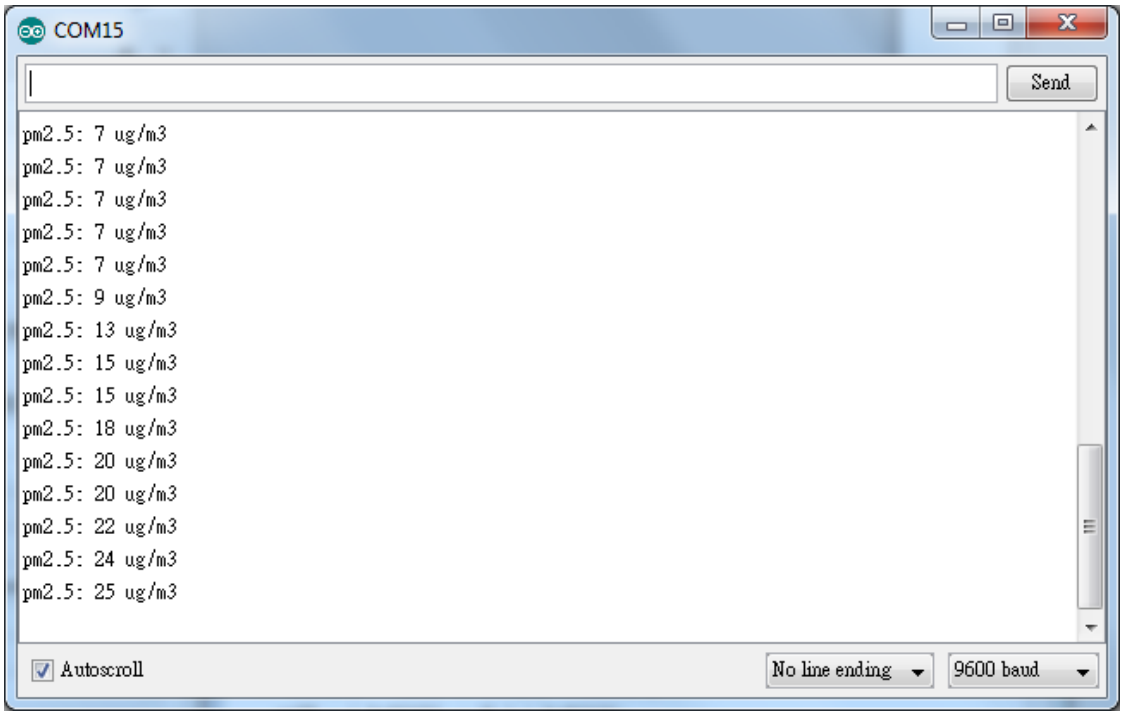


BW16 Wiring Diagram:



In this example, we do not use the “Set” and “Reset” pins.

Compile the code and upload it to Ameba. After pressing the Reset button, Ameba starts to output the PM 2.5 data to serial monitor.



Watchdog - Simple WDG Timer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

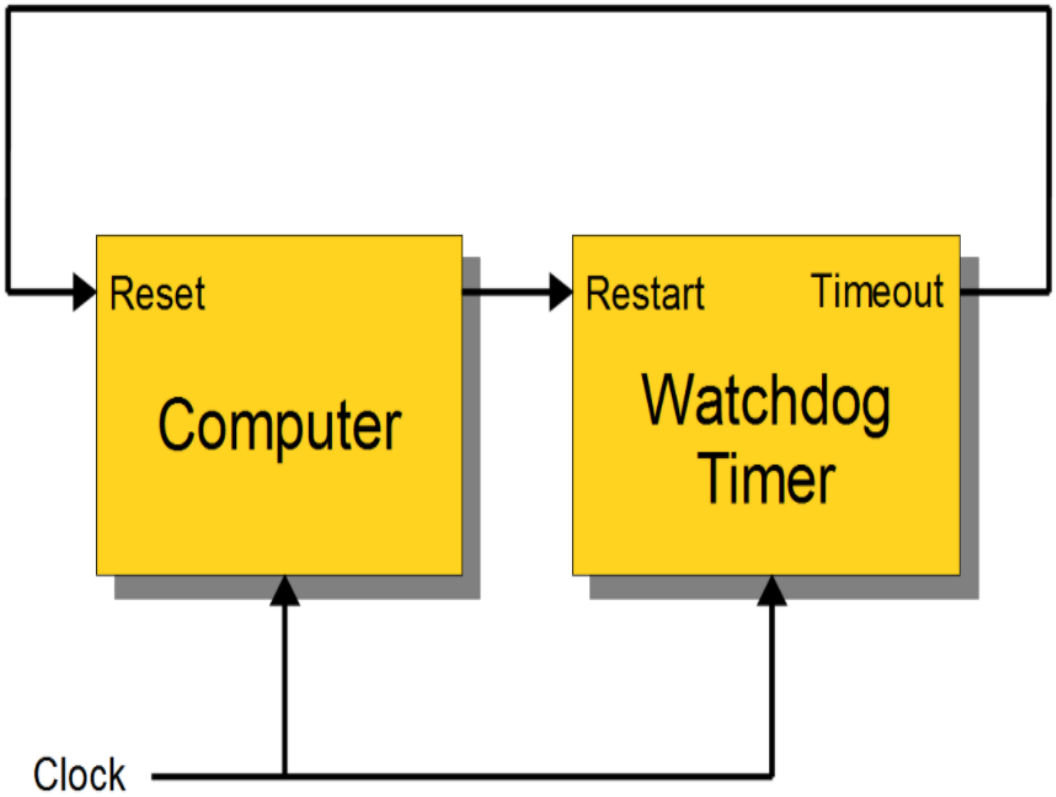
Example

In this example, we will use this simple watchdog timer example runs on the Ameba RTL8722 module to illustrate how to use the watchdog API. Before we get into the details of the example, let's briefly go through the definition of Watchdog as well as it's working principles.

Watchdog

Watchdog Timer (WDT) is a hardware timer that is used to detect the occurrence of a software fault, then automatically generates a system reset or a watchdog interrupt on the expiry of a programmed period.

In layman terms, imagine in the situation while your micro-controller is confused in an infinity loop, or any case like the micro-controller hang while performing some tasks. The normal troubleshooting method would be to press the reset button and jump out of the infinity loop. However, is it practically impossible to do press on the button all time, therefore, the watchdog timer that embedded inside the micro-controller would help with this situation.



Feed the Dog

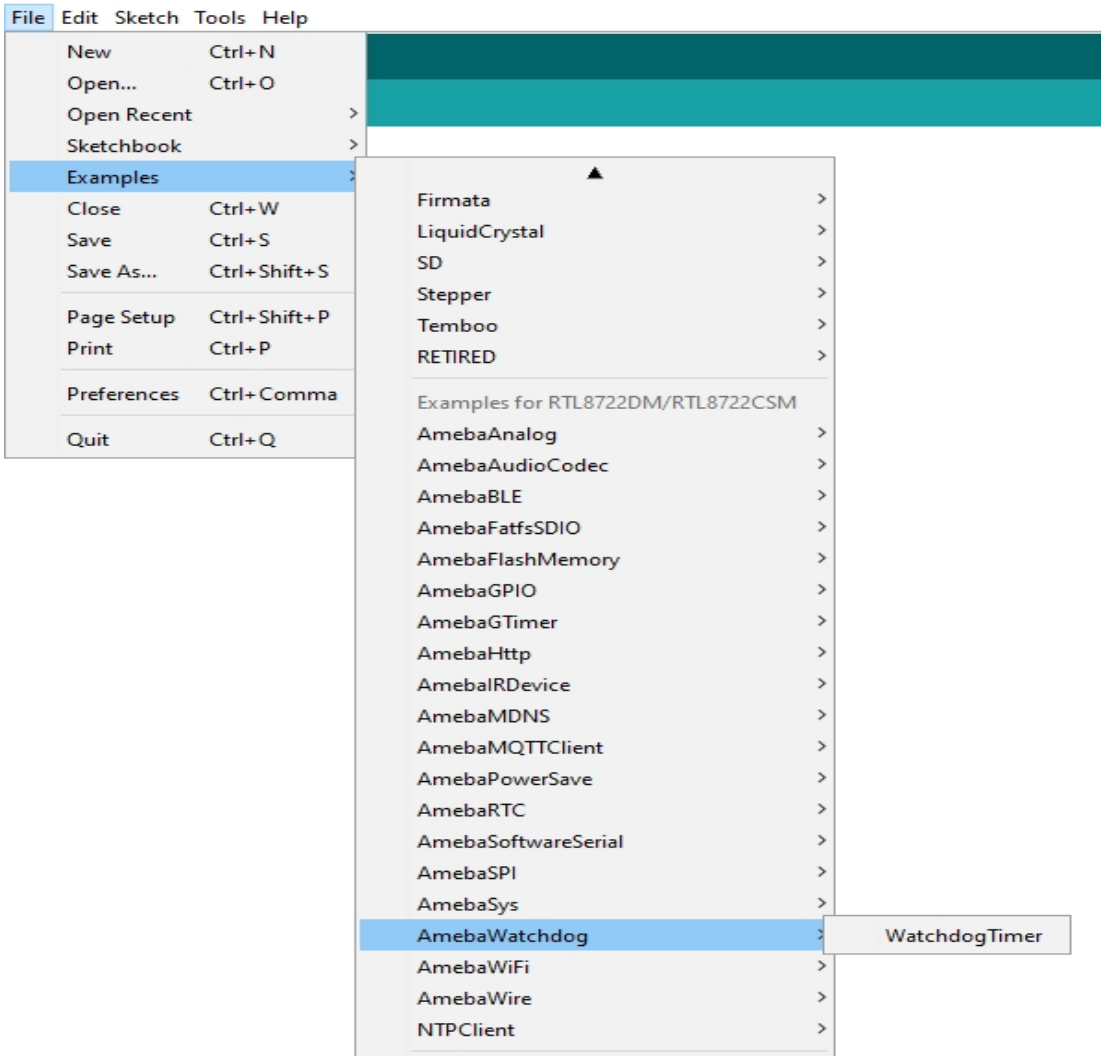
If you have a dog in your home. You need to feed that dog at a regular interval. if you can't feed one day, it will bite

you! And likewise, this is the working logic behind the watchdog timer.

In our example, we created 2 tasks that contain some loop that runs repeatedly, one is called “Small_Task” and the other is called “Big_Task”. We are enabling the watchdog timer is loaded with an initial value (5 seconds) greater than the total delay in the “Small_Task”, but shorter than the “Big_Task”.
For the successful case, the watchdog is being refreshed/feed within 5 seconds, however, for the failed case, the loop is under processing and the watchdog is not being fresh after 5 seconds, which triggers the watchdog (dog barks), an interrupt is generated to reset the processor. Likewise, the watchdog timer protects the micro-controller from the hanging case.

Then we move to the coding part for this example, for this example, you will only need the RTL8722CSM/RTL8722DM/RTL8722DM MINI Board itself.

Firstly, make sure the correct Ameba development board is selected in Arduino IDE: “`Tools`” -> “`Board`” -> “`RTL8722CSM/RTL8722DM`” (or “`RTL8722DM MINI`”). Then open the “Watchdog Timer” example in “`File`” -> “`Examples`” -> “`AmebaWatchdog`” -> “`Watchdog Timer`”:



Upon successfully upload the sample code, open the serial monitor, and press the reset button. You will find that the "Small_Task" can refresh the watchdog within the 5 seconds (initialized in the watchdog timer). However, the "Big_Task" will not be able to refresh the watchdog within 5 seconds, which the watchdog "barks" then the microcontroller reset.


```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
```

COM10

```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
watchdog barks!!!
doing dummy task #6
doing dummy task #7
doing dummy task #8
doing dummy task #9
doing dummy task #10
Big_Task finished refresh watchdog.
```

Community Examples

Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards.
To make contributions, please visit our official GitHub Wiki page.

Tip: Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards

1.1.4 Board HDK

EVB

- **HDK-AMEBAD_MB_4V2**
  - Layout
  - Schematic

- **HDK-AMEBAD_MB_4V0**
  - Layout
  - Schematic

RTL8722DM Module

- **HDK-AM8722DM01_6V2_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_6V1_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_4V1**
  - Layout
  - Schematic

1.1.5 API Documents

RTL8722DM ARDUINO Online API Documents

Analog

Class AmebaServo

AmebaServo Class

Description

Defines a class of manipulating servo motors connected to Arduino pins.

Syntax

class AmebaServo

Members

Public Constructors	
AmebaServo::AmebaServo	Constructs an AmebaServo object.
Public Methods	
AmebaServo::attach	Attach the given pin to the next free channel.
AmebaServo::detach	Detach the servo.
AmebaServo::write	Write value, if the value is < 200 it's treated as an angle, otherwise as pulse-width in microseconds.
AmebaServo::writeMicroseconds	Write pulse width in microseconds.
AmebaServo::read	Output current pulse width as an angle between 0 and 180 degrees.
AmebaServo::readMicroseconds	Output current pulse width in microseconds for this servo.
AmebaServo::attached	Check if the servo is attached.

AmebaServo::attach

Description

Attach the given pin to the next free channel, sets pinMode (including minimum and maximum values for writes), returns channel number, or 0 if failure.

Syntax

```
uint8_t attach(int pin);

uint8_t attach(int pin, int min, int max);
```

Parameters

pin: The Arduino pin number to be attached.

min: Minimum values for writes.

max: Maximum values for writes.

Returns

The function returns channel number or 0

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree.

Listing 1: ServoSweep.ino

```
1  /* Sweep
2  by BARRAGAN < http://barraganstudio.com >
3  This example code is in the public domain.
4  modified 8 Nov 2013
5  by Scott Fitzgerald
6  http://www.arduino.cc/en/Tutorial/Sweep
```

(continues on next page)

(continued from previous page)

```
refined 2016/03/18 by Realtek
*/

#include "AmebaServo.h"

// create servo object to control a servo
// 4 servo objects can be created correspond to PWM pins

AmebaServo myservo;

// variable to store the servo position
int pos = 0;

void setup() {
#if defined(BOARD_RTL8195A)
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#elif defined(BOARD_RTL8710)
    // attaches the servo on pin 13 to the servo object
    myservo.attach(13);
#elif defined(BOARD_RTL8721D)
    // attaches the servo on pin 8 to the servo object
    myservo.attach(8);
#else
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#endif
}

void loop() {
    // goes from 0 degrees to 180 degrees in steps of 1 degree
    for (pos = 0; pos <= 180; pos += 1) {
    // tell servo to go to position in variable 'pos'
    myservo.write(pos);
    // waits 15ms for the servo to reach the position
    delay(15);
    }
    // goes from 180 degrees to 0 degrees
    for (pos = 180; pos >= 0; pos -= 1) {
        // tell servo to go to position in variable 'pos'
        myservo.write(pos);
        // waits 15ms for the servo to reach the position
        delay(15);
    }
}
```

Notes and Warnings

Every time must include the header file “AmebaServo.h” in front of the project to use the class function.

AmebaServo::detach

Description

Detach the servo.

Syntax

void AmebaServo::detach(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::write

Description

Write an integer value to the function, if the value is < 200, it's being treated as an angle, otherwise as pulse-width in microseconds.

Syntax

void AmebaServo::write(int value);

Parameters

value: The value < 200 its treated as an angle; otherwise as pulse width in microseconds.

Returns

The function returns nothing.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::writeMicroseconds

Description

Write pulse width to the servo in microseconds.

Syntax

void AmebaServo::writeMicroseconds(int value);

Parameters

value: Write value the pulse width in microseconds.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::read

Description

The function reads current pulse width and returns as an angle between 0 and 180 degrees.

Syntax

int AmebaServo::read(void);

Parameters

The function requires no input parameter.

Returns

The pulse width as an angle between 0 ~ 180 degrees.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::readMicroseconds

Description

The function returns a Boolean value "true" if this servo is attached, otherwise returns "false".

Syntax

int AmebaServo::readMicroseconds(void);

Parameters

The function requires no input parameter.

Returns

The function returns current servo pulse width in microseconds.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::attached

Description

It returns true if this servo is attached, otherwise false.

Syntax

bool AmebaServo::attached(void);

Parameters

The function requires no input parameter.

Returns

The function returns a Boolean value as true or false.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AudioCodec

Class AudioCodec

Description

A class used for general control and management of the hardware audio codec functions.

Syntax

class AudioCodec

Members

Public Constructors

The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named Codec.

Public Methods

AudioCodec::begin	Configure and start the audio codec for transmit and receive operation
AudioCodec::end	Stop all audio codec operation
AudioCodec::getBufferPageSize	Get the byte size of a single page of the audio codec buffer
AudioCodec::setSampleRate	Configure the audio codec transmit and receive sampling rate
AudioCodec::setBitDepth	Configure the audio codec transmit and receive bit depth (bits per sample)
AudioCodec::setChannelCount	Configure the audio codec transmit and receive channel count
AudioCodec::setInputMicType	Configure for analog or digital input microphone type
AudioCodec::setInputLRMux	Configure input left right channel multiplexing
AudioCodec::setDMicBoost	Configure boost gain for digital microphone input
AudioCodec::setAMicBoost	Configure boost gain for analog microphone input
AudioCodec::setADCGain	Configure gain of ADC used to acquire analog input
AudioCodec::muteInput	Mute input audio data stream
AudioCodec::setOutputVolume	Configure output audio volume
AudioCodec::muteOutput	Mute output audio
AudioCodec::writeAvaliable	Check for free buffer page available for data write
AudioCodec::writeDataPage	Write audio data to an available buffer page
AudioCodec::readAvaliable	Check for buffer page with new data available for read
AudioCodec::readDataPage	Read audio data from a ready buffer page
AudioCodec::setWriteCallback	Set a callback function to be notified when a free buffer page is available for write
AudioCodec::setReadCallback	Set a callback function to be notified when a buffer page with new data is available for read

AudioCodec::begin

Description

Configure and start the audio codec for transmit and receive operation.

Syntax

void begin(bool input, bool output);

Parameters

input: enable audio codec data input

output: enable audio codec data output

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::end

Description

Stop all audio codec operation.

Syntax

void end();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::getBufferPageSize

Description

Get the byte size of a single page of the audio codec buffer.

Syntax

uint32_t getBufferPageSize();

Parameters

The function requires no input parameter.

Returns

The size of a audio codec buffer page, in number of bytes.

Example Code

NA

Notes and Warnings

The AudioCodec class includes a transmit and receive buffer to store audio sample data while transferring to and from the DAC output and ADC input. The buffer is divided into pages of fixed size, and audio data can be read and written one page at a time. Depending on the configured bit depth (bits per audio sample) and channel count, a buffer page may contain a different number of audio samples.

AudioCodec::setSampleRate

Description

Configure the audio codec transmit and receive sampling rate.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: desired audio codec sampling rate in Hz. Default value of 48000. Supported values: 8000, 16000, 32000, 44100, 48000, 88200, 96000.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

High sample rates above 48000Hz will require frequent buffer reads and writes to keep up with the large amount of data input and output. If there is insufficient processing time dedicated to this task, audio quality will be degraded.

AudioCodec::setBitDepth

Description

Configure the audio codec transmit and receive bit depth (bits per sample).

Syntax

void setBitDepth(uint8_t bitDepth);

Parameters

bitDepth: desired number of bits per sample. Default value of 16. Supported values: 8, 16, 24.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Setting a bit depth of 24 bits per sample will require 32 bits (4 bytes) of buffer space for storing each sample, with the most significant byte ignored.

AudioCodec::setChannelCount

Description

Configure the audio codec transmit and receive channel count.

Syntax

void setChannelCount(uint8_t monoStereo);

Parameters

monoStereo: number of channels. Default value of 1. Supported values: 1, 2.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setInputMicType

Description

Configure for analog or digital input microphone type.

Syntax

Void setInputMicType(Mic_Type micType);

Parameters

micType: Input microphone type. Default value ANALOGMIC. Valid values:

- ANALOGMIC – microphone with an analog output

- PDMMIC – digital microphone with a PDM output

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

For analog single-ended output, connect to PA_4 for the left channel and PA_2 for the right channel.

For digital PDM output, connect the PDM clock to PB_1 and PDM data to PB_2.

AudioCodec::setInputLRMux

Description

Configure input left right channel multiplexing.

Syntax

void setInputLRMux(uint32_t mux);

Parameters

mux: desired left right audio channel multiplexing setting. Default value RX_CH_LR. Valid values:

- RX_CH_LR
- RX_CH_RL
- RX_CH_LL
- RX_CH_RR

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In mono channel mode, both RX_CH_LR and RX_CH_LL will result in the audio codec sampling input data from the left channel microphone. Similarly, both RX_CH_RL and RX_CH_RR will result in the audio codec sampling input data from the right channel microphone.

In stereo channel mode, RX_CH_RL will switch the positions of input data sampled from the microphones. RX_CH_RR and RX_CH_LL will result in duplicated samples from the right and left microphones respectively.** **

AudioCodec::setDMicBoost

Description

Configure boost gain for digital microphone input.

Syntax

void setDMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel digital microphone input

rightBoost: boost gain for right channel digital microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 12dB
- 2 : 24dB
- 3 : 36dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setAMicBoost

Description

Configure boost gain for analog microphone input.

Syntax

void setAMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel analog microphone input

rightBoost: boost gain for right channel analog microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 20dB
- 2 : 30dB
- 3 : 40dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this function if additional gain is required after using setADCGain function.

AudioCodec::setADCGain

Description

Configure gain of ADC used to acquire analog input.

Syntax

void setADCGain(uint32_t leftGain, uint32_t rightGain);

Parameters

leftGain: Gain for left channel ADC

rightGain: Gain for right channel ADC

Valid value range is from 0x00 to 0x7f. Gain increases by 0.375dB for every increment in value:

- 0x00 : -17.625dB
- 0x01 : -17.25dB
- 0x2f : 0dB
- 0x30 : 0.375dB
- 0x7f : 30dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::muteInput

Description

Mute input audio data stream.

Syntax

void muteInput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel input, 0 to unmute

rightMute: 1 to mute right channel input, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setOutputVolume

Description

Configure output audio volume.

Syntax

void setOutputVolume(uint8_t leftVol, uint8_t rightVol);

Parameters

leftVol: left channel output volume

rightVol: right channel output volume

Valid value ranges from 0 to 100, corresponding to a volume of -65.625dB to 0dB.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::muteOutput

Description

Mute output audio.

Syntax

void muteOutput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel output, 0 to unmute

rightMute: 1 to mute right channel output, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::writeAvaliable

Description

Check for free buffer page available for data write.

Syntax

bool writeAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page that is available for writing data into. Returns false if all buffer pages are full.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::writeDataPage

Description

Write audio data to an available buffer page.

Syntax

uint32_t writeDataPage(int8_t* src, uint32_t len);

uint32_t writeDataPage(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing audio samples to write to audio codec.

len: number of audio samples in array.

Returns

The function returns the number of audio samples written to the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readAvaliable

Description

Check for buffer page with new data available for read.

Syntax

bool readAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page with new data that is ready for reading data from. Returns false if all buffer pages are empty.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readDataPage

Description

Read audio data from a ready buffer page.

Syntax

uint32_t readDataPage(int8_t* dst, uint32_t len);

uint32_t readDataPage(int16_t* dst, uint32_t len);

Parameters

dst: pointer to array to contain audio samples read from audio codec.

len: number of audio samples to read.

Returns

The function returns the number of audio samples read from the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setWriteCallback

Description

Set a callback function to be notified when a free buffer page is available for write.

Syntax

void setWriteCallback(**void** (*writeCB)(**void*));

Parameters

writeCB: function to be called when a buffer page becomes available for data write. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec finishes outputting the data in a buffer page.

AudioCodec::setReadCallback

Description

Set a callback function to be notified when a buffer page with new data is available for read.

Syntax

void setReadCallback(**void** (*readCB)(**void*));

Parameters

readCB: function to be called when a buffer page with new data becomes available for data read. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec fills up a buffer page with newly acquired audio samples.

Class FFT

Description

A class used for performing FFT calculations with real-number inputs and outputs.

Syntax

class FFT

Members

Public Constructors

FFT::FFT	Create an instance of the FFT class

Public Methods

FFT::setWindow	Configure the window function used in FFT calculations
FFT::calculate	Calculate FFT for an input array of values
FFT::getFrequencyBins	Get the FFT output frequency bins
FFT::getFFTSize	Get the size of FFT output for a given input size

FFT::FFT

Description

Create a FFT class object.

Syntax

void FFT();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

FFT::setWindow

Description

Configure the window function used in FFT calculations.

Syntax

void setWindow(FFTWindow_t window, uint16_t sampleCount);

Parameters

window: The window function to be used in FFT calculations. Valid values: None, Hann, Hamming.

sampleCount: Number of sample datapoints in the input.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

The window function is used to reduce the effects of discontinuities that occur when the input signal has frequencies that do not fit an integer number of periods in the sample datapoints.

More information on FFTs and window functions can be seen at:

https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf

https://en.wikipedia.org/wiki/Window_function

FFT::Calculate

Description

Calculate FFT for an input array of values.

Syntax

void calculate(float* inputBuf, float* outputBuf, uint16_t sampleCount);

void calculate(int16_t* inputBuf, float* outputBuf, uint16_t sampleCount);

Parameters

inputBuf: pointer to an array of sampleCount size, containing input sample datapoints, in float or uint16_t format.

outputBuf: pointer to a float array of sampleCount/2 size, for containing FFT output.

sampleCount: number of sample datapoints in the input array, valid values: 16, 32, 64, 128, 256, 512, 1024, 2048.

Returns

The function returns nothing.

Example Code

Example:FFT

Notes and Warnings

Large sample counts will require a longer time for FFT calculations, but will also return a result with higher frequency resolution.

FFT::getFrequencyBins

Description

Get the FFT output frequency bins.

Syntax

void getFrequencyBins(uint16_t* outputBuf, uint16_t sampleCount, uint32_t sampleRate);

Parameters

outputBuf: pointer to a uint16_t array of sampleCount/2 size, for containing the calculated center frequency of each FFT output element.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings NA

—

FFT::getFFTSize

Description

Get the size of FFT output for a given input size.

Syntax

uint16_t getFFTSize(uint16_t sampleCount);

Parameters

sampleCount: number of input sample datapoints.

Returns

The function returns the FFT output size for the given sampleCount, which is sampleCount/2.

Example Code

NA

Notes and Warnings NA

Class PlaybackWav

Description

A class used for control and playback of .wav file format audio data.

Syntax

class PlaybackWav

Members

Public Constructors

PlaybackWav::PlaybackWav	Create an instance of the PlaybackWav class

Public Methods

PlaybackWav::openFile	Open a .wav file for playback
PlaybackWav::closeFile	Close a previously opened file
PlaybackWav::fileOpened	Check if a .wav file is already opened
PlaybackWav::getSampleRate	Get the sample rate of the .wav file
PlaybackWav::getChannelCount	Get the number of audio channels in the .wav file
PlaybackWav::getBitDepth	Get the bit depth of each sample in the .wav file
PlaybackWav::getLengthMillis	Get the playback length of the .wav file in milliseconds
PlaybackWav::getPositionMillis	Get the current playback position in milliseconds
PlaybackWav::setPositionMillis	Set the current playback position in milliseconds
PlaybackWav::millisToBytes	Convert a playback duration to equivalent number of bytes
PlaybackWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
PlaybackWav::readAudioData	Read audio data from the .wav file

---

PlaybackWav::PlaybackWav

Description

Create a PlaybackWav class object.

Syntax

void PlaybackWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordPlaybackWav

Notes and Warnings

NA

PlaybackWav::getSampleRate

Description

Get the sample rate of the .wav file.

Syntax

uint32_t getSampleRate(void);

Parameters

The function requires no input parameter.

Returns

The function returns sampling rate encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getChannelCount

Description

Get the number of audio channels in the .wav file.

Syntax

uint16_t getChannelCount(void);

Parameters

The function requires no input parameter.

Returns

The function returns channel count encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getBitDepth

Description

Get the bit depth of each sample in the .wav file.

Syntax

uint16_t getBitDepth(void);

Parameters

The function requires no input parameter.

Returns

The function returns bit depth encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getLengthMillis

Description

Get the playback length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the total playback length of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getPositionMillis

Description

Get the current playback position in milliseconds.

Syntax

uint32_t getPositionMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current playback position of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::setPositionMillis

Description

Set the current playback position in milliseconds.

Syntax

void setPositionMillis(uint32_t pos);

Parameters

pos: The desired playback position expressed in milliseconds.

Returns

The function returns nothing.

Example Code

Example: PlaybackWavFile

Notes and Warnings

Any changes to playback position will only take effect on the next call to PlaybackWav::readAudioData. If the desired playback position is beyond the total playback length of the file, the playback position will be set to the end of file, and no audio data will be output on subsequent data reads.

PlaybackWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::readAudioData

Description

Read audio data from the .wav file.

Syntax

- `uint32_t readAudioData(int8_t* dst, uint32_t len);`
- `uint32_t readAudioData(int16_t* dst, uint32_t len);`

Parameters

- `dst`: pointer to array to store data read from .wav file.
- `len`: number of audio samples to read from .wav file.

Returns

The function returns number of audio samples read.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

Class RecordWav

Description

A class used for control and recording of .wav file format audio data.

Syntax

class RecordWav

Members

Public Constructors

RecordWav:: RecordWav	Create an instance of the RecordWav class

Public Methods

RecordWav::openFile	Open a .wav file for playback
RecordWav::closeFile	Close a previously opened file
RecordWav::fileOpened	Check if a .wav file is already opened
RecordWav::setSampleRate	Get the sample rate of the .wav file
RecordWav::setChannelCount	Set the number of audio channels in the .wav file
RecordWav::setBitDepth	Set the bit depth of each sample in the .wav file
RecordWav::getLengthMillis	Get the current record length of the .wav file in milliseconds
RecordWav::millisToBytes	Convert a playback duration to equivalent number of bytes
RecordWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
RecordWav::writeAudioData	Write audio data to the .wav file

RecordWav::RecordWav

Description

Create a RecordWav class object.

Syntax

void RecordWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::openFile

Description

Open a .wav file for recording.

Syntax

void openFile(const char* absFilepath);

Parameters

absFilepath: the filepath of the .wav file to open.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::closeFile

Description

Close a previously opened file.

Syntax

void closeFile(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

Any open .wav files should be closed after recording is complete, otherwise, loss of recorded audio data may occur.

RecordWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setSampleRate

Description

Set the recording sample rate of the .wav file.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: The desired recording sample rate.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setChannelCount

Description

Set the number of recording audio channels in the .wav file.

Syntax

void setChannelCount(uint16_t channelCount);

Parameters

channelCount: number of recording audio channels.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::setBitDepth

Description

Set the recording bit depth of each sample in the .wav file.

Syntax

void setBitDepth(uint16_t bitDepth);

Parameters

bitDepth: number of bits per sample.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::getLengthMillis

Description

Get the current recorded length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current recorded length of the currently open .wav file in milliseconds.

Example Code

NA

Notes and Warnings

NA

RecordWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::writeAudioData

Description

Write audio data to the .wav file.

Syntax

uint32_t writeAudioData(int8_t* src, uint32_t len); uint32_t writeAudioData(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing data to write to .wav file. len: number of audio samples to write to .wav file.

Returns

The function returns number of audio samples written.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

BLE

Class BLEAddr

BLEAddr Class

Description

A class used for managing Bluetooth addresses.

Syntax
class BLEAddr

Members

Public Constructors	
BLEAddr::BLEAddr	Constructs a BLEAddr object
Public Methods	
BLEAddr::str	Get the Bluetooth address represented as a formatted string
BLEAddr::data	Get the Bluetooth address represented as an integer array

BLEAddr::BLEAddr

Description
Constructs a BLEAddr object.

Syntax
BLEAddr::BLEAddr(void);
BLEAddr::BLEAddr(uint8_t (&addr)[6]);
BLEAddr::BLEAddr(const char* str);

Parameters
addr: An array of 6 bytes containing the desired Bluetooth address.
str: A character string representing the desired Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
When expressed as a string, the Bluetooth address should be written as 6 bytes in hexadecimal format, using a colon ":" to separate the bytes is acceptable (example – 00:11:22:33:EE:FF).

BLEAddr::str

Description
Get the Bluetooth address represented as a formatted string.

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a character string containing the hexadecimal representation of the Bluetooth address.

Example Code
Example: BLEScan

Notes and Warnings
The Bluetooth address expressed as a string will be written as 6 bytes in hexadecimal format, with a colon ":" separating the bytes (example – 00:11:22:33:EE:FF).

BLEAddr::data

Description
Get the Bluetooth address represented as an integer array.

Syntax
uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a 6 byte array containing the Bluetooth address.

Example Code
NA

Notes and Warnings

The Bluetooth address is stored with MSB at array index [5].

Class BLEAdvert

BLEAdvert Class

Description
A class used for managing BLE advertising settings.

Syntax
class BLEAdvert

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configAdvert().

Public Methods	
BLEAdvert::updateAdvertParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEAdvert::startAdv	Start BLE advertising
BLEAdvert::stopAdv	Stop BLE advertising
BLEAdvert::setAdvType	Set the BLE advertising type
BLEAdvert::setMinInterval	Set the BLE advertising minimum interval
BLEAdvert::setMaxInterval	Set the BLE advertising maximum interval
BLEAdvert::setAdvData	Set BLE advertising data
BLEAdvert::setScanRspData	Set BLE scan response data

BLEAdvert::updateAdvertParams

Description
Update the lower Bluetooth stack with the current advertising settings.

Syntax
void updateAdvertParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Please use the other class member functions to set the BLE advertising parameters first before using this function.

BLEAdvert::startAdv

Description
Start BLE advertising.

Syntax
void startAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly start the BLE advertising process without first registering the necessary callback and handler functions. Call BLEDevice::beginPeripheral() to register the necessary functions and start advertising for the first time.

BLEAdvert::stopAdv

Description
Stop BLE advertising.

Syntax
void stopAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly stop the BLE advertising process. Call BLEDevice::end() to stop advertising and free up used resources.

BLEAdvert::setAdvType

Description
Set the BLE advertising type.

Syntax
void setAdvType(uint8_t advType);

Parameters
advType: the desired advertisement type. Valid values:
– 0 = GAP_ADTYPE_ADV_IND : connectable undirected advertisement
– 1 = GAP_ADTYPE_ADV_HDC_DIRECT_IND : connectable high duty cycle directed
– 2 = GAP_ADTYPE_ADV_SCAN_IND : scannable undirected advertisement
– 3 = GAP_ADTYPE_ADV_NONCONN_IND : Non-connectable undirected advertisement
– 4 = GAP_ADTYPE_ADV_LDC_DIRECT_IND : connectable low duty cycle directed advertisement

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this function with the GAP_ADTYPE_ADV_IND argument if connection requests should be allowed, and GAP_ADTYPE_ADV_NONCONN_IND if all connection requests should be rejected.

BLEAdvert::setMinInterval

Description
Set the minimum BLE advertising interval.

Syntax
void setMinInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement minimum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setMaxInterval

Description
Set the maximum BLE advertising interval.

Syntax
void setMaxInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement maximum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setAdvData

Description
Set BLE advertising data.

Syntax
void setAdvData(BLEAdvertData adData);
void setAdvData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.
size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

BLEAdvert::setScanRspData

Description
Set BLE scan response data.

Syntax
void setScanRspData(BLEAdvertData adData);
void setScanRspData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.

size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

Class BLEAdvertData

BLEAdvertData Class

Description
A class used for managing BLE advertising data.

Syntax
class BLEAdvertData

Members

Public Constructors	
BLEAdvertData::BLEAdvertData	Constructs a BLEAdvertData object

Public Methods	
BLEAdvertData::clear	Clear all advertising data
BLEAdvertData::addData	Add binary advertising data
BLEAdvertData::addFlags	Add flags to advertising data
B LEAdvertData::addPartialServices	Add partial services to advertising data
BL EAdvertData::addCompleteServices	Add complete services to advertising data
BLEAdvertData::addAppearance	Add device appearance to advertising data
BLEAdvertData::addShortName	Add short device name to advertising data
BLEAdvertData::addCompleteName	Add complete device name to advertising data
BLEAdvertData::parseScanInfo	Parse advertising data received from a scan
BLEAdvertData::hasFlags	Check if received data includes advertising flags
BLEAdvertData::hasUUID	Check if received data includes UUIDs
BLEAdvertData::hasName	Check if received data includes device name
BLEAdvertData::hasManufacturer	Check if received data includes manufacturer data
BLEAdvertData::getAdvType	Get advertising type of received data
BLEAdvertData::getAddrType	Get Bluetooth address type of received data
BLEAdvertData::getAddr	Get Bluetooth address of received data
BLEAdvertData::getRSSI	Get RSSI of received data
BLEAdvertData::getFlags	Get advertising flags of received data
BLEAdvertData::getServiceCount	Get number of advertised services in received data
BLEAdvertData::getServiceList	Get array of advertised services in received data
BLEAdvertData::getName	Get advertised device name in received data
BLEAdvertData::getTxPower	Get advertised transmission power in received data
BLEAdvertData::getAppearance	Get advertised device appearance in received data
BLEAdvertData::getManufacturer	Get advertised manufacturer in received data
BLEAdver tData::getManufacturerDataLength	Get length of manufacturer data in received data
BL EAdvertData::getManufacturerData	Get advertised manufacturer data in received data

BLEAdvertData::BLEAdvertData

Description
Constructs a BLEAdvertData object.

Syntax
BLEAdvertData::BLEAdvertData(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class is used for managing BLE advertising data for two primary uses. First is to assemble advertising data for broadcasting as advertising packets. Second is to process and split up the advertising data received from a scan into separate types.

BLEAdvertData::clear

Description
Clear all advertising data currently saved in class object.

Syntax
void clear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addData

Description
Add binary advertising data.

Syntax
void addData(const uint8_t* data, uint8_t size);

Parameters
data: pointer to array containing desired advertising data.
size: number of bytes in array.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in adding BLE advertising data. Other functions should be used for adding advertising data if possible, as this function does not perform any checks on the validity of the data.

BLEAdvertData::addFlags

Description
Add flags to advertising data.

Syntax
uint8_t addFlags(uint8_t flags);

Parameters
flags: desired flags to add to advertising data. Valid values:
– GAP_ADTYPE_FLAGS_LIMITED
– GAP_ADTYPE_FLAGS_GENERAL
– GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_CONTROLLER
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_HOST

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::addPartialServices

Description
Add partial list of service UUIDs to advertising data.

Syntax
uint8_t addPartialServices(BLEUUID uuid);

Parameters
uuid: the desired UUID contained in BLEUUID class object.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addCompleteServices

Description
Add complete list of service UUIDs to advertising data.

Syntax
uint8_t addCompleteServices(BLEUUID uuid);
uint8_t addCompleteServices(uint8_t uuidBitLength);

Parameters
uuid: the desired UUID contained in BLEUUID class object.
uuidBitLength: UUID bit length for which a blank entry is to be added. Valid values: 16, 32, 128.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
uuidBitLength is used when it is desired to add a blank entry to the advertisement data, used to indicate that no services with UUIDs of a certain length are available.

BLEAdvertData::addAppearance

Description
Add device appearance to advertising data.

Syntax
uint8_t addAppearance(uint16_t appearance);

Parameters
appearance: the desired device appearance.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for a full list of device appearance values.

BLEAdvertData::addShortName

Description
Add shortened device name to advertising data.

Syntax
uint8_t addShortName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings

NA

BLEAdvertData::addCompleteName

Description
Add complete device name to advertising data.

Syntax
uint8_t addCompleteName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::parseScanInfo

Description
Parse advertising data received from a scan.

Syntax
void parseScanInfo(T_LE_CB_DATA *p_data);

Parameters
p_data: pointer to advertising data received from a Bluetooth scan.

Returns
The function returns nothing.

Example Code

Example: BLEBatteryClient

Notes and Warnings
Advertising data fields of parsed receive data can be access using member functions starting with “has” and “get”.

BLEAdvertData::hasFlags

Description
Check if received data includes advertising flags.

Syntax
bool hasFlags(void);

Parameters
The function requires no input parameter.

Returns
True if flags are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasUUID

Description
Check if received data includes service UUIDs.

Syntax
bool hasUUID(void);

Parameters
The function requires no input parameter.

Returns

True if service UUIDs are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasName

Description
Check if received data includes device name.

Syntax
bool hasName(void);

Parameters
The function requires no input parameter.

Returns
True if device name is present in received advertising data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::hasManufacturer

Description
Check if received data includes manufacturer specific data.

Syntax
bool hasManufacturer(void);

Parameters

The function requires no input parameter.

Returns
True if manufacturer specific data is present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAdvType

Description
Get advertising type of received data.

Syntax
T_GAP_ADV_EVT_TYPE getAdvType(void);

Parameters
The function requires no input parameter.

Returns
Advertising type of received advertising data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_ADV_EVT_TYPE_UNDIRECTED
– GAP_ADV_EVT_TYPE_DIRECTED
– GAP_ADV_EVT_TYPE_SCANNABLE
– GAP_ADV_EVT_TYPE_NON_CONNECTABEL
– GAP_ADV_EVT_TYPE_SCAN_RSP

BLEAdvertData::getAddrType

Description
Get Bluetooth address type of received data.

Syntax
T_GAP_REMOTE_ADDR_TYPE getAddrType(void);

Parameters
The function requires no input parameter.

Returns
Bluetooth address type of received data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

BLEAdvertData::getRSSI

Description
Get received signal strength indicator (RSSI) of received data.

Syntax
Int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
Received signal strength.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getFlags

Description
Get advertising flags of received data.

Syntax
uint8_t getFlags(void);

Parameters
The function requires no input parameter.

Returns
Advertising flags present in received advertising data, expressed as a single byte.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getServiceCount

Description
Get number of advertised services in received data.

Syntax
uint8_t getServiceCount(void);

Parameters
The function requires no input parameter.

Returns
Number of advertised service UUIDs in received data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getServiceList

Description
Get list of advertised service UUIDs in received data.

Syntax
BLEUUID* getServiceList(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a BLEUUID array containing all advertised service UUIDs.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getName

Description
Get advertised device name in received data.

Syntax
String getName(void);

Parameters
The function requires no input parameter.

Returns
Advertised device name contained in a String class object.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getTxPower

Description
Get advertised transmission power in received data.

Syntax
int8_t getTxPower(void);

Parameters
The function requires no input parameter.

Returns
Advertised transmission power.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAppearance

Description
Get advertised device appearance in received data.

Syntax
uint16_t getAppearance(void);

Parameters
The function requires no input parameter.

Returns
Advertised device appearance.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of device appearance values.

BLEAdvertData::getManufacturer

Description
Get advertised manufacturer in received data.

Syntax
uint16_t getManufacturer(void);

Parameters
The function requires no input parameter.

Returns
Advertised manufacturer.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of manufacturer codes.

BLEAdvertData::getManufacturerDataLength

Description
Get length of manufacturer data in received data.

Syntax
uint8_t getManufacturerDataLength(void);

Parameters
The function requires no input parameter.

Returns
Number of bytes of manufacturer data present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getManufacturerData

Description
Get manufacturer data in received data.

Syntax
uint8_t* getManufacturerData(void);

Parameters
The function requires no input parameter.

Returns
Pointer to array containing manufacturer data.

Example Code
NA

Notes and Warnings
NA

Class BLEBeacon

iBeacon Class

Description
A class used for managing iBeacon BLE advertising data.

Syntax
class iBeacon

Members

Public Constructors	
iBeacon::iBeacon	Create an instance of iBeacon advertising data
Public Methods	
iBeacon::getManufacturerId	Get current manufacturer ID value
iBeacon::getUUID	Get current UUID value
iBeacon::getMajor	Get current Major value
iBeacon::getMinor	Get current Minor value
iBeacon::getRSSI	Get current RSSI value
iBeacon::setManufacturerId	Set manufacturer ID value
iBeacon::setUUID	Set UUID value
iBeacon::setMajor	Set Major value
iBeacon::setMinor	Set Minor value
iBeacon::setRSSI	Set RSSI value
iBeacon::getAdvData	Get current advertising data
iBeacon::getScanRsp	Get current scan response data

altBeacon Class

Description
A class used for managing altBeacon BLE advertising data.

Syntax
class altBeacon

Members

Public Constructors	
altBeacon::altBeacon	Create an instance of altBeacon advertising data
Public Methods	
altBeacon::getManufacturerId	Get current manufacturer ID value
altBeacon::getUUID	Get current UUID value
altBeacon::getMajor	Get current Major value
altBeacon::getMinor	Get current Minor value
altBeacon::getRSSI	Get current RSSI value
altBeacon::getRSVD	Get current Reserved value
altBeacon::setManufacturerId	Set manufacturer ID value
altBeacon::setUUID	Set UUID value
altBeacon::setMajor	Set Major value
altBeacon::setMinor	Set Minor value
altBeacon::setRSSI	Set RSSI value
altBeacon::setRSVD	Set Reserved value
altBeacon::getAdvData	Get current advertising data
altBeacon::getScanRsp	Get current scan response data

iBeacon::iBeacon

Description
Create an iBeacon object.

Syntax
void iBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

altBeacon::altBeacon

Description
Create an altBeacon object.

Syntax
void altBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

iBeacon::getManufacturerId

altBeacon::getManufacturerId

Description
Get current Manufacturer ID value.

Syntax
uint16_t getManufacturerId(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Company ID.

Example Code
NA

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::getUUID

altBeacon::getUUID

Description
Get the current UUID value.

Syntax
void getUUID(uint8_t* UUID);

Parameters
UUID: pointer to a 16 element uint8_t array, current UUID will be copied into the array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::getMajor

altBeacon::getMajor

Description
Get current Major value.

Syntax
uint16_t getMajor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Major value.

Example Code

NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getMinor

altBeacon::getMinor

Description
Get current Minor value.

Syntax
uint16_t getMinor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Minor value.

Example Code
NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getRSSI

altBeacon::getRSSI

Description
Get the current RSSI value.

Syntax
int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit signed integer containing the currently set RSSI value.

Example Code
NA

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setManufacturerId

altBeacon::setManufacturerId

Description
Set Manufacturer ID value.

Syntax
void setManufacturerId(uint16_t id);

Parameters
id: desired Manufacturer ID

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::setUUID

altBeacon::setUUID

Description
Set UUID value.

Syntax
void setUUID(uint8_t* UUID);
void setUUID(const char* UUID);

Parameters
uint8_t* UUID: pointer to a 16 element uint8_t array containing the desired UUID
const char* UUID: desired UUID expressed as a character string

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::setMajor

altBeacon::setMajor

Description
Set Major value.

Syntax
void setMajor(uint16_t major);

Parameters
major: desired Major value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setMinor

altBeacon::setMinor

Description
Set Minor value.

Syntax
void setMinor(uint16_t minor);

Parameters
minor: desired Minor value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setRSSI

altBeacon::setRSSI

Description
Set RSSI value.

Syntax
void setRSSI(int8_t RSSI);

Parameters
RSSI: desired RSSI value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getAdvData

altBeacon::getAdvData

Description
Get current beacon advertising data.

Syntax
uint8_t* getAdvData(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

iBeacon::getScanRsp

altBeacon::getScanRsp

Description
Get current beacon advertising scan response data.

Syntax
uint8_t* getScanRsp(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising scan response data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

altBeacon::getRSVD

Description
Get current Reserved value.

Syntax
uint8_t getRSVD(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit unsigned integer containing the current Reserved value.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

altBeacon::setRSVD

Description
Set Reserved value.

Syntax
void setRSVD(uint8_t rsvd);

Parameters
rsvd: desired Reserved value

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

Class BLECharacteristic

BLECharacteristic Class

Description

A class used for creating and managing BLE GATT characteristics.

Syntax
class BLECharacteristic

Members

Public Constructors	
BLEC haracteristic::BLECharacteristic	Constructs a BLECharacteristic object
Public Methods	
BLECharacteristic::setUUID	Set the characteristic UUID
BLECharacteristic::getUUID	Get the characteristic UUID
BLECharacteristic::setBufferLen	Set the size of the internal data buffer
BLECharacteristic::getBufferLen	Get the current size of the internal data buffer
BL ECharacteristic::setReadProperty	Get the current size of the internal data bufferSet the characteristic r
BLE Characteristic::setWriteProperty	Set the characteristic write property
BLEC haracteristic::setNotifyProperty	Set the characteristic notify property
BLECha racteristic::setIndicateProperty	Set the characteristic indicate property

continues o

Table 3 – continued from previous page

Public Constructors	
BLECharacteristic::setProperties	Set the characteristic properties
BLECharacteristic::getProperties	Get the characteristic properties
BLECharacteristic::readString	Read the characteristic data buffer as a String object
BLECharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLECharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLECharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLECharacteristic::writeString	Write data to the characteristic data buffer as a String object or chara
BLECharacteristic::writeData8	Write data to the characteristic data buffer as an unsigned 8-bit integ
BLECharacteristic::writeData16	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::writeData32	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::setData	Write data to the characteristic data buffer
BLECharacteristic::getData	Read data from the characteristic data buffer
BLECharacteristic::getDataBuff	Get a pointer to the characteristic data buffer
BLECharacteristic::getDataLen	Get the number of bytes of data in the characteristic data buffer
BLECharacteristic::notify	Send a notification to a connected device
BLECharacteristic::indicate	Send an indication to a connected device
BLEC haracteristic::setUserDescriptor	Add a user description descriptor to characteristic
BLECha racteristic::setFormatDescriptor	Add a data format descriptor to characteristic
BLECharacteristic::Add a data format descriptor to characteristic	Set a user function as a read callback
BLE Characteristic::setWriteCallback	Set a user function as a write callback
BL ECharacteristic::setCCCDCallback	Set a user function as a CCCD write callback

BLECharacteristic::BLECharacteristic

Description
Constructs a BLECharacteristic object.

Syntax
BLECharacteristic::BLECharacteristic(BLEUUID uuid);
BLECharacteristic::BLECharacteristic(const char* uuid);

Parameters
uuid: characteristic UUID, expressed as a BLEUUID class object or a character array

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setUUID

Description
Set the characteristic UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: the new characteristic UUID, expressed with a BLEUUID class object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLECharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadProperty

Description
Set the characteristic read property.

Syntax
void setReadProperty(bool value);

Parameters
value: TRUE to allow connected devices to read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteProperty

Description
Set the characteristic write property.

Syntax
void setWriteProperty(bool value);

Parameters
value: TRUE to allow connected devices to write characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setNotifyProperty

Description
Set the characteristic notify property.

Syntax
void setNotifyProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data notifications.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setIndicateProperty

Description
Set the characteristic indicate property.

Syntax
void setIndicateProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data indications.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setProperties

Description
Set the characteristic properties.

Syntax
void setProperties(uint8_t value);

Parameters
value: desired characteristic properties

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getProperties

Description
Get the currently set characteristic properties.

Syntax
uint8_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set characteristic properties expressed as an unsigned 8-bit integer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readString

Description
Read the data in the characteristic internal buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a String class object.

Example Code
Example: BLEUartService

Notes and Warnings
Non-ASCII data may result in unexpected characters in the string.

BLECharacteristic::readData8

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData16

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Write data to the characteristic data buffer as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the characteristic buffer, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::writeData8

Description
Write data to the characteristic data buffer as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEBatteryService

Notes and Warnings

NA

BLECharacteristic::writeData16

Description
Write data to the characteristic data buffer as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::writeData32

Description
Write data to the characteristic data buffer as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setData

Description
Write data to the characteristic data buffer.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getData

Description
Read data from the characteristic data buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLECharacteristic::getDataBuff

Description
Get a pointer to the characteristic data buffer.

Syntax
uint8_t* getDataBuff();

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to the uint8_t array used as the characteristic internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getDataLen

Description
Get the number of bytes of data in the characteristic data buffer.

Syntax
uint16_t getDataLen

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes of data in the internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::notify

Description
Send a notification to a connected device.

Syntax
void notify(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send a notification to.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::indicate

Description
Send an indication to a connected device.

Syntax
void indicate(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send an indication to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setUserDescriptor

Description
Add a user description descriptor attribute (UUID 0x2901) to the characteristic.

Syntax
void setUserDescriptor(const char* description);

Parameters
description: the desired user description string expressed in a char array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setFormatDescriptor

Description
Add a data format descriptor attribute (UUID 0x2904) to the characteristic.

Syntax
void setFormatDescriptor(uint8_t format, uint8_t exponent, uint16_t unit, uint16_t description);

Parameters
format: refer to https://www.bluetooth.com/specifications/assigned-numbers/format-types/ for the valid values and associated format types.
exponent: base-10 exponent to be applied to characteristic data value.

unit: refer to https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf for the valid values and associated units.
descriptor: refer to https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-descriptors/ for the valid values and associated descriptors.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadCallback

Description
Set a user function to be called when the characteristic data is read by a connected device.

Syntax
void setReadCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing data read
conn_id: connection ID of connected device that read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteCallback

Description

Set a user function to be called when the characteristic data is written by a connected device.

Syntax
void setWriteCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setCCCDCallback

Description
Set a user function to be called when a connected device modifies the characteristic CCCD to enable or disable notifications or indications.

Syntax
void setCCCDCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id, uint16_t ccc_bits));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.
ccc_bits: the new CCCD data bits after modification by the connected device

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

Class BLEClient

BLEClient Class

Description
A class used for discovering and accessing BLE GATT services on a connected remote device.

Syntax
class BLEClient

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLEDevice::addClient().

Public Methods	
BLEClient::connected	Check if the corresponding remote device for the client is connected
BLEClient::discoverServices	Start service discovery process for connected device
BLEClient::discoveryDone	Determine if service discovery process has been completed
BLEClient::printServices	Format and print discovered services to serial port
BLEClient::getService	Get a specific service on the remote device
BLEClient::getConnId	
BLEClient::getClientId	Get corresponding client ID
BLEClient::setDisconnectCallback	Set a user function to be called when the remote device is disconnected

BLEClient::connected

Description
Check if the remote device associated with the client is still connected.

Syntax
bool connected();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the remote device is connected.

Example Code
NA

Notes and Warnings
NA

BLEClient::discoverServices

Description
Start the service discovery process for the connected remote device.

Syntax
void discoverServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::discoveryDone

Description
Check if the service discovery process has been completed.

Syntax
bool discoveryDone();

Parameters
The function requires no input parameter.

Returns
TThe function returns TRUE if the service discovery process has been completed successfully, FALSE if the service discovery process failed, is still in progress, or has yet to start.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::printServices

Description
Print out a formatted list of discovered services to the serial port.

Syntax
void printServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEClient::getService

Description
Get a service with the specified UUID on the remote device.

Syntax
BLERemoteService* getService(const char* uuid);
BLERemoteService* getService(BLEUUID uuid);

Parameters
uuid: the desired service UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found service as a BLERemoteService object pointer, otherwise nullptr is returned if a service with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::getConnId

Description
Get the connection ID associated with the remote device.

Syntax
uint8_t getConnId;

Parameters
The function requires no input parameter.

Returns
The function returns the connection ID for the connected remote device.

Example Code
NA

Notes and Warnings
NA

BLEClient::getClientId

Description
Get the client ID for the BLEClient object.

Syntax
T_CLIENT_ID getClientId();;

Parameters
The function requires no input parameter.

Returns
The function returns the BLEClient object's client ID.

Example Code
NA

Notes and Warnings
The client ID is used when calling internal GATT client API.

BLEClient::setDisconnectCallback

Description
Set a user function as a callback function when the remote device is disconnected.

Syntax
void setDisconnectCallback(void (*fCallback) (BLEClient* client));

Parameters
fCallback: A user callback function that returns void and takes one argument.
client: A pointer to the BLEClient object corresponding to the disconnected remote device

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The user callback function will be called after the remote device has disconnected, before the characteristics, services and client associated with the remote device are deleted.

Class BLEConnect

BLEConnect Class

Description
A class used for managing BLE connection settings.

Syntax
class BLEConnect

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configConnection.

Public Methods	
BLEConnect::connect	Connect to a target BLE device
BLEConnect::disconnect	Disconnect from a target BLE device
BLEConnect::setScanInterval	Set the BLE scanning interval when connecting
BLEConnect::setScanWindow	Set the BLE scanning window when connecting
BLEConnect::setConnInterval	Set the BLE connection interval duration
BLEConnect::setConnLatency	Set the BLE connection slave latency
BLEConnect::setConnTimeout	Set the BLE connection timeout value
BLEConnect::updateConnParams	Send new BLE connection parameters to a connected device
BLEConnect::getConnInfo	Get connection information
BLEConnect::getConnAddr	Get the Bluetooth address for a certain connection
BLEConnect::getConnId	Get the connection ID for a certain device

BLEConnect::connect

Description
Connect to a target BLE device.

Syntax
bool connect(char* btAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(uint8_t (&btAddr)[6], T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(BLEAdvertData targetDevice, uint16_t scanTimeout);
bool connect(BLEAddr destAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t (&btAddr): target device Bluetooth address contained in a 6 byte array.
destAddr: target device Bluetooth address contained in BLEAddr class object.

targetDevice: advertising data packet scanned from target device.
destAddrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM
scan timeout: duration in milliseconds for which to look for target device before giving up.

Returns
True if connection successful, false if connection failed.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEConnect::disconnect

Description
Disconnect from a target BLE device.

Syntax
bool disconnect(uint8_t connId);

Parameters
connId: connection ID for target device.

Returns
True if operation successful, false if otherwise.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanInterval

Description

Set the BLE scan interval when searching for a target device to connect to.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanWindow

Description
Set the BLE scan window when searching for a target device to connect to.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setConnInterval

Description
Set the BLE connection interval value.

Syntax
void setConnInterval(uint16_t min_ms, uint16_t max_ms);

Parameters
min_ms: minimum acceptable connection interval in milliseconds. Value range of 8 to 4000.
max_ms: maximum acceptable connection interval in milliseconds. Value range of 8 to 4000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection interval defines the period between successive connection events between a connected central and peripheral device. Even if there is no data to exchange, a connection event is required to maintain the connection. max_ms should be larger than or equal to min_ms.

BLEConnect::setConnLatency

Description
Set the BLE connection slave latency value.

Syntax
void setConnLatency(uint16_t latency);

Parameters
latency: Connection slave latency value. Value range of 0 to 499.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection slave latency defines the number of successive connection events a connected peripheral device can ignore without being considered as disconnected by the central device.

BLEConnect::setConnTimeout

Description
Set the BLE connection timeout value.

Syntax
void setConnTimeout(uint16_t timeout_ms);

Parameters
timeout_ms: connection timeout in milliseconds. Value range of 100 to 32000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection timeout defines the duration after a failed connection events before a peripheral or central device considers the connection broken.

BLEConnect::updateConnParams

Description
Update a connected device with new connection parameters.

Syntax
void updateConnParams(uint8_t conn_id);

Parameters
conn_id: connection ID of target device to update connection parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Update a connected device with previously set connection interval, slave latency and timeout values. The connected device may reject the new values if it is unable to conform to them.

BLEConnect::getConnInfo

Description
Get connection information.

Syntax
bool getConnInfo(uint8_t connId, T_GAP_CONN_INFO *pConnInfo);

Parameters
connId: connection ID to get connection information from.
pConnInfo: pointer to T_GAP_CONN_INFO structure to store obtained connection information.

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnAddr

Description
Get the Bluetooth address for a certain connection.

Syntax
bool getConnAddr(uint8_t connId, uint8_t* addr, uint8_t* addrType);

Parameters
connId: connection ID to get address information for

addr: pointer to 6 byte array to store retrieved Bluetooth address
addrType: pointer to uint8_t variable to store retrieved Bluetooth address type

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnId

Description
Get the connection ID for a certain device.

Syntax
int8_t getConnId(char* btAddr, uint8_t addrType);
int8_t getConnId(uint8_t* btAddr, uint8_t addrType);
int8_t getConnId(BLEAdvertData targetDevice);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t* btAddr: pointer to a 6 byte array containing target device Bluetooth address.
targetDevice: advertising data packet scanned from target device.
addrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

Returns
The function returns the requested connection ID. Returns -1 if failed to obtain connection ID.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

Class BLEDevice

BLEDevice Class

Description
A class used for general control and management of BLE functions.

Syntax
class BLEDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLE.

Public Methods	
BLEDevice::init	Allocate resources required for BLE functionality
BLEDevice::deinit	Free resources used by BLE functionality
BLEDevice::connected	Check if a BLE device is connected
BLEDevice::setDeviceName	Set BLE GAP device name
BLEDevice::setDeviceAppearance	Set BLE GAP device appearance
BLEDevice::configAdvert	Configure BLE advertising parameters
BLEDevice::configScan	Configure BLE scan parameters
BLEDevice::setScanCallback	Set callback function for BLE scans
BLEDevice::beginCentral	Start BLE stack in central mode
BLEDevice::beginPeripheral	Start BLE stack in peripheral mode
BLEDevice::end	Stop BLE stack
BLEDevice::configServer	Configure BLE stack for services
BLEDevice::addService	Add a service to the BLE stack
BLEDevice::configClient	Configure BLE stack for clients
BLEDevice::addClient	Add a client to the BLE stack
BLEDevice::getLocalAddr	Get local device Bluetooth address

BLEDevice::init

Description
Allocate resources required for BLE functionality.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this member function first before using any other member functions in the BLEDevice class.

BLEDevice::deinit

Description
Free up resources used for BLE functionality.

Syntax
void deinit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Call this member function last after all other BLE operations are stopped.

BLEDevice::connected

Description
Check if a BLE device is connected.

Syntax
bool connected(void);

Parameters
The function requires no input parameter.

Returns
TRUE if another BLE device is connected, FALSE if no BLE device is connected.

Example Code
NA

Notes and Warnings
NA

BLEDevice::setDeviceName

Description
Set the BLE GAP device name.

Syntax
void setDeviceName(String devName);

Parameters
devName: desired device name contained in an Arduino String object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The GAP device name has a maximum length of 39 characters. Other devices can see this name after a BLE connection is established. This name is separate and different from the device name sent in a BLE advertisement, the names should be the same but are not required.

BLEDevice::setDeviceAppearance

Description
Set the BLE GAP device appearance.

Syntax
void setDeviceAppearance(uint16_t devAppearance);

Parameters
devAppearance: desired device appearance expressed as a 16-bit unsigned integer.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to Bluetooth SIG assigned device appearances at https://www.bluetooth.com/specifications/gatt/characteristics/.

BLEDevice::configAdvert

Description
Configure BLE advertising parameters.

Syntax
BLEAdvert* configAdvert(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEAdvert class instance for configuring BLE advertising parameters.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function instead of creating a BLEAdvert class instance manually.

BLEDevice::configScan

Description
Configure BLE scanning parameters.

Syntax
BLEScan* configScan(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEScan class instance for configuring BLE scanning parameters.

Example Code
Example: BLEScan

```
#include “BLEDevice.h”
#include “BLEScan.h”
int dataCount = 0;
void scanFunction(T_LE_CB_DATA* p_data) {
printf(”rnScan Data %drn”, ++dataCount);
BLE.configScan()->printScanInfo(p_data);
}
void setup() {
BLE.init();
BLE.configScan()->setScanMode(GAP_SCAN_MODE_ACTIVE);
BLE.configScan()->setScanInterval(500); // Start a scan every 500ms
BLE.configScan()->setScanWindow(250); // Each scan lasts for 250ms
// Provide a callback function to process scan data.
// If no function is provided, default BLEScan::printScanInfo is used
BLE.setScanCallback(scanFunction);
BLE.beginCentral(0);
BLE.configScan()->startScan(5000); // Repeat scans for 5 seconds, then stop
}
void loop() {
}
```

Notes and Warnings

Use this member function instead of creating a BLEScan class instance manually.

BLEDevice::setScanCallback

Description
Set a callback function for processing BLE scan results.

Syntax
void setScanCallback(void (*scanCB)(T_LE_CB_DATA));

Parameters
scanCB: a function that returns nothing and takes in a scan data pointer of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to set a callback function that will be called for each BLE device scan result found.

BLEDevice::beginCentral

Description
Start the BLE stack in central mode.

Syntax
void beginCentral(uint8_t connCount);

Parameters
connCount: maximum number of allowed connected devices. If no argument is provided, default to maximum allowed connected devices for specific board.

Returns
The function returns nothing.

Example Code
Example: BLEScan
The function returns nothing.

Notes and Warnings
Use this member function to start the device in BLE central mode, after other BLE parameters are set correctly.

BLEDevice::beginPeripheral

Description
Start the BLE stack in peripheral mode.

Syntax
void beginPeripheral(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function to start the device in BLE peripheral mode, after other BLE parameters are set correctly.

BLEDevice::end

Description
Stop the BLE stack.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Use this member function to stop the device operating in either BLE peripheral mode or BLE central mode.

BLEDevice::configServer

Description
Configure the BLE stack for services.

Syntax
void configServer(uint8_t maxServiceCount);

Parameters
maxServiceCount: Maximum number of services that will run on the device

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function before adding any service to the BLE stack.

BLEDevice::addService

Description
Add a new service to the BLE stack.

Syntax
void addService(BLEService& newService);

Parameters
newService: the service to be added, defined using a BLEService class object.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
N/A

BLEDevice::configClient

Description
Configure the BLE stack for clients.

Syntax
void configClient();

Parameters
The function requries no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Use this member function before adding any client to the BLE stack.

BLEDevice::addClient

Description
Add a new client to the BLE stack.

Syntax
BLEClient* addClient(uint8_t connId);

Parameters
connId: the connection ID of the connected device to create a client for.

Returns
The function returns a pointer to a BLEClient class object, corresponding to the device with the specified connection ID, which can be used to access the services and characteristics on the connected device.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Only one client should be added per connected device.
The BLEClient object and any service, characteristic, descriptor associated with the connected device will be deleted when the device is disconnected.

BLEDevice::getLocalAddr

Description
Get local device Bluetooth address.

Syntax
void getLocalAddr(uint8_t (&addr)[GAP_BD_ADDR_LEN]);

Parameters
addr: 6 byte array to store local device Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Local device address is only available after starting in central or peripheral mode. This function will return all zeros for the address if central or peripheral mode is not in operation.

Class BLEHIDDevice

BLEHIDDevice Class

Description

A class used for creating and managing HID over GATT Profile (HOGP) services.

Syntax
class BLEHIDDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLEHIDDev.

Public Methods	
BLEHIDDevice::init	Initialize the HID Device Profile by creating the required services
BLEHIDD evice::setNumOutputReport	Configure the number of HID output reports
BLEHID Device::setNumInputReport	Configure the number of HID input reports
B LEHIDDevice::setReportMap	Configure the HID report map
BLEHIDDevice::inputReport	Send a HID input report
BLEHIDDevice ::setOutputReportCallback	Set a user callback function for receiving HID output reports
BLEHIDD evice::bootKeyboardReport	Send a HID boot keyboard input report
BLEHIDDevice::setHidInfo	Set HID info of the HID service
B LEHIDDevice::setBattLevel	Set battery level info of the Battery service
BLEHIDDevice::setPNPInfo	Set PNP information of the Device Information service
BLEHIDDevi ce::setManufacturerString	Set manufacturer information of the Device Information service
BLE HIDDevice::setModelString	Set model information of the Device Information service
BLEHIDDevice::hidService	Get reference to HID service
BLE HIDDevice::devInfoService	Get reference to Device Information service
BLEHIDDevice::battService	Get reference to Battery service

BLEHIDDevice::init

Description
Initialize the HID Device profile by creating the required services.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID Device object should be initialized before any HID reports can be sent.

BLEHIDDevice::setNumOutputReport

Description
Configure the number of HID output reports.

Syntax
void setNumOutputReport (uint8_t numOutputReports);

Parameters
numOutputReports: number of output reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of output reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setNumInputReport

Description
Configure the number of HID input reports.

Syntax
void setNumInputReport (uint8_t numInputReports);

Parameters
numInputReports: number of input reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of input reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setReportMap

Description
Configure the HID report map characteristic with a HID report descriptor.

Syntax
void setReportMap (uint8_t* report_map, uint16_t len);

Parameters
report_map: pointer to HID report descriptor
len: HID report descriptor length in bytes

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID report map characteristic can only be configured after BLEHIDDevice init() function is called.

BLEHIDDevice::inputReport

Description
Send a HID input report.

Syntax
void inputReport (uint8_t reportID, uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters

reportID: HID report ID of input report
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
HID input reports can only be sent after BLEHIDDevice init() function has been called.

BLEHIDDevice::setOutputReportCallback

Description
Set a user callback function for receiving HID output report data.

Syntax
void setOutputReportCallback (uint8_t reportID, void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
reportID: HID report ID of output report to link callback function with
chr: BLECharacteristic class object containing received HID output report data
conn_id: connection ID of device which sent HID report data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Setting a user callback function for output reports can only occur after BLEHIDDevice init() function has been called.

BLEHIDDevice::bootKeyboardReport

Description
Send a HID boot keyboard input report.

Syntax
void bootKeyboardReport (uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, the HID service Protocol Mode characteristic has boot mode disabled. To send boot keyboard input reports, the Protocol Mode characteristic needs to have boot mode enabled.

BLEHIDDevice::setHidInfo

Description
Set data of the HID Info characteristic of the HID service.

Syntax
void setHidInfo (uint16_t bcd, uint8_t country, uint8_t flags);

Parameters
bcd: 16-bit unsigned integer representing version number of base USB HID Specification implemented by HID Device
country: 8-bit integer identifying country HID Device hardware is localized for. Most hardware is not localized (value 0x00).
flags: Bit flags indicating remote-wake capability and advertising when bonded but not connected.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
For detailed information on the characteristic, refer to Bluetooth SIG HID Service specifications.

BLEHIDDevice::setBattLevel

Description
Set battery level data of the Battery service.

Syntax
void setBattLevel (uint8_t level);

Parameters
level: battery level expressed as % of full charge

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Battery level is set to 100% by default. For detailed information refer to Bluetooth SIG Battery service specifications.

BLEHIDDevice::setPNPInfo

Description
Set PNP data of the Device Information service.

Syntax
void setPNPInfo (uint8_t sig, uint16_t vid, uint16_t pid, uint16_t version);

Parameters
sig: The Vendor ID Source field designates which organization assigned the value used in the Vendor ID field value.
vid: The Vendor ID field is intended to uniquely identify the vendor of the device.
pid: The Product ID field is intended to distinguish between different products made by the vendor.
version: The Product Version field is a numeric expression identifying the device release number in Binary-Coded Decimal.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, sig and vid are configured to indicate Realtek as the vendor. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setManufacturerString

Description
Set manufacturer information of the Device Information service.

Syntax
void setManufacturerString (const char* manufacturer);

Parameters
manufacturer: pointer to character string containing manufacturer name info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Manufacturer is set to "Realtek" by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setModelString

Description
Set model information of the Device Information service.

Syntax
void setModelString (const char* model);

Parameters
model: pointer to character string containing device model info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Model is set to "Ameba_BLE_HID" by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::hidService

Description
Get reference to HID service

Syntax
BLEService& hidService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the HID service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::devInfoService

Description
Get reference to Device Information service

Syntax
BLEService& devInfoService ();

Parameters
The function requires no input parameter.

Returns

The function returns a reference to the BLEService class object for the Device Information service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::battService

Description
Get reference to Battery service

Syntax
BLEService& battService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the Battery service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLEHIDGamepad

BLEHIDGamepad Class

Description

A class used for creating and managing a BLE HID Gamepad.

Syntax
class BLEHIDGamepad

Members

Public Constructors	
BLEHIDGame pad::BLEHIDGamepad	Constructs a BLEHIDGamepad object
Public Methods	
BLEHIDGa mepad::setReportID	Set HID report ID for the HID Gamepad
BLEHIDGame pad::gamepadReport	Send a HID Gamepad report
BLEHIDGa mepad::buttonPress	Send a HID Gamepad report indicating buttons pressed
BLEHIDGame pad::buttonRelease	Send a HID Gamepad report indicating buttons released
BLEHIDGamepad ::buttonReleaseAll	Send a HID Gamepad report indicating no buttons pressed
BLE HIDGamepad::setHat	Send a HID Gamepad report indicating hat switch position
BLEH IDGamepad::setAxes	Send a HID Gamepad report indicating position of all axes
BLEHIDGam epad::setLeftStick	Send a HID Gamepad report indicating position of axes corresponding to left analog stick
BLEHIDGame pad::setRightStick	Send a HID Gamepad report indicating position of axes corresponding to right analog stick
BLEHIDGa mepad::setTriggers	Send a HID Gamepad report indicating position of axes corresponding to triggers

BLEHIDGamepad::BLEHIDGamepad

Description
Constructs a BLE object

Syntax
BLEHIDGamepad::BLEHIDGamepad();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
By default, the BLEHIDGamepad class assumes the HID report descriptor implements a gamepad device with 16 buttons, 6 16-bit axes and an 8-direction hat switch. This class will not work if a different gamepad report descriptor is implemented.

BLEHIDGamepad::setReportID

Description
Set HID report ID for the HID Gamepad.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the gamepad device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
HID report ID should start at 1. Some systems may consider a report ID of 0 as invalid.

BLEHIDGamepad::gamepadReport

Description
Send a HID Gamepad report.

Syntax
void gamepadReport (hid_gamepad_report_t* report);
void gamepadReport (uint16_t buttons, uint8_t hat, int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
report: pointer to gamepad report structure containing data on all inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::buttonPress

Description
Send a HID Gamepad report indicating buttons pressed.

Syntax
void buttonPress (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonRelease

Description
Send a HID Gamepad report indicating buttons released.

Syntax
void buttonRelease (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonReleaseAll

Description
Send a HID Gamepad report indicating no buttons pressed.

Syntax
void buttonReleaseAll (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setHat

Description
Send a HID Gamepad report indicating hat switch position.

Syntax
void setHat (uint8_t hat);

Parameters
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setAxes

Description
Send a HID Gamepad report indicating position of all axes.

Syntax
void setAxes (int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setLeftStick

Description
Send a HID Gamepad report indicating position of axes corresponding to left analog stick.

Syntax
void setLeftStick (int16_t x, int16_t y);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setRightStick

Description
Send a HID Gamepad report indicating position of axes corresponding to right analog stick.

Syntax
void setLeftStick (int16_t z, int16_t Rz);

Parameters
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setTriggers

Description
Send a HID Gamepad report indicating position of axes corresponding to triggers.

Syntax
void setTriggers (int16_t Rx, int16_t Ry);

Parameters
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

Class BLEHIDKeyboard

BLEHIDKeyboard Class

Description

A class used for creating and managing a BLE HID Keyboard.

Syntax
class BLEHIDKeyboard

Members

Public Constructors	
BLEHIDKeybo ard::BLEHIDKeyboard	Constructs a BLEHIDKeyboard object
Public Methods	
BLEHIDKe yboard::setReportID	Set HID report ID for the HID Keyboard and HID consumer control
BLEHIDKeybo ard::consumerReport	Send a HID Consumer report
BLEHIDKeybo ard::keyboardReport	Send a HID Keyboard report
BLEHIDKeyb oard::consumerPress	Send a HID Consumer report indicating button pressed
BLEHIDKeyboa rd::consumerRelease	Send a HID Consumer report indicating button released
BLEHI DKeyboard::keypress	Send a HID Keyboard report indicating keys pressed
BLEHIDK eyboard::keyRelease	Send a HID Keyboard report indicating keys released
BLEHIDKeyb oard::keyReleaseAll	Send a HID Keyboard report indicating no keys pressed
BLEHIDKey board::keyCharPress	Send a HID Keyboard report indicating keys pressed to output an ASCII character
BLEHIDKe yboard::keySequence	Send a HID Keyboard report indicating keys pressed to output an ASCII string

BLEHIDKeyboard::BLEHIDKeyboard

Description
Constructs a BLEHIDKeyboard object.

Syntax
BLEHIDKeyboard::BLEHIDKeyboard();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::setReportID

Description
Set HID report ID for the HID Keyboard and HID consumer control.

Syntax
void setReportID (uint8_t reportIDKeyboard, uint8_t reportIDConsumer);

Parameters
reportIDKeyboard: The report ID for the HID keyboard device, corresponding to the HID report descriptor.
reportIDConsumer: The report ID for the HID consumer control device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerReport

Description
Send a HID Consumer report.

Syntax
void consumerReport (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyboardReport

Description
Send a HID Keyboard report.

Syntax
void keyboardReport (void);
void keyboardReport (uint8_t modifiers, uint8_t keycode[6]);

Parameters
modifiers: bitmap indicating key modifiers pressed (CTRL, ALT, SHIFT).
keycode: byte array indicating keys pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerPress

Description
Send a HID Consumer report indicating button pressed.

Syntax
void consumerPress (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerRelease

Description
Send a HID Consumer report indicating button released.

Syntax
void consumerRelease (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keypress

Description
Send a HID Keyboard report indicating keys pressed.

Syntax
void keyPress (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyRelease

Description
Send a HID Keyboard report indicating keys released.

Syntax
void keyRelease (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyReleaseAll

Description
Send a HID Keyboard report indicating no keys pressed.

Syntax
void keyReleaseAll(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyCharPress

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII character.

Syntax
void keyCharPress (char ch);

Parameters
ch: ASCII character to output.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keySequence

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII string.

Syntax
void keySequence (const char* str, uint16_t delayTime);
void keySequence (String str, uint16_t delayTime);

Parameters
str: pointer to character string to output
str: String object containing character string to output

delayTime: time delay between key press and release, in milliseconds. Default value of 5.

Returns
The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

Class BLEHIDMouse

BLEHIDMouse Class

Description

A class used for creating and managing a BLE HID Mouse.

Syntax
class BLEHIDMouse

Members

Public Constructors	
BLE HIDMouse::BLEHIDMouse	Constructs a BLEHIDMouse object
Public Methods	
BLE HIDMouse::setReportID	Set HID report ID for the HID Mouse
BLE HIDMouse::mouseReport	Send a HID Mouse report
BL EHIDMouse::mousePress	Send a HID Mouse report indicating buttons pressed
BLEH IDMouse::mouseRelease	Send a HID Mouse report indicating buttons released
BLEHIDM ouse::mouseReleaseAll	Send a HID Mouse report indicating no buttons pressed
B LEHIDMouse::mouseMove	Send a HID Mouse report indicating mouse movement
BLE HIDMouse::mouseScroll	Send a HID Mouse report indicating mouse scroll wheel movement

BLEHIDMouse::BLEHIDMouse

Description
Constructs a BLEHIDMouse object.

Syntax
BLEHIDMouse::BLEHIDMouse();

Parameters

The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::setReportID

Description
Set HID report ID for the HID Mouse.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the HID mouse device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseReport

Description
Send a HID Mouse report.

Syntax
void mouseReport (hid_mouse_report_t* report);
void mouseReport (uint8_t buttons, int8_t x, int8_t y, int8_t scroll);

Parameters
report: pointer to mouse report structure containing data on mouse inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mousePress

Description
Send a HID Mouse report indicating buttons pressed.

Syntax
void mousePress (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseRelease

Description
Send a HID Mouse report indicating buttons released.

Syntax
void mouseRelease (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseReleaseAll

Description
Send a HID Mouse report indicating no buttons pressed.

Syntax
void mouseReleaseAll(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseMove

Description
Send a HID Mouse report indicating mouse movement.

Syntax
void mouseMove (int8_t x, int8_t y);

Parameters
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseScroll

Description
Send a HID Mouse report indicating mouse scroll wheel movement.

Syntax
void mouseScroll (int8_t scroll);

Parameters
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLERemoteCharacteristic

BLERemoteCharacteristic Class

Description
A class used for managing BLE GATT characteristics on connected remote devices.

Syntax
class BLERemoteCharacteristic

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteService::getCharacteristic().

Public Methods	
BLERem oteCharacteristic::getDescriptor	Get a specific descriptor on the remote device
BLERemoteCharacteristic::getUUID	Get the characteristic UUID
BLERe moteCharacteristic::setBufferLen	Set the size of the internal data buffer
BLERe moteCharacteristic::getBufferLen	Get the current size of the internal data buffer
BLERemoteCharacteristic::canRead	Determine if characteristic has read property enabled
B LERemoteCharacteristic::canWrite	Determine if characteristic has write property enabled
BL ERemoteCharacteristic::canNotify	Determine if characteristic has notify property enabled
BLER emoteCharacteristic::canIndicate	Determine if characteristic has indicate property enabled
BLERem oteCharacteristic::getProperties	Get the characteristic properties
BLE RemoteCharacteristic::readString	Read the characteristic data buffer as a String object
BL ERemoteCharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLE RemoteCharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLE RemoteCharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLER emoteCharacteristic::writeString	Write data to the characteristic as a String object or character array
BLE RemoteCharacteristic::writeData8	Write data to the characteristic as an unsigned 8-bit integer
BLER emoteCharacteristic::writeData16	Write data to the characteristic as an unsigned 16-bit integer
BLER emoteCharacteristic::writeData32	Write data to the characteristic as an unsigned 16-bit integer
BLERemoteCharacteristic::setData	Write data to the characteristic
BLERemoteCharacteristic::getData	Read data from the characteristic
BLERemoteChar acteristic::enableNotifyIndicate	Enable notification or indication for the characteristic
BLERemoteChara cteristic::disableNotifyIndicate	Disable notification and indication for the characteristic
BLERemoteC haracteristic::setNotifyCallback	Set a user function as a notification callback

BLERemoteCharacteristic::getDescriptor

Description
Get a descriptor with the specified UUID on the remote device.

Syntax
BLERemoteDescriptor* getDescriptor(const char* uuid);
BLERemoteDescriptor* getDescriptor(BLEUUID uuid);

Parameters
uuid: the desired descriptor UUID, expressed as a character array or a BLEUUID object

Returns
The function returns the found descriptor as a BLERemoteDescriptor object pointer, otherwise nullptr is returned if a descriptor with the UUID is not found.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteCharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canRead

Description
Determine if characteristic has read property enabled.

Syntax
bool canRead();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the read property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canWrite

Description
Determine if characteristic has write property enabled.

Syntax
bool canWrite();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the write property for the characteristic is enabled.

Example Code
NA

Notes and Warnings

NA

BLERemoteCharacteristic::canNotify

Description
Determine if characteristic has notify property enabled.

Syntax
bool canNotify();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the notify property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canIndicate

Description
Determine if characteristic has indicate property enabled.

Syntax
bool canIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the indicate property for the characteristic is enabled.

Example Code

NA

Notes and Warnings
NA

BLERemoteCharacteristic::getProperties

Description
Get the characteristic properties.

Syntax
uint16_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic properties.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readString

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns

The function returns the data in the characteristic buffer expressed as a String class object.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData8

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint8_t value.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData16

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readData32

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeString

Description
Write data to the remote device characteristic as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote characteristic, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData8

Description
Write data to the remote device characteristic as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData16

Description
Write data to the remote device characteristic as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData32

Description
Write data to the remote device characteristic as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setData

Description
Write data to the remote device characteristic.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getData

Description
Request for characteristic data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLERemoteCharacteristic::enableNotifyIndicate

Description
Enable the remote device to send notifications or indications for the characteristic.

Syntax
void enableNotifyIndicate(bool notify = 1);

Parameters
notify: TRUE to enable notifications, FALSE to enable indications.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::disableNotifyIndicate

Description
Disable receiving notifications and indications for the characteristic from the remote device.

Syntax
void disableNotifyIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setNotifyCallback

Description
Set a user function to be called when the characteristic receives a notification from the remote device.

Syntax
void setNotifyCallback(void (*fCallback) (BLERemoteCharacteristic* chr, uint8_t* data, uint16_t length));

Parameters
fCallback: A user callback function that returns void and takes three arguments.
chr: pointer to BLERemoteCharacteristic object associated with notification.
data: pointer to byte array containing notification data.
length: number of bytes of notification data in array.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLERemoteDescriptor

BLERemoteDescriptor Class

Description
A class used for managing BLE GATT descriptors on connected remote devices.

Syntax
class BLERemoteDescriptor

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteCharacteristic::getDescriptor().

Public Methods	
BLERemoteDescriptor::getUUID	Get the descriptor UUID
B LERemoteDescriptor::setBufferLen	Set the size of the internal data buffer
B LERemoteDescriptor::getBufferLen	Get the current size of the internal data buffer
BLERemoteDescriptor::readString	Read the descriptor data buffer as a String object
BLERemoteDescriptor::readData8	Read the descriptor data buffer as an unsigned 8-bit integer
BLERemoteDescriptor::readData16	Read the descriptor data buffer as an unsigned 16-bit integer
BLERemoteDescriptor::readData32	Read the descriptor data buffer as an unsigned 32-bit integer
BLERemoteDescriptor::writeString	Write data to the descriptor as a String object or character array
BLERemoteDescriptor::writeData8	Write data to the descriptor as an unsigned 8-bit integer
BLERemoteDescriptor::writeData16	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::writeData32	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::setData	Write data to the descriptor
BLERemoteDescriptor::getData	Read data from the descriptor

BLERemoteDescriptor::getUUID

Description
Get the descriptor UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the descriptor UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setBufferLen

Description
Set the size of the internal data buffer of the descriptor.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Descriptor data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteDescriptor::getBufferLen

Description
Get the size of the descriptor internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readString

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a String class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData8

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData16

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData32

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeString

Description
Write data to the remote device descriptor as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote descriptor, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData8

Description
Write data to the remote device descriptor as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData16

Description
Write data to the remote device descriptor as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData32

Description
Write data to the remote device descriptor as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the descriptor buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setData

Description
Write data to the remote device descriptor.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::getData

Description
Request for descriptor data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

Class BLERemoteService

BLERemoteService Class

Description
A class used for managing BLE GATT services on connected remote devices.

Syntax
class BLERemoteService

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLE-Client::getService().

Public Methods	
BLERemoteService::getUUID	Get the service UUID
BLE RemoteService::getCharacteristic	Get a specific characteristic on the remote device

BLERemoteService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteService::getCharacteristic

Description
Get a characteristic with the specified UUID on the remote device.

Syntax
BLERemoteCharacteristic* getCharacteristic (const char* uuid);
BLERemoteCharacteristic* getCharacteristic (BLEUUID uuid);

Parameters
uuid: the desired characteristic UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found characteristic as a BLERemoteCharacteristic object pointer, otherwise nullptr is returned if a characteristic with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLEScan

BLEScan Class

Description
A class used for managing BLE scanning settings.

Syntax
class BLEScan

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configScan

Public Methods	
BLEScan::updateScanParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEScan::startScan	Start a BLE scan
BLEScan::stopScan	Stop a BLE scan
BLEScan::setScanMode	Set the BLE scanning mode
BLEScan::setScanInterval	Set the BLE scanning interval
BLEScan::setScanWindow	Set the BLE scanning window
BLEScan::setScanDuplicateFilter	Set the BLE scan duplicate filter
BLEScan::scanInProgress	Check if a scan is currently in progress
BLEScan::printScanInfo	Print out scanned information

BLEScan::updateScanParams

Description
Update the lower Bluetooth stack with the current scan settings.

Syntax
void updateScanParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Stop any scans in progress first before using this function.

BLEScan::startScan

Description
Start BLE scanning.

Syntax
void startScan();
void startScan(uint32_t scanDuration_ms);

Parameters
scanDuration: BLE scan will stop after scanDuration milliseconds.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Set the scan parameters first before starting a scan. BLE scans will occur continuously for the duration set with BLEDevice::setScanWindow() and will repeat with a time interval set with BLEDevice::setScanInterval(). Call this member function without an argument to start scanning until BLEDevice::stopScan() is called.

BLEScan::stopScan

Description
Stop BLE scanning.

Syntax
void stopScan(void);

Parameters
The function requires no input paramter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEScan::setScanMode

Description
Set the BLE scan mode.

Syntax
void setScanMode(uint8_t scanMode);

Parameters
scanMode: GAP_SCAN_MODE_PASSIVE for passive scanning, GAP_SCAN_MODE_ACTIVE for active scanning

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Active scanning will request for scan response packets after discovering an advertising device. Passive scanning will only capture advertising data packets.

BLEScan::setScanInterval

Description
Set the BLE scan interval.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will repeat with a time interval set with this member function.

BLEScan::setScanWindow

Description
Set the BLE scan window.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will scan continuously for a window duration set with this member function. The scan window should be less than or equal to the scan interval.

BLEScan::setScanDuplicateFilter

Description
Set the scan duplicate filter.

Syntax
void setScanDuplicateFilter(bool dupeFilter);

Parameters
dupeFilter: TRUE to enable duplicate filtering.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling duplicate filters will ignore scan results for devices already discovered previously.

BLEScan::scanInProgress

Description
Set the scan duplicate filter.

Syntax
bool scanInProgress(void);

Parameters
The function requires no input paramter.

Returns
TRUE if BLE scanning is in progress.

Example Code
NA

Notes and Warnings
NA

BLEScan::printScanInfo

Description
Parse and print out scanned information.

Syntax
void printScanInfo(T_LE_CB_DATA* p_data);

Parameters
p_data: pointer to scan data of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to parse the various fields of received advertisement data packets and print the results out to the serial monitor.

Class BLEService

BLEService Class

Description
A class used for creating and managing BLE GATT services.

Syntax
class BLEService

Members

Public Constructors	
BLEService::BLEService	Constructs a BLEService object
Public Methods	
BLEService::setUUID	Set service UUID
BLEService::getUUID	Get service UUID
BLEService::addCharacteristic	Add a characteristic to service
BLEService::getCharacteristic	Get a previously added characteristic

BLEService::BLEService

Description

Constructs a BLEService object.

Syntax
BLEService::BLEService(BLEUUID uuid);
BLEService::BLEService(const char* uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object or a character array

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLEService::setUUID

Description
Set the service UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLEService::addCharacteristic

Description
Add a characteristic to the service.

Syntax
void addCharacteristic(BLECharacteristic& newChar);

Parameters
newChar: the BLECharacteristic to add to the service.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings

NA

BLEService::getCharacteristic

Description
Get a previously added characteristic.

Syntax
BLECharacteristic* getCharacteristic(uint8_t charIndex);

Parameters
charIndex: position index of characteristic.

Returns
The function returns a pointer to the BLECharacteristic at the requested position index.

Example Code
NA

Notes and Warnings
NA

Class BLEUUID

BLEUUID Class

Description
A class used for creating and managing UUIDs.

Syntax
class BLEUUID

Members

Public Constructors	
BLEUUID::BLEUUID	Create a UUID object
Public Methods	
BLEUUID::str	Get the character string representation of UUID
BLEUUID::data	Get the binary representation of UUID
BLEUUID::length	Get the length of UUID

BLEUUID::BLEUUID

Description
Create a UUID object from a UUID character string

Syntax
BLEUUID();
BLEUUID(const char* str);
BLEUUID(uint8_t* data, uint8_t length);

Parameters
str: UUID character string used to created object
data: pointer to byte array containing the desired UUID
length: number of bytes in array containing the desired UUID. Valid values of 2, 4 or 16

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLEUUID::str

Description
Get the character string representation of UUID

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a character string representation of the UUID

Example Code
NA

Notes and Warnings

BLEUUID::data

Description
Get the binary representation of UUID

Syntax
const uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
Pointer to an unsigned 8-bit integer array containing the UUID expressed in binary form

Example Code
NA

Notes and Warnings
Returned pointer is of const uint8_t* type and will not allow changing of the data.

BLEUUID::length

Description
Get the length of UUID

Syntax
uint8_t length(void);

Parameters
The function requires no input parameter.

Returns
Length of the UUID, in terms of bytes

Example Code

NA

Notes and Warnings
A 4-character UUID will be 16 bits / 2 bytes long.
A 32-character UUID will be 128 bits / 16 bytes long.

Class BLEWifiConfigService

BLEWifiConfigService Class

Description
A class used for managing a BLE WiFi configuration service running on the device.

Syntax
class BLEWifiConfigService

Members

Public Constructors	
BLEWifiCon figService::BLEWifiConfigService	Only one instance of this class should be created

Public Methods	
BLEWifiConfigService::begin	Start background thread to process WiFi configuration commands
BLEWifiConfigService::end	Stop background thread processing WiFi configuration commands
BLEWifiConfigService::addService	Add the service to the BLE stack
BLEWifiConfigService::advData	Get advertising data correctly formatted for WiFi configuration service

BLEWifiConfigService::BLEWifiConfigService

Description
Create an instance of the BLEWifiConfigService object.

Syntax
void BLEWifiConfigService ();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
Only one instance of this class / service should be created.

BLEWifiConfigService::begin

Description
Start background thread to process WiFi configuration commands.

Syntax
void begin();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::end

Description
Stop background thread processing WiFi configuration commands.

Syntax
void end();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEWifiConfigService::addService

Description
Add the WiFi configuration service to the BLE stack.

Syntax
void addService();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::advData

Description
Get advertising data correctly formatted for WiFi configuration service.

Syntax
BLEAdvertData advData();

Parameters
The function requires no input parameter.

Returns
The function returns a BLEAdvertData object that contains the required advertising data fields for the WiFi configuration service to work.

Example Code
Example: BLEWifiConfig

Notes and Warnings
The advertisement data needs to be correctly formatted for the corresponding smartphone app to recognise the device. WiFi configuration service advertisement data requires the local BT address, and should be called only after peripheral mode is started and may also require stopping and restarting the advertising process.

EPDIF

Class EpdIF

EpdIf Class

Description
A class used to control the electronic paper display internal functions.

Syntax
class EpdIf

Members

Public Constructors
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named EpdIf.

Public Methods	
EpdIf::EPD_Dis_Part	Put an image buffer to the frame memory, but not updating the display
EpdIf::EPD_SetFrame	Put display data to the frame memory, usually used for setup text display functions
EpdIf::EPD_SetRAMValue_BaseMap	To read image data stored in the RAM, but not display on the screen
EpdIf::EPD_SetFrameMemory	To read image data stored in the buffer, but not display on the screen
EpdIf::EPD_UpdateDisplay	Update the display
EpdIf::EPD_ClearScreen_White	Clear the frame memory with the White color, but not updating the display
EpdIf::EPD_ClearScreen_Black	Clear the frame memory with the Black color, but not updating the display
EpdIf::EPD_Busy	Wait until the Busy pin goes to low, which is the idle state
EpdIf::EPD_Reset	Used for the Epaper module reset. Often used to awaken the module in deep sleep
EpdIf::EPD_Sleep	After this command is transmitted, the chip would enter the deep-sleep mode to save power

EpdIf:: EPD_Dis_Part

Description

Put an image buffer to the frame memory, but not updating the display.

Syntax

void EPD_Dis_Part(unsigned int x_start, unsigned int y_start, const unsigned char* datas, unsigned int PART_COLUMN, unsigned int PART_LINE);

Parameters

x_start: starting position of the x-axis
y_start: starting position of the y-axis
datas: data to be displayed on the e-paper module
PART_COLUMN: height of the display area
PART_LINE: width of the display area

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

EpdIf:: EPD_SetFrame

Description

Put display data to the frame memory, usually used for setup text display functions.

Syntax
void EPD_SetFrame(const unsigned char* image_buffer, int x, int y, int image_width, int image_height);

Parameters
image_buffer: the buffer which stores the data to be displayed on the e-paper module, usually used to display texts.
x: starting position of the x-axis
y: starting position of the y-axis
image_width: width of the display area
image_height: height of the display area

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetRAMValue_BaseMap

Description
To read image data stored in the RAM, but not display on the screen.

Syntax
void EPD_SetRAMValue_BaseMap(const unsigned char* datas);

Parameters
datas: contains the black and white information that forms the image stored in RAM

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetFrameMemory

Description
To read image data stored in the buffer but not display on the screen.

Syntax
void EPD_SetFrameMemory(const unsigned char* image_buffer);

Parameters
image_buffer: the buffer where stores the image data in hexadecimal numbers

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_UpdateDisplay

Description
Update the ePaper display module. Always combined used with functions set the frames.

Syntax
void EPD_UpdateDisplay(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

There are 2 memory areas embedded in the e-paper display but once this function is called, then the next action of SetFrameMemory or ClearScreen will set the other memory area.

EpdIf:: EPD_ClearScreen_White

Description
Clear the frame memory with the White color.

Syntax
void EpdIf::EPD_ClearScreen_White(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_ClearScreen_Black

Description
Clear the frame memory with the Black color.

Syntax
void EpdIf::EPD_ClearScreen_Black(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Busy

Description
Wait until the busy_pin goes to low, which is the idle state.

Syntax
void EpdIf::EPD_Busy(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Reset

Description
This command will let the E-paper module reset, it is often used to awaken the module in while it's in the deep sleep mode, you will find more details in the function EpdIf:: EPD_Sleep().

Syntax
void EpdIf::EPD_Reset(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf::EPD_Sleep

Description
After this command is transmitted, the chip would enter the deep-sleep mode to save power. The deep sleep mode would return to standby by hardware reset. You can use EPD:: Init() to awaken the E-paper module.

Syntax
void EpdIf::EPD_Sleep(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

FatfsSDCard

Class SdFatFs

Description

Defines a class of SD FAT File system.

Syntax

class SdFatFs

Members

Public Constructors

SdFatFs::SdFatFs Constructs a SdFatFs object

SdFatFs::~SdFatFs Destructs a SdFatFs object

Public Methods

SdFatFs::begin	Initialize SD FAT File System
SdFatFs::end	Deinitialize SD FAT File System
SdFatFs::*getRootPath	Get the root path of the SD FAT File System
SdFatFs::readDir	List items under a specific folder
SdFatFs::mkdir	Create folder
SdFatFs::rm	Remove folder or file
SdFatFs::isDir	Check if a specific path is a directory
SdFatFs::isFile	Check if a specific path is a file
SdFatFs::getLastModTime	Get the last modified time for a file or directory
SdFatFs::setLastModTime	Set the last modified time for a file or directory
SdFatFs::status	Return the current status of SD
SdFatFs::open	Open a file

SdFatFs::begin

Description
Initialize SD FAT File System.

Syntax
int SdFatFs::begin(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::end

Description
De-initialize SD FAT File System.

Syntax
int SdFatFs::end(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::*getRootPath

Description
Get the root path of the SD FAT File System. The logical volume character is starting from ‘0’, so the root path would like “0:/”.

Syntax
char *SdFatFs::getRootPath(void);

Parameters
The function requires no input parameter.

Returns
The function returns the root path.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::readDir

Description
List items under a specific folder. List items under a specific folder and store the result in the buffer that user specified. Each item is separated by '0'.

Syntax
int SdFatFs::readDir(char *path, char *result_buf, unsigned int bufsize);

Parameters
path: The absolute directory path to be listed.
result_buf: The buffer to be stored results.
bufsize: The size of result_buf. If results exceed this size, then the results larger than this size would be discarded.

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: get_file_attribute; list_root_files

Notes and Warnings
Include "SdFatFs.h" to use the class function.

SdFatFs::mkdir

Description
Create folder.

Syntax
int SdFatFs::mkdir(char *absolute_path);

Parameters
absolute_path: The absolute directory path to be created

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: create_folder

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::rm

Description
Remove folder or file.

Syntax
int SdFatFs::rm(char *absolute_path);

Parameters
absolute_path: The absolute directory or file path to be deleted

Returns
Returns “0” if success, else returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isDir

Description
Check if a specific path is a directory.

Syntax
unsigned char SdFatFs::isDir(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isFile

Description
Check if a specific path is a file.

Syntax
unsigned char SdFatFs::isFile(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::getLastModTime

Description
Get the last modified time for a file or directory.

Syntax
int SdFatFs::getLastModTime(char *absolute_path, uint16_t *year, uint16_t *month, uint16_t *date, uint16_t *hour, uint16_t *minute, uint16_t *second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.

month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::setLastModTime

Description
Set the last modified time for a file or directory. Ameba doesn’t have built-in RTC. So we manually change file/directory last modified time.

Syntax
int SdFatFs::setLastModTime(char *absolute_path, uint16_t year, uint16_t month, uint16_t date, uint16_t hour, uint16_t minute, uint16_t second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.
month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::open

Description
Open a file.

Syntax
SdFatFile SdFatFs::open(char *absolute_path);

Parameters
absolute_path: The path to a file.

Returns
The file object is an instance of SdFatFile.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::status

Description
Return the current status of SD.

Syntax
int SdFatFs::status(void);

Parameters
The function requires no input parameter.

Returns
Function returns “1” if ready to use, else return “0” if the status is inactivating or abnormal.

Example Code
NA.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

Class SdFatFile

Description

Defines a class of SD FAT File.

Syntax
class SdFatFile

Members

Public Constructors	
SdFatFile::SdFatFile	Constructs a SdFatFile object
SdFatFile::~SdFatFile	Destructs a SdFatFile object
Public Methods	
SdFatFile::write	Write 1 byte/bytes to file
SdFatFile::read	Read 1 byte/bytes from the file
SdFatFile::peek	Read 1 byte from file without move curser
SdFatFile::available	Check if the cursor is at EOF (End-Of-File)
SdFatFile::bool	Check if file is opened
SdFatFile::seek	Change cursor to a specific position
SdFatFile::close	Close file

SdFatFile::write

Description
Write 1 byte or bytes to the file.

Syntax
size_t SdFatFile::write(uint8_t c);
size_t SdFatFile::write(const uint8_t *buf, size_t size);

Parameters
c: The character to be written.
buf: The buffer to be written.
size: The length of buffer to be written.

Returns

The function returns the number of byte count that has been successfully written to the file.

Example Code
NA.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: read

Description
Read 1 byte or bytes from the file.

Syntax
int SdFatFile::read(void);
int SdFatFile::read(void *buf, uint16_t nbyte);

Parameters
buf: The buffer to store the content.
nbyte: The buffer size. (Or can be regarded as the desired length to read).

Returns
The function returns a read character or the read size of the buffer.

Example Code

```
1. #include “FatFs_SD.h”
2.
3. char dirname[] = “testdir”;
4. char filename[] = “test.txt”;
5. char write_content[] = “hello world!”;
6.
7. FatFsSD fs;
8.
9. void setup() {
10. char buf[128];
11. char absolute_filename[128];
12.
13. fs.begin();
```

```
14.
15. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), dirname);
16. fs.mkdir(absolute_filename);
17. printf(“create dir at \”%s"rn”, absolute_filename);
18.
19. sprintf(absolute_filename, “%s%s/%s”, fs.getRootPath(), dirname, filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.println(write_content);
22. file.close();
23. printf(“create file at \”%s"rn”, absolute_filename);
24.
25. printf(“read back from \”%s"rn”, absolute_filename);
26. file = fs.open(absolute_filename);
27.
28. memset(buf, 0, sizeof(buf));
29. file.read(buf, sizeof(buf));
30.
31. file.close();
32. printf(“==== content ====rn”);
33. printf(“%s”, buf);
34. printf(“==== end ====rn”);
35.
36. fs.end();
37. }
38.
39. void loop() {
40. delay(1000);
41. }
42.
```

Example: create_folder;
This example shows how to create a folder and open a file under it.

```
1. #include “FatFs_SD.h”
2.
3. char filename[] = “test.txt”;
4. char write_content[] = “hello world!”;
```

```
5.
6.  FatFsSD fs;
7.
8.  void setup() {
9.  char buf[128];
10. char absolute_filename[128];
11.
12. fs.begin();
13.
14. printf("write something to \"%s"rn", filename);
15. sprintf(absolute_filename, "%s%s", fs.getRootPath(), filename);
16. SdFatFile file = fs.open(absolute_filename);
17.
18. file.println(write_content);
19.
20. file.close();
21. printf("write finishrnrn");
22.
23. printf("read back from \"%s"rn", filename);
24. file = fs.open(absolute_filename);
25.
26. memset(buf, 0, sizeof(buf));
27. file.read(buf, sizeof(buf));
28.
29. file.close();
30. printf("==== content ====rn");
31. printf("%s", buf);
32. printf("==== end ====rn");
33.
34. fs.end();
35. }
36.
37. void loop() {
38. delay(1000);
39. }
40.
```

Example: file_read_write;
This example shows how to open/close files and perform read/write to it.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: peek

Description
Read one byte from the file without moving the curser.

Syntax
int SdFatFile::peek(void);

Parameters
The function requires no input parameter.

Returns
The function returns the read character as an integer number.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: available

Description
Check if the cursor is at EOF.

Syntax
int SdFatFile::available(void);

Parameters
The function requires no input parameter.

Returns
The function returns “0” if the cursor is at EOF, else returns “1”.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: flush

Description
It is a nop. This is an inherited function from class Stream. And it does not affect SD File.

Syntax
void SdFatFile::flush(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: seek

Description
Change cursor to a specific position.

Syntax
int SdFatFile::seek(uint32_t pos);

Parameters
pos: The desired position.

Returns
The function returns 0 if success otherwise returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” in order to use the class function.

SdFatFile:: close

Description
Close file.

Syntax
int SdFatFile::close(void);

Parameters
The function requires no input parameter.

Returns
The function returns 0 if runs successfully otherwise it returns a negative value.

Example Code
Example: last_modified_time;
The example shows how to get and set last modified time of a file.

Example: create_folder;
This example shows how to create a folder and open a file under it. The details of the code can be found in the section of SdFatFile:: read.
Example: file_read_write;
This example shows how to open/close files and perform read/write to it. The details of the code can be found in the section of SdFatFile:: read.

1. #include <FatFs_SD.h>
2.
3. FatFsSD fs;
4.
5. char filename[] = “test.txt”;
6.
7. void setup() {
8. char absolute_filename[128];
9.

```
10. uint16_t year = 2021;
11. uint16_t month = 4;
12. uint16_t date = 4;
13. uint16_t hour = 12;
14. uint16_t minute = 12;
15. uint16_t second = 12;
16.
17. fs.begin();
18.
19. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.close();
22.
23. fs.setLastModTime(absolute_filename, year, month, date, hour, minute, second);
24.
25. fs.getLastModTime(absolute_filename, &year, &month, &date, &hour, &minute, &second);
26. printf(“filename:"%s"rn”, absolute_filename);
27. printf(“time mod:%04d/%02d/%02d %02d:%02d:%02drn”, year, month, date, hour, minute, second);
28.
29. fs.end();
30. }
31.
32. void loop() {
33. delay(1000);
34. }
35.
```

Notes and Warnings

Include “SdFatFile.h” in order to use the class function.

FlashMemory

Class EpdIF

FlashMemoryClass Class

Description
Defines a class of Flash memory API

Syntax
class FlashMemoryClass

Members

Public Constructors	
Fl ashMemoryClass::FlashMemoryClass	Constructs a FlashMemoryClass object
Fla shMemoryClass::~FlashMemoryClass	Deconstructs a FlashMemoryClass object
Public Methods	
FlashMemoryClass::begin	Initialize/Re-initialize the base address and size
FlashMemoryClass::read	Read the content to buf
FlashMemoryClass::update	Write buf back to flash memory
FlashMemoryClass::readWord	Read 4 bytes from flash memory
FlashMemoryClass::writeWord	Write 4 bytes into flash memory
FlashMemoryClass::buf_size	The buf size
FlashMemoryClass::*buf	The buf to be operated

FlashMemoryClass::FlashMemoryClass

Description
Constructs a FlashMemoryClass object.

Syntax
FlashMemoryClass(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address to operate.
_buf_size: The buf size for mirror a copy to reduce flash memory operation

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Direct read from flash memory is allowed. To write data into flash memory, each bit on flash memory can only change from '1' to '0' and it cannot change from '0' to '1'. To make sure the data are correctly written we do erase the flash memory sector before write data on it.

```
#include <FlashMemory.h>
void setup() {
FlashMemory.read();
if (FlashMemory.buf[0] == 0xFF) {
FlashMemory.buf[0] = 0x00;
FlashMemory.update();
Serial.println("write count to 0");
} else {
FlashMemory.buf[0]++;
FlashMemory.update();
Serial.print("Boot count: ");
Serial.println(FlashMemory.buf[0]);
}
}
void loop() {
delay(1000);
}
```

Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

```
#include <FlashMemory.h>
void setup() {
unsigned int value;
/* request flash size 0x4000 from 0xFC000 */
FlashMemory.begin(0xFC000, 0x4000);
/* read one word (32-bit) from 0xFC000 plus offset 0x3F00 */
value = FlashMemory.readWord(0x3F00);
printf("value is 0x%08Xrn", value);
if (value == 0xFFFFFFFF) {
value = 0;
```

```
} else {
value++;
}
/* write one word (32-bit) to 0xFC000 plus offset 0x3F00 */
FlashMemory.writeWord(0x3F00, value);
}
void loop() {
// put your main code here, to run repeatedly:
}
```

Notes and Warnings
Include “FlashMemory.h” to use the class function.

FlashMemoryClass::begin

Description
Initialize/Re-initialize the base address and size. The base address shell aligns with the size of 0x1000. And the size shell is multiple of 0x1000.

Syntax
void begin(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address
_buf_size: The desired work size

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba’s flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of “FlashMemoryClass:: FlashMemoryClass”.

Notes and Warnings

Include “FlashMemory.h” to use the class function.

FlashMemoryClass::read

Description
Read the content to buf. Read flash memory into the buf. The size would be 0x1000.

Syntax
void read(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba’s flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of “FlashMemoryClass:: FlashMemoryClass”.

Notes and Warnings
Include “FlashMemory.h” to use the class function.

FlashMemoryClass::update

Description
Write buf back to flash memory. Write flash memory with the content of the buffer. The size is 0x1000.

Syntax
void update(bool erase = true);

Parameters
erase: By default, it is true and erases flash memory before writing to it

Returns

The function returns nothing.

Example Code

Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::readWord

Description

Read 4 bytes from flash memory. Read 4 byte from specific offset based on base address.

Syntax

unsigned int readWord(unsigned int offset);

Parameters

offset: The offset according to the base address

Returns

The read data with a size of 4 bytes

Example Code

Example: ReadWriteOneWord

This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::writeWord

Description

Write 4 bytes into flash memory. It will try to write 4 bytes first. If the read data differ from the write data, then we buffer the sector of flash memory, erase it, and write correct data back to it.

Syntax
void writeWord(unsigned int offset, unsigned int data);

Parameters
offset: The offset according to the base address
data: The data to be written

Returns
The function returns nothing.

Example Code
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::buf_size

Description
The buf size (It can be regarded as work size).

Syntax
unsigned int buf_size;

Example Code
Example: FlashMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::*buf

Description
The buf to be operated. Modify buf won't change the content of the buf. It needs an update to write back to flash memory.

Syntax
unsigned char *buf;

Example Code
NA

Notes and Warnings
Include "FlashMemory.h" to use the class function.

GPIO

Class DHT

DHT Class

Description
Defines a class of using DHT temperature & humidity sensors

Syntax
class DHT

Members

Public Constructors	
DHT::DHT	Constructs a DHT object
Public Methods	
DHT::begin	Initialize the DHT sensor
DHT::readTemperature	Read temperature(Fahrenheit or Celcius) from the DHT sensor
DHT::convertCtoF	Convert a value from Celcius to Fahrenheit
DHT::convertFtoC	Convert a value from Fahrenheit to Celcius
DHT::readHumidity	Read humidity(%) from the DHT sensor
DHT::computeHeatIndex	Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations)
DHT::read	Check if the sensor is readable

DHT::DHT

Description
Constructs a DHT object.

Syntax
DHT::DHT(uint8_t pin, uint8_t type, uint8_t count)

Parameters
pin: The Arduino digital PIN connected
type: The DHT sensor type(DHT11, DHT22, or DHT21)
count: The count is now ignored as the DHT reading algorithm adjusts itself based on the speed of the processor

Returns
The function returns nothing.

Example Code
Example:DHTTester
The code demos basic testing for various DHT humidity & temperature sensors.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

// Example testing sketch for various DHT humidity/temperature sensors

// Written by ladyada, public domain

#include "DHT.h"

// The digital pin we're connected to.

#define DHTPIN 8

// Uncomment whatever type you're using!

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V

// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1

// to 3.3V instead of 5V!

// Connect pin 2 of the sensor to whatever your DHTPIN is

// Connect pin 4 (on the right) of the sensor to GROUND

// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.

// Note that older versions of this library took an optional third parameter to

```
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(115200);
Serial.println("DHTxx test!");
dht.begin();
}
void loop() {
// Wait a few seconds between measurements.
delay(2000);
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *Ft");
Serial.print("Heat index: ");
```

Serial.print(hic);

Serial.print(" *C ");

Serial.print(hif);

Serial.println(" *F");

}

DHT::begin

Description
Initialize the DHT sensor.

Syntax
void DHT::begin(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::readTemperature

Description
Read temperature(Fahrenheit or Celcius) from the DHT sensor.

Syntax
float DHT::readTemperature(bool S, bool force);

Parameters
S: Temperature scale, True is Fahrenheit and False is Celcius
force: Index of checking sensor readability, default is False

Returns
The function returns the current temperature as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertCtoF

Description
Convert a value from Celcius to Fahrenheit.

Syntax
float DHT::convertCtoF(float c);

Parameters
c: The value in Celcius

Returns
The function returns the temperature in Fahrenheit as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertFtoC

Description
Convert a value from Fahrenheit to Celcius.

Syntax
float DHT::convertFtoC(float f);

Parameters
f: The value in Fahrenheit

Returns
The function returns the temperature in Celcius as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::computeHeatIndex

Description
Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations). More details refer to http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml .

Syntax
float DHT::computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit);

Parameters
temperature: The temperature value
percentHumidity: The humidity percent value
isFahrenheit: True, temperature value in Fahrenheit (Default); False, temperature value in Celcius

Returns
The function returns the heat index in Fahrenheit or Celsius as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings

Every time must include the header file “DHT.h” in front of the project to use the class function.

DHT::readHumidity

Description
Reading temperature or humidity from the DHT sensor and return as a float value(%).

Syntax
float DHT::readHumidity(bool force);

Parameters
force: Ignored.

Returns
The function returns current humidity in a float number (in %).

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the “DHT: DHT” section.

Notes and Warnings
Every time must include the header file “DHT.h” in front of the project to use the class function. Reading temperature or humidity takes about 250 milliseconds! Sensor readings may also be up to 2 seconds.

DHT::read

Description
Check if the sensor is readable.

Syntax
boolean DHT::read(bool force);

Parameters
force: Index of whether checking the sensor was read less than two seconds ago or not. False, checking; True, not checking.

Returns

Return the last correct measurement of the sensor. False, low means not readable; True, high means readable.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

Class HttpClient

InterruptLock Class

Description
Defines a class of turning off/on interrupts temporarily

Syntax
class InterruptLock

Members

Public Constructors	
InterruptLock::InterruptLock	Constructs a InterruptLock object
InterruptLock::~ InterruptLock	Deconstructs a InterruptLock object

---

GTimer

Class EpdIF

GTimerClass Class

Description
GTimer is a hardware timer and this class is to operate it. The GTimer occupy same resource as PWM. Please make sure the timer is not conflict with you PWM index.

Syntax
class GTimerClass

Members

Public Constructors	
GTimerClass::GTimerClass	Constructs a GTimerClass object
Public Methods	
GTimerClass::begin	Initialize a timer and start it immediately
GTimerClass::stop	Stop a specific timer
GTimerClass::reload	Reload a specific timer
GTimerClass::read_us	Read current countdown value

GTimerClass::begin

Description
Initialize a timer and start it immediately.

Syntax
void GTimerClass::begin(uint32_t timerid, uint32_t duration_us, void (*handler)(uint32_t), bool periodical, uint32_t userdata);

Parameters
timerid: There are 5 valid GTimer with timer id 0~4.
duration_us: The duration of the timer. The time unit is microsecond and the precision is 32768Hz.
periodical: By default, the timer would keep periodically countdown and reload which means the handler would periodically be invoked.
userdate: The user data brings to the handler.

Returns
The function returns nothing.

Example Code
Example: TimerOneshot

/*

This sketch shows how to use several hardware timers in invoke handler only once for each timer.

*/

#include <GTimer.h>

void myhandler(**uint32_t** data) {

Serial.print("I am timer!");

Serial.println(data);

}

void setup() {

```
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// timerid 0, period 1s, invoke myhandler, invoke only once, user data is 0
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
// timerid 1, period 2s, invoke myhandler, invoke only once, user data is 1
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
GTimer.begin(2, 3 * 1000 * 1000, myhandler, false, 2);
GTimer.begin(3, 4 * 1000 * 1000, myhandler, false, 3);
}
void loop() {
delay(1000);
}
```

Example: TimerPeriodical

```
/*
This sketch shows how to use hardware timer and invoke interrupt handler periodically
*/
#include <GTimer.h>
int counter = 0;
void myhandler(uint32_t data) {
counter++;
Serial.print("counter: ");
Serial.println(counter);
if (counter >= 10) {
Serial.println("stop timer");
GTimer.stop(0);
}
}
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
```

```
// timerid 0, period 1s, invoke myhander
GTimer.begin(0, (1 * 1000 * 1000), myhandler);
}
void loop() {
delay(1000);
}
```

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::stop

Description
Stop a specific timer

Syntax
void GTimerClass::stop(uint32_t timerid);

Parameters
timerid: Stop the timer with this timer id

Returns
The function returns nothing.

Example Code
Example: TimerPeriodical, please refer to GTimerClass:: begin for more details.

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::reload

Description
Reload a specific timer. The GTimer is a countdown timer. Reload it would make it discard the current countdown value and restart countdown based on the duration.

Syntax

void GTimerClass::reload(uint32_t timerid, uint32_t duration_us);

Parameters
timerid: The timer to be modified
duration_us: The updated duration in unit of microseconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::read_us

Description
Read the current countdown value

Syntax
uint64_t GTimerClass::read_us(uint32_t timerid);

Parameters
timerid: The timer to be read

Returns
The function returns the current countdown value.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

Http

Class HttpClient

HttpClient Class

Description
Defines a class of using HttpClient

Syntax
class HttpClient

Members

Public Constructors	
HttpClient::HttpClient	Constructs a HttpClient object
Public Methods	
HttpClient::beginRequest	Start a more complex request
HttpClient::endRequest	End a more complex request
HttpClient::get	Connect to the server and start to send a GET request
HttpClient::post	Connect to the server and start to send a POST request
HttpClient::put	Connect to the server and start to send a PUT request
HttpClient::startRequest	Connect to the server and start to send the request
HttpClient::sendHeader	Send an additional header line
HttpClient::sendBasicAuth	Send a basic authentication header
HttpClient::finishRequest	Finish sending the HTTP request
HttpClient::responseStatusCode	Get the HTTP status code contained in the response
HttpClient::readHeader	Read the next character of the response headers
HttpClient::skipResponseHeaders	Skip any response headers to get to the body
HttpClient::endOfHeadersReached	Test whether all of the response headers have been consumed
HttpClient::endOfBodyReached	Test whether the end of the body has been reached
HttpClient::contentLength	Return the length of the body

HttpClient::HttpClient

Description
Constructs a HttpClient object. If Marco “PROXY_ENABLED” is defined, currently disabled as introduces a dependency on DNS.h in Ethernet.

Syntax
HttpClient::HttpClient(Client& aClient, const char* aProxy = NULL, uint16_t aProxyPort = 0);
HttpClient::HttpClient(Client& aClient);

Parameters
aClient: The object of class WiFiClient.

aProxy: The proxy name. The default proxy name is “NULL”.
aProxyPort: The proxy port. The default value for the proxy port is 0.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrate how to download the content from URL indicated in kHostname[].

```
#include <HttpClient.h>
#include <WiFi.h>
#include <WiFiClient.h>
char ssid[] = “YourNetwork”; // your network SSID (name)
char pass[] = “password”; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
// Name of the server we want to connect to
const char kHostname[] = “www.google.com”;
const char kPath[] = “/”;
// Number of milliseconds to wait without receiving any data before we give up
const int kNetworkTimeout = 30*1000;
// Number of milliseconds to wait if no data is available before trying again
const int kNetworkDelay = 1000;
int status = WL_IDLE_STATUS;
void setup() {
Serial.begin(9600);
while ( status != WL_CONNECTED) {
Serial.print(“Attempting to connect to SSID: “);
Serial.println(ssid);
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println(“Connected to wifi”);
printWifiStatus();
}
void loop() {
```

```
int err =0;
WiFiClient c;
HttpClient http(c);
err = http.get(kHostname, kPath);
if (err == 0)
{
Serial.println("startedRequest ok");
err = http.responseStatusCode();
if (err >= 0)
{
Serial.print("Got status code: ");
Serial.println(err);
// Usually you'd check that the response code is 200 or a
// similar "success" code (200-299) before carrying on,
// but we'll print out whatever response we get
err = http.skipResponseHeaders();
if (err >= 0)
{
int bodyLen = http.contentLength();
Serial.print("Content length is: ");
Serial.println(bodyLen);
Serial.println();
Serial.println("Body returned follows:");
// Now we've got to the body, so we can print it out
unsigned long timeoutStart = millis();
char c;
// Whilst we haven't timed out & haven't reached the end of the body
while ( (http.connected() || http.available()) &&
((millis() - timeoutStart) < kNetworkTimeout) )
{
if (http.available())
{
c = http.read();
// Print out this character
Serial.print(c);
bodyLen--;
```

```
// We read something, reset the timeout counter
timeoutStart = millis();
}
else
{
// We haven't got any data, so let's pause to allow some to arrive
delay(kNetworkDelay);
}
}
}
else
{
Serial.print("Failed to skip response headers: ");
Serial.println(err);
}
}
else
{
Serial.print("Getting response failed: ");
Serial.println(err);
}
}
else
{
Serial.print("Connect failed: ");
Serial.println(err);
}
http.stop();
// And just stop, now that we've tried a download
while(1);
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
```

```
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::beginRequest

Description
Start a more complex request. Use this when you need to send additional headers in the request, but you will also need to call endRequest() when you are finished.

Syntax
void HttpClient::beginRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::endRequest

Description

End a more complex request. Use this when you need to have sent additional headers in the request, but you will also need to call beginRequest() at the start.

Syntax

void HttpClient::endRequest(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::get

Description

Connect to the server and start to send a "GET" request. If the input parameter contains "aServerAddress", the connection will not perform a DNS lookup and just purely connect to the given IP address.

Syntax

int HttpClient::get(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: The name of the server being connected to. If aServerName is "NULL", the "Host" header line will not be sent.

aServerPort: The port on which server connected.

aURLPath: The URL to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::post

Description

Connect to the server and start to send a "POST" request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::post(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::put

Description

Connect to the server and start to send a PUT request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::put(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::startRequest

Description

Connect to the server and start to send the request.

Syntax

int HttpClient::startRequest(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

int HttpClient::startRequest(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

Parameters

aServerAddress: IP address of the server to connect to.

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aHttpMethod: Type of HTTP request to make, e.g. "GET", "POST", etc.

aUserAgent: User-Agent string to send. If NULL the default user-agent kUserAgent will be sent.

Returns

Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::sendHeader

Description

The function sends an additional header line.

The function void HttpClient:: sendHeader(const char* aHeader);can only be called in between the calls to startRequest and finishRequest.

The other 2 functions void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue); and void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue); are alternate form the previous one, which takes the header name and content as separately (as strings or integer). For example, to send an XXXXXX header, user might call sendHeader("XXXXX", "Something") or sendHeader("XXXXX", 123).And the call will add the ": " in the log to separate different header in the case of multiple headers.

Syntax

void HttpClient::sendHeader(const char* aHeader);

void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue);

void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue);

Parameters
aHeader: Header line to send, in its entirety (but without the trailing CRLF. E.g. "Authorization: Basic YQDDCAIGES".
aHeaderName: Type of header being sent.
aHeaderValue: Value for that header.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::sendBasicAuth

Description
The function sends a basic authentication header which will encode the given username and password, and send them in a suitable header line for doing Basic Authentication.

Syntax
void HttpClient::sendBasicAuth(const char* aUser, const char* aPassword);

Parameters
aUser: Username for the authorization.
aPassword: Password for the user aUser.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include “HttpClient.h” to use the class function.

HttpClient::finishRequest

Description
Finish sending the HTTP request. The function sends a blank line to signify the end of the request.

Syntax
void HttpClient::finishRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::responseStatusCode

Description
Get the HTTP status code contained in the response. For example, “200” for successful requests, “404” for file not found, etc.

Syntax
int HttpClient::responseStatusCode(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::readHeader

Description

The function reads the next character of the response headers. This functions the same as read() but to be used when reading through the headers which are slightly less efficient. The user might check whether the end of the headers has been reached by calling endOfHeadersReached(), although after that point this will still return data as read() would.

Syntax

int HttpClient::readHeader(void);

Parameters

The function requires no input parameter.

Returns

Return the next character of the response headers.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::skipResponseHeaders

Description

Skip any response headers to get to the body. Use this if you don't want to do any special processing of the headers returned in the response. You can also use it after you've found all of the headers you're interested in, and just want to get on with processing the body.

Syntax

int HttpClient::skipResponseHeaders(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfHeadersReached

Description
Test whether all of the response headers have been consumed.

Syntax
bool HttpClient::endOfHeadersReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now processing the response body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfBodyReached

Description

Test whether the end of the body has been reached. It only works if the Content-Length header was returned by the server.

Syntax
bool HttpClient::endOfBodyReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now at the end of the body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::contentLength

Description
The function returns the length of the body.

Syntax
int HttpClient::contentLength(void);

Parameters
The function requires no input parameter.

Returns
Return Length of the body, in bytes, or kNoContentLengthHeader if no Content-Length header was returned by the server.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

IRDevice

Class HttpClient

IRDevice Class

Description
A class used for managing, sending, and receiving data using IR.

Syntax
class IRDevice

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named IR.	

Public Methods	
IRDevice::getFreq	Get the current IR modulation frequency
IRDevice::begin	Allocate resources and start the IR device with a custom frequency
IRDevice::end	Stop the IR device operations and free up resources
IRDevice::send	Send IR raw data
IRDevice::beginNEC	Allocate resources and start the IR device with a frequency suitable for the NEC protocol
IRDevice::sendNEC	Send data using the NEC protocol
IRDevice::recvNEC	Receive data using the NEC protocol

IRDevice::getFreq

Description
Get the current IR modulation frequency.

Syntax
uint32_t getFreq(void);

Parameters
The function requires no input parameter.

Returns
Currently set IR modulation frequency in Hertz.

Example Code
NA

Notes and Warnings
NA

IRDevice::begin

Description
Allocate resources and start the IR device with a custom frequency.

Syntax
void begin(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode, uint32_t freq);

Parameters
receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX
freq: IR modulation frequency in Hertz

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
IR device can only operate in either transmit or receive mode.

IRDevice::end

Description
Stop the IR device operations and free up resources.

Syntax

void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

IRDevice::send

Description
Send IR raw data.

Syntax
void send(const unsigned int buf[ ] , uint16_t len);

Parameters
buf[ ] : IR raw signals (in us) in an array form.
len: total length of the IR raw signal array.

Returns
The function returns nothing.

Example Code

```
#include "IRDevice.h"

// User defined txPin, rxPin and carrier frequency

#define IR_RX_PIN 8

#define IR_TX_PIN 9

#define CARRIER_FREQ 38000

unsigned int irRawSignal[] = {

9000, 4500, // starting bit
```

560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560, // address 00100000 ? 4

560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~ address 11011111

560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, // data 00010000 ? 8

560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~ data 11101111

560 // stoping bit

};

int DataLen = sizeof(irRawSignal) / sizeof(irRawSignal[0]); // 284/ 4 = 71

void setup()

{

Serial.begin(115200);

IR.begin(IR_RX_PIN, IR_TX_PIN, IR_MODE_TX, CARRIER_FREQ);

}

void loop()

{

IR.send(irRawSignal, DataLen);

Serial.println("Finished Sending NEC Raw Data….");

delay(3000);

}

Notes and Warnings

IR Raw Data array contains information in the form of consecutive microseconds (us). For more details, please refer to: http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html.

IRDevice::beginNEC

Description

Allocate resources and start the IR device with a frequency suitable for the NEC protocol.

Syntax

void beginNEC(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode);

Parameters

receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX

Returns

The function returns nothing.

Example Code
Example: IRRecvNEC

```
#include "IRDevice.h"
uint8_t adr = 0;
uint8_t cmd = 0;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
IR.beginNEC(8, 9, IR_MODE_RX); // configure for NEC IR protocol
}
void loop() {
if (IR.recvNEC(adr, cmd, 1000)) {
Serial.print("Received ");
Serial.print(adr);
Serial.print(cmd);
Serial.println();
} else {
Serial.println("Received nothing, timed out");
}
//IR.end();
}
```

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::sendNEC

Description
Send data using the NEC protocol.

Syntax
void sendNEC(uint8_t adr, uint8_t cmd);

Parameters
adr: 8-bit address to transmit
cmd: 8-bit command to transmit

Returns
The function returns nothing.

Example Code
Example: IRSendNEC

```
#include "IRDevice.h"
uint8_t adr = 0;
uint8_t cmd = 0;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
IR.beginNEC(8, 9, IR_MODE_TX); // configure for NEC IR protocol
}
void loop() {
if (cmd++ >=255) {
adr++;
}
IR.sendNEC(adr, cmd);
Serial.print("Sent ");
Serial.print(adr);
Serial.print(cmd);
Serial.println();
//IR.end(); // Call this method to stop IR device and free up the pins for other uses
}
```

Notes and Warnings

IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::recvNEC

Description
Receive data using the NEC protocol.

Syntax
void recvNEC(uint8_t& adr, uint8_t& cmd uint32_t timeout);

Parameters
adr: variable to store received NEC address
cmd: variable to store received NEC command
timeout: time duration to wait for an incoming transmission

Returns
The function returns "1" if data has been received, returns "0" if no data has been received.

Example Code
Example: IRRecvNEC
Details of the code can be found in the previous section of IRDevice::beginNEC.

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

MDNS

Class HttpClient

MDNSClass Class

Description
A class used for registering and removing MDNS service records.

Syntax
class MDNSClass

Members

Public Constructors	
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named MDNS.	

Public Methods	
MDNSClass::begin	Start MDNS operations
MDNSClass::end	Stop MDNS operations
MDNSClass::registerService	Add a service record
MDNSClass::deregisterService	Remove service record
MDNSClass::updateService	Update service record

MDNSClass::begin

Description
Start MDNS operations to begin responding to MDNS queries.

Syntax
void begin(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE
This example shows how to register Ameba as a service that can be recognized by Arduino IDE. If both of the PC runs Arduino IDE and the Ameba board are connecting to the same local network. Then you can find Ameba in "Tools" -> "Port" -> "Arduino at 192.168.1.238 (Ameba RTL8195A), which means the Arduino IDE find Ameba via mDNS.

#include <WiFi.h>

#include <AmebaMDNS.h>

char ssid[] = "yourNetwork"; // your network SSID (name)

char pass[] = "secretPassword"; // your network password

MDNSService service("MyAmeba", "_arduino._tcp", "local", 5000);

void setup() {

printf("Try to connect to %srn", ssid);

while (WiFi.begin(ssid, pass) != WL_CONNECTED) {

printf(“Failed. Wait 1s and retry…rn”);

delay(1000);

}

printf(“Connected to %srn”, ssid);

service.addTxtRecord(“board”, strlen(“ameba_rtl8195a”), “ameba_rtl8195a”);

service.addTxtRecord(“auth_upload”, strlen(“no”), “no”);

service.addTxtRecord(“tcp_check”, strlen(“no”), “no”);

service.addTxtRecord(“ssh_upload”, strlen(“no”), “no”);

printf(“Start mDNS servicern”);

MDNS.begin();

printf(“register mDNS servicern”);

MDNS.registerService(service);

}

void loop() {

// put your main code here, to run repeatedly:

delay(1000);

}

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::end

Description
Stop MDNS operations and stop responding to MDNS queries.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::registerService

Description
Add a service record to be included in MDNS responses.

Syntax
void register service(MDNSService service);

Parameters
service: MDNSService class object with required MDNS service data

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::deregisterService

Description
Remove a service record from MDNS responses.

Syntax
void deregisterService(MDNSService service);

Parameters
service: MDNSService class object to be removed

Returns
The function returns nothing.

Example Code

Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::updateService

Description
Update a service record.

Syntax
void updateService(MDNSService service, unsigned int ttl);

Parameters
service: MDNSService class object to be updated
ttl: time-to-live(TTL) for service

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

Class HttpClient

MDNSService Class

Description
A class used for creating MDNS service records.

Syntax
class MDNSService

Members

Public Constructors	
MDNSService::MDNSService	Create a MDNS service record
Public Methods	
MDNSService::addTxtRecord	Add text to MDNS service record

MDNSService::MDNSService

Description
Create a MDNS service record.

Syntax
MDNSService(char* name, char* service_type, char* domain, unsigned short port, int bufsize);

Parameters
name: device name
service_type: MDNS service type
domain: host domain
port: network port
bufsize: size of buffer for MDNS text record

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSService::addTxtRecord

Description
Add text to MDNS service record.

Syntax
int addTextRecord(char* key, int value_len, char* value);

Parameters
key: record type expressed as character string
value_len: length of value string

value: record value expressed as character string

Returns
0 if add record successful

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MQTTClient

Class PMUClass

PubSubClient Class

Description
Defines a class of MQTT implementation for Ameba.

Syntax
class PubSubClient

Members

Public Constructors	
PubSubClient::PubSubClient	Constructs a PubSubClient object
Public Methods	
PubSubClient::setServer	Set MQTT server address and port
PubSubClient::setCallback	Set callback function
PubSubClient::setClient	Set WiFi client
PubSubClient::setStream	Set data stream
PubSubClient::connect	Attempt to connect to server
PubSubClient::disconnect	Disconnect from current session
PubSubClient::publish	Publish a message to server
PubSubClient::publish_P	Same as above
PubSubClient::subscribe	Subscribe to a topic
PubSubClient::unsubscribe	Unsubscribe to a topic
PubSubClient::loop	Keep MQTT session alive and process any queuing tasks
PubSubClient::connected	Check if client still connected
PubSubClient::state	Return connection state

PubSubClient::PubSubClient

Description
Constructs a PubSubClient object and, if applicable, sets server address, port, callback function, data stream and wifi client.

Syntax
PubSubClient::PubSubClient();
PubSubClient::PubSubClient(Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);

Parameters
client: the network client to use, for example WiFiClient
IPAddress: MQTT server address
port: port for MQTT, usually 1883 for unencrypted connection
MQTT_CALLBACK_SIGNATURE: callback function for MQTT
Stream: a stream to write received messages to

Returns
The function returns nothing.

Example Code
Example: MQTT_Basic

#include <WiFi.h>

#include <PubSubClient.h>

// Update these with values suitable for your network.

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int status = WL_IDLE_STATUS; // the Wifi radio’s status

char mqttServer[] = “test.mosquitto.org”;

```
char clientId[] = "amebaClient";
char publishTopic[] = "outTopic";
char publishPayload[] = "hello world";
char subscribeTopic[] = "inTopic";
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i=0;i<length;i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}
WiFiClient wifiClient;
PubSubClient client(wifiClient);
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection…");
// Attempt to connect
if (client.connect(clientId)) {
Serial.println("connected");
// Once connected, publish an announcement…
client.publish(publishTopic, publishPayload);
// … and resubscribe
client.subscribe(subscribeTopic);
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}
void setup()
```

```
{
Serial.begin(38400);
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
client.setServer(mqttServer, 1883);
client.setCallback(callback);
// Allow the hardware to sort itself out
delay(1500);
}
void loop()
{
if (!client.connected()) {
reconnect();
}
client.loop();
}
```

Notes and Warnings
PubSubClient::PubSubClient(Client& client) would suffice for normal MQTT connection

PubSubClient::setServer

Description
Sets the server details.

Syntax
PubSubClient& PubSubClient::setServer(uint8_t * ip, uint16_t port)
PubSubClient& PubSubClient::setServer(IPAddress ip, uint16_t port)
PubSubClient& PubSubClient::setServer(const char * domain, uint16_t port)

Parameters

ip: the address of the server
port: the port to connect to, default 1883
domain: the address of the server

Returns
The client instance, allowing the function to be chained

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setCallback

Description
Sets the message callback function.

Syntax
PubSubClient& PubSubClient::setCallback(MQTT_CALLBACK_SIGNATURE)

Parameters
MQTT_CALLBACK_SIGNATURE: a pointer to a message callback function called when a message arrives for a subscription created by this client.

Returns
The client instance, allowing the function to be chained.

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setClient

Description
Sets the network client instance to use.

Syntax
PubSubClient& PubSubClient::setClient(Client& client)

Parameters
client: the network client to use, for example WiFiClient

Returns
The client instance, allowing the function to be chained

Example Code
NA

Notes and Warnings
NA

PubSubClient::setStream

Description
Sets the stream to write received messages to.

Syntax
PubSubClient& PubSubClient::setStream(Stream& stream)

Parameters
stream: a stream to write received messages to

Returns
The client instance, allowing the function to be chained.

Example Code
NA

Notes and Warnings
NA

PubSubClient::connect

Description
Connects the client to the server.

Syntax

boolean PubSubClient::connect(const char *id)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass)

boolean PubSubClient::connect(const char *id, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

Parameters

id: Client ID, a unique string identifier

user: Username for authentication, default NULL

pass: Password for authentication, default NULL

willTopic: the topic to be used by the will message

willQoS: the quality of service to be used by the will message

willRetain: whether the will should be published with the retain flag

willMessage: the payload of the will message

Returns

True – connection succeeded

False – connection failed

Example Code

Example: MQTT_Basic

Notes and Warnings

Client ID is required and should always be unique else connection might be rejected by the server.

PubSubClient::disconnect

Description

Disconnect the client

Syntax

void PubSubClient::disconnect(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PubSubClient::publish

Description
Publishes a message to the specified topic.

Syntax
boolean PubSubClient::publish(const char* topic, const char* payload)
boolean PubSubClient::publish(const char* topic, const char* payload, boolean retained)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
Default max packet size is 128 bytes.

PubSubClient::publish_P

Description
Publishes a message stored in PROGMEM to the specified topic.

Syntax
boolean PubSubClient::publish_P(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::subscribe

Description
Subscribes to messages published to the specified topic.

Syntax
boolean PubSubClient::subscribe(const char* topic)
boolean PubSubClient::subscribe(const char* topic, uint8_t qos)

Parameters
topic: the topic to subscribe to
qos: the qos to subscribe at

Returns
False – sending the subscribe failed, either connection lost or message too large
True – sending the subscribe succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::unsubscribe

Description
Unsubscribes from the specified topic.

Syntax
boolean PubSubClient::unsubscribe(const char* topic)

Parameters
topic: the topic to unsubscribe to

Returns
False – sending the unsubscribe failed, either connection lost or message too large
True – sending the unsubscribe succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::loop

Description
A must method called regularly to allow the client to process incoming messages and maintain its connection to the server.

Syntax
boolean PubSubClient::loop(void)

Parameters
The function requires no input parameter.

Returns
False – the client is no longer connected
True – the client is still connected

Example Code
Example: MQTT_Basic

Notes and Warnings
A required method that should not be blocked for too long.

PubSubClient::connected

Description
Checks whether the client is connected to the server.

Syntax
boolean PubSubClient::connected(void)

Parameters
The function requires no input parameter.

Returns
False – the client is not connected
True – the client is connected

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::state

Description
Returns the current state of the client. If a connection attempt fails, this can be used to get more information about the failure.
All of the values have corresponding constants defined in PubSubClient.h.

Syntax
int PubSubClient::state(void)

Parameters
The function requires no input parameter.

Returns
-4 : MQTT_CONNECTION_TIMEOUT – the server didn't respond within the keepalive time
-3 : MQTT_CONNECTION_LOST – the network connection was broken
-2 : MQTT_CONNECT_FAILED – the network connection failed
-1 : MQTT_DISCONNECTED – the client is disconnected cleanly
0 : MQTT_CONNECTED – the client is connected
1 : MQTT_CONNECT_BAD_PROTOCOL – the server doesn't support the requested version of MQTT
2 : MQTT_CONNECT_BAD_CLIENT_ID – the server rejected the client identifier
3 : MQTT_CONNECT_UNAVAILABLE – the server was unable to accept the connection
4 : MQTT_CONNECT_BAD_CREDENTIALS – the username/password were rejected
5 : MQTT_CONNECT_UNAUTHORIZED – the client was not authorized to connect

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

Readme

The Ameba MQTT related APIs and examples are works based on the PubSubClient libraries written by Nicholas O'Leary
These include,

- `PubSubClient.cpp`
- `PubSubClient.h`

These libraries are under MIT License.

NTPClient

Readme

The NTPClient library is based on the NTPClient library written by Fabrice Weinberg, which can be found at https://github.com/arduino-libraries/NTPClient.
These include,

- `NTPClient.cpp`
- `NTPClient.h`

These libraries are licensed under MIT License.

PowerSave

Class PMUClass

PMUClass Class

Description
Defines a class of using Power Save API

Syntax
class PMUClass

Members

Public Constructors	
PMUClass::PMUClass	Constructs a PMUClass object
Public Methods	
PMUCLASS::begin	Initialize the PMUCLASS and select sleep mode
PMUCLASS::AONTimerDuration	Set the duration of AON Timer
PMUCLASS::AONTimerCmd	Disable the AON Timer for power save usage
PMUCLASS::RTCWakeSetup	Set up RTC Timer for power save usage
PMUCLASS::enable	Enable power save deep sleep mode
PMUCLASS::AONWakeReason	Check AON wakeup source
PMUCLASS::WakePinCheck	Check AON GPIO pin wakeup source
PMUCLASS::AONWakeClear	Clear all the AON wakeup source
PMUCLASS::DsleepWakeStatusGet	Check if deepsleep mode is set
PMUCLASS::TL_sysactive_time	Tickless mode system active time
PMUCLASS::TL_wakelock	Tickless mode wake lock, select acquire of release
PMUCLASS::DS_AON_TIMER_WAKEUP	Return the Wakeup source
PMUCLASS::DS_RTC_WAKEUP	Return the Wakeup source
PMUCLASS::TL_UART_WAKEUP	Return the Wakeup source
PMUCLASS::TL_RTC_WAKEUP	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA12	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA13	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA14	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA15	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA16	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA17	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA18	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA19	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA20	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA21	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA25	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA26	Return the Wakeup source

PMUCLASS::PMUCLASS

Description
Constructs a PMUCLASS object.

Syntax
PMUCLASS::PMUCLASS(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::begin

Description
Initialize the PMUCLASS and select sleep mode.

Syntax
void PMUClass::begin(uint32_t sleep_mode);

Parameters
sleep_mode: Selection value, “11” enters the DeepSleep Mode, “22” enters the Tickless Mode

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerDuration

Description
Set the duration of AON Timer

Syntax
void PMUClass::AONTimerDuration(uint32_t duration_ms);

Parameters
duration_ms: Timer duration between 0 to 32760000ms.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerCmd

Description
Disable the AON timer for power save usage.

Syntax
void PMUClass::AONTimerCmd(void);

Parameters
c: The value in Celcius.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::RTCWakeSetup

Description
Set up the RTC timer for power save usage.

Syntax
void PMUClass::RTCWakeSetu(uint32_t duration_d, unit32_t duration_h, uint32_t duration_m, uint32_t duration_s);

Parameters
duration_d: Set alarm for number of days from 0.
duration_h: Set alarm for number of hours from 0.
duration_m: Set alarm for number of minutes from 0.
duration_s: Set alarm for number of seconds from0.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::enable

Description
Enable power save deep sleep mode

Syntax
void PMUClass::enable(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeReason

Description
Check the AON wakeup source

Syntax
uint32_t PMUClass::AONWakeReason(void);

Parameters
The function requires no input parameter.

Returns
Returns the value of wakeup deepsleep source. “11” for AON pin, “22” for AON timer, “33” for RTC timer and “0” for none.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::WakePinCheck

Description
Check which AON GPIO pins are the wakeup source

Syntax
int PMUClass::WakePinCheck(void);

Parameters
The function requires no input parameter.

Returns
Return the pin number for indicating Arduino pin names.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeClear

Description
Clear all AON Wakeup source.

Syntax
void PMUClass::AONWakeClear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DsleepWakeStatusGet

Description
Check if deepsleep mode is set.

Syntax
bool PMUClass::DsleepWakeStatusGet(void);

Parameters
The function requires no input parameter.

Returns
Return TRUE when enter DeepSleep Mode or FALSE for negative.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_sysactive_time

Description
Tickless mode system active time.

Syntax
void PMUClass::TL_sysactive_time(uint32_t duration_ms);

Parameters
duration_ms: Set the duration of system active time. The unit is in milliseconds.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_wakelock

Description
Tickless mode wake lock, select acquire or release.

Syntax
void PMUClass::TL_wakelock(uint32_t select_lock);

Parameters
select_lock: Wake lock selection value, “1” for acquire or “0” for release.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_AON_TIMER_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_AON_TIMER_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON Timer as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_RTC_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_UART_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_UART_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns LOGUART as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_RTC_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA12

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA12(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA12 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA13

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA13(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA13 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA14

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA14(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA14 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA15

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA15(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA15 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA16

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA16(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA16 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA17

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA17(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA17 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA18

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA18(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA18 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA19

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA19(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA19 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA20

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA20(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA20 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA21

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA21(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA21 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA25

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA25(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA25 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA26

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA26(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA26 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

RTC

Class RTC

RTC Class

Description
A class used for displaying date and time and alarm configuration using RTC, the independent BCD (Binary-Coded-Decimal) timer.

Syntax
class RTC

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named RTC.	

Public Methods	
RTC:: Init	Initializes the RTC device, including the Clock, the RTC registers, and other functions
RTC:: DeInit	Deinitialize the RTC device
RTC:: Write	Set the specified timestamp in seconds to RTC
RTC:: Read	Get the current timestamp in seconds from RTC
RTC:: Wait	Wait for 1 second
RTC:: SetEpoch	Convert human-readable time to epoch time

RTC::Init

Description
Initializes the RTC device, including the Clock, the RTC registers, and other functions.

Syntax
void RTC::Init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC

```
/*
* This function describes how to use the RTC API.
* The RTC function is implemented by an independent BCD timer/counter.
* This example will print out the time information every second.
*/
#include <stdio.h>
#include <time.h>
#include "rtc.h"
#define YEAR 2020
#define MONTH 9
#define DAY 10
#define HOUR 20
#define MIN 30
#define SEC 40
/* Create an rtc object */
RTC rtc;
int32_t seconds;
struct tm *timeinfo;
void setup() {
Serial.begin(115200);
rtc.Init(); // initialize RTC
}
void loop() {
// step 1: convert user time to epoch
int epochTime = humanReadableToEpoch(YEAR, MONTH, DAY, HOUR, MIN, SEC);
// step 2: write epoch time to rtc
rtc.Write(epochTime);
while (1) {
seconds = rtc.Read();
printf("Epoch Time (in s) since January 1, 1970 = %dsn", seconds);
printf("Time as a basic string = %s", ctime(&seconds));
```

```
timeinfo = localtime(&seconds);
printf("Time as a custom formatted string = %d-%d-%d %d:%d:%dn",
(timeinfo->tm_year + 1900), (timeinfo->tm_mon + 1), timeinfo->tm_mday, timeinfo->tm_hour,
timeinfo->tm_min, timeinfo->tm_sec);
Serial.println();
rtc.wait(1);
}
}
// convert human readable time to epoch time
int humanReadableToEpoch(int year, int month, int day, int hour, int min, int sec) {
struct tm t;
time_t t_of_day;
t.tm_year = year - 1900; // Year - 1970
t.tm_mon = month - 1; // Month, where 0 = jan
t.tm_mday = day; // Day of the month
t.tm_hour = hour;
t.tm_min = min;
t.tm_sec = sec;
t.tm_isdst = -1; // Is DST on? 1 = yes, 0 = no, -1 = unknown
t_of_day = mktime(&t);
// printf("seconds since the Epoch: %dn", (long)t_of_day);
return t_of_day;
}
```

Notes and Warnings
NA

RTC::DeInit

Description
Deinitializes the RTC device.

Syntax
void RTC::DeInit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Write

Description
Set the specified timestamp in seconds to RTC. Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Syntax
void RTC::Write(int t);

Parameters
Parameters
t: Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC::Read

Description
Get the current timestamp in seconds from RTC. The current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Syntax
int32_t RTC::Read(void);

Parameters
The function requires no input parameter.

Returns
The function returns the current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Wait

Description
Send IR raw data.

Syntax
void RTC::wait(float s);

Parameters
s: unit microseconds (1 us)

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: SetEpoch

Description
Convert human-readable time to epoch time

Syntax
int RTC:: SetEpoch(int year, int month, int day, int hour, int min, int sec);

Parameters
year: user input year
month: user input month
day: user input day
hour: user input hour
min: user input minutes
sec: user input seconds

Returns
The function returns epoch time in seconds for RTC use.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

SoftwareSerial

Class Adafruit_GPS

Adafruit_GPS Class

Description
Defines a class to use GPS module on Ameba.

Syntax
class Adafruit_GPS

Members

Public Constructors	
Adafruit_GPS::Adafruit_GPS	Constructs an Adafruit_GPS object
Public Methods	
Adafruit_GPS::begin	Initialize serial communication
*Adafruit_GPS:: lastNMEA	Returns the last NMEA line received and unsets the received flag
Adafruit_GPS:: newNMEAreceived	Check to see if a new NMEA line has been received
Adafruit_GPS:: common_init	Initialization code used by all constructor types
Adafruit_GPS:: sendCommand	Send a command to the GPS device
Adafruit_GPS:: pause	Pause/unpause receiving new data
Adafruit_GPS:: parseHex	Read a Hex value and return the decimal equivalent
Adafruit_GPS:: read	Read one character from the GPS device
Adafruit_GPS:: parse	Parse an NMEA string
Adafruit_GPS:: wakeup	Wake the sensor up
Adafruit_GPS:: standby	Standby Mode Switches
Adafruit_GPS::waitForSentence	Wait for a specified sentence from the device
Adafruit_GPS::LOCUS_StartLogger	Start the LOCUS logger
Adafruit_GPS::LOCUS_StopLogger	Stop the LOCUS logger
Adafruit_GPS::LOCUS_ReadStatus	Read the logger status

Adafruit_GPS::Adafruit_GPS

Description

Constructs an Adafruit_GPS object and initialize serial using a SoftSerial object.

Syntax

Adafruit_GPS::Adafruit_GPS(SoftwareSerial *ser)

Adafruit_GPS::Adafruit_GPS(HardwareSerial *ser)

Parameters

ser: a Serial instance

Returns

The function returns nothing.

Example Code

Example: Adafruit_GPS_parsing

This example code from Adafruit demonstrates GPS modules using MTK3329/MTK3339 driver. This code shows how to listen to the GPS module in an interrupt which allows the program to have more 'freedom' – just parse when a new NMEA sentence is available! Then access data when desired.

#include <Adafruit_GPS.h>

#include <SoftwareSerial.h>

// If you're using a GPS module:

// Connect the GPS Power pin to 3.3V

```
// Connect the GPS Ground pin to ground
// Connect the GPS TX (transmit) pin to Digital 0
// Connect the GPS RX (receive) pin to Digital 1
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1);
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RTL8710 need change GPS TX/RX to pin 17 and 5
#else
SoftwareSerial mySerial(0, 1);
#endif
Adafruit_GPS GPS(&mySerial);
// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences.
#define GPSECHO false
void setup()
{
Serial.begin(38400);
Serial.println("Adafruit GPS library basic test!");
// 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
GPS.begin(9600);
// uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitude
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
// uncomment this line to turn on only the "minimum recommended" data
//GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
// For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
// the parser doesn't care about other sentences at this time
// Set the update rate
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
// For the parsing code to work nicely and have time to sort thru the data, and
// print it out we don't suggest using anything higher than 1 Hz
// Request updates on antenna status, comment out to keep quiet
GPS.sendCommand(PGCMD_ANTENNA);
delay(1000);
// Ask for firmware version
mySerial.println(PMTK_Q_RELEASE);
}
```

```
uint32_t timer = millis();
void loop() // run over and over again
{
// in case you are not using the interrupt above, you'll
// need to 'hand query' the GPS, not suggested :(
// read data from the GPS in the 'main loop'
char c = GPS.read();
// if you want to debug, this is a good time to do it!
if (GPSECHO)
if (c) Serial.print(c);
// if a sentence is received, we can check the checksum, parse it…
if (GPS.newNMEAreceived()) {
// a tricky thing here is if we print the NMEA sentence, or data
// we end up not listening and catching other sentences!
// so be very wary if using OUTPUT_ALLDATA and trytng to print out data
//Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false
if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to false
return; // we can fail to parse a sentence in which case we should just wait for another
}
// if millis() or timer wraps around, we'll just reset it
if (timer > millis()) timer = millis();
// approximately every 2 seconds or so, print out the current stats
if (millis() - timer > 2000) {
timer = millis(); // reset the timer
Serial.print("nTime: ");
Serial.print(GPS.hour, DEC); Serial.print(':');
Serial.print(GPS.minute, DEC); Serial.print(':');
Serial.print(GPS.seconds, DEC); Serial.print('.');
Serial.println(GPS.milliseconds);
Serial.print("Date: ");
Serial.print(GPS.day, DEC); Serial.print('/');
Serial.print(GPS.month, DEC); Serial.print("/20");
Serial.println(GPS.year, DEC);
Serial.print("Fix: "); Serial.print((int)GPS.fix);
Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
if (GPS.fix) {
```

Serial.print("Location: ");

Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);

Serial.print(", ");

Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);

Serial.print("Location (in degrees, works with Google Maps): ");

Serial.print(GPS.latitudeDegrees, 4);

Serial.print(", ");

Serial.println(GPS.longitudeDegrees, 4);

Serial.print("Speed (knots): "); Serial.println(GPS.speed);

Serial.print("Angle: "); Serial.println(GPS.angle);

Serial.print("Altitude: "); Serial.println(GPS.altitude);

Serial.print("Satellites: "); Serial.println((**int**)GPS.satellites);

}

}

}

Notes and Warnings
IMPORTANT: SoftSerial is using hardware serial so pin mapping cannot be altered.

Adafruit_GPS::begin

Description
Initialize serial communication

Syntax
void Adafruit_GPS::begin(uint16_t baud)

Parameters
baud: serial baud rate

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

***Adafruit_GPS::lastNMEA**

Description
Returns the last NMEA line received and unsets the received flag

Syntax
char *Adafruit_GPS::lastNMEA(void)

Parameters
The function requires no input parameter.

Returns
Pointer to the last line string

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::newNMEAreceived

Description
Check to see if a new NMEA line has been received

Syntax
boolean Adafruit_GPS::newNMEAreceived(void)

Parameters
The function requires no input parameter.

Returns
True if received, false if not

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::common_init

Description
Initialization code used by all constructor types

Syntax
void Adafruit_GPS::common_init(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::sendCommand

Description
Send a command to the GPS device

Syntax
void Adafruit_GPS::sendCommand(const char *str)

Parameters
str: Pointer to a string holding the command to send

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::pause

Description
Pause/unpause receiving new data

Syntax
void Adafruit_GPS::pause(boolean p)

Parameters
p: True = pause, false = unpause

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::parseHex

Description
Read a Hex value and return the decimal equivalent

Syntax
uint8_t Adafruit_GPS::parseHex(char c)

Parameters
c: Hex value

Returns
The decimal equivalent of the Hex value

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::read

Description
Read one character from the GPS device

Syntax
char Adafruit_GPS::read(void)

Parameters
The function requires no input parameter.

Returns
The character that we received, or 0 if nothing was available

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::parse

Description
Parse an NMEA string

Syntax
boolean Adafruit_GPS::parse(char *nmea)

Parameters
nmea: an NMEA string

Returns
True if we parsed it, false if it has invalid data

Example Code
Example: Adafruit_GPS_parsing

Notes and Warnings
NA

Adafruit_GPS::wakeup

Description
Wake the sensor up

Syntax
boolean Adafruit_GPS::wakeup(void)

Parameters
The function requires no input parameter.

Returns
True if woken up, false if not in standby or failed to wake

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::standby

Description
Standby Mode Switches

Syntax
boolean Adafruit_GPS::standby(void)

Parameters
The function requires no input parameter.

Returns
False if already in standby, true if it entered standby

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::waitForSentence

Description
Wait for a specified sentence from the device

Syntax
boolean Adafruit_GPS::waitForSentence(const char *wait4me, uint8_t max)

Parameters
wait4me: Pointer to a string holding the desired response
max: How long to wait, default is MAXWAITSENTENCE

Returns
True if we got what we wanted, false otherwise

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StartLogger

Description
Start the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StartLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StopLogger

Description
Stop the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StopLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings

NA

Adafruit_GPS::LOCUS_ReadStatus

Description
Read the logger status

Syntax
boolean Adafruit_GPS::LOCUS_ReadStatus(void)

Parameters
The function requires no input parameter.

Returns
True if we read the data, false if there was no response

Example Code
NA

Notes and Warnings
NA

Class HttpClient

PMS3003 Class

Description
Defines a class to work with PMS3003 air quality sensor on Ameba.

Syntax
class PMS3003

Members

Public Constructors	
PMS3003::PMS3003	Constructs a PMS3003 object
Public Methods	
PMS3003::begin	Initialize hardware UART
PMS3003::end	Free allocated space thus stopping UART
PMS3003::get_pm1p0_cf1	Get PM1.0 under correction factor = 1
PMS3003:: get_pm2p5_cf1	Get PM2.5 under correction factor = 1
PMS3003:: get_pm10_cf1	Get PM10 under correction factor = 1
PMS3003:: get_pm1p0_air	Get PM1.0 air quality
PMS3003:: get_pm2p5_air	Get PM2.5 air quality
PMS3003:: get_pm10_air	Get PM10 air quality
PMS3003:update_cache	Updates the cache memory
PMS3003::pms3003_handle_interrupt	Set up the serial event handler

PMS3003::PMS3003

Description
Constructs a PMS3003 object and initialize the pin mapping.

Syntax
PMS3003::PMS3003(int _rx, int _tx, int _set, int _reset)

Parameters
_rx: RX pin of UART
_tx: TX pin of UART
_set: Set pin
_reset: Reset pin

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::begin

Description
Initialize hardware UART and allocate space for serial buffer

Syntax
void PMS3003::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::end

Description
Free serial buffer space and stop UART

Syntax
void PMS3003::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_cf1

Description
Get PM1.0 under correction factor = 1

Syntax
int PMS3003::get_pm1p0_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value “pm1p0_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm2p5_cf1

Description
Get PM2.5 under correction factor = 1

Syntax
int PMS3003::get_pm2p5_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_cf1

Description
Get PM10 under correction factor = 1

Syntax
int PMS3003::get_pm10_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_air

Description
Get PM1.0 air quality

Syntax
int PMS3003::get_pm1p0_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm1p0_air” as an integer.

Example Code
NA

Notes and Warnings

NA

PMS3003::get_pm2p5_air

Description
Get PM2.5 air quality

Syntax
int PMS3003::get_pm2p5_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_air” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_air

Description
Get PM10 air quality

Syntax
int PMS3003::get_pm10_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_air” as an integer.

Example Code

NA

Notes and Warnings
NA

PMS3003::pms3003_handle_interrupt

Description
Set up the serial event handler

Syntax
void pms3003_handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: device identifier
event: Serial event for handling incoming data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::update_cache

Description
Serves the function of updating cache memory. One package has 32 bytes. Illustrate the formate by using below raw data: 42 4d 00 1c 00 1b 00 21 00 29 00 1a 00 21 00 29 2b fb 04 be 00 6b 00 10 00 04 00 04 67 00 04 46
42 4d : header signature
00 1c : frame length, 0x001c = 28 bytes (not include header and this field)
00 1b : PM1.0 under CF=1
00 21 : PM2.5 under CF=1
00 29 : PM10 under CF=1
00 1a : PM1.0 under air
00 21 : PM2.5 under air
00 29 : PM10 under air

2b fb : number of pariticle, diameter size 0.3 um in 0.1 liter air
04 be : number of pariticle, diameter size 0.5 um in 0.1 liter air
00 6b : number of pariticle, diameter size 1.0 um in 0.1 liter air
00 10 : number of pariticle, diameter size 2.5 um in 0.1 liter air
00 04 : number of pariticle, diameter size 5.0 um in 0.1 liter air
00 04 : number of pariticle, diameter size 10 um in 0.1 liter air
67 : serial number
00 : error code
04 46 :
checksum,0x42+0x4d+0x00+0x1c+0x00+0x1b+0x00+0x21+0x00+0x29+0x00+0x1a+0x00+0x21+0x00+0x29+
0x2b+0xfb+0x04+0xbe+0x00+0x6b+0x00+0x10+0x00+0x04+0x00+0x04+0x67+0x00 = 0x0446

Syntax
void PMS3003::update_cache(void)

Parameters
The function requires no input parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Class SoftwareSerial

SoftwareSerial Class

Description
Defines a class of software serial implementation for Ameba.

Syntax
class SoftwareSerial

Members

Public Constructors	
SoftwareSerial::SoftwareSerial	Constructs a SoftwareSerial object
Public Methods	
SoftwareSerial::begin	Sets the speed (baud rate) for the serial communication
SoftwareSerial::listen	Enables the selected software serial port to listen
SoftwareSerial::end	Same as stopListening
SoftwareSerial::stopListening	Stop listening on the port
SoftwareSerial::peek	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::write	Prints data to the transmit pin of the software serial port as raw bytes
SoftwareSerial::read	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::available	Get the number of bytes (characters) available for reading from a software serial port
SoftwareSerial::flush	Flush the received buffer
SoftwareSerial::setBufferSize	Set buffer size
SoftwareSerial::setAvailableCallback	Set available callback
SoftwareSerial::handle_interrupt	Private methods handles interrupt

SoftwareSerial::SoftwareSerial

Description

Constructs a SoftwareSerial object and sets RX and TX pin, and inverse logic.

Syntax

SoftwareSerial::SoftwareSerial(uint8_t receivePin, uint8_t transmitPin, bool inverse_logic /* = false */)

Parameters

receivePin: the pin on which to receive serial data
transmitPin: the pin on which to transmit serial data
inverse_logic: is used to invert the sense of incoming bits

Returns

The function returns nothing.

Example Code

Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX.

/*

The circuit: (BOARD RTL8195A)

* RX is digital pin 0 (connect to TX of other devices)

* TX is digital pin 1 (connect to RX of other devices)

*/

```
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(57600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
Serial.println("Goodnight moon!");
// set the data rate for the SoftwareSerial port
mySerial.begin(4800);
mySerial.println("Hello, world?");
}
void loop() { // run over and over
if (mySerial.available()) {
mySerial.write(mySerial.read());
}
}
```

Notes and Warnings
Software Serial is using hardware serial thus DO NOT change the default pins

SoftwareSerial::begin

Description
Sets the speed (baud rate) for the serial communication

Syntax
void SoftwareSerial::begin(long speed)
void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits)

void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits, int flowctrl, int rtsPin, int ctsPin)

Parameters
speed: the baud rate
data_bits: number of data bits, 8 bits(default) or 7 bits
stop_bits: number of stop bits, 1 bit(default), 1.5 bits or 2 bits
flowctrl: flow control pin
rtsPin: request to send pin
ctsPin: clear to send pin

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial_Basic:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::listen

Description
Enables the selected software serial port to listen

Syntax
bool SoftwareSerial::listen(void)

Parameters
The function requires no input parameter.

Returns
Returns true if it replaces another

Example Code
NA

Notes and Warnings

NA

SoftwareSerial::end

Description
Same as stopListening

Syntax
void SoftwareSerial::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::isListening

Description
Tests to see if requested software serial port is actively listening

Syntax
bool SoftwareSerial::isListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if the port is listening.

Example Code

NA

Notes and Warnings
NA

SoftwareSerial::stopListening

Description
Stop listening on the port

Syntax
bool SoftwareSerial::stopListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if listening on the port is stopped.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::peek

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::peek(void)

Parameters
The function requires no input parameter.

Returns

The function returns the character read, or returns "-1" if none is available.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::write

Description
Prints data to the transmit pin of the software serial port as raw bytes

Syntax
size_t SoftwareSerial::write(uint8_t b)

Parameters
b: byte to be written

Returns
The function returns the number of bytes written.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::read

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::read(void)

Parameters
The function requires no input parameter.

Returns
The function returns the character read, or -1 if none is available.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::available

Description
Get the number of bytes available for reading from a software serial port

Syntax
int SoftwareSerial::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available to read.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::flush

Description
Flush the received buffer

Syntax
void SoftwareSerial::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setBufferSize

Description
Set buffer size

Syntax
void SoftwareSerial::setBufferSize(uint32_t buffer_size)

Parameters
buffer_size: the size of the serial buffer

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setAvailableCallback

Description
Set available callback

Syntax
void SoftwareSerial::setAvailableCallback(void (*callback)(char c))

Parameters
*callback: user-defined serial callback function

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialIrqCallback
This example demonstrates the software serial testing using IRQ callback and semaphore. Set callback function "mySerialCalback" to software serial. Whenever there is data comes in, "mySerialCallback" is invoked. In this sketch, it does nothing until the end of the line. And then it sends a semaphore. The loop() uses a non-busy loop to wait for the semaphore. To test this sketch, you need to type something on software serial and then press Enter.

```
/*
The circuit: (BOARD RTL8195A)
RX is digital pin 0 (connect to TX of other devices)
TX is digital pin 1 (connect to RX of other devices)
*/
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
uint32_t semaID;
// The callback is hooking at UART IRQ handler and please don't do heavy task here.
void mySerialCallback(char c)
{
```

```
/* The parameter c is only for peeking. The actual data is
* still in the buffer of SoftwareSerial.
*/
if (c == 'r' || c == 'n') {
os_semaphore_release(semaID);
}
}
void setup() {
// use 1 count for binary semaphore
semaID = os_semaphore_create(1);
// There is a token in the semaphore, clear it.
os_semaphore_wait(semaID, 0xFFFFFFFF);
// set the data rate for the SoftwareSerial port
mySerial.begin(38400);
mySerial.setAvailableCallback(mySerialCallback);
}
void loop() { // run over and over
// wait semaphore for 5s timeout
if (os_semaphore_wait(semaID, 5 * 1000)) {
// we got data before timeout
while(mySerial.available()) {
mySerial.print((char)mySerial.read());
}
mySerial.println();
} else {
mySerial.println("No data comes in.");
}
}
```

Notes and Warnings
NA

SoftwareSerial::handle_interrupt

Description
A private method handles the interrupt

Syntax
void handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: the interupt id
event: interrupt event

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba Software Serial related APIs and examples are works based on libraries formerly known as `NewSoftSerial.h` by Mikal Hart (http://arduiniana.org/libraries/newsoftserial).
These include,

- `SoftwareSerial.cpp`
- `SoftwareSerial.h`

These libraries are under GNU Lesser General Public License.

The Ameba GPS related APIs and examples are works based on Adafruit GPS library written by Limor Fried/Ladyada for Adafruit Industries (http://www.adafruit.com/products/746).
These include,

- `Adafruit_GPS.cpp`
- `Adafruit_GPS.h`

These libraries are under BSD License.

SPI

Class AmebaILI9341

AmebaILI9341 Class

Description

Defines a class to use ILI9341 TFT SPI display for Ameba.

Syntax

class AmebaILI9341

Members

Public Constructors	
AmebaILI9341::AmebaILI9341	Constructs an AmebaILI9341 object
Public Methods	
AmebaILI9341::begin	Initialize SPI, pin mapping and screen configuration
AmebaILI9341::setAddress	Initialize image size and position
AmebaILI9341::writecommand	SPI transfer a command
AmebaILI9341::writedata	SPI transfer a piece of data
AmebaILI9341::setRotation	Set screen orientation
AmebaILI9341::fillScreen	Fill the screen with a color
AmebaILI9341::clr	Clear screen
AmebaILI9341::fillRectangle	Fill a rectangular space with a color
AmebaILI9341::drawPixel	Turn on a pixel on the screen
AmebaILI9341::drawChar	To print a character on the screen
AmebaILI9341::drawLine	Draw line on the screen
AmebaILI9341::drawRectangle	Draw a rectangle on the screen
AmebaILI9341::drawCircle	Draw a circle on the screen
AmebaILI9341::write	Same as drawChar
AmebaILI9341::getWidth	Return the width 240
AmebaILI9341::getHeight	Return the height 320
AmebaILI9341::setCursor	Set cursor to the desired position
AmebaILI9341::setForeground	Set foreground color
AmebaILI9341::setBackground	Set background color
AmebaILI9341::setFontSize	Set character font size
AmebaILI9341::reset	Reset pin to High or Low

AmebaILI9341::AmebaILI9341

Description

Constructs an AmebaILI9341 object and set CS, DC and RESET pins .

Syntax

AmebaILI9341::AmebaILI9341(int csPin, int dcPin, int resetPin)

Parameters

csPin: pin for Chip Select dcPin: pin for Data/Command resetPin: pin for Reset

Returns

The function returns nothing.

Example Code

Example: : PM25_ON_ILI9341_TFT_LCD

This example demonstrates how to read pm2.5 value on PMS 3003 air-condition sensor and display it on ILI9341 TFT LCD.

/*

PMS 3003 pin map is as follow:

PIN1 :VCC, connect to 5V

PIN2 :GND

PIN3 :SET, 0:Standby mode, 1:operating mode

PIN4 :RXD :Serial RX

PIN5 :TXD :Serial TX

PIN6 :RESET

PIN7 :NC

PIN8 :NC

In this example, we only use Serial to get PM 2.5 value.

The circuit:

* RX is digital pin 0 (connect to TX of PMS 3003)

* TX is digital pin 1 (connect to RX of PMS 3003)

For RTL8195A ILI9341 TFT LCD with SPI interface has these pins:

D/C : connect to pin 9

CS : connect to pin 10

MOSI : connect to pin 11

MISO : connect to pin 12

CLK : connect to pin 13

VCC : connect to 3V3

GND : connect to GND

*/

#include "SoftwareSerial.h"

#include "SPI.h"

#include "AmebaILI9341.h"

#if defined(BOARD_RTL8195A)

SoftwareSerial mySerial(0, 1); // RX, TX

#define TFT_RESET 8

#define TFT_DC 9

#define TFT_CS 10

#elif defined(BOARD_RTL8710)

```
SoftwareSerial mySerial(17, 5); // RX, TX
// IMPORTANT: Due to limit pin, we do not connect TFT_RESET pin.
#define TFT_RESET 0xFFFFFFFF
#define TFT_DC 2
#define TFT_CS 10
#endif
AmebaILI9341 tft = AmebaILI9341(TFT_CS, TFT_DC, TFT_RESET);
#define ILI9341_SPI_FREQUENCY 20000000
#define pmsDataLen 32
uint8_t buf[pmsDataLen];
int idx = 0;
int pm10 = 0;
int last_pm25 = 0;
int pm25 = 0;
int pm100 = 0;
uint16_t pm25color[] = {
0x9FF3,
0x37E0,
0x3660,
0xFFE0,
0xFE60,
0xFCC0,
0xFB2C,
0xF800,
0x9800,
0xC99F
};
void setup() {
Serial.begin(57600);
mySerial.begin(9600); // PMS 3003 UART has baud rate 9600
SPI.setDefaultFrequency(ILI9341_SPI_FREQUENCY);
tft.begin();
drawPictureFrames();
}
void loop() { // run over and over
uint8_t c;
```

```
idx = 0;
memset(buf, 0, pmsDataLen);
while (true) {
while (c != 0x42) {
while (!mySerial.available());
c = mySerial.read();
}
while (!mySerial.available());
c = mySerial.read();
if (c == 0x4d) {
// now we got a correct header)
buf[idx++] = 0x42;
buf[idx++] = 0x4d;
break;
}
}
while (idx != pmsDataLen) {
while(!mySerial.available());
buf[idx++] = mySerial.read();
}
pm10 = ( buf[10] << 8 ) | buf[11];
last_pm25 = pm25;
pm25 = ( buf[12] << 8 ) | buf[13];
pm100 = ( buf[14] << 8 ) | buf[15];
updateValueToTftScreen();
}
void drawPictureFrames() {
tft.setRotation(1);
tft.clr();
tft.setFontSize(1);
// Upper title
tft.setFontSize(1);
tft.setCursor(20,20);
tft.print("PM2.5 DETECTOR");
// PM 2.5 Circle Frame
tft.drawCircle(100,130,60, ILI9341_BLUE);
```

```
tft.drawCircle(100,130,61, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(90,85);
tft.print("PM2.5");
tft.setFontSize(1);
tft.setCursor(90,170);
tft.print("um/m3");
// PM 10 Circle Frame
tft.drawCircle(220,70,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(210,40);
tft.print("PM10");
tft.setFontSize(1);
tft.setCursor(205,95);
tft.print("um/m3");
// PM 1.0 Circle Frame
tft.drawCircle(220,170,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(205,140);
tft.print("PM1.0");
tft.setFontSize(1);
tft.setCursor(205,195);
tft.print("um/m3");
// right side bar, referenced from: http://taqm.epa.gov.tw/taqm/tw/
tft.fillRectangle(290, 30+ 0*2, 10, 12*2, pm25color[0]); // 0~11
tft.fillRectangle(290, 30+12*2, 10, 12*2, pm25color[1]); // 12-23
tft.fillRectangle(290, 30+24*2, 10, 12*2, pm25color[2]); // 24-35
tft.fillRectangle(290, 30+36*2, 10, 6*2, pm25color[3]); // 36-41
tft.fillRectangle(290, 30+42*2, 10, 6*2, pm25color[4]); // 42-47
tft.fillRectangle(290, 30+48*2, 10, 6*2, pm25color[5]); // 48-53
tft.fillRectangle(290, 30+54*2, 10, 6*2, pm25color[6]); // 54-58
tft.fillRectangle(290, 30+59*2, 10, 6*2, pm25color[7]); // 59-64
tft.fillRectangle(290, 30+65*2, 10, 6*2, pm25color[8]); // 65-70
tft.fillRectangle(290, 30+71*2, 10, 10*2, pm25color[9]); // >=71
tft.setCursor(302, 30);
tft.setFontSize(1);
```

```
tft.print("0");
tft.setCursor(302, 30+36*2);
tft.print("36");
tft.setCursor(302, 30+54*2);
tft.print("54");
tft.setCursor(302, 30+71*2);
tft.print("71");
// bottom right text
tft.setCursor(210,230);
tft.setFontSize(1);
tft.print("Powered by Realtek");
updateValueToTftScreen();
}
void updateValueToTftScreen() {
tft.setCursor(60, 111);
tft.setFontSize(5);
tft.setForeground( getPm25Color(pm25) );
if (pm25 < 10) {
tft.print(" ");
} else if (pm25 < 100) {
tft.print(" ");
}
tft.print(pm25);
tft.setCursor(195,60);
tft.setFontSize(3);
if (pm100 < 10) {
tft.print(" ");
} else if (pm100 < 100) {
tft.print(" ");
}
tft.print(pm100);
tft.setCursor(198,160);
if (pm10 < 10) {
tft.print(" ");
} else if (pm10 < 100) {
tft.print(" ");
```

```
}
tft.print(pm10);
tft.setFontSize(1);
tft.setForeground(ILI9341_WHITE);
if (last_pm25 > 80) {
tft.fillRectangle(275, 80*2+30-3, 12, 8, ILI9341_BLACK);
} else {
tft.fillRectangle(275, last_pm25*2+30-3, 12, 8, ILI9341_BLACK);
}
if (pm25 > 80) {
tft.setCursor(275, 80*2+30-3);
} else {
tft.setCursor(275, pm25*2+30-3);
}
tft.print("=>");
}
uint16_t getPm25Color(int v) {
if (v < 12) {
return pm25color[0];
} else if (v < 24) {
return pm25color[1];
} else if (v < 36) {
return pm25color[2];
} else if (v < 42) {
return pm25color[3];
} else if (v < 48) {
return pm25color[4];
} else if (v < 54) {
return pm25color[5];
} else if (v < 59) {
return pm25color[6];
} else if (v < 65) {
return pm25color[7];
} else if (v < 71) {
return pm25color[8];
} else {
```

return pm25color[9];

}

}

Notes and Warnings

NA

AmebaILI9341::begin

Description

Initialize hardware SPI, pin mapping and screen configuration

Syntax

void AmebaILI9341::begin(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

This method is required to run first before other operations on the display.

AmebaILI9341::setAddress

Description

Initialize image size and positioning on the display

Syntax

void AmebaILI9341::setAddress(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: rightmost coordinate of the image y1: bottom coordinate of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Do not use this to set the cursor, use the “setCursor” method instead.

AmebaILI9341::writecommand

Description

Write a single-byte command to display

Syntax

void AmebaILI9341::writecommand(uint8_t command)

Parameters

command: a single byte command

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::writedata

Description

Write 1 byte of data to display

Syntax

void AmebaILI9341::writedata(uint8_t data)

Parameters

data: 1 byte data

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this method to write 1 byte at a time.

AmebaILI9341::setRotation

Description

Setting screen orientation, “0” for no rotation, “1” for 90 degrees rotation and so on so forth.

Syntax

void AmebaILI9341::setRotation(uint8_t m)/span> **Parameters**

m: one of the 4 rotation modes -> “0” for no rotation, “1” for 90^0, “2” for 180^0, “3” for 270^0

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

if m=4, it’s equivalent to mode 0, and m=5 for mode 1, m=6 for mode 2 so on so forth.

AmebaILI9341::fillScreen

Description

Fill the entire screen with one color

Syntax

void AmebaILI9341::fillScreen(uint16_t color)

Parameters

color: a 16-bit color reference defined in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Refer to AmebaILI9341.h for available colors.

AmebaILI9341::clr

Description

Fill the entire screen with a certain background-color

Syntax

void AmebaILI9341::clr(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341

Notes and Warnings

background-color can be set by calling setBackground method.

AmebaILI9341::fillRectangle

Description

Fill a rectangular space with a color on the screen

Syntax

void AmebaILI9341::fillRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::drawPixel

Description

Turn on a pixel on the screen

Syntax

void AmebaILI9341::drawPixel(int16_t x, int16_t y, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawChar

Description

Draw character on the screen

Syntax

void AmebaILI9341::drawChar(unsigned char c) void AmebaILI9341::drawChar(int16_t x, int16_t y, unsigned char c, uint16_t _fontcolor, uint16_t _background, uint8_t _fontsize)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image c: a character _fontcolor: font color _background: background color _fontsize: font size

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In the actual example, the Print method is used to print a string of character on the screen instead of using this method.

AmebaILI9341::drawLine

Description

Draw a straight line on the screen

Syntax

void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1) void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: leftmost coordinate of the image y1: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawRectangle

Description

Draw a rectangular shape on the screen

Syntax

void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h) void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawCircle

Description

Draw a circular shape on the screen

Syntax

void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r) void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image r: radius of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Include “AmebaServo.h” to use the class function.

AmebaILI9341::write

Description

Same as drawChar, write a character on the screen

Syntax

size_t AmebaILI9341::write(uint8_t c)

Parameters

c: a character to be written on the screen

Returns

Number of bytes written

Example Code

NA

Notes and Warnings

This an inherited method from Print class and is seldom used.

AmebaILI9341::getWidth

Description

Get the width of the image

Syntax

int16_t AmebaILI9341::getWidth(void)

Parameters

The function requires no input parameter.

Returns

Width of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::getHeight

Description

Get the height of the image

Syntax

int16_t AmebaILI9341::getHeight(void)

Parameters

The function requires no input parameter.

Returns

Height of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::setCursor

Description

Set the cursor to a specific position on the screen

Syntax

void AmebaILI9341::setCursor(int16_t x, int16_t y)

Parameters

x: coordinate on the x-axis y: coordinate on the y-axis

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setForeground

Description

Set foreground color

Syntax

void AmebaILI9341::setForeground(uint16_t color)

Parameters

color: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setBackground

Description

Set background color

Syntax

void AmebaILI9341::setBackground(uint16_t _background)

Parameters

_background: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setFontSize

Description

Set the font size of the characters printed on the screen.

Syntax

void AmebaILI9341::setFontSize(uint8_t size)

Parameters

size: font size, default 1 for smallest, 5 for largest font size

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::reset

Description

Reset the pin to High or Low

Syntax

void AmebaILI9341::reset(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

Class SPISettings_SPIClass

SPISettings Class

Description
Defines a class to set SPI parameters.

Syntax
class SPISettings

Members

Public Constructors	
SPISettings::SPISettings	Create a SPISettings object and set SPI clock speed, bit order and data mode

SPISettings::SPISettings

Description
Construct an object and configure SPI parameters — clock speed, bit order and data model to the preferred default value.

Syntax
SPISettings YourObject(uint32_t clock, BitOrder bitOrder, uint8_t dataMode);

Parameters
clock: SPI clock speed, default is 4000000
bitOrder: order of bit stream, MSB first or LSB first, default is MSBFIRST
dataMode: There are 4 modes -> SPI_MODE0~3, default is SPI_MODE0

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class seldom used alone, it is always used with beginTransaction() as a parameter in SPIClass.

SPIClass Class

Description
Defines a class of SPI implementation for Ameba.

Syntax
class SPIClass

Members

Public Constructors	
SPIClass::SPIClass	Constructs an SPI object
Public Methods	
SPIClass::transfer	Transfer data through SPI
SPIClass::transfer16	Transfer a 16-bits data through SPI
SPIClass::beginTransaction	Set slave select pin and SPI initial settings
SPIClass::endTransaction	Stop SPI transaction
SPIClass::begin	Associate each SPI pin to Ameba pin using ameba HAL APIs
SPIClass::end	Stop SPI master mode
SPIClass::setBitOrder	Set MSB first or LSB first
SPIClass::setDataMode	Set to one of the four data modes
SPIClass::setClockDivider	Set to correct clock speed (no effect on Ameba)
SPIClass::setDefaultFrequency	Set default SPI frequency

SPIClass::SPIClass

Description
Construct an SPI object, create a pointer to the object, and attach “MOSI, MISO, CLK, and SS” to each pin on Ameba.

Syntax
SPIClass(void *pSpiObj, int mosi, int miso, int clk, int ss);

Parameters
pSpiObj: SPI pointer to the object
mosi: master out slave in
miso: master in slave out
clk: clock
ss: slave select

Returns
The function returns nothing.

Example Code

SPIClass SPI((void *)(&spi_obj0), 11, 12, 13, 10);

Notes and Warnings
2 SPI objects are created in the library for 2 different hardware SPI on Ameba (if applicable), use "SPI" for first hardware SPI and "SPI1" for the second.

SPIClass::transfer

Description
Calling HAL API to send data in the buffer to the slave

Syntax
byte SPIClass::transfer (byte _pin, uint8_t _data, SPITransferMode _mode);
byte SPIClass::transfer (uint8_t _data, SPITransferMode _mode);
void SPIClass::transfer (byte _pin, void *_buf, size_t _count, SPITransferMode _mode);
void SPIClass::transfer (void *_buf, size_t _count, SPITransferMode _mode);

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode
_count: number of bytes of data
_buf: data buffer

Returns
Void or "0" in case of error, "d" in case success

Example Code
NA

Notes and Warnings
NA

SPIClass::transfer16

Description
Same as "transfer" method above except data being of 16-bits.

Syntax
uint16_t SPIClass::transfer16(byte _pin, uint16_t _data, SPITransferMode _mode)
uint16_t SPIClass::transfer16(uint16_t _data, SPITransferMode _mode)

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode

Returns
The data being transferred

Example Code
NA

Notes and Warnings
NA

SPIClass::beginTransaction

Description
Set slave select pin and initialize SPI with default settings using SPISettings class.

Syntax
void SPIClass::beginTransaction(uint8_t pin, SPISettings settings)
void SPIClass::beginTransaction(SPISettings settings)

Parameters
pin: slave select pin
settings: an object of SPISettings class

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to SPISettings class for details of the initial settings.

SPIClass::endTransaction

Description
Set slave select pin to 1 and stop SPI transaction.

Syntax
void SPIClass::endTransaction(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::begin

Description
Calling HAL APIs to initialize SPI pins to physical Ameba pins and set SPI format and frequency

Syntax
void SPIClass::begin(void)
void SPIClass::begin(int ss)

Parameters
void or ss: slave select

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
This is a required method to use SPI on Ameba.

SPIClass::end

Description
Free hardware SPI from any activity.

Syntax
void SPIClass::end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setBitOrder

Description
A specific method to set bit order to either MSB first or LSB first and set slave select pin.

Syntax
void SPIClass::setBitOrder(uint8_t _pin, BitOrder _bitOrder)
void SPIClass::setBitOrder(BitOrder _order)

Parameters
_pin: slave select
_bitOrder: bit order -> either MSB first or LSB first
_order: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setDataMode

Description
A specific method to set data mode to one of the 4 modes (default: SPI_MODE0) and set slave lave select pin.

Syntax
void SPIClass::setDataMode(uint8_t _pin, uint8_t _mode)
void SPIClass::setDataMode(uint8_t _mode)

Parameters
_pin: slave select
_mode: one of the 4 modes (default: SPI_MODE0)

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setClockDivider

Description
A specific method to set to divider in order to get correct clock speed

Syntax
void SPIClass::setClockDivider(uint8_t _pin, uint8_t _divider)

void SPIClass::setClockDivider(uint8_t _div)

Parameters
_pin: slave select
_divider: clock divider
_div: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function does not affect the Ameba board.

SPIClass::setDefaultFrequency

Description
A specific method to set default SPI frequency

Syntax
void SPIClass::setDefaultFrequency(int _frequency)

Parameters
_frequency: the default SPI frequency

Returns
The function returns nothing.

Example Code
Example: PM25_on_ILI9341_TFT_LCD
Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings
Take note that defaultFrequency = _frequency.

Readme

The Ameba SPI related APIs and examples are works based on SPI Master library for arduino written by Cristian Maglie <c.maglie@arduino.cc> and Paul Stoffregen <paul@pjrc.com> (Transaction API).

These include,
SPI.cpp
SPI.h

These libraries are under GNU Lesser General Public License, version 2.1.

Sys

Wiring_OS_API

Wiring OS API

Description
A wrapper to CMSIS (Cortex Microcontroller Software Interface Standard) OS API which serve as a RTOS to create multi-threaded application with real-time behaviour.

Syntax
NA

Members

Public Methods	
os_thread_create_arduino	Create a thread and add it to Active Threads and set it to state READY
os_thread_get_id_arduino	Return the thread ID of the current running thread
os_thread_terminate_arduino	Terminate execution of a thread and remove it from Active Threads
os_thread_yield_arduino	Pass control to next thread that is in state READY
os_thread_set_priority_arduino	Change priority of an active thread
os_thread_get_priority_arduino	Get current priority of an active thread
os_signal_set_arduino	Set the specified Signal Flags of an active thread
os_signal_clear_arduino	Clear the specified Signal Flags of an active thread
os_signal_wait_arduino	Wait for one or more Signal Flags to become signaled for the current RUNNING thread
os_timer_create_arduino	Create a timer
os_timer_start_arduino	Start or restart a timer
os_timer_stop_arduino	Stop the timer
os_timer_delete_arduino	Delete a timer that was created by os_timer_create
os_semaphore_create_arduino	Create and Initialize a Semaphore object used for managing resources
os_semaphore_wait_arduino	Wait until a Semaphore token becomes available
os_semaphore_release_arduino	Release a Semaphore token
os_semaphore_delete_arduino	Delete a Semaphore that was created by os_semaphore_create
os_get_free_heap_size_arduino	Return the available heap memory space when called

os_thread_create_arduino

Description

Create a thread and add it to Active Threads and set it to state READY.

Syntax

uint32_t os_thread_create_arduino (void (*task)(const void *argument), void *argument, int priority, uint32_t stack_size);

Parameters

task: task Function pointer which is the thread body. It should not run into the end of function unless os_thread_terminate is invoked

argument: the data pointer which brings to task

priority: The underlying os is FreeRTOS. It executes tasks with highest priority which are not in idle state.

stack_size: The stack_size is used as memory heap only for this task.

Returns

The thread id which is used in thread operation and signalling.

Example Code

NA

Notes and Warnings

NA

os_thread_get_id_arduino

Description
Return the thread ID of the current running thread

Syntax
uint32_t os_thread_get_id_arduino (void);

Parameters
The function requires no input parameter.

Returns
Current thread id which calls os_thread_get_id.

Example Code
NA

Notes and Warnings
NA

os_thread_terminate_arduino

Description
Terminate execution of a thread and remove it from Active Threads

Syntax
uint32_t os_thread_terminate_arduino (uint32_t thread_id);

Parameters
thread_id: Terminate the thread with specific thread_id

Returns
os_status code

Example Code
NA

Notes and Warnings
Thread should not ended without terminate first.

os_thread_yield_arduino

Description
Pass control to next thread that is in state READY

Syntax
uint32_t os_thread_yield_arduino (void);

Parameters
The function requires no input parameter.

Returns
os_status code

Example Code
NA

Notes and Warnings
By default, the minimal execution unit is 1 millisecond. In a scenario that if a thread with smaller want to handout execution right to a thread with higher priority immediately without waiting for the ending of current 1 millisecond, then invoke os_thread_yield can transfer exection right to OS's idle task and check which is the next execution thread.

os_thread_set_priority_arduino

Description
Change priority of an active thread

Syntax
uint32_t os_thread_set_priority_arduino (uint32_t thread_id, int priority);

Parameters
thread_id: The target thread with the thread id to be changed
priority: The updated priority

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

os_thread_get_priority_arduino

Description
Get current priority of an active thread

Syntax
uint32_t os_thread_get_priority_arduino (uint32_t thread_id);

Parameters
thread_id: The target thread with the thread id to be searched

Returns
os_priority

Example Code
NA

Notes and Warnings
NA

os_signal_set_arduino

Description
Set the specified Signal Flags of an active thread

Syntax
int32_t os_signal_set_arduino (uint32_t thread_id, int32_t signals);

Parameters

thread_id: Send signal to a thread with the thread id
signals: the signals to be send

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_clear_arduino

Description
Clear the specified Signal Flags of an active thread

Syntax
int32_t os_signal_clear_arduino (uint32_t thread_id, int32_t signals);

Parameters
thread_id: Clear signal to a thread with the thread id
signals: The signals to be clear

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_wait_arduino

Description
Wait for one or more Signal Flags to become signaled for the current RUNNING thread

Syntax
os_event_t os_signal_wait_arduino (int32_t signals, uint32_t millisec);

Parameters
signals: the signals to be wait
millisec: the timeout value if no signal comes in. Fill in 0xFFFFFFFF for infinite wait

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_create_arduino

Description
Create a timer

Syntax
uint32_t os_timer_create_arduino (void (*callback)(void const *argument), uint8_t isPeriodic, void *argument);

Parameters
callback: The function to be invoke when timer timeout
isPeriodic: OS_TIMER_ONCE or OS_TIMER_PERIODIC
argument: The argument that is bring into callback function

Returns
timer id

Example Code
NA

Notes and Warnings
NA

os_timer_start_arduino

Description
Start or restart a timer

Syntax
uint32_t os_timer_start_arduino (uint32_t timer_id, uint32_t millisec);

Parameters
timer_id: The timer id obtained from by os_timer_create
millisec: The delays after timer starts

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_stop_arduino

Description
Stop the timer

Syntax
uint32_t os_timer_stop_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_delete_arduino

Description
Delete a timer that was created by os_timer_create

Syntax
uint32_t os_timer_delete_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_create_arduino

Description
Create and Initialize a Semaphore object used for managing resources

Syntax
uint32_t os_semaphore_create_arduino (int32_t count);

Parameters
count: The number of available resources

Returns
semaphore ID

Example Code
NA

Notes and Warnings
NA

os_semaphore_wait_arduino

Description
Wait until a Semaphore token becomes available

Syntax
int32_t os_semaphore_wait_arduino (uint32_t semaphore_id, uint32_t millisec);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create
millisec: timeout value

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_release_arduino

Description
Release a Semaphore token

Syntax
uint32_t os_semaphore_release_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_delete_arduino

Description
Delete a Semaphore that was created by os_semaphore_create

Syntax
uint32_t os_semaphore_delete_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_get_free_heap_size_arduino

Description
Return the available heap memory space when called

Syntax
size_t os_get_free_heap_size_arduino(void);

Parameters
The function requires no input parameter.

Returns
current free heap size

Example Code
Example: MemInfo

Notes and Warnings
NA

WDT

Class WDT

WDT Class

Description
A class used for initializing, starting, stopping watchdog timer.

Syntax
class WDT

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named WDT.	

Public Methods	
WDT:: InitWatchdog	Initializes the watchdog, include time setting, and mode register
WDT:: StartWatchdog	Start the watchdog counting
WDT:: StopWatchdog	Stop the watchdog counting
WDT:: RefreshWatchdog	Refresh the watchdog counting to prevent WDT timeout
WDT:: InitWatchdogIRQ	Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler

WDT:: InitWatchdog

Description
Initializes the watchdog, include time setting, and mode register.

Syntax
void InitWatchdog(uint32_t timeout_ms);

Parameters
timeout_ms: the watch-dog timer timeout value in millisecond (ms). The default action after watchdog timer timeout is to reset the whole system.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer

```
/*
* This example describes how to use watchdog api.
* In this example, watchdog is setup to 5s timeout.
* Watchdog won't bark if we refresh it before timeout in smallTask.
* The timer is also reloaded after refresh.
* Otherwise, while running bigTask, watchdog will restart system in default or call callback function if registered.
*/
#include "wdt.h"
#define RUN_CALLBACK_IF_WATCHDOG_BARKS (0)
WDT wdt;
void setup() {
Serial.begin(115200);
wdt.InitWatchdog(5000); // setup 5s watchdog
#if RUN_CALLBACK_IF_WATCHDOG_BARKS
wdt.InitWatchdogIRQ(my_watchdog_irq_handler, 0);
#else
// system would restart in default when watchdog barks
#endif
wdt.StartWatchdog(); // enable watchdog timer
successfulTask();
failedTask();
while (1)
```

```
;
}
void loop() {
}
void successfulTask(void) {
Serial.println("……doing small task……");
for (int i = 0; i < 50000000; i++) // dummy task
asm(" nop");
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
/*
* Doing this task will lead to failed refresh the
* watchdog timer within the time limits of 5 seconds
*/
void failedTask(void) {
Serial.println("……doing big task……");
for (int i = 0; i < 10; i++) {
Serial.print("doing dummy task #");
Serial.println(i, DEC);
for (int j = 0; j < 50000000; j++) // dummy task
asm(" nop");
}
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
void my_watchdog_irq_handler(uint32_t id) {
printf("watchdog barks!!!rn");
WDG_Cmd(DISABLE);
}
```

Notes and Warnings
NA

WDT:: StartWatchdog

Description
Start the watchdog counting.

Syntax
void StartWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: StopWatchdog

Description
Stop the watchdog counting.

Syntax
void StopWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: RefreshWatchdog

Description
Refresh the watchdog counting to prevent WDT timeout.

Syntax
void RefreshWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: InitWatchdogIRQ

Description
Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler. The interrupt handler will be called when the watchdog timer is timeout.

Syntax
void WDT::InitWatchdogIRQ(wdt_irq_handler handler, uint32_t id)

Parameters
handler: the callback function for WDT timeout interrupt.
id: the parameter for the callback function

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WiFi

Class WiFi

WiFiClass Class

Description
Defines a class of WiFi and network implementation for Ameba.

Syntax
class WiFiClass

Members

Public Constructors	
WiFiClass::WiFiClass	Constructs a WiFiClass object and initializes the WiFi libraries and network settings
Public Methods	
WiFi-Class::firmwareVersion	Get firmware version
WiFiClass:: begin	Start Wifi connection for OPEN networks
WiFiClass:: config	Configure network IP settings
WiFiClass:: setDNS	Set the DNS server IP address to use
WiFiClass:: disconnect	Disconnect from the network
WiFiClass:: macAddress	Get the interface MAC address
WiFiClass:: localIP	Get the interface IP address
WiFiClass:: subnetMask	Get the interface subnet mask address
WiFiClass:: gatewayIP	Get the gateway IP address
WiFiClass:: SSID	Return the current SSID associated with the network
WiFiClass:: BSSID	Return the current BSSID associated with the network
WiFiClass:: RSSI	Return the current RSSI (Received Signal Strength in dBm) associated with the network
WiFiClass:: encryption-Type	Return the Encryption Type associated with the network
WiFiClass:: scanNetworks	Start scan WiFi networks available
WiFiClass:: SSID	Return the SSID discovered during the network scan
WiFiClass:: encryption-Type	Return the encryption type of the networks discovered during the scanNetworks
WiFiClass:: encryption-TypeEx	Return the security type and encryption type of the networks discovered during the scanNetworks
WiFiClass:: RSSI	Return the RSSI of the networks discovered during the scanNetworks
WiFiClass:: status	Return Connection status
WiFiClass:: hostByName	Resolve the given hostname to an IP address
WiFiClass:: apbegin	Start AP mode
WiFiClass:: disablePower-Save	Disable power-saving mode

WiFiClass::WiFiClass

Description

Constructs a WiFiClass object and initializes the WiFi libraries and network settings.

Syntax

WiFiClass::WiFiClass()

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings
An instance of WiFiClass is created as WiFi inside WiFi.h and is extern for direct use.

WiFiClass::firmwareVersion

Description
Get firmware version

Syntax
char* WiFiClass::firmwareVersion()

Parameters
The function requires no input parameter.

Returns
WiFi firmware version

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFI network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>
// char ssid[] = "yourNetwork"; // your network SSID (name)
// char pass[] = "secretPassword"; // your network password
char ssid[] = "SINGTEL-D45F"; // your network SSID (name)
char pass[] = "mooxuteeth"; // your network key
int status = WL_IDLE_STATUS; // the Wifi radio's status
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
```

```
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to WPA SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
```

```
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
```

```
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::begin

Description
Start Wifi connection for OPEN networks

Syntax
int WiFiClass::begin(char* ssid)
int WiFiClass::begin(char* ssid, uint8_t key_idx, const char *key)
int WiFiClass::begin(char* ssid, const char *passphrase)

Parameters
ssid: Pointer to the SSID string
key_idx: The key index to set. Valid values are 0-3.
key: Key input buffer.
passphrase: Passphrase. Valid characters in a passphrase must be between ASCII 32-126 (decimal).

Returns
WiFi status

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::config

Description
Configure network settings for the WiFi network

Syntax
void WiFiClass::config(IPAddress local_ip)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway, IPAddress subnet)

Parameters
local_ip: Local device IP address to use on the network
dns_server: IP address of the DNS server to use
gateway: IP address of the gateway device on the network
subnet: Subnet mask for the network, expressed as a IP address

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This will disable the DHCP client when connecting to a network, and will require the network accepts a static IP. The configured IP addresses will also apply to AP mode, but the DHCP server will not be disabled in AP mode.

WiFiClass::setDNS

Description
Configure the IP address of the DNS server to use

Syntax
void WiFiClass::setDNS(IPAddress dns_server1)
void WiFiClass::setDNS(IPAddress dns_server1, IPAddress dns_server2)

Parameters
dns_server1: IP address of DNS server to use
dns_server2: IP address of DNS server to use

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClass::disconnect

Description
Disconnect from the network

Syntax
int WiFiClass::disconnect()

Parameters
The function requires no input parameter.

Returns
The function returns one value of wl_status_t enum as an integer.

Example Code
NA

Notes and Warnings
NA

WiFiClass::macAddress

Description
Get the interface MAC address

Syntax
uint8_t* WiFiClass::macAddress(uint8_t* mac)

Parameters
mac: an array to store MAC address

Returns
The function returns a pointer to uint8_t array with length WL_MAC_ADDR_LENGTH.

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::localIP

Description

Get the interface IP address

Syntax

IPAddress WiFiClass::localIP()

Parameters

The function requires no input parameter.

Returns

Ip address value

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::subnetMask

Description

Get the interface subnet mask address

Syntax

IPAddress WiFiClass::subnetMask()

Parameters
The function requires no input parameter.

Returns
subnet mask address value

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>
char ssid[] = "SINGTEL-D45F_5G"; // the name of your network
int status = WL_IDLE_STATUS; // the Wifi radio's status
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to open SSID: ");
Serial.println(ssid);
status = WiFi.begin(ssid);
// wait 10 seconds for connection:
delay(10000);
```

```
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
```

```
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
Serial.print("Gateway: ");
Serial.println(gateway);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
}
```

Notes and Warnings

NA

WiFiClass::gatewayIP

Description
Get the gateway IP address

Syntax
IPAddress WiFiClass::gatewayIP()

Parameters
The function requires no input parameter.

Returns
The function returns the value of the gateway IP address.

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details. Details of the code can be found in the section of WiFiClass:: subnetMask.

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the current SSID associated with the network

Syntax
char* WiFiClass::SSID()

Parameters
The function requires no input parameter.

Returns
The function returns current SSID associate with the network.

Example Code
Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::BSSID

Description
Return the current BSSID associated with the network

Syntax
uint8_t* WiFiClass::BSSID(uint8_t* bssid)

Parameters
bssid: an array to store bssid

Returns
pointer to uint8_t array with length WL_MAC_ADDR_LENGTH

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the current RSSI (Received Signal Strength in dBm) associated with the network

Syntax
int32_t WiFiClass::RSSI()

Parameters
The function requires no input parameter.

Returns
The function returns a signed-value signal strength

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the Encryption Type associated with the network

Syntax
uint8_t WiFiClass::encryptionType()

Parameters
The function requires no input parameter.

Returns
The function returns one unsigned integer value of wl_enc_type enum.

Example Code
Example: ConnectWithWPA

Notes and Warnings
NA

WiFiClass::scanNetworks

Description
Start scan WiFi networks available

Syntax
int8_t WiFiClass::scanNetworks()

Parameters
The function requires no input parameter.

Returns
The function returns the number of discovered networks as an integer.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified.

```
#include <WiFi.h>
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// Print WiFi MAC address:
printMacAddress();
}
void loop() {
// scan for existing networks:
Serial.println("Scanning available networks…");
listNetworks();
delay(10000);
}
```

```
void printMacAddress() {
// the MAC address of your Wifi shield
byte mac[6];
// print your MAC address:
WiFi.macAddress(mac);
Serial.print("MAC: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void listNetworks() {
// scan for nearby networks:
Serial.println("* Scan Networks *");
int numSsid = WiFi.scanNetworks();
if (numSsid == -1) {
Serial.println("Couldn't get a wifi connection");
while (true);
}
// print the list of networks seen:
Serial.print("number of available networks:");
Serial.println(numSsid);
// print the network number and name for each network found:
for (int thisNet = 0; thisNet < numSsid; thisNet++) {
Serial.print(thisNet);
Serial.print(") ");
Serial.print(WiFi.SSID(thisNet));
Serial.print("tSignal: ");
Serial.print(WiFi.RSSI(thisNet));
```

```
Serial.print(" dBm");
Serial.print("tEncryptionRaw: ");
printEncryptionTypeEx(WiFi.encryptionTypeEx(thisNet));
Serial.print("tEncryption: ");
printEncryptionType(WiFi.encryptionType(thisNet));
}
}
void printEncryptionTypeEx(uint32_t thisType) {
/* Arduino wifi api use encryption type to mapping to security type.
* This function demonstrate how to get more richful information of security type.
*/
switch (thisType) {
case SECURITY_OPEN:
Serial.print("Open");
break;
case SECURITY_WEP_PSK:
Serial.print("WEP");
break;
case SECURITY_WPA_TKIP_PSK:
Serial.print("WPA TKIP");
break;
case SECURITY_WPA_AES_PSK:
Serial.print("WPA AES");
break;
case SECURITY_WPA2_AES_PSK:
Serial.print("WPA2 AES");
break;
case SECURITY_WPA2_TKIP_PSK:
Serial.print("WPA2 TKIP");
break;
case SECURITY_WPA2_MIXED_PSK:
Serial.print("WPA2 Mixed");
break;
case SECURITY_WPA_WPA2_MIXED:
Serial.print("WPA/WPA2 AES");
break;
```

```
}
}
void printEncryptionType(int thisType) {
// read the encryption type and print out the name:
switch (thisType) {
case ENC_TYPE_WEP:
Serial.println("WEP");
break;
case ENC_TYPE_TKIP:
Serial.println("WPA");
break;
case ENC_TYPE_CCMP:
Serial.println("WPA2");
break;
case ENC_TYPE_NONE:
Serial.println("None");
break;
case ENC_TYPE_AUTO:
Serial.println("Auto");
break;
}
}
```

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the SSID discovered during the network scan

Syntax
char* WiFiClass::SSID(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
The function returns ssid string of the specified item on the networks scanned a list.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the encryption type of the networks discovered during the scanNetworks

Syntax
uint8_t WiFiClass::encryptionType(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
encryption type (enum wl_enc_type) of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionTypeEx

Description
Return the security type and encryption type of the networks discovered during the scanNetworks

Syntax
uint32_t WiFiClass::encryptionTypeEx(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
security and encryption type of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the RSSI of the networks discovered during the scanNetworks

Syntax
int32_t WiFiClass::RSSI(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
signed value of RSSI of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::status

Description
Return Connection status

Syntax
uint8_t WiFiClass::status()

Parameters
The function requires no input parameter.

Returns
The function returns one of the values defined in wl_status_t as an unsigned integer.

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::hostByName

Description
Resolve the given hostname to an IP address

Syntax
int WiFiClass::hostByName(const char* aHostname, IPAddress& aResult)

Parameters
aHostname: Name to be resolved
aResult: IPAddress structure to store the returned IP address

Returns
The function returns “1” if aIPAddrString was successfully converted to an IP address,else otherwise, it will return as an error code.

Example Code

NA

Notes and Warnings
NA

WiFiClass::apbegin

Description
Start AP mode

Syntax
int WiFiClass::apbegin(char* ssid, char* channel)
int WiFiClass::apbegin(char* ssid, char* password, char* channel)

Parameters
ssid: SSID of the AP network
channel: AP's channel, default 1
password: AP's password

Returns
The function will return the WiFi status.

Example Code
Example: WiFiAPMode

#include

char ssid[] = "yourNetwork"; //Set the AP's SSID

char pass[] = "Password"; //Set the AP's password

char channel[] = "1"; //Set the AP's channel

int status = WL_IDLE_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {

```
Serial.println("WiFi shield not present");
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to start AP:
while (status != WL_CONNECTED) {
Serial.print("Attempting to start AP with SSID: ");
Serial.println(ssid);
status = WiFi.apbegin(ssid, pass, channel);
delay(10000);
}
//AP MODE already started:
Serial.println("AP mode already started");
Serial.println();
printWifiData();
printCurrentNet();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
```

```
Serial.print("Gateway: ");
Serial.println(gateway);
Serial.println();
}
void printCurrentNet() {
// print the SSID of the AP:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of AP:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[0], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.println(bssid[5], HEX);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::disablePowerSave

Description
Disable power-saving mode

Syntax
int WiFiClass::disablePowerSave()

Parameters
The function requires no input parameter.

Returns
1 if disable success, 0 if failed

Example Code
NA

Notes and Warnings
NA

Class WiFiClient

WiFiClient Class

Description
Defines a class of WiFi Client implementation for Ameba.

Syntax
class WiFiClient

Members

Public Constructors	
WiFiClient::WiFiClient	Constructs a WiFiClient instance that connects to the specified IP address and port.
Public Methods	
WiFiClient::connect	Connect to the IP address and port
WiFiClient::write	Write a single byte into the packet
WiFiClient::available	Number of bytes remaining in the current packet
WiFiClient::read	Read a single byte from the current packet
WiFiClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiClient:: flush	Finish reading the current packet
WiFiClient::stop	Stop client connection
WiFiClient::connected	Check if client is connected, return 1 if connected, 0 if not
WiFiClient::setRecvTimeout	Set receiving timeout

WiFiClient::WiFiClient

Description

Constructs a WiFiClient instance that connects to the specified IP address and port.

Syntax
WiFiClient::WiFiClient()
WiFiClient::WiFiClient(uint8_t sock)

Parameters
sock: socket state, default -1.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient

```
#include <WiFi.h>
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "password"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
//IPAddress server(64,233,189,94); // numeric IP for Google (no DNS)
char server[] = "www.google.com"; // name address for Google (using DNS)
WiFiClient client;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
;
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
```

```
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 80)) {
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=ameba HTTP/1.1");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
```

```
// do nothing forevermore:
while (true);
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiClient::connect

Description
Connect to the IP address and port

Syntax
int WiFiClient::connect(IPAddress ip, uint16_t port)
int WiFiClient::connect(const char *host, uint16_t port)

Parameters
ip: IP address
host: Host name
port: the port to listen on

Returns

Returns “1”: if successful
Returns “0”: if failed

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiClient::write(uint8_t byte)
size_t WiFiClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message
size: the size of the buffer

Returns
The function returns single byte into the packet or returns bytes size from buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiClient::available(void)

Parameters
The function requires no input parameter.

Returns
• Function returns the number of bytes available in the current packet
Function returns 0: if no data available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiClient::read()
int WiFiClient::read(unsigned char* buf, size_t size)
int WiFiClient::read(char *buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiClient::flush

Description
Finish reading the current packet

Syntax
void WiFiClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClient::stop

Description
Disconnect from the server. Stop client connection

Syntax
void WiFiClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::connected

Description
Check if client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
0

Example Code
NA

Notes and Warnings
NA

Class WiFiServer

WiFiServer Class

Description
Defines a class of WiFi server implementation for Ameba.

Syntax
class WiFiServer

Members

Public Constructors	
WiFiServer::WiFiServer	Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port
Public Methods	
WiFiServer::available	Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop()
WiFiServer::begin	Tells the server to begin listening for incoming connections
WiFiServer::write	Write data to all the clients connected to a server

WiFiServer::WiFiServer

Description
Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port.

Syntax
WiFiServer::WiFiServer(uint16_t port)

Parameters
port: The port number being connected to.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi

#include <WiFi.h>

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

WiFiServer server(5000);

void setup() {

Serial.begin(9600); // initialize serial communication

pinMode(9, OUTPUT); // set the LED pin mode

```
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
while (true); // don't continue
}
String fv = WiFi.firmwareVersion();
if ( fv != "1.1.0" )
Serial.println("Please upgrade the firmware");
// attempt to connect to Wifi network:
while ( status != WL_CONNECTED) {
Serial.print("Attempting to connect to Network named: ");
Serial.println(ssid); // print the network name (SSID);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
server.begin(); // start the tcp server on port 5000
printWifiStatus(); // you're connected now, so print out the status
}
char buffer[256];
void loop() {
WiFiClient client = server.available();
while (client.connected()) {
memset(buffer, 0, 256);
int n = client.read((uint8_t*)(&buffer[0]), sizeof(buffer));
if (n > 0) {
for (int i=0; i<n; i++) {
Serial.print(buffer[i]);
}
n = client.write(buffer, n);
if (n <= 0) break;
}
}
client.stop();
}
```

```
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiServer::available

Description
Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop().

Syntax
WiFiClient WiFiServer::available(uint8_t* status)

Parameters
status: WiFi availability status

Returns
A Client object; if no Client has data available for reading, this object will evaluate to false in an if-statement

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::begin

Description
Tells the server to begin listening for incoming connections

Syntax
void WiFiServer::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::write

Description
Write data to all the clients connected to a server

Syntax
size_t WiFiServer::write(uint8_t b)
size_t WiFiServer::write(const uint8_t *buf, size_t size)

Parameters
b: byte to be written
buf: data buffer
size: Size of the data in the buffer

Returns
The function returns the number of bytes written. It is not necessary to read this.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

Class WiFiSSLClient

WiFiSSLClient Class

Description
Defines a class of WiFi Secure Socket Layer Client implementation for Ameba.

Syntax
class WiFiSSLClient

Members

Public Constructors	
WiFiSSLClient::WiFiSSLClient	Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port
Public Methods	
WiFiSSLClient::connect	Connect to the IP address and port
WiFiSSLClient::write	Write a single byte into the packet
WiFiSSLClient::available	Number of bytes remaining in the current packet
WiFiSSLClient::read	Read a single byte from the current packet
WiFiSSLClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiSSLClient:: flush	Finish reading the current packet
WiFiSSLClient::stop	Stop SSL client connection
WiFiSSLClient::connected	Check if SSL client is connected, return 1 if connected, 0 if not
WiFiSSLClient:: setRootCA	Set Root CA for authentication
WiFiSSLClient:: setClientCertificate	Set certificate of the client
WiFiSSLClient::setRecvTimeout	Set receiving timeout
WiFiSSLClient::setPreSharedKey	Set the pre shared key (PSK) to use for authentication

WiFiSSLClient::WiFiSSLClient

Description
Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port.

Syntax
WiFiSSLClient::WiFiSSLClient(void)
WiFiSSLClient::WiFiSSLClient(uint8_t sock)

Parameters
sock: socket state, default -1

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient

```
#include
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword";// your network password (use for WPA, or WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
char server[] = "www.google.com"; // name address for Google (using DNS)
//unsigned char test_client_key[] = ""; //For the usage of verifying client
//unsigned char test_client_cert[] = ""; //For the usage of verifying client
//unsigned char test_ca_cert[] = ""; //For the usage of verifying server
WiFiSSLClient client;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
```

```
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 443)) { //client.connect(server, 443, test_ca_cert, test_client_cert, test_client_key)
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=realtek HTTP/1.0");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
else
Serial.println("connected to server failed");
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
// do nothing forevermore:
while (true);
```

```
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiSSLClient::connect

Description
Connect to the IP address and port.

Syntax
int WiFiSSLClient::connect(IPAddress ip, uint16_t port)
int WiFiSSLClient::connect(const char *host, uint16_t port)
int WiFiSSLClient::connect(const char* host, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)
int WiFiSSLClient::connect(IPAddress ip, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)

Parameters
ip: IP address
host: Host name
port: the port to listen on
rootCABuff: buffer that store root CA
cli_cert: buffer that store client certificate
cli_key buffer that store client key pair

Returns
1: if successful
0: if failed

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiSSLClient::write(uint8_t byte)
size_t WiFiSSLClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message

size: the size of the buffer

Returns
The function returns single -byte into the packet or turns bytes size from the buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiSSLClient::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available in the current packet; else return “0:” if no data available.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiSSLClient::read()

int WiFiSSLClient::read(unsigned char* buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::peek

Description
Return the next byte from the current packet without moving on to the next byte.

Syntax
int WiFiSSLClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::flush

Description
Finish reading the current packet

Syntax
void WiFiSSLClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::stop

Description
Disconnect from the server. Stop SSL client connection

Syntax
void WiFiSSLClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::connected

Description
Check if SSL client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiSSLClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::setRootCA

Description
Set Root CA for authentication

Syntax
void WiFiSSLClient::setRootCA(unsigned char *rootCA)

Parameters
rootCA: a string of rootCA

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

WiFiSSLClient::setClientCertificate

Description
Set certificate of client

Syntax
void WiFiSSLClient::setClientCertificate(unsigned char *client_ca, unsigned char *private_key)

Parameters
client_ca: Client certificate
private_key: client's private key pair

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiSSLClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
The function returns "0".

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setPreSharedKey

Description
Set the pre shared key (PSK) to use for authentication

Syntax
void WiFiSSLClient::setPreSharedKey(unsigned char *pskIdent, unsigned char *psKey)

Parameters
pskIdent: identity for PSK
psKey: Pre shared key

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Do not set a root CA and client certificate if PSK should be used for authentication. If root CA, client certificate and PSK are all set, certificate based authentication will be used.

Class WiFiUdp

WiFiUDP Class

Description
Defines a class of WiFi UDP implementation for Ameba.

Syntax
class WiFiUDP

Members

Public Constructors	
WiFiUDP::WiFiUDP	Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages
Public Methods	
WiFiUDP:: begin	initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use
WiFiUDP:: stop	Finish with the UDP socket
WiFiUDP:: beginPacket	Start building up a packet to send to the remote host-specific in IP and port
WiFiUDP:: endPacket	Finish off this packet and send it
WiFiUDP:: write	Write a single byte into the packet
WiFiUDP:: writeImmediately	Send packet immediately from the buffer
WiFiUDP:: parsePacket	Start processing the next available incoming packet
WiFiUDP::available	Number of bytes remaining in the current packet
WiFiUDP::read	Read a single byte from the current packet
WiFiUDP:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiUDP:: flush	Finish reading the current packet
WiFiUDP:: remoteIP	Return the IP address of the host who sent the current incoming packet
WiFiUDP:: remotePort	Return the port of the host who sent the current incoming packet
WiFiUDP:: setRecvTimeout	Set receiving timeout

WiFiUDP::WiFiUDP

Description
Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages.

Syntax
WiFiUDP::WiFiUDP(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort.

#include <WiFi.h>

#include <WiFiUdp.h>

```
int status = WL_IDLE_STATUS;
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
unsigned int localPort = 2390; // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back
WiFiUDP Udp;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
```

```
// if you get a connection, report back via serial:
Udp.begin(localPort);
}
void loop() {
// if there's data available, read a packet
int packetSize = Udp.parsePacket();
if (packetSize) {
Serial.print("Received packet of size ");
Serial.println(packetSize);
Serial.print("From ");
IPAddress remoteIp = Udp.remoteIP();
Serial.print(remoteIp);
Serial.print(", port ");
Serial.println(Udp.remotePort());
// read the packet into packetBufffer
int len = Udp.read(packetBuffer, 255);
if (len > 0) {
packetBuffer[len] = 0;
}
Serial.println("Contents:");
Serial.println(packetBuffer);
// send a reply, to the IP address and port that sent us the packet we received
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
Udp.write(ReplyBuffer);
Udp.endPacket();
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
```

```
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
This constructor does not take in any parameter, thus use another method to set up the IP address and port number.

WiFiUDP::begin

Description
Initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use.

Syntax
uint8_t WiFiUDP::begin(uint16_t port)

Parameters
port: the local port to listen on

Returns
1: if successful
0: if there are no sockets available to use

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::stop

Description
Disconnect from the server. Release any resource being used during the UDP session.

Syntax
void WiFiUDP::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::beginPacket

Description
Start building up a packet to send to the remote host-specific in IP and port.

Syntax
int WiFiUDP::beginPacket(const char *host, uint16_t port)
int WiFiUDP::beginPacket(IPAddress ip, uint16_t port)

Parameters
host: hostname
port: port number
ip: IP address

Returns
1: if successful
0: if there was a problem with the supplied IP address or port

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::endPacket

Description
Finish off this packet and send it

Syntax
int WiFiUDP::endPacket(void)

Parameters
The function requires no input parameter.

Returns
1: if the packet was sent successfully
0: if there was an error

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::write

Description
Write a single byte into the packet.

Syntax
size_t WiFiUDP::write(uint8_t byte)
size_t WiFiUDP::write(const uint8_t *buffer, size_t size)

Parameters
byte: the outgoing byte

buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::writeImmediately

Description
Send packet immediately from the buffer

Syntax
size_t WiFiUDP::writeImmediately(const uint8_t *buffer, size_t size)

Parameters
buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::parsePacket

Description
Start processing the next available incoming packet

Syntax
int WiFiUDP::parsePacket(void)

Parameters
The function requires no input parameter.

Returns
The function returns the size of the packet in bytes or returns "0:" if no packets are available.

Example Code
Example: WiFiUdpSendReceiveString

Notes and Warnings
NA

WiFiUDP::available

Description
Number of bytes remaining in the current packet.

Syntax
int WiFiUDP::available(void)

Parameters
The function requires no input parameter.

Returns
the number of bytes available in the current packet
0: if parsePacket hasn't been called yet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::read

Description
Read a single byte from the current packet

Syntax
int WiFiUDP::read()
int WiFiUDP::read(unsigned char* buffer, size_t len)

Parameters
buffer: buffer to hold incoming packets (char*)
len: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiUdpSendReceiveString
his example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiUDP::peek(void)

Parameters

The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiUDP::flush

Description
Finish reading the current packet

Syntax
void WiFiUDP::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::remoteIP

Description
Return the IP address of the host who sent the current incoming packet

Syntax
IPAddress WiFiUDP::remoteIP(void)

Parameters
The function requires no input parameter.

Returns
IP address connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::remotePort

Description
Return the port of the host who sent the current incoming packet

Syntax
uint16_t WiFiUDP::remotePort(void)

Parameters
The function requires no input parameter.

Returns
The remote port connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::setRecvTimeout

Description
Set receiving timeout

Syntax
void WiFiUDP::setRecvTimeout(int timeout)

Parameters
timeout in seconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba WiFi related APIs and examples are works based on Arduino WiFI shield libraries (https://www.arduino.cc/en/Reference/WiFi).
These include,

- `WiFi.cpp`
- `WiFi.h`
- `WiFiServer.cpp`
- `WiFiServer.h`
- `WiFiUdp.cpp`
- `WiFiUdp.h`

Wire

Class TwoWire

TwoWire Class

Description
Defines a class of I2C API

Syntax
class TwoWire

Members

Public Constructors	
TwoWire::TwoWire	Constructs a TwoWire object
Public Methods	
TwoWire::begin	Initialize I2C master/slave
TwoWire::setClock	Set I2C frequency
TwoWire::beginTransmission	Begin I2C transmission
TwoWire::endTransmission	End I2C transmission
TwoWire::requestFrom	Set I2C requestFrom
TwoWire::write	Write data to I2C
TwoWire::available	Check if I2C is available
TwoWire::read	Read data from I2C
TwoWire::peek	Read peek from I2C
TwoWire::flush	Do nothing, use, and transmission(..) to force data transfer
TwoWire::onReceive	Callback function when I2C on receive
TwoWire::onRequest	Callback function when I2C on request

TwoWire::TwoWire

Description
Constructs a TwoWire object.

Syntax
TwoWire::TwoWire (uint32_t dwSDAPin, uint32_t dwSCLPin);

Parameters
dwSDAPin: The Arduino PIN to be set as an SDA pin.
dwSCLPin: The Arduino PIN to be set as an SCL pin.

Returns
The function returns nothing.

Example Code

Example: MasterWriter

This example demonstrates the use of the wire library writes to an I2C/TWI slave device.

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
}
byte x = 0;
void loop() {
Wire.beginTransmission(8); // transmit to device #8
Wire.write("x is "); // sends five bytes
Wire.write(x); // sends one byte
Wire.endTransmission(); // stop transmitting
x++;
delay(500);
}
```

Example: MasterReader

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
Serial.begin(9600); // start serial for output
}
void loop() {
Wire.requestFrom(8, 6); // request 6 bytes from slave device #8
while (Wire.available()) { // slave may send less than requested
char c = Wire.read(); // receive a byte as character
Serial.print(c); // print the character
}
delay(500);
}
```

This example demonstrates the use of the wire library reads data from an I2C/TWI slave device.

Notes and Warnings

Include "Wire.h" to use the class function.

TwoWire::begin

Description
Initialize I2C master/slave.

Syntax
void TwoWire::begin (void);
void TwoWire::begin (uint8_t address = 0);
void TwoWire::begin (int address);

Parameters
void: Set the I2C master mode.
address: Set the I2C master mode with slave address value.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::setClock

Description
Set I2C frequency.

Syntax
void TwoWire::setClock(uint32_t frequency);

Parameters
frequency: The frequency values.

Returns
The function returns nothing.

Example Code

Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::beginTransmission

Description
Begin I2C transmission.

Syntax
void TwoWire::beginTransmission (uint8_t address);
void TwoWire::beginTransmission (int address);

Parameters
address: The transmission address.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::endTransmission

Description
End I2C transmission. Originally, ‘endTransmission’ was an f(void) function. It has been modified to take one parameter indicating whether or not a STOP should be performed on the bus. Calling endTransmission(false) allows a sketch to perform a repeated start.

WARNING: Nothing in the library keeps track of whether the bus tenure has been properly ended with a STOP. It is very possible to leave the bus in a hung state if no call to endTransmission(true) is made. Some I2C devices will behave oddly if they do not see a STOP.

If the input parameter is void, this provides backward compatibility with the original definition, and expected behavior, of endTransmission.

Syntax
uint8_t TwoWire::endTransmission (uint8_t sendStop);
uint8_t TwoWire::endTransmission (void);

Parameters
sendStop: True to end the transmission

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::requestFrom

Description
Set I2C requestFrom.

Syntax
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity, uint8_t sendStop);
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity);
uint8_t TwoWire::requestFrom(int address, int quantity);
uint8_t TwoWire::requestFrom (int address, int quantity, int sendStop);

Parameters
address: I2C read address.
quantity: I2C read quantity.
sendStop: True to end the transmission.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::write

Description
Write data to I2C.

Syntax
size_t TwoWire::write (uint8_t data);
size_t TwoWire::write (const uint8_t *data, size_t quantity);

Parameters
data: The data to be transmitted.
quantity: The quantity of data.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::available

Description
Check if I2C is available.

Syntax
int TwoWire::available (void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::read

Description
Read data from I2C

Syntax
int TwoWire::read (void);

Parameters
The function requires no input parameter.

Returns
The read data from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::peek

Description
Read peek from I2C.

Syntax
int TwoWire::peek (void);

Parameters
The function requires no input parameter.

Returns
The peek data read from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::flush

Description
Do nothing, use endTransmission(..) to force data transfer.

Syntax
void TwoWire::flush (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter

Notes and Warnings
Include “Wire.h” in order to use the class function.

TwoWire::onReceive

Description
Callback function when I2C on receive.

Syntax
void TwoWire::onReceive (void(*function)(int));

Parameters
function: The callback function.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::onRequest

Description
Callback function when I2C on request.

Syntax
void TwoWire::onRequest (void(*function)(void));

Parameters
function: The callback function

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

Wire_Readme

The Ameba LCD related api and example are works based on "New LiquidCrystal library" (https://bitbucket.org/fmalpartida/new-liquidcrystal/).

These include,

LCD.h
LCD.cpp
I2CIO.h
I2CIO.cpp
LiquidCrystal_I2C.h
LiquidCrystal_I2C.cpp
examples/LcdHelloWorld/LcdHelloWorld.ino

These files inherit the licence of "New LiquidCrystal Library" which are under a Creative Commons Attribution-ShareAlike 3.0 Unported License. CC BY-SA 3.0.

1.1.6 Resources

Links

- AmebaD Arduino Github
- Arduino Website

1.1.7 Support

FAQ

Where to buy Ameba RTL8722DM Board?

Refer to Purchase link.

Which Bluetooth standards are supported by RTL8722CSM/RTL8722DM?

Both boards support BLE 5.0. Classic Bluetooth (BR/EDR) is not supported.

Which BLE roles are supported?

RTL8722CSM/RTL8722DM can operate as either a BLE Central or BLE Peripheral device.

Are all pins on RTL8722CSM/RTL8722DM usable?

No, those marked `NC` are not connected to any pin and thus unusable.

Is XIP (execute in place) supported on RTL8722CSM/RTL8722DM?

Yes, it is supported.

Does RTL8722CSM support 5G WiFi?

No. Only RTL8722DM supports dual band 2.4G + 5G WiFi. RTL8722CSM only supports single band 2.4G WiFi.

How to enter the download mode?

Press and hold the UART DOWNLOAD button. Then Press the RESET button and release both UART DOWNLOAD and RESET buttons.

Trouble shooting

RTL8722CSM/RTL8722DM cannot be found as a Bluetooth device.

Please make sure the antenna is connected properly. Check your code for the correct Bluetooth configurations.

My code is not behaving as I expected.

Try to debug your program using `printf()` and `Serial.print()` statements. If the issue persists, you can ask for help at Forums

Why is there no output on my serial terminal after connecting to RTL8722CSM/RTL8722DM UART?

RTL8722CSM/RTL8722DM is by default configured at 115200 baudrate, please check if your serial terminal is configured to 115200.

My program is not being downloaded into RTL8722CSM/RTL8722DM?

Please follow the procedure for the correct downloading?

1. Enter the download mode. The on-board Green LED will blink when entered download mode.
2. When downloading the image into board the on-board Red LED will blink
3. After a successful download, you will see log like this “All images sent successfully”.

Sometimes WiFi signal is weak?

The default antenna for RTL8722CSM/RTL8722DM uses the I-Pex Connector. Please change/connect the I-Pex Connector antenna.

Why is my board not powering up?

Please make sure the connector J38 beside resistor R43 is connected. The connector is used to link the power to IC.

If you have driver issue to connect board to your computer?

Please go to https://ftdichip.com/drivers/ for USB driver.

1.2 AMB23 (RTL8722DM MINI)

Welcome to AMB23 Arduino online documentation.

1.2.1 Getting Started

Ameba ARDUINO: Getting Started with AMB23

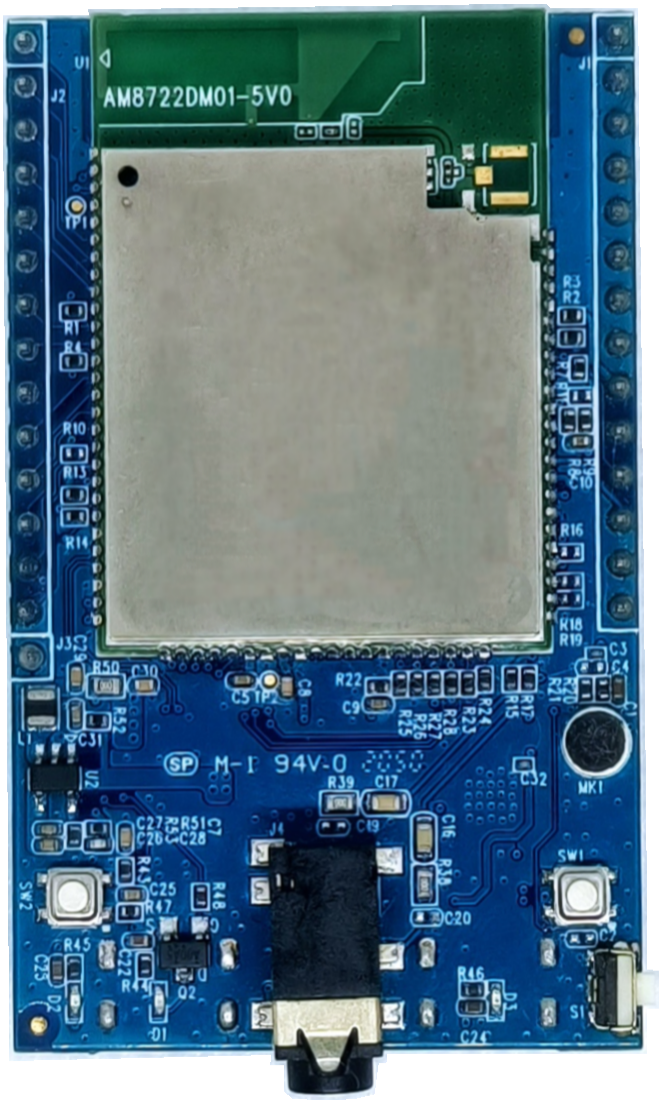
Required Environment

AMB23 board currently supports Windows OS 32-bits and 64-bits (WIN7/8/10), Linux OS (Ubuntu 18 LTS/20 LTS/latest) and macOS operating systems. Please use the latest OS version to have the best experiences. In this documentation, please use the latest version Arduino IDE (at least version 1.8.12).

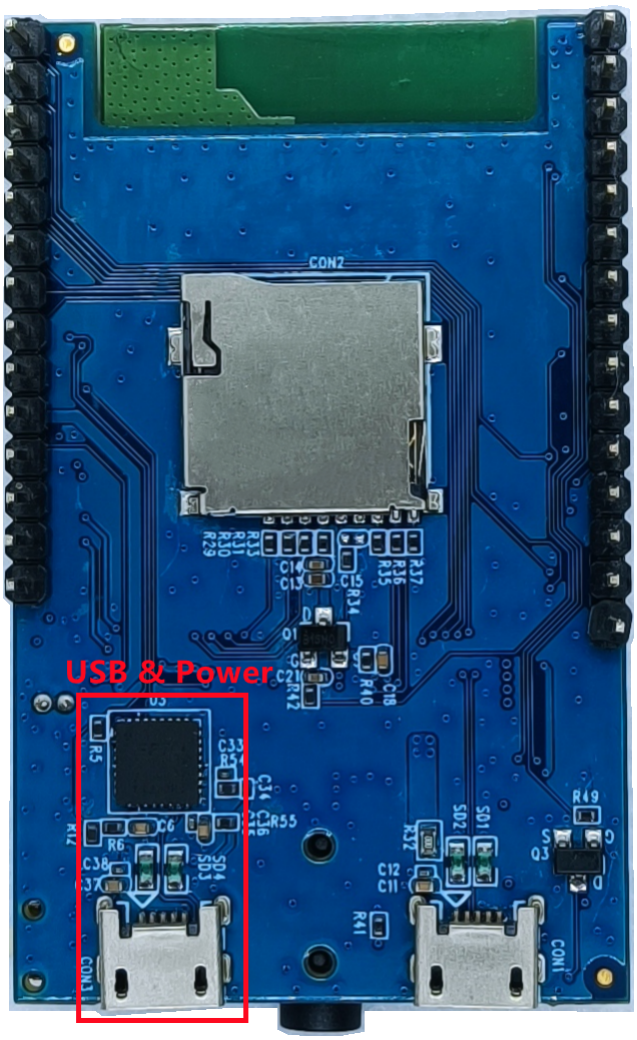
Introduction to AmebaD[AMB23]

Ameba is an easy-to-program platform for developing all kind of IoT applications. AmebaD is equipped with various peripheral interfaces, including WiFi, GPIO INT, I2C, UART, SPI, PWM, ADC. Through these interfaces, AmebaD can connect with electronic components such as LED, switches, manometer, hygrometer, PM2.5 dust sensors, …etc.

The collected data can be uploaded via WiFi and be utilized by applications on smart devices to realize IoT implementation.

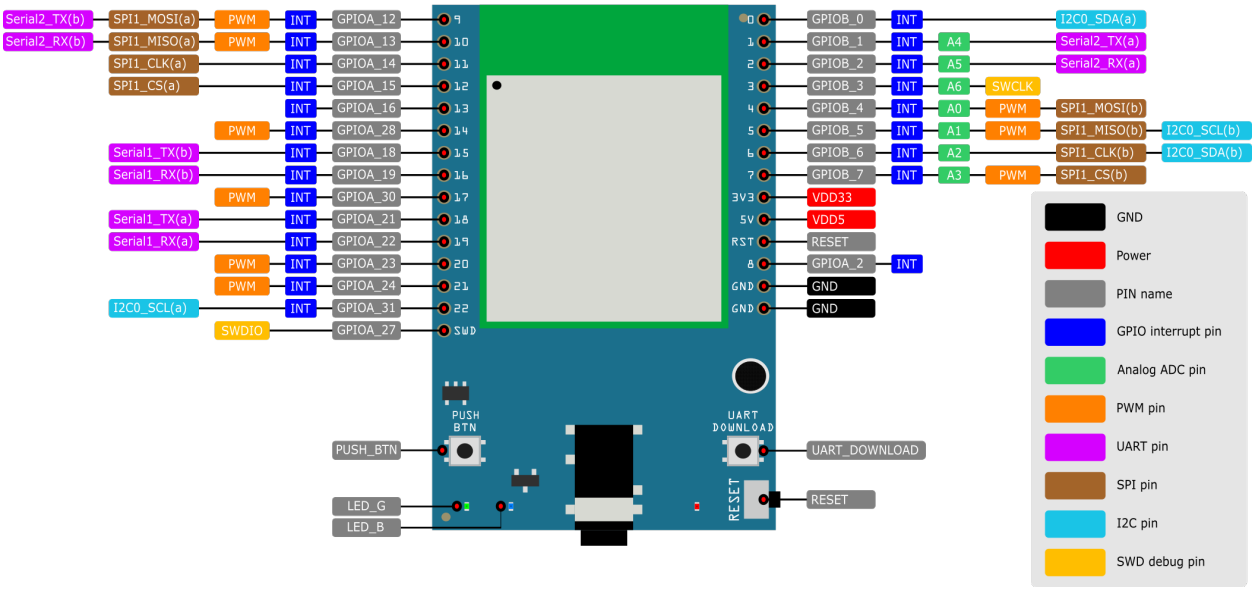


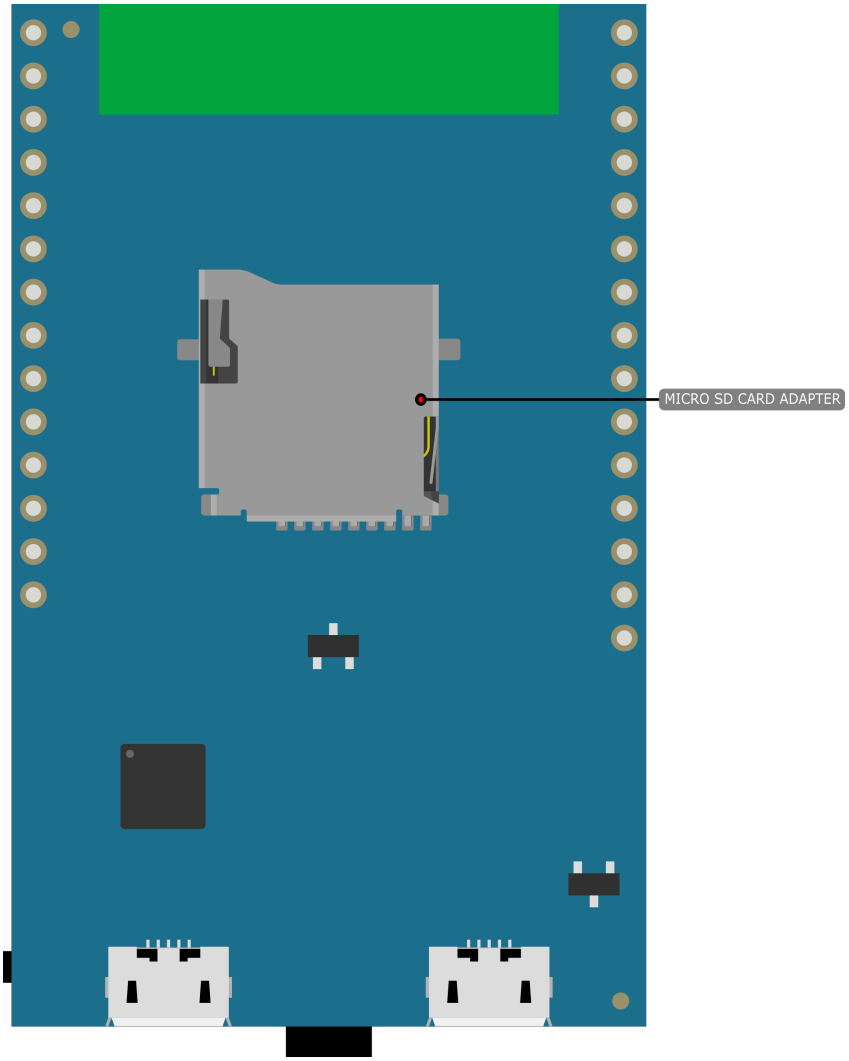
AMB23 has smaller size than Arduino Uno, as shown in the above figure.



AMB23 uses Micro USB to supply power, which is common in many smart devices.
Please refer to the following figure and table for the pin diagram and function of AMB23.

AMB23 EVB


Serial2_TX(b)
Serial2_RX(b)
SPI1_MOSI(a)
SPI1_MISO(a)
SPI1_CLK(a)
SPI1_CS(a)
Serial1_TX(b)
Serial1_RX(b)
Serial1_TX(a)
Serial1_RX(a)
I2C0_SCL(a)
SWDIO
PWM
INT
GPIOA_12
GPIOA_13
GPIOA_14
GPIOA_15
GPIOA_16
GPIOA_28
GPIOA_18
GPIOA_19
GPIOA_30
GPIOA_21
GPIOA_22
GPIOA_23
GPIOA_24
GPIOA_31
GPIOA_27
GPIOB_0
GPIOB_1
GPIOB_2
GPIOB_3
GPIOB_4
GPIOB_5
GPIOB_6
GPIOB_7
VDD33
VDD5
RESET
GPIOA_2
GND
A4
A5
A6
A0
A1
A2
A3
SWCLK
I2C0_SDA(a)
Serial2_TX(a)
Serial2_RX(a)
SPI1_MOSI(b)
SPI1_MISO(b)
SPI1_CLK(b)
SPI1_CS(b)
I2C0_SCL(b)
I2C0_SDA(b)
PUSH BTN
PUSH_BTN
UART DOWNLOAD
UART_DOWNLOAD
RESET
LED_G
LED_B
GND
Power
PIN name
GPIO interrupt pin
Analog ADC pin
PWM pin
UART pin
SPI pin
I2C pin
SWD debug pin

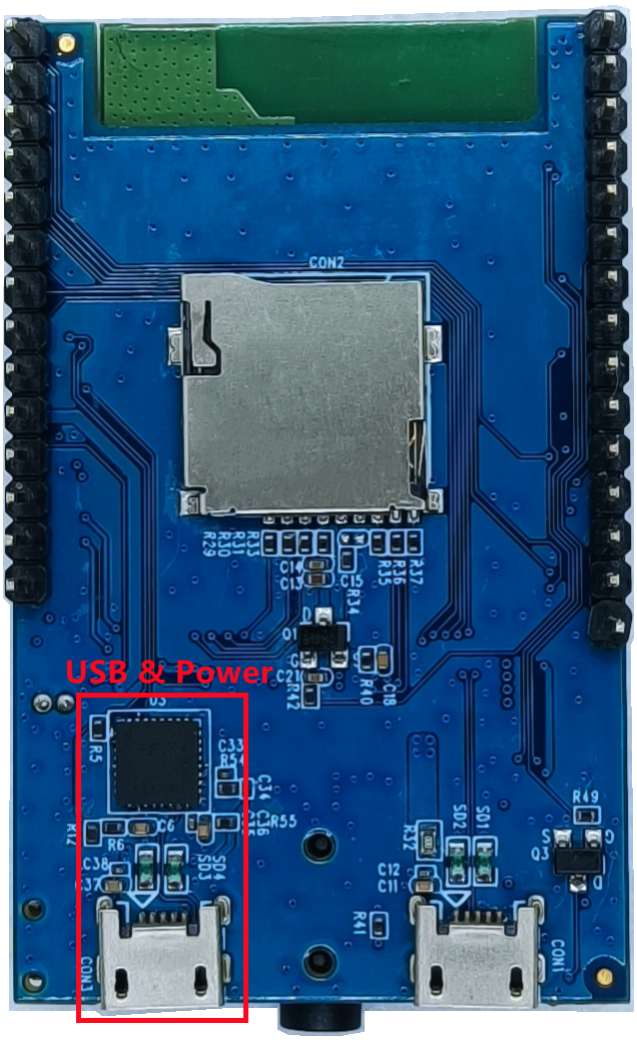

MICRO SD CARD ADAPTER

#	PIN name	GPIO INT	ADC	PWM	UART	SPI	I2C
D0	GPIOB_0	✓					I2C0 SDA
D1	GPIOB_1	✓	A4		Serial2_TX		
D2	GPIOB_2	✓	A5		Serial2_RX		
D3	GPIOB_3	✓	A6				
D4	GPIOB_4	✓	A0	✓			
D5	GPIOB_5	✓	A1	✓			I2C0 SCL
D6	GPIOB_6	✓	A2				I2C0 SDA
D7	GPIOB_7	✓	A3	✓			
D8	GPIOA_2	✓					
D9	GPIOA_12	✓		✓	Serial2_TX	SPI1_MOSI	
D10	GPIOA_13	✓		✓	Serial2_RX	SPI1_MISO	
D11	GPIOA_14	✓				SPI1_CLK	
D12	GPIOA_15	✓				SPI1_CS	
D13	GPIOA_16	✓					
D14	GPIOA_28	✓		✓			
D15	GPIOA_18	✓			Serial1_TX		
D16	GPIOA_19	✓			Serial1_RX		
D17	GPIOA_30	✓		✓			
D18	GPIOA_21	✓			Serial1_TX		
D19	GPIOA_22	✓			Serial1_RX		
D20	GPIOA_23	✓		✓			
D21	GPIOA_24	✓		✓			
D22	GPIOA_31	✓					I2C0 SCL

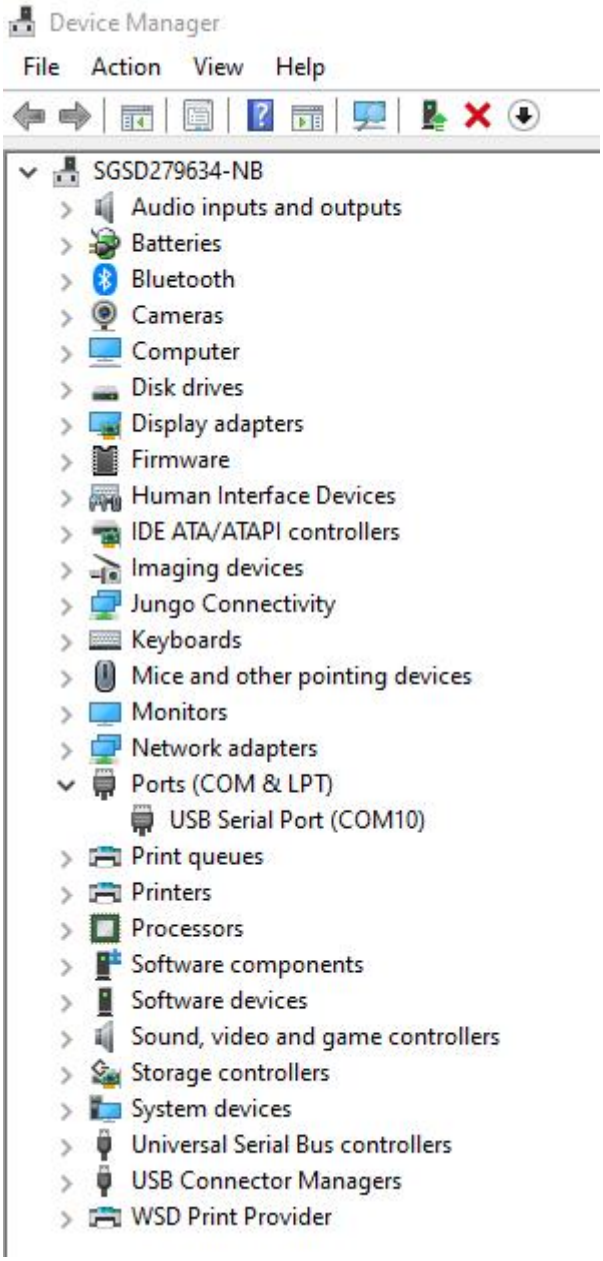
Setting up Development Environment

Step 1. Installing the Driver

First, connect AMB23 to the computer via Micro USB (same as power):



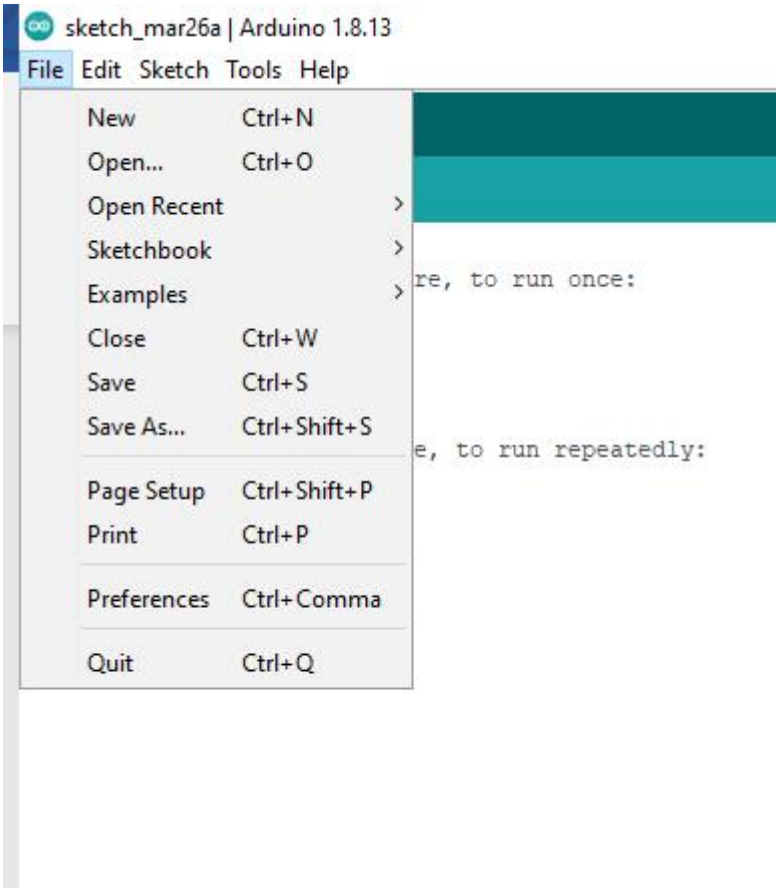
If this is the first time you connect AMB23 to your computer, the USB driver for AMB23 will be automatic installed.
If you have driver issue of connect board to your computer please go to here for USB driver.
You can check the COM port number in Device Manager of your computer:



Step 2. Set up Arduino IDE

From version 1.6.5, Arduino IDE supports third-party hardware. Therefore, we can use Arduino IDE to develop applications on AMB23, and the examples of Arduino can run on AMB23 too. Refer to basic example link.

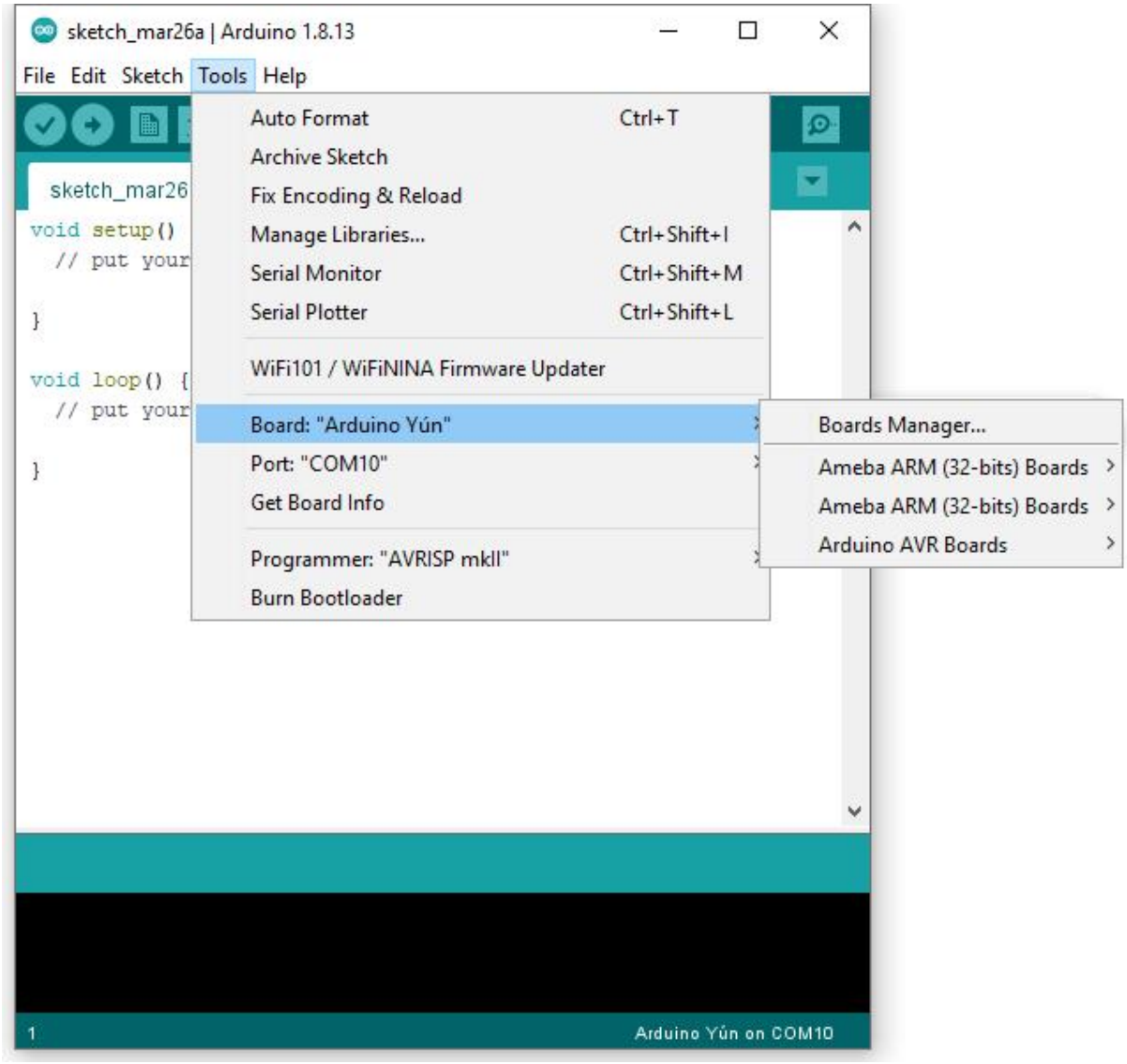
Arduino IDE can be downloaded in the Arduino website:
https://www.arduino.cc/en/Main/Software
When the installation is finished, open Arduino IDE. To set up AMB23 correctly in Arduino IDE, go to *"File"* -> *"Preferences"*.



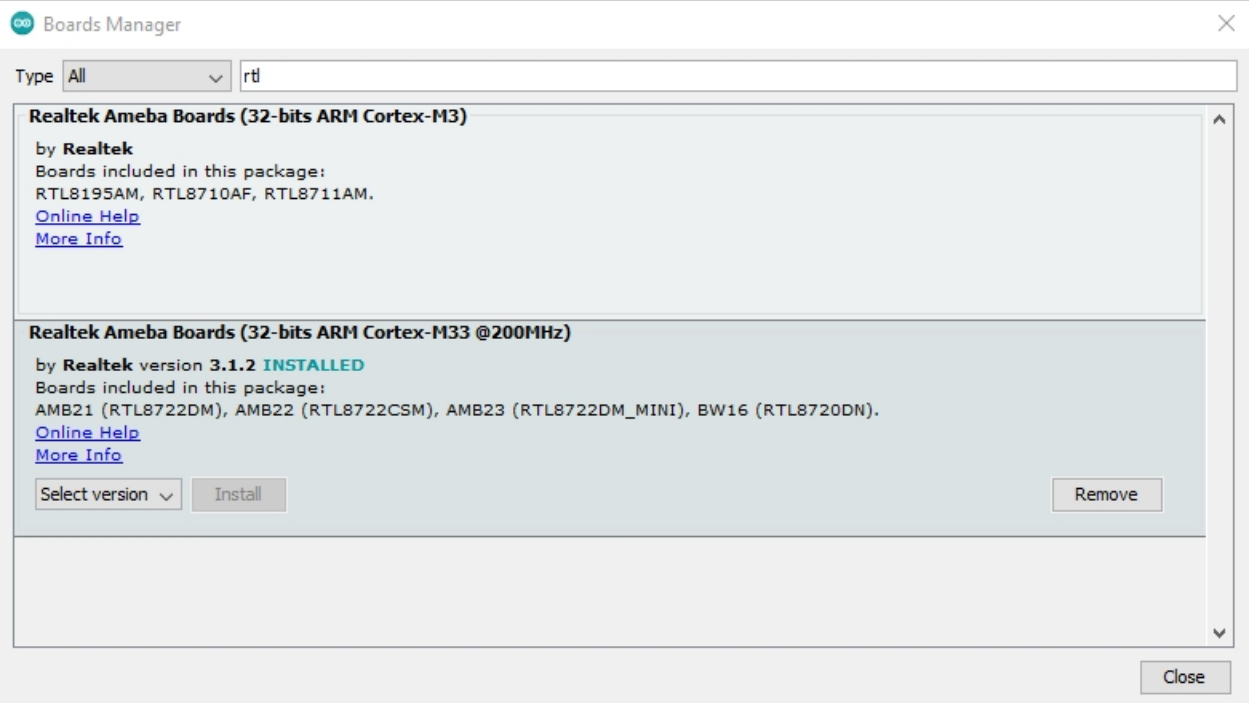
And paste the following URL into *"Additional Boards Manager URLs"* field:

```
https://github.com/ambiot/ambd_arduino/raw/master/Arduino_package/package_realtek.com_
↪amebad_index.json
```

Next, go to *"Tools"* -> *"Board"* -> *"Boards Manager"*:



The *"Boards Manager"* requires about 10~20 seconds to refresh all hardware files (if the network is in bad condition, it may take longer). Every time the new hardware is connected, we need to reopen the Board Manager. So, we close the *"Boards Manager"*, and then open it again. Find *"Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)"* in the list, click *"Install"*, then the Arduino IDE starts to download required files for AmebaD.



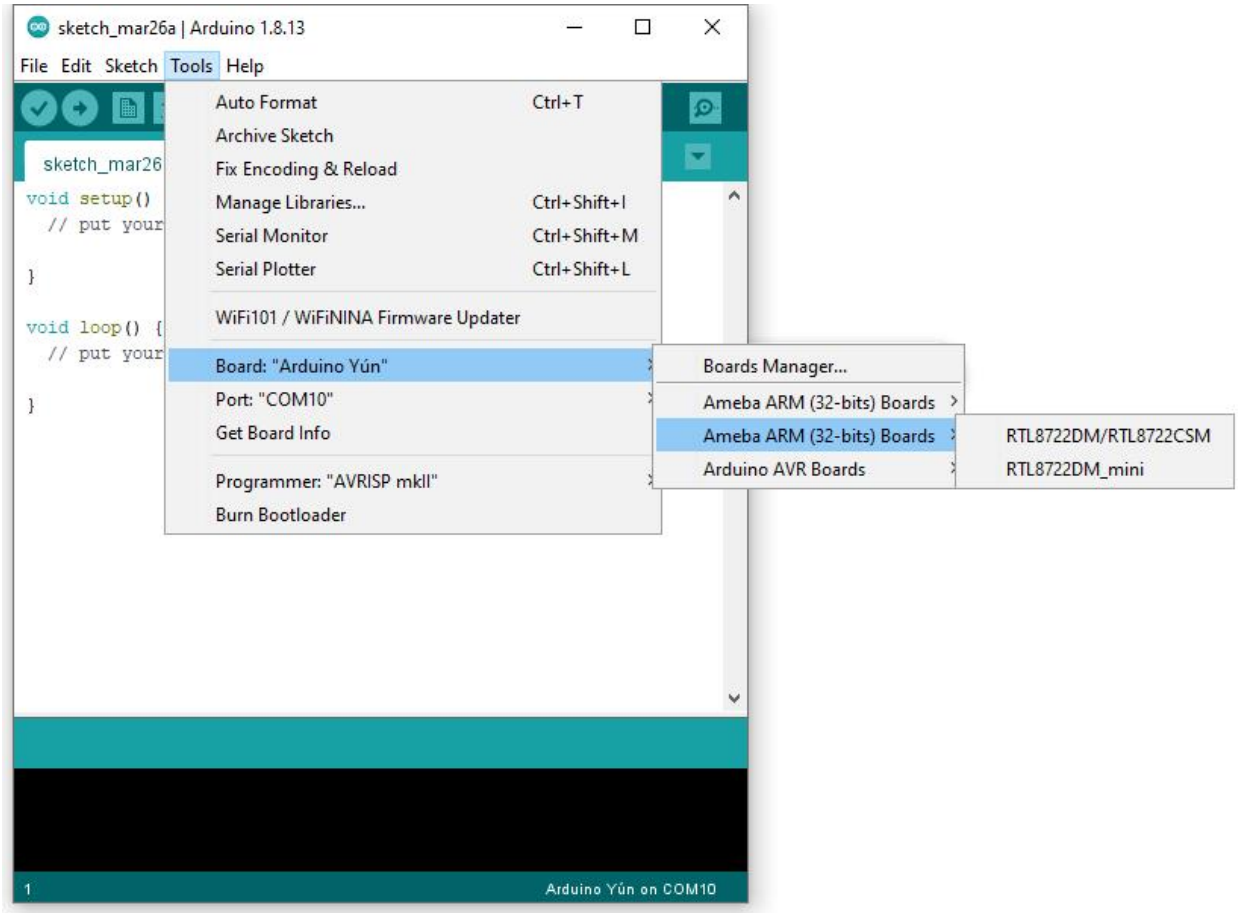
If you are facing GitHub downloading issue, please refer to the following link at Download/Software Development Kit. There are 3 sections:

1. "AmebaD_Arduino_patch1_SDK", please select at least 1 of the SDKs. There are 5 latest released SDK options.
2. "AmebaD_Arduino_patch2_Tools", please select according to your operation system. There are Windows, Linux and MacOS.
3. "AmebaD_Arduino_Source_Code", this section is optional download only wants to refer the latest source code.

Download the files selected, then unzip (patch1 and patch2 are compulsory). There are "Install.doc"/"Install.pdf" for you to refer installation steps. According to your system, please run the installation tool in the "Offline_SDK_installation_tool" folder.

After the installation tool running successfully, you may open Arduino IDE and proceed to *"Tools"* -> *"Board"* -> *"Boards Manager…"*. Try to find *"Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)"* in the list, click *"Install"*, then the Arduino IDE starts to download required files for AmebaD.

Finally, we select AmebaD as current connected board in *"Tools"* -> *"Board"* -> *"Ameba ARM (32-bits) Boards"* -> *"RTL8722DM MINI"*

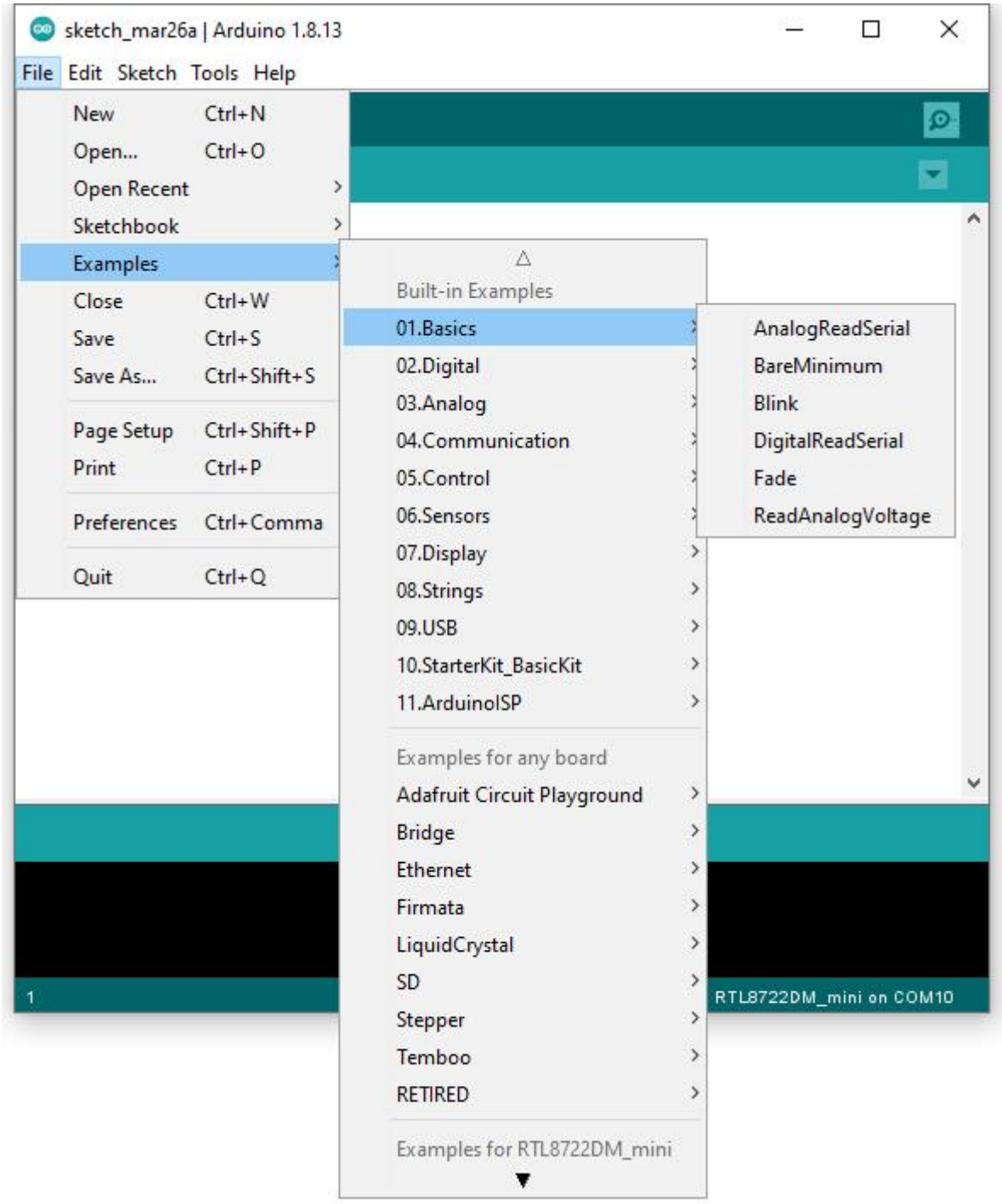


Try the First Example

Step 1. Compile & Upload

Arduino IDE provides many built-in examples, which can be compiled, uploaded and run directly on the boards. Here, we take the "Blink" example as the first try.

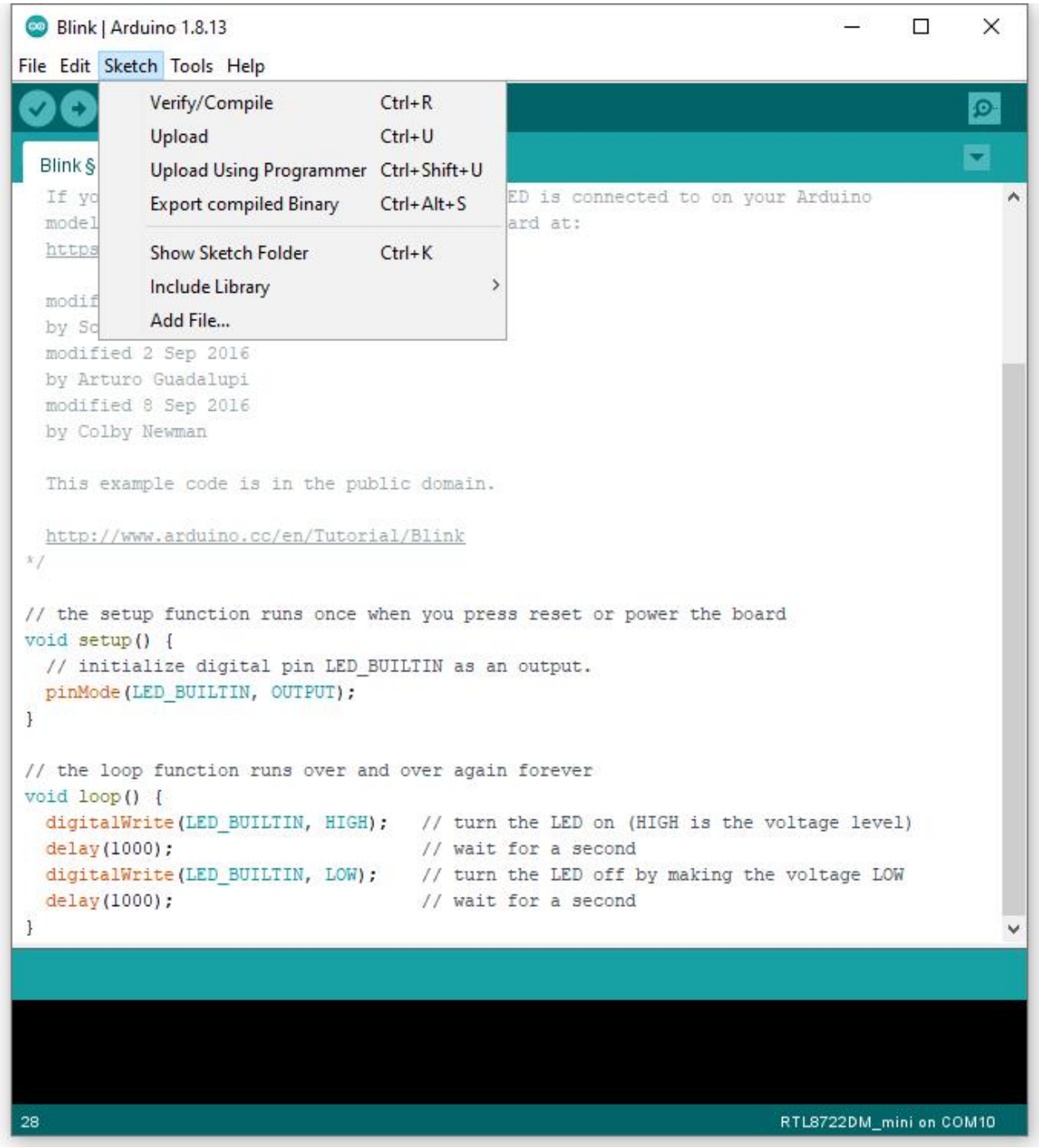
Open *"File" -> "Examples" -> "01.Basics" -> "Blink"*:



Arduino IDE opens a new window with the complete sample code.

There are onboard LED of AMB23, the default “LED_BUILTIN” is blue onboard LED.
Change “LED_BUILTIN” to “LED_B” or “LED_G” for different colour. Onboard LEDs options LED_B and LED_G (blue and green).

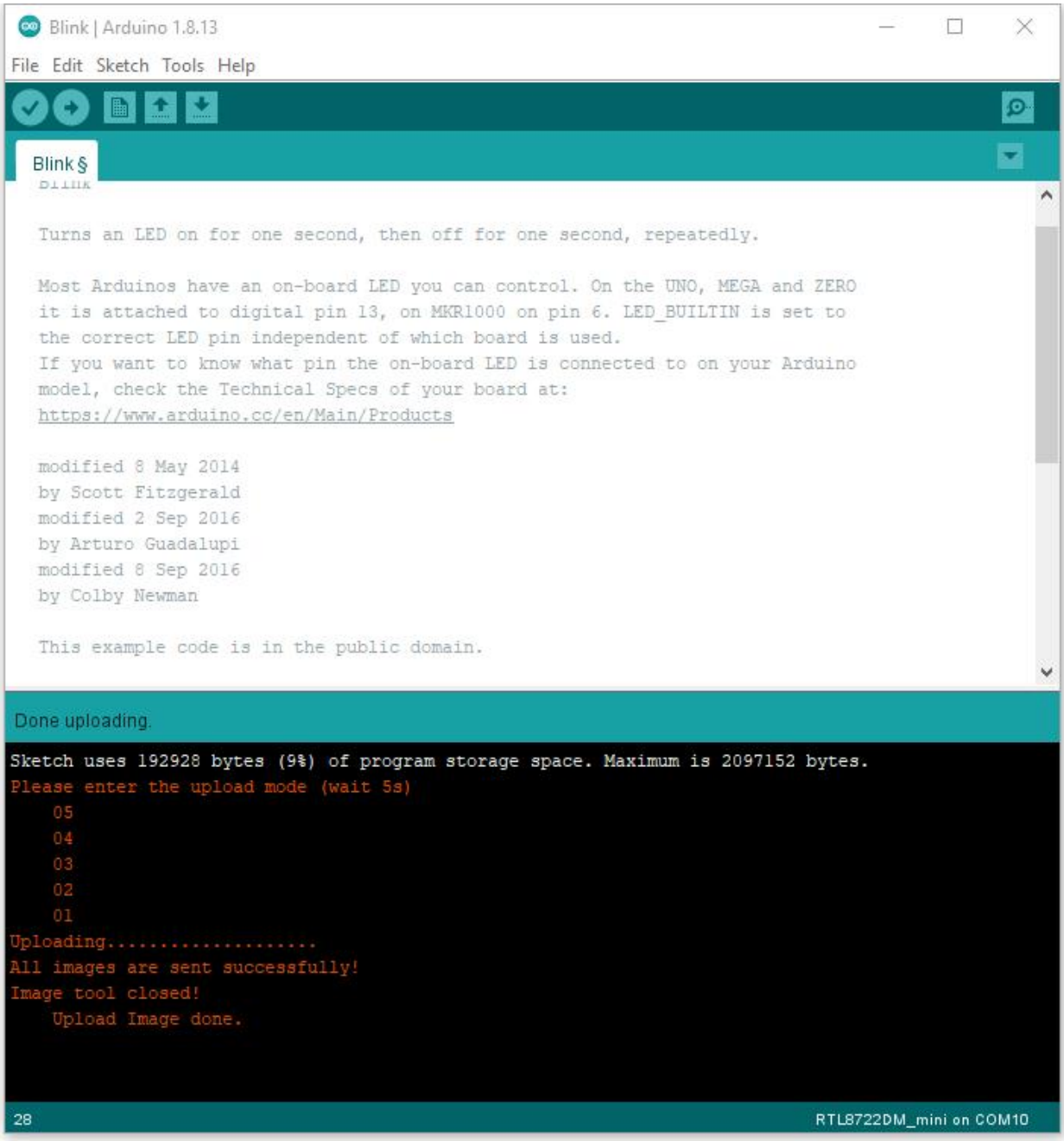
Next, we compile the sample code directly; click *"Sketch"* -> *"Verify/Compile"*



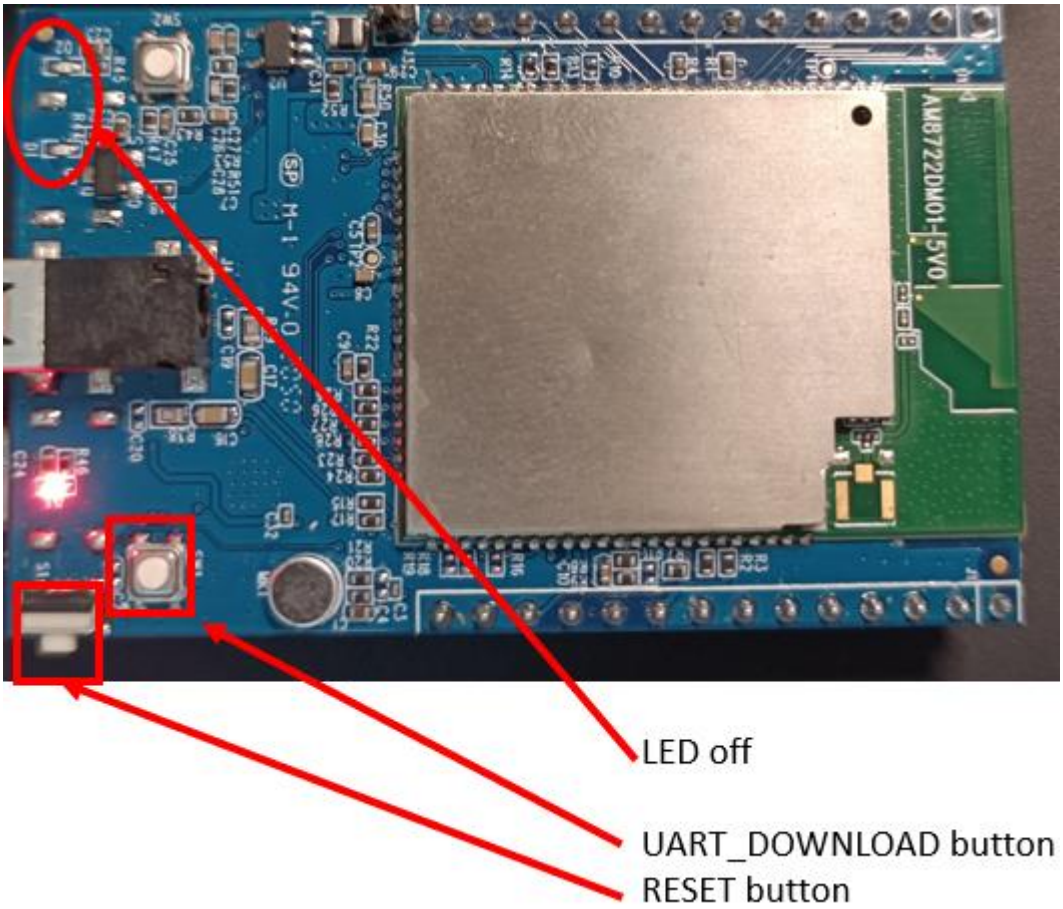
Arduino IDE prints the compiling messages in the bottom area of the IDE window. When the compilation is finished, you will get the message similar to the following figure:

Afterwards, we will upload the compiled code to AMB23.
Please make sure AMB23 is connected to your computer, then click *"Sketch" -> "Upload"*.

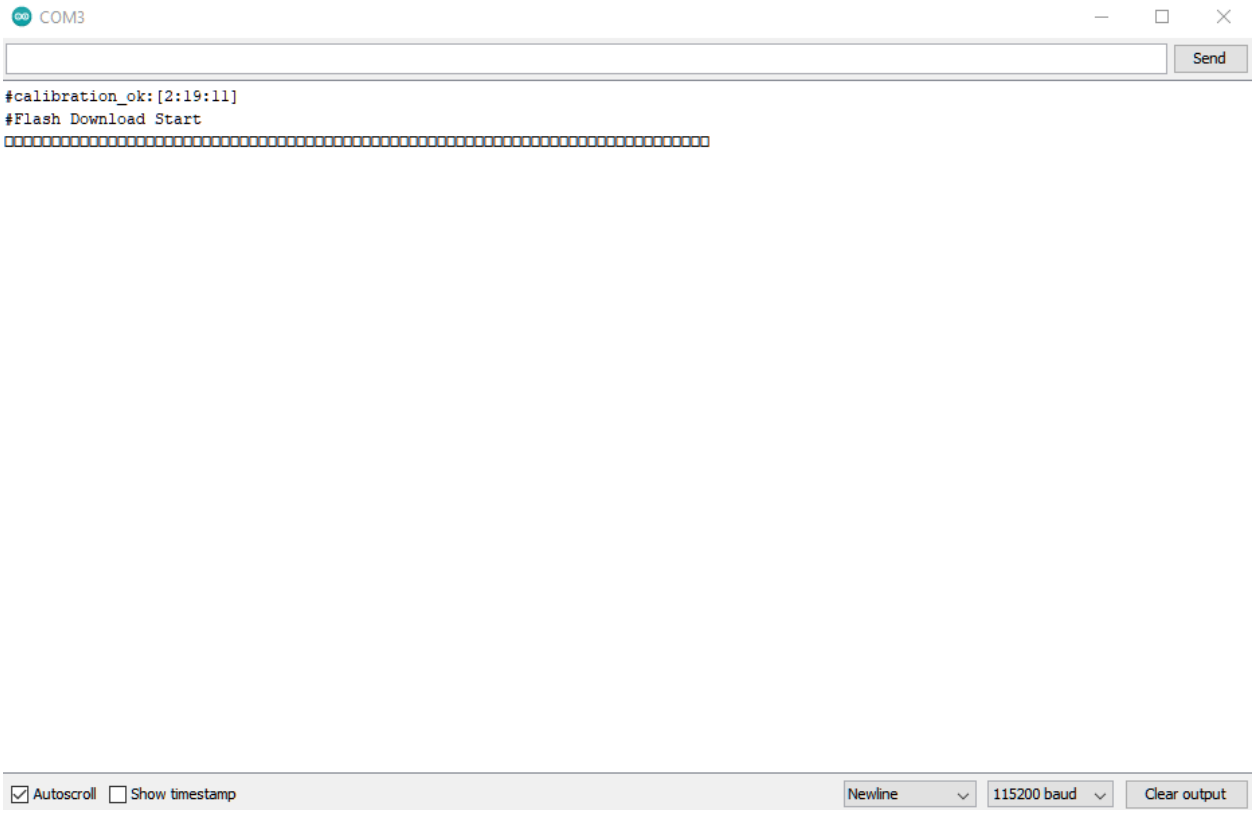
The Arduino IDE will compile first then upload. During the uploading process, users are required to enter the upload mode of the board. Arduino IDE will wait 5s for DEV board to enter the upload mode.



To enter the upload mode, first press and hold the *UART_DOWNLOAD* button, then press the *RESET* button. If success, you should see the onboard green LED and blue LED all turned off.



It is optional for users to check if the board entered the upload mode. Open serial monitor/terminal and look for "#Flash Download Start". Note, it is normal that some serial terminals may show unknown characters as following picture.



Again, during the uploading procedure the IDE prints messages. Uploading procedure takes considerably longer time (about 30 seconds to 1 minute). When upload completed, the "Done uploading" message is printed.

Step 2.Run the Blink example

In each example, Arduino not only provides sample code, but also detailed documentation, including wiring diagram, sample code explanation, technical details, …etc. These examples can be directly used on AMB23.
So, we find the detailed information of the "Blink" example:
https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink

In short, for AMB23, the example can be run on both onboard LEDs (green or blue) or external LED (use any GPIO pins for signal output).
Finally, press the *RESET* button, and you can see the *LED* blinking.

(End)

Note: If you face any issue, please refer to the FAQ and Trouble shooting sections on ../../support/index page.

1.2.2 Peripherals & Examples

Basic Examples

AMB23 (RTL8722DM_MINI) Supported ARDUINO built-in example table

There are many built-in examples in Arduino. In the table below, we list all examples that are compatible with Ameba.

Please refer to the following link to set up Ameba for Arduino IDE.
https://www.amebaiot.com/en/amebad-mini-arduino-getting-started/
Please refer to the following link for Arduino built-in example details.
https://www.arduino.cc/en/Tutorial/BuiltInExamples//

Category	Name	Comment
01. Basics	AnalogRead Serial	Connect potentiometer to 3.3V.
	Bare Minimum	
	Blink	Pin LED_BUILTIN sets to LED_B
	DigitalRead Serial	
	Fade	Replace “led = 9;” by a PWM pin (D4, D5, D7, D12, D13, D14, D17, D20, or D
	ReadAnalog Voltage	
02. Digital	BlinkWitout Delay	The onboard blue LED (LED_B) has been used.
	Button	
	Debounce	
	DigitalInput Pullup	
	StateChange Detection	
	toneKeyboard	Replace “tone(8, notes[thisSensor], 20);” by a PWM pin (D4, D5, D7, D12, D13
	toneMelody	
	tone Multiple	
	tonePitch Follower	
03. Analog	AnalogIn OutSerial	Replace “analogOutPin = 9;” by a PWM pin (D4, D5, D7, D12, D13, D14, D17
	AnalogInput	
	Analog Write Mega	Use PWM pins D4, D5, D7, D12, D13, D14, D17, D20, or D21.
	Calibration	
	Fading	Replace “ledPin = 9;” by a PWM pin (D4, D5, D7, D12, D13, D14, D17, D20, o
	Smoothing	
04. Communication	ASCIITable	
	Dimmer	
	Graph	Connect potentiometer to 3.3V.
	Midi	Use Serial1 and pin D18, or use Serial2 and pin D1.
	MultiSerial	
	Physical Pixel	
	ReadASCII String	Use PWM pin for LED (D4, D5, D7, D12, D13, D14, D17, D20, or D21).
	SerialCall Response	
	Serial CallResponse ASCII	
	SerialEvent	
	SerialPa ssthrough	

Category	Name	Comment
	VirtualColor Mixer	
05. Control	Arrays	Use pins D1, D2, D3, D4, D5, D6.
	ForLoop Iteration	Use pins D1, D2, D3, D4, D5, D6.
	IfStatement Conditional	
	switchCase	
	switchCase2	Use pins D1, D2, D3, D4, D5, D6.
	WhileStatement Conditional	Replace “ledPin = 9;” by a PWM pin (D4, D5, D7, D12, D13, D14, D17, D20, c
06. Display	barGraph	
	RowColumn Scanning	
07. Strings	Character Analysis	
	String Addition Operator	
	String Append Operator	
	StringCase Changes	
	String Characters	
	String Comparison Operators	
	String IndexOf	
	String Length	
	StringLength Trim	
	String Replace	
	String StartsWith EndsWith	
	String Substring	
	String ToInt	

Network Examples

BLE - BLE Battery Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery service is set up on the Ameba Bluetooth stack. A mobile phone is used to connect to the Ameba peripheral device and read the battery data.

Procedure

Ensure that the following Bluetooth apps are installed on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.

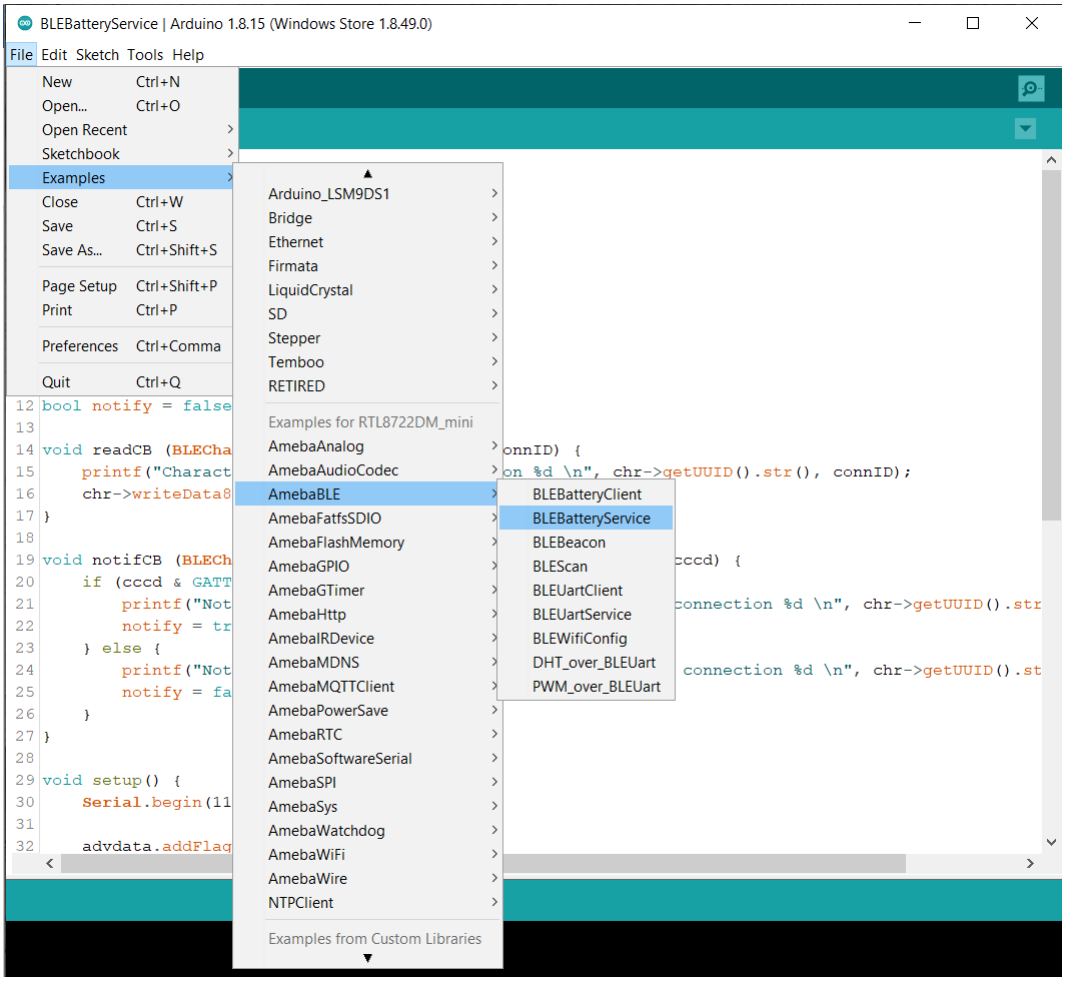
The recommended application is nRF connect, and is available at the links below:

- Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS : https://apps.apple.com/us/app/nrf-connect/id1054362403

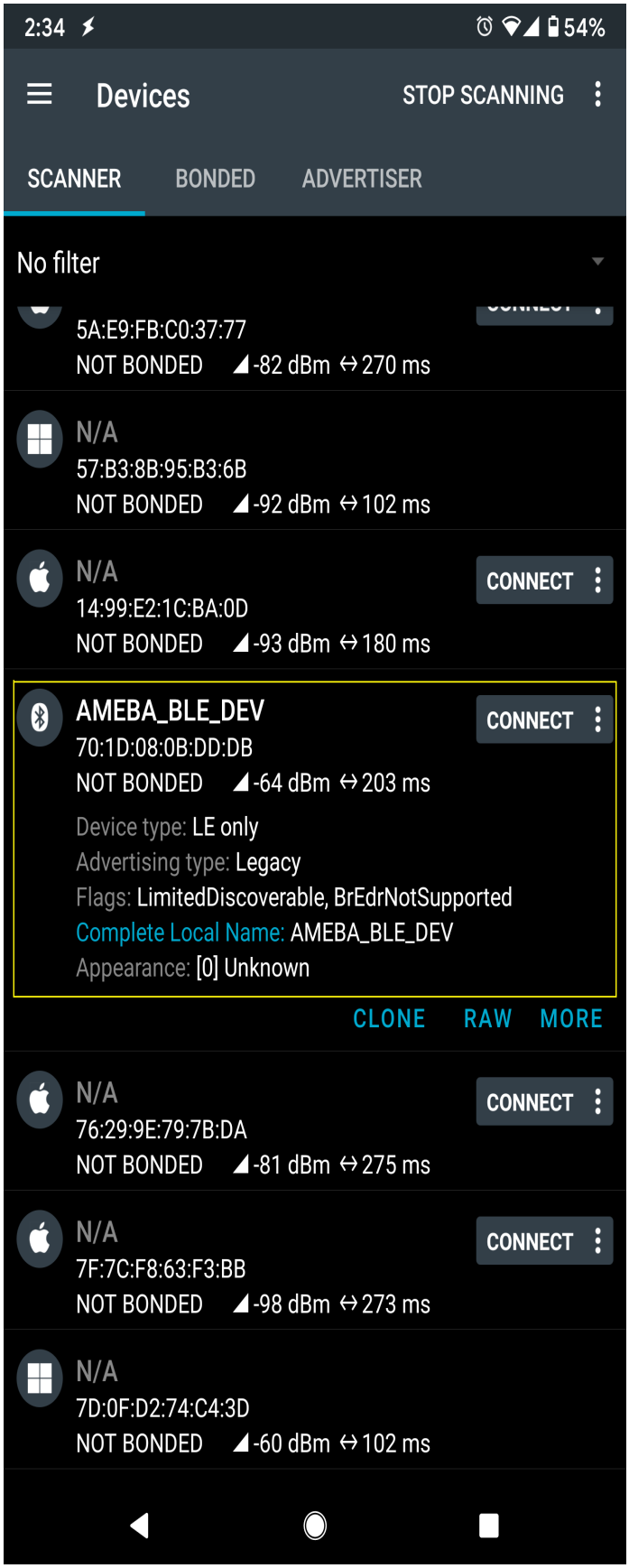
LightBlue is an alternative application that can also be used, but has less features:

- Android: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS : https://apps.apple.com/us/app/lightblue/id557428110

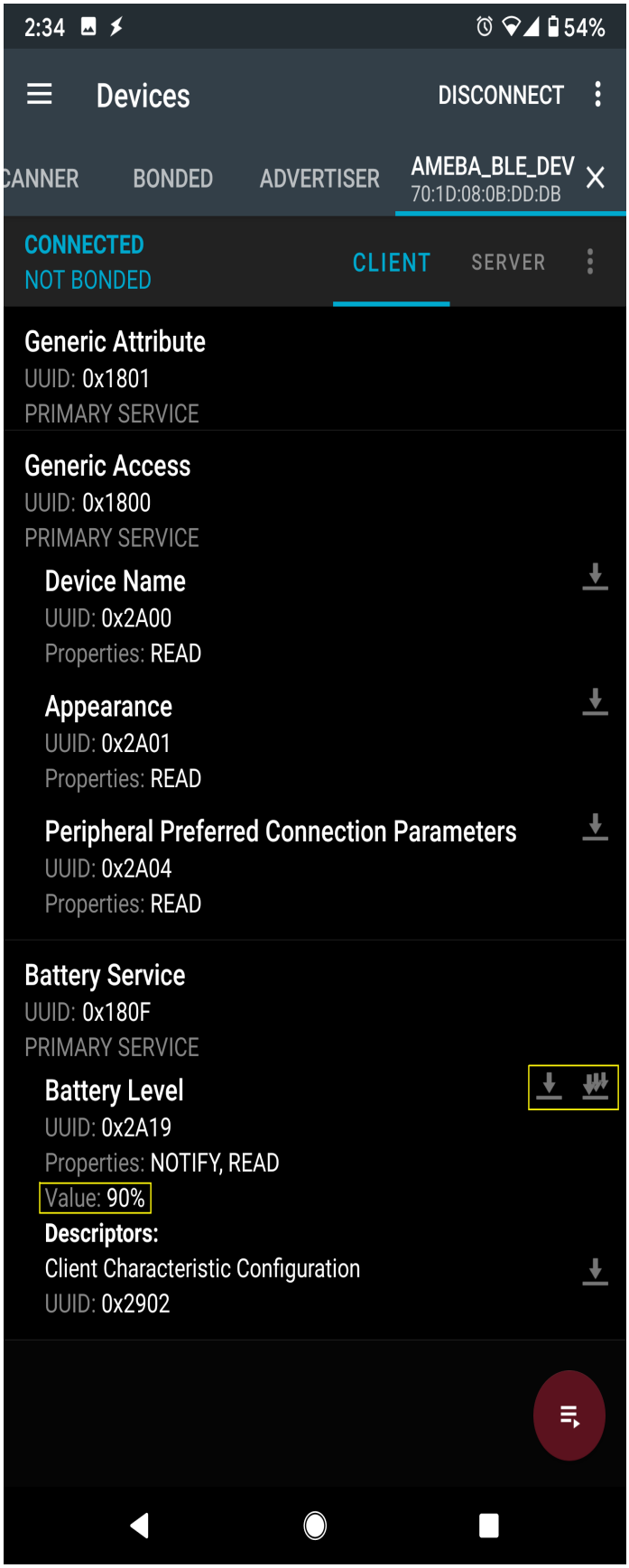
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEBatteryService`”



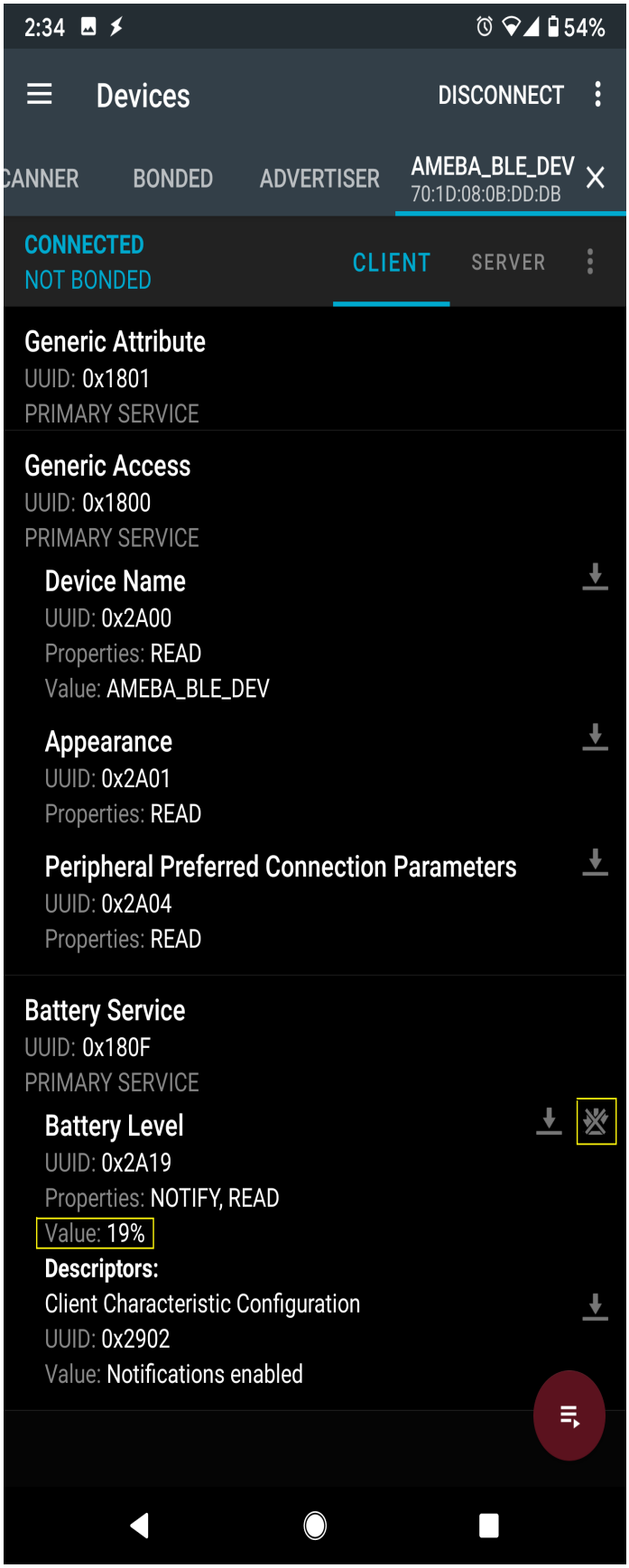
Upload the code and press the reset button on Ameba once the upload is finished.
On your mobile phone, open the Bluetooth app and scan for the Bluetooth signal broadcast by Ameba, it should appear as a device named “AMEBA_BLE_DEV”.


2:34
54%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
No filter
5A:E9:FB:C0:37:77
NOT BONDED -82 dBm 270 ms
N/A
57:B3:8B:95:B3:6B
NOT BONDED -92 dBm 102 ms
N/A
CONNECT
14:99:E2:1C:BA:0D
NOT BONDED -93 dBm 180 ms
AMEBA_BLE_DEV
CONNECT
70:1D:08:0B:DD:DB
NOT BONDED -64 dBm 203 ms
Device type: LE only
Advertising type: Legacy
Flags: LimitedDiscoverable, BrEdrNotSupported
Complete Local Name: AMEBA_BLE_DEV
Appearance: [0] Unknown
CLONE
RAW
MORE
N/A
CONNECT
76:29:9E:79:7B:DA
NOT BONDED -81 dBm 275 ms
N/A
CONNECT
7F:7C:F8:63:F3:BB
NOT BONDED -98 dBm 273 ms
N/A
7D:0F:D2:74:C4:3D
NOT BONDED -60 dBm 102 ms

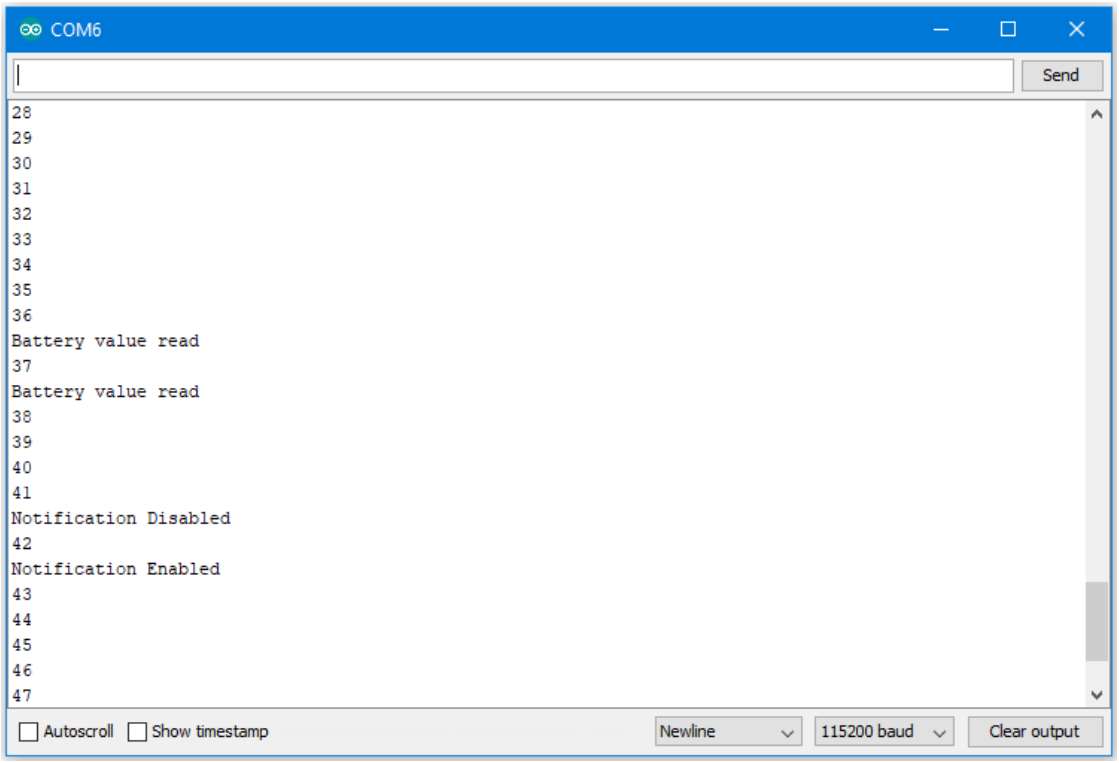
Connect to the Ameba Bluetooth device, and a list of available services should appear. Click on the battery service to expand it, and you can see the battery level data value. The arrows highlighted in the box on the right are used to read data and subscribe to notifications. Click on the single arrow to read the battery level value, and a 90% value will appear.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 90%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Click on the triple arrow to subscribe to updates on the battery level value, and the battery value will start updating by itself.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Value: AMEBA_BLE_DEV
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 19%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

The serial monitor will show the sketch increasing the battery level every second. When you click on either of the arrows, the sketch running on the Ameba will be notified, and will print out the action taken.



Code Reference

BLEService and BLECharacteristic classes are used to create and define the battery service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(GAP_ADTYPE_ADV_IND)` is used to set the advertisement type to a general undirected advertisement that allows for connections.

`setReadCallback()` and `setCCCDCallback()` is used to register functions that will be called when the battery level data is read, or notification is enabled by the user.

`BLE.configServer(1)` is used to tell the Bluetooth stack that there will be one service running.

`addService()` registers the battery service to the Bluetooth stack.

BLE – BLE Beacon

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

A BLE beacon broadcasts its identity to nearby Bluetooth devices, to enable the other devices to determine their location relative to the beacon, and to perform actions based on information broadcasted by the beacon.

Example applications of beacons include indoor positioning system, location-based advertising and more.

From the definition of its purpose as a broadcast device, a BLE beacon thus cannot be connected to, and can only send information in its Bluetooth advertisement packets.

There are several BLE beacon protocols. The Ameba BLEBeacon library supports the iBeacon and AltBeacon protocols.

Procedure

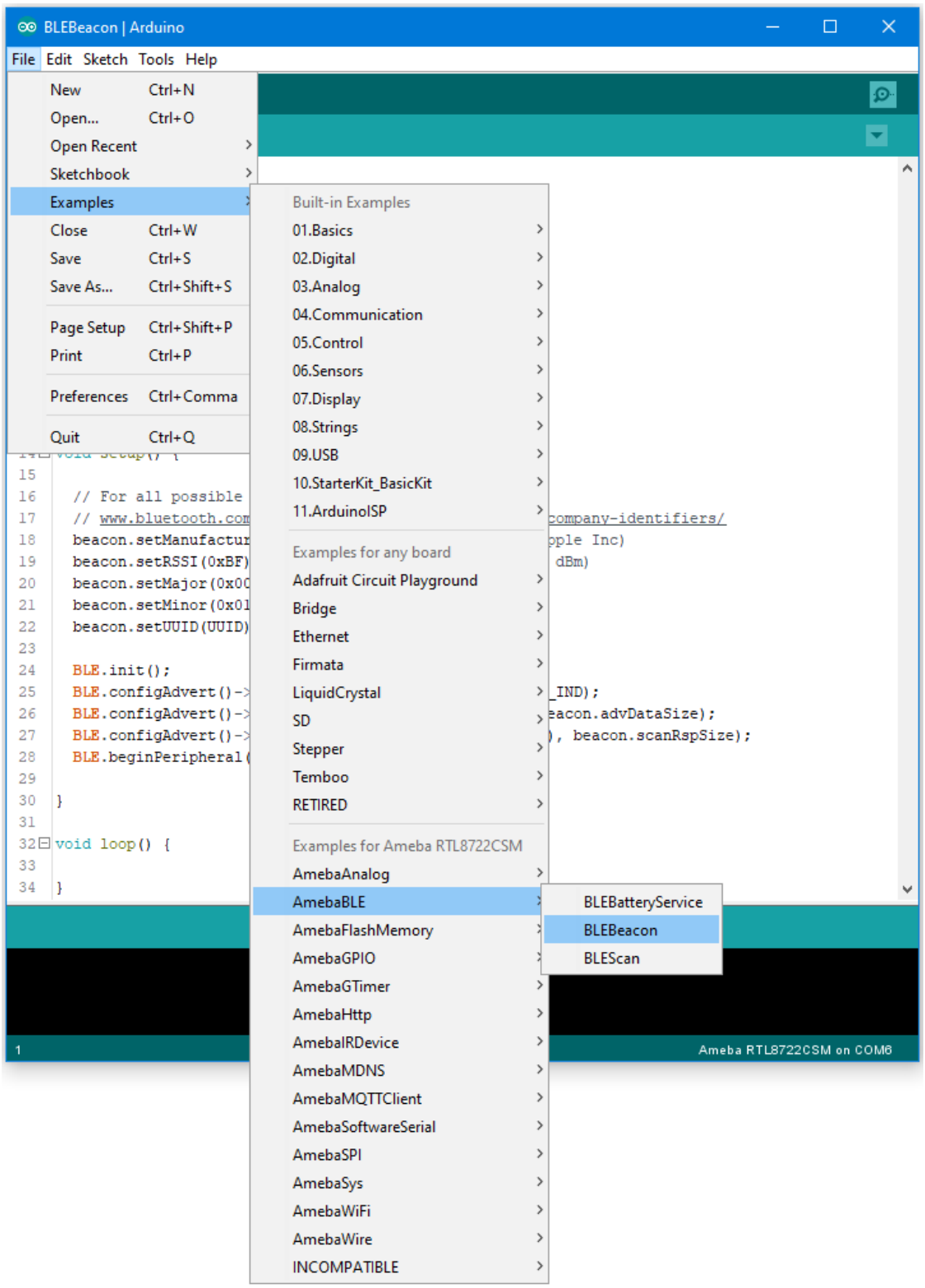
First, you need to install some Bluetooth apps on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.
The recommended application is nRF connect, and is available at the links below:

- **Android** : https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS :https://apps.apple.com/us/app/nrf-connect/id1054362403

LightBlue is an alternative application that can also be used, but has less features:

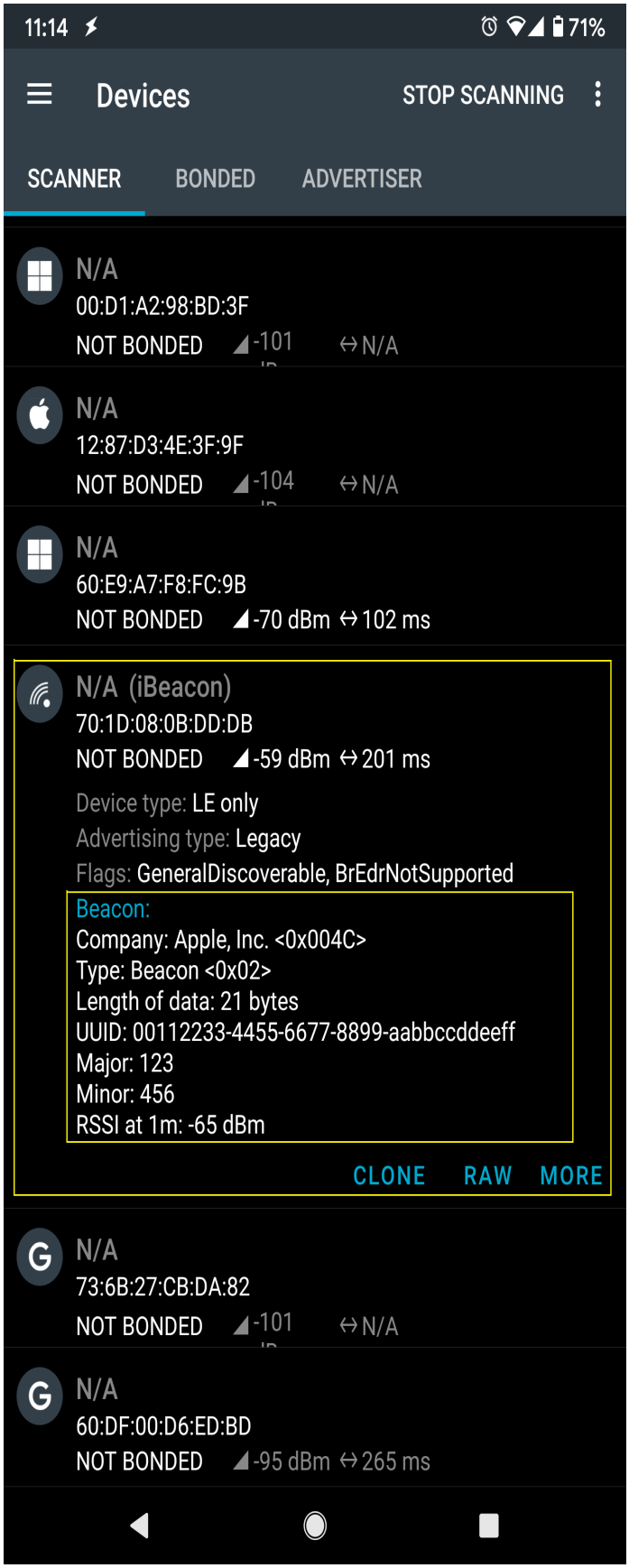
- **Android** : https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS :https://apps.apple.com/us/app/lightblue/id557428110

Open the example, "`Files" -> "Examples" -> "AmebaBLE" -> "BLEBeacon`"



Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Bluetooth app and scan for the beacon signal broadcast by Ameba.


11:14
71%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
N/A
00:D1:A2:98:BD:3F
NOT BONDED
-101
N/A
N/A
12:87:D3:4E:3F:9F
NOT BONDED
-104
N/A
N/A
60:E9:A7:F8:FC:9B
NOT BONDED
-70 dBm
102 ms
N/A (iBeacon)
70:1D:08:0B:DD:DB
NOT BONDED
-59 dBm
201 ms
Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 00112233-4455-6677-8899-aabbccddeeff
Major: 123
Minor: 456
RSSI at 1m: -65 dBm
CLONE
RAW
MORE
N/A
73:6B:27:CB:DA:82
NOT BONDED
-101
N/A
N/A
60:DF:00:D6:ED:BD
NOT BONDED
-95 dBm
265 ms

If you happen to be in an environment with multiple BLE beacons, you can tap the entries to expand them, and verify that the beacon data is identical to the data in the sketch.

Code Reference

`setRssi()` is used to set the received signal strength indicator (rssi) data field for a beacon. The specification states that this should be the received signal strength from the beacon at a 1 meter distance. With no method to measure this, it is set to -65dBm as an estimate.

`setMajor()` and `setMinor()` are used to set the two data fields. The purpose of these data are left for the manufacturer of the beacon to define, and can be used in any way.

`setUUID()` is used to give the beacon a universally unique identifier (UUID). This is a 128-bit number usually expressed as a hexadecimal string. It is used to identify each unique beacon, and can be randomly generated for free online.

The BLEBeacon library includes both iBeacon and AltBeacon classes, replace line 6 iBeacon with altBeacon to create an AltBeacon instead. The data fields are mostly the same, with only minor changes, please look at the header files for more details.

`BLE.init()` is used to allocate memory and prepare Ameba for starting the Bluetooth stack.

`BLE.configAdvert()` is used to configure the Bluetooth advertisement settings, to which we pass the beacon data and set the device as non-connectable.

`BLE.beginPeripheral()` starts Ameba in Bluetooth peripheral mode, after which it will begin to advertise with the beacon data provided.

BLE – BLE Scan

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

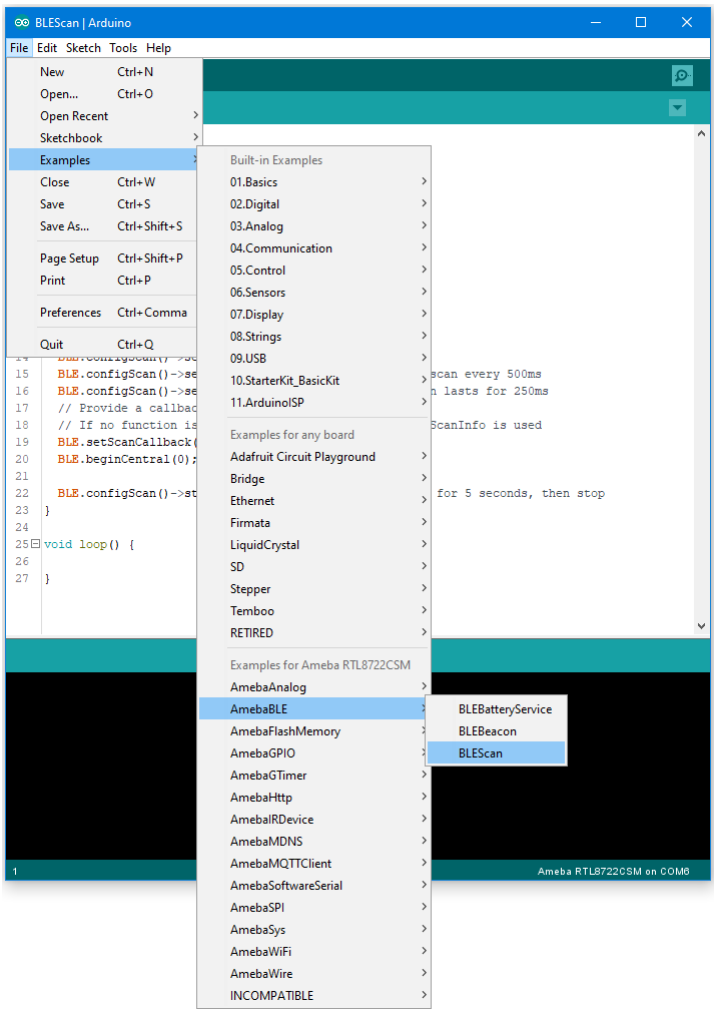
Example

Introduction

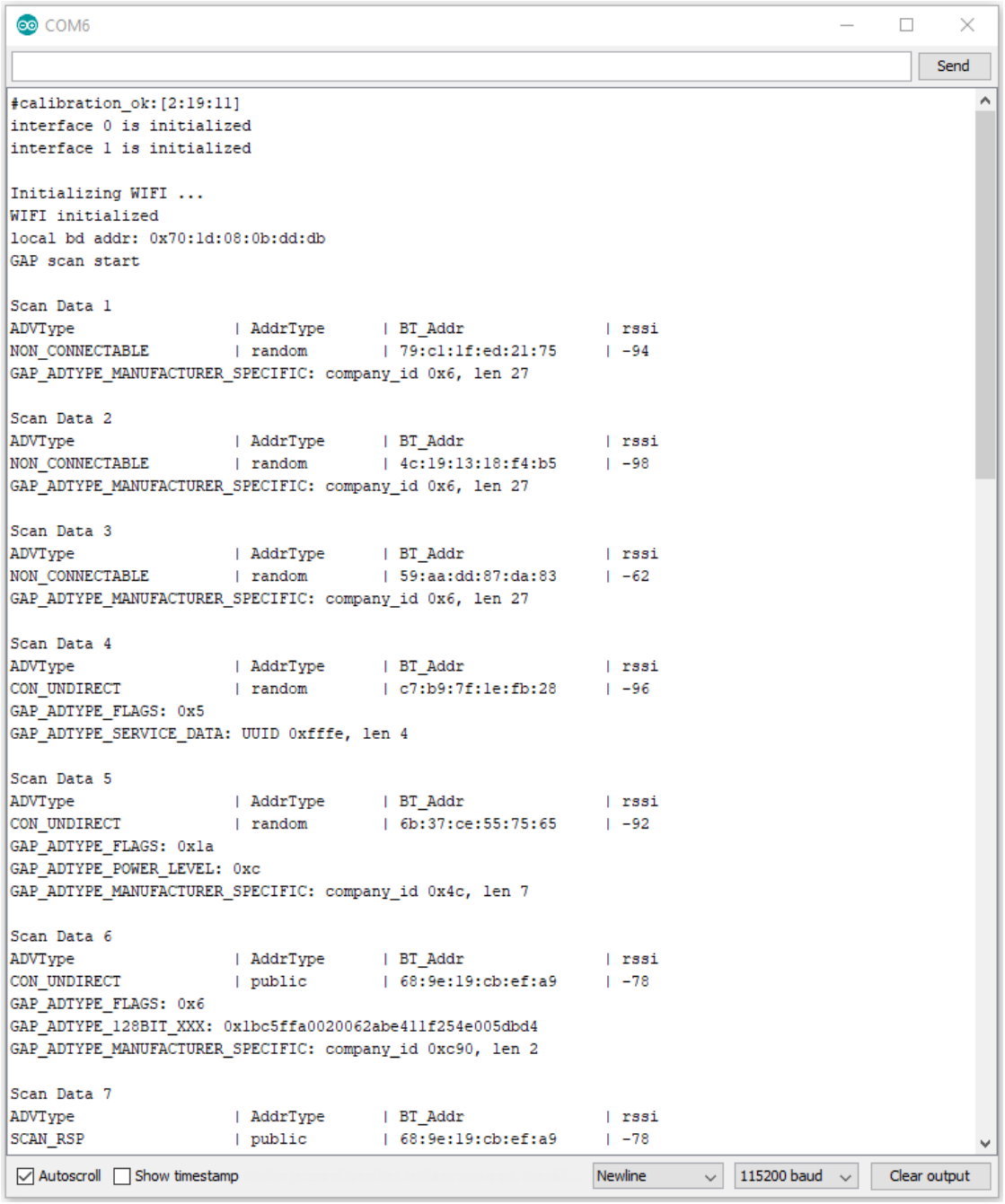
This example configures the Ameba as a Bluetooth central device, uses the scan functionality to scan for other Bluetooth devices, and prints out the results to the serial monitor.

Procedure

Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEScan`”



Upload the code and press the reset button on Ameba once the upload is finished.
Open the Arduino serial monitor, and you should see the scan results of nearby Bluetooth devices formatted and printed out.



If you have the Bluetooth app nRF Connect installed, you can also use it to send out Bluetooth advertisements for the Ameba to pick up.

Code Reference

`setScanMode(GAP_SCAN_MODE_ACTIVE)` is used to set the scan mode. Active scanning will request for an additional scan response data packet from a device when it is found. Passive scanning will only look at the advertisement data, and not request for additional data.

`setScanInterval()` and `setScanWindow()` are used to set the frequency and duration of scans in milliseconds. A scan will start every interval duration, and each scan will last for the scan window duration. The scan window duration

should be lesser or equal to the scan interval. Set a short interval to discover devices rapidly, set a long interval to conserve power.

`setScanCallback(scanFunction)` is used to register a function to be called when scan results are received. This can be used to set a user function for additional processing of scan data, such as looking for a specific device. If no function is registered, the scan results are formatted and printed to the serial monitor by default.

`beginCentral(0)` is used to start the Bluetooth stack in Central mode. The argument 0 is used to indicate that no clients will be operating in central mode.

`startScan(5000)` is used to start the scanning process for a specified duration of 5000 milliseconds. The scan will repeat according to the set scan interval and scan window values. After 5000 milliseconds, the scan process will stop, and will be ready to be started again.

BLE – Battery Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Introduction

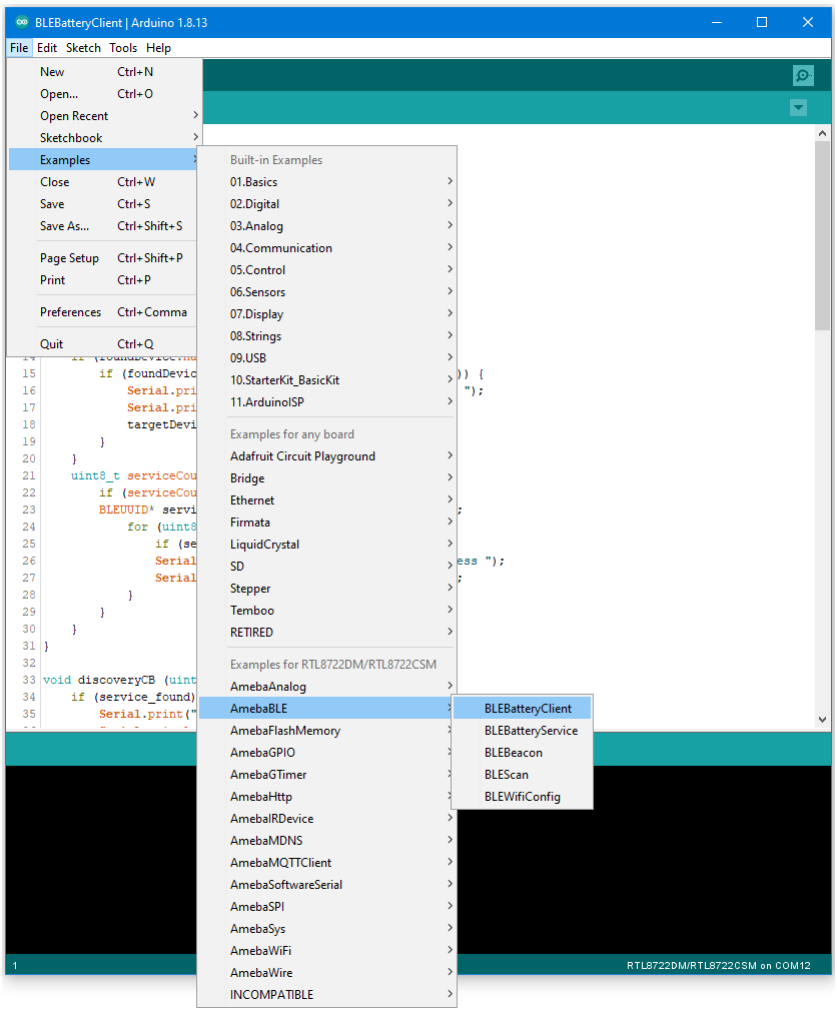
BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery client is set up on the Ameba Bluetooth stack. The client connects to another Ameba board running the corresponding BLE battery service to read the battery level data.

Procedure

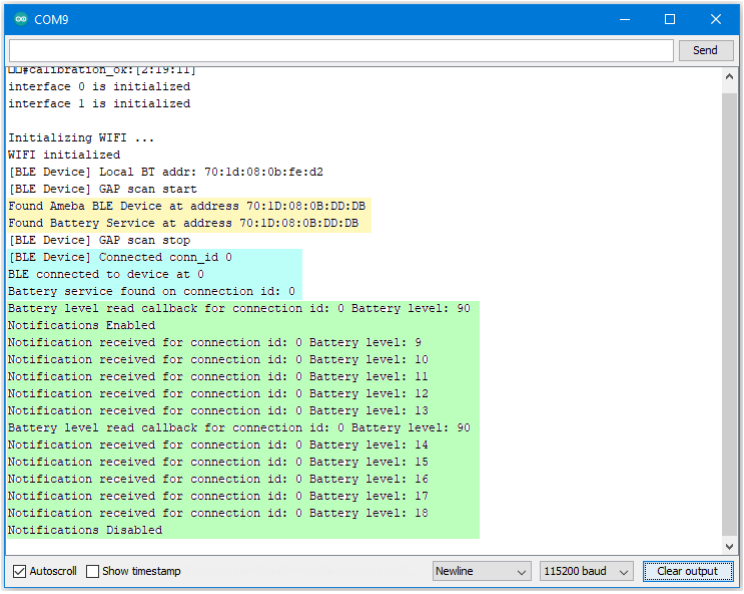
On the first Ameba board, upload the BLEBatteryService example code and let it run.

For the second Ameba board, open the example "`Files`" -> "`Examples`" -> "`AmebaBLE`" -> "`BLEBatteryClient`".



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor and observe the log messages as the Ameba board with the battery client scans, connects, and reads data from the Ameba board with the battery service.



Highlighted in yellow, the Ameba board with the battery client first scans for advertising BLE devices with the advertised device name "AMEBA_BLE_DEV" and the advertised service UUID of 0x180F representing the battery service.

After finding the target device, the Ameba board with the battery client forms a BLE connection and searches for a battery service on the connected device, highlighted in blue.

With the client connected to the service, the battery client begins to read data using both regular data reads and notifications, highlighted in green.

Code Reference

BLEClient is used to create a client object to discover services and characteristics on the connected device.

- `setNotifyCallback()` is used to register a function that will be called when a battery level notification is received.
- `BLE.configClient()` is used to configure the Bluetooth stack for client operation.
- `addClient(connID)` creates a new BLEClient object that corresponds to the connected device.

BLE – WiFi Configuration Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

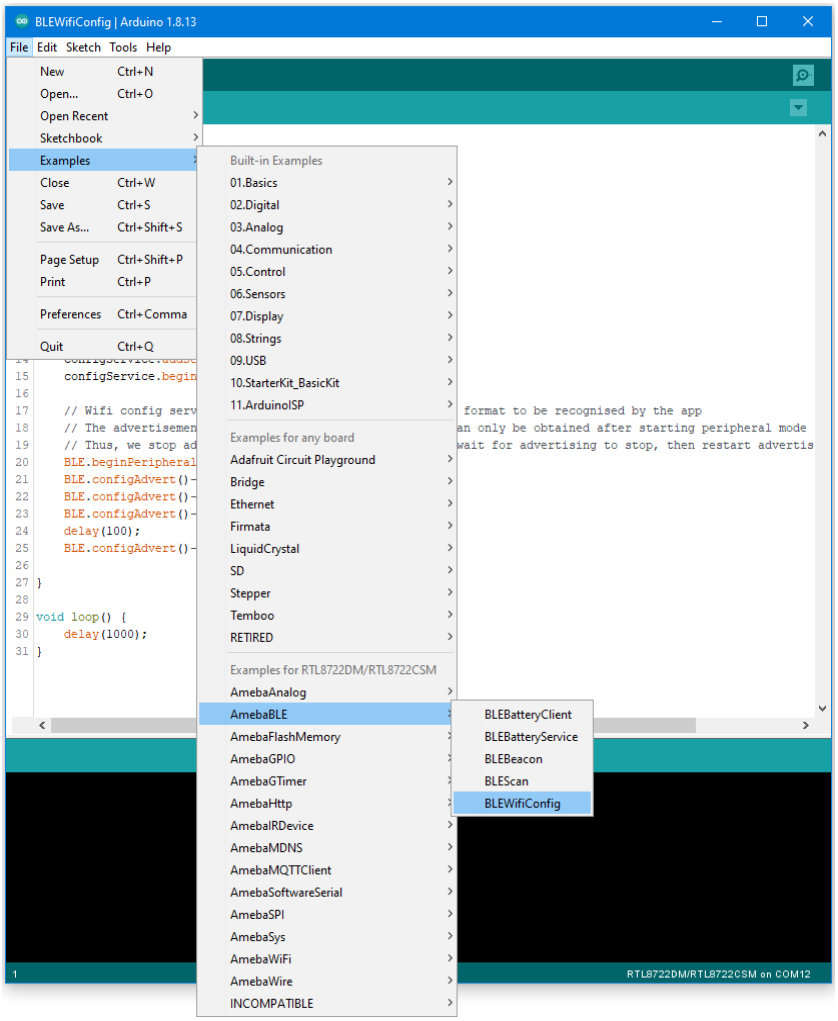
In this example, a WiFi configuration service is set up on the Ameba Bluetooth stack. A mobile phone with the configuration app connects to the Ameba device using BLE and configures the Ameba to connect to the correct WiFi access point.

Procedure

Ensure that the Realtek WiFi configuration app is installed on your mobile phone, it is available at:

– Google Play Store: https://play.google.com/store/apps/details?id=com.rtk.btconfig
– Apple App Store: https://apps.apple.com/sg/app/easy-wifi-config/id1194919510

Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEWifiConfigService`”.

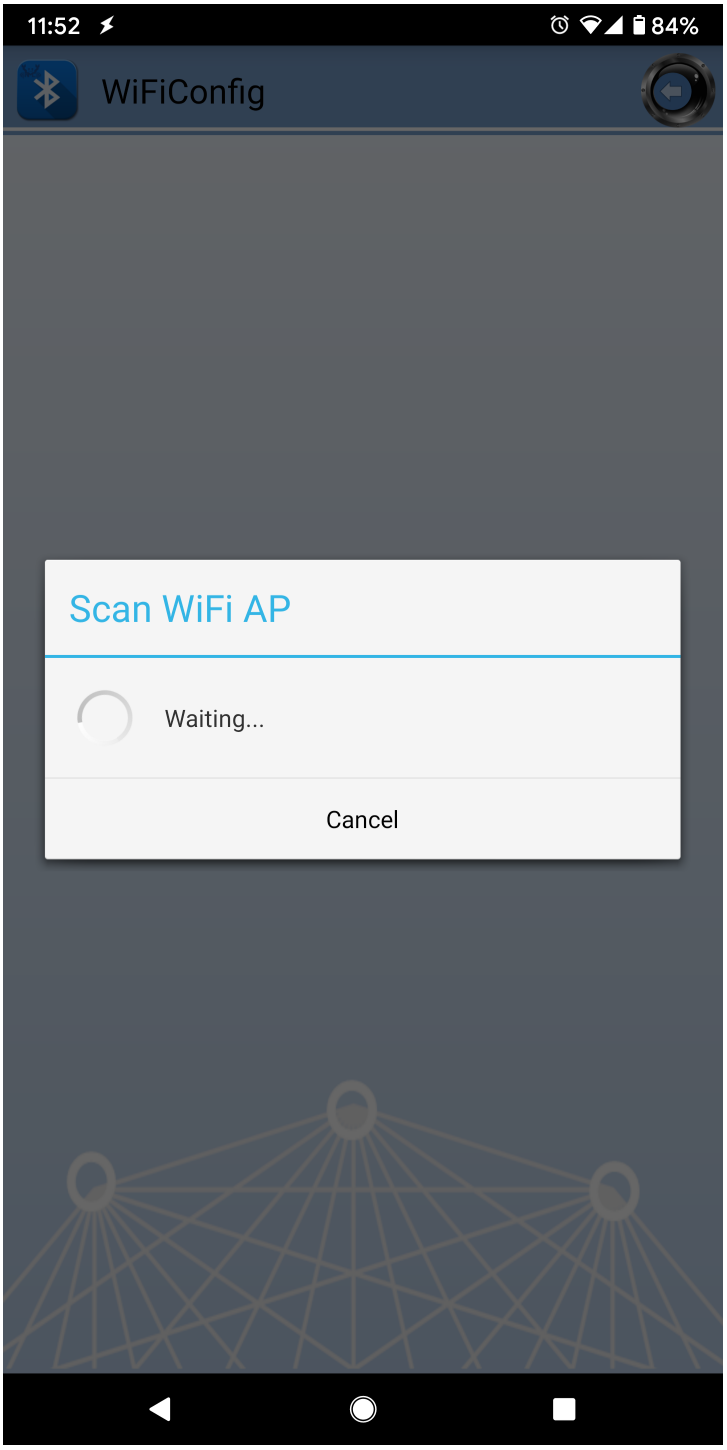


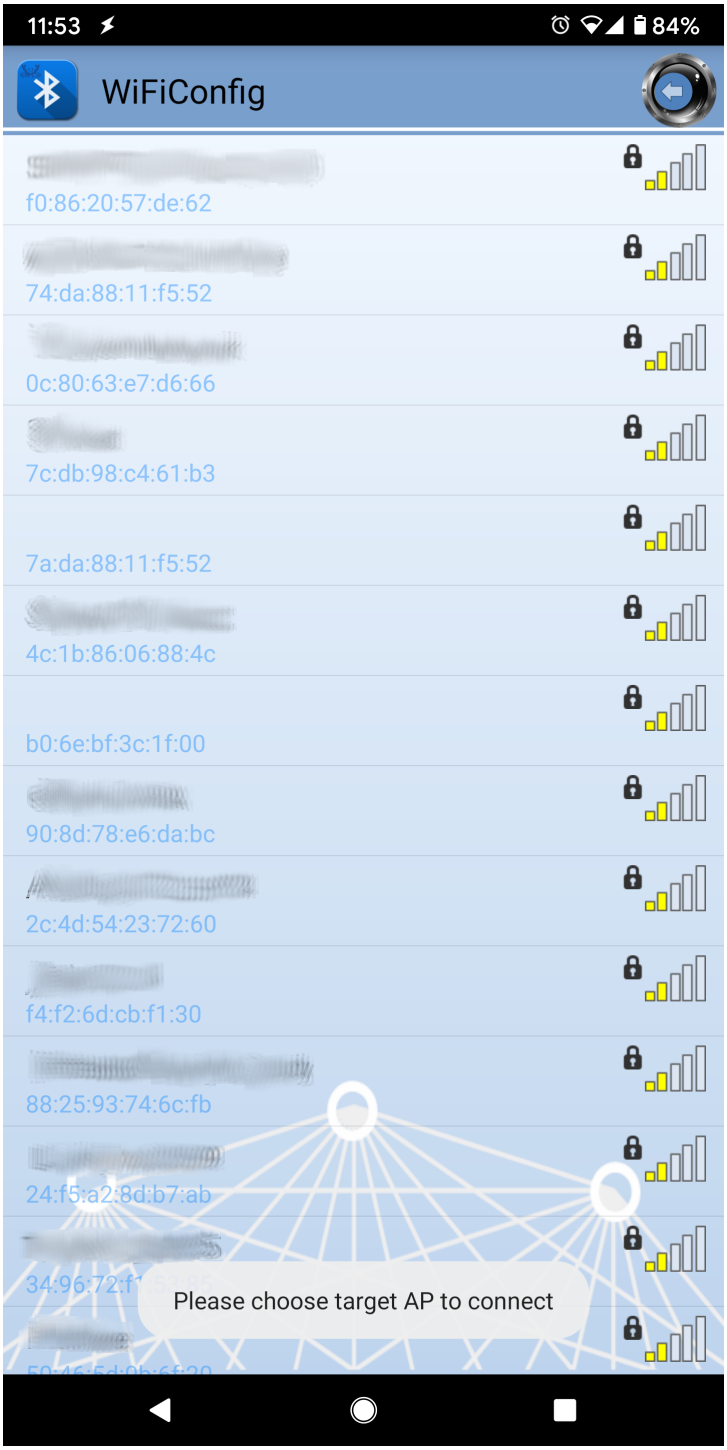
Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Realtek WiFiConfig app and tap the round button to scan for Ameba boards.

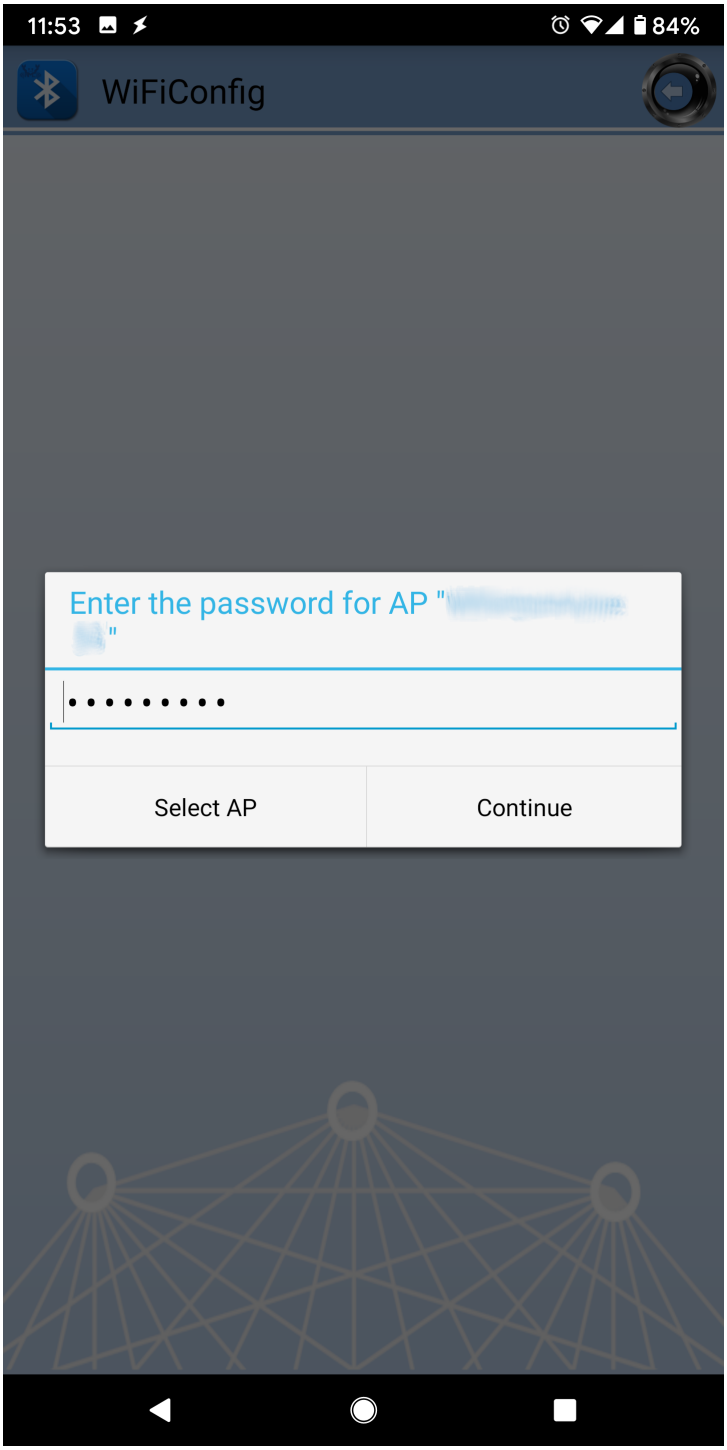
Select the correct Ameba board from the scan results. The app will connect to the Ameba board and ask the board to scan for WiFi networks and send the scan results back to the app using BLE.


4:16
82%
WiFiConfig
Choose device to configure
Ameba_0BFED2
70:1D:08:0B:FE:D2
Cancel

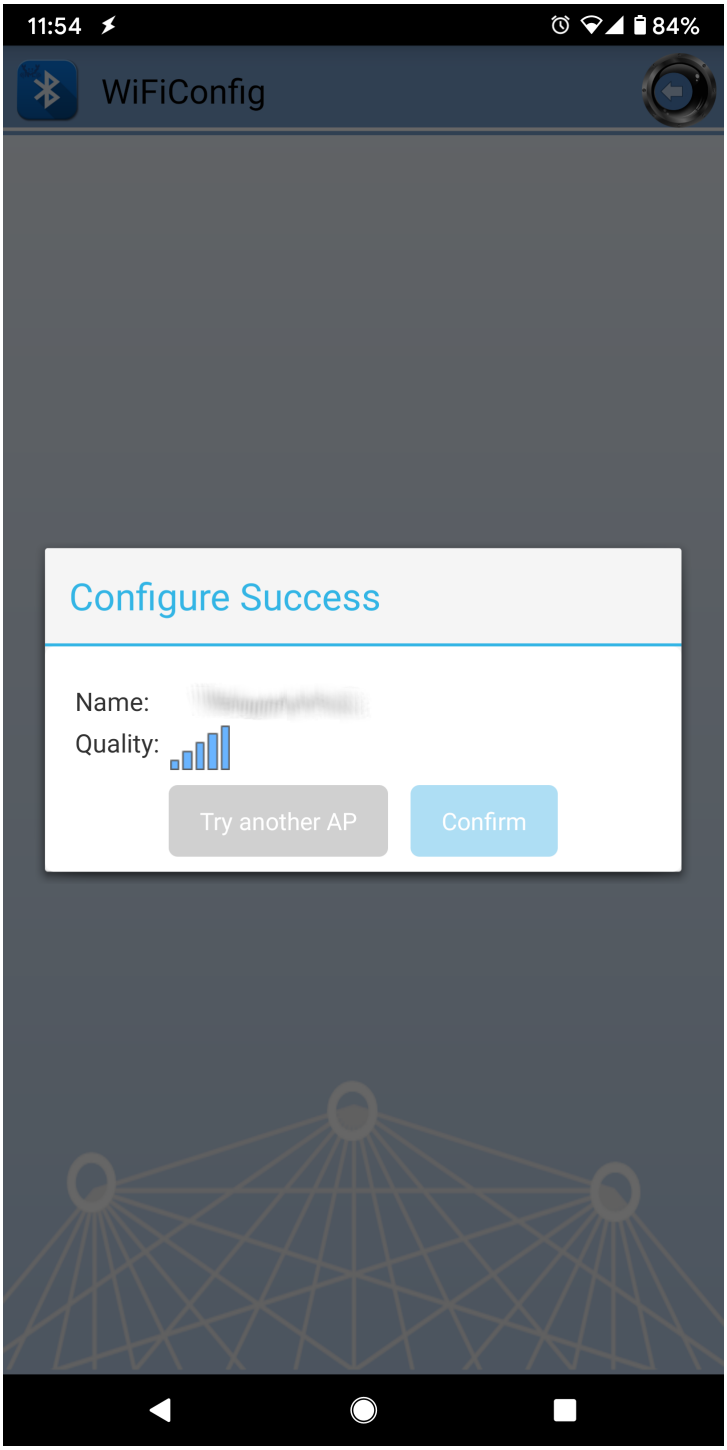

11:52
84%
WiFiConfig
Scan WiFi AP
Waiting...
Cancel



If your phone is currently connected to a WiFi network, the app will ask for the WiFi password to connect the Ameba board to the same WiFi network. Tap “Select AP” to choose another WiFi network, or enter the password and tap continue to connect Ameba to the selected WiFi network.



After the Ameba board connects to the WiFi network, the following message will be shown. Tap “Try another AP” to connect to another WiFi network or tap “Confirm” to keep the current WiFi network and disconnect BLE from the Ameba board.



Code Reference

BLEWifiConfigService is used to create an instance of the WiFi configuration service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(configService.advData())` is used to set the correct advertisement data necessary for the phone app to find the Ameba Bluetooth device.

BLE – BLE UART Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

Example

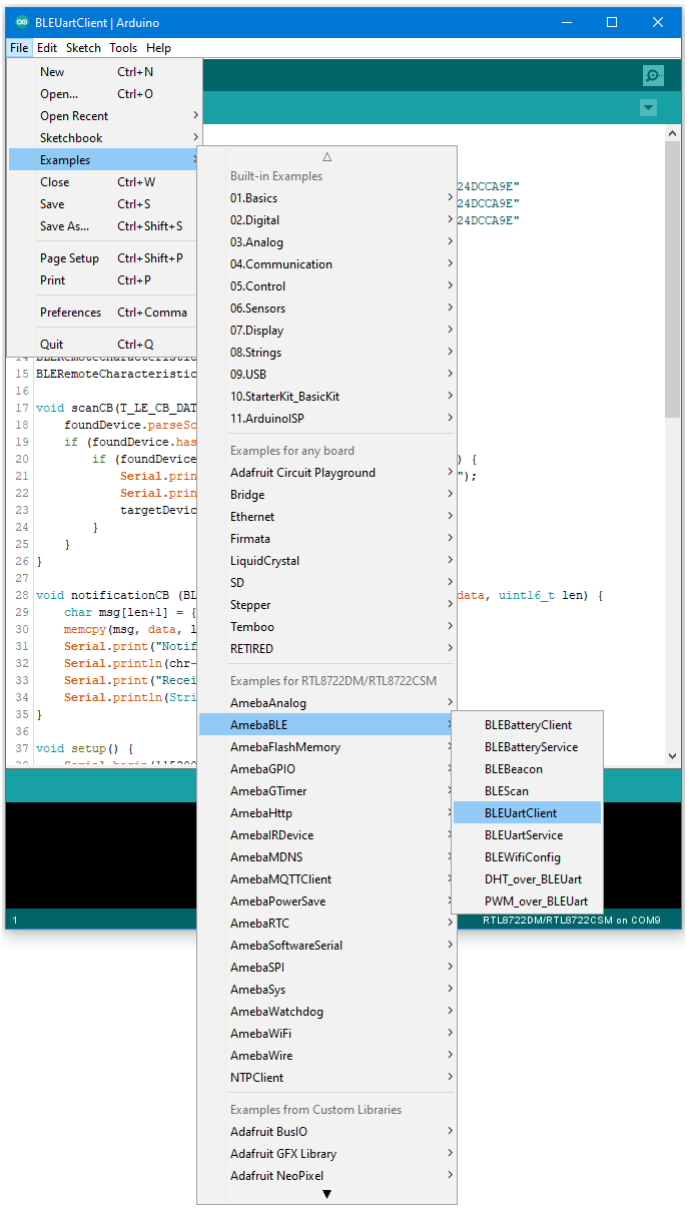
Introduction

In this example, two RTL8722 boards are connected using BLE. One board runs a BLE UART service, while the other connects to the service using a client and both boards are able to communicate with text messages over the UART service.

Procedure

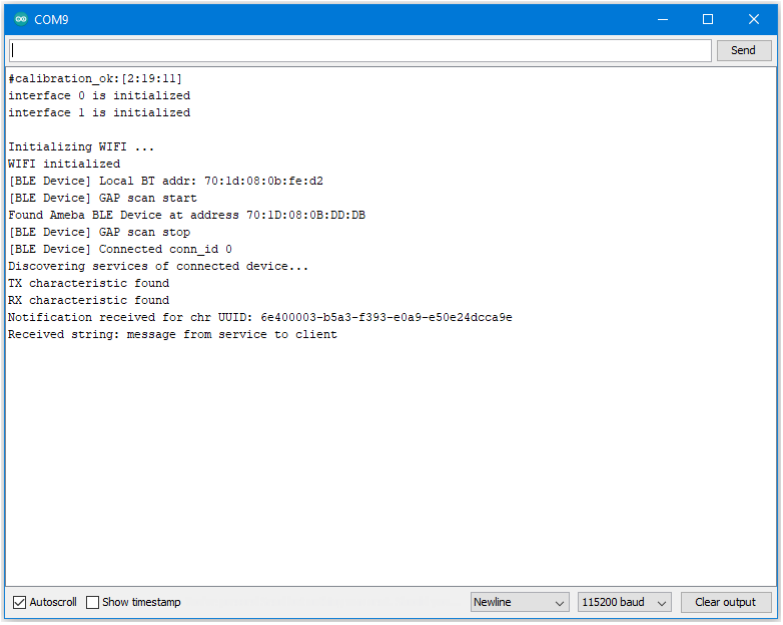
On the first board, upload the BLE UART service example code. Refer to the example guide for detailed instructions.

For the second board, open the example, "`Files" -> "Examples" -> "AmebaBLE" -> "BLEUart-Client`".



Upload the code and press the reset button on Ameba once the upload is finished.

Reset the UART service board first, wait for the BLE advertisement process to begin, and reset the UART client board. The client board should scan, discover, and connect to the service board. After connecting, the client board will verify that the correct UART service exists on the service board, before enabling notifications on the TX characteristic. Any message typed in the serial terminal will be sent to the other board using the UART service.



Code Reference

The BLEClient class is used to discover the services that exist on a connected BLE device. The discovery process will create BLERemoteService, BLERemoteCharacteristic and BLERemoteDescriptor objects corresponding to the services, characteristics and descriptors that exist on the connected device. These objects can then be used to read and write data to the connected device.

BLE – BLE UART Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS smartphone

Example

Introduction

With BLE, application data is sent and received using the GATT system. GATT uses services, characteristics, and attributes to organise data and control how the data can be read from and written to. The Bluetooth SIG specification for BLE includes several predefined services for common applications, but users are free to implement custom services and characteristics to best fit their data structure and application needs

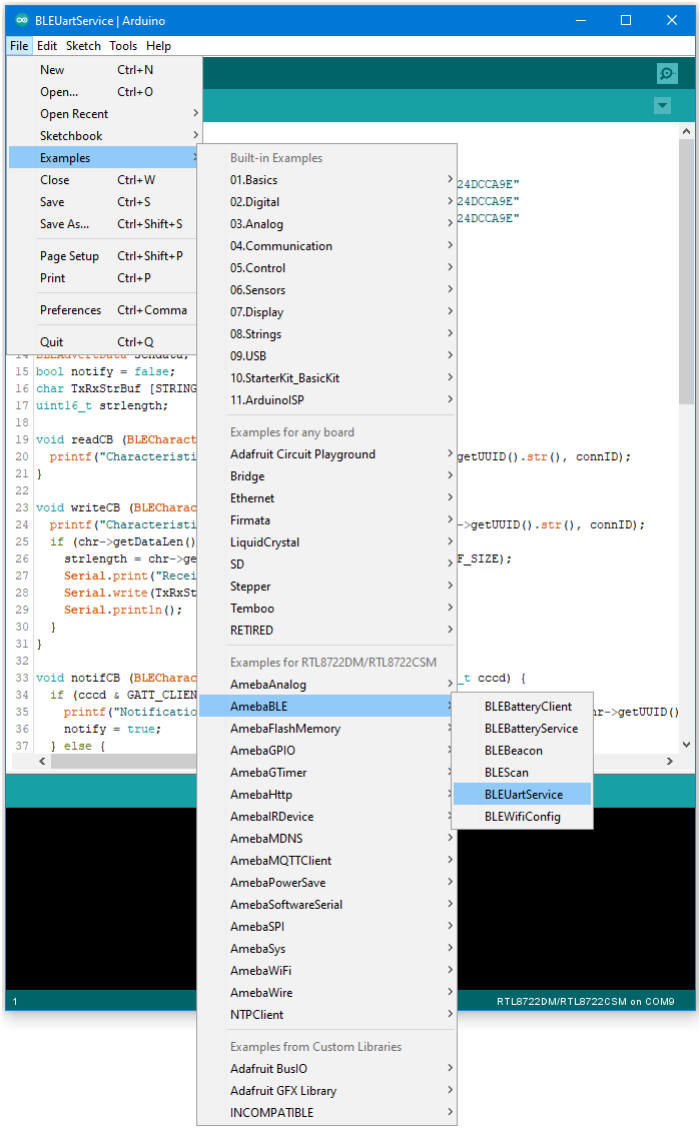
In this example, the BLEService and BLECharacteristic classes are used to implement a custom service for transmitting ASCII characters similar to regular UART. This custom service is the Nordic UART Service, which is supported in several smartphone apps.

Procedure

Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

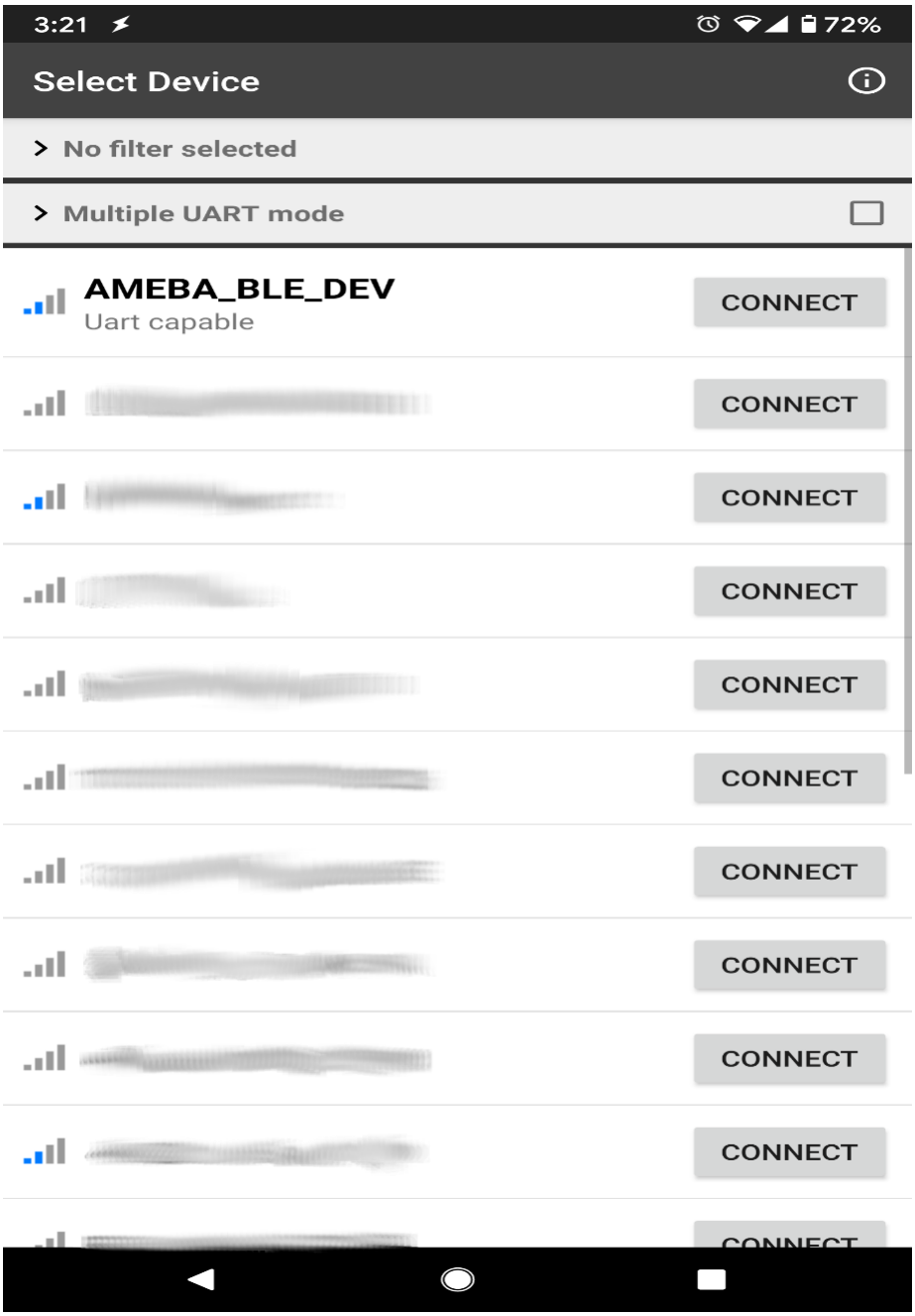
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

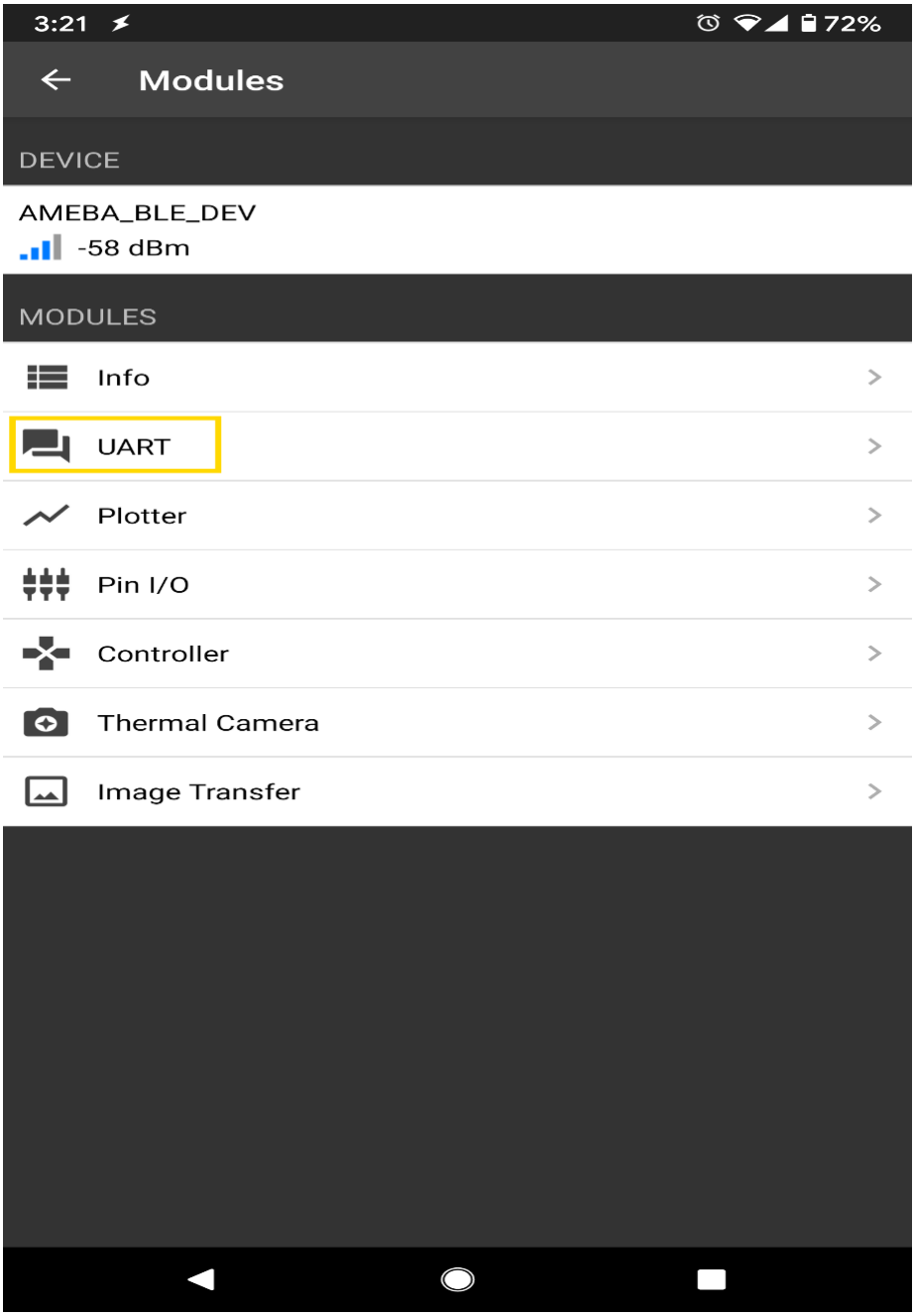
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEUartService`”.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app. Note that the BLE UART service on the Ameba board will only work with the UART and Plotter functions in the Bluefruit Connect app, other functions (Pin I/O, Image Transfer) require other BLE services

that are not included in this example.

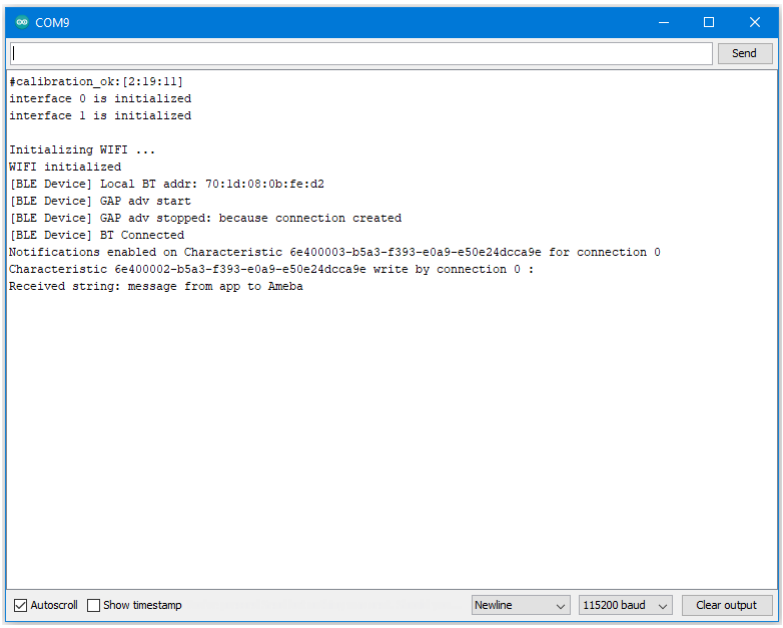




In the UART terminal section of the app, enter a message and click send. You should see the message appear in the Arduino serial monitor.

In the Arduino serial monitor, enter a message and click send. The message will appear in the smartphone app.


3:22
72%
UART
MQTT
message from app to Ameba
message from Ameba to smartphone app
SEND
Sent: 26 bytes
Received: 37 bytes



Code Reference

The BLECharacteristic class is used to create two characteristics, one for receive (Rx) and one for transmit (Tx), and added to a service created with the BLEService class.

The required read/write/notify properties are set for each characteristic using the `set__Property()` methods, and callback functions are registered using the `set__Callback()` methods. The required buffer size is also set for each characteristic so that it has enough memory to store a complete string.

When data is written to the receive characteristic, the registered callback function is called, which prints out the received data as a string to the serial monitor.

When data is received on the serial port, it is copied into the transmit characteristic buffer, and the `notify()` method is used to inform the connected device of the new data.

BLE – DHT over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21
- Android / iOS smartphone

Example

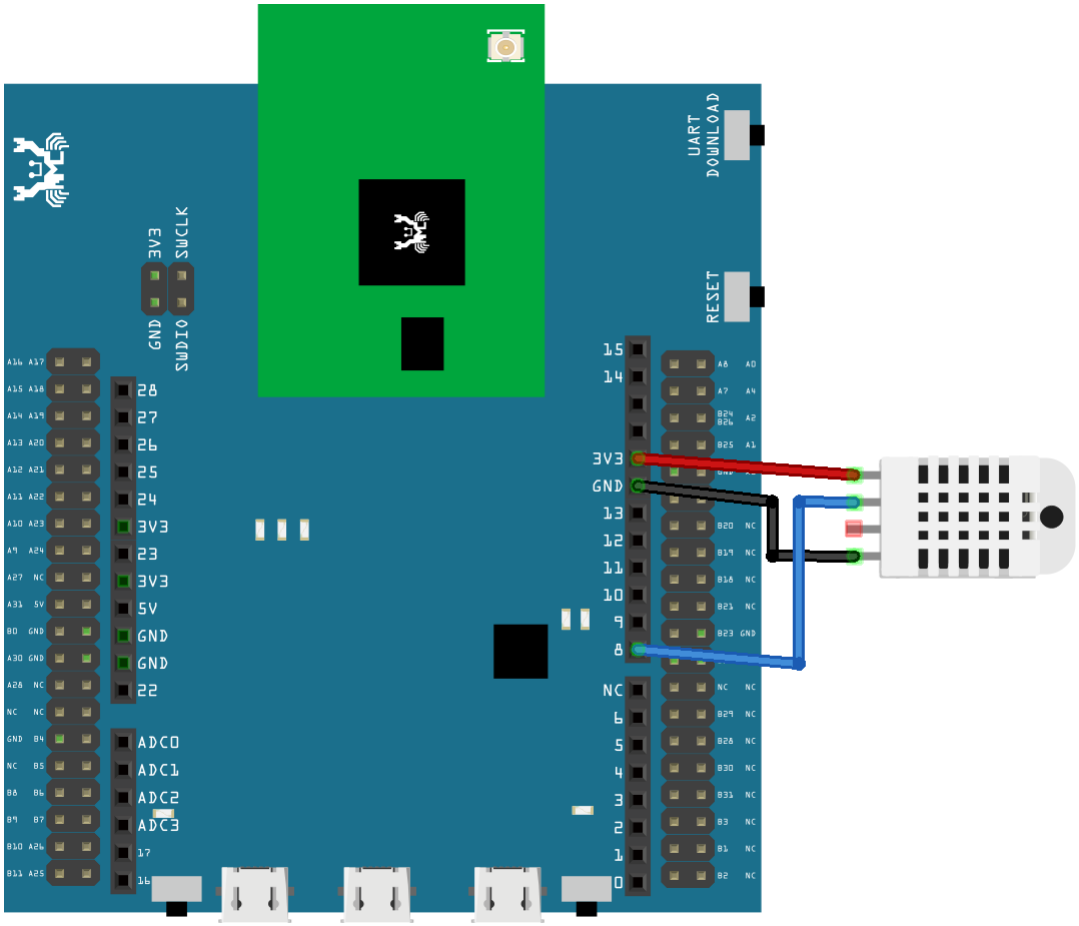
Introduction

In this example, the data obtained from a DHT temperature and humidity sensor are transmitted over a BLE UART service to a smartphone. Refer to the other examples for detailed explanations of using the DHT sensor and the BLE UART service.

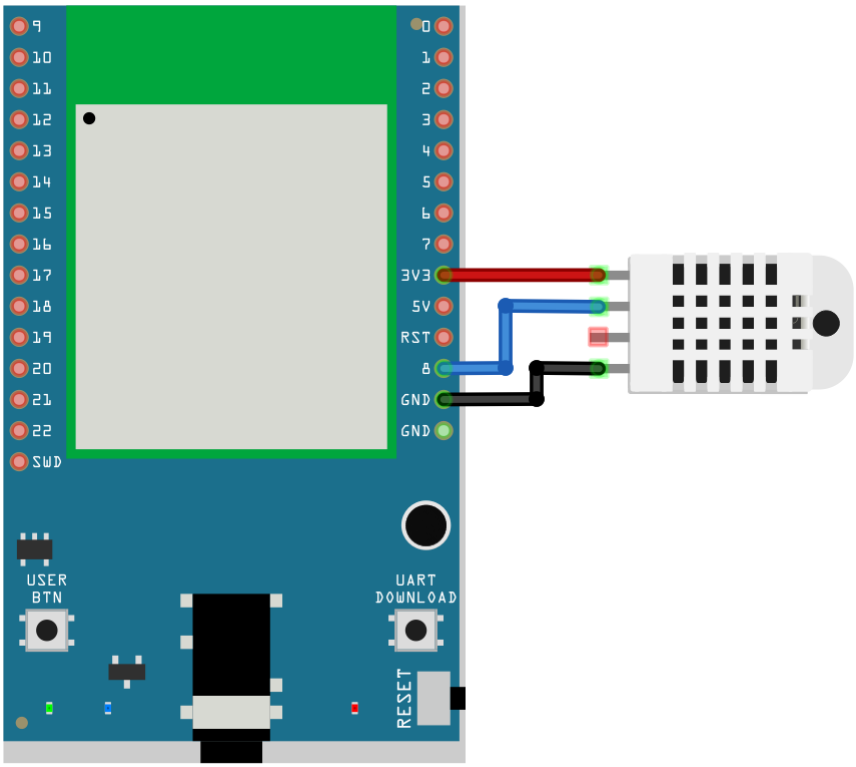
Procedure

Connect the DHT sensor to the Ameba board following the diagram.

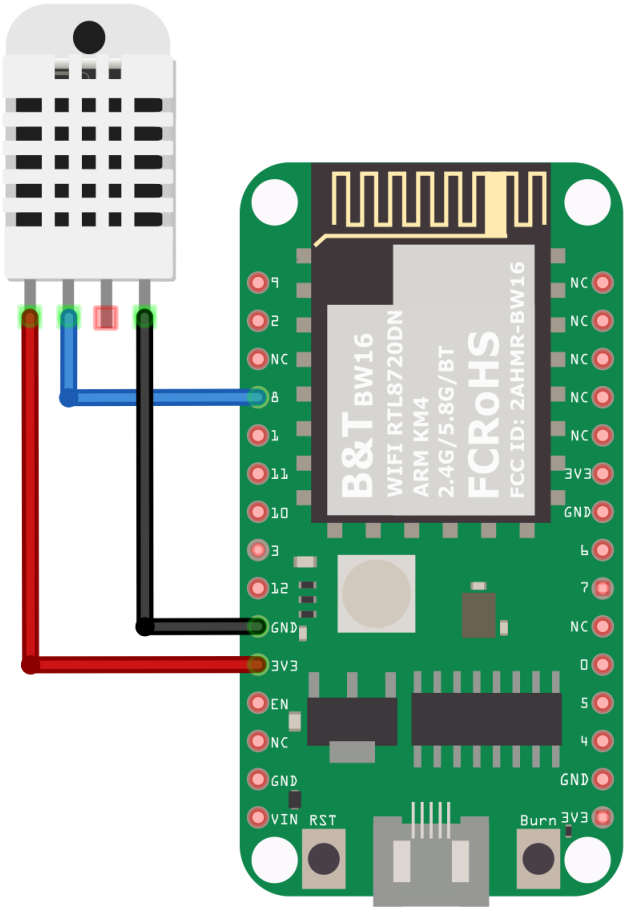
AMB21 / AMB22:



AMB23:


9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

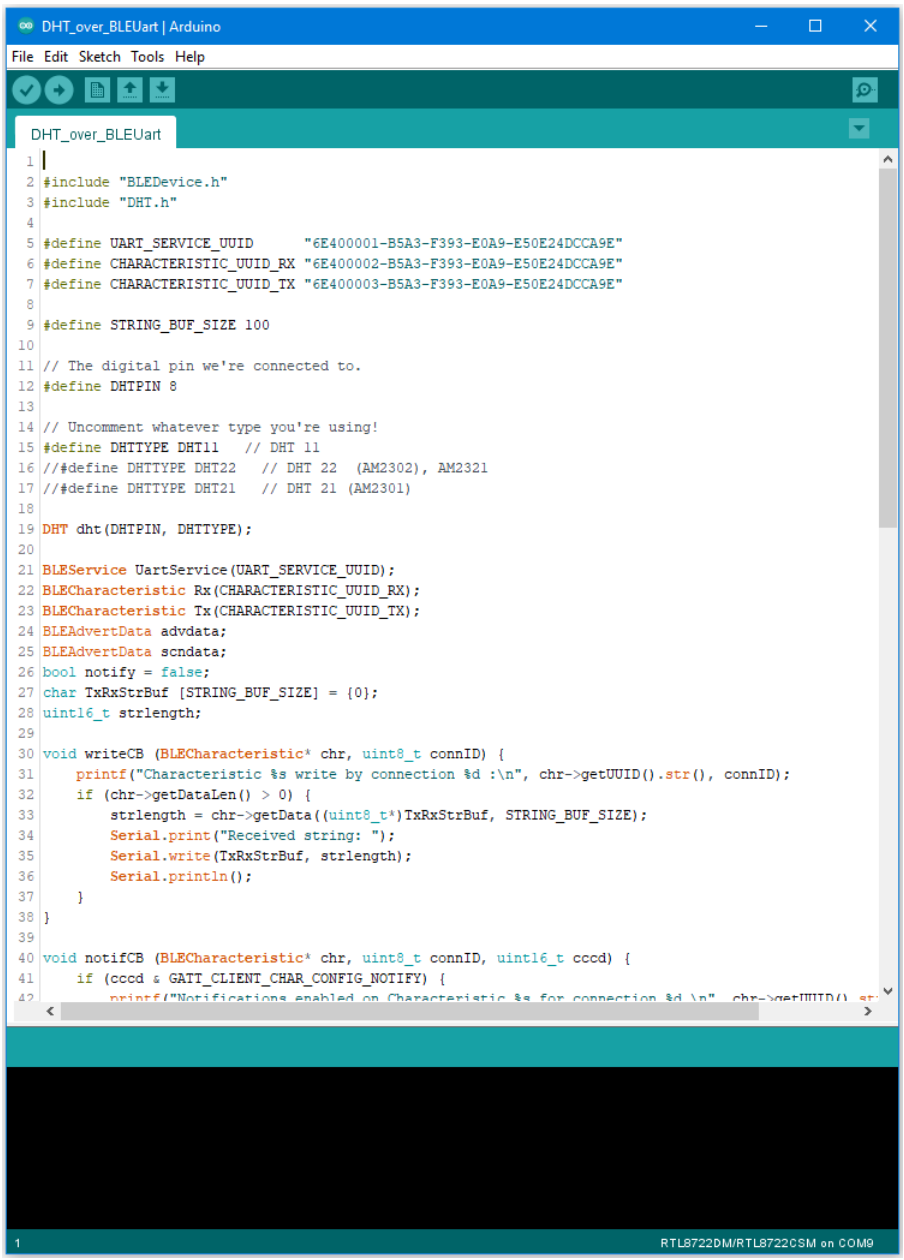
BW16:



Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
- Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connecta>https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

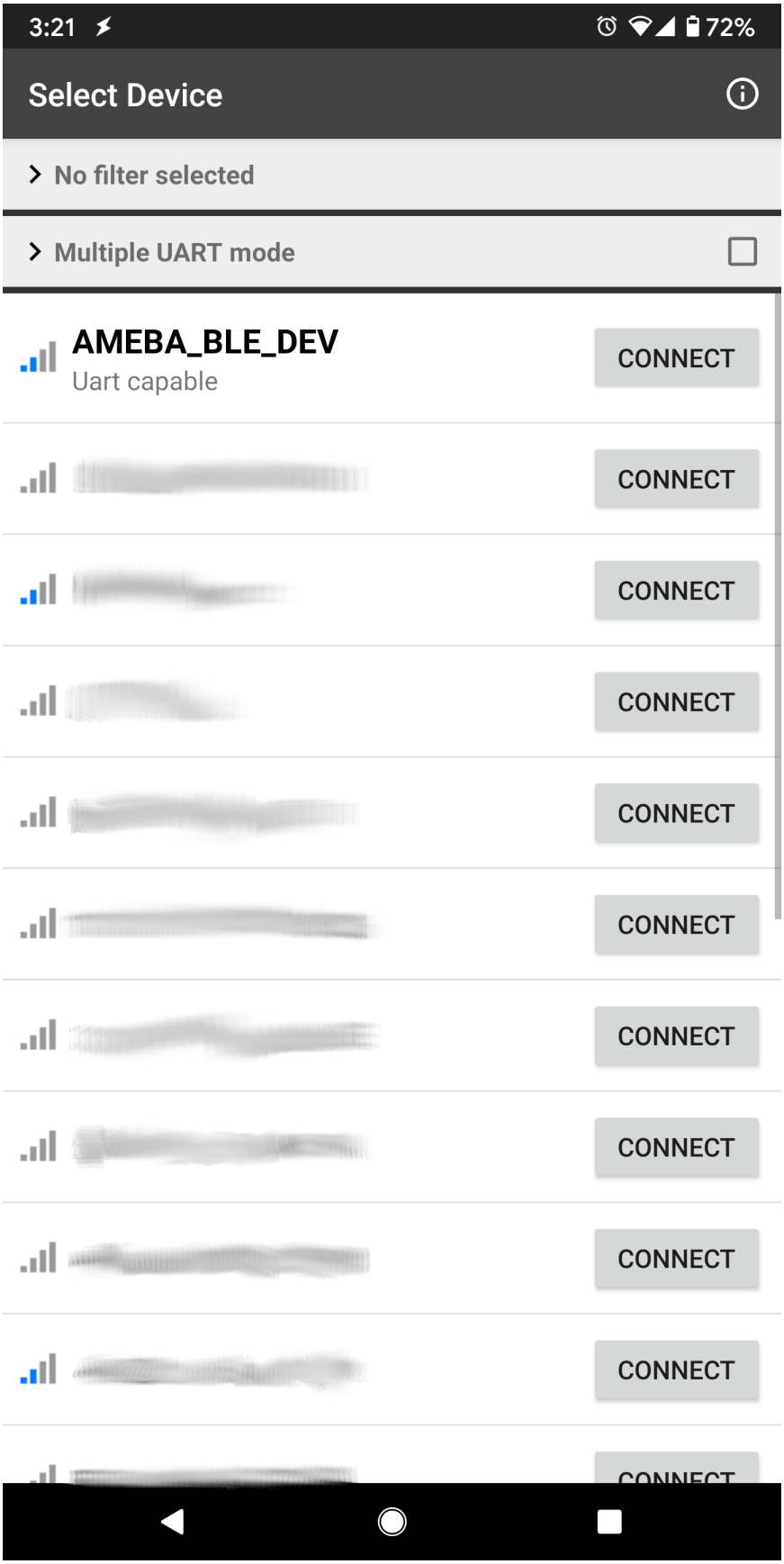
- Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

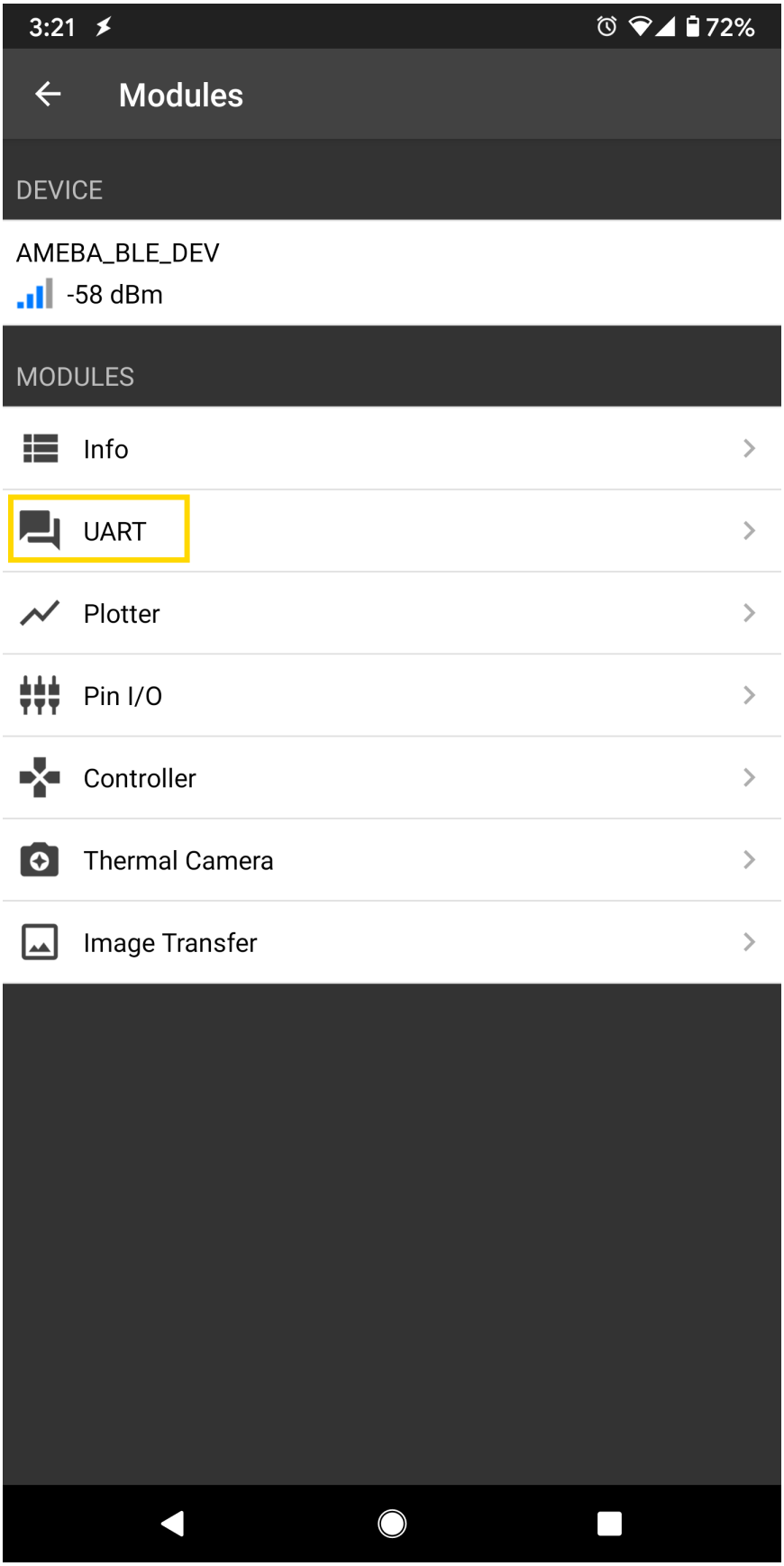
Open the example, "`Files`" -> "`Examples`" -> "`AmebaBLE`" -> "`DHT_over_BLEUart`".



Upload the code and press the reset button on Ameba once the upload is finished.

Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer

After starting the UART function, notifications should be received every 5 seconds containing the measured temperature and humidity.


4:20
81%
UART
MQTT
Humidity: 59% Temperature: 27°C
Humidity: 59% Temperature: 27°C
Humidity: 62% Temperature: 27°C
Humidity: 67% Temperature: 27°C
Humidity: 69% Temperature: 27°C
Humidity: 88% Temperature: 27°C
SEND
Sent: 0 bytes
Received: 198 bytes

BLE – PWM over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- RGB LED
- Android / iOS smartphone

Example

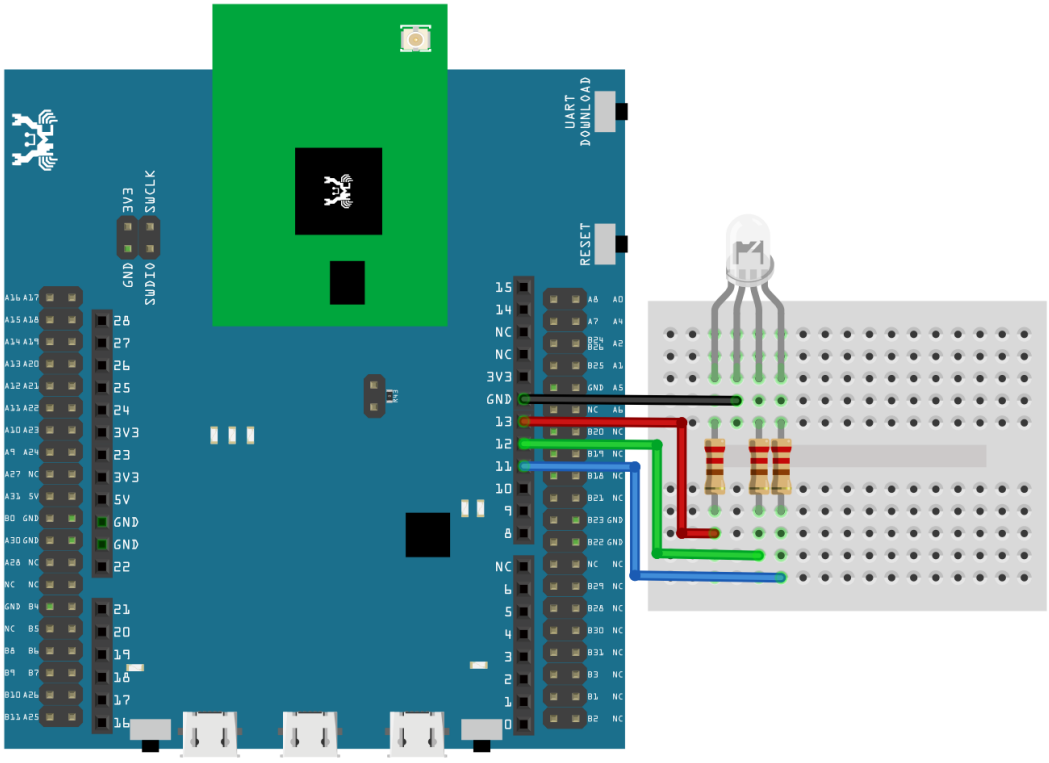
Introduction

In this example, a smartphone app is used to transmit commands over BLE UART to control the PWM outputs and change the color of a RGB LED. Refer to the other example guides for detailed explanations of the BLE UART service.

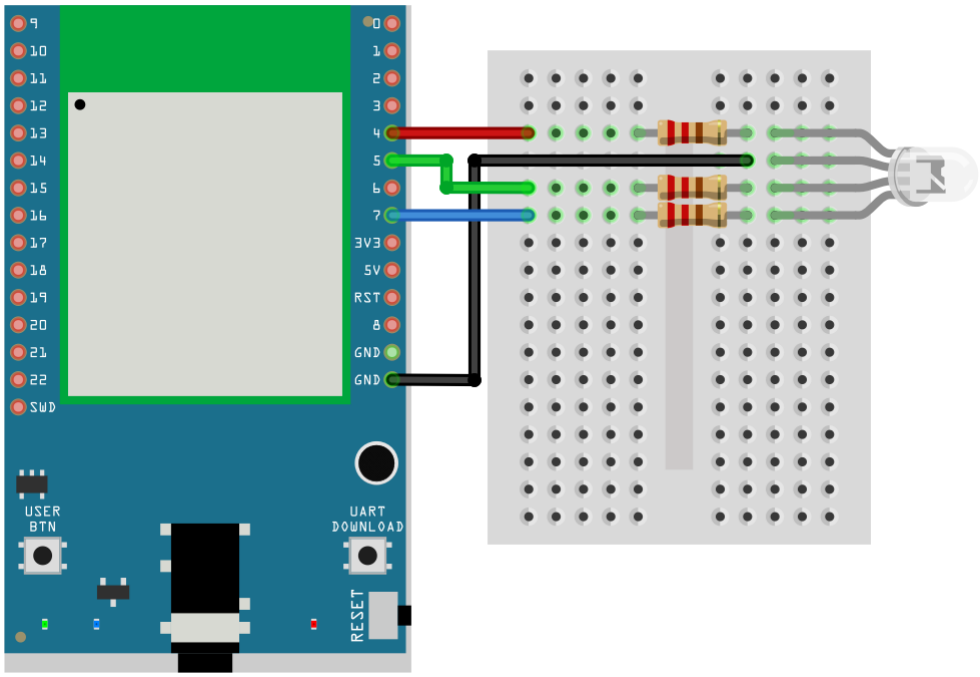
Procedure

Connect the RGB LED to the RTL8722 board following the diagram, the common LED pin may need to connect to 3.3V or GND depending on the type of LED (common anode / common cathode).

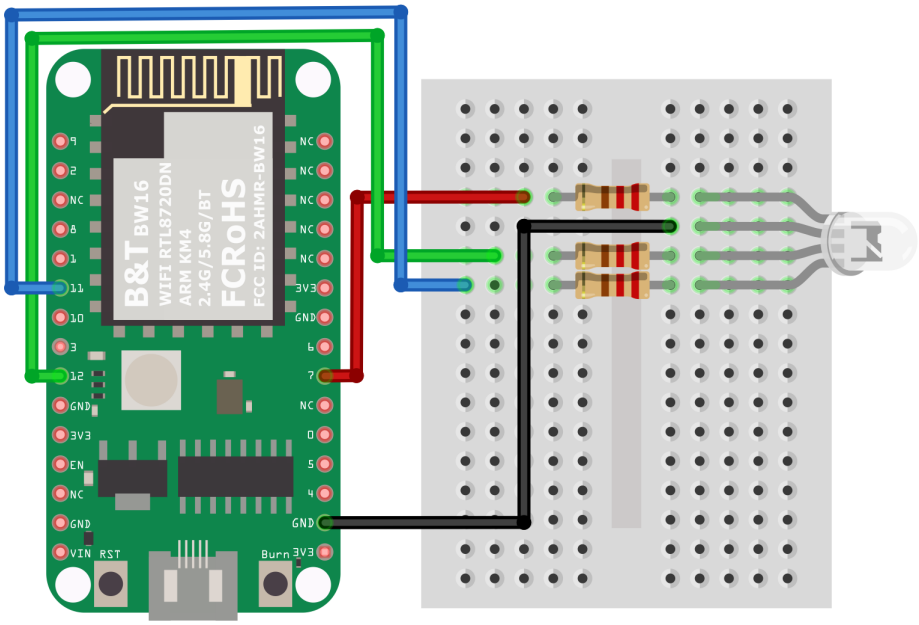
AMB21 /AMB22:



AMB23:



BW16:

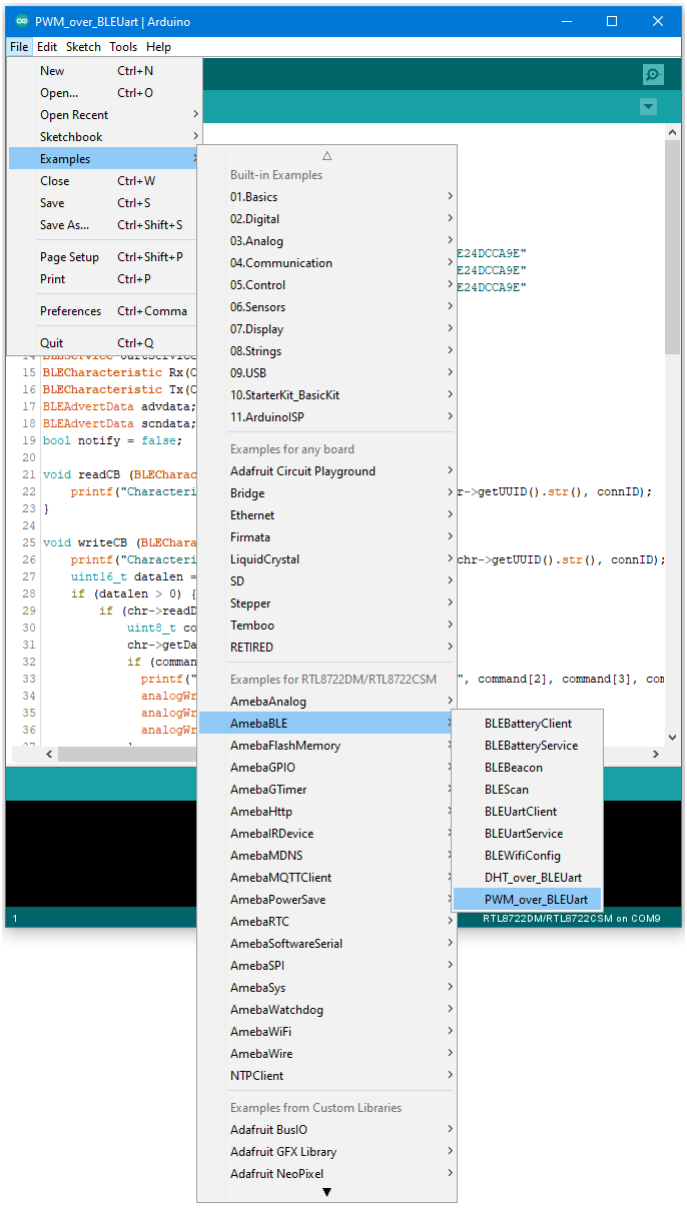


Ensure that the required app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect

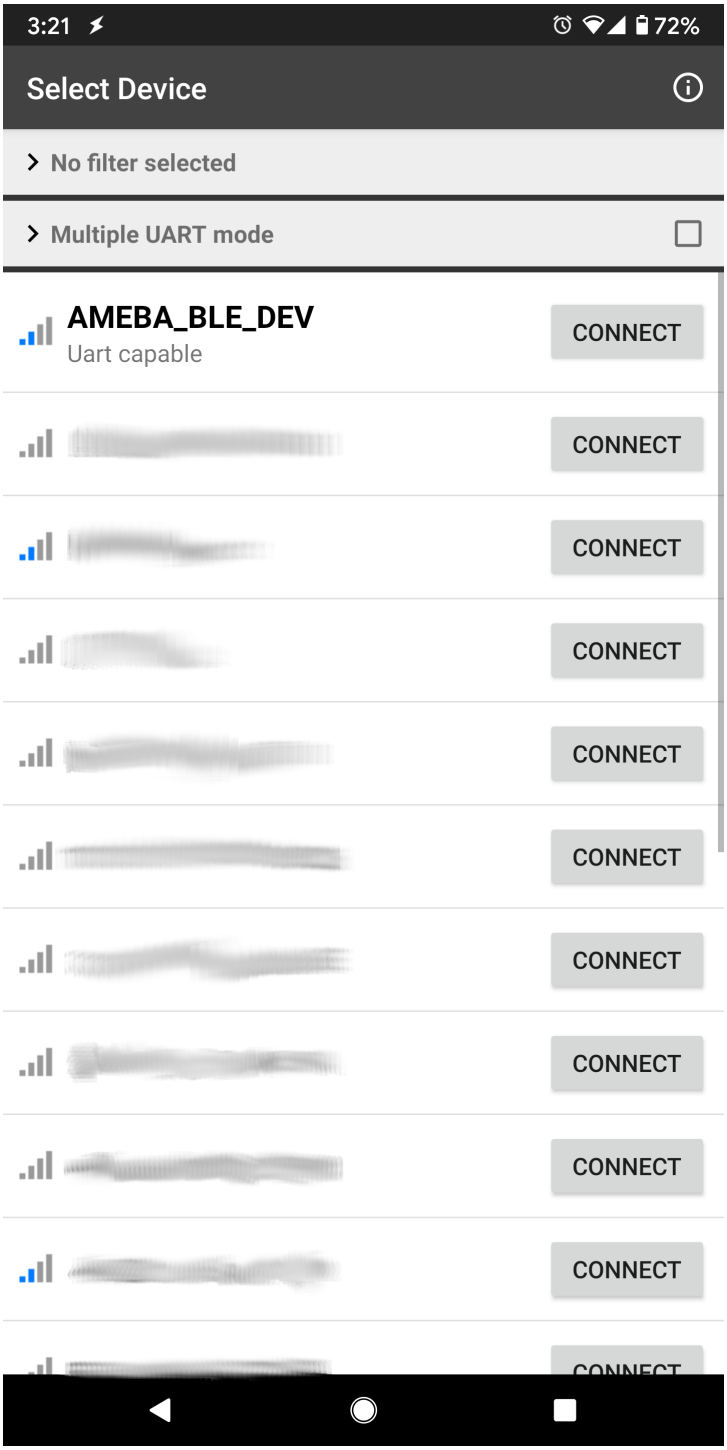
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

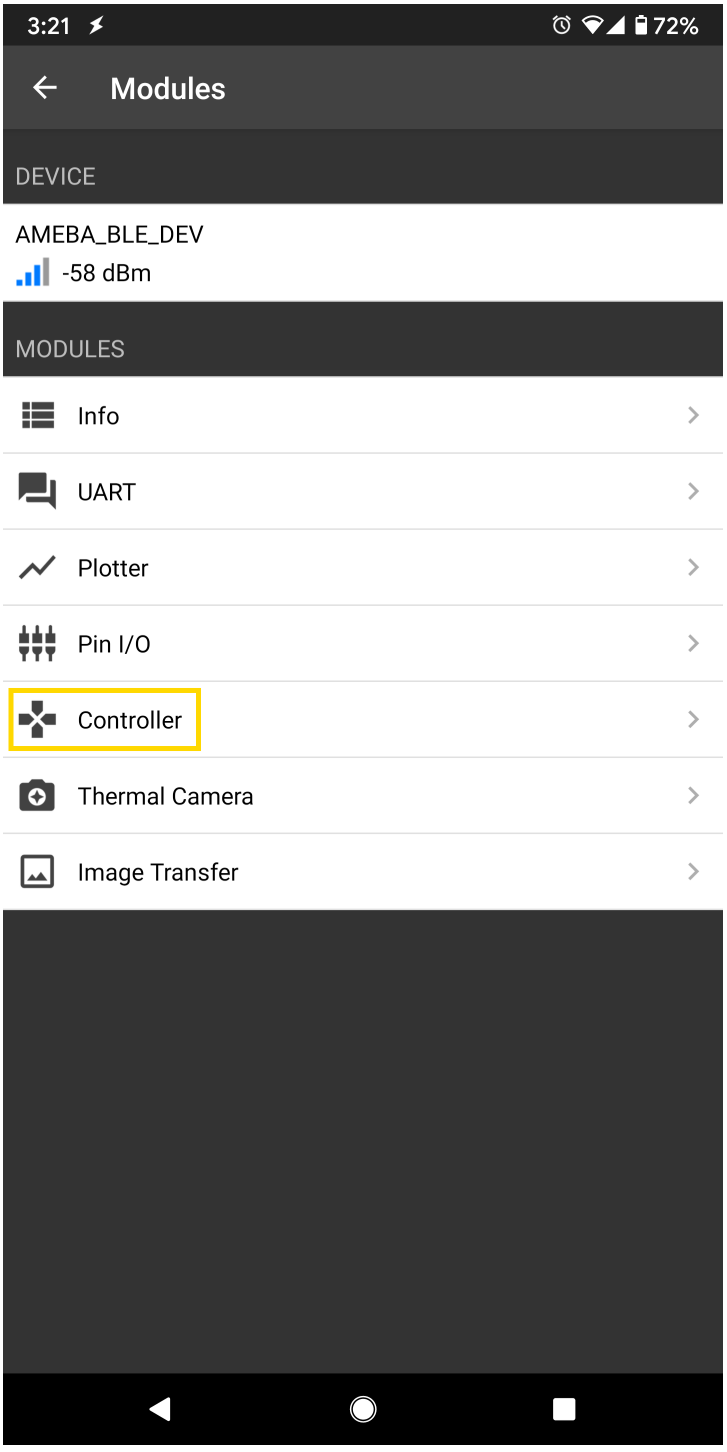
Open the example, “Files” -> “Examples” -> “AmebaBLE” -> “PWM_over_BLEUart”.

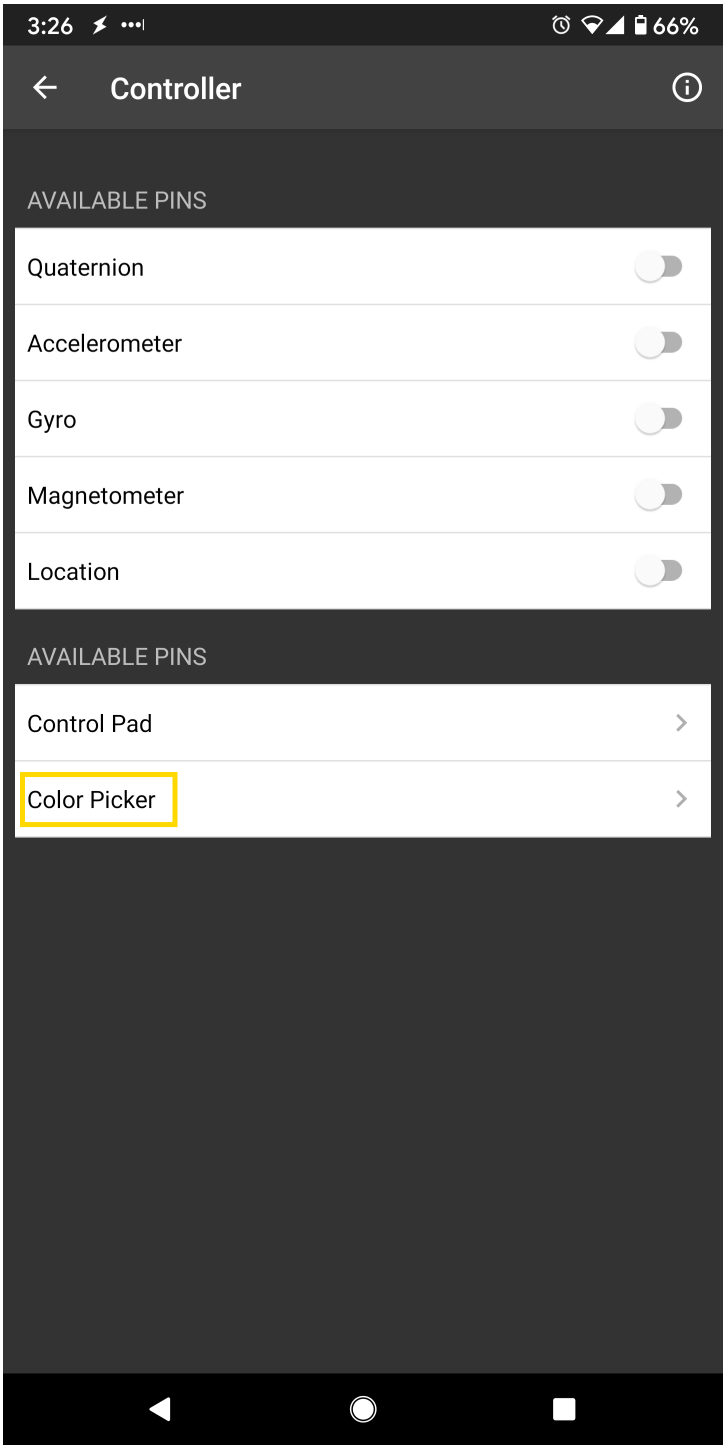
Upload the code and press the reset button on Ameba once the upload is finished.



Open the app on your smartphone, scan and connect to the board shown as “AMEBA_BLE_DEV” and choose the controller -> color picker function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer



Using the color selection wheel, saturation, and brightness sliders, choose a desired color and click select to send the RGB values to the board. You should see the RGB LED change to the matching color.

Code Reference

The RGB values are sent as three consecutive bytes prefixed by "!C" characters. The "!" exclamation mark is used to indicate that the following data is a command, and the "C" character is used to indicate that the data is RGB values. The received UART message is checked in the callback function for "!C" first, otherwise it is treated as a regular message and printed to the serial terminal.

BLE - HID Gamepad

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

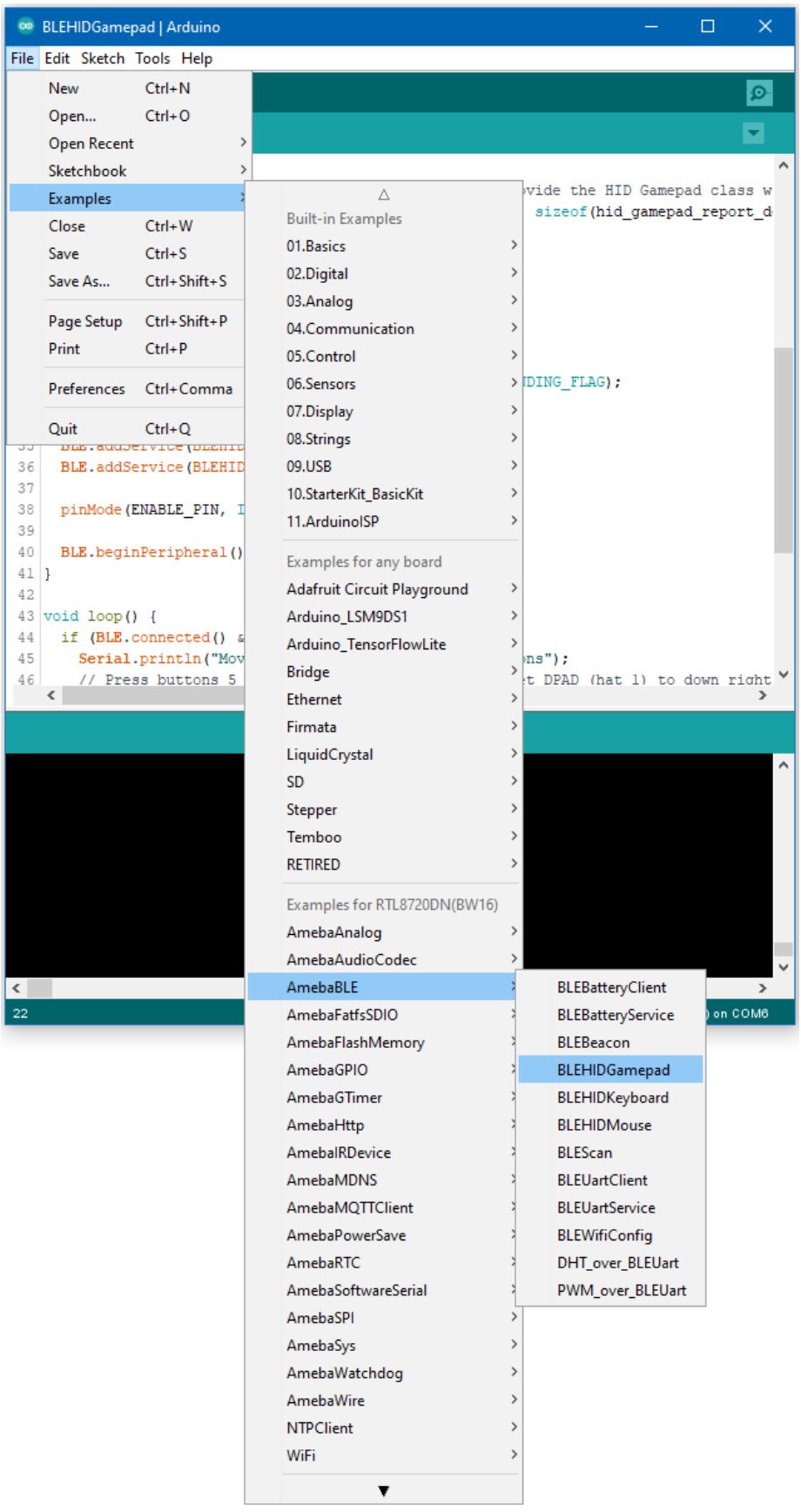
Example

Introduction

In this example, the RTL8722 board emulates a HID gamepad connected using BLE.

Procedure

Open the example, "`Files" -> "Examples" -> "AmebaBLE" -> BLEHIDGamepad`.

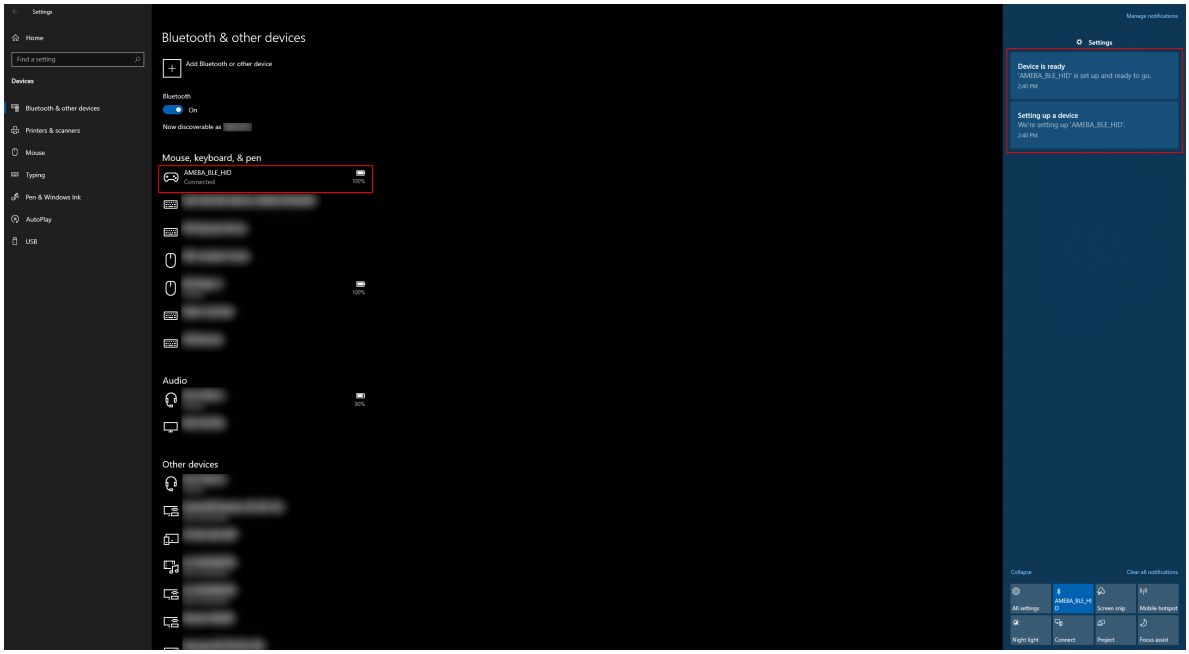

BLEHIDGamepad | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

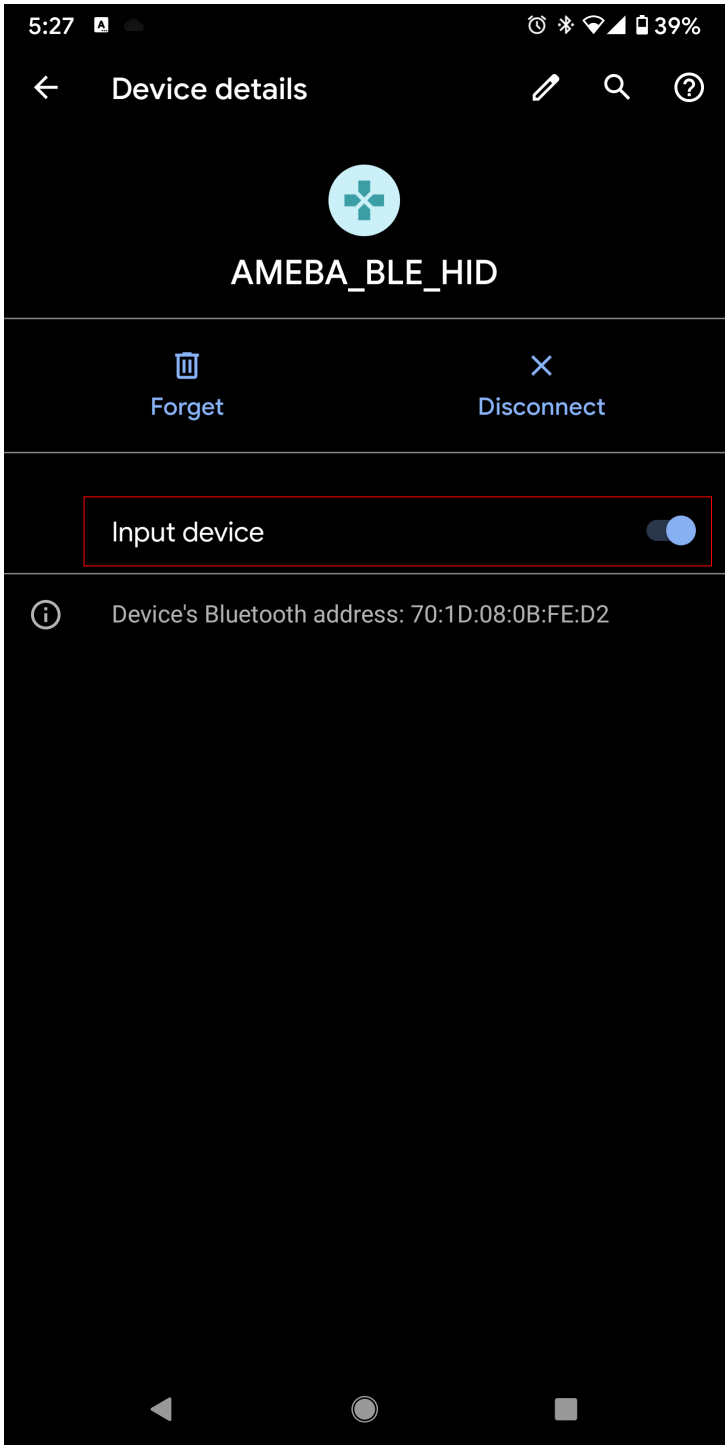
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.



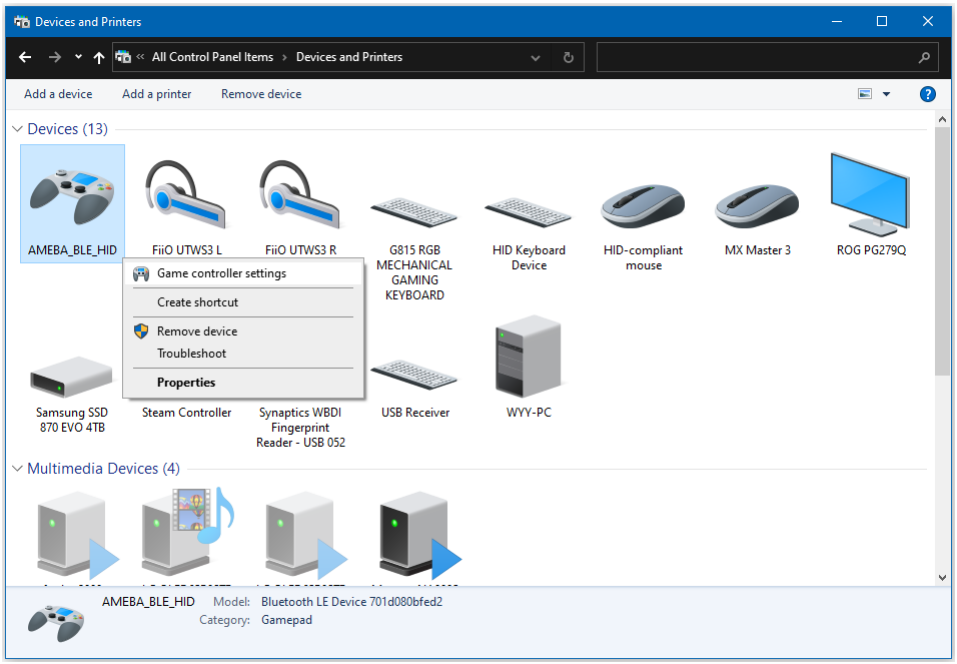
On Android, ensure that “Input device” is enabled for the board.

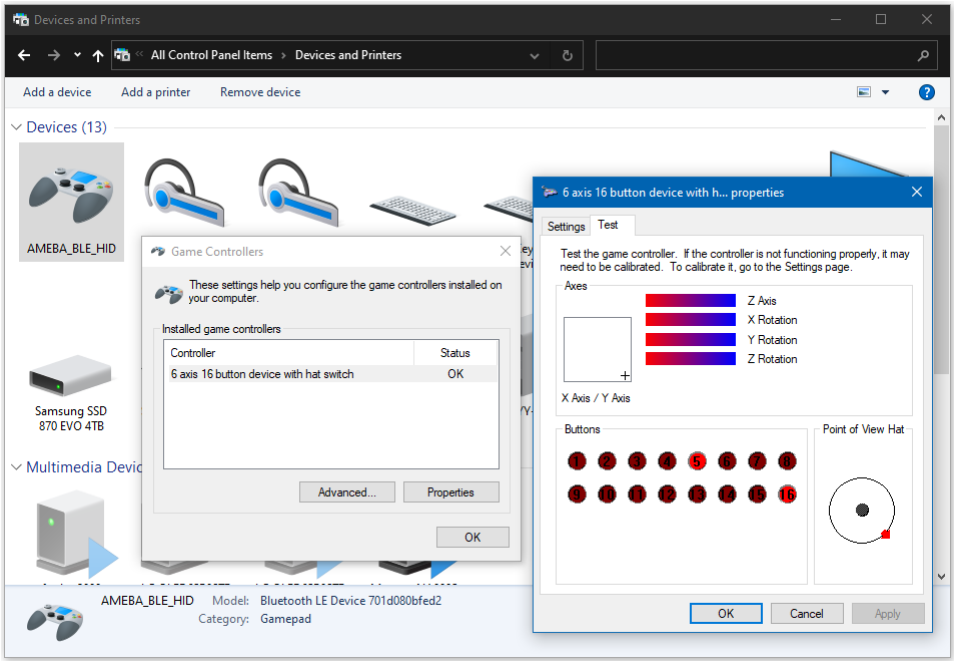


After the Bluetooth connection process is completed, the board is ready to send gamepad input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

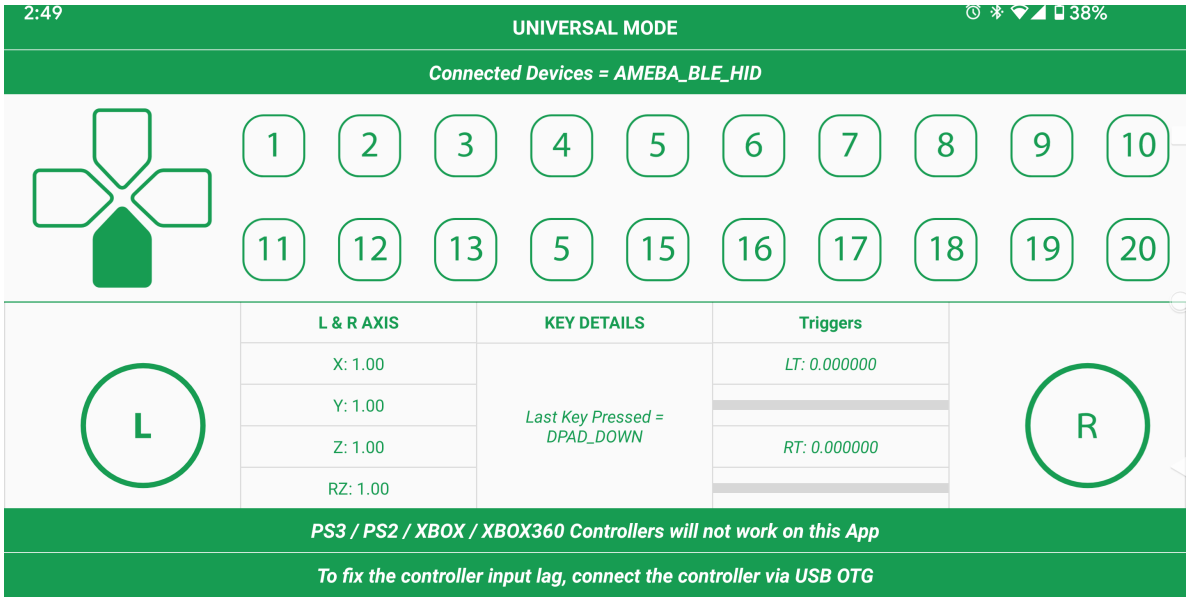
To view the input, open a browser window and go to Gamepad Tester. The connected gamepad device should show up here, and some of the buttons and axes should show changing values.

On Windows, gamepad input can also be viewed by going to “Control Panel” -> “Devices and Printers” -> “AMEBA_BLE_HID” -> “Game Controller Settings” -> “Properties”. The buttons and axes should also show changing values here.





On Android, gamepad testing apps such as Andriod Gamepad Tester can also be used to view the gamepad input.



Code Reference

By default, the board emulates a gamepad with an 8-direction hat switch (d-pad), 6 analog axes and 16 buttons. How the inputs are interpreted is dependent on the host device, and the button ordering may differ between devices. Also, some axes or buttons may be disabled or missing on certain host devices.

BLE - HID Keyboard

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

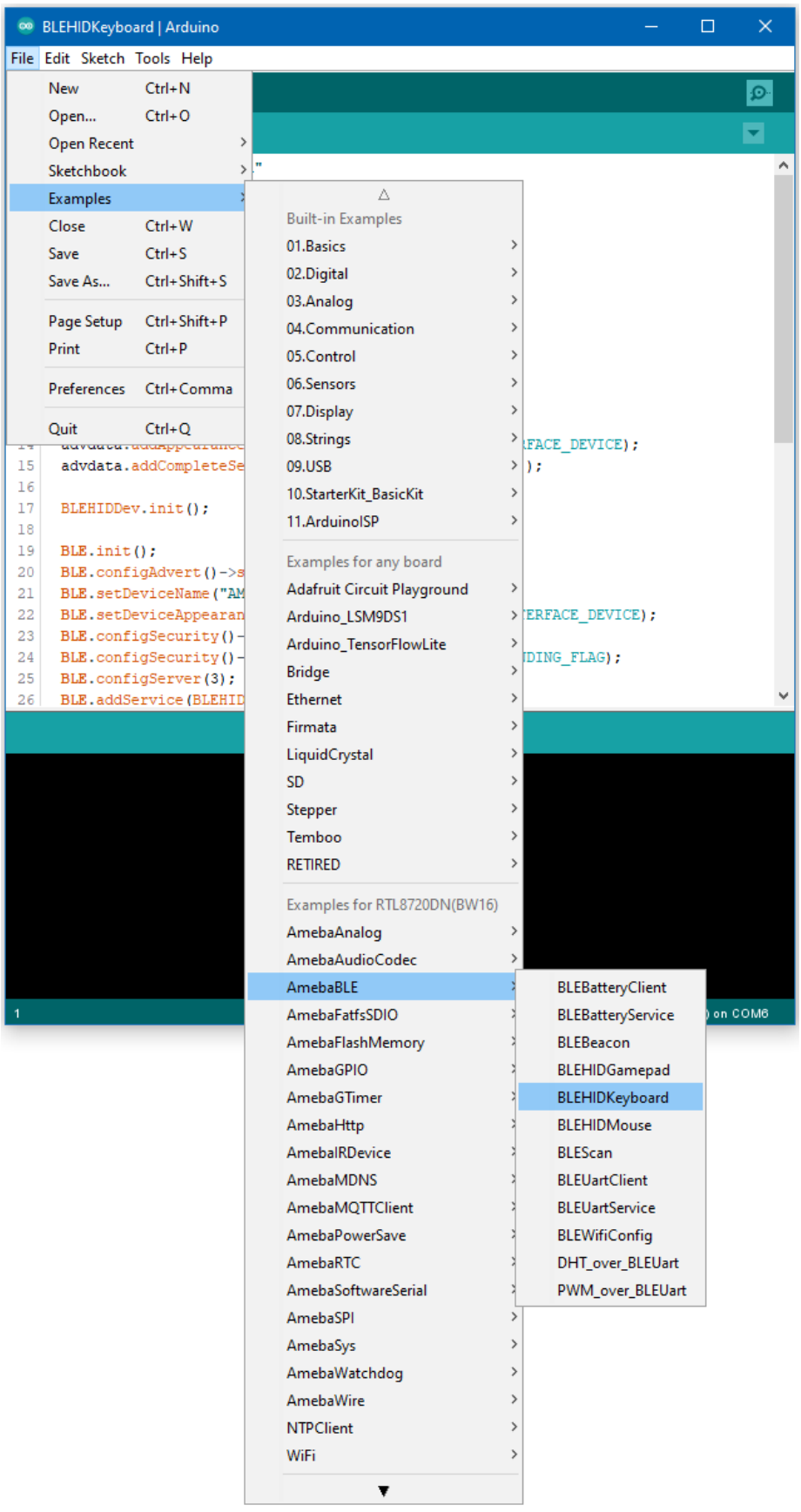
Example

Introduction

In this example, the RTL8722 board emulates a HID keyboard connected using BLE.

Procedure

Open the example, "`Files" -> "Examples" -> "AmebaBLE" -> BLEHIDKeyboard`.

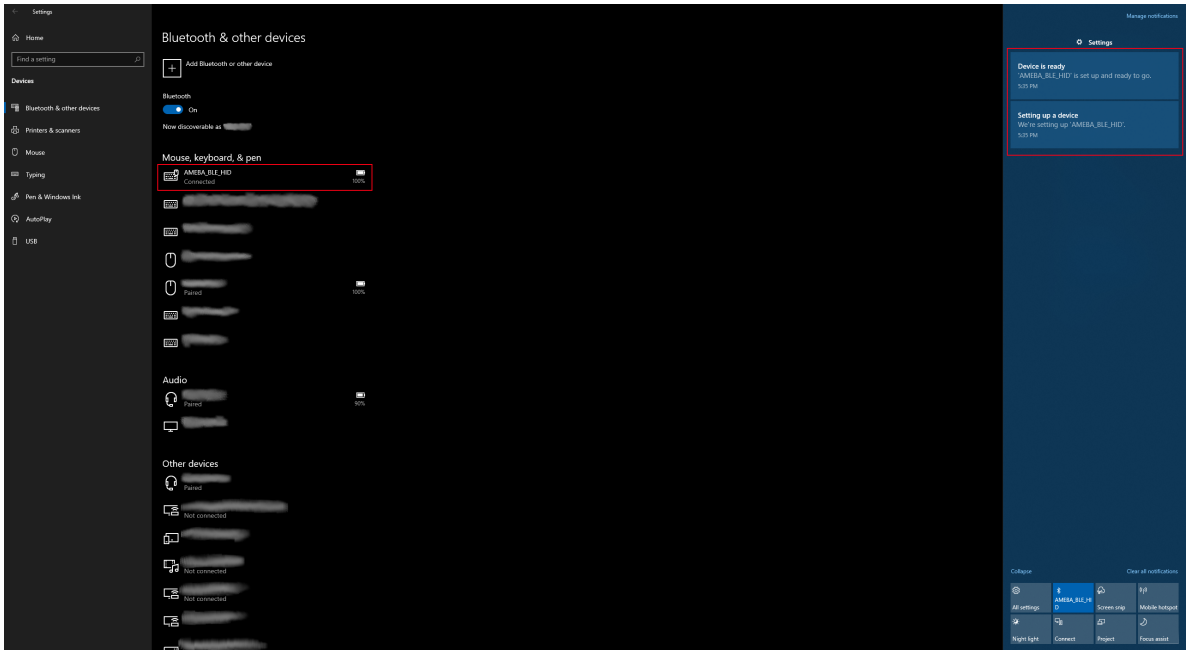

BLEHIDKeyboard | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

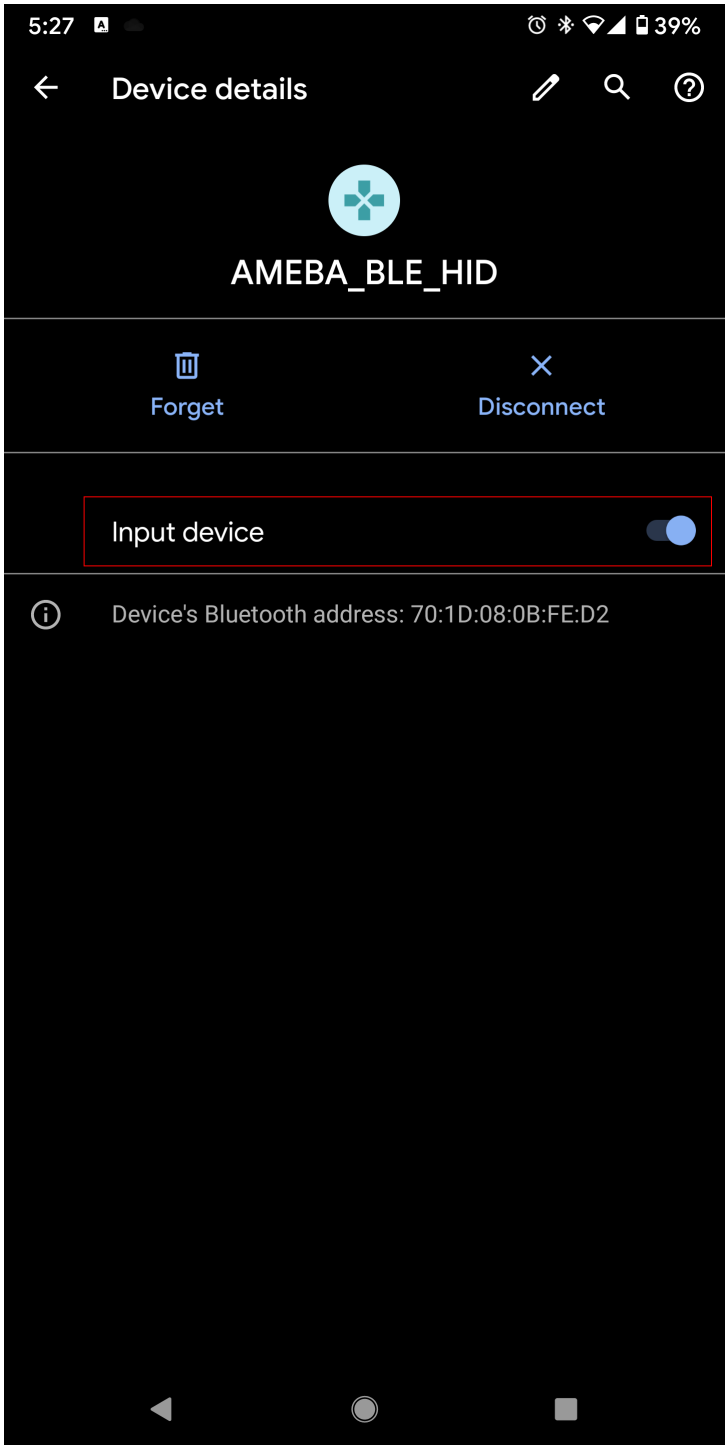
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.

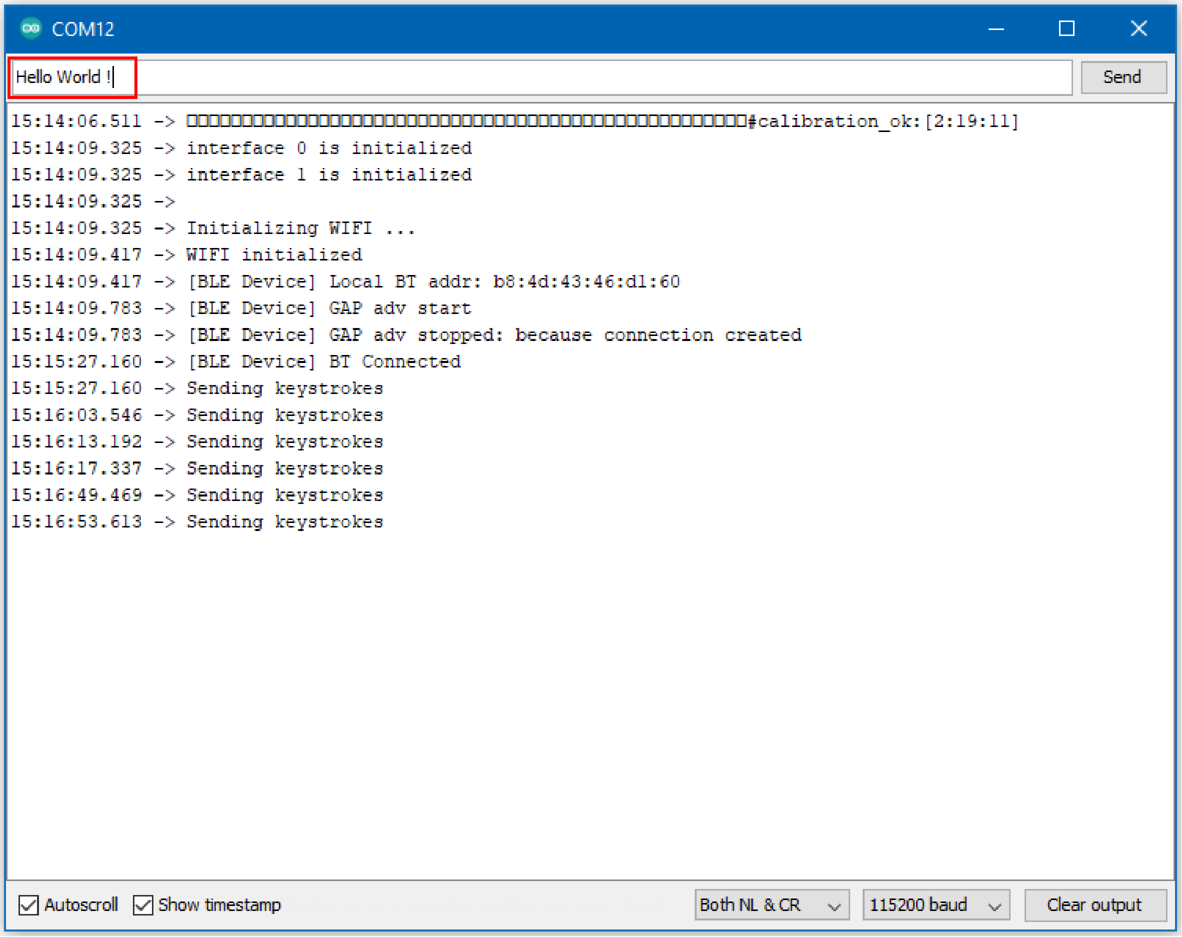


On Android, ensure that “Input device” is enabled for the board.



After the Bluetooth connection process is completed, the board is ready to send mouse input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

You should see the text “Hello World !” typed out and deleted repeatedly.



BLE - HID Mouse

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- BLE capable host device [Windows / Linux / MacOS / Android

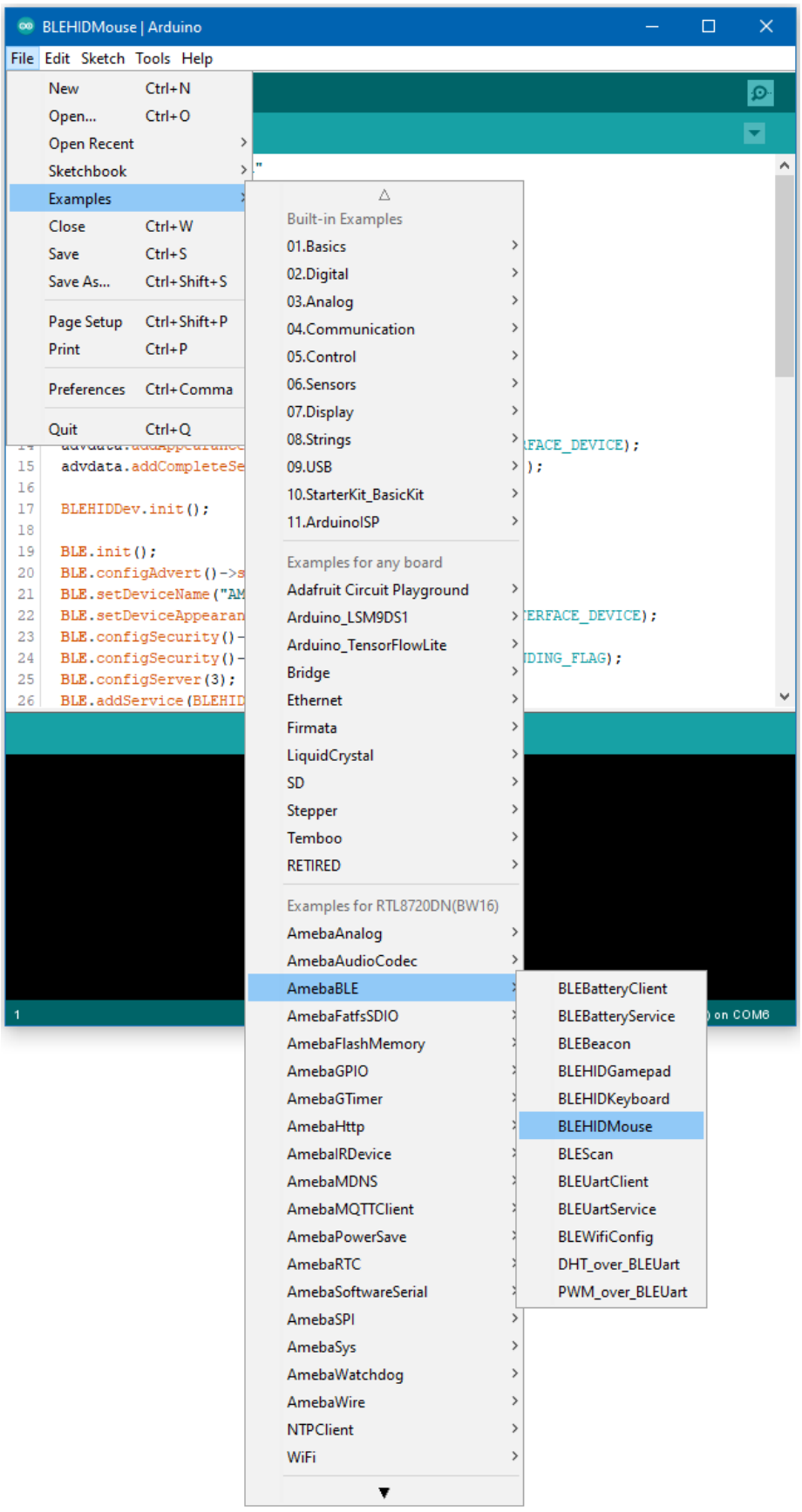
Example

Introduction

In this example, the RTL8722 board emulates a HID mouse connected using BLE.

Procedure

Open the example, "`Files`" -> "`Examples`" -> "`AmebaBLE`" -> "`BLEHIDMouse`".

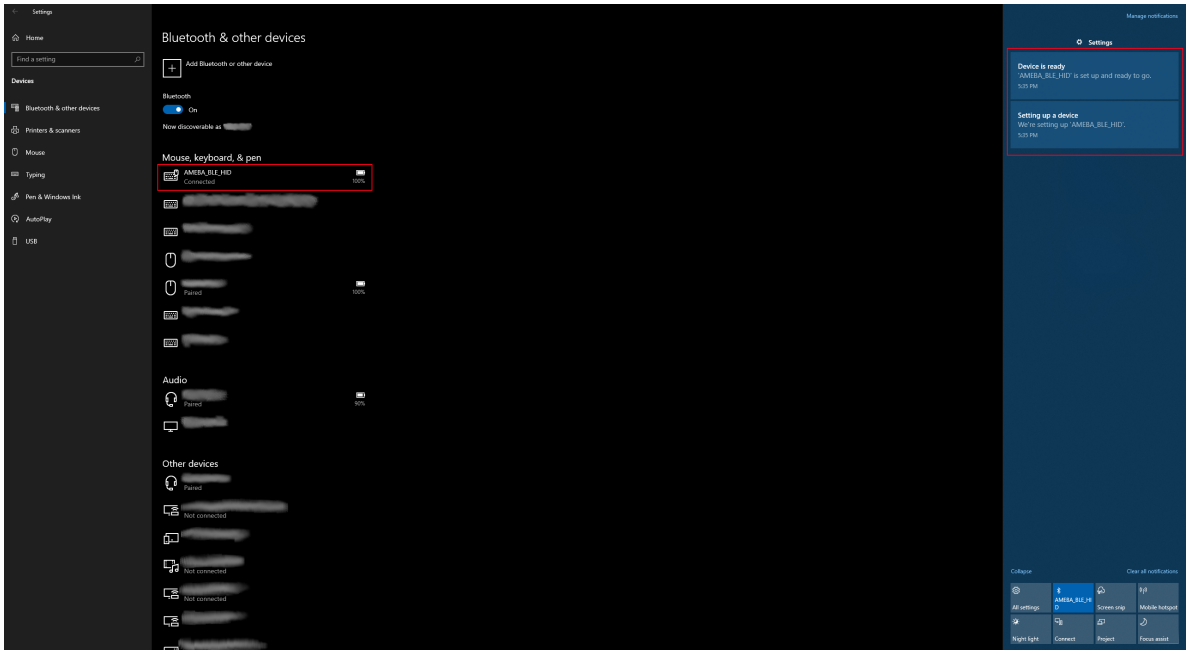

BLEHIDMouse | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arduino_LSM9DS1
Arduino_TensorFlowLite
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8720DN(BW16)
AmebaAnalog
AmebaAudioCodec
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWire
NTPClient
WiFi
BLEBatteryClient
BLEBatteryService
BLEBeacon
BLEHIDGamepad
BLEHIDKeyboard
BLEHIDMouse
BLEScan
BLEUartClient
BLEUartService
BLEWifiConfig
DHT_over_BLEUart
PWM_over_BLEUart

Upload the code and press the reset button once the upload is finished.

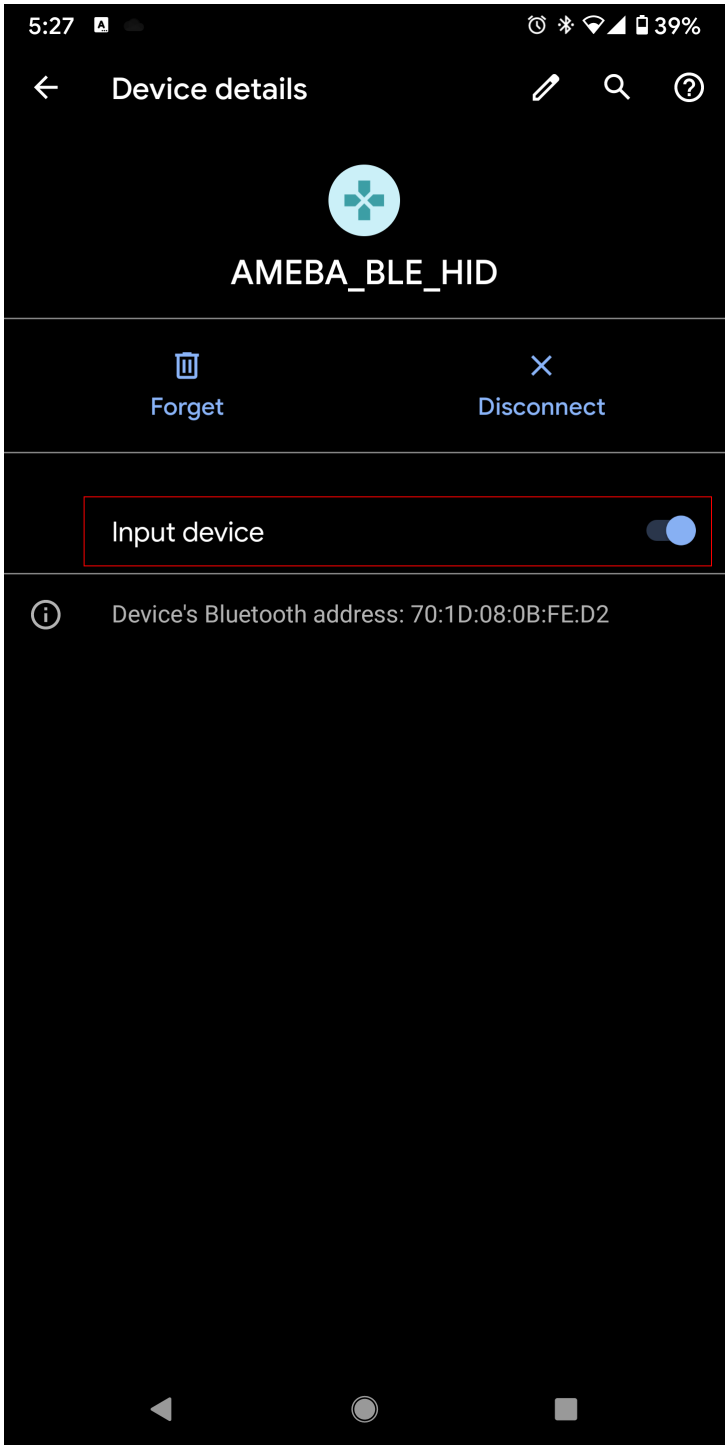
Immediately after reset, the board will begin BLE advertising as “AMEBA_BLE_HID”. On your host device, go to the Bluetooth settings menu, scan, and connect to the board.

You should ensure that the connection process is completed before proceeding.

On Windows, ensure that any driver installation is finished, and the board shows up in the Bluetooth menu under the “Mouse, keyboard & pen” category.



On Android, ensure that “Input device” is enabled for the board.



After the Bluetooth connection process is completed, the board is ready to send mouse input to the host device. Connect digital pin 8 to 3.3V to start sending input, and connect to GND to stop.

You should see the mouse cursor move around four points in a square, performing right and left clicks, and scrolling up and down.

Code Reference

How the mouse input is interpreted is dependent on the host system. Some systems, such as mobile operating systems, may not support all mouse button input functions.

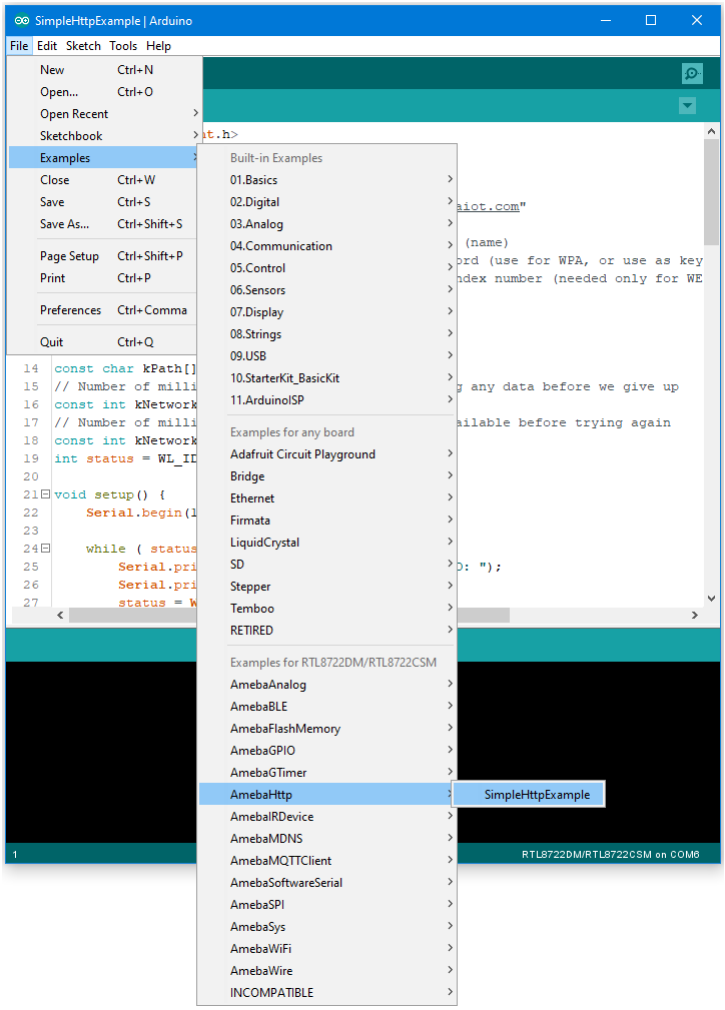
HTTP - Retrieve HTTP websites from the Internet

Materials

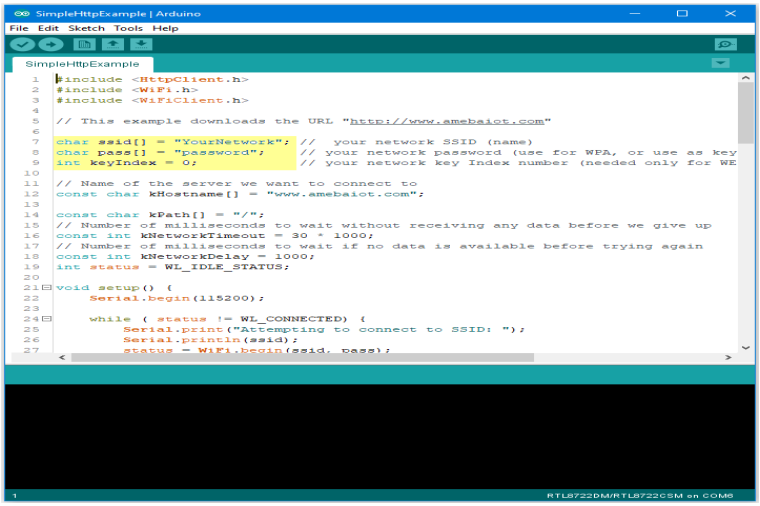
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

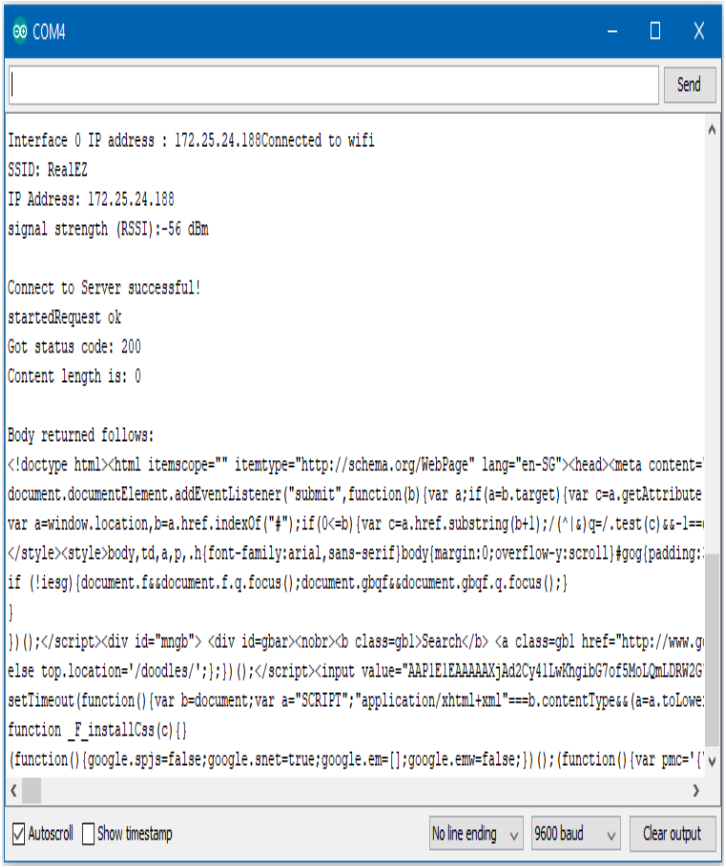
In this example, the HttpClient library is used to retrieve a webpage using the HTTP protocol.
First, make sure that the correct Ameba development board is selected in "Tools" -> "Board"
Then open "`File" -> "Examples" -> "AmebaHttp" -> "SimpleHttpExample`"



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in the Arduino IDE and you can see the information retrieved from the website.



Code Reference

Use `WiFi.begin()` to establish WiFi connection:
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use WiFiClient to create a client to handle the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiClient
Use HTTPClient to create a client to handle the HTTP connection.

Use `http.get()` to send a GET request to the website.

HTTP - Set up Server to Control LED

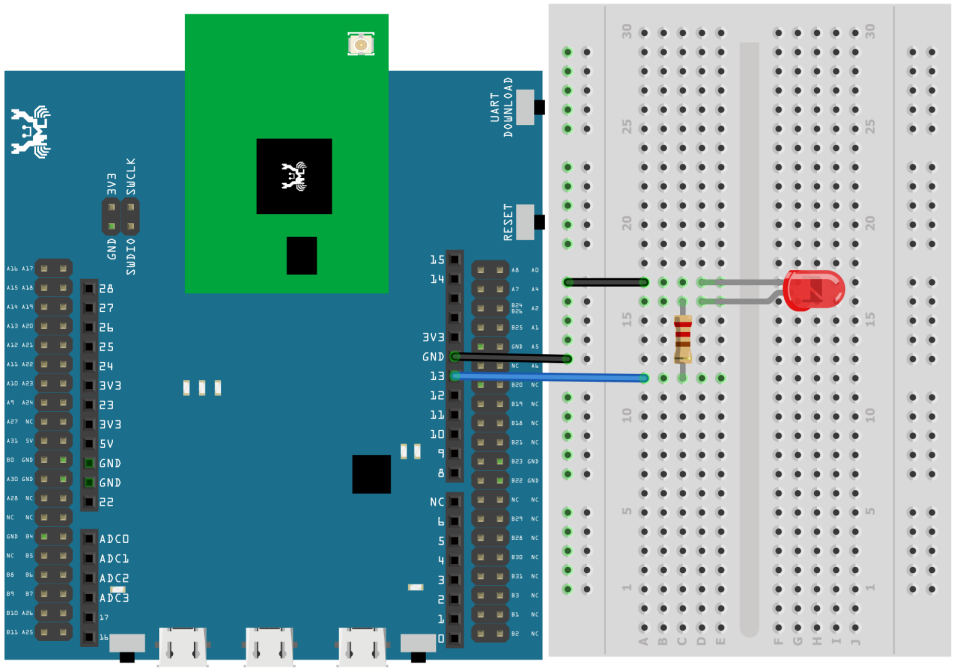
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Breadboard x 1
- LED x 1
- 1KΩ Resistor x 1

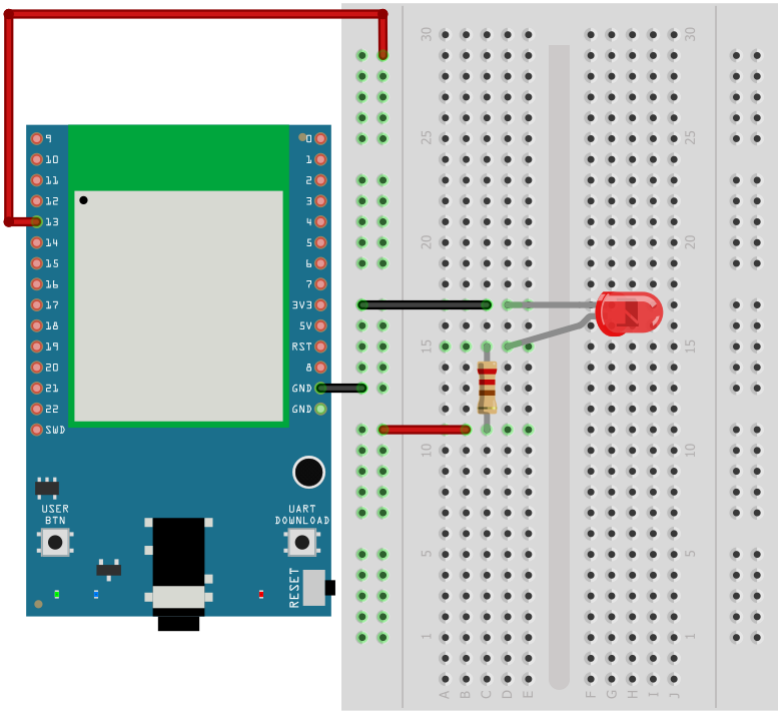
Procedure

In this example, we connect Ameba to WiFi and use Ameba as server, the user can control the LED on/off through a webpage.
First, connect Ameba with the LED.
In a LED, the longer pin is the positive pole, and the shorter pin is the negative pole. So, we connect the shorter pin to GND and connect the longer pin to D13. Additionally, to avoid the electric current exceeds the tolerance of the LED and causes damage, we connect a resistance on the positive pole.

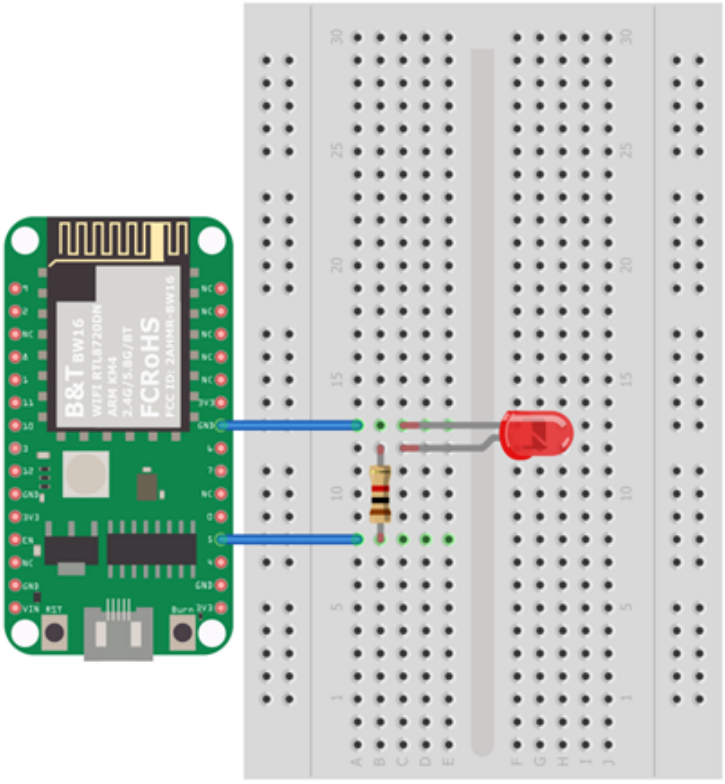
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

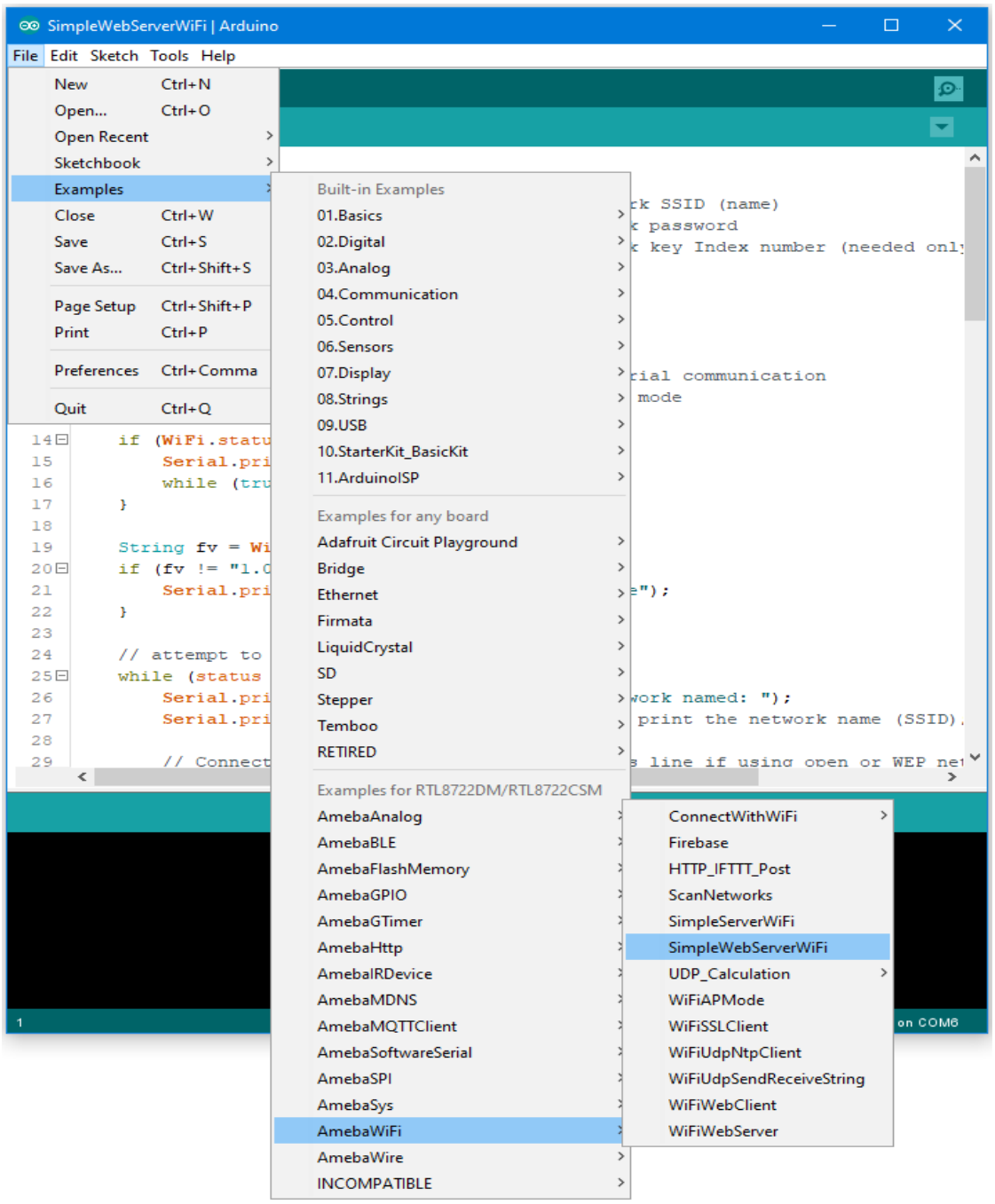


BW16 Wiring Diagram:

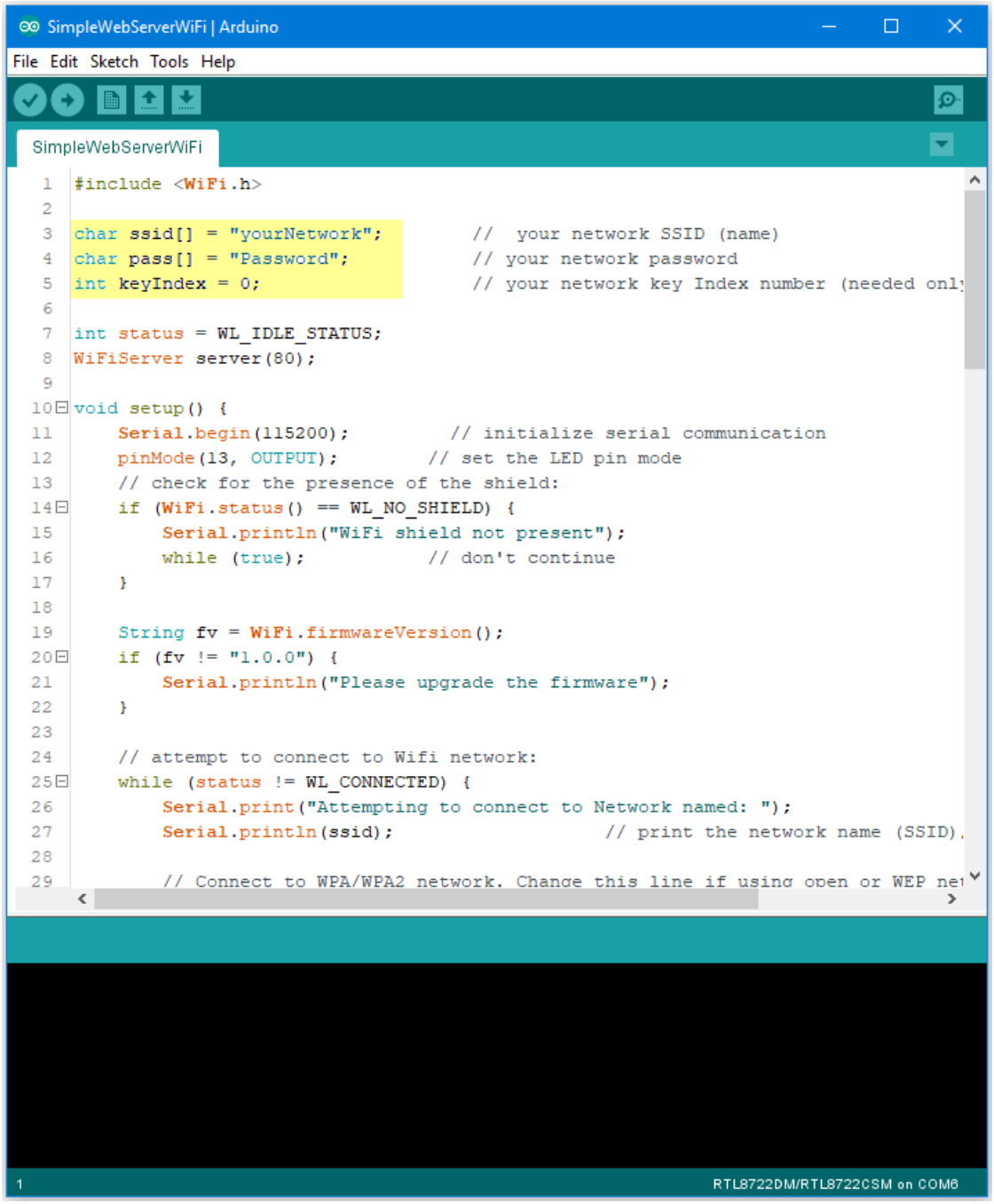


Note:
For BW16 board, you may consider to re-define "LED_PIN" macro to **10** for built-in green LED, or **11** for blue built-in LED, or **12** for red built-in LED to avoid using extra components.

Then open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`SimpleWebServerWiFi`"



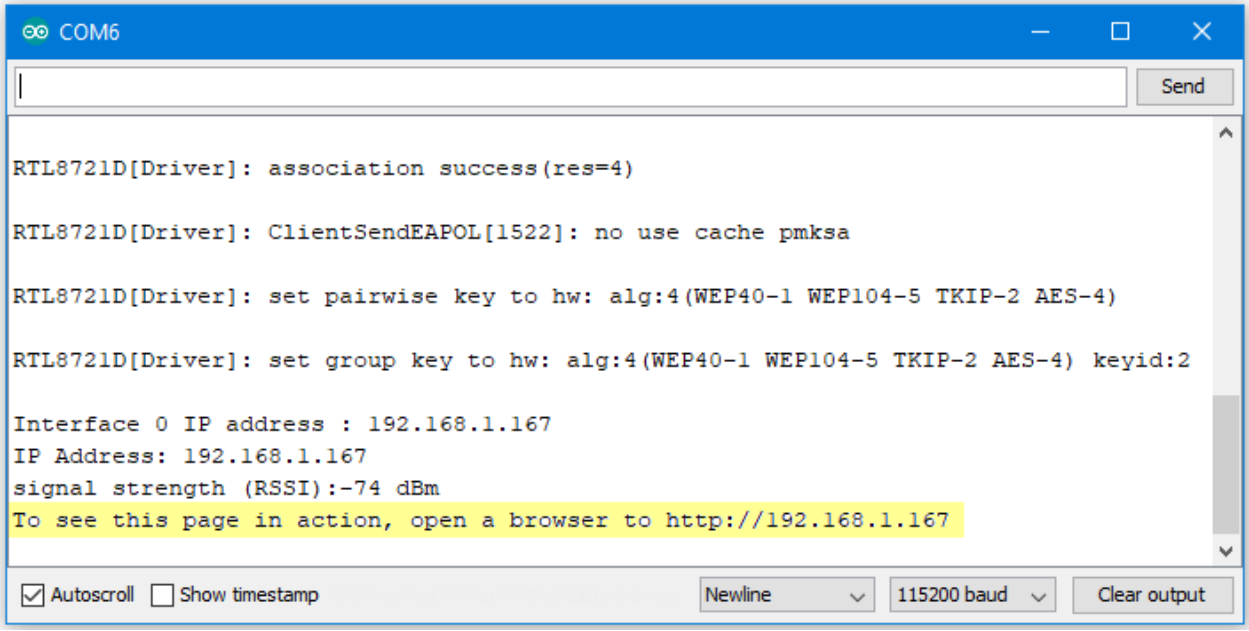
In the sample code, modify the highlighted snippet to corresponding information.



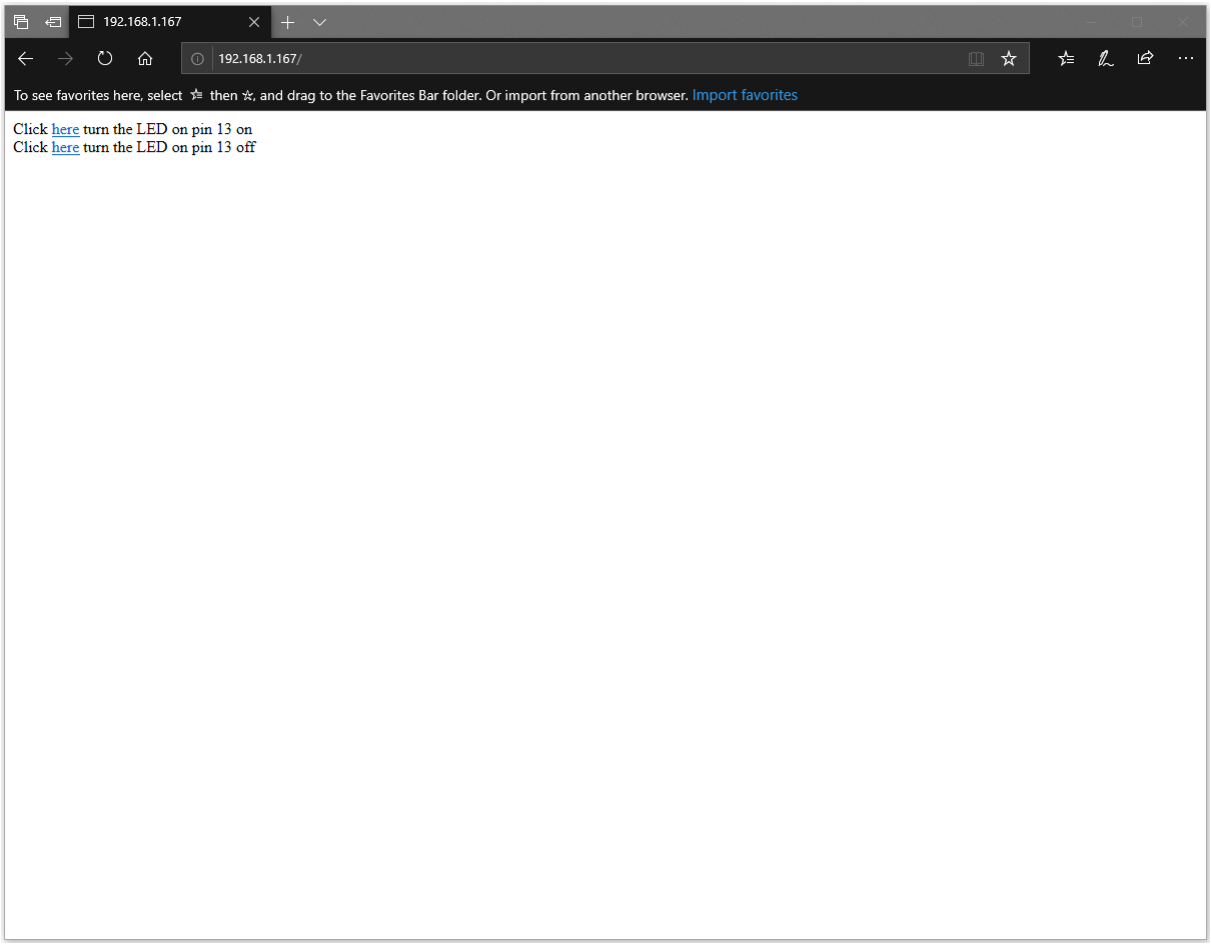
Upload the code and press the reset button on Ameba. When the connection is established, you will see the message:

```
"To see this page in action, open a browser to http://xxx.xxx.xxx.xxx"
```

in the Arduino IDE as shown in the figure:



Next, open the browser of a computer or a cell phone under the same WiFi domain, enter the address in the message.



In the webpage, you can turn on/off the LED.

Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use WiFi.RSSI() to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to get whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

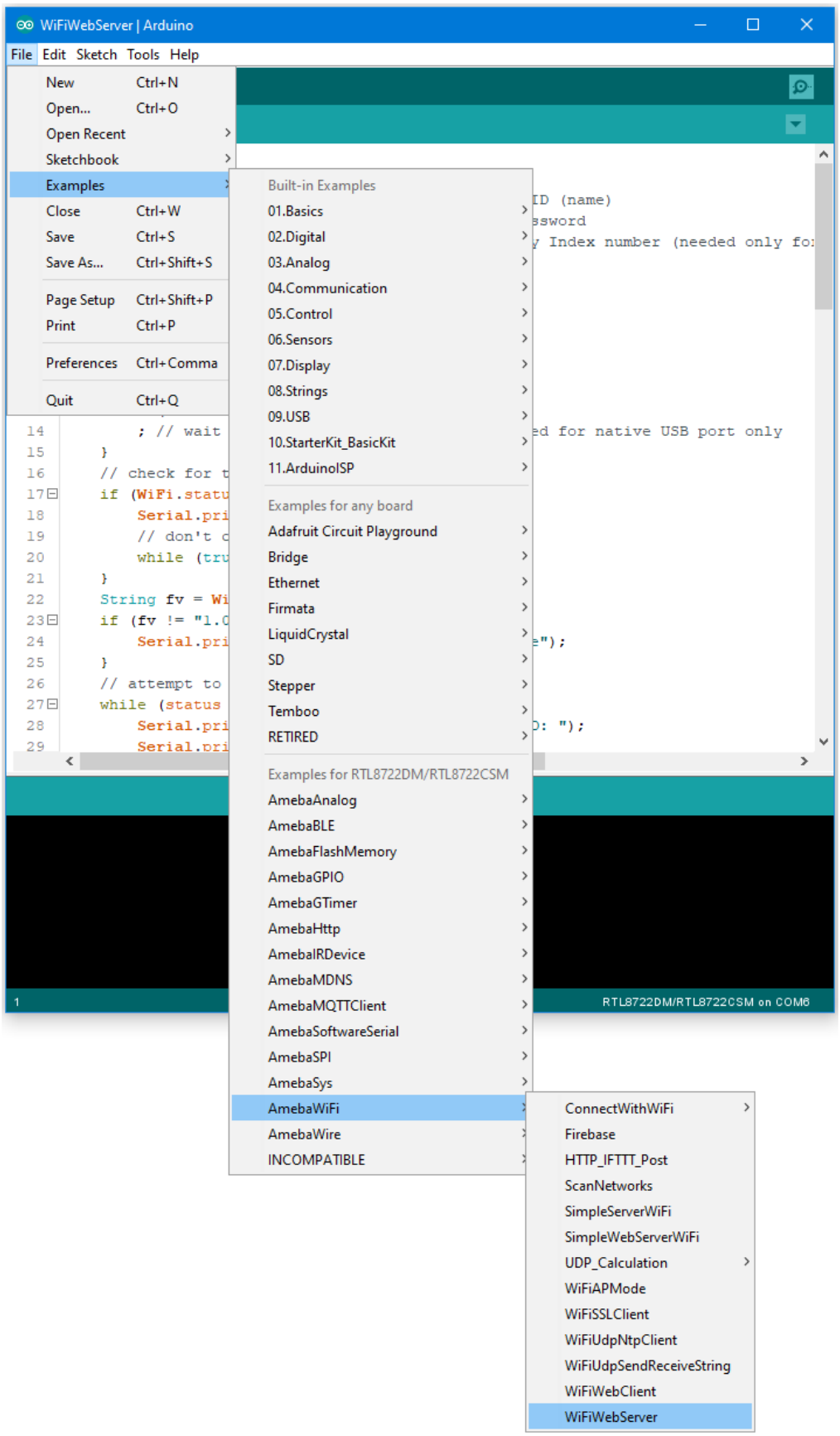
HTTP - Set up Server to Get the Ameba Status

Materials

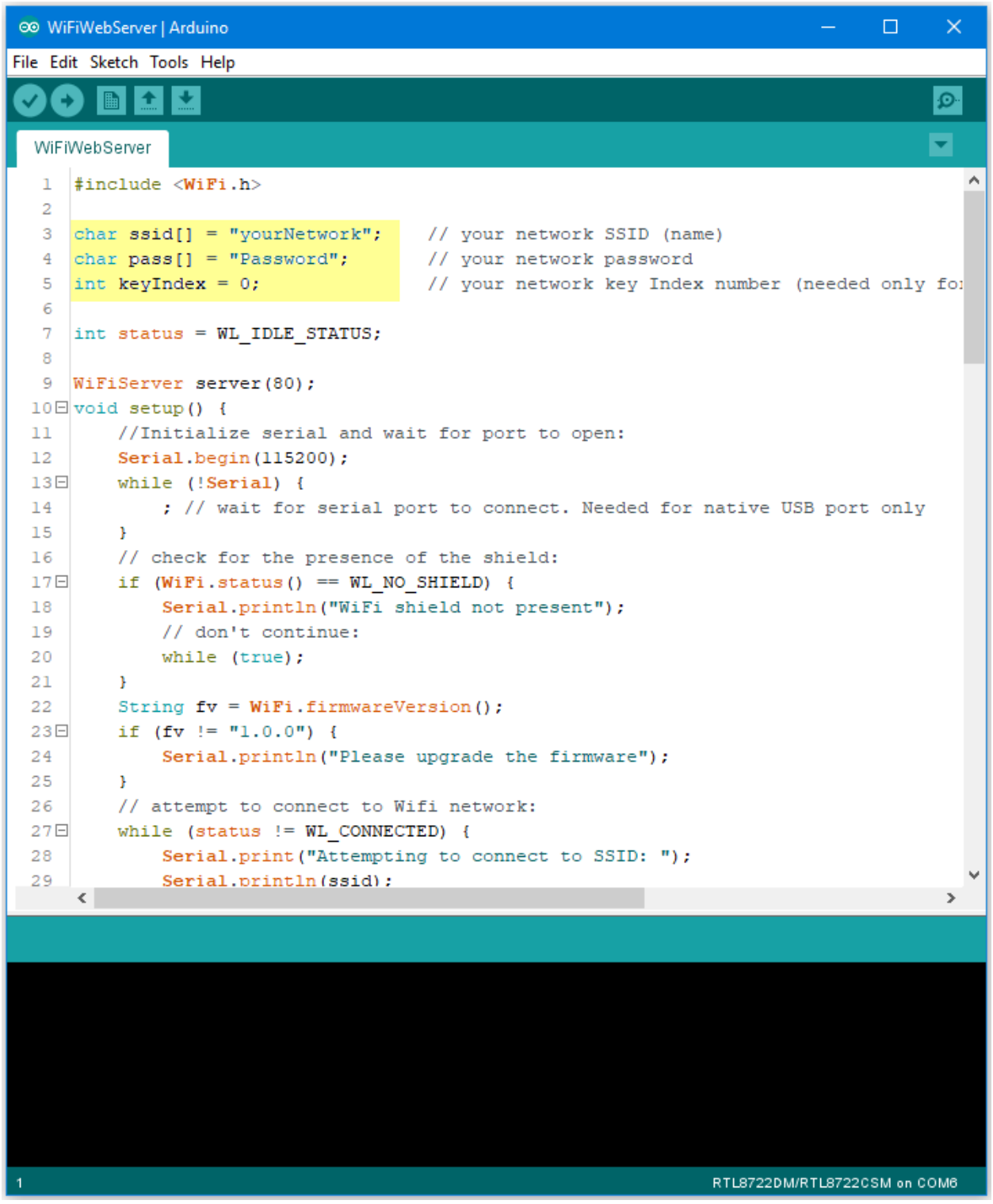
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

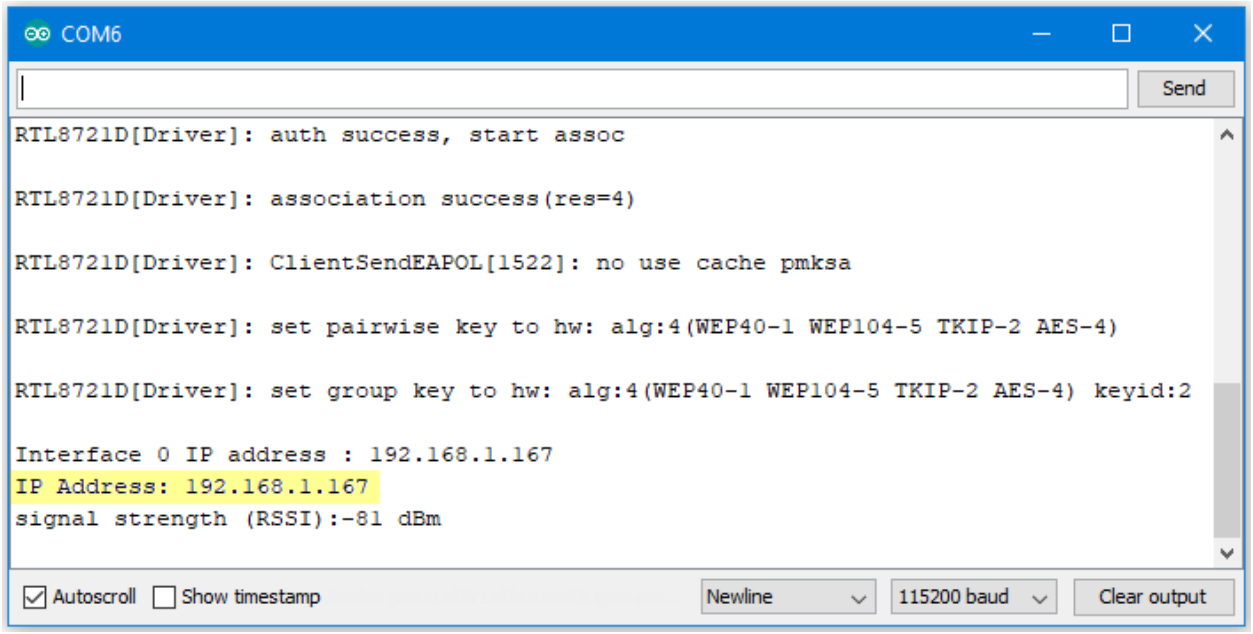
In this example, we connect Ameba to WiFi and use Ameba as server to send message to connected client.
First, open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiWebServer`"


WiFiWebServer | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
ConnectWithWiFi
Firebase
HTTP_IFTTT_Post
ScanNetworks
SimpleServerWiFi
SimpleWebServerWiFi
UDP_Calculation
WiFiAPMode
WiFiSSLClient
WiFiUdpNtpClient
WiFiUdpSendReceiveString
WiFiWebClient
WiFiWebServer
RTL8722DM/RTL8722CSM on COM6

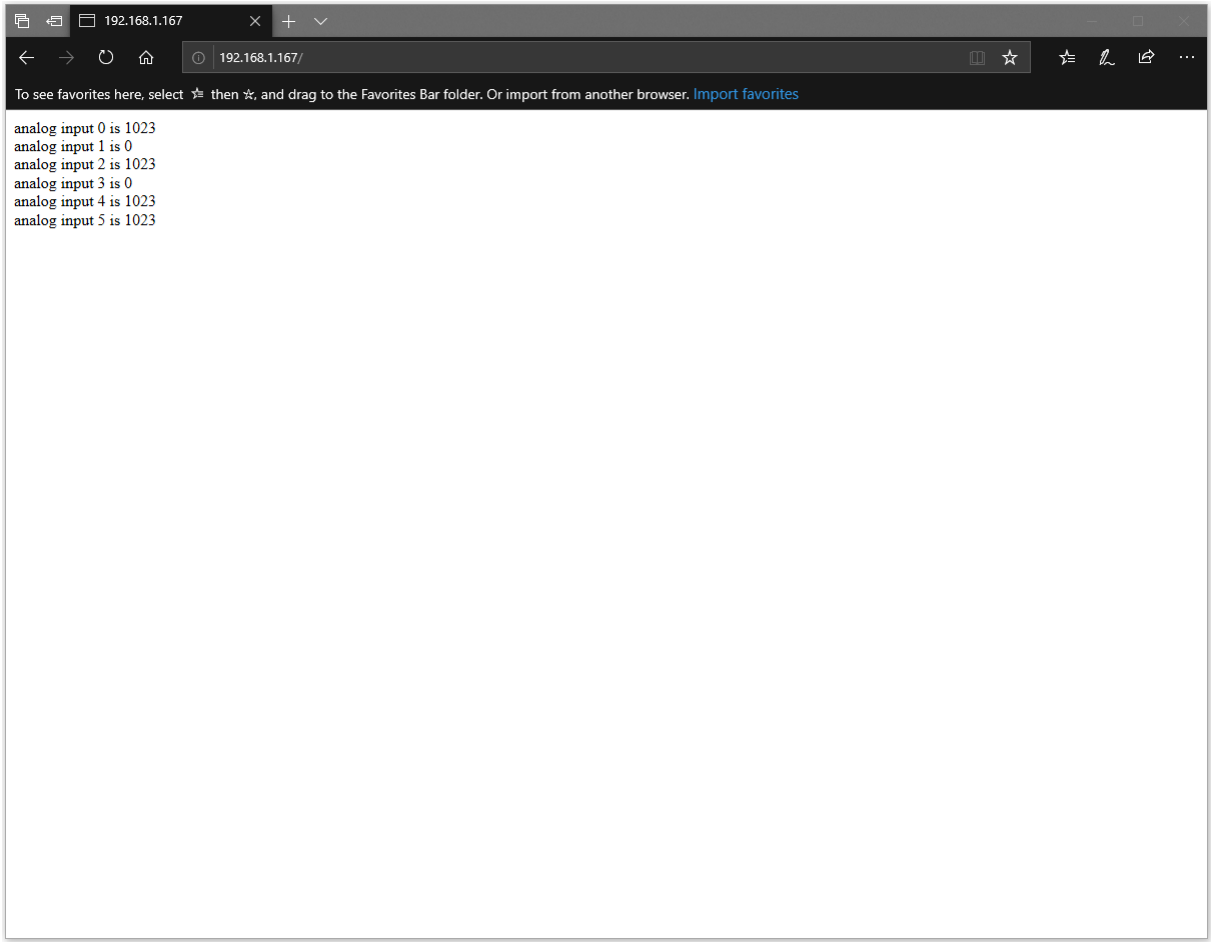
In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. After connecting to WiFi, Ameba starts to run as server. The IP of the server is shown in the serial monitor, and port is 80.



We connect to the server in a browser, and we can see the data sent from the server.



Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
se `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to check whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

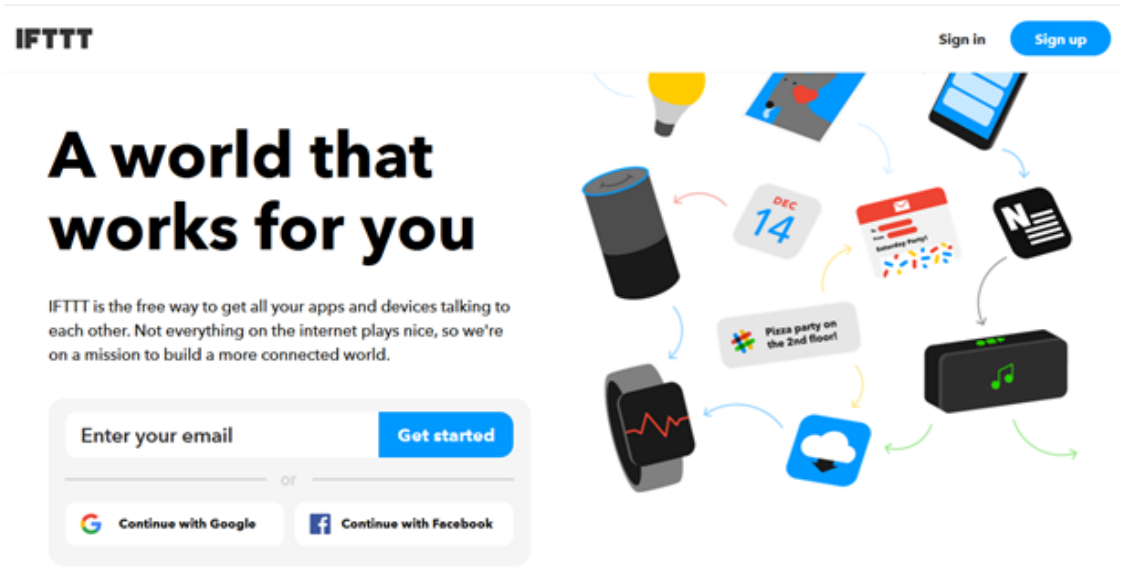
HTTP - Use IFTTT for Web Service

Introduction to IFTTT

IFTTT, known as If This Then That, is a website and mobile app and free web-based service to create the applets, or the chains of simple conditional statements. The applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, Pinterest etc.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- An account from https://ifttt.com/ , in order to access IFTTT service*



Note: Upon log in, there are several cloud and online services that are integrated with IFTTT platforms.

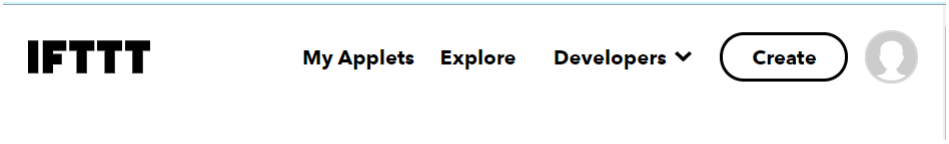
Example

- Generate Applet from IFTTT

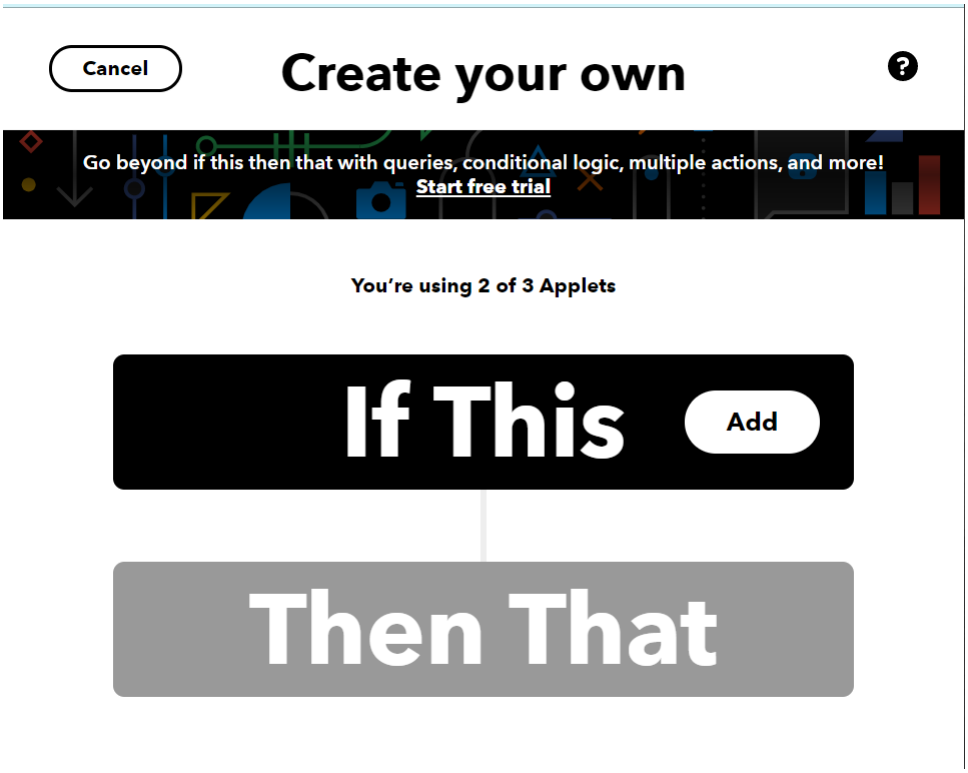
  In this example, we obtain an example of IFTTT Applet to send email to specified recipient.

  To run the example, HTTP POST feature of the Ameba is used to post a simple webhook service that is received by IFTTT platform and in turn be used to trigger a response (sending an email).

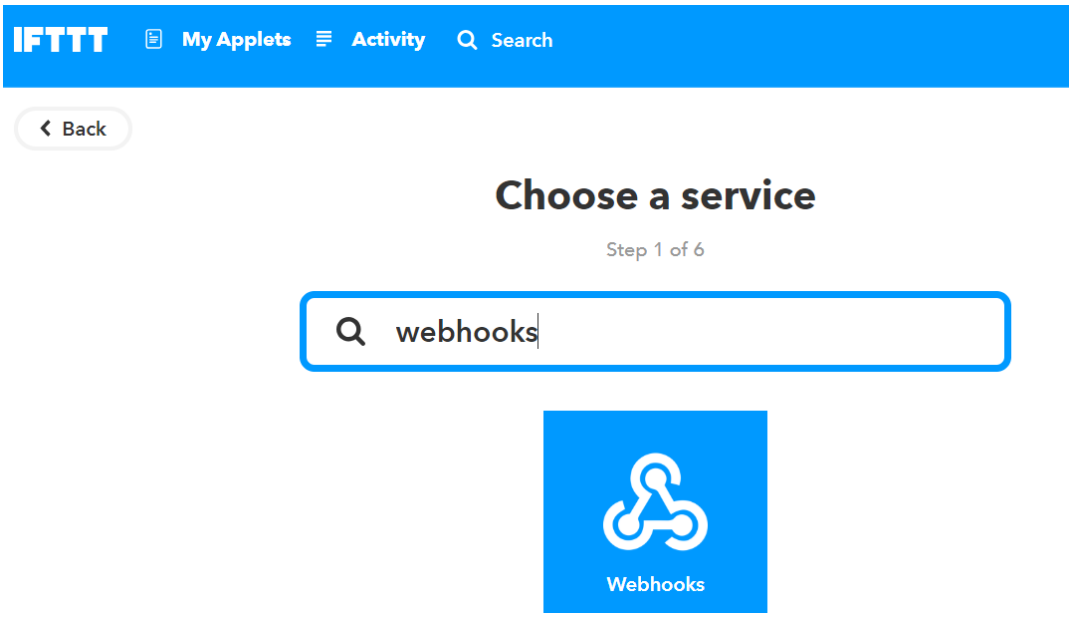
  After logging in https://ifttt.com/, click **Create** from the top bar.

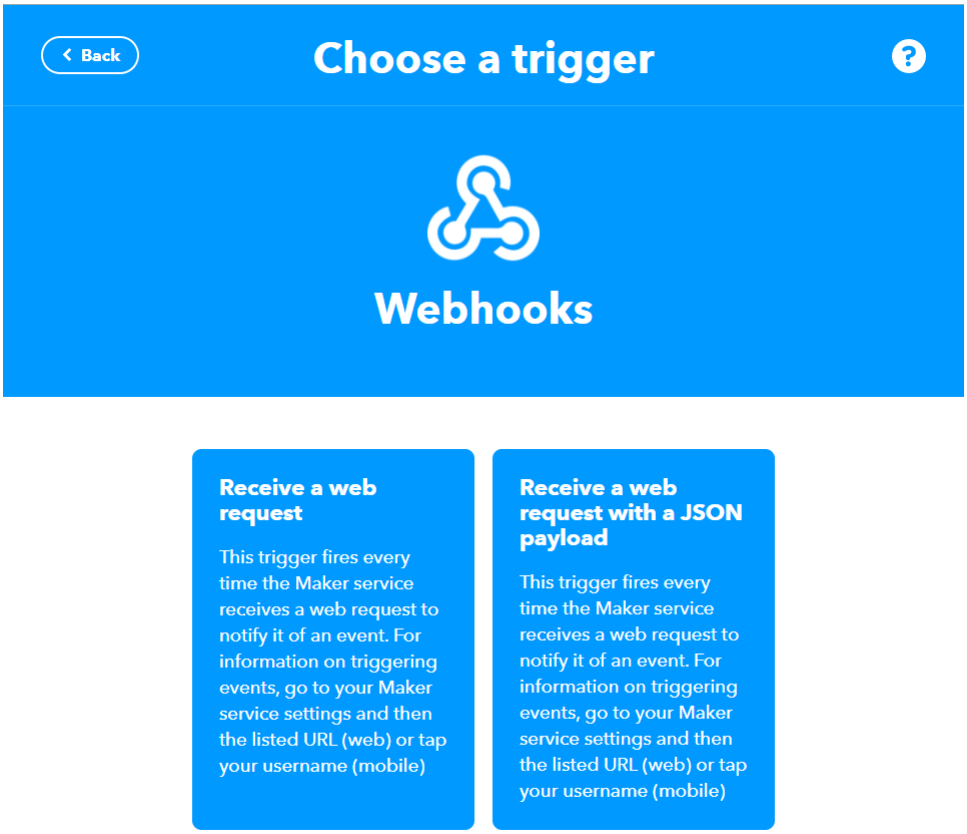
  Click **"Add"** to add the trigger.



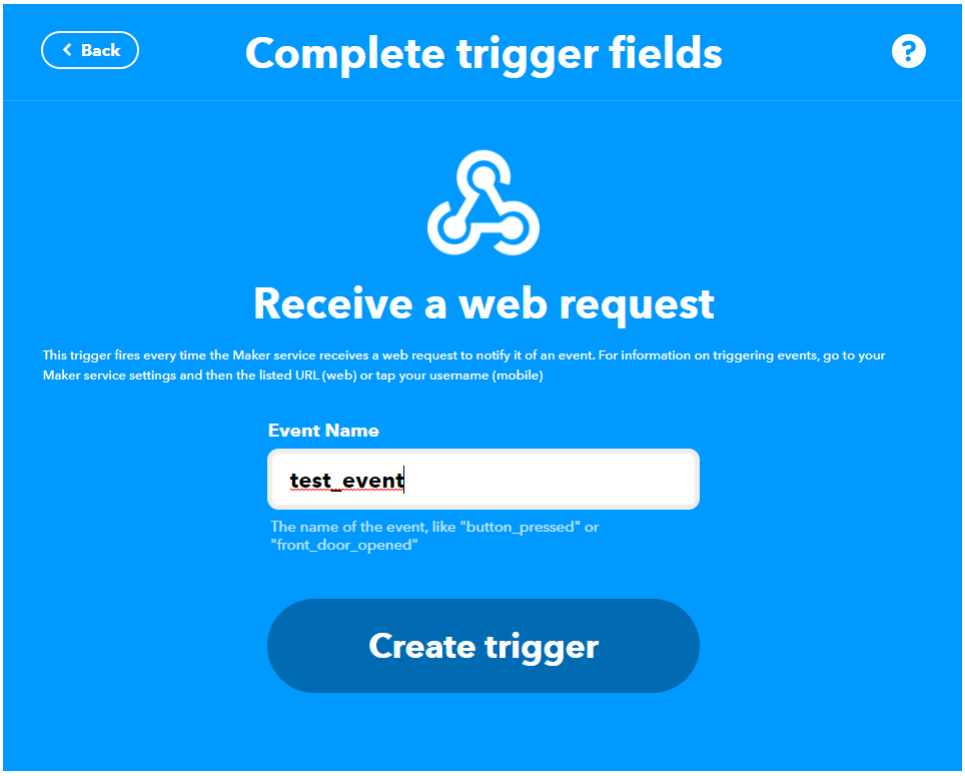
Choose Webhooks service as shown below. Alternatively, search the service by typing into the search bar.



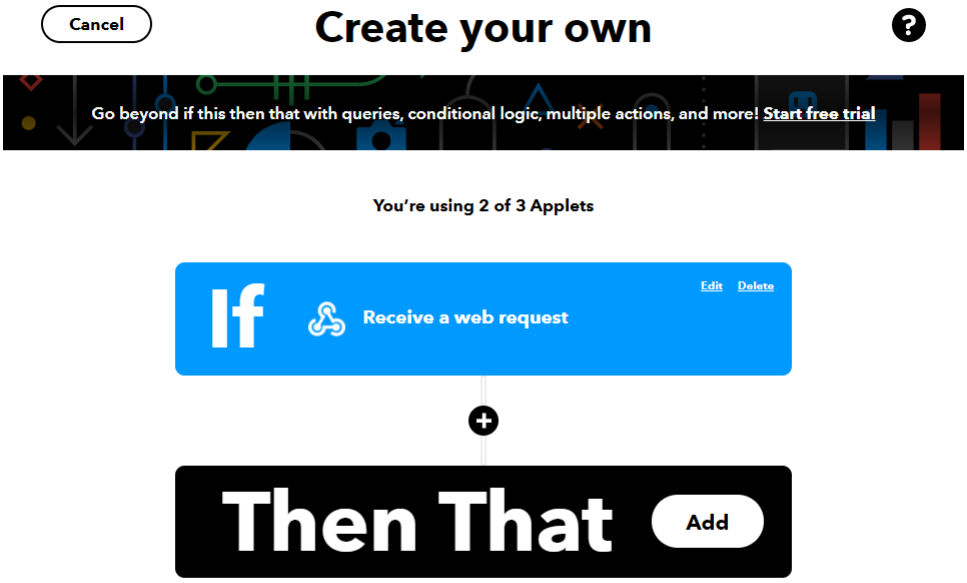
After that, the available triggers will appear. Choose Receive a Web request.



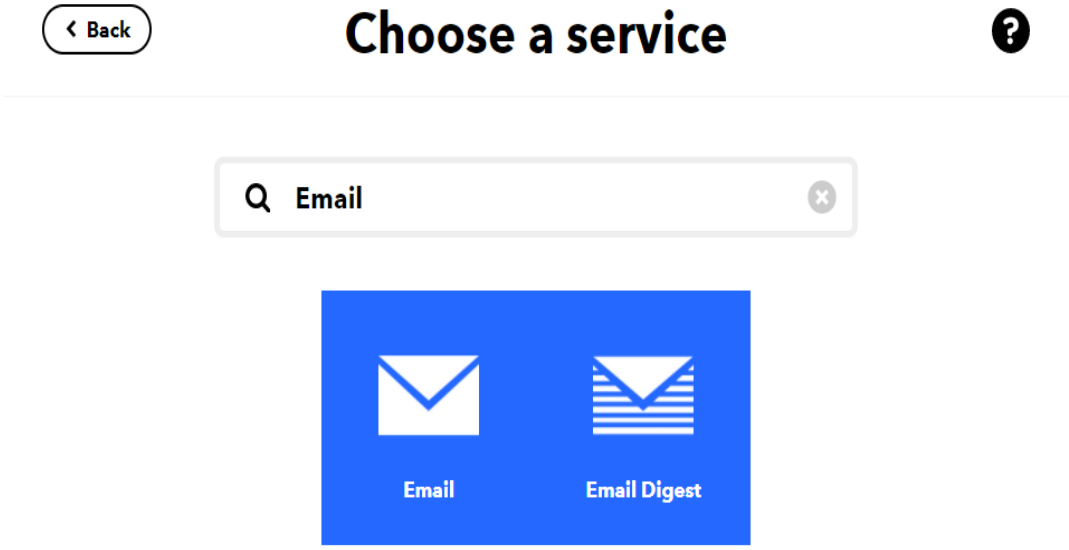
Next, an Event Name is required to identify the trigger successfully. In this example, set the Event name as “test_event”.



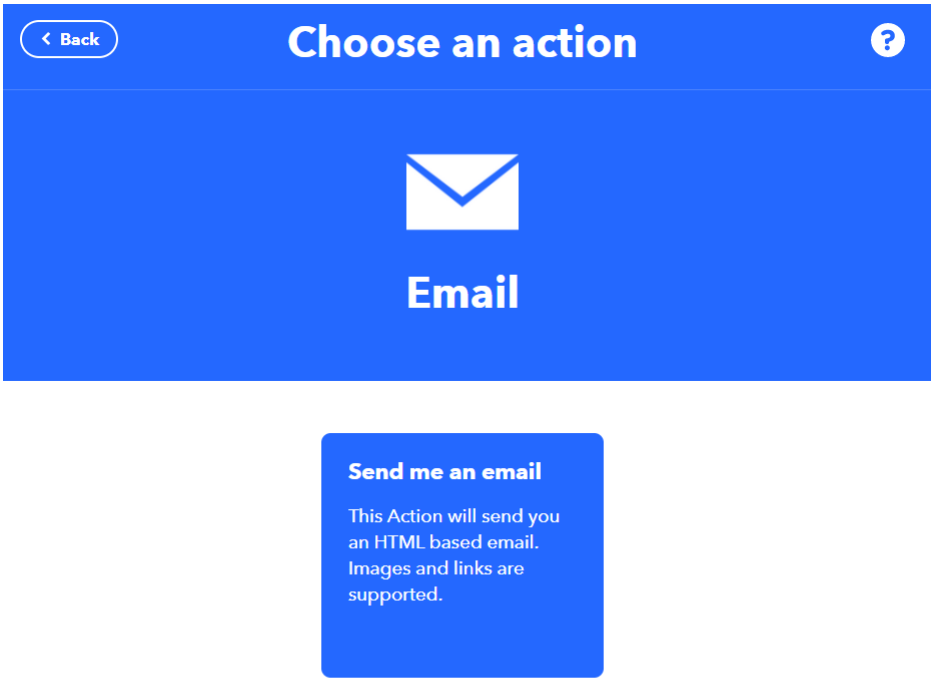
Next, click **Add** in Then That field to create the action service taken in response to the last trigger.



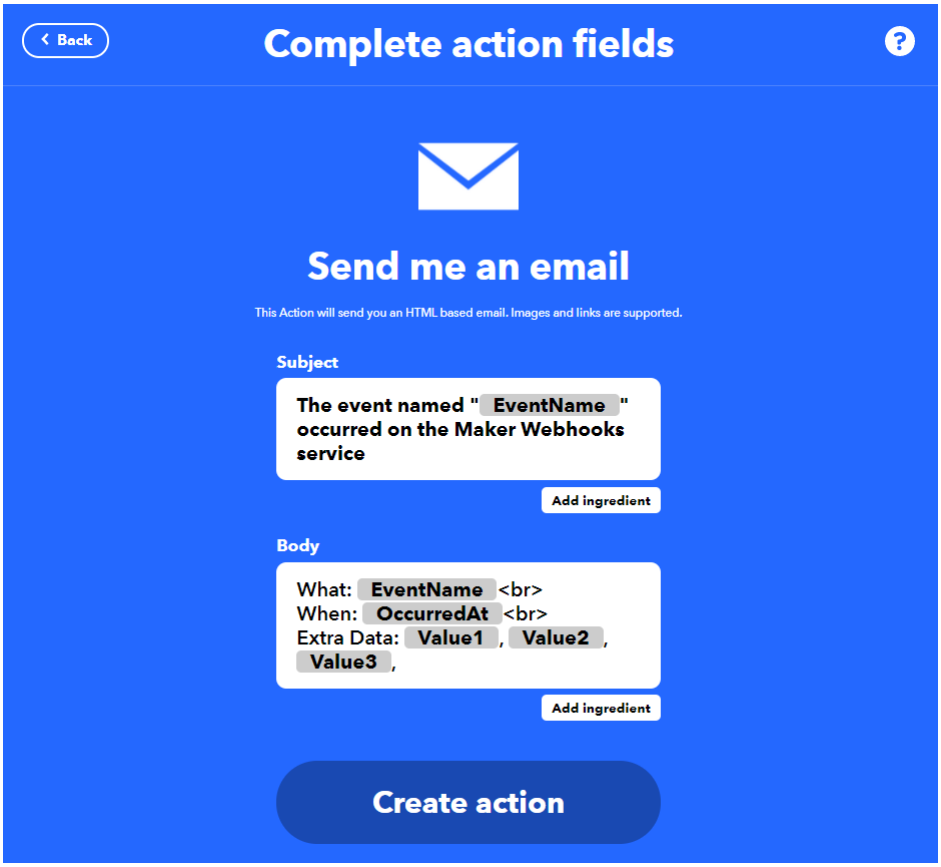
Choose Email as the action service.



Click on Send me an email.



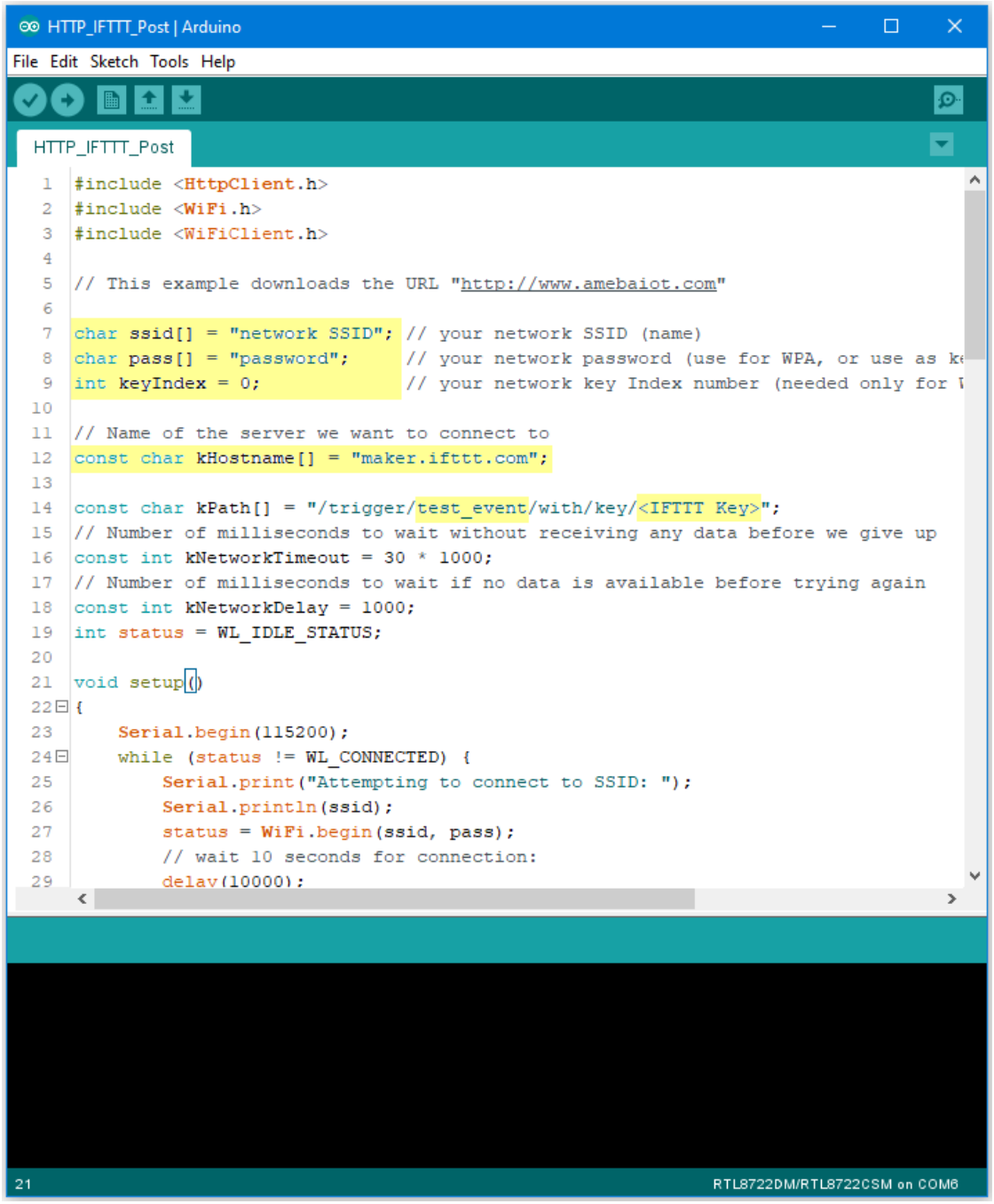
Under the template of **Send me an Email**, the contents of the email, such as subject and body is editable. Click **Create Action** to complete the action. Take note that **Email service** is offered to the email address registered under IFTTT account.



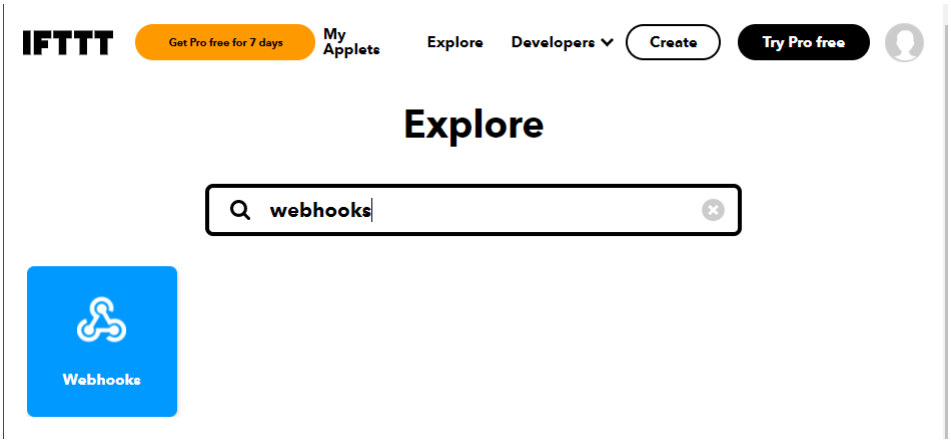
- Post the Trigger via Ameba

Once the Applet is ready in the IFTTT dashboard, the example program can be flashed onto the Ameba board to post the HTTP request.
Open the example code in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`HTTP_IFTTT_Post`"
In the example program, edit the following 3 items inside the code to make the program work.

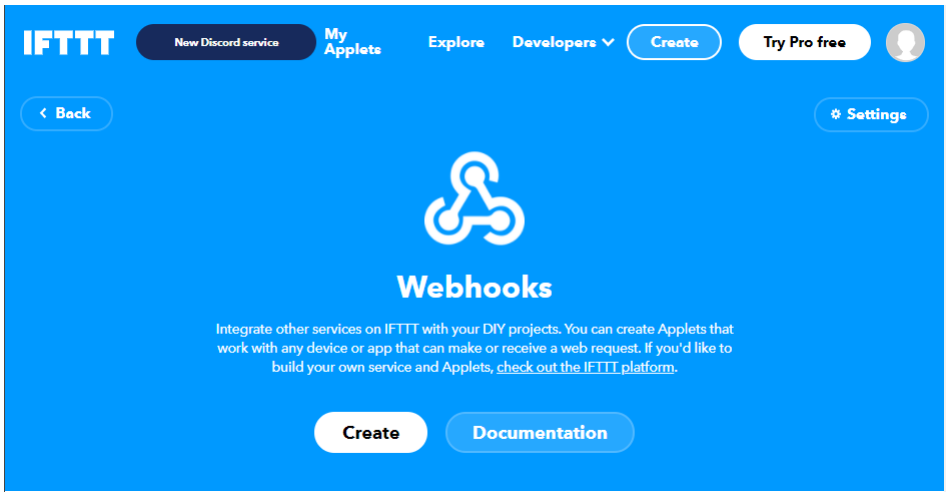
1. The WiFi credentials to connect to the Wi-Fi hotspot or access point of desirable choice.
2. Under the Host name field, enter the host name of the IFTTT service "maker.ifttt.com".
3. Under the Path name field, enter the Event name and key field "/trigger/Event name/with/key/Key Field"
   - Event name: The event name should be the same as the one specified in the IFTTT applet. In this example, the event name is "test_event".
   - Key Field: Available under webhook service in individual IFTTT account. See the next step for the steps to obtain the Key Field.



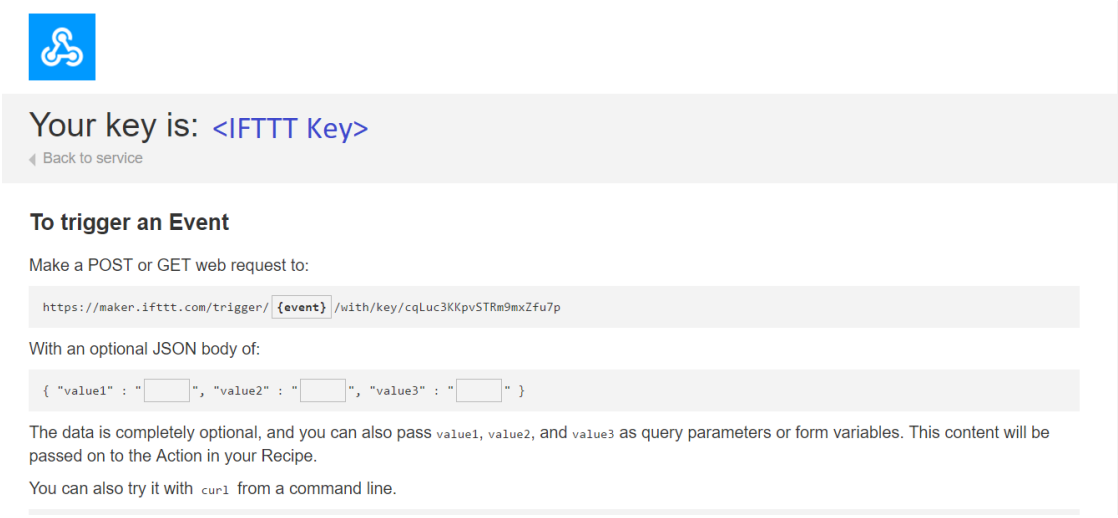
To obtain a key from documentation tab of the Webhooks, find the webhook service in the Explore tab.



On the Webhooks service page, click on the Documentation tab.

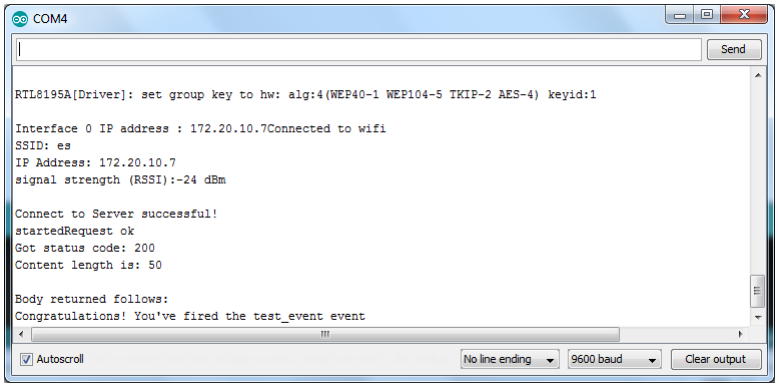


The key can be found in the documentation page. Also, information on how HTTP request can be used.

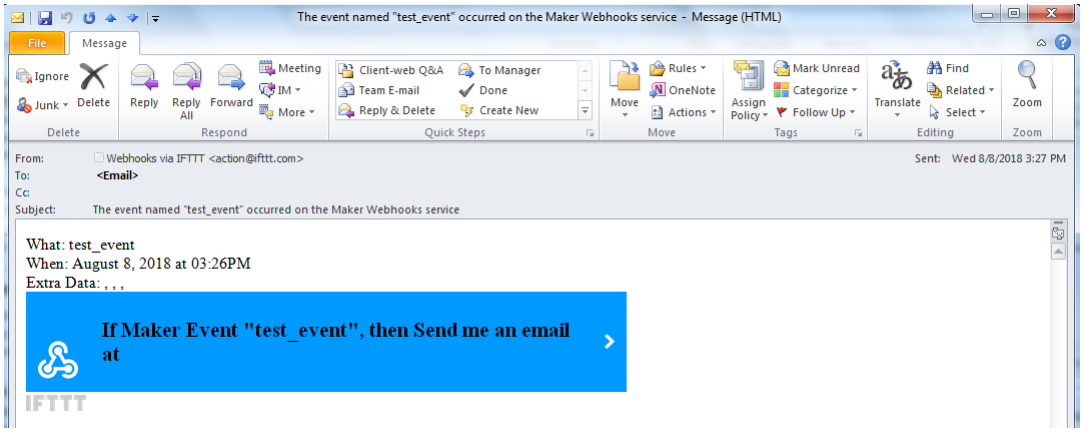


Once the example is ready, connect to Ameba board via USB Cable.

On the Arduino IDE, compile the code and upload the code onto Ameba and press the reset button. After the event has been successfully fired, “Congratulations! You have fired the test_event event” can be seen on the serial monitor and an email reminder for this event will be delivered.



Thereafter an email is sent to recipient email account registered at IFTTT Applet and email notification will be received.



IPv6 – Ameba as IPv6 Server/Client over TCP

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

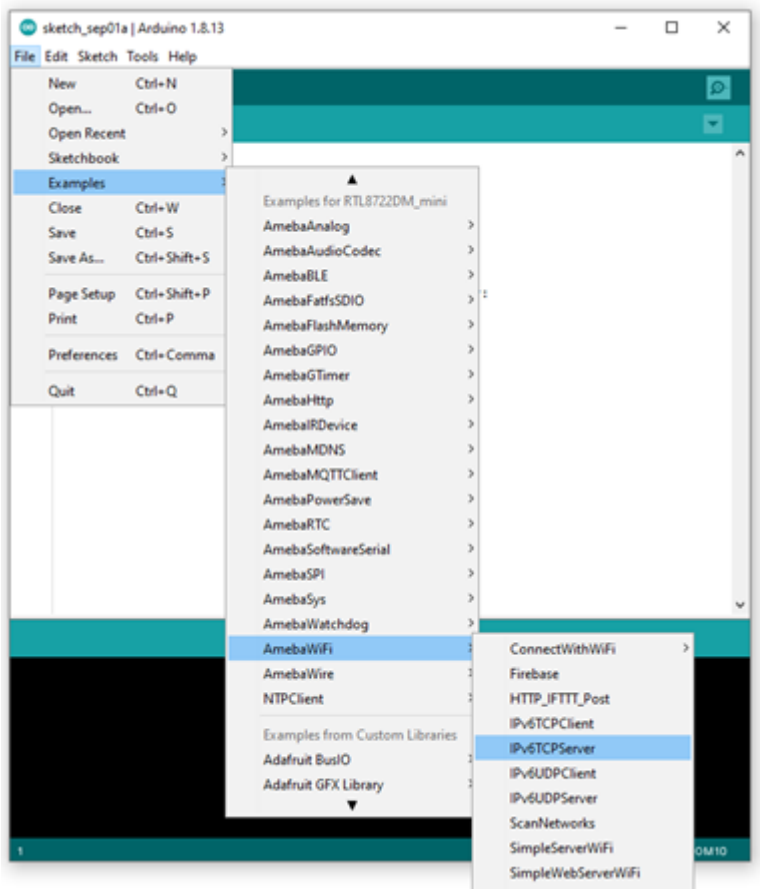
Example

Introduction

This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over TCP. Note that this example only works after you have set up the server and then configure the client accordingly.

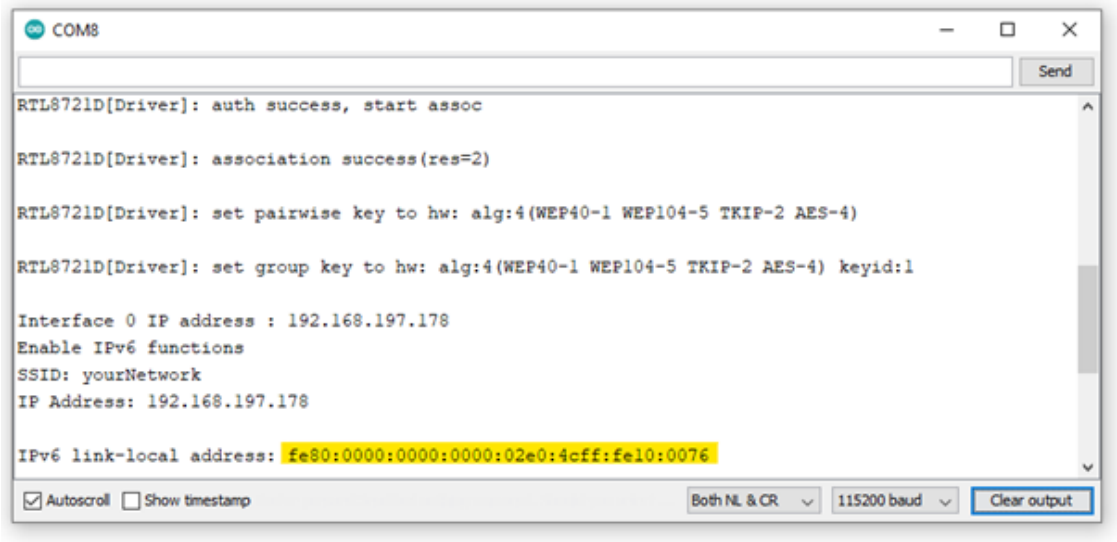
Procedure

Step 1. IPv6TCPServer Open the example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6TCPServer`”.

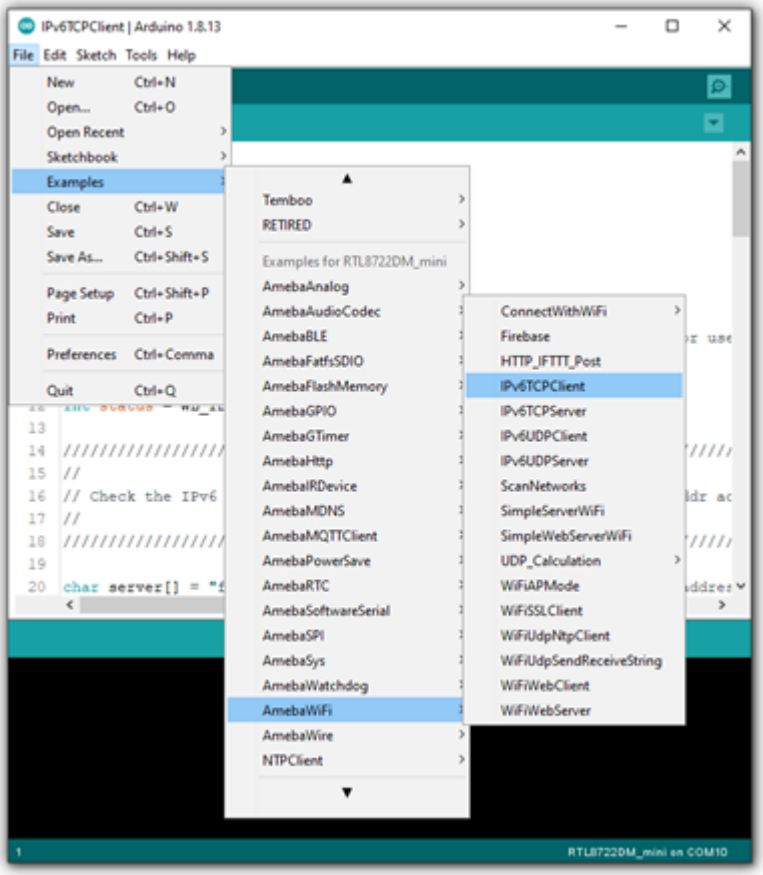


In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

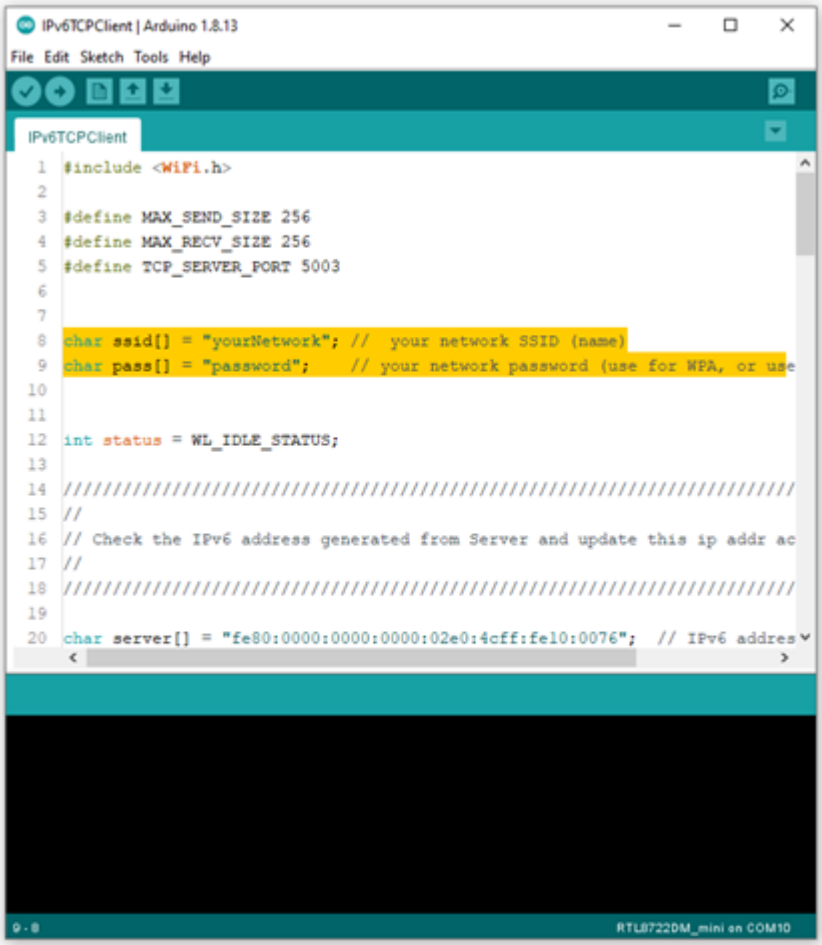
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



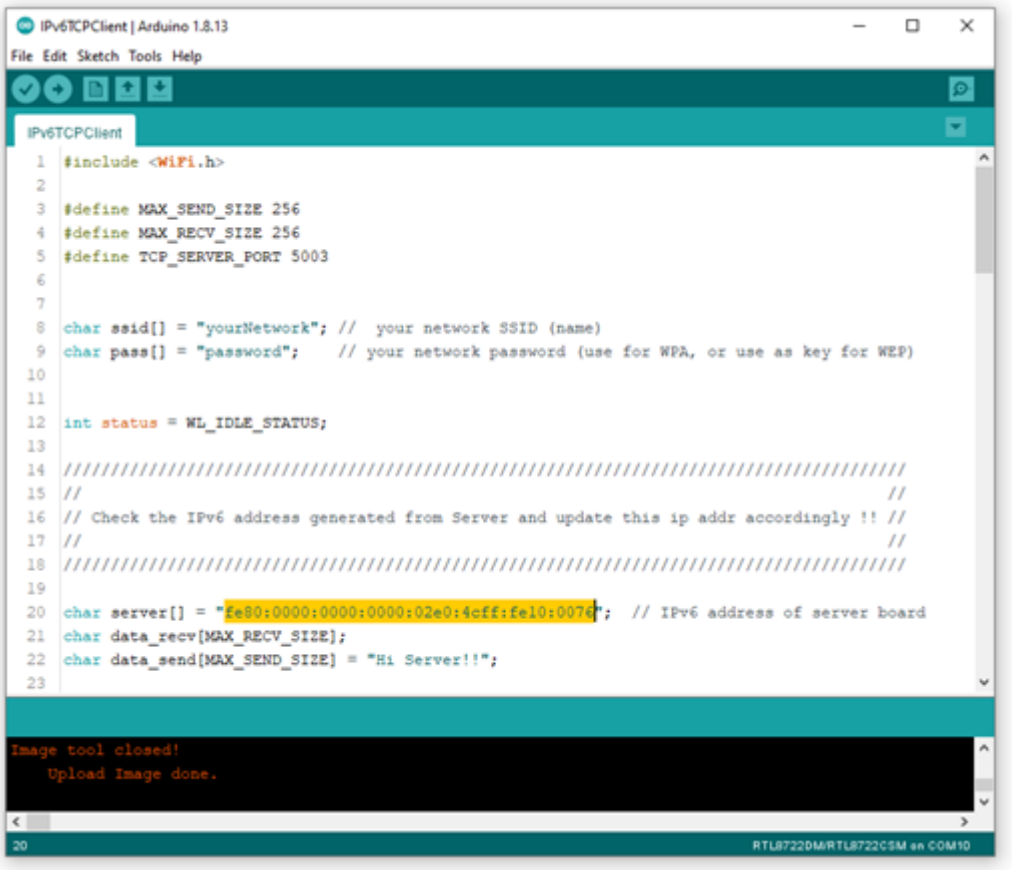
Step 2. IPv6TCPClient Now take the second Ameba D and open another example, "`Files` -> "`Examples` -> "`AmebaWiFi` -> "`IPv6TCPClient`".



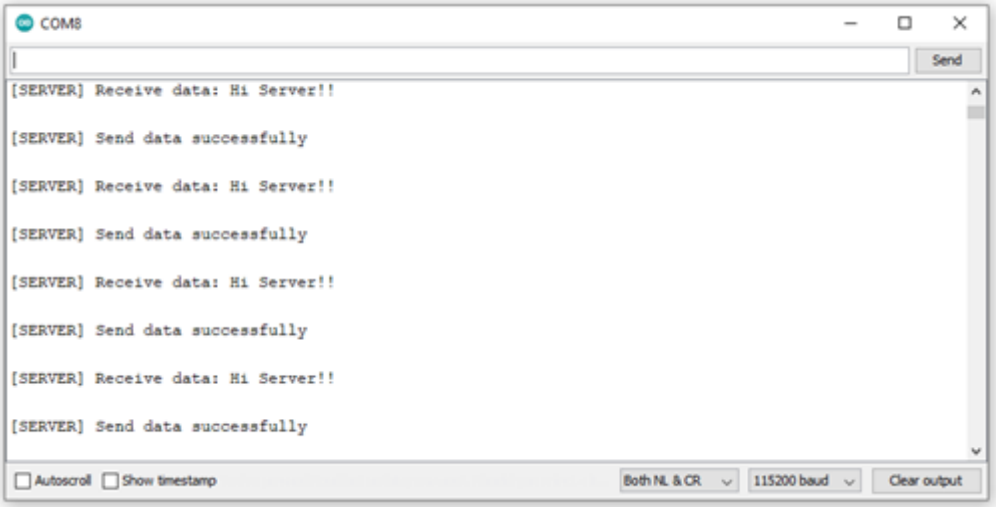
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

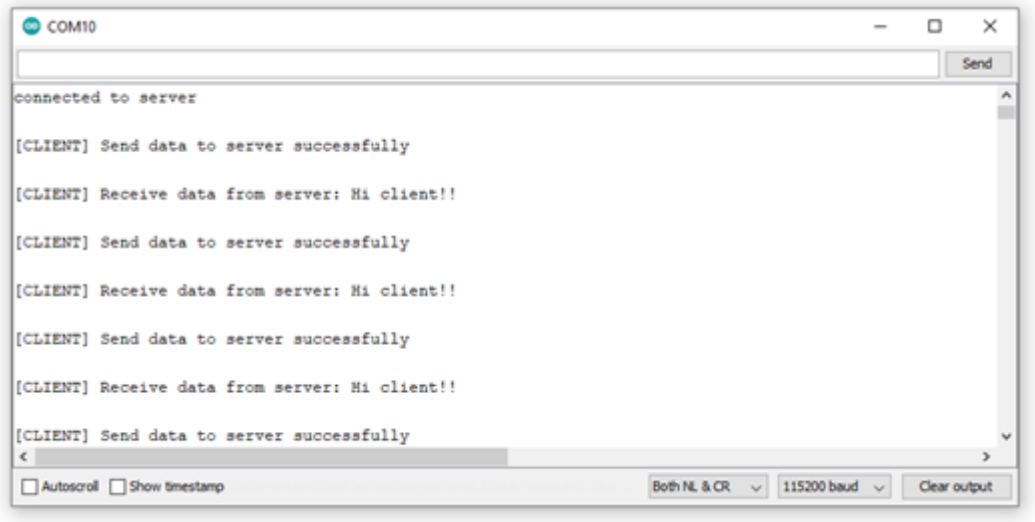


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6TCPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





IPv6 – Ameba as IPv6 Server/Client over UDP

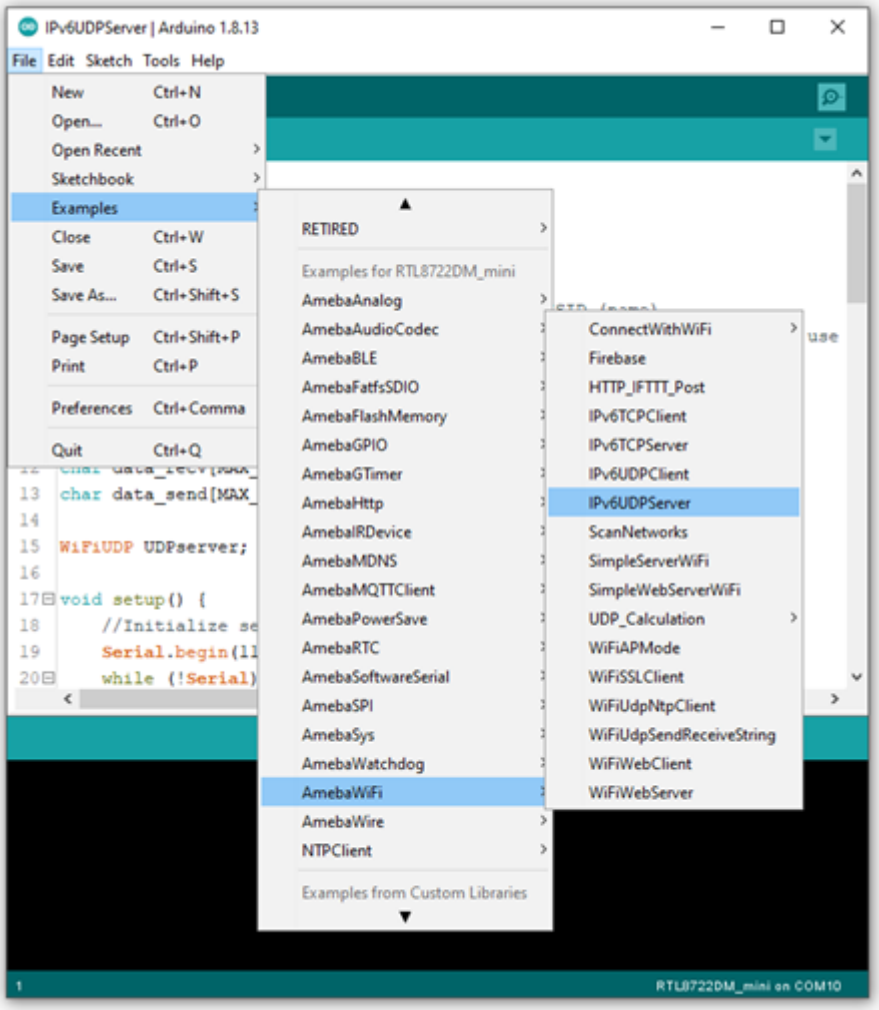
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

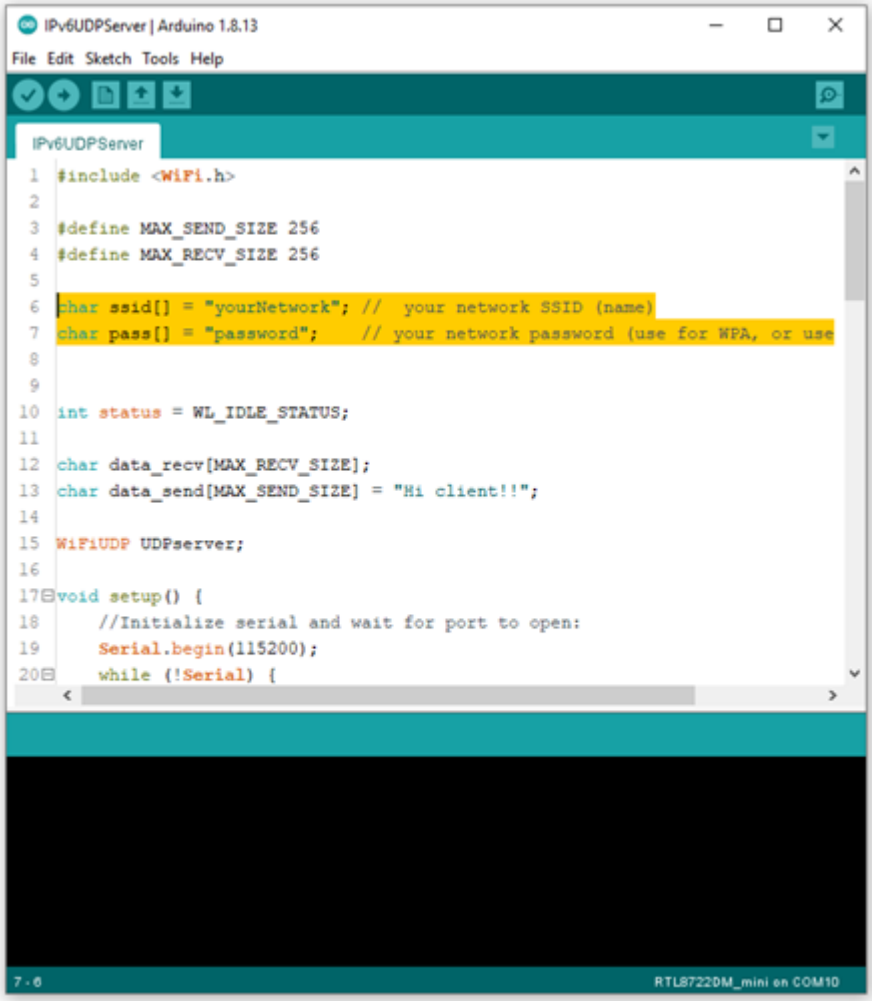
Example

Introduction

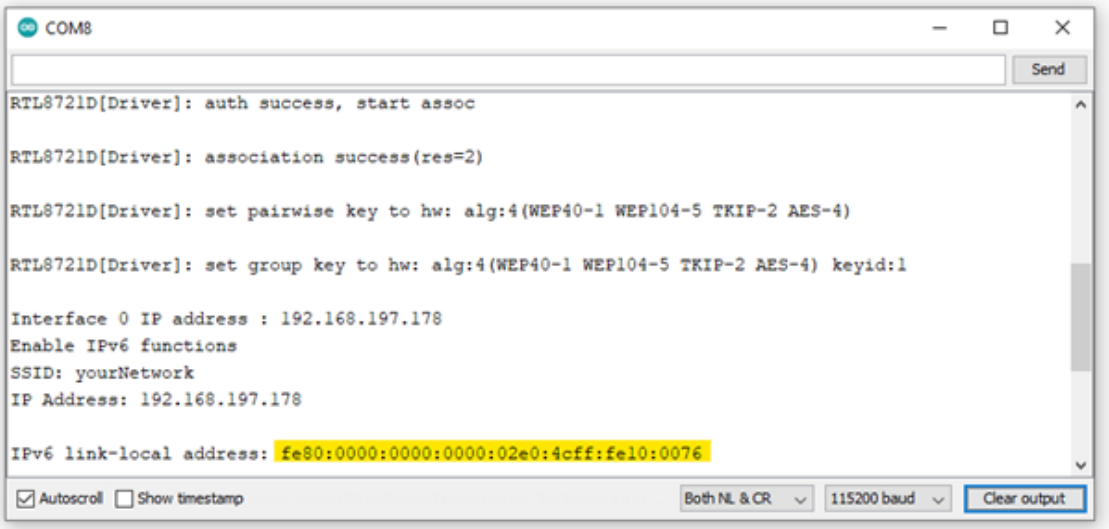
This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over UDP. Note that this example only works after you have set up the server and then configure the client accordingly.



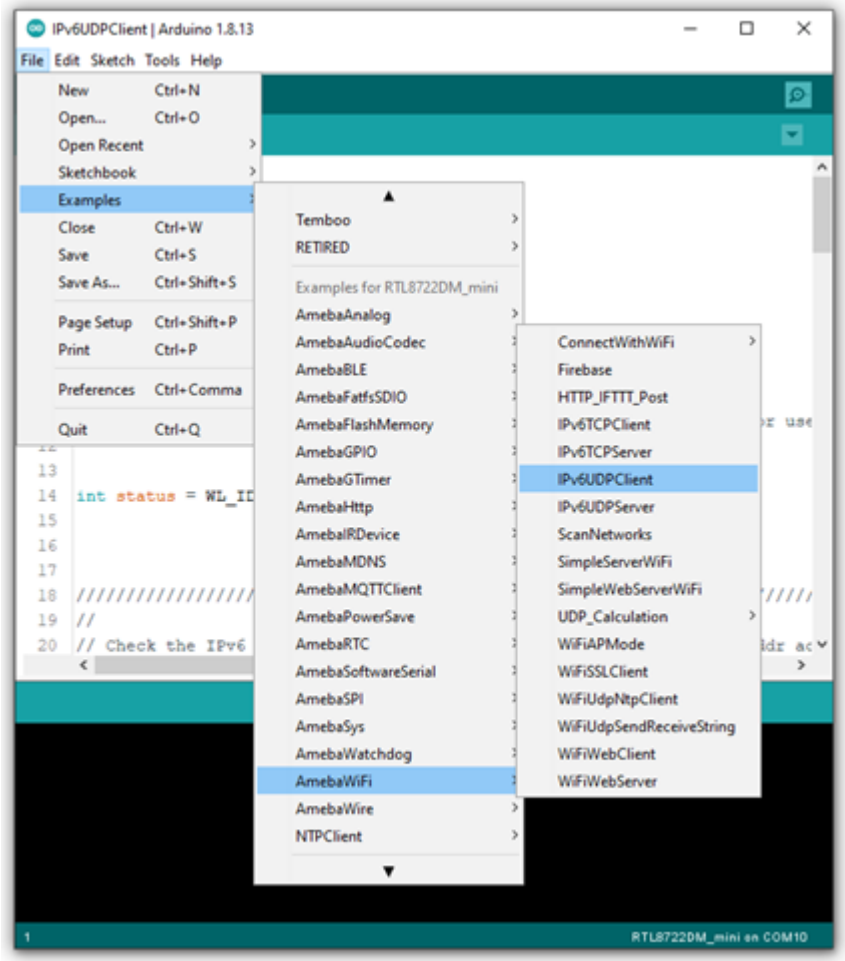
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.



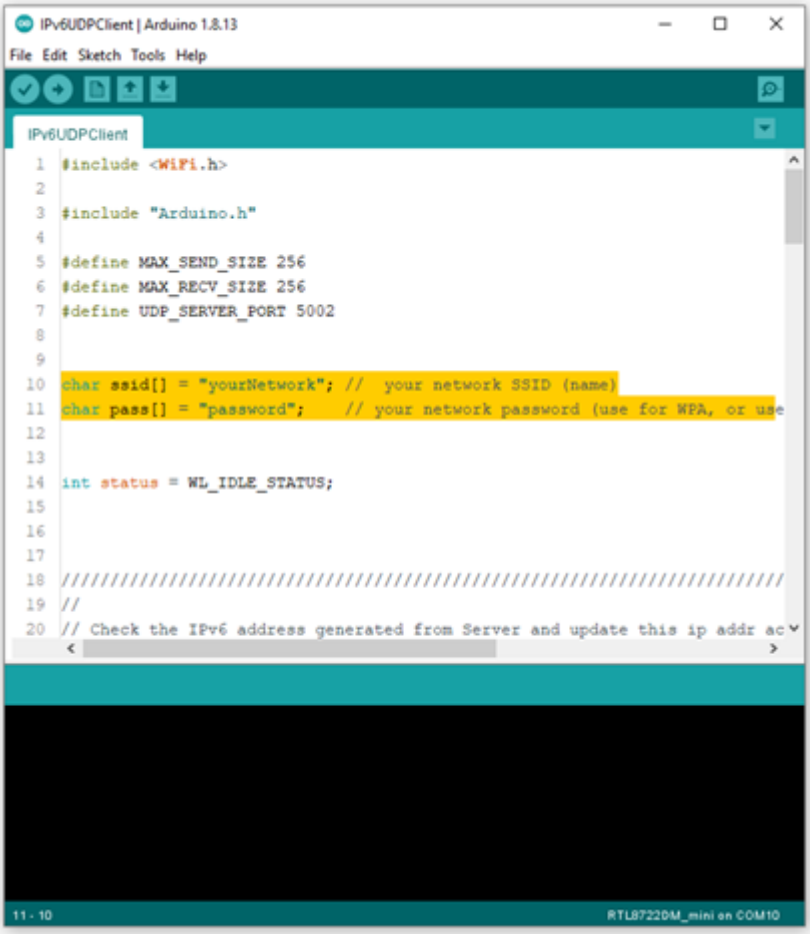
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



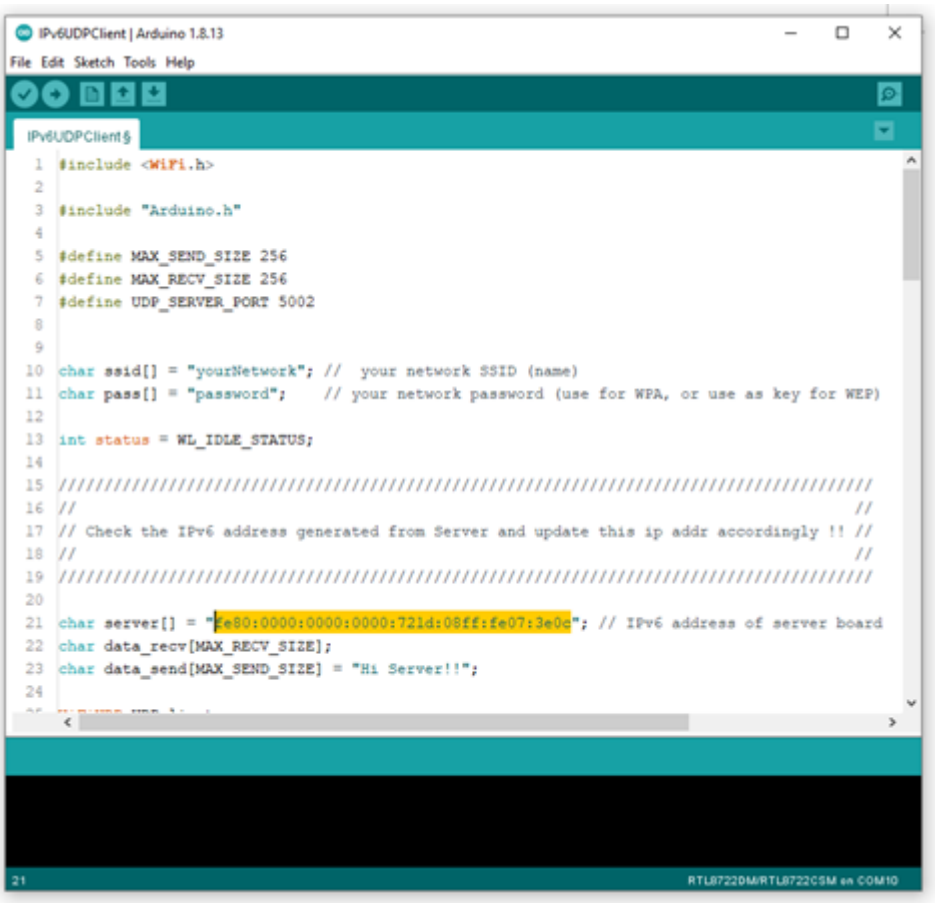
Step 2. IPv6UDPClient Now take the second Ameba D and open another example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6UDPClient`”.



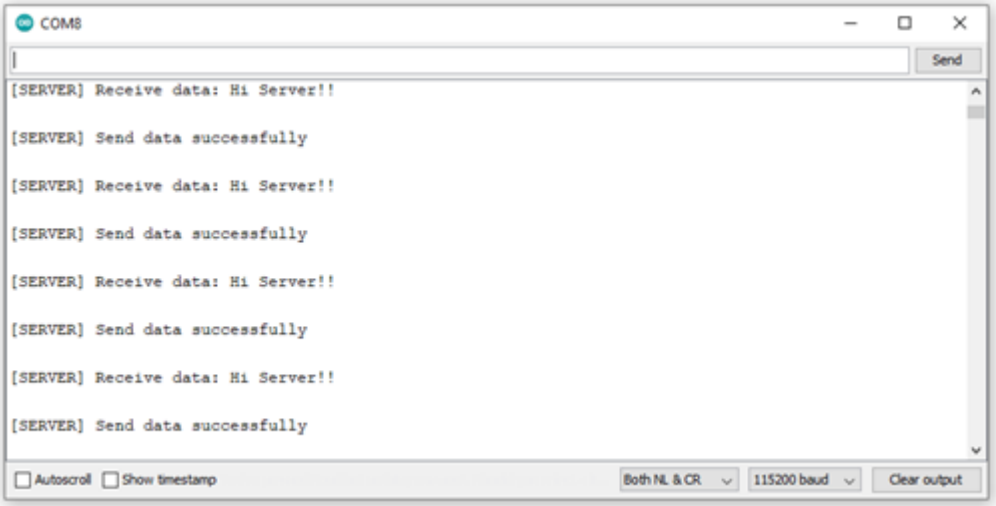
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

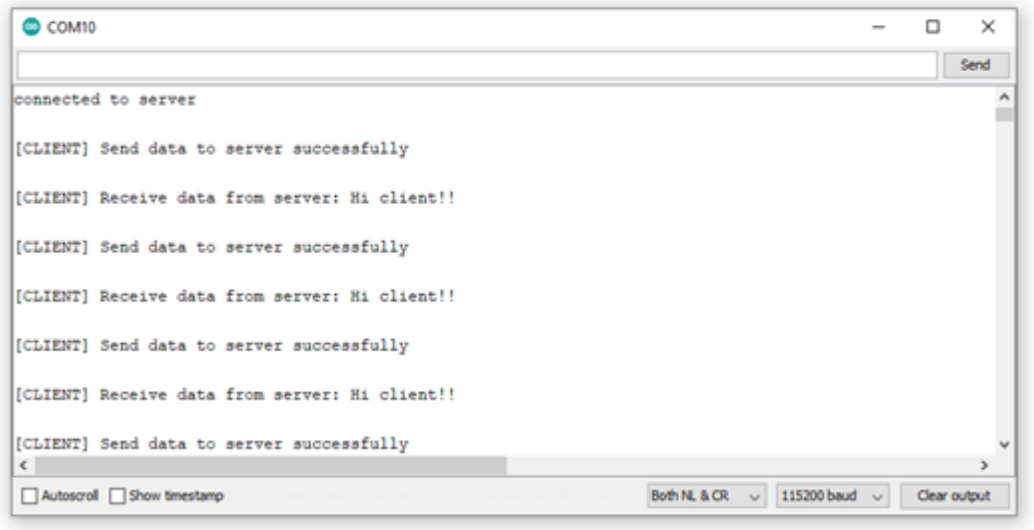


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6UDPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





MDNS - Set up mDNS Client on Arduino IDE

Preparation

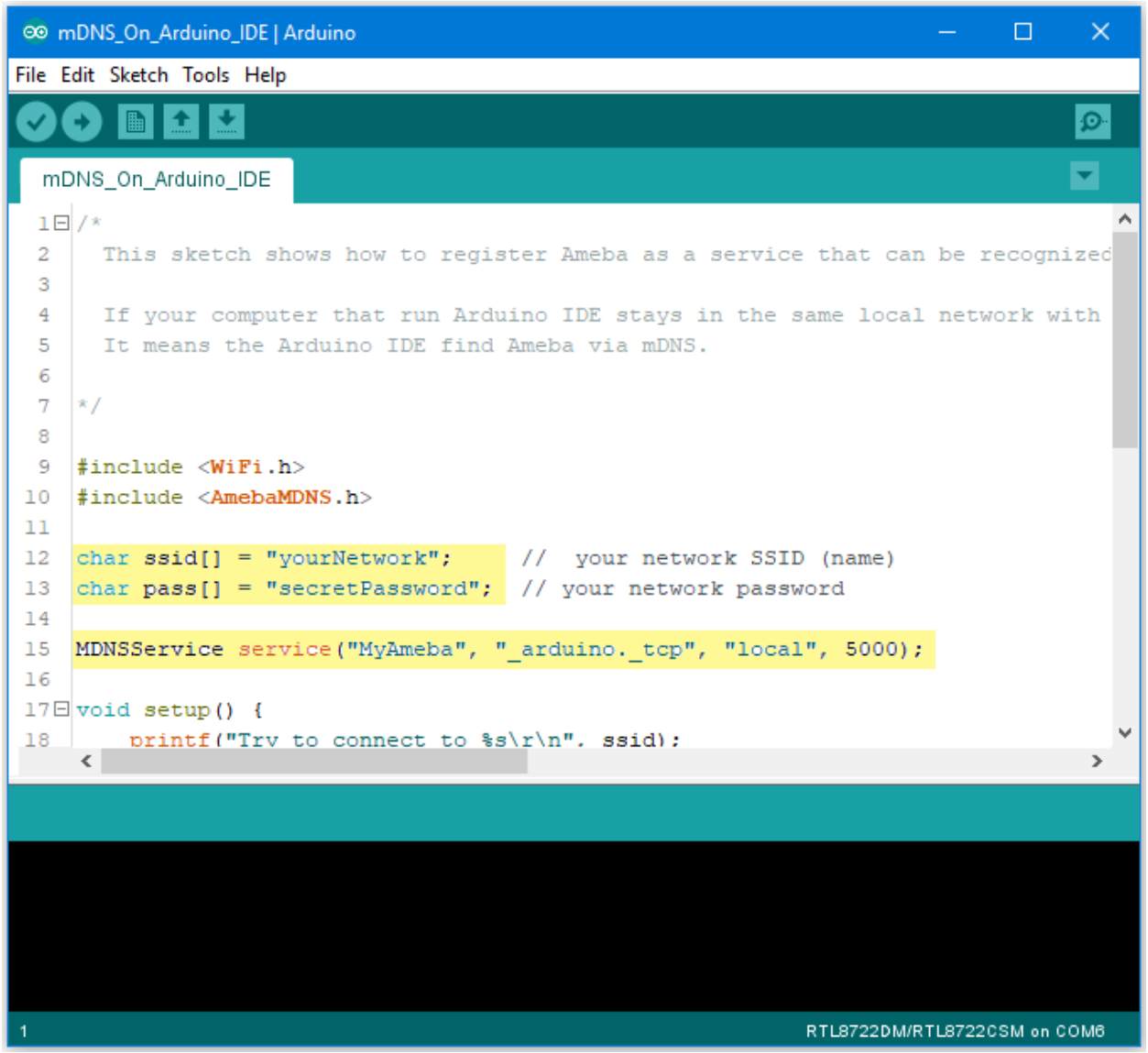
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

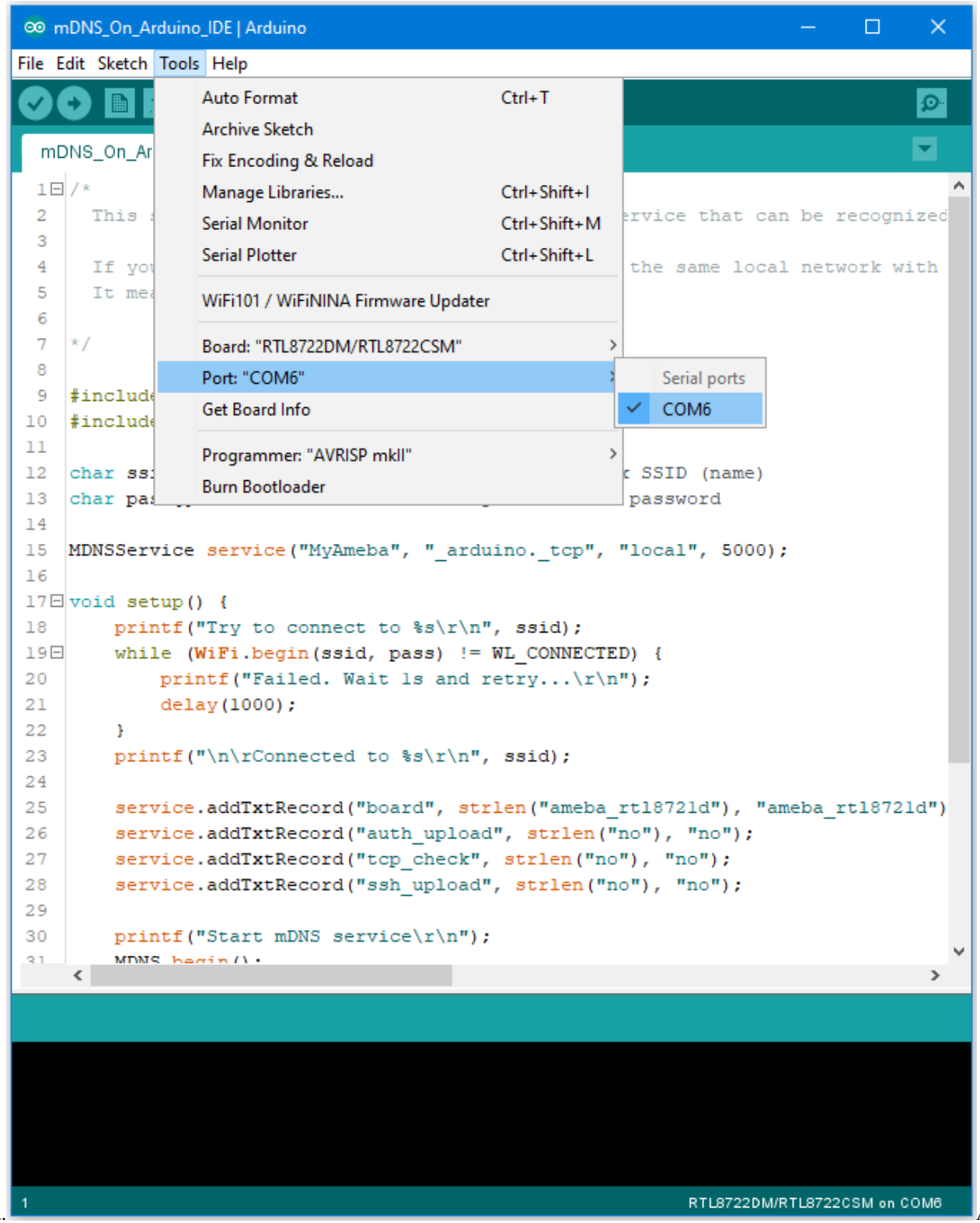
mDNS (Multicast DNS) is a protocol used in the local area network. It delivers the network information like IP address and provided services to others. mDNS is based on the UDP protocol, and it sends packets to 224.0.0.251 with port 5353 under IPv4 address. The naming style for the service follows the format: **{Instance Name}.{Protocol Name}.{Domain}**

- Instance Name: used to identify the name of the service
- Protocol Name: Divided into two parts, the front end is in regard to the name of the service, and it adds baseline as a prefix. The rear end is in regard to the transport protocol name it used, and it also adds baseline as a prefix
- Domain: Local area network in normal cases

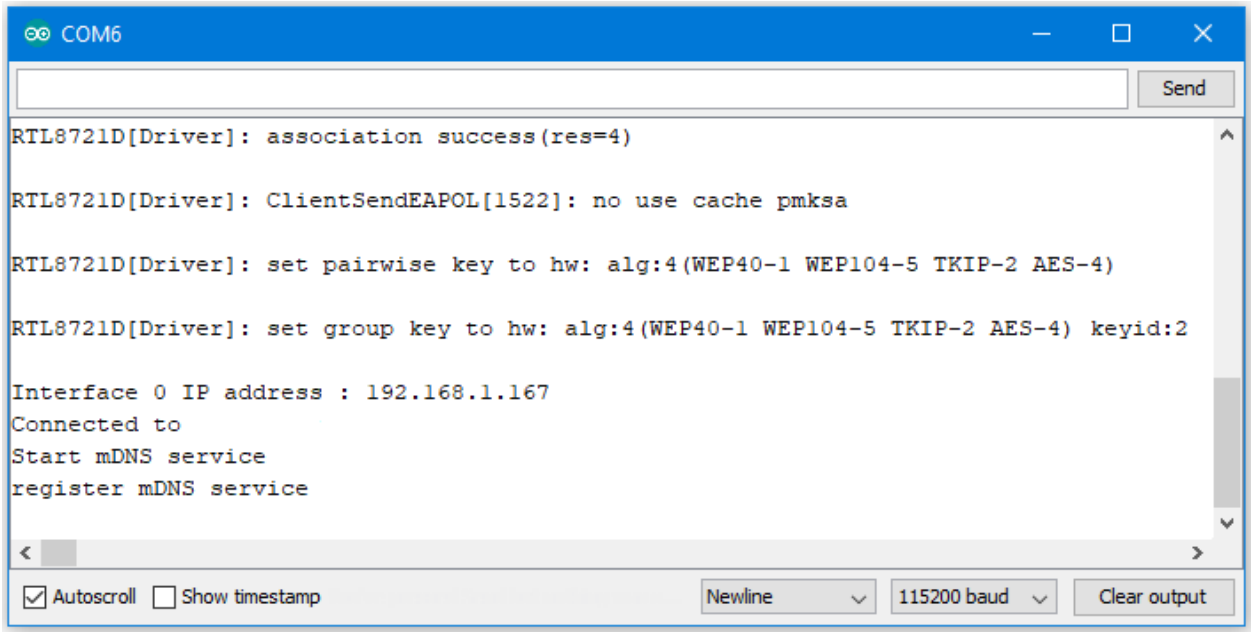
For example, Arduino IDE adopts the naming for the mDNS service which is used in OTA as following: **MyAmeba._arduino._tcp.local**
Among the naming example, “MyAmeba” can identify the Ameba device name and the name “MyAmeba” is changeable. “_arduino._tcp” is the protocol that Arduino IDE adopts, and the Domain is set as local in common.
Open the example, “`File`” -> “`Examples`” -> “`AmebaMDNS`” -> “`mdns_on_arduino_ide`”
You need to input ssid and password of the AP because the example will use WiFi connection.
And you can find out the naming of the service at the place where it declares MDNS Service. The example uses the default name “MyAmeba” and the name is changeable.



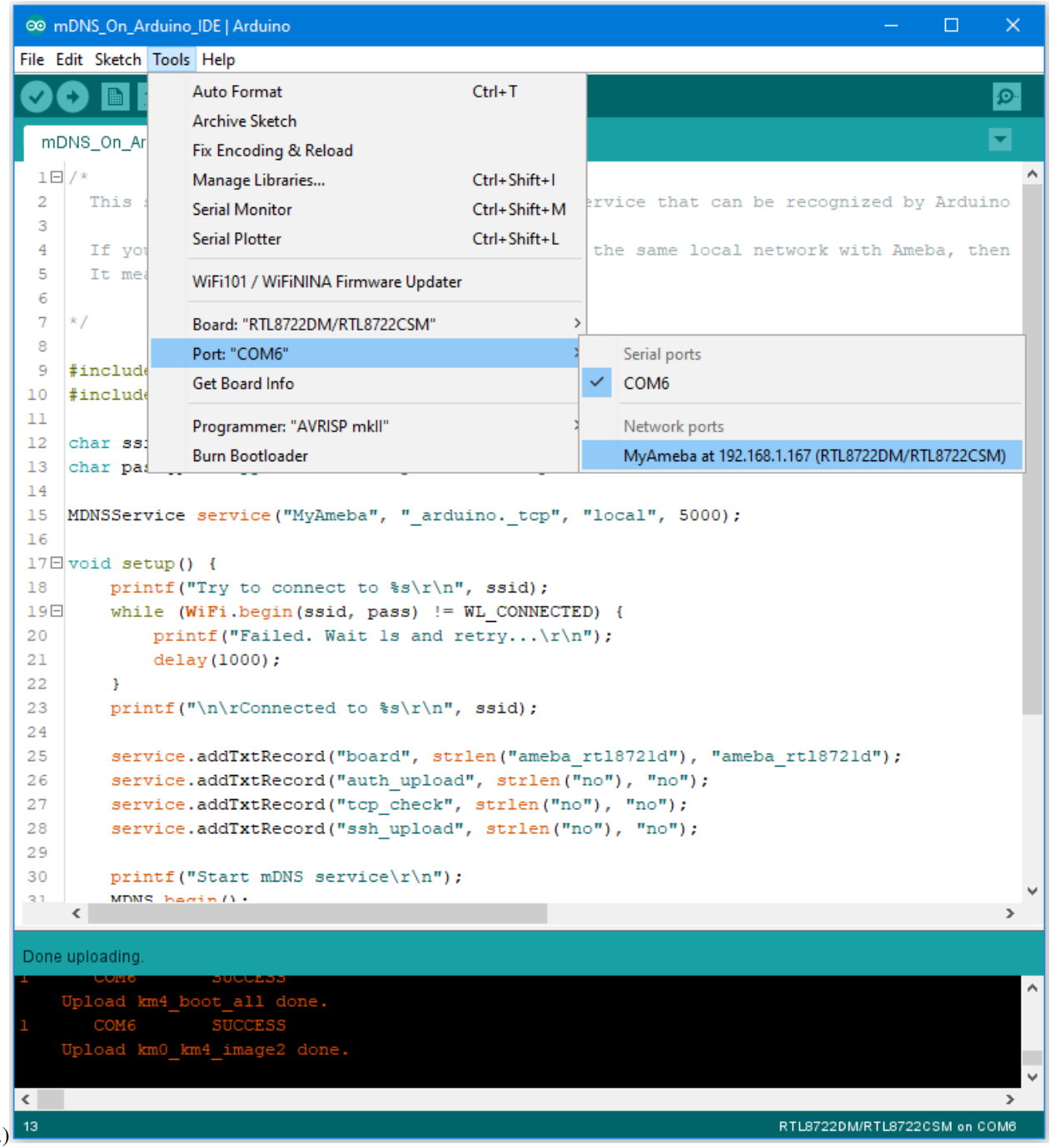
Next, go to ("Tools" -> "Port"), and you can find out at least one Serial Port. This port is simulated by Ameba board via USB. Choose this port and upload the compiled code to



Ameba. After uploading the code, press the reset button on Ameba and waiting for Ameba to connect with AP and activate the mDNS service after a while. You can see the Log at the bottom of the Serial Monitor.



Then you can find out the added item "Network Ports" **"MyAmeba at 192.168.1.167 (Ameba RTL8722DM/RTL8722CSM)"**, "MyAmeba" is the device name we set up, and "IP" is the IP address that AP assigned to Ameba, the IP address should be the same with the IP shown in the Serial Monitor. Last, "Ameba RTL8722DM/RTL8722CSM" is the type name of the board, and it means that Ameba can let Arduino IDE identify the mDNS service successfully.(We still can not use the Internet to upload the code, and we will explain this part in the OTA



example.)

If you cannot find the Network ports on your Arduino IDE, please check

- Does your computer in the same local area network with the Ameba?
- Restart the Arduino IDE, and it will find the mDNS service again
- Check the Log in Serial Monitor if the Ameba connects to the AP and activate mDNS service successfully

Code Reference

The program set up the mDNS service in the beginning, the first parameter is Instance Name, and it is changeable in this example. The second parameter is the protocol that the service used, and it would be "_arduino._tcp" for Arduino IDE. The third parameter is Domain, and it would be "local" in common. The fourth parameter is the port number for the service, it is 5000 here and we doesn't use it in the example.

```
MDNSService service("MyAmeba", "_arduino._tcp", "local", 5000);
```

After connected to the network, we set up some text fields for the service. For the following example, "board" is the name of the field, "ameba_rtl8721d" is the value of the field. "board" is used to let Arduino IDE check installed SDK to see if it exists known device or not. We will use the name of the device if there is known device, users can change "ameba_rtl8721d" to "yun" or other names to find out what's the difference if interested.

```
service.addTxtRecord("board", strlen("ameba_rtl8721d"),"ameba_rtl8721d");
```

Then we add three text fields "auth_upload", "tcp_check", and "ssh_upload", this example does not activate these services.

```
service.addTxtRecord("auth_upload", strlen("no"), "no");
service.addTxtRecord("tcp_check", strlen("no"), "no");
service.addTxtRecord("ssh_upload", strlen("no"), "no");
```

Next we activate MDNS

```
MDNS.begin();
```

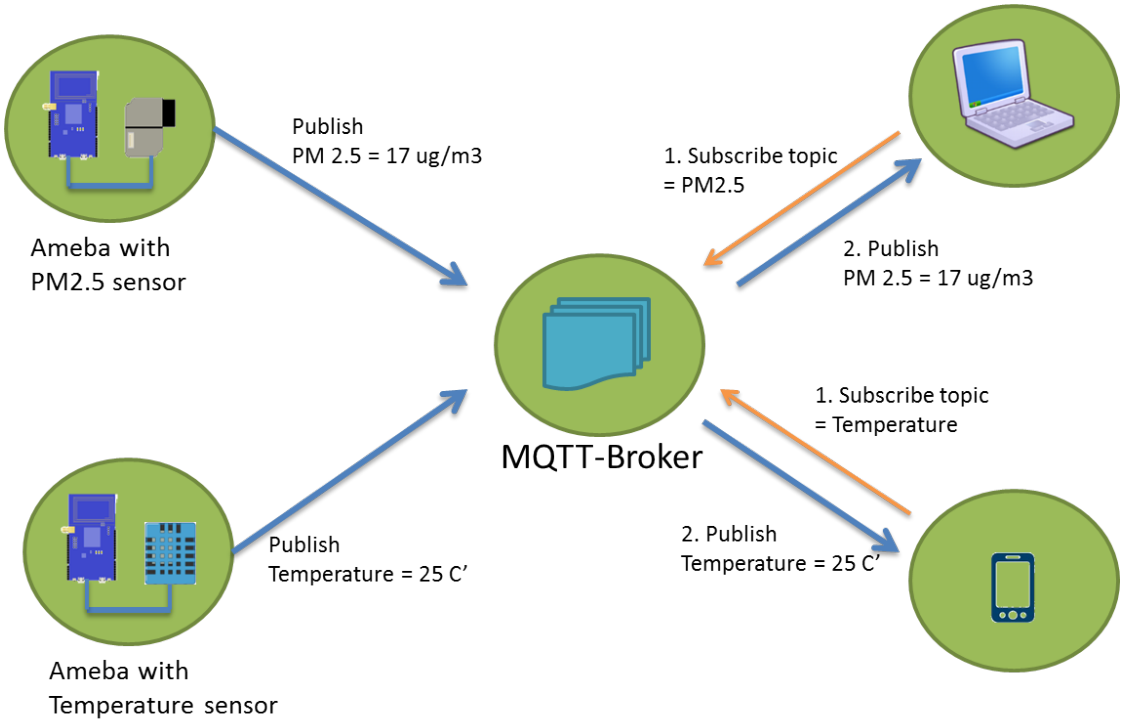
and register to the mDNS service.

```
MDNS.registerService(service);
```

MQTT - Set up MQTT Client to Communicate with Broker

Intro to MQTT

MQTT (Message Queuing Telemetry Transport) is a protocol proposed by IBM and Eurotech. The introduction in MQTT Official Website: MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. We can say MQTT is a protocol designed for IoT. MQTT is based on TCP/IP and transmits/receives data via publish/subscribe. Please refer to the figure below:



In the operation of MQTT, there are several roles:

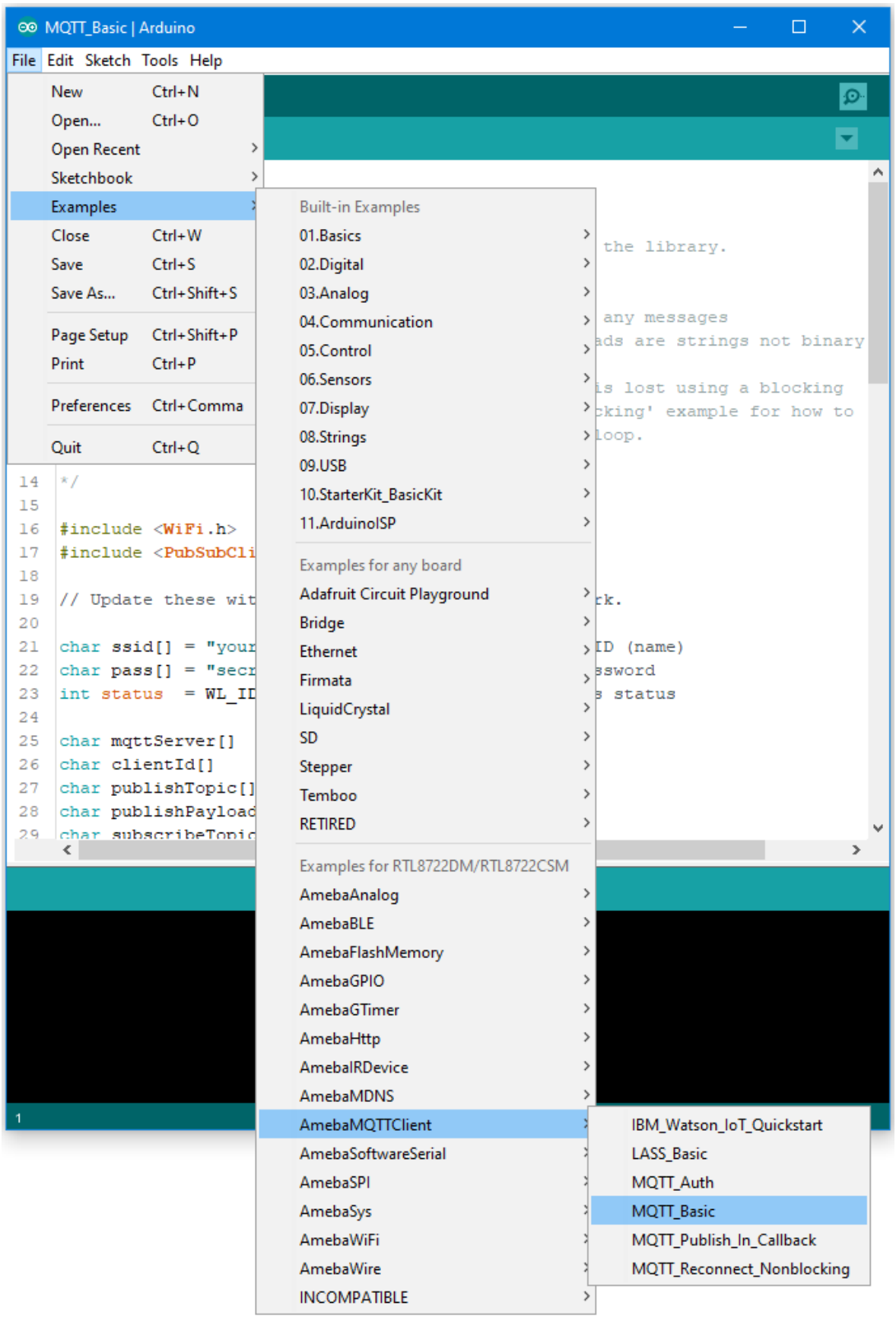
- **Publisher**: Usually publishers are the devices equipped with sensors (ex. Ameba). Publishers uploads the data of the sensors to MQTT-Broker, which serves as a database with MQTT service.
- **Subscriber**: Subscribers are referred to the devices which receive and observe messages, such as a laptop or a mobile phone.
- **Topic**: Topic is used to categorized the messages, for example the topic of a message can be "PM2.5" or "Temperature". Subscribers can choose messages of which topics they want to receive.

Preparation

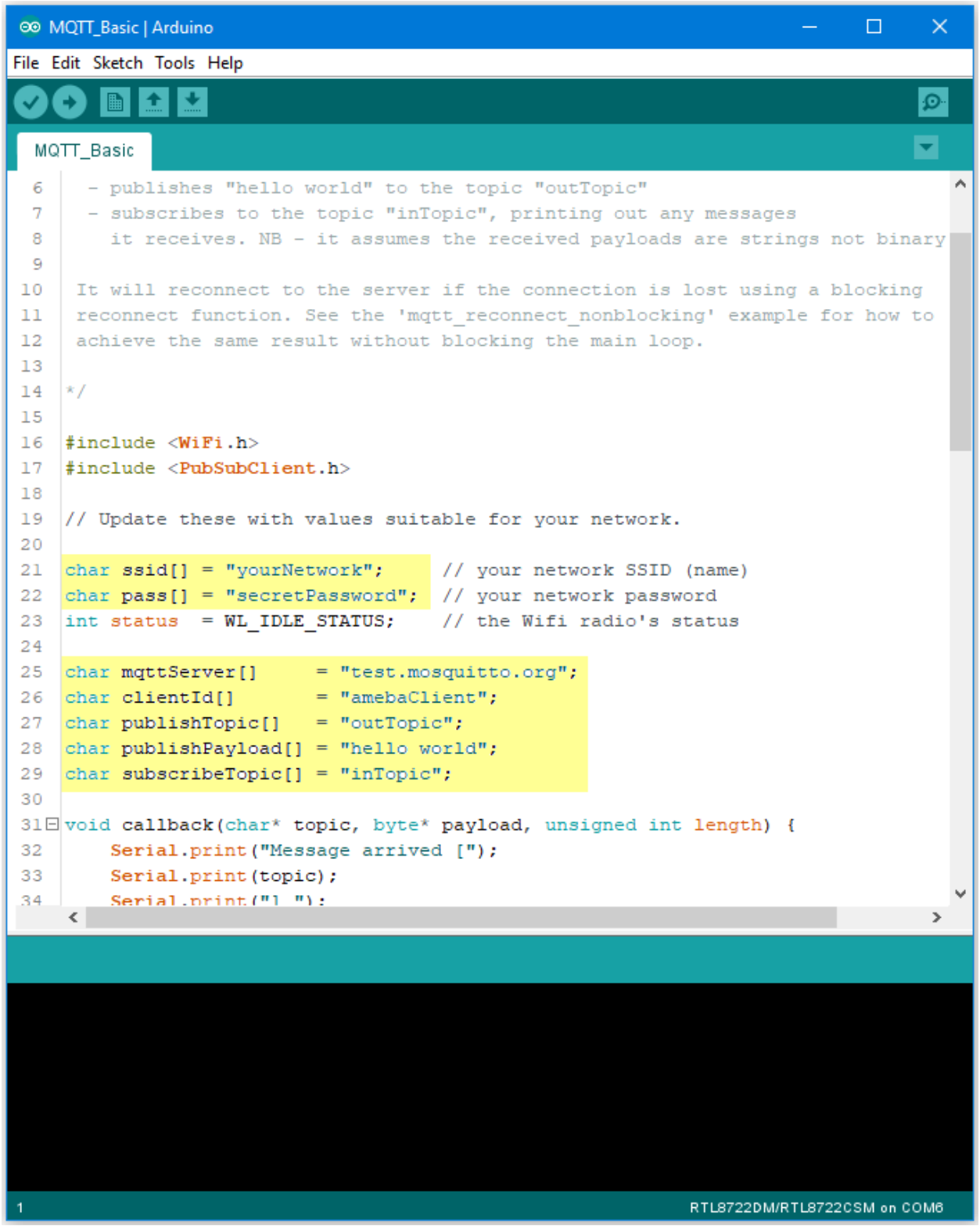
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to MQTT-Broker. Then send messages as publisher and receive messages from MQTT-Broker as subscriber.
Open the MQTT example "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`MQTT_Basic`"


MQTT_Basic | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking

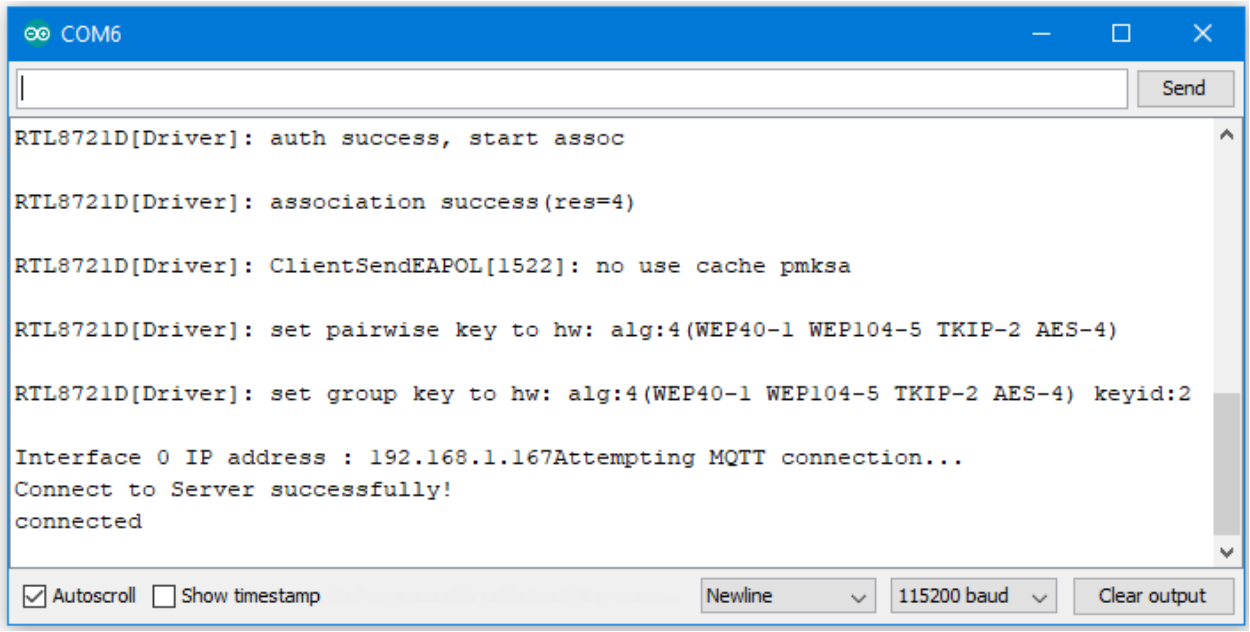
Please modify some WiFi-related parameters.
And some information related to MQTT:



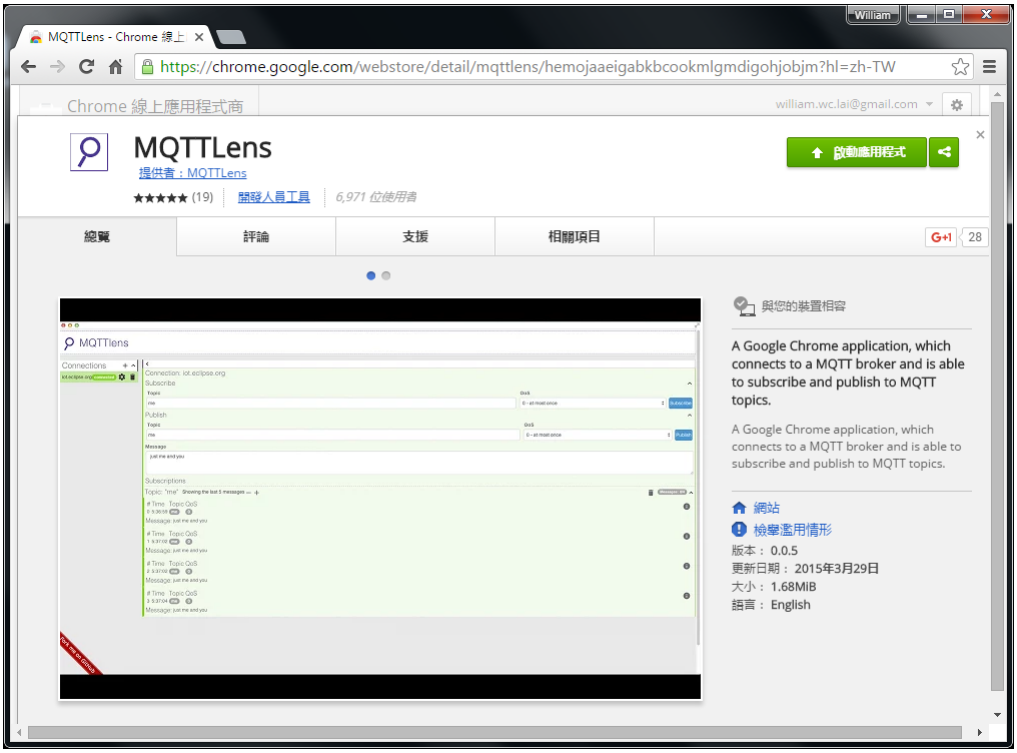
The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.

- “clientId” is an identifier for MQTT-Broker to identify the connected device.
- “publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
- “publishPayload” is the content to be published.
- “subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor



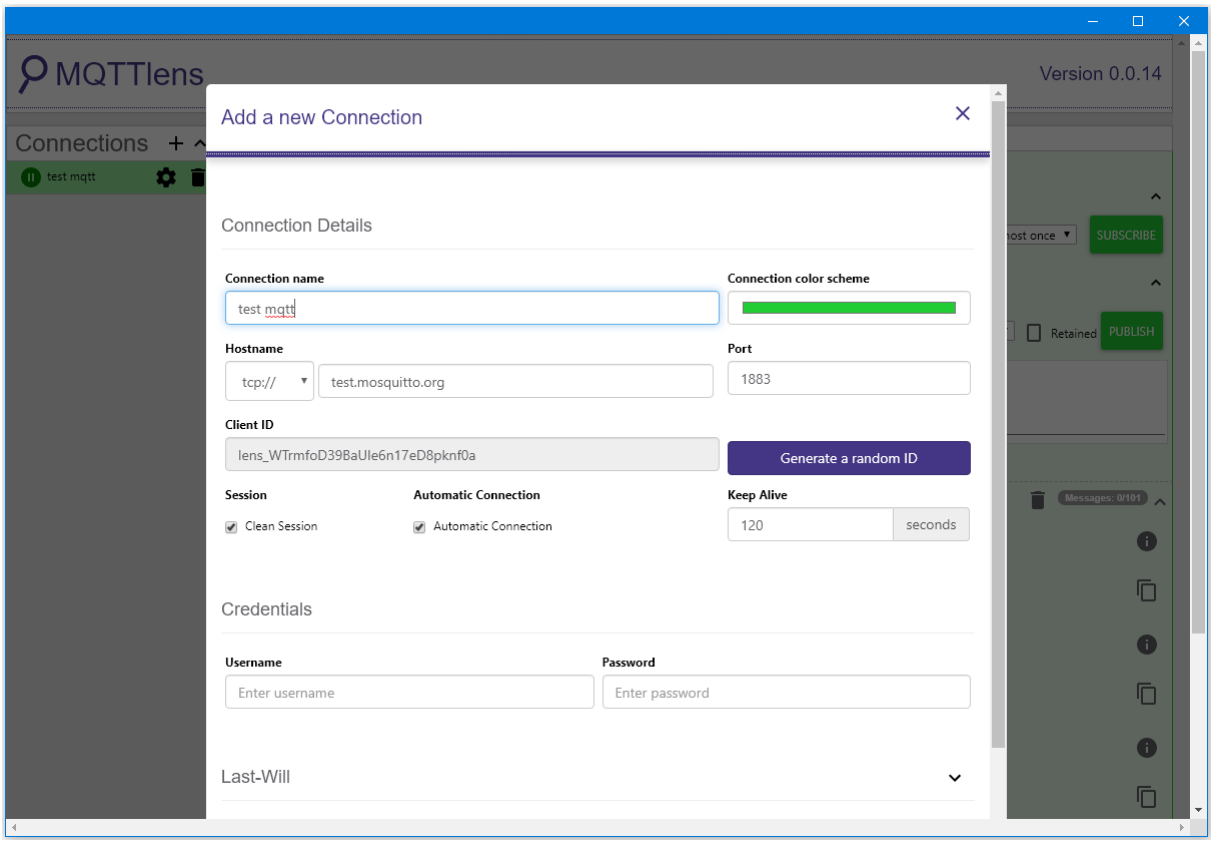
After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”.
To see the message, we need another MQTT client.
Here we use a chrome plugin “MQTTLens” to be the MQTT client. You can find it in google webstore.



Install and open the MQTTLens, click “+” next to “Connection” on the left, and fill in the required information

- **Connection Name**: Used to identify the connection, you can choose a name you like.
- **Hostname**: The MQTT-Broker server, here we use “iot.eclipse.org”
- **Client ID**: We use the default randomly generated ID.

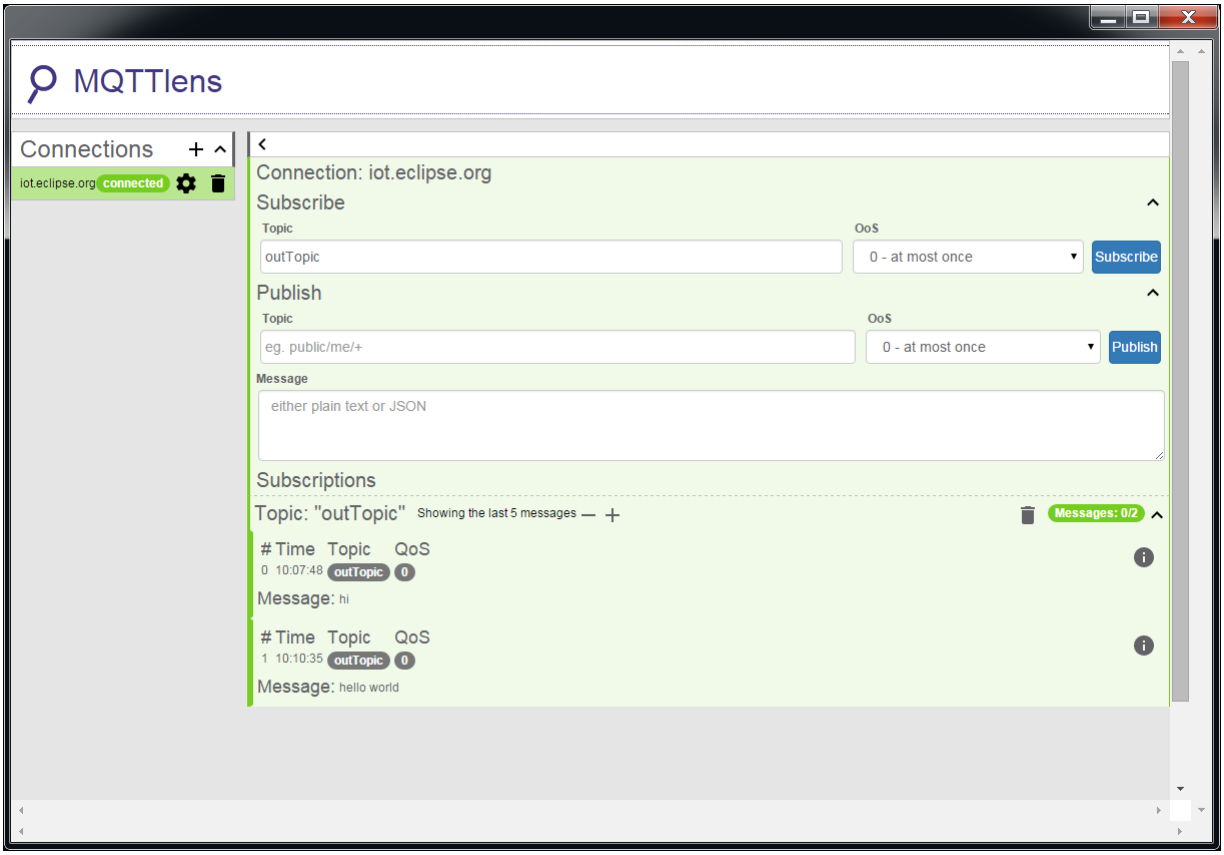
Then click “CREATE CONNECTION”.



Since we have not registered the topic we want to listen to, we would not receive any messages now.

Fill in “outTopic” in the “Topic” field and click “Subscribe”.

Wait for Ameba to send next message (or you can press the reset button). Then you can see the “hello world” message show up.



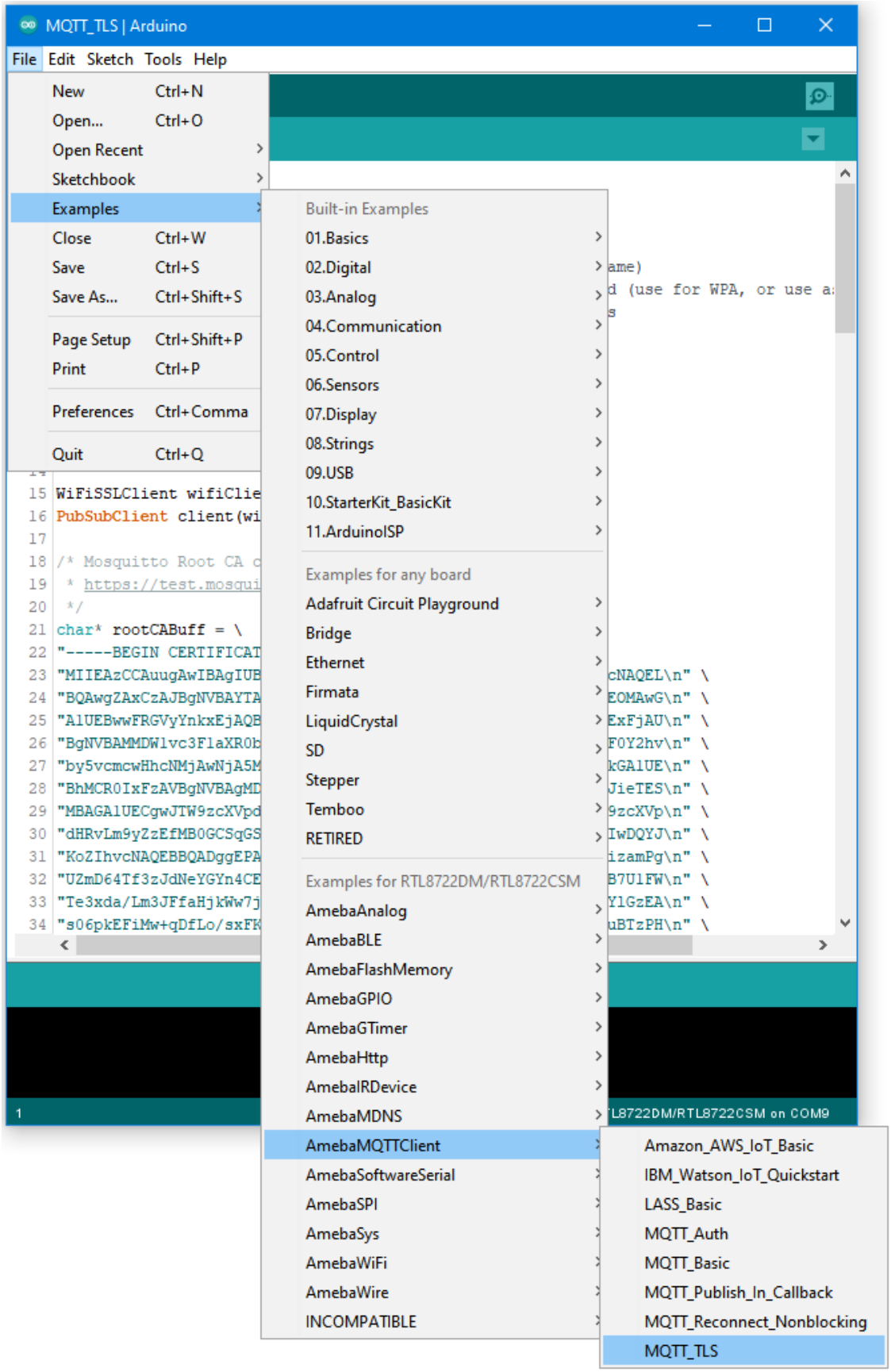
MQTT - Set up MQTT Client over TLS

Preparation

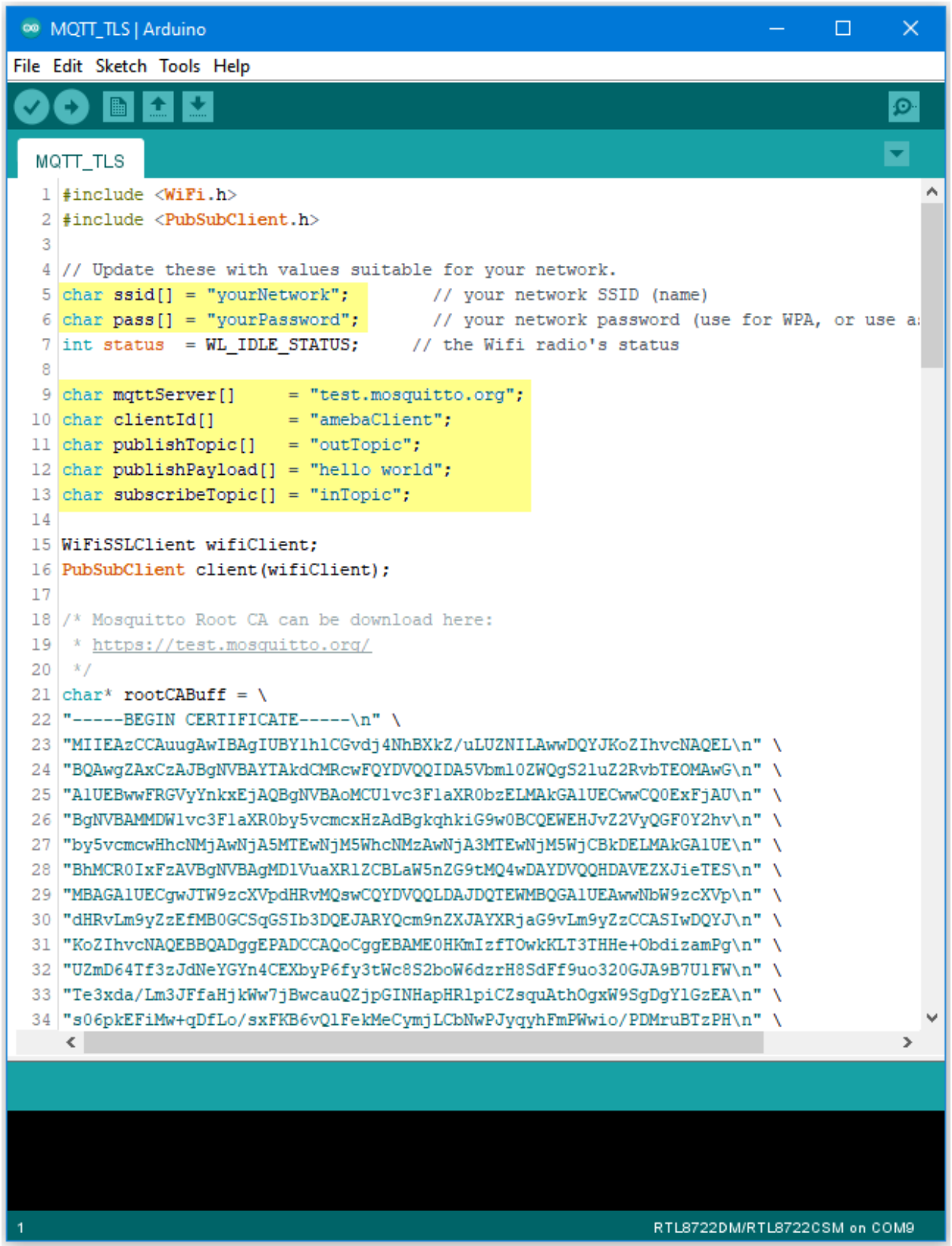
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to a MQTT broker using TLS authentication. Then send messages as a publisher and receive messages from as a subscriber. Open the MQTT example "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`MQTT_TLS`"

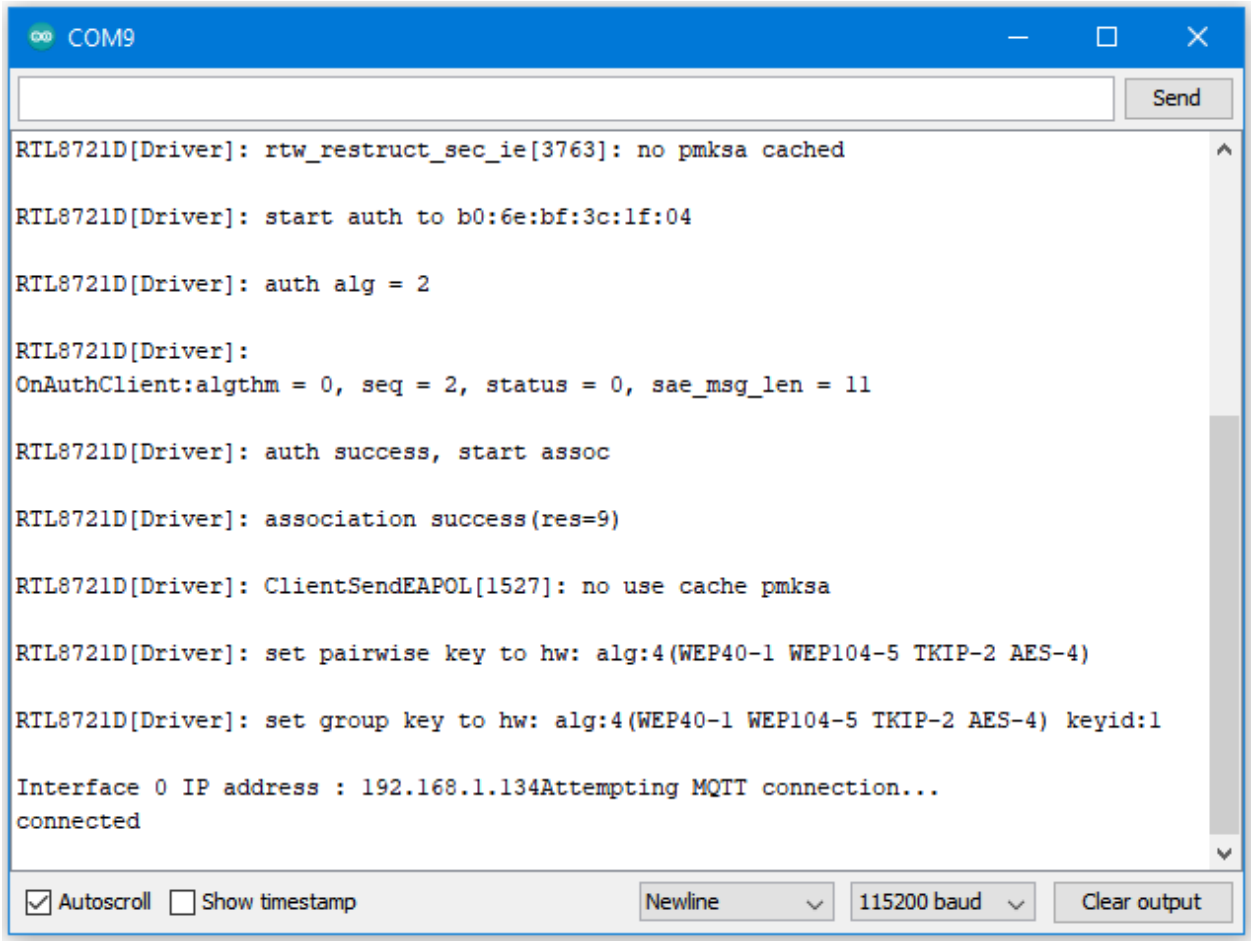

MQTT_TLS | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
Amazon_AWS_IoT_Basic
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking
MQTT_TLS

Please modify the WiFi-related parameters to connect to your WiFi network.
Modify the MQTT parameters to fit your application:



The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.
“clientId” is an identifier for MQTT-Broker to identify the connected device.
“publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
“publishPayload” is the content to be published.
“subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

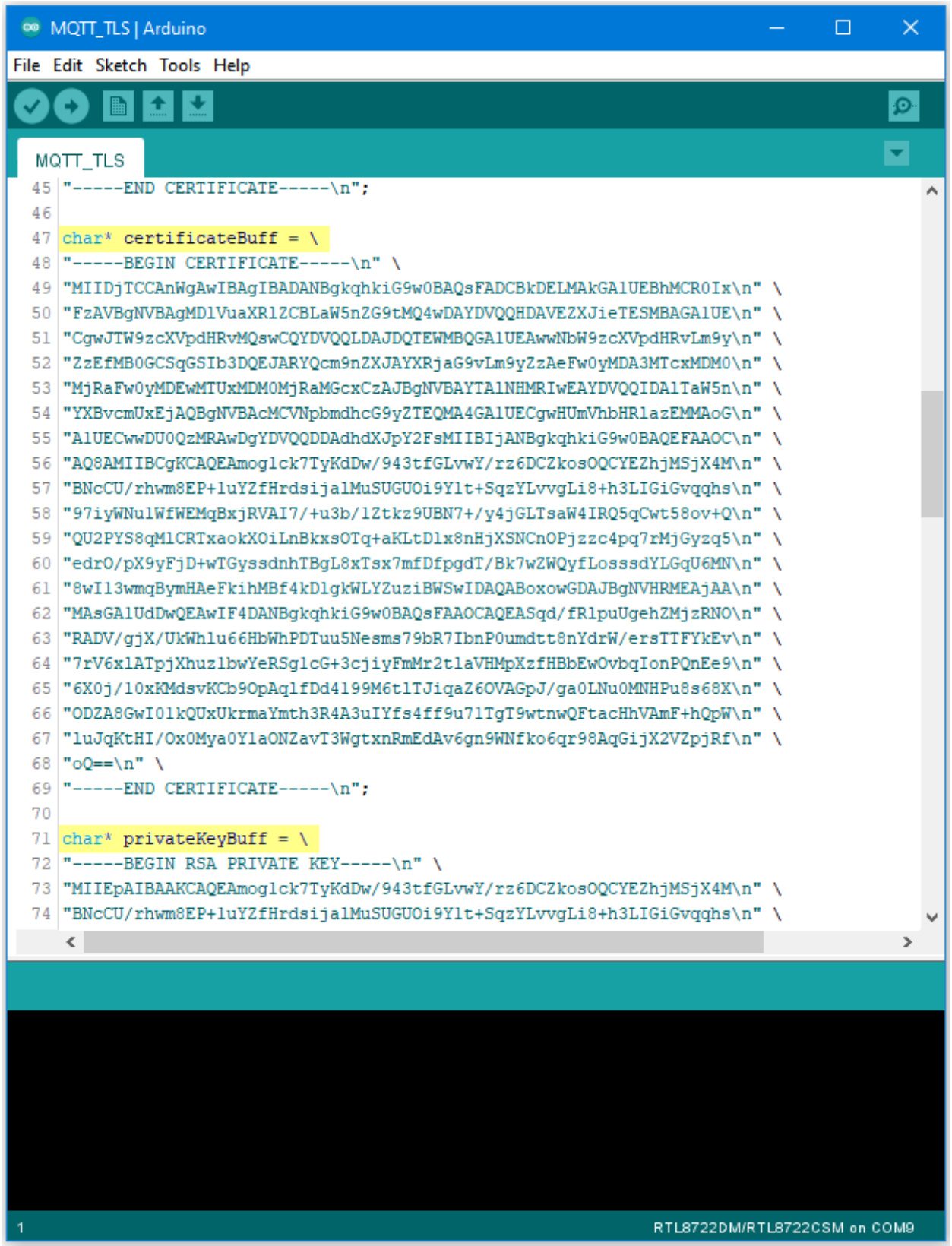
Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor

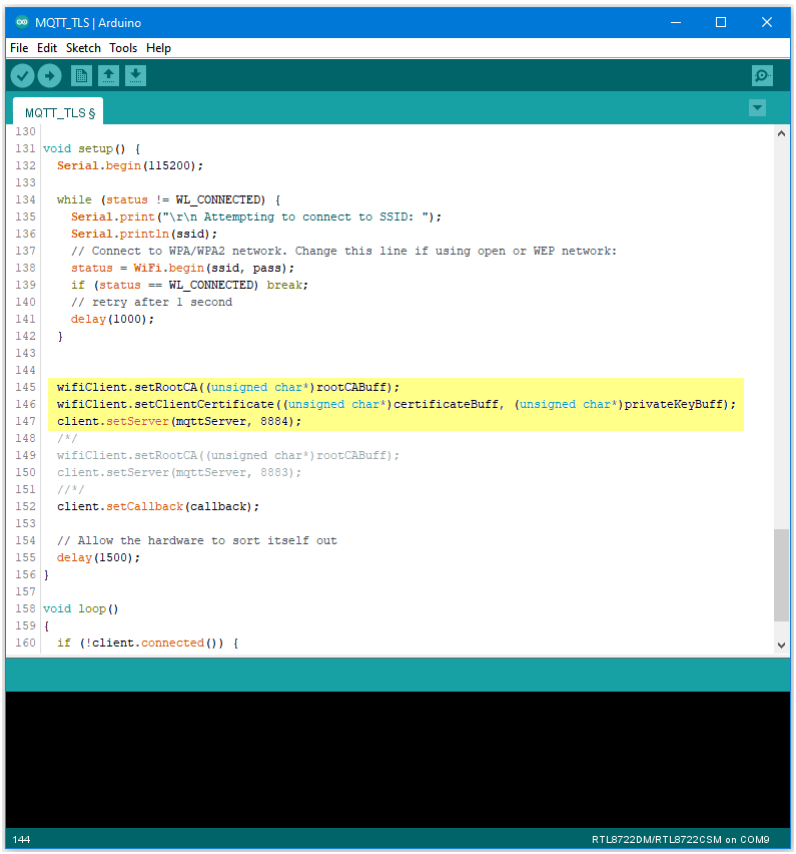


After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”. To see the message, use another MQTT client. Refer to the MQTT_Basic example guide on how to setup a PC-based MQTT client.

If you wish to use TLS client authentication in addition to server authentication, you will need to generate an OpenSSL private key and obtain a signed certificate from the server. For testing purposes, signed certificates can be obtained from test.mosquitto.org by following the guide at https://test.mosquitto.org/ssl/.

Replace the character strings “certificateBuff” and “privateKeyBuff” with your signed certificate and OpenSSL private key, ensuring that they are formatted the same way as the shown in the example code. Also uncomment the highlighted code to enable client authentication, and to change the MQTT port number.


MQTT_TLS | Arduino
File Edit Sketch Tools Help
MQTT_TLS
"-----END CERTIFICATE-----\n";
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDjTCCAnWgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix\n" \
"FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE\n" \
"CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y\n" \
"ZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzAeFw0yMDA3MTcxMDM0\n" \
"MjRaFw0yMDEwMTUxMDM0MjRaMGcxCzAJBgNVBAYTAlNHMRIwEAYDVQQIDAlTaW5n\n" \
"YXBvcmUxEjAQBgNVBAcMCVNpbmdhcG9yZTEQMA4GA1UECgwHUmVhbHRlazEMMAoG\n" \
"A1UECwwDU0QzMRAwDgYDVQQDDAdhdXJpY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOC\n" \
"AQ8AMIIBCgKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
"97iyWNulWfWEMqBxjRVAI7/+u3b/lZtkz9UBN7+/y4jGLTsaW4IRQ5qCwt58ov+Q\n" \
"QU2PYS8qMlCRTxaokXOiLnBkxsOTq+aKLtDlx8nHjXSNCnOPjzzc4pq7rMjGyzq5\n" \
"edrO/pX9yFjD+wTGyssdnhTBgL8xTsx7mfDfpgdT/Bk7wZWQyfLosssdYLGqU6MN\n" \
"8wIl3wmqBymHAeFkihMBf4kDlgkWLYZuziBWSwIDAQABoxowGDAJBgNVHRMEAjAA\n" \
"MAsGA1UdDwQEAwIF4DANBgkqhkiG9w0BAQsFAAOCAQEASqd/fRlpuUgehZMjzRNO\n" \
"RADV/gjX/UkWhlu66HbWhPDTuu5Nesms79bR7IbnP0umdtt8nYdrW/ersTTFYkEv\n" \
"7rV6xlATpjXhuzlbwYeRSglcG+3cjiyFmMr2tlaVHMpXzfHBbEwOvbqIonPQnEe9\n" \
"6X0j/l0xKMdsvKCb9OpAqlfDd4199M6tlTJiqaZ6OVAGpJ/ga0LNu0MNHPu8s68X\n" \
"ODZA8GwI0lkQUxUkrmaYmth3R4A3uIYfs4ff9u7lTgT9wtnwQFtacHhVAmF+hQpW\n" \
"luJqKtHI/Ox0Mya0YlaONZavT3WgtxnRmEdAv6gn9WNfko6qr98AqGijX2VZpjRf\n" \
"oQ==\n" \
"-----END CERTIFICATE-----\n";
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEpAIBAAKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
1
RTL8722DM/RTL8722CSM on COM9



MQTT - Use Amazon AWS IoT Shadow Service

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

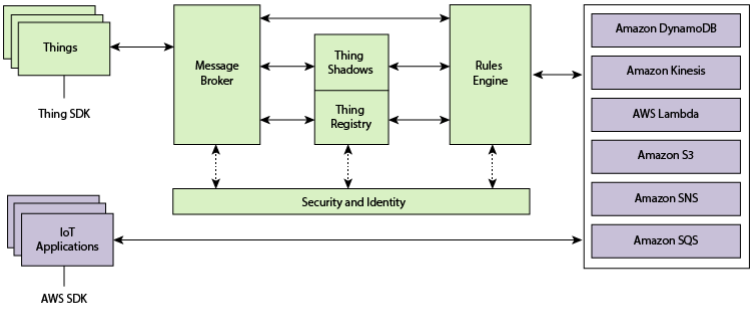
Example

Introduction

Amazon AWS IoT is a cloud IoT service platform:

Amazon AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline. (https://aws.amazon.com/iot/how-it-works/)

The service architecture of AWS IoT:

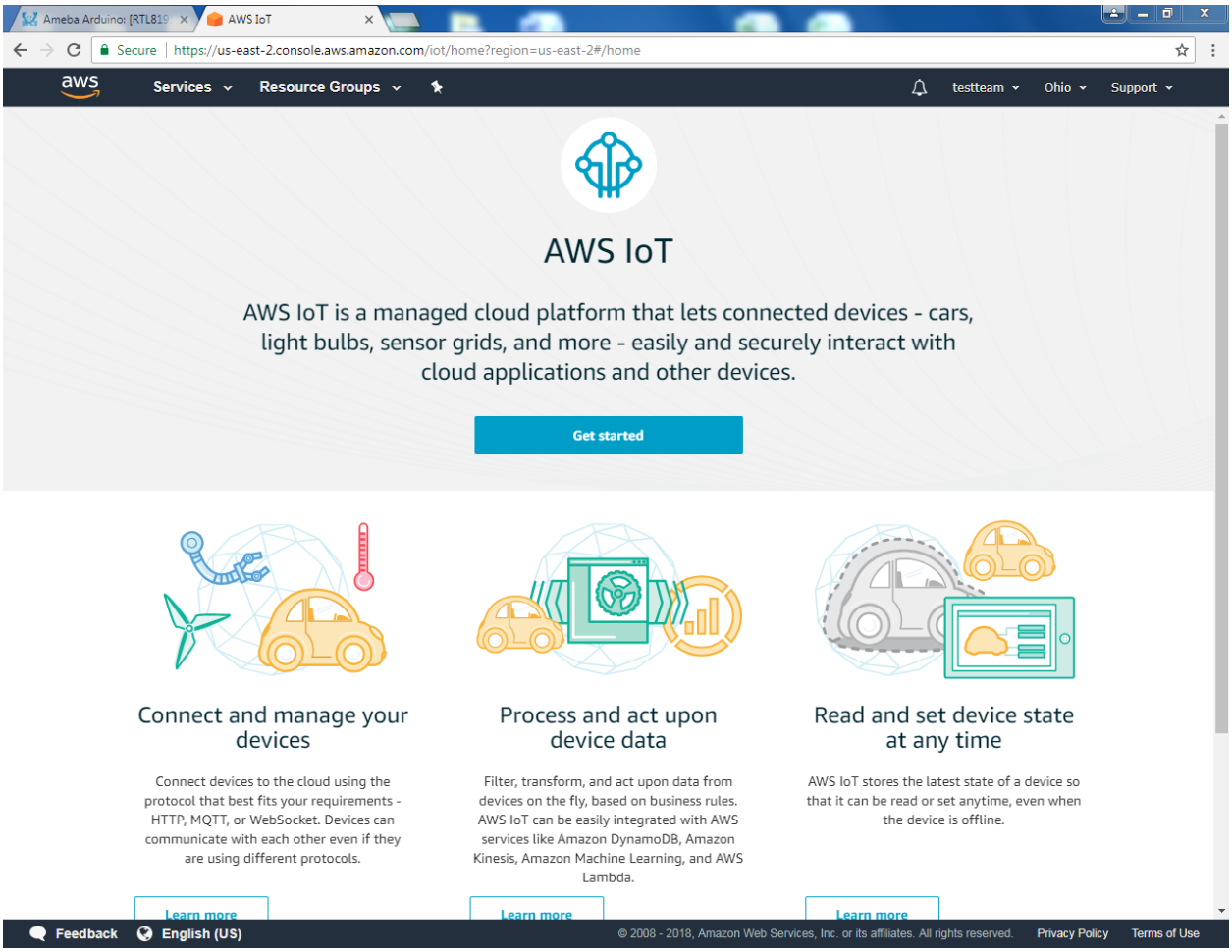


(Picture from http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html)
In the architecture, Ameba belongs to the upper-left “Things” block. A TLS secure channel will be established between “Things” and the MQTT Message Broker. Afterwards, “Things” and “Message Broker” communicate using MQTT Protocol via this secure channel. Behind the “Message Broker”, the “Thing Shadows” keeps messages temporarily when Ameba is offline, and sends the control message to Ameba next time it is connected. The “Rules Engine” allows you to place restrictions to the behavior of Things or to connect Things to other services of Amazon.

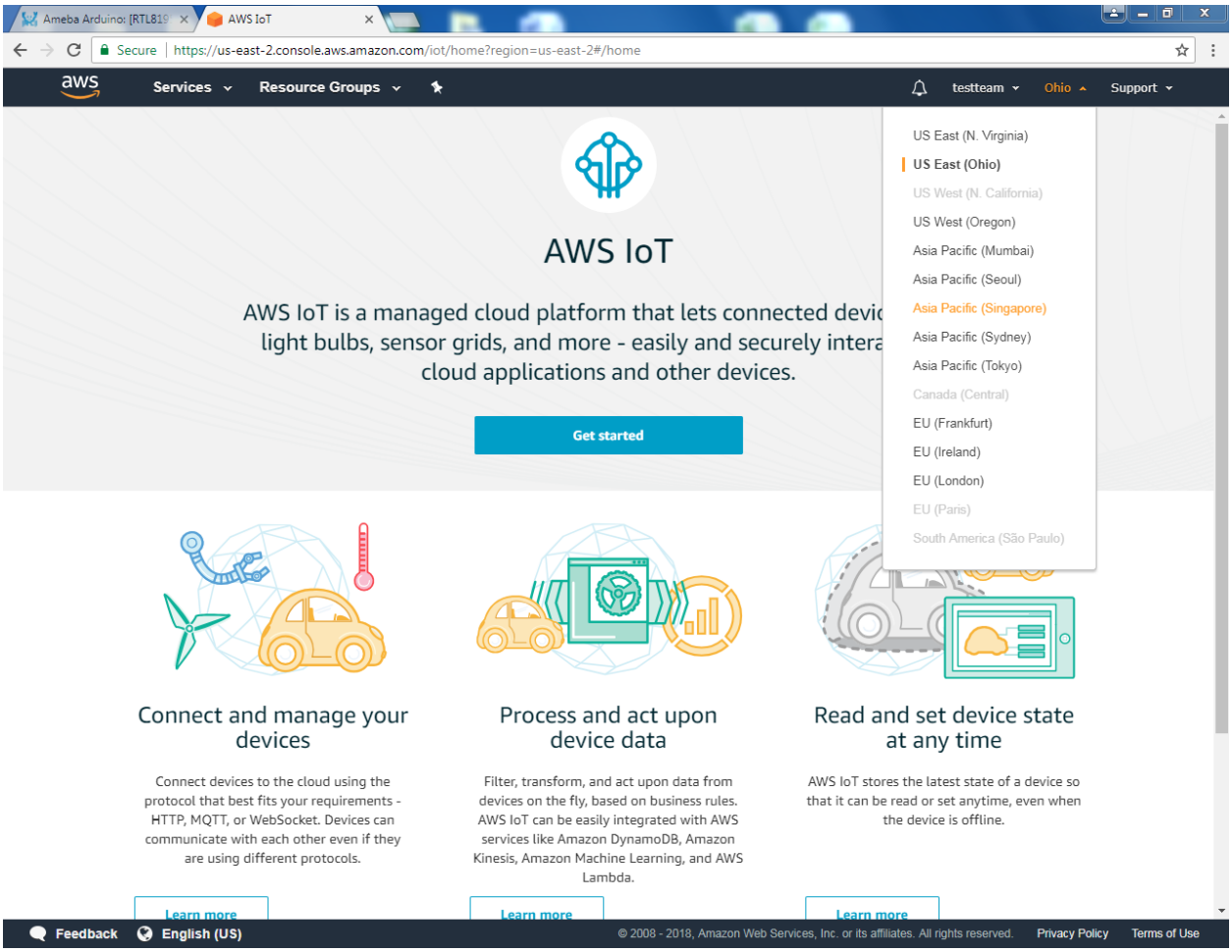
AWS Management Console

First, create an account and sign up for AWS IoT service:https://aws.amazon.com/
Afterwards, log in to the Amazon Management Console and click “IoT Core” found under services -> Internet of Things.

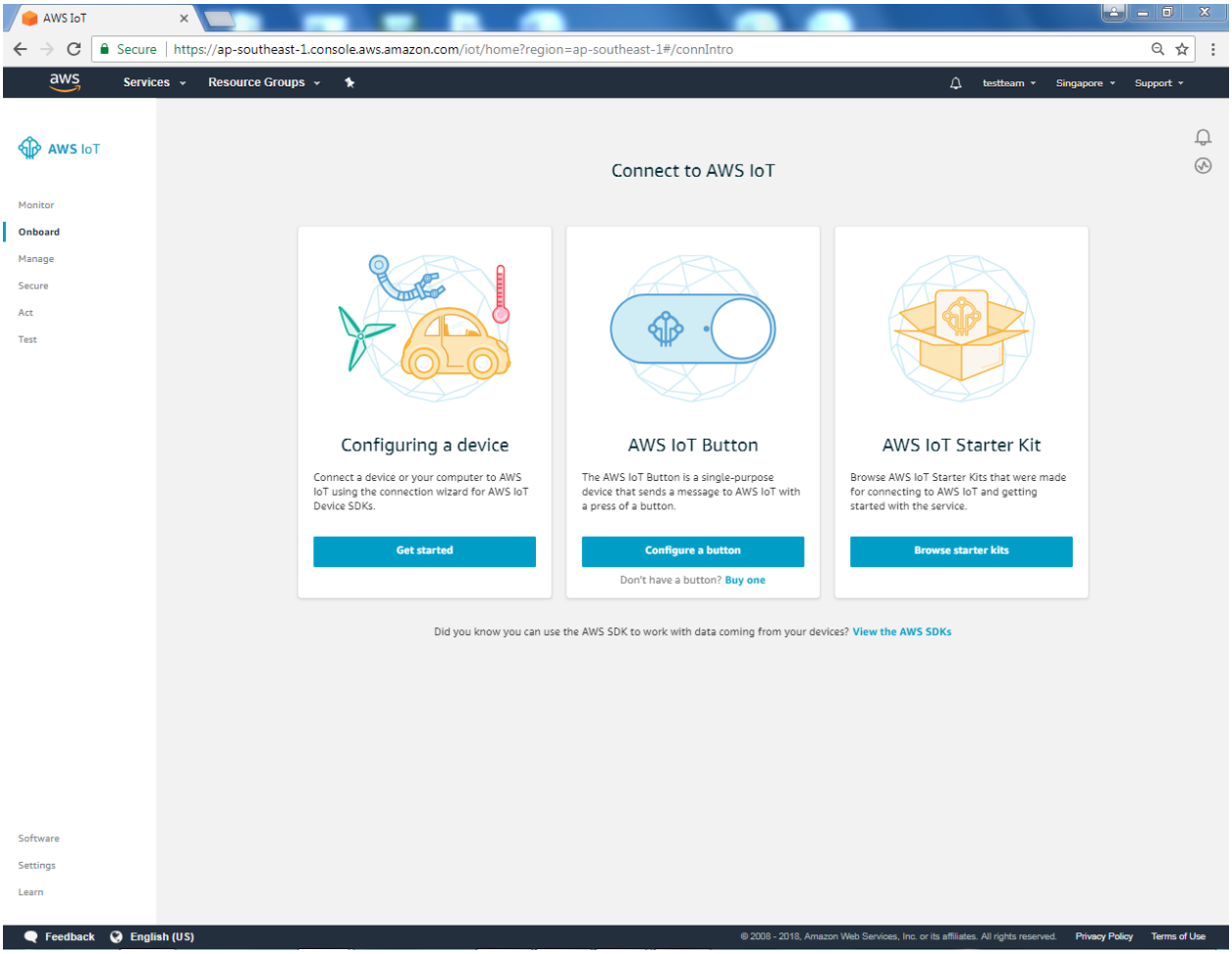
Then you will enter the home page of AWS IoT. To offer the best service quality, Amazon offers servers in different regions for users to choose from.
Click the region dropdown menu at the upper-right:



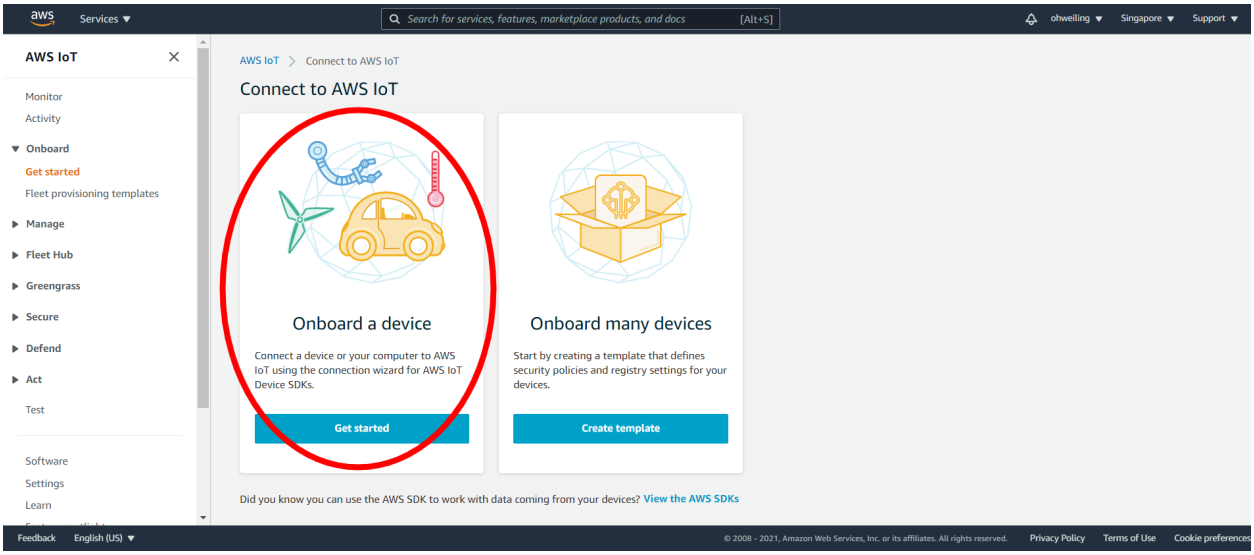
Choose a nearby region.



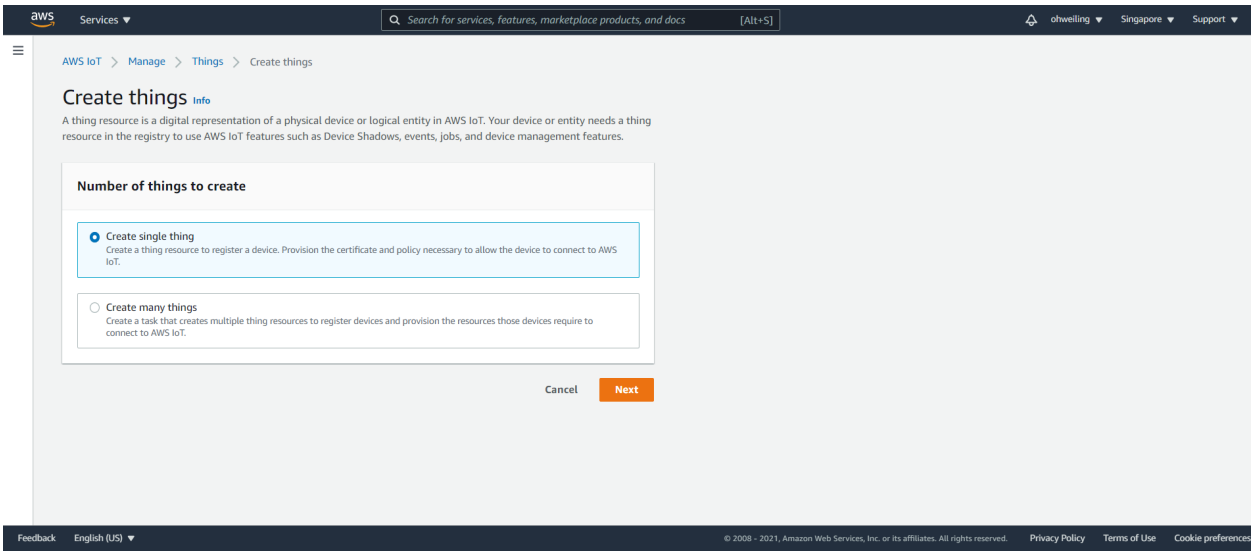
Then from Services, go to Onboard then Get Started.



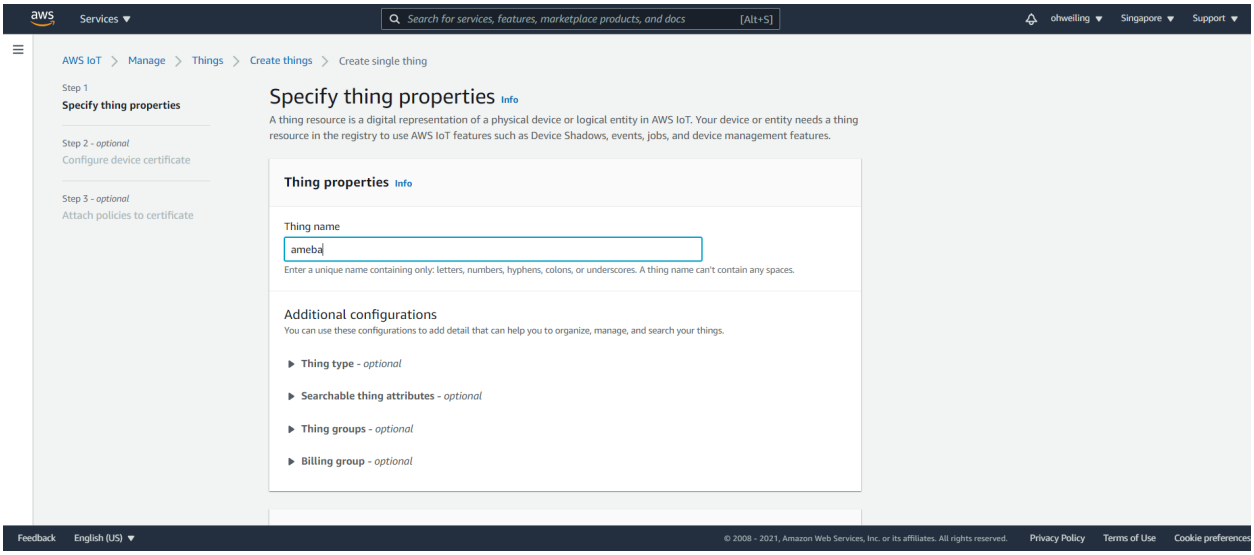
Enter the main page of AWS IoT. Under the Onboard a device, click Get started.



Click Create single thing

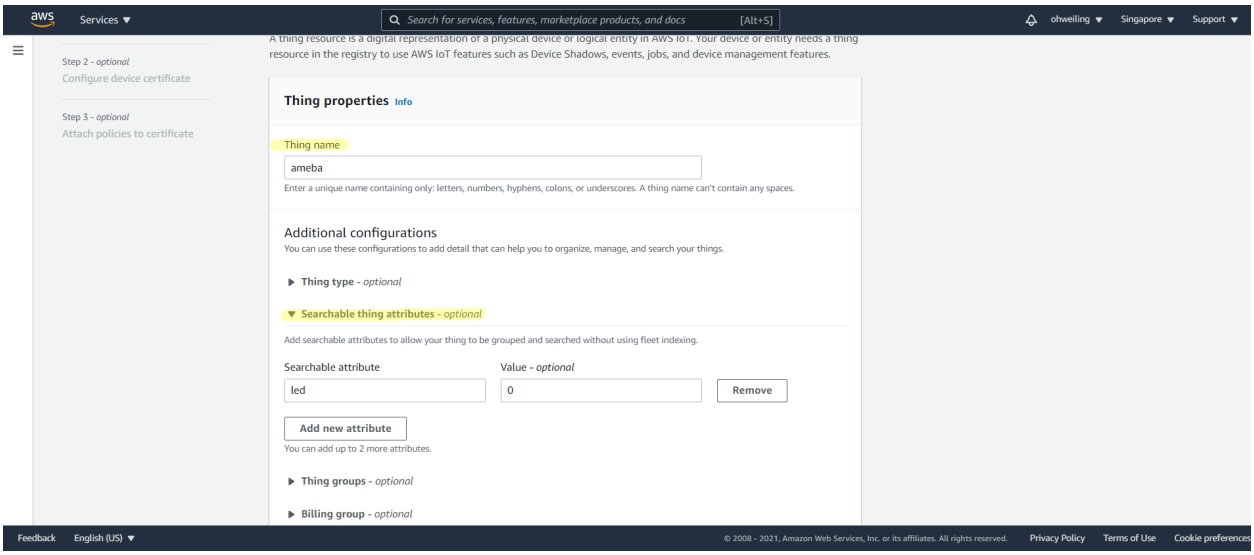


Fill in “ameba” on the name field. Attributes represent the status of Ameba.

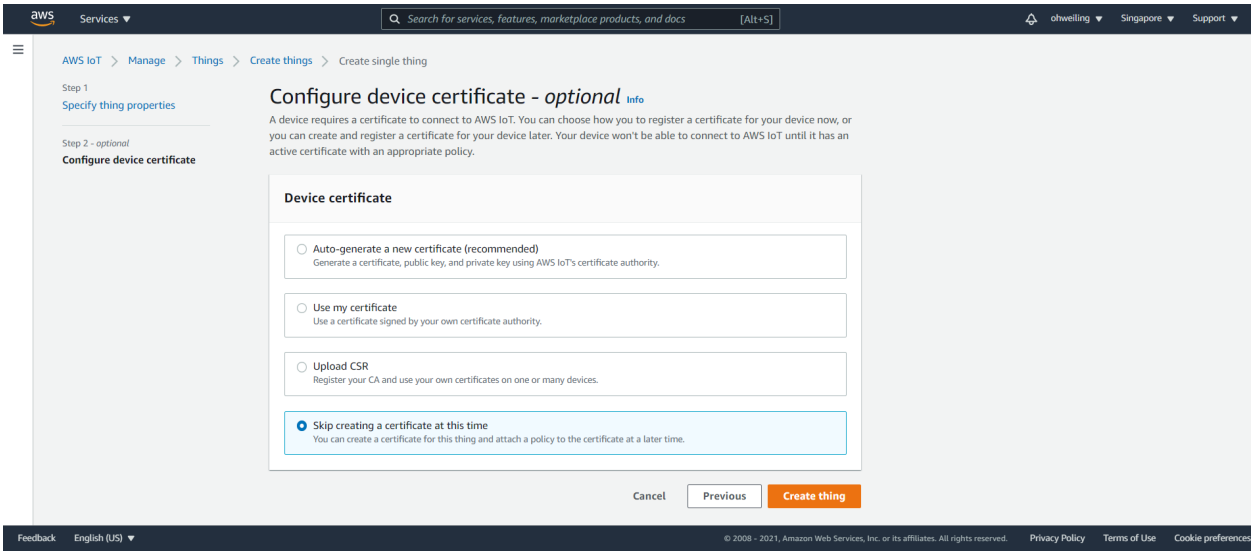


Under the searchable thing attributes. The value of the attributes can be updated directly by Ameba or by the control side and control side can request Ameba to set the attribute to desired value.

Here we add an attribute named “led” with value “0” and click “Next”.



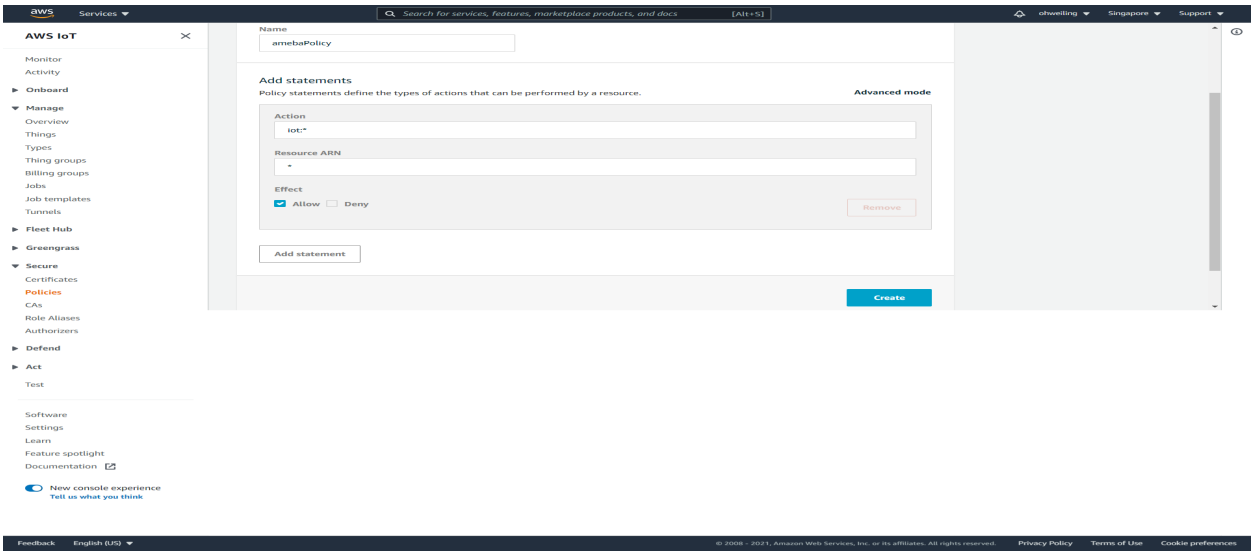
Click Skip creating a certificate at this time and then Create thing



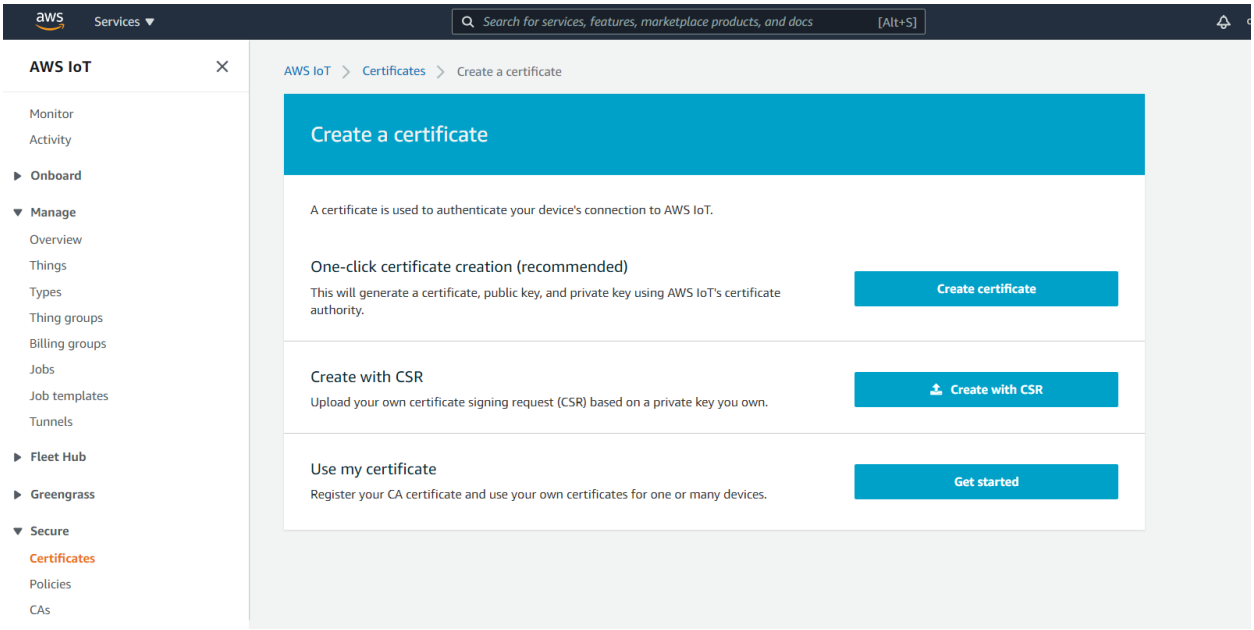
Next, click Policy¸ and create a policy. Policy is used to restrict the functions that a “thing” can do, it can limit the MQTT actions or specific topic that can be performed. Learn more about policy:

http://docs.aws.amazon.com/iot/latest/developerguide/authorization.html

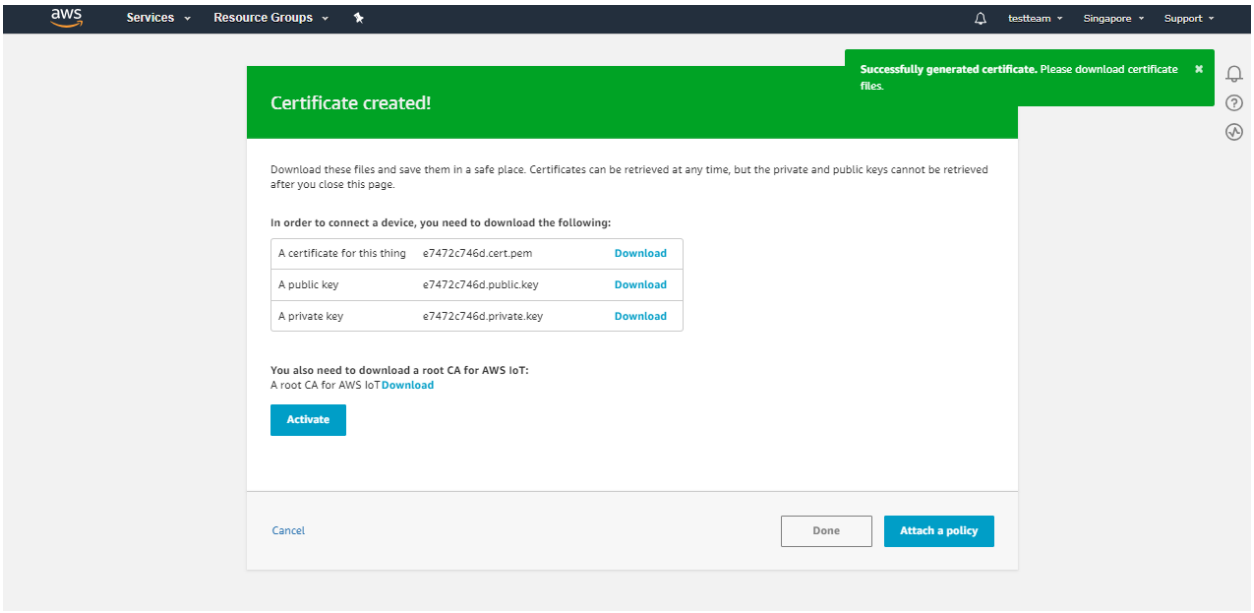
Here we do not place policy on Ameba. Fill in “amebaPolicy” in the Name field, “iot:” *in Action field and* “” in resources field. Then “Allow”. Finally, click “Create”.



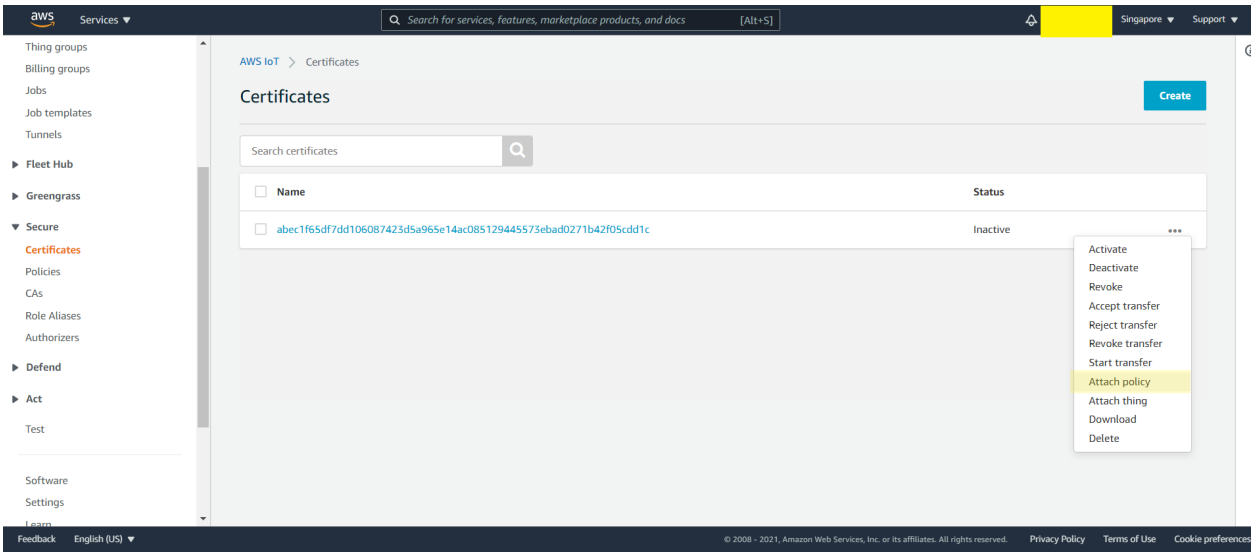
Next, we have to setup the TLS certificate. You can choose to user-defined or generate a certificate by AWS IoT. In this example we click Create Certificate to generate a TLS certificate.



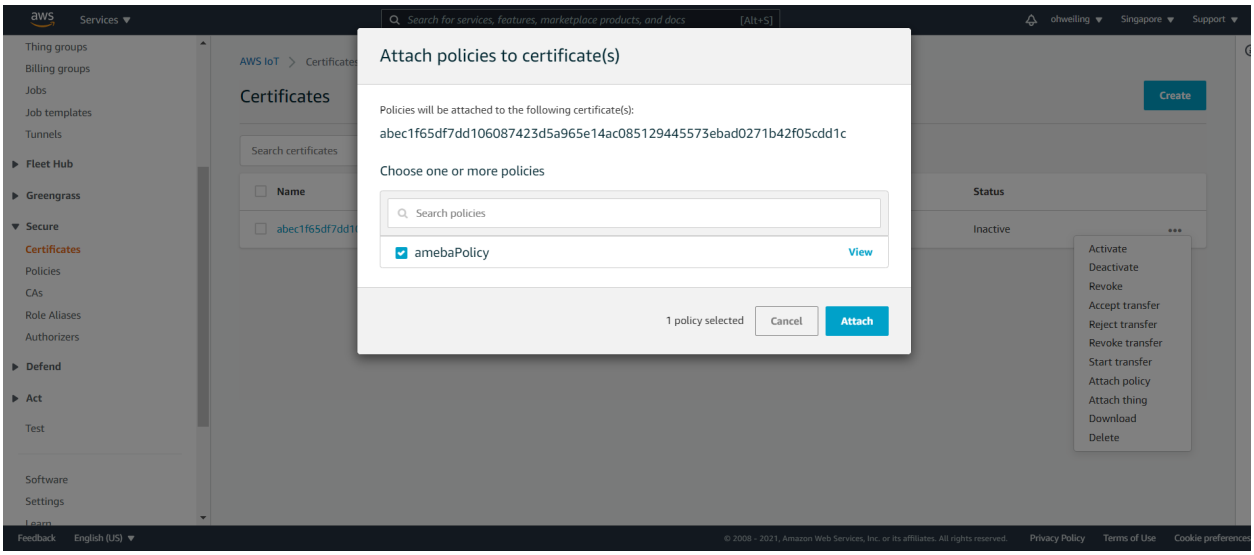
You can see 4 Links. Please download each of the link, “public key”, “private key”, “Certificate” and “rootCA”. After downloading the 4 files, click Done and go back to the certificate main page.



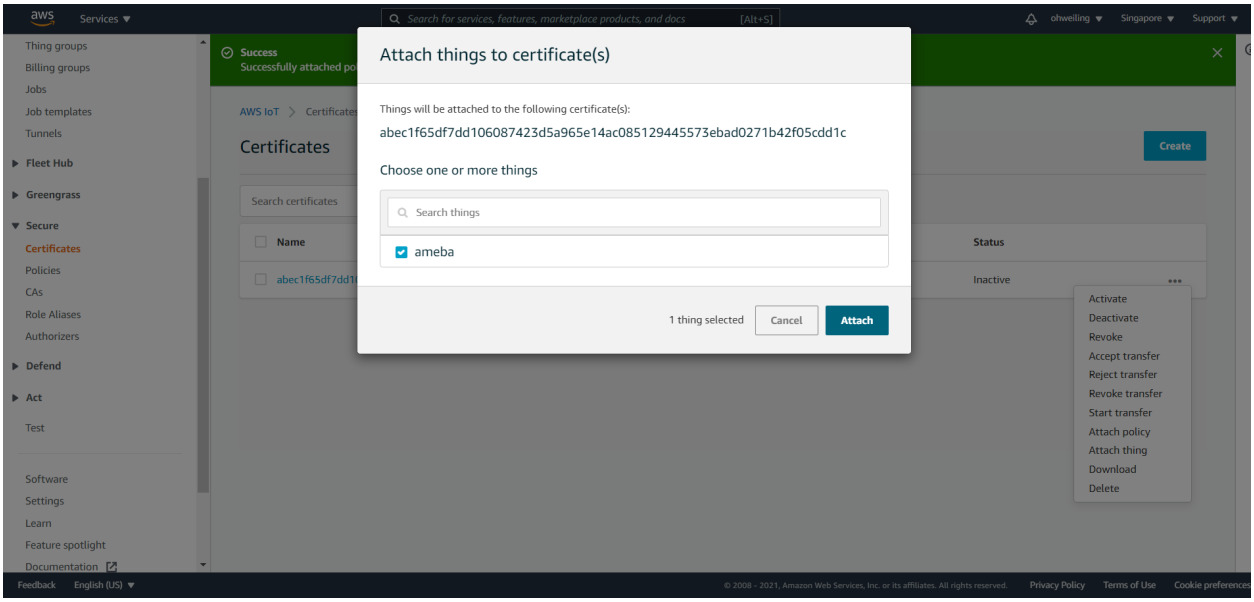
Click Attach a policy in the Actions dropdown menu.



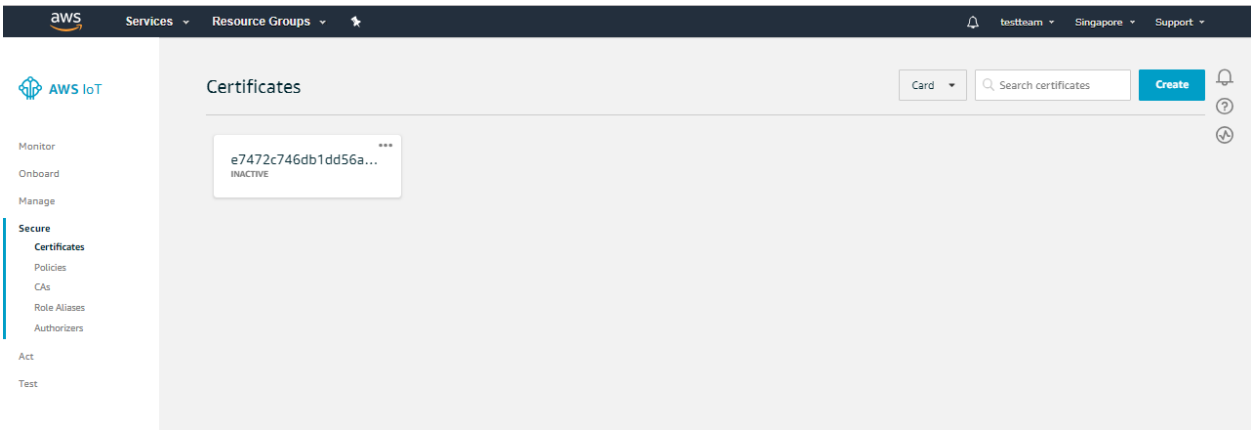
Choose amebaPolicy and click attach.



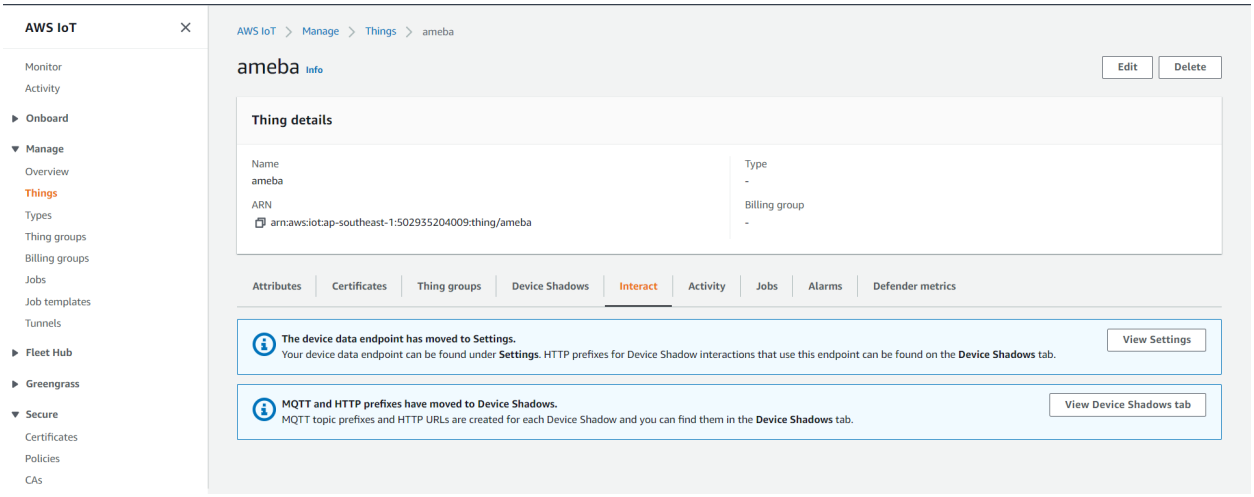
Then go back to the "Actions" drop-down menu at the top right of the certificates homepage, click on "Attach thing", select the thing "ameba" you just created when the window below appears, then click on "Attach"



Go back to certificate main page and click Certificate and click Activate in the Actions drop down menu.



Next, click Manage, and click Things, then click “ameba” the thing we created just now.
Click on Interact and View settings.

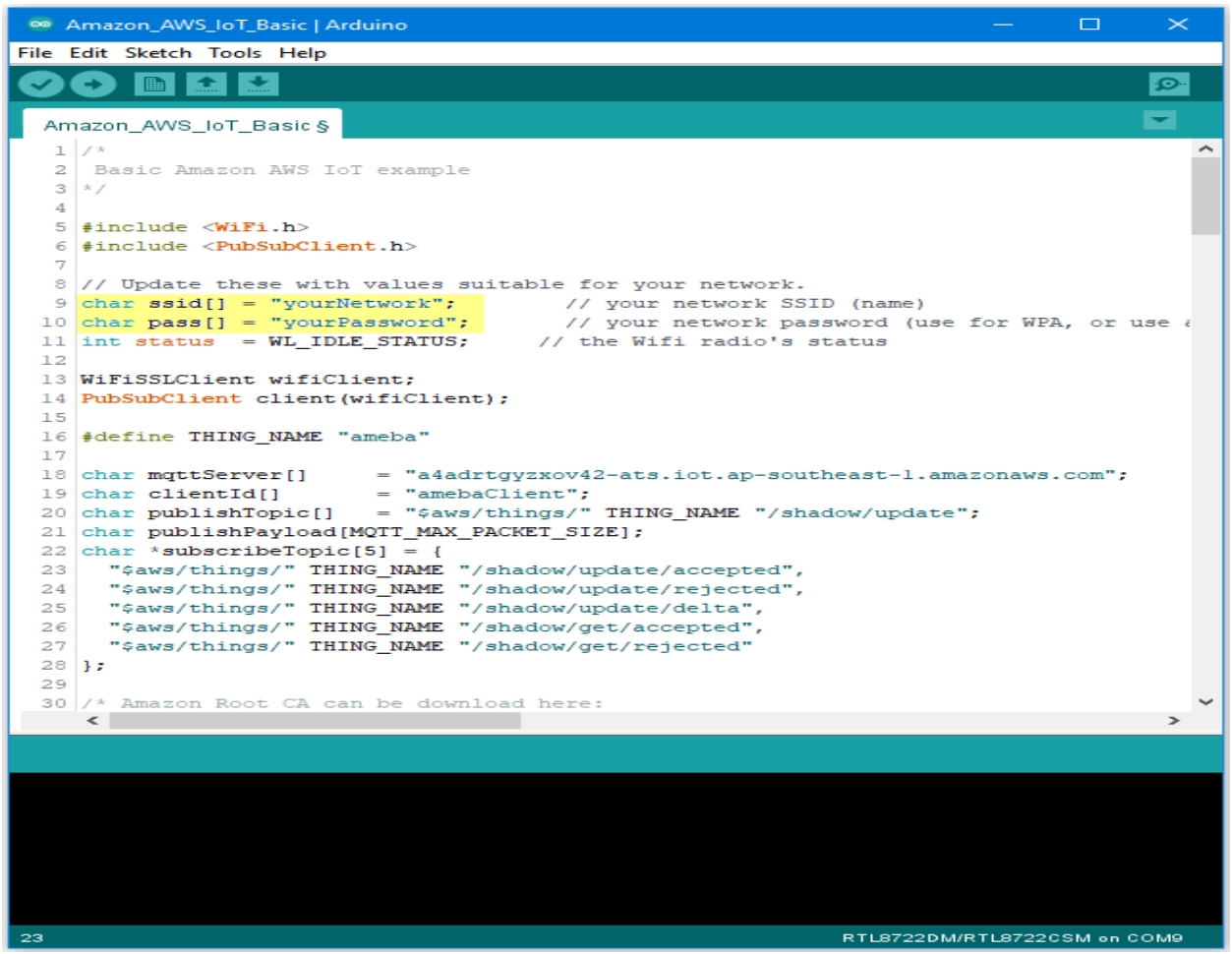


Find out the information of Rest API Endpoint to set Amazon Alexa:

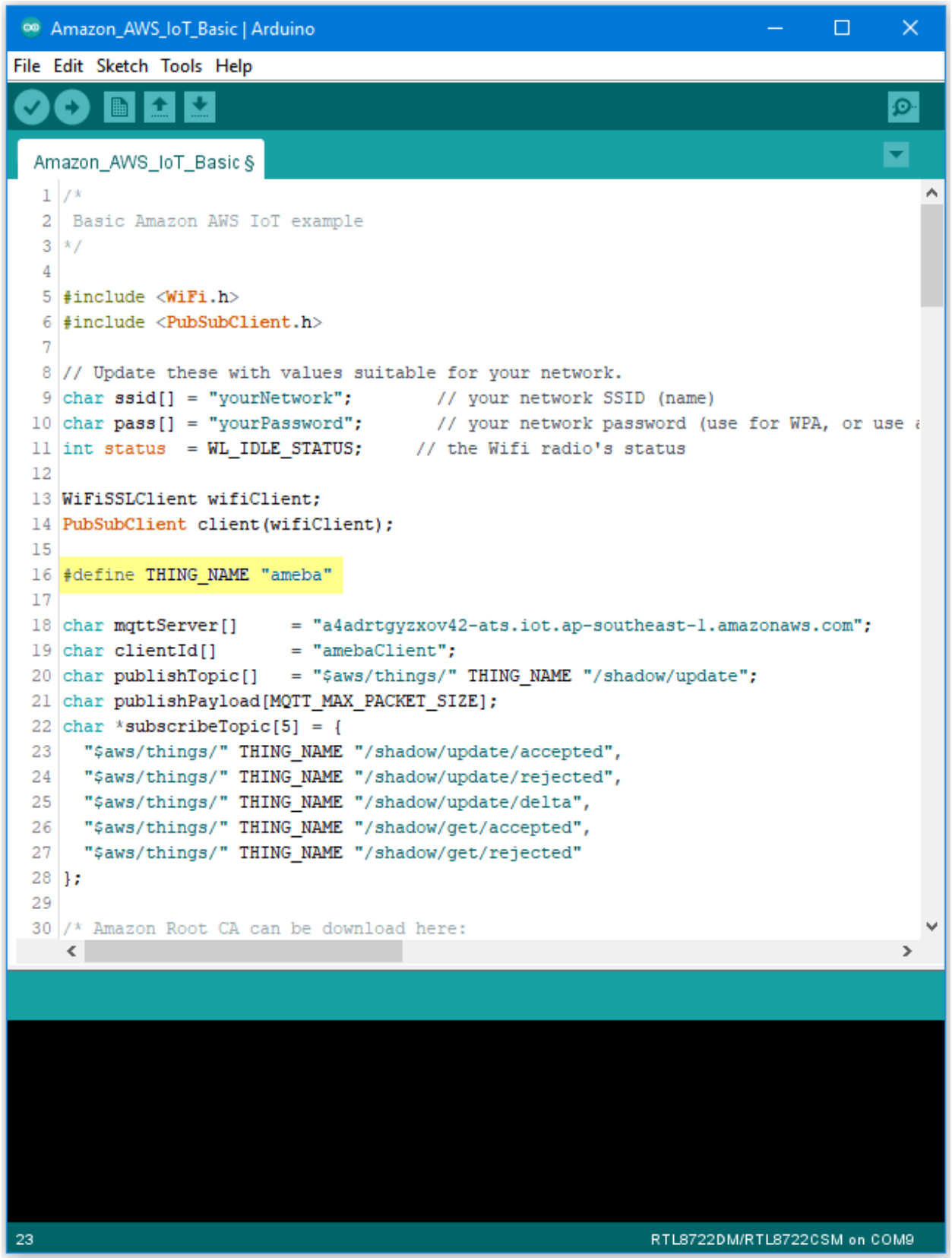
- REST API endpoint: In the value “https://a1a7oo4baosgyy.iot.us-east-1.amazonaws.com/things/ameba/shadow”, the part “a1a7oo4baosgyy.iot.us-east-1.amazonaws.com” is the MQTT Broker server address.
- MQTT topic：The value “$aws/things/ameba/shadow/update” represents the MQTT topic we will use in the AWS IoT Shadow service (if we use MQTT only, without AWS IoT Shadow service, then we can specify other topic name). It is recommended to use “$aws/things/ameba/shadow/update” here.

Ameba setting

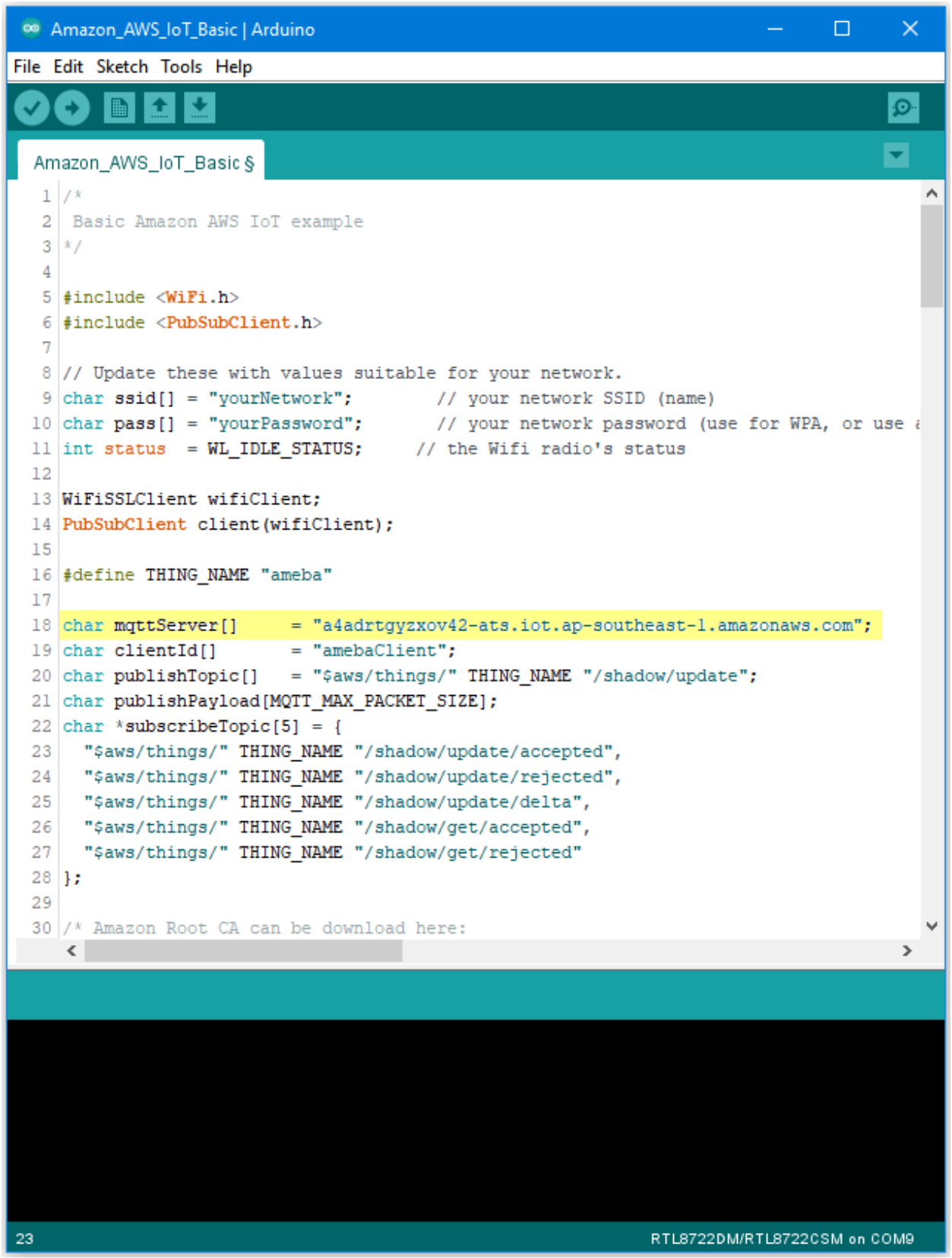
Open “`File` -> “`Examples`” -> “`AmebaMQTTClient`” -> “`Amazon_AWS_IoT_Basic`”
In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



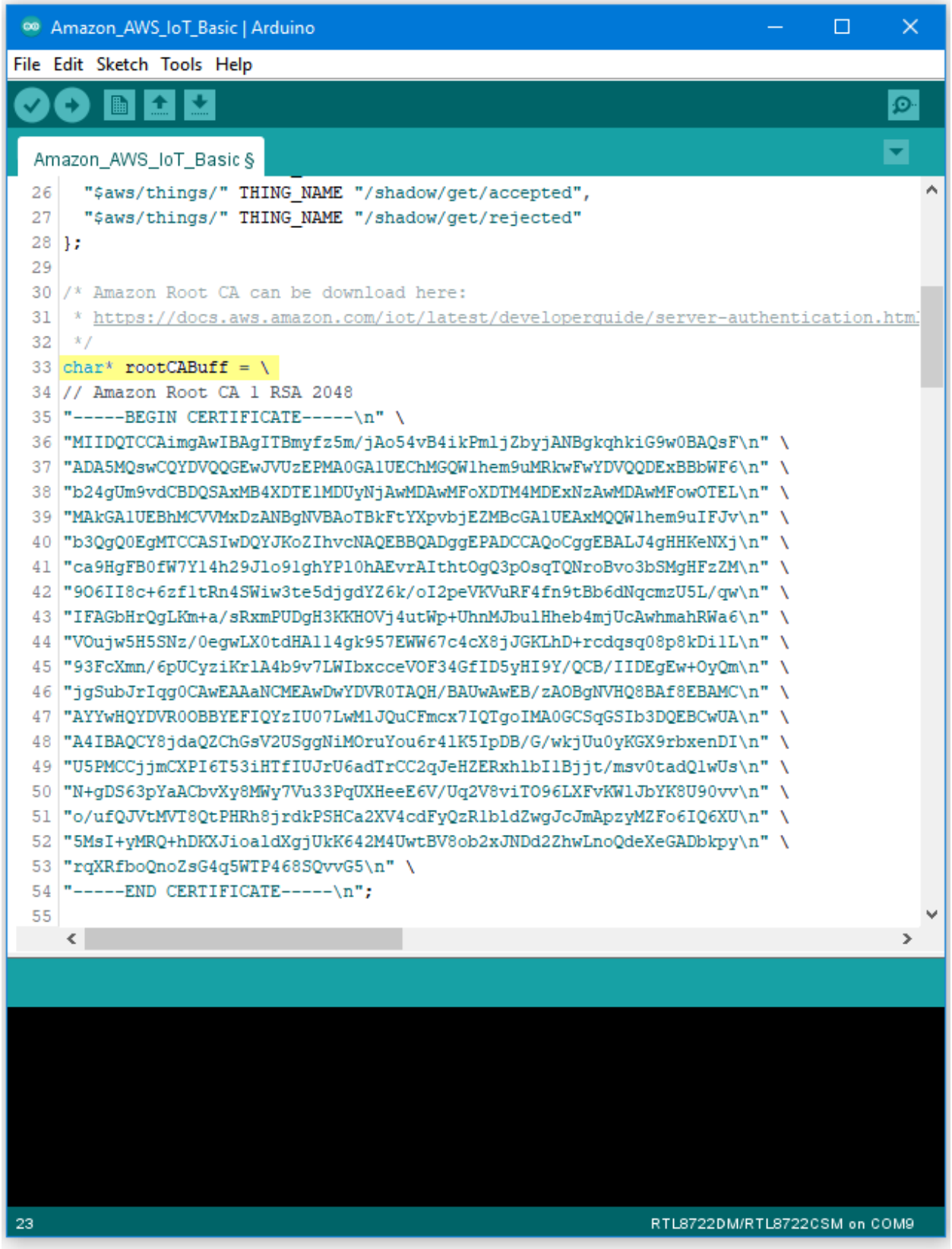
Then fill in the “thing” name “ameba”.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/*
Basic Amazon AWS IoT example
*/
#include <WiFi.h>
#include <PubSubClient.h>
// Update these with values suitable for your network.
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "yourPassword"; // your network password (use for WPA, or use
int status = WL_IDLE_STATUS; // the Wifi radio's status
WiFiSSLClient wifiClient;
PubSubClient client(wifiClient);
#define THING_NAME "ameba"
char mqttServer[] = "a4adrtgyzxov42-ats.iot.ap-southeast-1.amazonaws.com";
char clientId[] = "amebaClient";
char publishTopic[] = "$aws/things/" THING_NAME "/shadow/update";
char publishPayload[MQTT_MAX_PACKET_SIZE];
char *subscribeTopic[5] = {
"$aws/things/" THING_NAME "/shadow/update/accepted",
"$aws/things/" THING_NAME "/shadow/update/rejected",
"$aws/things/" THING_NAME "/shadow/update/delta",
"$aws/things/" THING_NAME "/shadow/get/accepted",
"$aws/things/" THING_NAME "/shadow/get/rejected"
};
/* Amazon Root CA can be download here:
23
RTL8722DM/RTL8722CSM on COM9

And the MQTT Broker server address we found earlier in AWS IoT.



Next, fill in the root CA used in TLS. Download and make sure the downloaded root CA contents conforms to the root CA used in the sketch.



Next, fill in the certificate we created in the AWS IoT Console (i.e., client certificate), usually its file name ends with "-certificate.pem.crt" (e.g., "efae24a533-certificate.pem.crt"). Open the certificate with a text editor, and adjust its format as follows to use in the sketch:

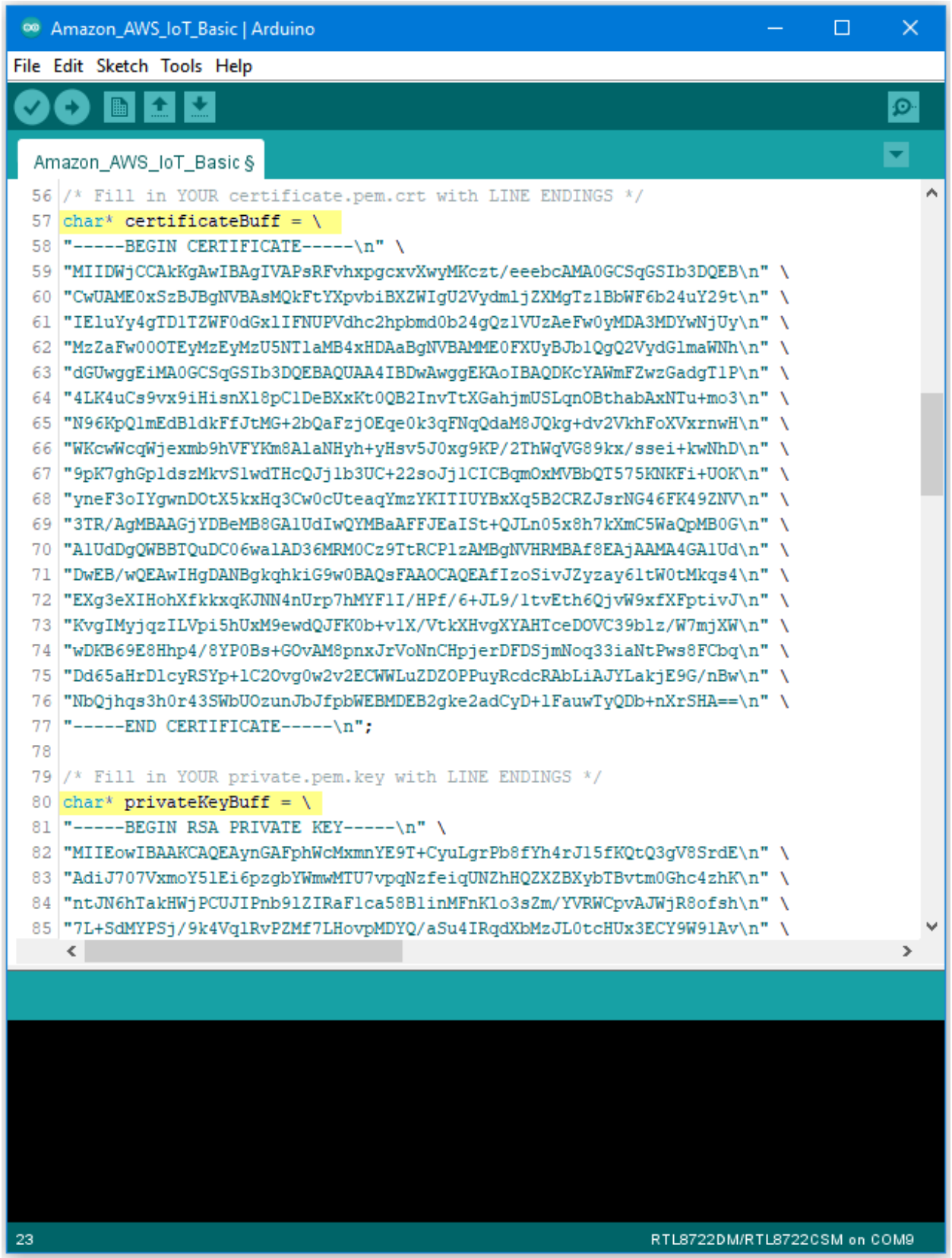
– Add the new line character "n" at the end of each line.

– Add double-quote at the beginning and the end of each line.

– To concatenate each line as a string, add "" at the end of each line.

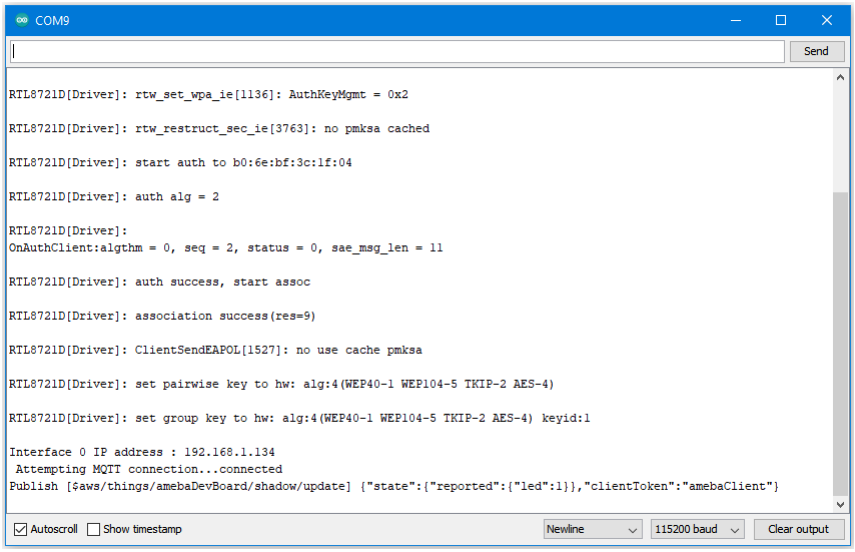
– The last line ends with semicolon.

Adjust the format of the private key in the same way and add it to privateKeyBuff.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/* Fill in YOUR certificate.pem.crt with LINE ENDINGS */
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDWjCCAkKgAwIBAgIVAPsRFvhxpgcxvXwyMKczt/eeebcAMA0GCSqGSIb3DQEB\n" \
"CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTzlBbWF6b24uY29t\n" \
"IEluYy4gTDlTZWF0dGxlIFNUPVdhc2hpbmd0b24gQzlVUzAeFw0yMDA3MDYwNjUy\n" \
"MzZaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJblQgQ2VydGlmaWNh\n" \
"dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDKcYAWmFZwzGadgTlP\n" \
"4LK4uCs9vx9iHisnXl8pClDeBXxKt0QB2InvTtXGahjmUSLqnOBthabAxNTu+mo3\n" \
"N96KpQlmEdBldkFfJtMG+2bQaFzjOEqe0k3qFNqQdaM8JQkg+dv2VkhFoXVxrnwH\n" \
"WKcwWcqWjexmb9hVFYKm8AlaNHyh+yHsv5J0xg9KP/2ThWqVG89kx/ssei+kwNhD\n" \
"9pK7ghGpldszMkvSlwdTHcQJjlb3UC+22soJjlCICBqmOxMVBbQT575KNKFi+UOK\n" \
"yneF3oIYgwnDOtX5kxHq3Cw0cUteaqYmzYKITIUYBxXq5B2CRZJsrNG46FK49ZNV\n" \
"3TR/AgMBAAGjYDBeMB8GA1UdIwQYMBaAFFJEaISt+QJLn05x8h7kXmC5WaQpMB0G\n" \
"A1UdDgQWBBTQuDC06walAD36MRM0Cz9TtRCPlzAMBgNVHRMBAf8EAjAAMA4GA1Ud\n" \
"DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAfIzoSivJZyzay6ltW0tMkqs4\n" \
"EXg3eXIHohXfkkxqKJNN4nUrp7hMYFlI/HPf/6+JL9/ltvEth6QjvW9xfXFptivJ\n" \
"KvgIMyjqzILVpi5hUxM9ewdQJFK0b+vlX/VtkXHvgXYAHTceDOVC39blz/W7mjXW\n" \
"wDKB69E8Hhp4/8YP0Bs+GOvAM8pnxJrVoNnCHpjerDFDSjmNoq33iaNtPws8FCbq\n" \
"Dd65aHrDlcyRSYp+lC2Ovg0w2v2ECWWLuZDZOPPuyRcdcRAbLiAJYLakjE9G/nBw\n" \
"NbQjhqs3h0r43SWbUOzunJbJfpbWEBMDEB2gke2adCyD+lFauwTyQDb+nXrSHA==\n" \
"-----END CERTIFICATE-----\n";
/* Fill in YOUR private.pem.key with LINE ENDINGS */
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEowIBAAKCAQEAynGAFphWcMxmnYE9T+CyuLgrPb8fYh4rJ15fKQtQ3gV8SrdE\n" \
"AdiJ707VxmoY5lEi6pzgbYWmwMTU7vpqNzfeiqUNZhHQZXZBXybTBvtm0Ghc4zhK\n" \
"ntJN6hTakHWjPCUJIPnb9lZIRaFlca58BlinMFnKlo3sZm/YVRWCpvAJWjR8ofsh\n" \
"7L+SdMYPSj/9k4VqlRvPZMf7LHovpMDYQ/aSu4IRqdXbMzJL0tcHUx3ECY9W9lAv\n" \
23
RTL8722DM/RTL8722CSM on COM9

Compile and run

Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor in the Arduino IDE and observe as Ameba connects to the AWS IoT server and sends updates on the LED state variable.



Alternatives

Ameba can also retrieve the current LED status variable from the AWS shadow. This is done by sending a message to the “shadow/get” topic. Refer to the Amazon_AWS_IoT_with_ACK example code for more information.

Code Reference

Change led state:
In this example, we use GPIO interface to control the led. We set led_pin to 10 and led_state to 1 by default in the sample code.

```
pinMode(led_pin, OUTPUT);
digitalWrite(led_pin, led_state);
```

Set up certificate:
Note that we use the WiFiSSLClient type of wifiClient.

```
WiFiSSLClient wifiClient;
```

WiFiSSLClient inherits Client, so it can be passed as the parameter of PubSubClient constructor.
Next, set up TLS certificate required in connection.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
wifiClient.setClientCertificate((unsigned char*)certificateBuff,(unsigned
↪char*)privateKeyBuff);
```

Configure MQTT Broker server
Then MQTT PubClient set MQTT Broker server to connect

```
client.setServer(mqttServer, 8883);
client.setCallback(callback);
```

Connect to MQTT Broker server:
In `loop()`, call `reconnect()` function and try to connect to MQTT Broker server and do the certificate verification.

```
while (!client.connected()) {
```

Subscribe & Publish
Next, subscribe to topics.

```
for (int i=0; i<5; i++) {
  client.subscribe(subscribeTopic[i]);
}
```

There are some common topics:
“$aws/things/ameba/shadow/update/accepted”,
“$aws/things/ameba/shadow/update/rejected”,
“$aws/things/ameba/shadow/update/delta”,
“$aws/things/ameba/shadow/get/accepted”,
“$aws/things/ameba/shadow/get/rejected”
Related documentation:
http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-data-flow.html

Then publish current status::

```
sprintf(publishPayload,
"{\"state\":{\"reported\":{\"led\":%d}},\"clientToken\":\"%s\"}",
led_state, clientId);
```

```
client.publish(publishTopic, publishPayload);
```

Listen to topic and make response:
In the callback function, we listen to the 5 subscribed topics and check if there are messages of "/shadow/get/accepted":

```
if (strstr(topic, "/shadow/get/accepted") != NULL) {
```

If there is, the message is from the control side. If the attribute state in the message is different from current state, publish the new state.

```
updateLedState(desired_led_state);
```

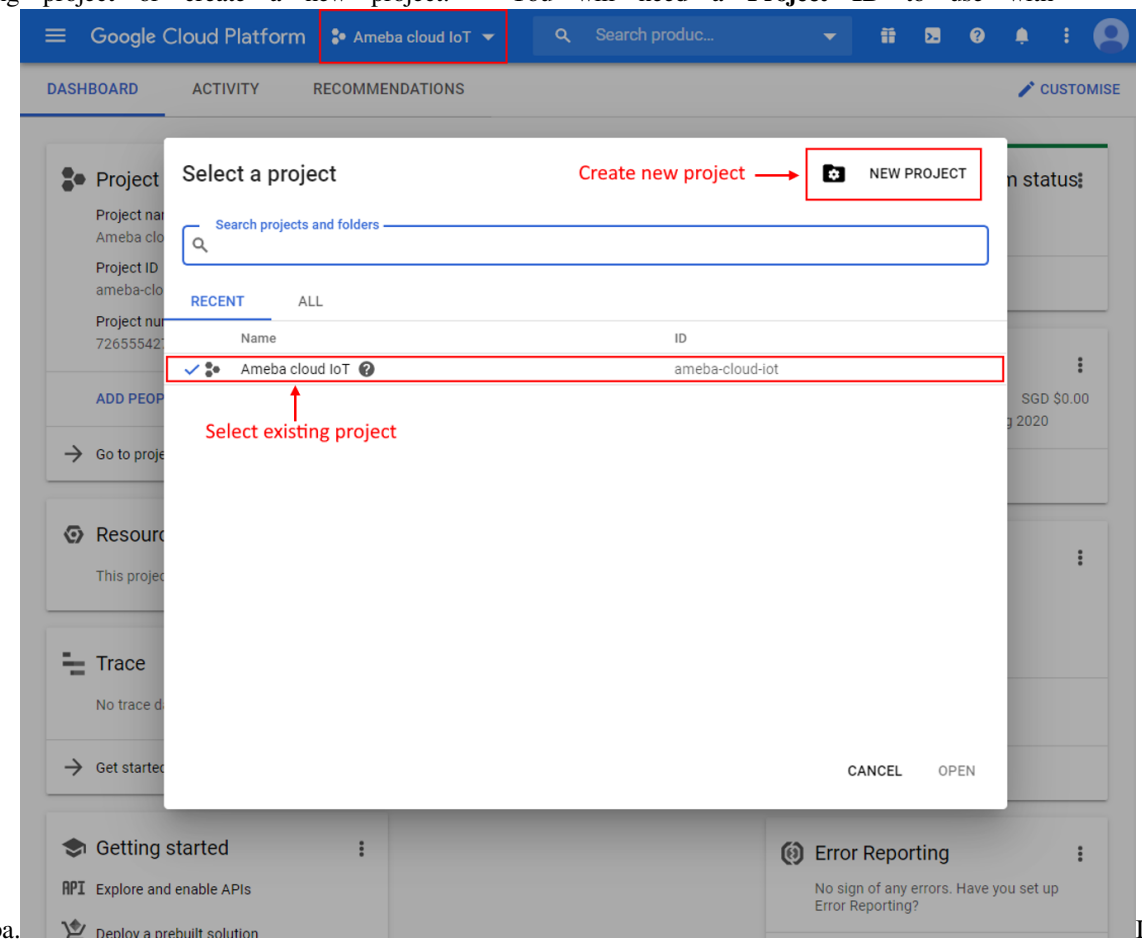
MQTT - Use Google Cloud IoT

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

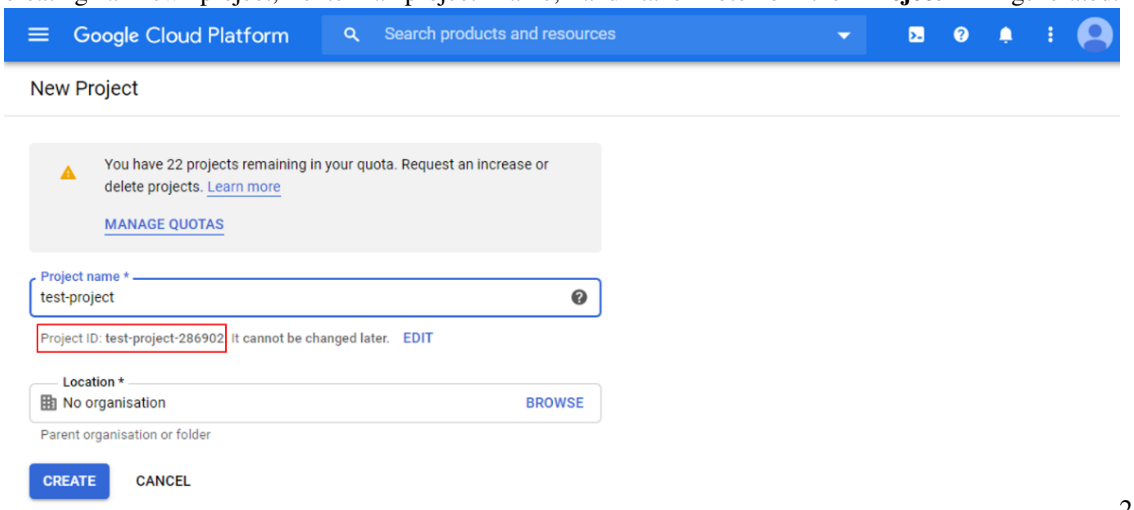
Google Cloud IoT Configuration

1. Select or create a Cloud Platform project In the Google Cloud Console, select an existing project or create a new project. You will need a **Project ID** to use with Ameba.

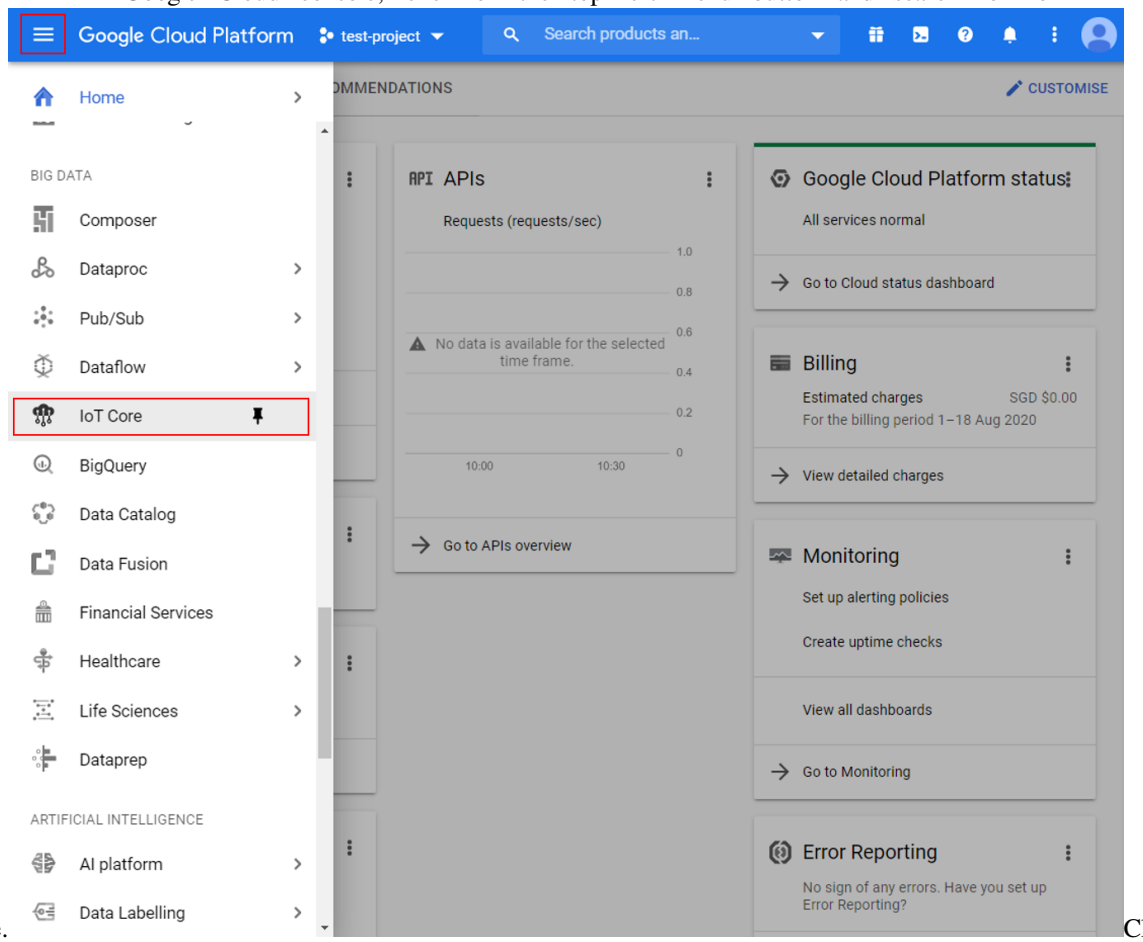


If

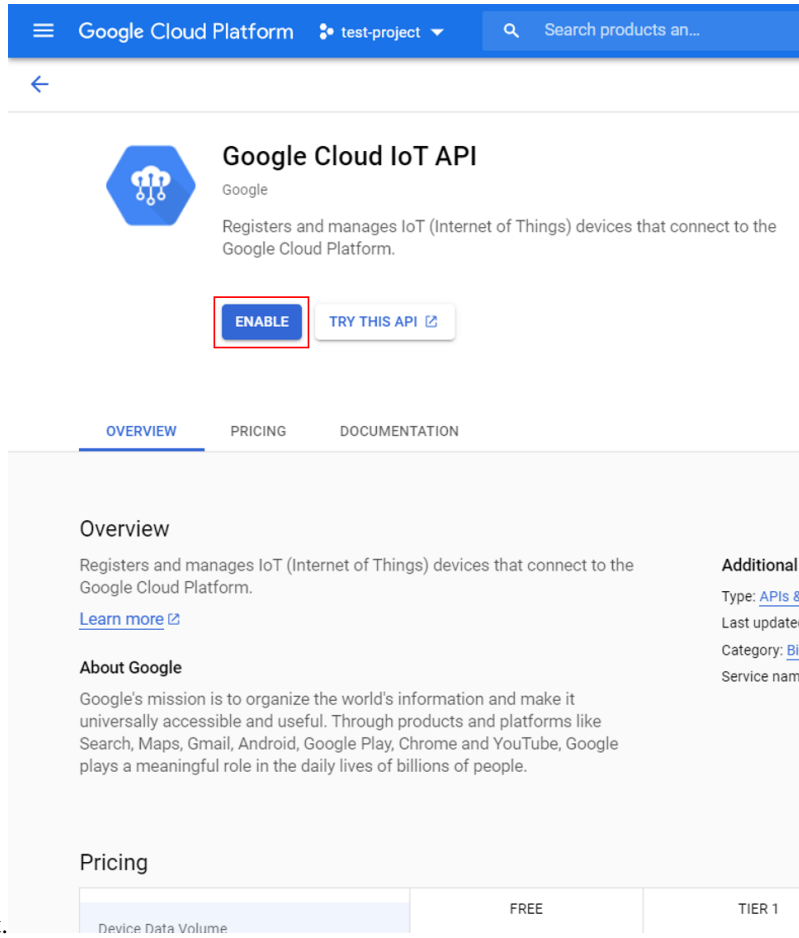
creating a new project, enter a project name, and take note of the **Project ID** generated.



2. Enable billing for your project Billing needs to be enabled for your project to use Google Cloud Platform features. Follow the guide in Google cloud documentation to enable billing. https://cloud.google.com/billing/docs/how-to/modify-project 3. Enable the Cloud IoT Core API In Google Cloud console, click on the top left menu button and search for IoT Core.

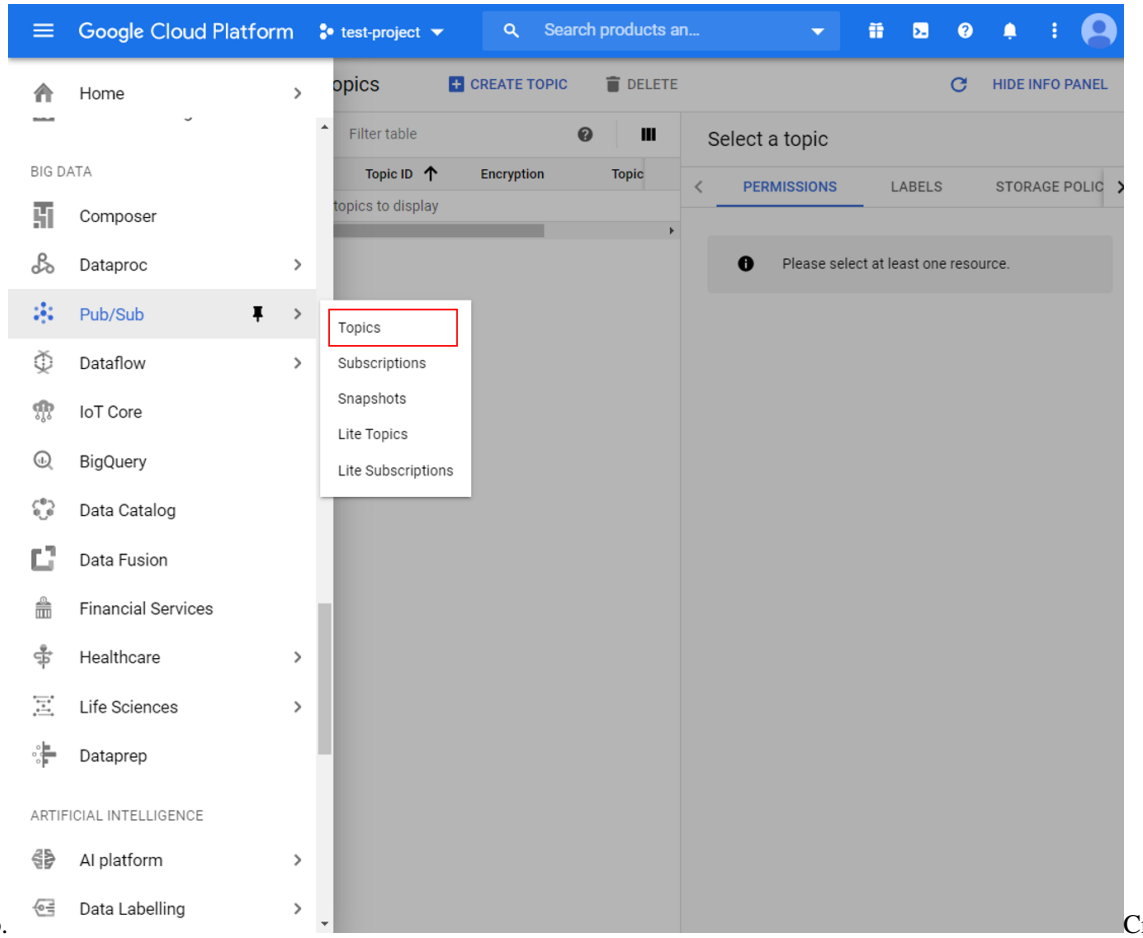


Click

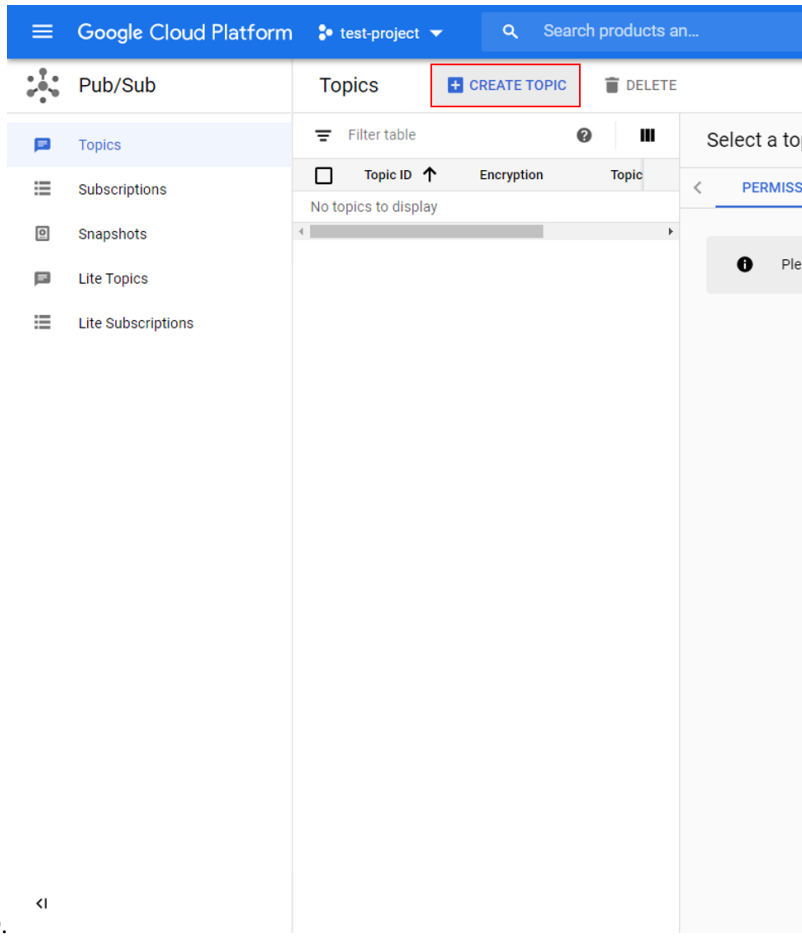


enable to activate Google Cloud IoT API for your project.

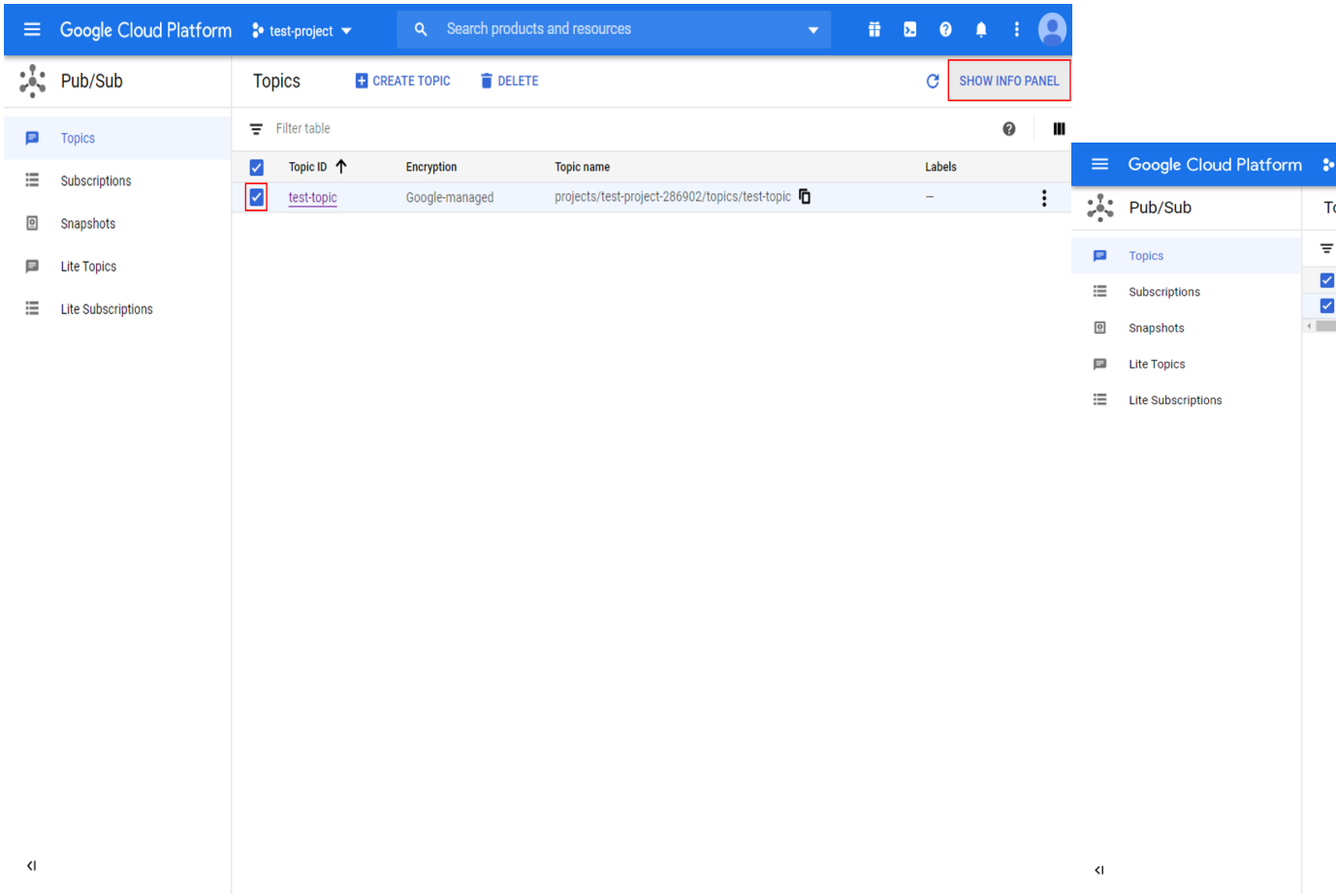
Create a Cloud Pub/Sub topic In Google Cloud console, click on the top left menu button and search for



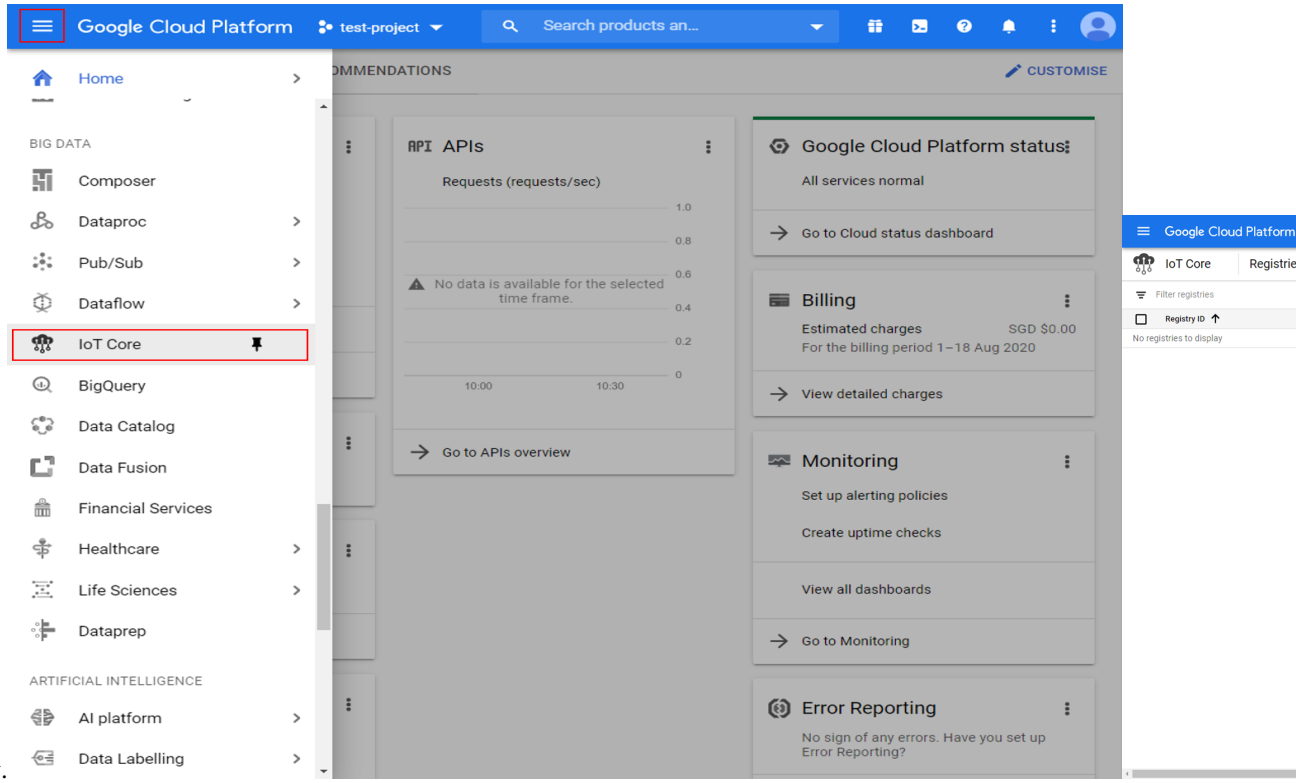
Pub/Sub. Create



a new topic for your project and give it a suitable topic ID. the topic is created, go to the permissions tab of the info panel, and add "cloud-iot@system.gserviceaccount.com" with the role of "Pub/Sub Publisher".

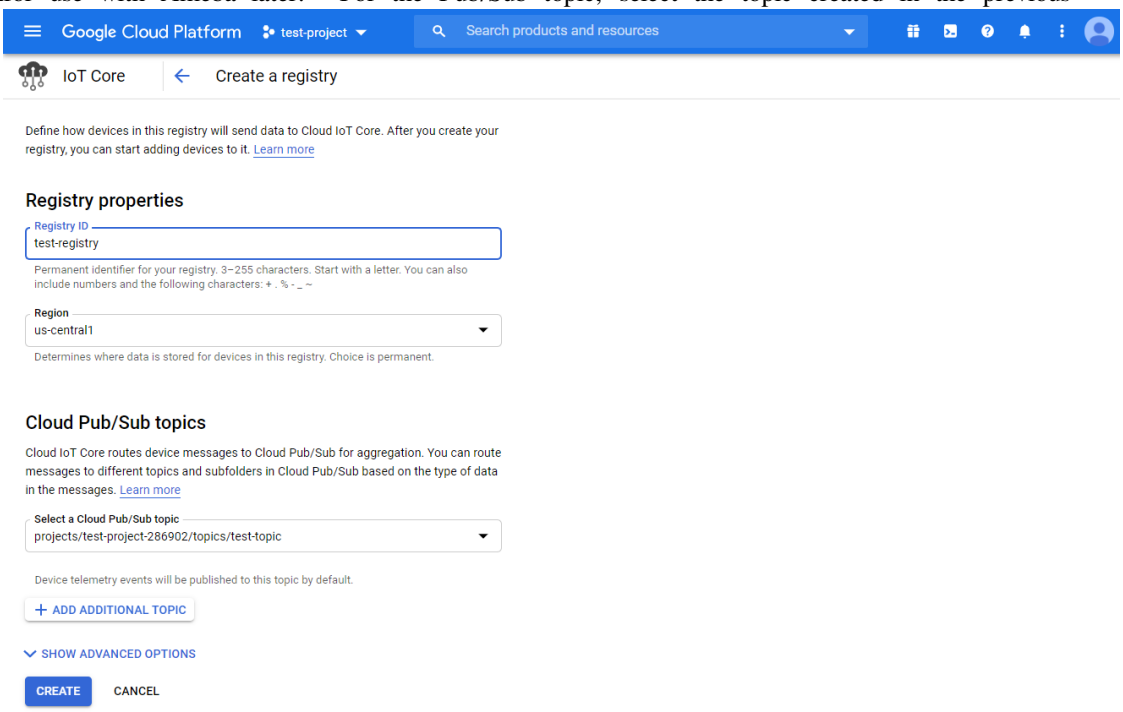


a device registry Go back to the IoT Core settings page and create a new



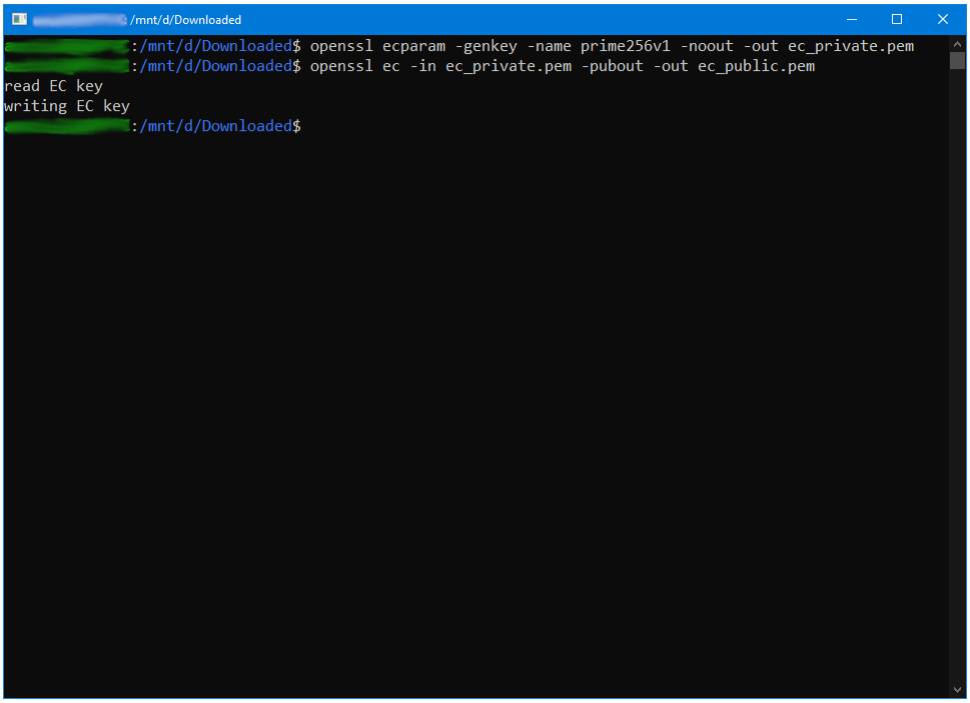
registry.

a suitable **Registry ID** and **in which to store data.** **Remember the **Registry ID** and **Region**for use with Ameba later. For the Pub/Sub topic, select the topic created in the previous



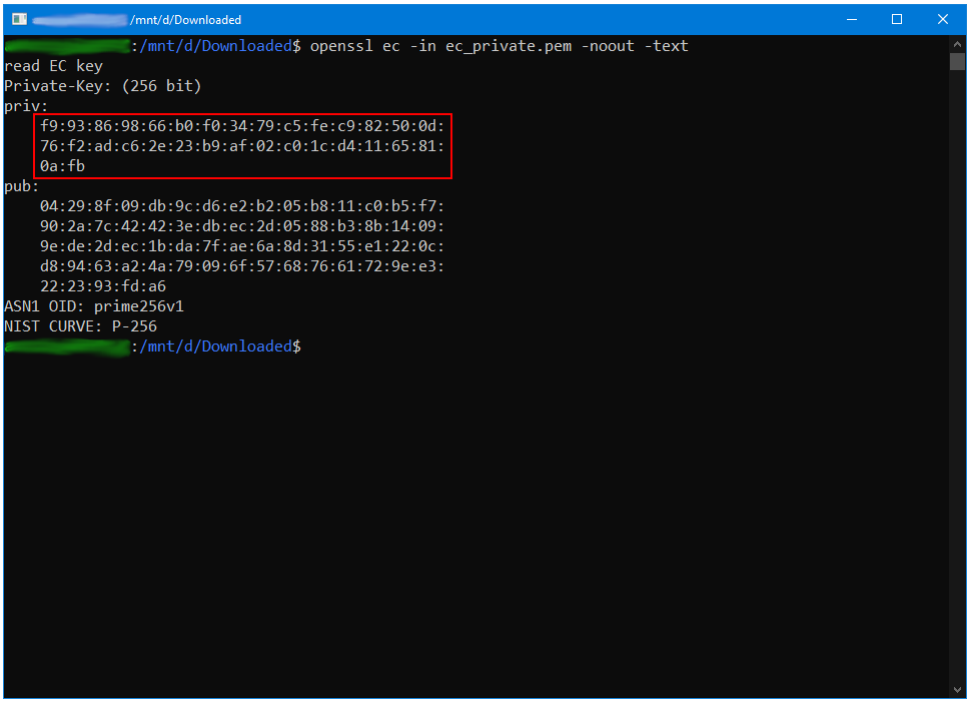
step. 6. Create a public/private key pair Using Openssl in a terminal in Windows/Linux/MacOs, run the following commands to generate a private and public key pair. Two files will be created by these commands, “ec_private.pem” containing the private key, and “ec_public.pem” containing the public key.

```
$ openssl ecparam -genkey -name prime256v1 -noout -out ec_private.pem
$ openssl ec -in ec_private.pem -pubout -out ec_public.pem
```

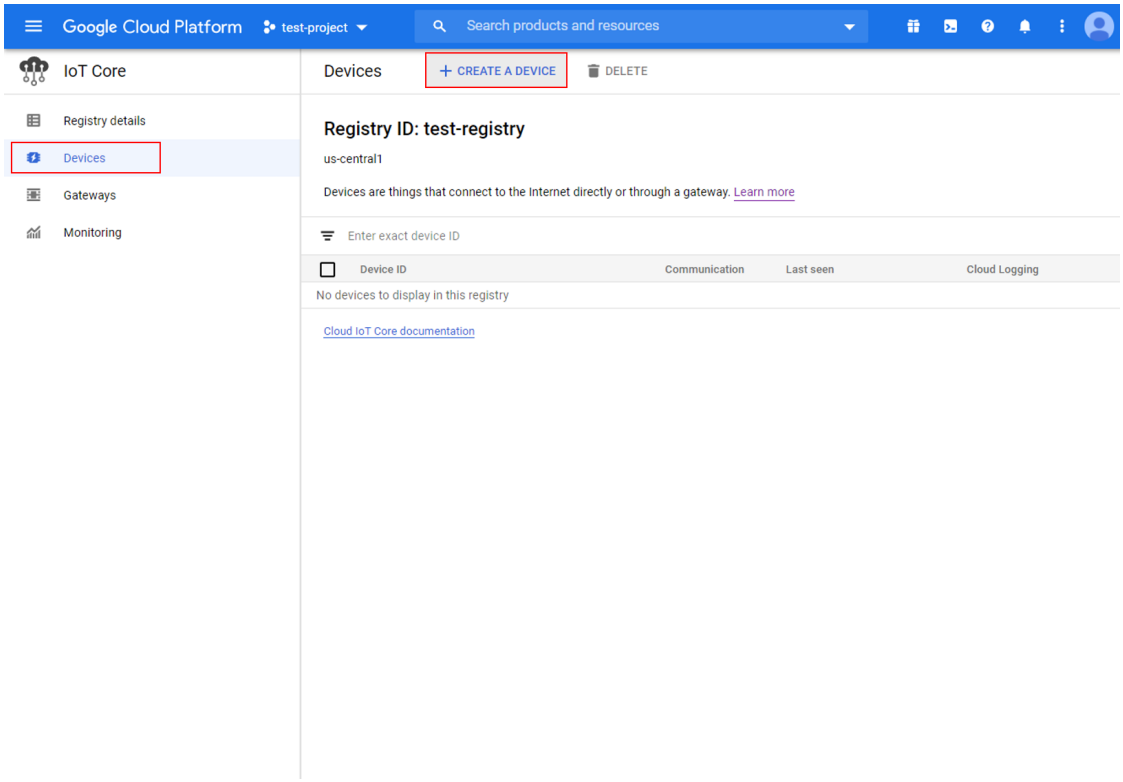


Run the next command to extract out the private key, and remember the highlighted string of hexadecimal numbers for use with Ameba later.

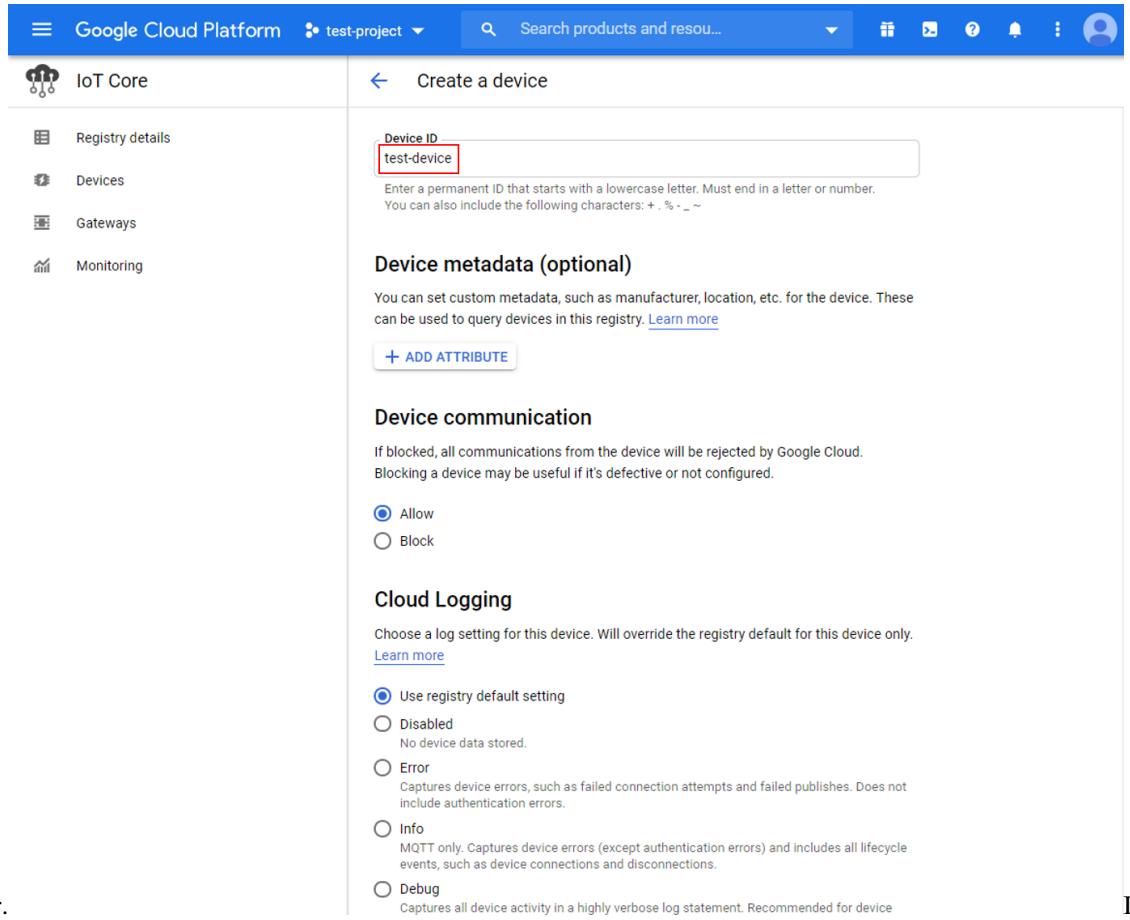
```
$ openssl ec -in ec_private.pem -noout -text
```



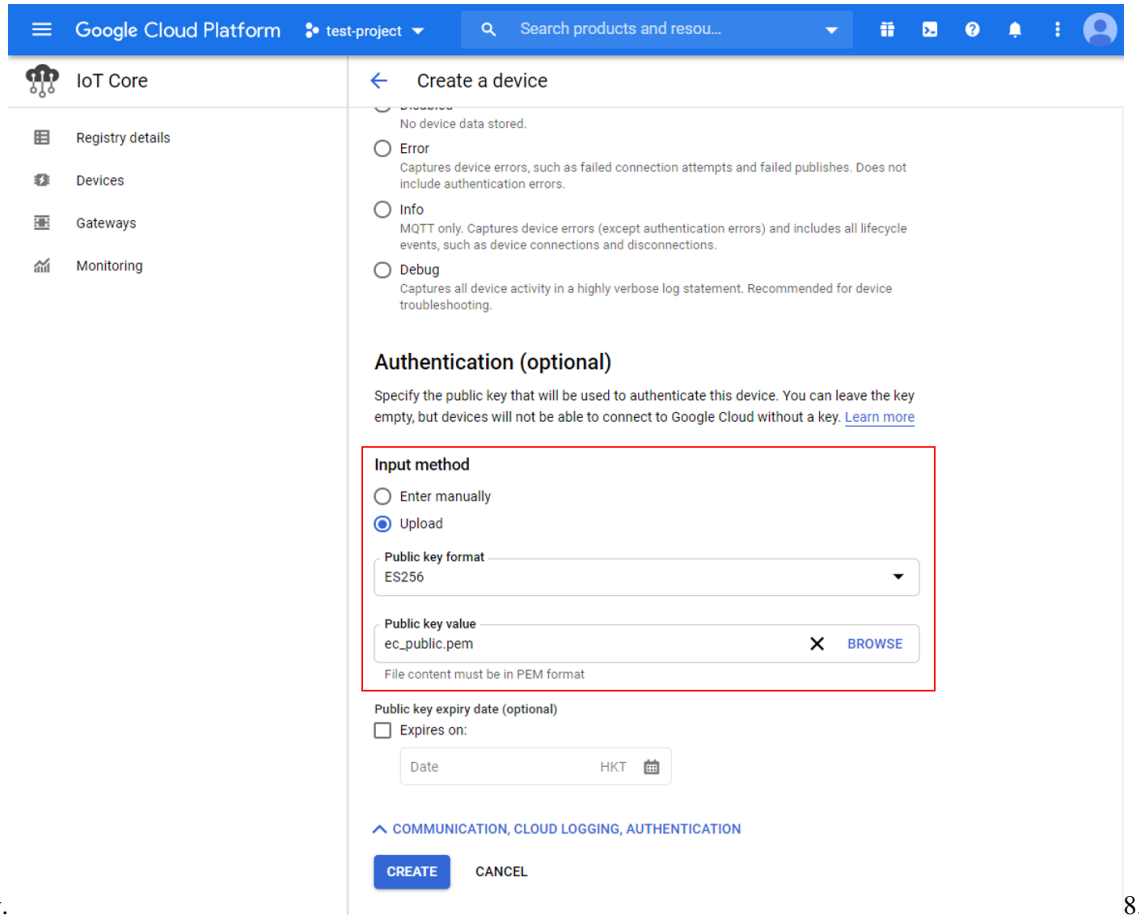
7. Create a device Go back to the IoT Core settings page and create a new device.



Give the device a suitable **Device ID** and remember it for use with Ameba

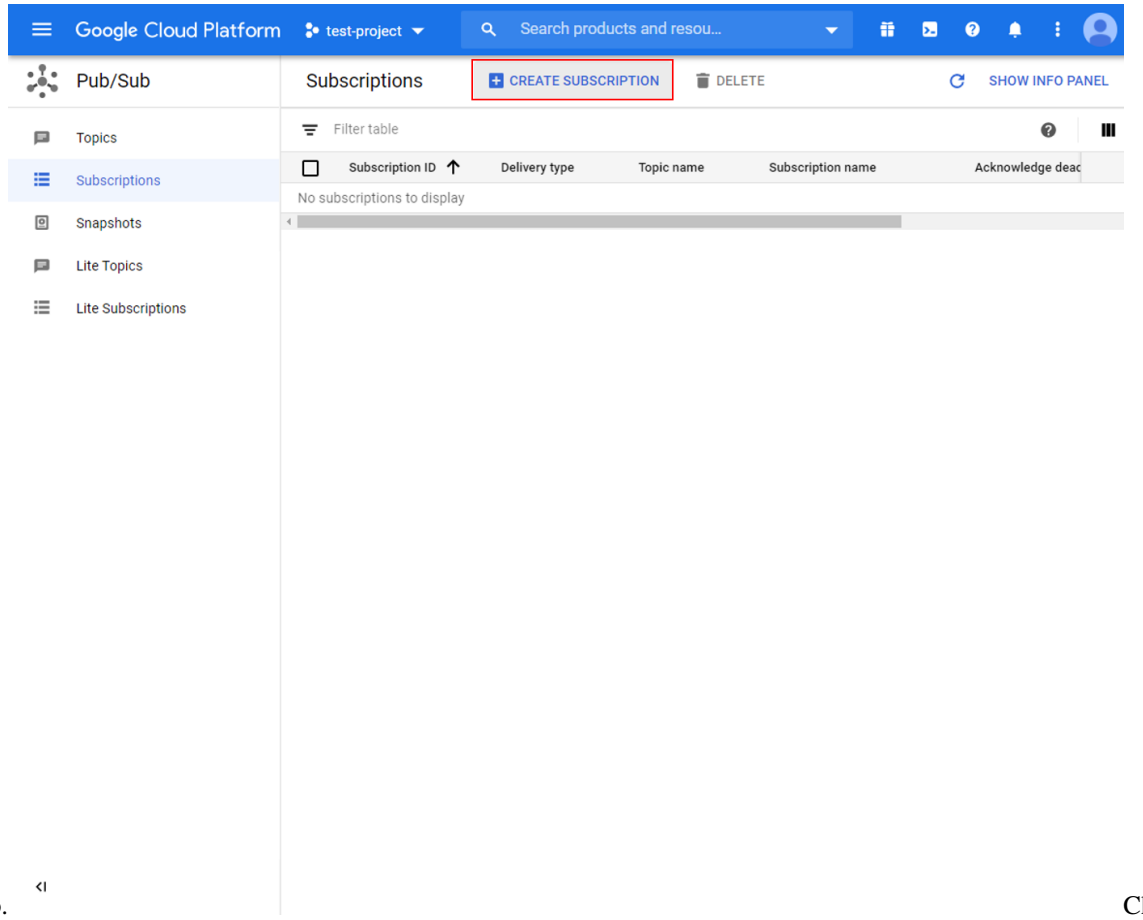


later. In the authentication section of the additional options, upload the previously generated “ec_public.pem” public

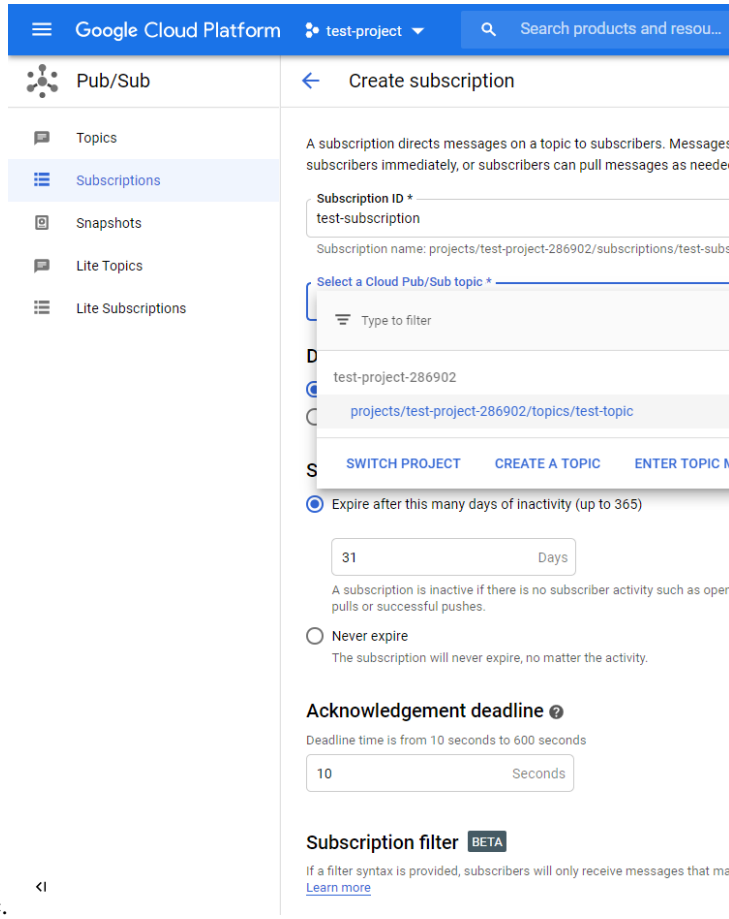


key.

8. Create a Cloud Pub/Sub subscription To observe messages sent by Ameba, create a subscription in



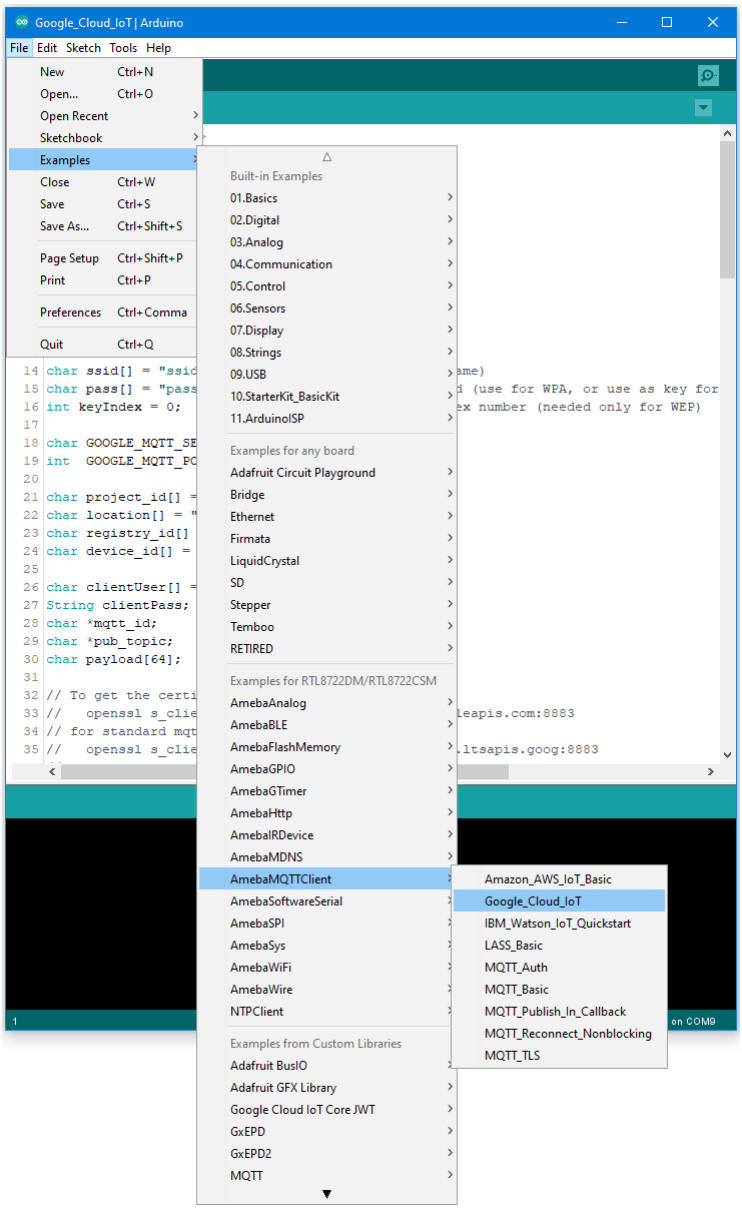
Pub/Sub. Choose



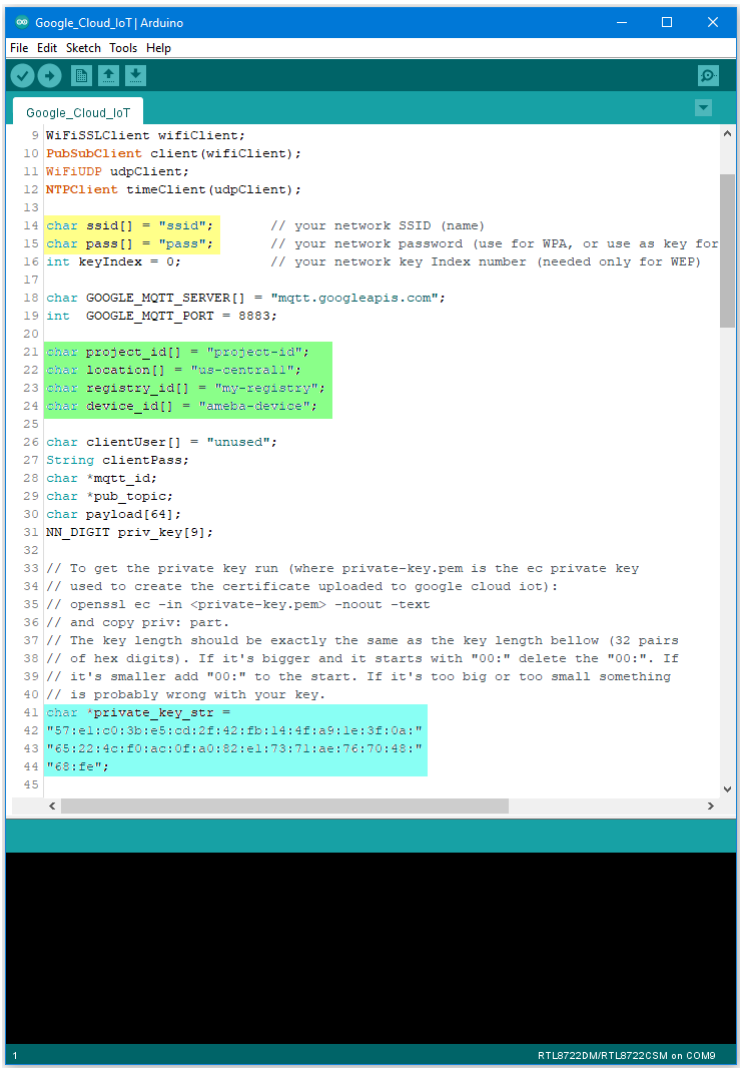
a suitable subscription ID and select the previously created topic.

Example

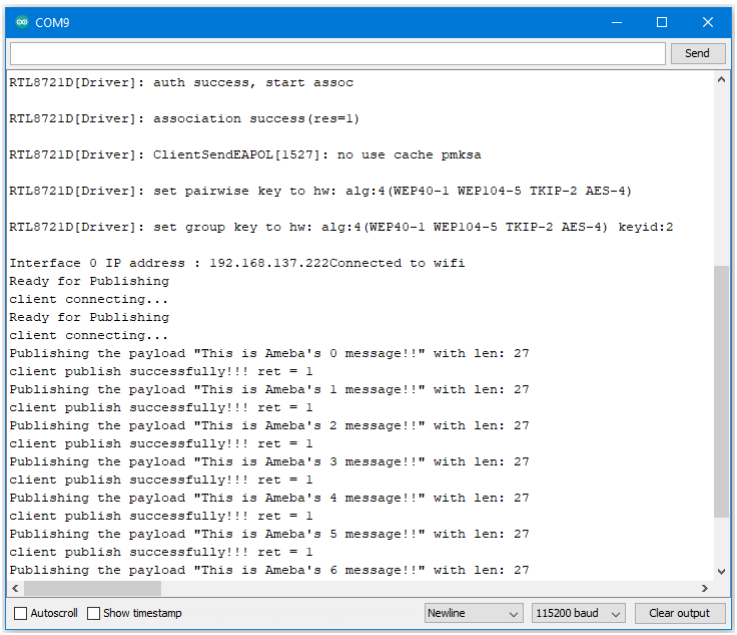
Open the example in "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`Google_Cloud_IoT`".



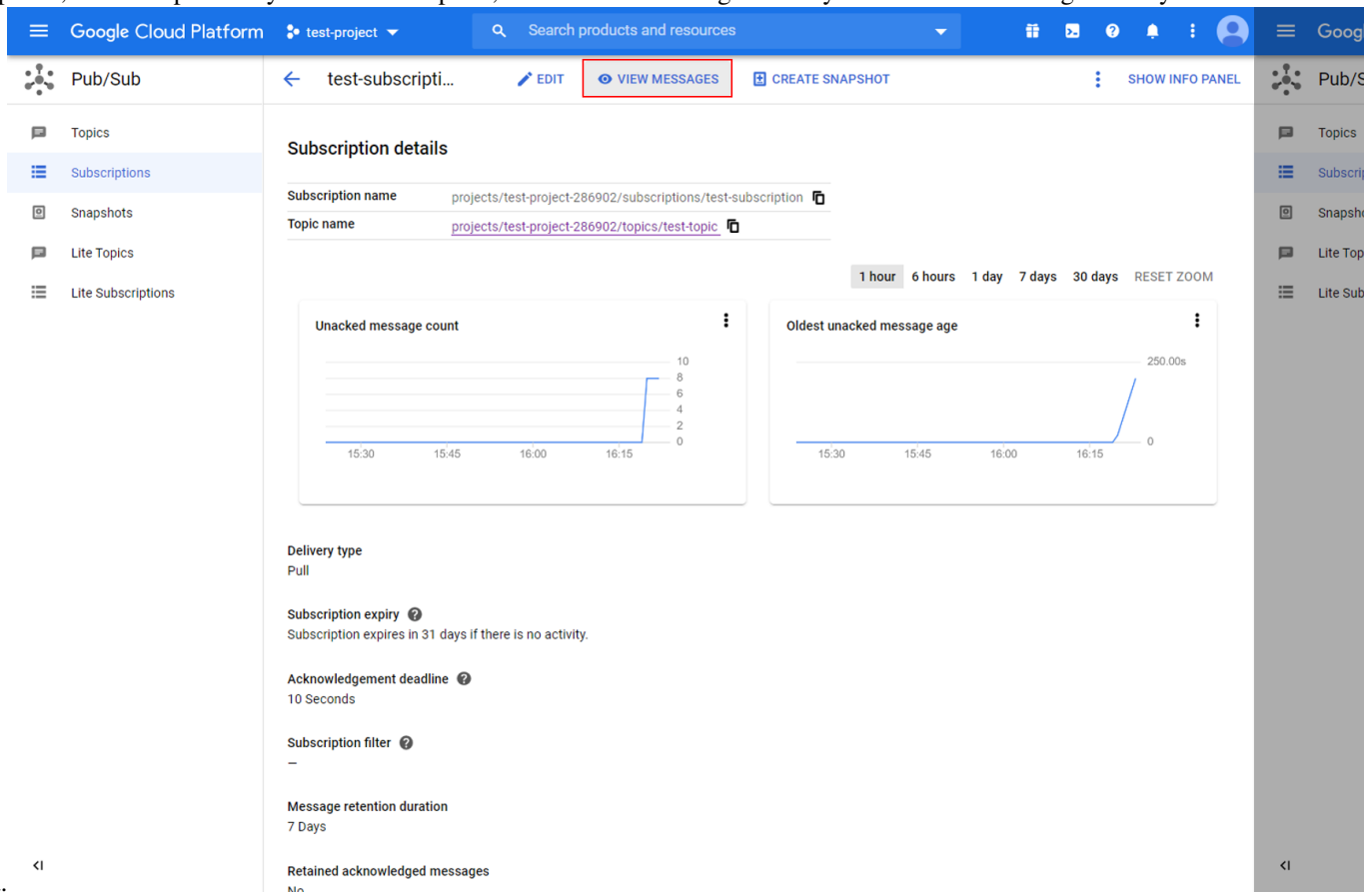
Enter the required information in the highlighted sections below.



In the yellow section, enter the SSID and password required to connect to your WiFi network. In the green section, enter the Project ID, server Region, Registry ID and Device ID previously configured in Google Cloud console. In the blue section, enter the hexadecimal string previously extracted from the private key. Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor and observe as Ameba connects and sends messages to Google Cloud IoT.



In Google Cloud console, go to Pub/Sub subscriptions, select the previously created subscription, and click view messages. Here you can view the messages sent by Ameba.



Code Reference

In `setup()`, we set up RootCA which is required to form a TLS connection with Google's servers.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
```

In `loop()`, each loop checks the Internet status and re-connect to it when the environment has a problem.

```
if (WiFi.status() != WL_CONNECTED) {
   while (WiFi.begin(ssid, pass) != WL_CONNECTED)
   {
      delay(1000);
   }
   Serial.println("Connected to wifi");
}
```

To publish messages, mqtt_id , clientPass and pub_topic are required. mqtt_id is generated by printing the project ID, server location, registry ID and device ID in the required format:

```
mqtt_id = (char *)malloc(strlen("projects/") + strlen(project_id) + strlen("/
↪locations/us-central1/registries/") + strlen(registry_id) + strlen("/devices/") +␣
↪strlen(device_id) + 1);
sprintf(mqtt_id, "projects/%s/locations/us-central1/registries/%s/devices/%s",␣
↪project_id, registry_id, device_id);
```

`clientPass` is generated using a JSON web token (JWT) generator function, which requires the project ID and current time, and signs it with the private key:

```
clientPass = CreateJwt(project_id, timeClient.getEpochTime(), priv_key);
```

`pub_topic` is generated by printing the project ID and topic in the required format:

```
pub_topic = (char *)malloc(strlen("/devices/") + strlen(device_id) + strlen("/events
↪") + 1);
sprintf(pub_topic, "/devices/%s/events", device_id);
```

MQTT Server setting:

```
client.setServer(GOOGLE_MQTT_SERVER, GOOGLE_MQTT_PORT);
client.setPublishQos(MQTTQOS1);
client.waitForAck(true);
```

Connect to google cloud and publish messages:

```
if (client.connect(mqtt_id, clientUser, clientPass.c_str())){
   // ...
        for(int i = 0; i < count; i++){
      // ...
      sprintf(payload, "This is Ameba's %d message!!", i);
      ret = client.publish(pub_topic, payload);
      // ...
   }
   // ...
   client.disconnect();
}
free(mqtt_id);
free(pub_topic);
```

MQTT - Upload PM2.5 Data to LASS System

Intro to LASS

The LASS stands for "Location Aware Sensor System". It is an open project and was started only for the interest of public welfare. Find detailed introduction here.
Practically, LASS is based on MQTT protocol to collect all kinds of uploaded data, and for those who need these data can subscribe top as well.
Find more LASS information at their official hackpad.

Preparation

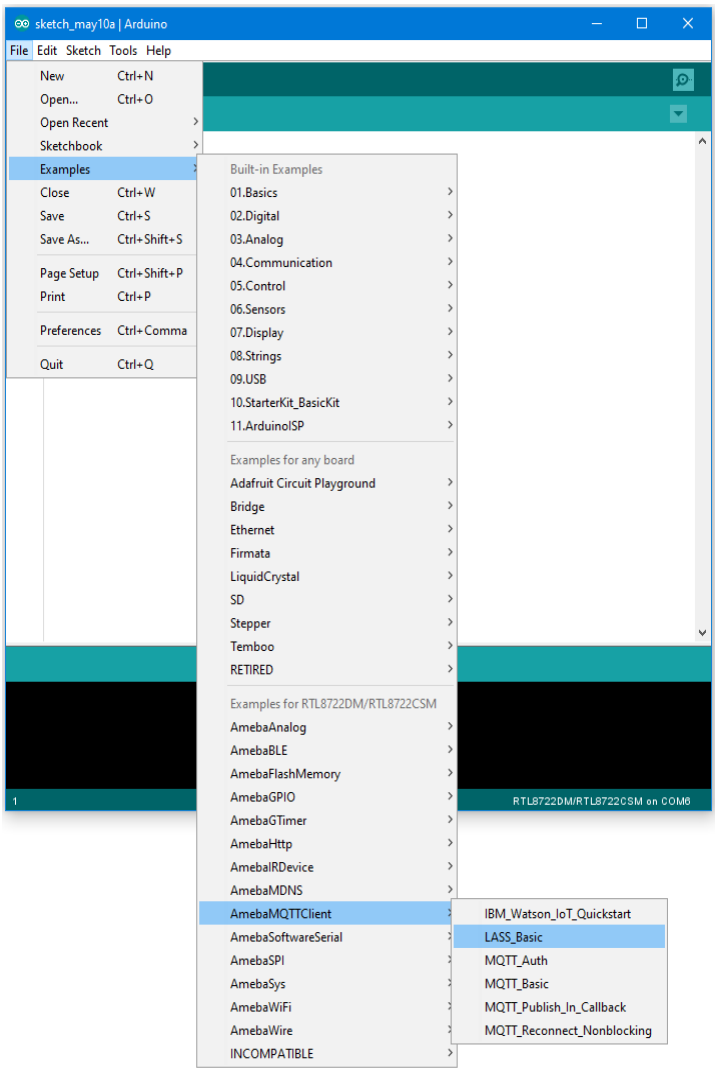
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x1

Example

In this example, we use applications mentioned at our website, including:

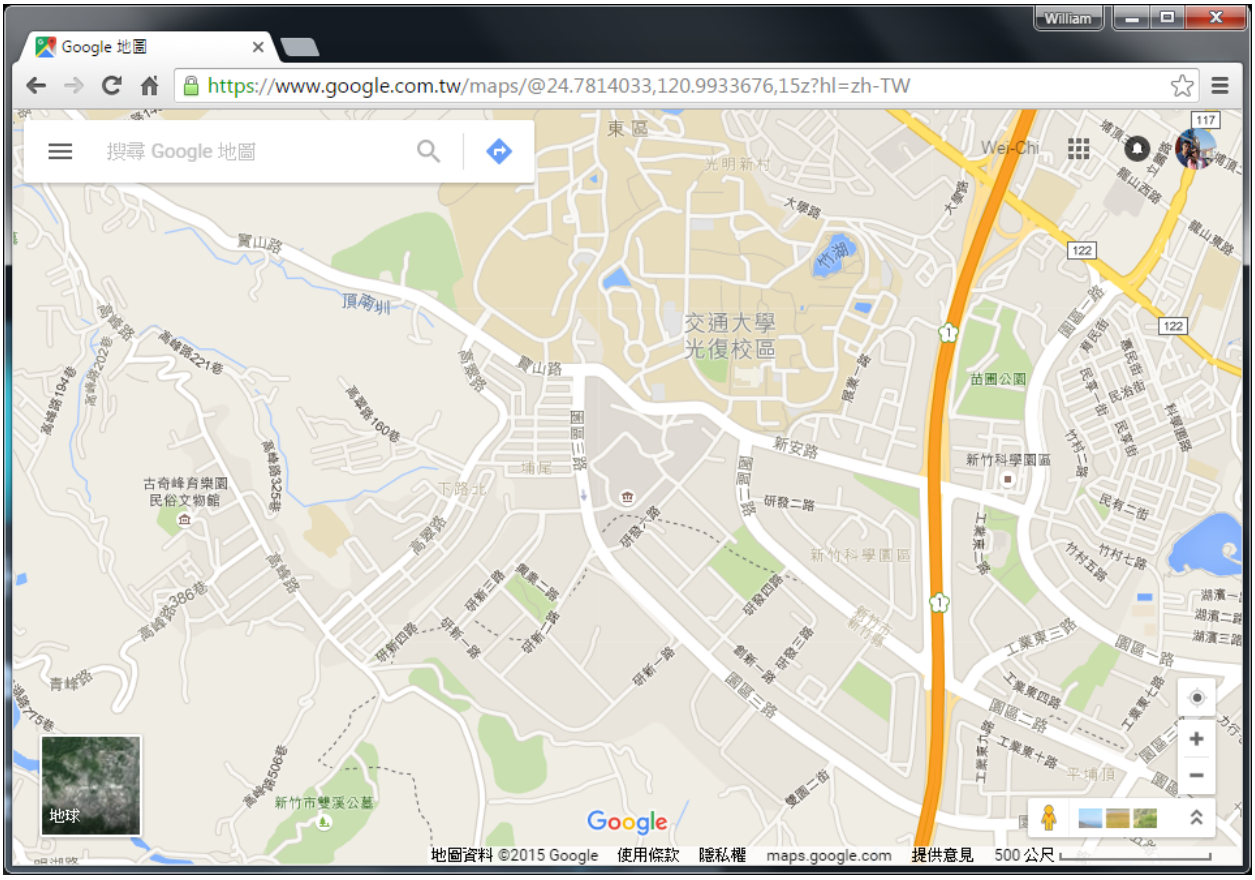
- MQTT: a MQTT-Broker to connect to LASS. The Client is "FT1_0XXXX", the XXXX are the four last digits of Ameba's Wi-Fi MAC, and the outTopic is "LASS/Test/Pm25Ameba/**clientID**", where clientID is the actual Ameba's MQTT client ID.
- NTP: uploaded data must have time notation
- PM2.5: uploaded data includes PM2.5 information

Open the example. "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`lass_basic`"

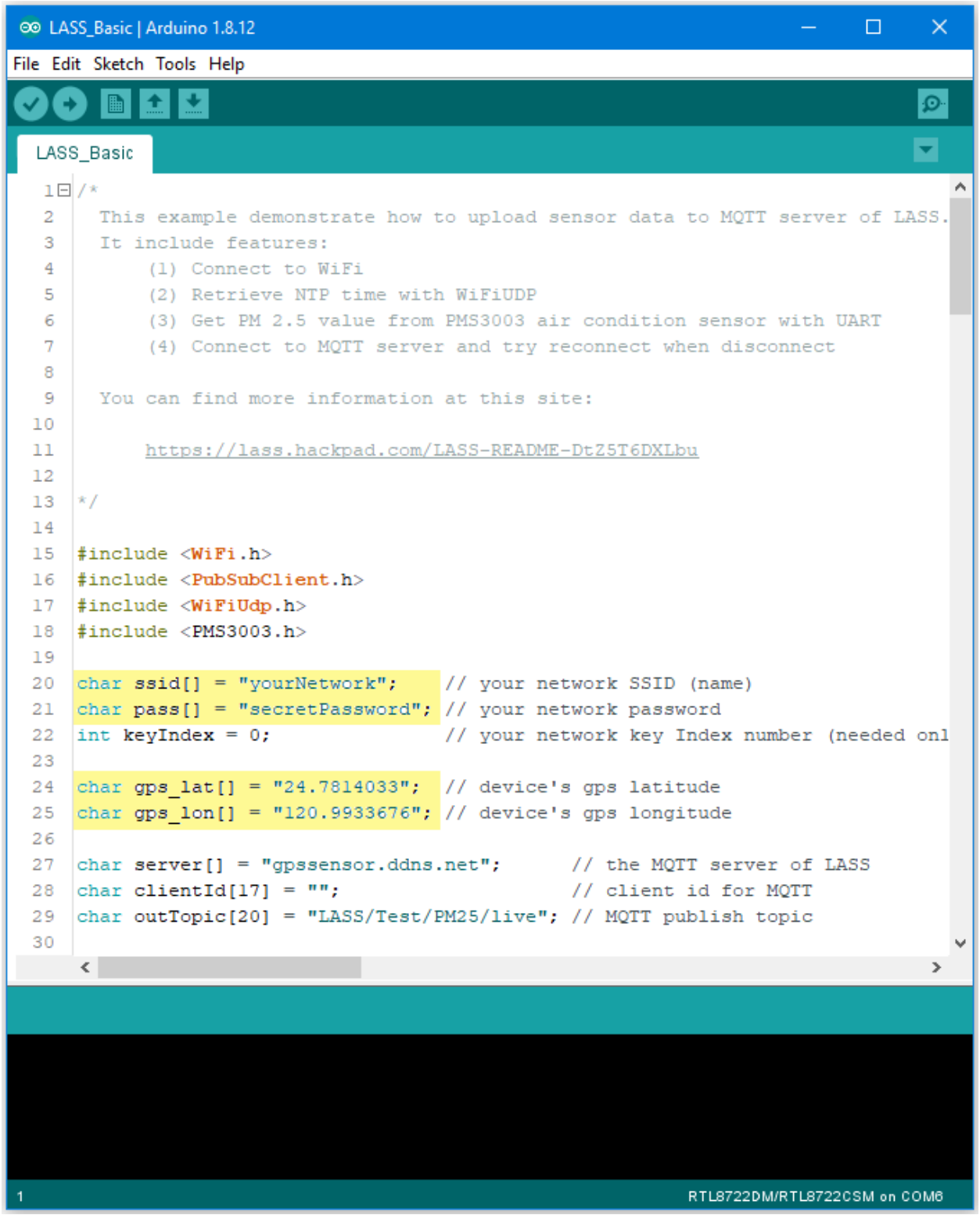


This example requires internet connection, so make sure you fill in SSID and PASS into AP information that you wish to connect.

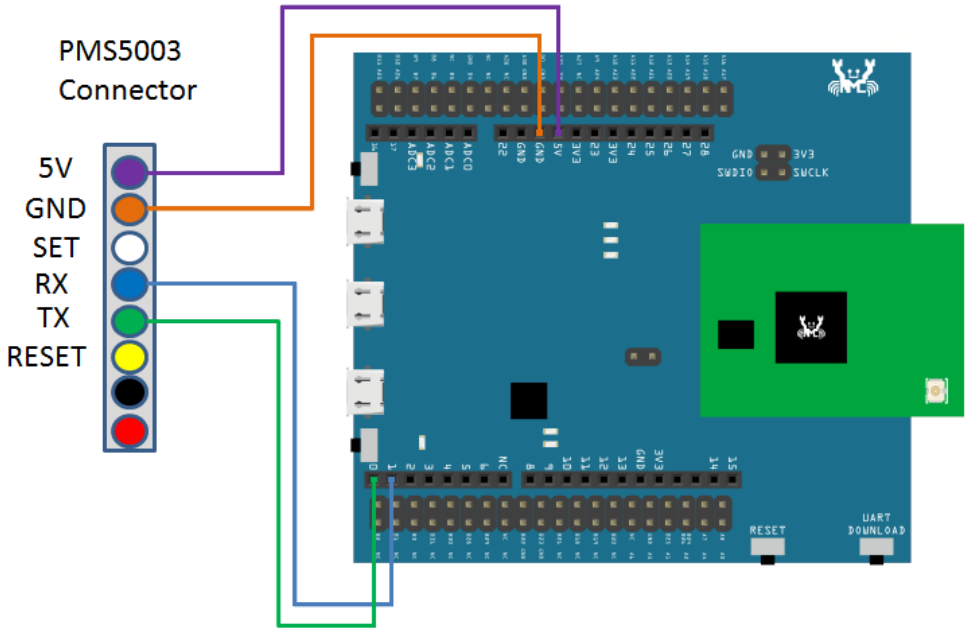
Also, LASS requires GPS information. There is no GPS sensor included in this example, so you must manually provide GPS information. Use Google Map to find the coordinates you plan to place your Ameba. You can see in this example that the latitude is 24.7814033, and the longitude is 120.9933676



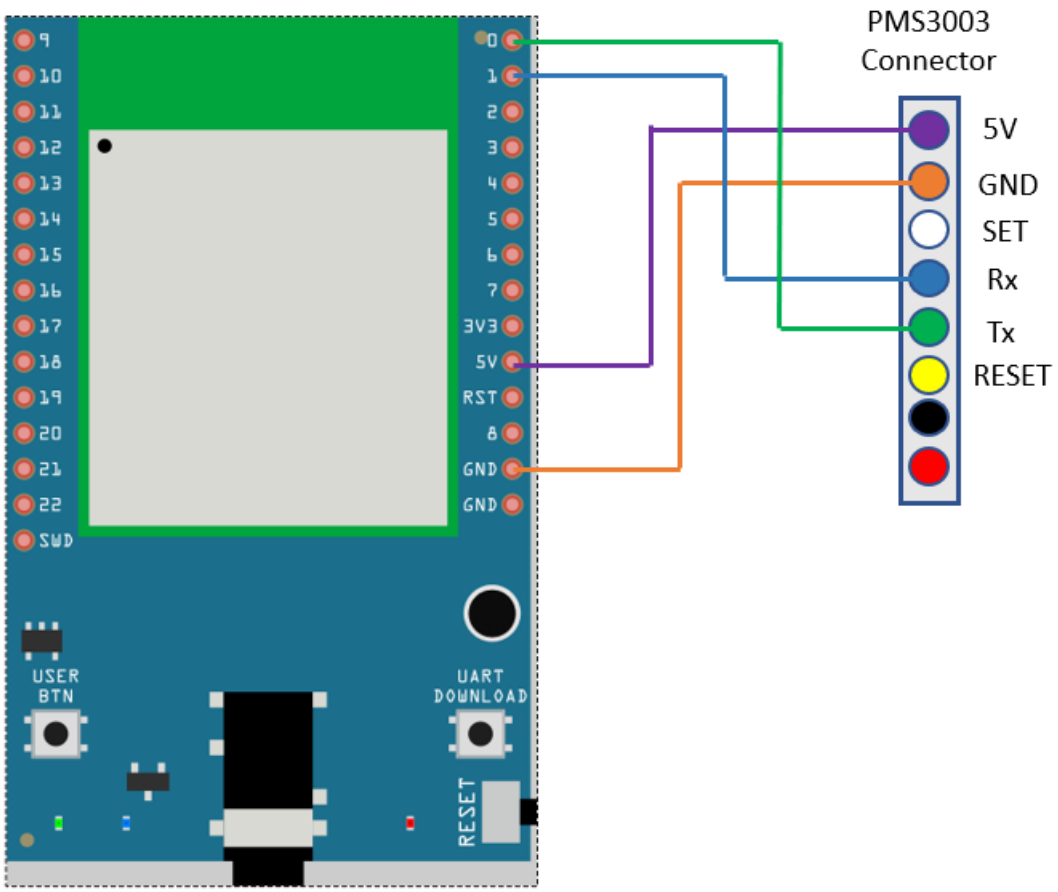
Fill in GPS info at `gps_lat` and `gps_lon`.



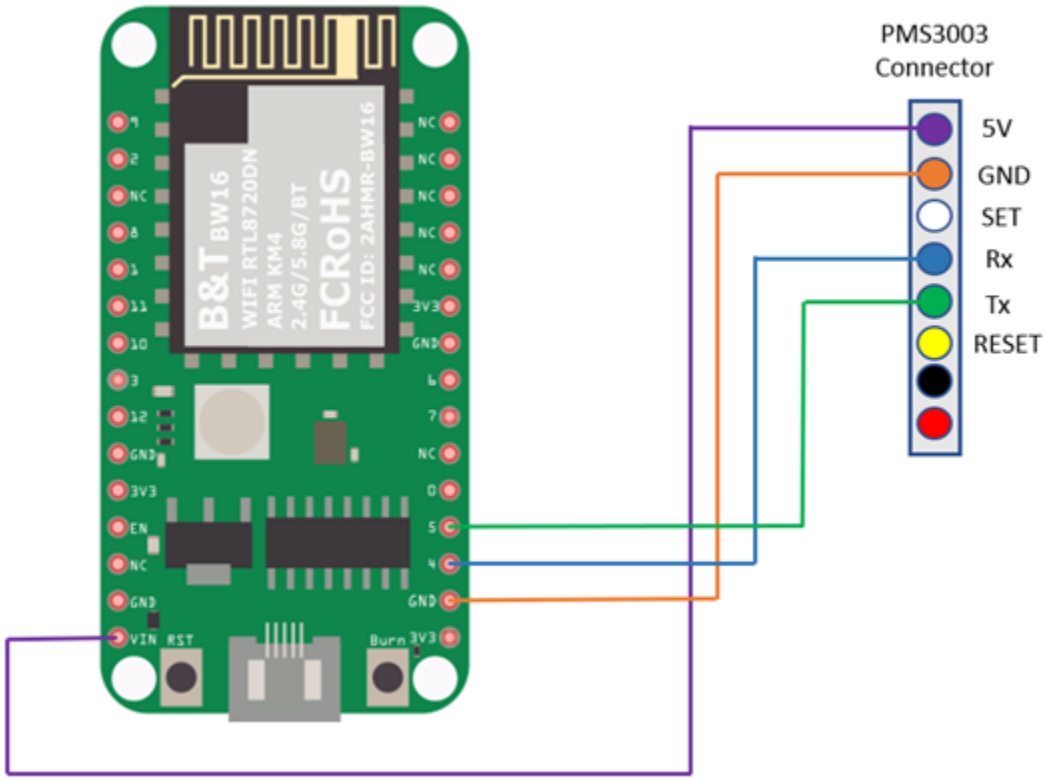
Then connect sensors according to UART-PlanTower PMS3003 wiring example.
AMB21 / AMB22:


PMS5003
Connector
5V
GND
SET
RX
TX
RESET
RESET
UART
DOWNLOAD

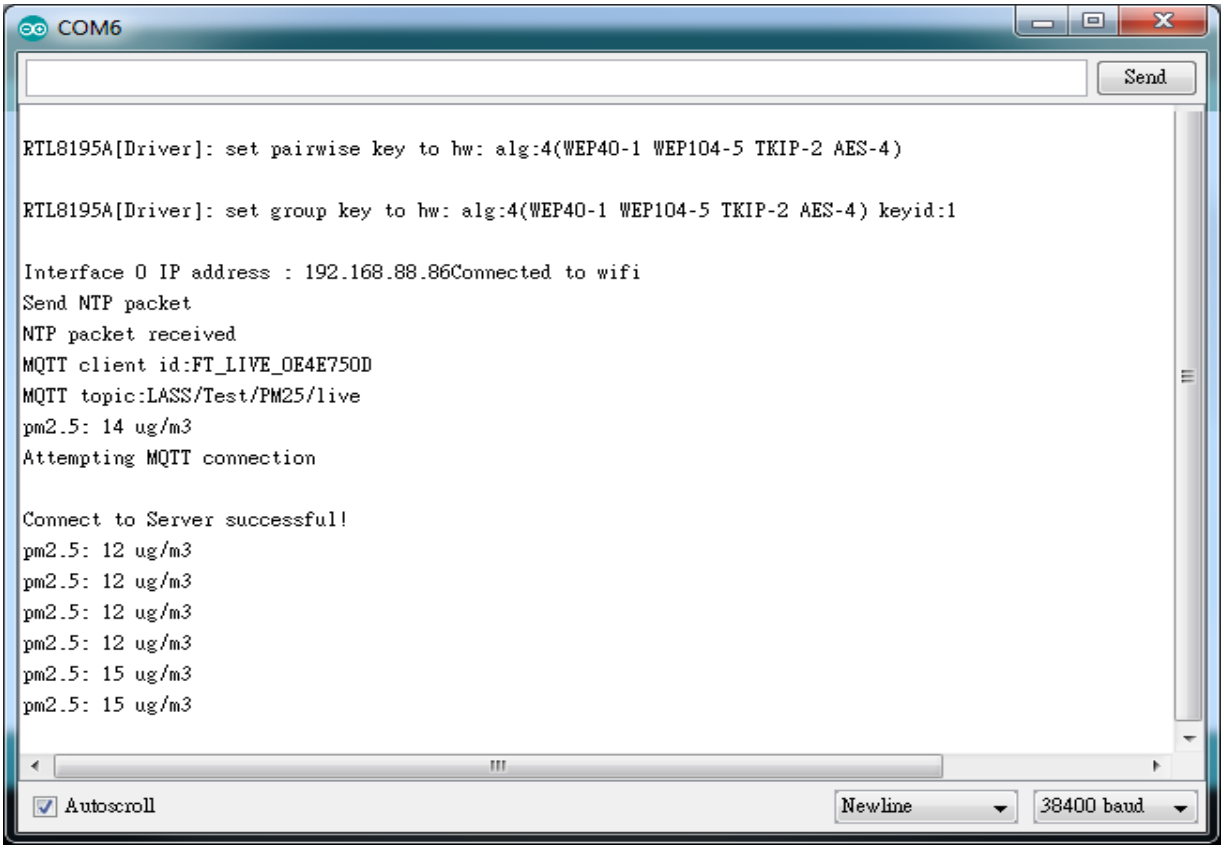
AMB23:


PMS3003
Connector
5V
GND
SET
Rx
Tx
RESET
9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

BW16:



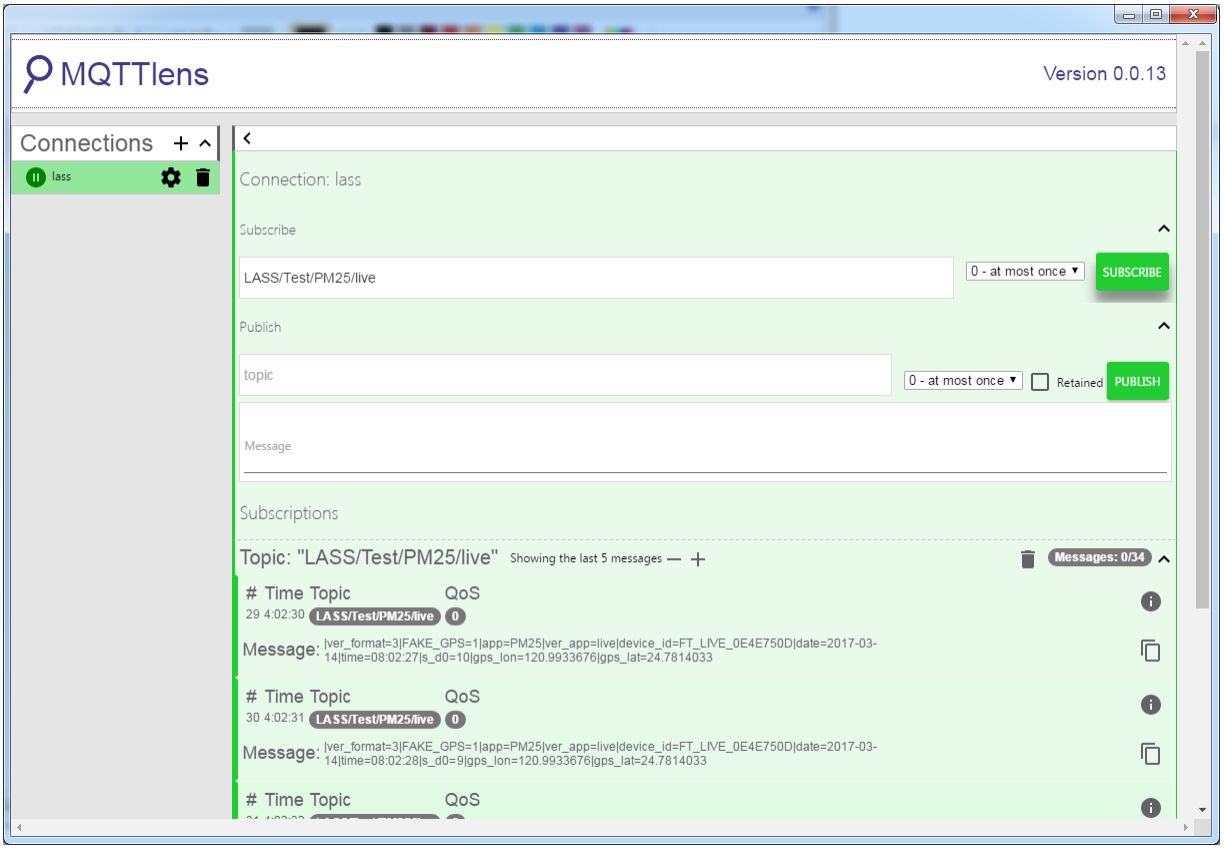
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba will attempt to read PM2.5 data every minute and upload it to LASS MQTT-Broker. Open Serial Monitor to see the uploaded data, including client id, topic, and current PM2.5 status.



We can also use MQTTlens to verify if the data is properly uploaded.

Enter “gpssensor.ddns.net” as the MQTT-Broker server and “LASS/Test/PM25/live” as the subscribe topic to receive data.

The time uses UTC format, and the PM2.5 data stores in s-d0. In the figure, s_d0 = 9 represents that the PM2.5 is 9, meaning that the entire publish/subscribe process is working successfully.



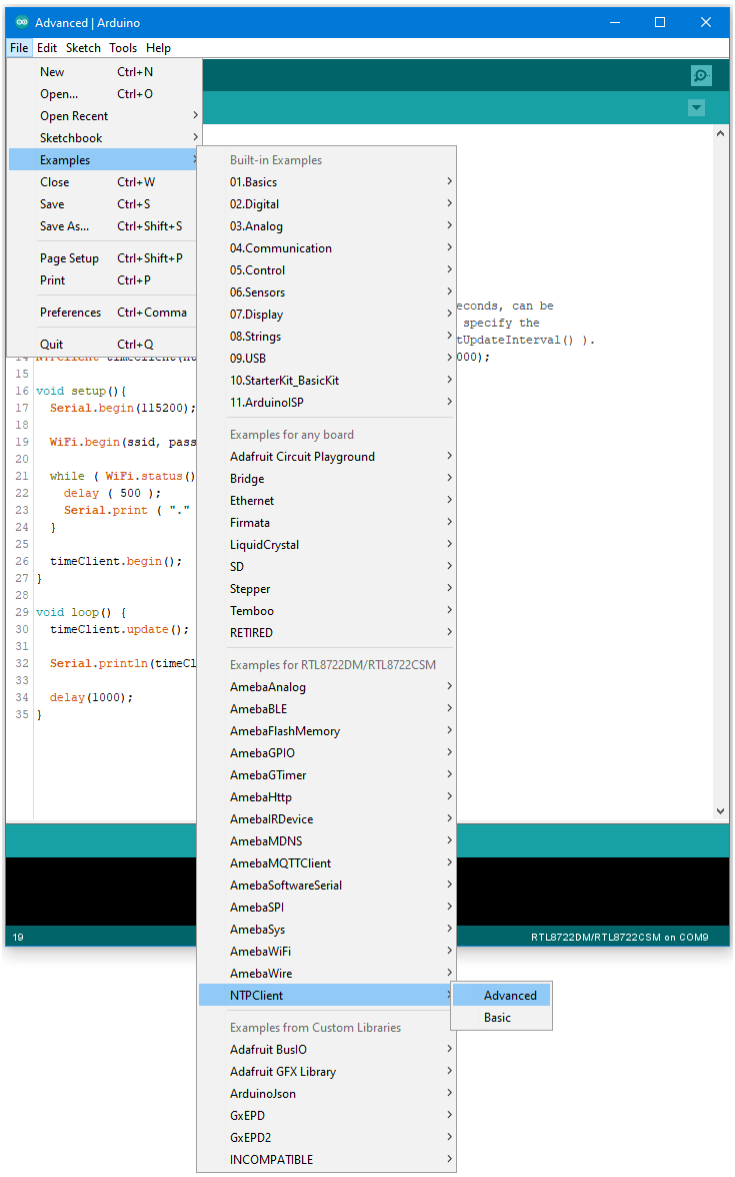
NTP - Retrieve Universal Time (UTC) by NTPClient library

Preparation

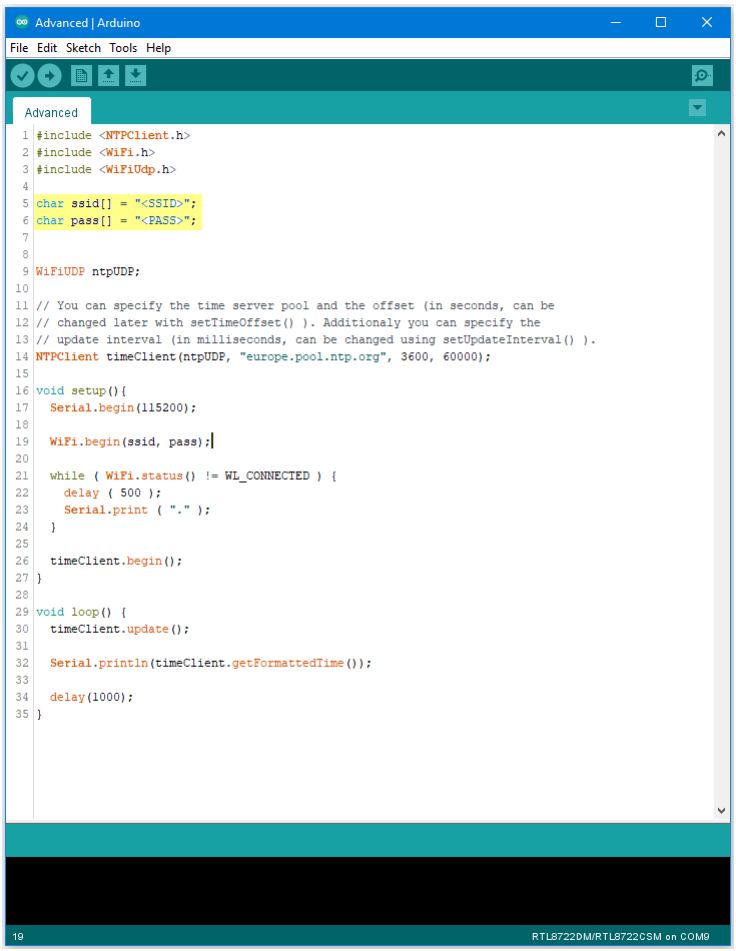
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

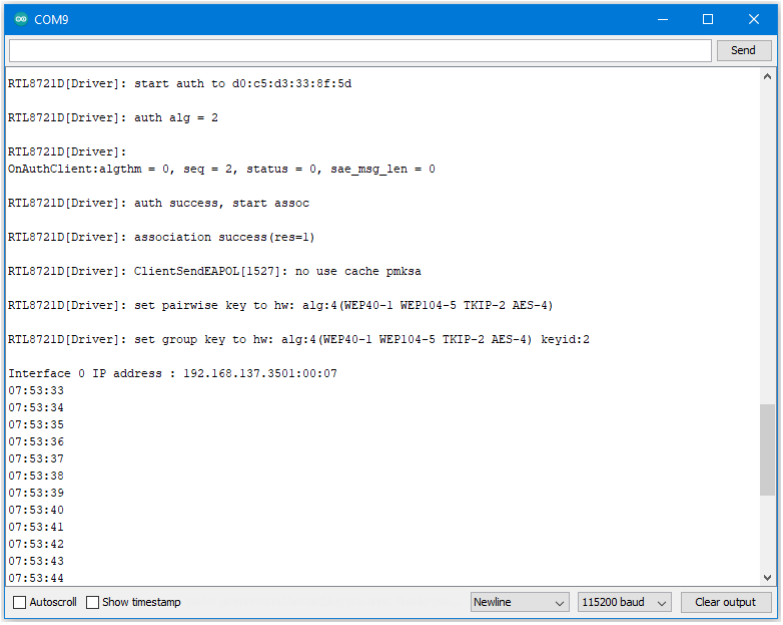
In this example, we use an NTP client to sync with NTP servers using UDP and keep track of time locally. Open the example. “`File” -> “Examples”-> “NTPClient” -> “Advanced`”



Modify the highlighted code section (ssid, password) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi, gets the UTC time from the NTP server, and prints out the current time with time zone offset to the serial monitor.



Code Reference

Configure NTP client:

The NTPClient needs to use a UDP client for communications. A WiFiUDP client is declared and passed to the NTPClient constructor, along with an NTP server address, time zone offset in seconds, and update interval in milliseconds. If detailed configuration is not needed, just passing in the UDP client is also sufficient, refer to the "NTPClient" -> "Basic" example.

```
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 3600, 60000);
```

Start NTP client:

After connecting to WiFi, the NTPClient is started using the `begin()` function, which causes the client to sync with the NTP server and get the UTC time.

```
WiFiUDP ntpUDP;
timeClient.begin();
```

Get local time:

`getFormattedTime()` is used to format the received UTC time into the local time zone. `update()` is called every loop so that the NTPClient will sync with the NTP server once every update interval.

```
timeClient.update();
timeClient.getFormattedTime();
```

WiFi - Approximate UDP Receive Delay

Materials

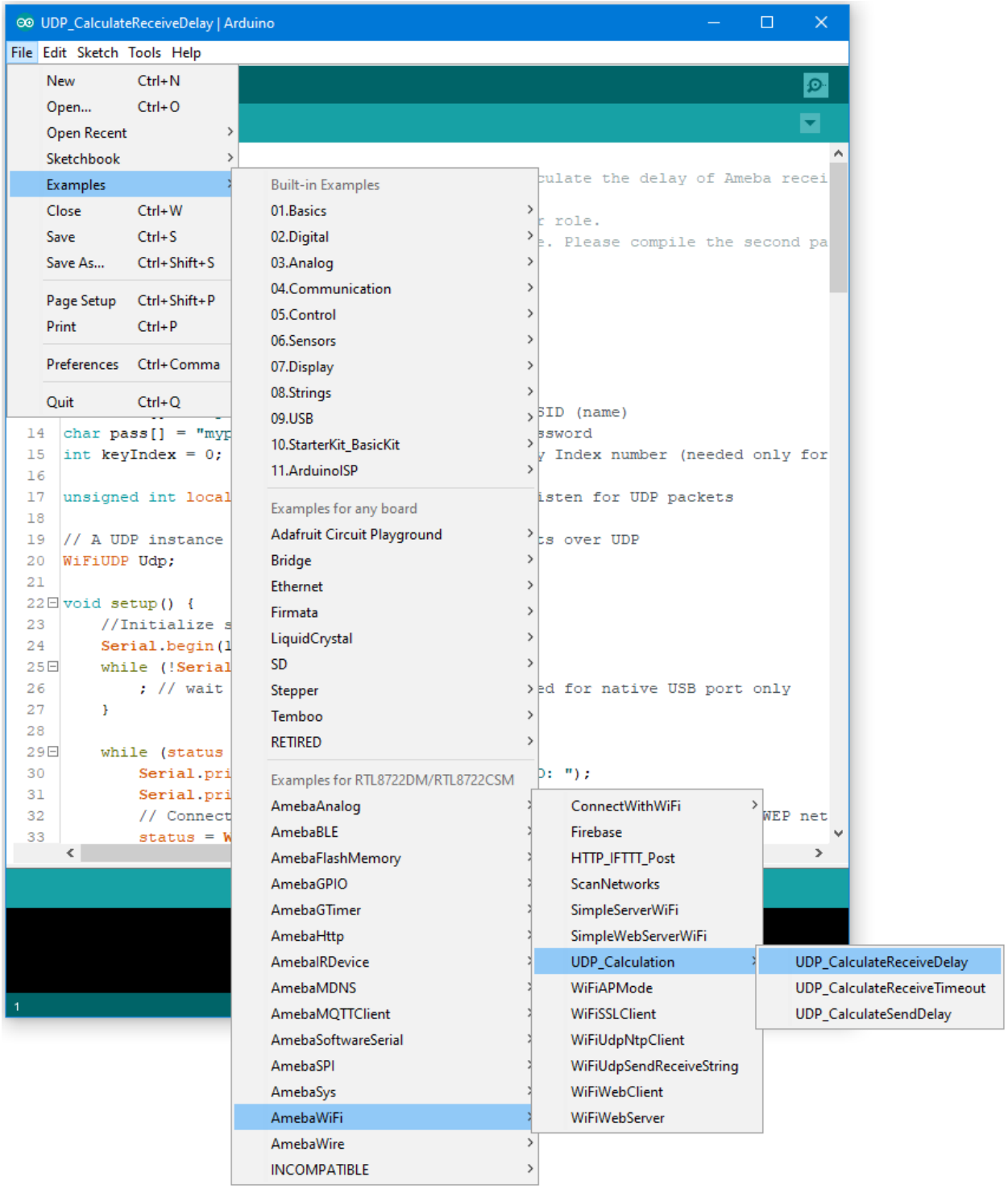
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

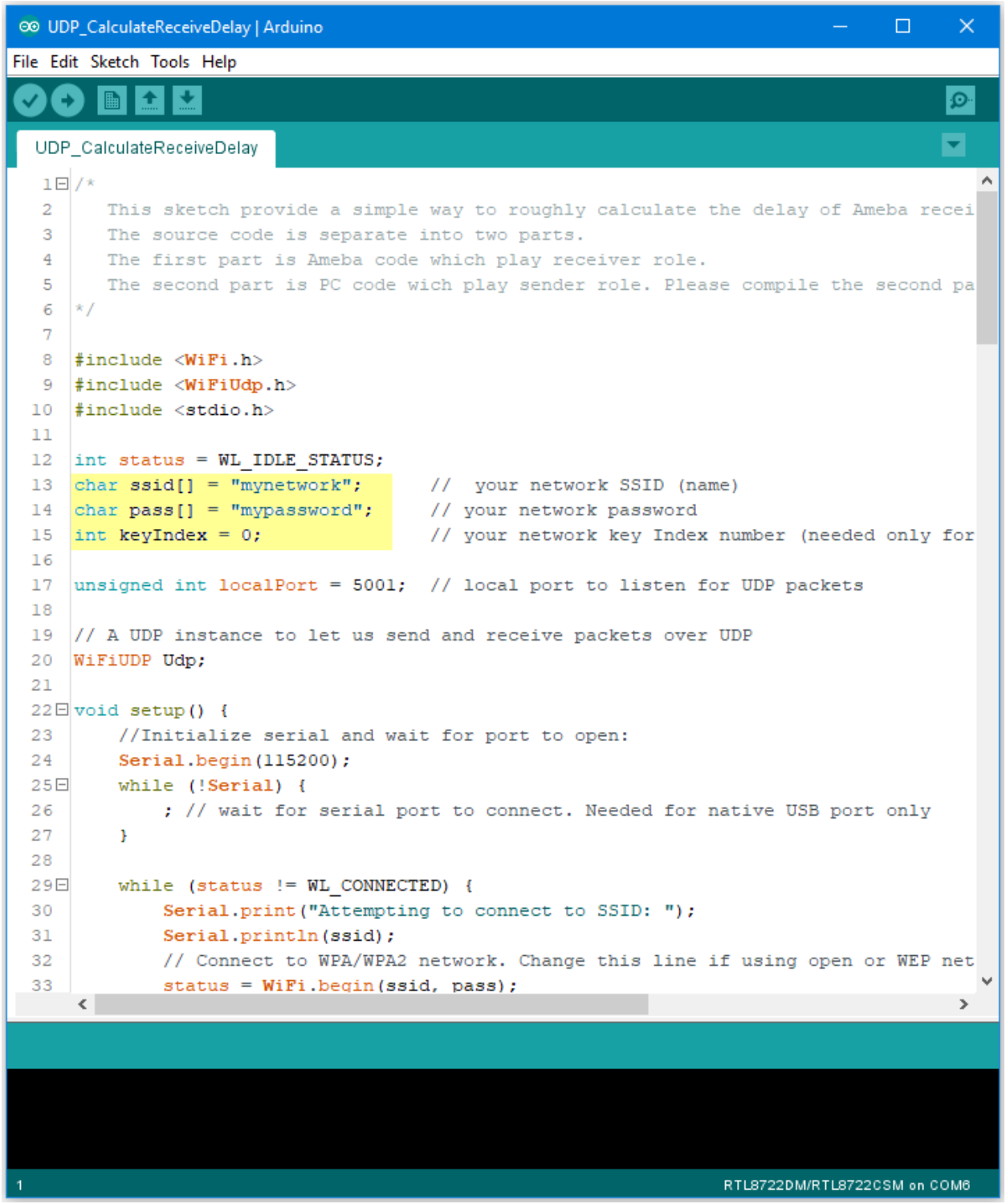
This example uses Ameba to receive UDP packets from a computer and calculates the UDP receive delay.

Ameba Preparation

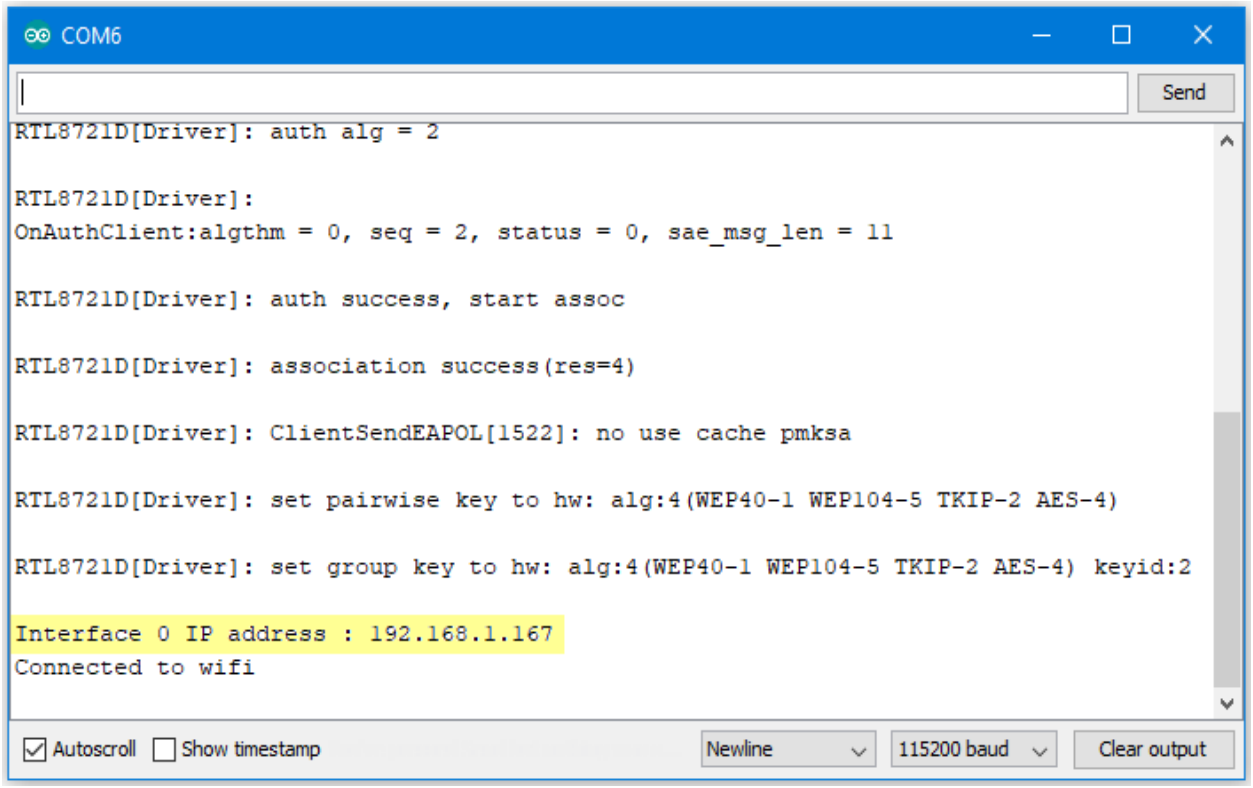
Open the "CalculateUdpReceiveDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation`" -> "`CalculateUdpReceiveDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



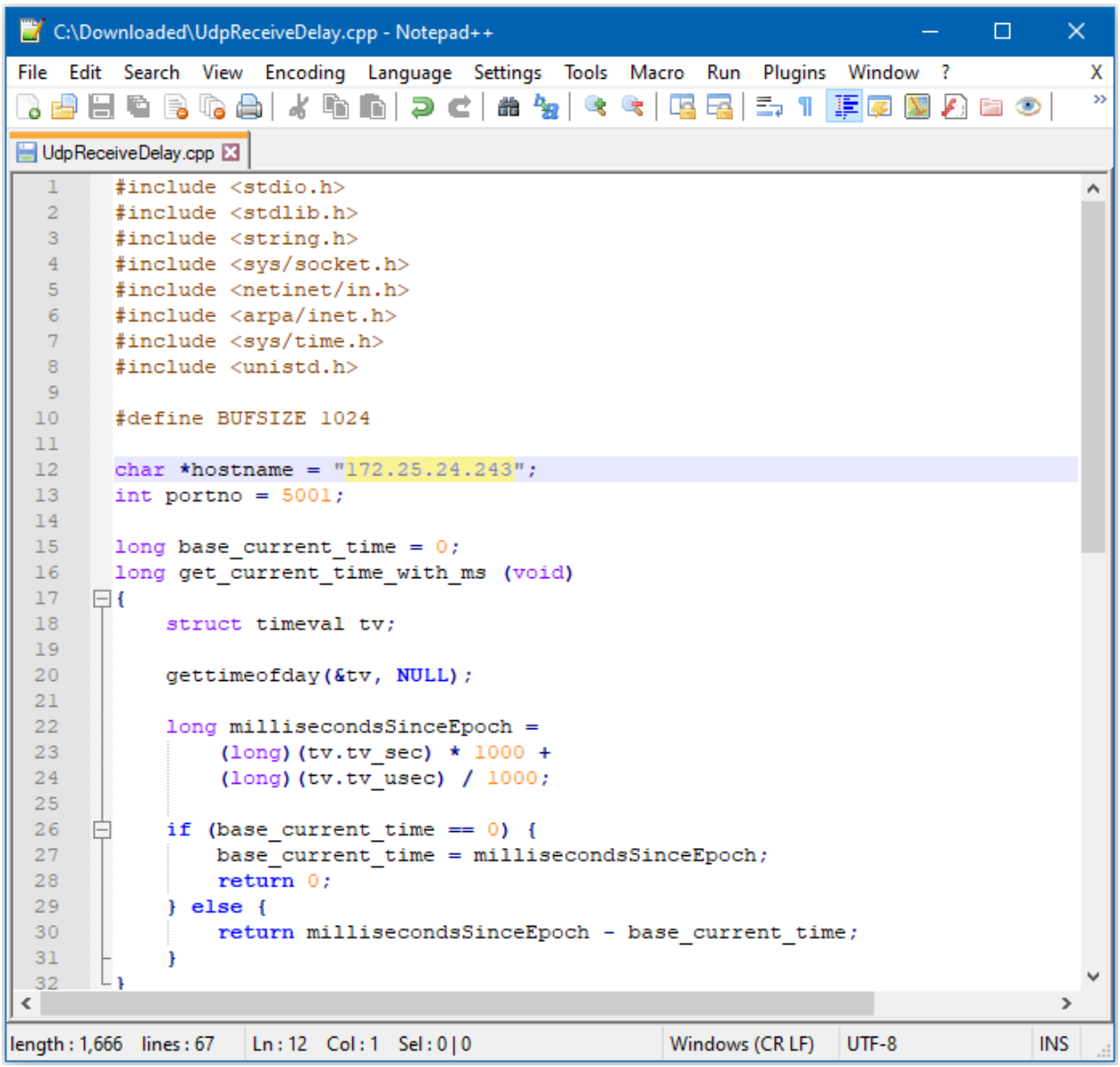
Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

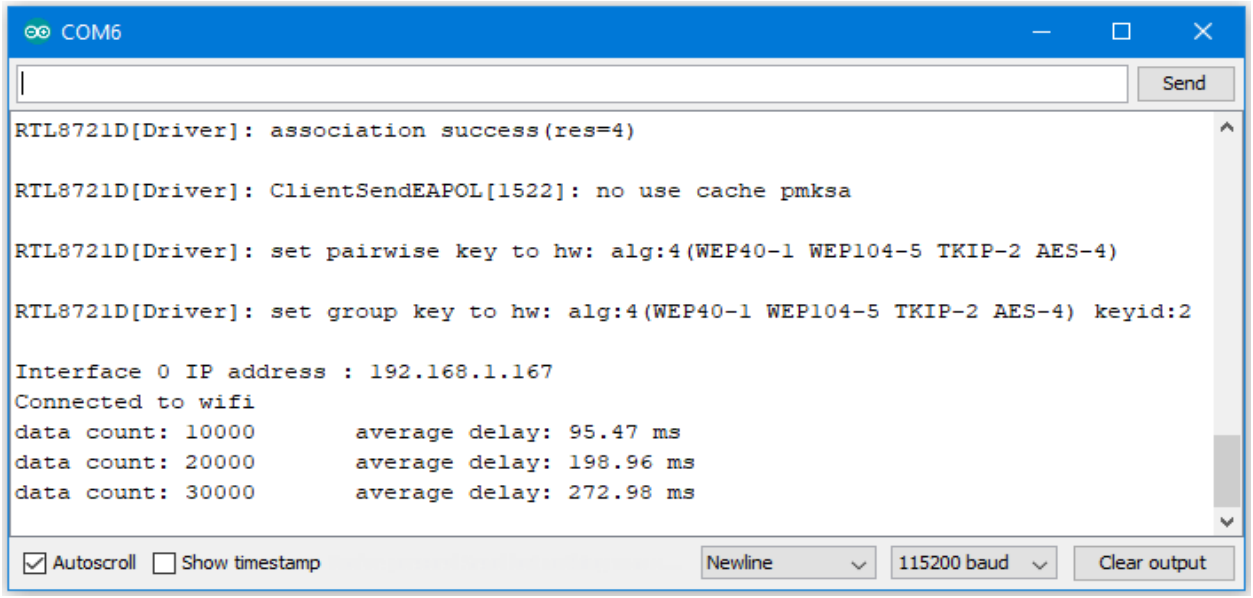
Follow the instructions there to install it. Next, from the "CalculateUdpReceiveDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveDelay.cpp", and use the command "g++ UdpReceiveDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpDelay.exe file, and the computer will begin to send packets to Ameba. Once 10000 packets have been received, Ameba will calculate the average delay and print out the result to the serial monitor. It may take up to a few minutes for 10000 packets to be sent.



WiFi - Approximate UDP Sending Delay

Materials

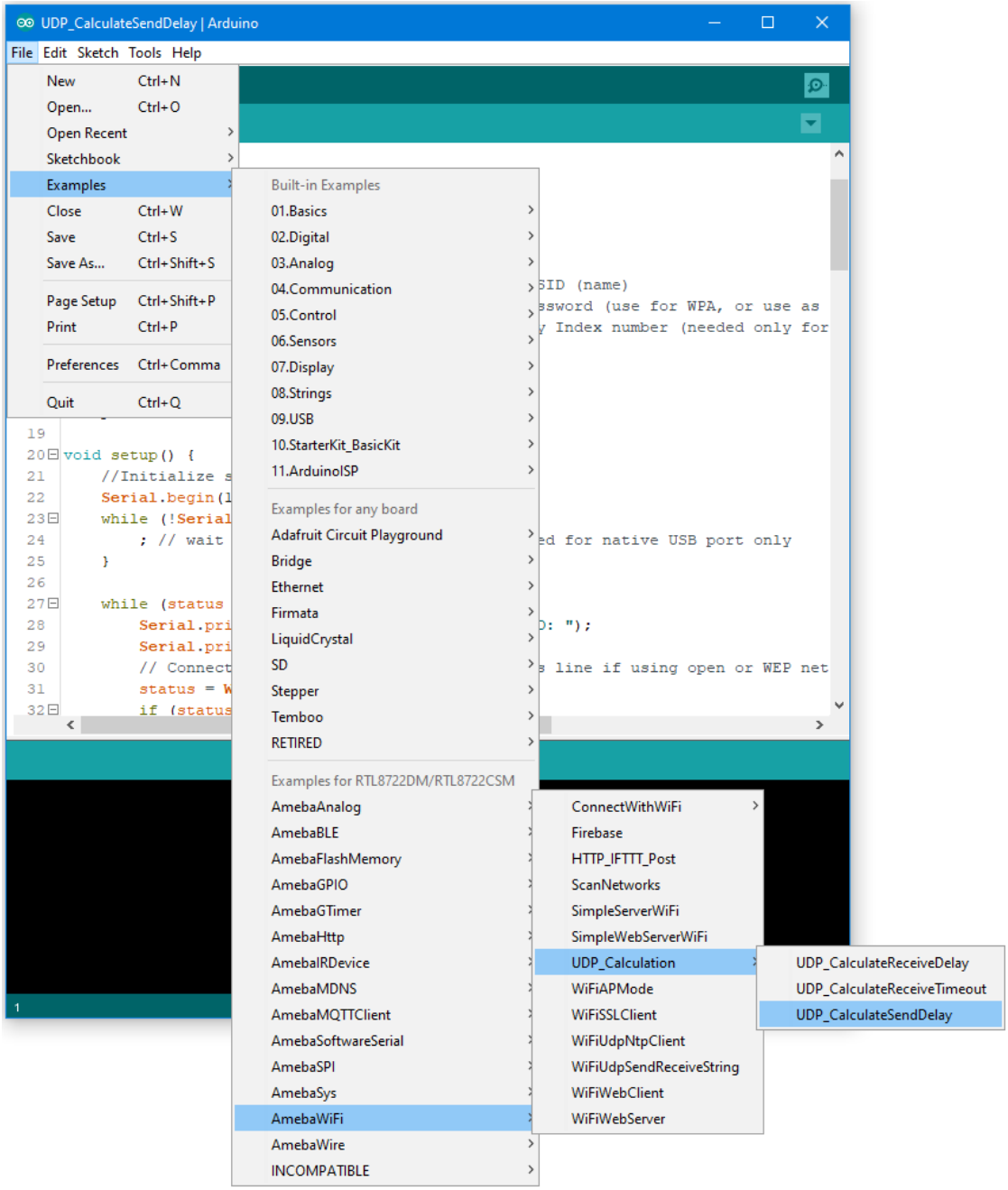
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

This example uses Ameba to send UDP packets to a computer and calculates the UDP sending delay.

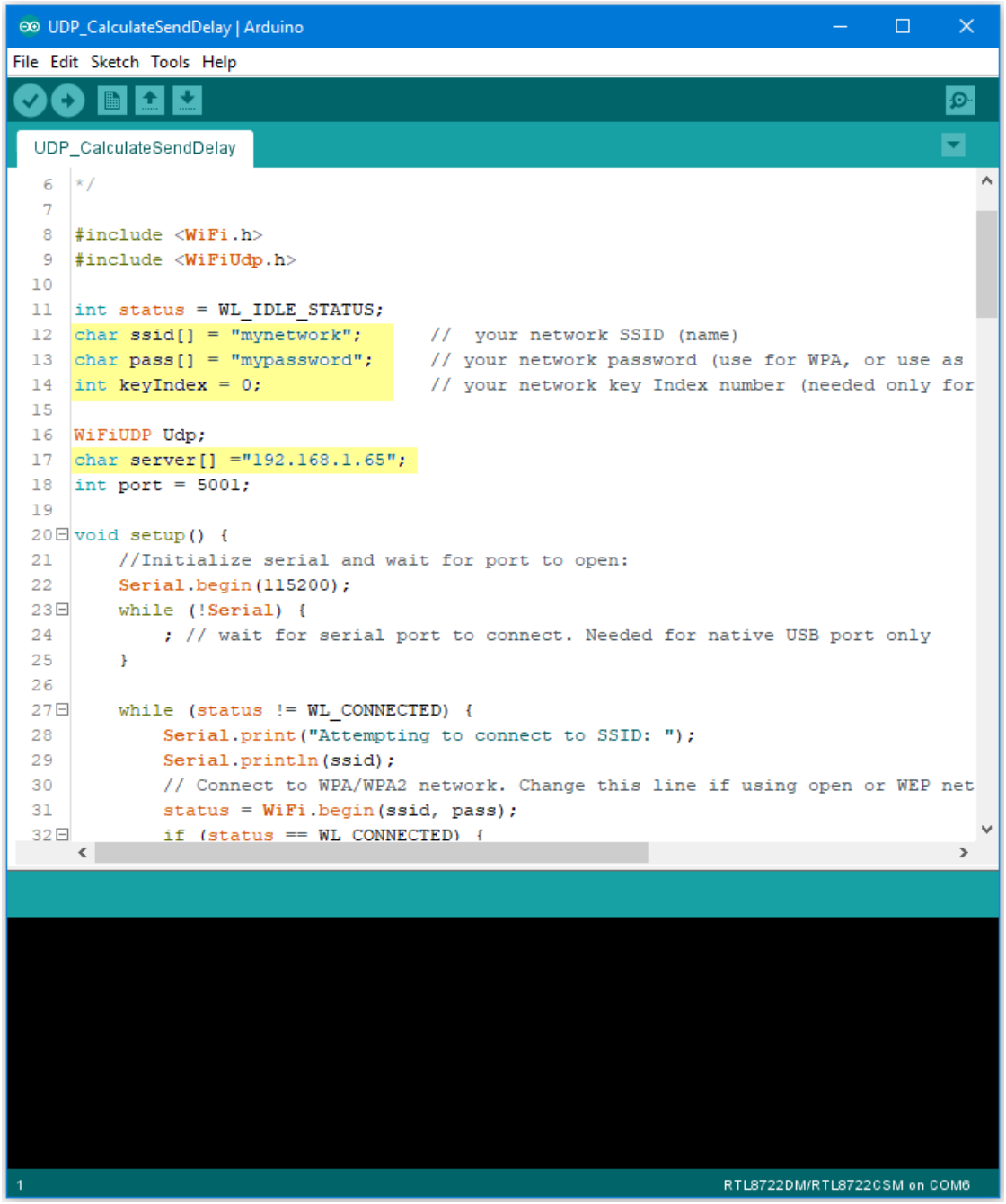
Ameba Preparation

Open the "CalculateUdpSendDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation`" -> "`CalculateUdpSendDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.

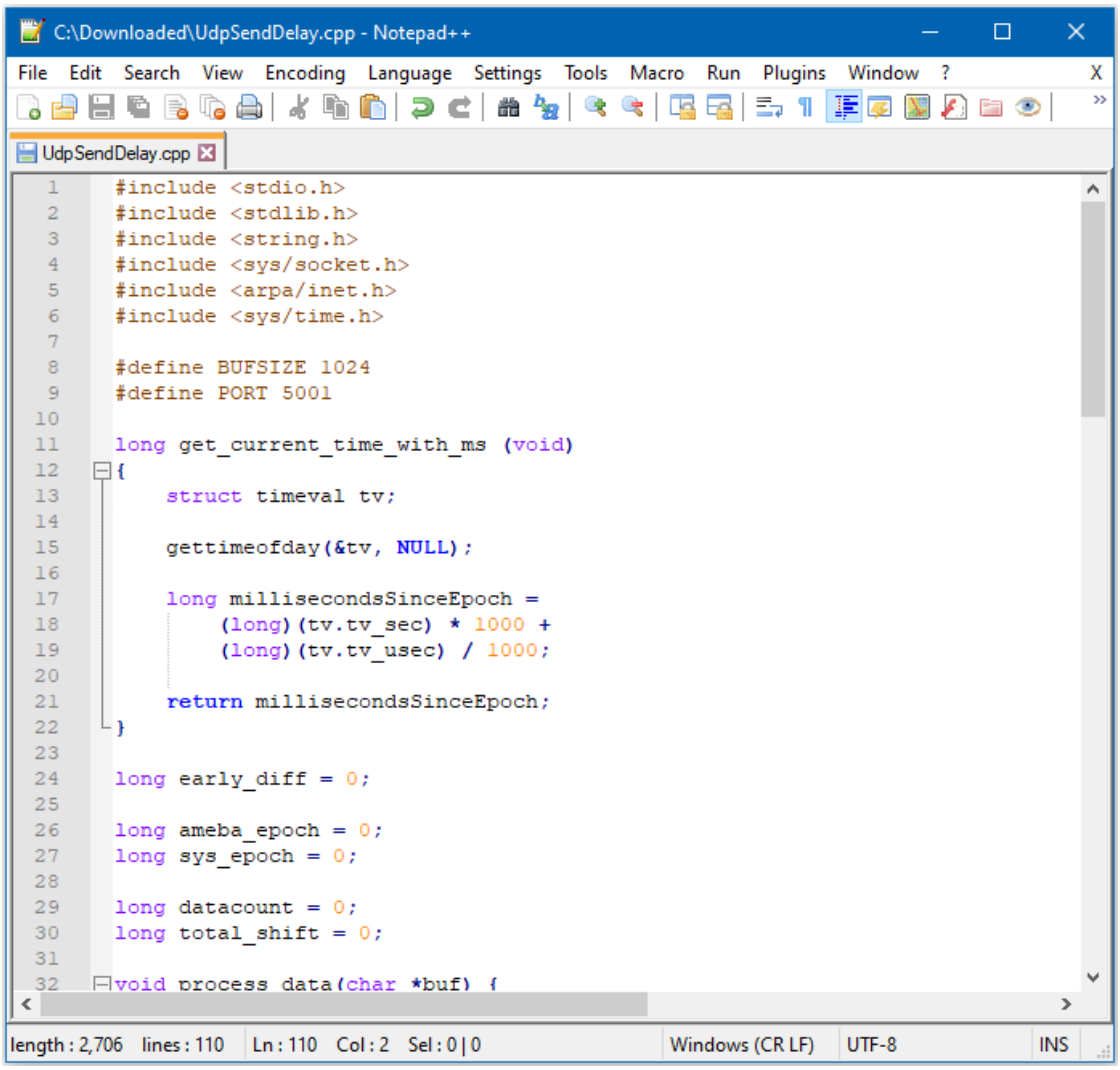
The server variable also needs to be changed to match the IP address of your computer. You can find the IP address using the “ipconfig” command in a terminal window.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpSendDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file and rename the file to "UdpSendDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpSendDelay.cpp", and use the command "g++ UdpSendDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

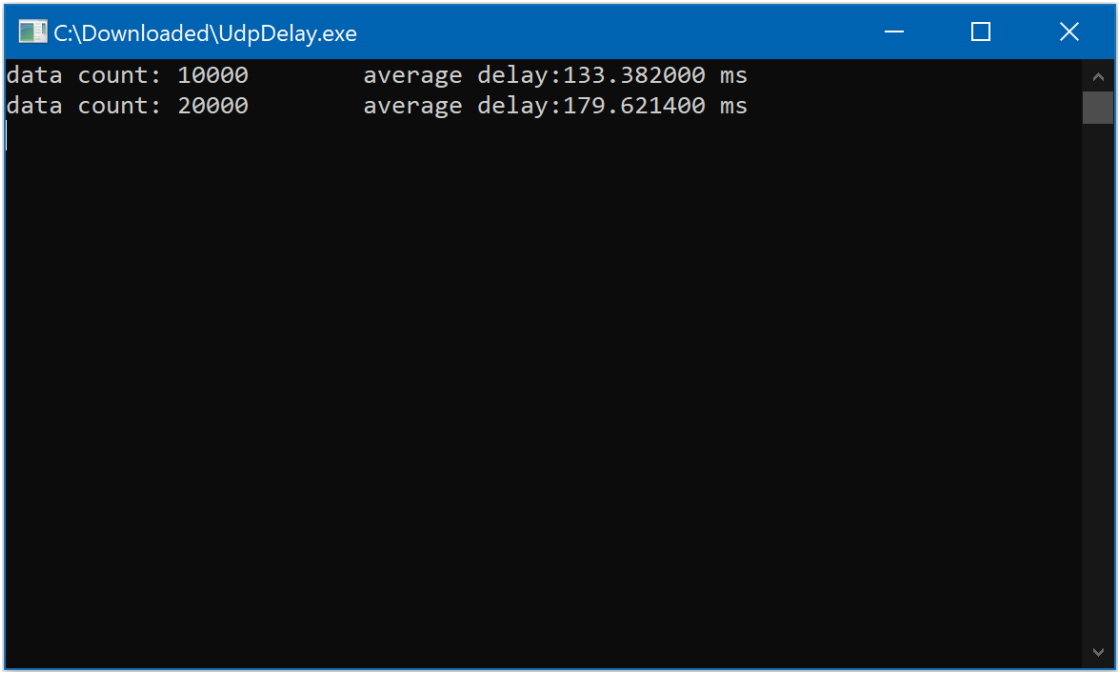
Running the Example

First, on the computer, run the UdpDelay.exe file, and the computer will begin to listen for packets from Ameba.

Next, compile and upload the code from the Arduino IDE to Ameba and press the reset button when the upload is complete.

The Ameba will begin to send UDP packets to the computer. Once 10000 packets have been received, the computer will calculate the average delay and print out the result.

It will take some time for 10000 packets to be sent.



WiFi - Approximate UDP Receive Timeout

Materials

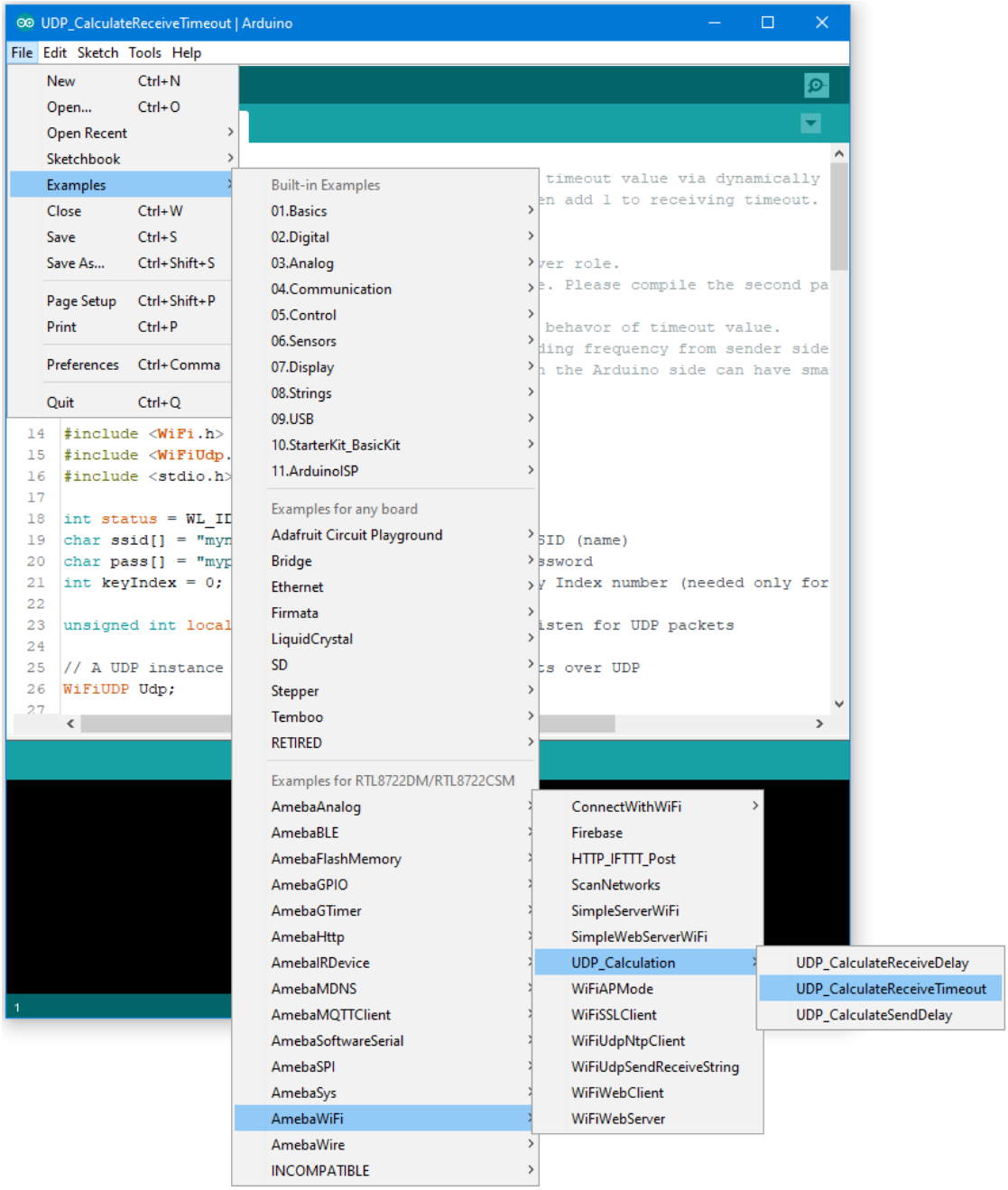
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

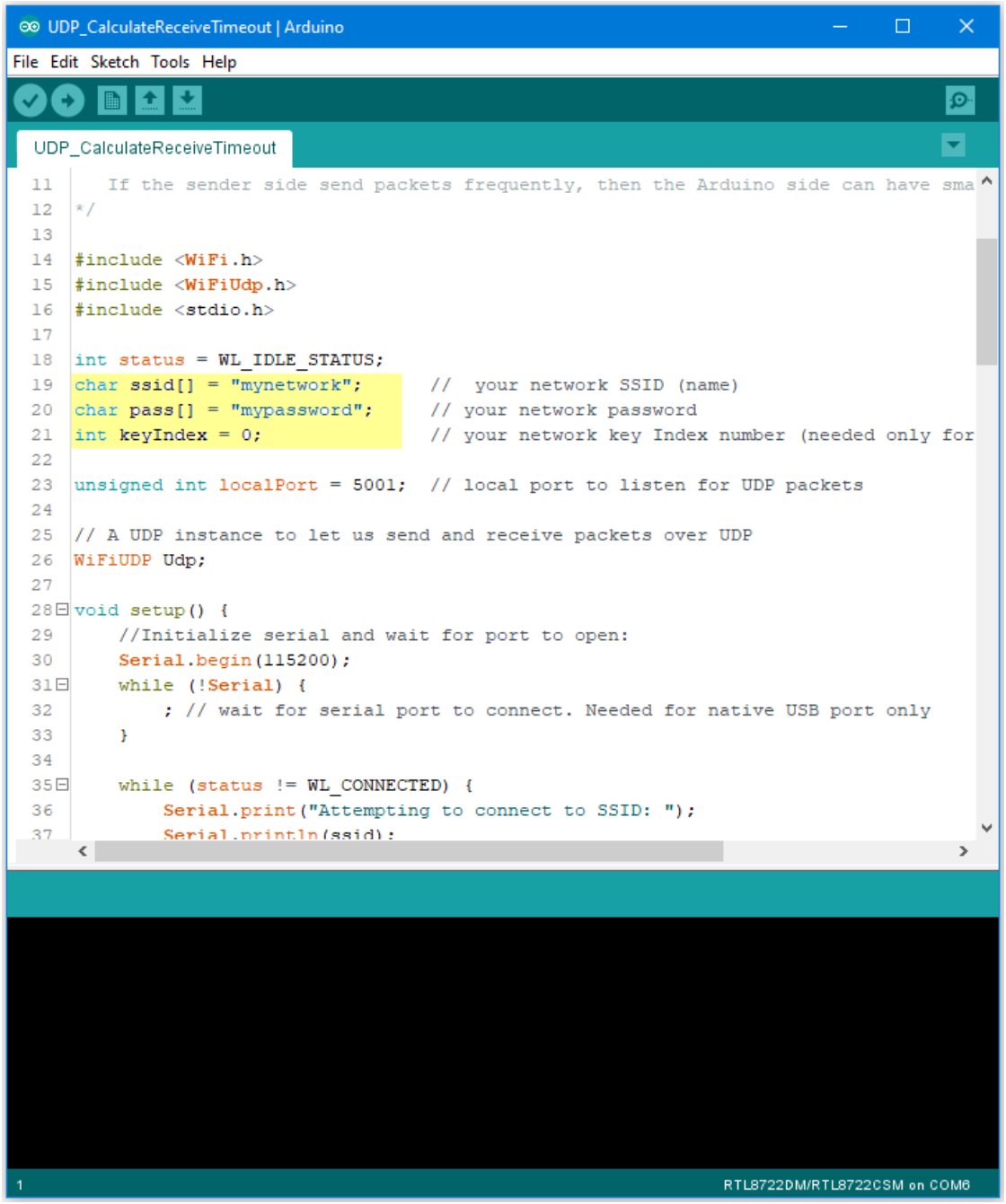
This example uses Ameba to receive UDP packets from a computer and calculates the allowed UDP receive timeout setting.

Ameba Preparation

Open the "CalculateUdpReceiveTimeout" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation `" -> "`CalculateUdpReceiveTimeout`".

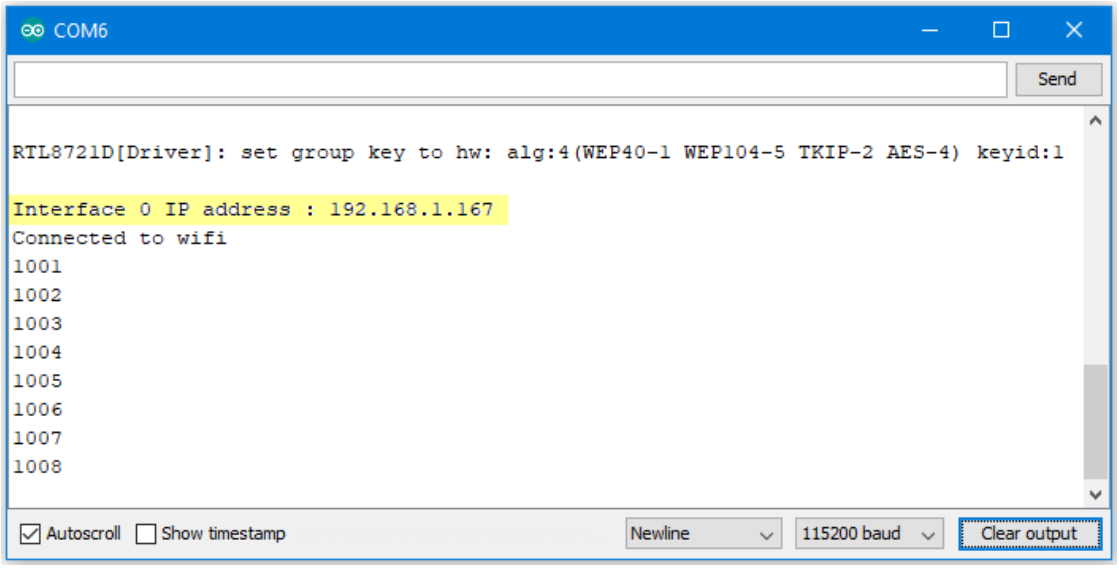


In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished.

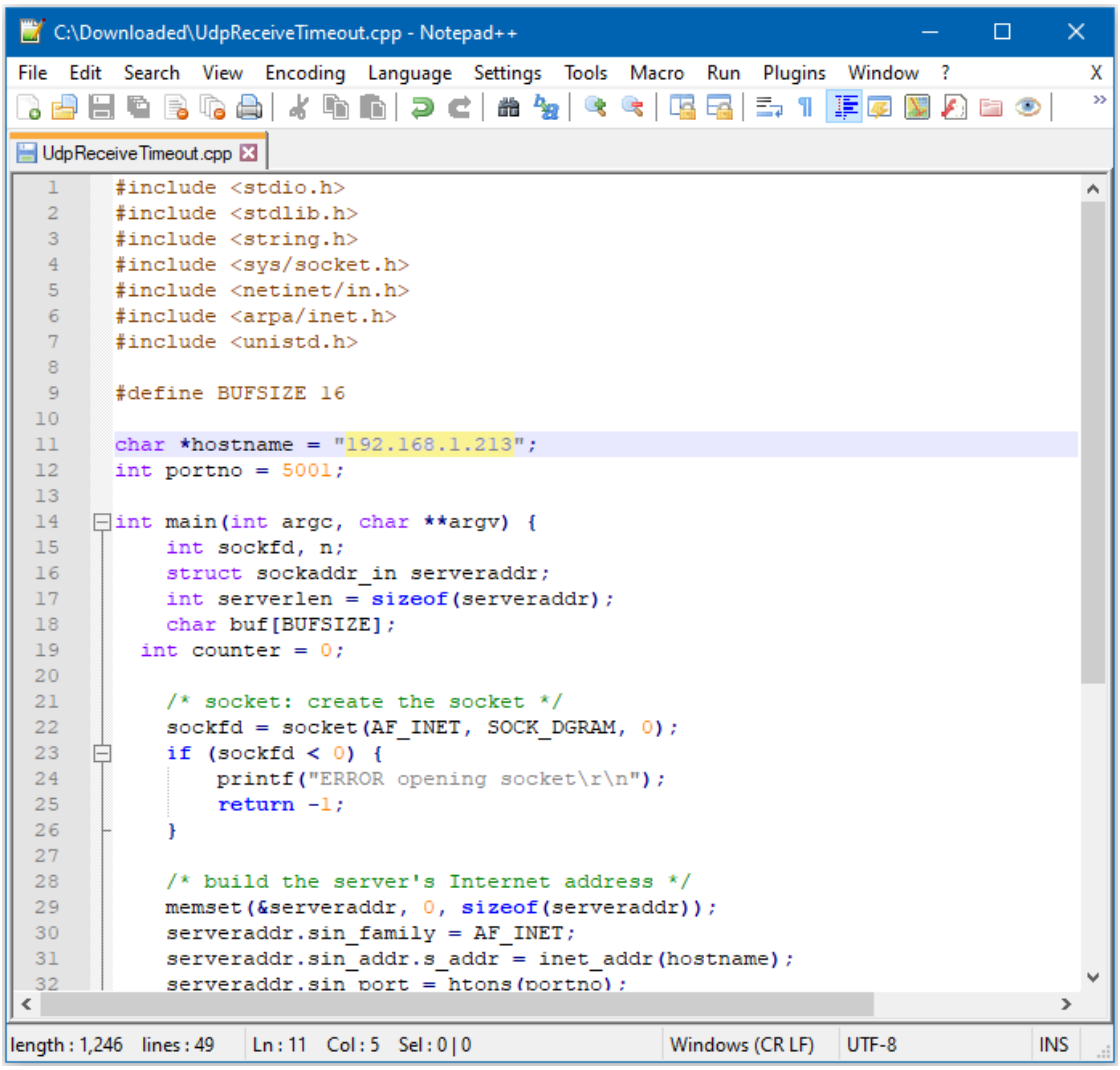
Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpReceiveTimeout" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveTimeout.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveTimeout.cpp", and use the command "g++ UdpReceiveTimeout.cpp -o UdpTimeout" to compile the code. A file named "UdpTimeout.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpTimeout.exe file, and the computer will begin to send packets continuously to Ameba.

The timeout value is set to 1000ms initially. For each packet received successfully, Ameba decreases the timeout value. The next packet must be received within the timeout period, otherwise Ameba registers a failed packet and increases the timeout value. Open the serial monitor and observe the timeout value converge to a minimum value.

WiFi - Connect to WiFi networks

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Procedure

There three common encryption type in WiFi connection. The first one is “OPEN”, which means there is no password needed to connect to this network. The second type of encryption is WPA, which requires the correct password to access. The third type is WEP, which requires a hexadecimal password and a keyindex.

In the following, we will give a brief introduction on how to establish WiFi connection with these three types of encryption on Ameba.

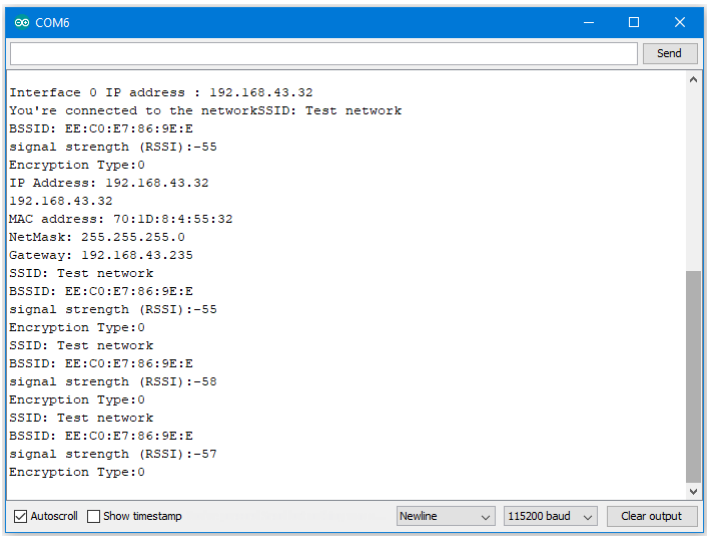
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”.

- Open (WiFi connection without password)

  Open the “ConnectNoEncryption” example in “File” -> “Examples” -> “AmebaWiFi” -> “ConnectWithWiFi” -> “ConnectNoEncryption”

  In the sample code, modify “ssid” to be the same as the WiFi SSID to be connected to.

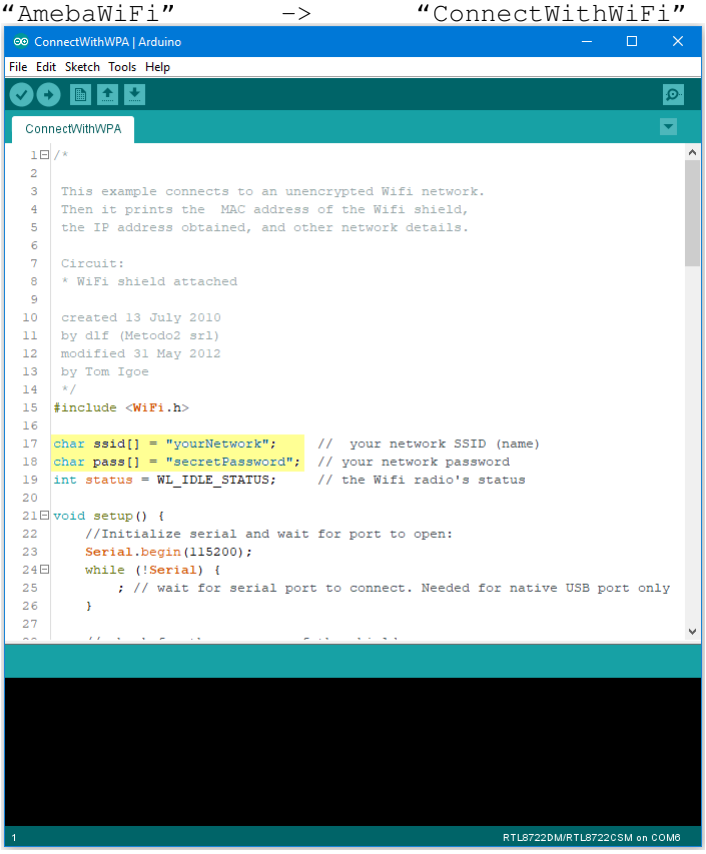
  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message “You’re connected to the networkSSID: XXXXX”, and the information of this WiFi connection is printed in the



serial monitor every 10 seconds.

- WiFi connection with WPA encryption

  Open the "ConnectWithWPA" example in "File" -> "Examples" -> "AmebaWiFi" -> "ConnectWithWiFi" -> "ConnectWithWPA"



  In the sample code, modify "ssid" to the WiFi SSID to be connected to and "pass" to the network password.

  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message "You're connected to the networkSSID: XXXXX", and the information of this WiFi connection is printed in the serial monitor every 10 seconds.

- WiFi connection with WEP encryption

Open the “ConnectWithWEP” example in “File” -> “Examples” -> “AmebaWiFi” -> “ConnectWithWiFi” -> “ConnectWithWEP”

In the sample code, modify “ssid” to the SSID to be connected, “key” to the hexadecimal password, “keyIndex” to your key index number.

Next, upload the sample code, and press the reset button on Ameba. Then you will see a message “You’re connected to the networkSSID: XXXXX”, and the information of this WiFi connection is printed in the IDE every 10 seconds.

Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.encryptionType()` to get the encryption type of the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiEncryptionType
Use `WiFi.BSSID()` to get the MAC address of the router you are connected to.
https://www.arduino.cc/en/Reference/WiFiBSSID
To get the information of Ameba:
Use `WiFi.macAddress()` to get the MAC address of Ameba.
https://www.arduino.cc/en/Reference/WiFiMACAddress

Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFi.subnetMask()` to get the subnet mask.
https://www.arduino.cc/en/Reference/WiFiSubnetMask
Use `WiFi.gatewayIP()` to get the WiFi shield's gateway IP address.
https://www.arduino.cc/en/Reference/WiFiGatewayIP

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

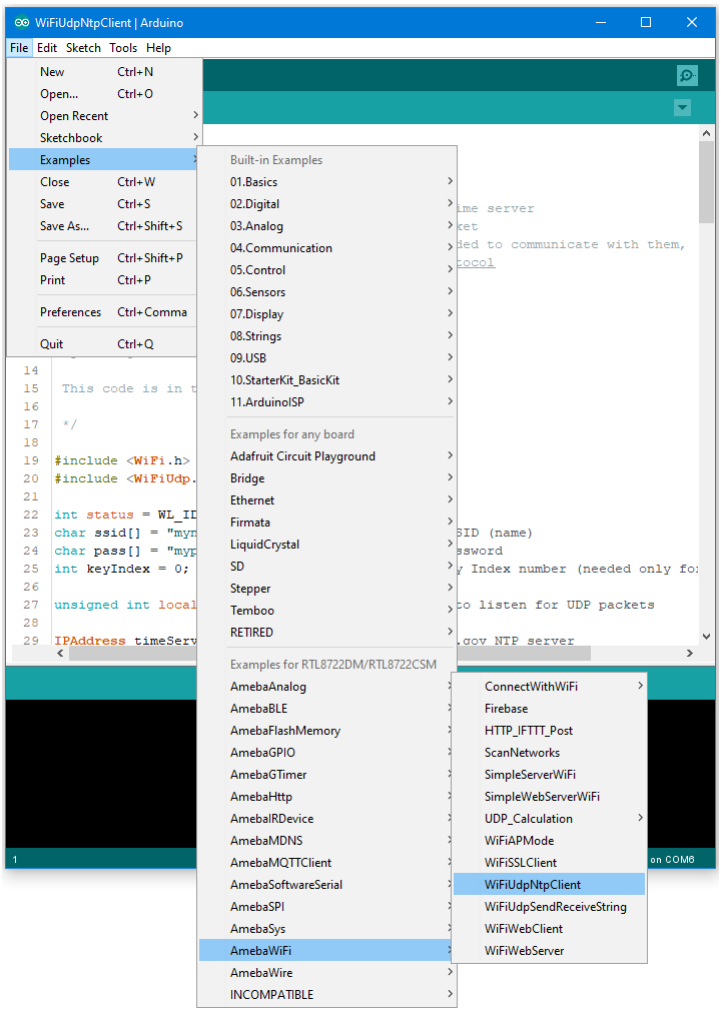
WiFi - Retrieve Universal Time (UTC) by UDP

Preparation

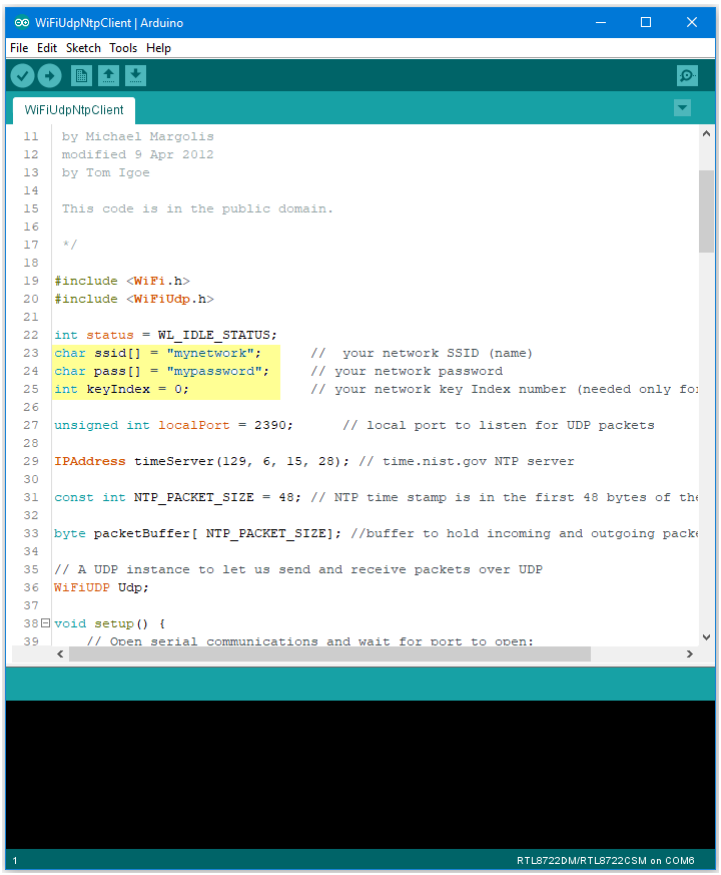
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to WiFi. Then send NTP (Network Time Protocol, RFC 1305) request to NTP server using UDP. After receiving the NTP request, the NTP server replies current UTC (Coordinated Universal Time) packet. We will parse the UTC packet to show current UTC time in the serial monitor.
Open the example: `"File" -> "Examples" -> "AmebaWiFi" -> "WiFiUdpNtpClient"`

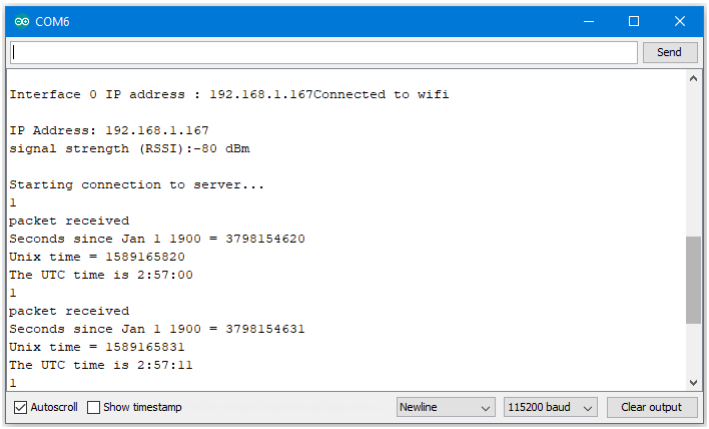


Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and sends NTP request packet to NTP server “129.6.15.28”.

We parse the replied packet and show UTC time in serial monitor:



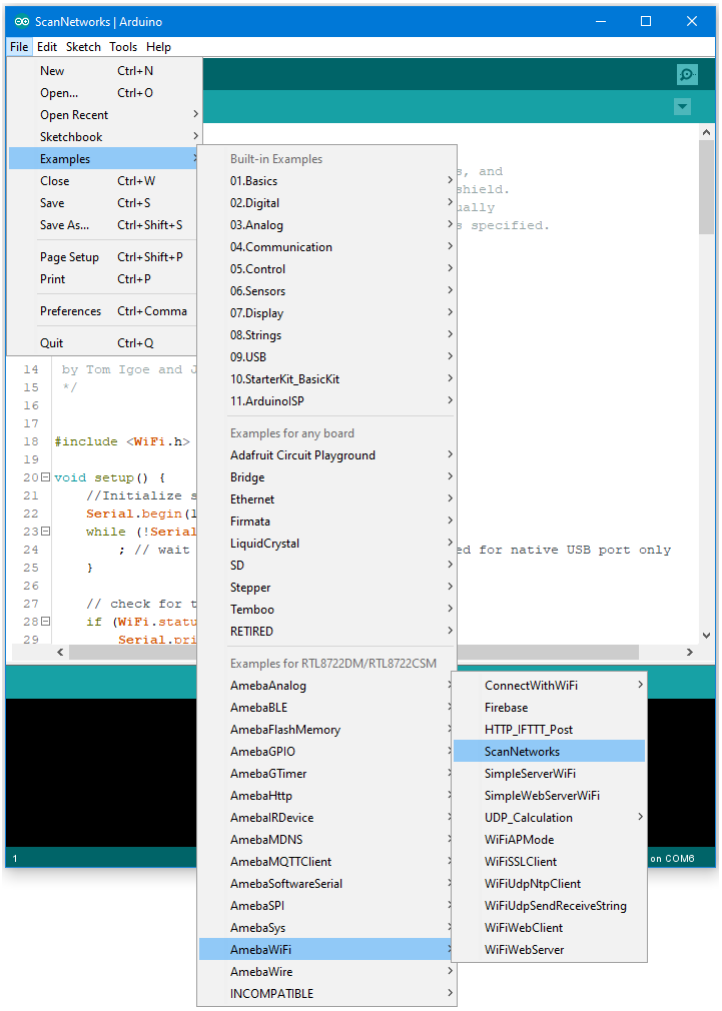
WiFi - Scan the surrounding WiFi networks

Materials

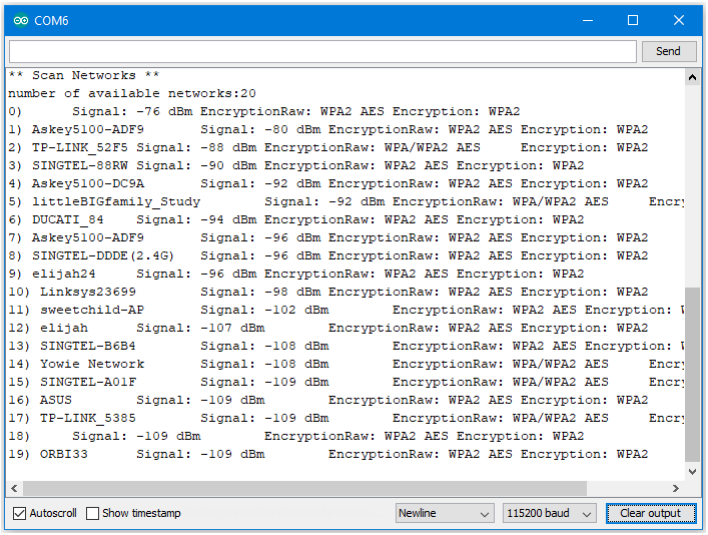
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Antenna x 1
- **AmebaD [:raw-html:`<p style="color:#1A76B4;">AMB21(RTL8722DM/CSM)</p>` / AMB23(RTL8722DM_MINI) / BW16(RTL8729DN)] x 1**

Example

In this example, we use Ameba to scan available WiFi hotspots in the surroundings, and prints the SSID, encryption type, signal strength information of each detected hotspot.
First, make sure the correct Ameba development board is selected in Arduino IDE: "`Tools`" -> "`Board`"
Open the "ScanNetworks" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`ScanNetworks`":



Then upload the sample code and press the reset button on Ameba. Afterwards, you can see "**Scan Networks**" message appears, with the detected WiFi hotspots and the information of each hotspot.



Code Reference

- First we use `WiFi.macAddress(mac)` to get the MAC address of Ameba: https://www.arduino.cc/en/Reference/WiFiMACAddress
- Then we use `WiFi.scanNetworks()` to detect WiFi hotspots: https://www.arduino.cc/en/Reference/WiFiScanNetworks
- To get information of detected WiFi hotspot: We use `WiFi.SSID(thisNet)` to retrieve SSID of a network: https://www.arduino.cc/en/Reference/WiFiSSID We use `WiFi.RSSI(thisNet)` to get the signal strength of the connection to the router: https://www.arduino.cc/en/Reference/WiFiRSSI
- We use `WiFi.encryptionType(thisNet)` to get the encryption type of the network: https://www.arduino.cc/en/Reference/WiFiEncryptionType

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

WiFi - Set up Server to communicate

Materials

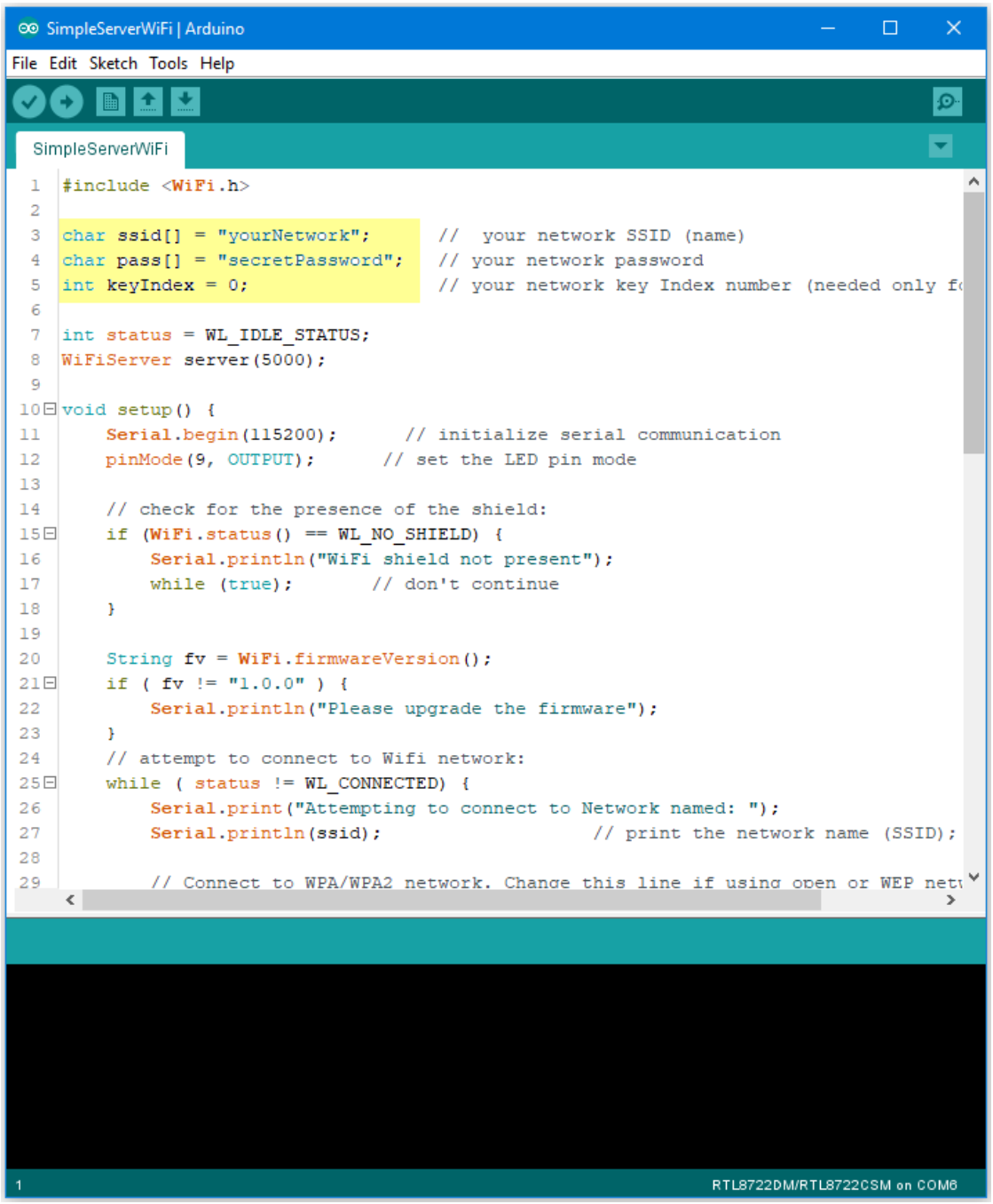
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Laptop□Make sure it is connected to the same network domain as Ameba, and tcp tools are installed.□

Example

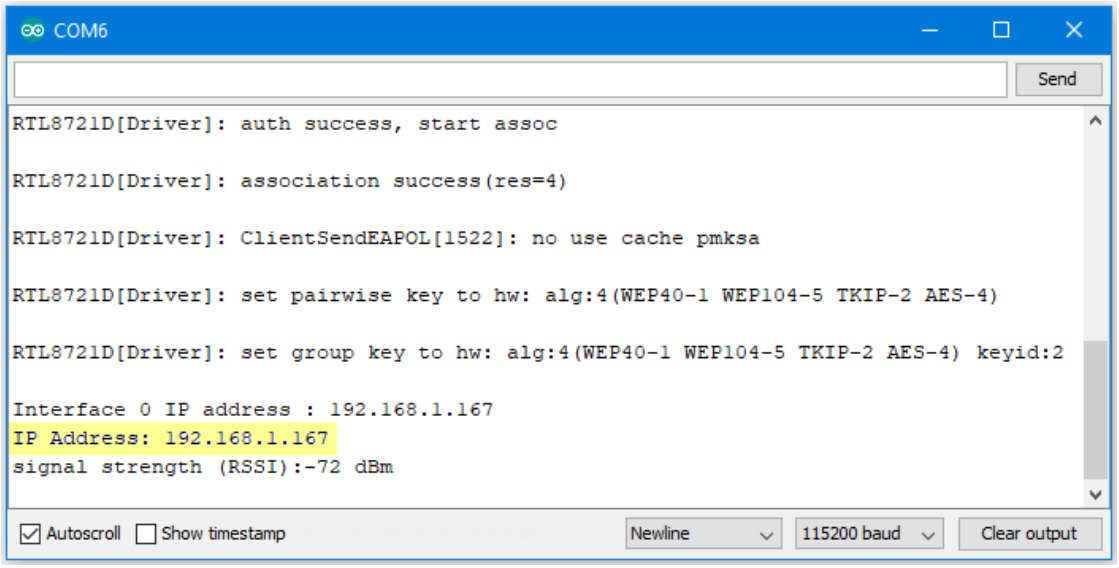
In this example, we first connect Ameba to WiFi, then we use Ameba as server to communicate with client.

First, we make sure the correct Ameba development board is set in “Tools” -> “Board”

Then, open the Simple WiFi Server example in “`File` -> “`Examples`” -> “`AmebaWiFi`” -> “`SimpleServerWiFi`”

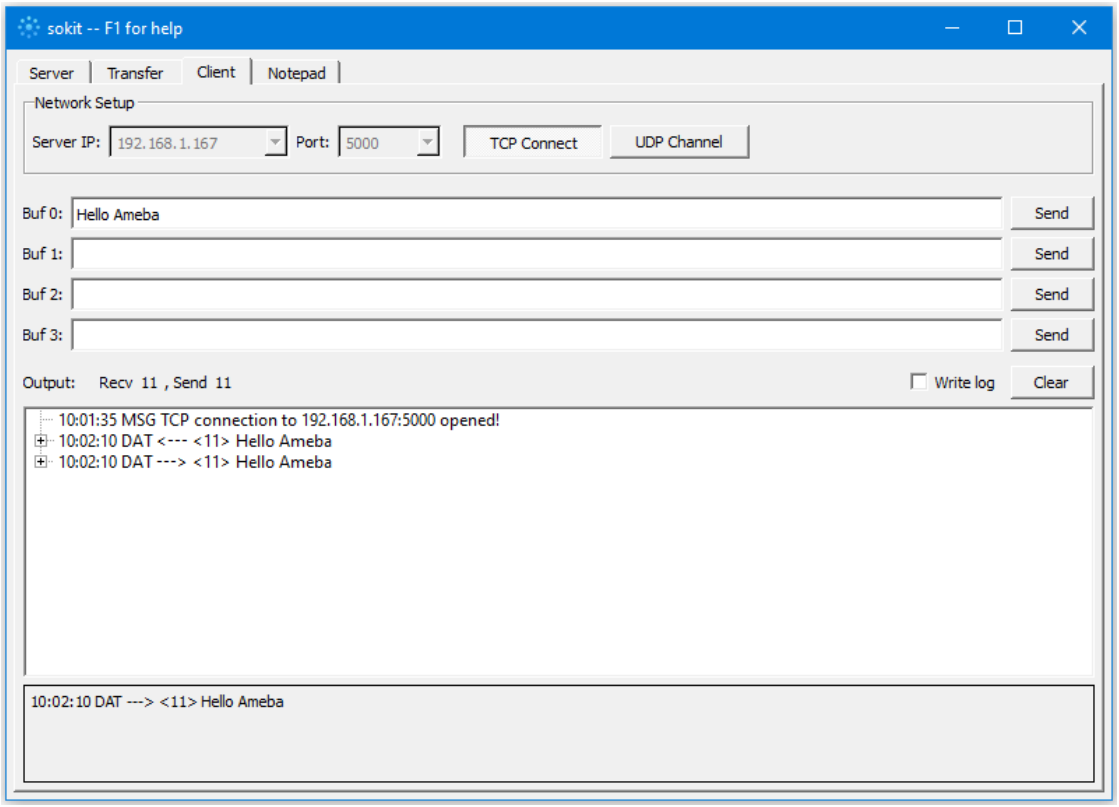


In the sample code, modify the highlighted parameters and enter the ssid and password for your WiFi connection.



Next, upload the code, then press the reset button on Ameba. At this moment, you will see the connection information is displayed in the console.

Next, we use the socket tool in the laptop to be the client and connect to the IP address of the Ameba board shown in the connection information at port 5000. (Note: The socket tool we used in this example is “sokit”)

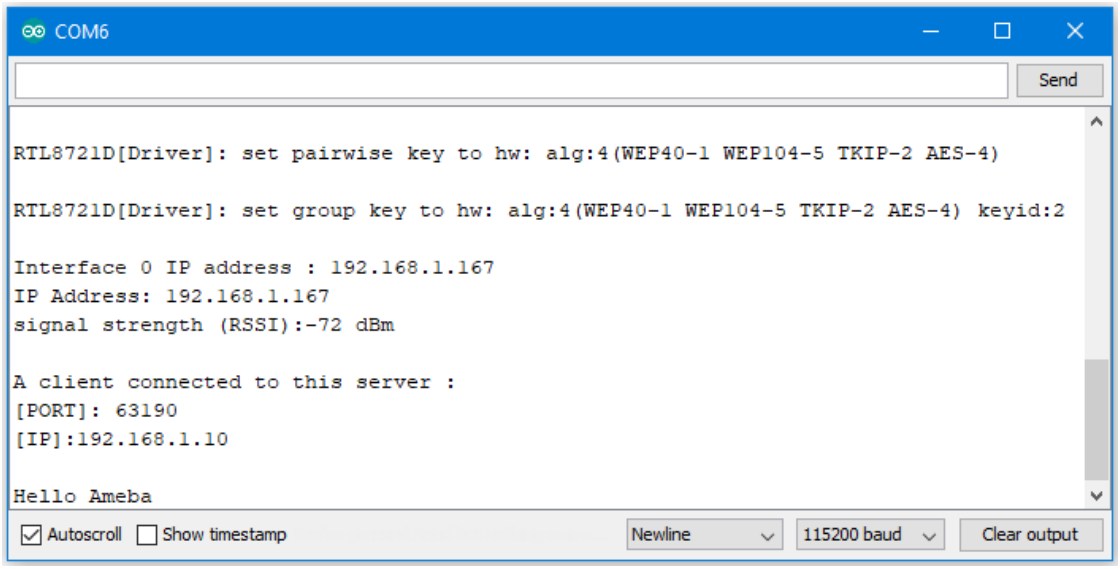


Click on the “Client” tab to choose the client mode, specify the IP and port of the server, then click “TCP Connect”.

If the connection is established successfully, the server shows a message: “A client connected to this Server”, and the IP and port of the connected client.

In this example, when the client and server are connected and the client sends a string to Ameba server, the Ameba server

returns the identical string back to the client.



The string sent to server is returned and showed at the client side.

Code Reference

Use `WiFi.begin()` to establish WiFi connection;
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the Ameba WiFi shield's IP address.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Create server and transmitting data:
Use `Server(port)` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.read()` to read the next byte received from the server.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.write()` to write data to the server.
https://www.arduino.cc/en/Reference/WiFiClientWrite
Use `client.stop()` to disconnect from the server.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up Client to Retrieve Google Search Information

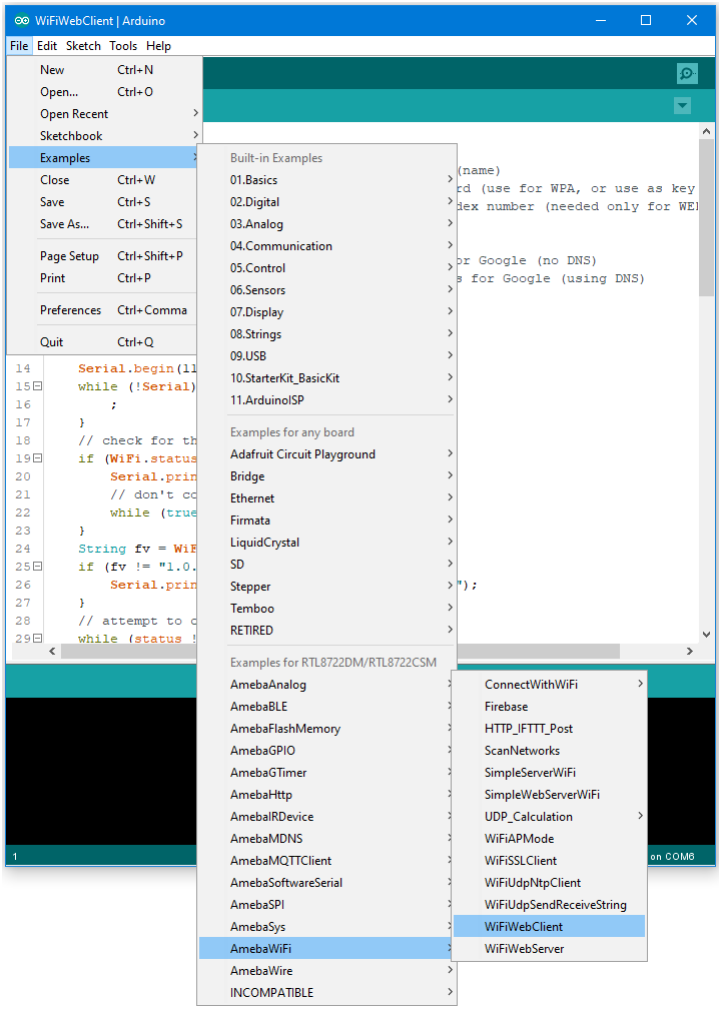
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

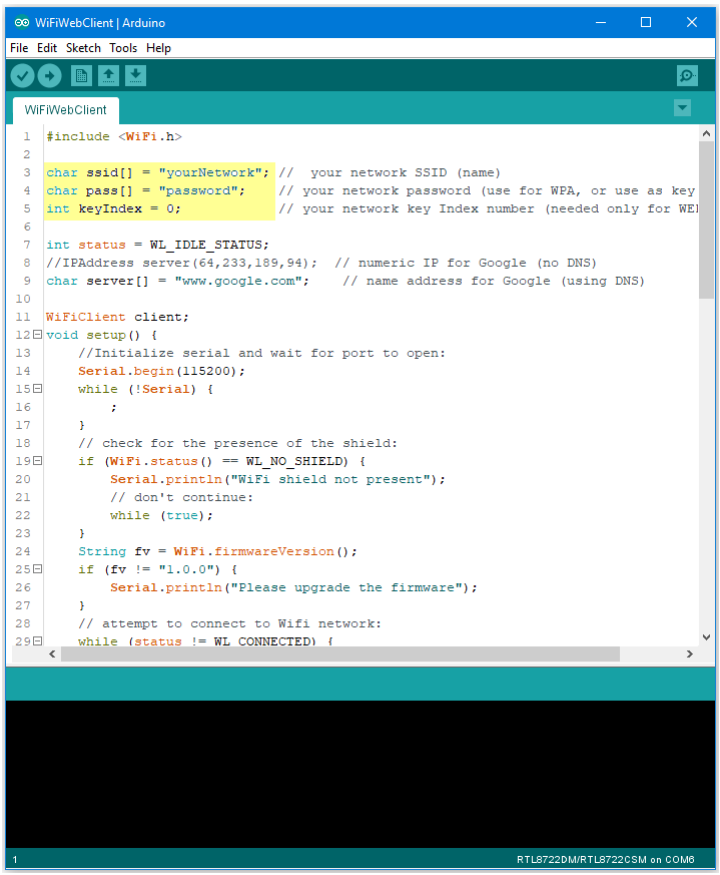
Example

In this example, we use Ameba to be a web client to retrieve information from the Internet.
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”

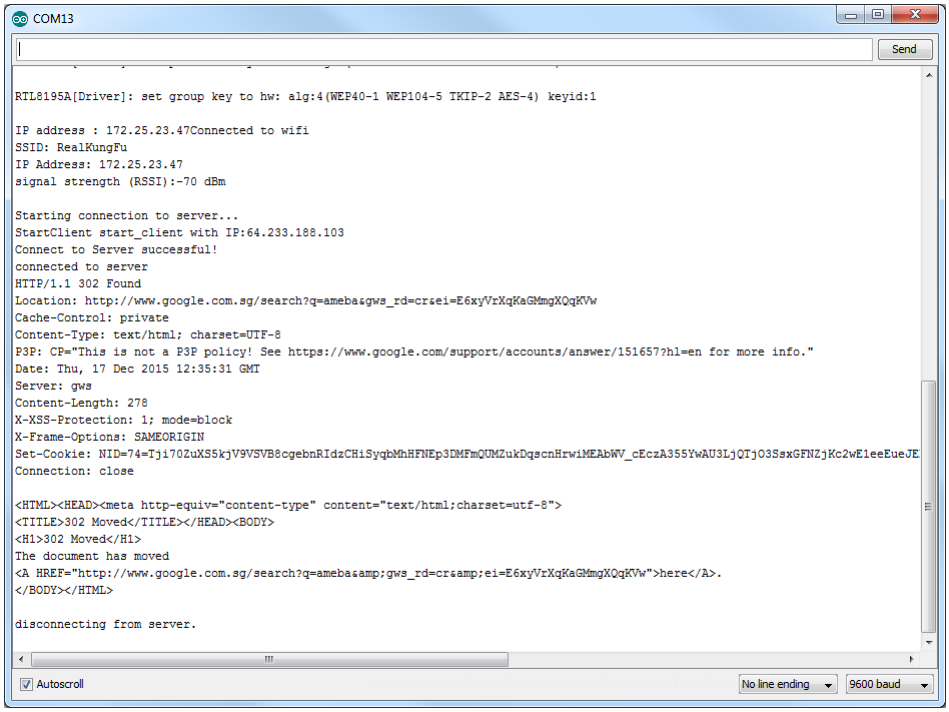
Then open “`File” -> “Examples” -> “AmebaWiFi” -> “WiFiWebClient`”



In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. Then you can see the information retrieved from Google is shown in the Arduino serial monitor.



Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection: Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiClient()` to create a client.
https://www.arduino.cc/en/Reference/WiFiClient
Use `client.connect()` to connect to the IP address and port specified.
https://www.arduino.cc/en/Reference/WiFiClientConnect
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up UDP Server to Communicate

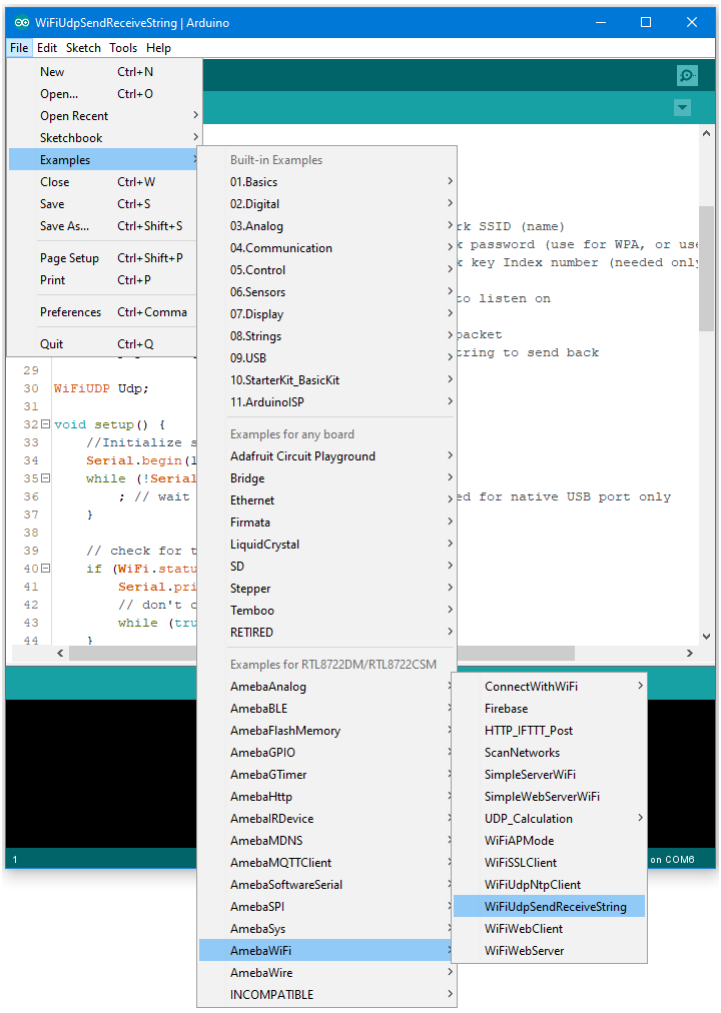
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

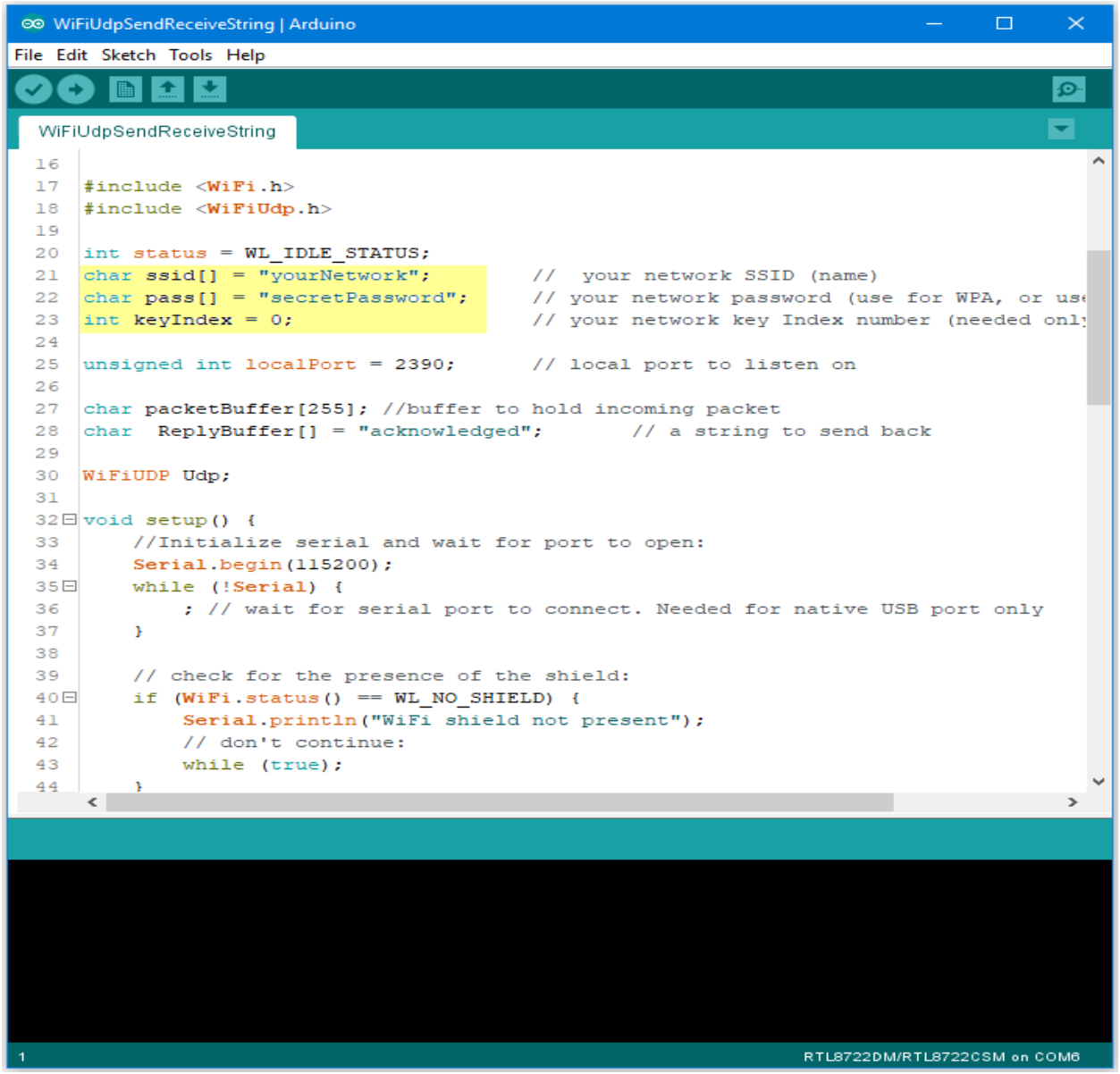
Example

In this example, we connect Ameba to WiFi and use Ameba to be an UDP server. When Ameba receives a message from UDP client, it replies "acknowledged" message to client.

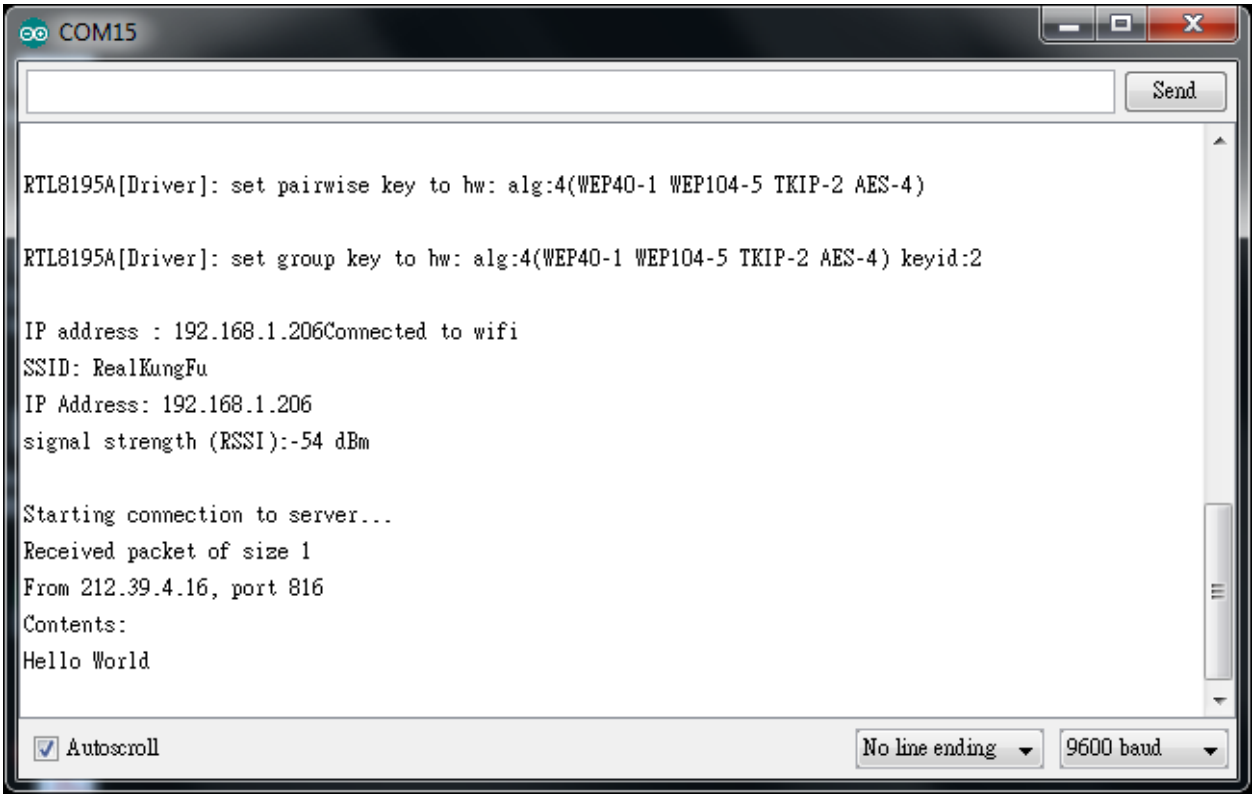
Open the WiFi Web Server example. "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiUdpSendReceiveString`"



Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



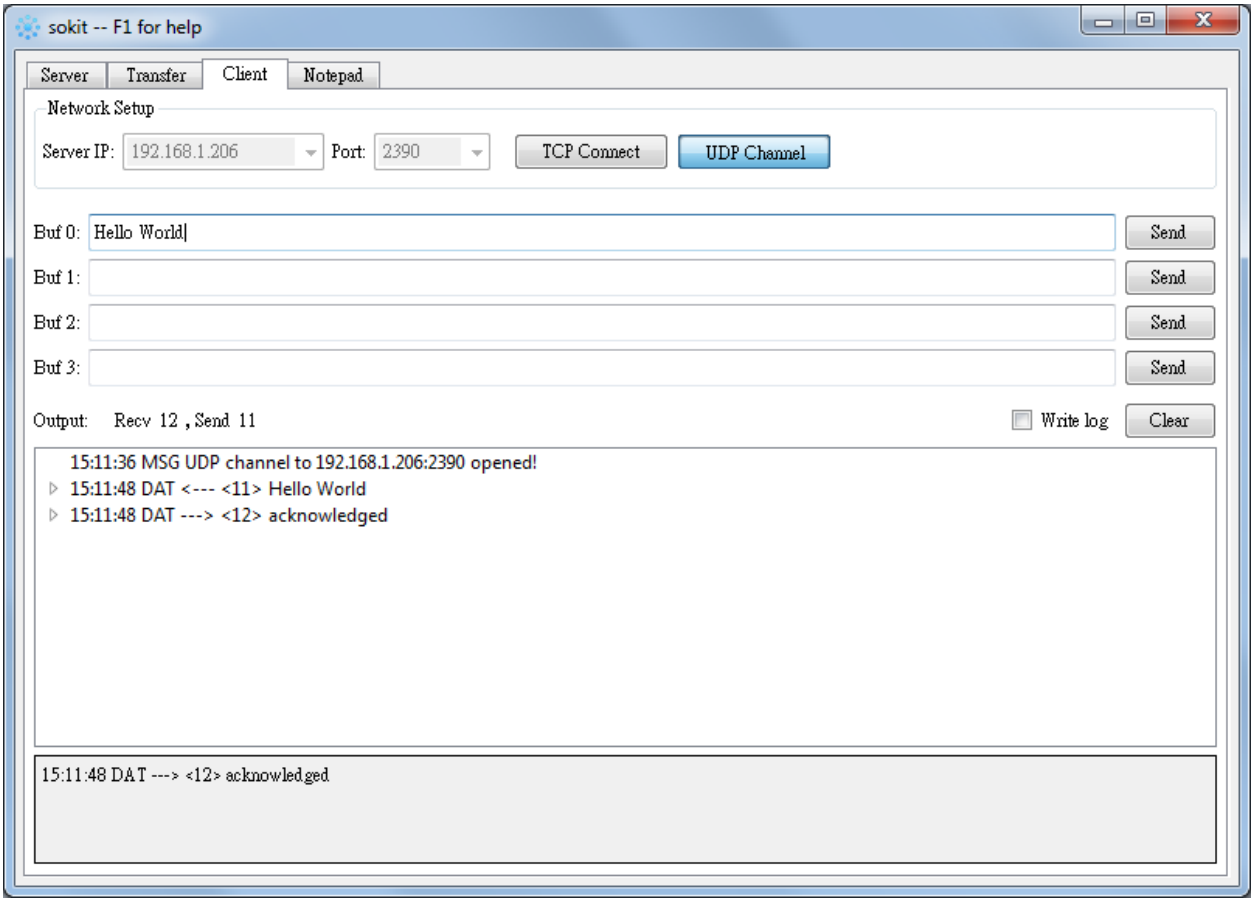
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and starts the UDP server with port 2390. After the UDP server starts service, Ameba prints the “Starting connection to server” message and waits for client connection.



As to the UDP client, we use “sokit” program in the computer to connect to UDP server.

Choose client mode and fill in the IP of UDP server (which is the IP of Ameba) and port 2390, then click “UDP Connect”.

After the connection is established, fill in “Hello World” in the Buf 0 field in sokit and click “Send”. Then you can see the Ameba UDP server replies “acknowledged”.



Code Reference

Refer to the Arduino tutorial for detailed information about this example.
https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString

First, use `begin()` to open an UDP port on Ameba.
https://www.arduino.cc/en/Reference/WiFiUDPBegin

Use `parsePacket()` to wait for data from client.
https://www.arduino.cc/en/Reference/WiFiUDPParsePacket

When a connection is established, use `remoteIP()` and `remotePort()` to get the IP and port of the client.
https://www.arduino.cc/en/Reference/WiFiUDPRemoteIP

Then use `read()` to read the data sent by client.
https://www.arduino.cc/en/Reference/WiFiUDPRead

To send reply, use `beginPacket()`, `write()`, `end()`.
https://www.arduino.cc/en/Reference/WiFiUDPBeginPacket
https://www.arduino.cc/en/Reference/WiFiUDPWrite
https://www.arduino.cc/en/Reference/WiFiUDPEndPacket

WiFi - Set up WiFi AP Mode

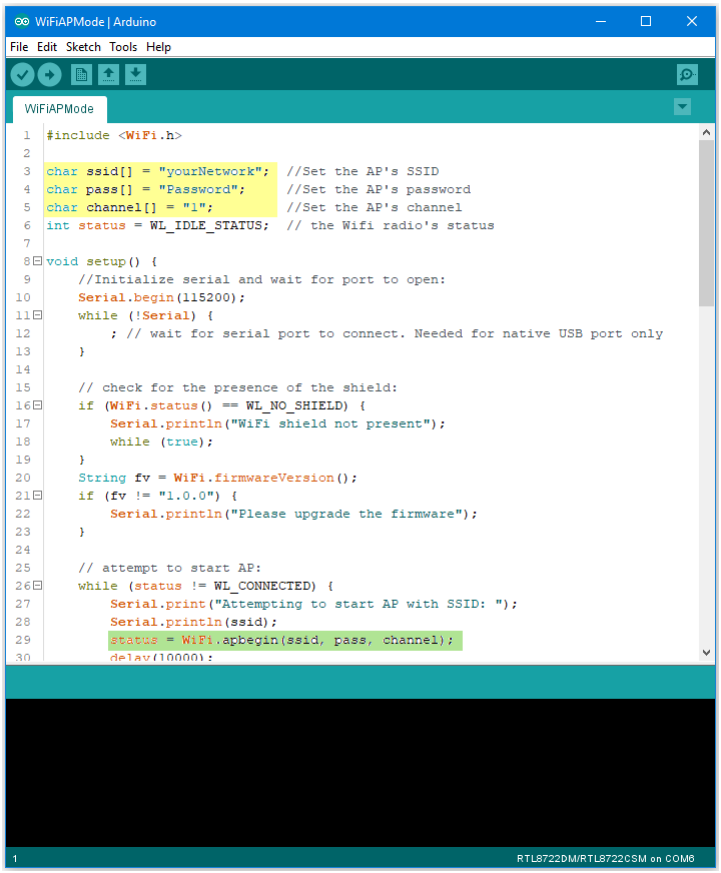
In AP mode, Ameba can accept at most 3 station connections, and can be set to open mode or WPA2 mode.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we turn on the AP mode of Ameba and connect station to Ameba.
Open the WiFi AP example, "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiAPMode`"



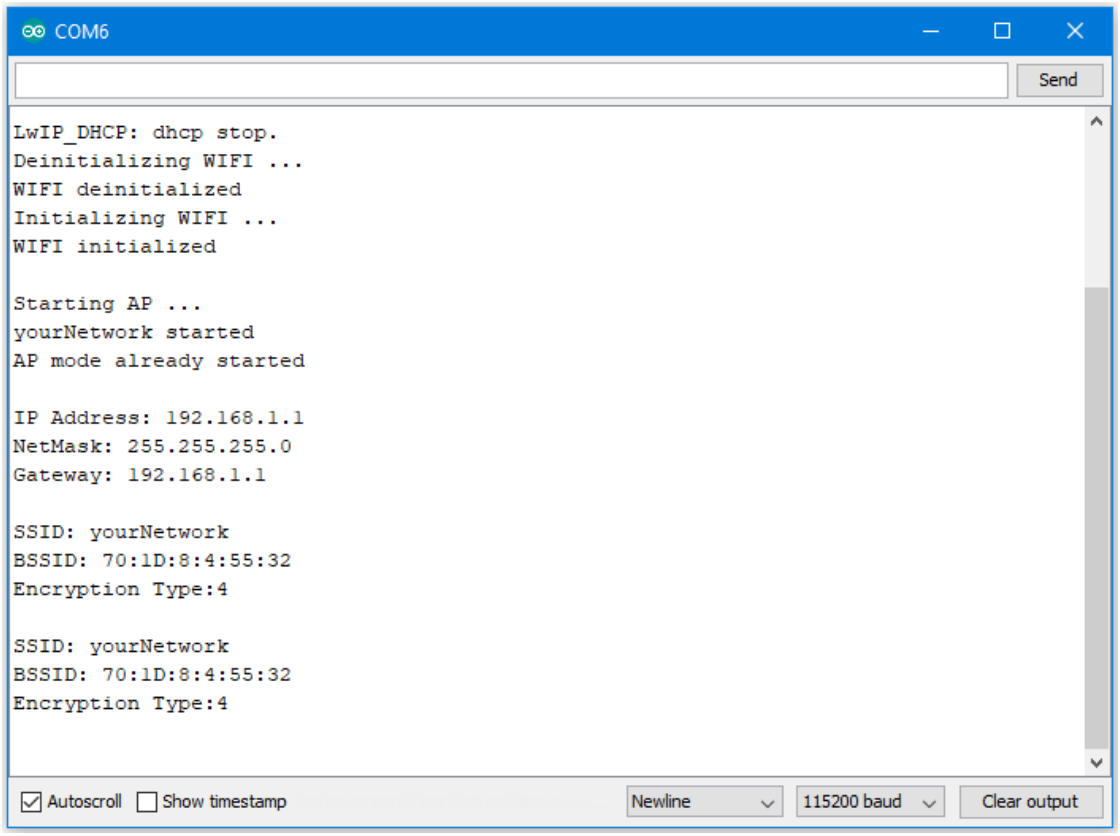
In the highlighted code snippet, fill in your SSID, PASSWORD and CHANNEL.

The code highlighted in green is the API we used to turn on the AP mode in security mode.

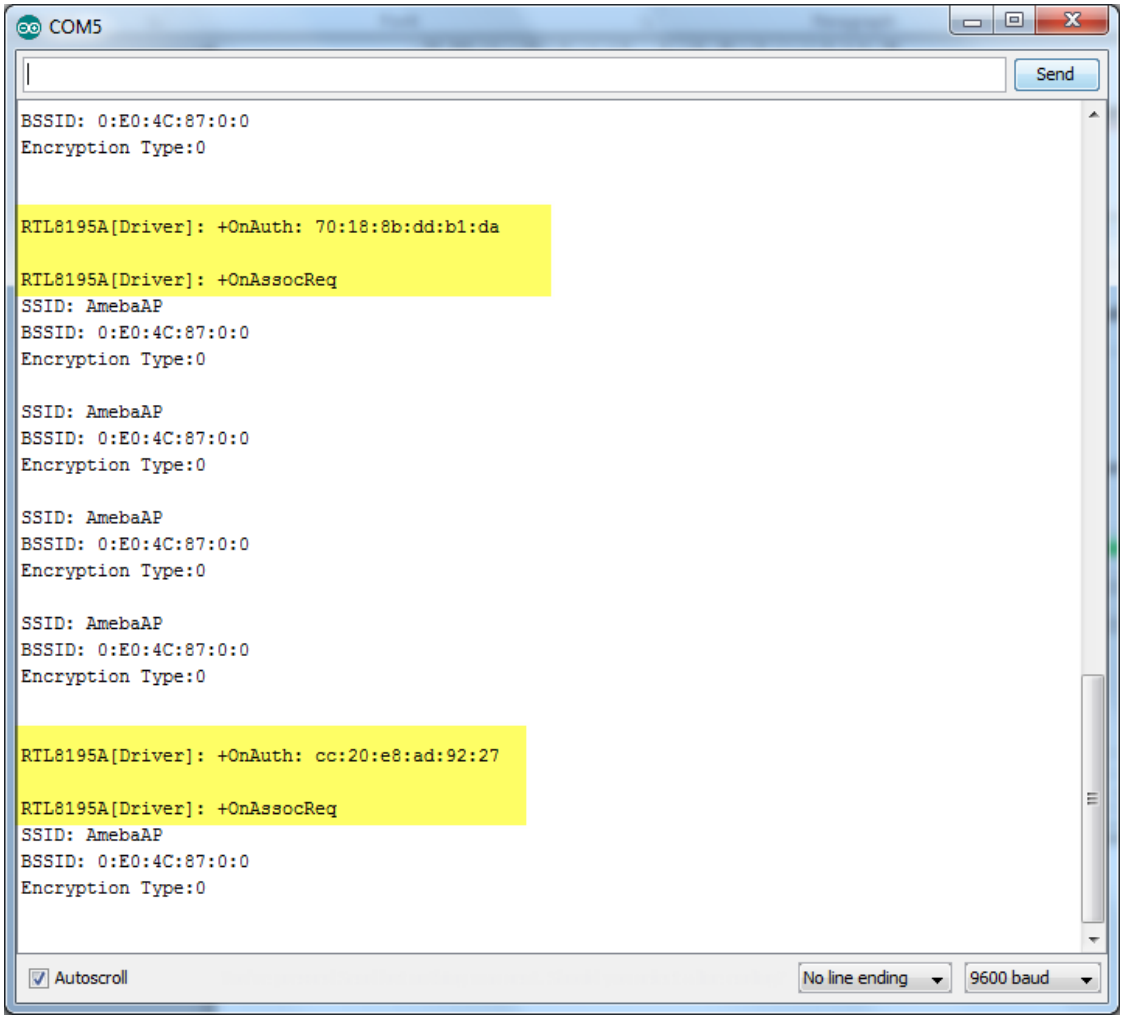
If you want to turn on the AP mode in open mode, please modify the code to

```
status = WiFi.apbegin(ssid, channel);
```

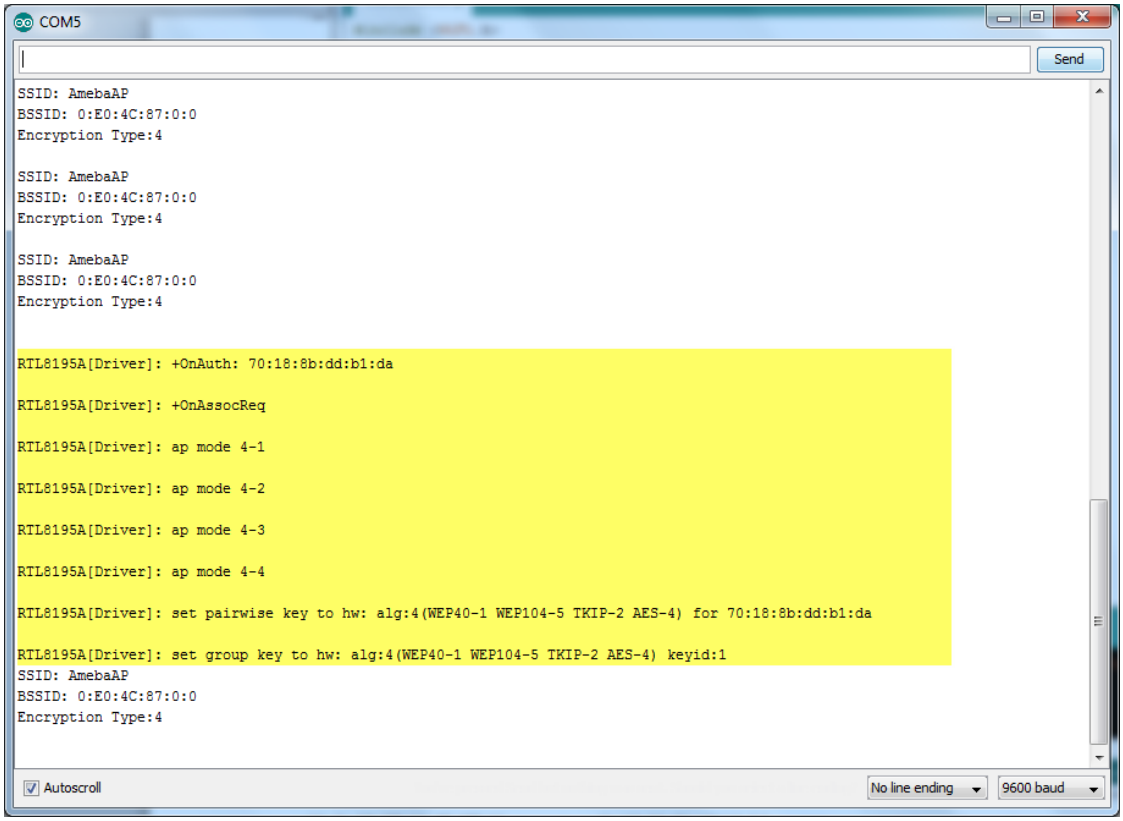
Then upload the sample code and press reset, and you can see related information shown in serial monitor.



In the figure below, we show the messages shown in serial monitor when two stations connect to Ameba AP in open mode:



In the figure below, we show the messages shown in serial monitor when a station connects to Ameba AP in security mode:



WiFi - Set up SSL Client for HTTPS Communication

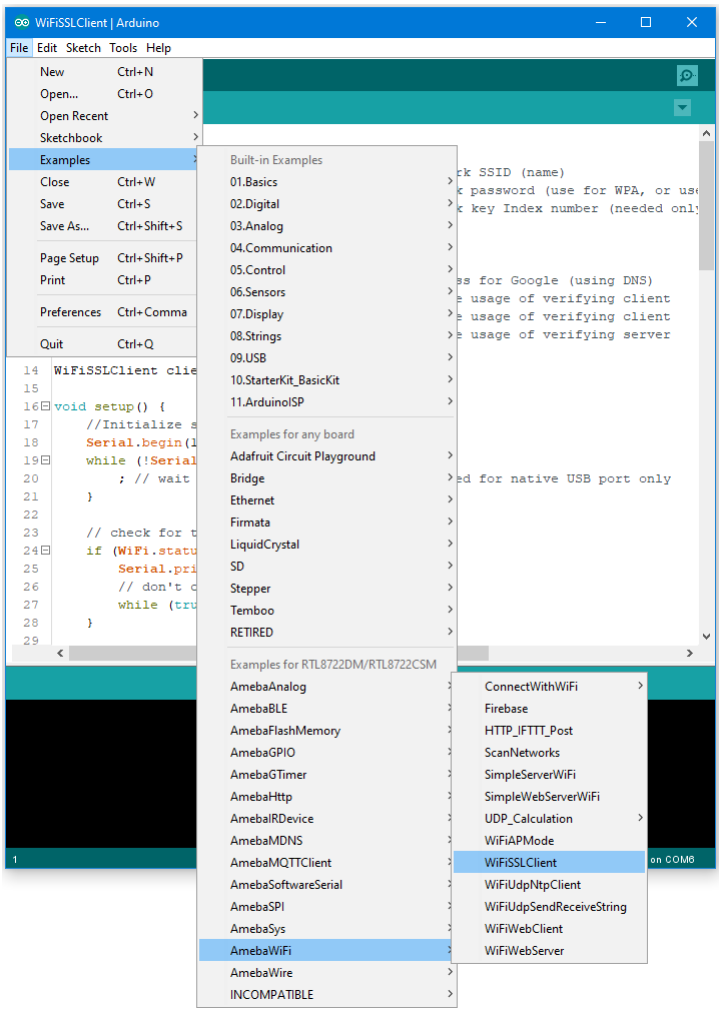
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

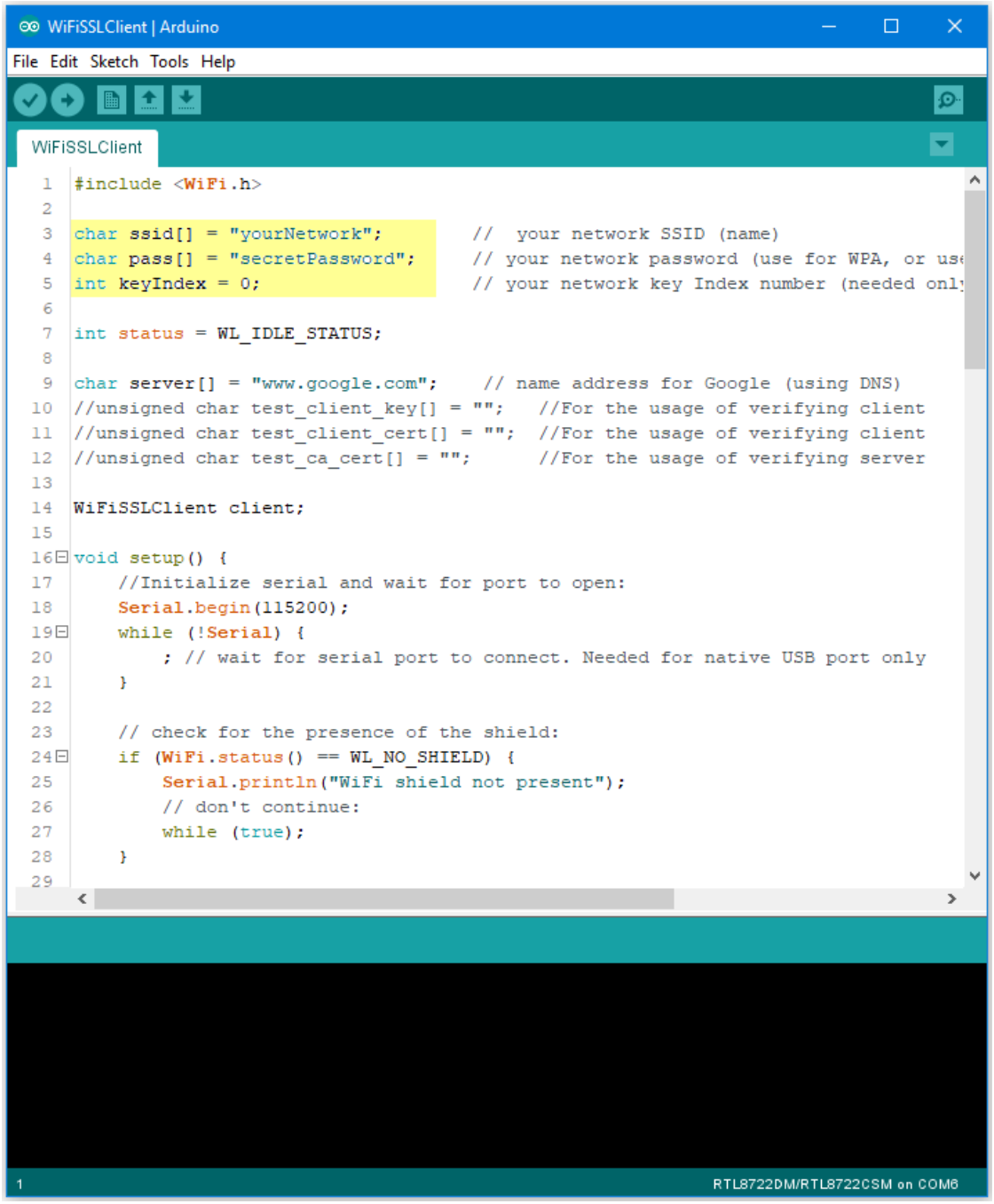
Example

This example uses Ameba to securely retrieve information from the internet using SSL. SSL is an acronym for Secure Sockets Layer. It is a cryptographic protocol designed to provide communications security over a computer network, by encrypting the messages passed between server and client.

Open the "WiFiSSLClient" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiSSLClient`".

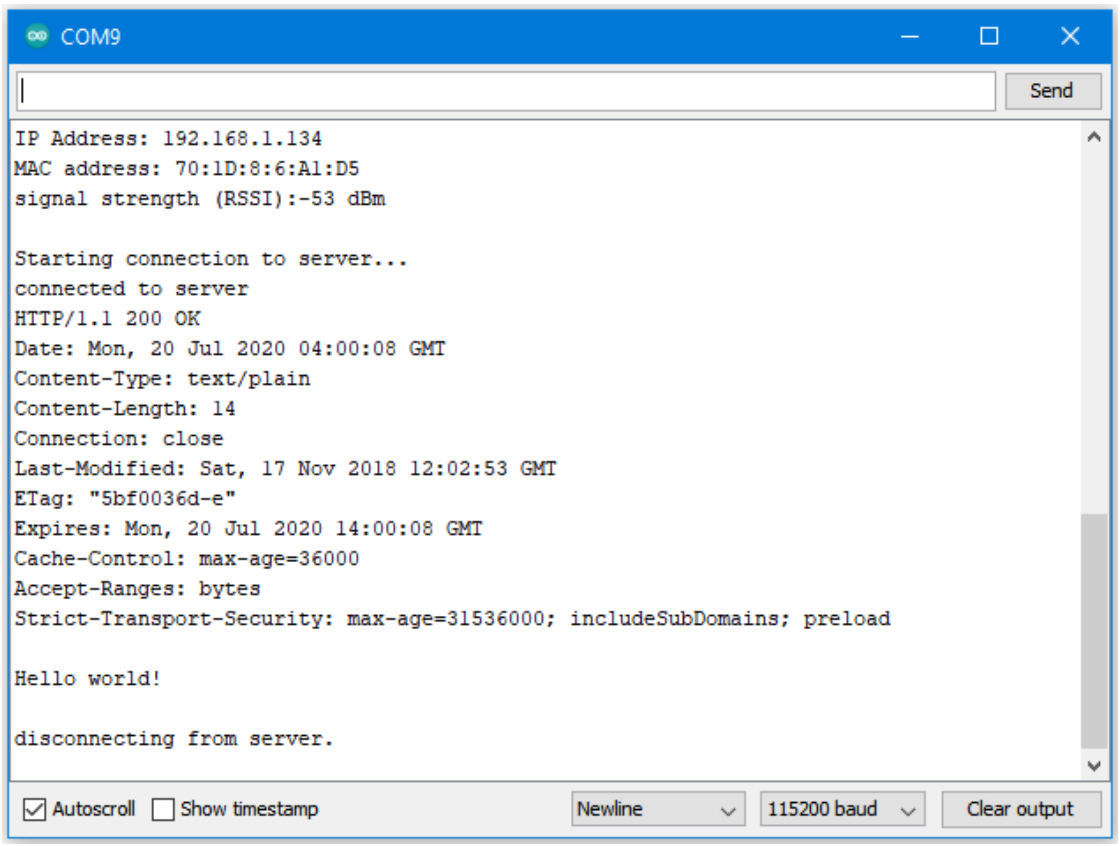


In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor in the Arduino IDE and observe as Ameba retrieves a text file from os.mbed.com.

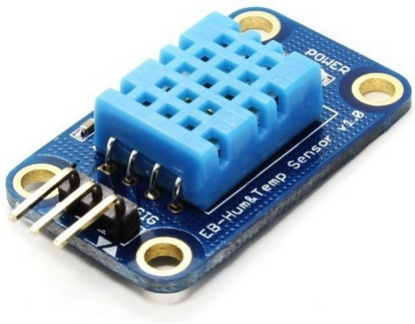
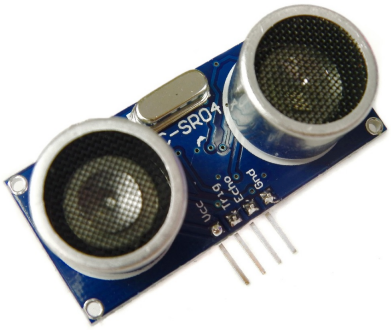
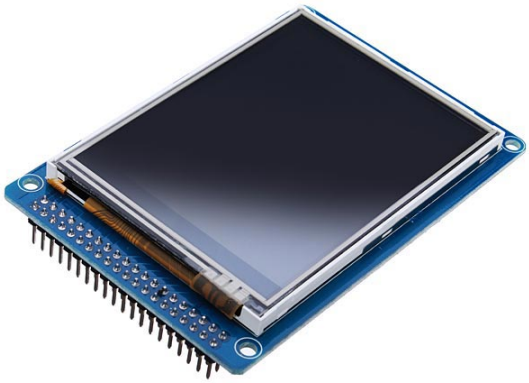
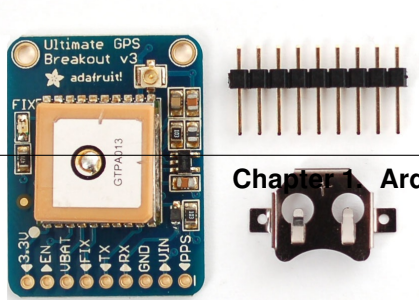


Code Reference

Use “WiFiSSLClient client;” to create a client that uses SSL. After creation, the client can be used in the same way as a regular client.

Components Used

Table 5: Components used in Examples

	
DHT11_DHT22 Humidity & temperature sensor	HC_SR04 Distance measurement function
	
ILI9341 TFT LCD TFT LCD display with SPI interface	PMS3003/5003 Air quality sensor that detects concentration of micro particulate matters
	

Peripheral Examples

Audio Codec – Basic Input Output

Materials

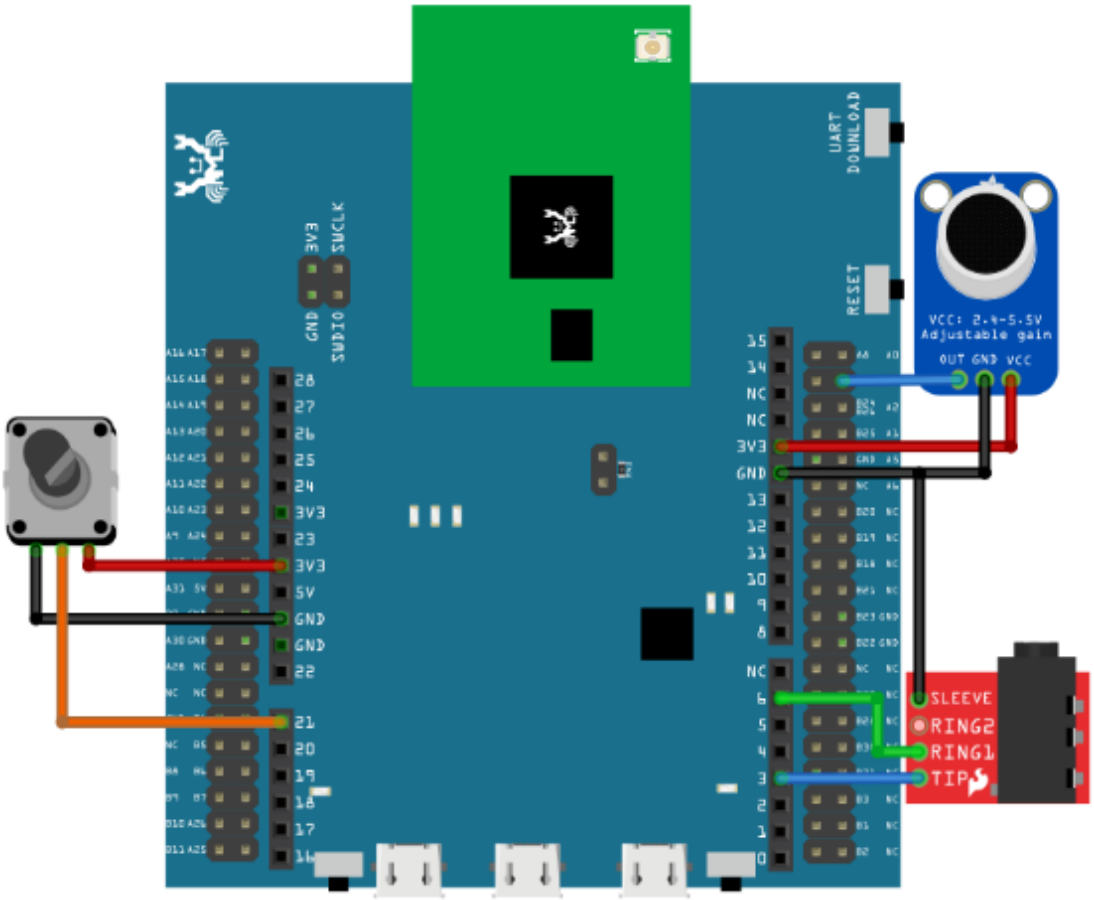
- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Potentiometer x 1
- Analog microphone x 1 (e.g., Adafruit 1063 / 1064)
- 3.5mm TRS/TRRS breakout x 1 (e.g., Adafruit 2791 / Sparkfun 11570)

Example

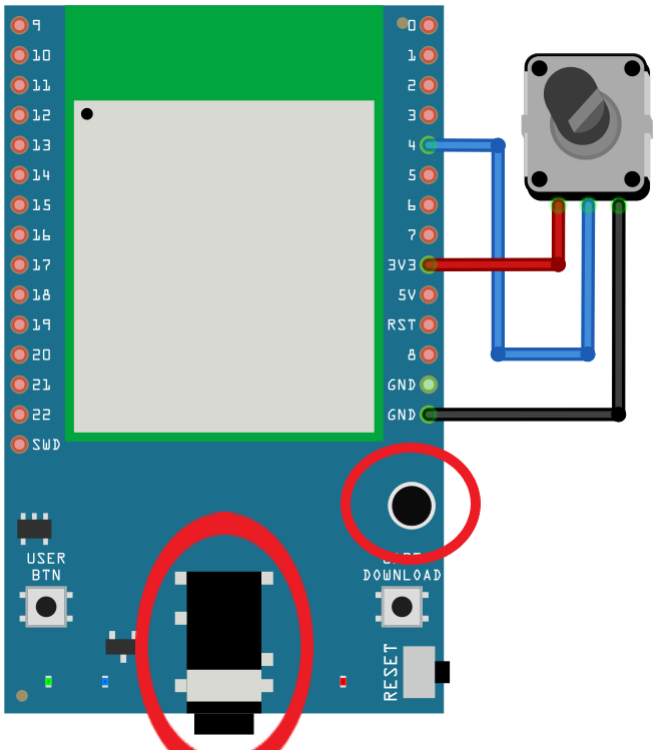
Procedure

Connect the potentiometer, microphone and 3.5mm connector to the RTL8722 board following the diagram.

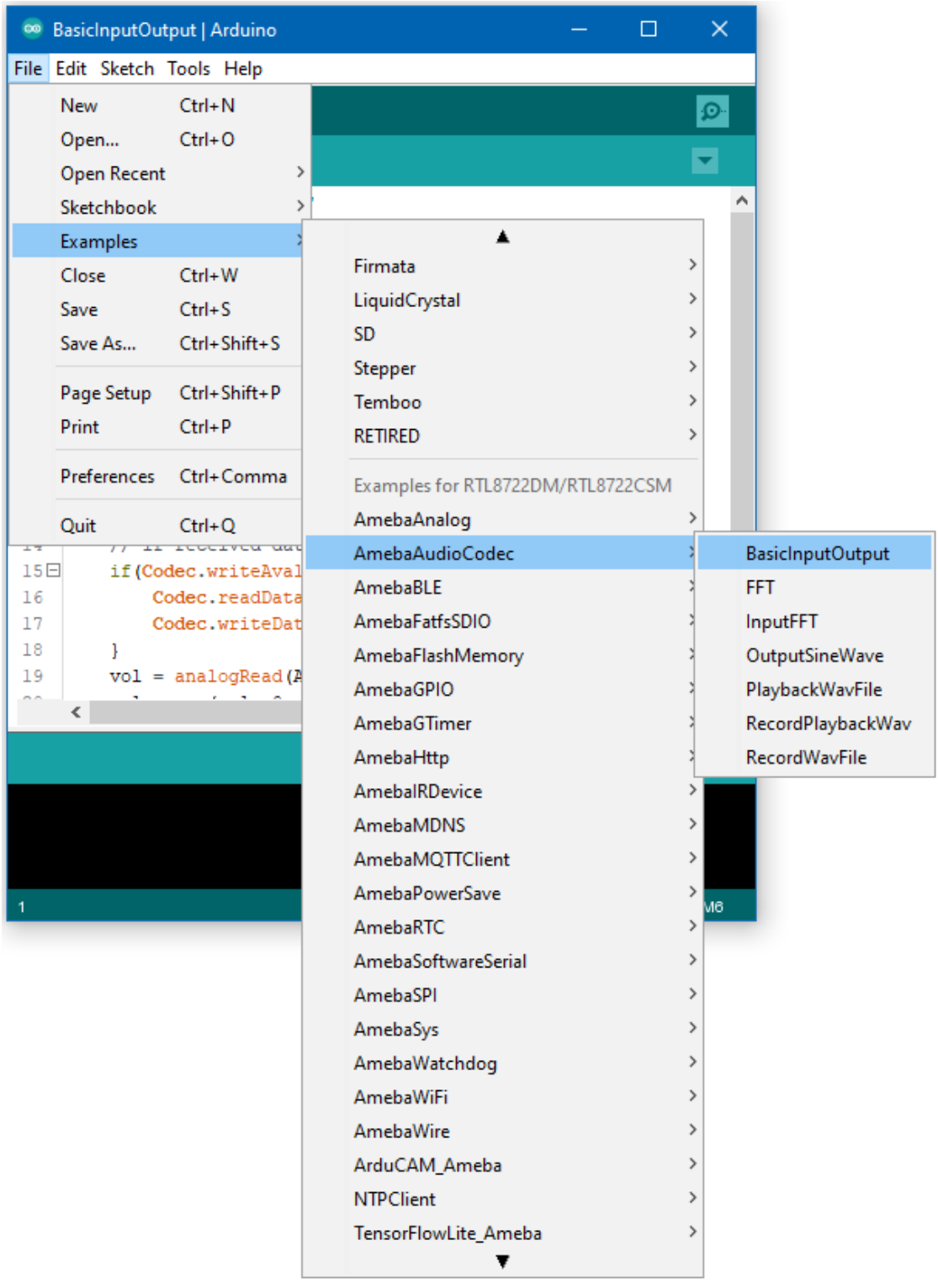
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



Open the example, `"Files" -> "Examples" -> “AmebaAudioCodec” -> “BasicInputOutput”`.



Upload the code and press the reset button on Ameba once the upload is finished.

Connect a pair of wired headphones to the 3.5mm audio jack, blow at the microphone, and you should hear the sounds picked-up by the microphone replayed in the headphones. Adjust the potentiometer and the output volume will change as well. Note: if you are using a microphone with an amplifier included, such as Adafruit 1063, the amplifier can lead to the microphone picking up more noise.

Audio Codec - FFT

Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1

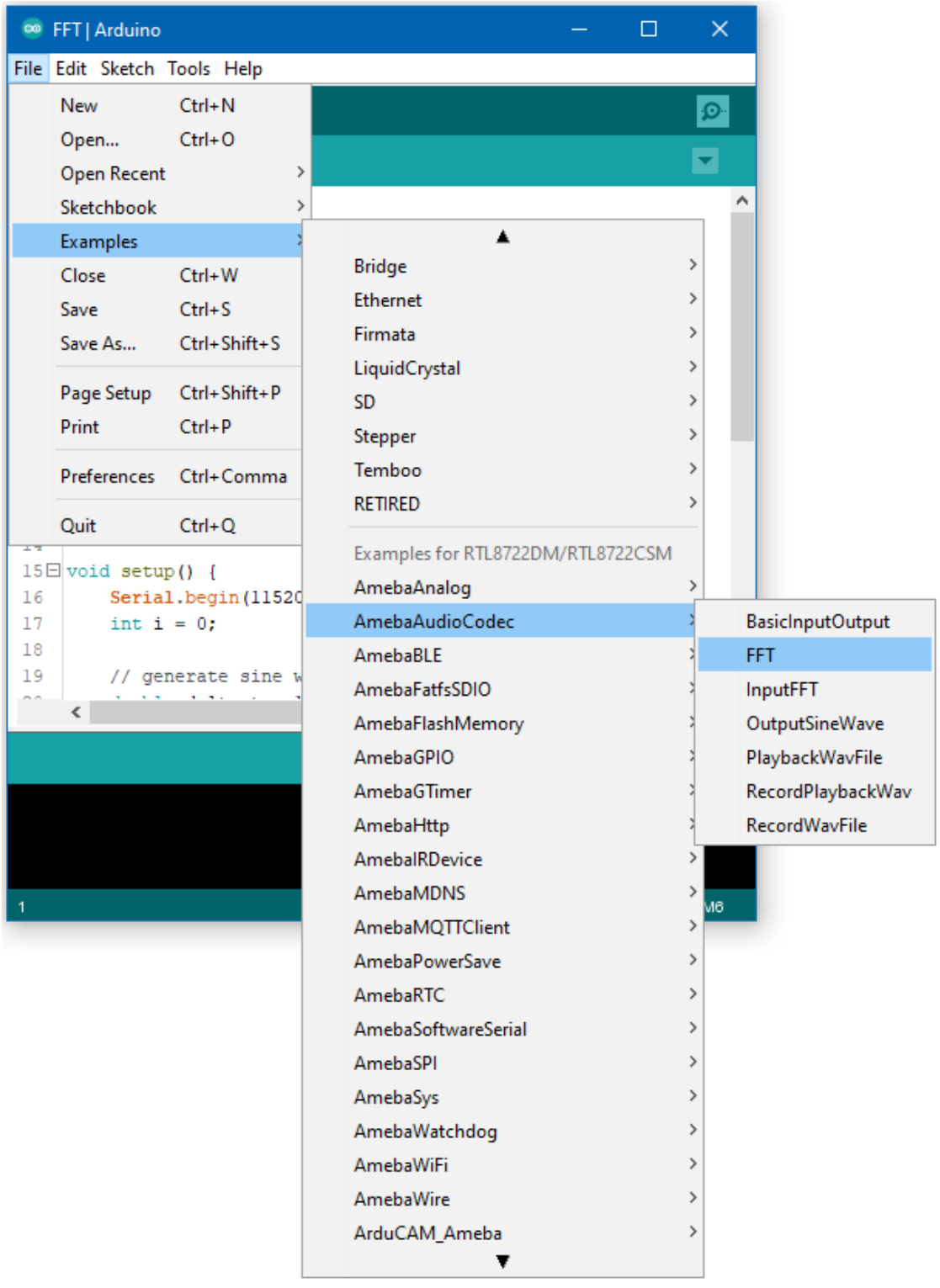
Example

Introduction

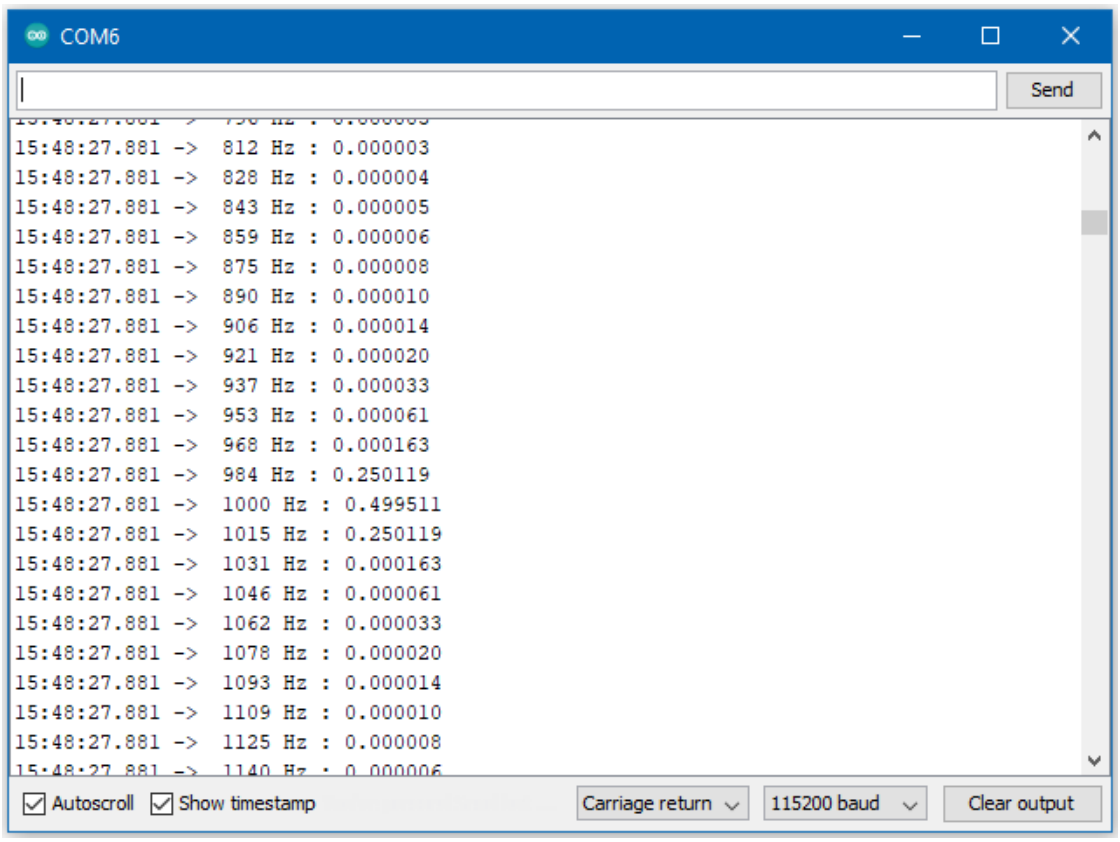
This example shows how to use the AudioCodec_FFT class to calculate the fast Fourier transform of a signal to extract the frequencies present in the signal.

Procedure

Open the example, `"Files" -> "Examples" -> “AmebaAudioCodec” -> “FFT”`.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor, and the output results of the AudioCodec_FFT calculation will be displayed.



Audio Codec - Input FFT

Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Analog microphone x 1 (e.g., Adafruit 1063 / 1064)

Example

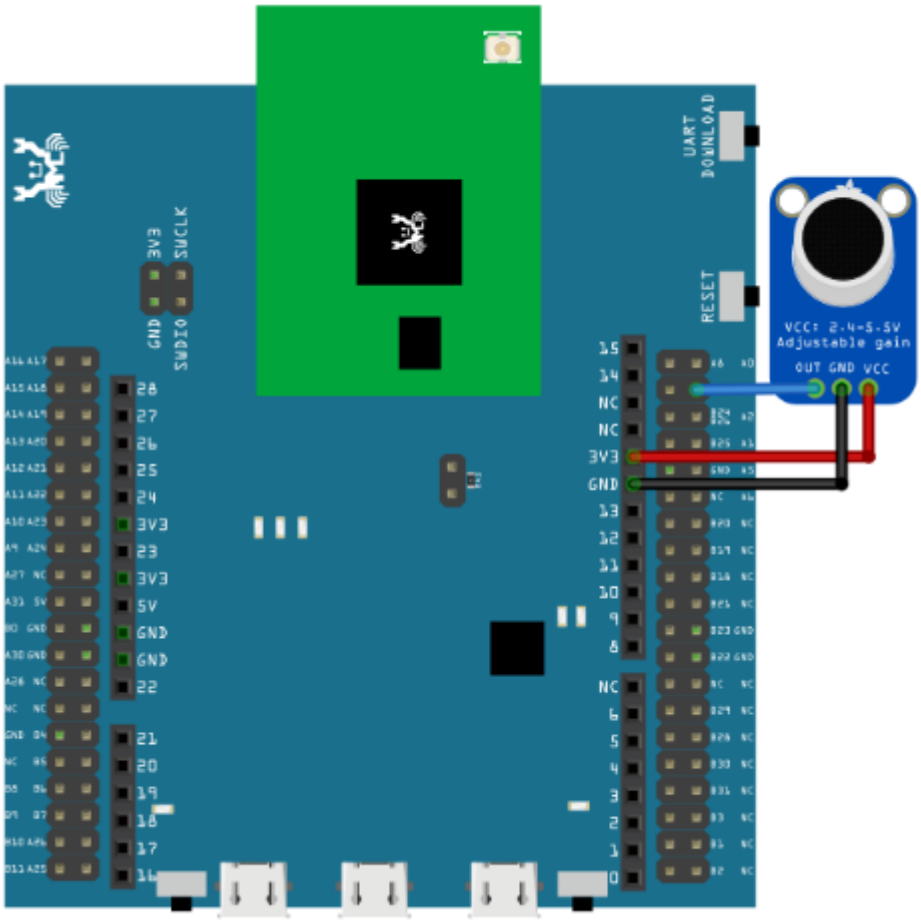
Introduction

This example shows how to use the FFT class to calculate the fast Fourier transform of the audio signal recorded by the microphone.

Procedure

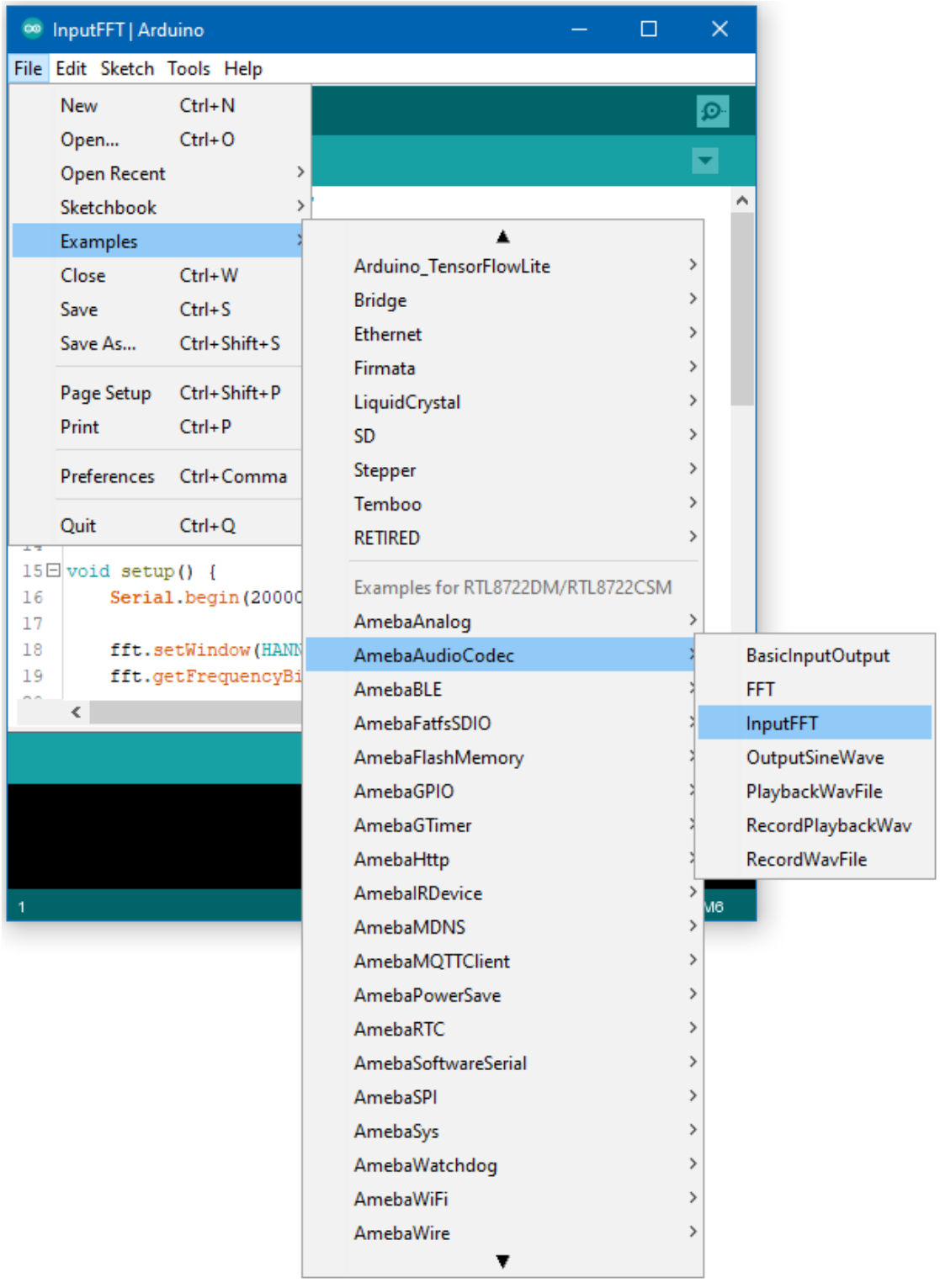
Connect the microphone to the RTL8722 board following the diagram.

AMB21 / AMB22 Wiring Diagram:

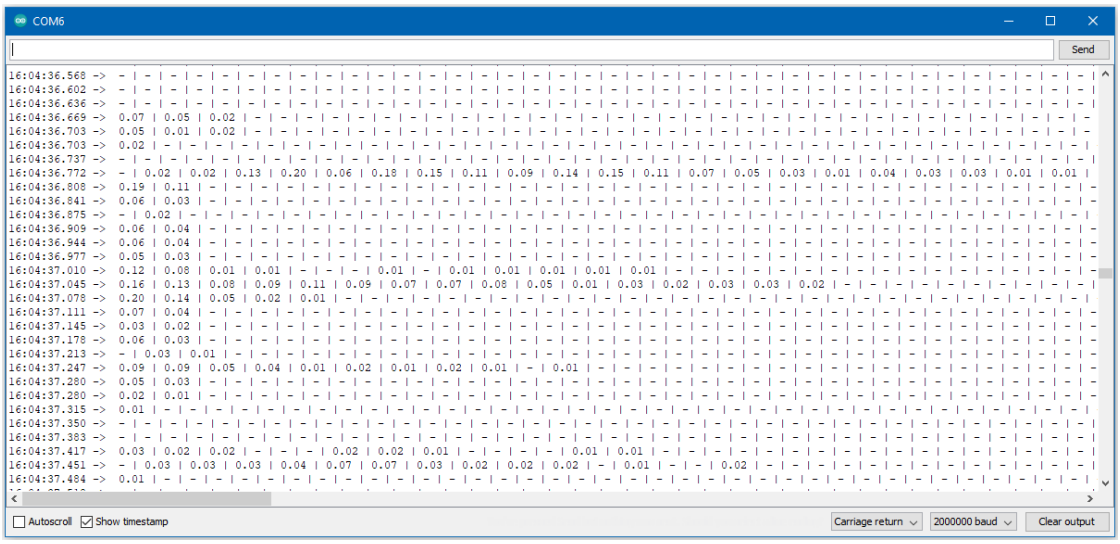


AMB23 Wiring Diagram:
As AMB23 have a built in microphone on the board, there is no need for any external microphone.

Next, open the example, `"Files" -> "Examples" -> "AmebaAudioCodec" -> "InputFFT"`.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor and change the baud rate to 2000000. A stream of FFT results of audio samples will be displayed. Try playing music or use a smartphone app to generate a sine wave into the microphone, and you should be able to see the FFT output change.



Audio Codec – Output Sine Wave

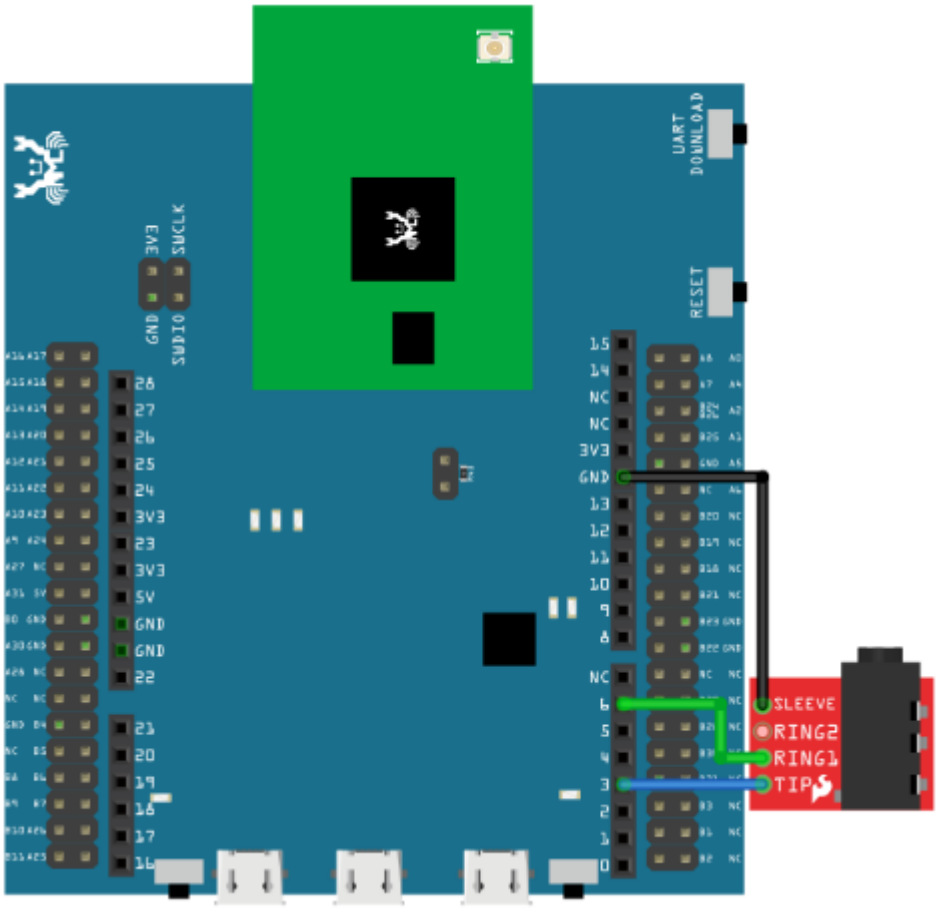
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- 3.5mm TRS/TRRS breakout x 1 (e.g., Adafruit 2791 / Sparkfun 11570)

Example

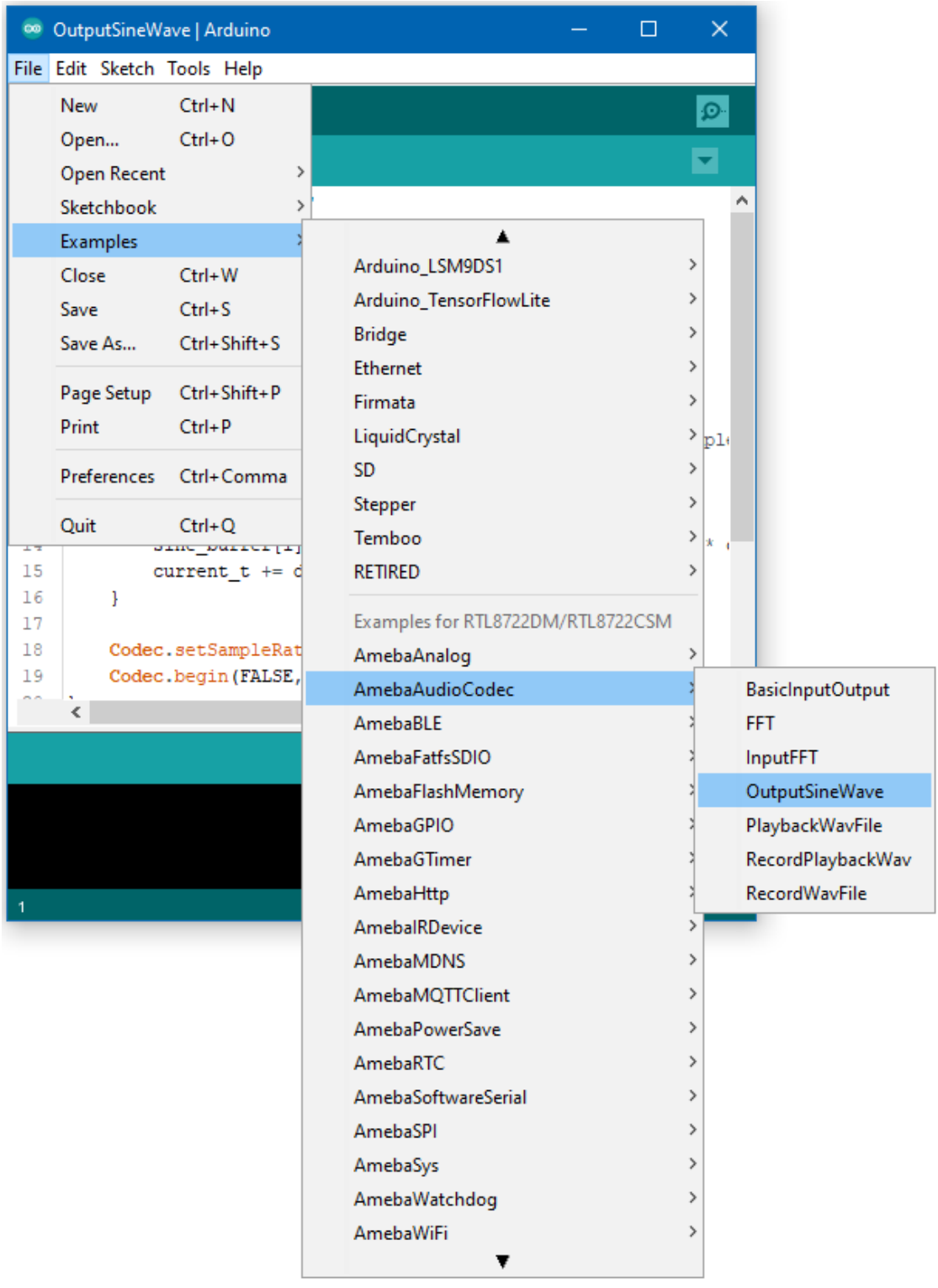
Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the 3.5mm connector to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
As AMB23 have a built in microphone on the board, there is no need for any external microphone.

Open the example, `"Files" -> "Examples" -> "AmebaAudioCodec" -> "OutputSineWave"`.



Upload the code and press the reset button on Ameba once the upload is finished.
Connect a pair of wired headphones to the 3.5mm audio jack and you should hear the generate single sinusoidal tone.

Audio Codec – Play and Record Wav Files

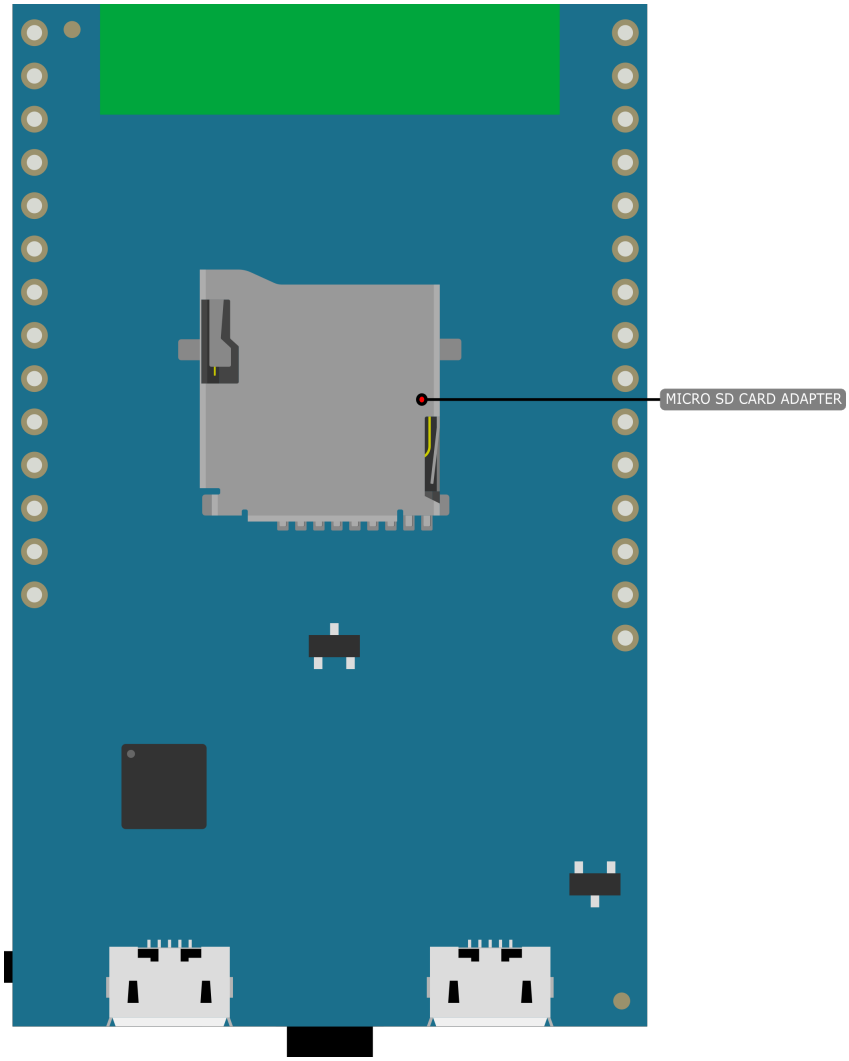
Materials

- AmebaD [AMB23] x 1
- MicroSD card

Example

Procedure

As AMB23 have a built in microphone on the board, there is no need for any external microphone. Copy a sample wav file into the MicroSD card for demo. (In this example, the sample name is "Test_Audio_48khz_16bit_stereo.wav".) Then insert the MicroSD card into the adapter at the back of the board.



Example 01 PlaybackWavFile Open the example, "`Files`" -> "`Examples`" -> "`AmebaAudioCodec`" -> "`PlaybackWavFile`".


```
#include "FatFs_SD.h"
#include "PlaybackWav.h"
#include "AudioCodec.h"

char filename[] = "Test_Audio_48khz_16bit_stereo.wav";

#define BUFFERSIZE 512
int16_t buffer[BUFFERSIZE] = {0};

FatFsSD fs;
PlaybackWav playWav;

// Callback function to feed audio codec with additional data
void writeCBFunc() {
    if(Codec.writeAvaliable()) {
        playWav.readAudioData(buffer, BUFFERSIZE);
        Codec.writeDataPage(buffer, BUFFERSIZE);
    }
}
```

Upload the code and press the reset button on Ameba once the upload is finished. Insert earphone/speaker into the onboard jack for playing the sample sound.

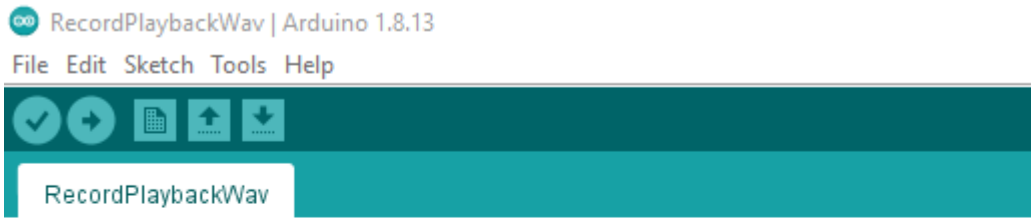
Example 02 RecordWavFile Open the example, “`Files`” -> “`Examples`” -> “`AmebaAudioCodec`” -> “`RecordWavFile`”.

Define a GPIO/button(Input high to active) for "RECORDBTN". Define the "filename[]" for name of the storge wav file. In this example the name is "Test_Recording_48khz_16bit_mono.wav".

Upload the code and press the reset button on Ameba once the upload is finished.

Input high to "RECORDBTN", then record voice by on board mic. When input is low the record stops. The recorded voice will be stored in "Test_Recording_48khz_16bit_mono.wav" that located at MicroSD card.

Example 03 RecordPlaybackWav Open the example, "`Files`" -> "`Examples`" -> "`AmebaAudioCodec`" -> "`RecordPlaybackWav`".




```
#include "FatFs_SD.h"
#include "PlaybackWav.h"
#include "RecordWav.h"
#include "AudioCodec.h"

#define RECORDBTN 9
#define SAMPLERATE 48000

#define BUFFERSIZE 512
int16_t buffer[BUFFERSIZE] = {0};

char filename[] = "Test_Recording_48khz_16bit_mono.wav";
char absolute_filename[128];

FatFsSD fs;
RecordWav recWav;
PlaybackWav playWav;

void readCBFunc() {
    if (Codec.readAvaliable() && recWav.fileOpened()) {
        Codec.readDataPage(buffer, BUFFERSIZE);
        recWav.writeAudioData(buffer, BUFFERSIZE);
    }
}
```

This example is a combination of Example01 and Example02. You can record by Example02 then play it by jack as the method of Example01.

Define a GPIO/button(Input high to active) for “RECORDBTN”. Define the “filename[]” for name of the storge wav file. In this example the name is

“Test_Recording_48khz_16bit_mono.wav”.

Upload the code and press the reset button on Ameba once the upload is finished.

Input high to “RECORDBTN”, then record voice by on board mic. When input is low the record stops. The recorded voice will be stored in

“Test_Recording_48khz_16bit_mono.wav” that located at MicroSD card.

Insert earphone/speaker into the onboard jack for playing the sample sound.

E-Paper - Display Images

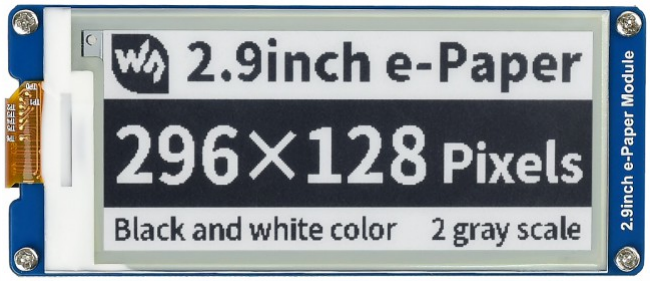
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

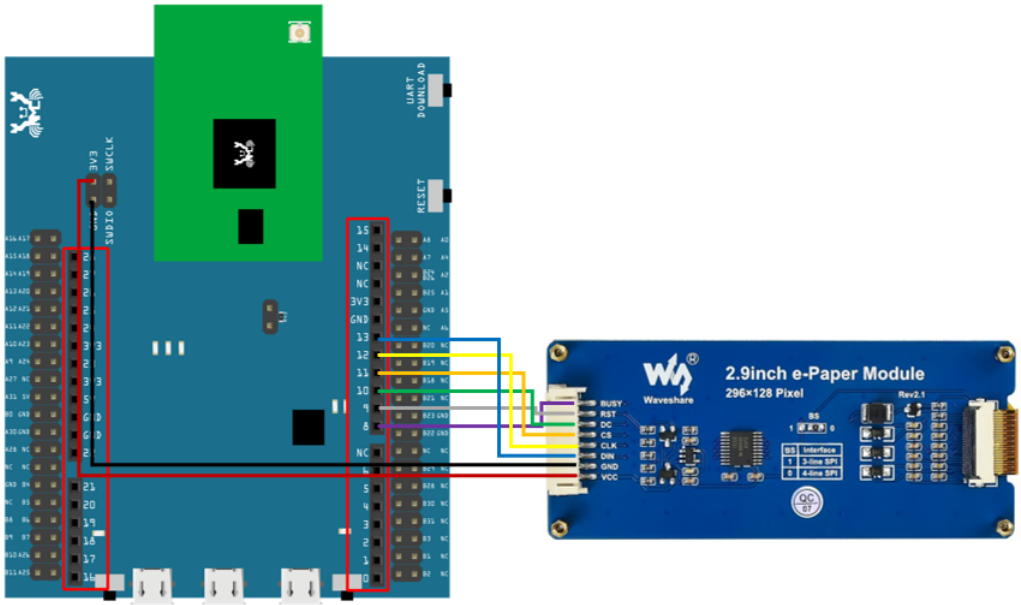
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

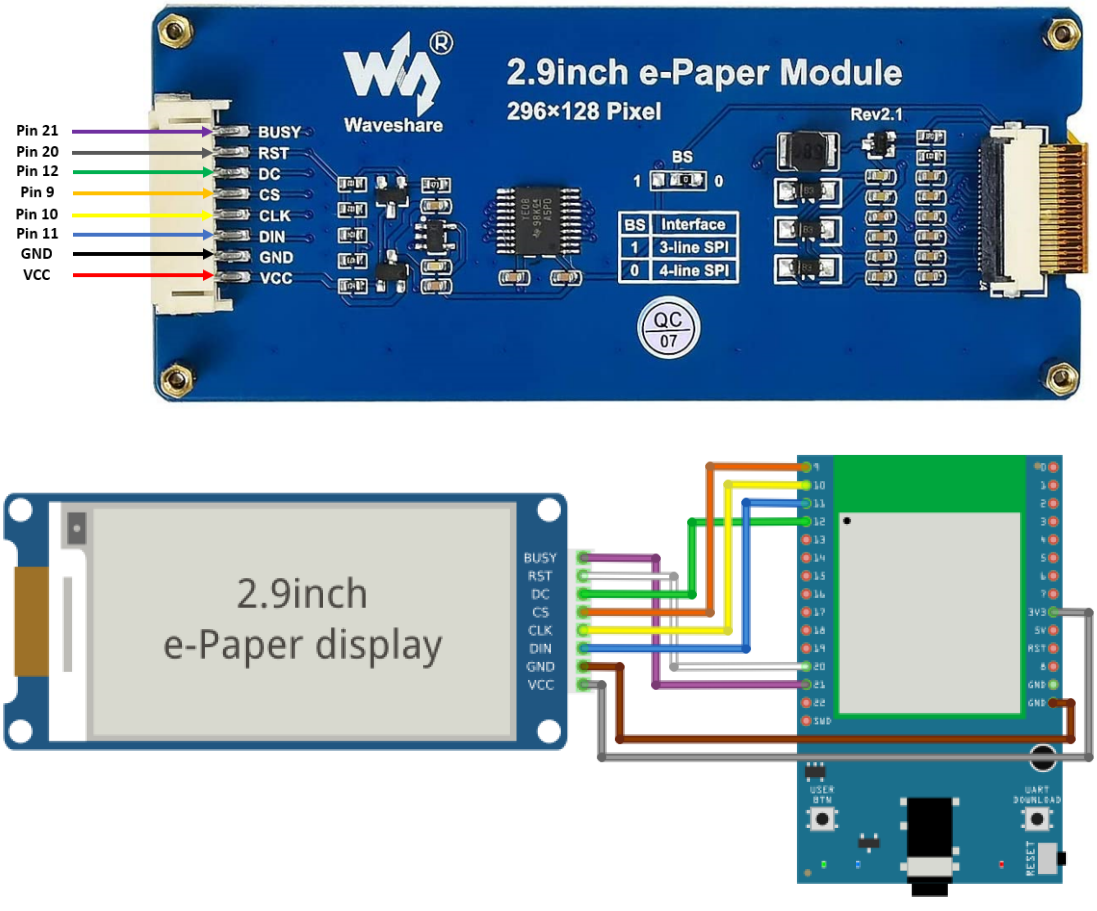
Front view of the e-Paper Module:



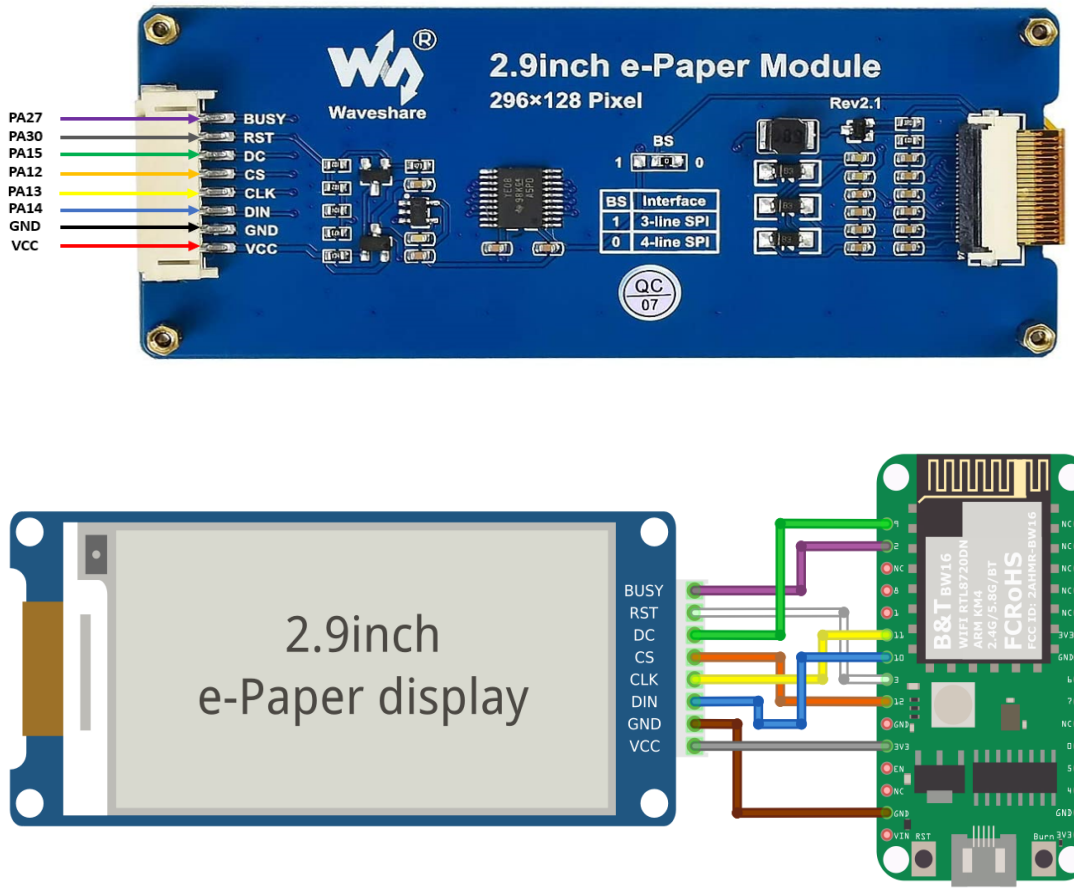
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:

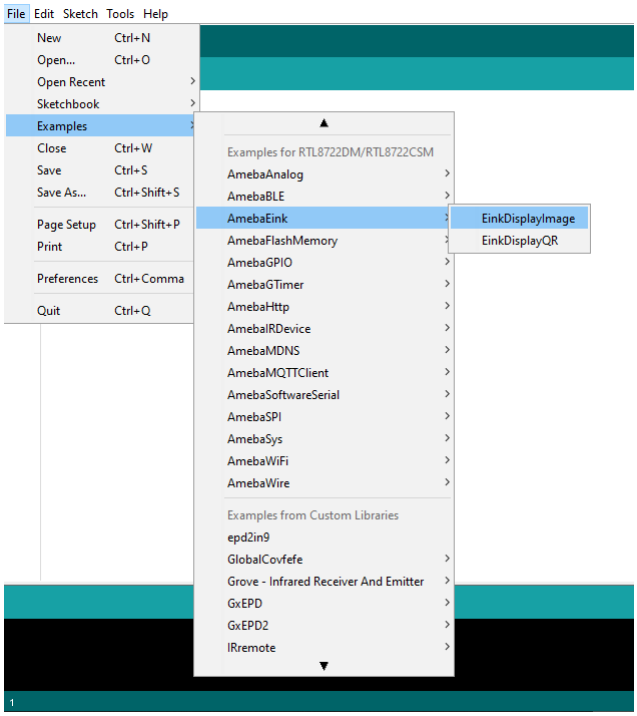


Firstly, you need to prepare a picture/photo in the format of 296×128 pixels. We can easily find a photo resizing tool online, for example, the Online Image Resizer.

Following the instructions on the website, then download the generated image in JPG format.

Secondly, we use the Image2LCD tool to transfer the downloaded 296×128 image into hexadecimal codes. You can visit this YouTube link to get detailed instructions.

Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`EinkDisplayImage`":



Press the reset button after uploading the sample code, you will need to wait for around 1-2 seconds for the e-Paper module to fresh its screen. Then the screen will start to display an image for 5 seconds first, then 3 different QR codes will be displayed every 5 seconds (showing in the screenshot below, you may scan the QR codes and find out more information if you wish to). Lastly, a gif which comes in form of 3 frames will be displayed for a few seconds.

Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html

E-Paper - Display Text

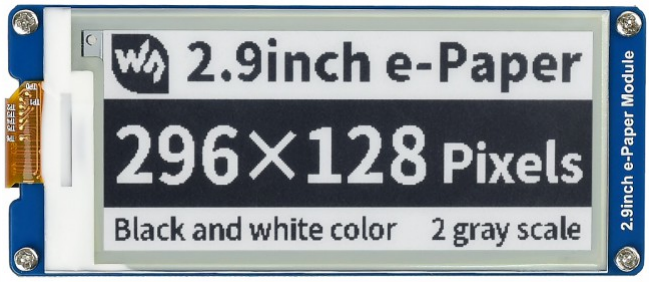
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

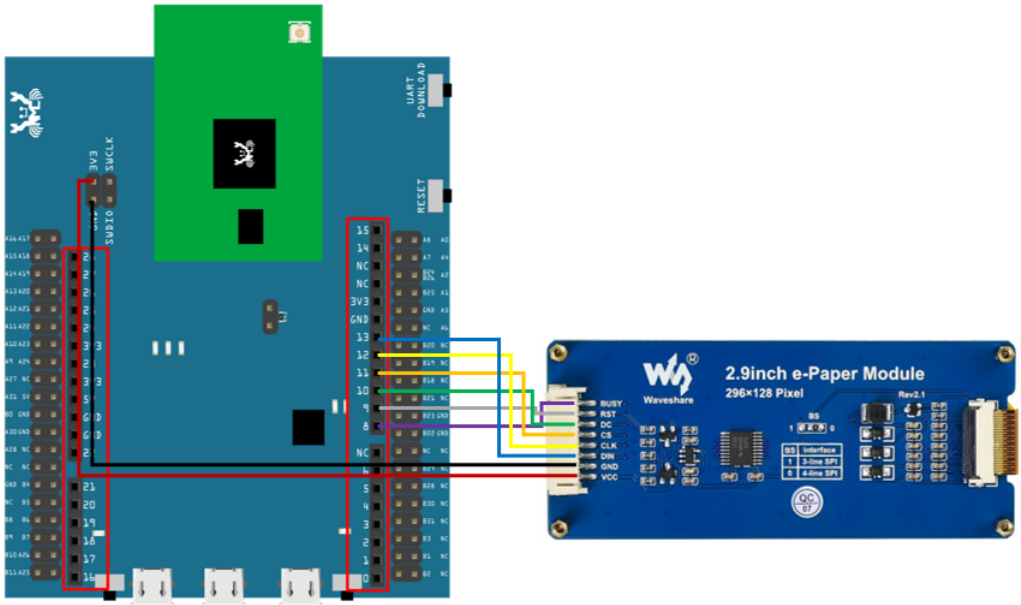
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

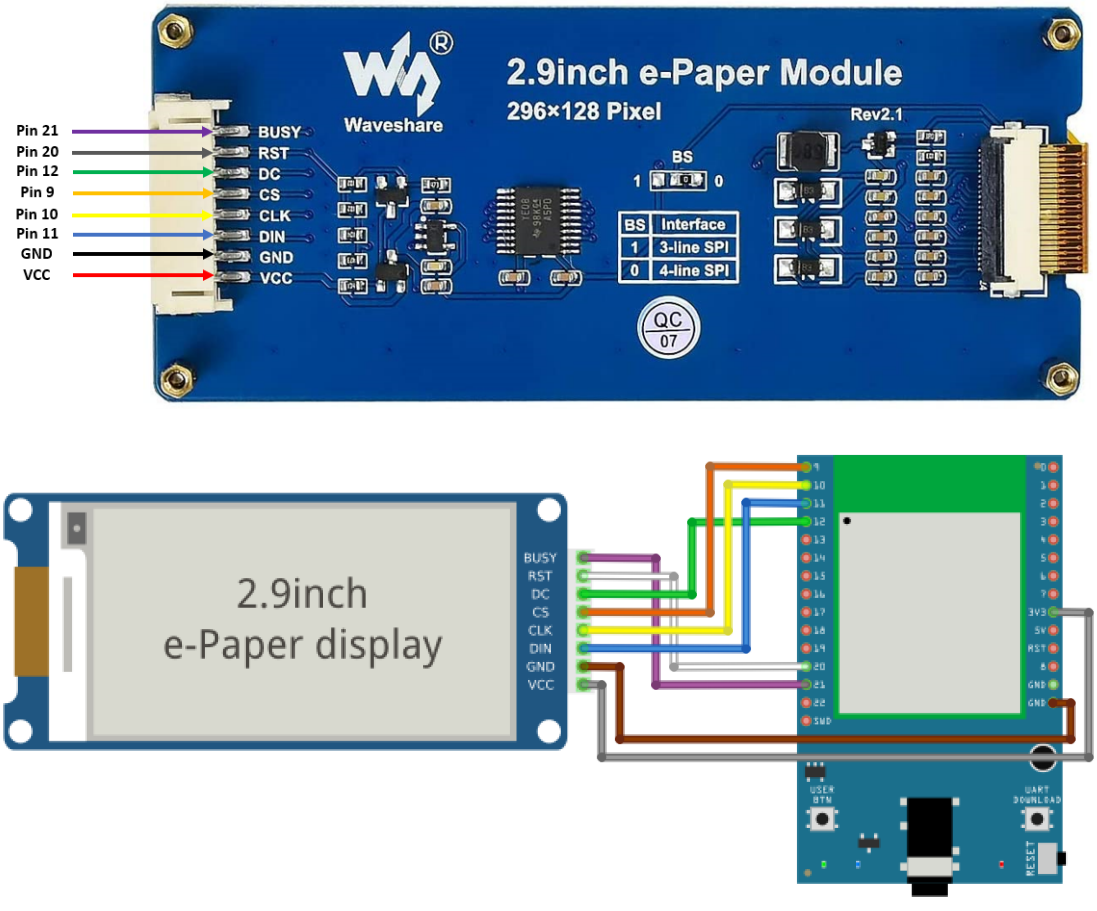
Front view of the e-Paper Module:



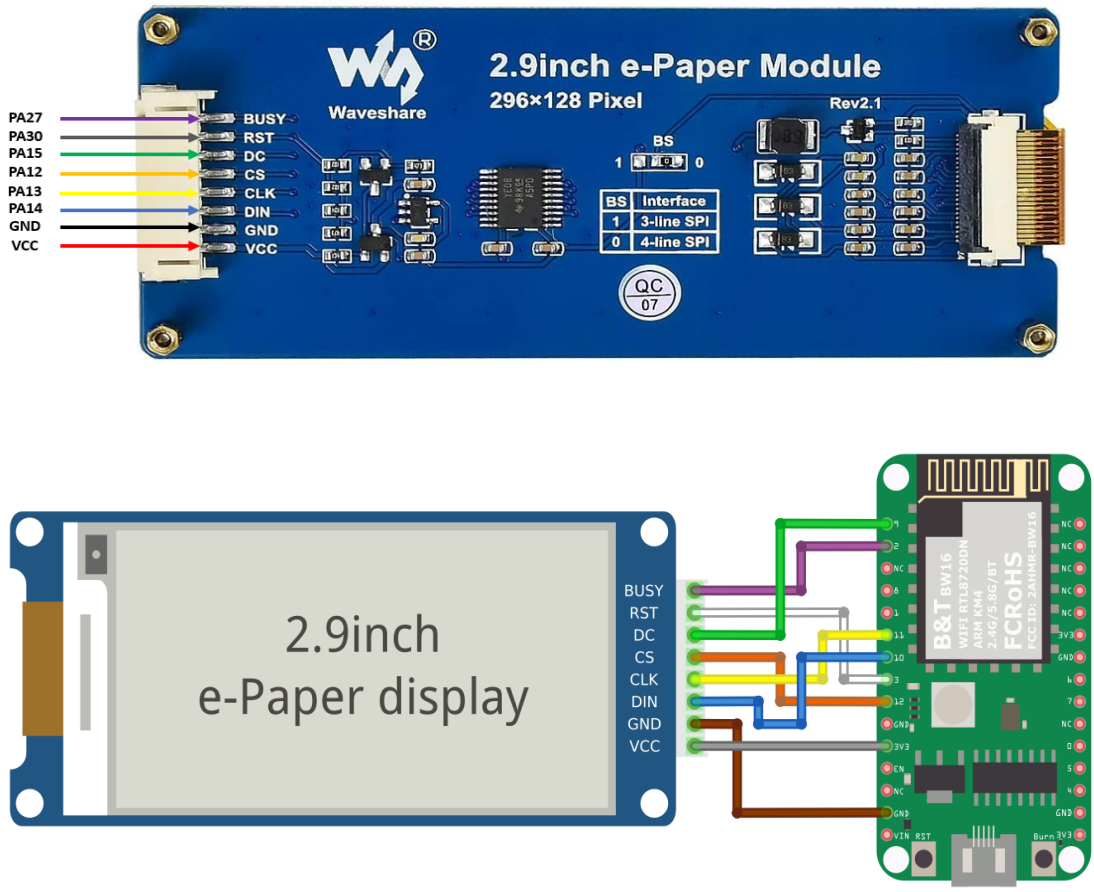
AMB21 / AMB22 Wiring Diagram:



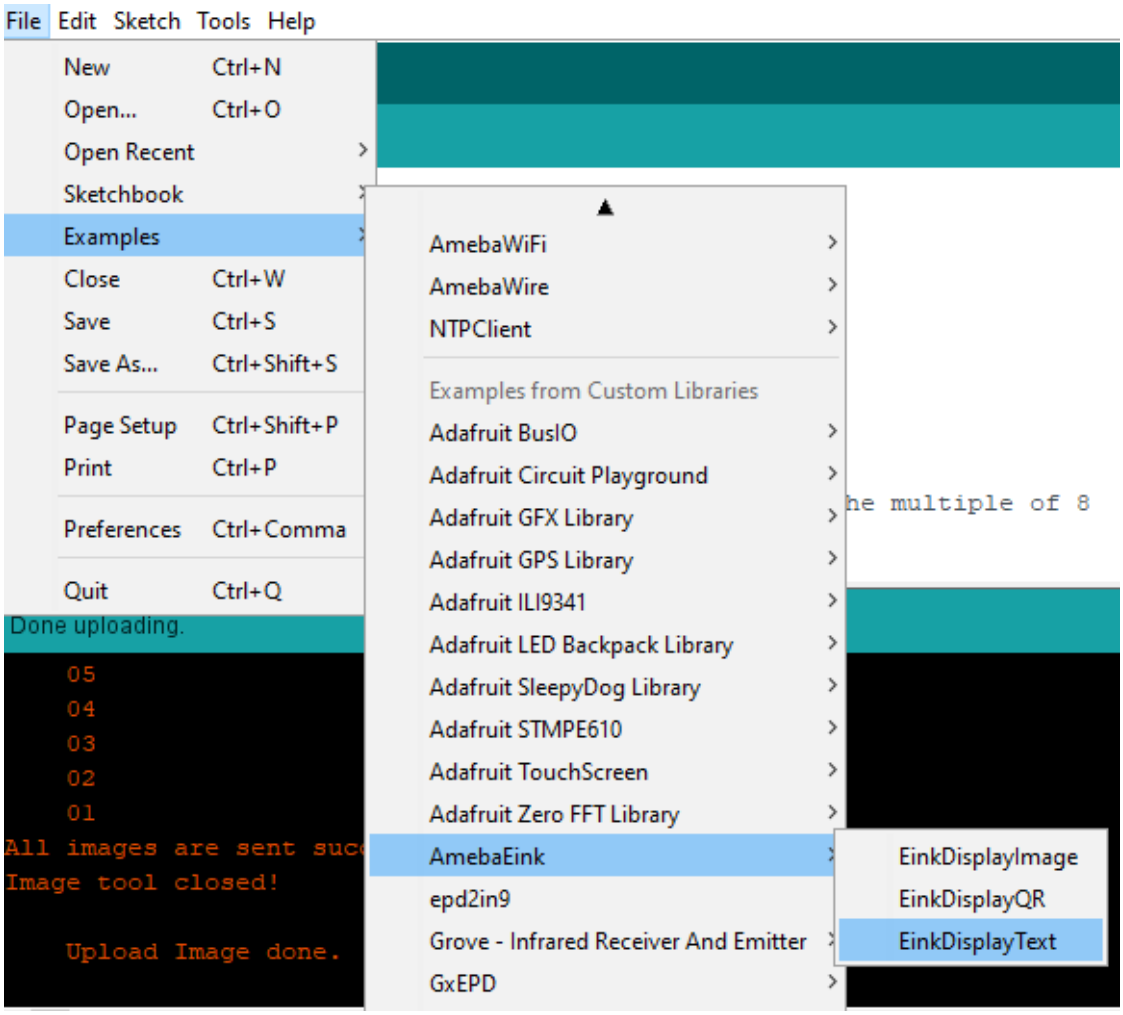
AMB23 Wiring Diagram:



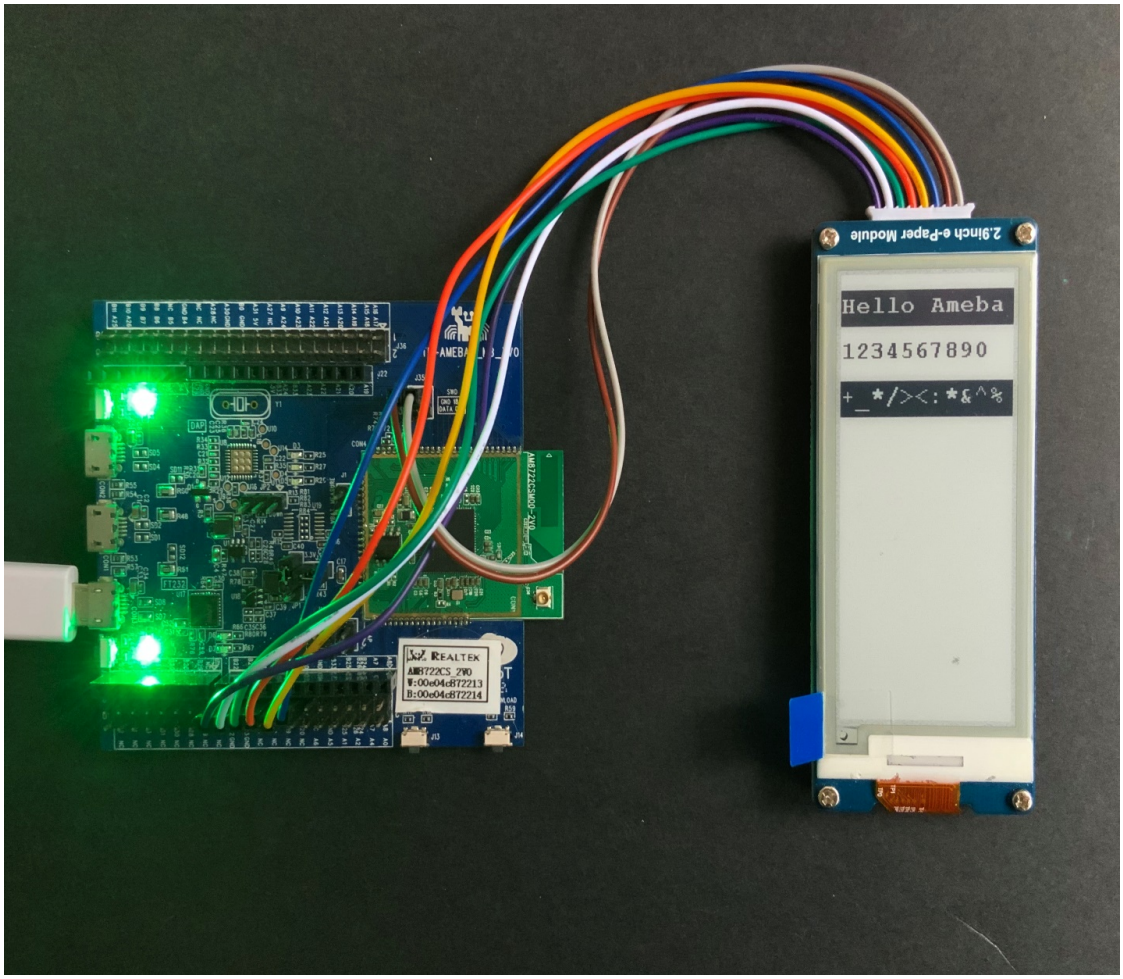
BW16 Wiring Diagram:



Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" -> "`Examples`" -> "`AmebaEink`" -> "`EinkDisplayText`":



Upload the code to the board and press the Reset button after the uploading is done. You will find these texts displayed on the board:



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html

E-Paper - Display User-generated QR Code

Materials

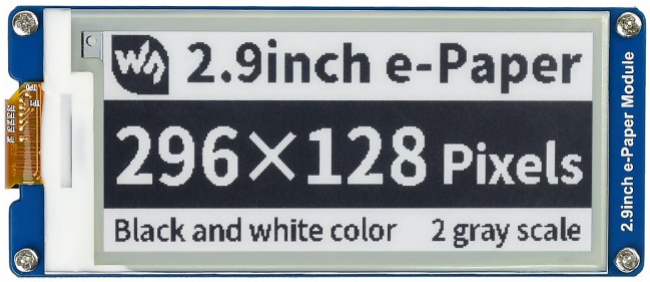
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

Example

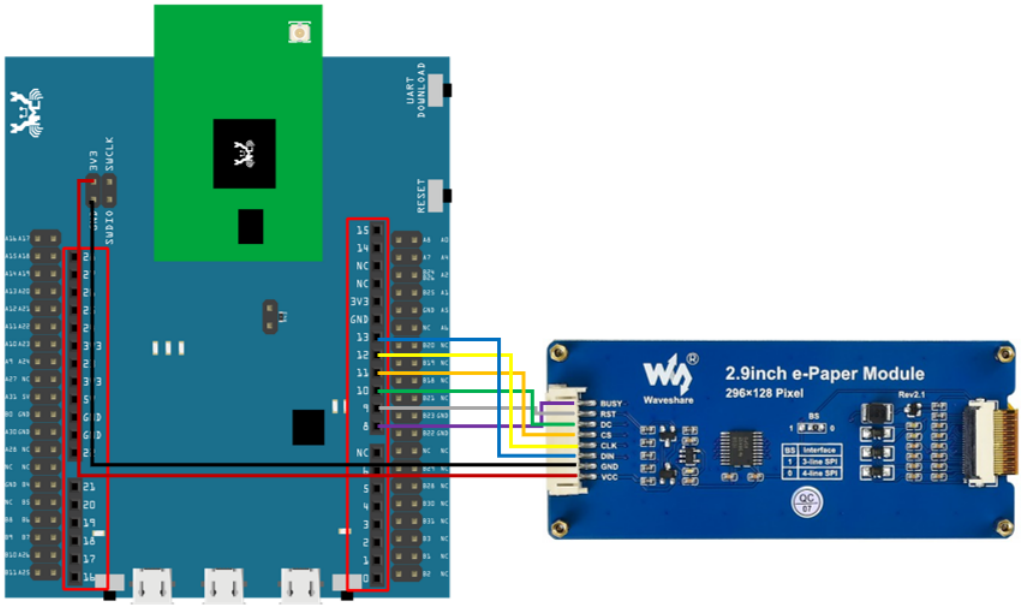
In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.

The 2.9” active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

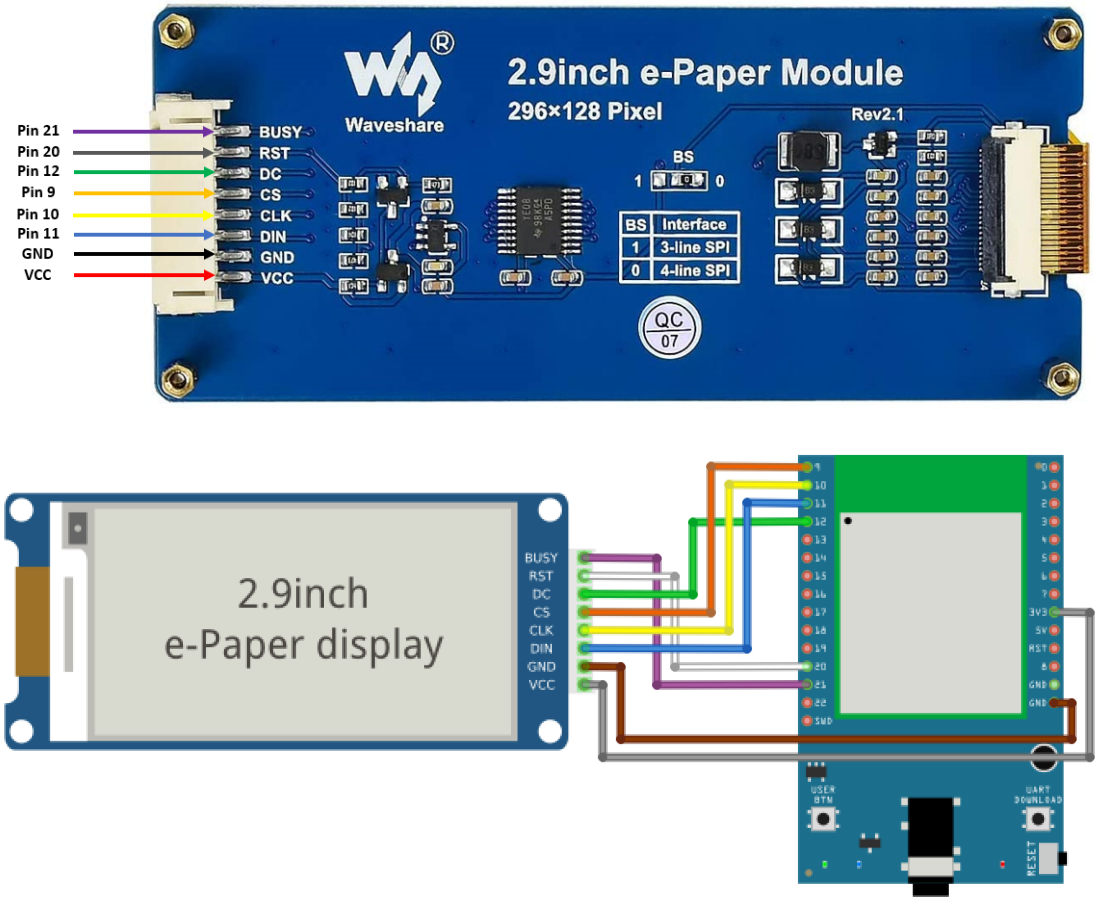
Front view of the e-Paper Module:



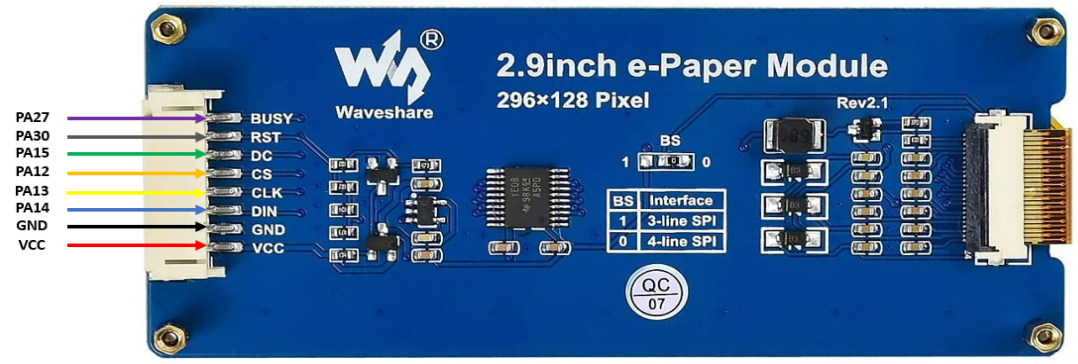
AMB21 / AMB22 Wiring Diagram:

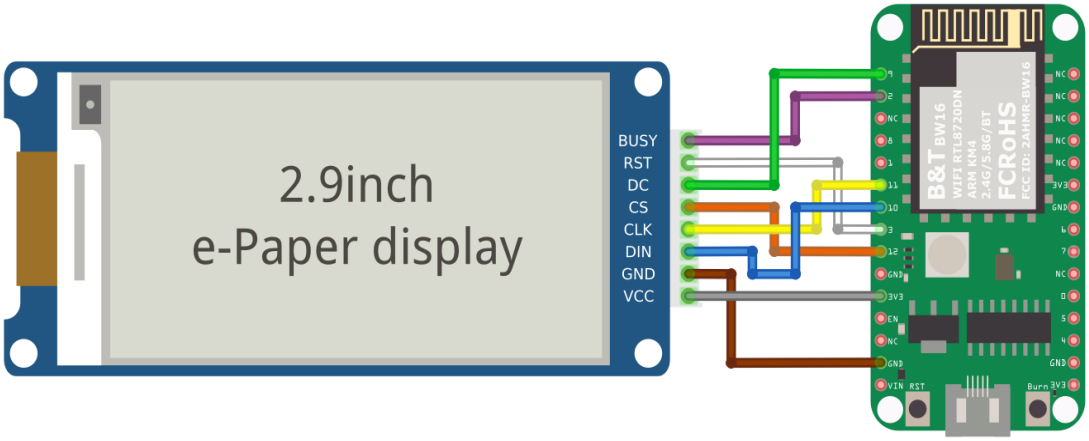


AMB23 Wiring Diagram:

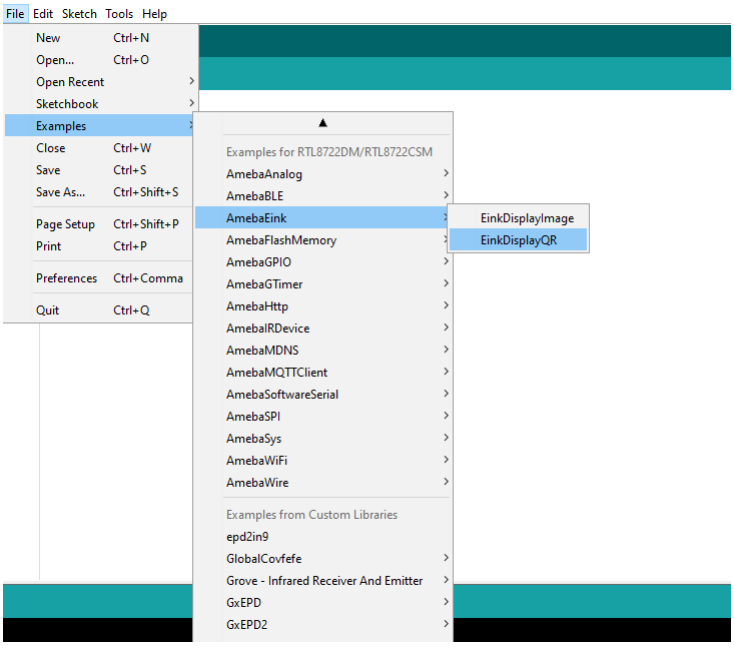


BW16 Wiring Diagram:

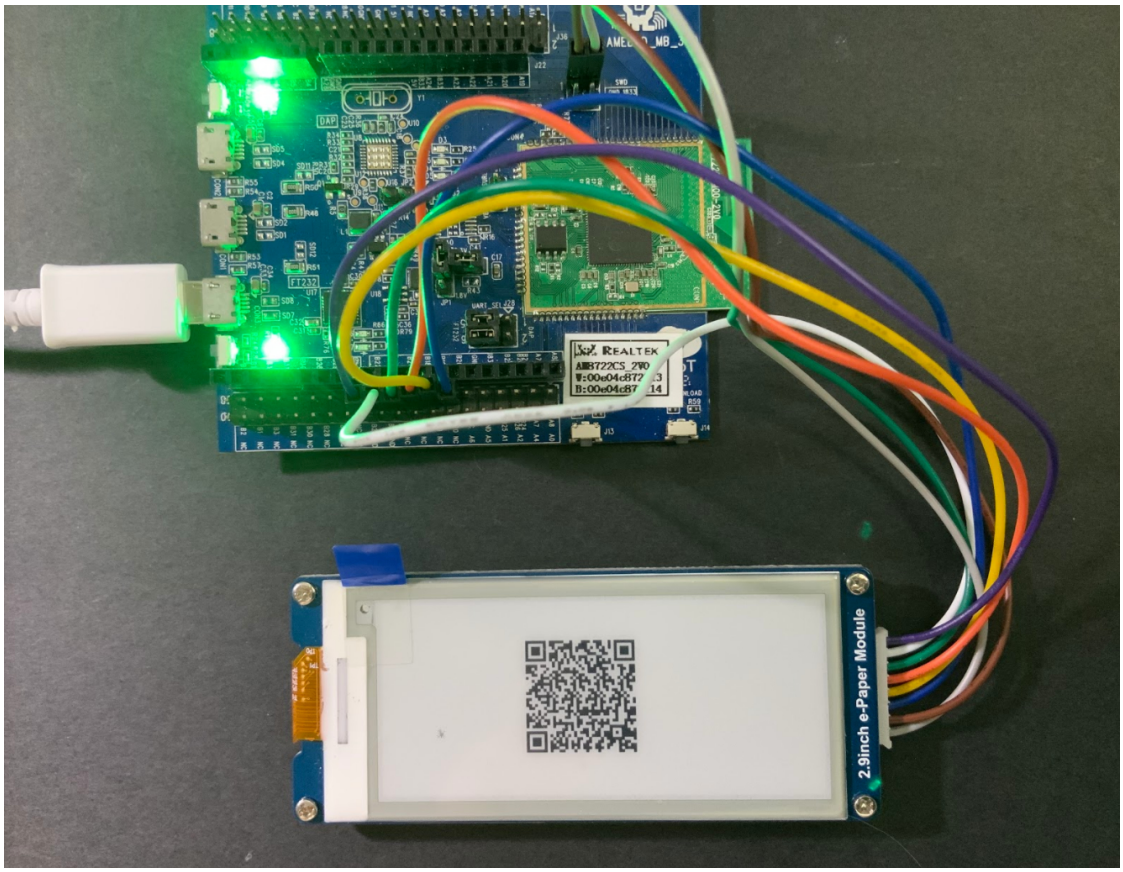




Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`DisplayQR`":



Modify the URL in the loop() section as your wish, after that, verify and upload the code to the Ameba board. Upon successfully upload the sample code and press the reset button, a QR code generated based on the URL of your input will be shown on the E-Paper module. The QR code showing below leads to our Ameba IoT official website: Ameba ARDUINO



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html
[3] A simple library for generating QR codes in C, optimized for processing and memory-constrained systems: https://github.com/ricmoo/QRCode#data-capacities

FatfsSDIO – File system in SD card

Materials

- AmebaD [AMB23] x 1
- MicroSD card

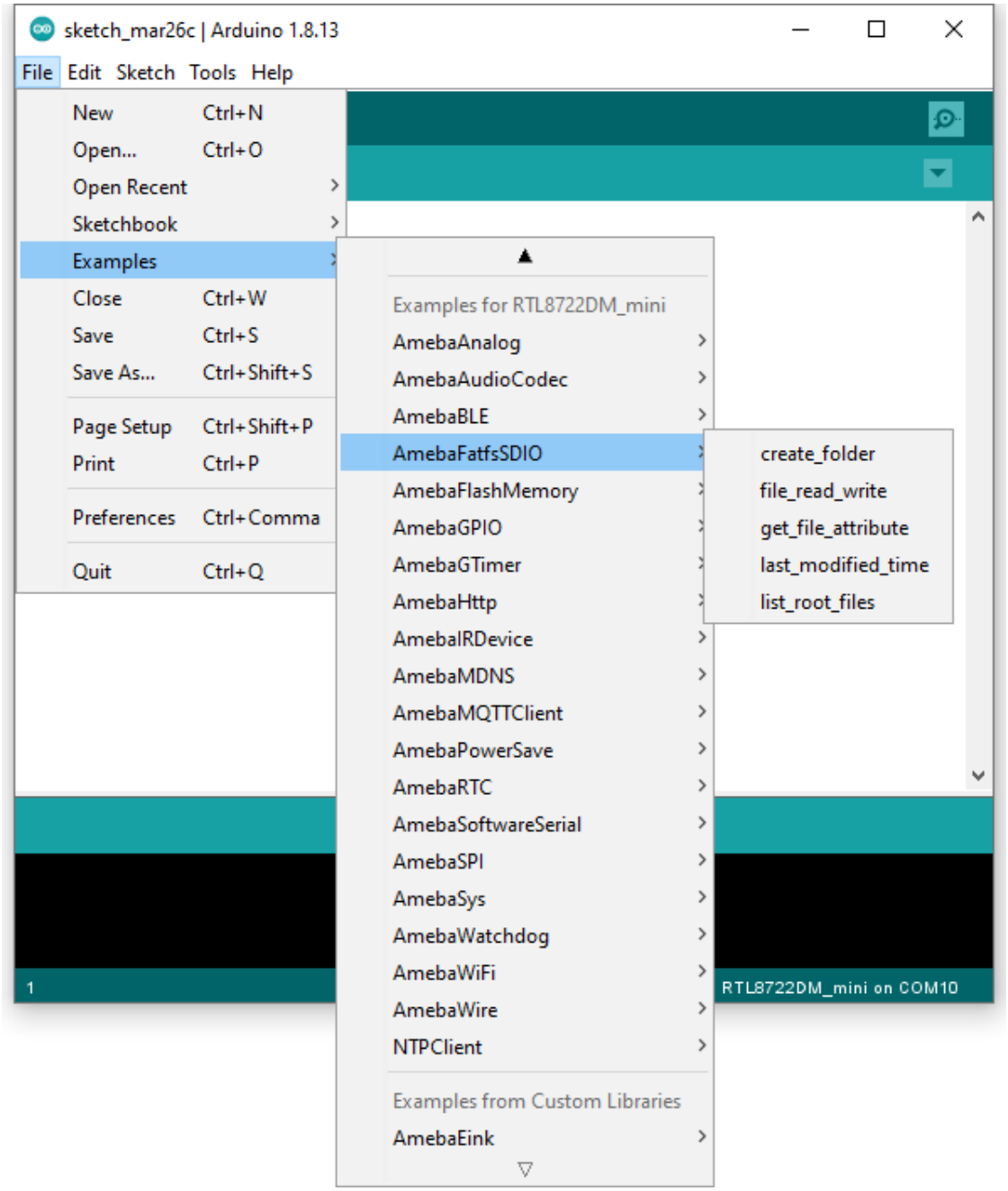
Example

Procedure

Insert a MicroSD card into the onboard SD card reader of RTL8722DM MINI board.

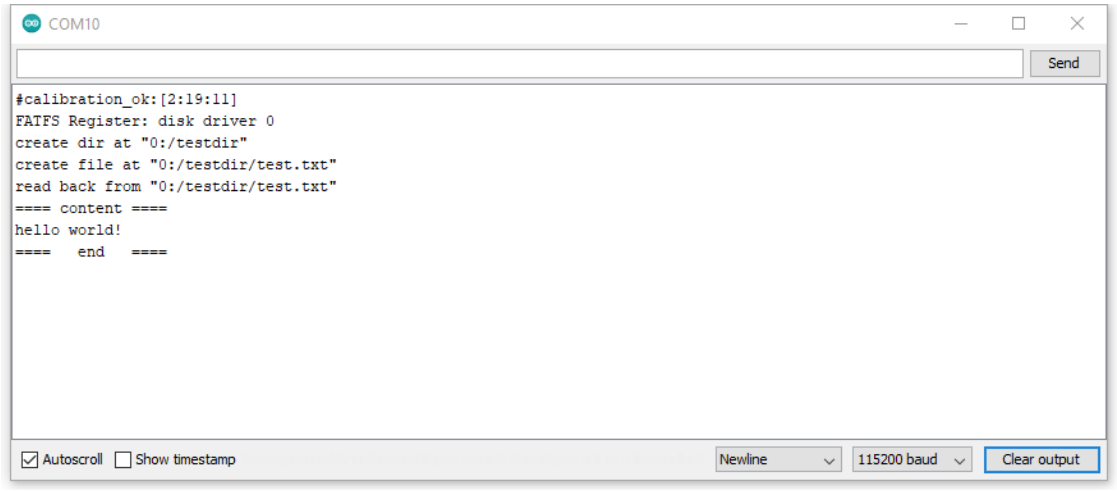
Example 01 create_folder

Open the example, `"Files" -> "Examples" -> “AmebaFatfsSDIO” -> “create_folder”`.

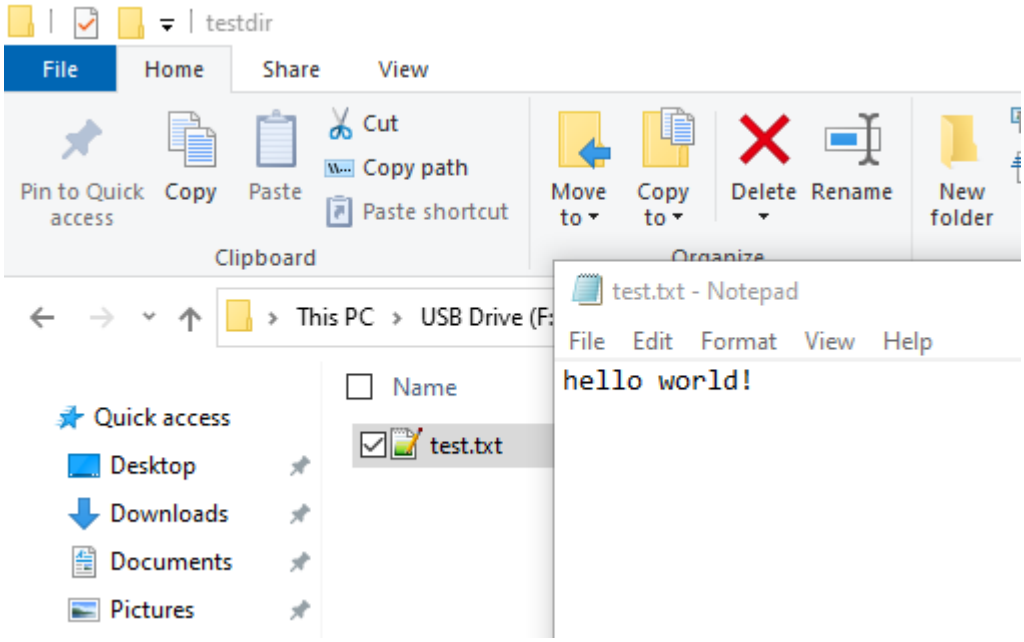


Upload the code and press the reset button on Ameba once the upload is finished.

In the sample code, we first create a folder “testdir”, then text file “test.txt” with content “hello world!”. Read the file and print content to serial monitor.



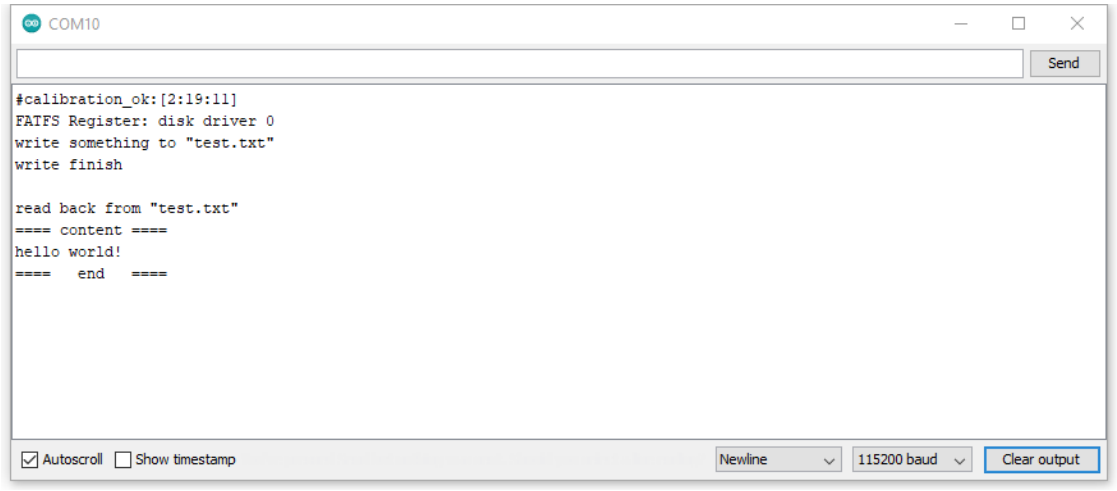
Next, insert SD card into card reader, and check whether the operations succeeded.



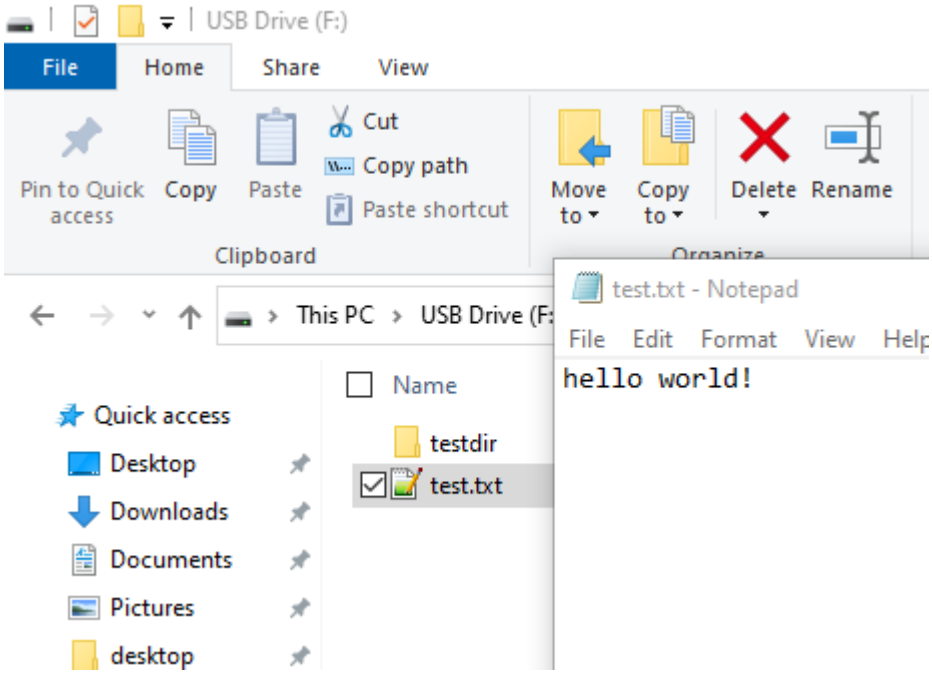
Example 02 file_read_write

Open the example, `"Files" -> "Examples" -> “AmebaFatfsSDIO” -> “file_read_write”`. Upload the code and press the reset button on Ameba once the upload is finished.

In the sample code, we create text file “test.txt” with content “hello world!”. Read the file and print content to serial monitor.

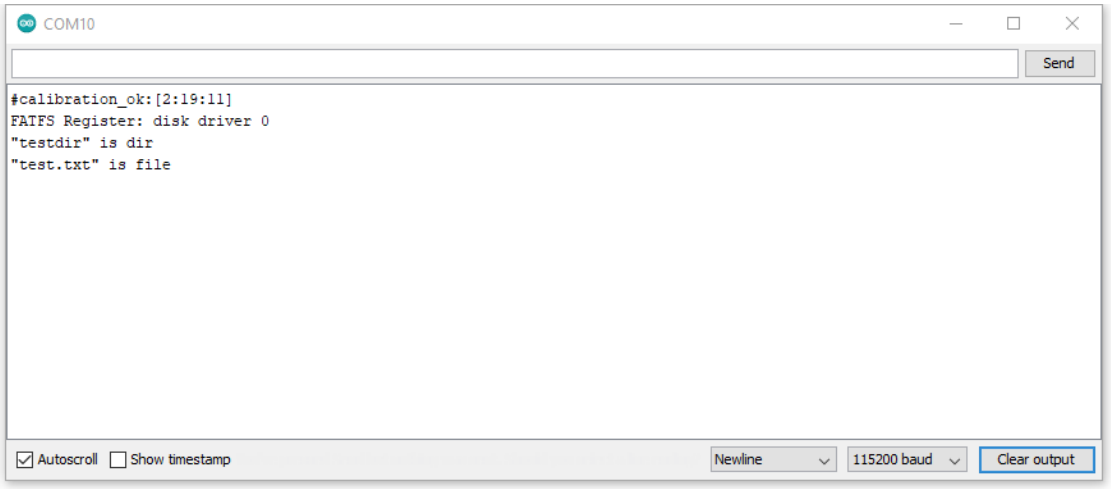


Next, insert SD card into card reader, and check whether the operations succeeded.



Example 03 get_file_attribute

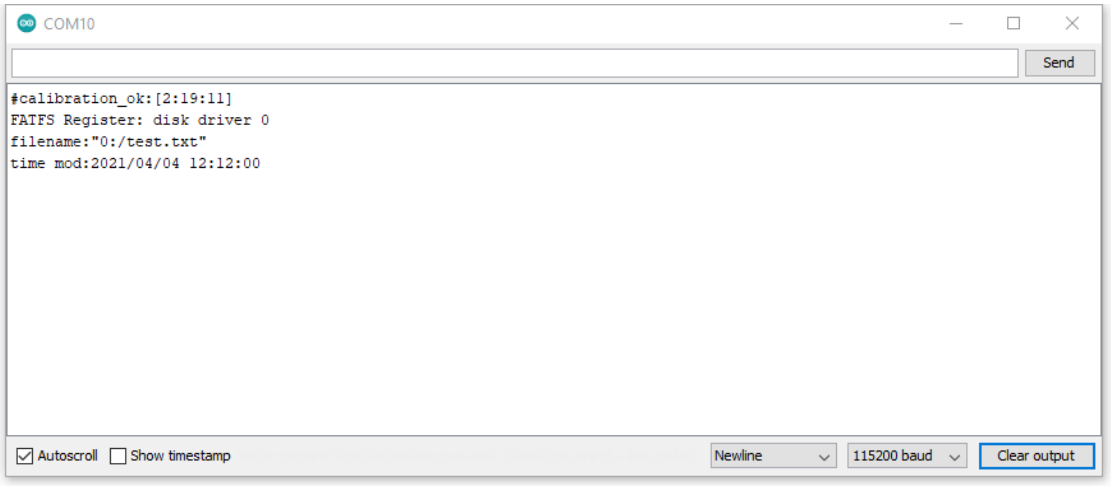
Open the example, `"Files" -> "Examples" -> “AmebaFatfsSDIO” -> “get_file_attribute”`.
Upload the code and press the reset button on Ameba once the upload is finished.
In the sample code, system will print put all file attribute to serial monitor.



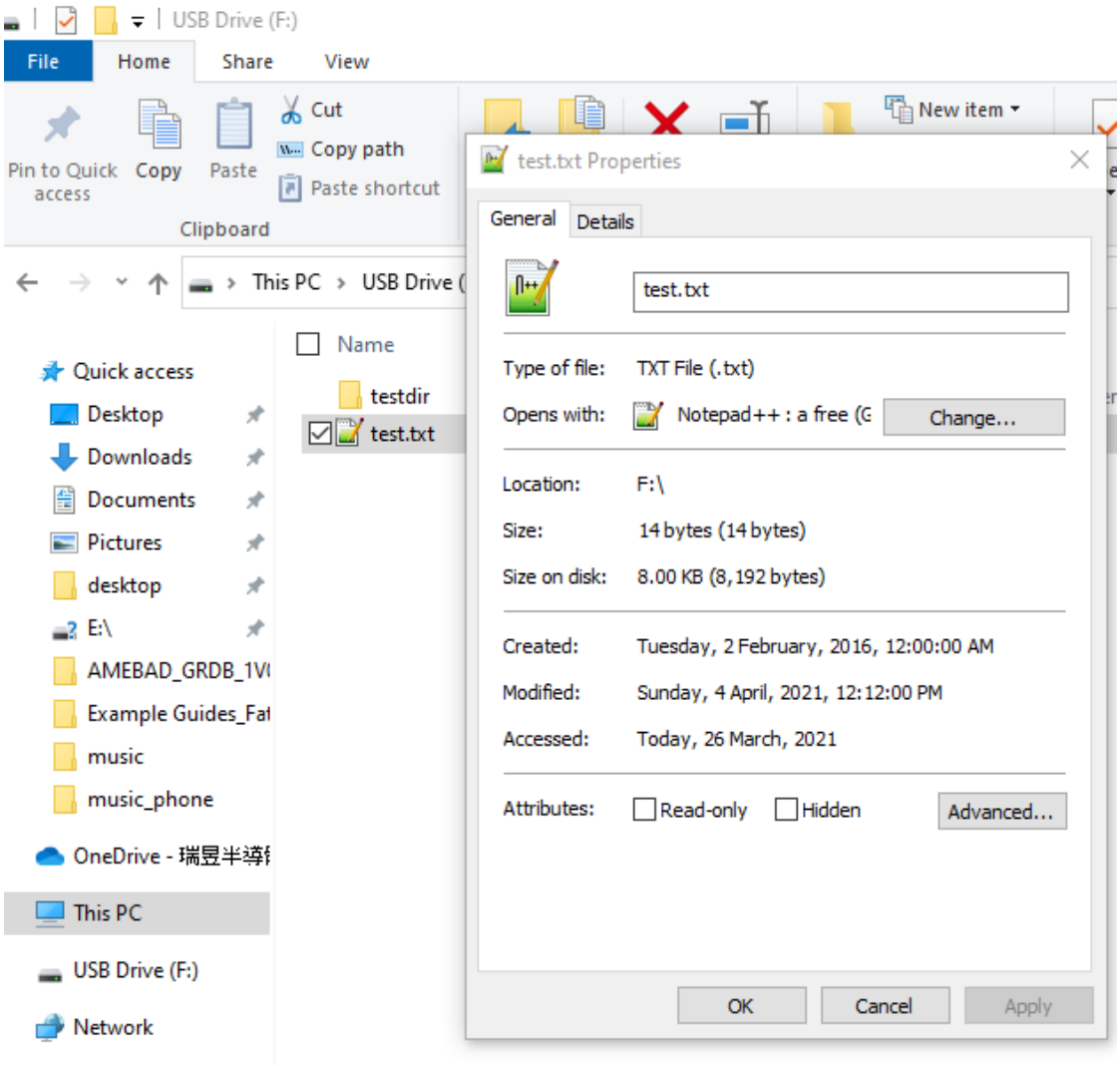
Next, insert SD card into card reader, and check whether the operations succeeded. In this case, we already know the attribute should be folder “testdir” and text file “test.txt”by refer the above pictures.

Example 04 last_modified_time

Open the example, `"Files" -> "Examples" -> “AmebaFatfsSDIO” -> “last_modified_time”`.
Upload the code and press the reset button on Ameba once the upload is finished.
In the sample code, system will print put the target file last modified time to serial monitor.

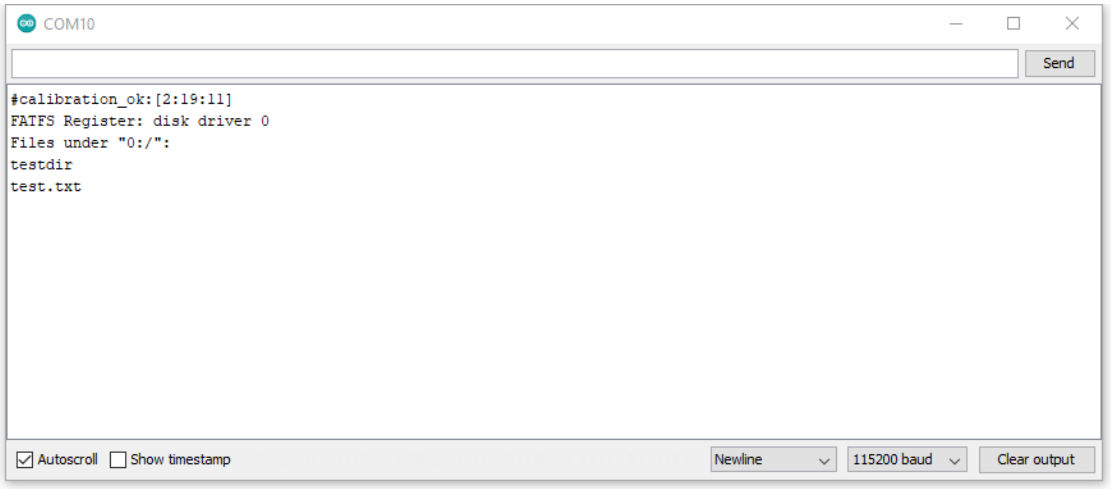


Next, insert SD card into card reader, and check whether the operations succeeded.



Example 05 list_root_files

Open the example, `"Files" -> "Examples" -> “AmebaFatfsSDIO” -> “list_root_files”`.
Upload the code and press the reset button on Ameba once the upload is finished.
In the sample code, system will print put all root file to serial monitor.



Next, insert SD card into card reader, and check whether the operations succeeded. In this case, we already know the root files folder "testdir" and text file "test.txt"by refer the above pictures.

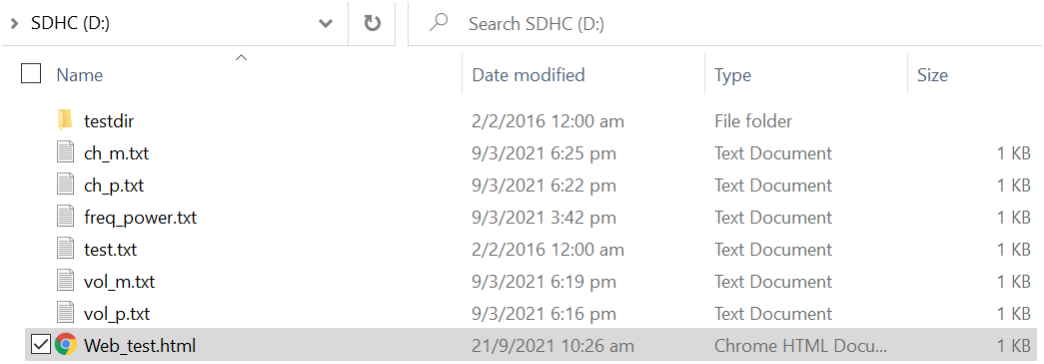
FatfsSDIO – Read And Open HTML File From SD Card

Materials

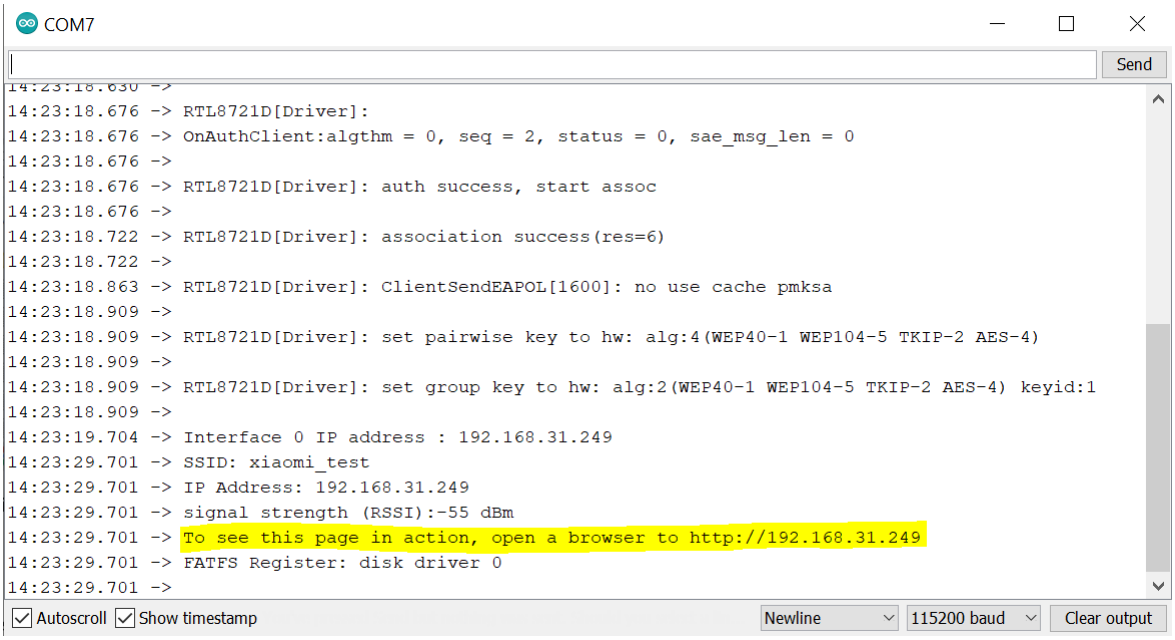
- AmebaD [AMB23] x 1
- MicroSD card

Example

Insert the MicroSD card into your computer and copy the HTML file to your SD card (Note: put the file at outside and do not put it inside of any folder in the SD card). Here is a HTML sample for testing, "Web_test.html".



Then insert the MicroSD card into the onboard SD card reader of RTL8722DM MINI board.
Open the example, "`Files`" -> "`Examples`" -> "`AmebaFatfsSDIO`" -> "`read_html_from_SD_card`"
Upload the code and press the reset button on Ameba once the upload is finished. When the connection is established, you should be able to see the message "To see this page in action, open a browser to http://xxx.xxx.xxx.xxx" in the serial monitor as shown in the figure:



Next, open the address stated in serial monitor in the browser of your laptop or cell phone under the same WiFi domain. You will see the following display in your browser:

Now you have successfully read and opened the html file saved in your SD card.

Flash Memory - Store data in FlashEEProm

Preparation

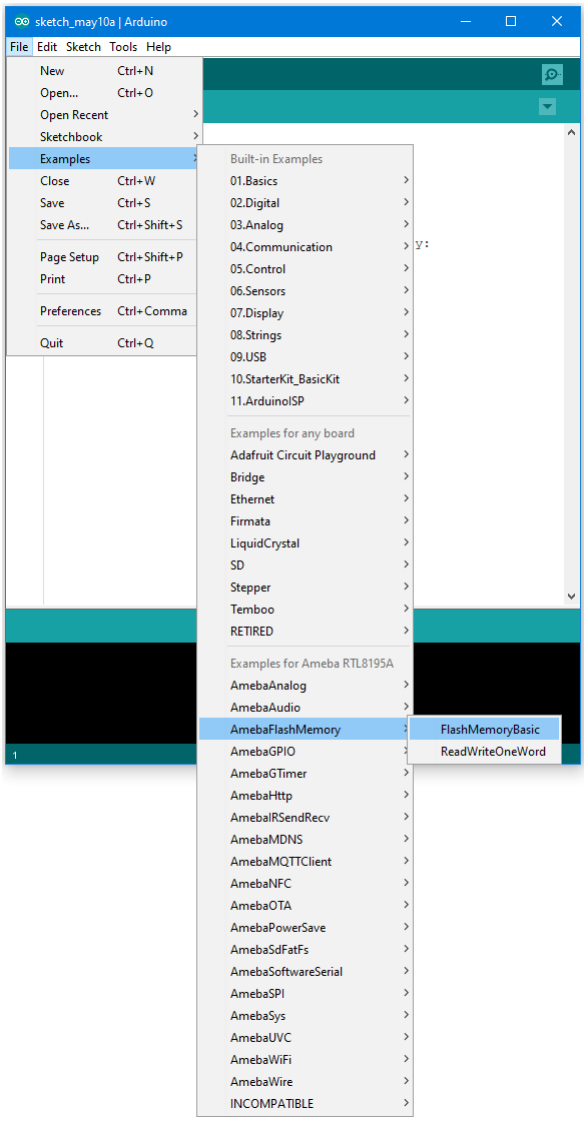
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

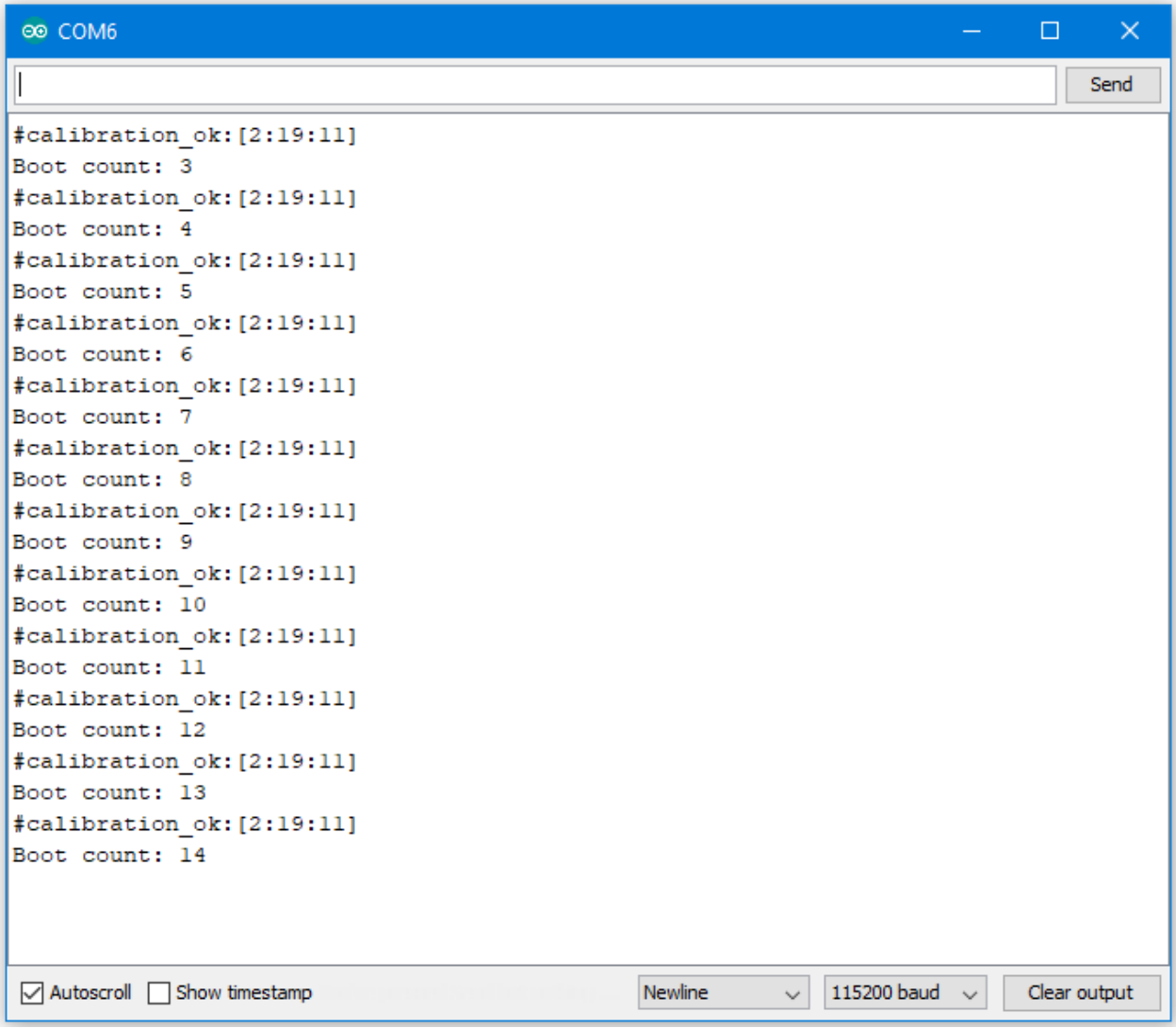
Ameba provides Flash Memory component for data storage and the data can be preserved when the power is off if necessary, e.g., compiled program. To avoid the memory space overlapped with the program on Ameba, the Flash API uses the tail part of the address space, with sector size 4K.

In this example, we store the value of boot times in flash memory. Every time Ameba reboots, it reads the boot times from flash, increases the value by 1, and writes it back to flash memory.

First open the example, “File” -> “Example” -> “AmebaFlashMemory” -> “FlashMemoryBasic”



Compile and upload to Ameba, then press the reset button.
Open the Serial Monitor, press the reset button for a few times. Then you can see the boot times value increases.



Code Reference

By default, the Flash Memory API uses address 0xFF000~0xFFFFF to store data.

There is limitation when writing to flash memory. That is, you can not directly write data to the same address you used in last write. To do that correctly, you need erase the sector first. The Flash API of Ameba uses a 4K SRAM to record the user modification and do the erase/write task together.

Use `FlashMemory.read()` to read from Flash memory.
Use `FlashMemory.buf[0] = 0x00;` to manipulate the 4K buf.
Use `FlashMemory.update();` to update the data in buf to Flash Memory.

Flash Memory - Use Flash Memory Larger Than 4K

Preparation

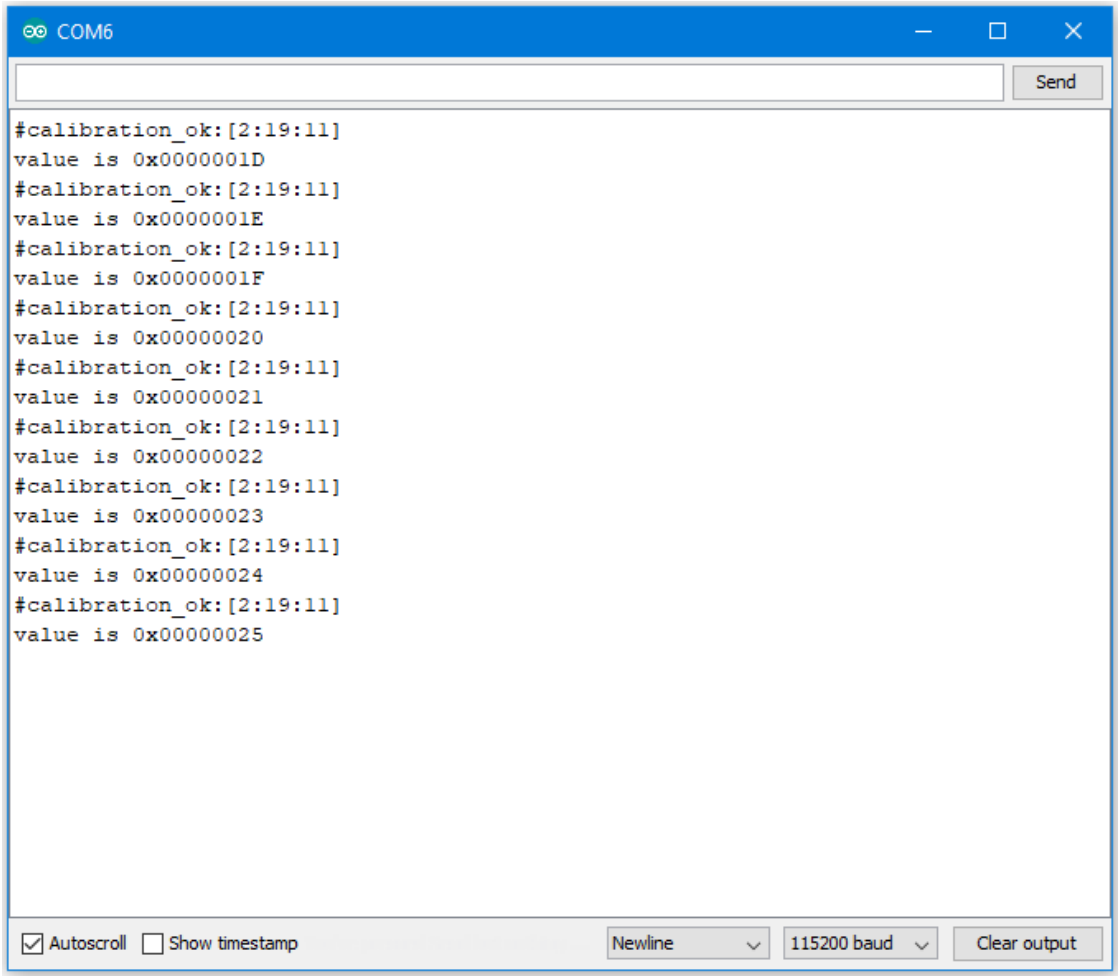
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Flash Memory API uses memory of 4K bytes, which is normally sufficient for most application. However, larger memory can be provided by specifying a specific memory address and required size.

First, open the sample code in "`File`" -> "`Examples`" -> "`AmebaFlashMemory`" -> "`ReadWriteOneWord`"

In this example, we specify the starting address of flash memory is 0xFC000 and size is 0x4000 (The default starting address is 0xFF000 and size is 0x1000).
Then calculate correct address according to the specified offset and perform read/write operation. In the sample code we use offset 0x3F00, that is, 0xFC000 + 0x3F00 = 0xFFF00 in flash. We set the value to 0 at first, then increase by 1 every time Ameba reboots.



Code Reference

We can use the flash api we used in previous flash memory example, but we need to use `begin()` function to specify the desired starting address and memory size.

```
FlashMemory.begin(0xFC000, 0x4000);
```

Use `readWord()` to read the value stored in a memory address. In the example, we read the value stored in memory offset 0x3F00, that is 0xFC000 + 0x3F00 = 0xFFF00. `readWord()` function reads a 32-bit value and returns it.

```
value = FlashMemory.readWord(0x3F00);
```

Use `writeWord()` to write to a memory address. The first argument is the memory offset, the second argument is the value to write to memory.

```
FlashMemory.writeWord(0x3F0C, value);
```

GPIO - Measure Distance By Ultrasound Module

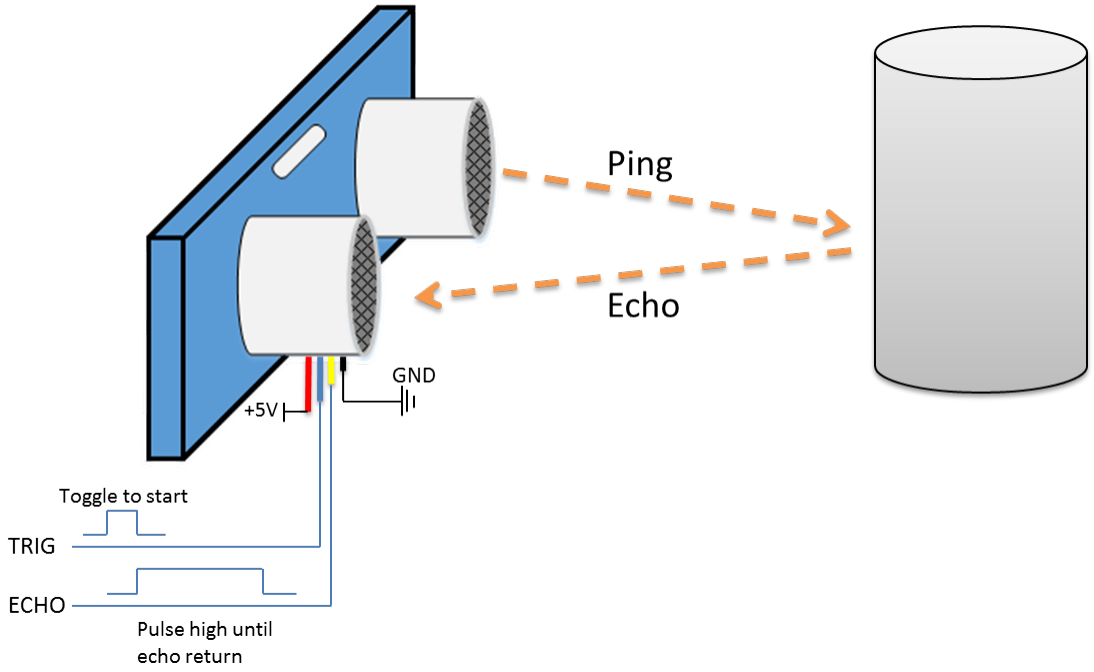
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- HC-SR04 Ultrasonic x 1
- Dropping resistor or Level converter

Example

HC-SR04 is a module that uses ultrasound to measure the distance. It looks like a pair of eyes in its appearance, therefore it's often installed onto robot-vehicle or mechanical bugs to be their eyes.

The way it works is that first we "toggle high" the TRIG pin (that is to pull high then pull low). The HC-SR04 would send eight 40kHz sound wave signal and pull high the ECHO pin. When the sound wave returns back, it pull low the ECHO pin.

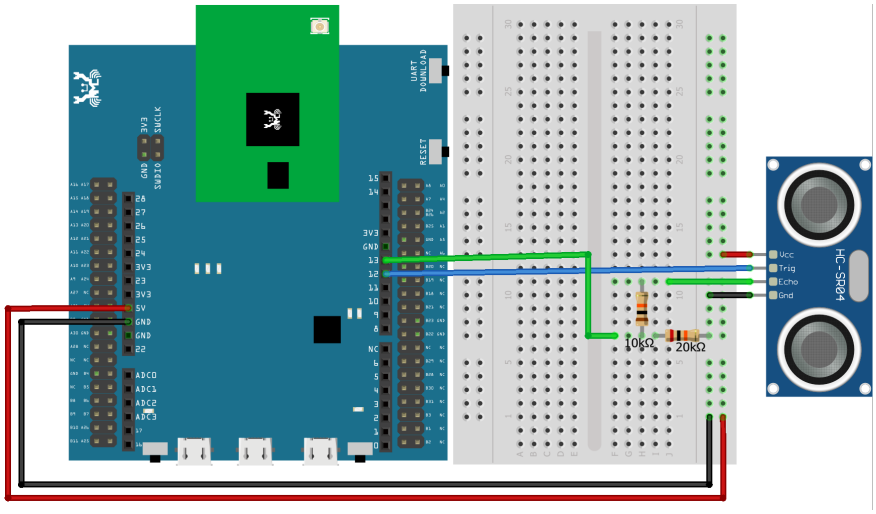


Assume the velocity of sound is 340 m/s, the time it takes for the sound to advance 1 cm in the air is 340*100*10^-6 = 29 us.

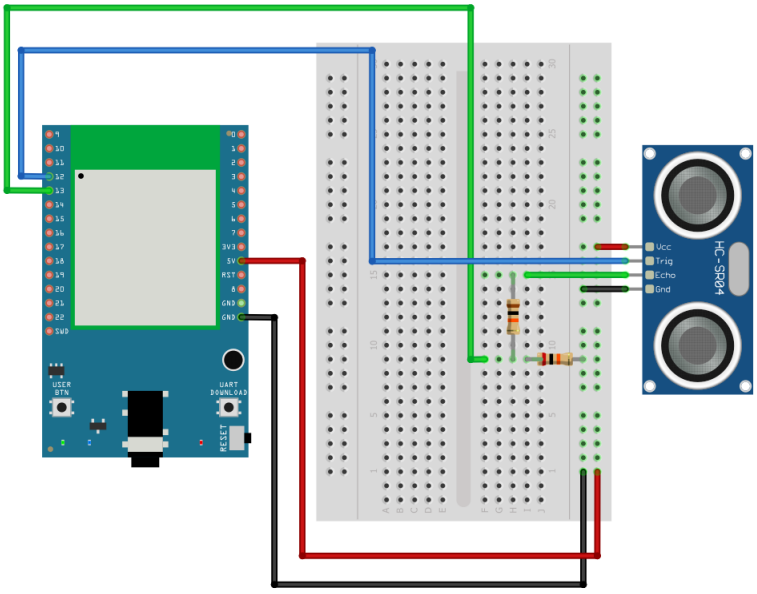
The sound wave actually travels twice the distance between HC-SR04 and the object, therefore the distance can be calculated by (time/29) / 2 = time / 58.

The working voltage of HC-SR04 is 5V. When we pull high the ECHO pin to 5V, the voltage might cause damage to the GPIO pin of Ameba. To avoid this situation, we need to drop the voltage as follows:

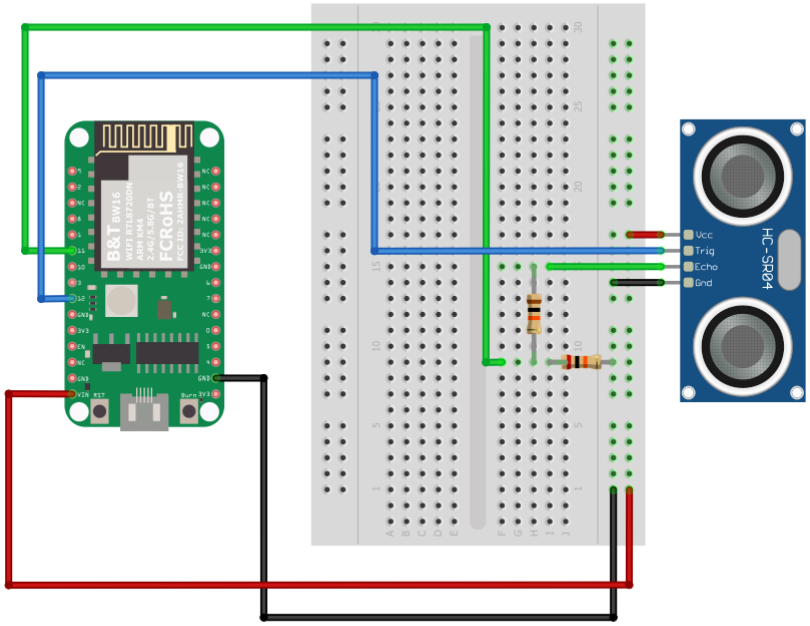
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

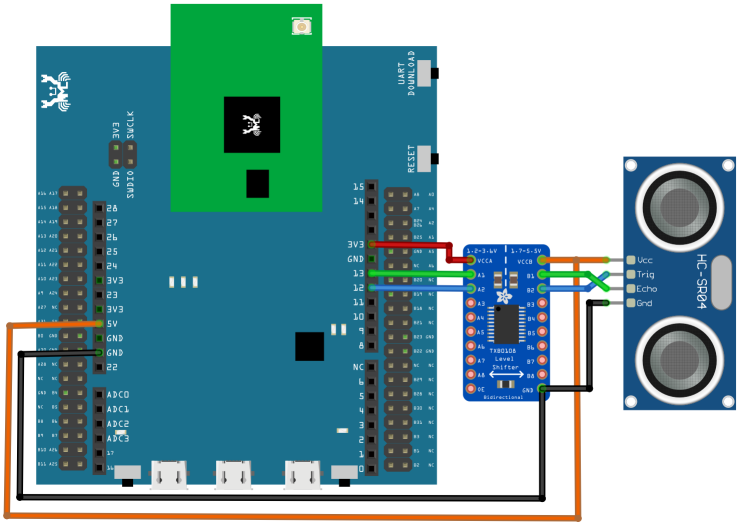


BW16 Wiring Diagram:

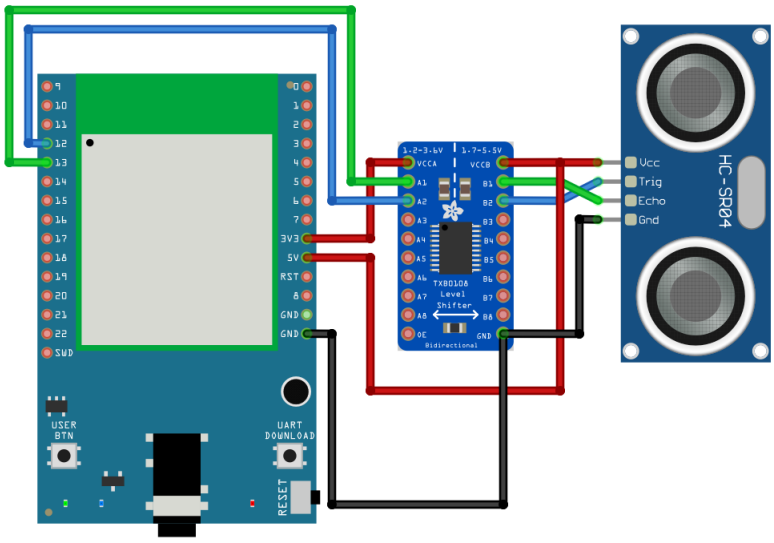


We pick the resistors with resistance 1:2, in the example we use 10kΩ and 20kΩ.
If you do not have resistors in hand, you can use level converter instead.The TXB0108 8 channel level converter is a suitable example:

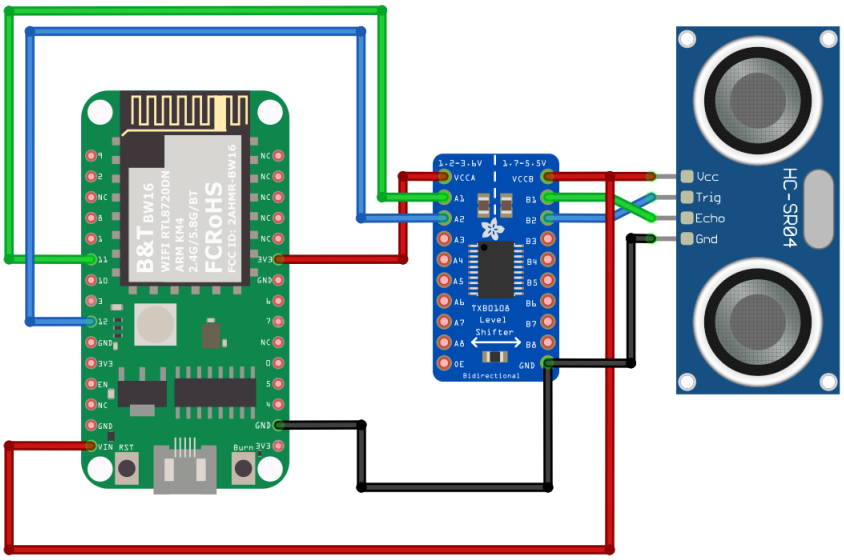
AMB21 / AMB22 Wiring Diagram:



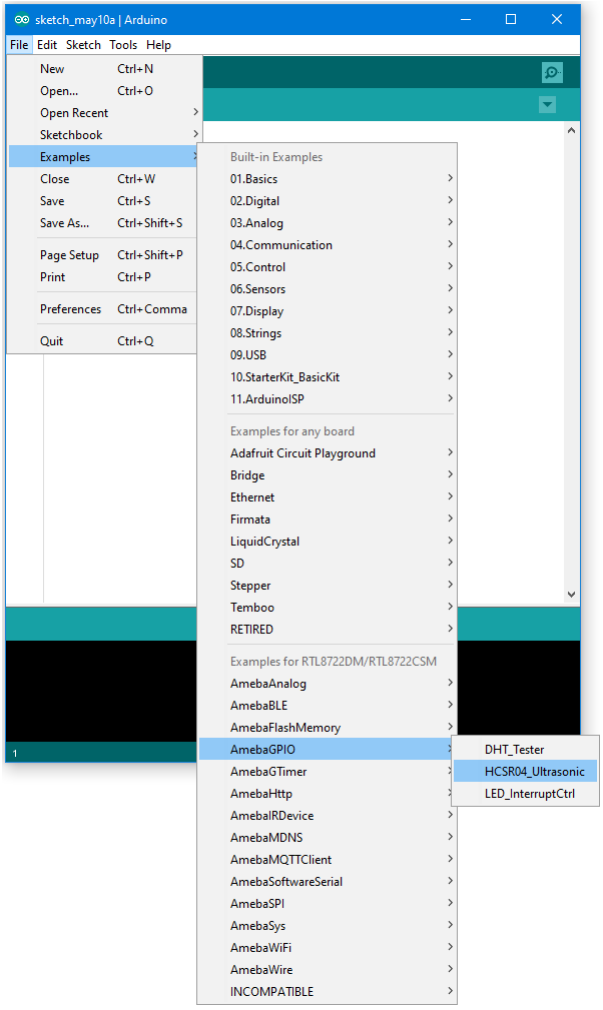
AMB23 Wiring Diagram:



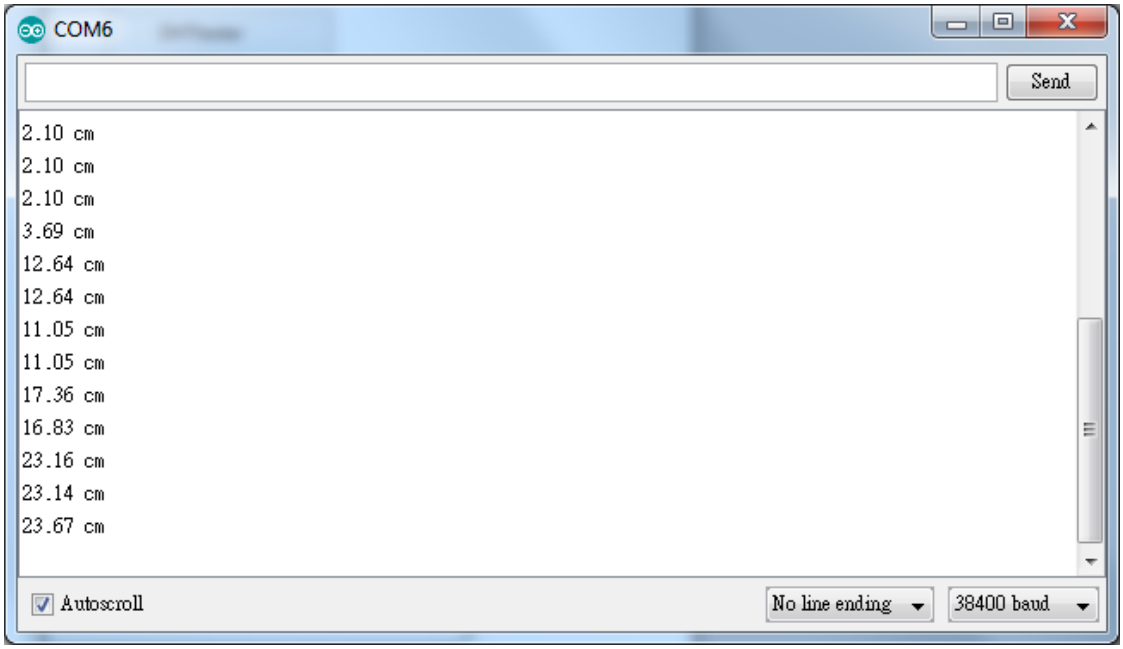
BW16 Wiring Diagram:



Next, open the sample code in "`File`" -> "`Examples`" -> "`AmebaGPIO`" -> "`HCSR04_Ultrasonic`"



Compile and upload to Ameba, then press the reset button. Open the Serial Monitor, the calculated result is output to serial monitor every 2 seconds.



Note that the HCSR04 module uses the reflection of sound wave to calculate the distance, thus the result can be affected by the surface material of the object (e.g., harsh surface tends to cause scattering of sound wave, and soft surface may cause the sound wave to be absorbed).

Code Reference

Before the measurement starts, we need to pull high the TRIG pin for 10us and then pull low. By doing this, we are telling the HC-SR04 that we are about to start the measurement:

```
digitalWrite(trigger_pin, HIGH);
delayMicroseconds(10);
digitalWrite(trigger_pin, LOW);
```

Next, use pulseIn to measure the time when the ECHO pin is pulled high.

```
duration = pulseIn (echo_pin, HIGH);
```

Finally, use the formula to calculate the distance.

```
distance = duration / 58;
```

GPIO - Measure Temperature and Humidity

Preparation

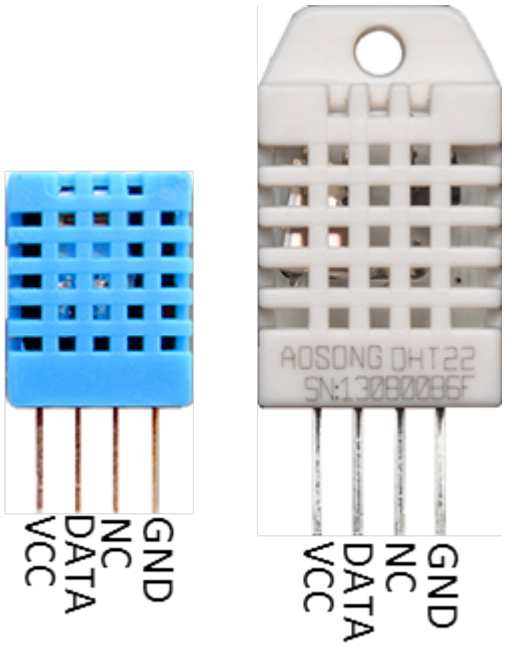
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21

Example

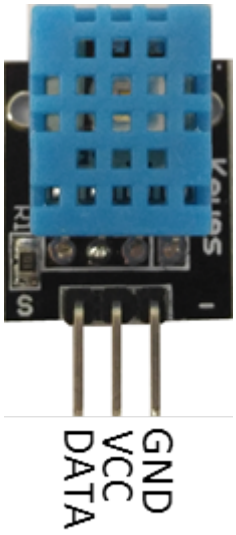
DHT11 is a temperature and humidity sensor which operates at voltage 3.3V~5V. At room temperature, the measurable range of the humidity is 20% ~ 90%RH with ±5%RH precision, the measurable range of the temperature is 0 ~ 50℃ with ±2℃ precision.

Another choice of temperature and humidity sensor is DHT22 sensor, which has better precision. Its measurable range of the humidity is 0%~100%RH with ±5%RH precision, the measurable range of the temperature is -40~125 ℃ with ±0.2℃ precision.

There are 4 pins on the sensor:



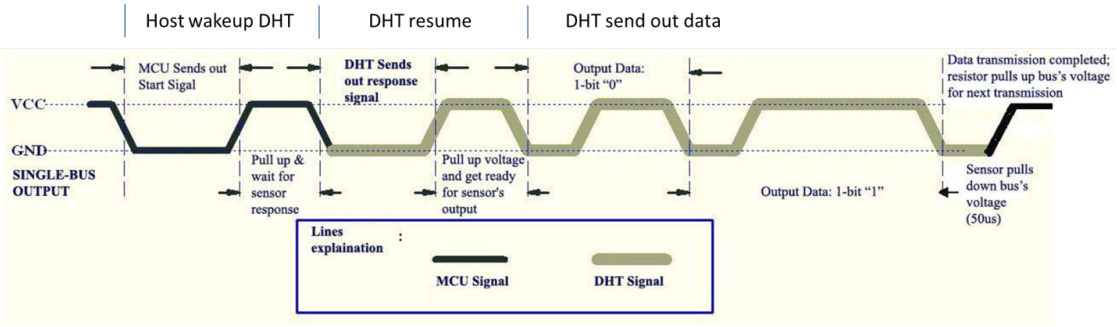
Since one of the 4 pins has no function, there are temperature/humidity sensors with only 3 pins on the market:



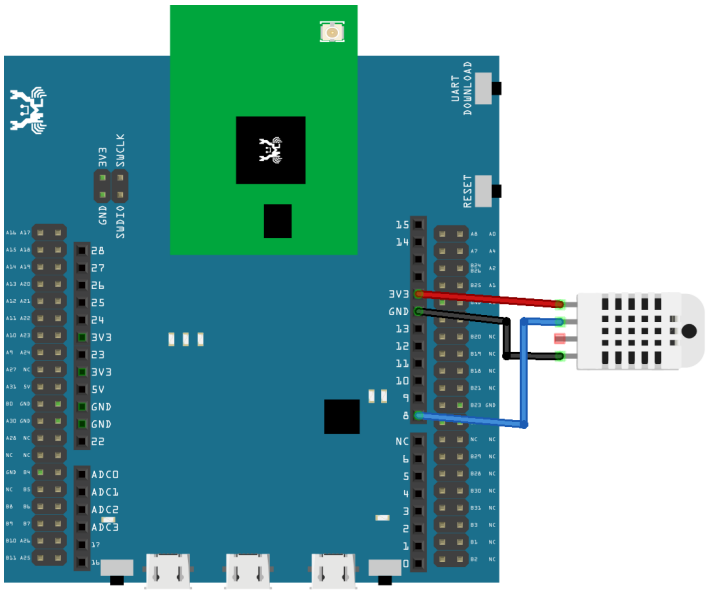
DHT is normally in the sleeping mode. To get the temperature/humidity data, please follow the steps:

1. Awake DHT: Ameba toggles low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital out to Ameba.

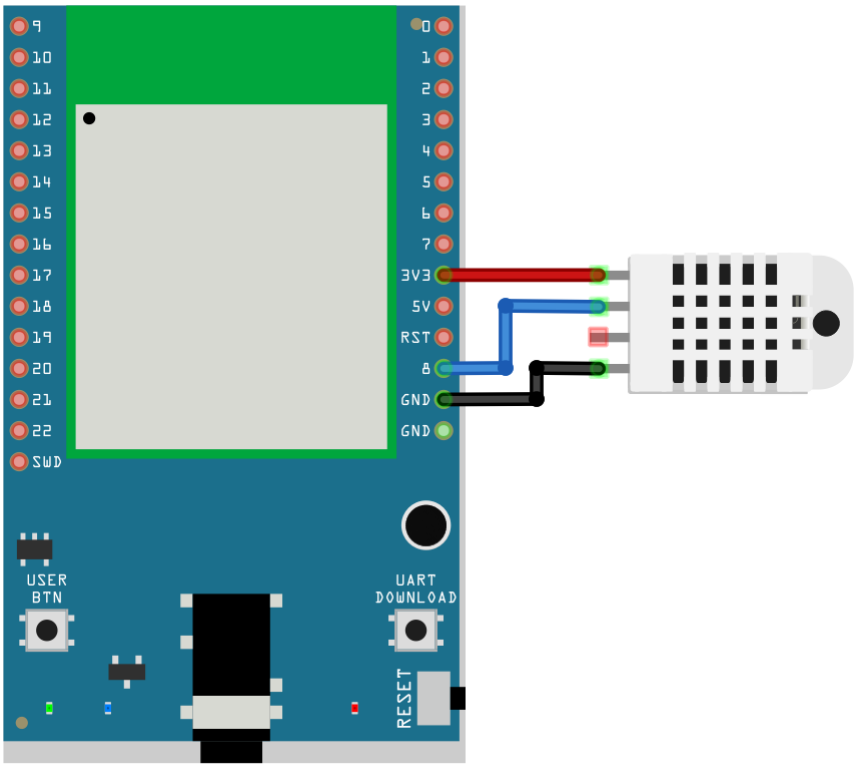
2. DHT response: DHT also toggle low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital in for Ameba.
3. DHT sends data: DHT sends out the temperature/humidity data (which has size 5 bytes) in a bit by bit manner. To represent each bit, DHT first pull low the DATA GPIO pin for a while and then pull high. If the duration of high is smaller than low, it stands for bit 0. Otherwise it stands for bit 1.



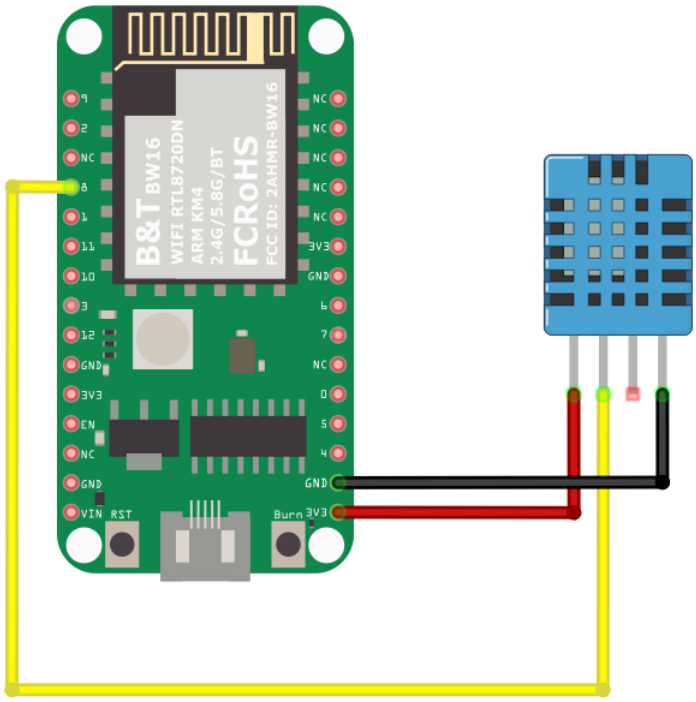
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

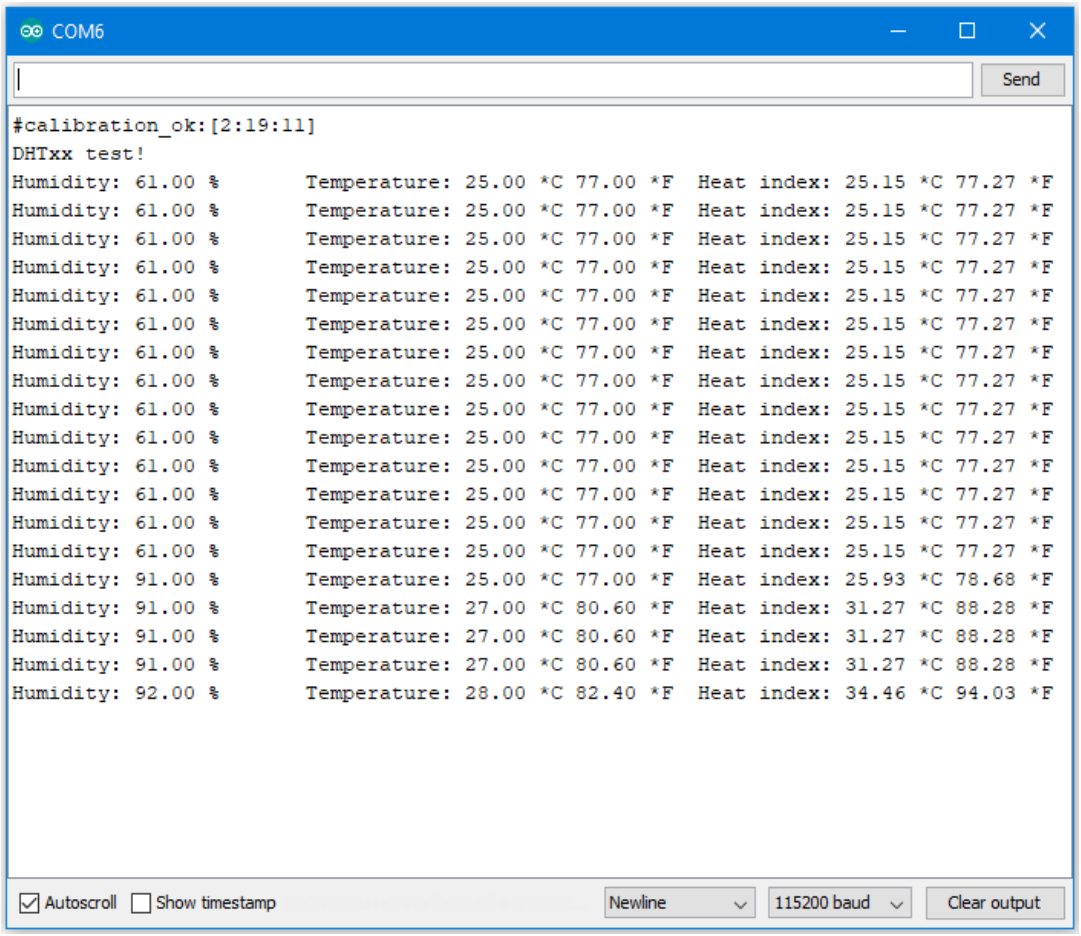


BW16 Wiring Diagram:



Open the sample code in "`Files`" -> "`Examples`" -> "`AmebaGPIO`" -> "`DHT_Tester`". Compile and

upload to Ameba, then press the reset button. The result would be shown on the Serial Monitor.



```
#calibration_ok:[2:19:11]
DHTxx test!
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.15 *C 77.27 *F
Humidity: 91.00 %       Temperature: 25.00 *C 77.00 *F  Heat index: 25.93 *C 78.68 *F
Humidity: 91.00 %       Temperature: 27.00 *C 80.60 *F  Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %       Temperature: 27.00 *C 80.60 *F  Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %       Temperature: 27.00 *C 80.60 *F  Heat index: 31.27 *C 88.28 *F
Humidity: 92.00 %       Temperature: 28.00 *C 82.40 *F  Heat index: 34.46 *C 94.03 *F
```

Code Reference

Use `dht.readHumidity()` read the humidity value, and use `dht.readTemperature()` to read the temperature value.

Every time we read the temperature/humidity data, Ameba uses the buffered temperature/humidity data unless it found the data has expired (i.e., has not been updated for over 2 seconds). If the data is expired, Ameba issues a request to DHT to read the latest data.

GPIO - Use GPIO Interrupt To Control LED

Preparation

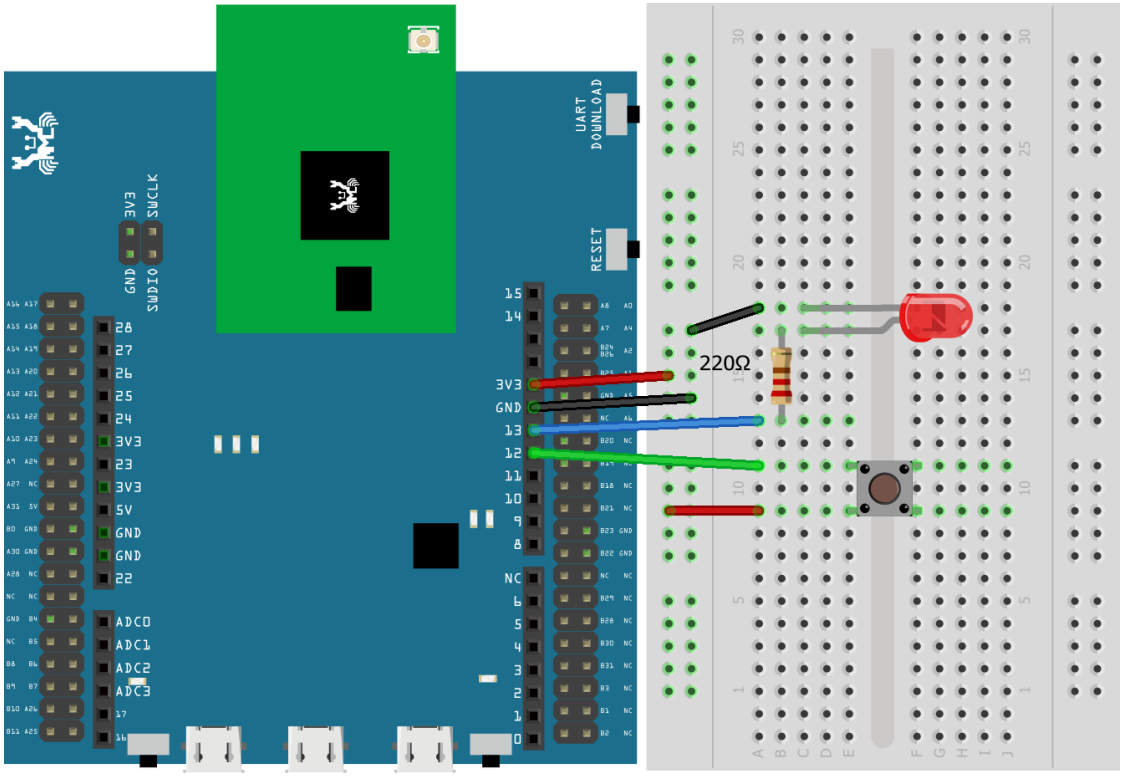
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- LED x 1
- Button x 1

Example

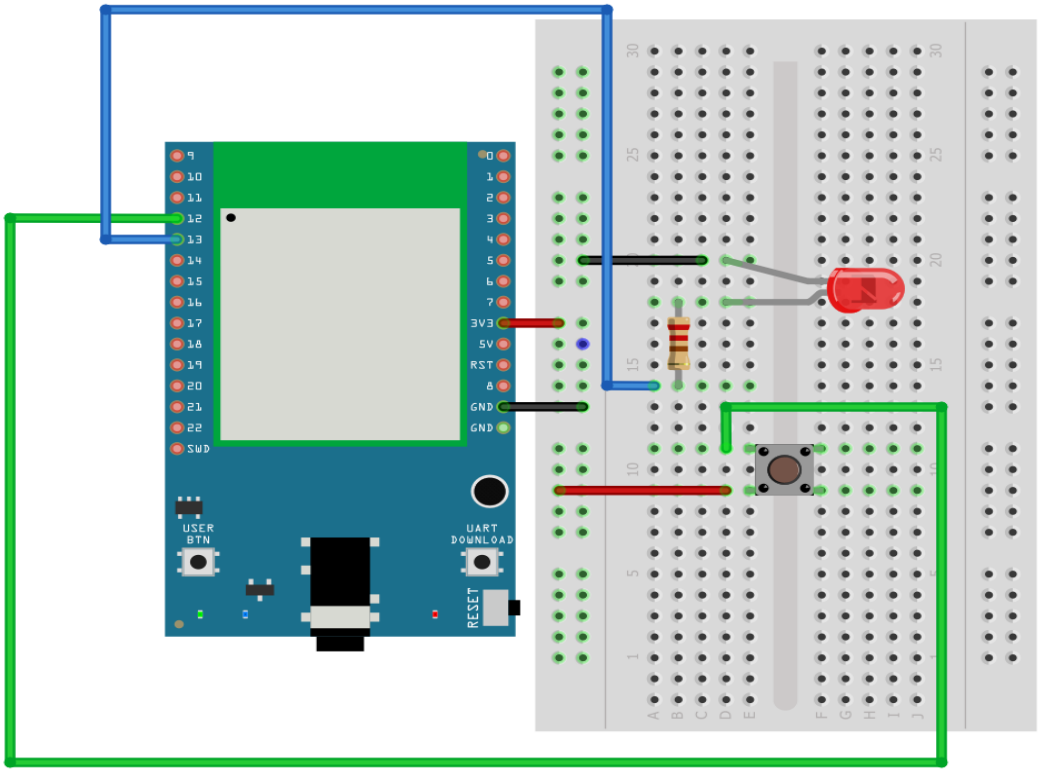
In this example, we use a button to trigger interrupt and control the LED. When we press and release the button, the LED dims, press and release the button again, and the LED lights.Note that in the Arduino example “Button and LED”, LED only lights when the button is pressed and hold, when we release the button, the LED dims.

Open the example, “`Files`” -> “`Examples`” -> “`AmebaGPIO`” -> “`LED_InterruptCtrl`”

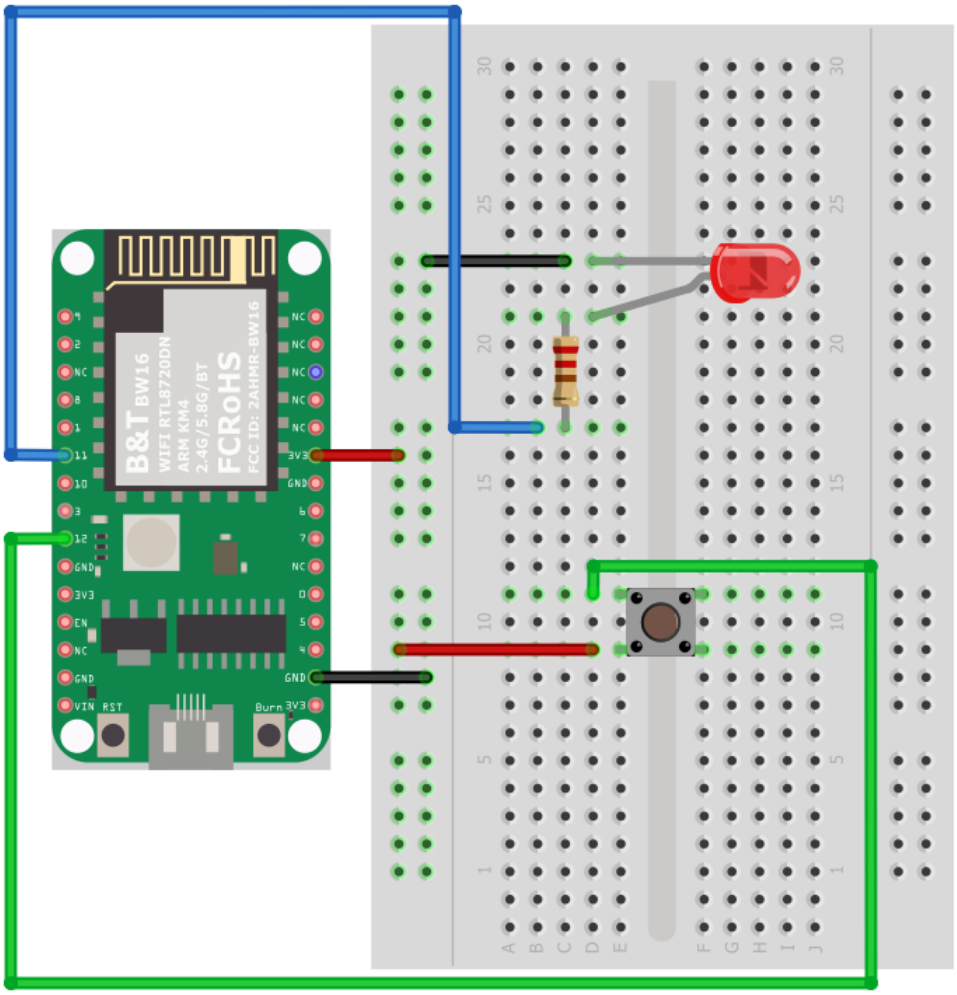
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Compile and upload the program, press reset.
The LED lights at first. Press and release the button, then the LED should dim. Press again, then the LED should light.

Code Reference

In

```
setup()
```

we set Pin 12 to

```
INPUT_IRQ_RISE
```

, this means that an interrupt occurs when the voltage of this pin changes from GND to 3V3. Therefore, we connect the other side of the button to 3V3, so as to trigger interrupt event when the button is pressed.

```
pinMode(button, INPUT_IRQ_RISE);
```

On the other hand, we can set pin 12 to

```
INPUT_IRQ_FALL
```

, this means that an interrupt occurs when the voltage of this pin changes from 3V3 to GND. In this case, the other side of the button is connected to GND.Next, we need to specify the funtion to be execute to handle the interrupt:

```
digitalSetIrqHandler(button, button_handler);
```

The second parameter is a function pointer, with prototype:

```
void button_handler(uint32_t id, uint32_t event)
```

In this handler, every time we press and release the button, we trigger an interrupt, and change the status of the LED.

GTimer - Using The Periodic GTimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Ameba provides 4 hardware GTimer for users to use. The timers' resolutions are at microseconds scale.
The timer can be set to be periodic or for single use. The periodic timers reset periodically, and the single-use timers do not.

Open the example, "`File`" -> "`Examples`" -> "`AmebaGTimer`" -> "`TimerPeriodical`". Compile and upload to Ameba, and press reset.
In the Serial Monitor, you can see the counter value is increased periodically.

Code Reference

The first argument of begin() is the timer id (0~3).
The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s.
The third argument specifies the function to call when the time is up. In the example, we call the "myhandler" function to increase the counter value by 1 and print the counter value to serial monitor.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler);
```

The GTimer is periodic by default, therefore "myhandler" function is called every second. When we want to stop the GTimer, use `stop()`:

```
GTimer.stop(0);
```

GTimer - Using the One-Time Gtimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we will use 4 One-Time GTimer, and pass user data to each timer.

Open the example "File" -> "Examples" -> "AmebaGTimer" -> "TimerOneshot". Compile and upload to Ameba, and press reset. Then you can see the 4 timer log printed to the serial monitor in series.

Code Reference

The first argument of begin() is the Timer ID (0~3). The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s. The third argument specifies the function to call when the time is up. The fourth argument is to set whether this timer is a periodic timer, we use "false" here to begin a single-use timer. The fifth argument is the user data, we give 0 here to represent that this is timer 0.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
```

Next, we set up the second timer, which has timer value 2 seconds, and user data 1. And other timers are set similarly.

```
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
```

In myhandler function, we print the user data to serial monitor. Since the 4 timers are separately set to count for 1, 2, 3, 4 seconds, from 1 second to 4 second, the user data of each timer are printed on the serial monitor in order. After 4 second, no log will be printed.

I2C - Send Data to Arduino UNO

Introduction of I2C

There are two roles in the operation of I2C, one is "master", the other is "slave". Only one master is allowed and can be connected to many slaves. Each slave has its unique address, which is used in the communication between master and the slave. I2C uses two pins, one is for data transmission (SDA), the other is for the clock (SCL). Master uses the SCL to inform slave of the upcoming data transmission, and the data is transmitted through SDA. The I2C example was named "Wire" in the Arduino example.

Materials

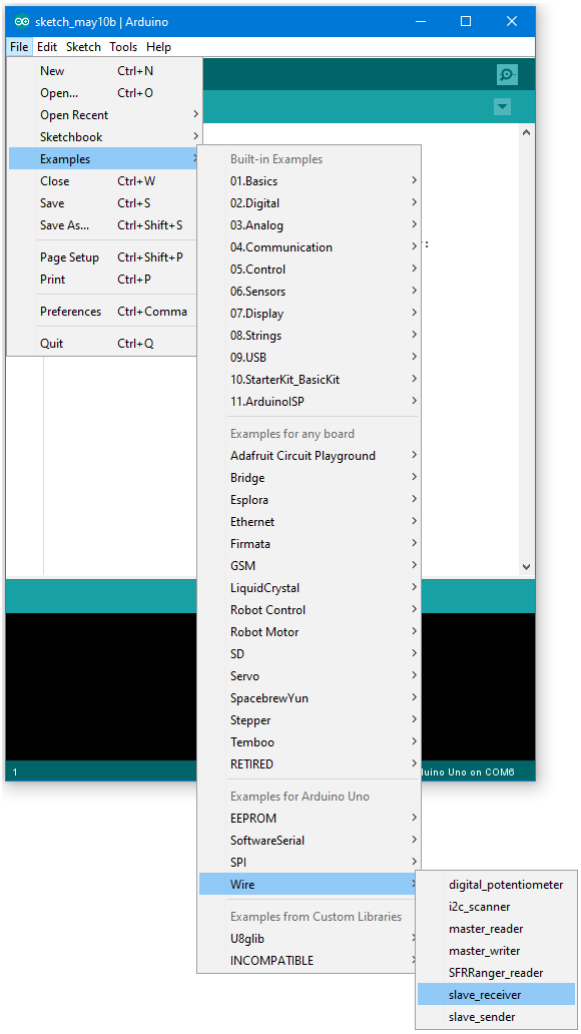
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In this example, we use Ameba as the I2C master writer, and use Arduino as the I2C slave receiver.
When the I2C slave receives string sent from I2C master, it prints the received string.

- Setting up Arduino Uno to be I2C Slave

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`"
Open the "Slave Receiver" example in "`Examples`" -> "`Wire`" -> "`slave_receiver`":

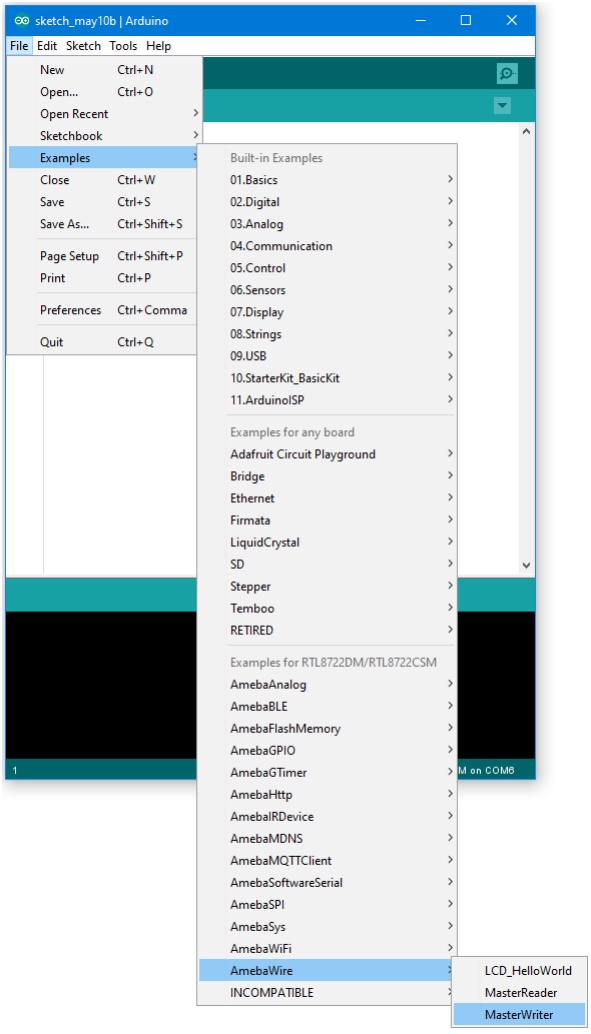


Then click "`Sketch`" -> "`Upload`" to compile and upload the example to Arduino Uno.

- Setting up Ameba to be I2C Master

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE:
"`Tools`" -> "`Board`"

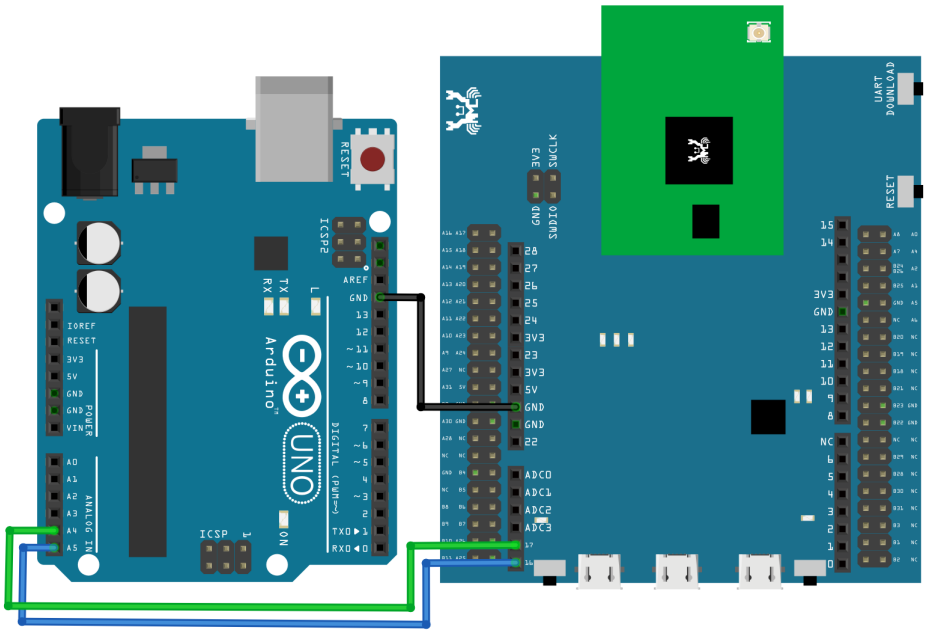
Then open the “Master Writer” example in “`File`” -> “`Examples`” -> “`AmebaWire`” -> “`MasterWriter`”



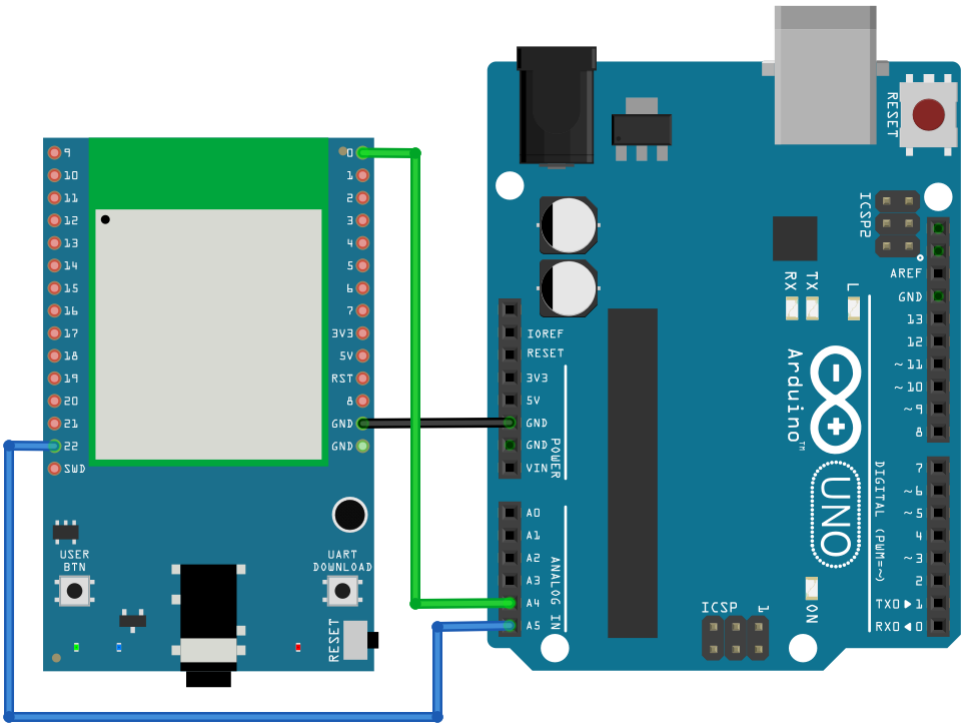
- Wiring

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

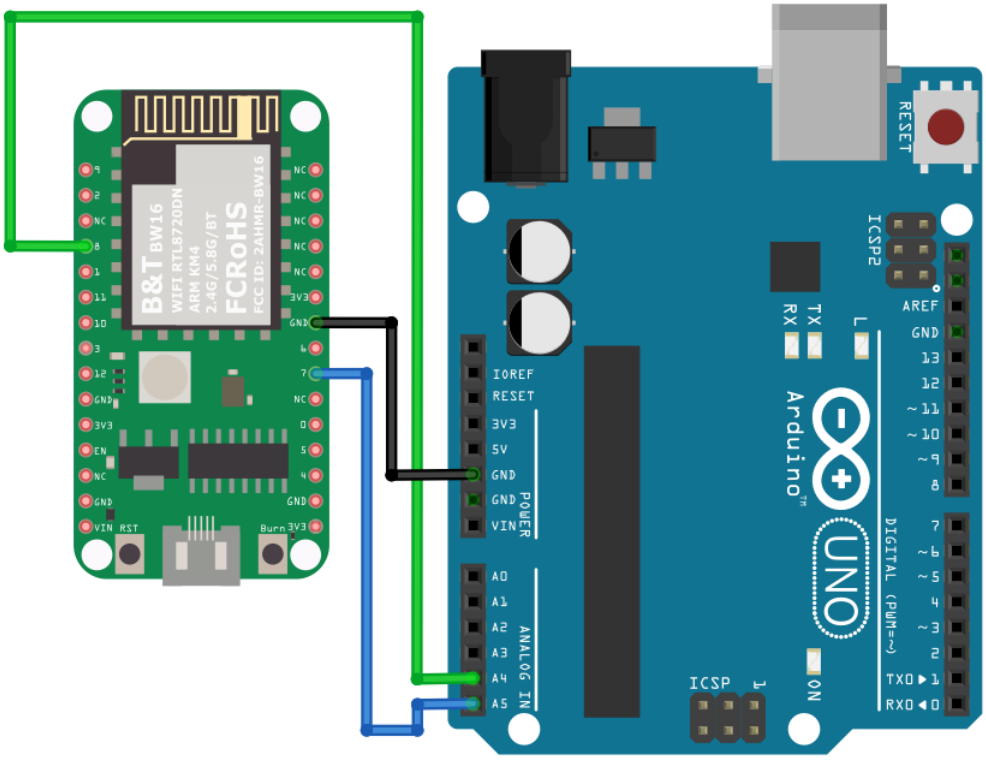
AMB21/ AMB22 Wiring Diagram:



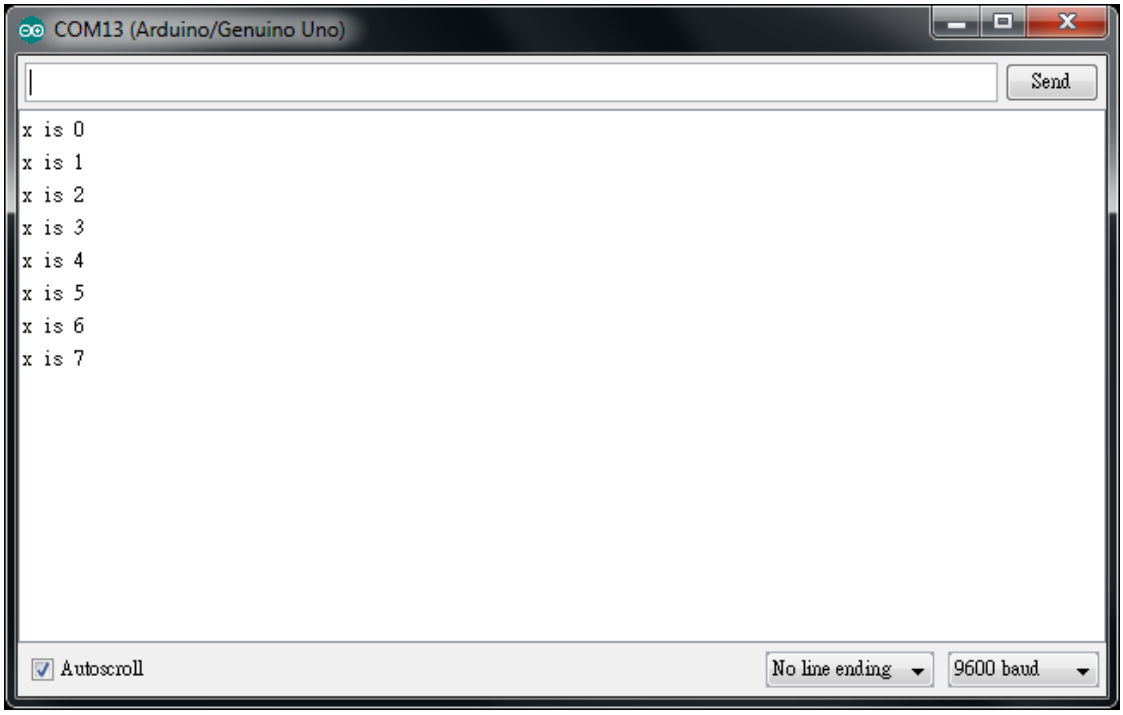
AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the Arduino IDE of the Arduino Uno and open the serial monitor ("Tools" -> "Serial Monitor").
In the Serial Monitor, you can see the messages printed from Arduino Uno.
Next, press the reset button on Arduino Uno. Now the Arduino Uno is waiting for the connection from I2C Master.
We press the reset button on Ameba to start to send messages. Then observe the serial monitor, you can see the messages show up every half second.



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterWriter

First use Wire.begin()/Wire.begin(address) to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses Wire.beginTransmission(address) to begin a transmission to the I2C slave with the given address:
https://www.arduino.cc/en/Reference/WireBeginTransmission

Uses Wire.write() to send data, and finally use Wire.endTransmission() to end a transmission to a Slave and transmits the bytes that were queued:
https://www.arduino.cc/en/Reference/WireEndTransmission

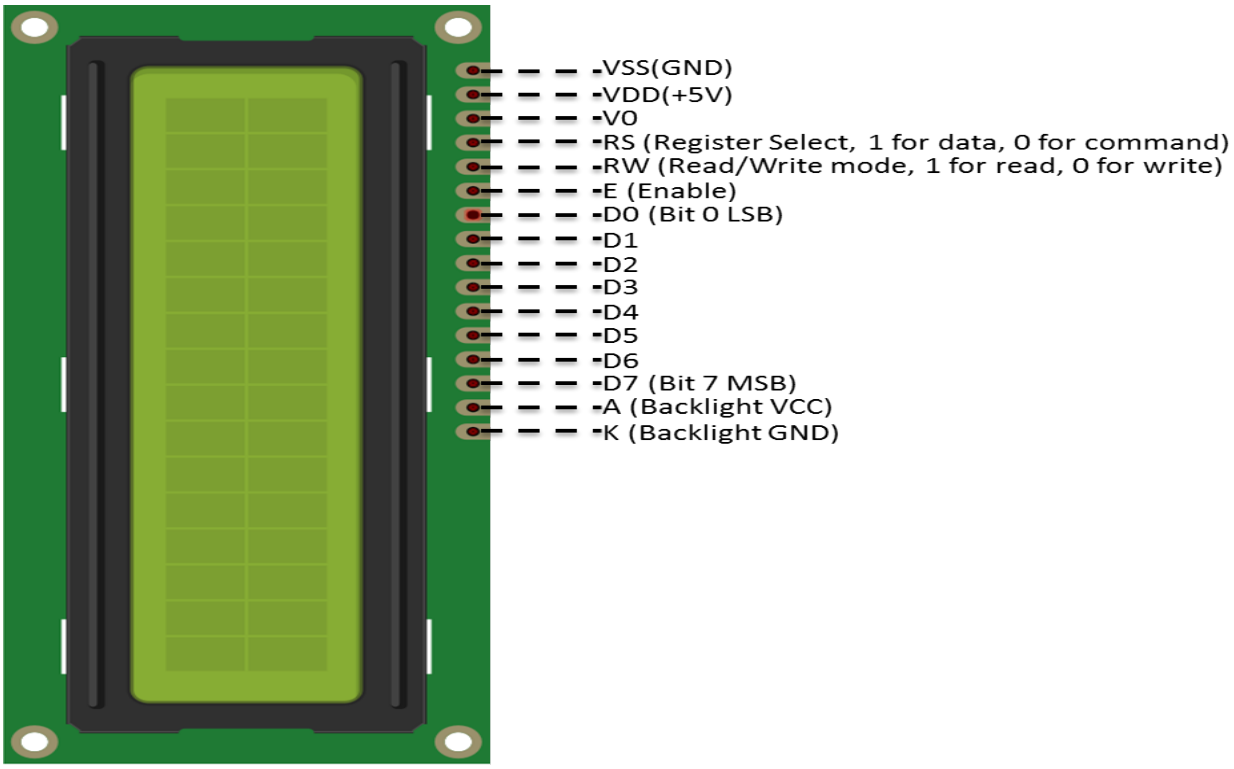
I2C - Display Data On LCD Screen

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- I2C 2×16 LCD

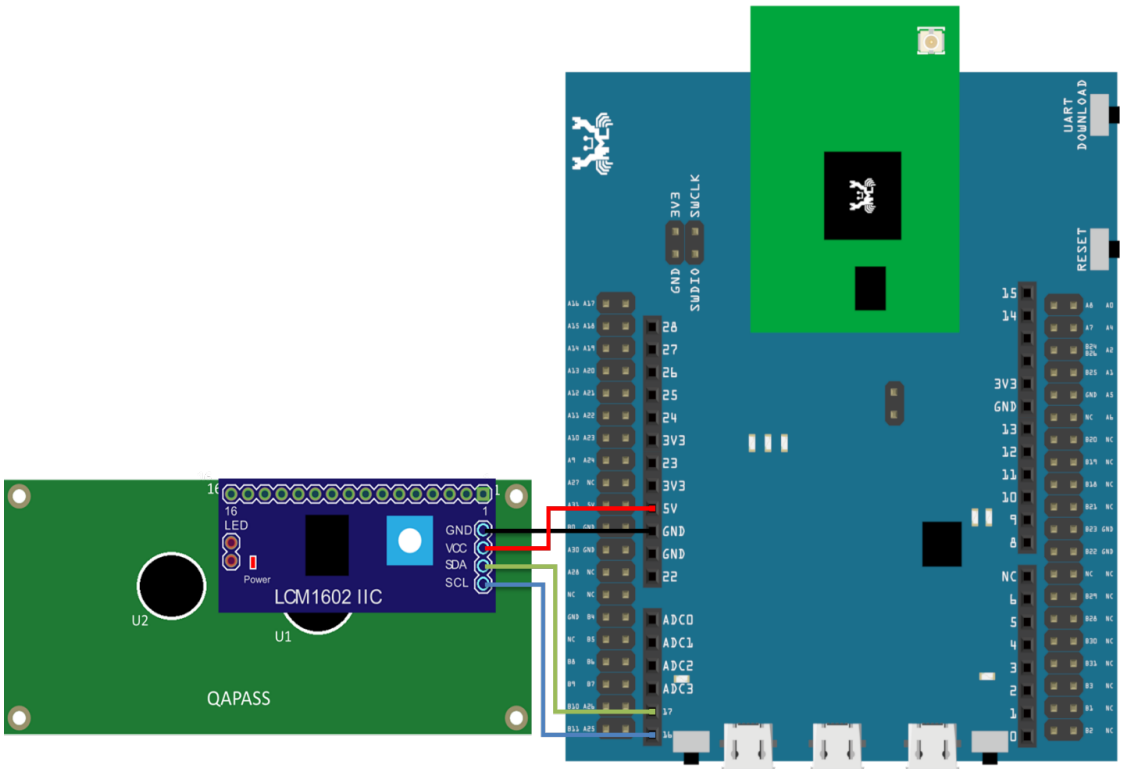
Example

Normally there are many pins on an LCD display, as shown in below figure.

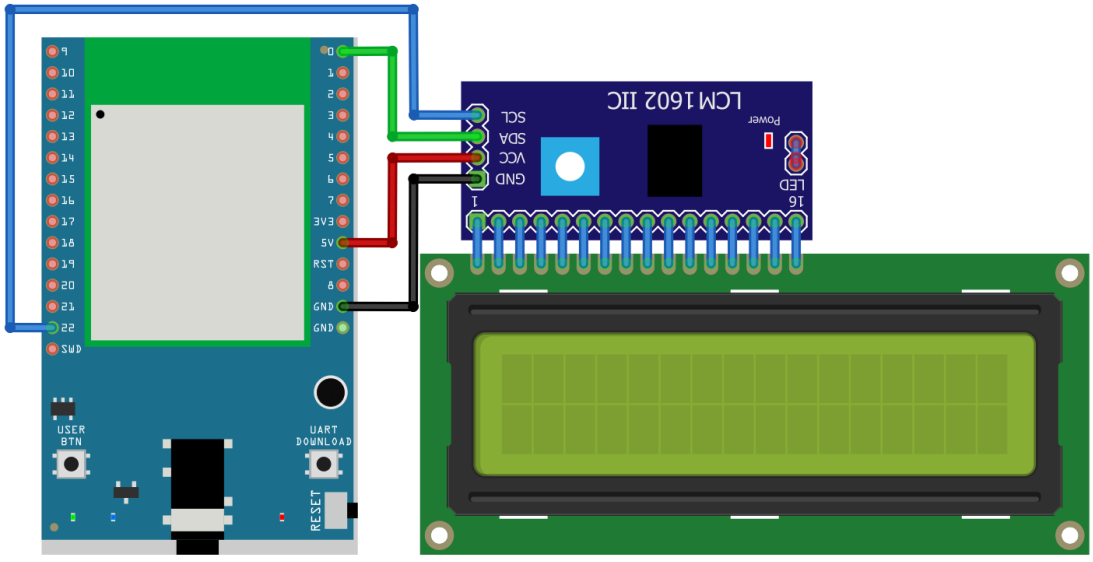


An LCD display can be equipped with an additional processing chip to process the data. The processing chip can connect to a microcontroller using the I2C interface.

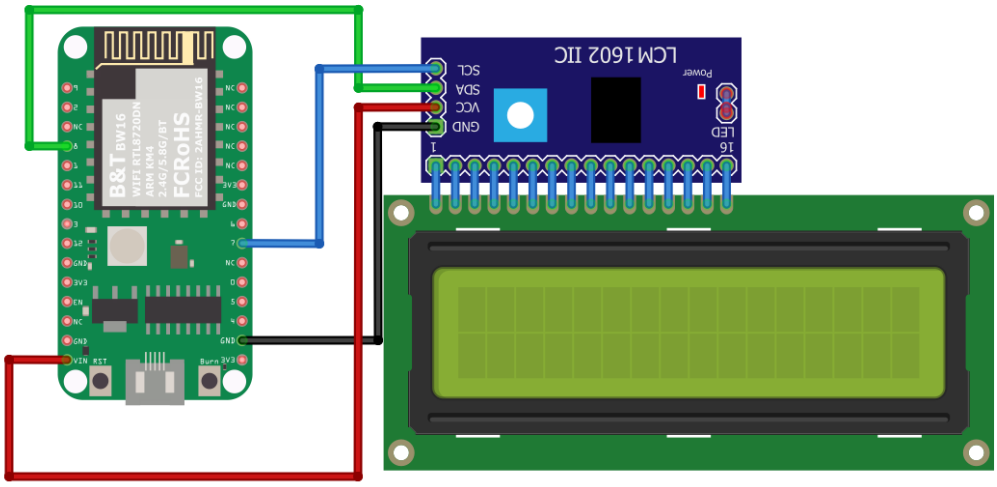
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

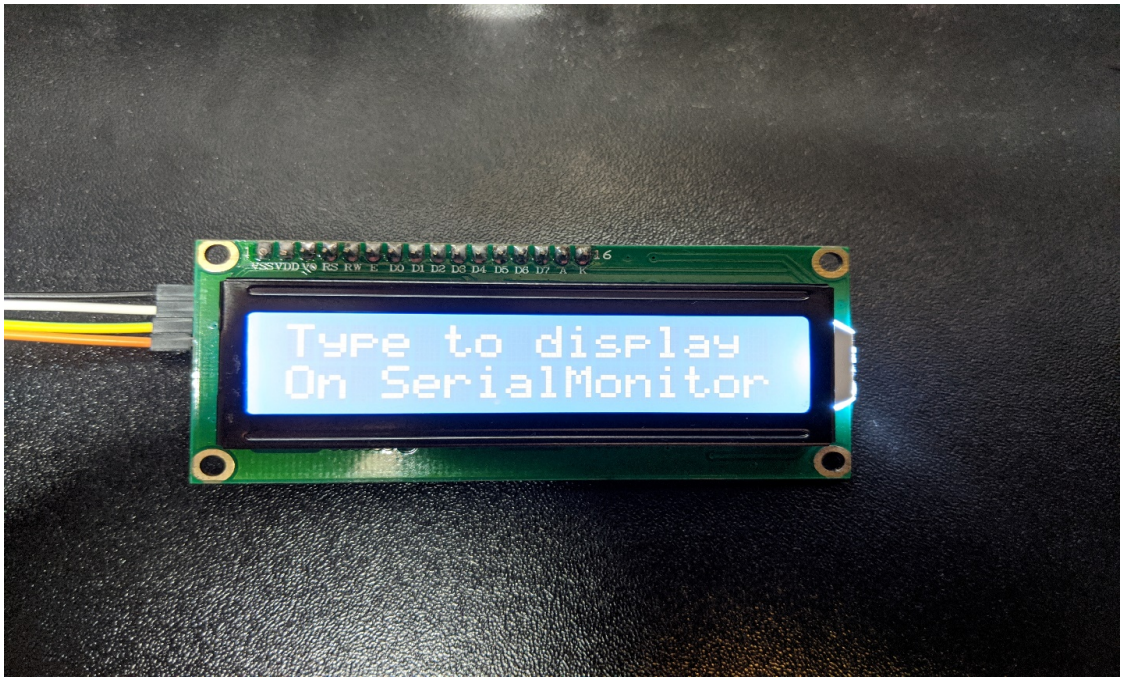


BW16 Wiring Diagram:



Open the example in "`File` -> "`Examples`" -> "`AmebaWire`" -> "`LCD_HelloWorld`".
Compile and upload to Ameba, then press the reset button.
Then you can see "Hello World" in the first line, and "Ameba" in the second line displayed on the LCD screen.

After 8 seconds, you can input to the Serial Monitor the string you would like to display on the LCD.



For example, we enter “123456789” and press “Send”:

Code Reference

The required settings of each model of LCD might be different, the constructor we use in this example is:

```
LiquidCrystal_I2C(uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs,
                  uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7,
                  uint8_t backlighPin, t_backlighPol pol);
```

And the setting parameters are as follows:

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C␣
↪address
```

The first parameter 0x27 is the address of I2C. Each of the following 8 parameters represents the meaning of each bit in a byte, i.e., En is bit 2, Rw is bit 1, Rs is bit 0, d4 is bit 4, and so forth.

Call `backlight()` to light the screen,
Call `setCursor(0, 0)` to set the position of the cursor.
LCD inherits the Print class, so we can use `lcd.print()` to output string on the screen.

I2C - Receive Data from Arduino UNO

Materials

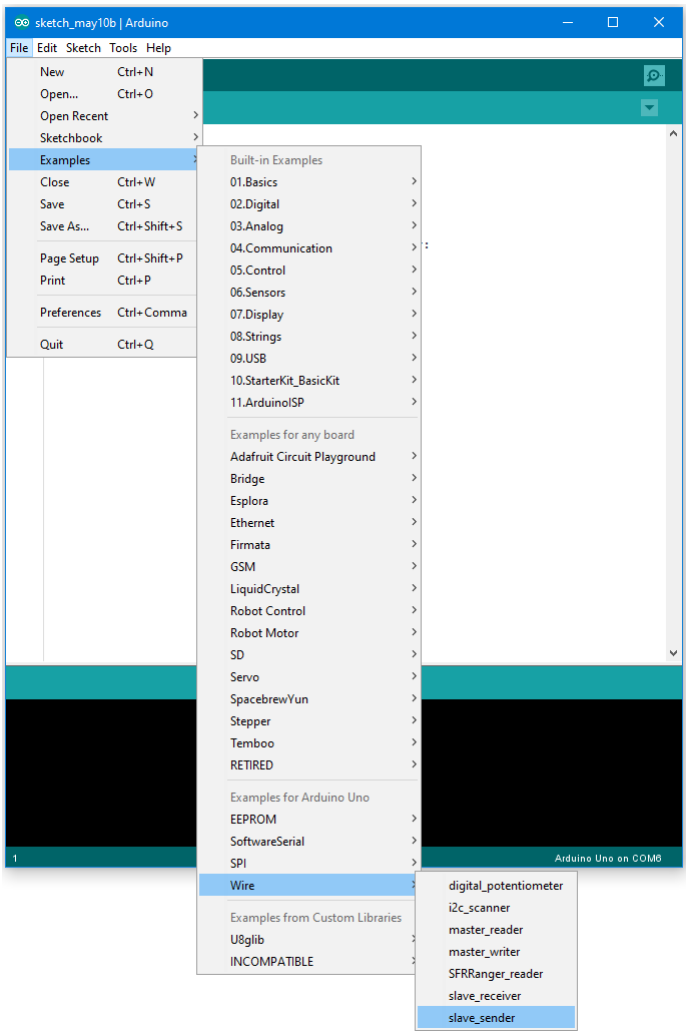
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In the previous example "I2C – Communicate with Arduino UNO via I2C" , Ameba, the I2C master, transmits data to the Arduino UNO, the I2C slave.
As to this example, Ameba is the I2C master, and receives data from the Arduino UNO, which is the I2C slave.

- **Setting up Arduino Uno to be I2C Slave**

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`":
Open "`Examples`" -> "`Wire`" -> "`slave_sender`"

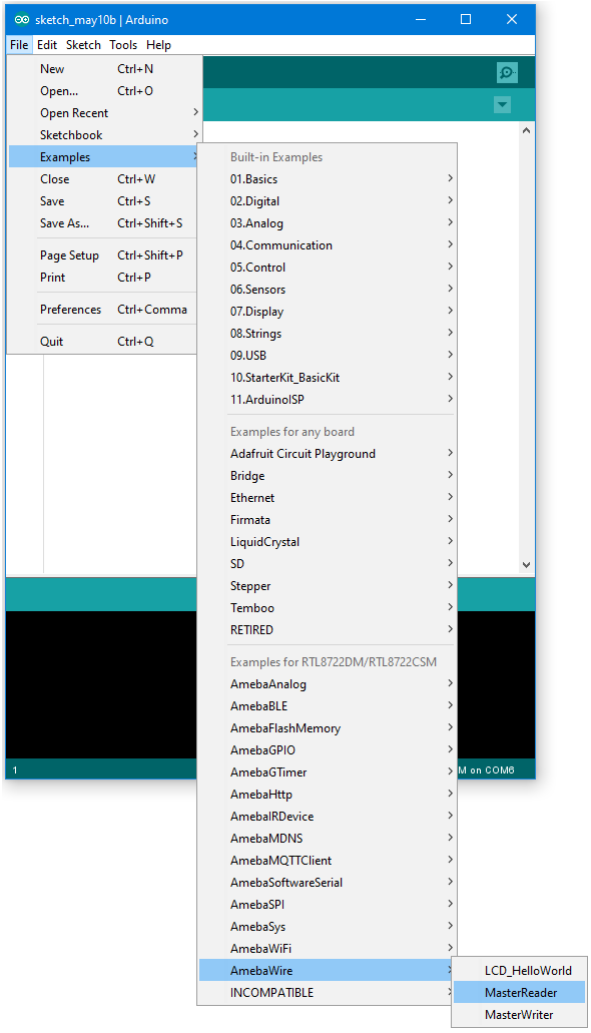


Then click “Sketch” -> “Upload” to compile and upload the example to Arduino Uno.

- **Setting up Ameba to be I2C Master**

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE: “Tools” -> “Board”

Open “`File`” -> “`Examples`” -> “`AmebaWire`” -> “`MasterReader`”

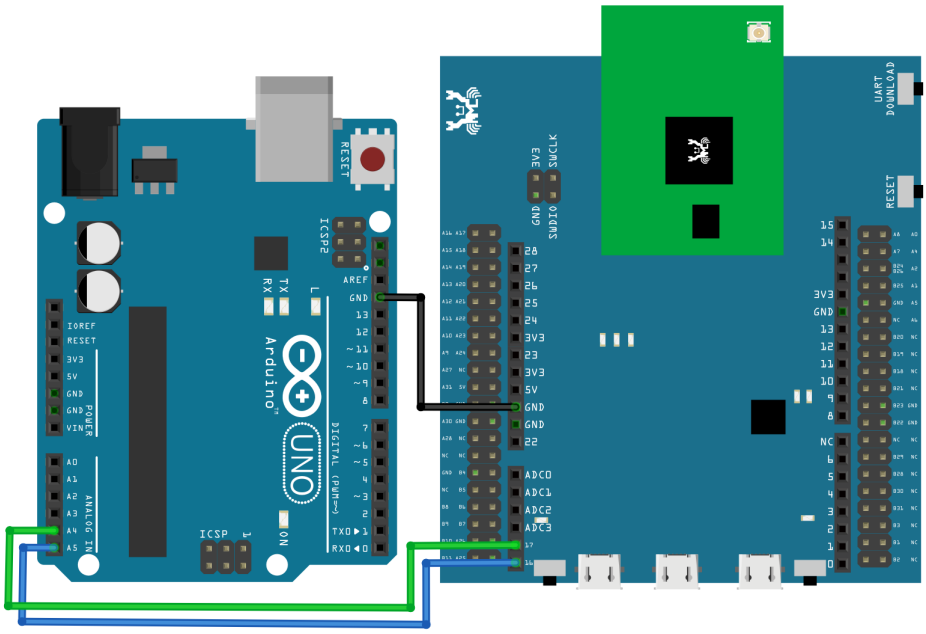


Click “Sketch” -> “Upload” to compile and upload the example to Ameba.

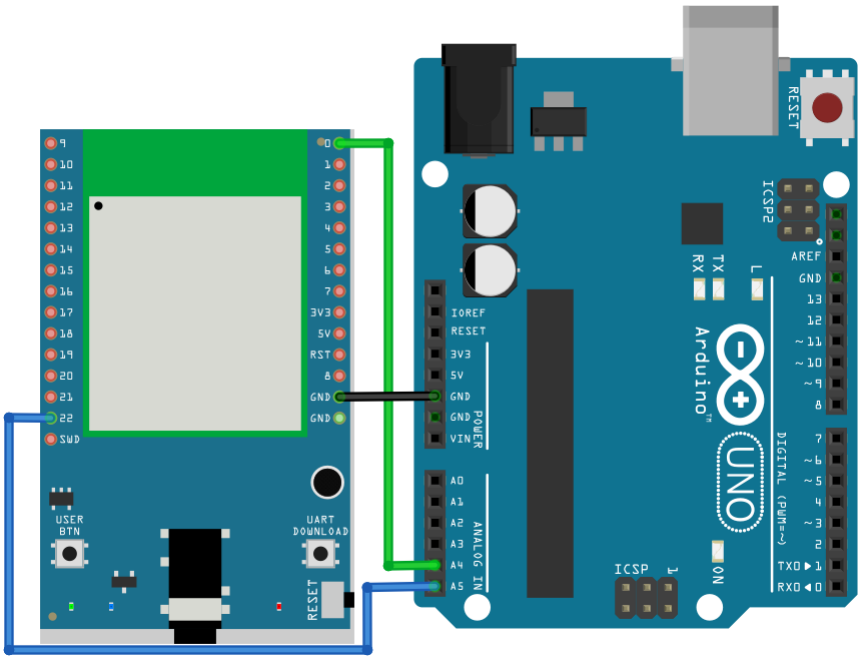
- **Wiring**

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

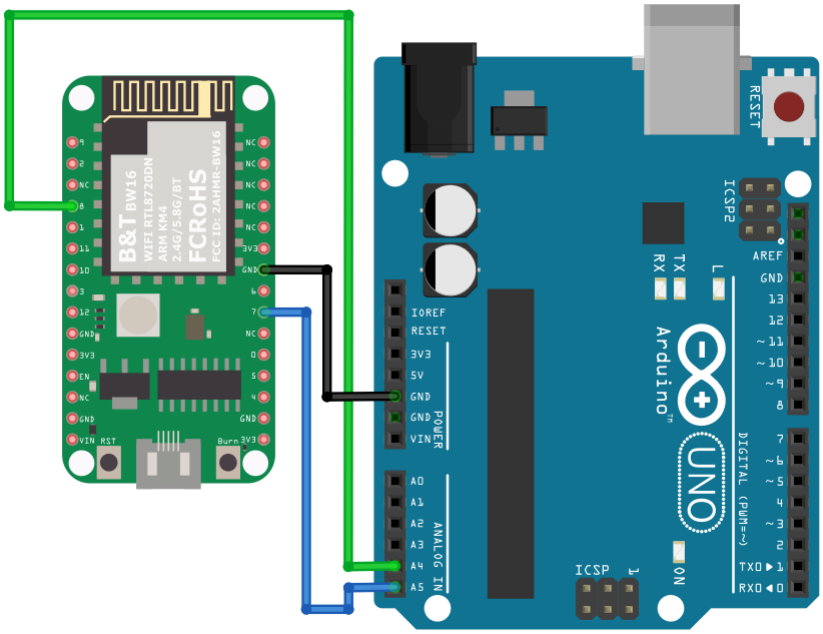
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

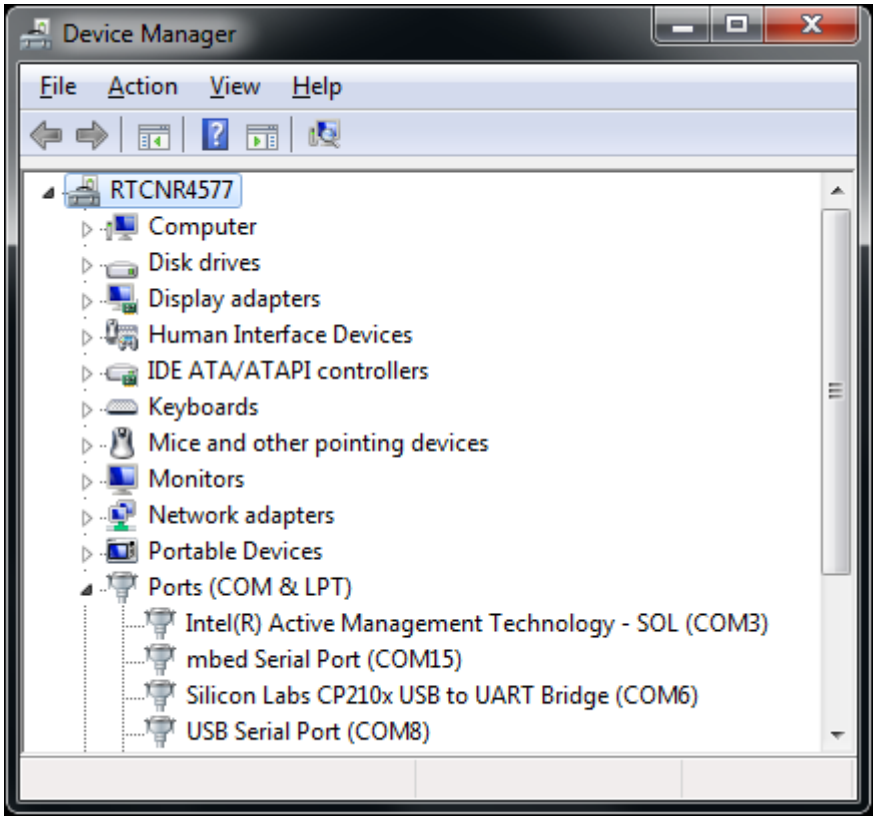


BW16 Wiring Diagram:



Next, we will observe the data receive by Ameba in the Serial Monitor.

(Note: If you do not know which port the Ameba development board is connected to, please find it in the Device Manager of Windows first. Ameba is connected as “mbed Serial Port”. For example, if you find mbed Serial Port (COM15) means Ameba is connected to port COM15.)



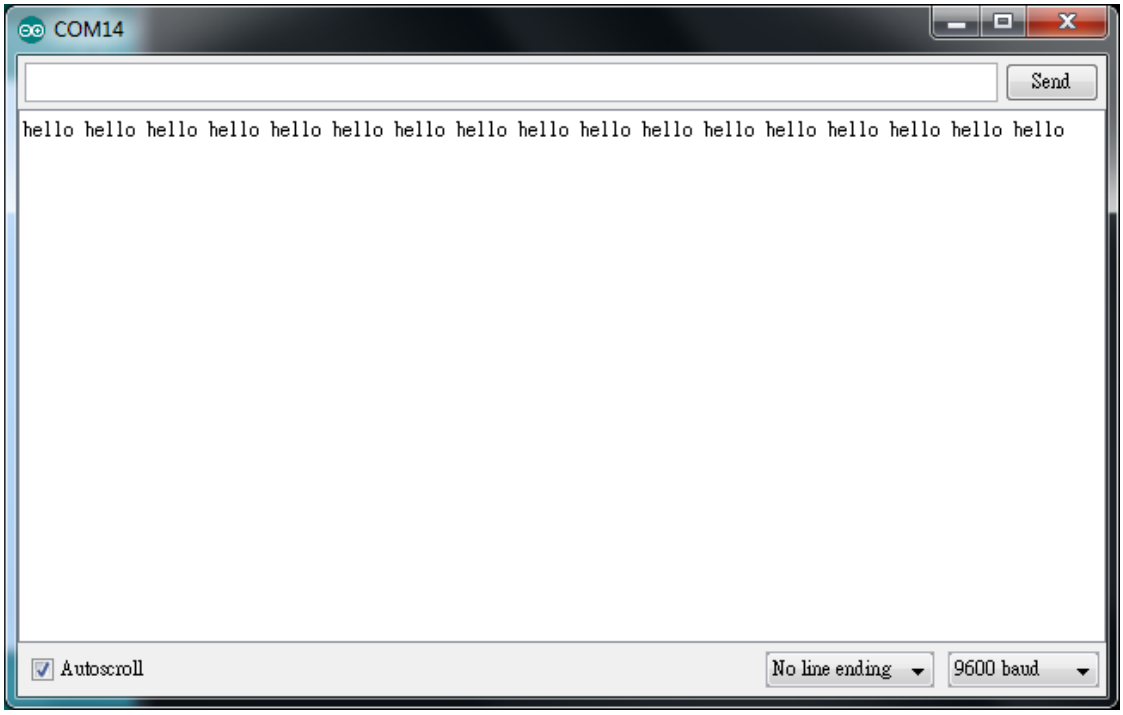
We select the port in “Tools” -> “Port” -> “COM15” (the port connected to Ameba)

Open the Arduino IDE window of the Ameba, go to “Tools” -> “Serial Monitor” to display the messages printed by Ameba.

Press the reset button on Arduino Uno, Arduino Uno now waits for connection from I2C master.

Then press the reset button on Ameba, Ameba will start to receive messages from Arduino Uno. And you can see the “hello ” message printed every half second in serial monitor.

(NOTE: If the message does not show in the Serial Monitor of Ameba, please close and open the serial monitor again.)



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterReader

First use `Wire.begin()` / `Wire.begin(address)` to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses `Wire.requestFrom()` to specify from which Slave to request data.
https://www.arduino.cc/en/Reference/WireRequestFrom

IR - Transmit IR NEC Raw Data And Decode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2
- Grove – Infrared Emitter x1 (Figure 1)
- Grove – Infrared Receiver x1 (Figure 2)

Example

In this example, we use two Ameba RTL8722 modules that connecting with an infrared (IR) Emitter and an IR Receiver separately to transmit and receive IR NEC Raw data.

Figure 1: Grove – Infrared Receiver

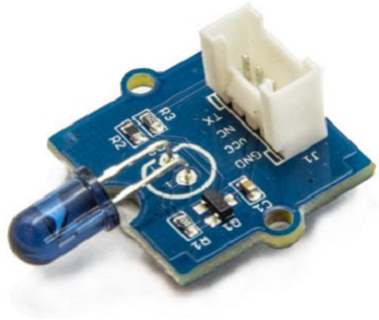


Figure 2: Grove – Infrared Emitter

On the transmission side, the transmitter will send IR NEC raw data. The raw data can be seen as consecutive durations of "marks" and "spaces" (Figure 3) in microseconds (us).

- Mark: a specific period of sending pulses
- Space: a specific period of sending nothing

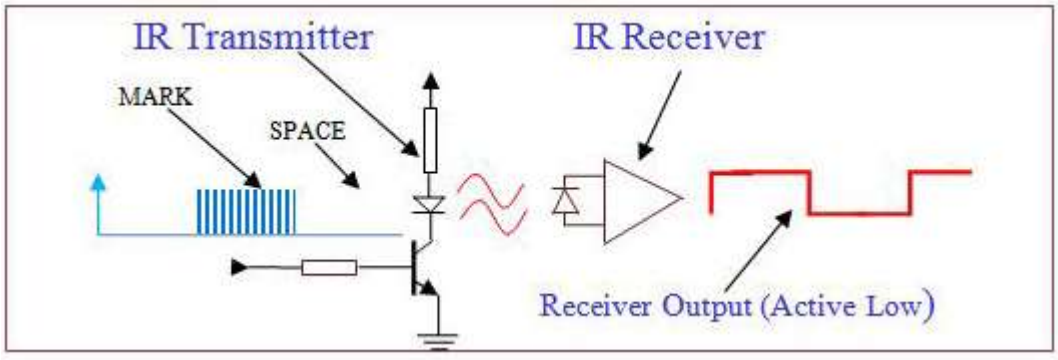


Figure 3: A typical IR transmission and reception setup implementation

For more details, please refer to SB-Projects' topic of IR Remote Control Theory to learn the theory of IR remote controls operation and a collection of IR protocol descriptions. In this example, we are going to use NEC (Now Renesas, also known as Japanese Format) as the transmission protocol.

NEC Features

- 8-bit address and 8-bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- The carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

Modulation

NEC protocol uses Pulse Distance Encoding of the bits for data communication (Figure 4). A logical “1” is represented by total duration of 2250us, with 560us of “marks” and (2250-560) us of “spaces”. While logical ”0” is represented by total duration of 1120us, with 560us “marks” and (1120-560) us of “spaces”.

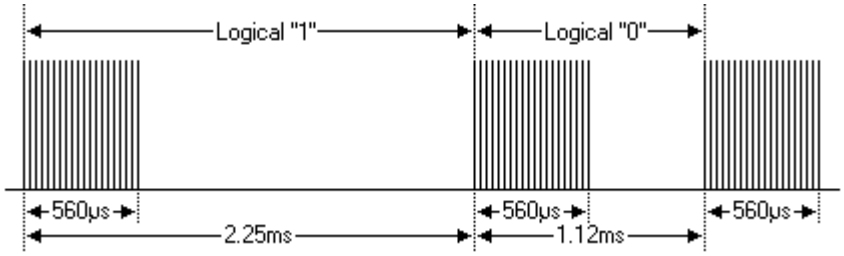


Figure 4: Modulation of NEC

Since a total number of 32-bit data together with the header and the end-bit will be transferred (Figure 5). If we separate the data in the time-frame (in us), there will be (2 + 32) x 2 + 1 = 69 “marks” / “spaces” to be transmitted (Figure 6), which forms the raw NEC data we would like to transmit in our Arduino “*.ino” file. This part of the code can be modified by users. Details of how to obtain raw data code for your remote devices, you may refer to Ken Shirriff’s blog, where it provides multiple libraries provided online.

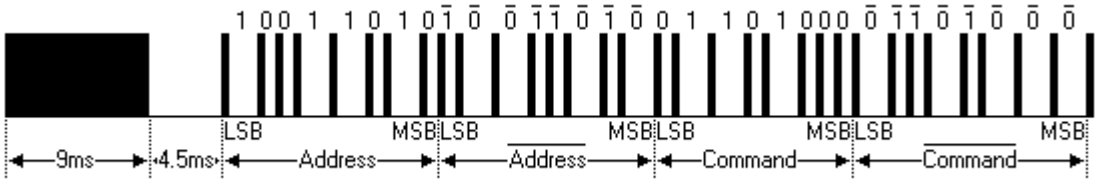


Figure 5: Sample of a Full NEC Data (in logic1 or 0)

```
9000, 4500,                                                                 // starting bit
560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560,       // address
560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~address
560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560,       // data
560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~data
560                                                                         // stoping bit
```

Figure 6: Sample of a Full NEC RAW Data (in us)

Figure 7 and 8 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722 board.

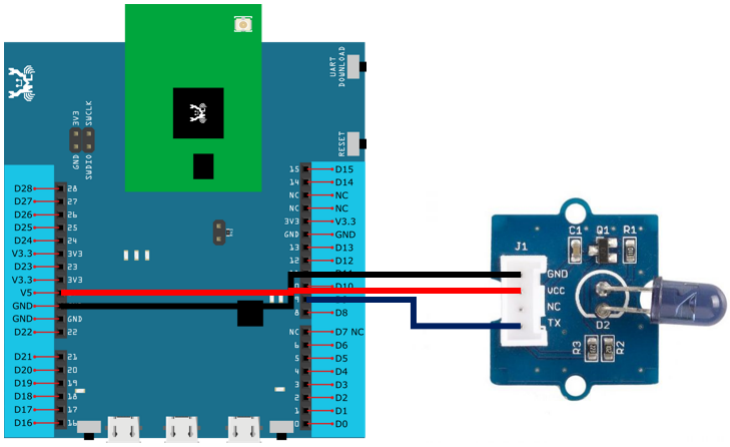


Figure 7: Pin configuration of IR Emitter and AMB21/AMB22

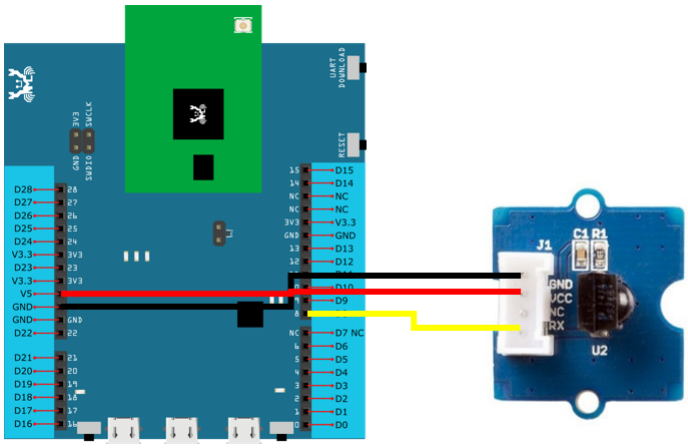


Figure 8: Pin configuration of the IR Receiver and Ameba RTL8722

Figure 9 and Figure 10 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722DM MINI.

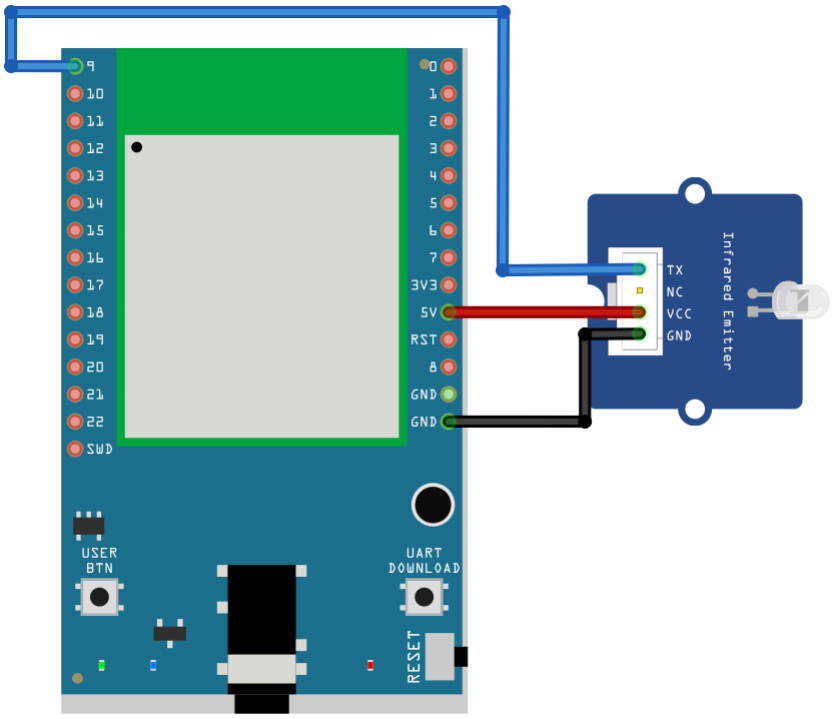


Figure 9: Pin configuration of IR Emitter and AMB23

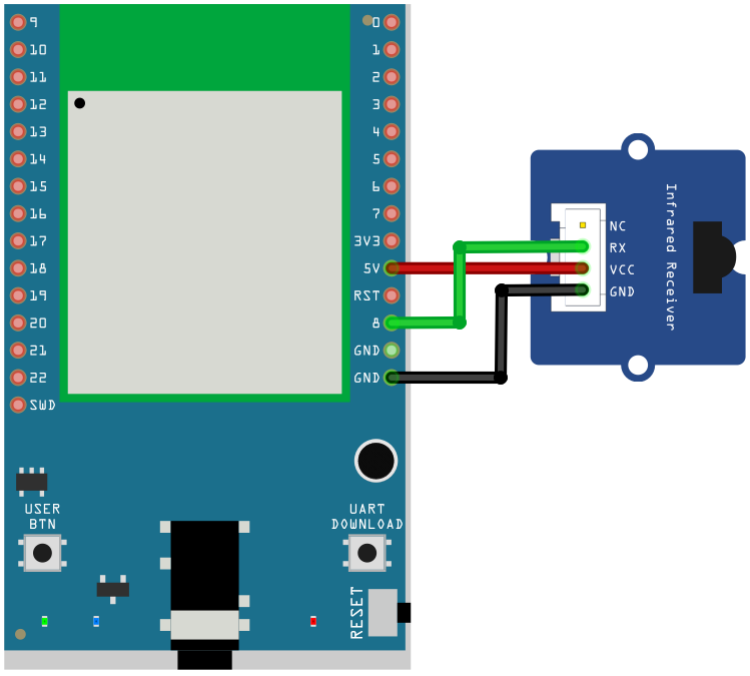


Figure 10: Pin configuration of the IR Receiver and AMB23

Figure 11 and Figure 12 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8720DN (BW16).

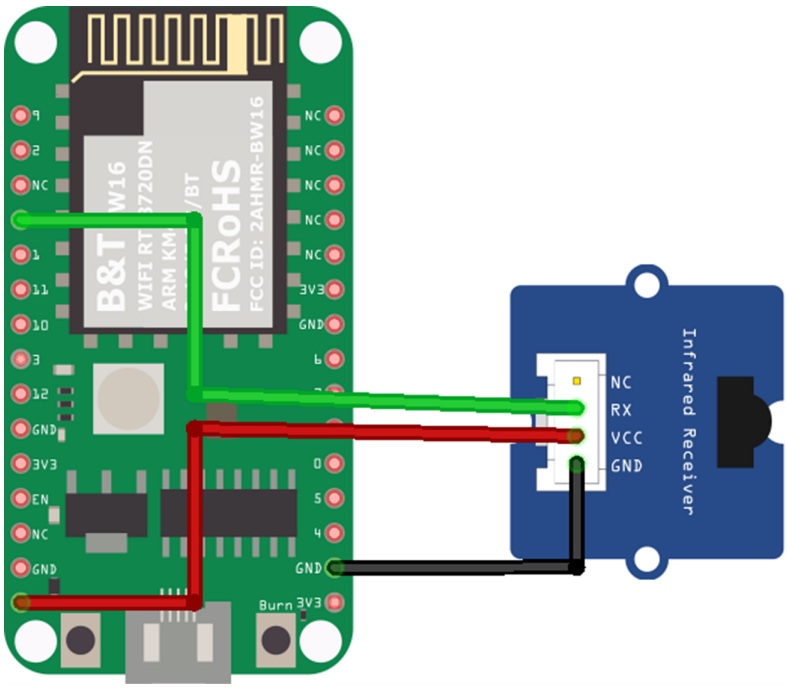


Figure 11: Pin configuration of IR Emitter and BW16

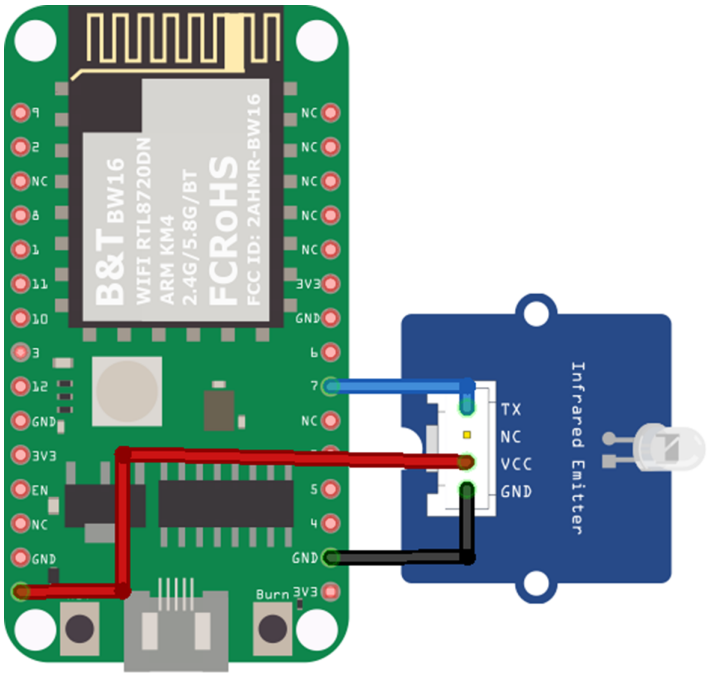


Figure 12: Pin configuration of IR Receiver and BW16

After the connection is being set up correctly, we will move to the coding part for this example. First, make sure the correct Ameba development board is selected in Arduino IDE: “Tools” -> “Board”.

Open the “IRSendRAW” example in “`File`” -> “`Examples`” -> “`AmebaIRDevice`” -> “`IRSendRAW`” (Figure 11) and upload to 1st board connected with IR Emitter:

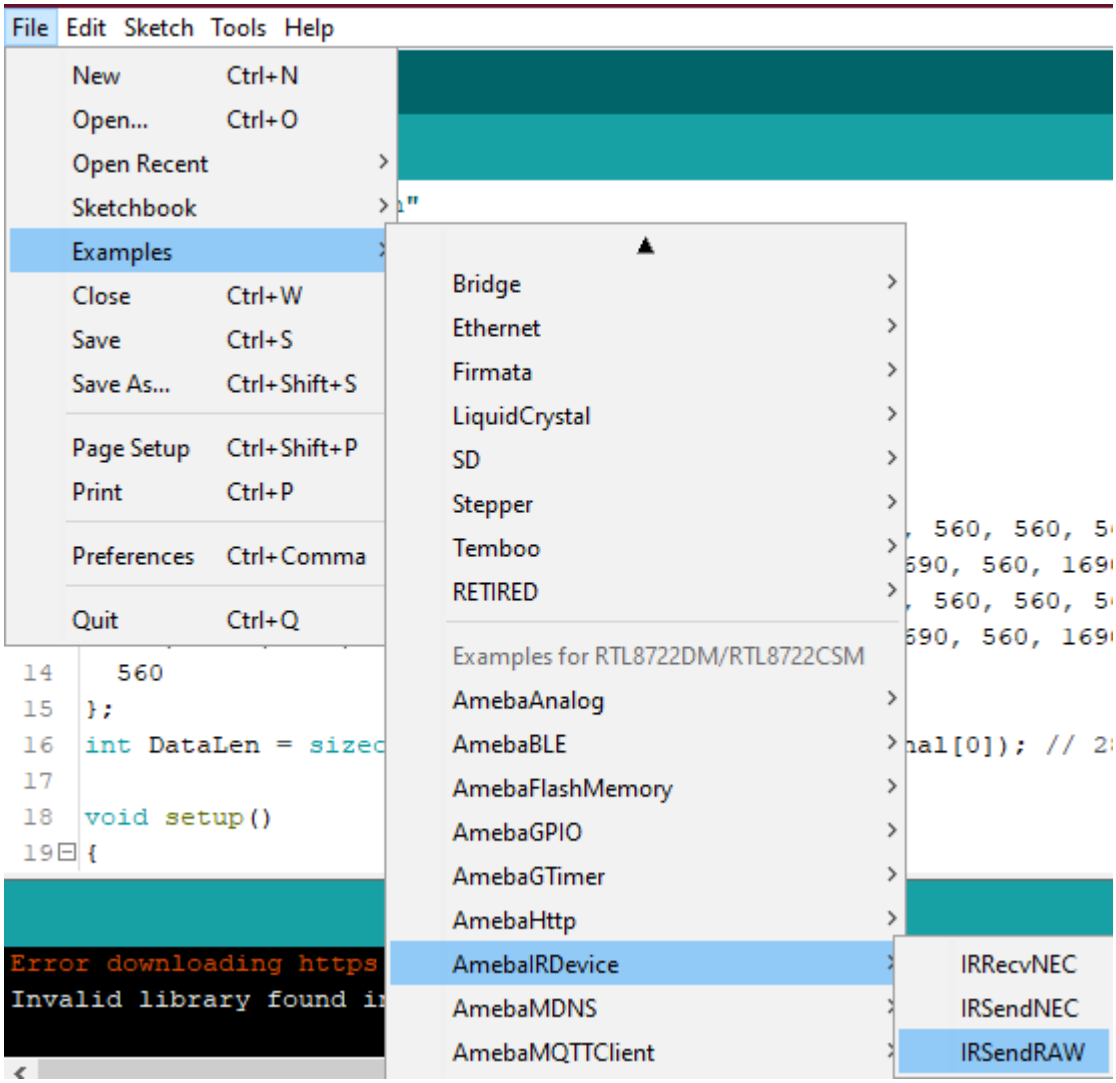


Figure 13: Example Location of IRSendRaw and IRRecvNEC

After successfully upload the sample code for IRSendRaw, you might need to upload the IRRecvNEC example for the 2nd board connected with IR Receiver from "`File`" -> "`Examples`" -> "`AmebaIRDevice`" -> "`IRRecvNEC`".

After opening the serial monitor on the IR Receiver side and press the reset buttons on two boards, the data "48" will be received every 3 seconds (due to the delays () function, not compulsory to wait). After decoding the signal from the receiving Pin D8 and transmitting Pin D9 with Logic Analyser and Pulse View (Figure 10), the result is also shown as "48" after decoding the receiving data with IR NEC Protocol.

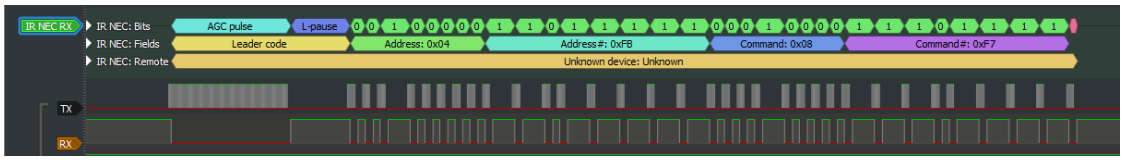


Figure 14: Pulse View results from sending and receiving pin

Code Reference

[1] Seeed Official website for Grove – Infrared Receiver
https://wiki.seeedstudio.com/Grove-Infrared_Receiver/

[2] Seed Official website for Grove – Infrared Emitter
https://wiki.seeedstudio.com/Grove-Infrared_Emitter/

[3] Ken SHirriff's blog on A Multi-Protocol Infrared Remote Library for the Arduino
http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

[4] SB-Projects: IR Remote Control Project
https://www.sbprojects.net/knowledge/ir/index.php

Power Save - Deep Sleep Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

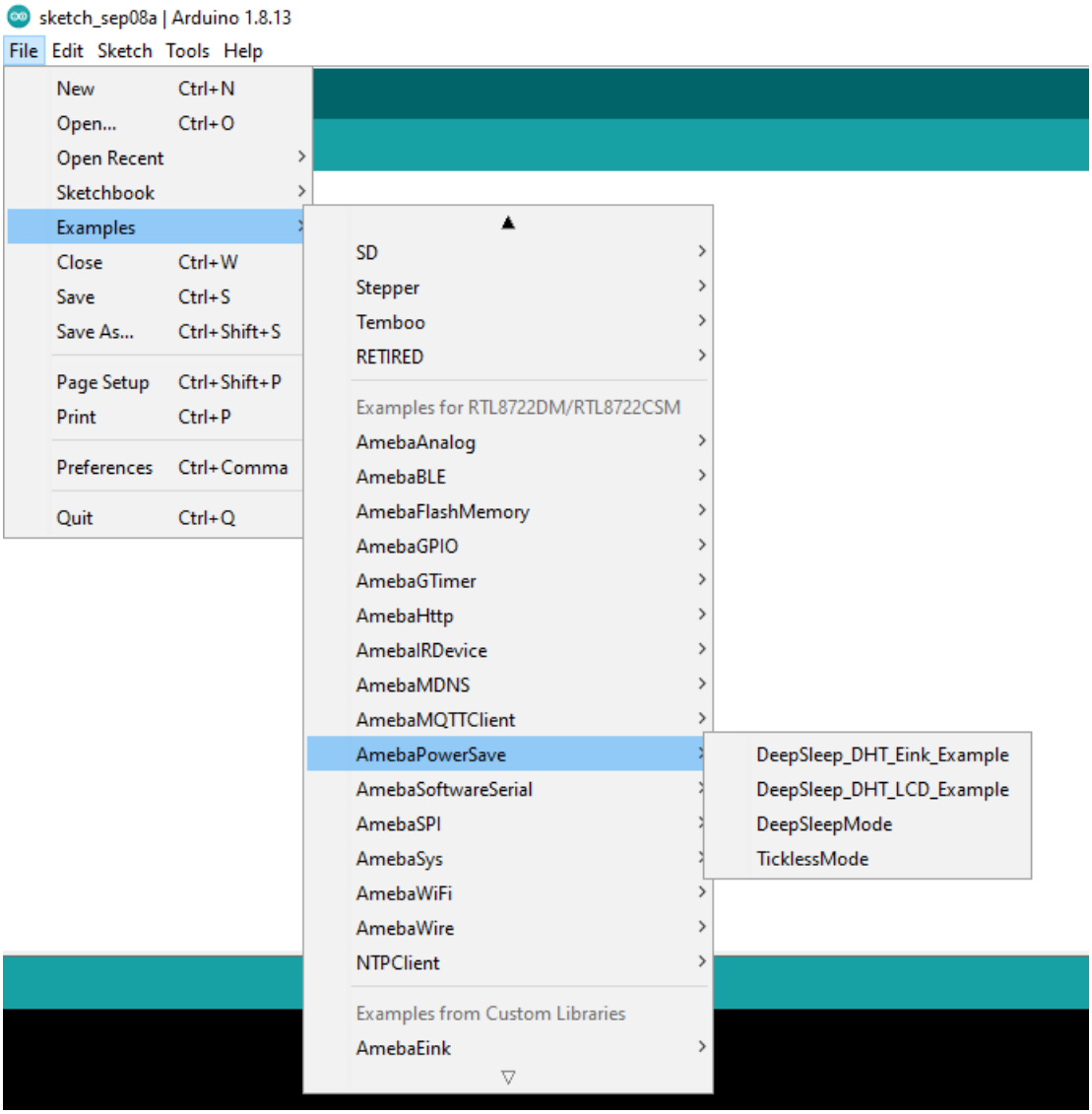
Example

Introduction

Ameba-D supports 2 low power modes which are deepsleep mode and sleep mode. Deep Sleep mode turns off more power domain than sleep mode. The power consumption of Deep Sleep mode is around 7μA to 8μA as compared to normal state which is around 22mA. This example describes how to enter Deep Sleep mode and configure the wakeup source

Procedure

Open "`File` -> "`Examples`" -> "`AmebaPowerSave`" -> "`DeepSleepMode`"



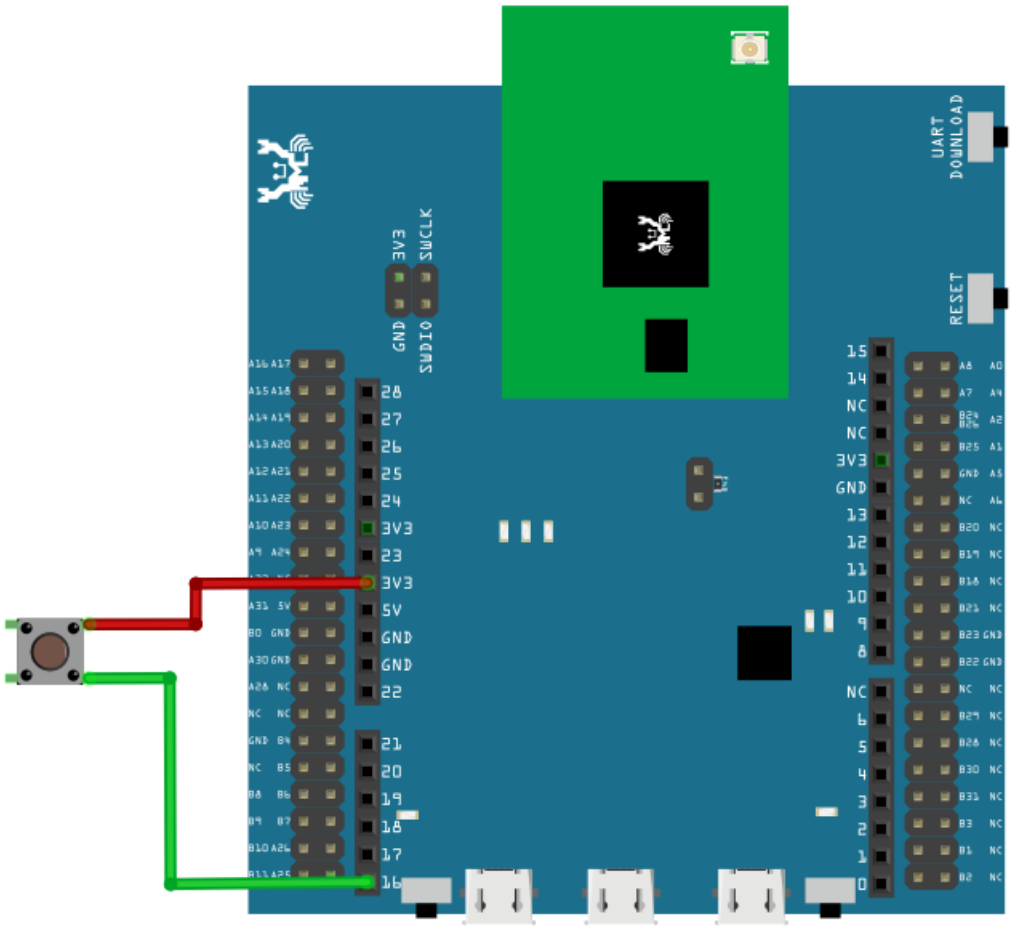
Set condition values as picture below.
"DS_WAKEUP_SOURCE" is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON Timer(SET_DS_AON_TIMER_WAKEUP);
AON GPIO pins(SET_AONWAKEPIN_WAKEUP);
RTC Timer(SET_DS_RTC_WAKEUP);
```

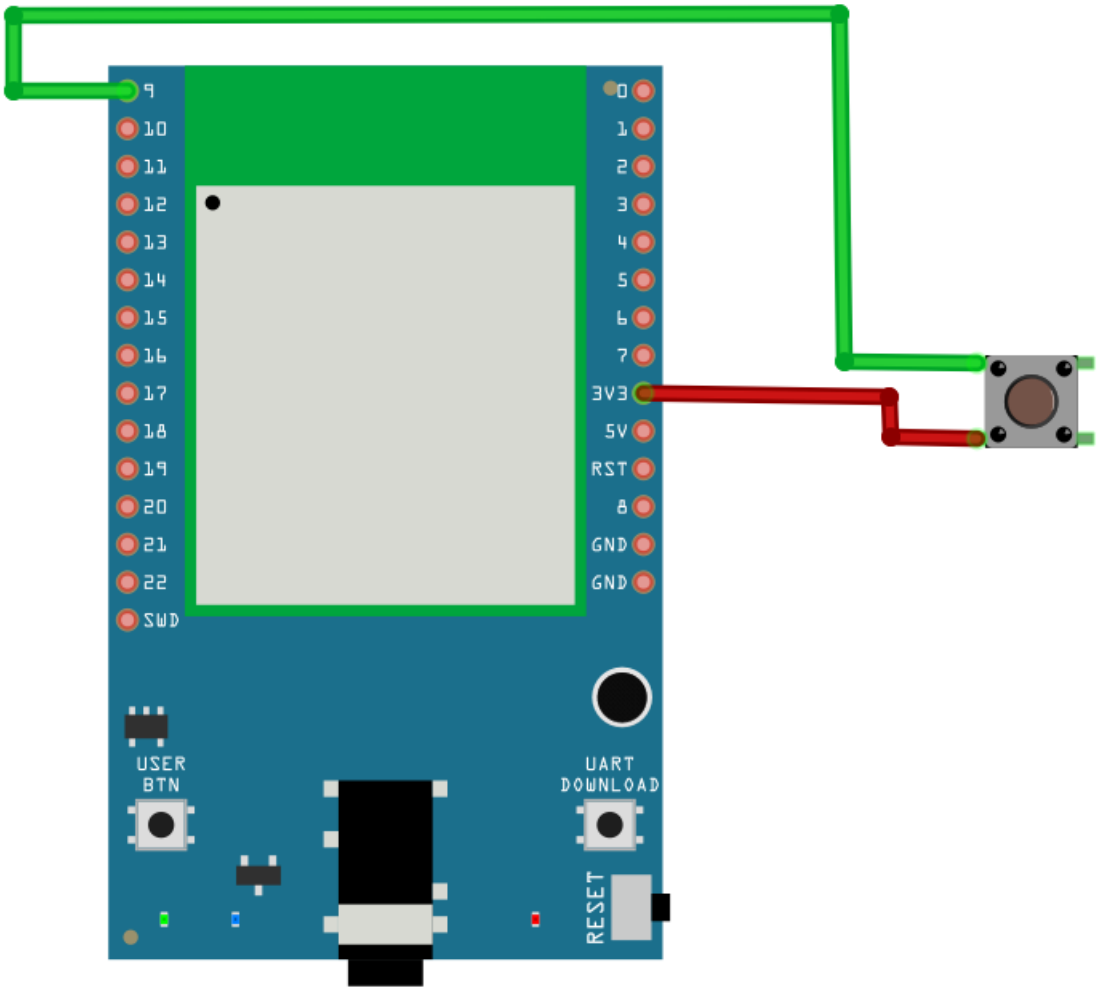
Using AON Timer as wakeup source
AON Timer can be set from 0 to 32760000ms range by `AON_TIMER_SLEEP_DURATION`

Using AON GPIO pins as wakeup source
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16),

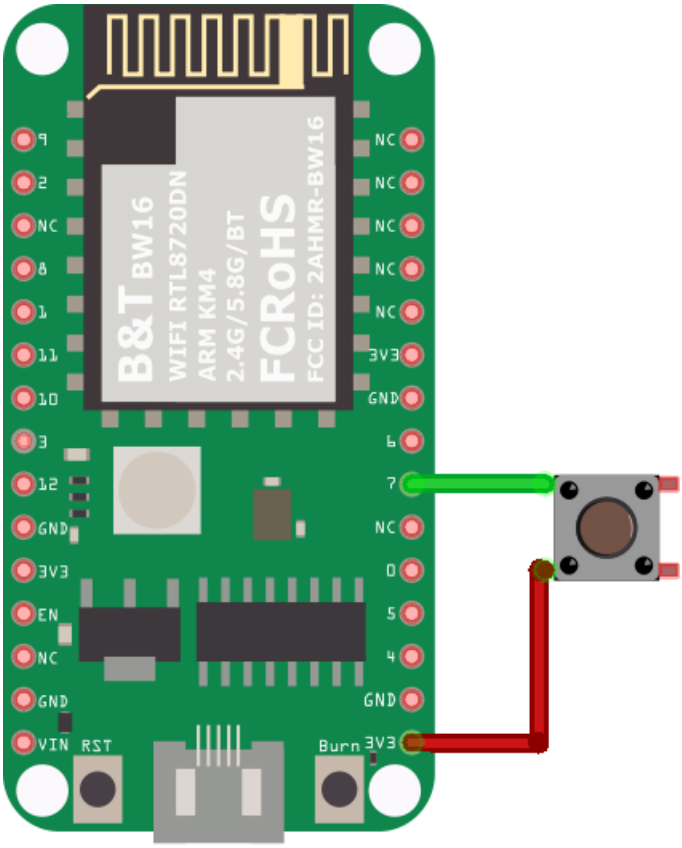
GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).



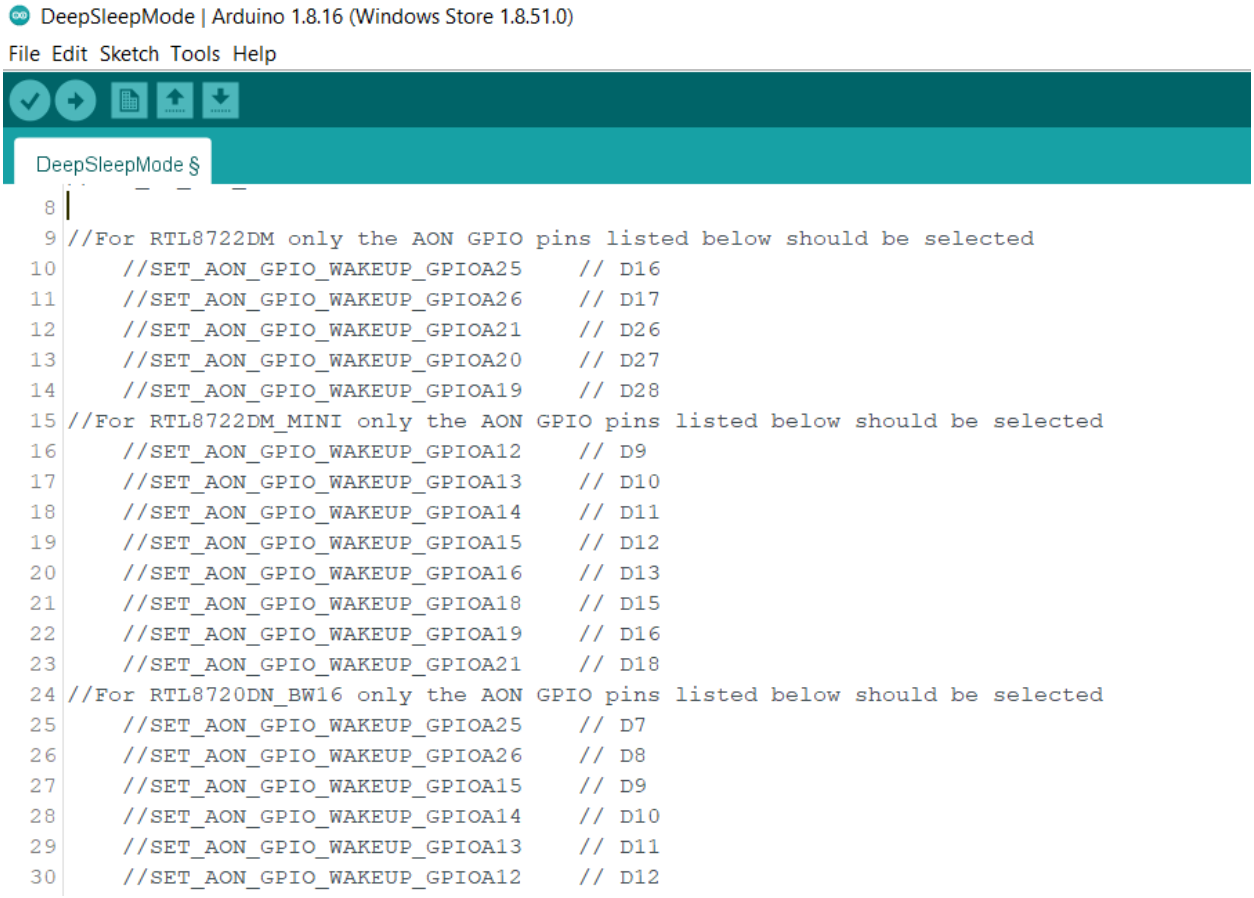
For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

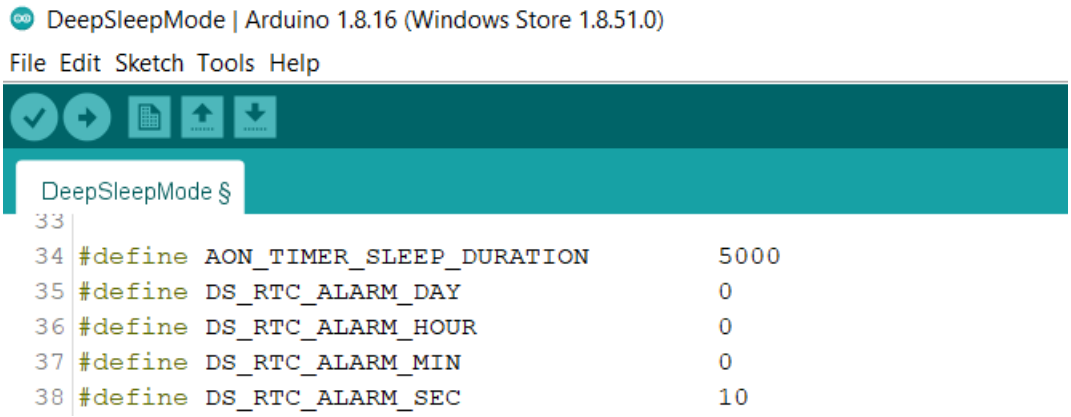


These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below



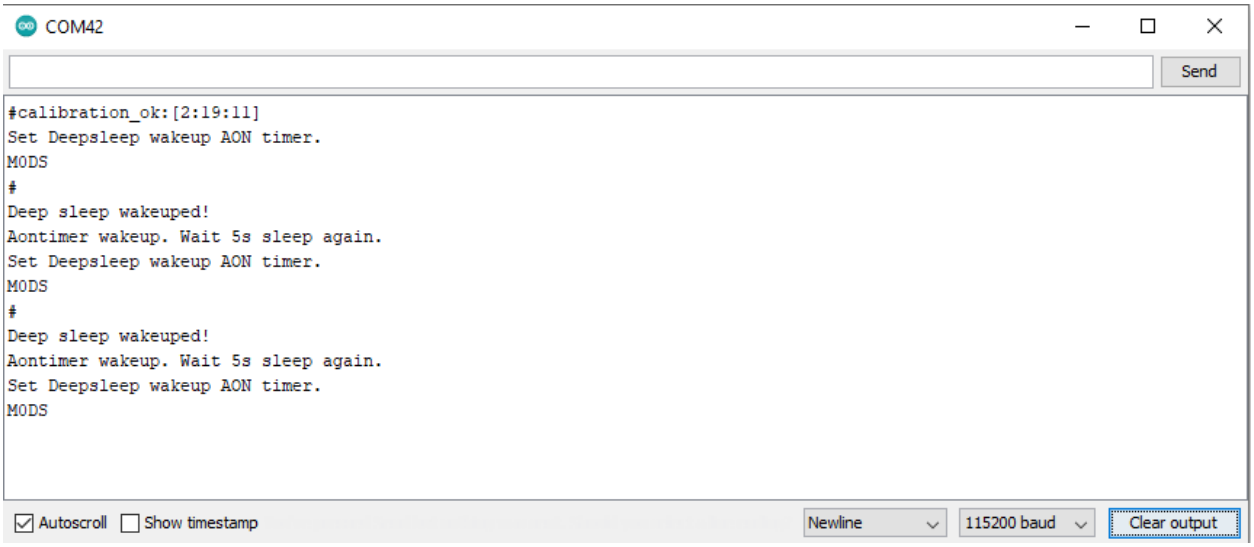
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

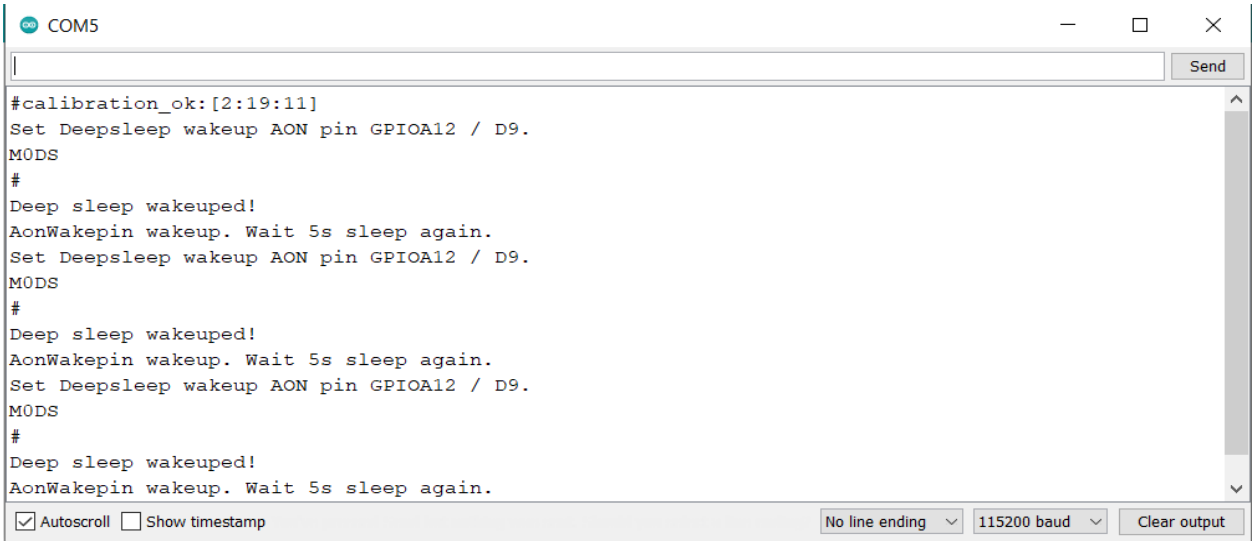


When all the condition values are set, the system will run and switch between normal and deep sleep mode which is controlled by the wakeup source. The serial monitor will display the switching log as shown below.

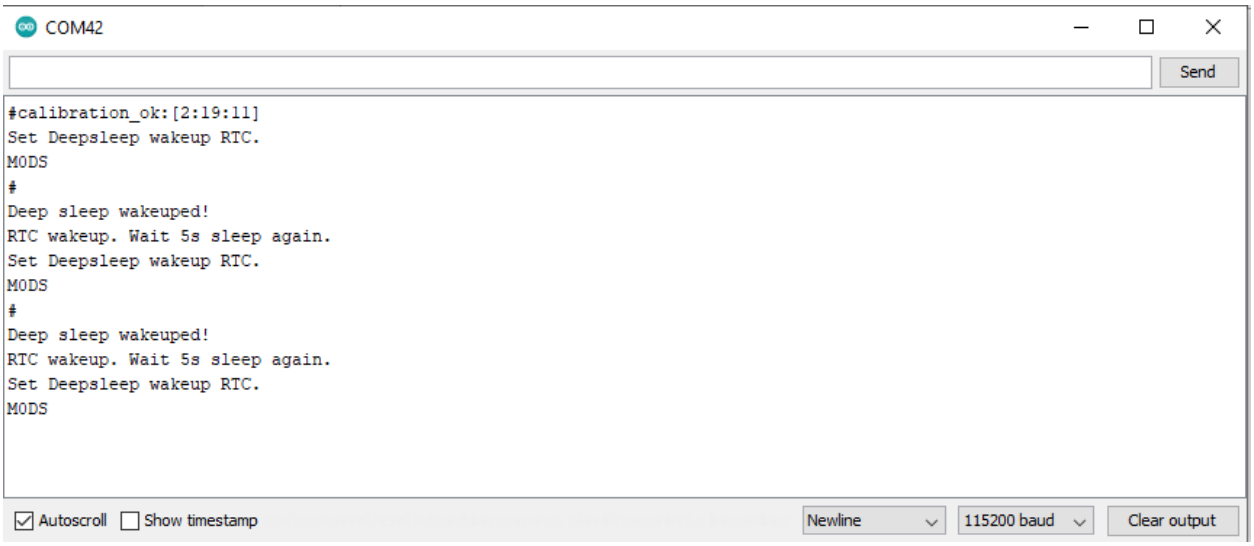
AON Timer



AON GPIO Pin



RTC Timer



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and E-paper

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

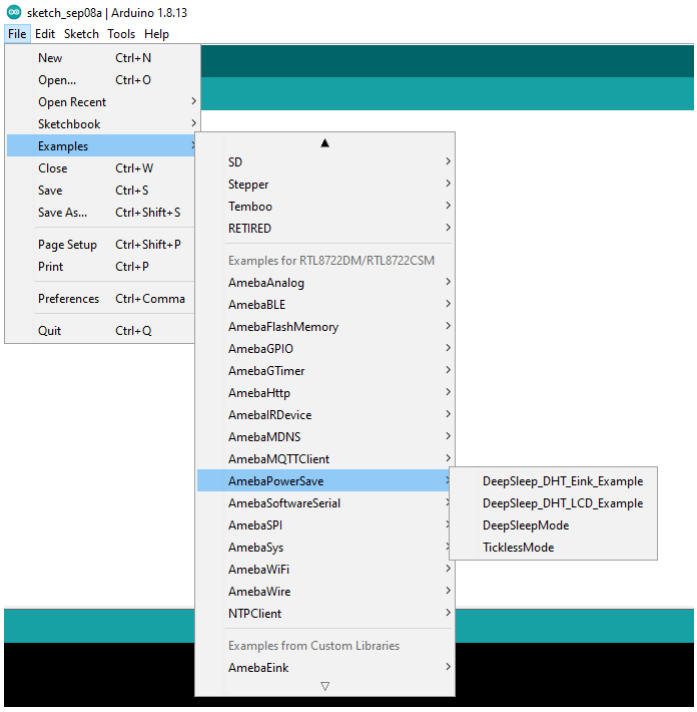
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between "working" and "sleep"(power save).Using DHT sensor to read data and display on Eink screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Download the Eink zip library, AmebaEink.zip, at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries. Then install the AmebaEink.zip.
Open "`File" -> "Examples" -> "AmebaPowerSave" -> "DeepSleep_DHT_Eink_Example`"



Set condition values as picture below.
`DS_WAKEUP_SOURCE` is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

Using AON Timer as wakeup source

AON timer can be set from 0 to 32760000 range (unit ms) by `AON_TIMER_SLEEP_DURATION`

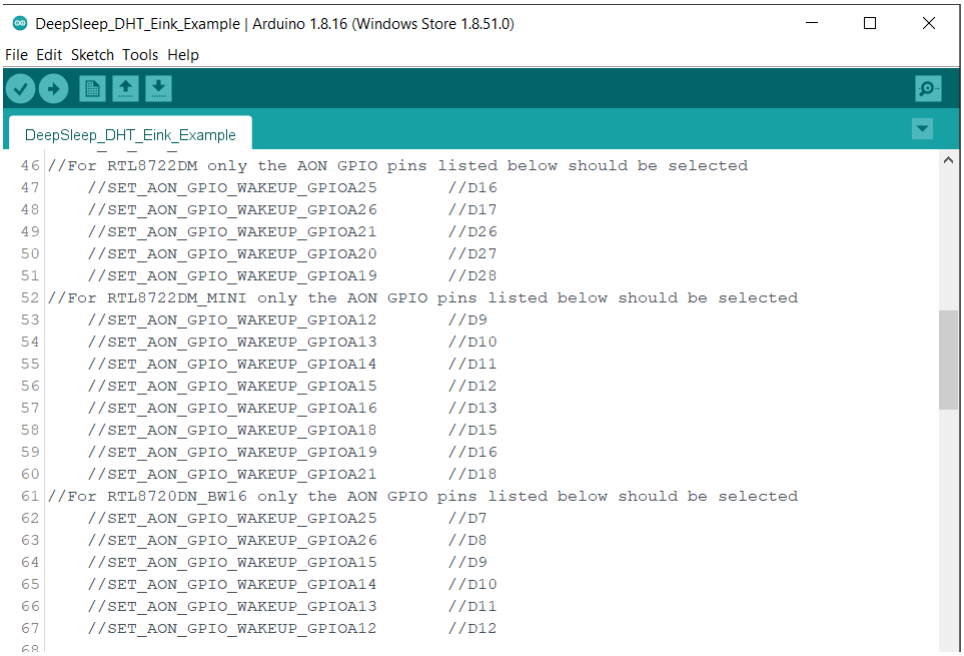
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

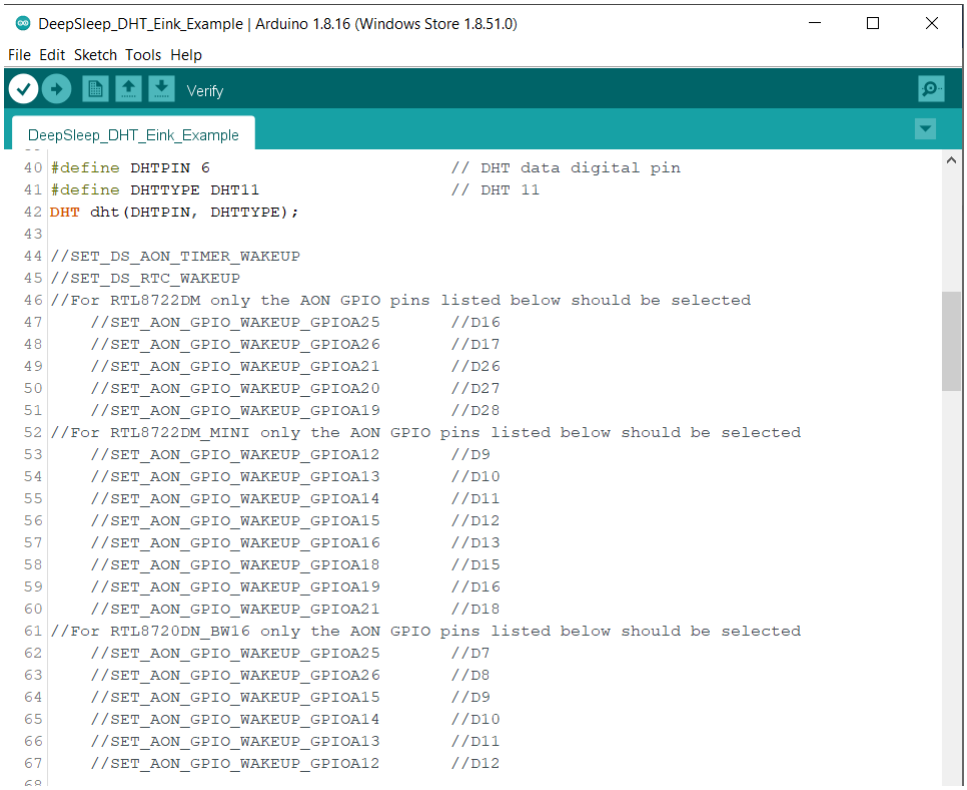
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. Eink screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and LCD

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

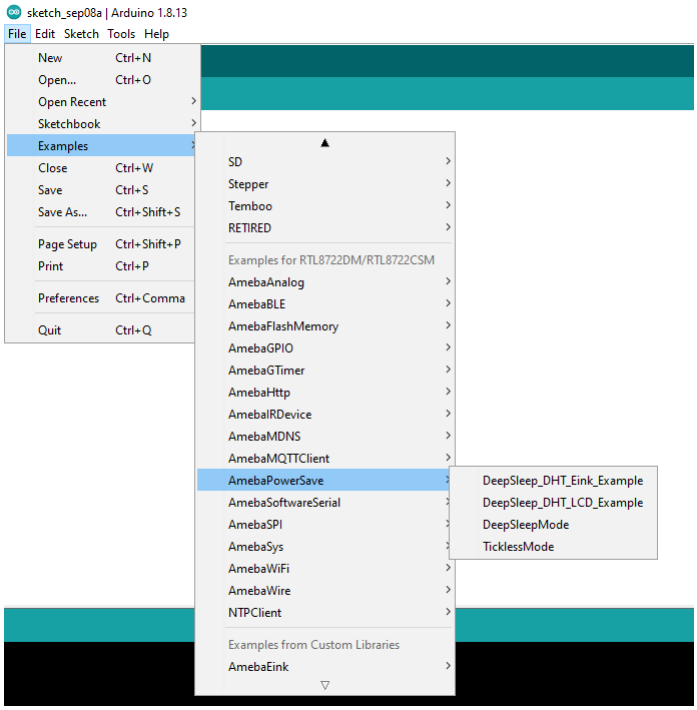
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between “working” and “sleep”(power save).Using DHT sensor to read data and display on LCD screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Open “File” -> “Examples” -> “AmebaPowerSave” -> “DeepSleep_DHT_LCD_Example”



Set condition values as picture below.

`DS_WAKEUP_SOURCE` is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

Using AON Timer as wakeup source

AON timer can be set from 0 to 32760000 range (unit ms) by `AON_TIMER_SLEEP_DURATION`

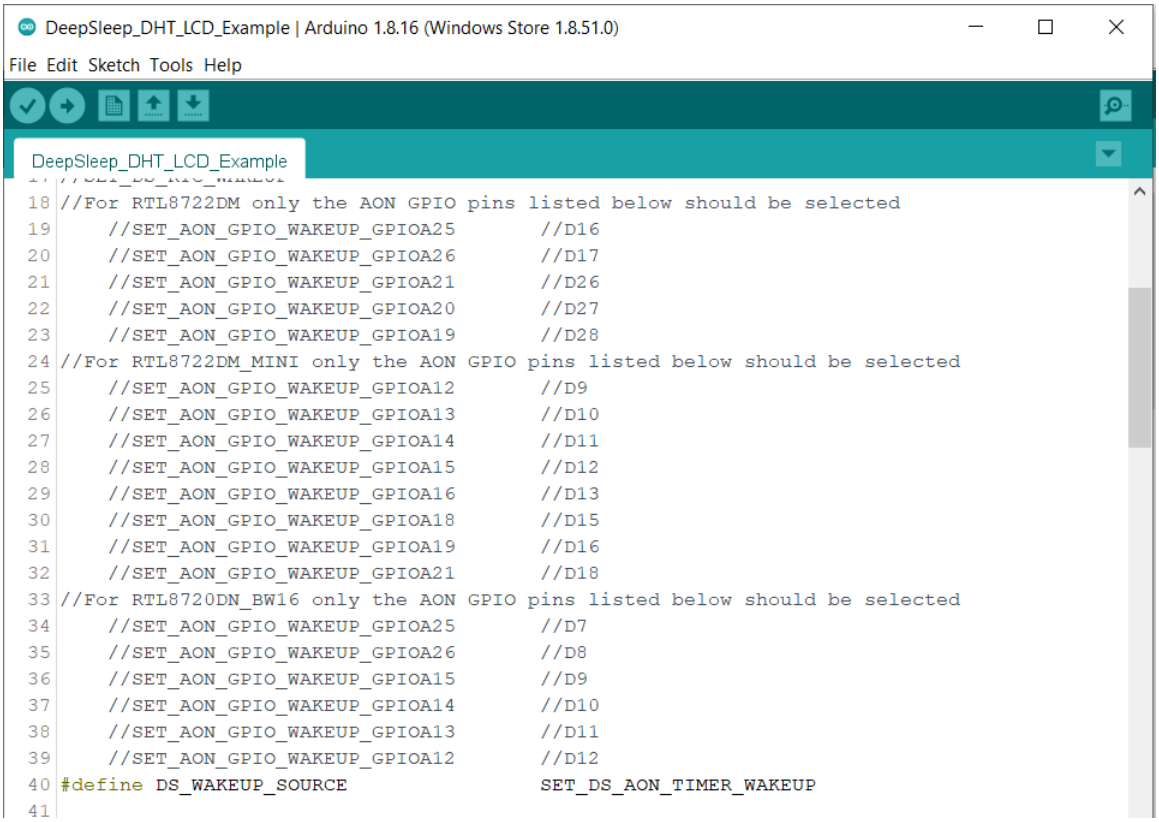
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

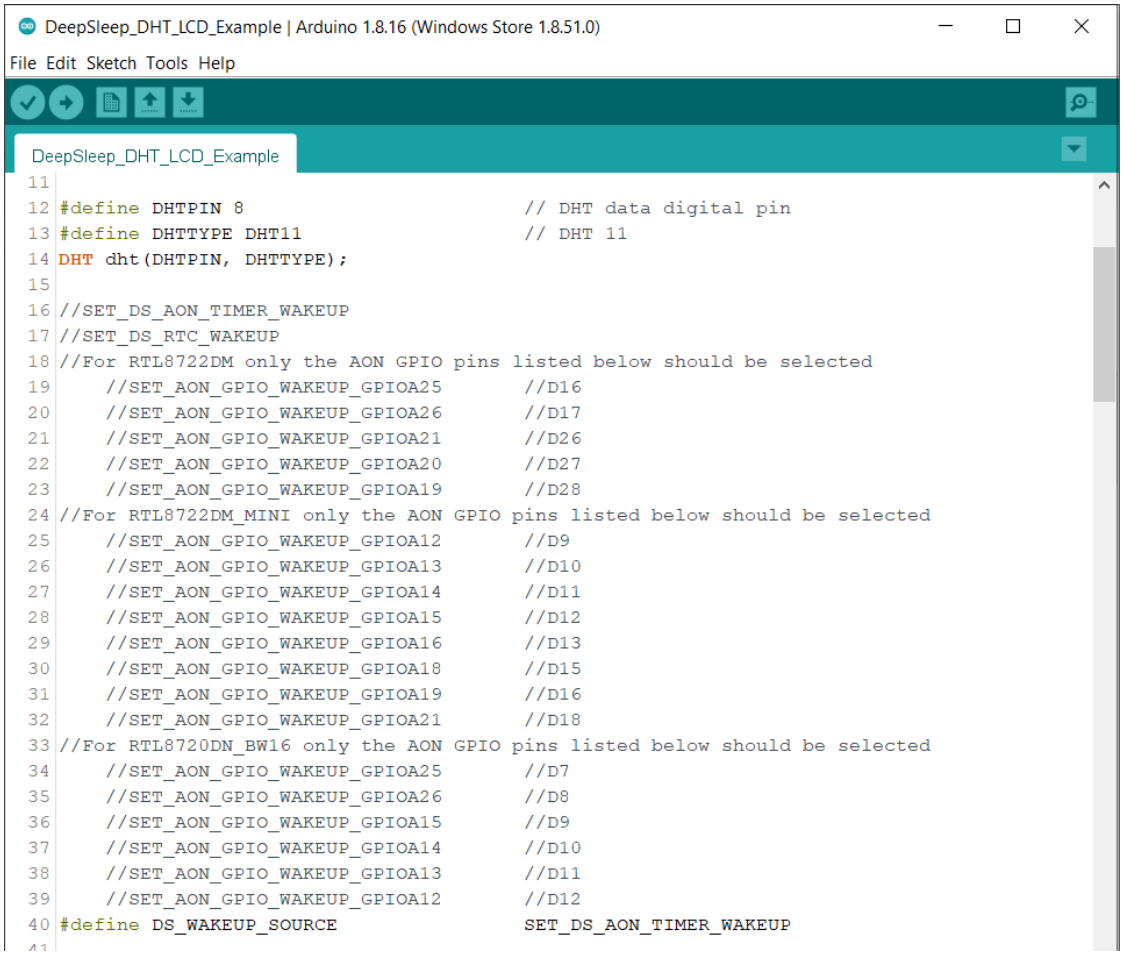
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. LCD screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Tickless Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

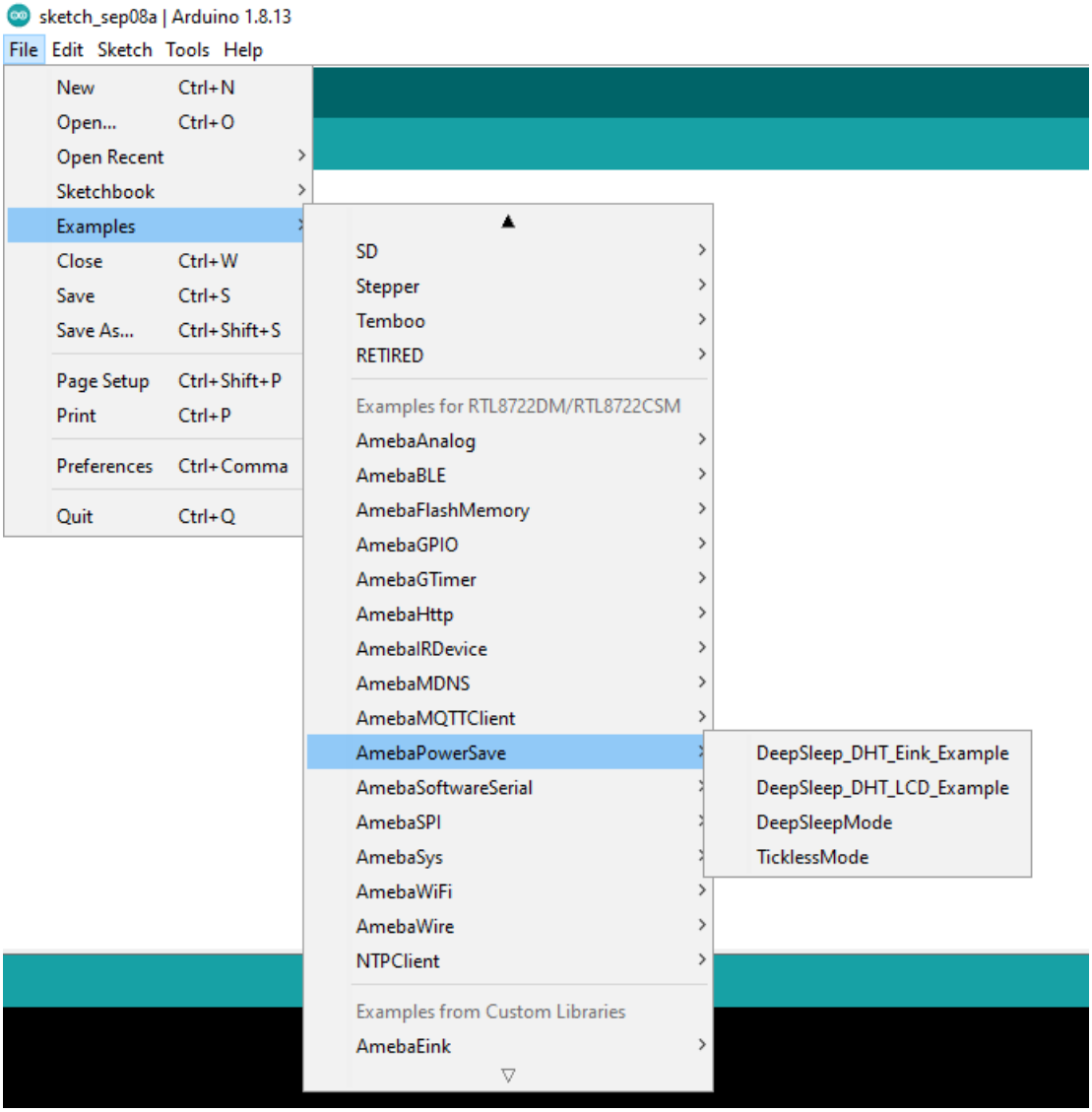
Example

Introduction

Ameba-D supports two low power modes which are deepsleep mode and sleep mode. The power consumptions of Tickless Sleep Mode is around 28uA to 30uA compare to normal state around 15mA. This example describes how to use freertos tickless with uart interruptable interface.

Procedure

Open “File” -> “Examples” -> “AmebaPowerSave” -> “TicklessMode”



Set condition values as picture below.

“TL_WAKEUP_SOURCE” is used to set the wake-up source, user can chose 3 wake up sources now,

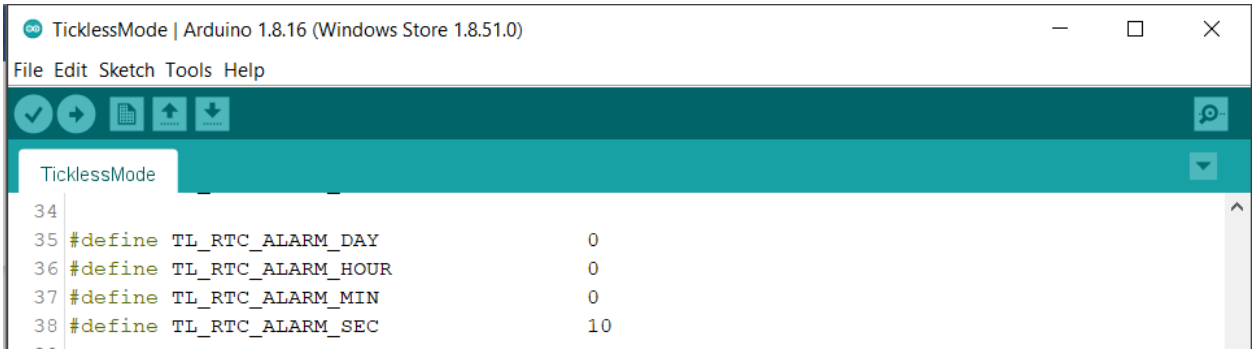
```
LOGUART(SET_TL_UART_WAKEUP);
RTC Timer(SET_TL_RTC_WAKEUP);
AON pins(SET_AON_WAKEPIN_WAKEUP);
```

Using LOGUART as wakeup source

When the LOGUART is selected as the wakeup source, the “TL_Suspend_function” will enter sleep mode. And then it is kept alive for 13s then enter sleep mode. To wakeup, press enter.

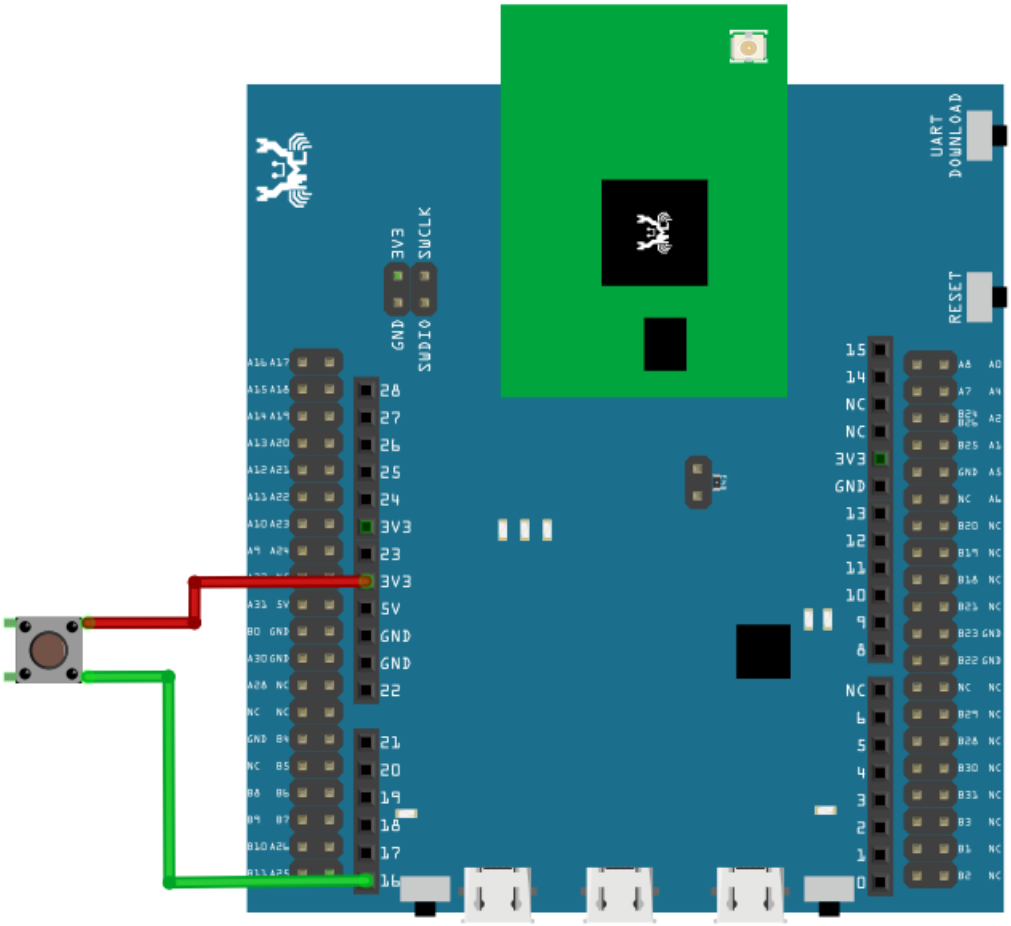
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values to be set, day, hour, min and sec. All 4 values can be set by DS_RTC_ALARM_DAY, DS_RTC_ALARM_HOUR, DS_RTC_ALARM_MIN, and DS_RTC_ALARM_SEC.

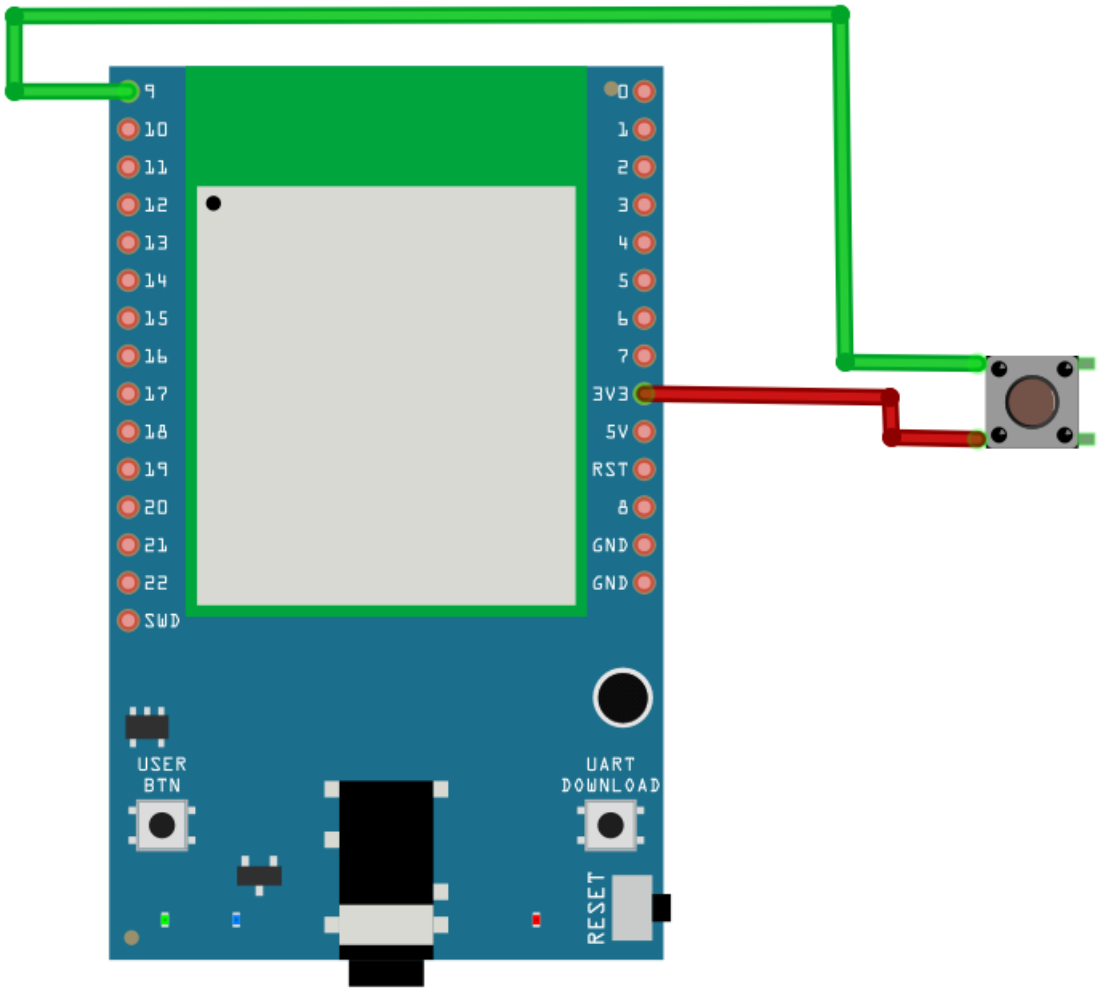


Using AON GPIO pins as wakeup source

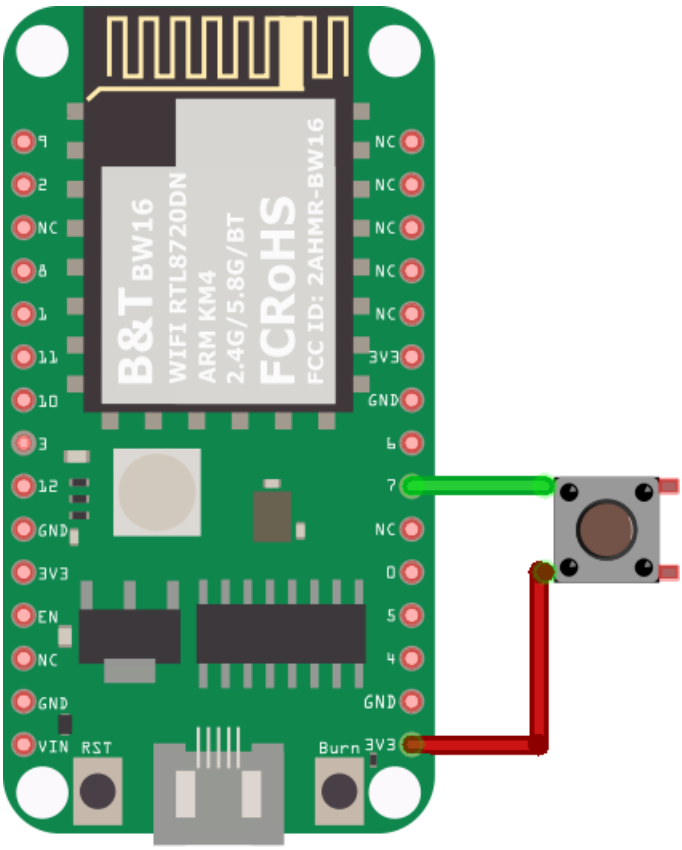
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

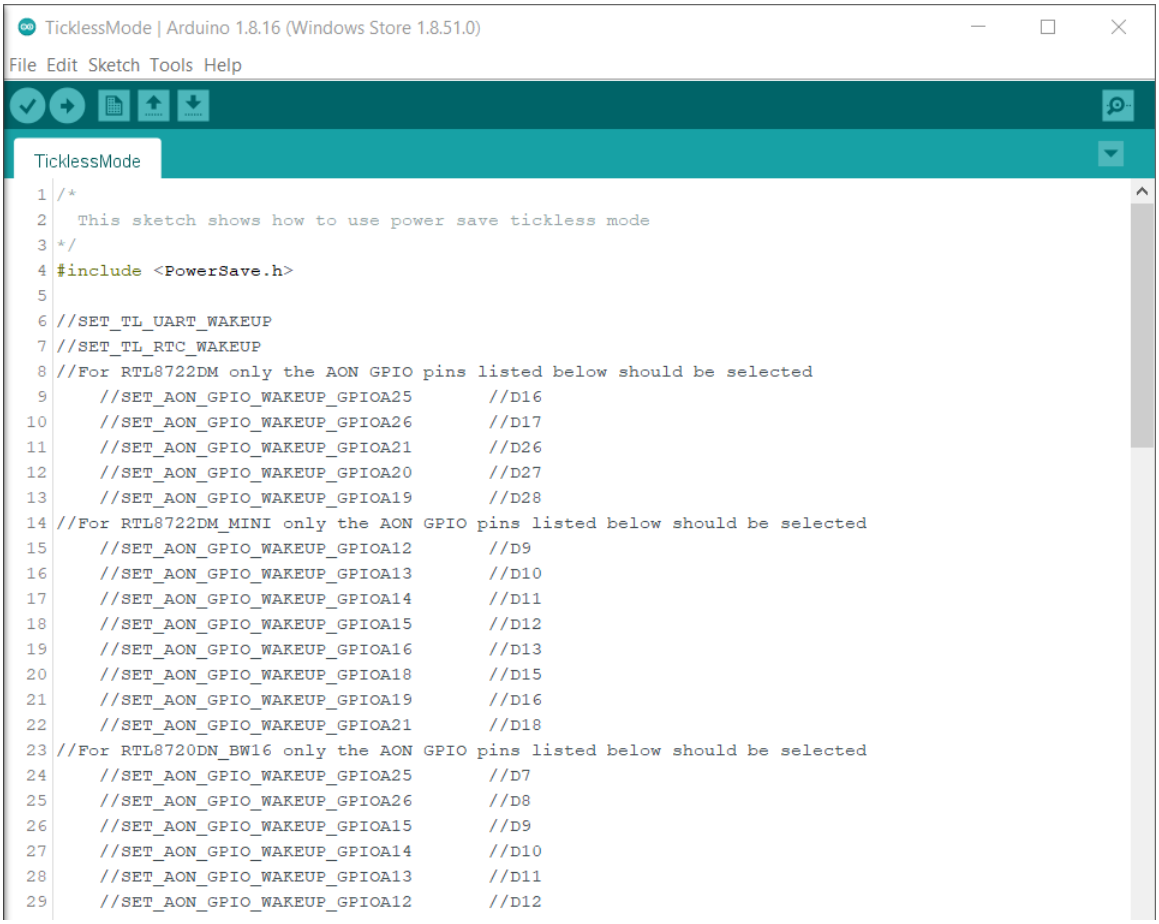


For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



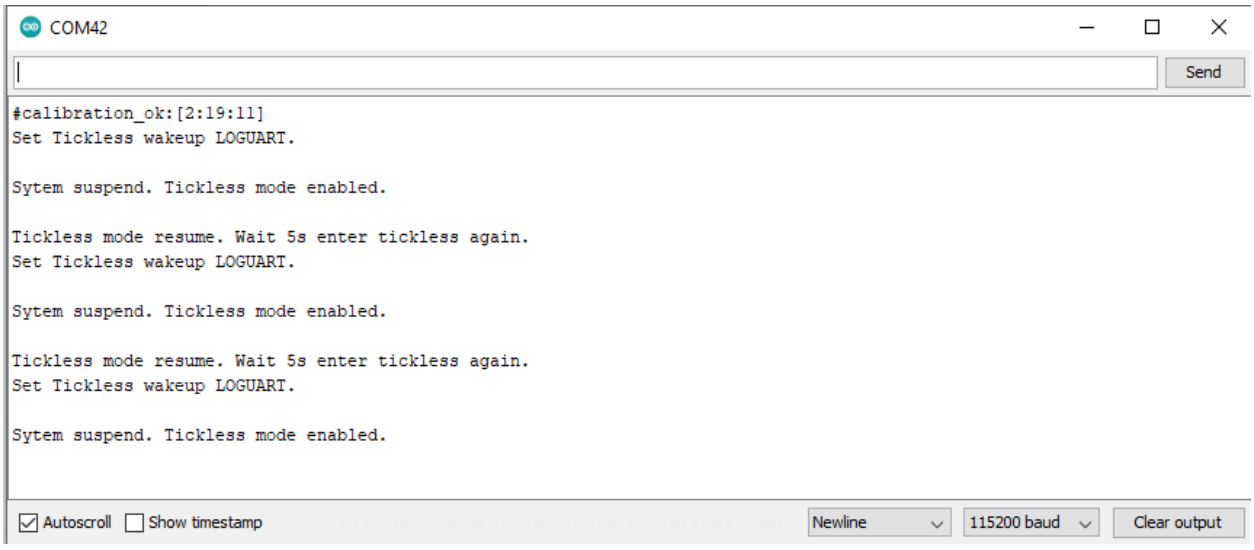
For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).


B&T BW16
WIFI RTL8720DN
ARM KM4
2.4G/5.8G/BT
FCC ID: 2AHMR-BW16
9
2
NC
8
1
11
10
3
12
GND
3V3
EN
NC
GND
VIN
RST
NC
NC
NC
NC
NC
3V3
GND
6
7
NC
0
5
4
GND
3V3
Burn

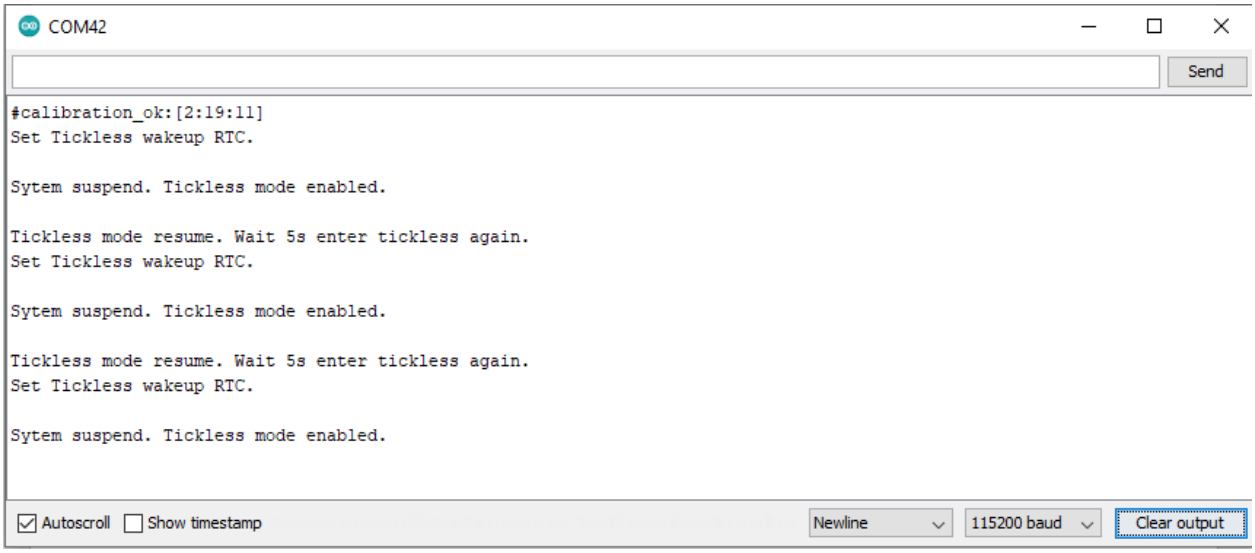


TL_SYSACTIVE_TIME is for setting time duration of the system to keep alive. (Unit ms)

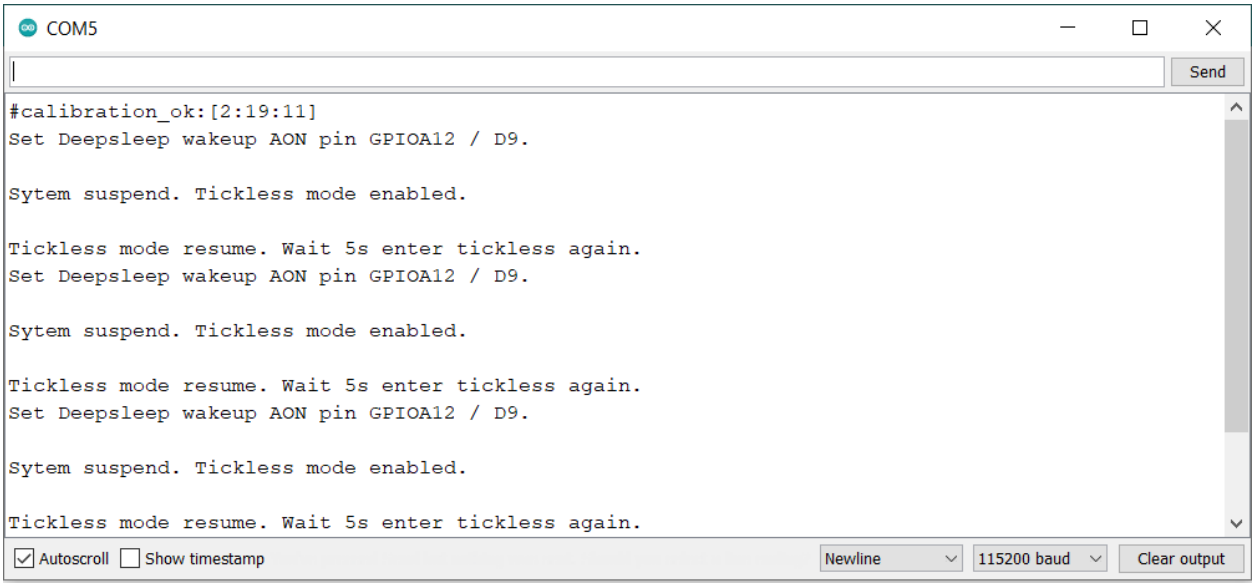
LOGUART



RTC Timer



AON GPIO Pins



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

PWM - Play Music by Buzzer

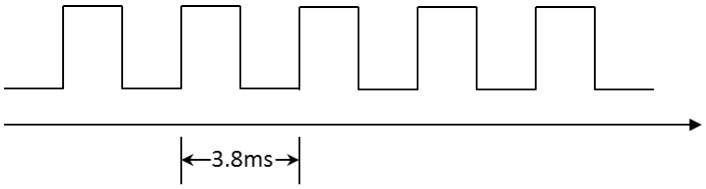
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Buzzer x 1

Example

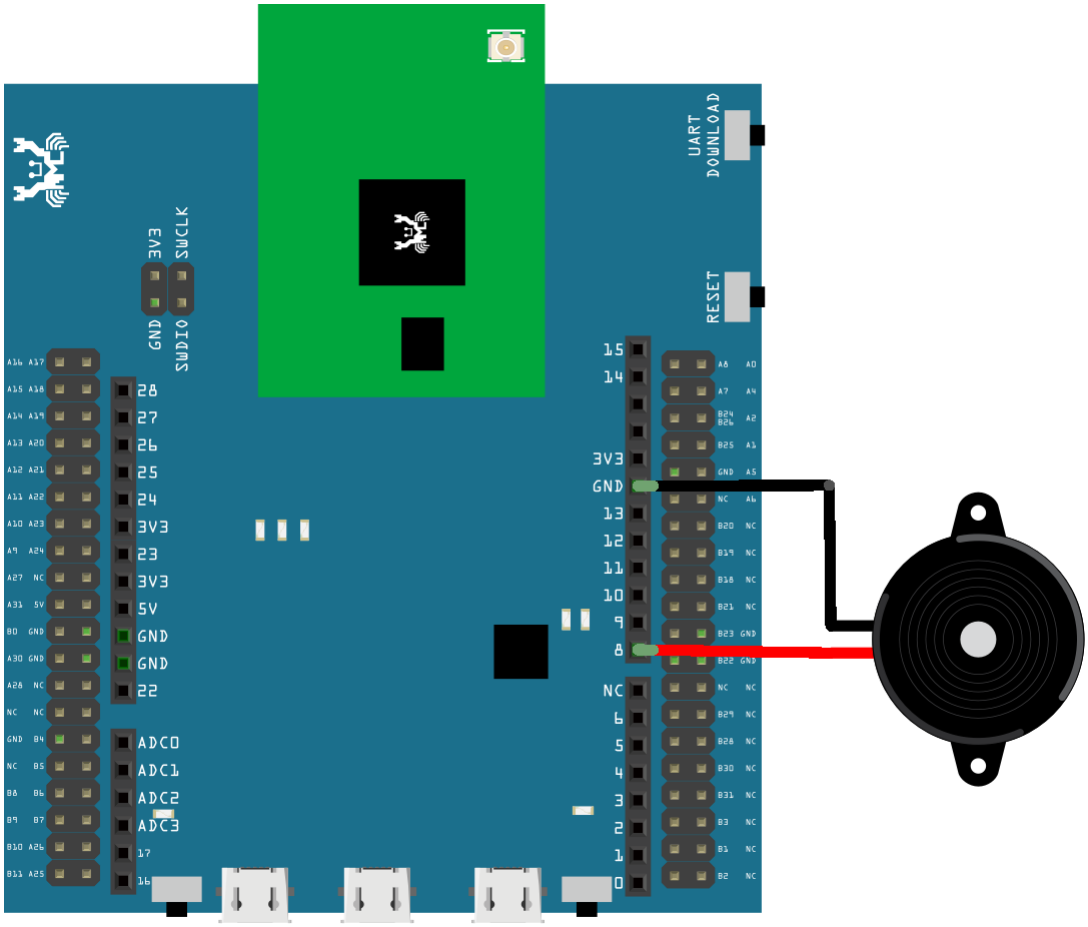
A sound is composed of volume, tone and timbre. Volume is determined by the amplitude of the sound wave. Tone is determined by the frequency of the sound wave. Timbre is determined by the waveform of the sound wave.

In this example, we use PWM to control the buzzer to emit sound with desired tone. As PWM outputs square wave, if we wish to emit tone C4 (frequency=262Hz), we have to make PWM to output square wave with wavelength 1/262 = 3.8ms:

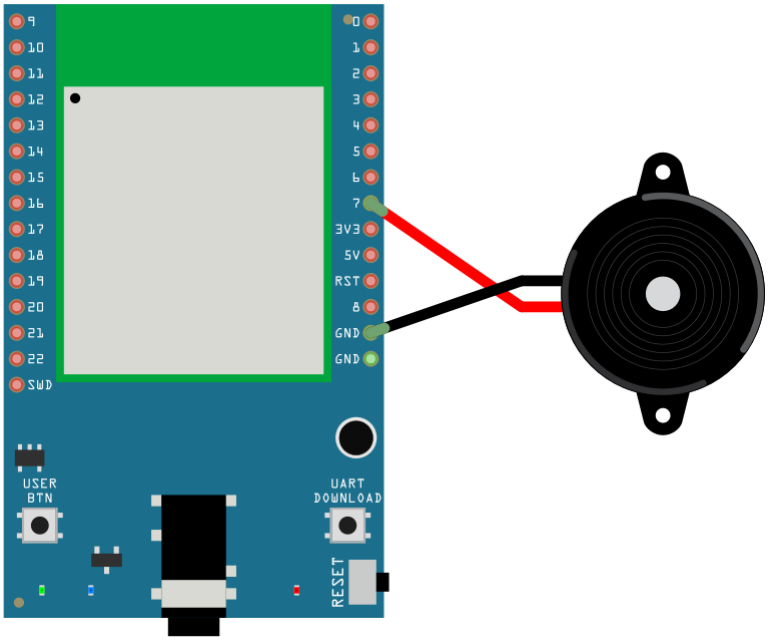


We use PWM to output sound wave with different frequency, so as to play music by the buzzer.
Connect the buzzer to the PWM output pin shown in the following diagrams.

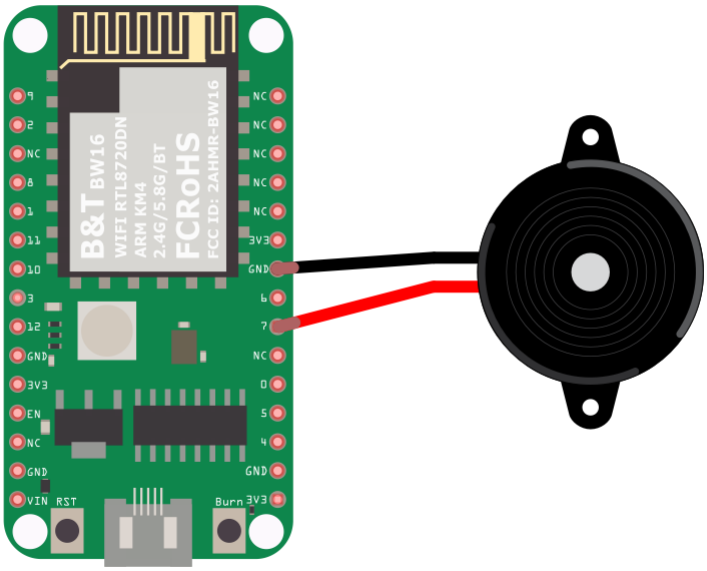
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example code in "`Examples`" -> "`AmebaAnalog`" -> "`TonePlayMelody`"
Compile and upload to Ameba, press the reset button. Then you can hear the buzzer playing music.

Code Reference

Ameba implement the tone() and noTone() API of Arduino:
https://www.arduino.cc/en/Reference/Tone
https://www.arduino.cc/en/Reference/NoTone

In the sample code, we initiate a melody array, which stores the tones to make. Another array, noteDurations, contains the length of each tone, 4 represents quarter note (equals to 3000ms/4 = 750ms, and plus an extra 30% time pause), 8 represents eighth note.

PWM - Servo Control

Preparation

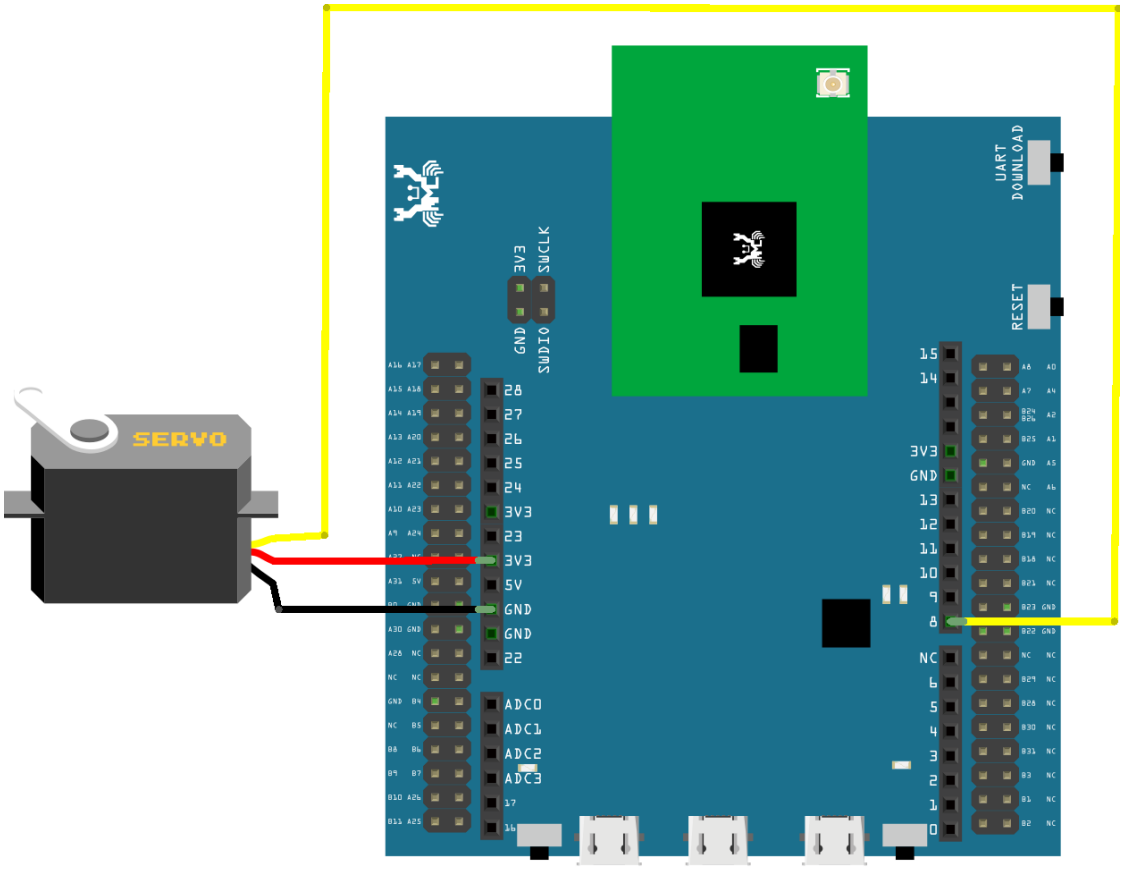
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Servo x 1 (Ex. Tower Pro SG90)

Example

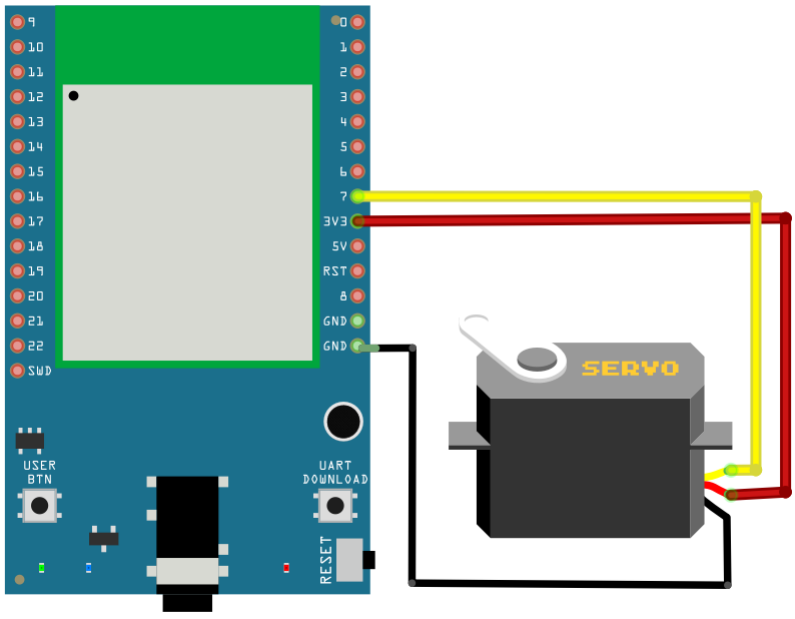
A typical servo has 3 wires, the red wire is for power, black or brown one should be connected to GND, and the other one is for signal data. We use PWM signal to control the rotation angle of the axis of the servo. The frequency of the signal is 50Hz, that is length 20ms. Each servo defines its pulse bandwidth, which is usually 1ms~2ms.

To control the rotation angle, for example if 1ms-length pulse rotates the axis to degree 0, then 1.5 ms pulse rotates the axis to 90 degrees, and 2 ms pulse rotates the axis to 180 degrees. Furthermore, a servo defines the “dead bandwidth”, which stands for the required minimum difference of the length of two consecutive pulse for the servo to work.

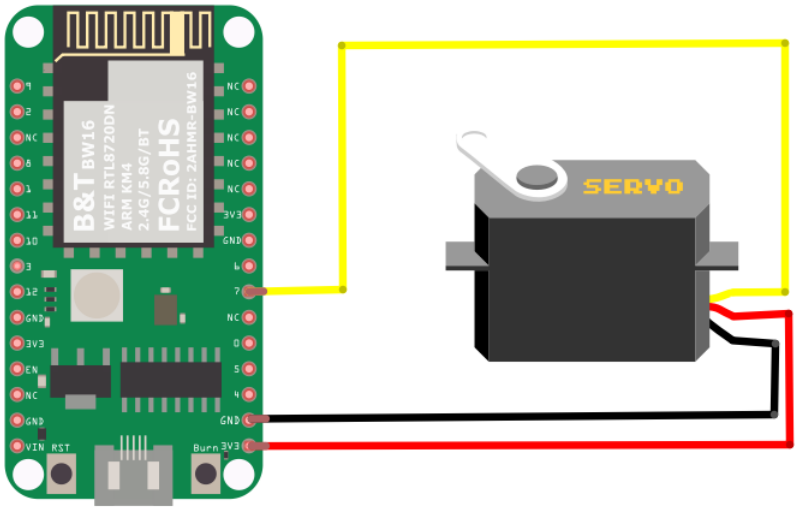
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example, "`File" -> "Examples" -> "AmebaAnalog" -> "ServoSweep`"
This example makes the servo to rotate from degree 0 to 180, and then rotate back to degree 0.

Code Reference

The Servo API of Ameba is similar to the API of Arduino. To distinguish from the original API of Arduino, we name the header file "AmebaServo.h" and the Class "AmebaServo", the usage is identical to the Arduino API.

The default pulse bandwidth of Arduino Servo is 0.5ms~2.4ms, which is the same as Tower Pro SG90. Therefore, we set the attached pin directly:

```
myservo.attach(9);
```

Next, rotate the axis to desired position:

```
myservo.write(pos);
```

RTC - Simple RTC

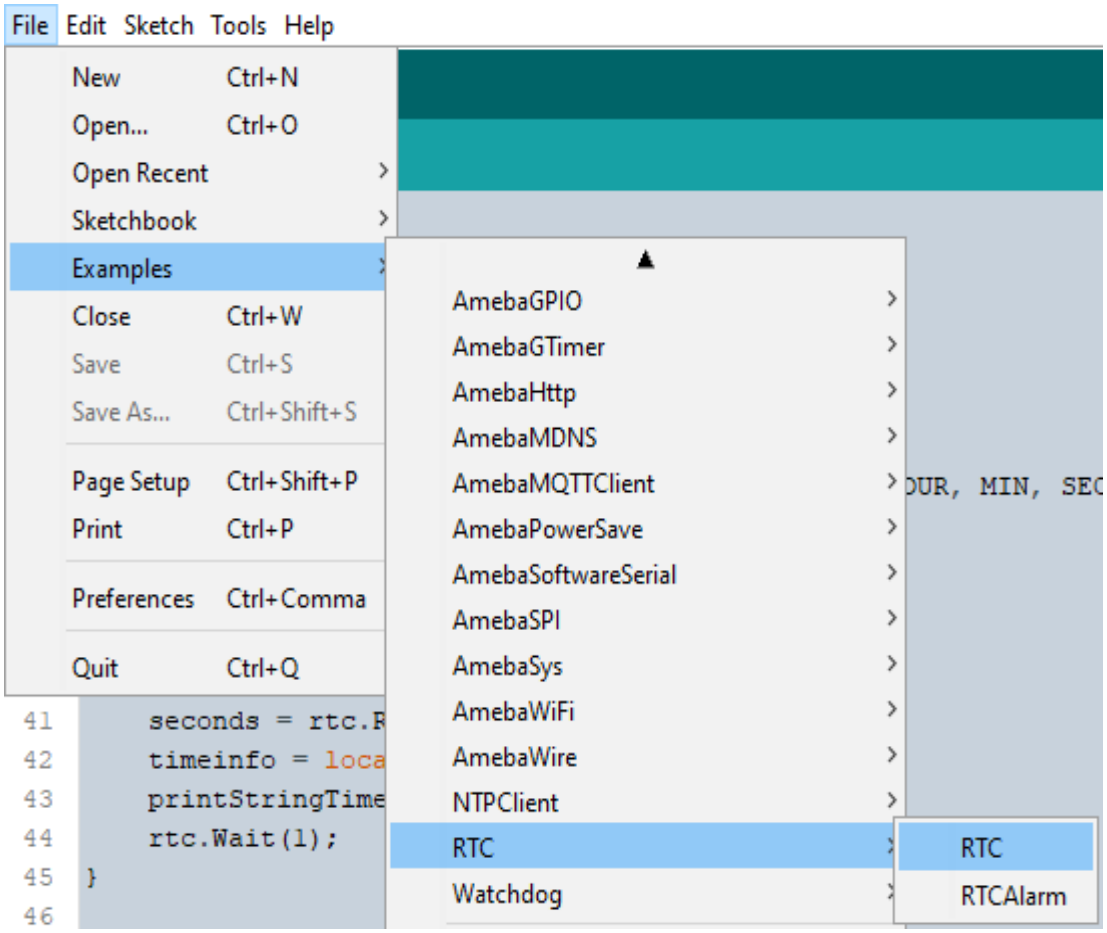
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

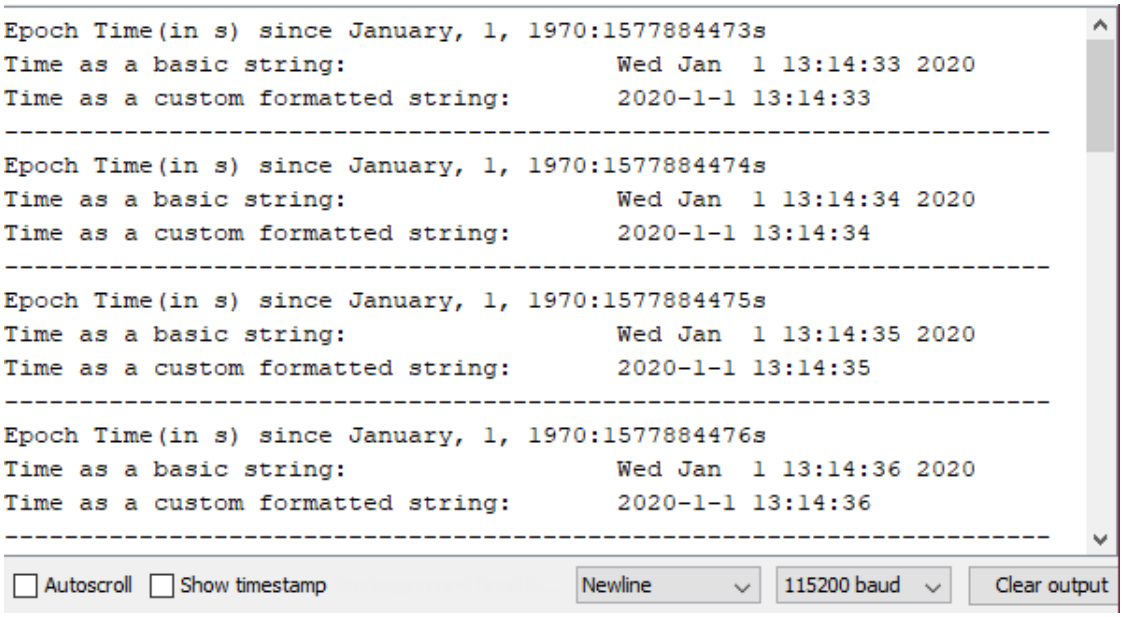
Example

This example demonstrates how to use the RTC library methods. This function describes how to use the RTC API. The RTC function is implemented by an independent BCD timer/counter.

Select the correct Ameba development board from the Arduino IDE: “Tools” -> “Board”.
Then open the “RTC” example from:
`"File" -> "Examples" -> "AmebaRTC" -> "RTC":`



Upon successfully upload the sample code and press the reset button, this example will print out time information since the user initialized time every second in the Serial Monitor.



Code Reference

[1] Simple RTC example from Arduino Tutorials:
https://www.arduino.cc/en/Tutorial/SimpleRTC

RTC - Simple RTC Alarm

Materials

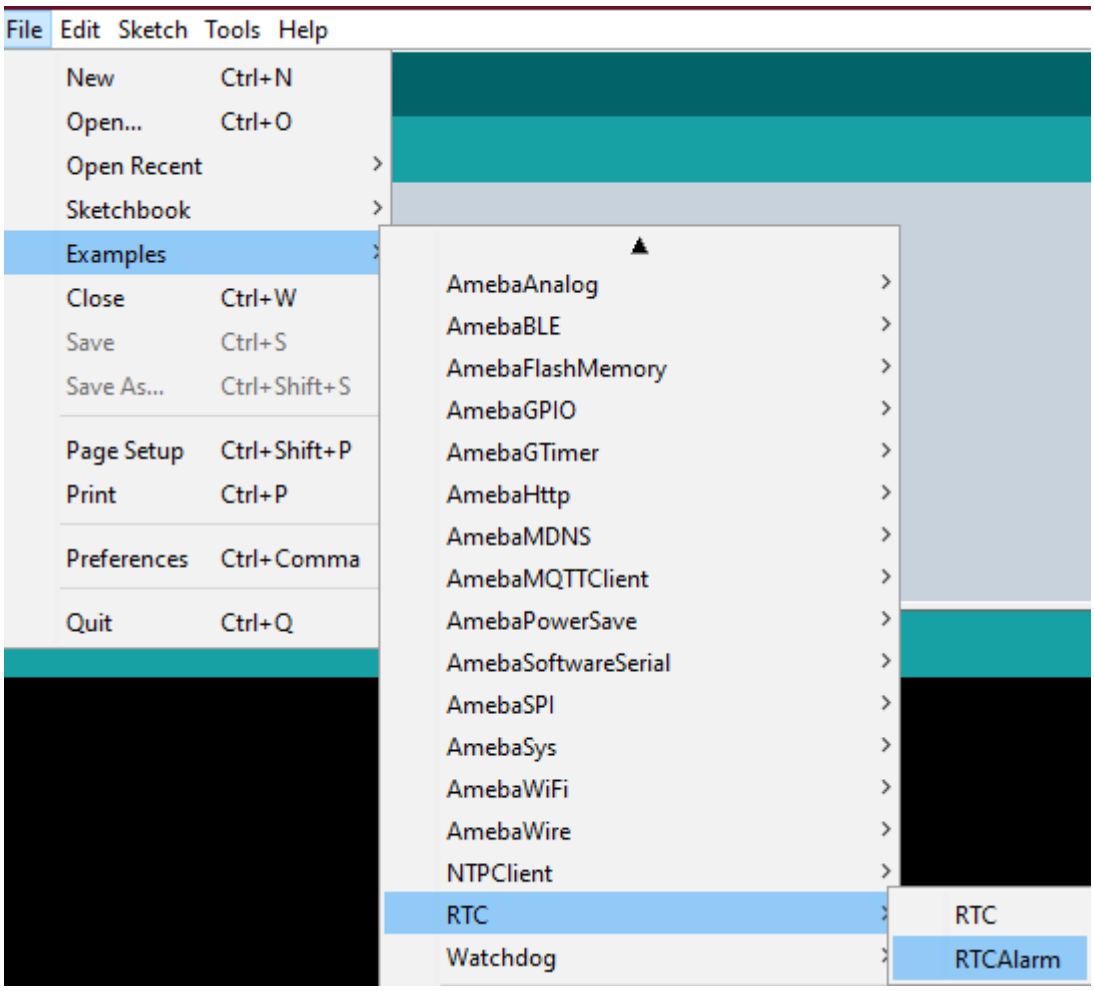
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

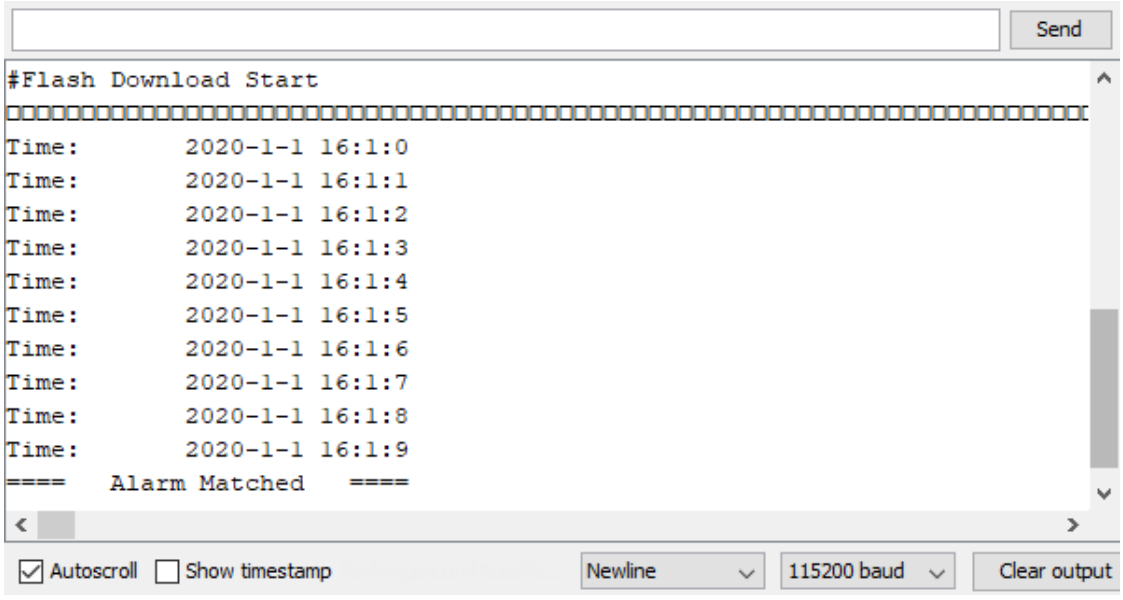
This example demonstrates how to use the RTC library methods to create a RTC Alarm, so that to do some tasks when an alarm is matched. In particular, the RTC time is set at 16:00:00 and an alarm at 16:00:10. When the time matches, "Alarm Match" information will be printed on the serial monitor.

First, select the correct Ameba development board from the Arduino IDE: "Tools" -> "Board".

Then open the "RTCAlarm" example from: `"File" -> "Examples" -> "RTC" -> "RTCAlarm"`:



In the example, the RTC time is set at 16:00:00 and an alarm is set at 16:00:10. Upon successfully upload the sample code and press the reset button. When the alarm time (10 seconds) is reached the attached interrupt function will print the following information: “Alarm Matched!” showing in this figure below.



SPI – Print Image And Text On LCD Screen

If you are not familiar with SPI, please read Introduction to SPI first.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1

Example

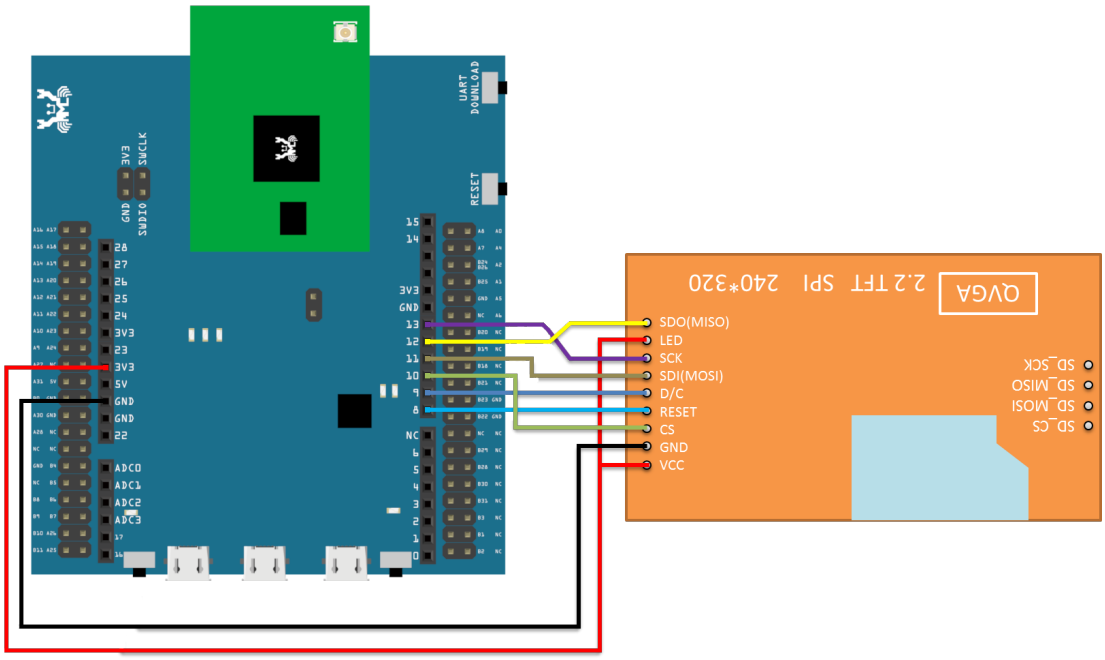
We have tested the following two models of ILI9341 TFT LCD with SPI interface:

- Adafruit 2.8″ TFT LCD (with touch screen)
  - https://www.adafruit.com/products/1651
  - https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2?view=all
- QVGA 2.2″ TFT LCD
  - http://www.lcdwiki.com/2.2inch_SPI_Module_ILI9341_SKU:MSP2202

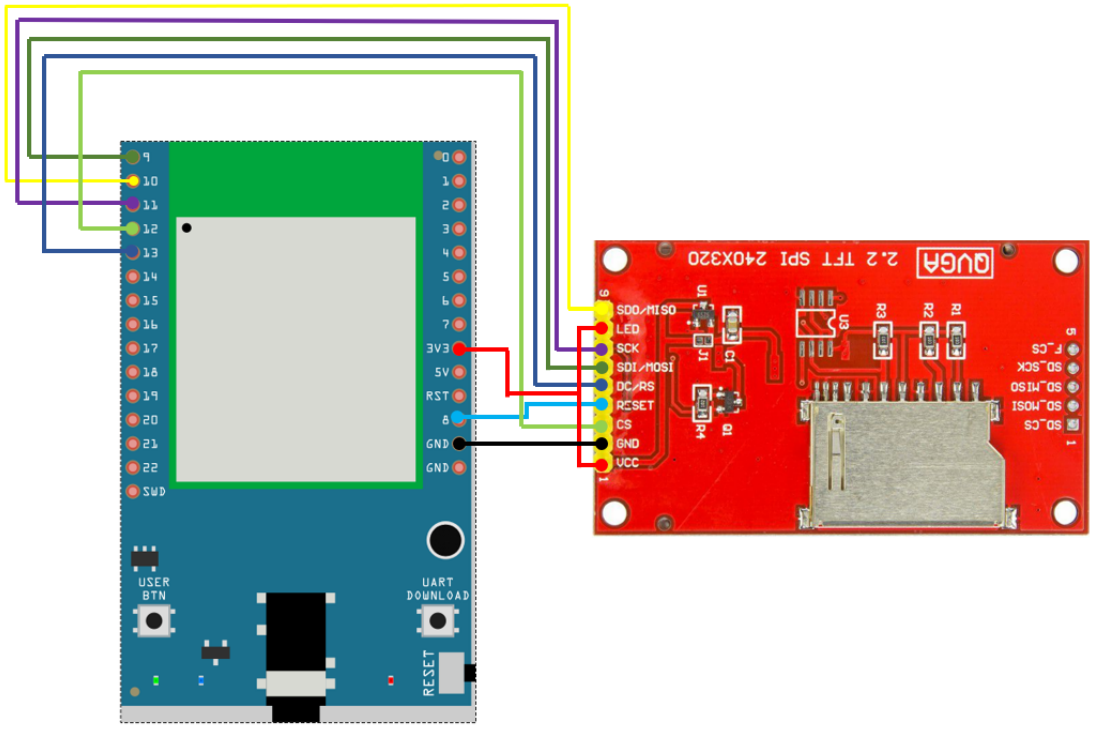
Common pins in ILI9341 TFT LCD with SPI interface:

- MOSI: Standard SPI Pin
- MISO: Standard SPI Pin
- SLK: Standard SPI Pin
- CS: Standard SPI Pin
- RESET: Used to reboot LCD.
- D/C: Data/Command. When it is at Low, the signal transmitted are commands, otherwise the data transmitted are data.
- LED (or BL): Adapt the screen backlight. Can be controlled by PWM or connected to VCC for 100% backlight.
- VCC: Connected to 3V or 5V, depends on its spec.
- GND: Connected to GND.

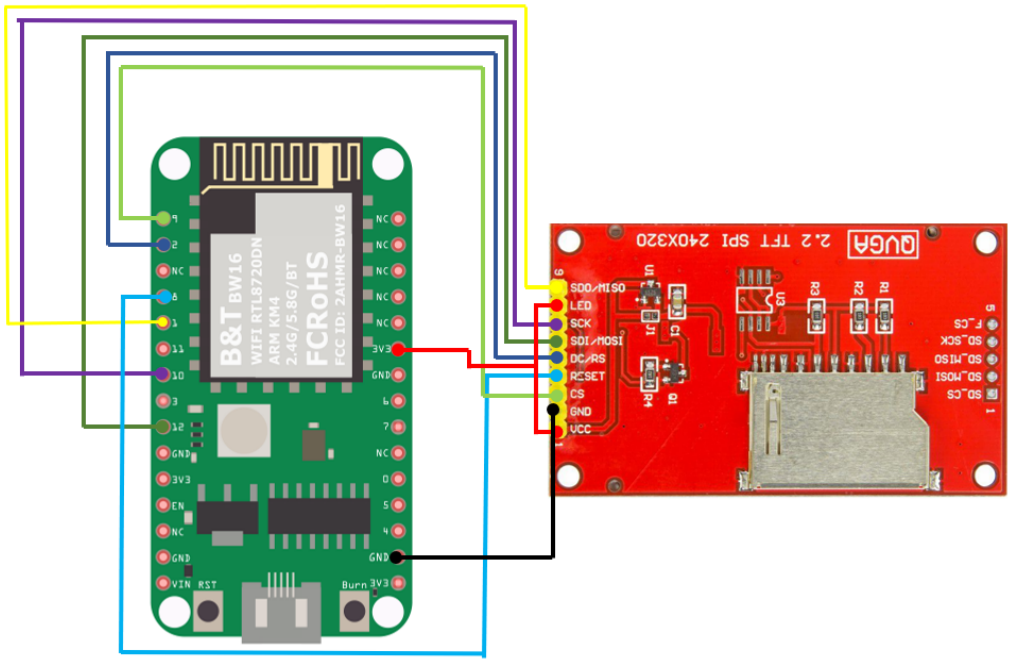
AMB21/ AMB22 and QVGA TFT LCD Wiring Diagram:



AMB23 and QVGA TFT LCD Wiring Diagram:



BW16 and QVGA TFT LCD Wiring Diagram:

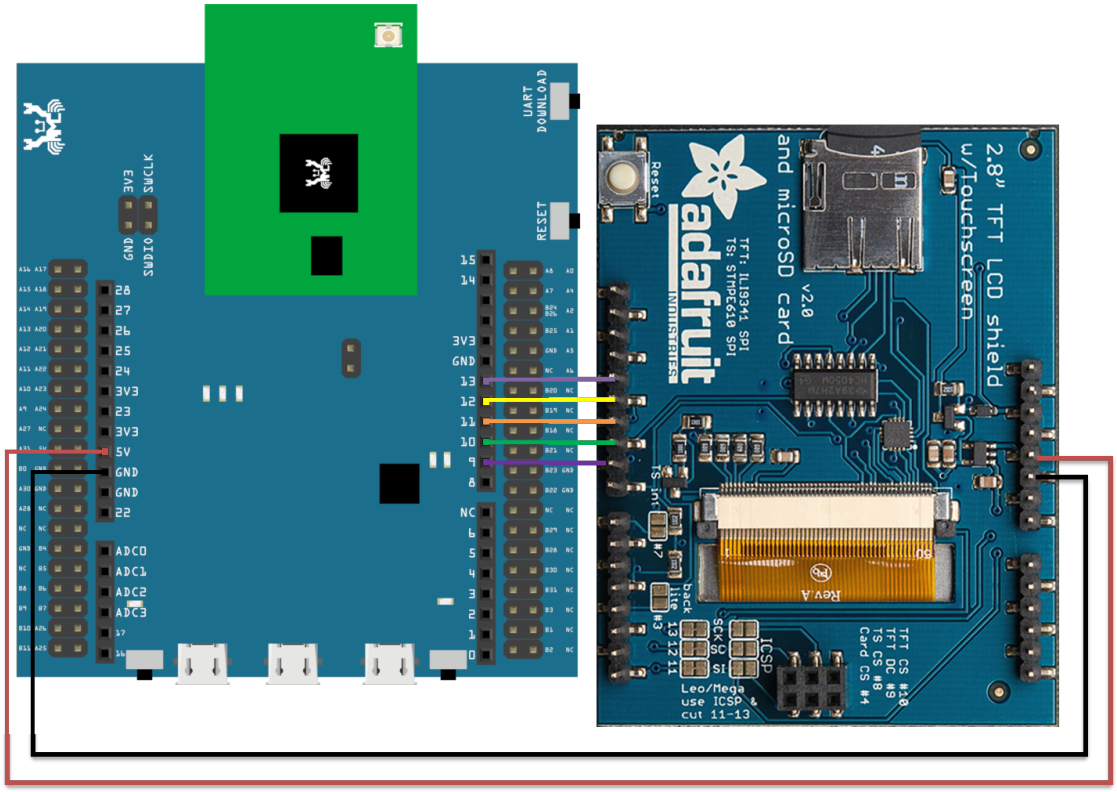


Wiring example of Adafruit 2.8” TFT LCD touch shield:

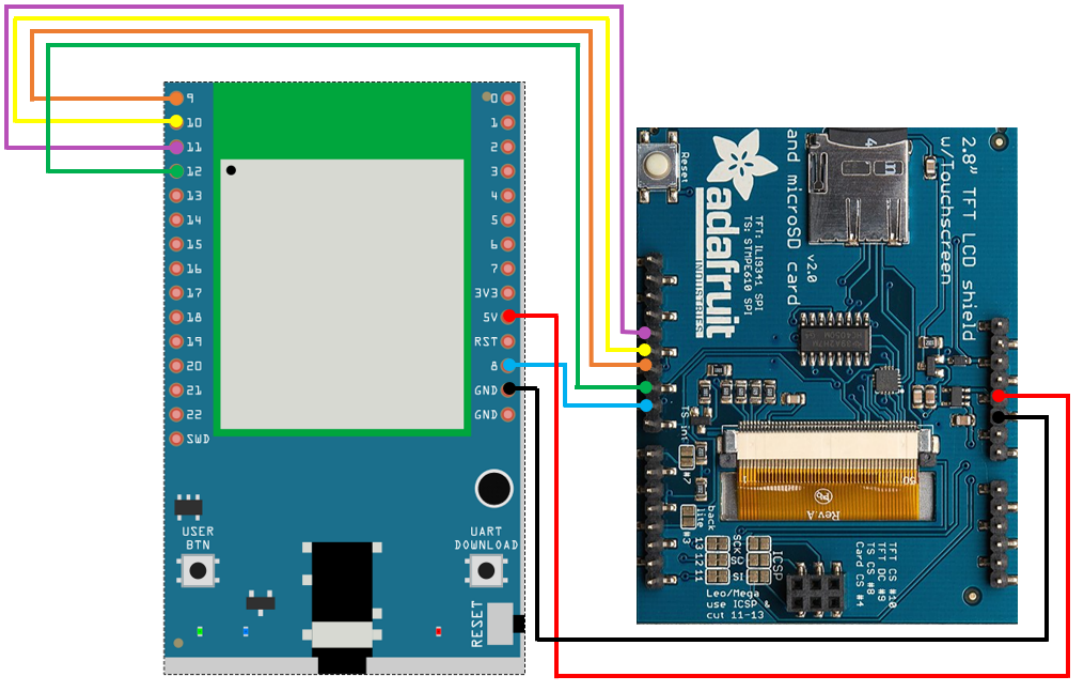
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.

AMB21 / AMB22 and Adafruit 2.8” TFT LCD touch shield Wiring Diagram:

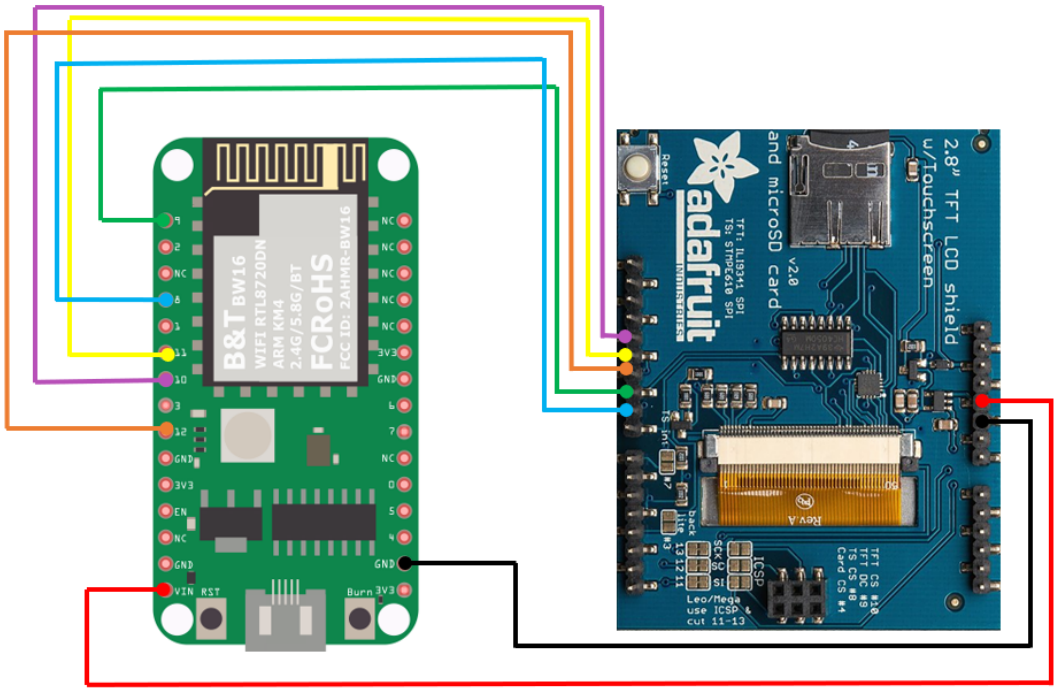
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.



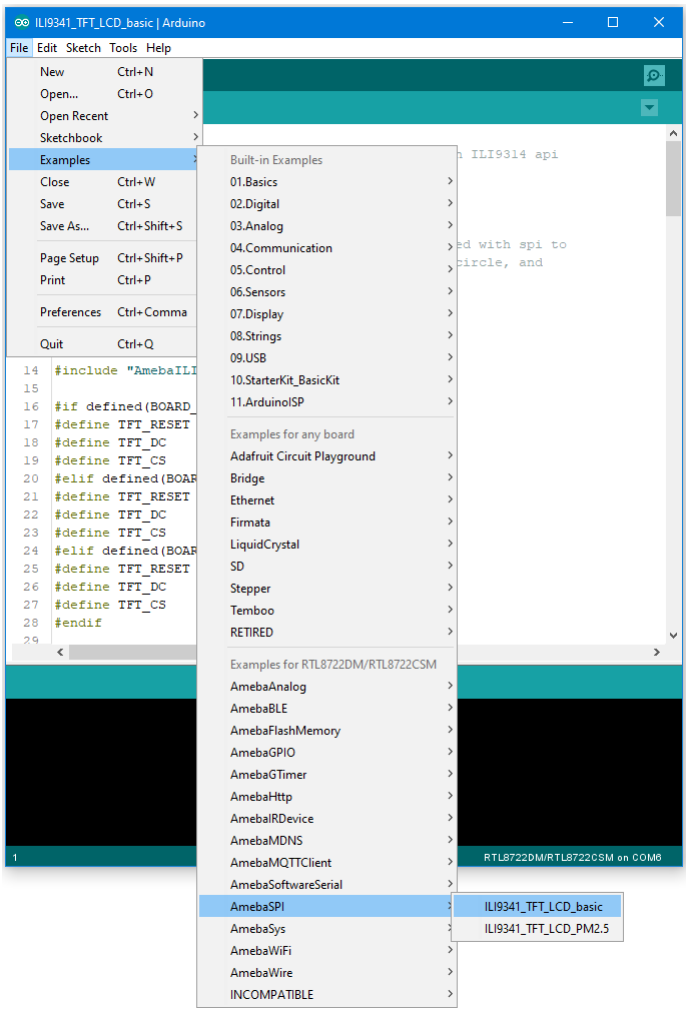
AMB23 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:



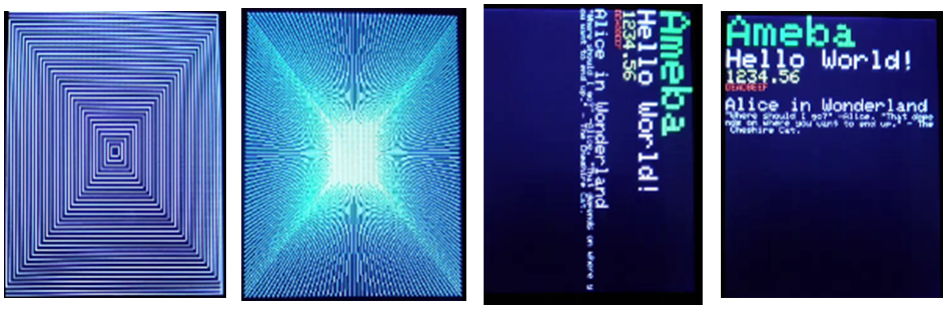
BW16 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:



Open the example, “Files” -> “Examples” -> “AmebaSPI” -> “ILI9341_TFT_LCD_basic”



Compile and upload to Ameba, then press the reset button.
Then you can see some display tests appear on the LCD screen, such as displaying different colors, drawing vertical and horizontal lines, drawing circles, etc.…



Code Reference

- **RGB 16-bit**

  ILI9341 uses RGB 16-bit to display colors. Different from RGB 24-bit, it uses 5 bits for red, 6 bits for green, 5

bits for blue. For example, the RGB 24-bit representation of sky blue is 0x87CEFF, that is in binary:

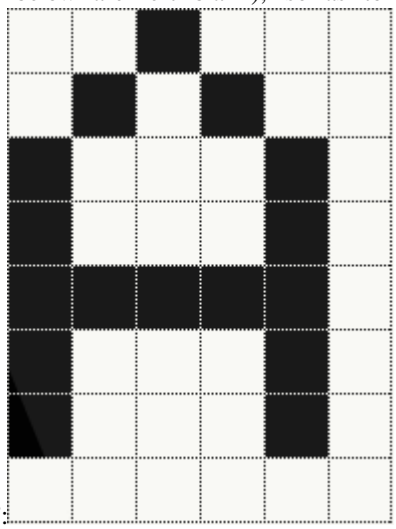
  - Red: 0x87 = B10000111
  - Green: 0xCE = B11001110
  - Blue: 0xFF = B11111111

and converted to RGB 16-bit:

  - Red: B10000
  - Green: B110011
  - Blue: B11111

Then concatenate them, which forms B1000011001111111 = 0x867F

- **Drawing of ILI9341**
  - First you must specify the range of the rectangle to draw, then pass the 2-byte RGB 16-bit color to ILI9341 corresponding to each pixel one by one, in this way ILI9341 fills each color to each pixel.
  - You still must specify the drawing range even though the range covers only one pixel.
  - From the rules we mentioned above, we can conclude that drawing vertical or horizontal lines are faster than diagonal lines.
- **Printing text on ILI9341**
  - In our API, each character is 5×7 but each character is printed to size 6×8 (its right side and below are left blank), so as to separate from next character. For example, the character "A":



  - The font size represents the dot size. For example, if the font size is 2, each dot in the character is a 2×2 rectangle
- **Screen rotation**
  - ILI9341 provides 0, 90, 180, 270 degrees screen rotation.
  - If the original width is 240 and original height is 320, when the screen rotates 90 degrees, the width becomes 320 and the height becomes 240.

SPI – Show PM2.5 Concentration On ILI9341 TFT LCD

If you are not familiar with SPI, please read Introduction to SPI first.

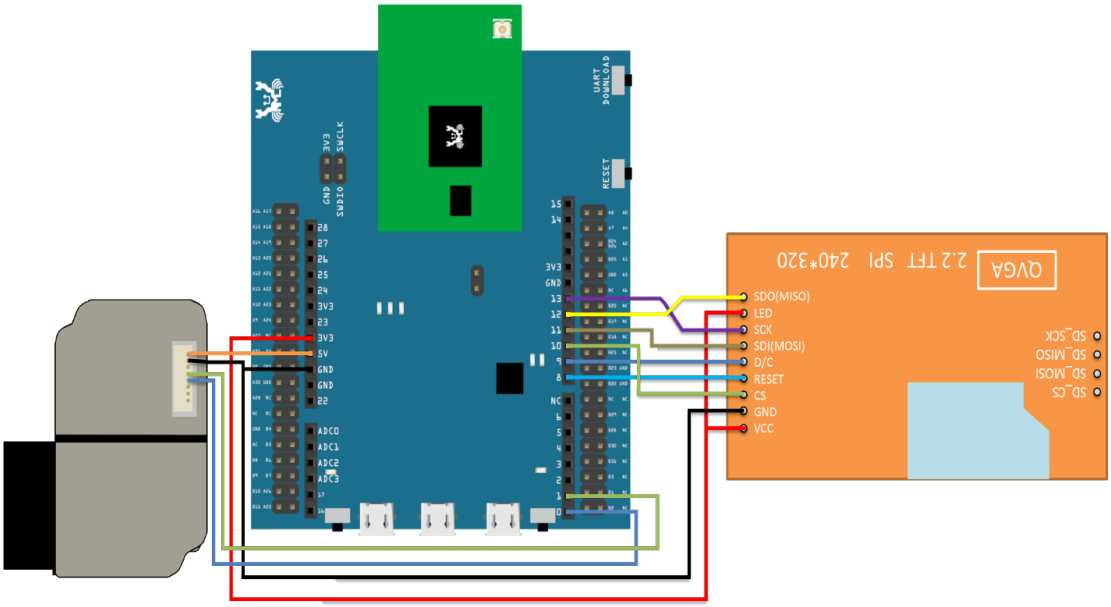
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1
- Plantower PMS3003 or PMS5003 x 1

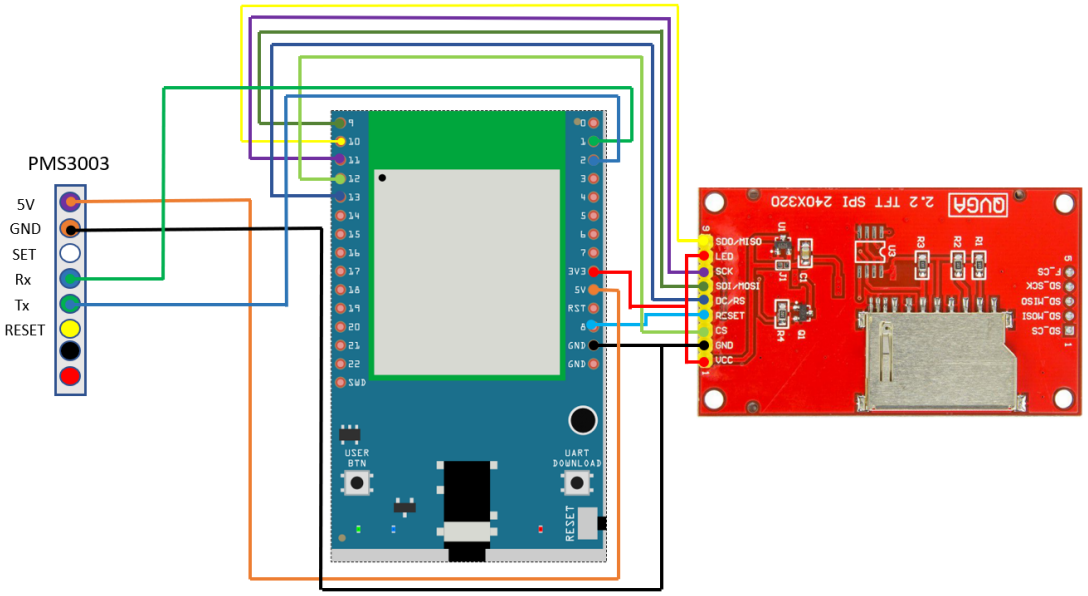
Example

This example extends previous PM2.5 example to show the PM2.5 concentration on the LCD.

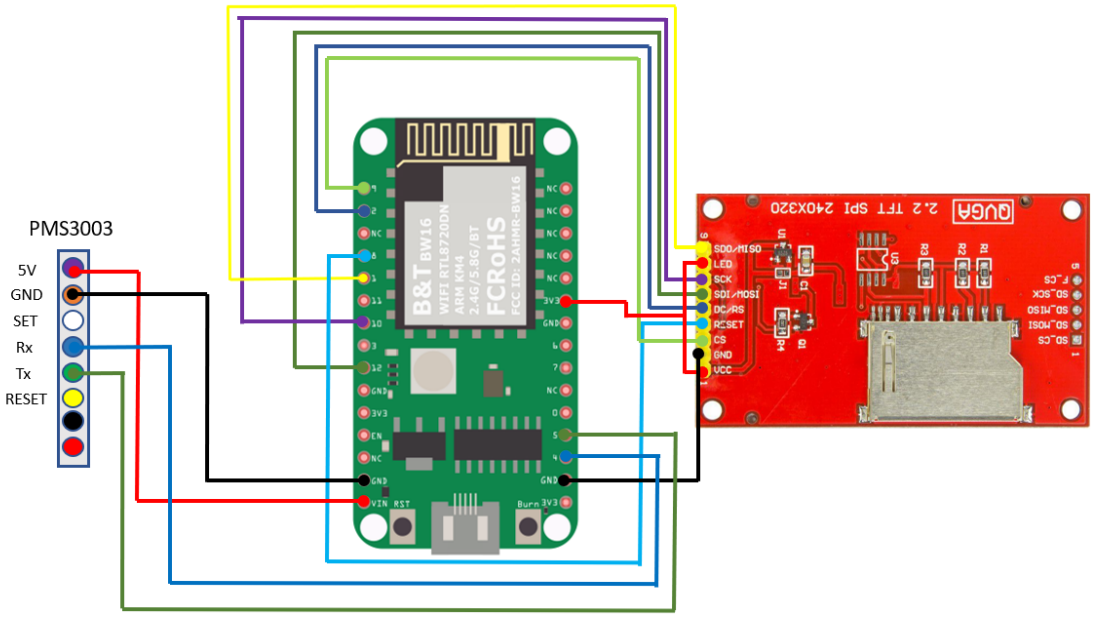
AMB21 / AMB22 and QVGA TFT LCD Wiring Diagram:
(Note: PMS3003/PMS5003 sensor requires 5V voltage)



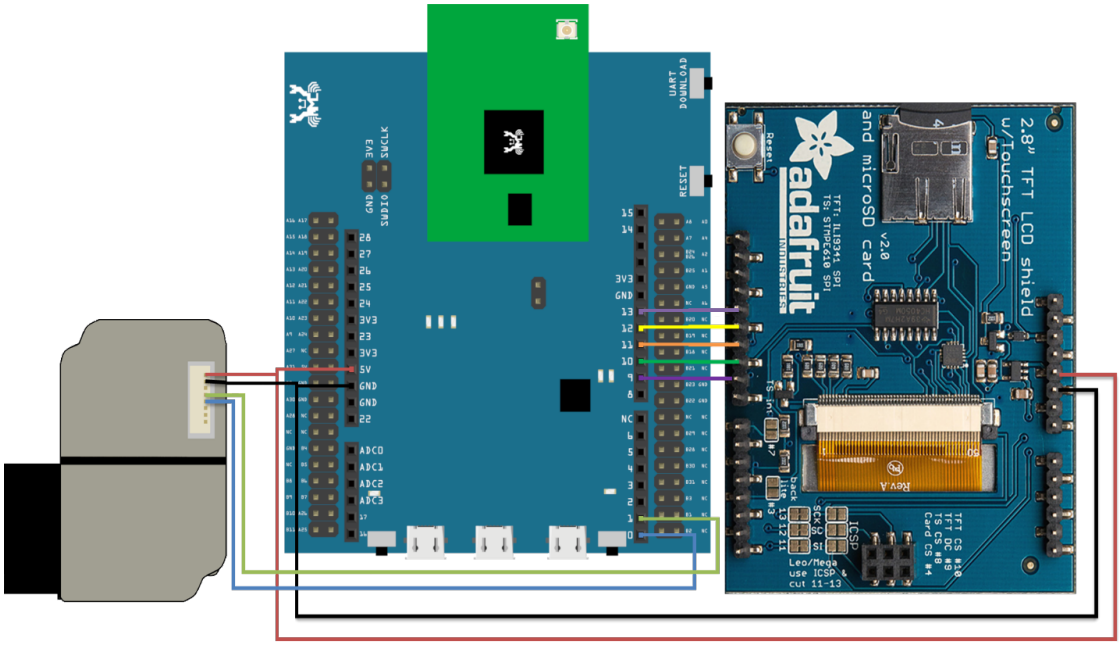
AMB23 and QVGA TFT LCD Wiring Diagram:



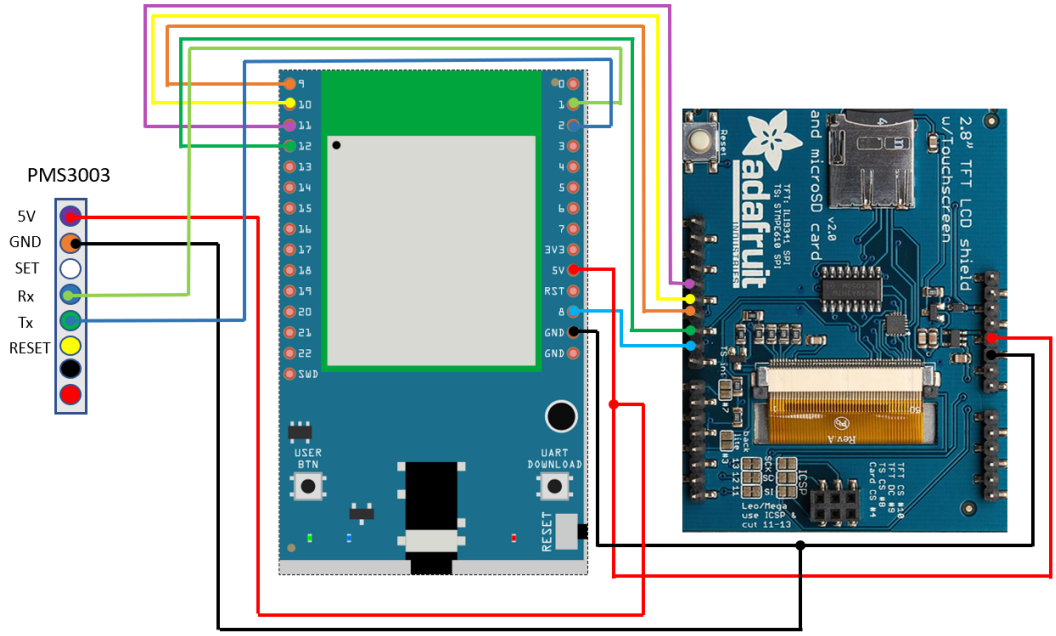
BW16 and QVGA TFT LCD Wiring Diagram:



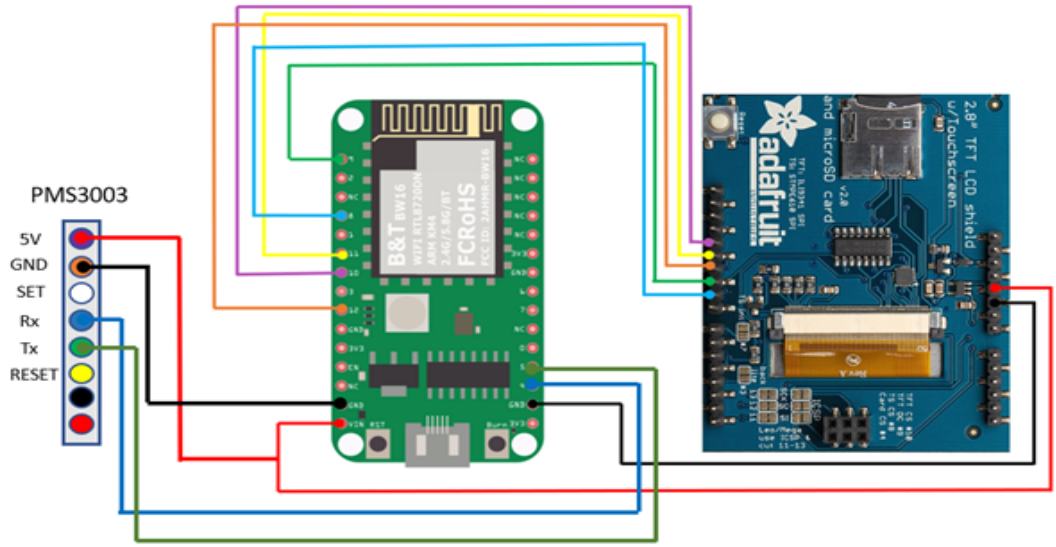
AMB21 / AMB22 and Adafruit 2.8" TFT LCD Wiring Diagram:



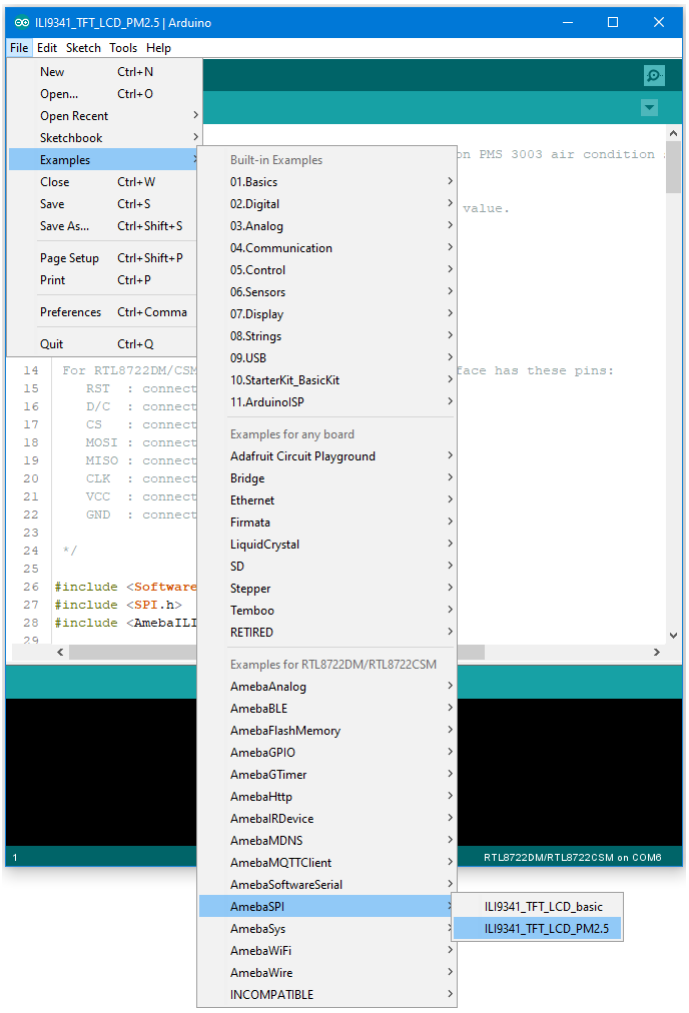
AMB23 and and Adafruit 2.8" TFT LCD Wiring Diagram:



BW16 and and Adafruit 2.8" TFT LCD Wiring Diagram:

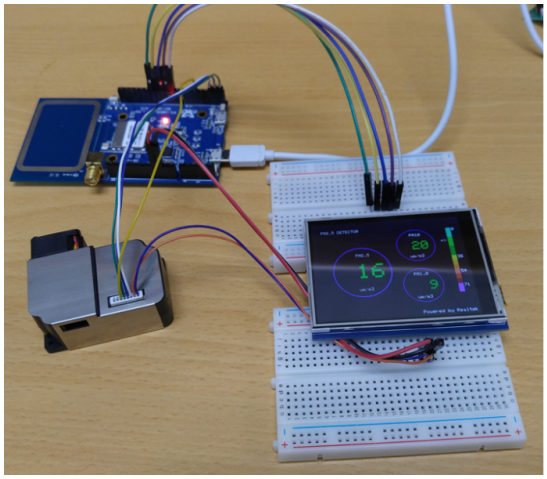
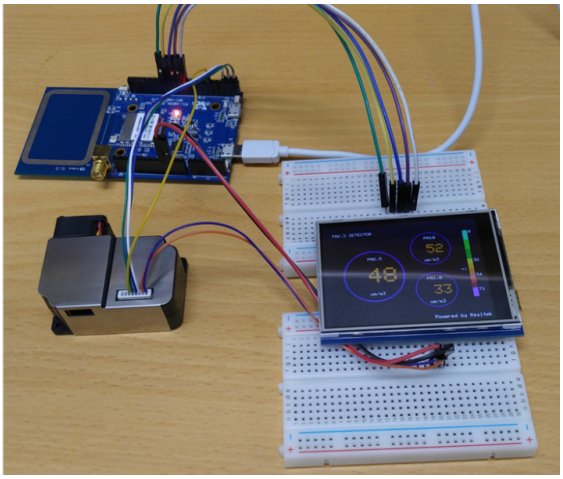


Open the example, "`Files` -> "`Examples`" -> "`AmebaSPI`" -> "`PM25_on_ILI9341_TFT_LCD`"



Compile and upload to Ameba, then press the reset button.

Then you can see the concentration value of PM1.0, PM2.5 and PM10 on the LCD.

Code Reference

In this example, first rotate the screen by 90 degrees, and draw the static components such as the circles, the measuring scale, and the title text. After the concentration value is detected, it is printed inside the circle.

TensorFlow Lite - Hello World

Materials

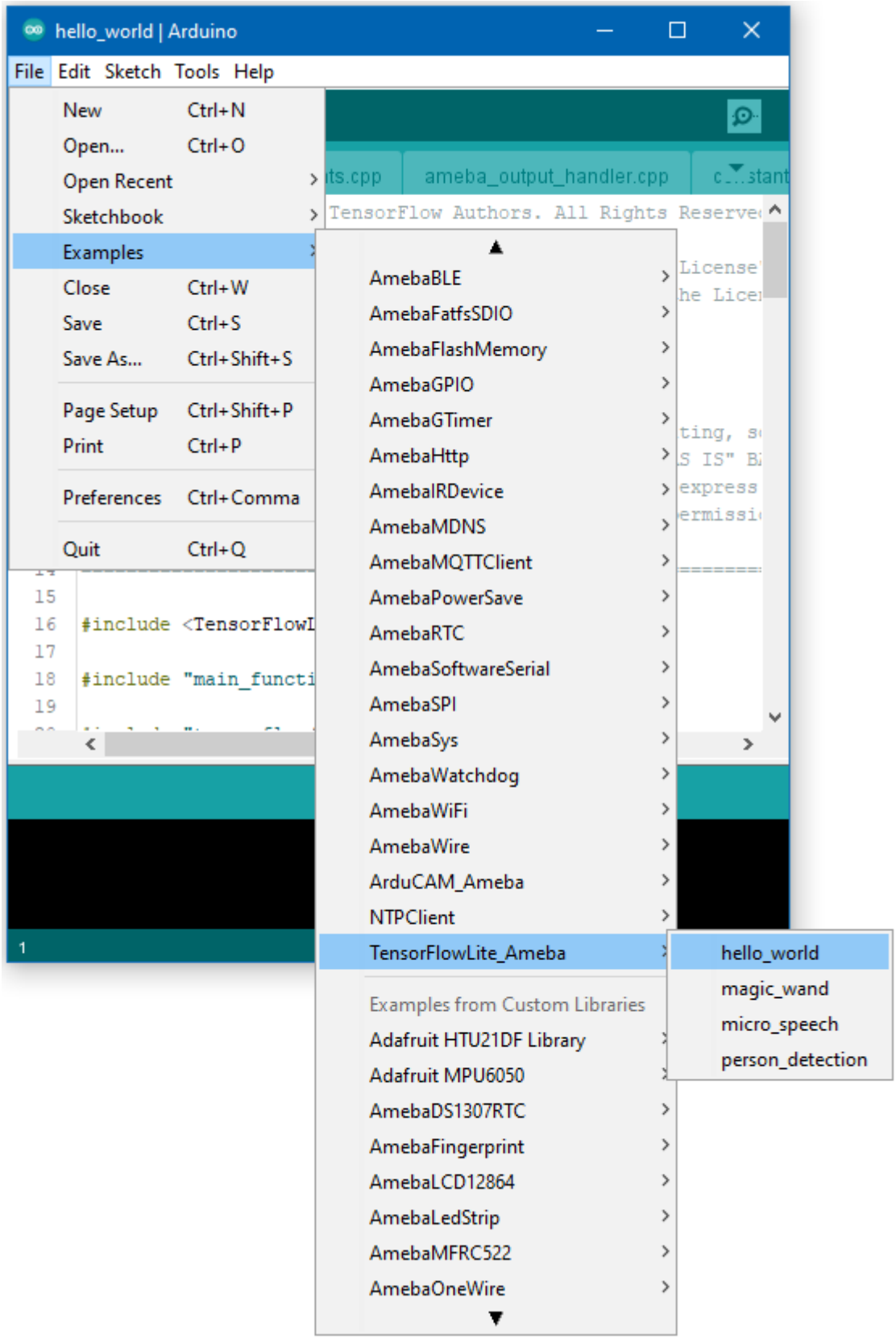
- AmebaD [AMB21 / AMB22 / AMB23] x 1
- LED x 1

Example

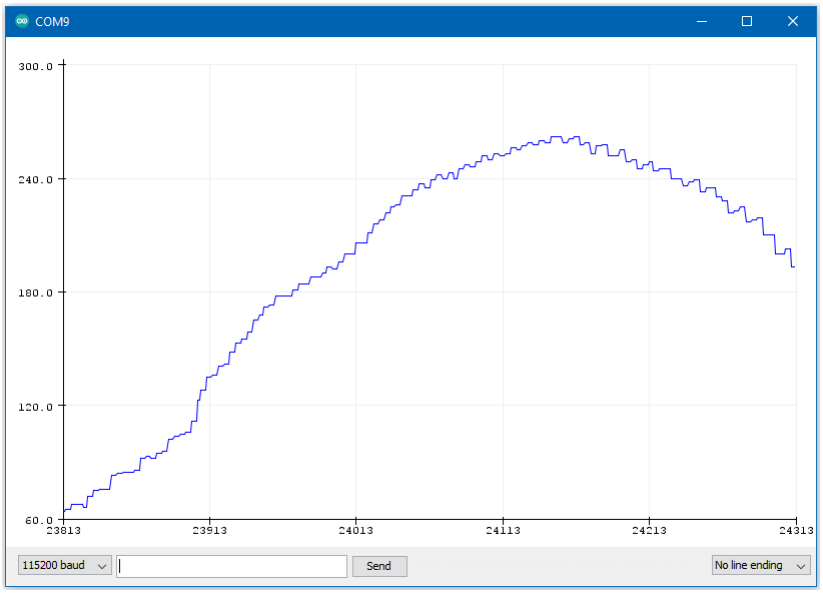
Procedure

Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries. Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it. Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

Open the example, `"Files" -> "Examples" -> “TensorFlowLite_Ameba” -> “hello_world”`.


hello_world | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
ameba_output_handler.cpp
TensorFlow Authors. All Rights Reserve
AmebaBLE
AmebaFatfsSDIO
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
hello_world
magic_wand
micro_speech
person_detection
#include <TensorFlowL
#include "main_functi

Upload the code and press the reset button on Ameba once the upload is finished.
Connect the LED to digital pin 10 and ground, ensuring that the polarity is correct. You should see the LED fade in and out rapidly.
In the Arduino serial plotter, you can see the output value of the Tensorflow model plotted as a graph, it should resemble a sine wave.



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Magic Wand

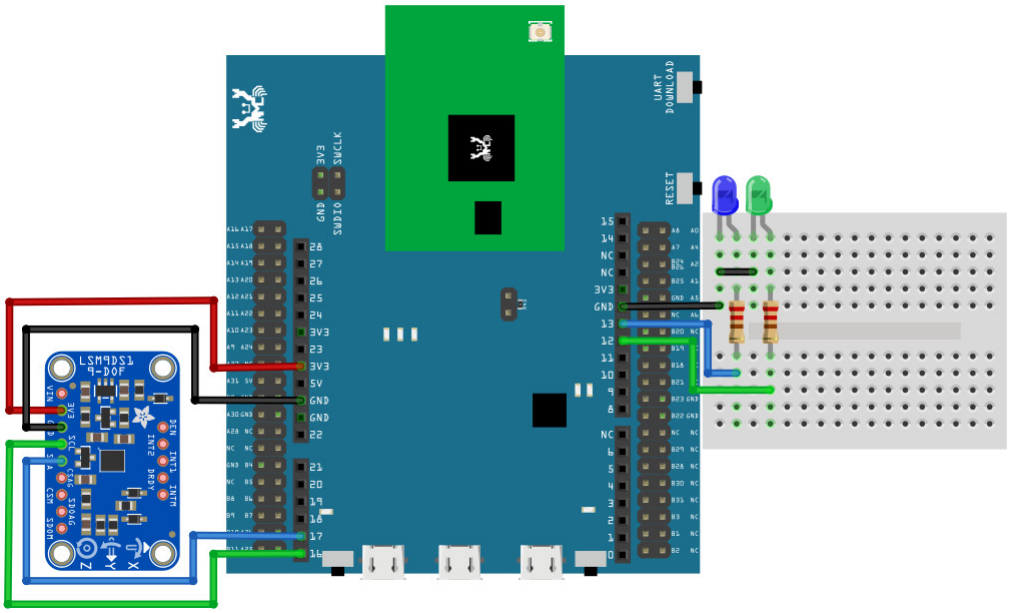
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Adafruit LSM9DS1 accelerometer
- LED x 2

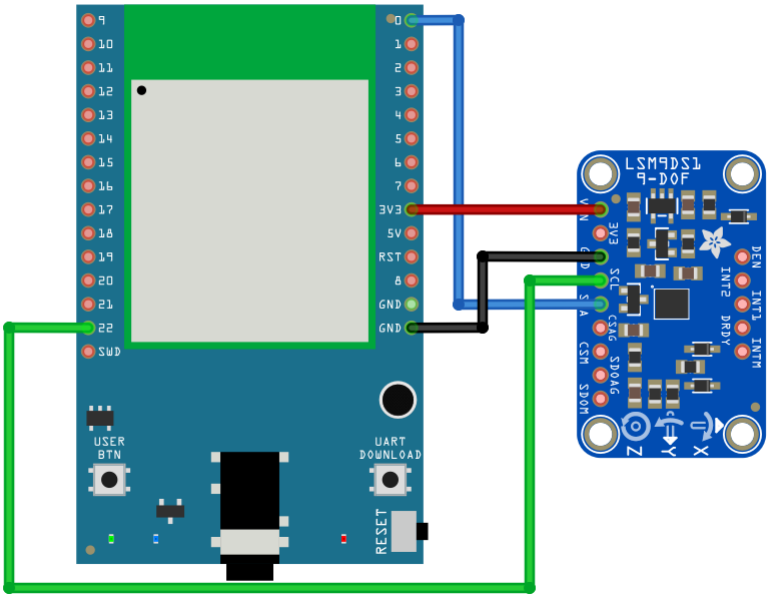
Example

Procedure

AMB21* / AMB22 Wiring Diagram:
Connect the accelerometer and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
For RTL8722DM MINI, we will use the onboard LEDs on the board itself.



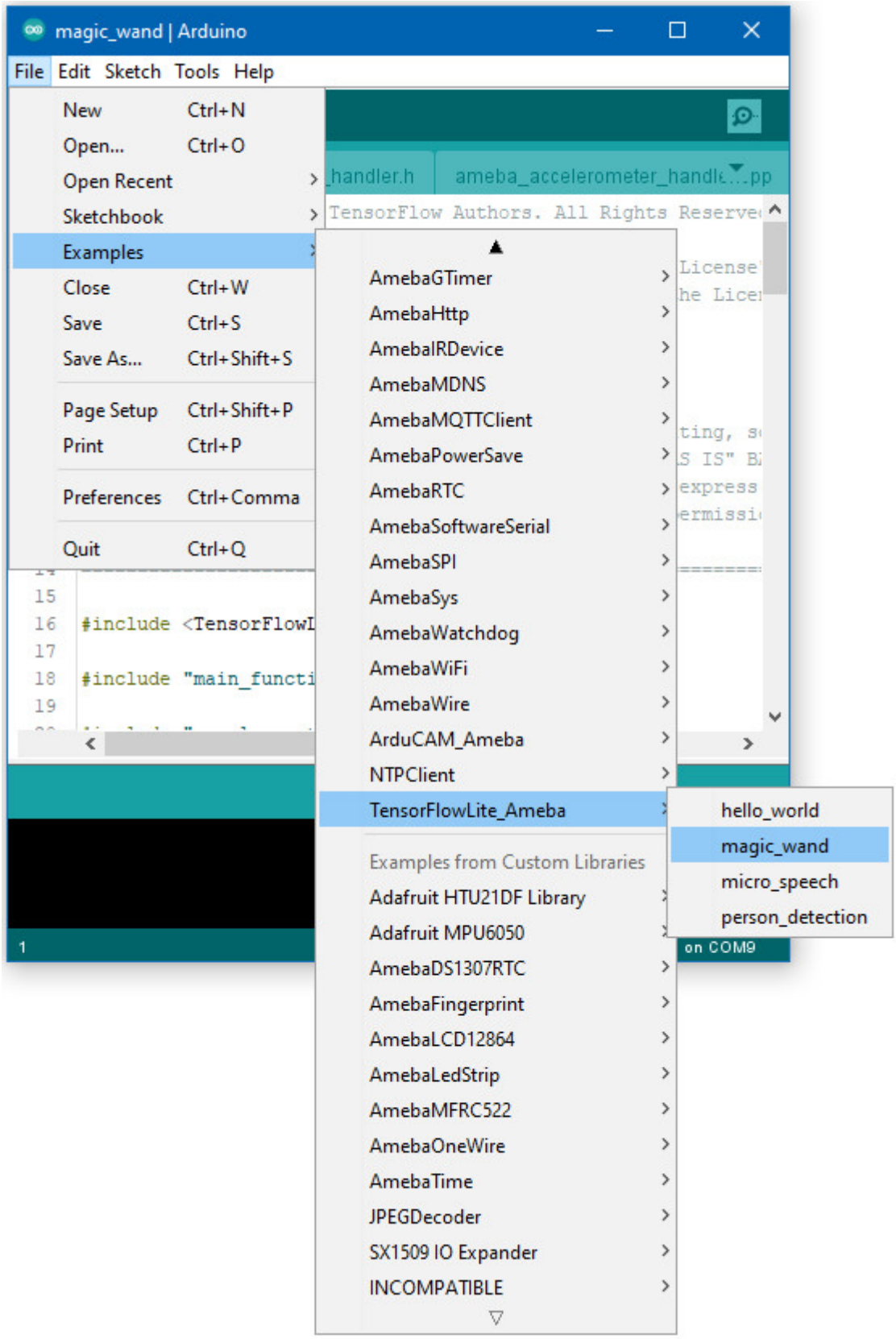
Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it.

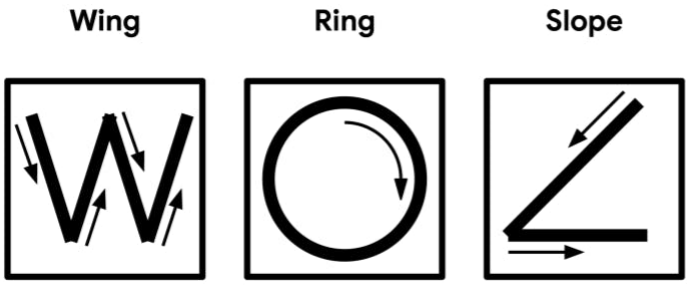
Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

In the Arduino IDE library manager, install the Arduino_LSM9DS1 library. This example has been tested with version 1.1.0 of the LSM9DS1 library.

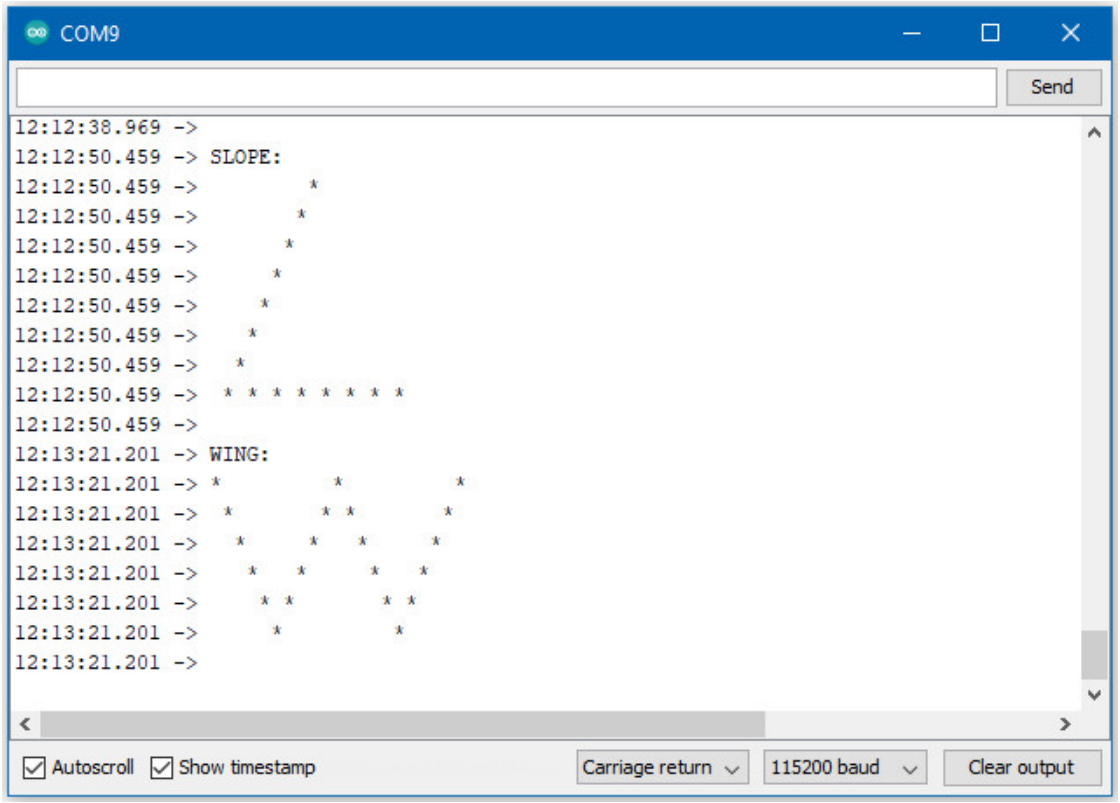
Open the example, "Files" -> "Examples" -> "TensorFlowLite_Ameba" -> "magic_wand".


magic_wand | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
AmebaTime
JPEGDecoder
SX1509 IO Expander
INCOMPATIBLE
hello_world
magic_wand
micro_speech
person_detection
TensorFlow Authors. All Rights Reserve
#include <TensorFlowL
#include "main_functi
on COM9

Upload the code and press the reset button on Ameba once the upload is finished.
Holding the accelerometer steady, with the positive x-axis pointing to the right and the positive z-axis pointing upwards, move it following the shapes as shown, moving it in a smooth motion over 1 to 2 seconds, avoiding any sharp movements.



If the movement is recognised by the Tensorflow Lite model, you should see the same shape output to the Arduino serial monitor. Different LEDs will light up corresponding to different recognized gestures.
Note that the wing shape is easy to achieve, while the slope and ring shapes tend to be harder to get right.



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Micro Speech

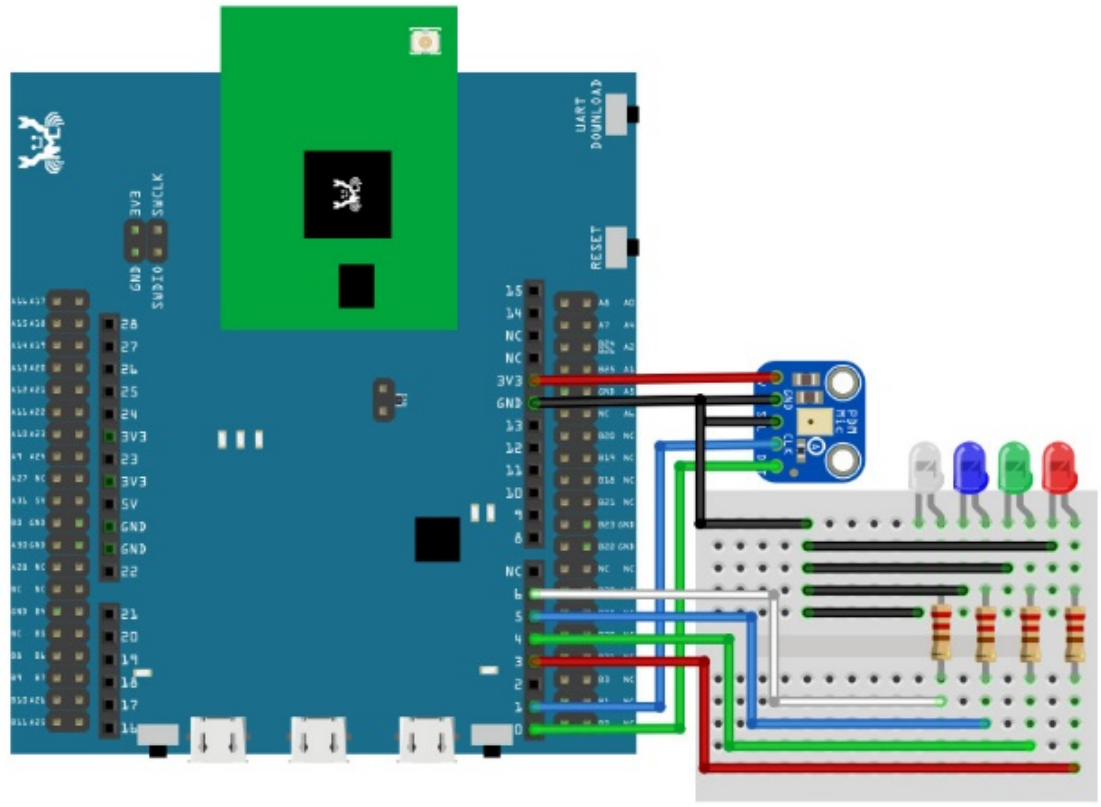
Preparation

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Adafruit PDM MEMS microphone
- LED x 4

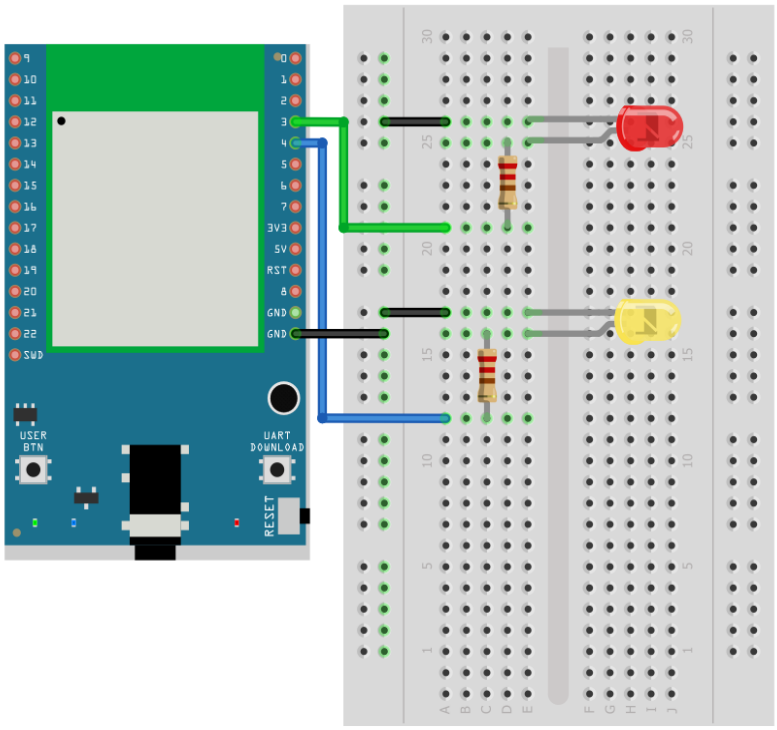
Example

Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the microphone and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:
As RTL8722DM MINI have a built in microphone on the board, there is no need for any external microphone. For the LEDs, we will only connect two LEDs and then use the two onboard LEDs (Blue and Green).

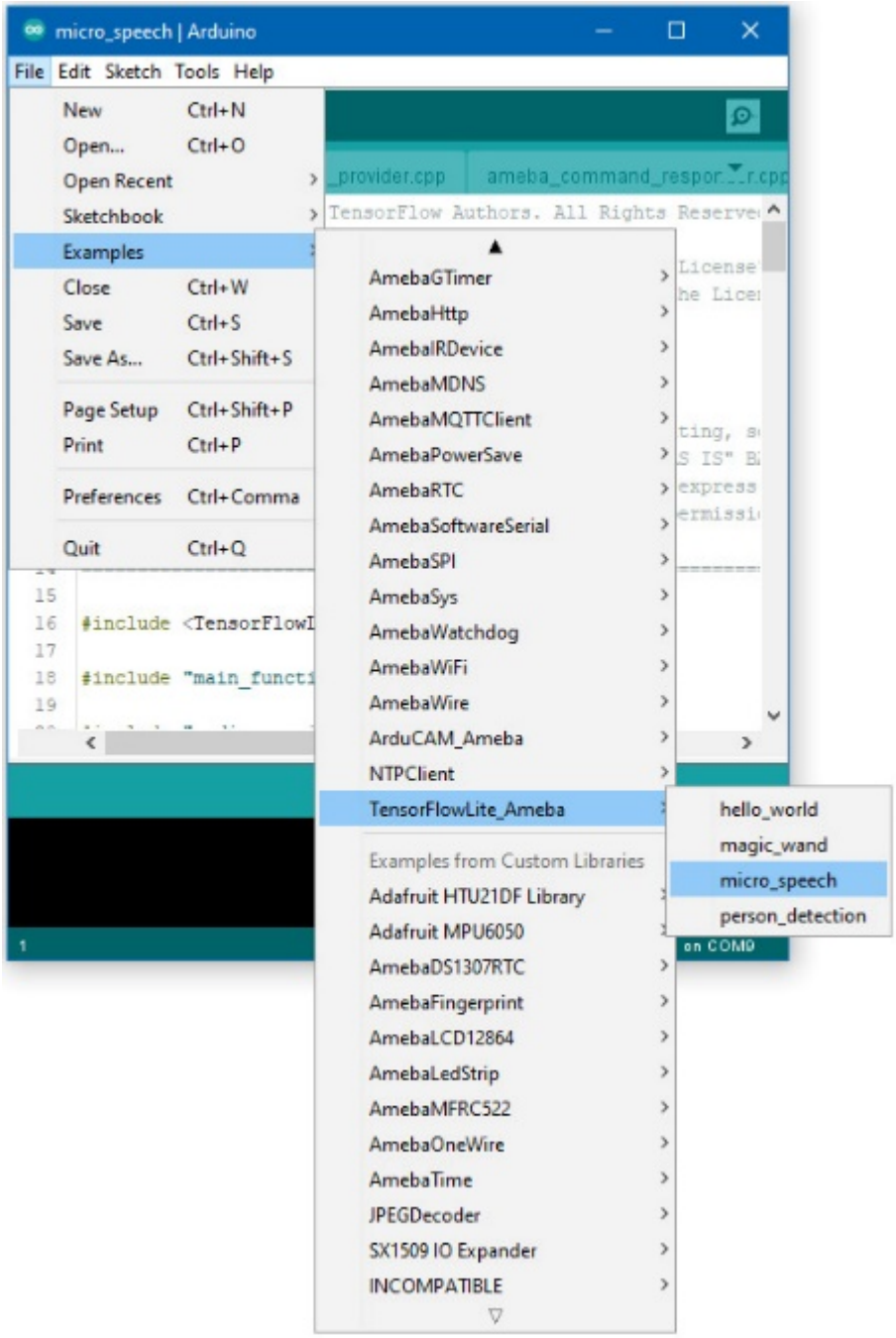


Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it.

Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

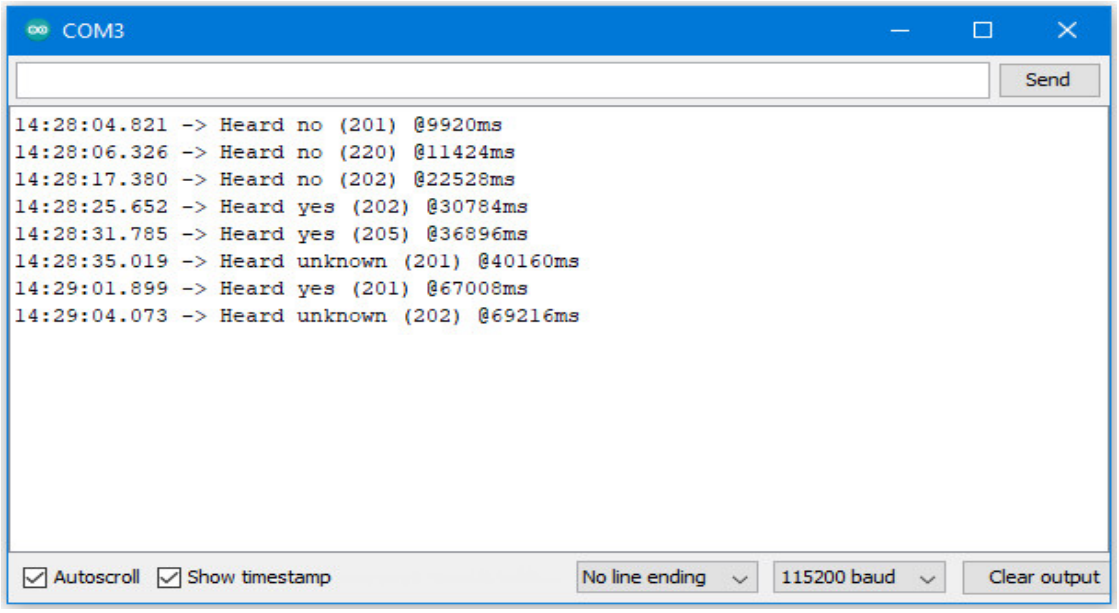
Open the example, `"Files" -> "Examples" -> “TensorFlowLite_Ameba” -> “micro_speech”`.



Upload the code and press the reset button on Ameba once the upload is finished.

Once it is running, you should see one of the LEDs flashing, indicating that it is processing audio. Saying the word “yes” will cause the green LED to light up. Saying the word “no” will cause the red LED to light up. If the word is not recognized, the blue LED will to light up.

The inference results are also output to the Arduino serial monitor, which appear as follows:



If you are having trouble in getting the words recognized, here are some tips:

- Ensure that your surroundings are quiet with minimal noise.
- Experiment with varying the distance of the microphone, starting with it at an arm's length.
- Experiment with different tones and volume when saying the words.
- Depending on how you pronounce the words, the characteristics of the microphone used, getting one keyword recognized may be easier than the other.

Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

TensorFlow Lite - Person Detection

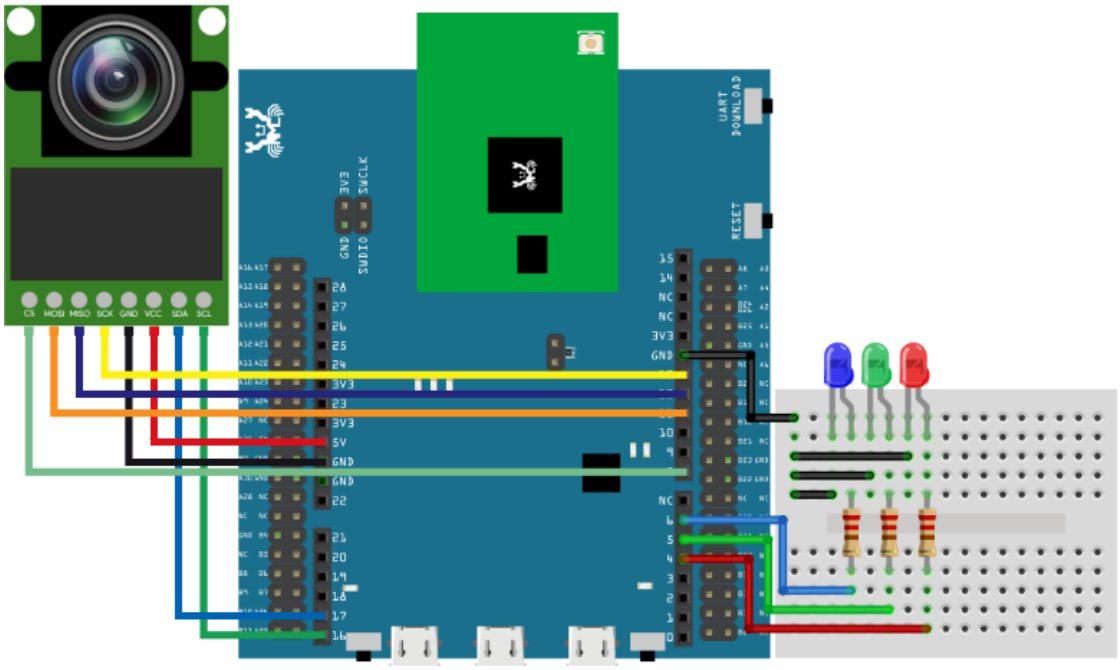
Materials

- AmebaD [AMB21 / AMB22 / AMB23] x 1
- Arducam Mini 2MP Plus OV2640 SPI Camera Module x 1
- LED x 3

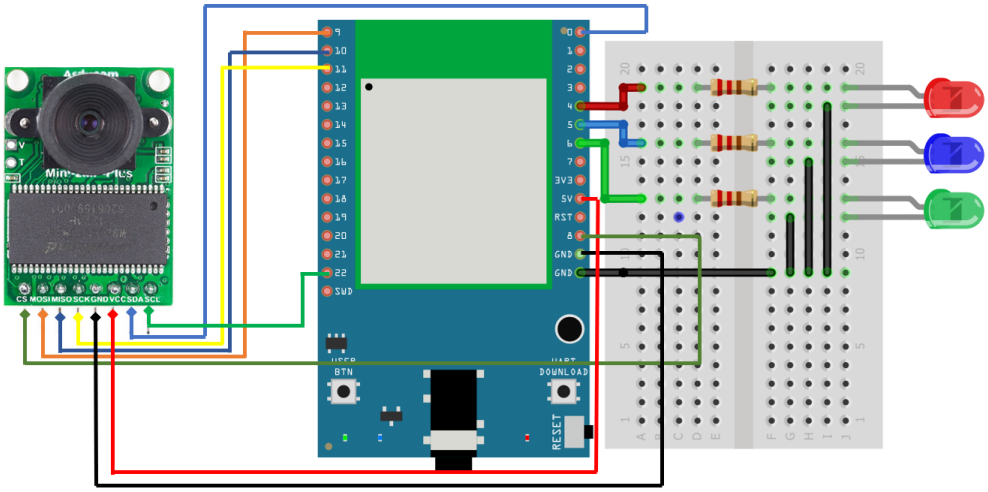
Example

Procedure

AMB21 / AMB22 Wiring Diagram:
Connect the camera and LEDs to the RTL8722 board following the diagram.



AMB23 Wiring Diagram:



Download the Ameba customized version of TensorFlow Lite for Microcontrollers library at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries.

Follow the instructions at https://www.arduino.cc/en/guide/libraries to install it. Ensure that the patch files found at https://github.com/ambiot/ambd_arduino/tree/master/Ameba_misc/ are also installed.

You will also need to install the Ameba_ArduCAM library, found together with the TensorFlow Lite library.

In the Arduino IDE library manager, install the JPEGDecoder library. This example has been tested with version 1.8.0 of the JPEGDecoder library.

Once the library has installed, you will need to configure it to disable some optional components that are not compatible

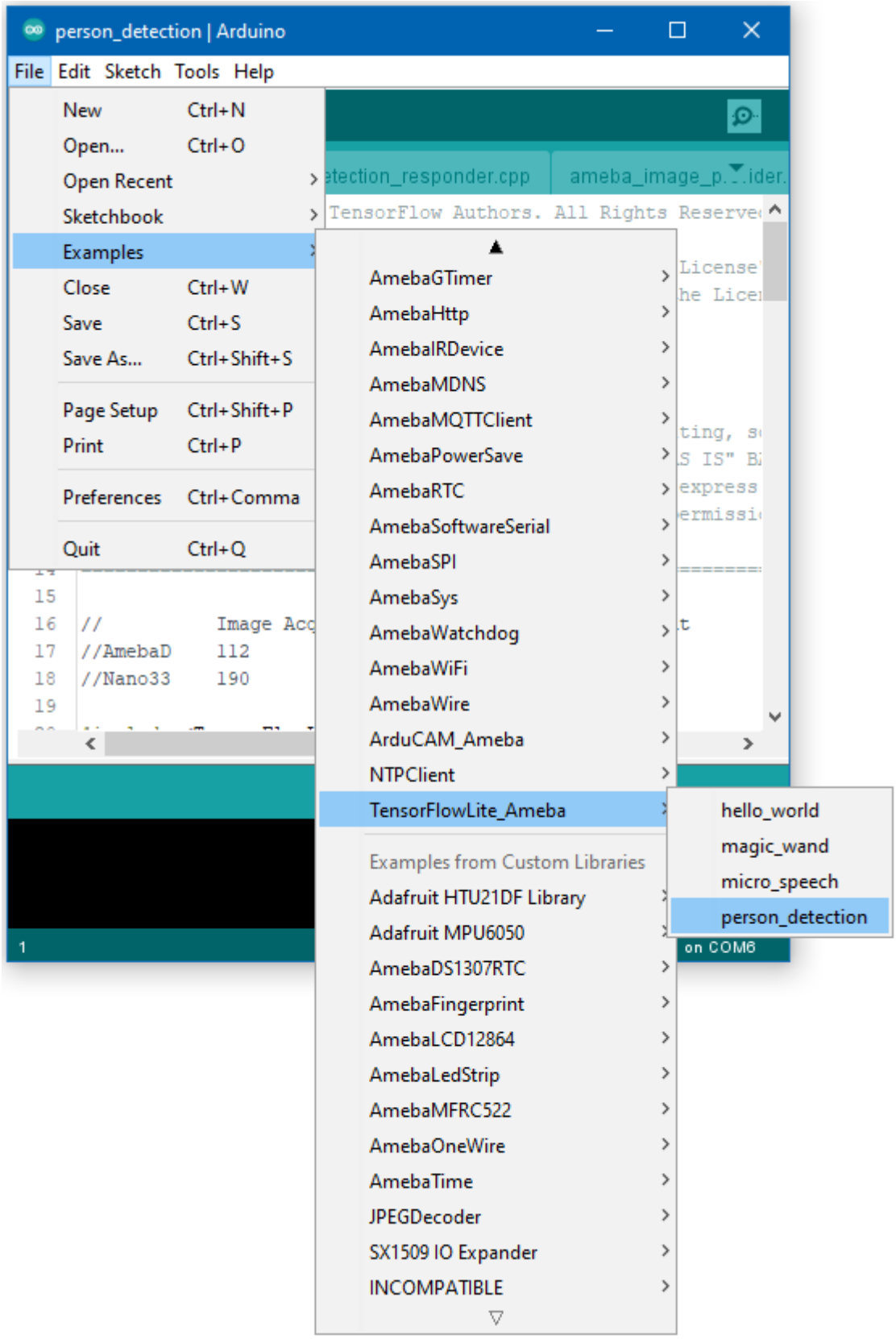
with the RTL8722DM. Open the following file:

```
Arduino/libraries/JPEGDecoder/src/User_Config.h
```

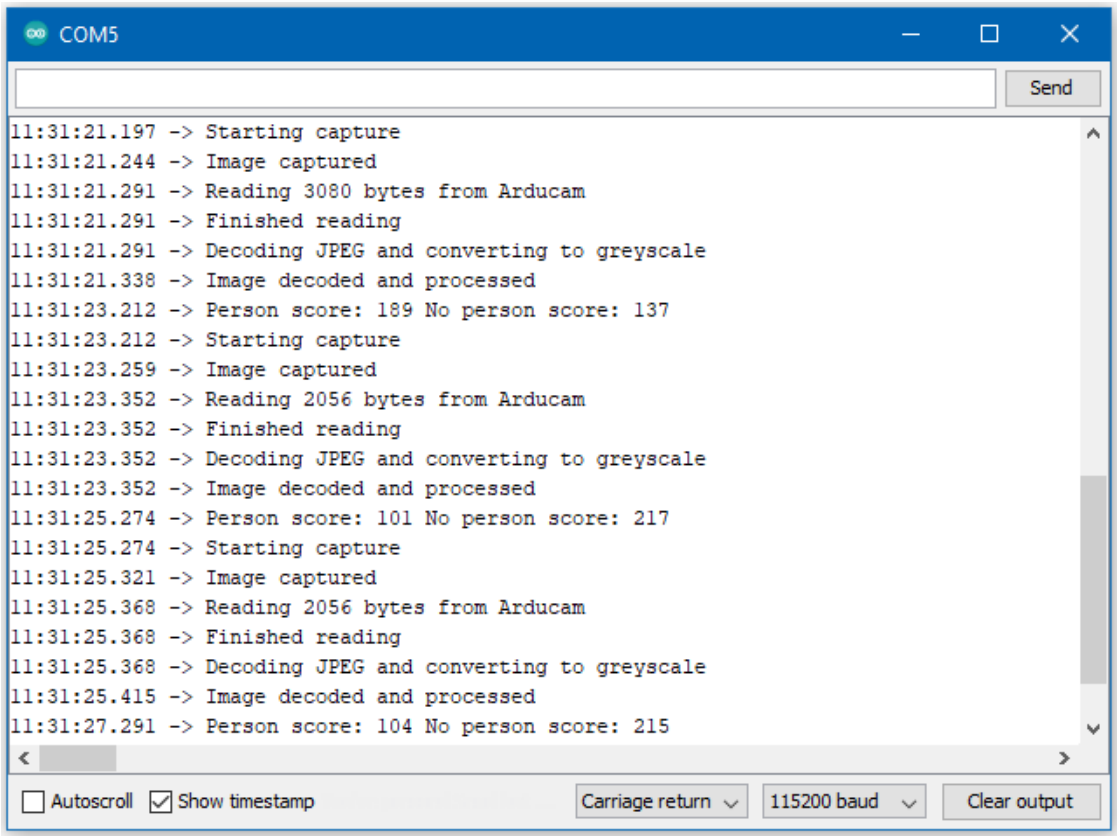
Make sure that both `#define LOAD_SD_LIBRARY` and `#define LOAD_SDFAT_LIBRARY` are commented out, as shown in this excerpt from the file:

```
//#define LOAD_SD_LIBRARY // Default SD Card library
//#define LOAD_SDFAT_LIBRARY // Use SdFat library instead, so SD Card SPI can be bit␣
↪bashed
```

Open the example, `"Files" -> "Examples" -> "TensorFlowLite_Ameba" -> "person_detection"`.


person_detection | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaPowerSave
AmebaRTC
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWatchdog
AmebaWiFi
AmebaWire
ArduCAM_Ameba
NTPClient
TensorFlowLite_Ameba
Examples from Custom Libraries
Adafruit HTU21DF Library
Adafruit MPU6050
AmebaDS1307RTC
AmebaFingerprint
AmebaLCD12864
AmebaLedStrip
AmebaMFRC522
AmebaOneWire
AmebaTime
JPEGDecoder
SX1509 IO Expander
INCOMPATIBLE
hello_world
magic_wand
micro_speech
person_detection

Upload the code and press the reset button on Ameba once the upload is finished.
Once it is running, you should see the blue LED flashing once every few seconds, indicating that it has finished processing an image. The red LED will light up if it determines that there is no person in the previous image captured, and the green LED will light up if it determines that there is a person.
The inference results are also output to the Arduino serial monitor, which appear as follows:



Code Reference

More information on TensorFlow Lite for Microcontrollers can be found at: https://www.tensorflow.org/lite/microcontrollers

UART - Communicate with PC over USB to Serial Module

Introduction of UART

UART uses two wire, one for transmitting and the other one for receiving, so the data transmission is bidirectional. The communication uses a predefined frequency (baud rate) to transmit data. In Arduino, UART is called “Serial”. There is only one hardware UART on Arduino Uno and it is primarily used to read the log and messages printed by Arduino (so it is also called “Log UART”). If we use the hardware UART for other purposes, the Log UART does not have resources to function. To provide more UART connections, it is possible to use a GPIO pin to simulate the behavior of UART with a software approach, this is called Software Serial. Ameba is equipped with several hardware UART ports, but it is also compatible with the Software Serial library.

Materials

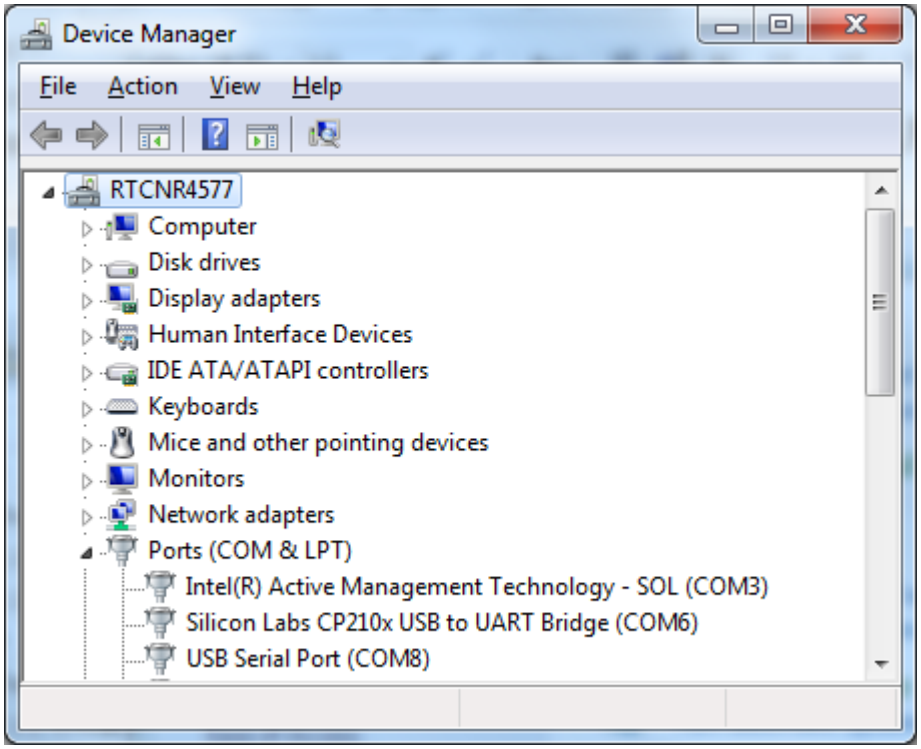
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

Example

In this example, we use UART to connect USB to TTL adapter to Ameba.
USB to TTL adapter sends data to Ameba, the data would be returned by Ameba, and showed on the screen.

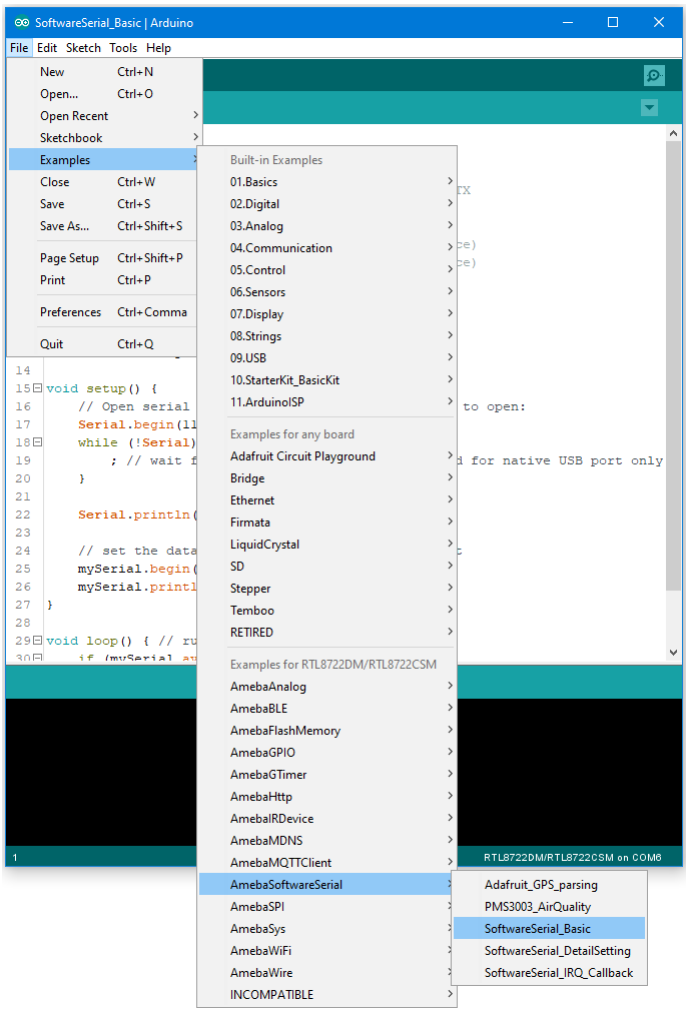
- **Install USB to TTL Adapter**

USB to TTL adapter converts USB to serial interface. Normally, there are at least 4 pins on the adapter, that is 3V3 (or 5V), GND, TX and RX. Generally, installing the driver for the USB to TTL adapter would be required before using it. If the adapter uses the chip of FTDI, Windows will search and install the driver automatically, otherwise, you may need to install corresponding driver yourself.
Afterwards, open device manager. You can find corresponding serial port number of the USB to TTL adapter:



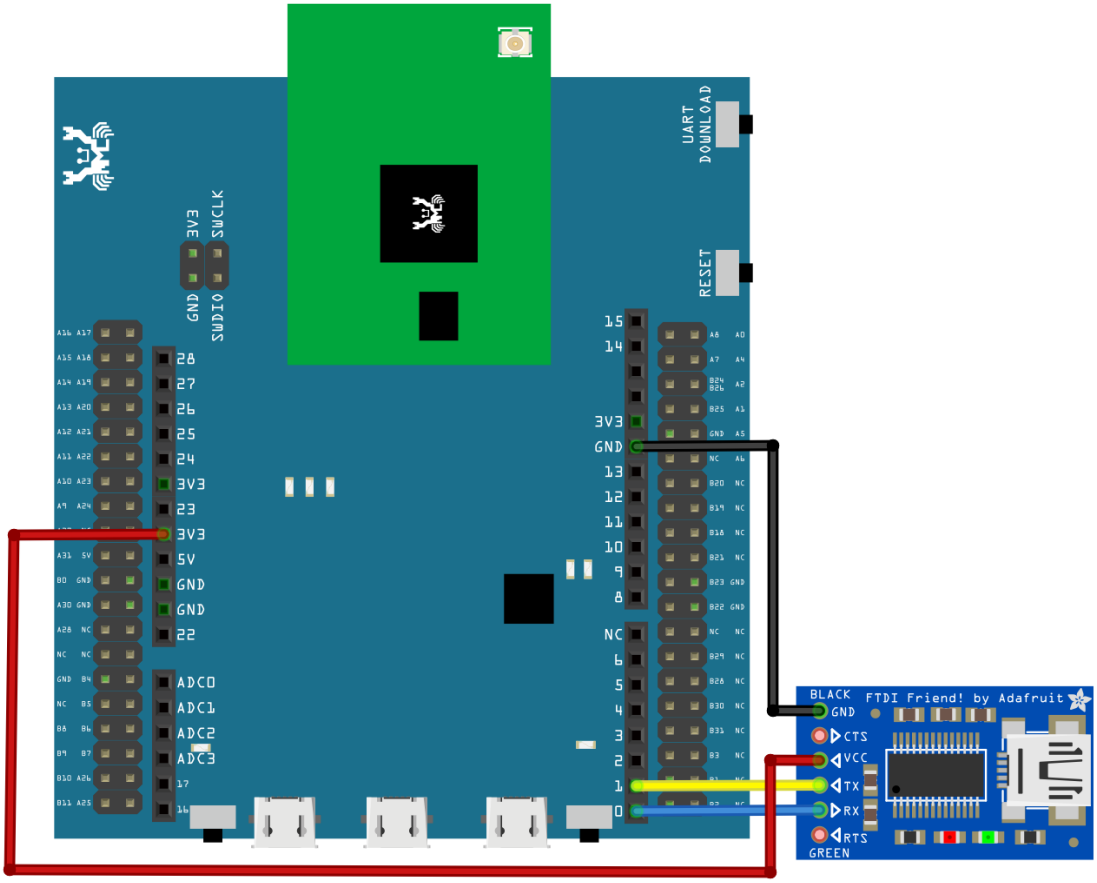
- Executing the Example

Open the “SoftwareSerialExample” example in “`File`” -> “`Examples`” -> “`AmebaSoftwareSerial`” -> “`SoftwareSerial_Basic`”:

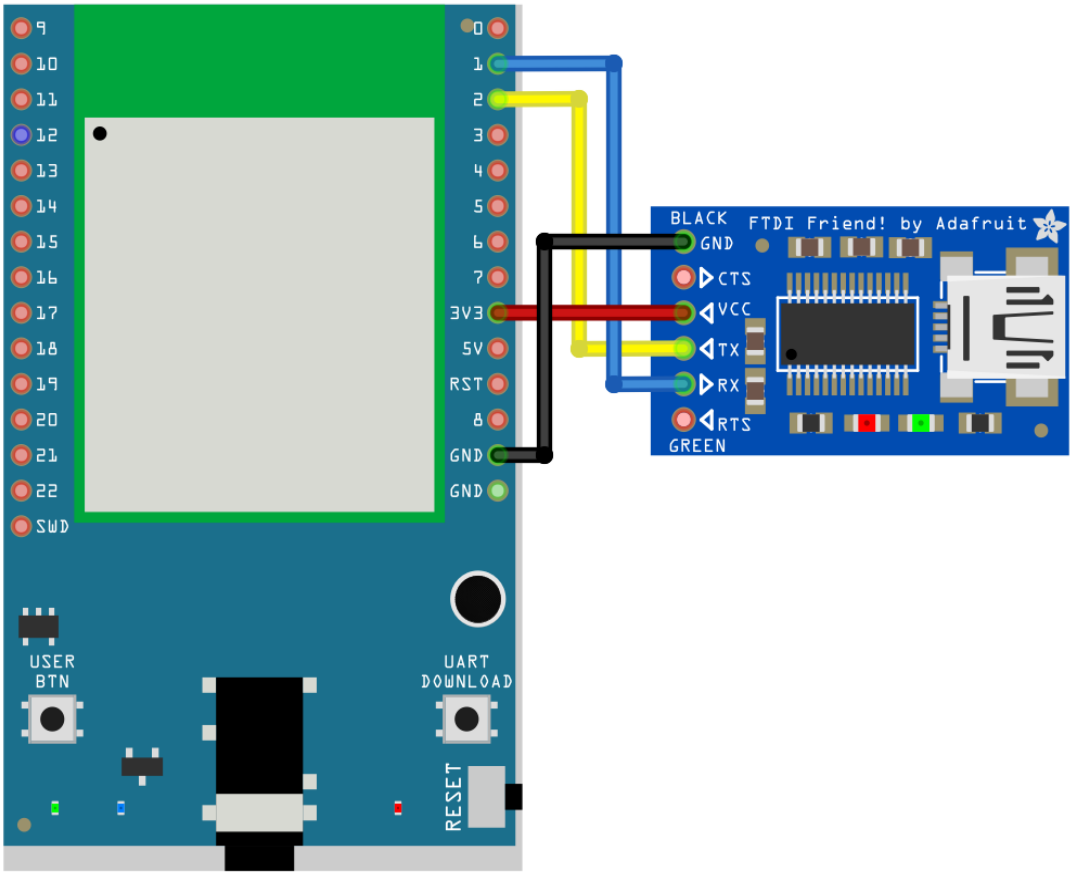


Connect the wire as the following diagrams show. The TX pin of USB to TTL adapter is connected to the RX of Ameba, and the RX pin of USB to TTL adapter is connected to the TX of Ameba.

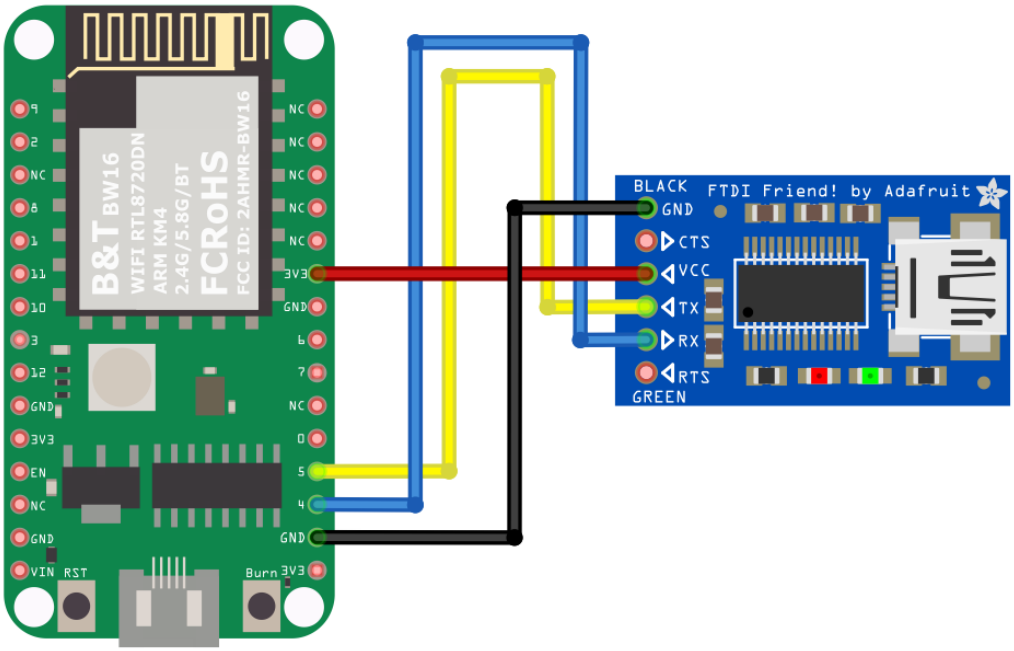
AMB21 / AMB22 Wiring Diagram:



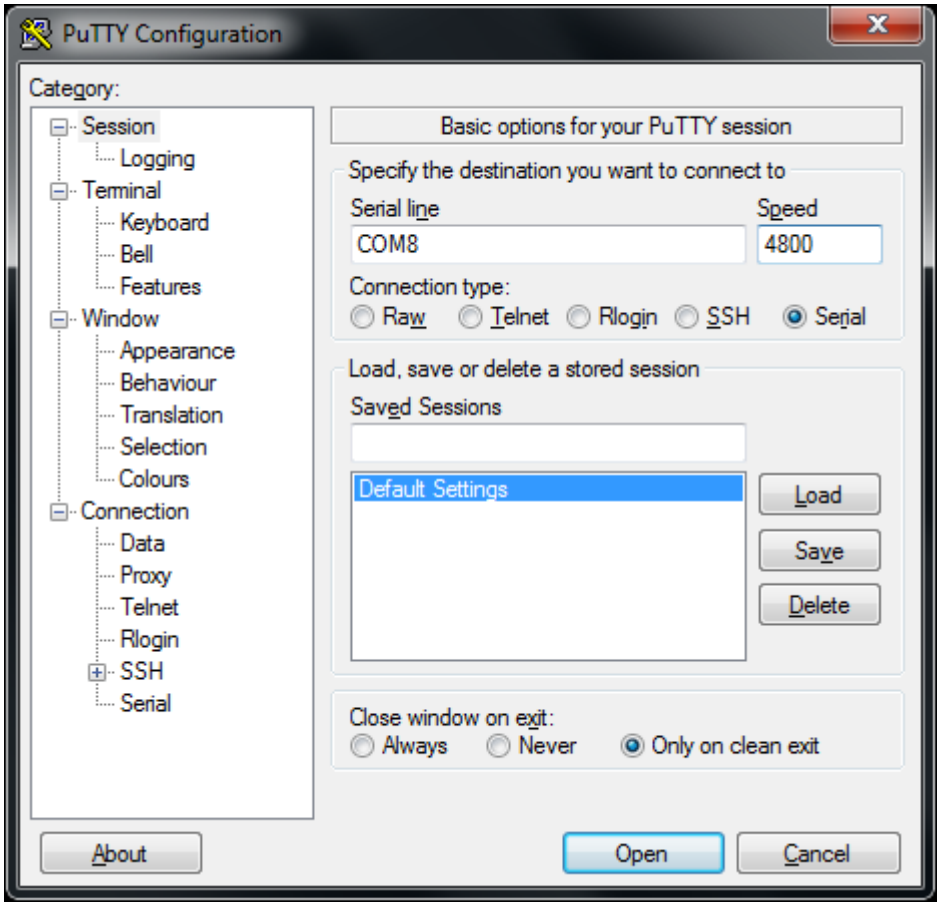
AMB23 Wiring Diagram:



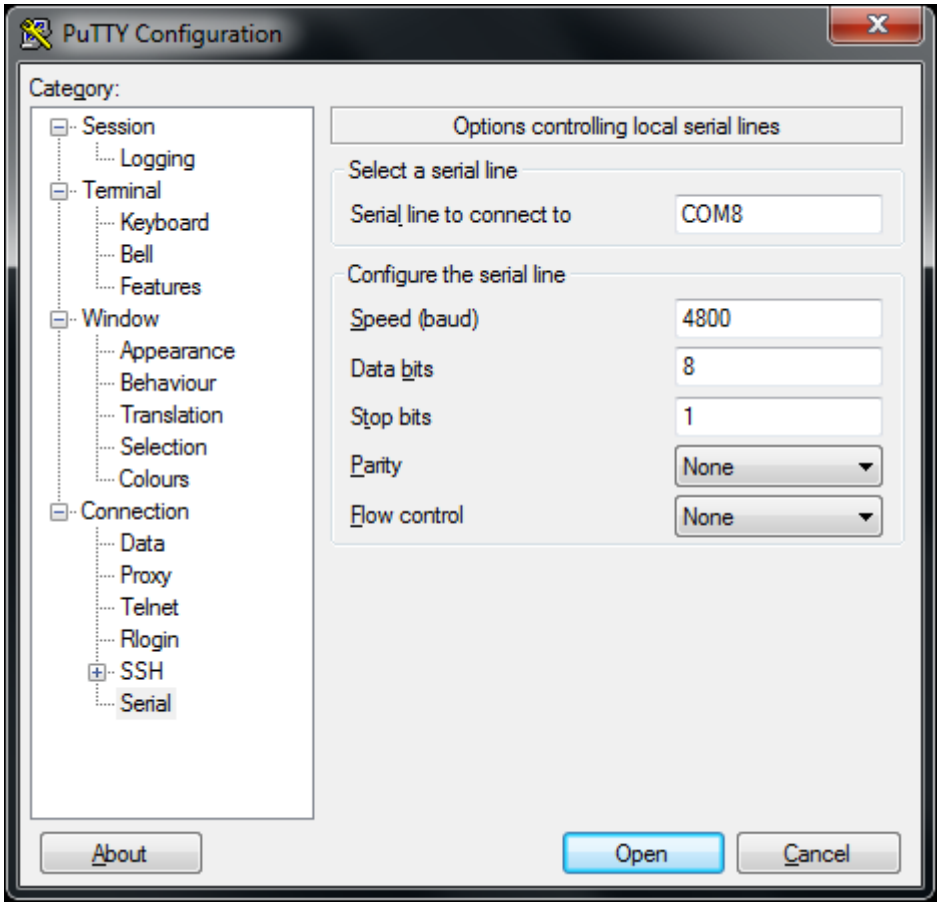
BW16 Wiring Diagram:



Next, open a serial port terminal, such as Putty or Tera Term. (Putty is used in this example). Open the Putty window, choose "Serial" in connection type, and specify the port number of the USB to TTL adapter (e.g. COM8). In the speed field, fill in the baud rate of this connection. Note that both sides of the connection should use the same baud rate. In this example we set baud rate 4800.



Next, select “Serial” on the left side. Set data bits to 8, stop bits to 1, parity to none, and flow control to none.



Then click Open and press the reset button on Ameba. You can see the "Hello, world?" message appears in Putty. If characters are typed into Putty, the input characters would be sent to Serial RX of Ameba by TX of USB to TTL Adapter, and returned by Serial TX of Ameba. Finally, RX of USB to TTL Adapter receives the returned characters and prints them in Putty. Therefore, if you insert "I am fine", you will get something like this:

Code Reference

First, use `SoftwareSerial:begin(speed)` to set the baud rate for the serial communication:
https://www.arduino.cc/en/Reference/SoftwareSerialBegin

Use `write()` to send data, and use `SoftwareSerial:available()` to get the number of bytes available for reading from a software serial port:

https://www.arduino.cc/en/Reference/SoftwareSerialAvailable
If there are data available to read, use read() to read from serial port.

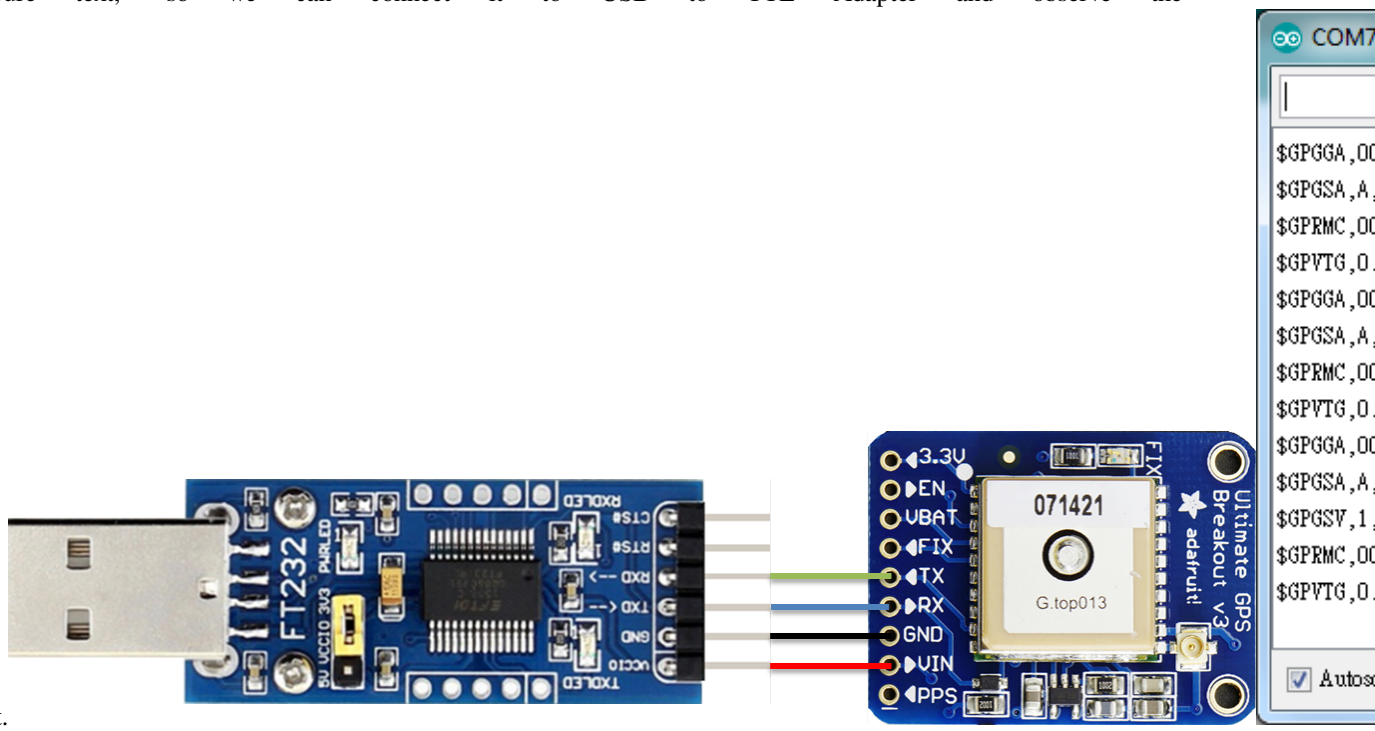
UART - Retrieve GPS Position

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Adafruit Ultimate GPS Breakout x 1 (Refer to official document)

Example

In this example, we use Adafruit Ultimate GPS Breakout. Its data format is pure text, so we can connect it to USB to TTL Adapter and observe the output.



It follows the NMEA sentence format (refer to http://aprs.gids.nl/nmea/) The GPS signal is weak in indoor environment. The status that the GPS signal is not received is called “not fix”. Bring the GPS module outdoors, when the GPS signal is “fix”, you would get message similar to the figure

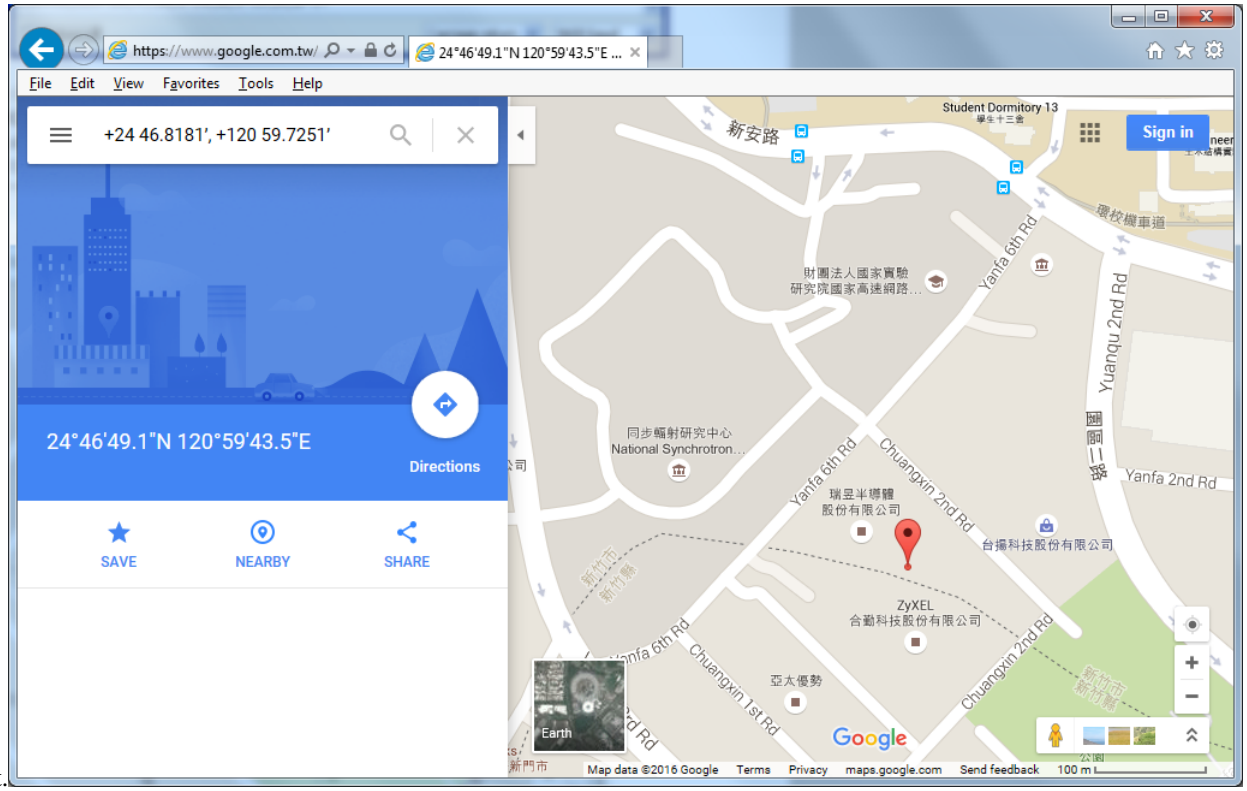
below.

In this example we are only interested in the "$GPRMC (Global Positioning Recommended Minimum Coordinates)":
$GPRMC,032122.000,A,2446.8181,N,12059.7251,E,0.39,78.89,270116,,,A*53
Each field is separated by a comma.

- First field is the GMT time (Greenwich Mean Time), that is 032122.000 in this example. The time format is HH:MM:SS.SSS, i.e., 03:21:22.000. Note that the time zone and the daylight-saving time adjustment should be handled on your own.
- Second field represents the status code
  - V: Void (Invalid)
  - A: Active, meaning the GPS signal is fix.
- The third to sixth fields represent the geolocation

In this example, 2446.8181,N represents 24 degrees 46.8181 minutes north latitude, and 12059.7251,E represents 120 degrees 59.7251 minutes east longitude.

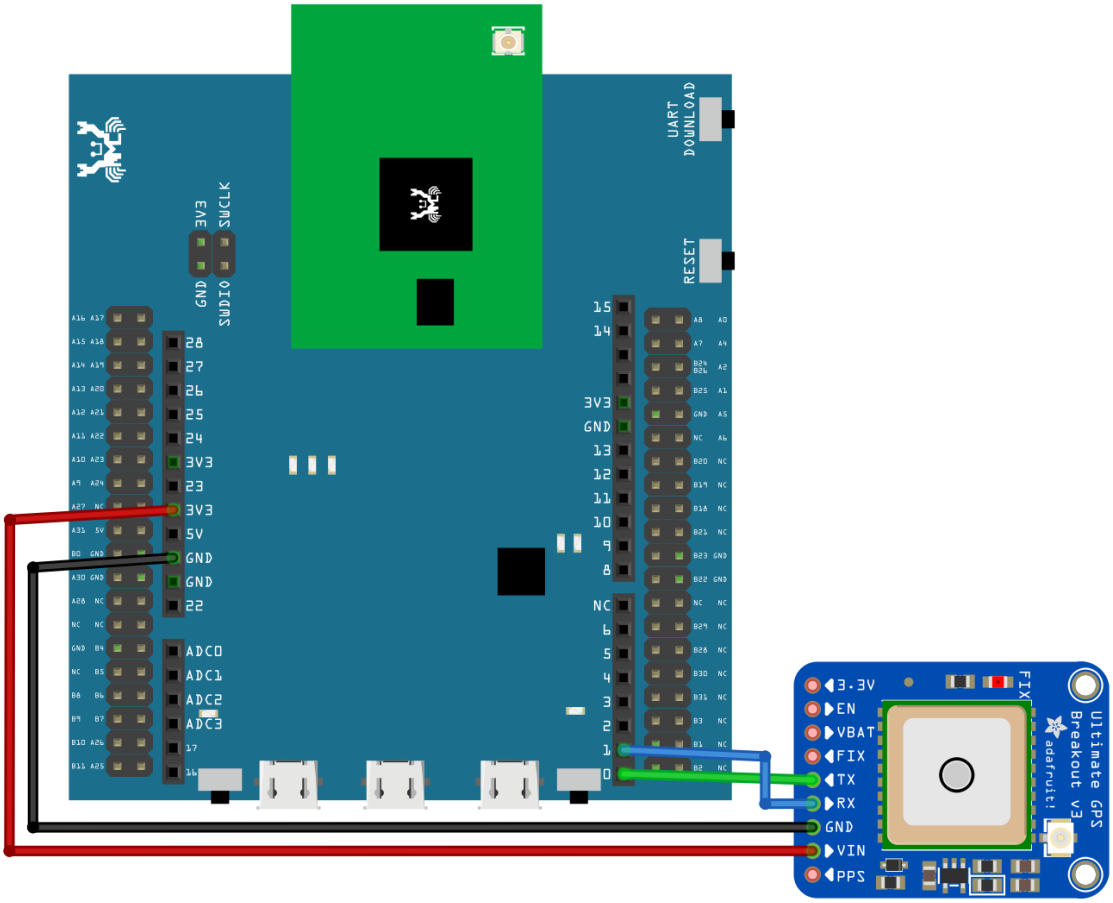
We can search **+24 46.8181', +120 59.7251'** in Google map to check whether the position is



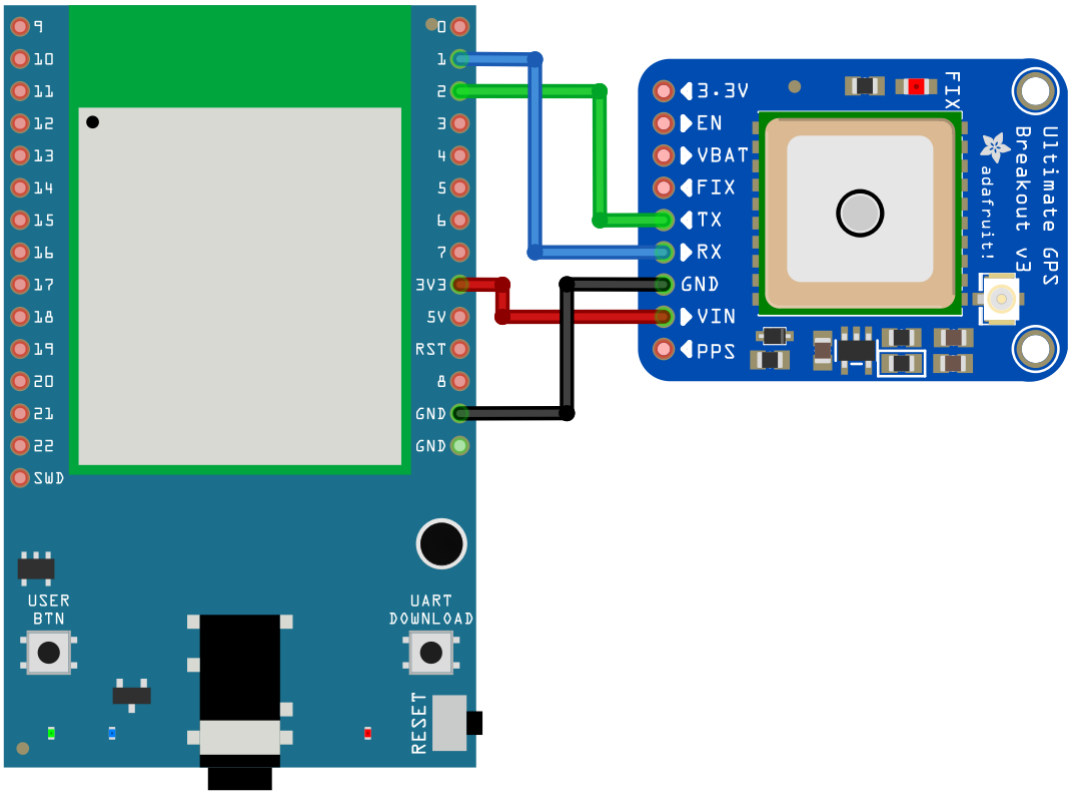
correct.

- The seventh field is relative speed(knot). 1 knot = 1.852km/hr, in this example the relative speed is 0.39 knot.
- The eighth field is the moving angle, which is calculated by its moving orbit.
- The ninth field is the date with format ddMMyy. In this example, “270116” stands for day 27, January, year 2016.
- The last field is checksum. In the example we have *53 as checksum.

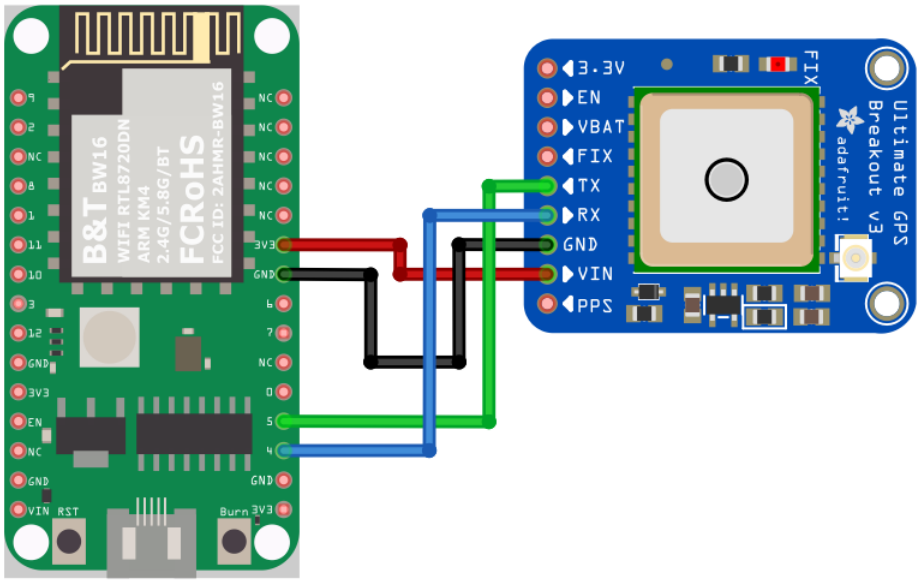
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

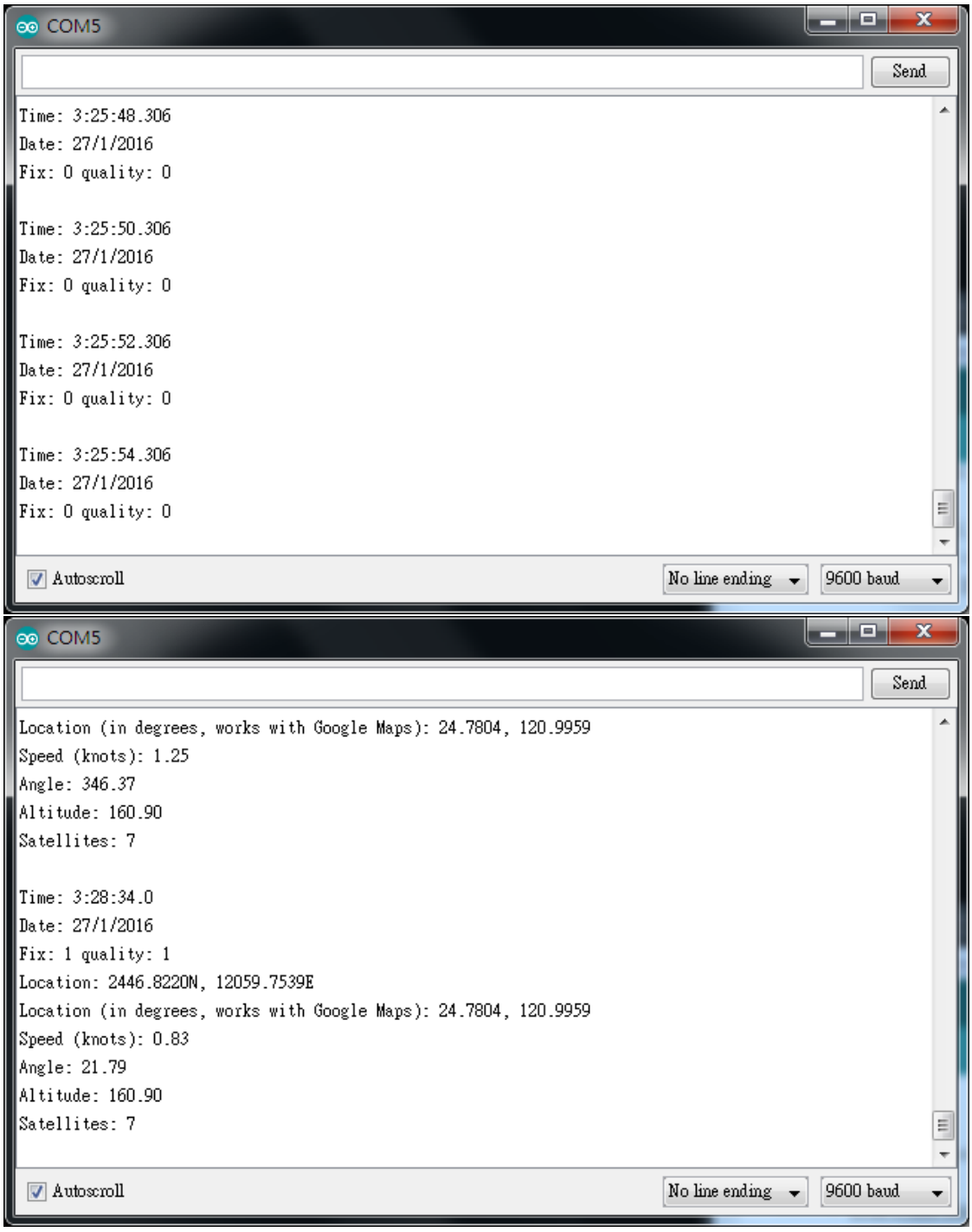


AMB23 Wiring Diagram:



Open the example in "Files" -> "Examples" -> "AmebaSoftwareSerial" -> "Adafruit_GPS_parsing".

Compile and upload to Ameba, then press the reset button.
The result will be output to Serial Monitor:



UART – Set Callback Function For UART Communications

Materials

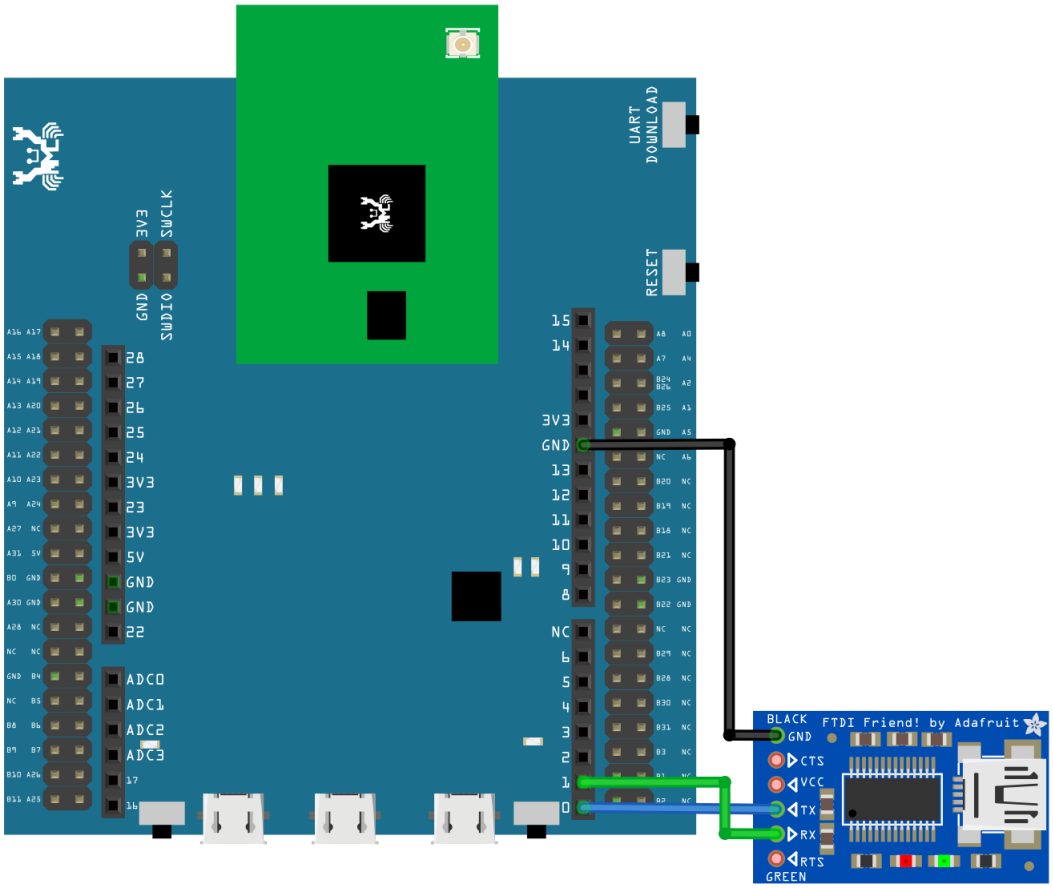
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

Example

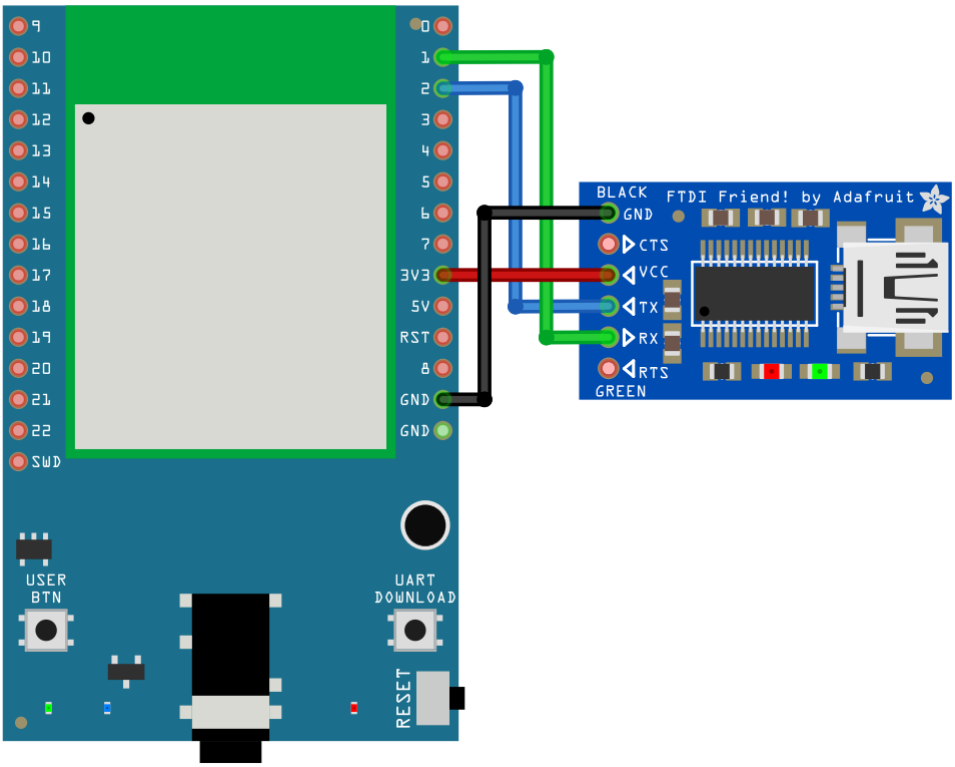
This example shows how to set a callback function for UART communication to process the UART data.

A USB to TTL adapter is required for this example. Ensure that you have the driver installed and connect it to the Ameba board as shown.

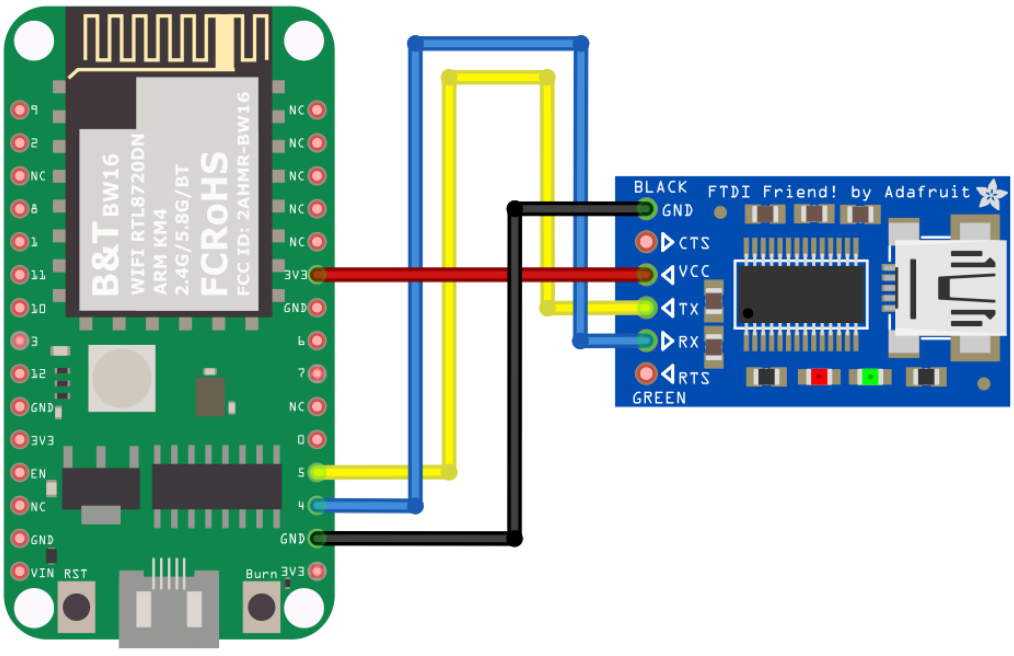
AMB21 / AMB22 Wiring Diagram:



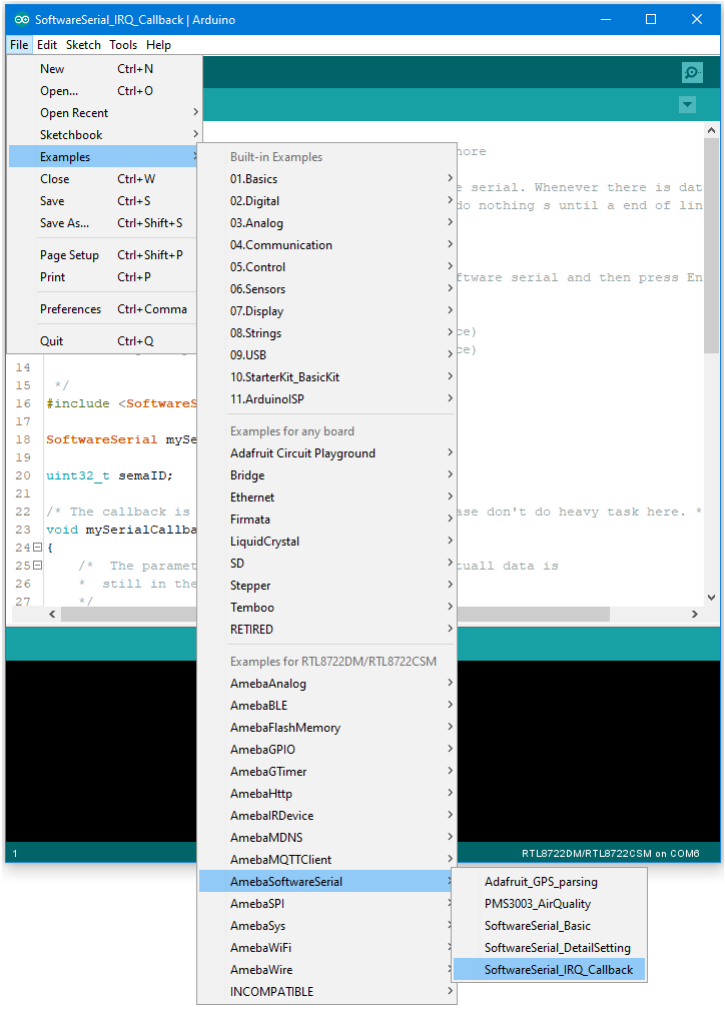
AMB23 Wiring Diagram:



BW16 Wiring Diagram:

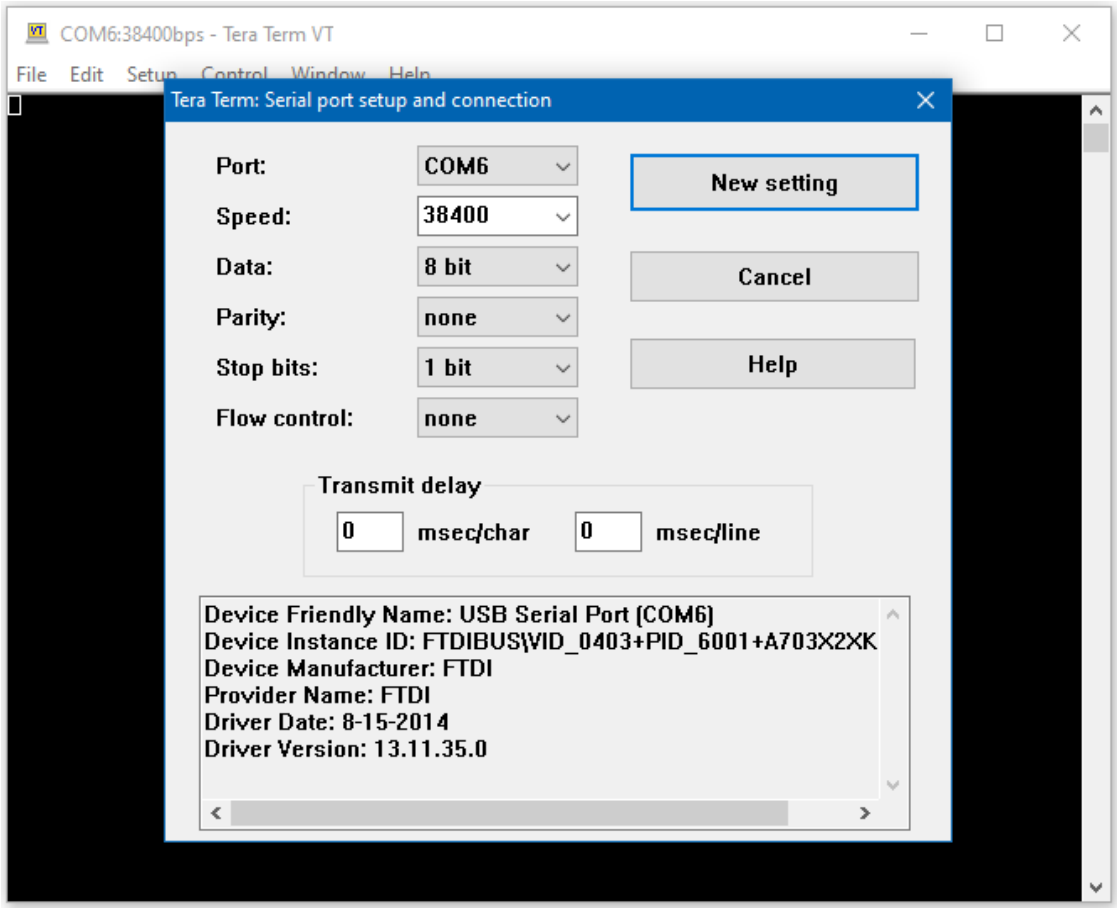


Open the example in "`File`" -> "`Examples`" -> "`AmebaSoftwareSerial`" -> "`SoftwareSerial_Irq_Callback`"

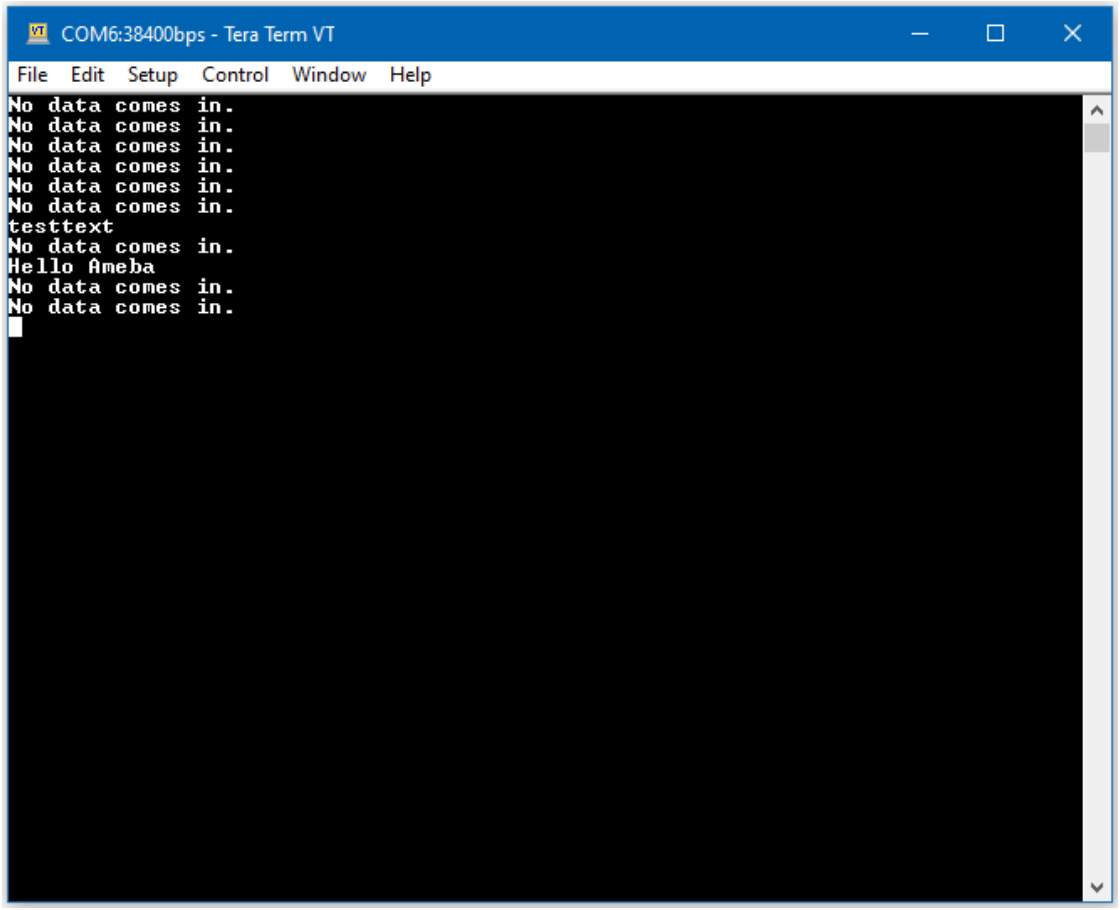


Upload the code and press the reset button on Ameba once the upload is finished.
Next, using a terminal program, such as TeraTerm or PuTTY, open a serial port and configure it according to the settings. Make sure the serial port number corresponds to the USB to TTL adapter.

- Speed: 38400
- Data: 8 bit
- Parity: none
- Stop bits: 1 bit
- Flow control: none



Once the serial port is open, type in the terminal and press the enter key, and you will see the corresponding output.



Code Reference

`mySerial.setAvailableCallback(mySerialCallback);` is used to set the function mySerialCallback as a callback function for software serial. When a new character is received, the callback function checks if the character corresponds to the enter key, and releases the semaphore if it is true, which in turn allows the main loop to print out all the previously received characters.

UART - PM2.5 Concentration in The Air

Preparation

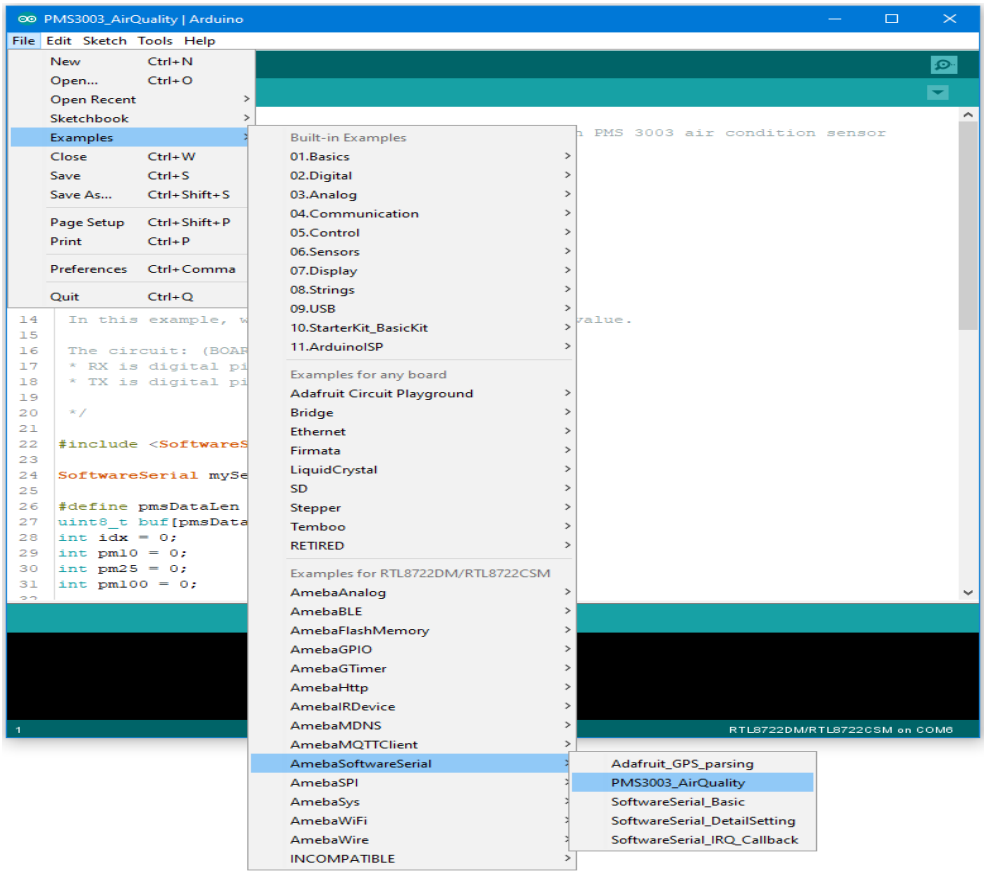
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x 1

Example

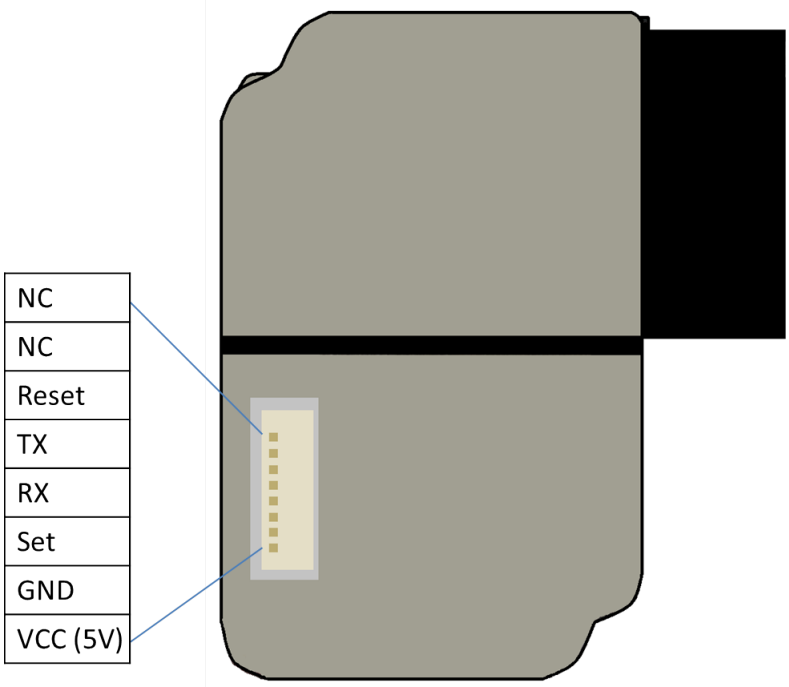
PMS3003 (or PMS5003) is a sensor of air quality, it can detect the concentration of those 0.3 to 10 micrometer particulate matters in the air. The sensor output its data via UART.

The PMS3003 (or PMS5003) sensor detects the concentration value of PM 1.0, PM 2.5, PM 10. Take PM 2.5 for example, it stands for the fine particles with a diameter of 2.5 micrometers or less.

Open the example in “File” -> “Examples” -> “AmebaSoftwareSerial” -> “PMS3003_AirQuality”



There are 8 pins in PMS3003:

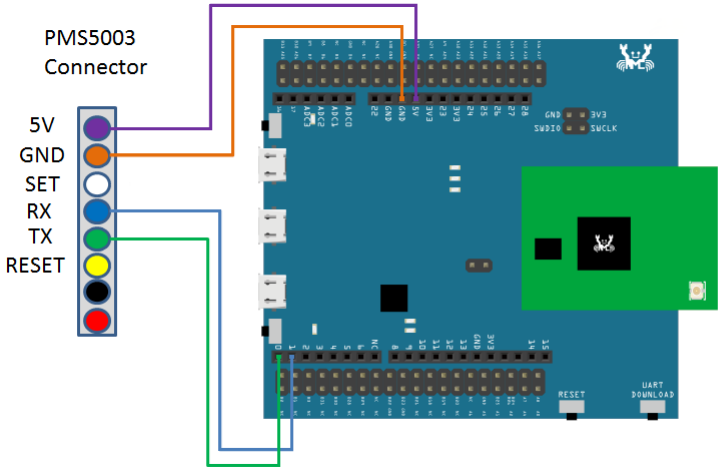


PMS3003 requires 5V power, but the working voltage of its IC is 3.3C. Therefore, the working voltage of Reset, TX, RX, Set are 3.3 as well. If the "Set" pin is pulled to high, the PMS3003 is put to operating mode. If the "Set" pin is pulled low, the PMS3003 is put to standby mode.

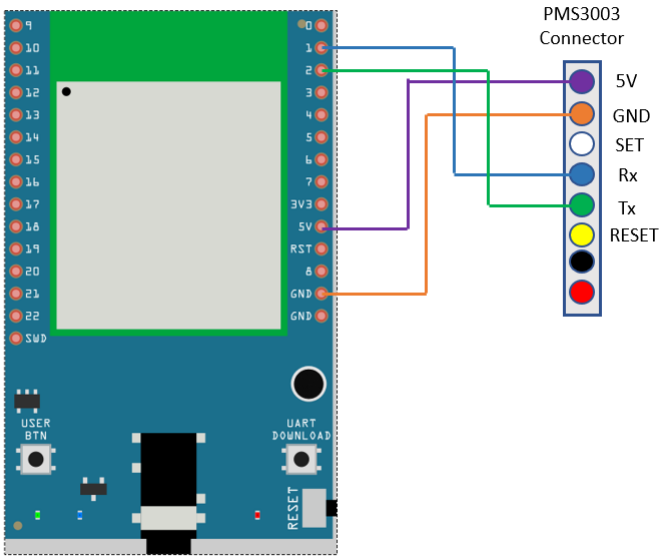
TX/RX pins are for UART connection. Under operating mode, PMS3003 output the data it reads continuously. Each data is of 32 byte, please refer to the following article for detailed data format information:

https://www.dfrobot.com/wiki/index.php?title=PM2.5_laser_dust_sensor_SKU:SEN0177 RTL8722

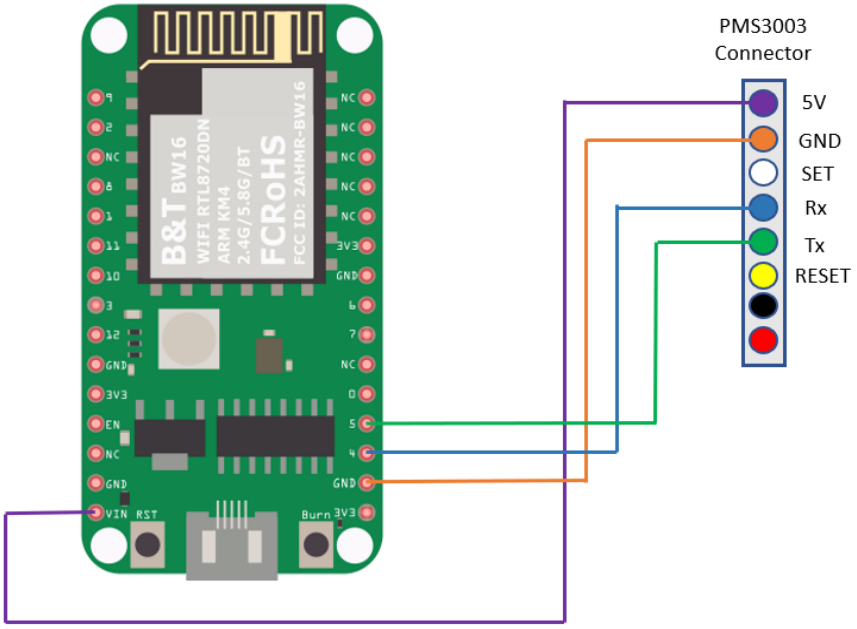
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

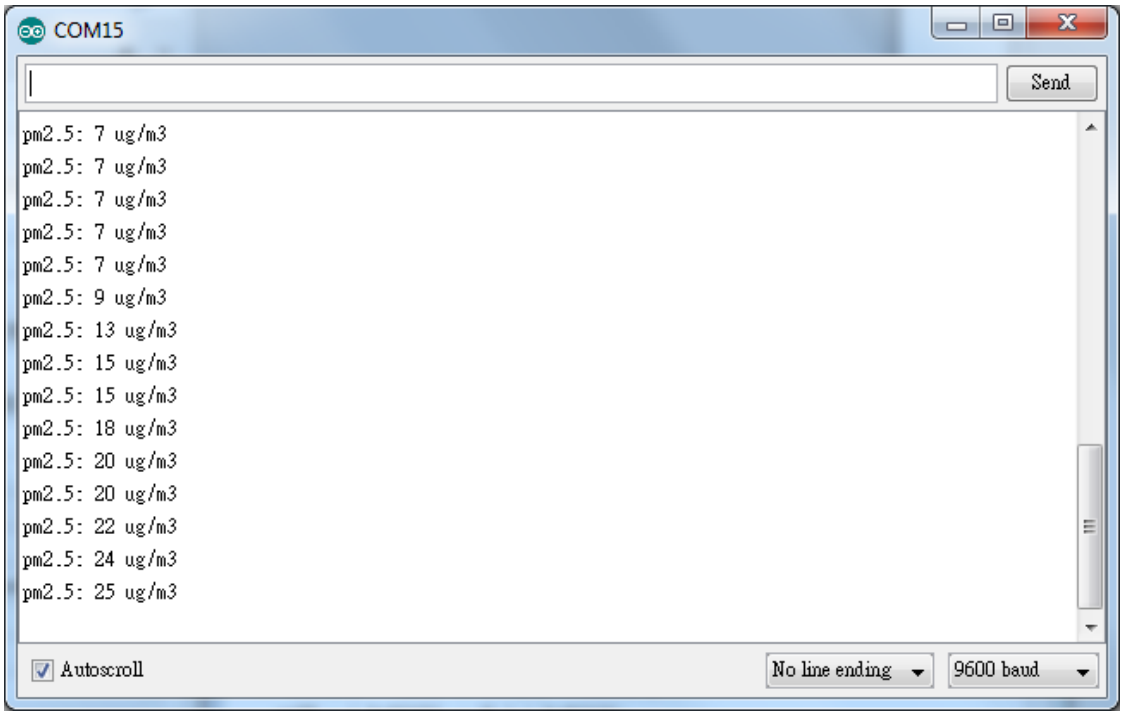


BW16 Wiring Diagram:



In this example, we do not use the “Set” and “Reset” pins.

Compile the code and upload it to Ameba. After pressing the Reset button, Ameba starts to output the PM 2.5 data to serial monitor.



Watchdog - Simple WDG Timer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

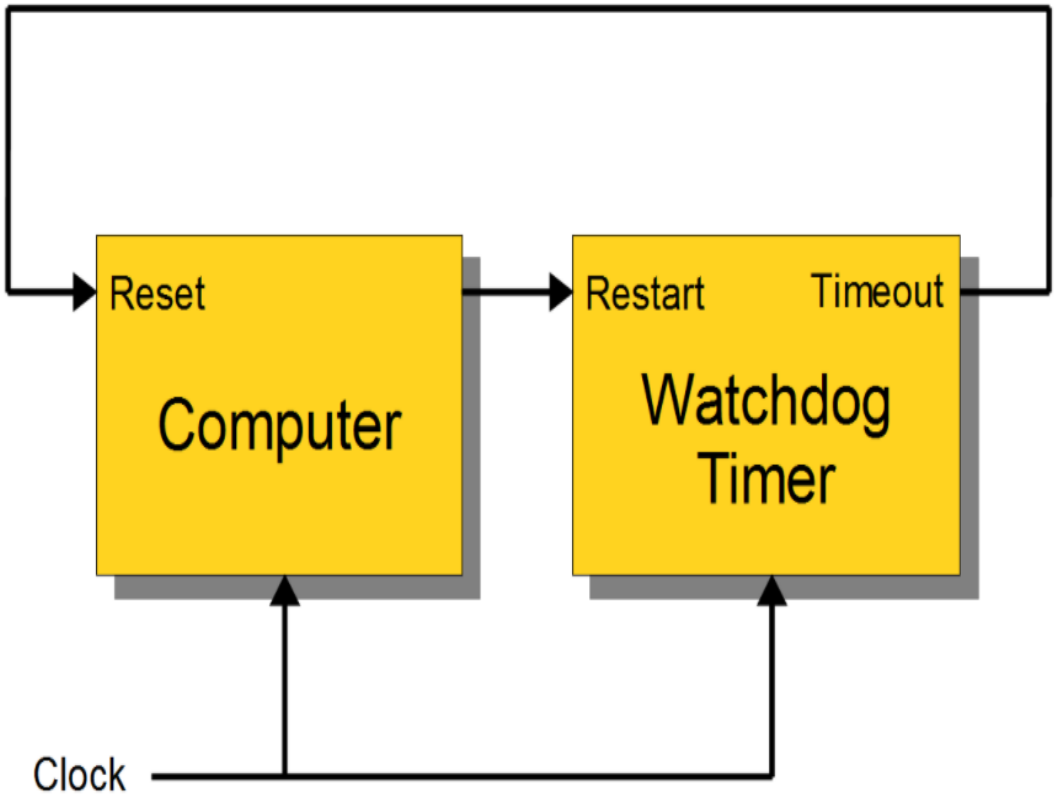
Example

In this example, we will use this simple watchdog timer example runs on the Ameba RTL8722 module to illustrate how to use the watchdog API. Before we get into the details of the example, let's briefly go through the definition of Watchdog as well as it's working principles.

Watchdog

Watchdog Timer (WDT) is a hardware timer that is used to detect the occurrence of a software fault, then automatically generates a system reset or a watchdog interrupt on the expiry of a programmed period.

In layman terms, imagine in the situation while your micro-controller is confused in an infinity loop, or any case like the micro-controller hang while performing some tasks. The normal troubleshooting method would be to press the reset button and jump out of the infinity loop. However, is it practically impossible to do press on the button all time, therefore, the watchdog timer that embedded inside the micro-controller would help with this situation.



Feed the Dog

If you have a dog in your home. You need to feed that dog at a regular interval. if you can't feed one day, it will bite

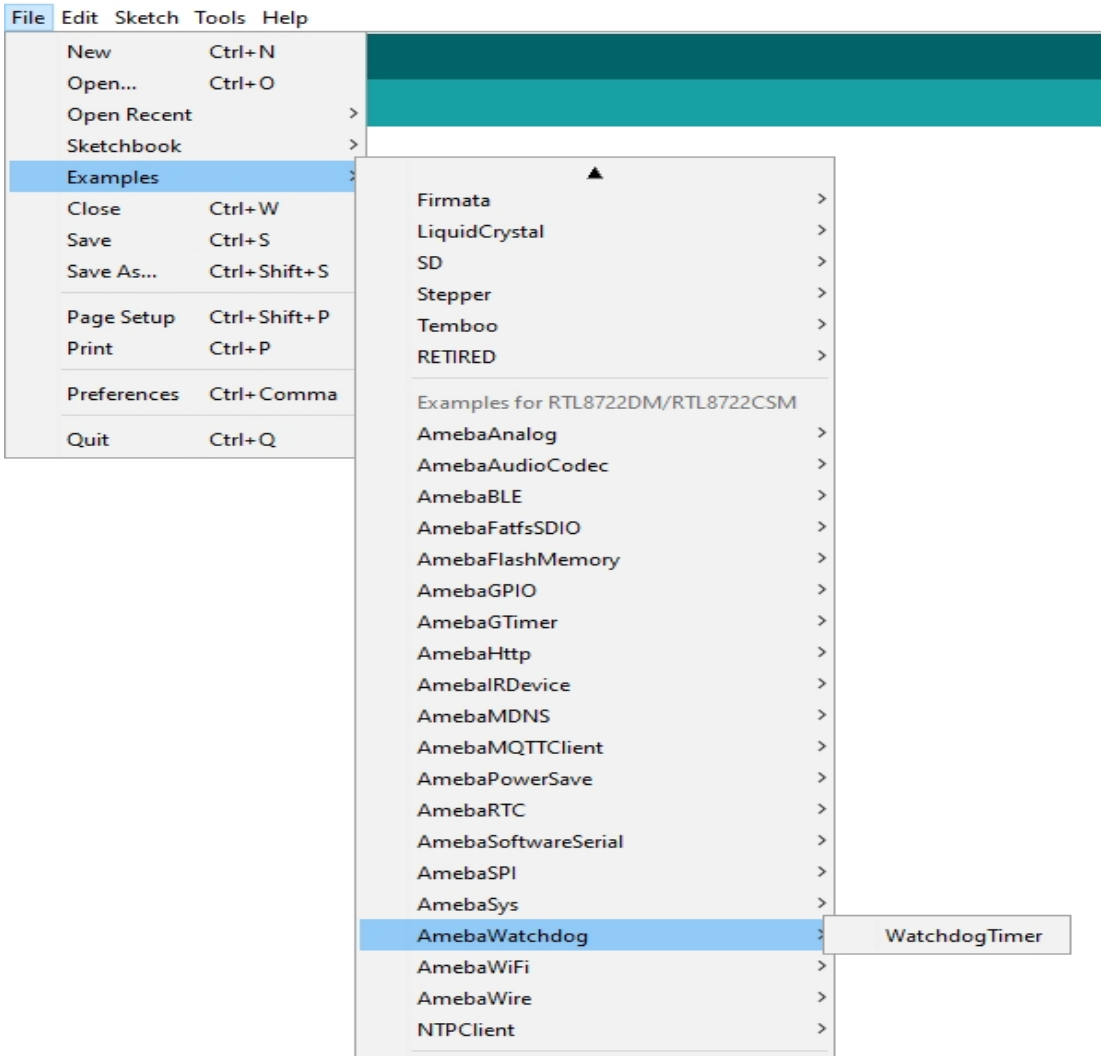
you! And likewise, this is the working logic behind the watchdog timer.

In our example, we created 2 tasks that contain some loop that runs repeatedly, one is called “Small_Task” and the other is called “Big_Task”. We are enabling the watchdog timer is loaded with an initial value (5 seconds) greater than the total delay in the “Small_Task”, but shorter than the “Big_Task”.

For the successful case, the watchdog is being refreshed/feed within 5 seconds, however, for the failed case, the loop is under processing and the watchdog is not being fresh after 5 seconds, which triggers the watchdog (dog barks), an interrupt is generated to reset the processor. Likewise, the watchdog timer protects the micro-controller from the hanging case.

Then we move to the coding part for this example, for this example, you will only need the RTL8722CSM/RTL8722DM/RTL8722DM MINI Board itself.

Firstly, make sure the correct Ameba development board is selected in Arduino IDE: “`Tools`” -> “`Board`” -> “`RTL8722CSM/RTL8722DM`” (or “`RTL8722DM MINI`”). Then open the “Watchdog Timer” example in “`File`” -> “`Examples`” -> “`AmebaWatchdog`” -> “`Watchdog Timer`”:



Upon successfully upload the sample code, open the serial monitor, and press the reset button. You will find that the "Small_Task" can refresh the watchdog within the 5 seconds (initialized in the watchdog timer). However, the "Big_Task" will not be able to refresh the watchdog within 5 seconds, which the watchdog "barks" then the microcontroller reset.


```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
```

COM10

```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
watchdog barks!!!
doing dummy task #6
doing dummy task #7
doing dummy task #8
doing dummy task #9
doing dummy task #10
Big_Task finished refresh watchdog.
```

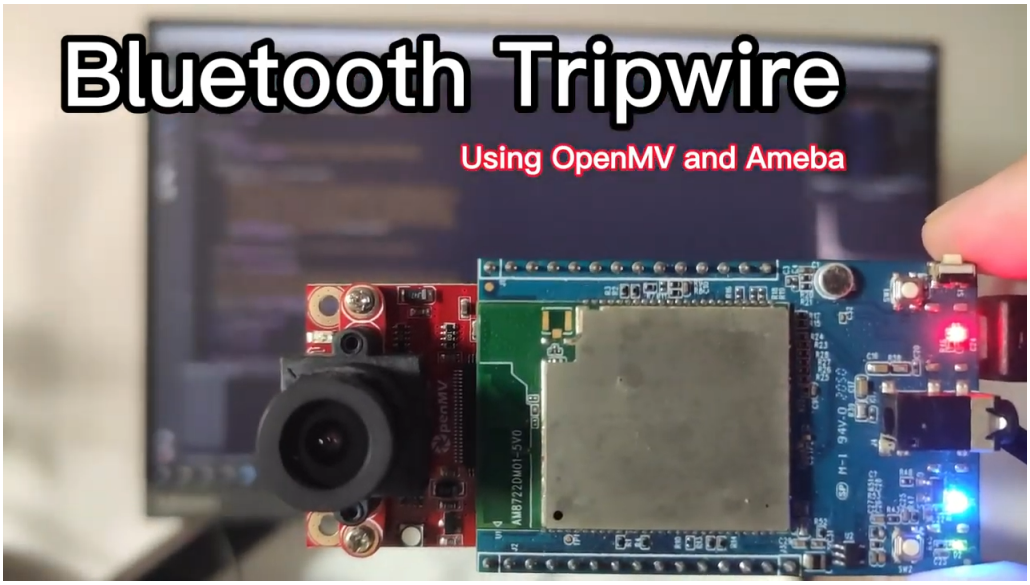
Community Examples

Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards.
To make contributions, please visit our official GitHub Wiki page.

OpenMV Bluetooth TripWire

CONTRIBUTED BY: SIMON XI (https://github.com/xidameng)

Hi if you haven't watched the demo video, feel free to play this short clip below to see what it's capable of.



Introduction

This project took the inspiration from the another open-source project **daytripper** (link 1) which uses 2 seperate devices to detect movement and control your PC to switch Apps. However, I think if we go with the Computer Vision solution instead, we might reduce the number of hardware to just 1, and we can even push it a little further by adding some more cool features like face recognition, speed detection and even more.

That's how I came up with this idea – using **OpenMV**, which is littler and easier to deploy, and a **IoT Microcontroller**, in this case Ameba RTL8722DM_MINI, together we can achieve the same function as **daytripper** and more.

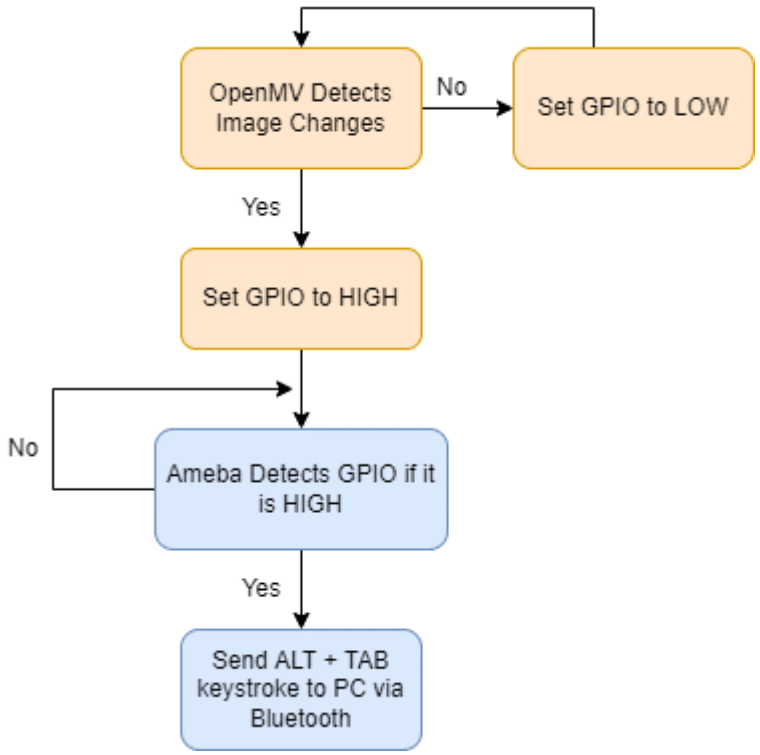
Components

1. AMB23 (RTL8722DM MINI) dev board x1
2. OpenMV(any model) dev board x1

Connection is simple, just connect P0 pin on OpenMV to pin 9 on Ameba Board.

Function Flow

This is how it works,



Code

OpenMV

```
# Advanced Frame Differencing Example
#
# This example demonstrates using frame differencing with your OpenMV Cam. This
# example is advanced because it preforms a background update to deal with the
# backgound image changing overtime.

import sensor, image, pyb, os, time
from pyb import Pin

p_out = Pin('P0', Pin.OUT_PP)
p_out.low()

TRIGGER_THRESHOLD = 5

BG_UPDATE_FRAMES = 50 # How many frames before blending.
BG_UPDATE_BLEND = 128 # How much to blend by... ([0-256]==[0.0-1.0]).

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.RGB565
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or others)
sensor.skip_frames(time = 2000) # Let new settings take affect.
sensor.set_auto_whitebal(False) # Turn off white balance.
clock = time.clock() # Tracks FPS.

# Take from the main frame buffer's RAM to allocate a second frame buffer.
```

(continues on next page)

(continued from previous page)

```
# There's a lot more RAM in the frame buffer than in the MicroPython heap.
# However, after doing this you have a lot less RAM for some algorithms...
# So, be aware that it's a lot easier to get out of RAM issues now. However,
# frame differencing doesn't use a lot of the extra space in the frame buffer.
# But, things like AprilTags do and won't work if you do this...
extra_fb = sensor.alloc_extra_fb(sensor.width(), sensor.height(), sensor.RGB565)

print("About to save background image...")
sensor.skip_frames(time = 2000) # Give the user time to get ready.
extra_fb.replace(sensor.snapshot())
print("Saved background image - Now frame differencing!")

triggered = False

frame_count = 0
while(True):
    clock.tick() # Track elapsed milliseconds between snapshots().
    img = sensor.snapshot() # Take a picture and return the image.

    frame_count += 1
    if (frame_count > BG_UPDATE_FRAMES):
        frame_count = 0
        # Blend in new frame. We're doing 256-alpha here because we want to
        # blend the new frame into the backgound. Not the background into the
        # new frame which would be just alpha. Blend replaces each pixel by
        # ((NEW*(alpha))+(OLD*(256-alpha)))/256. So, a low alpha results in
        # low blending of the new image while a high alpha results in high
        # blending of the new image. We need to reverse that for this update.
        img.blend(extra_fb, alpha=(256-BG_UPDATE_BLEND))
        extra_fb.replace(img)

# Replace the image with the "abs(NEW-OLD)" frame difference.
img.difference(extra_fb)

hist = img.get_histogram()
# This code below works by comparing the 99th percentile value (e.g. the
# non-outlier max value against the 90th percentile value (e.g. a non-max
# value. The difference between the two values will grow as the difference
# image seems more pixels change.
diff = hist.get_percentile(0.99).l_value() - hist.get_percentile(0.98).l_value()
triggered = diff > TRIGGER_THRESHOLD

if triggered == True:
    p_out.high()
else:
    p_out.low()


print(clock.fps(), triggered) # Note: Your OpenMV Cam runs about half as fast while
# connected to your computer. The FPS should increase once disconnected.
```

AMB23

```
#include "BLEHIDDevice.h"
#include "BLEHIDKeyboard.h"
#include "BLEDevice.h"

BLEHIDKeyboard keyboardDev;
BLEAdvertData advdata;

#define ENABLE_PIN 9

void setup() {
Serial.begin(115200);
advdata.addFlags();
advdata.addCompleteName("AMEBA_BLE_HID");
advdata.addAppearance(GAP_GATT_APPEARANCE_HUMAN_INTERFACE_DEVICE);
advdata.addCompleteServices(BLEUUID(HID_SERVICE_UUID));

BLEHIDDev.init();

BLE.init();
BLE.configAdvert()->setAdvData(advdata);
BLE.setDeviceName("AMEBA_BLE_HID");
BLE.setDeviceAppearance(GAP_GATT_APPEARANCE_HUMAN_INTERFACE_DEVICE);
BLE.configSecurity()->setPairable(true);
BLE.configSecurity()->setAuthFlags(GAP_AUTHEN_BIT_BONDING_FLAG);
BLE.configServer(3);
BLE.addService(BLEHIDDev.hidService());
BLE.addService(BLEHIDDev.battService());
BLE.addService(BLEHIDDev.devInfoService());

pinMode(ENABLE_PIN, INPUT);

BLE.beginPeripheral();
}

int flag = 0;

void loop() {
if (BLE.connected() && digitalRead(ENABLE_PIN) && flag == 0) {
    Serial.println("Sending keystrokes");
    keyboardDev.keyReleaseAll();
    delay(100);
    keyboardDev.keyPress(HID_KEY_ALT_LEFT);
    delay(100);
    keyboardDev.keyPress(HID_KEY_TAB);
    keyboardDev.keyReleaseAll();
    delay(100);
    flag = 1;
} else {
    flag = 0;
    delay(100);
}
}
```

Conclusion

This project is not perfect as it's done in a rush, so if anyone wants to perfect it you may go ahead and change my code however you like, or leave a comment below if you have a question or want to discuss something with me~

Until next time, happy coding.

Tip: Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards

---

1.2.3 Release History

Version 3.1.2 - 2021/12/28

- Feature:
  - Update SimpleWebServerWiFi example
  - Support BW16
- API Updates:
  - Update Wlan related naming from "AmebaWiFi" become "WiFi"
  - Update RTC library for minor bug fix
- Misc:
  - Update all Fritzing files for new name updates
  - AMB21, AMB22, AMB23, and BW16

Version 3.1.1 - 2021/12/25

- Feature:
  - Add BLE HID and examples
  - BLEHIDGamepad, BLEHIDKeyboard, and BLEHIDMouse
  - Update PowerSave examples
  - Support RTL8722DM MINI and RTL8720DN/BW16
  - Enable LwIP hostname edit
- API Updates:
  - Update API for PowerSave
  - Update ameba_d_tools 1.0.7 for all 3 platforms
  - Support RTL8720DN/BW16 and RTL8722DM MINI
  - Add more Aon wake up pins
  - Update API for IR
  - Removed requirement to define both IR TX and RX pins in IRDevice::begin
  - Removed previous limit on number of time durations IRDevice::send can accept
  - Update GPIO Int

  - Enable INPUT_IRQ_CHANGE
  - Add definition inside wiring_constants.h and wiring_digital.c, also complete the TODO part for attachInterrupt() as well
  - Update UART, for RTL8720DN/BW16 not showing log issue
  - Fix wrong attribute permissions for characteristic CCCD descriptor. Remove unused variable warnings
  - Update GTimer, for the internal timer ID validation test
  - Updated SPI connection for RTL8720DN/BW16
  - Update Google_Cloud_IoT example with new Google TLS cert
  - Update Analog Pin remove A0 and A1
  - Update Platform.txt for Windows OS with User Name having a space in between
  - Update all libs
- Misc:
  - Update AmebaEink.zip, SPI connection for RTL8720DN/BW16
  - Add Autoflash_patch folder
  - Update the Fritzing of RTL8720DN/BW16, remove A0 and A1

Version 3.1.0 - 2021/11/05

- Feature:
  - Support board RTL8720DN(BW16)
  - Add WiFiControlCar example
  - Add Arduboy zip library
  - Add WPA3 support
  - Add Amebad_HMI_MQTT zip library
  - Add support for IPV6 wiht 4 examples
  - WLAN lib update
  - Minor bug fix
- API Updates:
  - Support Microsoft Azure IoT cloud
  - Enable “strnlen” from rom
  - Add “#define yield” for compilation
  - Update PubSubClient lib
  - Update APIs for RTL8720DN(BW16) (SPI, I2C, Fatfs, Audiocodec and UART
  - Update jtag enable functions
  - Update wifi security option
  - Remove the unused libs lib_wifi_fw.a lib_wifi_ucps_fw.a
  - Update watchdog

  - Update AudioCodec
  - Pin mapping updates
  - Remove unused marcos
  - RTL8720DN(BW16) related naming update for all examples
  - Update PowerSave
- Misc
  - Add RTL8720DN_BW16 frizting folder
  - Move RTL8720DN_BW16 frizting files to correct folder
  - Rename folder name to short the length of path
  - Add Offline_SDK_installation_tool (Windows, Linux and MacOS)
  - Update linux tools for compatibility issue
  - Update RTL8722DM MINI and RTL8720DN(BW16) Fritzing and Pinmux
  - Update ameba_d_tools V1.0.6
  - Add Image_Releated folder
  - Correct the core from Cortex-M4 to Cortex-M33

Version 3.0.11 - 2021/10/26

- Feature:
  - Add example, FatfsSDIO - Read and open HTML file from SD card
- API Updates:
  - RTL8720DN/BW16 related compatibility update for all examples
- Misc
  - Update RTL8722DM MINI and RTL8720DN Fritzing and Pinmux

Version 3.0.10 - 2021/09/22

- Feature:
  - Add AudioCodec wav examples
- API Updates:
  - Pin mapping updates for RTL8722DM MINI
  - Remove unused marcos
  - Update platform.txt for bin files process
  - rollback for “wifi.h” update
  - Minor bug fix patch

Version 3.0.9 - 2021/09/13

- API Updates:

  - Pin mapping updates
  - Remove unused marcos
  - “wifi.h” related files change to “Amebawifi.h”

Version 3.0.8 - 2021/05/06

- Feature:
  - Add RTL8722DM_mini board
  - Add fatfs for SD card
  - Add AudioCodec
  - Add TensorFlow lite support with examples
  - Add zip libraries for TensorFlow lite support
  - Update SDK for supporting Arduino IDE 2.0
  - Update wlan lib
- API Updates:
  - Update zip libraries of Eink
  - ADC updates, Change calculation method to use EFUSE calibration parameters and SDK formula to improve accuracy
  - writing_analog updates, minor bug fix and support for mini board
  - SPI updates, minor bug fix and support for mini board
  - I2S updates, minor bug fix and support for mini board
  - IRDevice updates, minor bug fix

Version 3.0.7 - 2020/11/19

- Feature:
  - Add AmebaIRDevice example IRSendSONY
  - Update Ameba Arduino IRDevice API
  - Update Ameba Arduino SSL related API
  - Update Ameba Arduino Wlan API to support static IP function

---

Version 3.0.6 - 2020/10/28

- Feature:
  - Add Ameba RTC support
  - Add AmebaRTC example RTC and RTCAlarm
  - Add Ameba Watchdog support
  - Add AmebaWatchdog example WatchdogTimer
  - Update Ameba BLE support

  - Add AmebaBLE example BLEUartService, DHT_over_BLEUart
  - Update Ameba Wlan library
  - Update Ameba Wlan SDK structure, add AP mode hidden SSID support

---

Version 3.0.5 - 2020/09/09

- Feature:
  - Build in tool updates V1.0.4
  - Add zip lib AmebaEink
  - Add AmebaEink example EinkDisplayImage, EinkDisplayQR, and EinkDisplayText
  - Add google cloud examples
  - Update Amazon AWS related examples
  - Add power save support
  - Add AmebaPowerSave example TicklessMode, DeepSleepMode, DeepSleep_DHT_LCD_Example, and DeepSleep_DHT_Eink_Example

---

Version 3.0.4 - 2020/07/27

- Feature:
  - Update BLE library. Add example BLEBatteryClient and BLEWIfiConfig
  - Update from polarssl to mbedtls 2.4.0

Version 3.0.3 - 2020/07/03

- Feature:
  - Build in Image tool updates V1.0.3
  - Upload log clean up

Version 3.0.2 - 2020/06/30

- Feature:
  - Windows, Linux and macOS X support
  - Build in Image tool updates

Version 3.0.1 - 2020/05/15

- Feature:
  - Official release of AmebaD Arduino SDK
  - warning cleaning
  - I2C lib updates

Version 3.0.0 - 2020/05/01

- Feature:
  - Support Boards Manager and Arduino IDE development
  - WiFi scan AP, connect to AP, TCP Server/Client, including 5G
  - Bluetooth, BLE
  - GPIO digital in/out and interrupt
  - ADC analog in/out (0 ~ 3.3V)
  - PWM getting analog results with digital means
  - SPI master and slave mode
  - UART 1 for log, 2 for customize usage
  - I2C master mode

1.2.4 Board HDK

- `Layout`
- `Schematic`

1.2.5 API Documents

RTL8722DM ARDUINO Online API Documents

Analog

Class AmebaServo

AmebaServo Class

Description

Defines a class of manipulating servo motors connected to Arduino pins.

Syntax

class AmebaServo

Members

Public Constructors	
AmebaServo::AmebaServo	Constructs an AmebaServo object.
Public Methods	
AmebaServo::attach	Attach the given pin to the next free channel.
AmebaServo::detach	Detach the servo.
AmebaServo::write	Write value, if the value is < 200 it's treated as an angle, otherwise as pulse-width in microseconds.
AmebaServo::writeMicroseconds	Write pulse width in microseconds.
AmebaServo::read	Output current pulse width as an angle between 0 and 180 degrees.
AmebaServo::readMicroseconds	Output current pulse width in microseconds for this servo.
AmebaServo::attached	Check if the servo is attached.

AmebaServo::attach

Description

Attach the given pin to the next free channel, sets pinMode (including minimum and maximum values for writes), returns channel number, or 0 if failure.

Syntax

```
uint8_t attach(int pin);

uint8_t attach(int pin, int min, int max);
```

Parameters

pin: The Arduino pin number to be attached.

min: Minimum values for writes.

max: Maximum values for writes.

Returns

The function returns channel number or 0

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree.

Listing 2: ServoSweep.ino

```
/* Sweep
by BARRAGAN < http://barraganstudio.com >
This example code is in the public domain.
modified 8 Nov 2013
by Scott Fitzgerald
http://www.arduino.cc/en/Tutorial/Sweep
refined 2016/03/18 by Realtek
*/

```

(continues on next page)

(continued from previous page)

```
#include "AmebaServo.h"

// create servo object to control a servo
// 4 servo objects can be created correspond to PWM pins

AmebaServo myservo;

// variable to store the servo position
int pos = 0;

void setup() {
#if defined(BOARD_RTL8195A)
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#elif defined(BOARD_RTL8710)
    // attaches the servo on pin 13 to the servo object
    myservo.attach(13);
#elif defined(BOARD_RTL8721D)
    // attaches the servo on pin 8 to the servo object
    myservo.attach(8);
#else
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#endif
}

void loop() {
    // goes from 0 degrees to 180 degrees in steps of 1 degree
    for (pos = 0; pos <= 180; pos += 1) {
    // tell servo to go to position in variable 'pos'
    myservo.write(pos);
    // waits 15ms for the servo to reach the position
    delay(15);
    }
    // goes from 180 degrees to 0 degrees
    for (pos = 180; pos >= 0; pos -= 1) {
        // tell servo to go to position in variable 'pos'
        myservo.write(pos);
        // waits 15ms for the servo to reach the position
        delay(15);
    }
}
```

Notes and Warnings

Every time must include the header file “AmebaServo.h” in front of the project to use the class function.

AmebaServo::detach

Description

Detach the servo.

Syntax

void AmebaServo::detach(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::write

Description

Write an integer value to the function, if the value is < 200, it's being treated as an angle, otherwise as pulse-width in microseconds.

Syntax

void AmebaServo::write(int value);

Parameters

value: The value < 200 its treated as an angle; otherwise as pulse width in microseconds.

Returns

The function returns nothing.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::writeMicroseconds

Description

Write pulse width to the servo in microseconds.

Syntax

void AmebaServo::writeMicroseconds(int value);

Parameters

value: Write value the pulse width in microseconds.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file “AmebaServo.h” in front of the project to use the class function.

AmebaServo::read

Description

The function reads current pulse width and returns as an angle between 0 and 180 degrees.

Syntax

int AmebaServo::read(void);

Parameters

The function requires no input parameter.

Returns

The pulse width as an angle between 0 ~ 180 degrees.

Example Code

NA

Notes and Warnings

Every time must include the header file “AmebaServo.h” in front of the project to use the class function.

AmebaServo::readMicroseconds

Description

The function returns a Boolean value “true” if this servo is attached, otherwise returns “false”.

Syntax

int AmebaServo::readMicroseconds(void);

Parameters

The function requires no input parameter.

Returns

The function returns current servo pulse width in microseconds.

Example Code

NA

Notes and Warnings

Every time must include the header file “AmebaServo.h” in front of the project to use the class function.

AmebaServo::attached

Description

It returns true if this servo is attached, otherwise false.

Syntax

bool AmebaServo::attached(void);

Parameters

The function requires no input parameter.

Returns

The function returns a Boolean value as true or false.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AudioCodec

Class AudioCodec

Description

A class used for general control and management of the hardware audio codec functions.

Syntax

class AudioCodec

Members

Public Constructors

The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named Codec.

Public Methods

AudioCodec::begin	Configure and start the audio codec for transmit and receive operation
AudioCodec::end	Stop all audio codec operation
AudioCodec::getBufferPageSize	Get the byte size of a single page of the audio codec buffer
AudioCodec::setSampleRate	Configure the audio codec transmit and receive sampling rate
AudioCodec::setBitDepth	Configure the audio codec transmit and receive bit depth (bits per sample)
AudioCodec::setChannelCount	Configure the audio codec transmit and receive channel count
AudioCodec::setInputMicType	Configure for analog or digital input microphone type
AudioCodec::setInputLRMux	Configure input left right channel multiplexing
AudioCodec::setDMicBoost	Configure boost gain for digital microphone input
AudioCodec::setAMicBoost	Configure boost gain for analog microphone input
AudioCodec::setADCGain	Configure gain of ADC used to acquire analog input
AudioCodec::muteInput	Mute input audio data stream
AudioCodec::setOutputVolume	Configure output audio volume
AudioCodec::muteOutput	Mute output audio
AudioCodec::writeAvaliable	Check for free buffer page available for data write
AudioCodec::writeDataPage	Write audio data to an available buffer page
AudioCodec::readAvaliable	Check for buffer page with new data available for read
AudioCodec::readDataPage	Read audio data from a ready buffer page
AudioCodec::setWriteCallback	Set a callback function to be notified when a free buffer page is available for write
AudioCodec::setReadCallback	Set a callback function to be notified when a buffer page with new data is available for read

AudioCodec::begin

Description

Configure and start the audio codec for transmit and receive operation.

Syntax

void begin(bool input, bool output);

Parameters

input: enable audio codec data input

output: enable audio codec data output

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::end

Description

Stop all audio codec operation.

Syntax

void end();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::getBufferPageSize

Description

Get the byte size of a single page of the audio codec buffer.

Syntax

uint32_t getBufferPageSize();

Parameters

The function requires no input parameter.

Returns

The size of a audio codec buffer page, in number of bytes.

Example Code

NA

Notes and Warnings

The AudioCodec class includes a transmit and receive buffer to store audio sample data while transferring to and from the DAC output and ADC input. The buffer is divided into pages of fixed size, and audio data can be read and written one page at a time. Depending on the configured bit depth (bits per audio sample) and channel count, a buffer page may contain a different number of audio samples.

AudioCodec::setSampleRate

Description

Configure the audio codec transmit and receive sampling rate.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: desired audio codec sampling rate in Hz. Default value of 48000. Supported values: 8000, 16000, 32000, 44100, 48000, 88200, 96000.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

High sample rates above 48000Hz will require frequent buffer reads and writes to keep up with the large amount of data input and output. If there is insufficient processing time dedicated to this task, audio quality will be degraded.

AudioCodec::setBitDepth

Description

Configure the audio codec transmit and receive bit depth (bits per sample).

Syntax

void setBitDepth(uint8_t bitDepth);

Parameters

bitDepth: desired number of bits per sample. Default value of 16. Supported values: 8, 16, 24.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Setting a bit depth of 24 bits per sample will require 32 bits (4 bytes) of buffer space for storing each sample, with the most significant byte ignored.

AudioCodec::setChannelCount

Description

Configure the audio codec transmit and receive channel count.

Syntax

void setChannelCount(uint8_t monoStereo);

Parameters

monoStereo: number of channels. Default value of 1. Supported values: 1, 2.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setInputMicType

Description

Configure for analog or digital input microphone type.

Syntax

Void setInputMicType(Mic_Type micType);

Parameters

micType: Input microphone type. Default value ANALOGMIC. Valid values:

- ANALOGMIC – microphone with an analog output

- PDMMIC – digital microphone with a PDM output

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

For analog single-ended output, connect to PA_4 for the left channel and PA_2 for the right channel.

For digital PDM output, connect the PDM clock to PB_1 and PDM data to PB_2.

AudioCodec::setInputLRMux

Description

Configure input left right channel multiplexing.

Syntax

void setInputLRMux(uint32_t mux);

Parameters

mux: desired left right audio channel multiplexing setting. Default value RX_CH_LR. Valid values:

- RX_CH_LR
- RX_CH_RL
- RX_CH_LL
- RX_CH_RR

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In mono channel mode, both RX_CH_LR and RX_CH_LL will result in the audio codec sampling input data from the left channel microphone. Similarly, both RX_CH_RL and RX_CH_RR will result in the audio codec sampling input data from the right channel microphone.

In stereo channel mode, RX_CH_RL will switch the positions of input data sampled from the microphones. RX_CH_RR and RX_CH_LL will result in duplicated samples from the right and left microphones respectively.** **

AudioCodec::setDMicBoost

Description

Configure boost gain for digital microphone input.

Syntax

void setDMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel digital microphone input

rightBoost: boost gain for right channel digital microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 12dB
- 2 : 24dB
- 3 : 36dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setAMicBoost

Description

Configure boost gain for analog microphone input.

Syntax

void setAMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel analog microphone input

rightBoost: boost gain for right channel analog microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 20dB
- 2 : 30dB
- 3 : 40dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this function if additional gain is required after using setADCGain function.

AudioCodec::setADCGain

Description

Configure gain of ADC used to acquire analog input.

Syntax

void setADCGain(uint32_t leftGain, uint32_t rightGain);

Parameters

leftGain: Gain for left channel ADC

rightGain: Gain for right channel ADC

Valid value range is from 0x00 to 0x7f. Gain increases by 0.375dB for every increment in value:

- 0x00 : -17.625dB
- 0x01 : -17.25dB
- 0x2f : 0dB
- 0x30 : 0.375dB
- 0x7f : 30dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::muteInput

Description

Mute input audio data stream.

Syntax

void muteInput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel input, 0 to unmute

rightMute: 1 to mute right channel input, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setOutputVolume

Description

Configure output audio volume.

Syntax

void setOutputVolume(uint8_t leftVol, uint8_t rightVol);

Parameters

leftVol: left channel output volume

rightVol: right channel output volume

Valid value ranges from 0 to 100, corresponding to a volume of -65.625dB to 0dB.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::muteOutput

Description

Mute output audio.

Syntax

void muteOutput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel output, 0 to unmute

rightMute: 1 to mute right channel output, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::writeAvaliable

Description

Check for free buffer page available for data write.

Syntax

bool writeAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page that is available for writing data into. Returns false if all buffer pages are full.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::writeDataPage

Description

Write audio data to an available buffer page.

Syntax

uint32_t writeDataPage(int8_t* src, uint32_t len);

uint32_t writeDataPage(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing audio samples to write to audio codec.

len: number of audio samples in array.

Returns

The function returns the number of audio samples written to the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readAvaliable

Description

Check for buffer page with new data available for read.

Syntax

bool readAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page with new data that is ready for reading data from. Returns false if all buffer pages are empty.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readDataPage

Description

Read audio data from a ready buffer page.

Syntax

uint32_t readDataPage(int8_t* dst, uint32_t len);

uint32_t readDataPage(int16_t* dst, uint32_t len);

Parameters

dst: pointer to array to contain audio samples read from audio codec.

len: number of audio samples to read.

Returns

The function returns the number of audio samples read from the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setWriteCallback

Description

Set a callback function to be notified when a free buffer page is available for write.

Syntax

void setWriteCallback(**void** (*writeCB)(**void*));

Parameters

writeCB: function to be called when a buffer page becomes available for data write. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec finishes outputting the data in a buffer page.

AudioCodec::setReadCallback

Description

Set a callback function to be notified when a buffer page with new data is available for read.

Syntax

void setReadCallback(**void** (*readCB)(**void*));

Parameters

readCB: function to be called when a buffer page with new data becomes available for data read. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec fills up a buffer page with newly acquired audio samples.

Class FFT

Description

A class used for performing FFT calculations with real-number inputs and outputs.

Syntax

class FFT

Members

Public Constructors

FFT::FFT	Create an instance of the FFT class

Public Methods

FFT::setWindow	Configure the window function used in FFT calculations
FFT::calculate	Calculate FFT for an input array of values
FFT::getFrequencyBins	Get the FFT output frequency bins
FFT::getFFTSize	Get the size of FFT output for a given input size

FFT::FFT

Description

Create a FFT class object.

Syntax

void FFT();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

FFT::setWindow

Description

Configure the window function used in FFT calculations.

Syntax

void setWindow(FFTWindow_t window, uint16_t sampleCount);

Parameters

window: The window function to be used in FFT calculations. Valid values: None, Hann, Hamming.

sampleCount: Number of sample datapoints in the input.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

The window function is used to reduce the effects of discontinuities that occur when the input signal has frequencies that do not fit an integer number of periods in the sample datapoints.

More information on FFTs and window functions can be seen at:

https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf

https://en.wikipedia.org/wiki/Window_function

FFT::Calculate

Description

Calculate FFT for an input array of values.

Syntax

void calculate(float* inputBuf, float* outputBuf, uint16_t sampleCount);

void calculate(int16_t* inputBuf, float* outputBuf, uint16_t sampleCount);

Parameters

inputBuf: pointer to an array of sampleCount size, containing input sample datapoints, in float or uint16_t format.

outputBuf: pointer to a float array of sampleCount/2 size, for containing FFT output.

sampleCount: number of sample datapoints in the input array, valid values: 16, 32, 64, 128, 256, 512, 1024, 2048.

Returns

The function returns nothing.

Example Code

Example:FFT

Notes and Warnings

Large sample counts will require a longer time for FFT calculations, but will also return a result with higher frequency resolution.

FFT::getFrequencyBins

Description

Get the FFT output frequency bins.

Syntax

void getFrequencyBins(uint16_t* outputBuf, uint16_t sampleCount, uint32_t sampleRate);

Parameters

outputBuf: pointer to a uint16_t array of sampleCount/2 size, for containing the calculated center frequency of each FFT output element.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings NA

—

FFT::getFFTSize

Description

Get the size of FFT output for a given input size.

Syntax

uint16_t getFFTSize(uint16_t sampleCount);

Parameters

sampleCount: number of input sample datapoints.

Returns

The function returns the FFT output size for the given sampleCount, which is sampleCount/2.

Example Code

NA

Notes and Warnings NA

Class PlaybackWav

Description

A class used for control and playback of .wav file format audio data.

Syntax

class PlaybackWav

Members

Public Constructors

PlaybackWav::PlaybackWav	Create an instance of the PlaybackWav class

Public Methods

PlaybackWav::openFile	Open a .wav file for playback
PlaybackWav::closeFile	Close a previously opened file
PlaybackWav::fileOpened	Check if a .wav file is already opened
PlaybackWav::getSampleRate	Get the sample rate of the .wav file
PlaybackWav::getChannelCount	Get the number of audio channels in the .wav file
PlaybackWav::getBitDepth	Get the bit depth of each sample in the .wav file
PlaybackWav::getLengthMillis	Get the playback length of the .wav file in milliseconds
PlaybackWav::getPositionMillis	Get the current playback position in milliseconds
PlaybackWav::setPositionMillis	Set the current playback position in milliseconds
PlaybackWav::millisToBytes	Convert a playback duration to equivalent number of bytes
PlaybackWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
PlaybackWav::readAudioData	Read audio data from the .wav file

---

PlaybackWav::PlaybackWav

Description

Create a PlaybackWav class object.

Syntax

void PlaybackWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordPlaybackWav

Notes and Warnings

NA

PlaybackWav::getSampleRate

Description

Get the sample rate of the .wav file.

Syntax

uint32_t getSampleRate(void);

Parameters

The function requires no input parameter.

Returns

The function returns sampling rate encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getChannelCount

Description

Get the number of audio channels in the .wav file.

Syntax

uint16_t getChannelCount(void);

Parameters

The function requires no input parameter.

Returns

The function returns channel count encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getBitDepth

Description

Get the bit depth of each sample in the .wav file.

Syntax

uint16_t getBitDepth(void);

Parameters

The function requires no input parameter.

Returns

The function returns bit depth encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getLengthMillis

Description

Get the playback length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the total playback length of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getPositionMillis

Description

Get the current playback position in milliseconds.

Syntax

uint32_t getPositionMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current playback position of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::setPositionMillis

Description

Set the current playback position in milliseconds.

Syntax

void setPositionMillis(uint32_t pos);

Parameters

pos: The desired playback position expressed in milliseconds.

Returns

The function returns nothing.

Example Code

Example: PlaybackWavFile

Notes and Warnings

Any changes to playback position will only take effect on the next call to PlaybackWav::readAudioData. If the desired playback position is beyond the total playback length of the file, the playback position will be set to the end of file, and no audio data will be output on subsequent data reads.

PlaybackWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::readAudioData

Description

Read audio data from the .wav file.

Syntax

- `uint32_t readAudioData(int8_t* dst, uint32_t len);`
- `uint32_t readAudioData(int16_t* dst, uint32_t len);`

Parameters

- `dst`: pointer to array to store data read from .wav file.
- `len`: number of audio samples to read from .wav file.

Returns

The function returns number of audio samples read.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

Class RecordWav

Description

A class used for control and recording of .wav file format audio data.

Syntax

class RecordWav

Members

Public Constructors

RecordWav:: RecordWav	Create an instance of the RecordWav class

Public Methods

RecordWav::openFile	Open a .wav file for playback
RecordWav::closeFile	Close a previously opened file
RecordWav::fileOpened	Check if a .wav file is already opened
RecordWav::setSampleRate	Get the sample rate of the .wav file
RecordWav::setChannelCount	Set the number of audio channels in the .wav file
RecordWav::setBitDepth	Set the bit depth of each sample in the .wav file
RecordWav::getLengthMillis	Get the current record length of the .wav file in milliseconds
RecordWav::millisToBytes	Convert a playback duration to equivalent number of bytes
RecordWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
RecordWav::writeAudioData	Write audio data to the .wav file

RecordWav::RecordWav

Description

Create a RecordWav class object.

Syntax

void RecordWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::openFile

Description

Open a .wav file for recording.

Syntax

void openFile(const char* absFilepath);

Parameters

absFilepath: the filepath of the .wav file to open.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::closeFile

Description

Close a previously opened file.

Syntax

void closeFile(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

Any open .wav files should be closed after recording is complete, otherwise, loss of recorded audio data may occur.

RecordWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setSampleRate

Description

Set the recording sample rate of the .wav file.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: The desired recording sample rate.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setChannelCount

Description

Set the number of recording audio channels in the .wav file.

Syntax

void setChannelCount(uint16_t channelCount);

Parameters

channelCount: number of recording audio channels.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::setBitDepth

Description

Set the recording bit depth of each sample in the .wav file.

Syntax

void setBitDepth(uint16_t bitDepth);

Parameters

bitDepth: number of bits per sample.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::getLengthMillis

Description

Get the current recorded length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current recorded length of the currently open .wav file in milliseconds.

Example Code

NA

Notes and Warnings

NA

RecordWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::writeAudioData

Description

Write audio data to the .wav file.

Syntax

uint32_t writeAudioData(int8_t* src, uint32_t len); uint32_t writeAudioData(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing data to write to .wav file. len: number of audio samples to write to .wav file.

Returns

The function returns number of audio samples written.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

BLE

Class BLEAddr

BLEAddr Class

Description

A class used for managing Bluetooth addresses.

Syntax
class BLEAddr

Members

Public Constructors	
BLEAddr::BLEAddr	Constructs a BLEAddr object
Public Methods	
BLEAddr::str	Get the Bluetooth address represented as a formatted string
BLEAddr::data	Get the Bluetooth address represented as an integer array

BLEAddr::BLEAddr

Description
Constructs a BLEAddr object.

Syntax
BLEAddr::BLEAddr(void);
BLEAddr::BLEAddr(uint8_t (&addr)[6]);
BLEAddr::BLEAddr(const char* str);

Parameters
addr: An array of 6 bytes containing the desired Bluetooth address.
str: A character string representing the desired Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
When expressed as a string, the Bluetooth address should be written as 6 bytes in hexadecimal format, using a colon ":" to separate the bytes is acceptable (example – 00:11:22:33:EE:FF).

BLEAddr::str

Description
Get the Bluetooth address represented as a formatted string.

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a character string containing the hexadecimal representation of the Bluetooth address.

Example Code
Example: BLEScan

Notes and Warnings
The Bluetooth address expressed as a string will be written as 6 bytes in hexadecimal format, with a colon ":" separating the bytes (example – 00:11:22:33:EE:FF).

BLEAddr::data

Description
Get the Bluetooth address represented as an integer array.

Syntax
uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a 6 byte array containing the Bluetooth address.

Example Code
NA

Notes and Warnings

The Bluetooth address is stored with MSB at array index [5].

Class BLEAdvert

BLEAdvert Class

Description
A class used for managing BLE advertising settings.

Syntax
class BLEAdvert

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configAdvert().

Public Methods	
BLEAdvert::updateAdvertParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEAdvert::startAdv	Start BLE advertising
BLEAdvert::stopAdv	Stop BLE advertising
BLEAdvert::setAdvType	Set the BLE advertising type
BLEAdvert::setMinInterval	Set the BLE advertising minimum interval
BLEAdvert::setMaxInterval	Set the BLE advertising maximum interval
BLEAdvert::setAdvData	Set BLE advertising data
BLEAdvert::setScanRspData	Set BLE scan response data

BLEAdvert::updateAdvertParams

Description
Update the lower Bluetooth stack with the current advertising settings.

Syntax
void updateAdvertParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Please use the other class member functions to set the BLE advertising parameters first before using this function.

BLEAdvert::startAdv

Description
Start BLE advertising.

Syntax
void startAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly start the BLE advertising process without first registering the necessary callback and handler functions. Call BLEDevice::beginPeripheral() to register the necessary functions and start advertising for the first time.

BLEAdvert::stopAdv

Description
Stop BLE advertising.

Syntax
void stopAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly stop the BLE advertising process. Call BLEDevice::end() to stop advertising and free up used resources.

BLEAdvert::setAdvType

Description
Set the BLE advertising type.

Syntax
void setAdvType(uint8_t advType);

Parameters
advType: the desired advertisement type. Valid values:
– 0 = GAP_ADTYPE_ADV_IND : connectable undirected advertisement
– 1 = GAP_ADTYPE_ADV_HDC_DIRECT_IND : connectable high duty cycle directed
– 2 = GAP_ADTYPE_ADV_SCAN_IND : scannable undirected advertisement
– 3 = GAP_ADTYPE_ADV_NONCONN_IND : Non-connectable undirected advertisement
– 4 = GAP_ADTYPE_ADV_LDC_DIRECT_IND : connectable low duty cycle directed advertisement

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this function with the GAP_ADTYPE_ADV_IND argument if connection requests should be allowed, and GAP_ADTYPE_ADV_NONCONN_IND if all connection requests should be rejected.

BLEAdvert::setMinInterval

Description
Set the minimum BLE advertising interval.

Syntax
void setMinInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement minimum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setMaxInterval

Description
Set the maximum BLE advertising interval.

Syntax
void setMaxInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement maximum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setAdvData

Description
Set BLE advertising data.

Syntax
void setAdvData(BLEAdvertData adData);
void setAdvData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.
size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

BLEAdvert::setScanRspData

Description
Set BLE scan response data.

Syntax
void setScanRspData(BLEAdvertData adData);
void setScanRspData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.

size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

Class BLEAdvertData

BLEAdvertData Class

Description
A class used for managing BLE advertising data.

Syntax
class BLEAdvertData

Members

Public Constructors	
BLEAdvertData::BLEAdvertData	Constructs a BLEAdvertData object

Public Methods	
BLEAdvertData::clear	Clear all advertising data
BLEAdvertData::addData	Add binary advertising data
BLEAdvertData::addFlags	Add flags to advertising data
B LEAdvertData::addPartialServices	Add partial services to advertising data
BL EAdvertData::addCompleteServices	Add complete services to advertising data
BLEAdvertData::addAppearance	Add device appearance to advertising data
BLEAdvertData::addShortName	Add short device name to advertising data
BLEAdvertData::addCompleteName	Add complete device name to advertising data
BLEAdvertData::parseScanInfo	Parse advertising data received from a scan
BLEAdvertData::hasFlags	Check if received data includes advertising flags
BLEAdvertData::hasUUID	Check if received data includes UUIDs
BLEAdvertData::hasName	Check if received data includes device name
BLEAdvertData::hasManufacturer	Check if received data includes manufacturer data
BLEAdvertData::getAdvType	Get advertising type of received data
BLEAdvertData::getAddrType	Get Bluetooth address type of received data
BLEAdvertData::getAddr	Get Bluetooth address of received data
BLEAdvertData::getRSSI	Get RSSI of received data
BLEAdvertData::getFlags	Get advertising flags of received data
BLEAdvertData::getServiceCount	Get number of advertised services in received data
BLEAdvertData::getServiceList	Get array of advertised services in received data
BLEAdvertData::getName	Get advertised device name in received data
BLEAdvertData::getTxPower	Get advertised transmission power in received data
BLEAdvertData::getAppearance	Get advertised device appearance in received data
BLEAdvertData::getManufacturer	Get advertised manufacturer in received data
BLEAdver tData::getManufacturerDataLength	Get length of manufacturer data in received data
BL EAdvertData::getManufacturerData	Get advertised manufacturer data in received data

BLEAdvertData::BLEAdvertData

Description
Constructs a BLEAdvertData object.

Syntax
BLEAdvertData::BLEAdvertData(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class is used for managing BLE advertising data for two primary uses. First is to assemble advertising data for broadcasting as advertising packets. Second is to process and split up the advertising data received from a scan into separate types.

BLEAdvertData::clear

Description
Clear all advertising data currently saved in class object.

Syntax
void clear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addData

Description
Add binary advertising data.

Syntax
void addData(const uint8_t* data, uint8_t size);

Parameters
data: pointer to array containing desired advertising data.
size: number of bytes in array.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in adding BLE advertising data. Other functions should be used for adding advertising data if possible, as this function does not perform any checks on the validity of the data.

BLEAdvertData::addFlags

Description
Add flags to advertising data.

Syntax
uint8_t addFlags(uint8_t flags);

Parameters
flags: desired flags to add to advertising data. Valid values:
– GAP_ADTYPE_FLAGS_LIMITED
– GAP_ADTYPE_FLAGS_GENERAL
– GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_CONTROLLER
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_HOST

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::addPartialServices

Description
Add partial list of service UUIDs to advertising data.

Syntax
uint8_t addPartialServices(BLEUUID uuid);

Parameters
uuid: the desired UUID contained in BLEUUID class object.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addCompleteServices

Description
Add complete list of service UUIDs to advertising data.

Syntax
uint8_t addCompleteServices(BLEUUID uuid);
uint8_t addCompleteServices(uint8_t uuidBitLength);

Parameters
uuid: the desired UUID contained in BLEUUID class object.
uuidBitLength: UUID bit length for which a blank entry is to be added. Valid values: 16, 32, 128.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
uuidBitLength is used when it is desired to add a blank entry to the advertisement data, used to indicate that no services with UUIDs of a certain length are available.

BLEAdvertData::addAppearance

Description
Add device appearance to advertising data.

Syntax
uint8_t addAppearance(uint16_t appearance);

Parameters
appearance: the desired device appearance.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for a full list of device appearance values.

BLEAdvertData::addShortName

Description
Add shortened device name to advertising data.

Syntax
uint8_t addShortName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings

NA

BLEAdvertData::addCompleteName

Description
Add complete device name to advertising data.

Syntax
uint8_t addCompleteName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::parseScanInfo

Description
Parse advertising data received from a scan.

Syntax
void parseScanInfo(T_LE_CB_DATA *p_data);

Parameters
p_data: pointer to advertising data received from a Bluetooth scan.

Returns
The function returns nothing.

Example Code

Example: BLEBatteryClient

Notes and Warnings
Advertising data fields of parsed receive data can be access using member functions starting with “has” and “get”.

BLEAdvertData::hasFlags

Description
Check if received data includes advertising flags.

Syntax
bool hasFlags(void);

Parameters
The function requires no input parameter.

Returns
True if flags are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasUUID

Description
Check if received data includes service UUIDs.

Syntax
bool hasUUID(void);

Parameters
The function requires no input parameter.

Returns

True if service UUIDs are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasName

Description
Check if received data includes device name.

Syntax
bool hasName(void);

Parameters
The function requires no input parameter.

Returns
True if device name is present in received advertising data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::hasManufacturer

Description
Check if received data includes manufacturer specific data.

Syntax
bool hasManufacturer(void);

Parameters

The function requires no input parameter.

Returns
True if manufacturer specific data is present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAdvType

Description
Get advertising type of received data.

Syntax
T_GAP_ADV_EVT_TYPE getAdvType(void);

Parameters
The function requires no input parameter.

Returns
Advertising type of received advertising data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_ADV_EVT_TYPE_UNDIRECTED
– GAP_ADV_EVT_TYPE_DIRECTED
– GAP_ADV_EVT_TYPE_SCANNABLE
– GAP_ADV_EVT_TYPE_NON_CONNECTABEL
– GAP_ADV_EVT_TYPE_SCAN_RSP

BLEAdvertData::getAddrType

Description
Get Bluetooth address type of received data.

Syntax
T_GAP_REMOTE_ADDR_TYPE getAddrType(void);

Parameters
The function requires no input parameter.

Returns
Bluetooth address type of received data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

BLEAdvertData::getRSSI

Description
Get received signal strength indicator (RSSI) of received data.

Syntax
Int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
Received signal strength.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getFlags

Description
Get advertising flags of received data.

Syntax
uint8_t getFlags(void);

Parameters
The function requires no input parameter.

Returns
Advertising flags present in received advertising data, expressed as a single byte.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getServiceCount

Description
Get number of advertised services in received data.

Syntax
uint8_t getServiceCount(void);

Parameters
The function requires no input parameter.

Returns
Number of advertised service UUIDs in received data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getServiceList

Description
Get list of advertised service UUIDs in received data.

Syntax
BLEUUID* getServiceList(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a BLEUUID array containing all advertised service UUIDs.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getName

Description
Get advertised device name in received data.

Syntax
String getName(void);

Parameters
The function requires no input parameter.

Returns
Advertised device name contained in a String class object.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getTxPower

Description
Get advertised transmission power in received data.

Syntax
int8_t getTxPower(void);

Parameters
The function requires no input parameter.

Returns
Advertised transmission power.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAppearance

Description
Get advertised device appearance in received data.

Syntax
uint16_t getAppearance(void);

Parameters
The function requires no input parameter.

Returns
Advertised device appearance.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of device appearance values.

BLEAdvertData::getManufacturer

Description
Get advertised manufacturer in received data.

Syntax
uint16_t getManufacturer(void);

Parameters
The function requires no input parameter.

Returns
Advertised manufacturer.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of manufacturer codes.

BLEAdvertData::getManufacturerDataLength

Description
Get length of manufacturer data in received data.

Syntax
uint8_t getManufacturerDataLength(void);

Parameters
The function requires no input parameter.

Returns
Number of bytes of manufacturer data present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getManufacturerData

Description
Get manufacturer data in received data.

Syntax
uint8_t* getManufacturerData(void);

Parameters
The function requires no input parameter.

Returns
Pointer to array containing manufacturer data.

Example Code
NA

Notes and Warnings
NA

Class BLEBeacon

iBeacon Class

Description
A class used for managing iBeacon BLE advertising data.

Syntax
class iBeacon

Members

Public Constructors	
iBeacon::iBeacon	Create an instance of iBeacon advertising data
Public Methods	
iBeacon::getManufacturerId	Get current manufacturer ID value
iBeacon::getUUID	Get current UUID value
iBeacon::getMajor	Get current Major value
iBeacon::getMinor	Get current Minor value
iBeacon::getRSSI	Get current RSSI value
iBeacon::setManufacturerId	Set manufacturer ID value
iBeacon::setUUID	Set UUID value
iBeacon::setMajor	Set Major value
iBeacon::setMinor	Set Minor value
iBeacon::setRSSI	Set RSSI value
iBeacon::getAdvData	Get current advertising data
iBeacon::getScanRsp	Get current scan response data

altBeacon Class

Description
A class used for managing altBeacon BLE advertising data.

Syntax
class altBeacon

Members

Public Constructors	
altBeacon::altBeacon	Create an instance of altBeacon advertising data
Public Methods	
altBeacon::getManufacturerId	Get current manufacturer ID value
altBeacon::getUUID	Get current UUID value
altBeacon::getMajor	Get current Major value
altBeacon::getMinor	Get current Minor value
altBeacon::getRSSI	Get current RSSI value
altBeacon::getRSVD	Get current Reserved value
altBeacon::setManufacturerId	Set manufacturer ID value
altBeacon::setUUID	Set UUID value
altBeacon::setMajor	Set Major value
altBeacon::setMinor	Set Minor value
altBeacon::setRSSI	Set RSSI value
altBeacon::setRSVD	Set Reserved value
altBeacon::getAdvData	Get current advertising data
altBeacon::getScanRsp	Get current scan response data

iBeacon::iBeacon

Description
Create an iBeacon object.

Syntax
void iBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

altBeacon::altBeacon

Description
Create an altBeacon object.

Syntax
void altBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

iBeacon::getManufacturerId

altBeacon::getManufacturerId

Description
Get current Manufacturer ID value.

Syntax
uint16_t getManufacturerId(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Company ID.

Example Code
NA

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::getUUID

altBeacon::getUUID

Description
Get the current UUID value.

Syntax
void getUUID(uint8_t* UUID);

Parameters
UUID: pointer to a 16 element uint8_t array, current UUID will be copied into the array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::getMajor

altBeacon::getMajor

Description
Get current Major value.

Syntax
uint16_t getMajor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Major value.

Example Code

NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getMinor

altBeacon::getMinor

Description
Get current Minor value.

Syntax
uint16_t getMinor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Minor value.

Example Code
NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getRSSI

altBeacon::getRSSI

Description
Get the current RSSI value.

Syntax
int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit signed integer containing the currently set RSSI value.

Example Code
NA

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setManufacturerId

altBeacon::setManufacturerId

Description
Set Manufacturer ID value.

Syntax
void setManufacturerId(uint16_t id);

Parameters
id: desired Manufacturer ID

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::setUUID

altBeacon::setUUID

Description
Set UUID value.

Syntax
void setUUID(uint8_t* UUID);
void setUUID(const char* UUID);

Parameters
uint8_t* UUID: pointer to a 16 element uint8_t array containing the desired UUID
const char* UUID: desired UUID expressed as a character string

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::setMajor

altBeacon::setMajor

Description
Set Major value.

Syntax
void setMajor(uint16_t major);

Parameters
major: desired Major value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setMinor

altBeacon::setMinor

Description
Set Minor value.

Syntax
void setMinor(uint16_t minor);

Parameters
minor: desired Minor value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setRSSI

altBeacon::setRSSI

Description
Set RSSI value.

Syntax
void setRSSI(int8_t RSSI);

Parameters
RSSI: desired RSSI value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getAdvData

altBeacon::getAdvData

Description
Get current beacon advertising data.

Syntax
uint8_t* getAdvData(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

iBeacon::getScanRsp

altBeacon::getScanRsp

Description
Get current beacon advertising scan response data.

Syntax
uint8_t* getScanRsp(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising scan response data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

altBeacon::getRSVD

Description
Get current Reserved value.

Syntax
uint8_t getRSVD(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit unsigned integer containing the current Reserved value.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

altBeacon::setRSVD

Description
Set Reserved value.

Syntax
void setRSVD(uint8_t rsvd);

Parameters
rsvd: desired Reserved value

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

Class BLECharacteristic

BLECharacteristic Class

Description

A class used for creating and managing BLE GATT characteristics.

Syntax
class BLECharacteristic

Members

Public Constructors	
BLEC haracteristic::BLECharacteristic	Constructs a BLECharacteristic object
Public Methods	
BLECharacteristic::setUUID	Set the characteristic UUID
BLECharacteristic::getUUID	Get the characteristic UUID
BLECharacteristic::setBufferLen	Set the size of the internal data buffer
BLECharacteristic::getBufferLen	Get the current size of the internal data buffer
BL ECharacteristic::setReadProperty	Get the current size of the internal data bufferSet the characteristic r
BLE Characteristic::setWriteProperty	Set the characteristic write property
BLEC haracteristic::setNotifyProperty	Set the characteristic notify property
BLECha racteristic::setIndicateProperty	Set the characteristic indicate property

continues o

Table 6 – continued from previous page

Public Constructors	
BLECharacteristic::setProperties	Set the characteristic properties
BLECharacteristic::getProperties	Get the characteristic properties
BLECharacteristic::readString	Read the characteristic data buffer as a String object
BLECharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLECharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLECharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLECharacteristic::writeString	Write data to the characteristic data buffer as a String object or chara
BLECharacteristic::writeData8	Write data to the characteristic data buffer as an unsigned 8-bit integ
BLECharacteristic::writeData16	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::writeData32	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::setData	Write data to the characteristic data buffer
BLECharacteristic::getData	Read data from the characteristic data buffer
BLECharacteristic::getDataBuff	Get a pointer to the characteristic data buffer
BLECharacteristic::getDataLen	Get the number of bytes of data in the characteristic data buffer
BLECharacteristic::notify	Send a notification to a connected device
BLECharacteristic::indicate	Send an indication to a connected device
BLEC haracteristic::setUserDescriptor	Add a user description descriptor to characteristic
BLECha racteristic::setFormatDescriptor	Add a data format descriptor to characteristic
BLECharacteristic::Add a data format descriptor to characteristic	Set a user function as a read callback
BLE Characteristic::setWriteCallback	Set a user function as a write callback
BL ECharacteristic::setCCCDCallback	Set a user function as a CCCD write callback

BLECharacteristic::BLECharacteristic

Description
Constructs a BLECharacteristic object.

Syntax
BLECharacteristic::BLECharacteristic(BLEUUID uuid);
BLECharacteristic::BLECharacteristic(const char* uuid);

Parameters
uuid: characteristic UUID, expressed as a BLEUUID class object or a character array

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setUUID

Description
Set the characteristic UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: the new characteristic UUID, expressed with a BLEUUID class object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLECharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadProperty

Description
Set the characteristic read property.

Syntax
void setReadProperty(bool value);

Parameters
value: TRUE to allow connected devices to read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteProperty

Description
Set the characteristic write property.

Syntax
void setWriteProperty(bool value);

Parameters
value: TRUE to allow connected devices to write characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setNotifyProperty

Description
Set the characteristic notify property.

Syntax
void setNotifyProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data notifications.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setIndicateProperty

Description
Set the characteristic indicate property.

Syntax
void setIndicateProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data indications.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setProperties

Description
Set the characteristic properties.

Syntax
void setProperties(uint8_t value);

Parameters
value: desired characteristic properties

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getProperties

Description
Get the currently set characteristic properties.

Syntax
uint8_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set characteristic properties expressed as an unsigned 8-bit integer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readString

Description
Read the data in the characteristic internal buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a String class object.

Example Code
Example: BLEUartService

Notes and Warnings
Non-ASCII data may result in unexpected characters in the string.

BLECharacteristic::readData8

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData16

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Write data to the characteristic data buffer as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the characteristic buffer, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::writeData8

Description
Write data to the characteristic data buffer as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEBatteryService

Notes and Warnings

NA

BLECharacteristic::writeData16

Description
Write data to the characteristic data buffer as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::writeData32

Description
Write data to the characteristic data buffer as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setData

Description
Write data to the characteristic data buffer.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getData

Description
Read data from the characteristic data buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLECharacteristic::getDataBuff

Description
Get a pointer to the characteristic data buffer.

Syntax
uint8_t* getDataBuff();

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to the uint8_t array used as the characteristic internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getDataLen

Description
Get the number of bytes of data in the characteristic data buffer.

Syntax
uint16_t getDataLen

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes of data in the internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::notify

Description
Send a notification to a connected device.

Syntax
void notify(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send a notification to.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::indicate

Description
Send an indication to a connected device.

Syntax
void indicate(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send an indication to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setUserDescriptor

Description
Add a user description descriptor attribute (UUID 0x2901) to the characteristic.

Syntax
void setUserDescriptor(const char* description);

Parameters
description: the desired user description string expressed in a char array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setFormatDescriptor

Description
Add a data format descriptor attribute (UUID 0x2904) to the characteristic.

Syntax
void setFormatDescriptor(uint8_t format, uint8_t exponent, uint16_t unit, uint16_t description);

Parameters
format: refer to https://www.bluetooth.com/specifications/assigned-numbers/format-types/ for the valid values and associated format types.
exponent: base-10 exponent to be applied to characteristic data value.

unit: refer to https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf for the valid values and associated units.
descriptor: refer to https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-descriptors/ for the valid values and associated descriptors.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadCallback

Description
Set a user function to be called when the characteristic data is read by a connected device.

Syntax
void setReadCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing data read
conn_id: connection ID of connected device that read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteCallback

Description

Set a user function to be called when the characteristic data is written by a connected device.

Syntax
void setWriteCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setCCCDCallback

Description
Set a user function to be called when a connected device modifies the characteristic CCCD to enable or disable notifications or indications.

Syntax
void setCCCDCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id, uint16_t ccc_bits));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.
ccc_bits: the new CCCD data bits after modification by the connected device

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

Class BLEClient

BLEClient Class

Description
A class used for discovering and accessing BLE GATT services on a connected remote device.

Syntax
class BLEClient

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLEDevice::addClient().

Public Methods	
BLEClient::connected	Check if the corresponding remote device for the client is connected
BLEClient::discoverServices	Start service discovery process for connected device
BLEClient::discoveryDone	Determine if service discovery process has been completed
BLEClient::printServices	Format and print discovered services to serial port
BLEClient::getService	Get a specific service on the remote device
BLEClient::getConnId	
BLEClient::getClientId	Get corresponding client ID
BLEClient::setDisconnectCallback	Set a user function to be called when the remote device is disconnected

BLEClient::connected

Description
Check if the remote device associated with the client is still connected.

Syntax
bool connected();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the remote device is connected.

Example Code
NA

Notes and Warnings
NA

BLEClient::discoverServices

Description
Start the service discovery process for the connected remote device.

Syntax
void discoverServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::discoveryDone

Description
Check if the service discovery process has been completed.

Syntax
bool discoveryDone();

Parameters
The function requires no input parameter.

Returns
TThe function returns TRUE if the service discovery process has been completed successfully, FALSE if the service discovery process failed, is still in progress, or has yet to start.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::printServices

Description
Print out a formatted list of discovered services to the serial port.

Syntax
void printServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEClient::getService

Description
Get a service with the specified UUID on the remote device.

Syntax
BLERemoteService* getService(const char* uuid);
BLERemoteService* getService(BLEUUID uuid);

Parameters
uuid: the desired service UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found service as a BLERemoteService object pointer, otherwise nullptr is returned if a service with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::getConnId

Description
Get the connection ID associated with the remote device.

Syntax
uint8_t getConnId;

Parameters
The function requires no input parameter.

Returns
The function returns the connection ID for the connected remote device.

Example Code
NA

Notes and Warnings
NA

BLEClient::getClientId

Description
Get the client ID for the BLEClient object.

Syntax
T_CLIENT_ID getClientId();;

Parameters
The function requires no input parameter.

Returns
The function returns the BLEClient object's client ID.

Example Code
NA

Notes and Warnings
The client ID is used when calling internal GATT client API.

BLEClient::setDisconnectCallback

Description
Set a user function as a callback function when the remote device is disconnected.

Syntax
void setDisconnectCallback(void (*fCallback) (BLEClient* client));

Parameters
fCallback: A user callback function that returns void and takes one argument.
client: A pointer to the BLEClient object corresponding to the disconnected remote device

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The user callback function will be called after the remote device has disconnected, before the characteristics, services and client associated with the remote device are deleted.

Class BLEConnect

BLEConnect Class

Description
A class used for managing BLE connection settings.

Syntax
class BLEConnect

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configConnection.

Public Methods	
BLEConnect::connect	Connect to a target BLE device
BLEConnect::disconnect	Disconnect from a target BLE device
BLEConnect::setScanInterval	Set the BLE scanning interval when connecting
BLEConnect::setScanWindow	Set the BLE scanning window when connecting
BLEConnect::setConnInterval	Set the BLE connection interval duration
BLEConnect::setConnLatency	Set the BLE connection slave latency
BLEConnect::setConnTimeout	Set the BLE connection timeout value
BLEConnect::updateConnParams	Send new BLE connection parameters to a connected device
BLEConnect::getConnInfo	Get connection information
BLEConnect::getConnAddr	Get the Bluetooth address for a certain connection
BLEConnect::getConnId	Get the connection ID for a certain device

BLEConnect::connect

Description
Connect to a target BLE device.

Syntax
bool connect(char* btAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(uint8_t (&btAddr)[6], T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(BLEAdvertData targetDevice, uint16_t scanTimeout);
bool connect(BLEAddr destAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t (&btAddr): target device Bluetooth address contained in a 6 byte array.
destAddr: target device Bluetooth address contained in BLEAddr class object.

targetDevice: advertising data packet scanned from target device.
destAddrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM
scan timeout: duration in milliseconds for which to look for target device before giving up.

Returns
True if connection successful, false if connection failed.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEConnect::disconnect

Description
Disconnect from a target BLE device.

Syntax
bool disconnect(uint8_t connId);

Parameters
connId: connection ID for target device.

Returns
True if operation successful, false if otherwise.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanInterval

Description

Set the BLE scan interval when searching for a target device to connect to.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanWindow

Description
Set the BLE scan window when searching for a target device to connect to.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setConnInterval

Description
Set the BLE connection interval value.

Syntax
void setConnInterval(uint16_t min_ms, uint16_t max_ms);

Parameters
min_ms: minimum acceptable connection interval in milliseconds. Value range of 8 to 4000.
max_ms: maximum acceptable connection interval in milliseconds. Value range of 8 to 4000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection interval defines the period between successive connection events between a connected central and peripheral device. Even if there is no data to exchange, a connection event is required to maintain the connection. max_ms should be larger than or equal to min_ms.

BLEConnect::setConnLatency

Description
Set the BLE connection slave latency value.

Syntax
void setConnLatency(uint16_t latency);

Parameters
latency: Connection slave latency value. Value range of 0 to 499.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection slave latency defines the number of successive connection events a connected peripheral device can ignore without being considered as disconnected by the central device.

BLEConnect::setConnTimeout

Description
Set the BLE connection timeout value.

Syntax
void setConnTimeout(uint16_t timeout_ms);

Parameters
timeout_ms: connection timeout in milliseconds. Value range of 100 to 32000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection timeout defines the duration after a failed connection events before a peripheral or central device considers the connection broken.

BLEConnect::updateConnParams

Description
Update a connected device with new connection parameters.

Syntax
void updateConnParams(uint8_t conn_id);

Parameters
conn_id: connection ID of target device to update connection parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Update a connected device with previously set connection interval, slave latency and timeout values. The connected device may reject the new values if it is unable to conform to them.

BLEConnect::getConnInfo

Description
Get connection information.

Syntax
bool getConnInfo(uint8_t connId, T_GAP_CONN_INFO *pConnInfo);

Parameters
connId: connection ID to get connection information from.
pConnInfo: pointer to T_GAP_CONN_INFO structure to store obtained connection information.

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnAddr

Description
Get the Bluetooth address for a certain connection.

Syntax
bool getConnAddr(uint8_t connId, uint8_t* addr, uint8_t* addrType);

Parameters
connId: connection ID to get address information for

addr: pointer to 6 byte array to store retrieved Bluetooth address
addrType: pointer to uint8_t variable to store retrieved Bluetooth address type

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnId

Description
Get the connection ID for a certain device.

Syntax
int8_t getConnId(char* btAddr, uint8_t addrType);
int8_t getConnId(uint8_t* btAddr, uint8_t addrType);
int8_t getConnId(BLEAdvertData targetDevice);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t* btAddr: pointer to a 6 byte array containing target device Bluetooth address.
targetDevice: advertising data packet scanned from target device.
addrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

Returns
The function returns the requested connection ID. Returns -1 if failed to obtain connection ID.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

Class BLEDevice

BLEDevice Class

Description
A class used for general control and management of BLE functions.

Syntax
class BLEDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLE.

Public Methods	
BLEDevice::init	Allocate resources required for BLE functionality
BLEDevice::deinit	Free resources used by BLE functionality
BLEDevice::connected	Check if a BLE device is connected
BLEDevice::setDeviceName	Set BLE GAP device name
BLEDevice::setDeviceAppearance	Set BLE GAP device appearance
BLEDevice::configAdvert	Configure BLE advertising parameters
BLEDevice::configScan	Configure BLE scan parameters
BLEDevice::setScanCallback	Set callback function for BLE scans
BLEDevice::beginCentral	Start BLE stack in central mode
BLEDevice::beginPeripheral	Start BLE stack in peripheral mode
BLEDevice::end	Stop BLE stack
BLEDevice::configServer	Configure BLE stack for services
BLEDevice::addService	Add a service to the BLE stack
BLEDevice::configClient	Configure BLE stack for clients
BLEDevice::addClient	Add a client to the BLE stack
BLEDevice::getLocalAddr	Get local device Bluetooth address

BLEDevice::init

Description
Allocate resources required for BLE functionality.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this member function first before using any other member functions in the BLEDevice class.

BLEDevice::deinit

Description
Free up resources used for BLE functionality.

Syntax
void deinit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Call this member function last after all other BLE operations are stopped.

BLEDevice::connected

Description
Check if a BLE device is connected.

Syntax
bool connected(void);

Parameters
The function requires no input parameter.

Returns
TRUE if another BLE device is connected, FALSE if no BLE device is connected.

Example Code
NA

Notes and Warnings
NA

BLEDevice::setDeviceName

Description
Set the BLE GAP device name.

Syntax
void setDeviceName(String devName);

Parameters
devName: desired device name contained in an Arduino String object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The GAP device name has a maximum length of 39 characters. Other devices can see this name after a BLE connection is established. This name is separate and different from the device name sent in a BLE advertisement, the names should be the same but are not required.

BLEDevice::setDeviceAppearance

Description
Set the BLE GAP device appearance.

Syntax
void setDeviceAppearance(uint16_t devAppearance);

Parameters
devAppearance: desired device appearance expressed as a 16-bit unsigned integer.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to Bluetooth SIG assigned device appearances at https://www.bluetooth.com/specifications/gatt/characteristics/.

BLEDevice::configAdvert

Description
Configure BLE advertising parameters.

Syntax
BLEAdvert* configAdvert(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEAdvert class instance for configuring BLE advertising parameters.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function instead of creating a BLEAdvert class instance manually.

BLEDevice::configScan

Description
Configure BLE scanning parameters.

Syntax
BLEScan* configScan(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEScan class instance for configuring BLE scanning parameters.

Example Code
Example: BLEScan

```
#include “BLEDevice.h”
#include “BLEScan.h”
int dataCount = 0;
void scanFunction(T_LE_CB_DATA* p_data) {
printf(”rnScan Data %drn”, ++dataCount);
BLE.configScan()->printScanInfo(p_data);
}
void setup() {
BLE.init();
BLE.configScan()->setScanMode(GAP_SCAN_MODE_ACTIVE);
BLE.configScan()->setScanInterval(500); // Start a scan every 500ms
BLE.configScan()->setScanWindow(250); // Each scan lasts for 250ms
// Provide a callback function to process scan data.
// If no function is provided, default BLEScan::printScanInfo is used
BLE.setScanCallback(scanFunction);
BLE.beginCentral(0);
BLE.configScan()->startScan(5000); // Repeat scans for 5 seconds, then stop
}
void loop() {
}
```

Notes and Warnings

Use this member function instead of creating a BLEScan class instance manually.

BLEDevice::setScanCallback

Description
Set a callback function for processing BLE scan results.

Syntax
void setScanCallback(void (*scanCB)(T_LE_CB_DATA));

Parameters
scanCB: a function that returns nothing and takes in a scan data pointer of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to set a callback function that will be called for each BLE device scan result found.

BLEDevice::beginCentral

Description
Start the BLE stack in central mode.

Syntax
void beginCentral(uint8_t connCount);

Parameters
connCount: maximum number of allowed connected devices. If no argument is provided, default to maximum allowed connected devices for specific board.

Returns
The function returns nothing.

Example Code
Example: BLEScan
The function returns nothing.

Notes and Warnings
Use this member function to start the device in BLE central mode, after other BLE parameters are set correctly.

BLEDevice::beginPeripheral

Description
Start the BLE stack in peripheral mode.

Syntax
void beginPeripheral(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function to start the device in BLE peripheral mode, after other BLE parameters are set correctly.

BLEDevice::end

Description
Stop the BLE stack.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Use this member function to stop the device operating in either BLE peripheral mode or BLE central mode.

BLEDevice::configServer

Description
Configure the BLE stack for services.

Syntax
void configServer(uint8_t maxServiceCount);

Parameters
maxServiceCount: Maximum number of services that will run on the device

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function before adding any service to the BLE stack.

BLEDevice::addService

Description
Add a new service to the BLE stack.

Syntax
void addService(BLEService& newService);

Parameters
newService: the service to be added, defined using a BLEService class object.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
N/A

BLEDevice::configClient

Description
Configure the BLE stack for clients.

Syntax
void configClient();

Parameters
The function requries no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Use this member function before adding any client to the BLE stack.

BLEDevice::addClient

Description
Add a new client to the BLE stack.

Syntax
BLEClient* addClient(uint8_t connId);

Parameters
connId: the connection ID of the connected device to create a client for.

Returns
The function returns a pointer to a BLEClient class object, corresponding to the device with the specified connection ID, which can be used to access the services and characteristics on the connected device.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Only one client should be added per connected device.
The BLEClient object and any service, characteristic, descriptor associated with the connected device will be deleted when the device is disconnected.

BLEDevice::getLocalAddr

Description
Get local device Bluetooth address.

Syntax
void getLocalAddr(uint8_t (&addr)[GAP_BD_ADDR_LEN]);

Parameters
addr: 6 byte array to store local device Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Local device address is only available after starting in central or peripheral mode. This function will return all zeros for the address if central or peripheral mode is not in operation.

Class BLEHIDDevice

BLEHIDDevice Class

Description

A class used for creating and managing HID over GATT Profile (HOGP) services.

Syntax
class BLEHIDDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLEHIDDev.

Public Methods	
BLEHIDDevice::init	Initialize the HID Device Profile by creating the required services
BLEHIDD evice::setNumOutputReport	Configure the number of HID output reports
BLEHID Device::setNumInputReport	Configure the number of HID input reports
B LEHIDDevice::setReportMap	Configure the HID report map
BLEHIDDevice::inputReport	Send a HID input report
BLEHIDDevice ::setOutputReportCallback	Set a user callback function for receiving HID output reports
BLEHIDD evice::bootKeyboardReport	Send a HID boot keyboard input report
BLEHIDDevice::setHidInfo	Set HID info of the HID service
B LEHIDDevice::setBattLevel	Set battery level info of the Battery service
BLEHIDDevice::setPNPInfo	Set PNP information of the Device Information service
BLEHIDDevi ce::setManufacturerString	Set manufacturer information of the Device Information service
BLE HIDDevice::setModelString	Set model information of the Device Information service
BLEHIDDevice::hidService	Get reference to HID service
BLE HIDDevice::devInfoService	Get reference to Device Information service
BLEHIDDevice::battService	Get reference to Battery service

BLEHIDDevice::init

Description
Initialize the HID Device profile by creating the required services.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID Device object should be initialized before any HID reports can be sent.

BLEHIDDevice::setNumOutputReport

Description
Configure the number of HID output reports.

Syntax
void setNumOutputReport (uint8_t numOutputReports);

Parameters
numOutputReports: number of output reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of output reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setNumInputReport

Description
Configure the number of HID input reports.

Syntax
void setNumInputReport (uint8_t numInputReports);

Parameters
numInputReports: number of input reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of input reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setReportMap

Description
Configure the HID report map characteristic with a HID report descriptor.

Syntax
void setReportMap (uint8_t* report_map, uint16_t len);

Parameters
report_map: pointer to HID report descriptor
len: HID report descriptor length in bytes

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID report map characteristic can only be configured after BLEHIDDevice init() function is called.

BLEHIDDevice::inputReport

Description
Send a HID input report.

Syntax
void inputReport (uint8_t reportID, uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters

reportID: HID report ID of input report
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
HID input reports can only be sent after BLEHIDDevice init() function has been called.

BLEHIDDevice::setOutputReportCallback

Description
Set a user callback function for receiving HID output report data.

Syntax
void setOutputReportCallback (uint8_t reportID, void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
reportID: HID report ID of output report to link callback function with
chr: BLECharacteristic class object containing received HID output report data
conn_id: connection ID of device which sent HID report data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Setting a user callback function for output reports can only occur after BLEHIDDevice init() function has been called.

BLEHIDDevice::bootKeyboardReport

Description
Send a HID boot keyboard input report.

Syntax
void bootKeyboardReport (uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, the HID service Protocol Mode characteristic has boot mode disabled. To send boot keyboard input reports, the Protocol Mode characteristic needs to have boot mode enabled.

BLEHIDDevice::setHidInfo

Description
Set data of the HID Info characteristic of the HID service.

Syntax
void setHidInfo (uint16_t bcd, uint8_t country, uint8_t flags);

Parameters
bcd: 16-bit unsigned integer representing version number of base USB HID Specification implemented by HID Device
country: 8-bit integer identifying country HID Device hardware is localized for. Most hardware is not localized (value 0x00).
flags: Bit flags indicating remote-wake capability and advertising when bonded but not connected.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
For detailed information on the characteristic, refer to Bluetooth SIG HID Service specifications.

BLEHIDDevice::setBattLevel

Description
Set battery level data of the Battery service.

Syntax
void setBattLevel (uint8_t level);

Parameters
level: battery level expressed as % of full charge

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Battery level is set to 100% by default. For detailed information refer to Bluetooth SIG Battery service specifications.

BLEHIDDevice::setPNPInfo

Description
Set PNP data of the Device Information service.

Syntax
void setPNPInfo (uint8_t sig, uint16_t vid, uint16_t pid, uint16_t version);

Parameters
sig: The Vendor ID Source field designates which organization assigned the value used in the Vendor ID field value.
vid: The Vendor ID field is intended to uniquely identify the vendor of the device.
pid: The Product ID field is intended to distinguish between different products made by the vendor.
version: The Product Version field is a numeric expression identifying the device release number in Binary-Coded Decimal.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, sig and vid are configured to indicate Realtek as the vendor. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setManufacturerString

Description
Set manufacturer information of the Device Information service.

Syntax
void setManufacturerString (const char* manufacturer);

Parameters
manufacturer: pointer to character string containing manufacturer name info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Manufacturer is set to “Realtek” by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setModelString

Description
Set model information of the Device Information service.

Syntax
void setModelString (const char* model);

Parameters
model: pointer to character string containing device model info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Model is set to "Ameba_BLE_HID" by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::hidService

Description
Get reference to HID service

Syntax
BLEService& hidService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the HID service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::devInfoService

Description
Get reference to Device Information service

Syntax
BLEService& devInfoService ();

Parameters
The function requires no input parameter.

Returns

The function returns a reference to the BLEService class object for the Device Information service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::battService

Description
Get reference to Battery service

Syntax
BLEService& battService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the Battery service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLEHIDGamepad

BLEHIDGamepad Class

Description

A class used for creating and managing a BLE HID Gamepad.

Syntax
class BLEHIDGamepad

Members

Public Constructors	
BLEHIDGame pad::BLEHIDGamepad	Constructs a BLEHIDGamepad object
Public Methods	
BLEHIDGa mepad::setReportID	Set HID report ID for the HID Gamepad
BLEHIDGame pad::gamepadReport	Send a HID Gamepad report
BLEHIDGa mepad::buttonPress	Send a HID Gamepad report indicating buttons pressed
BLEHIDGame pad::buttonRelease	Send a HID Gamepad report indicating buttons released
BLEHIDGamepad ::buttonReleaseAll	Send a HID Gamepad report indicating no buttons pressed
BLE HIDGamepad::setHat	Send a HID Gamepad report indicating hat switch position
BLEH IDGamepad::setAxes	Send a HID Gamepad report indicating position of all axes
BLEHIDGam epad::setLeftStick	Send a HID Gamepad report indicating position of axes corresponding to left analog stick
BLEHIDGame pad::setRightStick	Send a HID Gamepad report indicating position of axes corresponding to right analog stick
BLEHIDGa mepad::setTriggers	Send a HID Gamepad report indicating position of axes corresponding to triggers

BLEHIDGamepad::BLEHIDGamepad

Description
Constructs a BLE object

Syntax
BLEHIDGamepad::BLEHIDGamepad();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
By default, the BLEHIDGamepad class assumes the HID report descriptor implements a gamepad device with 16 buttons, 6 16-bit axes and an 8-direction hat switch. This class will not work if a different gamepad report descriptor is implemented.

BLEHIDGamepad::setReportID

Description
Set HID report ID for the HID Gamepad.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the gamepad device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
HID report ID should start at 1. Some systems may consider a report ID of 0 as invalid.

BLEHIDGamepad::gamepadReport

Description
Send a HID Gamepad report.

Syntax
void gamepadReport (hid_gamepad_report_t* report);
void gamepadReport (uint16_t buttons, uint8_t hat, int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
report: pointer to gamepad report structure containing data on all inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::buttonPress

Description
Send a HID Gamepad report indicating buttons pressed.

Syntax
void buttonPress (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonRelease

Description
Send a HID Gamepad report indicating buttons released.

Syntax
void buttonRelease (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonReleaseAll

Description
Send a HID Gamepad report indicating no buttons pressed.

Syntax
void buttonReleaseAll (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setHat

Description
Send a HID Gamepad report indicating hat switch position.

Syntax
void setHat (uint8_t hat);

Parameters
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setAxes

Description
Send a HID Gamepad report indicating position of all axes.

Syntax
void setAxes (int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setLeftStick

Description
Send a HID Gamepad report indicating position of axes corresponding to left analog stick.

Syntax
void setLeftStick (int16_t x, int16_t y);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setRightStick

Description
Send a HID Gamepad report indicating position of axes corresponding to right analog stick.

Syntax
void setLeftStick (int16_t z, int16_t Rz);

Parameters
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setTriggers

Description
Send a HID Gamepad report indicating position of axes corresponding to triggers.

Syntax
void setTriggers (int16_t Rx, int16_t Ry);

Parameters
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

Class BLEHIDKeyboard

BLEHIDKeyboard Class

Description

A class used for creating and managing a BLE HID Keyboard.

Syntax
class BLEHIDKeyboard

Members

Public Constructors	
BLEHIDKeybo ard::BLEHIDKeyboard	Constructs a BLEHIDKeyboard object
Public Methods	
BLEHIDKe yboard::setReportID	Set HID report ID for the HID Keyboard and HID consumer control
BLEHIDKeybo ard::consumerReport	Send a HID Consumer report
BLEHIDKeybo ard::keyboardReport	Send a HID Keyboard report
BLEHIDKeyb oard::consumerPress	Send a HID Consumer report indicating button pressed
BLEHIDKeyboa rd::consumerRelease	Send a HID Consumer report indicating button released
BLEHI DKeyboard::keypress	Send a HID Keyboard report indicating keys pressed
BLEHIDK eyboard::keyRelease	Send a HID Keyboard report indicating keys released
BLEHIDKeyb oard::keyReleaseAll	Send a HID Keyboard report indicating no keys pressed
BLEHIDKey board::keyCharPress	Send a HID Keyboard report indicating keys pressed to output an ASCII character
BLEHIDKe yboard::keySequence	Send a HID Keyboard report indicating keys pressed to output an ASCII string

BLEHIDKeyboard::BLEHIDKeyboard

Description
Constructs a BLEHIDKeyboard object.

Syntax
BLEHIDKeyboard::BLEHIDKeyboard();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::setReportID

Description
Set HID report ID for the HID Keyboard and HID consumer control.

Syntax
void setReportID (uint8_t reportIDKeyboard, uint8_t reportIDConsumer);

Parameters
reportIDKeyboard: The report ID for the HID keyboard device, corresponding to the HID report descriptor.
reportIDConsumer: The report ID for the HID consumer control device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerReport

Description
Send a HID Consumer report.

Syntax
void consumerReport (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyboardReport

Description
Send a HID Keyboard report.

Syntax
void keyboardReport (void);
void keyboardReport (uint8_t modifiers, uint8_t keycode[6]);

Parameters
modifiers: bitmap indicating key modifiers pressed (CTRL, ALT, SHIFT).
keycode: byte array indicating keys pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerPress

Description
Send a HID Consumer report indicating button pressed.

Syntax
void consumerPress (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerRelease

Description
Send a HID Consumer report indicating button released.

Syntax
void consumerRelease (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keypress

Description
Send a HID Keyboard report indicating keys pressed.

Syntax
void keyPress (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyRelease

Description
Send a HID Keyboard report indicating keys released.

Syntax
void keyRelease (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyReleaseAll

Description
Send a HID Keyboard report indicating no keys pressed.

Syntax
void keyReleaseAll(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyCharPress

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII character.

Syntax
void keyCharPress (char ch);

Parameters
ch: ASCII character to output.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keySequence

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII string.

Syntax
void keySequence (const char* str, uint16_t delayTime);
void keySequence (String str, uint16_t delayTime);

Parameters
str: pointer to character string to output
str: String object containing character string to output

delayTime: time delay between key press and release, in milliseconds. Default value of 5.

Returns
The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

Class BLEHIDMouse

BLEHIDMouse Class

Description

A class used for creating and managing a BLE HID Mouse.

Syntax
class BLEHIDMouse

Members

Public Constructors	
BLE HIDMouse::BLEHIDMouse	Constructs a BLEHIDMouse object
Public Methods	
BLE HIDMouse::setReportID	Set HID report ID for the HID Mouse
BLE HIDMouse::mouseReport	Send a HID Mouse report
BL EHIDMouse::mousePress	Send a HID Mouse report indicating buttons pressed
BLEH IDMouse::mouseRelease	Send a HID Mouse report indicating buttons released
BLEHIDM ouse::mouseReleaseAll	Send a HID Mouse report indicating no buttons pressed
B LEHIDMouse::mouseMove	Send a HID Mouse report indicating mouse movement
BLE HIDMouse::mouseScroll	Send a HID Mouse report indicating mouse scroll wheel movement

BLEHIDMouse::BLEHIDMouse

Description
Constructs a BLEHIDMouse object.

Syntax
BLEHIDMouse::BLEHIDMouse();

Parameters

The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::setReportID

Description
Set HID report ID for the HID Mouse.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the HID mouse device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseReport

Description
Send a HID Mouse report.

Syntax
void mouseReport (hid_mouse_report_t* report);
void mouseReport (uint8_t buttons, int8_t x, int8_t y, int8_t scroll);

Parameters
report: pointer to mouse report structure containing data on mouse inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mousePress

Description
Send a HID Mouse report indicating buttons pressed.

Syntax
void mousePress (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseRelease

Description
Send a HID Mouse report indicating buttons released.

Syntax
void mouseRelease (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseReleaseAll

Description
Send a HID Mouse report indicating no buttons pressed.

Syntax
void mouseReleaseAll(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseMove

Description
Send a HID Mouse report indicating mouse movement.

Syntax
void mouseMove (int8_t x, int8_t y);

Parameters
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseScroll

Description
Send a HID Mouse report indicating mouse scroll wheel movement.

Syntax
void mouseScroll (int8_t scroll);

Parameters
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLERemoteCharacteristic

BLERemoteCharacteristic Class

Description
A class used for managing BLE GATT characteristics on connected remote devices.

Syntax
class BLERemoteCharacteristic

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteService::getCharacteristic().

Public Methods	
BLERem oteCharacteristic::getDescriptor	Get a specific descriptor on the remote device
BLERemoteCharacteristic::getUUID	Get the characteristic UUID
BLERe moteCharacteristic::setBufferLen	Set the size of the internal data buffer
BLERe moteCharacteristic::getBufferLen	Get the current size of the internal data buffer
BLERemoteCharacteristic::canRead	Determine if characteristic has read property enabled
B LERemoteCharacteristic::canWrite	Determine if characteristic has write property enabled
BL ERemoteCharacteristic::canNotify	Determine if characteristic has notify property enabled
BLER emoteCharacteristic::canIndicate	Determine if characteristic has indicate property enabled
BLERem oteCharacteristic::getProperties	Get the characteristic properties
BLE RemoteCharacteristic::readString	Read the characteristic data buffer as a String object
BL ERemoteCharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLE RemoteCharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLE RemoteCharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLER emoteCharacteristic::writeString	Write data to the characteristic as a String object or character array
BLE RemoteCharacteristic::writeData8	Write data to the characteristic as an unsigned 8-bit integer
BLER emoteCharacteristic::writeData16	Write data to the characteristic as an unsigned 16-bit integer
BLER emoteCharacteristic::writeData32	Write data to the characteristic as an unsigned 16-bit integer
BLERemoteCharacteristic::setData	Write data to the characteristic
BLERemoteCharacteristic::getData	Read data from the characteristic
BLERemoteChar acteristic::enableNotifyIndicate	Enable notification or indication for the characteristic
BLERemoteChara cteristic::disableNotifyIndicate	Disable notification and indication for the characteristic
BLERemoteC haracteristic::setNotifyCallback	Set a user function as a notification callback

BLERemoteCharacteristic::getDescriptor

Description
Get a descriptor with the specified UUID on the remote device.

Syntax
BLERemoteDescriptor* getDescriptor(const char* uuid);
BLERemoteDescriptor* getDescriptor(BLEUUID uuid);

Parameters
uuid: the desired descriptor UUID, expressed as a character array or a BLEUUID object

Returns
The function returns the found descriptor as a BLERemoteDescriptor object pointer, otherwise nullptr is returned if a descriptor with the UUID is not found.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteCharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canRead

Description
Determine if characteristic has read property enabled.

Syntax
bool canRead();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the read property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canWrite

Description
Determine if characteristic has write property enabled.

Syntax
bool canWrite();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the write property for the characteristic is enabled.

Example Code
NA

Notes and Warnings

NA

BLERemoteCharacteristic::canNotify

Description
Determine if characteristic has notify property enabled.

Syntax
bool canNotify();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the notify property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canIndicate

Description
Determine if characteristic has indicate property enabled.

Syntax
bool canIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the indicate property for the characteristic is enabled.

Example Code

NA

Notes and Warnings
NA

BLERemoteCharacteristic::getProperties

Description
Get the characteristic properties.

Syntax
uint16_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic properties.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readString

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns

The function returns the data in the characteristic buffer expressed as a String class object.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData8

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint8_t value.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData16

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readData32

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeString

Description
Write data to the remote device characteristic as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote characteristic, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData8

Description
Write data to the remote device characteristic as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData16

Description
Write data to the remote device characteristic as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData32

Description
Write data to the remote device characteristic as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setData

Description
Write data to the remote device characteristic.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getData

Description
Request for characteristic data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLERemoteCharacteristic::enableNotifyIndicate

Description
Enable the remote device to send notifications or indications for the characteristic.

Syntax
void enableNotifyIndicate(bool notify = 1);

Parameters
notify: TRUE to enable notifications, FALSE to enable indications.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::disableNotifyIndicate

Description
Disable receiving notifications and indications for the characteristic from the remote device.

Syntax
void disableNotifyIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setNotifyCallback

Description
Set a user function to be called when the characteristic receives a notification from the remote device.

Syntax
void setNotifyCallback(void (*fCallback) (BLERemoteCharacteristic* chr, uint8_t* data, uint16_t length));

Parameters
fCallback: A user callback function that returns void and takes three arguments.
chr: pointer to BLERemoteCharacteristic object associated with notification.
data: pointer to byte array containing notification data.
length: number of bytes of notification data in array.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLERemoteDescriptor

BLERemoteDescriptor Class

Description
A class used for managing BLE GATT descriptors on connected remote devices.

Syntax
class BLERemoteDescriptor

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteCharacteristic::getDescriptor().

Public Methods	
BLERemoteDescriptor::getUUID	Get the descriptor UUID
B LERemoteDescriptor::setBufferLen	Set the size of the internal data buffer
B LERemoteDescriptor::getBufferLen	Get the current size of the internal data buffer
BLERemoteDescriptor::readString	Read the descriptor data buffer as a String object
BLERemoteDescriptor::readData8	Read the descriptor data buffer as an unsigned 8-bit integer
BLERemoteDescriptor::readData16	Read the descriptor data buffer as an unsigned 16-bit integer
BLERemoteDescriptor::readData32	Read the descriptor data buffer as an unsigned 32-bit integer
BLERemoteDescriptor::writeString	Write data to the descriptor as a String object or character array
BLERemoteDescriptor::writeData8	Write data to the descriptor as an unsigned 8-bit integer
BLERemoteDescriptor::writeData16	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::writeData32	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::setData	Write data to the descriptor
BLERemoteDescriptor::getData	Read data from the descriptor

BLERemoteDescriptor::getUUID

Description
Get the descriptor UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the descriptor UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setBufferLen

Description
Set the size of the internal data buffer of the descriptor.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Descriptor data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteDescriptor::getBufferLen

Description
Get the size of the descriptor internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readString

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a String class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData8

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData16

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData32

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeString

Description
Write data to the remote device descriptor as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote descriptor, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData8

Description
Write data to the remote device descriptor as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData16

Description
Write data to the remote device descriptor as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData32

Description
Write data to the remote device descriptor as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the descriptor buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setData

Description
Write data to the remote device descriptor.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::getData

Description
Request for descriptor data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

Class BLERemoteService

BLERemoteService Class

Description
A class used for managing BLE GATT services on connected remote devices.

Syntax
class BLERemoteService

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLE-Client::getService().

Public Methods	
BLERemoteService::getUUID	Get the service UUID
BLE RemoteService::getCharacteristic	Get a specific characteristic on the remote device

BLERemoteService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteService::getCharacteristic

Description
Get a characteristic with the specified UUID on the remote device.

Syntax
BLERemoteCharacteristic* getCharacteristic (const char* uuid);
BLERemoteCharacteristic* getCharacteristic (BLEUUID uuid);

Parameters
uuid: the desired characteristic UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found characteristic as a BLERemoteCharacteristic object pointer, otherwise nullptr is returned if a characteristic with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLEScan

BLEScan Class

Description
A class used for managing BLE scanning settings.

Syntax
class BLEScan

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configScan

Public Methods	
BLEScan::updateScanParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEScan::startScan	Start a BLE scan
BLEScan::stopScan	Stop a BLE scan
BLEScan::setScanMode	Set the BLE scanning mode
BLEScan::setScanInterval	Set the BLE scanning interval
BLEScan::setScanWindow	Set the BLE scanning window
BLEScan::setScanDuplicateFilter	Set the BLE scan duplicate filter
BLEScan::scanInProgress	Check if a scan is currently in progress
BLEScan::printScanInfo	Print out scanned information

BLEScan::updateScanParams

Description
Update the lower Bluetooth stack with the current scan settings.

Syntax
void updateScanParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Stop any scans in progress first before using this function.

BLEScan::startScan

Description
Start BLE scanning.

Syntax
void startScan();
void startScan(uint32_t scanDuration_ms);

Parameters
scanDuration: BLE scan will stop after scanDuration milliseconds.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Set the scan parameters first before starting a scan. BLE scans will occur continuously for the duration set with BLEDevice::setScanWindow() and will repeat with a time interval set with BLEDevice::setScanInterval(). Call this member function without an argument to start scanning until BLEDevice::stopScan() is called.

BLEScan::stopScan

Description
Stop BLE scanning.

Syntax
void stopScan(void);

Parameters
The function requires no input paramter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEScan::setScanMode

Description
Set the BLE scan mode.

Syntax
void setScanMode(uint8_t scanMode);

Parameters
scanMode: GAP_SCAN_MODE_PASSIVE for passive scanning, GAP_SCAN_MODE_ACTIVE for active scanning

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Active scanning will request for scan response packets after discovering an advertising device. Passive scanning will only capture advertising data packets.

BLEScan::setScanInterval

Description
Set the BLE scan interval.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will repeat with a time interval set with this member function.

BLEScan::setScanWindow

Description
Set the BLE scan window.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will scan continuously for a window duration set with this member function. The scan window should be less than or equal to the scan interval.

BLEScan::setScanDuplicateFilter

Description
Set the scan duplicate filter.

Syntax
void setScanDuplicateFilter(bool dupeFilter);

Parameters
dupeFilter: TRUE to enable duplicate filtering.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling duplicate filters will ignore scan results for devices already discovered previously.

BLEScan::scanInProgress

Description
Set the scan duplicate filter.

Syntax
bool scanInProgress(void);

Parameters
The function requires no input paramter.

Returns
TRUE if BLE scanning is in progress.

Example Code
NA

Notes and Warnings
NA

BLEScan::printScanInfo

Description
Parse and print out scanned information.

Syntax
void printScanInfo(T_LE_CB_DATA* p_data);

Parameters
p_data: pointer to scan data of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to parse the various fields of received advertisement data packets and print the results out to the serial monitor.

Class BLEService

BLEService Class

Description
A class used for creating and managing BLE GATT services.

Syntax
class BLEService

Members

Public Constructors	
BLEService::BLEService	Constructs a BLEService object
Public Methods	
BLEService::setUUID	Set service UUID
BLEService::getUUID	Get service UUID
BLEService::addCharacteristic	Add a characteristic to service
BLEService::getCharacteristic	Get a previously added characteristic

BLEService::BLEService

Description

Constructs a BLEService object.

Syntax
BLEService::BLEService(BLEUUID uuid);
BLEService::BLEService(const char* uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object or a character array

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLEService::setUUID

Description
Set the service UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLEService::addCharacteristic

Description
Add a characteristic to the service.

Syntax
void addCharacteristic(BLECharacteristic& newChar);

Parameters
newChar: the BLECharacteristic to add to the service.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings

NA

BLEService::getCharacteristic

Description
Get a previously added characteristic.

Syntax
BLECharacteristic* getCharacteristic(uint8_t charIndex);

Parameters
charIndex: position index of characteristic.

Returns
The function returns a pointer to the BLECharacteristic at the requested position index.

Example Code
NA

Notes and Warnings
NA

Class BLEUUID

BLEUUID Class

Description
A class used for creating and managing UUIDs.

Syntax
class BLEUUID

Members

Public Constructors	
BLEUUID::BLEUUID	Create a UUID object
Public Methods	
BLEUUID::str	Get the character string representation of UUID
BLEUUID::data	Get the binary representation of UUID
BLEUUID::length	Get the length of UUID

BLEUUID::BLEUUID

Description
Create a UUID object from a UUID character string

Syntax
BLEUUID();
BLEUUID(const char* str);
BLEUUID(uint8_t* data, uint8_t length);

Parameters
str: UUID character string used to created object
data: pointer to byte array containing the desired UUID
length: number of bytes in array containing the desired UUID. Valid values of 2, 4 or 16

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLEUUID::str

Description
Get the character string representation of UUID

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a character string representation of the UUID

Example Code
NA

Notes and Warnings

BLEUUID::data

Description
Get the binary representation of UUID

Syntax
const uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
Pointer to an unsigned 8-bit integer array containing the UUID expressed in binary form

Example Code
NA

Notes and Warnings
Returned pointer is of const uint8_t* type and will not allow changing of the data.

BLEUUID::length

Description
Get the length of UUID

Syntax
uint8_t length(void);

Parameters
The function requires no input parameter.

Returns
Length of the UUID, in terms of bytes

Example Code

NA

Notes and Warnings
A 4-character UUID will be 16 bits / 2 bytes long.
A 32-character UUID will be 128 bits / 16 bytes long.

Class BLEWifiConfigService

BLEWifiConfigService Class

Description
A class used for managing a BLE WiFi configuration service running on the device.

Syntax
class BLEWifiConfigService

Members

Public Constructors	
BLEWifiCon figService::BLEWifiConfigService	Only one instance of this class should be created

Public Methods	
BLEWifiConfigService::begin	Start background thread to process WiFi configuration commands
BLEWifiConfigService::end	Stop background thread processing WiFi configuration commands
BLEWifiConfigService::addService	Add the service to the BLE stack
BLEWifiConfigService::advData	Get advertising data correctly formatted for WiFi configuration service

BLEWifiConfigService::BLEWifiConfigService

Description
Create an instance of the BLEWifiConfigService object.

Syntax
void BLEWifiConfigService ();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
Only one instance of this class / service should be created.

BLEWifiConfigService::begin

Description
Start background thread to process WiFi configuration commands.

Syntax
void begin();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::end

Description
Stop background thread processing WiFi configuration commands.

Syntax
void end();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEWifiConfigService::addService

Description
Add the WiFi configuration service to the BLE stack.

Syntax
void addService();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::advData

Description
Get advertising data correctly formatted for WiFi configuration service.

Syntax
BLEAdvertData advData();

Parameters
The function requires no input parameter.

Returns
The function returns a BLEAdvertData object that contains the required advertising data fields for the WiFi configuration service to work.

Example Code
Example: BLEWifiConfig

Notes and Warnings
The advertisement data needs to be correctly formatted for the corresponding smartphone app to recognise the device. WiFi configuration service advertisement data requires the local BT address, and should be called only after peripheral mode is started and may also require stopping and restarting the advertising process.

EPDIF

Class EpdIF

EpdIf Class

Description
A class used to control the electronic paper display internal functions.

Syntax
class EpdIf

Members

Public Constructors
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named EpdIf.

Public Methods	
EpdIf::EPD_Dis_Part	Put an image buffer to the frame memory, but not updating the display
EpdIf::EPD_SetFrame	Put display data to the frame memory, usually used for setup text display functions
EpdIf::EPD_SetRAMValue_BaseMap	To read image data stored in the RAM, but not display on the screen
EpdIf::EPD_SetFrameMemory	To read image data stored in the buffer, but not display on the screen
EpdIf::EPD_UpdateDisplay	Update the display
EpdIf::EPD_ClearScreen_White	Clear the frame memory with the White color, but not updating the display
EpdIf::EPD_ClearScreen_Black	Clear the frame memory with the Black color, but not updating the display
EpdIf::EPD_Busy	Wait until the Busy pin goes to low, which is the idle state
EpdIf::EPD_Reset	Used for the Epaper module reset. Often used to awaken the module in deep sleep
EpdIf::EPD_Sleep	After this command is transmitted, the chip would enter the deep-sleep mode to save power

EpdIf:: EPD_Dis_Part

Description

Put an image buffer to the frame memory, but not updating the display.

Syntax

void EPD_Dis_Part(unsigned int x_start, unsigned int y_start, const unsigned char* datas, unsigned int PART_COLUMN, unsigned int PART_LINE);

Parameters

x_start: starting position of the x-axis
y_start: starting position of the y-axis
datas: data to be displayed on the e-paper module
PART_COLUMN: height of the display area
PART_LINE: width of the display area

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

EpdIf:: EPD_SetFrame

Description

Put display data to the frame memory, usually used for setup text display functions.

Syntax
void EPD_SetFrame(const unsigned char* image_buffer, int x, int y, int image_width, int image_height);

Parameters
image_buffer: the buffer which stores the data to be displayed on the e-paper module, usually used to display texts.
x: starting position of the x-axis
y: starting position of the y-axis
image_width: width of the display area
image_height: height of the display area

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetRAMValue_BaseMap

Description
To read image data stored in the RAM, but not display on the screen.

Syntax
void EPD_SetRAMValue_BaseMap(const unsigned char* datas);

Parameters
datas: contains the black and white information that forms the image stored in RAM

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetFrameMemory

Description
To read image data stored in the buffer but not display on the screen.

Syntax
void EPD_SetFrameMemory(const unsigned char* image_buffer);

Parameters
image_buffer: the buffer where stores the image data in hexadecimal numbers

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_UpdateDisplay

Description
Update the ePaper display module. Always combined used with functions set the frames.

Syntax
void EPD_UpdateDisplay(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

There are 2 memory areas embedded in the e-paper display but once this function is called, then the next action of SetFrameMemory or ClearScreen will set the other memory area.

EpdIf:: EPD_ClearScreen_White

Description
Clear the frame memory with the White color.

Syntax
void EpdIf::EPD_ClearScreen_White(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_ClearScreen_Black

Description
Clear the frame memory with the Black color.

Syntax
void EpdIf::EPD_ClearScreen_Black(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Busy

Description
Wait until the busy_pin goes to low, which is the idle state.

Syntax
void EpdIf::EPD_Busy(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Reset

Description
This command will let the E-paper module reset, it is often used to awaken the module in while it's in the deep sleep mode, you will find more details in the function EpdIf:: EPD_Sleep().

Syntax
void EpdIf::EPD_Reset(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf::EPD_Sleep

Description
After this command is transmitted, the chip would enter the deep-sleep mode to save power. The deep sleep mode would return to standby by hardware reset. You can use EPD:: Init() to awaken the E-paper module.

Syntax
void EpdIf::EPD_Sleep(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

FatfsSDCard

Class SdFatFs

Description

Defines a class of SD FAT File system.

Syntax

class SdFatFs

Members

Public Constructors

SdFatFs::SdFatFs Constructs a SdFatFs object

SdFatFs::~SdFatFs Destructs a SdFatFs object

Public Methods

SdFatFs::begin	Initialize SD FAT File System
SdFatFs::end	Deinitialize SD FAT File System
SdFatFs::*getRootPath	Get the root path of the SD FAT File System
SdFatFs::readDir	List items under a specific folder
SdFatFs::mkdir	Create folder
SdFatFs::rm	Remove folder or file
SdFatFs::isDir	Check if a specific path is a directory
SdFatFs::isFile	Check if a specific path is a file
SdFatFs::getLastModTime	Get the last modified time for a file or directory
SdFatFs::setLastModTime	Set the last modified time for a file or directory
SdFatFs::status	Return the current status of SD
SdFatFs::open	Open a file

SdFatFs::begin

Description
Initialize SD FAT File System.

Syntax
int SdFatFs::begin(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::end

Description
De-initialize SD FAT File System.

Syntax
int SdFatFs::end(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::*getRootPath

Description
Get the root path of the SD FAT File System. The logical volume character is starting from ‘0’, so the root path would like “0:/”.

Syntax
char *SdFatFs::getRootPath(void);

Parameters
The function requires no input parameter.

Returns
The function returns the root path.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::readDir

Description
List items under a specific folder. List items under a specific folder and store the result in the buffer that user specified. Each item is separated by '0'.

Syntax
int SdFatFs::readDir(char *path, char *result_buf, unsigned int bufsize);

Parameters
path: The absolute directory path to be listed.
result_buf: The buffer to be stored results.
bufsize: The size of result_buf. If results exceed this size, then the results larger than this size would be discarded.

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: get_file_attribute; list_root_files

Notes and Warnings
Include "SdFatFs.h" to use the class function.

SdFatFs::mkdir

Description
Create folder.

Syntax
int SdFatFs::mkdir(char *absolute_path);

Parameters
absolute_path: The absolute directory path to be created

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: create_folder

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::rm

Description
Remove folder or file.

Syntax
int SdFatFs::rm(char *absolute_path);

Parameters
absolute_path: The absolute directory or file path to be deleted

Returns
Returns “0” if success, else returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isDir

Description
Check if a specific path is a directory.

Syntax
unsigned char SdFatFs::isDir(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isFile

Description
Check if a specific path is a file.

Syntax
unsigned char SdFatFs::isFile(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::getLastModTime

Description
Get the last modified time for a file or directory.

Syntax
int SdFatFs::getLastModTime(char *absolute_path, uint16_t *year, uint16_t *month, uint16_t *date, uint16_t *hour, uint16_t *minute, uint16_t *second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.

month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::setLastModTime

Description
Set the last modified time for a file or directory. Ameba doesn’t have built-in RTC. So we manually change file/directory last modified time.

Syntax
int SdFatFs::setLastModTime(char *absolute_path, uint16_t year, uint16_t month, uint16_t date, uint16_t hour, uint16_t minute, uint16_t second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.
month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::open

Description
Open a file.

Syntax
SdFatFile SdFatFs::open(char *absolute_path);

Parameters
absolute_path: The path to a file.

Returns
The file object is an instance of SdFatFile.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::status

Description
Return the current status of SD.

Syntax
int SdFatFs::status(void);

Parameters
The function requires no input parameter.

Returns
Function returns “1” if ready to use, else return “0” if the status is inactivating or abnormal.

Example Code
NA.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

Class SdFatFile

Description

Defines a class of SD FAT File.

Syntax
class SdFatFile

Members

Public Constructors	
SdFatFile::SdFatFile	Constructs a SdFatFile object
SdFatFile::~SdFatFile	Destructs a SdFatFile object
Public Methods	
SdFatFile::write	Write 1 byte/bytes to file
SdFatFile::read	Read 1 byte/bytes from the file
SdFatFile::peek	Read 1 byte from file without move curser
SdFatFile::available	Check if the cursor is at EOF (End-Of-File)
SdFatFile::bool	Check if file is opened
SdFatFile::seek	Change cursor to a specific position
SdFatFile::close	Close file

SdFatFile::write

Description
Write 1 byte or bytes to the file.

Syntax
size_t SdFatFile::write(uint8_t c);
size_t SdFatFile::write(const uint8_t *buf, size_t size);

Parameters
c: The character to be written.
buf: The buffer to be written.
size: The length of buffer to be written.

Returns

The function returns the number of byte count that has been successfully written to the file.

Example Code
NA.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: read

Description
Read 1 byte or bytes from the file.

Syntax
int SdFatFile::read(void);
int SdFatFile::read(void *buf, uint16_t nbyte);

Parameters
buf: The buffer to store the content.
nbyte: The buffer size. (Or can be regarded as the desired length to read).

Returns
The function returns a read character or the read size of the buffer.

Example Code

```
1. #include “FatFs_SD.h”
2.
3. char dirname[] = “testdir”;
4. char filename[] = “test.txt”;
5. char write_content[] = “hello world!”;
6.
7. FatFsSD fs;
8.
9. void setup() {
10. char buf[128];
11. char absolute_filename[128];
12.
13. fs.begin();
```

```
14.
15. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), dirname);
16. fs.mkdir(absolute_filename);
17. printf(“create dir at \”%s"rn”, absolute_filename);
18.
19. sprintf(absolute_filename, “%s%s/%s”, fs.getRootPath(), dirname, filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.println(write_content);
22. file.close();
23. printf(“create file at \”%s"rn”, absolute_filename);
24.
25. printf(“read back from \”%s"rn”, absolute_filename);
26. file = fs.open(absolute_filename);
27.
28. memset(buf, 0, sizeof(buf));
29. file.read(buf, sizeof(buf));
30.
31. file.close();
32. printf(“==== content ====rn”);
33. printf(“%s”, buf);
34. printf(“==== end ====rn”);
35.
36. fs.end();
37. }
38.
39. void loop() {
40. delay(1000);
41. }
42.
```

Example: create_folder;
This example shows how to create a folder and open a file under it.

```
1. #include “FatFs_SD.h”
2.
3. char filename[] = “test.txt”;
4. char write_content[] = “hello world!”;
```

```
5.
6.  FatFsSD fs;
7.
8.  void setup() {
9.  char buf[128];
10. char absolute_filename[128];
11.
12. fs.begin();
13.
14. printf("write something to \"%s"rn", filename);
15. sprintf(absolute_filename, "%s%s", fs.getRootPath(), filename);
16. SdFatFile file = fs.open(absolute_filename);
17.
18. file.println(write_content);
19.
20. file.close();
21. printf("write finishrnrn");
22.
23. printf("read back from \"%s"rn", filename);
24. file = fs.open(absolute_filename);
25.
26. memset(buf, 0, sizeof(buf));
27. file.read(buf, sizeof(buf));
28.
29. file.close();
30. printf("==== content ====rn");
31. printf("%s", buf);
32. printf("==== end ====rn");
33.
34. fs.end();
35. }
36.
37. void loop() {
38. delay(1000);
39. }
40.
```

Example: file_read_write;
This example shows how to open/close files and perform read/write to it.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: peek

Description
Read one byte from the file without moving the curser.

Syntax
int SdFatFile::peek(void);

Parameters
The function requires no input parameter.

Returns
The function returns the read character as an integer number.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: available

Description
Check if the cursor is at EOF.

Syntax
int SdFatFile::available(void);

Parameters
The function requires no input parameter.

Returns
The function returns “0” if the cursor is at EOF, else returns “1”.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: flush

Description
It is a nop. This is an inherited function from class Stream. And it does not affect SD File.

Syntax
void SdFatFile::flush(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: seek

Description
Change cursor to a specific position.

Syntax
int SdFatFile::seek(uint32_t pos);

Parameters
pos: The desired position.

Returns
The function returns 0 if success otherwise returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” in order to use the class function.

SdFatFile:: close

Description
Close file.

Syntax
int SdFatFile::close(void);

Parameters
The function requires no input parameter.

Returns
The function returns 0 if runs successfully otherwise it returns a negative value.

Example Code
Example: last_modified_time;
The example shows how to get and set last modified time of a file.

Example: create_folder;
This example shows how to create a folder and open a file under it. The details of the code can be found in the section of SdFatFile:: read.
Example: file_read_write;
This example shows how to open/close files and perform read/write to it. The details of the code can be found in the section of SdFatFile:: read.

1. #include <FatFs_SD.h>
2.
3. FatFsSD fs;
4.
5. char filename[] = “test.txt”;
6.
7. void setup() {
8. char absolute_filename[128];
9.

```
10. uint16_t year = 2021;
11. uint16_t month = 4;
12. uint16_t date = 4;
13. uint16_t hour = 12;
14. uint16_t minute = 12;
15. uint16_t second = 12;
16.
17. fs.begin();
18.
19. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.close();
22.
23. fs.setLastModTime(absolute_filename, year, month, date, hour, minute, second);
24.
25. fs.getLastModTime(absolute_filename, &year, &month, &date, &hour, &minute, &second);
26. printf(“filename:"%s"rn”, absolute_filename);
27. printf(“time mod:%04d/%02d/%02d %02d:%02d:%02drn”, year, month, date, hour, minute, second);
28.
29. fs.end();
30. }
31.
32. void loop() {
33. delay(1000);
34. }
35.
```

Notes and Warnings

Include “SdFatFile.h” in order to use the class function.

FlashMemory

Class EpdIF

FlashMemoryClass Class

Description
Defines a class of Flash memory API

Syntax
class FlashMemoryClass

Members

Public Constructors	
Fl ashMemoryClass::FlashMemoryClass	Constructs a FlashMemoryClass object
Fla shMemoryClass::~FlashMemoryClass	Deconstructs a FlashMemoryClass object
Public Methods	
FlashMemoryClass::begin	Initialize/Re-initialize the base address and size
FlashMemoryClass::read	Read the content to buf
FlashMemoryClass::update	Write buf back to flash memory
FlashMemoryClass::readWord	Read 4 bytes from flash memory
FlashMemoryClass::writeWord	Write 4 bytes into flash memory
FlashMemoryClass::buf_size	The buf size
FlashMemoryClass::*buf	The buf to be operated

FlashMemoryClass::FlashMemoryClass

Description
Constructs a FlashMemoryClass object.

Syntax
FlashMemoryClass(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address to operate.
_buf_size: The buf size for mirror a copy to reduce flash memory operation

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Direct read from flash memory is allowed. To write data into flash memory, each bit on flash memory can only change from '1' to '0' and it cannot change from '0' to '1'. To make sure the data are correctly written we do erase the flash memory sector before write data on it.

```
#include <FlashMemory.h>
void setup() {
FlashMemory.read();
if (FlashMemory.buf[0] == 0xFF) {
FlashMemory.buf[0] = 0x00;
FlashMemory.update();
Serial.println("write count to 0");
} else {
FlashMemory.buf[0]++;
FlashMemory.update();
Serial.print("Boot count: ");
Serial.println(FlashMemory.buf[0]);
}
}
void loop() {
delay(1000);
}
```

Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

```
#include <FlashMemory.h>
void setup() {
unsigned int value;
/* request flash size 0x4000 from 0xFC000 */
FlashMemory.begin(0xFC000, 0x4000);
/* read one word (32-bit) from 0xFC000 plus offset 0x3F00 */
value = FlashMemory.readWord(0x3F00);
printf("value is 0x%08Xrn", value);
if (value == 0xFFFFFFFF) {
value = 0;
```

```
} else {
value++;
}
/* write one word (32-bit) to 0xFC000 plus offset 0x3F00 */
FlashMemory.writeWord(0x3F00, value);
}
void loop() {
// put your main code here, to run repeatedly:
}
```

Notes and Warnings
Include “FlashMemory.h” to use the class function.

FlashMemoryClass::begin

Description
Initialize/Re-initialize the base address and size. The base address shell aligns with the size of 0x1000. And the size shell is multiple of 0x1000.

Syntax
void begin(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address
_buf_size: The desired work size

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba’s flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of “FlashMemoryClass:: FlashMemoryClass”.

Notes and Warnings

Include “FlashMemory.h” to use the class function.

FlashMemoryClass::read

Description
Read the content to buf. Read flash memory into the buf. The size would be 0x1000.

Syntax
void read(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba’s flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of “FlashMemoryClass:: FlashMemoryClass”.

Notes and Warnings
Include “FlashMemory.h” to use the class function.

FlashMemoryClass::update

Description
Write buf back to flash memory. Write flash memory with the content of the buffer. The size is 0x1000.

Syntax
void update(bool erase = true);

Parameters
erase: By default, it is true and erases flash memory before writing to it

Returns

The function returns nothing.

Example Code

Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::readWord

Description

Read 4 bytes from flash memory. Read 4 byte from specific offset based on base address.

Syntax

unsigned int readWord(unsigned int offset);

Parameters

offset: The offset according to the base address

Returns

The read data with a size of 4 bytes

Example Code

Example: ReadWriteOneWord

This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::writeWord

Description

Write 4 bytes into flash memory. It will try to write 4 bytes first. If the read data differ from the write data, then we buffer the sector of flash memory, erase it, and write correct data back to it.

Syntax
void writeWord(unsigned int offset, unsigned int data);

Parameters
offset: The offset according to the base address
data: The data to be written

Returns
The function returns nothing.

Example Code
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::buf_size

Description
The buf size (It can be regarded as work size).

Syntax
unsigned int buf_size;

Example Code
Example: FlashMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::*buf

Description
The buf to be operated. Modify buf won't change the content of the buf. It needs an update to write back to flash memory.

Syntax
unsigned char *buf;

Example Code
NA

Notes and Warnings
Include "FlashMemory.h" to use the class function.

GPIO

Class DHT

DHT Class

Description
Defines a class of using DHT temperature & humidity sensors

Syntax
class DHT

Members

Public Constructors	
DHT::DHT	Constructs a DHT object
Public Methods	
DHT::begin	Initialize the DHT sensor
DHT::readTemperature	Read temperature(Fahrenheit or Celcius) from the DHT sensor
DHT::convertCtoF	Convert a value from Celcius to Fahrenheit
DHT::convertFtoC	Convert a value from Fahrenheit to Celcius
DHT::readHumidity	Read humidity(%) from the DHT sensor
DHT::computeHeatIndex	Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations)
DHT::read	Check if the sensor is readable

DHT::DHT

Description
Constructs a DHT object.

Syntax
DHT::DHT(uint8_t pin, uint8_t type, uint8_t count)

Parameters
pin: The Arduino digital PIN connected
type: The DHT sensor type(DHT11, DHT22, or DHT21)
count: The count is now ignored as the DHT reading algorithm adjusts itself based on the speed of the processor

Returns
The function returns nothing.

Example Code
Example:DHTTester
The code demos basic testing for various DHT humidity & temperature sensors.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

// Example testing sketch for various DHT humidity/temperature sensors

// Written by ladyada, public domain

#include "DHT.h"

// The digital pin we're connected to.

#define DHTPIN 8

// Uncomment whatever type you're using!

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V

// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1

// to 3.3V instead of 5V!

// Connect pin 2 of the sensor to whatever your DHTPIN is

// Connect pin 4 (on the right) of the sensor to GROUND

// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.

// Note that older versions of this library took an optional third parameter to

```
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(115200);
Serial.println("DHTxx test!");
dht.begin();
}
void loop() {
// Wait a few seconds between measurements.
delay(2000);
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *Ft");
Serial.print("Heat index: ");
```

Serial.print(hic);

Serial.print(" *C ");

Serial.print(hif);

Serial.println(" *F");

}

DHT::begin

Description
Initialize the DHT sensor.

Syntax
void DHT::begin(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::readTemperature

Description
Read temperature(Fahrenheit or Celcius) from the DHT sensor.

Syntax
float DHT::readTemperature(bool S, bool force);

Parameters
S: Temperature scale, True is Fahrenheit and False is Celcius
force: Index of checking sensor readability, default is False

Returns
The function returns the current temperature as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertCtoF

Description
Convert a value from Celcius to Fahrenheit.

Syntax
float DHT::convertCtoF(float c);

Parameters
c: The value in Celcius

Returns
The function returns the temperature in Fahrenheit as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertFtoC

Description
Convert a value from Fahrenheit to Celcius.

Syntax
float DHT::convertFtoC(float f);

Parameters
f: The value in Fahrenheit

Returns
The function returns the temperature in Celcius as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::computeHeatIndex

Description
Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations). More details refer to http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml .

Syntax
float DHT::computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit);

Parameters
temperature: The temperature value
percentHumidity: The humidity percent value
isFahrenheit: True, temperature value in Fahrenheit (Default); False, temperature value in Celcius

Returns
The function returns the heat index in Fahrenheit or Celsius as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings

Every time must include the header file “DHT.h” in front of the project to use the class function.

DHT::readHumidity

Description
Reading temperature or humidity from the DHT sensor and return as a float value(%).

Syntax
float DHT::readHumidity(bool force);

Parameters
force: Ignored.

Returns
The function returns current humidity in a float number (in %).

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the “DHT: DHT” section.

Notes and Warnings
Every time must include the header file “DHT.h” in front of the project to use the class function. Reading temperature or humidity takes about 250 milliseconds! Sensor readings may also be up to 2 seconds.

DHT::read

Description
Check if the sensor is readable.

Syntax
boolean DHT::read(bool force);

Parameters
force: Index of whether checking the sensor was read less than two seconds ago or not. False, checking; True, not checking.

Returns

Return the last correct measurement of the sensor. False, low means not readable; True, high means readable.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

Class HttpClient

InterruptLock Class

Description
Defines a class of turning off/on interrupts temporarily

Syntax
class InterruptLock

Members

Public Constructors	
InterruptLock::InterruptLock	Constructs a InterruptLock object
InterruptLock::~ InterruptLock	Deconstructs a InterruptLock object

---

GTimer

Class EpdIF

GTimerClass Class

Description
GTimer is a hardware timer and this class is to operate it. The GTimer occupy same resource as PWM. Please make sure the timer is not conflict with you PWM index.

Syntax
class GTimerClass

Members

Public Constructors	
GTimerClass::GTimerClass	Constructs a GTimerClass object
Public Methods	
GTimerClass::begin	Initialize a timer and start it immediately
GTimerClass::stop	Stop a specific timer
GTimerClass::reload	Reload a specific timer
GTimerClass::read_us	Read current countdown value

GTimerClass::begin

Description
Initialize a timer and start it immediately.

Syntax
void GTimerClass::begin(uint32_t timerid, uint32_t duration_us, void (*handler)(uint32_t), bool periodical, uint32_t userdata);

Parameters
timerid: There are 5 valid GTimer with timer id 0~4.
duration_us: The duration of the timer. The time unit is microsecond and the precision is 32768Hz.
periodical: By default, the timer would keep periodically countdown and reload which means the handler would periodically be invoked.
userdate: The user data brings to the handler.

Returns
The function returns nothing.

Example Code
Example: TimerOneshot

/*

This sketch shows how to use several hardware timers in invoke handler only once for each timer.

*/

#include <GTimer.h>

void myhandler(**uint32_t** data) {

Serial.print("I am timer!");

Serial.println(data);

}

void setup() {

```
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// timerid 0, period 1s, invoke myhandler, invoke only once, user data is 0
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
// timerid 1, period 2s, invoke myhandler, invoke only once, user data is 1
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
GTimer.begin(2, 3 * 1000 * 1000, myhandler, false, 2);
GTimer.begin(3, 4 * 1000 * 1000, myhandler, false, 3);
}
void loop() {
delay(1000);
}
```

Example: TimerPeriodical

```
/*
This sketch shows how to use hardware timer and invoke interrupt handler periodically
*/
#include <GTimer.h>
int counter = 0;
void myhandler(uint32_t data) {
counter++;
Serial.print("counter: ");
Serial.println(counter);
if (counter >= 10) {
Serial.println("stop timer");
GTimer.stop(0);
}
}
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
```

```
// timerid 0, period 1s, invoke myhander
GTimer.begin(0, (1 * 1000 * 1000), myhandler);
}
void loop() {
delay(1000);
}
```

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::stop

Description
Stop a specific timer

Syntax
void GTimerClass::stop(uint32_t timerid);

Parameters
timerid: Stop the timer with this timer id

Returns
The function returns nothing.

Example Code
Example: TimerPeriodical, please refer to GTimerClass:: begin for more details.

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::reload

Description
Reload a specific timer. The GTimer is a countdown timer. Reload it would make it discard the current countdown value and restart countdown based on the duration.

Syntax

void GTimerClass::reload(uint32_t timerid, uint32_t duration_us);

Parameters
timerid: The timer to be modified
duration_us: The updated duration in unit of microseconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::read_us

Description
Read the current countdown value

Syntax
uint64_t GTimerClass::read_us(uint32_t timerid);

Parameters
timerid: The timer to be read

Returns
The function returns the current countdown value.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

Http

Class HttpClient

HttpClient Class

Description
Defines a class of using HttpClient

Syntax
class HttpClient

Members

Public Constructors	
HttpClient::HttpClient	Constructs a HttpClient object
Public Methods	
HttpClient::beginRequest	Start a more complex request
HttpClient::endRequest	End a more complex request
HttpClient::get	Connect to the server and start to send a GET request
HttpClient::post	Connect to the server and start to send a POST request
HttpClient::put	Connect to the server and start to send a PUT request
HttpClient::startRequest	Connect to the server and start to send the request
HttpClient::sendHeader	Send an additional header line
HttpClient::sendBasicAuth	Send a basic authentication header
HttpClient::finishRequest	Finish sending the HTTP request
HttpClient::responseStatusCode	Get the HTTP status code contained in the response
HttpClient::readHeader	Read the next character of the response headers
HttpClient::skipResponseHeaders	Skip any response headers to get to the body
HttpClient::endOfHeadersReached	Test whether all of the response headers have been consumed
HttpClient::endOfBodyReached	Test whether the end of the body has been reached
HttpClient::contentLength	Return the length of the body

HttpClient::HttpClient

Description
Constructs a HttpClient object. If Marco “PROXY_ENABLED” is defined, currently disabled as introduces a dependency on DNS.h in Ethernet.

Syntax
HttpClient::HttpClient(Client& aClient, const char* aProxy = NULL, uint16_t aProxyPort = 0);
HttpClient::HttpClient(Client& aClient);

Parameters
aClient: The object of class WiFiClient.

aProxy: The proxy name. The default proxy name is “NULL”.
aProxyPort: The proxy port. The default value for the proxy port is 0.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrate how to download the content from URL indicated in kHostname[].

```
#include <HttpClient.h>
#include <WiFi.h>
#include <WiFiClient.h>
char ssid[] = “YourNetwork”; // your network SSID (name)
char pass[] = “password”; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
// Name of the server we want to connect to
const char kHostname[] = “www.google.com”;
const char kPath[] = “/”;
// Number of milliseconds to wait without receiving any data before we give up
const int kNetworkTimeout = 30*1000;
// Number of milliseconds to wait if no data is available before trying again
const int kNetworkDelay = 1000;
int status = WL_IDLE_STATUS;
void setup() {
Serial.begin(9600);
while ( status != WL_CONNECTED) {
Serial.print(“Attempting to connect to SSID: “);
Serial.println(ssid);
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println(“Connected to wifi”);
printWifiStatus();
}
void loop() {
```

```
int err =0;
WiFiClient c;
HttpClient http(c);
err = http.get(kHostname, kPath);
if (err == 0)
{
Serial.println("startedRequest ok");
err = http.responseStatusCode();
if (err >= 0)
{
Serial.print("Got status code: ");
Serial.println(err);
// Usually you'd check that the response code is 200 or a
// similar "success" code (200-299) before carrying on,
// but we'll print out whatever response we get
err = http.skipResponseHeaders();
if (err >= 0)
{
int bodyLen = http.contentLength();
Serial.print("Content length is: ");
Serial.println(bodyLen);
Serial.println();
Serial.println("Body returned follows:");
// Now we've got to the body, so we can print it out
unsigned long timeoutStart = millis();
char c;
// Whilst we haven't timed out & haven't reached the end of the body
while ( (http.connected() || http.available()) &&
((millis() - timeoutStart) < kNetworkTimeout) )
{
if (http.available())
{
c = http.read();
// Print out this character
Serial.print(c);
bodyLen--;
```

```
// We read something, reset the timeout counter
timeoutStart = millis();
}
else
{
// We haven't got any data, so let's pause to allow some to arrive
delay(kNetworkDelay);
}
}
}
else
{
Serial.print("Failed to skip response headers: ");
Serial.println(err);
}
}
else
{
Serial.print("Getting response failed: ");
Serial.println(err);
}
}
else
{
Serial.print("Connect failed: ");
Serial.println(err);
}
http.stop();
// And just stop, now that we've tried a download
while(1);
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
```

```
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::beginRequest

Description
Start a more complex request. Use this when you need to send additional headers in the request, but you will also need to call endRequest() when you are finished.

Syntax
void HttpClient::beginRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::endRequest

Description

End a more complex request. Use this when you need to have sent additional headers in the request, but you will also need to call beginRequest() at the start.

Syntax

void HttpClient::endRequest(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::get

Description

Connect to the server and start to send a "GET" request. If the input parameter contains "aServerAddress", the connection will not perform a DNS lookup and just purely connect to the given IP address.

Syntax

int HttpClient::get(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: The name of the server being connected to. If aServerName is "NULL", the "Host" header line will not be sent.

aServerPort: The port on which server connected.

aURLPath: The URL to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::post

Description

Connect to the server and start to send a "POST" request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::post(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::put

Description

Connect to the server and start to send a PUT request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::put(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::startRequest

Description

Connect to the server and start to send the request.

Syntax

int HttpClient::startRequest(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

int HttpClient::startRequest(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

Parameters

aServerAddress: IP address of the server to connect to.

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aHttpMethod: Type of HTTP request to make, e.g. "GET", "POST", etc.

aUserAgent: User-Agent string to send. If NULL the default user-agent kUserAgent will be sent.

Returns

Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::sendHeader

Description

The function sends an additional header line.

The function void HttpClient:: sendHeader(const char* aHeader);can only be called in between the calls to startRequest and finishRequest.

The other 2 functions void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue); and void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue); are alternate form the previous one, which takes the header name and content as separately (as strings or integer). For example, to send an XXXXXX header, user might call sendHeader("XXXXX", "Something") or sendHeader("XXXXX", 123).And the call will add the ": " in the log to separate different header in the case of multiple headers.

Syntax

void HttpClient::sendHeader(const char* aHeader);

void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue);

void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue);

Parameters
aHeader: Header line to send, in its entirety (but without the trailing CRLF. E.g. "Authorization: Basic YQDDCAIGES".
aHeaderName: Type of header being sent.
aHeaderValue: Value for that header.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::sendBasicAuth

Description
The function sends a basic authentication header which will encode the given username and password, and send them in a suitable header line for doing Basic Authentication.

Syntax
void HttpClient::sendBasicAuth(const char* aUser, const char* aPassword);

Parameters
aUser: Username for the authorization.
aPassword: Password for the user aUser.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include “HttpClient.h” to use the class function.

HttpClient::finishRequest

Description
Finish sending the HTTP request. The function sends a blank line to signify the end of the request.

Syntax
void HttpClient::finishRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::responseStatusCode

Description
Get the HTTP status code contained in the response. For example, “200” for successful requests, “404” for file not found, etc.

Syntax
int HttpClient::responseStatusCode(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::readHeader

Description

The function reads the next character of the response headers. This functions the same as read() but to be used when reading through the headers which are slightly less efficient. The user might check whether the end of the headers has been reached by calling endOfHeadersReached(), although after that point this will still return data as read() would.

Syntax

int HttpClient::readHeader(void);

Parameters

The function requires no input parameter.

Returns

Return the next character of the response headers.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::skipResponseHeaders

Description

Skip any response headers to get to the body. Use this if you don't want to do any special processing of the headers returned in the response. You can also use it after you've found all of the headers you're interested in, and just want to get on with processing the body.

Syntax

int HttpClient::skipResponseHeaders(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfHeadersReached

Description
Test whether all of the response headers have been consumed.

Syntax
bool HttpClient::endOfHeadersReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now processing the response body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfBodyReached

Description

Test whether the end of the body has been reached. It only works if the Content-Length header was returned by the server.

Syntax
bool HttpClient::endOfBodyReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now at the end of the body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::contentLength

Description
The function returns the length of the body.

Syntax
int HttpClient::contentLength(void);

Parameters
The function requires no input parameter.

Returns
Return Length of the body, in bytes, or kNoContentLengthHeader if no Content-Length header was returned by the server.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

IRDevice

Class HttpClient

IRDevice Class

Description
A class used for managing, sending, and receiving data using IR.

Syntax
class IRDevice

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named IR.	

Public Methods	
IRDevice::getFreq	Get the current IR modulation frequency
IRDevice::begin	Allocate resources and start the IR device with a custom frequency
IRDevice::end	Stop the IR device operations and free up resources
IRDevice::send	Send IR raw data
IRDevice::beginNEC	Allocate resources and start the IR device with a frequency suitable for the NEC protocol
IRDevice::sendNEC	Send data using the NEC protocol
IRDevice::recvNEC	Receive data using the NEC protocol

IRDevice::getFreq

Description
Get the current IR modulation frequency.

Syntax
uint32_t getFreq(void);

Parameters
The function requires no input parameter.

Returns
Currently set IR modulation frequency in Hertz.

Example Code
NA

Notes and Warnings
NA

IRDevice::begin

Description
Allocate resources and start the IR device with a custom frequency.

Syntax
void begin(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode, uint32_t freq);

Parameters
receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX
freq: IR modulation frequency in Hertz

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
IR device can only operate in either transmit or receive mode.

IRDevice::end

Description
Stop the IR device operations and free up resources.

Syntax

void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

IRDevice::send

Description
Send IR raw data.

Syntax
void send(const unsigned int buf[ ] , uint16_t len);

Parameters
buf[ ] : IR raw signals (in us) in an array form.
len: total length of the IR raw signal array.

Returns
The function returns nothing.

Example Code

```
#include "IRDevice.h"

// User defined txPin, rxPin and carrier frequency

#define IR_RX_PIN 8

#define IR_TX_PIN 9

#define CARRIER_FREQ 38000

unsigned int irRawSignal[] = {

9000, 4500, // starting bit
```

560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560, // address 00100000 ? 4

560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~ address 11011111

560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, // data 00010000 ? 8

560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~ data 11101111

560 // stoping bit

};

int DataLen = sizeof(irRawSignal) / sizeof(irRawSignal[0]); // 284/ 4 = 71

void setup()

{

Serial.begin(115200);

IR.begin(IR_RX_PIN, IR_TX_PIN, IR_MODE_TX, CARRIER_FREQ);

}

void loop()

{

IR.send(irRawSignal, DataLen);

Serial.println("Finished Sending NEC Raw Data….");

delay(3000);

}

Notes and Warnings

IR Raw Data array contains information in the form of consecutive microseconds (us). For more details, please refer to: http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html.

IRDevice::beginNEC

Description

Allocate resources and start the IR device with a frequency suitable for the NEC protocol.

Syntax

void beginNEC(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode);

Parameters

receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX

Returns

The function returns nothing.

Example Code
Example: IRRecvNEC

```
#include "IRDevice.h"
uint8_t adr = 0;
uint8_t cmd = 0;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
IR.beginNEC(8, 9, IR_MODE_RX); // configure for NEC IR protocol
}
void loop() {
if (IR.recvNEC(adr, cmd, 1000)) {
Serial.print("Received ");
Serial.print(adr);
Serial.print(cmd);
Serial.println();
} else {
Serial.println("Received nothing, timed out");
}
//IR.end();
}
```

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::sendNEC

Description
Send data using the NEC protocol.

Syntax
void sendNEC(uint8_t adr, uint8_t cmd);

Parameters
adr: 8-bit address to transmit
cmd: 8-bit command to transmit

Returns
The function returns nothing.

Example Code
Example: IRSendNEC

#include "IRDevice.h"

uint8_t adr = 0;

uint8_t cmd = 0;

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(115200);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

IR.beginNEC(8, 9, IR_MODE_TX); // configure for NEC IR protocol

}

void loop() {

if (cmd++ >=255) {

adr++;

}

IR.sendNEC(adr, cmd);

Serial.print("Sent ");

Serial.print(adr);

Serial.print(cmd);

Serial.println();

//IR.end(); // Call this method to stop IR device and free up the pins for other uses

}

Notes and Warnings

IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::recvNEC

Description
Receive data using the NEC protocol.

Syntax
void recvNEC(uint8_t& adr, uint8_t& cmd uint32_t timeout);

Parameters
adr: variable to store received NEC address
cmd: variable to store received NEC command
timeout: time duration to wait for an incoming transmission

Returns
The function returns "1" if data has been received, returns "0" if no data has been received.

Example Code
Example: IRRecvNEC
Details of the code can be found in the previous section of IRDevice::beginNEC.

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

MDNS

Class HttpClient

MDNSClass Class

Description
A class used for registering and removing MDNS service records.

Syntax
class MDNSClass

Members

Public Constructors	
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named MDNS.	

Public Methods	
MDNSClass::begin	Start MDNS operations
MDNSClass::end	Stop MDNS operations
MDNSClass::registerService	Add a service record
MDNSClass::deregisterService	Remove service record
MDNSClass::updateService	Update service record

MDNSClass::begin

Description
Start MDNS operations to begin responding to MDNS queries.

Syntax
void begin(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE
This example shows how to register Ameba as a service that can be recognized by Arduino IDE. If both of the PC runs Arduino IDE and the Ameba board are connecting to the same local network. Then you can find Ameba in "Tools" -> "Port" -> "Arduino at 192.168.1.238 (Ameba RTL8195A), which means the Arduino IDE find Ameba via mDNS.

#include <WiFi.h>

#include <AmebaMDNS.h>

char ssid[] = "yourNetwork"; // your network SSID (name)

char pass[] = "secretPassword"; // your network password

MDNSService service("MyAmeba", "_arduino._tcp", "local", 5000);

void setup() {

printf("Try to connect to %srn", ssid);

while (WiFi.begin(ssid, pass) != WL_CONNECTED) {

```
printf("Failed. Wait 1s and retry…rn");
delay(1000);
}
printf("Connected to %srn", ssid);
service.addTxtRecord("board", strlen("ameba_rtl8195a"), "ameba_rtl8195a");
service.addTxtRecord("auth_upload", strlen("no"), "no");
service.addTxtRecord("tcp_check", strlen("no"), "no");
service.addTxtRecord("ssh_upload", strlen("no"), "no");
printf("Start mDNS servicern");
MDNS.begin();
printf("register mDNS servicern");
MDNS.registerService(service);
}
void loop() {
// put your main code here, to run repeatedly:
delay(1000);
}
```

Notes and Warnings
Include "AmebaMDNS.h" to use the class function.

MDNSClass::end

Description
Stop MDNS operations and stop responding to MDNS queries.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::registerService

Description
Add a service record to be included in MDNS responses.

Syntax
void register service(MDNSService service);

Parameters
service: MDNSService class object with required MDNS service data

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::deregisterService

Description
Remove a service record from MDNS responses.

Syntax
void deregisterService(MDNSService service);

Parameters
service: MDNSService class object to be removed

Returns
The function returns nothing.

Example Code

Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::updateService

Description
Update a service record.

Syntax
void updateService(MDNSService service, unsigned int ttl);

Parameters
service: MDNSService class object to be updated
ttl: time-to-live(TTL) for service

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

Class HttpClient

MDNSService Class

Description
A class used for creating MDNS service records.

Syntax
class MDNSService

Members

Public Constructors	
MDNSService::MDNSService	Create a MDNS service record
Public Methods	
MDNSService::addTxtRecord	Add text to MDNS service record

MDNSService::MDNSService

Description
Create a MDNS service record.

Syntax
MDNSService(char* name, char* service_type, char* domain, unsigned short port, int bufsize);

Parameters
name: device name
service_type: MDNS service type
domain: host domain
port: network port
bufsize: size of buffer for MDNS text record

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSService::addTxtRecord

Description
Add text to MDNS service record.

Syntax
int addTextRecord(char* key, int value_len, char* value);

Parameters
key: record type expressed as character string
value_len: length of value string

value: record value expressed as character string

Returns
0 if add record successful

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MQTTClient

Class PMUClass

PubSubClient Class

Description
Defines a class of MQTT implementation for Ameba.

Syntax
class PubSubClient

Members

Public Constructors	
PubSubClient::PubSubClient	Constructs a PubSubClient object
Public Methods	
PubSubClient::setServer	Set MQTT server address and port
PubSubClient::setCallback	Set callback function
PubSubClient::setClient	Set WiFi client
PubSubClient::setStream	Set data stream
PubSubClient::connect	Attempt to connect to server
PubSubClient::disconnect	Disconnect from current session
PubSubClient::publish	Publish a message to server
PubSubClient::publish_P	Same as above
PubSubClient::subscribe	Subscribe to a topic
PubSubClient::unsubscribe	Unsubscribe to a topic
PubSubClient::loop	Keep MQTT session alive and process any queuing tasks
PubSubClient::connected	Check if client still connected
PubSubClient::state	Return connection state

PubSubClient::PubSubClient

Description
Constructs a PubSubClient object and, if applicable, sets server address, port, callback function, data stream and wifi client.

Syntax
PubSubClient::PubSubClient();
PubSubClient::PubSubClient(Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);

Parameters
client: the network client to use, for example WiFiClient
IPAddress: MQTT server address
port: port for MQTT, usually 1883 for unencrypted connection
MQTT_CALLBACK_SIGNATURE: callback function for MQTT
Stream: a stream to write received messages to

Returns
The function returns nothing.

Example Code
Example: MQTT_Basic

#include <WiFi.h>

#include <PubSubClient.h>

// Update these with values suitable for your network.

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int status = WL_IDLE_STATUS; // the Wifi radio’s status

char mqttServer[] = “test.mosquitto.org”;

```
char clientId[] = "amebaClient";
char publishTopic[] = "outTopic";
char publishPayload[] = "hello world";
char subscribeTopic[] = "inTopic";
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i=0;i<length;i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}
WiFiClient wifiClient;
PubSubClient client(wifiClient);
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection…");
// Attempt to connect
if (client.connect(clientId)) {
Serial.println("connected");
// Once connected, publish an announcement…
client.publish(publishTopic, publishPayload);
// … and resubscribe
client.subscribe(subscribeTopic);
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}
void setup()
```

```
{
Serial.begin(38400);
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
client.setServer(mqttServer, 1883);
client.setCallback(callback);
// Allow the hardware to sort itself out
delay(1500);
}
void loop()
{
if (!client.connected()) {
reconnect();
}
client.loop();
}
```

Notes and Warnings
PubSubClient::PubSubClient(Client& client) would suffice for normal MQTT connection

PubSubClient::setServer

Description
Sets the server details.

Syntax
PubSubClient& PubSubClient::setServer(uint8_t * ip, uint16_t port)
PubSubClient& PubSubClient::setServer(IPAddress ip, uint16_t port)
PubSubClient& PubSubClient::setServer(const char * domain, uint16_t port)

Parameters

ip: the address of the server
port: the port to connect to, default 1883
domain: the address of the server

Returns
The client instance, allowing the function to be chained

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setCallback

Description
Sets the message callback function.

Syntax
PubSubClient& PubSubClient::setCallback(MQTT_CALLBACK_SIGNATURE)

Parameters
MQTT_CALLBACK_SIGNATURE: a pointer to a message callback function called when a message arrives for a subscription created by this client.

Returns
The client instance, allowing the function to be chained.

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setClient

Description
Sets the network client instance to use.

Syntax
PubSubClient& PubSubClient::setClient(Client& client)

Parameters
client: the network client to use, for example WiFiClient

Returns
The client instance, allowing the function to be chained

Example Code
NA

Notes and Warnings
NA

PubSubClient::setStream

Description
Sets the stream to write received messages to.

Syntax
PubSubClient& PubSubClient::setStream(Stream& stream)

Parameters
stream: a stream to write received messages to

Returns
The client instance, allowing the function to be chained.

Example Code
NA

Notes and Warnings
NA

PubSubClient::connect

Description
Connects the client to the server.

Syntax

boolean PubSubClient::connect(const char *id)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass)

boolean PubSubClient::connect(const char *id, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

Parameters

id: Client ID, a unique string identifier

user: Username for authentication, default NULL

pass: Password for authentication, default NULL

willTopic: the topic to be used by the will message

willQoS: the quality of service to be used by the will message

willRetain: whether the will should be published with the retain flag

willMessage: the payload of the will message

Returns

True – connection succeeded

False – connection failed

Example Code

Example: MQTT_Basic

Notes and Warnings

Client ID is required and should always be unique else connection might be rejected by the server.

PubSubClient::disconnect

Description

Disconnect the client

Syntax

void PubSubClient::disconnect(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PubSubClient::publish

Description
Publishes a message to the specified topic.

Syntax
boolean PubSubClient::publish(const char* topic, const char* payload)
boolean PubSubClient::publish(const char* topic, const char* payload, boolean retained)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
Default max packet size is 128 bytes.

PubSubClient::publish_P

Description
Publishes a message stored in PROGMEM to the specified topic.

Syntax
boolean PubSubClient::publish_P(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::subscribe

Description
Subscribes to messages published to the specified topic.

Syntax
boolean PubSubClient::subscribe(const char* topic)
boolean PubSubClient::subscribe(const char* topic, uint8_t qos)

Parameters
topic: the topic to subscribe to
qos: the qos to subscribe at

Returns
False – sending the subscribe failed, either connection lost or message too large
True – sending the subscribe succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::unsubscribe

Description
Unsubscribes from the specified topic.

Syntax
boolean PubSubClient::unsubscribe(const char* topic)

Parameters
topic: the topic to unsubscribe to

Returns
False – sending the unsubscribe failed, either connection lost or message too large
True – sending the unsubscribe succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::loop

Description
A must method called regularly to allow the client to process incoming messages and maintain its connection to the server.

Syntax
boolean PubSubClient::loop(void)

Parameters
The function requires no input parameter.

Returns
False – the client is no longer connected
True – the client is still connected

Example Code
Example: MQTT_Basic

Notes and Warnings
A required method that should not be blocked for too long.

PubSubClient::connected

Description
Checks whether the client is connected to the server.

Syntax
boolean PubSubClient::connected(void)

Parameters
The function requires no input parameter.

Returns
False – the client is not connected
True – the client is connected

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::state

Description
Returns the current state of the client. If a connection attempt fails, this can be used to get more information about the failure.
All of the values have corresponding constants defined in PubSubClient.h.

Syntax
int PubSubClient::state(void)

Parameters
The function requires no input parameter.

Returns
-4 : MQTT_CONNECTION_TIMEOUT – the server didn't respond within the keepalive time
-3 : MQTT_CONNECTION_LOST – the network connection was broken
-2 : MQTT_CONNECT_FAILED – the network connection failed
-1 : MQTT_DISCONNECTED – the client is disconnected cleanly
0 : MQTT_CONNECTED – the client is connected
1 : MQTT_CONNECT_BAD_PROTOCOL – the server doesn't support the requested version of MQTT
2 : MQTT_CONNECT_BAD_CLIENT_ID – the server rejected the client identifier
3 : MQTT_CONNECT_UNAVAILABLE – the server was unable to accept the connection
4 : MQTT_CONNECT_BAD_CREDENTIALS – the username/password were rejected
5 : MQTT_CONNECT_UNAUTHORIZED – the client was not authorized to connect

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

Readme

The Ameba MQTT related APIs and examples are works based on the PubSubClient libraries written by Nicholas O'Leary
These include,

- `PubSubClient.cpp`
- `PubSubClient.h`

These libraries are under MIT License.

NTPClient

Readme

The NTPClient library is based on the NTPClient library written by Fabrice Weinberg, which can be found at https://github.com/arduino-libraries/NTPClient.
These include,

- `NTPClient.cpp`
- `NTPClient.h`

These libraries are licensed under MIT License.

PowerSave

Class PMUClass

PMUClass Class

Description
Defines a class of using Power Save API

Syntax
class PMUClass

Members

Public Constructors	
PMUClass::PMUClass	Constructs a PMUClass object
Public Methods	
PMUCLASS::begin	Initialize the PMUCLASS and select sleep mode
PMUCLASS::AONTimerDuration	Set the duration of AON Timer
PMUCLASS::AONTimerCmd	Disable the AON Timer for power save usage
PMUCLASS::RTCWakeSetup	Set up RTC Timer for power save usage
PMUCLASS::enable	Enable power save deep sleep mode
PMUCLASS::AONWakeReason	Check AON wakeup source
PMUCLASS::WakePinCheck	Check AON GPIO pin wakeup source
PMUCLASS::AONWakeClear	Clear all the AON wakeup source
PMUCLASS::DsleepWakeStatusGet	Check if deepsleep mode is set
PMUCLASS::TL_sysactive_time	Tickless mode system active time
PMUCLASS::TL_wakelock	Tickless mode wake lock, select acquire of release
PMUCLASS::DS_AON_TIMER_WAKEUP	Return the Wakeup source
PMUCLASS::DS_RTC_WAKEUP	Return the Wakeup source
PMUCLASS::TL_UART_WAKEUP	Return the Wakeup source
PMUCLASS::TL_RTC_WAKEUP	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA12	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA13	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA14	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA15	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA16	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA17	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA18	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA19	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA20	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA21	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA25	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA26	Return the Wakeup source

PMUCLASS::PMUCLASS

Description
Constructs a PMUCLASS object.

Syntax
PMUCLASS::PMUCLASS(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::begin

Description
Initialize the PMUCLASS and select sleep mode.

Syntax
void PMUClass::begin(uint32_t sleep_mode);

Parameters
sleep_mode: Selection value, “11” enters the DeepSleep Mode, “22” enters the Tickless Mode

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerDuration

Description
Set the duration of AON Timer

Syntax
void PMUClass::AONTimerDuration(uint32_t duration_ms);

Parameters
duration_ms: Timer duration between 0 to 32760000ms.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerCmd

Description
Disable the AON timer for power save usage.

Syntax
void PMUClass::AONTimerCmd(void);

Parameters
c: The value in Celcius.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::RTCWakeSetup

Description
Set up the RTC timer for power save usage.

Syntax
void PMUClass::RTCWakeSetu(uint32_t duration_d, unit32_t duration_h, uint32_t duration_m, uint32_t duration_s);

Parameters
duration_d: Set alarm for number of days from 0.
duration_h: Set alarm for number of hours from 0.
duration_m: Set alarm for number of minutes from 0.
duration_s: Set alarm for number of seconds from0.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::enable

Description
Enable power save deep sleep mode

Syntax
void PMUClass::enable(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeReason

Description
Check the AON wakeup source

Syntax
uint32_t PMUClass::AONWakeReason(void);

Parameters
The function requires no input parameter.

Returns
Returns the value of wakeup deepsleep source. “11” for AON pin, “22” for AON timer, “33” for RTC timer and “0” for none.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::WakePinCheck

Description
Check which AON GPIO pins are the wakeup source

Syntax
int PMUClass::WakePinCheck(void);

Parameters
The function requires no input parameter.

Returns
Return the pin number for indicating Arduino pin names.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeClear

Description
Clear all AON Wakeup source.

Syntax
void PMUClass::AONWakeClear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DsleepWakeStatusGet

Description
Check if deepsleep mode is set.

Syntax
bool PMUClass::DsleepWakeStatusGet(void);

Parameters
The function requires no input parameter.

Returns
Return TRUE when enter DeepSleep Mode or FALSE for negative.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_sysactive_time

Description
Tickless mode system active time.

Syntax
void PMUClass::TL_sysactive_time(uint32_t duration_ms);

Parameters
duration_ms: Set the duration of system active time. The unit is in milliseconds.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_wakelock

Description
Tickless mode wake lock, select acquire or release.

Syntax
void PMUClass::TL_wakelock(uint32_t select_lock);

Parameters
select_lock: Wake lock selection value, “1” for acquire or “0” for release.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_AON_TIMER_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_AON_TIMER_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON Timer as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_RTC_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_UART_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_UART_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns LOGUART as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_RTC_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA12

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA12(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA12 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA13

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA13(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA13 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA14

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA14(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA14 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA15

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA15(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA15 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA16

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA16(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA16 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA17

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA17(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA17 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA18

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA18(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA18 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA19

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA19(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA19 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA20

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA20(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA20 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA21

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA21(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA21 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA25

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA25(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA25 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA26

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA26(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA26 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

RTC

Class RTC

RTC Class

Description
A class used for displaying date and time and alarm configuration using RTC, the independent BCD (Binary-Coded-Decimal) timer.

Syntax
class RTC

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named RTC.	

Public Methods	
RTC:: Init	Initializes the RTC device, including the Clock, the RTC registers, and other functions
RTC:: DeInit	Deinitialize the RTC device
RTC:: Write	Set the specified timestamp in seconds to RTC
RTC:: Read	Get the current timestamp in seconds from RTC
RTC:: Wait	Wait for 1 second
RTC:: SetEpoch	Convert human-readable time to epoch time

RTC::Init

Description
Initializes the RTC device, including the Clock, the RTC registers, and other functions.

Syntax
void RTC::Init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC

```
/*
* This function describes how to use the RTC API.
* The RTC function is implemented by an independent BCD timer/counter.
* This example will print out the time information every second.
*/
#include <stdio.h>
#include <time.h>
#include "rtc.h"
#define YEAR 2020
#define MONTH 9
#define DAY 10
#define HOUR 20
#define MIN 30
#define SEC 40
/* Create an rtc object */
RTC rtc;
int32_t seconds;
struct tm *timeinfo;
void setup() {
Serial.begin(115200);
rtc.Init(); // initialize RTC
}
void loop() {
// step 1: convert user time to epoch
int epochTime = humanReadableToEpoch(YEAR, MONTH, DAY, HOUR, MIN, SEC);
// step 2: write epoch time to rtc
rtc.Write(epochTime);
while (1) {
seconds = rtc.Read();
printf("Epoch Time (in s) since January 1, 1970 = %dsn", seconds);
printf("Time as a basic string = %s", ctime(&seconds));
```

```
timeinfo = localtime(&seconds);
printf("Time as a custom formatted string = %d-%d-%d %d:%d:%dn",
(timeinfo->tm_year + 1900), (timeinfo->tm_mon + 1), timeinfo->tm_mday, timeinfo->tm_hour,
timeinfo->tm_min, timeinfo->tm_sec);
Serial.println();
rtc.wait(1);
}
}
// convert human readable time to epoch time
int humanReadableToEpoch(int year, int month, int day, int hour, int min, int sec) {
struct tm t;
time_t t_of_day;
t.tm_year = year - 1900; // Year - 1970
t.tm_mon = month - 1; // Month, where 0 = jan
t.tm_mday = day; // Day of the month
t.tm_hour = hour;
t.tm_min = min;
t.tm_sec = sec;
t.tm_isdst = -1; // Is DST on? 1 = yes, 0 = no, -1 = unknown
t_of_day = mktime(&t);
// printf("seconds since the Epoch: %dn", (long)t_of_day);
return t_of_day;
}
```

Notes and Warnings
NA

RTC::DeInit

Description
Deinitializes the RTC device.

Syntax
void RTC::DeInit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Write

Description
Set the specified timestamp in seconds to RTC. Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Syntax
void RTC::Write(int t);

Parameters
Parameters
t: Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC::Read

Description
Get the current timestamp in seconds from RTC. The current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Syntax
int32_t RTC::Read(void);

Parameters
The function requires no input parameter.

Returns
The function returns the current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Wait

Description
Send IR raw data.

Syntax
void RTC::wait(float s);

Parameters
s: unit microseconds (1 us)

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: SetEpoch

Description
Convert human-readable time to epoch time

Syntax
int RTC:: SetEpoch(int year, int month, int day, int hour, int min, int sec);

Parameters
year: user input year
month: user input month
day: user input day
hour: user input hour
min: user input minutes
sec: user input seconds

Returns
The function returns epoch time in seconds for RTC use.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

SoftwareSerial

Class Adafruit_GPS

Adafruit_GPS Class

Description
Defines a class to use GPS module on Ameba.

Syntax
class Adafruit_GPS

Members

Public Constructors	
Adafruit_GPS::Adafruit_GPS	Constructs an Adafruit_GPS object
Public Methods	
Adafruit_GPS::begin	Initialize serial communication
*Adafruit_GPS:: lastNMEA	Returns the last NMEA line received and unsets the received flag
Adafruit_GPS:: newNMEAreceived	Check to see if a new NMEA line has been received
Adafruit_GPS:: common_init	Initialization code used by all constructor types
Adafruit_GPS:: sendCommand	Send a command to the GPS device
Adafruit_GPS:: pause	Pause/unpause receiving new data
Adafruit_GPS:: parseHex	Read a Hex value and return the decimal equivalent
Adafruit_GPS:: read	Read one character from the GPS device
Adafruit_GPS:: parse	Parse an NMEA string
Adafruit_GPS:: wakeup	Wake the sensor up
Adafruit_GPS:: standby	Standby Mode Switches
Adafruit_GPS::waitForSentence	Wait for a specified sentence from the device
Adafruit_GPS::LOCUS_StartLogger	Start the LOCUS logger
Adafruit_GPS::LOCUS_StopLogger	Stop the LOCUS logger
Adafruit_GPS::LOCUS_ReadStatus	Read the logger status

Adafruit_GPS::Adafruit_GPS

Description

Constructs an Adafruit_GPS object and initialize serial using a SoftSerial object.

Syntax

Adafruit_GPS::Adafruit_GPS(SoftwareSerial *ser)

Adafruit_GPS::Adafruit_GPS(HardwareSerial *ser)

Parameters

ser: a Serial instance

Returns

The function returns nothing.

Example Code

Example: Adafruit_GPS_parsing

This example code from Adafruit demonstrates GPS modules using MTK3329/MTK3339 driver. This code shows how to listen to the GPS module in an interrupt which allows the program to have more 'freedom' – just parse when a new NMEA sentence is available! Then access data when desired.

#include <Adafruit_GPS.h>

#include <SoftwareSerial.h>

// If you're using a GPS module:

// Connect the GPS Power pin to 3.3V

```
// Connect the GPS Ground pin to ground
// Connect the GPS TX (transmit) pin to Digital 0
// Connect the GPS RX (receive) pin to Digital 1
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1);
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RTL8710 need change GPS TX/RX to pin 17 and 5
#else
SoftwareSerial mySerial(0, 1);
#endif
Adafruit_GPS GPS(&mySerial);
// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences.
#define GPSECHO false
void setup()
{
Serial.begin(38400);
Serial.println("Adafruit GPS library basic test!");
// 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
GPS.begin(9600);
// uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitude
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
// uncomment this line to turn on only the "minimum recommended" data
//GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
// For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
// the parser doesn't care about other sentences at this time
// Set the update rate
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
// For the parsing code to work nicely and have time to sort thru the data, and
// print it out we don't suggest using anything higher than 1 Hz
// Request updates on antenna status, comment out to keep quiet
GPS.sendCommand(PGCMD_ANTENNA);
delay(1000);
// Ask for firmware version
mySerial.println(PMTK_Q_RELEASE);
}
```

```
uint32_t timer = millis();
void loop() // run over and over again
{
// in case you are not using the interrupt above, you'll
// need to 'hand query' the GPS, not suggested :(
// read data from the GPS in the 'main loop'
char c = GPS.read();
// if you want to debug, this is a good time to do it!
if (GPSECHO)
if (c) Serial.print(c);
// if a sentence is received, we can check the checksum, parse it…
if (GPS.newNMEAreceived()) {
// a tricky thing here is if we print the NMEA sentence, or data
// we end up not listening and catching other sentences!
// so be very wary if using OUTPUT_ALLDATA and trytng to print out data
//Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false
if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to false
return; // we can fail to parse a sentence in which case we should just wait for another
}
// if millis() or timer wraps around, we'll just reset it
if (timer > millis()) timer = millis();
// approximately every 2 seconds or so, print out the current stats
if (millis() - timer > 2000) {
timer = millis(); // reset the timer
Serial.print("nTime: ");
Serial.print(GPS.hour, DEC); Serial.print(':');
Serial.print(GPS.minute, DEC); Serial.print(':');
Serial.print(GPS.seconds, DEC); Serial.print('.');
Serial.println(GPS.milliseconds);
Serial.print("Date: ");
Serial.print(GPS.day, DEC); Serial.print('/');
Serial.print(GPS.month, DEC); Serial.print("/20");
Serial.println(GPS.year, DEC);
Serial.print("Fix: "); Serial.print((int)GPS.fix);
Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
if (GPS.fix) {
```

Serial.print(“Location: “);

Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);

Serial.print(”, “);

Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);

Serial.print(“Location (in degrees, works with Google Maps): “);

Serial.print(GPS.latitudeDegrees, 4);

Serial.print(”, “);

Serial.println(GPS.longitudeDegrees, 4);

Serial.print(“Speed (knots): “); Serial.println(GPS.speed);

Serial.print(“Angle: “); Serial.println(GPS.angle);

Serial.print(“Altitude: “); Serial.println(GPS.altitude);

Serial.print(“Satellites: “); Serial.println((**int**)GPS.satellites);

}

}

}

Notes and Warnings
IMPORTANT: SoftSerial is using hardware serial so pin mapping cannot be altered.

Adafruit_GPS::begin

Description
Initialize serial communication

Syntax
void Adafruit_GPS::begin(uint16_t baud)

Parameters
baud: serial baud rate

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

***Adafruit_GPS::lastNMEA**

Description
Returns the last NMEA line received and unsets the received flag

Syntax
char *Adafruit_GPS::lastNMEA(void)

Parameters
The function requires no input parameter.

Returns
Pointer to the last line string

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::newNMEAreceived

Description
Check to see if a new NMEA line has been received

Syntax
boolean Adafruit_GPS::newNMEAreceived(void)

Parameters
The function requires no input parameter.

Returns
True if received, false if not

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::common_init

Description
Initialization code used by all constructor types

Syntax
void Adafruit_GPS::common_init(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::sendCommand

Description
Send a command to the GPS device

Syntax
void Adafruit_GPS::sendCommand(const char *str)

Parameters
str: Pointer to a string holding the command to send

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::pause

Description
Pause/unpause receiving new data

Syntax
void Adafruit_GPS::pause(boolean p)

Parameters
p: True = pause, false = unpause

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::parseHex

Description
Read a Hex value and return the decimal equivalent

Syntax
uint8_t Adafruit_GPS::parseHex(char c)

Parameters
c: Hex value

Returns
The decimal equivalent of the Hex value

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::read

Description
Read one character from the GPS device

Syntax
char Adafruit_GPS::read(void)

Parameters
The function requires no input parameter.

Returns
The character that we received, or 0 if nothing was available

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::parse

Description
Parse an NMEA string

Syntax
boolean Adafruit_GPS::parse(char *nmea)

Parameters
nmea: an NMEA string

Returns
True if we parsed it, false if it has invalid data

Example Code
Example: Adafruit_GPS_parsing

Notes and Warnings
NA

Adafruit_GPS::wakeup

Description
Wake the sensor up

Syntax
boolean Adafruit_GPS::wakeup(void)

Parameters
The function requires no input parameter.

Returns
True if woken up, false if not in standby or failed to wake

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::standby

Description
Standby Mode Switches

Syntax
boolean Adafruit_GPS::standby(void)

Parameters
The function requires no input parameter.

Returns
False if already in standby, true if it entered standby

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::waitForSentence

Description
Wait for a specified sentence from the device

Syntax
boolean Adafruit_GPS::waitForSentence(const char *wait4me, uint8_t max)

Parameters
wait4me: Pointer to a string holding the desired response
max: How long to wait, default is MAXWAITSENTENCE

Returns
True if we got what we wanted, false otherwise

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StartLogger

Description
Start the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StartLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StopLogger

Description
Stop the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StopLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings

NA

Adafruit_GPS::LOCUS_ReadStatus

Description
Read the logger status

Syntax
boolean Adafruit_GPS::LOCUS_ReadStatus(void)

Parameters
The function requires no input parameter.

Returns
True if we read the data, false if there was no response

Example Code
NA

Notes and Warnings
NA

Class HttpClient

PMS3003 Class

Description
Defines a class to work with PMS3003 air quality sensor on Ameba.

Syntax
class PMS3003

Members

Public Constructors	
PMS3003::PMS3003	Constructs a PMS3003 object
Public Methods	
PMS3003::begin	Initialize hardware UART
PMS3003::end	Free allocated space thus stopping UART
PMS3003::get_pm1p0_cf1	Get PM1.0 under correction factor = 1
PMS3003:: get_pm2p5_cf1	Get PM2.5 under correction factor = 1
PMS3003:: get_pm10_cf1	Get PM10 under correction factor = 1
PMS3003:: get_pm1p0_air	Get PM1.0 air quality
PMS3003:: get_pm2p5_air	Get PM2.5 air quality
PMS3003:: get_pm10_air	Get PM10 air quality
PMS3003:update_cache	Updates the cache memory
PMS3003::pms3003_handle_interrupt	Set up the serial event handler

PMS3003::PMS3003

Description
Constructs a PMS3003 object and initialize the pin mapping.

Syntax
PMS3003::PMS3003(int _rx, int _tx, int _set, int _reset)

Parameters
_rx: RX pin of UART
_tx: TX pin of UART
_set: Set pin
_reset: Reset pin

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::begin

Description
Initialize hardware UART and allocate space for serial buffer

Syntax
void PMS3003::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::end

Description
Free serial buffer space and stop UART

Syntax
void PMS3003::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_cf1

Description
Get PM1.0 under correction factor = 1

Syntax
int PMS3003::get_pm1p0_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value “pm1p0_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm2p5_cf1

Description
Get PM2.5 under correction factor = 1

Syntax
int PMS3003::get_pm2p5_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_cf1

Description
Get PM10 under correction factor = 1

Syntax
int PMS3003::get_pm10_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_air

Description
Get PM1.0 air quality

Syntax
int PMS3003::get_pm1p0_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm1p0_air” as an integer.

Example Code
NA

Notes and Warnings

NA

PMS3003::get_pm2p5_air

Description
Get PM2.5 air quality

Syntax
int PMS3003::get_pm2p5_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_air” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_air

Description
Get PM10 air quality

Syntax
int PMS3003::get_pm10_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_air” as an integer.

Example Code

NA

Notes and Warnings
NA

PMS3003::pms3003_handle_interrupt

Description
Set up the serial event handler

Syntax
void pms3003_handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: device identifier
event: Serial event for handling incoming data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::update_cache

Description
Serves the function of updating cache memory. One package has 32 bytes. Illustrate the formate by using below raw data: 42 4d 00 1c 00 1b 00 21 00 29 00 1a 00 21 00 29 2b fb 04 be 00 6b 00 10 00 04 00 04 67 00 04 46
42 4d : header signature
00 1c : frame length, 0x001c = 28 bytes (not include header and this field)
00 1b : PM1.0 under CF=1
00 21 : PM2.5 under CF=1
00 29 : PM10 under CF=1
00 1a : PM1.0 under air
00 21 : PM2.5 under air
00 29 : PM10 under air

2b fb : number of pariticle, diameter size 0.3 um in 0.1 liter air
04 be : number of pariticle, diameter size 0.5 um in 0.1 liter air
00 6b : number of pariticle, diameter size 1.0 um in 0.1 liter air
00 10 : number of pariticle, diameter size 2.5 um in 0.1 liter air
00 04 : number of pariticle, diameter size 5.0 um in 0.1 liter air
00 04 : number of pariticle, diameter size 10 um in 0.1 liter air
67 : serial number
00 : error code
04 46 :
checksum,0x42+0x4d+0x00+0x1c+0x00+0x1b+0x00+0x21+0x00+0x29+0x00+0x1a+0x00+0x21+0x00+0x29+
0x2b+0xfb+0x04+0xbe+0x00+0x6b+0x00+0x10+0x00+0x04+0x00+0x04+0x67+0x00 = 0x0446

Syntax
void PMS3003::update_cache(void)

Parameters
The function requires no input parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Class SoftwareSerial

SoftwareSerial Class

Description
Defines a class of software serial implementation for Ameba.

Syntax
class SoftwareSerial

Members

Public Constructors	
SoftwareSerial::SoftwareSerial	Constructs a SoftwareSerial object
Public Methods	
SoftwareSerial::begin	Sets the speed (baud rate) for the serial communication
SoftwareSerial::listen	Enables the selected software serial port to listen
SoftwareSerial::end	Same as stopListening
SoftwareSerial::stopListening	Stop listening on the port
SoftwareSerial::peek	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::write	Prints data to the transmit pin of the software serial port as raw bytes
SoftwareSerial::read	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::available	Get the number of bytes (characters) available for reading from a software serial port
SoftwareSerial::flush	Flush the received buffer
SoftwareSerial::setBufferSize	Set buffer size
SoftwareSerial::setAvailableCallback	Set available callback
SoftwareSerial::handle_interrupt	Private methods handles interrupt

SoftwareSerial::SoftwareSerial

Description

Constructs a SoftwareSerial object and sets RX and TX pin, and inverse logic.

Syntax

SoftwareSerial::SoftwareSerial(uint8_t receivePin, uint8_t transmitPin, bool inverse_logic /* = false */)

Parameters

receivePin: the pin on which to receive serial data
transmitPin: the pin on which to transmit serial data
inverse_logic: is used to invert the sense of incoming bits

Returns

The function returns nothing.

Example Code

Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX.

/*

The circuit: (BOARD RTL8195A)

* RX is digital pin 0 (connect to TX of other devices)

* TX is digital pin 1 (connect to RX of other devices)

*/

```
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(57600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
Serial.println("Goodnight moon!");
// set the data rate for the SoftwareSerial port
mySerial.begin(4800);
mySerial.println("Hello, world?");
}
void loop() { // run over and over
if (mySerial.available()) {
mySerial.write(mySerial.read());
}
}
```

Notes and Warnings
Software Serial is using hardware serial thus DO NOT change the default pins

SoftwareSerial::begin

Description
Sets the speed (baud rate) for the serial communication

Syntax
void SoftwareSerial::begin(long speed)
void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits)

void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits, int flowctrl, int rtsPin, int ctsPin)

Parameters
speed: the baud rate
data_bits: number of data bits, 8 bits(default) or 7 bits
stop_bits: number of stop bits, 1 bit(default), 1.5 bits or 2 bits
flowctrl: flow control pin
rtsPin: request to send pin
ctsPin: clear to send pin

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial_Basic:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::listen

Description
Enables the selected software serial port to listen

Syntax
bool SoftwareSerial::listen(void)

Parameters
The function requires no input parameter.

Returns
Returns true if it replaces another

Example Code
NA

Notes and Warnings

NA

SoftwareSerial::end

Description
Same as stopListening

Syntax
void SoftwareSerial::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::isListening

Description
Tests to see if requested software serial port is actively listening

Syntax
bool SoftwareSerial::isListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if the port is listening.

Example Code

NA

Notes and Warnings
NA

SoftwareSerial::stopListening

Description
Stop listening on the port

Syntax
bool SoftwareSerial::stopListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if listening on the port is stopped.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::peek

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::peek(void)

Parameters
The function requires no input parameter.

Returns

The function returns the character read, or returns "-1" if none is available.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::write

Description
Prints data to the transmit pin of the software serial port as raw bytes

Syntax
size_t SoftwareSerial::write(uint8_t b)

Parameters
b: byte to be written

Returns
The function returns the number of bytes written.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::read

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::read(void)

Parameters
The function requires no input parameter.

Returns
The function returns the character read, or -1 if none is available.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::available

Description
Get the number of bytes available for reading from a software serial port

Syntax
int SoftwareSerial::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available to read.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::flush

Description
Flush the received buffer

Syntax
void SoftwareSerial::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setBufferSize

Description
Set buffer size

Syntax
void SoftwareSerial::setBufferSize(uint32_t buffer_size)

Parameters
buffer_size: the size of the serial buffer

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setAvailableCallback

Description
Set available callback

Syntax
void SoftwareSerial::setAvailableCallback(void (*callback)(char c))

Parameters
*callback: user-defined serial callback function

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialIrqCallback
This example demonstrates the software serial testing using IRQ callback and semaphore. Set callback function "mySerialCalback" to software serial. Whenever there is data comes in, "mySerialCallback" is invoked. In this sketch, it does nothing until the end of the line. And then it sends a semaphore. The loop() uses a non-busy loop to wait for the semaphore. To test this sketch, you need to type something on software serial and then press Enter.

```
/*
The circuit: (BOARD RTL8195A)
RX is digital pin 0 (connect to TX of other devices)
TX is digital pin 1 (connect to RX of other devices)
*/
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
uint32_t semaID;
// The callback is hooking at UART IRQ handler and please don't do heavy task here.
void mySerialCallback(char c)
{
```

```
/* The parameter c is only for peeking. The actual data is
* still in the buffer of SoftwareSerial.
*/
if (c == 'r' || c == 'n') {
os_semaphore_release(semaID);
}
}
void setup() {
// use 1 count for binary semaphore
semaID = os_semaphore_create(1);
// There is a token in the semaphore, clear it.
os_semaphore_wait(semaID, 0xFFFFFFFF);
// set the data rate for the SoftwareSerial port
mySerial.begin(38400);
mySerial.setAvailableCallback(mySerialCallback);
}
void loop() { // run over and over
// wait semaphore for 5s timeout
if (os_semaphore_wait(semaID, 5 * 1000)) {
// we got data before timeout
while(mySerial.available()) {
mySerial.print((char)mySerial.read());
}
mySerial.println();
} else {
mySerial.println("No data comes in.");
}
}
```

Notes and Warnings
NA

SoftwareSerial::handle_interrupt

Description
A private method handles the interrupt

Syntax
void handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: the interupt id
event: interrupt event

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba Software Serial related APIs and examples are works based on libraries formerly known as `NewSoftSerial.h` by Mikal Hart (http://arduiniana.org/libraries/newsoftserial).
These include,

- `SoftwareSerial.cpp`
- `SoftwareSerial.h`

These libraries are under GNU Lesser General Public License.

The Ameba GPS related APIs and examples are works based on Adafruit GPS library written by Limor Fried/Ladyada for Adafruit Industries (http://www.adafruit.com/products/746).
These include,

- `Adafruit_GPS.cpp`
- `Adafruit_GPS.h`

These libraries are under BSD License.

SPI

Class AmebaILI9341

AmebaILI9341 Class

Description

Defines a class to use ILI9341 TFT SPI display for Ameba.

Syntax

class AmebaILI9341

Members

Public Constructors	
AmebaILI9341::AmebaILI9341	Constructs an AmebaILI9341 object
Public Methods	
AmebaILI9341::begin	Initialize SPI, pin mapping and screen configuration
AmebaILI9341::setAddress	Initialize image size and position
AmebaILI9341::writecommand	SPI transfer a command
AmebaILI9341::writedata	SPI transfer a piece of data
AmebaILI9341::setRotation	Set screen orientation
AmebaILI9341::fillScreen	Fill the screen with a color
AmebaILI9341::clr	Clear screen
AmebaILI9341::fillRectangle	Fill a rectangular space with a color
AmebaILI9341::drawPixel	Turn on a pixel on the screen
AmebaILI9341::drawChar	To print a character on the screen
AmebaILI9341::drawLine	Draw line on the screen
AmebaILI9341::drawRectangle	Draw a rectangle on the screen
AmebaILI9341::drawCircle	Draw a circle on the screen
AmebaILI9341::write	Same as drawChar
AmebaILI9341::getWidth	Return the width 240
AmebaILI9341::getHeight	Return the height 320
AmebaILI9341::setCursor	Set cursor to the desired position
AmebaILI9341::setForeground	Set foreground color
AmebaILI9341::setBackground	Set background color
AmebaILI9341::setFontSize	Set character font size
AmebaILI9341::reset	Reset pin to High or Low

AmebaILI9341::AmebaILI9341

Description

Constructs an AmebaILI9341 object and set CS, DC and RESET pins .

Syntax

AmebaILI9341::AmebaILI9341(int csPin, int dcPin, int resetPin)

Parameters

csPin: pin for Chip Select dcPin: pin for Data/Command resetPin: pin for Reset

Returns

The function returns nothing.

Example Code

Example: : PM25_ON_ILI9341_TFT_LCD

This example demonstrates how to read pm2.5 value on PMS 3003 air-condition sensor and display it on ILI9341 TFT LCD.

/*

PMS 3003 pin map is as follow:

PIN1 :VCC, connect to 5V

PIN2 :GND

PIN3 :SET, 0:Standby mode, 1:operating mode

PIN4 :RXD :Serial RX

PIN5 :TXD :Serial TX

PIN6 :RESET

PIN7 :NC

PIN8 :NC

In this example, we only use Serial to get PM 2.5 value.

The circuit:

* RX is digital pin 0 (connect to TX of PMS 3003)

* TX is digital pin 1 (connect to RX of PMS 3003)

For RTL8195A ILI9341 TFT LCD with SPI interface has these pins:

D/C : connect to pin 9

CS : connect to pin 10

MOSI : connect to pin 11

MISO : connect to pin 12

CLK : connect to pin 13

VCC : connect to 3V3

GND : connect to GND

*/

#include "SoftwareSerial.h"

#include "SPI.h"

#include "AmebaILI9341.h"

#if defined(BOARD_RTL8195A)

SoftwareSerial mySerial(0, 1); // RX, TX

#define TFT_RESET 8

#define TFT_DC 9

#define TFT_CS 10

#elif defined(BOARD_RTL8710)

```
SoftwareSerial mySerial(17, 5); // RX, TX
// IMPORTANT: Due to limit pin, we do not connect TFT_RESET pin.
#define TFT_RESET 0xFFFFFFFF
#define TFT_DC 2
#define TFT_CS 10
#endif
AmebaILI9341 tft = AmebaILI9341(TFT_CS, TFT_DC, TFT_RESET);
#define ILI9341_SPI_FREQUENCY 20000000
#define pmsDataLen 32
uint8_t buf[pmsDataLen];
int idx = 0;
int pm10 = 0;
int last_pm25 = 0;
int pm25 = 0;
int pm100 = 0;
uint16_t pm25color[] = {
0x9FF3,
0x37E0,
0x3660,
0xFFE0,
0xFE60,
0xFCC0,
0xFB2C,
0xF800,
0x9800,
0xC99F
};
void setup() {
Serial.begin(57600);
mySerial.begin(9600); // PMS 3003 UART has baud rate 9600
SPI.setDefaultFrequency(ILI9341_SPI_FREQUENCY);
tft.begin();
drawPictureFrames();
}
void loop() { // run over and over
uint8_t c;
```

```
idx = 0;
memset(buf, 0, pmsDataLen);
while (true) {
while (c != 0x42) {
while (!mySerial.available());
c = mySerial.read();
}
while (!mySerial.available());
c = mySerial.read();
if (c == 0x4d) {
// now we got a correct header)
buf[idx++] = 0x42;
buf[idx++] = 0x4d;
break;
}
}
while (idx != pmsDataLen) {
while(!mySerial.available());
buf[idx++] = mySerial.read();
}
pm10 = ( buf[10] << 8 ) | buf[11];
last_pm25 = pm25;
pm25 = ( buf[12] << 8 ) | buf[13];
pm100 = ( buf[14] << 8 ) | buf[15];
updateValueToTftScreen();
}
void drawPictureFrames() {
tft.setRotation(1);
tft.clr();
tft.setFontSize(1);
// Upper title
tft.setFontSize(1);
tft.setCursor(20,20);
tft.print("PM2.5 DETECTOR");
// PM 2.5 Circle Frame
tft.drawCircle(100,130,60, ILI9341_BLUE);
```

```
tft.drawCircle(100,130,61, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(90,85);
tft.print("PM2.5");
tft.setFontSize(1);
tft.setCursor(90,170);
tft.print("um/m3");
// PM 10 Circle Frame
tft.drawCircle(220,70,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(210,40);
tft.print("PM10");
tft.setFontSize(1);
tft.setCursor(205,95);
tft.print("um/m3");
// PM 1.0 Circle Frame
tft.drawCircle(220,170,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(205,140);
tft.print("PM1.0");
tft.setFontSize(1);
tft.setCursor(205,195);
tft.print("um/m3");
// right side bar, referenced from: http://taqm.epa.gov.tw/taqm/tw/
tft.fillRectangle(290, 30+ 0*2, 10, 12*2, pm25color[0]); // 0~11
tft.fillRectangle(290, 30+12*2, 10, 12*2, pm25color[1]); // 12-23
tft.fillRectangle(290, 30+24*2, 10, 12*2, pm25color[2]); // 24-35
tft.fillRectangle(290, 30+36*2, 10, 6*2, pm25color[3]); // 36-41
tft.fillRectangle(290, 30+42*2, 10, 6*2, pm25color[4]); // 42-47
tft.fillRectangle(290, 30+48*2, 10, 6*2, pm25color[5]); // 48-53
tft.fillRectangle(290, 30+54*2, 10, 6*2, pm25color[6]); // 54-58
tft.fillRectangle(290, 30+59*2, 10, 6*2, pm25color[7]); // 59-64
tft.fillRectangle(290, 30+65*2, 10, 6*2, pm25color[8]); // 65-70
tft.fillRectangle(290, 30+71*2, 10, 10*2, pm25color[9]); // >=71
tft.setCursor(302, 30);
tft.setFontSize(1);
```

```
tft.print("0");
tft.setCursor(302, 30+36*2);
tft.print("36");
tft.setCursor(302, 30+54*2);
tft.print("54");
tft.setCursor(302, 30+71*2);
tft.print("71");
// bottom right text
tft.setCursor(210,230);
tft.setFontSize(1);
tft.print("Powered by Realtek");
updateValueToTftScreen();
}
void updateValueToTftScreen() {
tft.setCursor(60, 111);
tft.setFontSize(5);
tft.setForeground( getPm25Color(pm25) );
if (pm25 < 10) {
tft.print(" ");
} else if (pm25 < 100) {
tft.print(" ");
}
tft.print(pm25);
tft.setCursor(195,60);
tft.setFontSize(3);
if (pm100 < 10) {
tft.print(" ");
} else if (pm100 < 100) {
tft.print(" ");
}
tft.print(pm100);
tft.setCursor(198,160);
if (pm10 < 10) {
tft.print(" ");
} else if (pm10 < 100) {
tft.print(" ");
```

```
}
tft.print(pm10);
tft.setFontSize(1);
tft.setForeground(ILI9341_WHITE);
if (last_pm25 > 80) {
tft.fillRectangle(275, 80*2+30-3, 12, 8, ILI9341_BLACK);
} else {
tft.fillRectangle(275, last_pm25*2+30-3, 12, 8, ILI9341_BLACK);
}
if (pm25 > 80) {
tft.setCursor(275, 80*2+30-3);
} else {
tft.setCursor(275, pm25*2+30-3);
}
tft.print("=>");
}
uint16_t getPm25Color(int v) {
if (v < 12) {
return pm25color[0];
} else if (v < 24) {
return pm25color[1];
} else if (v < 36) {
return pm25color[2];
} else if (v < 42) {
return pm25color[3];
} else if (v < 48) {
return pm25color[4];
} else if (v < 54) {
return pm25color[5];
} else if (v < 59) {
return pm25color[6];
} else if (v < 65) {
return pm25color[7];
} else if (v < 71) {
return pm25color[8];
} else {
```

return pm25color[9];

}

}

Notes and Warnings

NA

AmebaILI9341::begin

Description

Initialize hardware SPI, pin mapping and screen configuration

Syntax

void AmebaILI9341::begin(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

This method is required to run first before other operations on the display.

AmebaILI9341::setAddress

Description

Initialize image size and positioning on the display

Syntax

void AmebaILI9341::setAddress(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: rightmost coordinate of the image y1: bottom coordinate of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Do not use this to set the cursor, use the “setCursor” method instead.

AmebaILI9341::writecommand

Description

Write a single-byte command to display

Syntax

void AmebaILI9341::writecommand(uint8_t command)

Parameters

command: a single byte command

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::writedata

Description

Write 1 byte of data to display

Syntax

void AmebaILI9341::writedata(uint8_t data)

Parameters

data: 1 byte data

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this method to write 1 byte at a time.

AmebaILI9341::setRotation

Description

Setting screen orientation, “0” for no rotation, “1” for 90 degrees rotation and so on so forth.

Syntax

void AmebaILI9341::setRotation(uint8_t m)/span> **Parameters**

m: one of the 4 rotation modes -> “0” for no rotation, “1” for 90^0, “2” for 180^0, “3” for 270^0

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

if m=4, it’s equivalent to mode 0, and m=5 for mode 1, m=6 for mode 2 so on so forth.

AmebaILI9341::fillScreen

Description

Fill the entire screen with one color

Syntax

void AmebaILI9341::fillScreen(uint16_t color)

Parameters

color: a 16-bit color reference defined in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Refer to AmebaILI9341.h for available colors.

AmebaILI9341::clr

Description

Fill the entire screen with a certain background-color

Syntax

void AmebaILI9341::clr(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341

Notes and Warnings

background-color can be set by calling setBackground method.

AmebaILI9341::fillRectangle

Description

Fill a rectangular space with a color on the screen

Syntax

void AmebaILI9341::fillRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::drawPixel

Description

Turn on a pixel on the screen

Syntax

void AmebaILI9341::drawPixel(int16_t x, int16_t y, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawChar

Description

Draw character on the screen

Syntax

void AmebaILI9341::drawChar(unsigned char c) void AmebaILI9341::drawChar(int16_t x, int16_t y, unsigned char c, uint16_t _fontcolor, uint16_t _background, uint8_t _fontsize)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image c: a character _fontcolor: font color _background: background color _fontsize: font size

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In the actual example, the Print method is used to print a string of character on the screen instead of using this method.

AmebaILI9341::drawLine

Description

Draw a straight line on the screen

Syntax

void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1) void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: leftmost coordinate of the image y1: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawRectangle

Description

Draw a rectangular shape on the screen

Syntax

void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h) void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawCircle

Description

Draw a circular shape on the screen

Syntax

void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r) void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image r: radius of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Include “AmebaServo.h” to use the class function.

AmebaILI9341::write

Description

Same as drawChar, write a character on the screen

Syntax

size_t AmebaILI9341::write(uint8_t c)

Parameters

c: a character to be written on the screen

Returns

Number of bytes written

Example Code

NA

Notes and Warnings

This an inherited method from Print class and is seldom used.

AmebaILI9341::getWidth

Description

Get the width of the image

Syntax

int16_t AmebaILI9341::getWidth(void)

Parameters

The function requires no input parameter.

Returns

Width of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::getHeight

Description

Get the height of the image

Syntax

int16_t AmebaILI9341::getHeight(void)

Parameters

The function requires no input parameter.

Returns

Height of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::setCursor

Description

Set the cursor to a specific position on the screen

Syntax

void AmebaILI9341::setCursor(int16_t x, int16_t y)

Parameters

x: coordinate on the x-axis y: coordinate on the y-axis

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setForeground

Description

Set foreground color

Syntax

void AmebaILI9341::setForeground(uint16_t color)

Parameters

color: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setBackground

Description

Set background color

Syntax

void AmebaILI9341::setBackground(uint16_t _background)

Parameters

_background: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setFontSize

Description

Set the font size of the characters printed on the screen.

Syntax

void AmebaILI9341::setFontSize(uint8_t size)

Parameters

size: font size, default 1 for smallest, 5 for largest font size

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::reset

Description

Reset the pin to High or Low

Syntax

void AmebaILI9341::reset(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

Class SPISettings_SPIClass

SPISettings Class

Description
Defines a class to set SPI parameters.

Syntax
class SPISettings

Members

Public Constructors	
SPISettings::SPISettings	Create a SPISettings object and set SPI clock speed, bit order and data mode

SPISettings::SPISettings

Description
Construct an object and configure SPI parameters — clock speed, bit order and data model to the preferred default value.

Syntax
SPISettings YourObject(uint32_t clock, BitOrder bitOrder, uint8_t dataMode);

Parameters
clock: SPI clock speed, default is 4000000
bitOrder: order of bit stream, MSB first or LSB first, default is MSBFIRST
dataMode: There are 4 modes -> SPI_MODE0~3, default is SPI_MODE0

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class seldom used alone, it is always used with beginTransaction() as a parameter in SPIClass.

SPIClass Class

Description
Defines a class of SPI implementation for Ameba.

Syntax
class SPIClass

Members

Public Constructors	
SPIClass::SPIClass	Constructs an SPI object
Public Methods	
SPIClass::transfer	Transfer data through SPI
SPIClass::transfer16	Transfer a 16-bits data through SPI
SPIClass::beginTransaction	Set slave select pin and SPI initial settings
SPIClass::endTransaction	Stop SPI transaction
SPIClass::begin	Associate each SPI pin to Ameba pin using ameba HAL APIs
SPIClass::end	Stop SPI master mode
SPIClass::setBitOrder	Set MSB first or LSB first
SPIClass::setDataMode	Set to one of the four data modes
SPIClass::setClockDivider	Set to correct clock speed (no effect on Ameba)
SPIClass::setDefaultFrequency	Set default SPI frequency

SPIClass::SPIClass

Description
Construct an SPI object, create a pointer to the object, and attach “MOSI, MISO, CLK, and SS” to each pin on Ameba.

Syntax
SPIClass(void *pSpiObj, int mosi, int miso, int clk, int ss);

Parameters
pSpiObj: SPI pointer to the object
mosi: master out slave in
miso: master in slave out
clk: clock
ss: slave select

Returns
The function returns nothing.

Example Code

SPIClass SPI((void *)(&spi_obj0), 11, 12, 13, 10);

Notes and Warnings
2 SPI objects are created in the library for 2 different hardware SPI on Ameba (if applicable), use "SPI" for first hardware SPI and "SPI1" for the second.

SPIClass::transfer

Description
Calling HAL API to send data in the buffer to the slave

Syntax
byte SPIClass::transfer (byte _pin, uint8_t _data, SPITransferMode _mode);
byte SPIClass::transfer (uint8_t _data, SPITransferMode _mode);
void SPIClass::transfer (byte _pin, void *_buf, size_t _count, SPITransferMode _mode);
void SPIClass::transfer (void *_buf, size_t _count, SPITransferMode _mode);

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode
_count: number of bytes of data
_buf: data buffer

Returns
Void or "0" in case of error, "d" in case success

Example Code
NA

Notes and Warnings
NA

SPIClass::transfer16

Description
Same as "transfer" method above except data being of 16-bits.

Syntax
uint16_t SPIClass::transfer16(byte _pin, uint16_t _data, SPITransferMode _mode)
uint16_t SPIClass::transfer16(uint16_t _data, SPITransferMode _mode)

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode

Returns
The data being transferred

Example Code
NA

Notes and Warnings
NA

SPIClass::beginTransaction

Description
Set slave select pin and initialize SPI with default settings using SPISettings class.

Syntax
void SPIClass::beginTransaction(uint8_t pin, SPISettings settings)
void SPIClass::beginTransaction(SPISettings settings)

Parameters
pin: slave select pin
settings: an object of SPISettings class

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to SPISettings class for details of the initial settings.

SPIClass::endTransaction

Description
Set slave select pin to 1 and stop SPI transaction.

Syntax
void SPIClass::endTransaction(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::begin

Description
Calling HAL APIs to initialize SPI pins to physical Ameba pins and set SPI format and frequency

Syntax
void SPIClass::begin(void)
void SPIClass::begin(int ss)

Parameters
void or ss: slave select

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
This is a required method to use SPI on Ameba.

SPIClass::end

Description
Free hardware SPI from any activity.

Syntax
void SPIClass::end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setBitOrder

Description
A specific method to set bit order to either MSB first or LSB first and set slave select pin.

Syntax
void SPIClass::setBitOrder(uint8_t _pin, BitOrder _bitOrder)
void SPIClass::setBitOrder(BitOrder _order)

Parameters
_pin: slave select
_bitOrder: bit order -> either MSB first or LSB first
_order: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setDataMode

Description
A specific method to set data mode to one of the 4 modes (default: SPI_MODE0) and set slave lave select pin.

Syntax
void SPIClass::setDataMode(uint8_t _pin, uint8_t _mode)
void SPIClass::setDataMode(uint8_t _mode)

Parameters
_pin: slave select
_mode: one of the 4 modes (default: SPI_MODE0)

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setClockDivider

Description
A specific method to set to divider in order to get correct clock speed

Syntax
void SPIClass::setClockDivider(uint8_t _pin, uint8_t _divider)

void SPIClass::setClockDivider(uint8_t _div)

Parameters
_pin: slave select
_divider: clock divider
_div: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function does not affect the Ameba board.

SPIClass::setDefaultFrequency

Description
A specific method to set default SPI frequency

Syntax
void SPIClass::setDefaultFrequency(int _frequency)

Parameters
_frequency: the default SPI frequency

Returns
The function returns nothing.

Example Code
Example: PM25_on_ILI9341_TFT_LCD
Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings
Take note that defaultFrequency = _frequency.

Readme

The Ameba SPI related APIs and examples are works based on SPI Master library for arduino written by Cristian Maglie <c.maglie@arduino.cc> and Paul Stoffregen <paul@pjrc.com> (Transaction API).

These include,
SPI.cpp
SPI.h

These libraries are under GNU Lesser General Public License, version 2.1.

Sys

Wiring_OS_API

Wiring OS API

Description
A wrapper to CMSIS (Cortex Microcontroller Software Interface Standard) OS API which serve as a RTOS to create multi-threaded application with real-time behaviour.

Syntax
NA

Members

Public Methods	
os_thread_create_arduino	Create a thread and add it to Active Threads and set it to state READY
os_thread_get_id_arduino	Return the thread ID of the current running thread
os_thread_terminate_arduino	Terminate execution of a thread and remove it from Active Threads
os_thread_yield_arduino	Pass control to next thread that is in state READY
os_thread_set_priority_arduino	Change priority of an active thread
os_thread_get_priority_arduino	Get current priority of an active thread
os_signal_set_arduino	Set the specified Signal Flags of an active thread
os_signal_clear_arduino	Clear the specified Signal Flags of an active thread
os_signal_wait_arduino	Wait for one or more Signal Flags to become signaled for the current RUNNING thread
os_timer_create_arduino	Create a timer
os_timer_start_arduino	Start or restart a timer
os_timer_stop_arduino	Stop the timer
os_timer_delete_arduino	Delete a timer that was created by os_timer_create
os_semaphore_create_arduino	Create and Initialize a Semaphore object used for managing resources
os_semaphore_wait_arduino	Wait until a Semaphore token becomes available
os_semaphore_release_arduino	Release a Semaphore token
os_semaphore_delete_arduino	Delete a Semaphore that was created by os_semaphore_create
os_get_free_heap_size_arduino	Return the available heap memory space when called

os_thread_create_arduino

Description

Create a thread and add it to Active Threads and set it to state READY.

Syntax

uint32_t os_thread_create_arduino (void (*task)(const void *argument), void *argument, int priority, uint32_t stack_size);

Parameters

task: task Function pointer which is the thread body. It should not run into the end of function unless os_thread_terminate is invoked

argument: the data pointer which brings to task

priority: The underlying os is FreeRTOS. It executes tasks with highest priority which are not in idle state.

stack_size: The stack_size is used as memory heap only for this task.

Returns

The thread id which is used in thread operation and signalling.

Example Code

NA

Notes and Warnings

NA

os_thread_get_id_arduino

Description
Return the thread ID of the current running thread

Syntax
uint32_t os_thread_get_id_arduino (void);

Parameters
The function requires no input parameter.

Returns
Current thread id which calls os_thread_get_id.

Example Code
NA

Notes and Warnings
NA

os_thread_terminate_arduino

Description
Terminate execution of a thread and remove it from Active Threads

Syntax
uint32_t os_thread_terminate_arduino (uint32_t thread_id);

Parameters
thread_id: Terminate the thread with specific thread_id

Returns
os_status code

Example Code
NA

Notes and Warnings
Thread should not ended without terminate first.

os_thread_yield_arduino

Description
Pass control to next thread that is in state READY

Syntax
uint32_t os_thread_yield_arduino (void);

Parameters
The function requires no input parameter.

Returns
os_status code

Example Code
NA

Notes and Warnings
By default, the minimal execution unit is 1 millisecond. In a scenario that if a thread with smaller want to handout execution right to a thread with higher priority immediately without waiting for the ending of current 1 millisecond, then invoke os_thread_yield can transfer exection right to OS's idle task and check which is the next execution thread.

os_thread_set_priority_arduino

Description
Change priority of an active thread

Syntax
uint32_t os_thread_set_priority_arduino (uint32_t thread_id, int priority);

Parameters
thread_id: The target thread with the thread id to be changed
priority: The updated priority

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

os_thread_get_priority_arduino

Description
Get current priority of an active thread

Syntax
uint32_t os_thread_get_priority_arduino (uint32_t thread_id);

Parameters
thread_id: The target thread with the thread id to be searched

Returns
os_priority

Example Code
NA

Notes and Warnings
NA

os_signal_set_arduino

Description
Set the specified Signal Flags of an active thread

Syntax
int32_t os_signal_set_arduino (uint32_t thread_id, int32_t signals);

Parameters

thread_id: Send signal to a thread with the thread id
signals: the signals to be send

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_clear_arduino

Description
Clear the specified Signal Flags of an active thread

Syntax
int32_t os_signal_clear_arduino (uint32_t thread_id, int32_t signals);

Parameters
thread_id: Clear signal to a thread with the thread id
signals: The signals to be clear

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_wait_arduino

Description
Wait for one or more Signal Flags to become signaled for the current RUNNING thread

Syntax
os_event_t os_signal_wait_arduino (int32_t signals, uint32_t millisec);

Parameters
signals: the signals to be wait
millisec: the timeout value if no signal comes in. Fill in 0xFFFFFFFF for infinite wait

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_create_arduino

Description
Create a timer

Syntax
uint32_t os_timer_create_arduino (void (*callback)(void const *argument), uint8_t isPeriodic, void *argument);

Parameters
callback: The function to be invoke when timer timeout
isPeriodic: OS_TIMER_ONCE or OS_TIMER_PERIODIC
argument: The argument that is bring into callback function

Returns
timer id

Example Code
NA

Notes and Warnings
NA

os_timer_start_arduino

Description
Start or restart a timer

Syntax
uint32_t os_timer_start_arduino (uint32_t timer_id, uint32_t millisec);

Parameters
timer_id: The timer id obtained from by os_timer_create
millisec: The delays after timer starts

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_stop_arduino

Description
Stop the timer

Syntax
uint32_t os_timer_stop_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_delete_arduino

Description
Delete a timer that was created by os_timer_create

Syntax
uint32_t os_timer_delete_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_create_arduino

Description
Create and Initialize a Semaphore object used for managing resources

Syntax
uint32_t os_semaphore_create_arduino (int32_t count);

Parameters
count: The number of available resources

Returns
semaphore ID

Example Code
NA

Notes and Warnings
NA

os_semaphore_wait_arduino

Description
Wait until a Semaphore token becomes available

Syntax
int32_t os_semaphore_wait_arduino (uint32_t semaphore_id, uint32_t millisec);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create
millisec: timeout value

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_release_arduino

Description
Release a Semaphore token

Syntax
uint32_t os_semaphore_release_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_delete_arduino

Description
Delete a Semaphore that was created by os_semaphore_create

Syntax
uint32_t os_semaphore_delete_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_get_free_heap_size_arduino

Description
Return the available heap memory space when called

Syntax
size_t os_get_free_heap_size_arduino(void);

Parameters
The function requires no input parameter.

Returns
current free heap size

Example Code
Example: MemInfo

Notes and Warnings
NA

WDT

Class WDT

WDT Class

Description
A class used for initializing, starting, stopping watchdog timer.

Syntax
class WDT

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named WDT.	

Public Methods	
WDT:: InitWatchdog	Initializes the watchdog, include time setting, and mode register
WDT:: StartWatchdog	Start the watchdog counting
WDT:: StopWatchdog	Stop the watchdog counting
WDT:: RefreshWatchdog	Refresh the watchdog counting to prevent WDT timeout
WDT:: InitWatchdogIRQ	Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler

WDT:: InitWatchdog

Description
Initializes the watchdog, include time setting, and mode register.

Syntax
void InitWatchdog(uint32_t timeout_ms);

Parameters
timeout_ms: the watch-dog timer timeout value in millisecond (ms). The default action after watchdog timer timeout is to reset the whole system.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer

```
/*
* This example describes how to use watchdog api.
* In this example, watchdog is setup to 5s timeout.
* Watchdog won't bark if we refresh it before timeout in smallTask.
* The timer is also reloaded after refresh.
* Otherwise, while running bigTask, watchdog will restart system in default or call callback function if registered.
*/
#include "wdt.h"
#define RUN_CALLBACK_IF_WATCHDOG_BARKS (0)
WDT wdt;
void setup() {
Serial.begin(115200);
wdt.InitWatchdog(5000); // setup 5s watchdog
#if RUN_CALLBACK_IF_WATCHDOG_BARKS
wdt.InitWatchdogIRQ(my_watchdog_irq_handler, 0);
#else
// system would restart in default when watchdog barks
#endif
wdt.StartWatchdog(); // enable watchdog timer
successfulTask();
failedTask();
while (1)
```

```
;
}
void loop() {
}
void successfulTask(void) {
Serial.println("……doing small task……");
for (int i = 0; i < 50000000; i++) // dummy task
asm(" nop");
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
/*
* Doing this task will lead to failed refresh the
* watchdog timer within the time limits of 5 seconds
*/
void failedTask(void) {
Serial.println("……doing big task……");
for (int i = 0; i < 10; i++) {
Serial.print("doing dummy task #");
Serial.println(i, DEC);
for (int j = 0; j < 50000000; j++) // dummy task
asm(" nop");
}
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
void my_watchdog_irq_handler(uint32_t id) {
printf("watchdog barks!!!rn");
WDG_Cmd(DISABLE);
}
```

Notes and Warnings
NA

WDT:: StartWatchdog

Description
Start the watchdog counting.

Syntax
void StartWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: StopWatchdog

Description
Stop the watchdog counting.

Syntax
void StopWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: RefreshWatchdog

Description
Refresh the watchdog counting to prevent WDT timeout.

Syntax
void RefreshWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: InitWatchdogIRQ

Description
Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler. The interrupt handler will be called when the watchdog timer is timeout.

Syntax
void WDT::InitWatchdogIRQ(wdt_irq_handler handler, uint32_t id)

Parameters
handler: the callback function for WDT timeout interrupt.
id: the parameter for the callback function

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WiFi

Class WiFi

WiFiClass Class

Description
Defines a class of WiFi and network implementation for Ameba.

Syntax
class WiFiClass

Members

Public Constructors	
WiFiClass::WiFiClass	Constructs a WiFiClass object and initializes the WiFi libraries and network settings
Public Methods	
WiFi-Class::firmwareVersion	Get firmware version
WiFiClass:: begin	Start Wifi connection for OPEN networks
WiFiClass:: config	Configure network IP settings
WiFiClass:: setDNS	Set the DNS server IP address to use
WiFiClass:: disconnect	Disconnect from the network
WiFiClass:: macAddress	Get the interface MAC address
WiFiClass:: localIP	Get the interface IP address
WiFiClass:: subnetMask	Get the interface subnet mask address
WiFiClass:: gatewayIP	Get the gateway IP address
WiFiClass:: SSID	Return the current SSID associated with the network
WiFiClass:: BSSID	Return the current BSSID associated with the network
WiFiClass:: RSSI	Return the current RSSI (Received Signal Strength in dBm) associated with the network
WiFiClass:: encryption-Type	Return the Encryption Type associated with the network
WiFiClass:: scanNetworks	Start scan WiFi networks available
WiFiClass:: SSID	Return the SSID discovered during the network scan
WiFiClass:: encryption-Type	Return the encryption type of the networks discovered during the scanNetworks
WiFiClass:: encryption-TypeEx	Return the security type and encryption type of the networks discovered during the scanNetworks
WiFiClass:: RSSI	Return the RSSI of the networks discovered during the scanNetworks
WiFiClass:: status	Return Connection status
WiFiClass:: hostByName	Resolve the given hostname to an IP address
WiFiClass:: apbegin	Start AP mode
WiFiClass:: disablePower-Save	Disable power-saving mode

WiFiClass::WiFiClass

Description

Constructs a WiFiClass object and initializes the WiFi libraries and network settings.

Syntax

WiFiClass::WiFiClass()

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings
An instance of WiFiClass is created as WiFi inside WiFi.h and is extern for direct use.

WiFiClass::firmwareVersion

Description
Get firmware version

Syntax
char* WiFiClass::firmwareVersion()

Parameters
The function requires no input parameter.

Returns
WiFi firmware version

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFI network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>
// char ssid[] = "yourNetwork"; // your network SSID (name)
// char pass[] = "secretPassword"; // your network password
char ssid[] = "SINGTEL-D45F"; // your network SSID (name)
char pass[] = "mooxuteeth"; // your network key
int status = WL_IDLE_STATUS; // the Wifi radio's status
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
```

```
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to WPA SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
```

```
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
```

```
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::begin

Description
Start Wifi connection for OPEN networks

Syntax
int WiFiClass::begin(char* ssid)
int WiFiClass::begin(char* ssid, uint8_t key_idx, const char *key)
int WiFiClass::begin(char* ssid, const char *passphrase)

Parameters
ssid: Pointer to the SSID string
key_idx: The key index to set. Valid values are 0-3.
key: Key input buffer.
passphrase: Passphrase. Valid characters in a passphrase must be between ASCII 32-126 (decimal).

Returns
WiFi status

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::config

Description
Configure network settings for the WiFi network

Syntax
void WiFiClass::config(IPAddress local_ip)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway, IPAddress subnet)

Parameters
local_ip: Local device IP address to use on the network
dns_server: IP address of the DNS server to use
gateway: IP address of the gateway device on the network
subnet: Subnet mask for the network, expressed as a IP address

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This will disable the DHCP client when connecting to a network, and will require the network accepts a static IP. The configured IP addresses will also apply to AP mode, but the DHCP server will not be disabled in AP mode.

WiFiClass::setDNS

Description
Configure the IP address of the DNS server to use

Syntax
void WiFiClass::setDNS(IPAddress dns_server1)
void WiFiClass::setDNS(IPAddress dns_server1, IPAddress dns_server2)

Parameters
dns_server1: IP address of DNS server to use
dns_server2: IP address of DNS server to use

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClass::disconnect

Description
Disconnect from the network

Syntax
int WiFiClass::disconnect()

Parameters
The function requires no input parameter.

Returns
The function returns one value of wl_status_t enum as an integer.

Example Code
NA

Notes and Warnings
NA

WiFiClass::macAddress

Description
Get the interface MAC address

Syntax
uint8_t* WiFiClass::macAddress(uint8_t* mac)

Parameters
mac: an array to store MAC address

Returns
The function returns a pointer to uint8_t array with length WL_MAC_ADDR_LENGTH.

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::localIP

Description

Get the interface IP address

Syntax

IPAddress WiFiClass::localIP()

Parameters

The function requires no input parameter.

Returns

Ip address value

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::subnetMask

Description

Get the interface subnet mask address

Syntax

IPAddress WiFiClass::subnetMask()

Parameters
The function requires no input parameter.

Returns
subnet mask address value

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>

char ssid[] = "SINGTEL-D45F_5G"; // the name of your network

int status = WL_IDLE_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while (true);

}

String fv = WiFi.firmwareVersion();

if (fv != "1.1.0") {

Serial.println("Please upgrade the firmware");

}

// attempt to connect to Wifi network:

while (status != WL_CONNECTED) {

Serial.print("Attempting to connect to open SSID: ");

Serial.println(ssid);

status = WiFi.begin(ssid);

// wait 10 seconds for connection:

delay(10000);
```

```
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
```

```
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
Serial.print("Gateway: ");
Serial.println(gateway);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
}
```

Notes and Warnings

NA

WiFiClass::gatewayIP

Description
Get the gateway IP address

Syntax
IPAddress WiFiClass::gatewayIP()

Parameters
The function requires no input parameter.

Returns
The function returns the value of the gateway IP address.

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details. Details of the code can be found in the section of WiFiClass:: subnetMask.

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the current SSID associated with the network

Syntax
char* WiFiClass::SSID()

Parameters
The function requires no input parameter.

Returns
The function returns current SSID associate with the network.

Example Code
Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::BSSID

Description
Return the current BSSID associated with the network

Syntax
uint8_t* WiFiClass::BSSID(uint8_t* bssid)

Parameters
bssid: an array to store bssid

Returns
pointer to uint8_t array with length WL_MAC_ADDR_LENGTH

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the current RSSI (Received Signal Strength in dBm) associated with the network

Syntax
int32_t WiFiClass::RSSI()

Parameters
The function requires no input parameter.

Returns
The function returns a signed-value signal strength

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the Encryption Type associated with the network

Syntax
uint8_t WiFiClass::encryptionType()

Parameters
The function requires no input parameter.

Returns
The function returns one unsigned integer value of wl_enc_type enum.

Example Code
Example: ConnectWithWPA

Notes and Warnings
NA

WiFiClass::scanNetworks

Description
Start scan WiFi networks available

Syntax
int8_t WiFiClass::scanNetworks()

Parameters
The function requires no input parameter.

Returns
The function returns the number of discovered networks as an integer.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified.

```
#include <WiFi.h>
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// Print WiFi MAC address:
printMacAddress();
}
void loop() {
// scan for existing networks:
Serial.println("Scanning available networks…");
listNetworks();
delay(10000);
}
```

```
void printMacAddress() {
// the MAC address of your Wifi shield
byte mac[6];
// print your MAC address:
WiFi.macAddress(mac);
Serial.print("MAC: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void listNetworks() {
// scan for nearby networks:
Serial.println("* Scan Networks *");
int numSsid = WiFi.scanNetworks();
if (numSsid == -1) {
Serial.println("Couldn't get a wifi connection");
while (true);
}
// print the list of networks seen:
Serial.print("number of available networks:");
Serial.println(numSsid);
// print the network number and name for each network found:
for (int thisNet = 0; thisNet < numSsid; thisNet++) {
Serial.print(thisNet);
Serial.print(") ");
Serial.print(WiFi.SSID(thisNet));
Serial.print("tSignal: ");
Serial.print(WiFi.RSSI(thisNet));
```

```
Serial.print(" dBm");
Serial.print("tEncryptionRaw: ");
printEncryptionTypeEx(WiFi.encryptionTypeEx(thisNet));
Serial.print("tEncryption: ");
printEncryptionType(WiFi.encryptionType(thisNet));
}
}
void printEncryptionTypeEx(uint32_t thisType) {
/* Arduino wifi api use encryption type to mapping to security type.
* This function demonstrate how to get more richful information of security type.
*/
switch (thisType) {
case SECURITY_OPEN:
Serial.print("Open");
break;
case SECURITY_WEP_PSK:
Serial.print("WEP");
break;
case SECURITY_WPA_TKIP_PSK:
Serial.print("WPA TKIP");
break;
case SECURITY_WPA_AES_PSK:
Serial.print("WPA AES");
break;
case SECURITY_WPA2_AES_PSK:
Serial.print("WPA2 AES");
break;
case SECURITY_WPA2_TKIP_PSK:
Serial.print("WPA2 TKIP");
break;
case SECURITY_WPA2_MIXED_PSK:
Serial.print("WPA2 Mixed");
break;
case SECURITY_WPA_WPA2_MIXED:
Serial.print("WPA/WPA2 AES");
break;
```

```
}
}
void printEncryptionType(int thisType) {
// read the encryption type and print out the name:
switch (thisType) {
case ENC_TYPE_WEP:
Serial.println("WEP");
break;
case ENC_TYPE_TKIP:
Serial.println("WPA");
break;
case ENC_TYPE_CCMP:
Serial.println("WPA2");
break;
case ENC_TYPE_NONE:
Serial.println("None");
break;
case ENC_TYPE_AUTO:
Serial.println("Auto");
break;
}
}
```

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the SSID discovered during the network scan

Syntax
char* WiFiClass::SSID(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
The function returns ssid string of the specified item on the networks scanned a list.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the encryption type of the networks discovered during the scanNetworks

Syntax
uint8_t WiFiClass::encryptionType(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
encryption type (enum wl_enc_type) of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionTypeEx

Description
Return the security type and encryption type of the networks discovered during the scanNetworks

Syntax
uint32_t WiFiClass::encryptionTypeEx(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
security and encryption type of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the RSSI of the networks discovered during the scanNetworks

Syntax
int32_t WiFiClass::RSSI(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
signed value of RSSI of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::status

Description
Return Connection status

Syntax
uint8_t WiFiClass::status()

Parameters
The function requires no input parameter.

Returns
The function returns one of the values defined in wl_status_t as an unsigned integer.

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::hostByName

Description
Resolve the given hostname to an IP address

Syntax
int WiFiClass::hostByName(const char* aHostname, IPAddress& aResult)

Parameters
aHostname: Name to be resolved
aResult: IPAddress structure to store the returned IP address

Returns
The function returns “1” if aIPAddrString was successfully converted to an IP address,else otherwise, it will return as an error code.

Example Code

NA

Notes and Warnings
NA

WiFiClass::apbegin

Description
Start AP mode

Syntax
int WiFiClass::apbegin(char* ssid, char* channel)
int WiFiClass::apbegin(char* ssid, char* password, char* channel)

Parameters
ssid: SSID of the AP network
channel: AP's channel, default 1
password: AP's password

Returns
The function will return the WiFi status.

Example Code
Example: WiFiAPMode

#include

char ssid[] = "yourNetwork"; //Set the AP's SSID

char pass[] = "Password"; //Set the AP's password

char channel[] = "1"; //Set the AP's channel

int status = WL_IDLE_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {

```
Serial.println("WiFi shield not present");
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to start AP:
while (status != WL_CONNECTED) {
Serial.print("Attempting to start AP with SSID: ");
Serial.println(ssid);
status = WiFi.apbegin(ssid, pass, channel);
delay(10000);
}
//AP MODE already started:
Serial.println("AP mode already started");
Serial.println();
printWifiData();
printCurrentNet();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
```

```
Serial.print("Gateway: ");
Serial.println(gateway);
Serial.println();
}
void printCurrentNet() {
// print the SSID of the AP:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of AP:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[0], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.println(bssid[5], HEX);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::disablePowerSave

Description
Disable power-saving mode

Syntax
int WiFiClass::disablePowerSave()

Parameters
The function requires no input parameter.

Returns
1 if disable success, 0 if failed

Example Code
NA

Notes and Warnings
NA

Class WiFiClient

WiFiClient Class

Description
Defines a class of WiFi Client implementation for Ameba.

Syntax
class WiFiClient

Members

Public Constructors	
WiFiClient::WiFiClient	Constructs a WiFiClient instance that connects to the specified IP address and port.
Public Methods	
WiFiClient::connect	Connect to the IP address and port
WiFiClient::write	Write a single byte into the packet
WiFiClient::available	Number of bytes remaining in the current packet
WiFiClient::read	Read a single byte from the current packet
WiFiClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiClient:: flush	Finish reading the current packet
WiFiClient::stop	Stop client connection
WiFiClient::connected	Check if client is connected, return 1 if connected, 0 if not
WiFiClient::setRecvTimeout	Set receiving timeout

WiFiClient::WiFiClient

Description

Constructs a WiFiClient instance that connects to the specified IP address and port.

Syntax
WiFiClient::WiFiClient()
WiFiClient::WiFiClient(uint8_t sock)

Parameters
sock: socket state, default -1.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient

```
#include <WiFi.h>
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "password"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
//IPAddress server(64,233,189,94); // numeric IP for Google (no DNS)
char server[] = "www.google.com"; // name address for Google (using DNS)
WiFiClient client;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
;
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
```

```
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 80)) {
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=ameba HTTP/1.1");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
```

```
// do nothing forevermore:
while (true);
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiClient::connect

Description
Connect to the IP address and port

Syntax
int WiFiClient::connect(IPAddress ip, uint16_t port)
int WiFiClient::connect(const char *host, uint16_t port)

Parameters
ip: IP address
host: Host name
port: the port to listen on

Returns

Returns “1”: if successful
Returns “0”: if failed

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiClient::write(uint8_t byte)
size_t WiFiClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message
size: the size of the buffer

Returns
The function returns single byte into the packet or returns bytes size from buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiClient::available(void)

Parameters
The function requires no input parameter.

Returns
• Function returns the number of bytes available in the current packet
Function returns 0: if no data available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiClient::read()
int WiFiClient::read(unsigned char* buf, size_t size)
int WiFiClient::read(char *buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiClient::flush

Description
Finish reading the current packet

Syntax
void WiFiClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClient::stop

Description
Disconnect from the server. Stop client connection

Syntax
void WiFiClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::connected

Description
Check if client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
0

Example Code
NA

Notes and Warnings
NA

Class WiFiServer

WiFiServer Class

Description
Defines a class of WiFi server implementation for Ameba.

Syntax
class WiFiServer

Members

Public Constructors	
WiFiServer::WiFiServer	Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port
Public Methods	
WiFiServer::available	Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop()
WiFiServer::begin	Tells the server to begin listening for incoming connections
WiFiServer::write	Write data to all the clients connected to a server

WiFiServer::WiFiServer

Description
Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port.

Syntax
WiFiServer::WiFiServer(uint16_t port)

Parameters
port: The port number being connected to.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi

#include <WiFi.h>

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

WiFiServer server(5000);

void setup() {

Serial.begin(9600); // initialize serial communication

pinMode(9, OUTPUT); // set the LED pin mode

```
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
while (true); // don't continue
}
String fv = WiFi.firmwareVersion();
if ( fv != "1.1.0" )
Serial.println("Please upgrade the firmware");
// attempt to connect to Wifi network:
while ( status != WL_CONNECTED) {
Serial.print("Attempting to connect to Network named: ");
Serial.println(ssid); // print the network name (SSID);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
server.begin(); // start the tcp server on port 5000
printWifiStatus(); // you're connected now, so print out the status
}
char buffer[256];
void loop() {
WiFiClient client = server.available();
while (client.connected()) {
memset(buffer, 0, 256);
int n = client.read((uint8_t*)(&buffer[0]), sizeof(buffer));
if (n > 0) {
for (int i=0; i<n; i++) {
Serial.print(buffer[i]);
}
n = client.write(buffer, n);
if (n <= 0) break;
}
}
client.stop();
}
```

```
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiServer::available

Description
Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop().

Syntax
WiFiClient WiFiServer::available(uint8_t* status)

Parameters
status: WiFi availability status

Returns
A Client object; if no Client has data available for reading, this object will evaluate to false in an if-statement

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::begin

Description
Tells the server to begin listening for incoming connections

Syntax
void WiFiServer::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::write

Description
Write data to all the clients connected to a server

Syntax
size_t WiFiServer::write(uint8_t b)
size_t WiFiServer::write(const uint8_t *buf, size_t size)

Parameters
b: byte to be written
buf: data buffer
size: Size of the data in the buffer

Returns
The function returns the number of bytes written. It is not necessary to read this.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

Class WiFiSSLClient

WiFiSSLClient Class

Description
Defines a class of WiFi Secure Socket Layer Client implementation for Ameba.

Syntax
class WiFiSSLClient

Members

Public Constructors	
WiFiSSLClient::WiFiSSLClient	Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port
Public Methods	
WiFiSSLClient::connect	Connect to the IP address and port
WiFiSSLClient::write	Write a single byte into the packet
WiFiSSLClient::available	Number of bytes remaining in the current packet
WiFiSSLClient::read	Read a single byte from the current packet
WiFiSSLClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiSSLClient:: flush	Finish reading the current packet
WiFiSSLClient::stop	Stop SSL client connection
WiFiSSLClient::connected	Check if SSL client is connected, return 1 if connected, 0 if not
WiFiSSLClient:: setRootCA	Set Root CA for authentication
WiFiSSLClient:: setClientCertificate	Set certificate of the client
WiFiSSLClient::setRecvTimeout	Set receiving timeout
WiFiSSLClient::setPreSharedKey	Set the pre shared key (PSK) to use for authentication

WiFiSSLClient::WiFiSSLClient

Description
Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port.

Syntax
WiFiSSLClient::WiFiSSLClient(void)
WiFiSSLClient::WiFiSSLClient(uint8_t sock)

Parameters
sock: socket state, default -1

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient

#include

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”;// your network password (use for WPA, or WEP)

int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

char server[] = “www.google.com”; // name address for Google (using DNS)

//unsigned char test_client_key[] = “”; //For the usage of verifying client

//unsigned char test_client_cert[] = “”; //For the usage of verifying client

//unsigned char test_ca_cert[] = “”; //For the usage of verifying server

WiFiSSLClient client;

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {

Serial.println(“WiFi shield not present”);

// don’t continue:

while (**true**);

}

// attempt to connect to Wifi network:

while (status != WL_CONNECTED) {

```
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 443)) { //client.connect(server, 443, test_ca_cert, test_client_cert, test_client_key)
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=realtek HTTP/1.0");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
else
Serial.println("connected to server failed");
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
// do nothing forevermore:
while (true);
```

```
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiSSLClient::connect

Description
Connect to the IP address and port.

Syntax
int WiFiSSLClient::connect(IPAddress ip, uint16_t port)
int WiFiSSLClient::connect(const char *host, uint16_t port)
int WiFiSSLClient::connect(const char* host, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)
int WiFiSSLClient::connect(IPAddress ip, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)

Parameters
ip: IP address
host: Host name
port: the port to listen on
rootCABuff: buffer that store root CA
cli_cert: buffer that store client certificate
cli_key buffer that store client key pair

Returns
1: if successful
0: if failed

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiSSLClient::write(uint8_t byte)
size_t WiFiSSLClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message

size: the size of the buffer

Returns
The function returns single -byte into the packet or turns bytes size from the buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiSSLClient::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available in the current packet; else return “0:” if no data available.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiSSLClient::read()

int WiFiSSLClient::read(unsigned char* buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::peek

Description
Return the next byte from the current packet without moving on to the next byte.

Syntax
int WiFiSSLClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::flush

Description
Finish reading the current packet

Syntax
void WiFiSSLClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::stop

Description
Disconnect from the server. Stop SSL client connection

Syntax
void WiFiSSLClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::connected

Description
Check if SSL client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiSSLClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::setRootCA

Description
Set Root CA for authentication

Syntax
void WiFiSSLClient::setRootCA(unsigned char *rootCA)

Parameters
rootCA: a string of rootCA

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

WiFiSSLClient::setClientCertificate

Description
Set certificate of client

Syntax
void WiFiSSLClient::setClientCertificate(unsigned char *client_ca, unsigned char *private_key)

Parameters
client_ca: Client certificate
private_key: client's private key pair

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiSSLClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
The function returns "0".

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setPreSharedKey

Description
Set the pre shared key (PSK) to use for authentication

Syntax
void WiFiSSLClient::setPreSharedKey(unsigned char *pskIdent, unsigned char *psKey)

Parameters
pskIdent: identity for PSK
psKey: Pre shared key

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Do not set a root CA and client certificate if PSK should be used for authentication. If root CA, client certificate and PSK are all set, certificate based authentication will be used.

Class WiFiUdp

WiFiUDP Class

Description
Defines a class of WiFi UDP implementation for Ameba.

Syntax
class WiFiUDP

Members

Public Constructors	
WiFiUDP::WiFiUDP	Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages
Public Methods	
WiFiUDP:: begin	initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use
WiFiUDP:: stop	Finish with the UDP socket
WiFiUDP:: beginPacket	Start building up a packet to send to the remote host-specific in IP and port
WiFiUDP:: endPacket	Finish off this packet and send it
WiFiUDP:: write	Write a single byte into the packet
WiFiUDP:: writeImmediately	Send packet immediately from the buffer
WiFiUDP:: parsePacket	Start processing the next available incoming packet
WiFiUDP::available	Number of bytes remaining in the current packet
WiFiUDP::read	Read a single byte from the current packet
WiFiUDP:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiUDP:: flush	Finish reading the current packet
WiFiUDP:: remoteIP	Return the IP address of the host who sent the current incoming packet
WiFiUDP:: remotePort	Return the port of the host who sent the current incoming packet
WiFiUDP:: setRecvTimeout	Set receiving timeout

WiFiUDP::WiFiUDP

Description
Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages.

Syntax
WiFiUDP::WiFiUDP(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort.

#include <WiFi.h>

#include <WiFiUdp.h>

```
int status = WL_IDLE_STATUS;
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
unsigned int localPort = 2390; // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back
WiFiUDP Udp;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
```

```
// if you get a connection, report back via serial:
Udp.begin(localPort);
}
void loop() {
// if there's data available, read a packet
int packetSize = Udp.parsePacket();
if (packetSize) {
Serial.print("Received packet of size ");
Serial.println(packetSize);
Serial.print("From ");
IPAddress remoteIp = Udp.remoteIP();
Serial.print(remoteIp);
Serial.print(", port ");
Serial.println(Udp.remotePort());
// read the packet into packetBufffer
int len = Udp.read(packetBuffer, 255);
if (len > 0) {
packetBuffer[len] = 0;
}
Serial.println("Contents:");
Serial.println(packetBuffer);
// send a reply, to the IP address and port that sent us the packet we received
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
Udp.write(ReplyBuffer);
Udp.endPacket();
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
```

```
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
This constructor does not take in any parameter, thus use another method to set up the IP address and port number.

WiFiUDP::begin

Description
Initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use.

Syntax
uint8_t WiFiUDP::begin(uint16_t port)

Parameters
port: the local port to listen on

Returns
1: if successful
0: if there are no sockets available to use

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::stop

Description
Disconnect from the server. Release any resource being used during the UDP session.

Syntax
void WiFiUDP::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::beginPacket

Description
Start building up a packet to send to the remote host-specific in IP and port.

Syntax
int WiFiUDP::beginPacket(const char *host, uint16_t port)
int WiFiUDP::beginPacket(IPAddress ip, uint16_t port)

Parameters
host: hostname
port: port number
ip: IP address

Returns
1: if successful
0: if there was a problem with the supplied IP address or port

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::endPacket

Description
Finish off this packet and send it

Syntax
int WiFiUDP::endPacket(void)

Parameters
The function requires no input parameter.

Returns
1: if the packet was sent successfully
0: if there was an error

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::write

Description
Write a single byte into the packet.

Syntax
size_t WiFiUDP::write(uint8_t byte)
size_t WiFiUDP::write(const uint8_t *buffer, size_t size)

Parameters
byte: the outgoing byte

buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::writeImmediately

Description
Send packet immediately from the buffer

Syntax
size_t WiFiUDP::writeImmediately(const uint8_t *buffer, size_t size)

Parameters
buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::parsePacket

Description
Start processing the next available incoming packet

Syntax
int WiFiUDP::parsePacket(void)

Parameters
The function requires no input parameter.

Returns
The function returns the size of the packet in bytes or returns "0:" if no packets are available.

Example Code
Example: WiFiUdpSendReceiveString

Notes and Warnings
NA

WiFiUDP::available

Description
Number of bytes remaining in the current packet.

Syntax
int WiFiUDP::available(void)

Parameters
The function requires no input parameter.

Returns
the number of bytes available in the current packet
0: if parsePacket hasn't been called yet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::read

Description
Read a single byte from the current packet

Syntax
int WiFiUDP::read()
int WiFiUDP::read(unsigned char* buffer, size_t len)

Parameters
buffer: buffer to hold incoming packets (char*)
len: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiUdpSendReceiveString
his example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiUDP::peek(void)

Parameters

The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiUDP::flush

Description
Finish reading the current packet

Syntax
void WiFiUDP::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::remoteIP

Description
Return the IP address of the host who sent the current incoming packet

Syntax
IPAddress WiFiUDP::remoteIP(void)

Parameters
The function requires no input parameter.

Returns
IP address connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::remotePort

Description
Return the port of the host who sent the current incoming packet

Syntax
uint16_t WiFiUDP::remotePort(void)

Parameters
The function requires no input parameter.

Returns
The remote port connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::setRecvTimeout

Description
Set receiving timeout

Syntax
void WiFiUDP::setRecvTimeout(int timeout)

Parameters
timeout in seconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba WiFi related APIs and examples are works based on Arduino WiFI shield libraries (https://www.arduino.cc/en/Reference/WiFi).
These include,

- `WiFi.cpp`
- `WiFi.h`
- `WiFiServer.cpp`
- `WiFiServer.h`
- `WiFiUdp.cpp`
- `WiFiUdp.h`

Wire

Class TwoWire

TwoWire Class

Description
Defines a class of I2C API

Syntax
class TwoWire

Members

Public Constructors	
TwoWire::TwoWire	Constructs a TwoWire object
Public Methods	
TwoWire::begin	Initialize I2C master/slave
TwoWire::setClock	Set I2C frequency
TwoWire::beginTransmission	Begin I2C transmission
TwoWire::endTransmission	End I2C transmission
TwoWire::requestFrom	Set I2C requestFrom
TwoWire::write	Write data to I2C
TwoWire::available	Check if I2C is available
TwoWire::read	Read data from I2C
TwoWire::peek	Read peek from I2C
TwoWire::flush	Do nothing, use, and transmission(..) to force data transfer
TwoWire::onReceive	Callback function when I2C on receive
TwoWire::onRequest	Callback function when I2C on request

TwoWire::TwoWire

Description
Constructs a TwoWire object.

Syntax
TwoWire::TwoWire (uint32_t dwSDAPin, uint32_t dwSCLPin);

Parameters
dwSDAPin: The Arduino PIN to be set as an SDA pin.
dwSCLPin: The Arduino PIN to be set as an SCL pin.

Returns
The function returns nothing.

Example Code

Example: MasterWriter

This example demonstrates the use of the wire library writes to an I2C/TWI slave device.

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
}
byte x = 0;
void loop() {
Wire.beginTransmission(8); // transmit to device #8
Wire.write("x is "); // sends five bytes
Wire.write(x); // sends one byte
Wire.endTransmission(); // stop transmitting
x++;
delay(500);
}
```

Example: MasterReader

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
Serial.begin(9600); // start serial for output
}
void loop() {
Wire.requestFrom(8, 6); // request 6 bytes from slave device #8
while (Wire.available()) { // slave may send less than requested
char c = Wire.read(); // receive a byte as character
Serial.print(c); // print the character
}
delay(500);
}
```

This example demonstrates the use of the wire library reads data from an I2C/TWI slave device.

Notes and Warnings

Include "Wire.h" to use the class function.

TwoWire::begin

Description
Initialize I2C master/slave.

Syntax
void TwoWire::begin (void);
void TwoWire::begin (uint8_t address = 0);
void TwoWire::begin (int address);

Parameters
void: Set the I2C master mode.
address: Set the I2C master mode with slave address value.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::setClock

Description
Set I2C frequency.

Syntax
void TwoWire::setClock(uint32_t frequency);

Parameters
frequency: The frequency values.

Returns
The function returns nothing.

Example Code

Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::beginTransmission

Description
Begin I2C transmission.

Syntax
void TwoWire::beginTransmission (uint8_t address);
void TwoWire::beginTransmission (int address);

Parameters
address: The transmission address.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::endTransmission

Description
End I2C transmission. Originally, ‘endTransmission’ was an f(void) function. It has been modified to take one parameter indicating whether or not a STOP should be performed on the bus. Calling endTransmission(false) allows a sketch to perform a repeated start.

WARNING: Nothing in the library keeps track of whether the bus tenure has been properly ended with a STOP. It is very possible to leave the bus in a hung state if no call to endTransmission(true) is made. Some I2C devices will behave oddly if they do not see a STOP.

If the input parameter is void, this provides backward compatibility with the original definition, and expected behavior, of endTransmission.

Syntax
uint8_t TwoWire::endTransmission (uint8_t sendStop);
uint8_t TwoWire::endTransmission (void);

Parameters
sendStop: True to end the transmission

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::requestFrom

Description
Set I2C requestFrom.

Syntax
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity, uint8_t sendStop);
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity);
uint8_t TwoWire::requestFrom(int address, int quantity);
uint8_t TwoWire::requestFrom (int address, int quantity, int sendStop);

Parameters
address: I2C read address.
quantity: I2C read quantity.
sendStop: True to end the transmission.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::write

Description
Write data to I2C.

Syntax
size_t TwoWire::write (uint8_t data);
size_t TwoWire::write (const uint8_t *data, size_t quantity);

Parameters
data: The data to be transmitted.
quantity: The quantity of data.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::available

Description
Check if I2C is available.

Syntax
int TwoWire::available (void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::read

Description
Read data from I2C

Syntax
int TwoWire::read (void);

Parameters
The function requires no input parameter.

Returns
The read data from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::peek

Description
Read peek from I2C.

Syntax
int TwoWire::peek (void);

Parameters
The function requires no input parameter.

Returns
The peek data read from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::flush

Description
Do nothing, use endTransmission(..) to force data transfer.

Syntax
void TwoWire::flush (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter

Notes and Warnings
Include “Wire.h” in order to use the class function.

TwoWire::onReceive

Description
Callback function when I2C on receive.

Syntax
void TwoWire::onReceive (void(*function)(int));

Parameters
function: The callback function.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::onRequest

Description
Callback function when I2C on request.

Syntax
void TwoWire::onRequest (void(*function)(void));

Parameters
function: The callback function

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

Wire_Readme

The Ameba LCD related api and example are works based on "New LiquidCrystal library" (https://bitbucket.org/fmalpartida/new-liquidcrystal/).

These include,

LCD.h
LCD.cpp
I2CIO.h
I2CIO.cpp
LiquidCrystal_I2C.h
LiquidCrystal_I2C.cpp
examples/LcdHelloWorld/LcdHelloWorld.ino

These files inherit the licence of "New LiquidCrystal Library" which are under a Creative Commons Attribution-ShareAlike 3.0 Unported License. CC BY-SA 3.0.

1.2.6 Resources

Links

- AmebaD Arduino Github
- Arduino Website

1.2.7 Support

FAQ

Where to buy Ameba RTL8722DM Board?

Refer to Purchase link.

Which Bluetooth standards are supported by RTL8722CSM/RTL8722DM?

Both boards support BLE 5.0. Classic Bluetooth (BR/EDR) is not supported.

Which BLE roles are supported?

RTL8722CSM/RTL8722DM can operate as either a BLE Central or BLE Peripheral device.

Are all pins on RTL8722CSM/RTL8722DM usable?

No, those marked `NC` are not connected to any pin and thus unusable.

Is XIP (execute in place) supported on RTL8722CSM/RTL8722DM?

Yes, it is supported.

Does RTL8722CSM support 5G WiFi?

No. Only RTL8722DM supports dual band 2.4G + 5G WiFi. RTL8722CSM only supports single band 2.4G WiFi.

How to enter the download mode?

Press and hold the UART DOWNLOAD button. Then Press the RESET button and release both UART DOWNLOAD and RESET buttons.

Trouble shooting

RTL8722CSM/RTL8722DM cannot be found as a Bluetooth device.

Please make sure the antenna is connected properly. Check your code for the correct Bluetooth configurations.

My code is not behaving as I expected.

Try to debug your program using `printf()` and `Serial.print()` statements. If the issue persists, you can ask for help at Forums

Why is there no output on my serial terminal after connecting to RTL8722CSM/RTL8722DM UART?

RTL8722CSM/RTL8722DM is by default configured at 115200 baudrate, please check if your serial terminal is configured to 115200.

My program is not being downloaded into RTL8722CSM/RTL8722DM?

Please follow the procedure for the correct downloading?

1. Enter the download mode. The on-board Green LED will blink when entered download mode.
2. When downloading the image into board the on-board Red LED will blink
3. After a successful download, you will see log like this "All images sent successfully".

Sometimes WiFi signal is weak?

The default antenna for RTL8722CSM/RTL8722DM uses the I-Pex Connector. Please change/connect the I-Pex Connector antenna.

Why is my board not powering up?

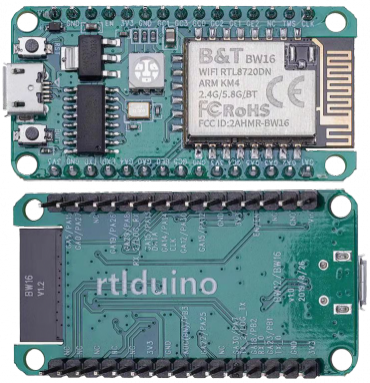
Please make sure the connector J38 beside resistor R43 is connected. The connector is used to link the power to IC.

If you have driver issue to connect board to your computer?

Please go to https://ftdichip.com/drivers/ for USB driver.

1.3 BW16 (RTL8720DN) by Ai-Thinker

Welcome to BW16 (RTL8720DN) online documentation.



1.3.1 Getting Started

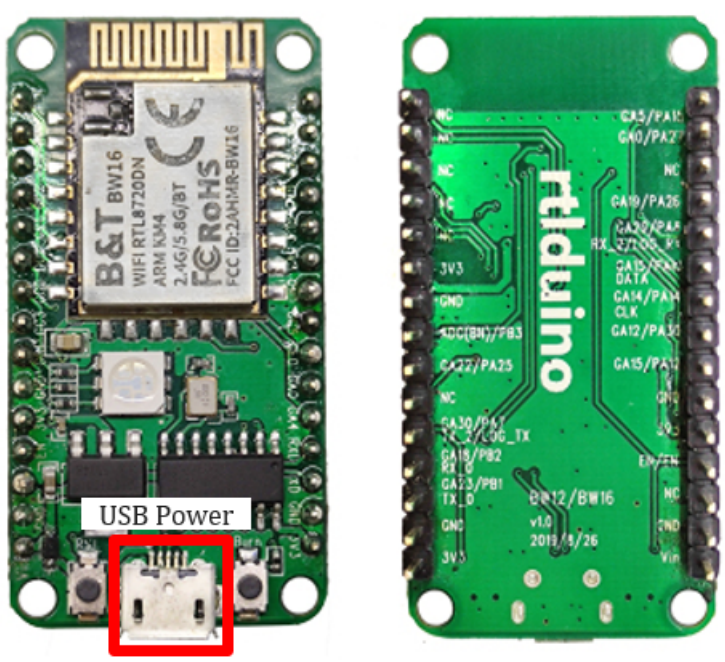
Ameba ARDUINO: Getting Started with BW16

Required Environment

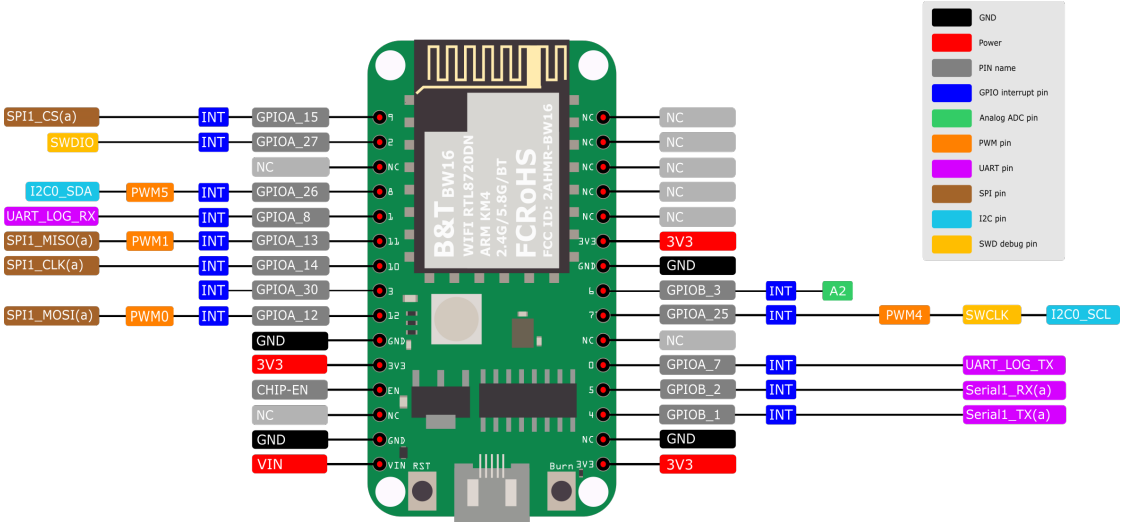
BW16 Dual-Band Wi-Fi board currently supports Windows XP/7/8/10 32-bits and 64-bits operating systems. In this documentation, please use Arduino IDE with version 1.8.15 or later.

Introduction to BW16

Realtek RTL8720DN is a Wi-Fi and Bluetooth IC that supports 2.4GHz and 5GHz dual bands for Wi-Fi communication, and Bluetooth Low Energy (BLE) 5.0. BW16 is a module manufactured by B&T, this module is a highly integrated Wi-Fi and Bluetooth module with the RTL8720DN as the main SoC (System on Chip), it can be regarded as an SoC for the Wi-Fi and Bluetooth application with typical SBCs.



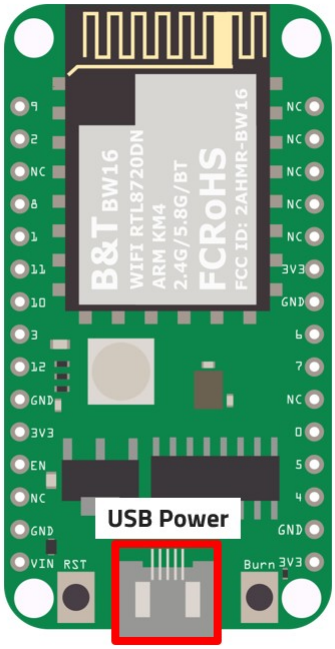
BW16 has a smaller size than AMB21 and AMB23 as shown in the above figure. It uses Micro USB to supply power, which is common in many smart devices. Please refer to the following figure and table for the pin diagram and function of BW16.



#	PIN name	GPIO	ADC	PWM	UART	SPI	I2C
D0(PA7)	GPIOA_7	✓			UART_LOG_TX		
D1(PA8)	GPIOA_8	✓			UART_LOG_RX		
D2(PA27)	GPIOA_27	✓					
D3(PA30)	GPIOA_30	✓					
D4(PB1)	GPIOB_1	✓			Serial_TX		
D5(PB2)	GPIOB_2	✓			Serial_RX		
D6(PB3)	GPIOB_3	✓	A2				
D7(PA25)	GPIOA_25				✓		I2C0_CLK
D8(PA26)	GPIOA_26	✓		✓			I2C0_SDA
D9(PA15)	GPIOA_15	✓				SPI_CS	
D10(PA14)	GPIOA_14	✓				SPI_CLK	
D11(PA13)	GPIOA_13	✓		✓		SPI_MISO	
D12(PA12)	GPIOA_12	✓		✓		SPI_MOSI	

Setting up Development Environment

Step 1. Installing the Driver

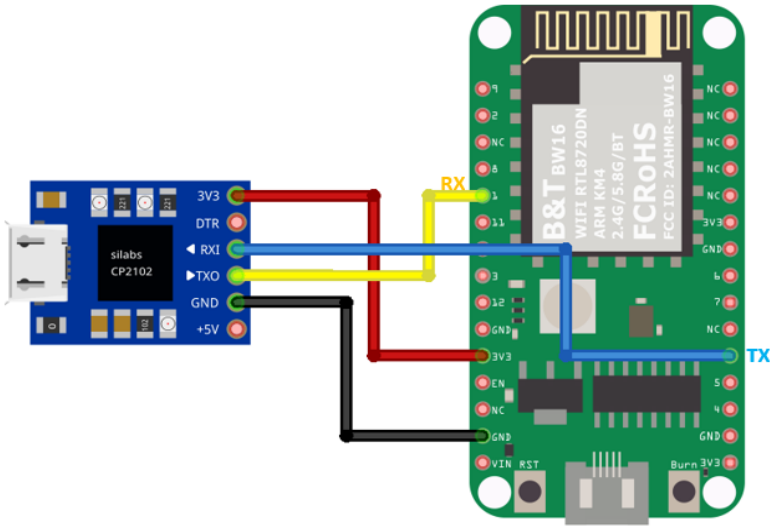


First, connect BW16 to the computer via Micro USB:

If this is the first time you connect BW16 to your computer, here is something that you might take note of:

From the pinmap above, we know D0 and D1 pins are used for program uploading. However, according to the schematic design of AI Thinker, the onboard USB-to-UART module is connected to D4 and D5 which cannot be directly used for program upload.

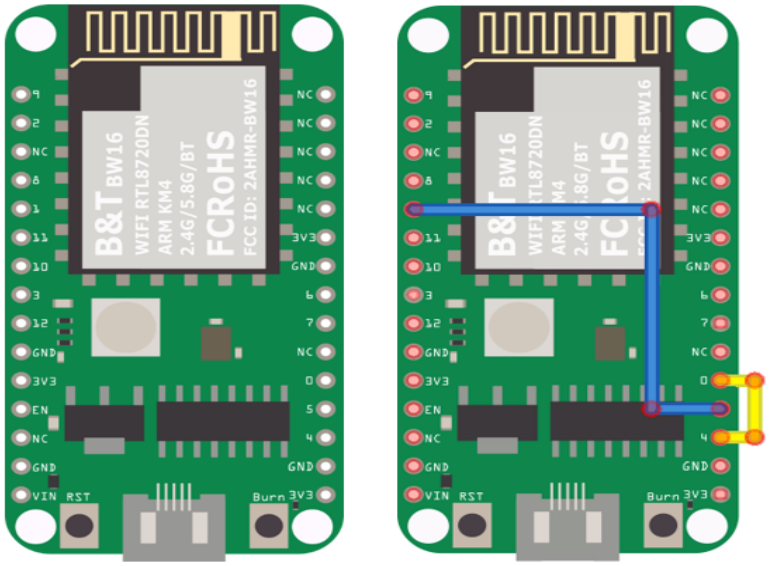
In order to upload firmware to this board, we suggested that you could choose to add in an external USB-to-UART module connecting to D0 and D1 as shown in the pin connection below:



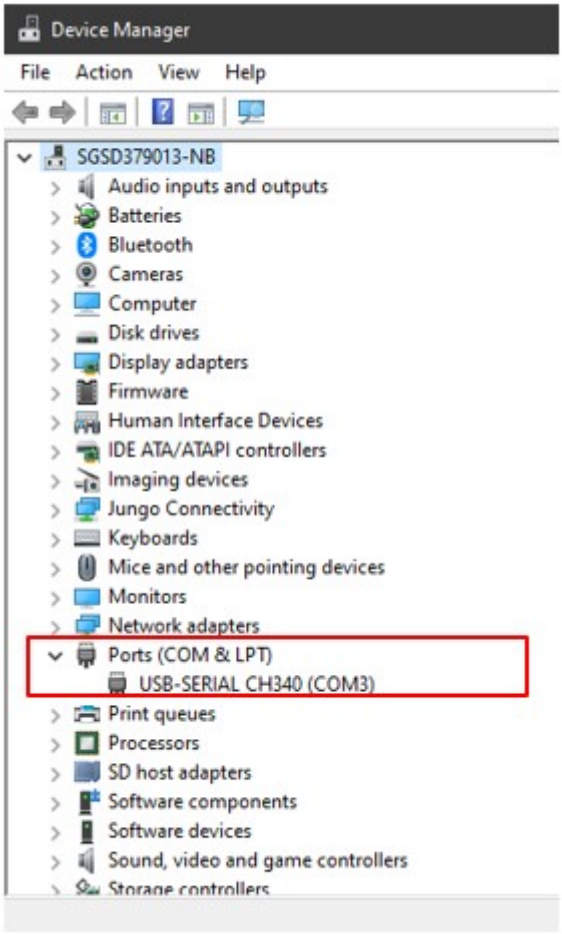
Optionally, you could short the pins indicated below to use the on-board USB:

D1 —— D5

D0 —— D4



After connecting accordingly, the USB driver for BW16 will be automatically installed. If you have driver issue of connecting board, please go to http://www.wch-ic.com/downloads/CH341SER_ZIP.html for USB driver. You can check the COM Port number in your Device Manager:



Step 2. Set up Arduino IDE

From version 1.6.5, Arduino IDE supports third-party hardware. TTherefore, we can use Arduino IDE to develop applications on BW16, and the basic examples of Arduino can run on BW16 too. Refer to the *Basic Examples*.

Arduino IDE can be downloaded in the Arduino website.

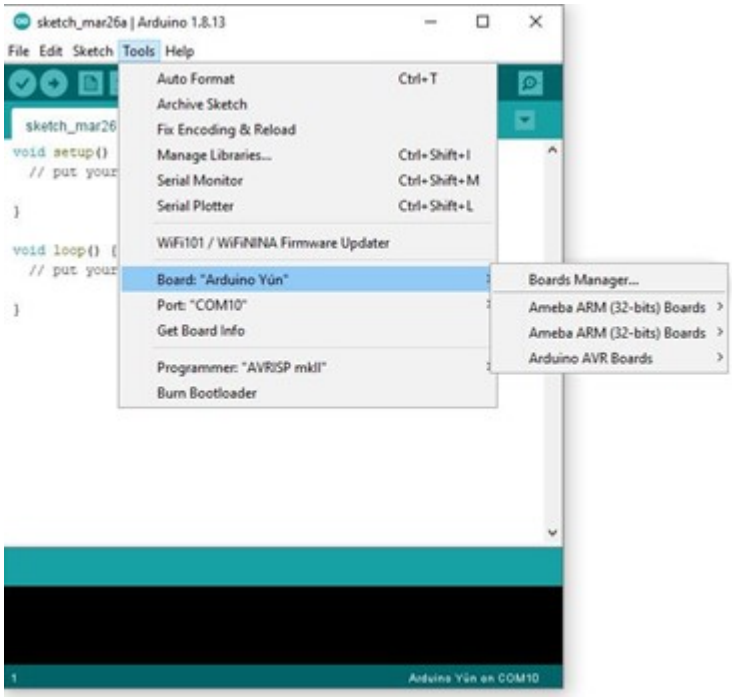
When the installation is finished, open Arduino IDE. To set up BW16 correctly in Arduino IDE, go to *"File" -> "Preferences"*.

And paste the following URL into *"Additional Boards Manager URLs"* field:

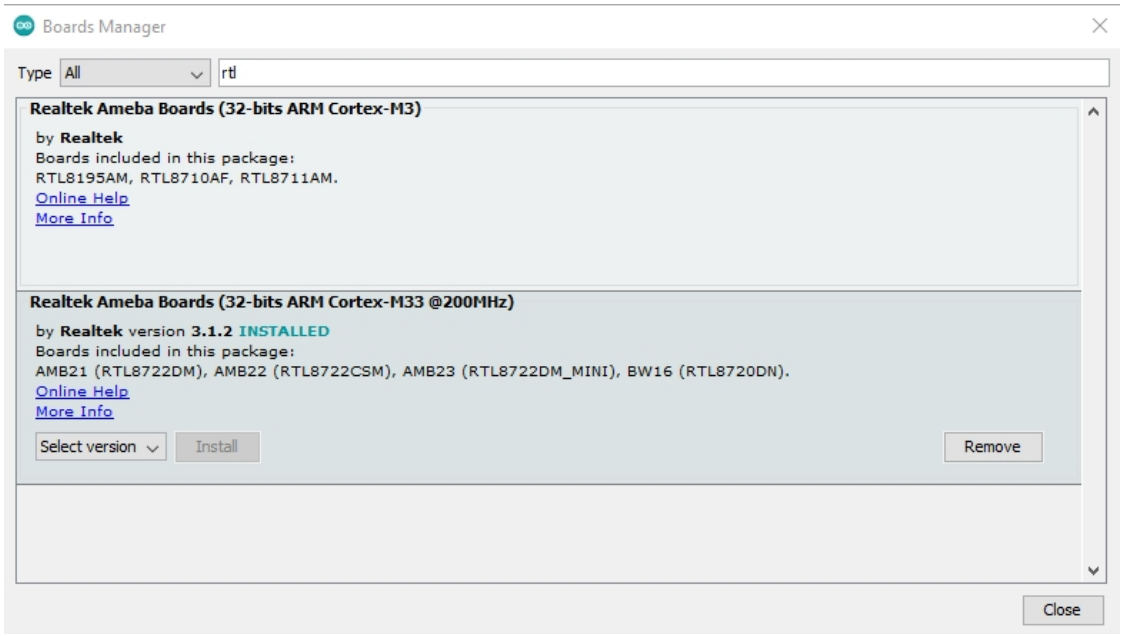
```
https://github.com/ambiot/ambd_arduino/raw/master/Arduino_package/package_realtek.com_
↪amebad_index.json
```

BW16 will be supported from v3.0.8 officially.

Next, go to *"Tools" -> "Board" -> "Boards Manager"*:



The *"Boards Manager"* requires about 10~20 seconds to refresh all hardware files (if the network is in bad condition, it may take longer). Every time the new hardware is connected, we need to reopen the Board Manager. So, we close the *"Boards Manager"*, and then open it again. Find *"Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)"* in the list, click *"Install"*, then the Arduino IDE starts to download required files for RTL8722DM.



If you are facing GitHub downloading issue, please refer to the following link at Download/Software Development Kit. There are 3 sections:

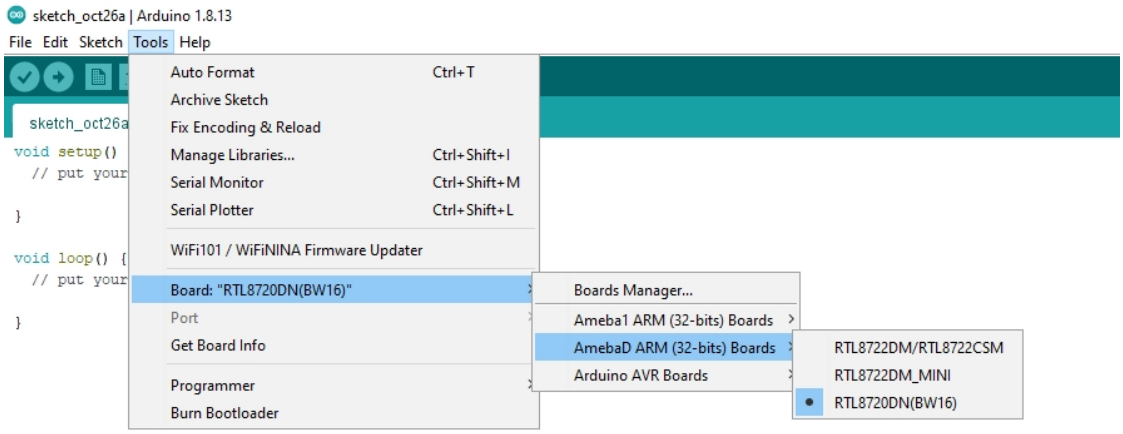
1. "AmebaD_Arduino_patch1_SDK", please select at least 1 of the SDKs. There are 5 latest released SDK options.

2. “AmebaD_Arduino_patch2_Tools”, please select according to your operation system. There are Windows, Linux and MacOS.
3. “AmebaD_Arduino_Source_Code”, this section is optional download only wants to refer the latest source code.

   Download the files selected, then unzip (patch1 and patch2 are compulsory). There are “Install.doc”/“Install.pdf” for you to refer installation steps. According to your system, please run the installation tool in the “Offline_SDK_installation_tool” folder.

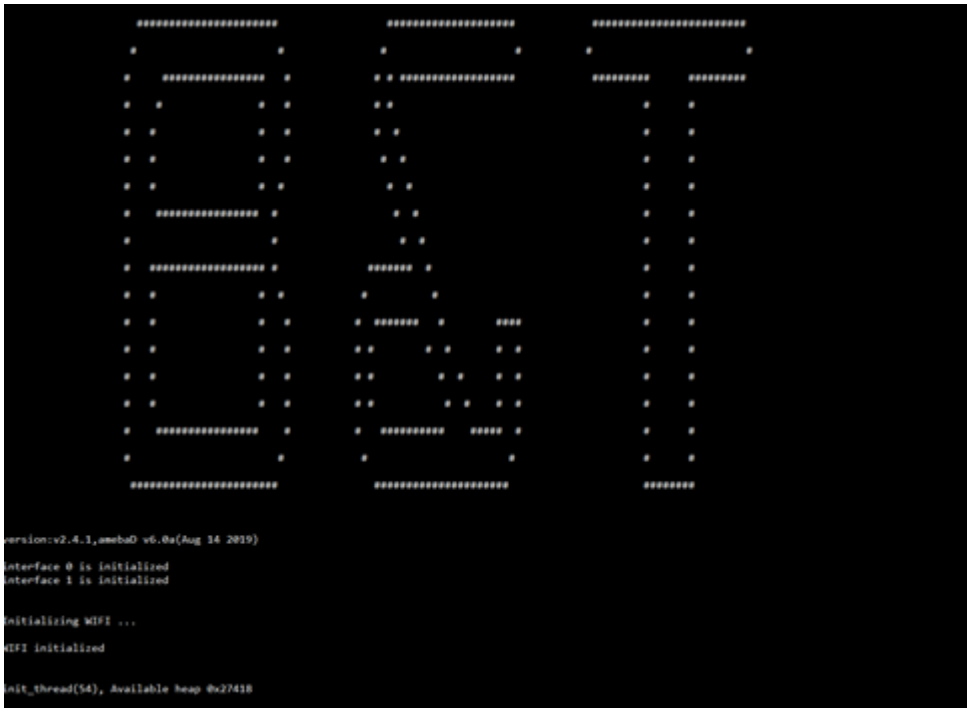
   After the installation tool running successfully, you may open Arduino IDE and proceed to *“Tools” -> “Board“ -> “Boards Manager…”*. Try to find *“Realtek AmebaD Boards (32-bits ARM Cortex-M33 @200MHz)”* in the list, click *“Install”*, then the Arduino IDE starts to download required files for AmebaD.

Finally, we select RTL8722DM as current connected board in *“Tools” -> “Board” -> “Ameba ARM (32-bits) Boards” -> “RTL8722DM”*.



How to upload firmware into BW16

Depending on the batch of manufacturing, some BW16 modules on the development board might have built-in the default B&T firmware, the firmware information is shown in the image below:



This will cause Arduino Image unable to flash into the module. Although information of "All images are sent successfully! Image tool closed! Upload Image did." is showing in the Image Tool, however, the factory image is unable to be erased. Unfortunately after press the onboard RST button, you will find the factory image still remains in the flash.

Arduino IDE provides many built-in examples, which can be compiled, uploaded and run directly on the boards. Here, we take the "Blink" example as the first try.
Open *"File"* -> *"Examples"* -> *"01.Basics"* -> *"Blink"*:

Uploading Solution

Method 1: Use AmebaD Image Tool to erase flash

The B&T default factory image can be washed using "Erase" function provided by Realtek's Image Tool. Using Image Tool to erase the flash image memory starting from memory address: 0x8000_0000 till the end of 2MB memory location, later on, we need to upload Realtek's image back to the module again using Arduino IDE.

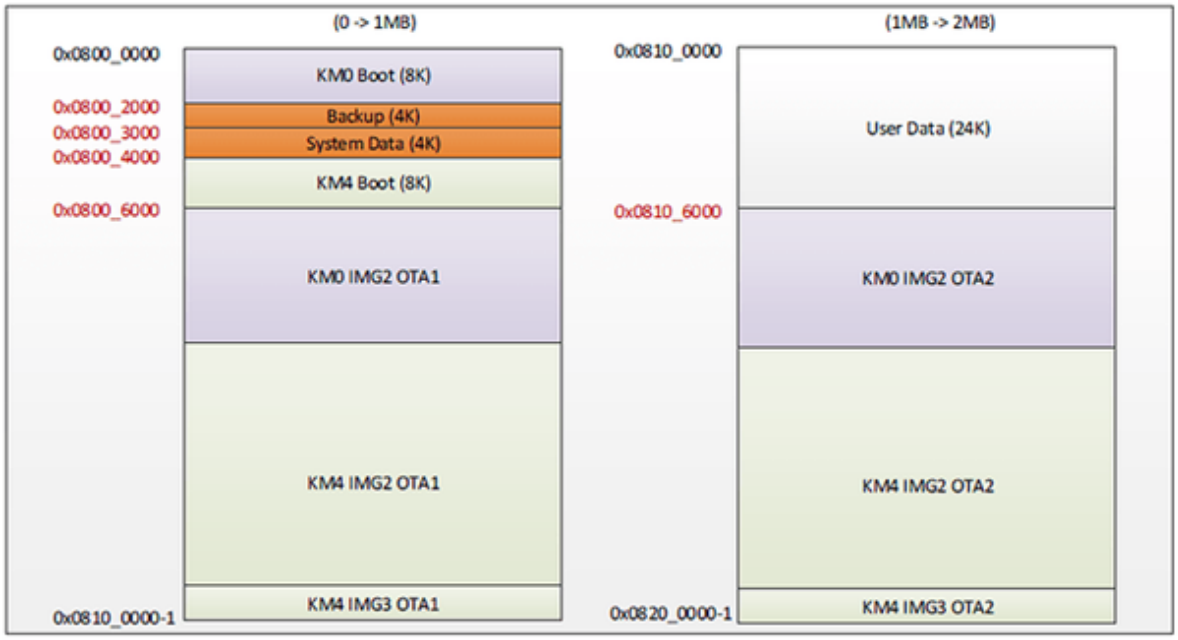


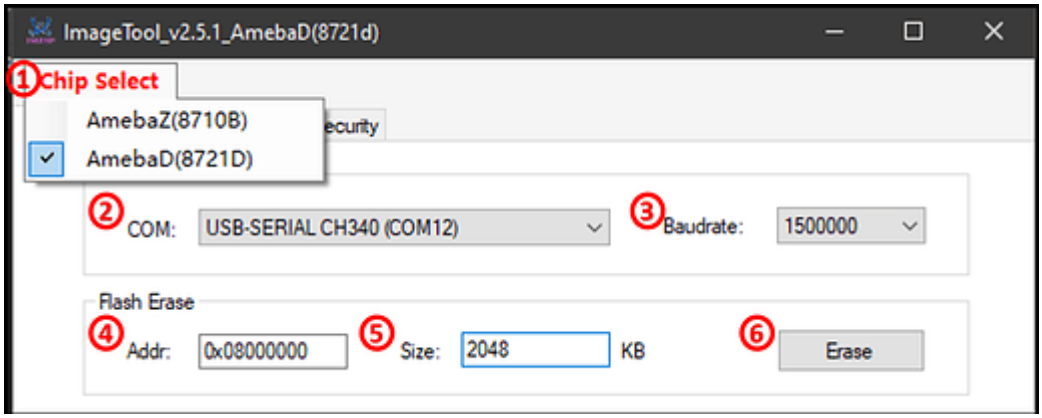
Fig 9-1 1M/2M Flash program layout

Step 1 – Download and prepare the Image Tool

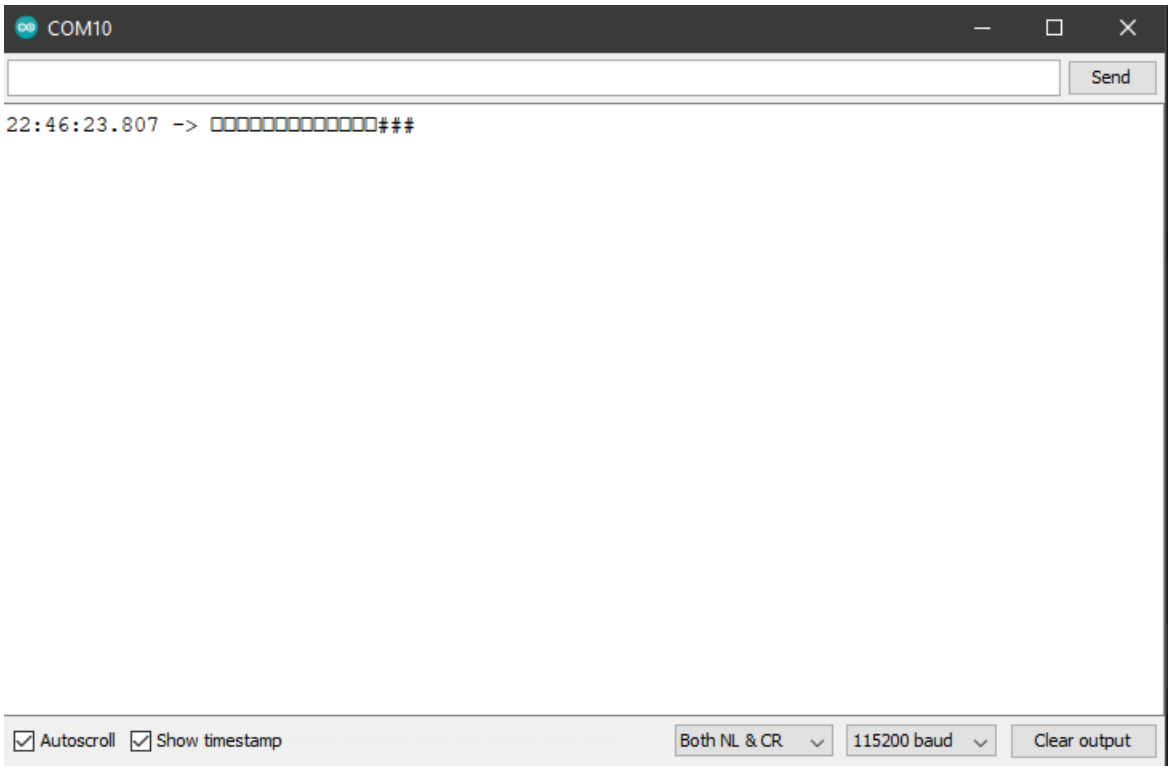
Download ambd_sdk from the link ambiot GitHub: https://github.com/ambiot/ambd_sdk.
The Realtek's Image Tool can be found under the following file path:
"ambd_sdk\tools\AmbaD\Image_tool\image_tool.exe"
Arduino IDE opens a new window with the complete sample code.

Step 2 – Setup the Image Tool

1. In the "Chip Select" option, choose "AmebaD(8721D)" which is also suitable for RTL8720DN chip.
2. Select correct COM Port that you are using.
3. Set the Baudrate to "115200".
4. Then key in the Flash Erase starting position from Memory Address of 0x0800 0000.
5. The size to be 2048 KB.
6. Set the module to "Download mode" first, then click the "Erase" button.

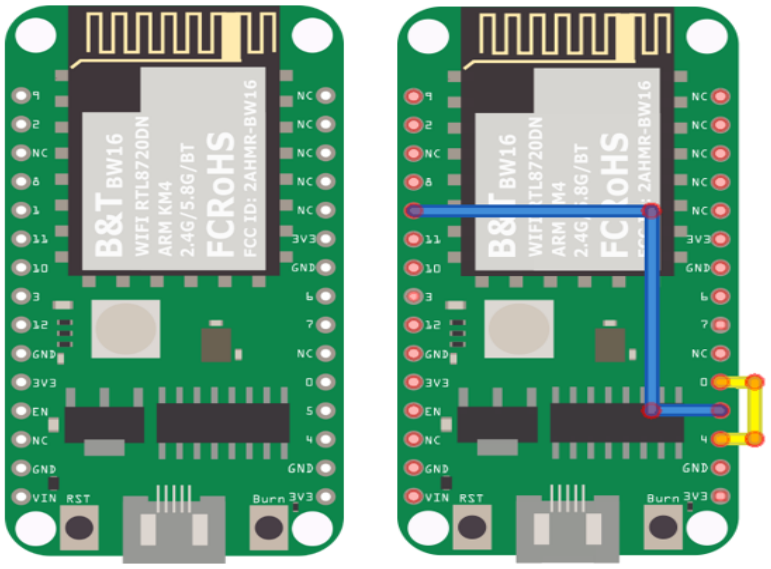


Upon finishing the above image erase and press the reset button, we could find that the `"#calibration"` will no longer pop out, only `"#"` will appear in the Serial Monitor.



Step 3 – Download Image using Arduino IDE

Now you are able to download the program via UART in Ardunio IDE. In order to upload the program, you could choose to either use an external USB-to-UART module connecting to D0 and D1, or short the pins indicated below to use the on-board USB:



D1 —— D5
D0 —— D4

Optional Uploading Solution

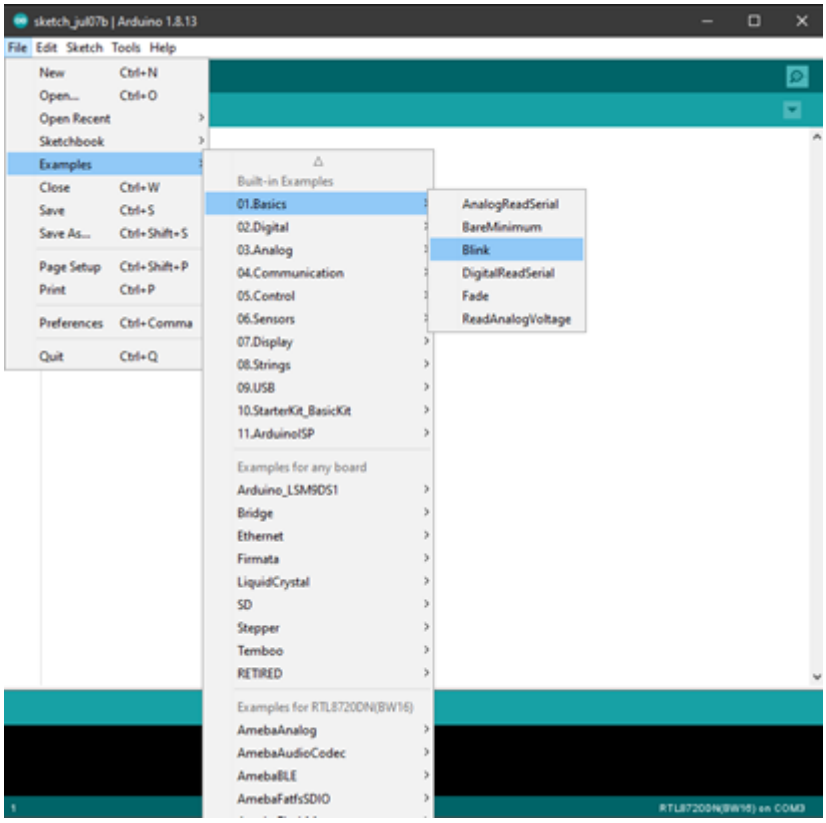
OTA (Over The Air)

Ai-Thinker is providing a guide for OTA firmware upload in Section 6.1 of B&T "RTL8720D AT Command User Manual" of which can be retrieved from this link here.

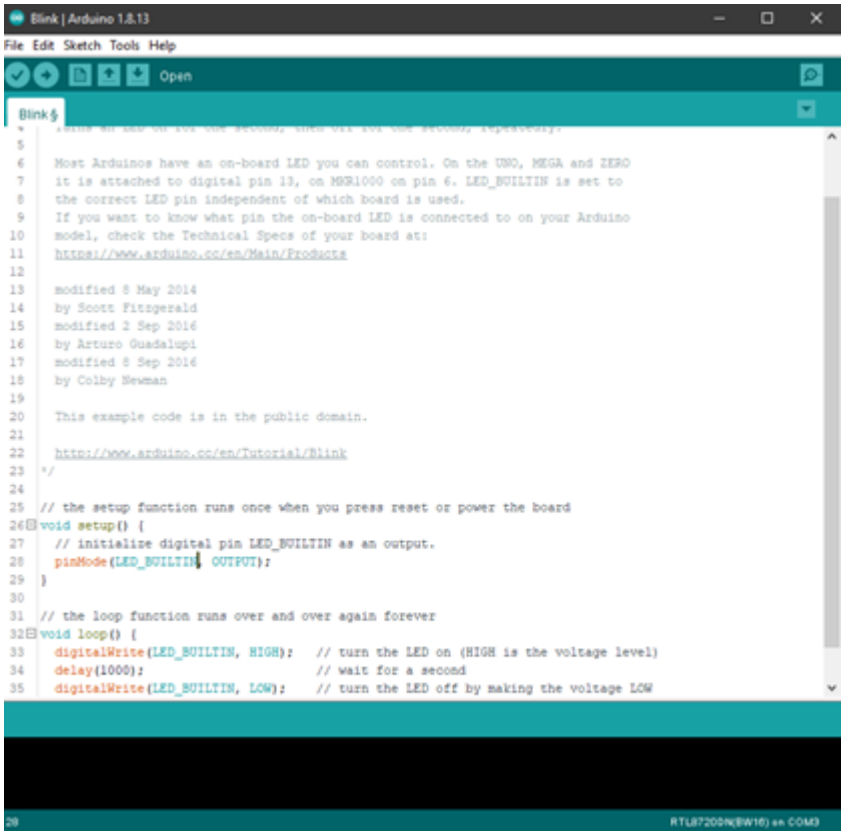
Try the First Example

Step 1. Compile & Upload

Arduino IDE provides many built-in examples, which can be compiled, uploaded, and run directly on the boards.
Here, we take the "Blink" example as the first try.
Open *"File" -> "Examples" -> "01.Basics" -> "Blink"*:

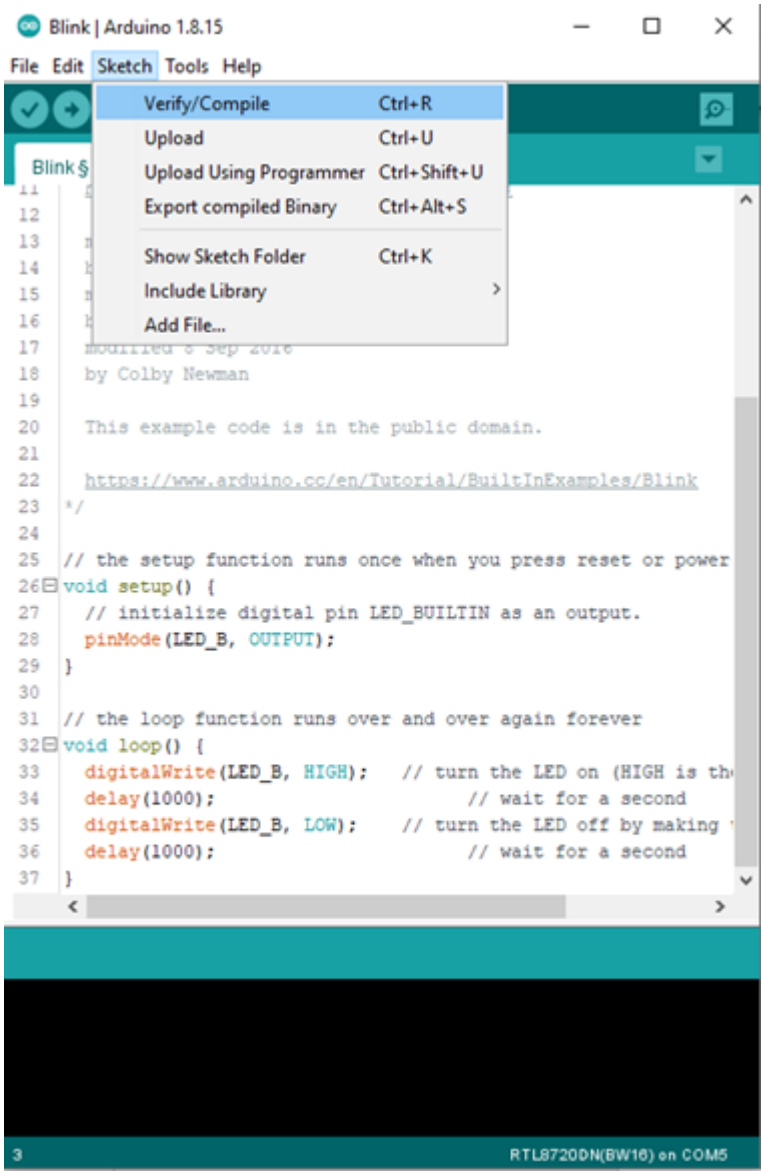


Arduino IDE opens a new window with the complete sample code.



There is an onboard LED of BW16, the default `LED_BUILTIN` is a green onboard LED. Change `LED_BUILTIN` to

`LED_B` or `LED_R` for different colors such as blue and red. Here we use `LED_B` for demonstration purpose.
Next, we compile the sample code directly; click *"Sketch"* -> *"Verify/Compile"*

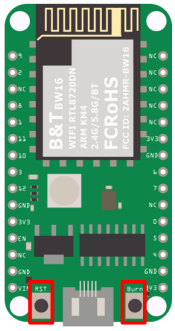


Arduino IDE prints the compiling messages in the bottom area of the IDE window. When the compilation is finished, you will get the message similar to the following figure:

Afterwards, we will upload the compiled code to BW16.

Please make sure BW16 is connected to your computer, then click *"Sketch" -> "Upload"*.

The Arduino IDE will compile first then upload. During the uploading process, users are required to enter the upload mode of the board. To enter the upload mode, first press and hold the BW16 "Burn" button, press the "RST" button, and then release the "Burn" button.



Press "Upload" button in Arduino IDE to compile and upload the program. Arduino IDE will wait for 5s for the development board to enter the upload mode.

Again, during the uploading procedure the IDE prints messages. Uploading procedure takes considerably longer time (about 30 seconds to 1 minute). When upload completed, the "Done uploading" message is printed.

Step 2.Run the Blink example

In each example, Arduino not only provides sample code, but also detailed documentation, including wiring diagram, sample code explanation, technical details, …etc. These examples can be directly used on BW16.
So, we find the detailed information of the Blink example.

In short, for BW16, the example can be run on both the onboard RGB LED or external LED (use any GPIO pins for signal output). Finally, press the "RST" button, and you can see the RGB LED turns into blue and keep blinking.

References

1. Introduction of BW16 on Instructable: https://www.instructables.com/RTL8720DN/
2. Load Arduino image into BW16: How to load BW16 program with Arduino – #13
3. BW16 IMG2 SIGN Invalid Solution: RTL8720DN(BW16) IMG2 SIGN Invalid Solution
4. FTDI Driver Download from here: https://ftdichip.com/wp-content/uploads/2021/02/CDM21228_Setup.zip

(End)

Note: If you face any issue, please refer to the FAQ and Trouble shooting sections on ../../support/index page.

1.3.2 Peripherals & Examples

Basic Examples

BW16 (RTL8720DN) Supported ARDUINO built-in example table

There are many built-in examples in Arduino. In the table below, we list all examples that are compatible with Ameba.

Please refer to the following link to set up Ameba for Arduino IDE.
https://www.amebaiot.com/en/amebad-bw16-arduino-getting-started/
Please refer to the following link for Arduino built-in example details.
https://www.arduino.cc/en/Tutorial/BuiltInExamples//

Category	Name	Comment
01. Basics	Analog ReadSerial	Connect potentiometer to 3.3V. Use ADC pin A2(PB3).
	BareMinimum	
	Blink	Pin LED_BUILTIN sets to LED_G
	Digital ReadSerial	
	Fade	Use PWM pins D7(PA25), D8(PA26), D11(PA13), D12(PA12)
	Read Analog Voltage	Use ADC pin A2(PB3).
02. Digital	BlinkWithout Delay	Pin LED_BUILTIN sets to LED_G
	Button	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	Debounce	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	Digital InputPullup	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	StateChange Detection	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	toneKeyboard	Replace “tone(8, notes[thisSensor], 20);” by a PWM pin D7(PA25), D8(PA26),
	toneMelody	
	tone Multiple	
	tonePitch Follower	
03. Analog	Analog InOutSerial	Replace “analogOutPin = 9;” by a PWM pin (D7(PA25), D8(PA26), D11(PA13
	AnalogInput	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;

Category	Name	Comment
	Analog Write Mega	Use PWM pins D7(PA25), D8(PA26), D11(PA13), D12(PA12)
	Calibration	Connect another LED to pin D13. Use ADC pin A2(PB3).
	Fading	Use PWM pins D7(PA25), D8(PA26), D11(PA13), D12(PA12).
	Smoothing	Use ADC pin A2(PB3).
04. Communication	ASCIITable	
	Dimmer	
	Graph	Connect potentiometer to 3.3V. Use ADC pin A2(PB3).
	Midi	Use Serial1 and pin D4(PB1).
	MultiSerial	Required external USB-to-UART module.
	Physical Pixel	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	Read ASCIIString	Use PWM pins for LED, D7(PA25), D8(PA26), D11(PA13), D12(PA12).
	SerialEvent	
	Serial Passthrough	Required external USB-to-UART module.
05. Control	Arrays	Use pins D9, D2, D8, D1, D11, D10.
	ForLoop Iteration	Use pins D9, D2, D8, D1, D11, D10.
	IfStatement Conditional	Replace “ledPin = 13;” by available digital pins. e.g. “ledPin = LED_BUILTIN;
	switchCase	Use ADC pin A2(PB3).
	switchCase2	Use pins D9, D2, D8, D1, D11, D10.
	While Statement Conditional	Use ADC pin A2(PB3). Replace “indicatorLedPin = 13;” by available digital pi
06. Display	barGraph	Use ADC pin A2(PB3).
07. Strings	Character Analysis	
	StringAddition Operator	
	StringAppend Operator	
	String CaseChanges	
	String Characters	
	String Comparision Operators	Use ADC pin A2(PB3).
	StringIndexOf	
	StringLength	
	String LengthTrim	
	String Replace	
	String Starts WithEndsWith	
	String Substring	
	StringToInt	

Network Examples

BLE - BLE Battery Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery service is set up on the Ameba Bluetooth stack. A mobile phone is used to connect to the Ameba peripheral device and read the battery data.

Procedure

Ensure that the following Bluetooth apps are installed on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.

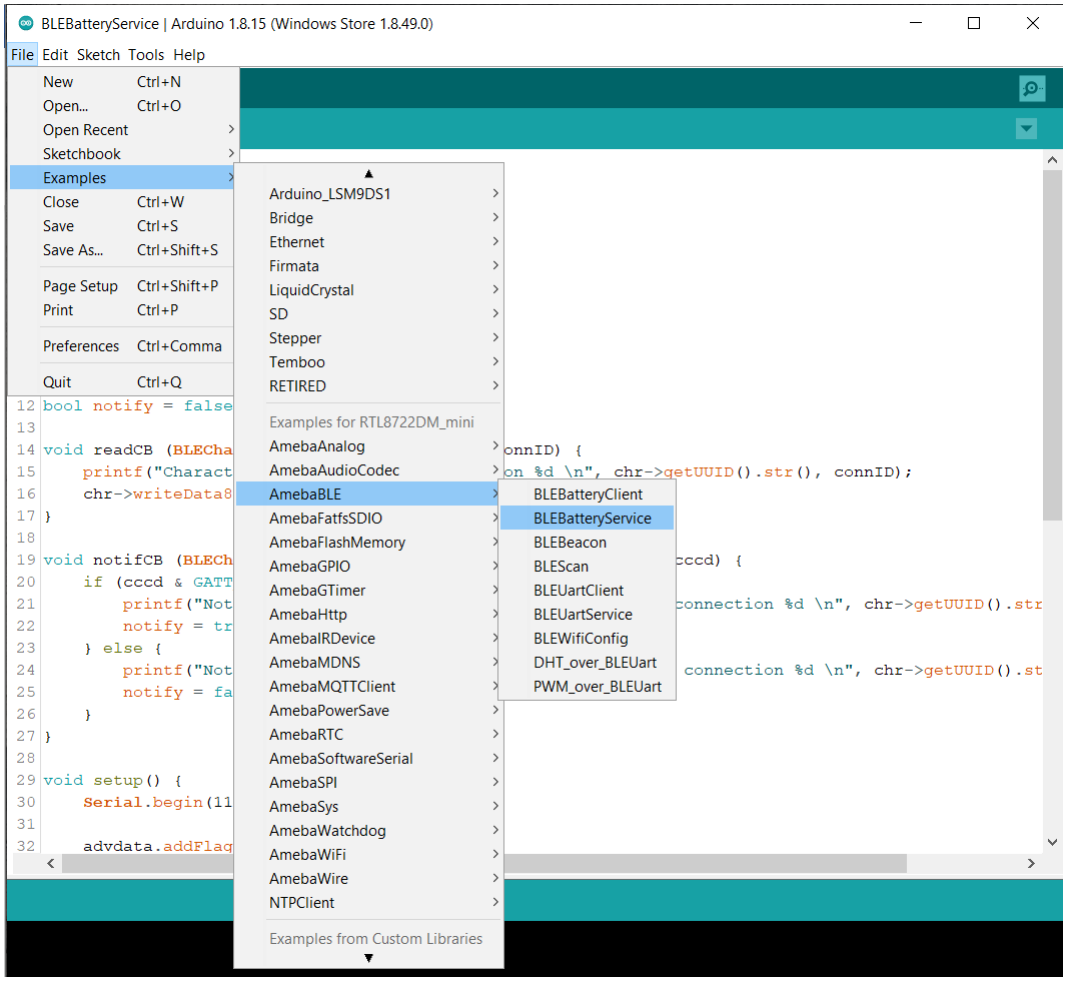
The recommended application is nRF connect, and is available at the links below:

- Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS : https://apps.apple.com/us/app/nrf-connect/id1054362403

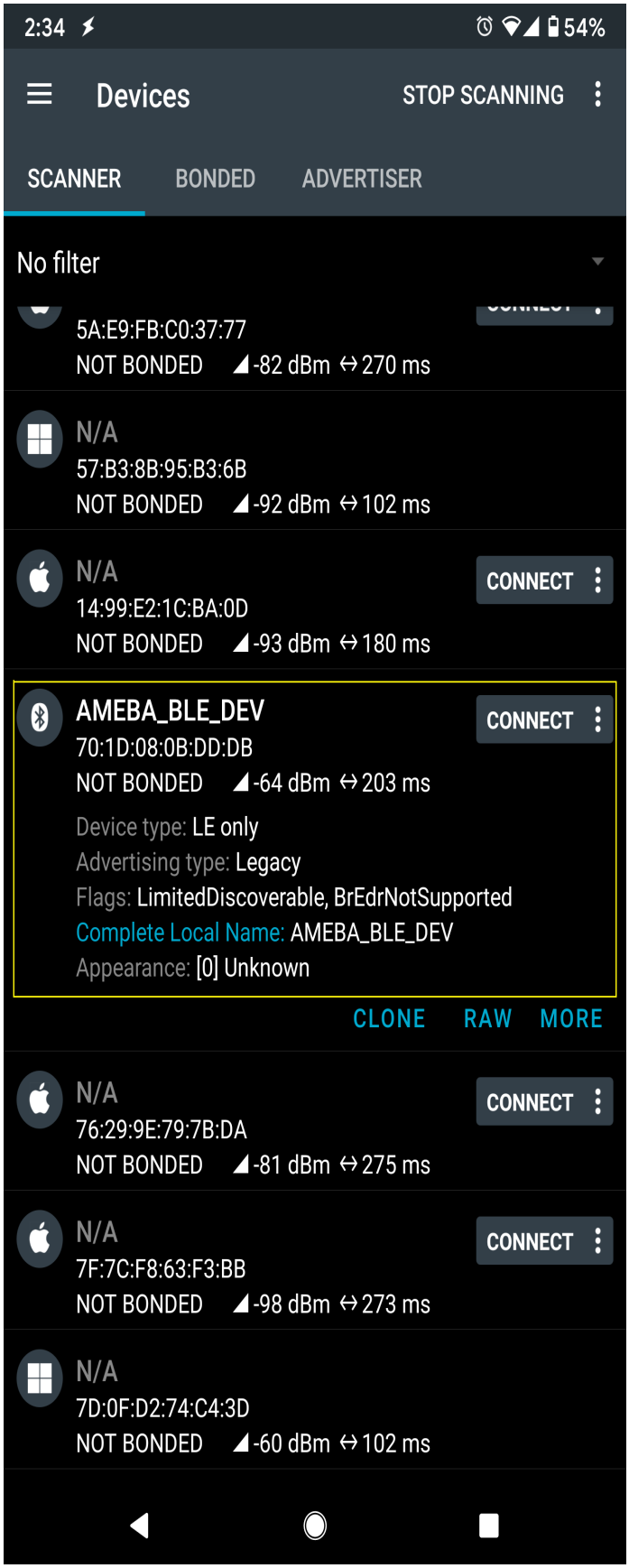
LightBlue is an alternative application that can also be used, but has less features:

- Android: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS : https://apps.apple.com/us/app/lightblue/id557428110

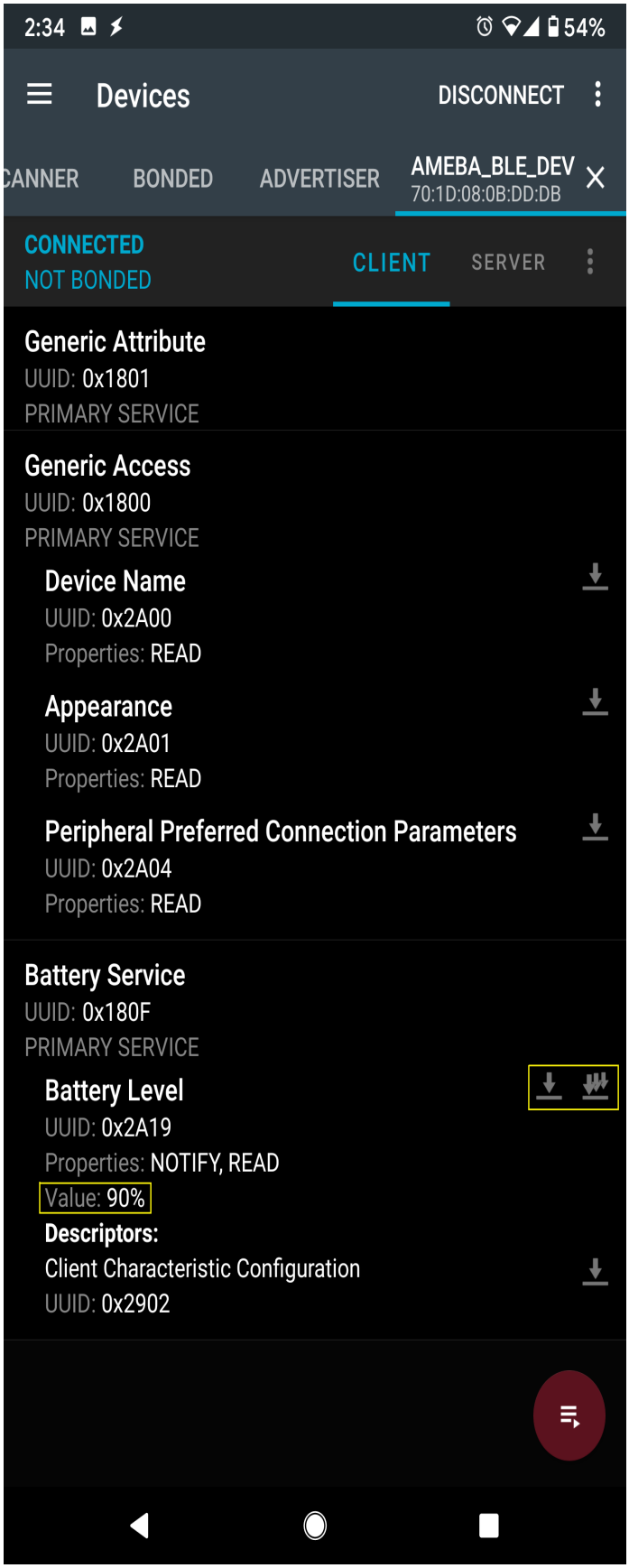
Open the example, "`Files` -> "`Examples`" -> "`AmebaBLE`" -> "`BLEBatteryService`"



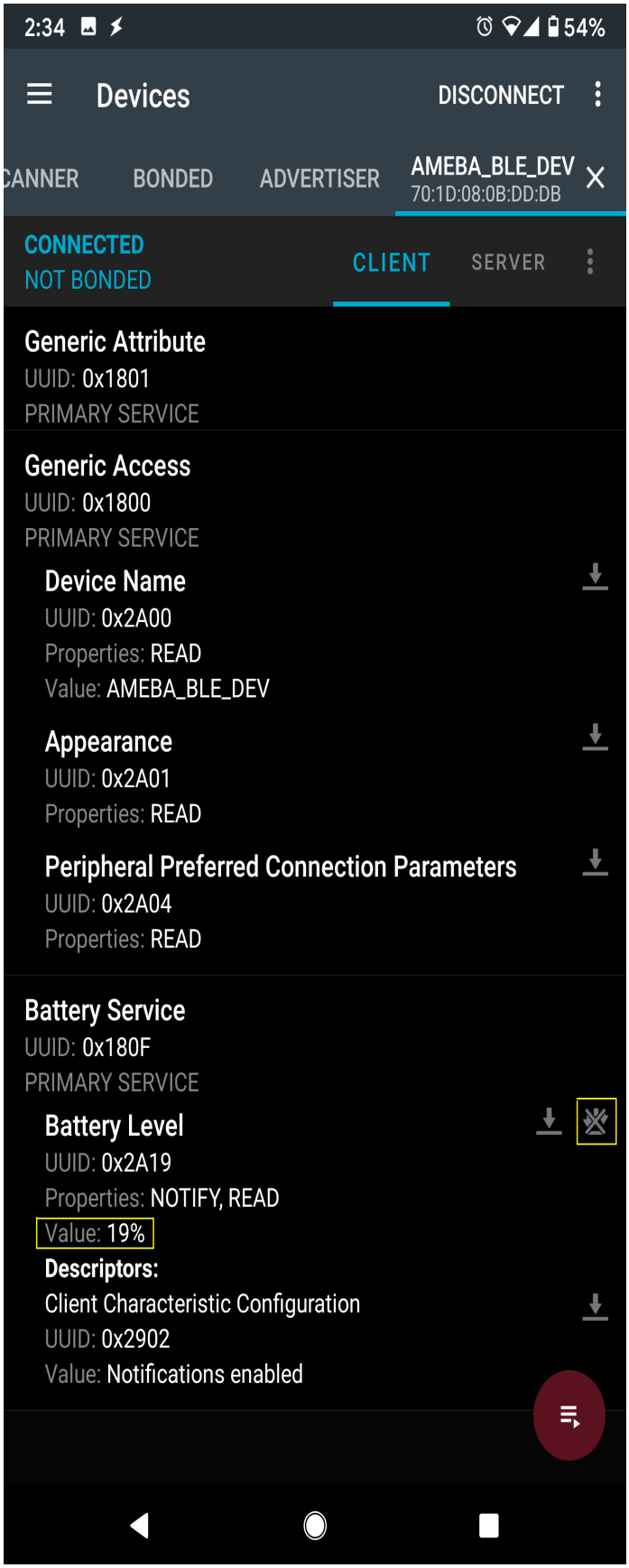
Upload the code and press the reset button on Ameba once the upload is finished.
On your mobile phone, open the Bluetooth app and scan for the Bluetooth signal broadcast by Ameba, it should appear as a device named “AMEBA_BLE_DEV”.


2:34
54%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
No filter
5A:E9:FB:C0:37:77
NOT BONDED -82 dBm 270 ms
N/A
57:B3:8B:95:B3:6B
NOT BONDED -92 dBm 102 ms
N/A
CONNECT
14:99:E2:1C:BA:0D
NOT BONDED -93 dBm 180 ms
AMEBA_BLE_DEV
CONNECT
70:1D:08:0B:DD:DB
NOT BONDED -64 dBm 203 ms
Device type: LE only
Advertising type: Legacy
Flags: LimitedDiscoverable, BrEdrNotSupported
Complete Local Name: AMEBA_BLE_DEV
Appearance: [0] Unknown
CLONE
RAW
MORE
N/A
CONNECT
76:29:9E:79:7B:DA
NOT BONDED -81 dBm 275 ms
N/A
CONNECT
7F:7C:F8:63:F3:BB
NOT BONDED -98 dBm 273 ms
N/A
7D:0F:D2:74:C4:3D
NOT BONDED -60 dBm 102 ms

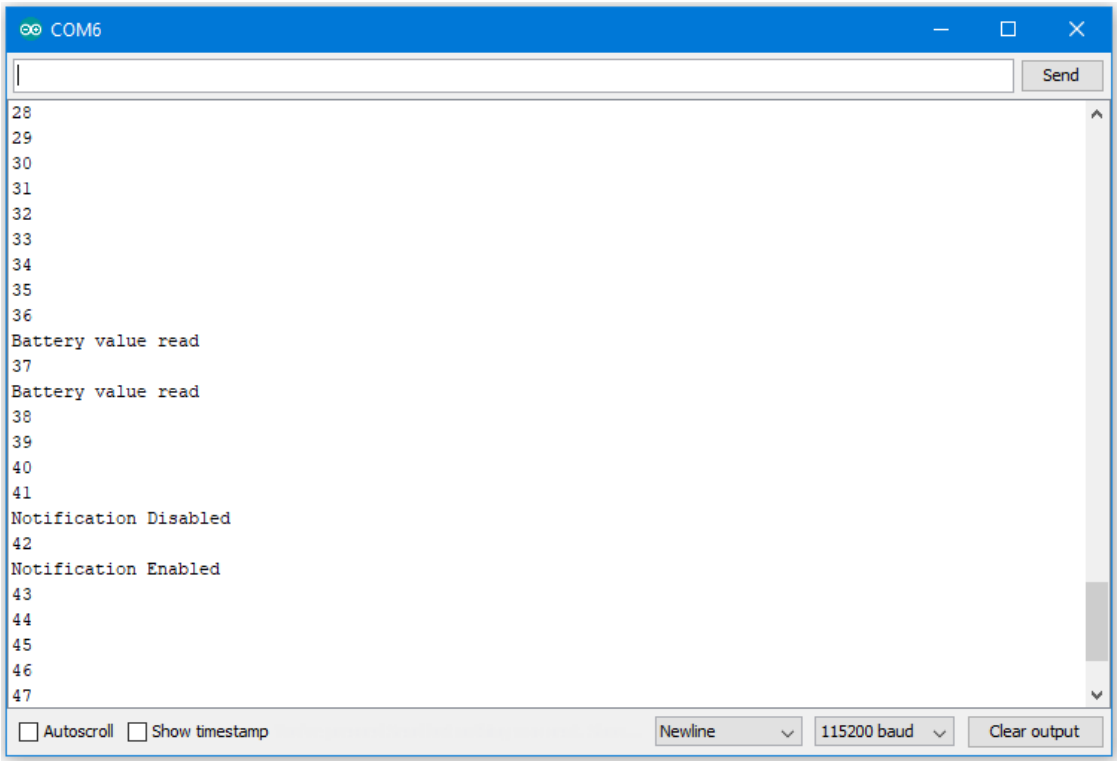
Connect to the Ameba Bluetooth device, and a list of available services should appear. Click on the battery service to expand it, and you can see the battery level data value. The arrows highlighted in the box on the right are used to read data and subscribe to notifications. Click on the single arrow to read the battery level value, and a 90% value will appear.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 90%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Click on the triple arrow to subscribe to updates on the battery level value, and the battery value will start updating by itself.


2:34
54%
Devices
DISCONNECT
CANNER
BONDED
ADVERTISER
AMEBA_BLE_DEV
70:1D:08:0B:DD:DB
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ
Value: AMEBA_BLE_DEV
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Battery Service
UUID: 0x180F
PRIMARY SERVICE
Battery Level
UUID: 0x2A19
Properties: NOTIFY, READ
Value: 19%
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

The serial monitor will show the sketch increasing the battery level every second. When you click on either of the arrows, the sketch running on the Ameba will be notified, and will print out the action taken.



Code Reference

BLEService and BLECharacteristic classes are used to create and define the battery service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(GAP_ADTYPE_ADV_IND)` is used to set the advertisement type to a general undirected advertisement that allows for connections.

`setReadCallback()` and `setCCCDCallback()` is used to register functions that will be called when the battery level data is read, or notification is enabled by the user.

`BLE.configServer(1)` is used to tell the Bluetooth stack that there will be one service running.

`addService()` registers the battery service to the Bluetooth stack.

BLE – BLE Beacon

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

A BLE beacon broadcasts its identity to nearby Bluetooth devices, to enable the other devices to determine their location relative to the beacon, and to perform actions based on information broadcasted by the beacon.

Example applications of beacons include indoor positioning system, location-based advertising and more.

From the definition of its purpose as a broadcast device, a BLE beacon thus cannot be connected to, and can only send information in its Bluetooth advertisement packets.

There are several BLE beacon protocols. The Ameba BLEBeacon library supports the iBeacon and AltBeacon protocols.

Procedure

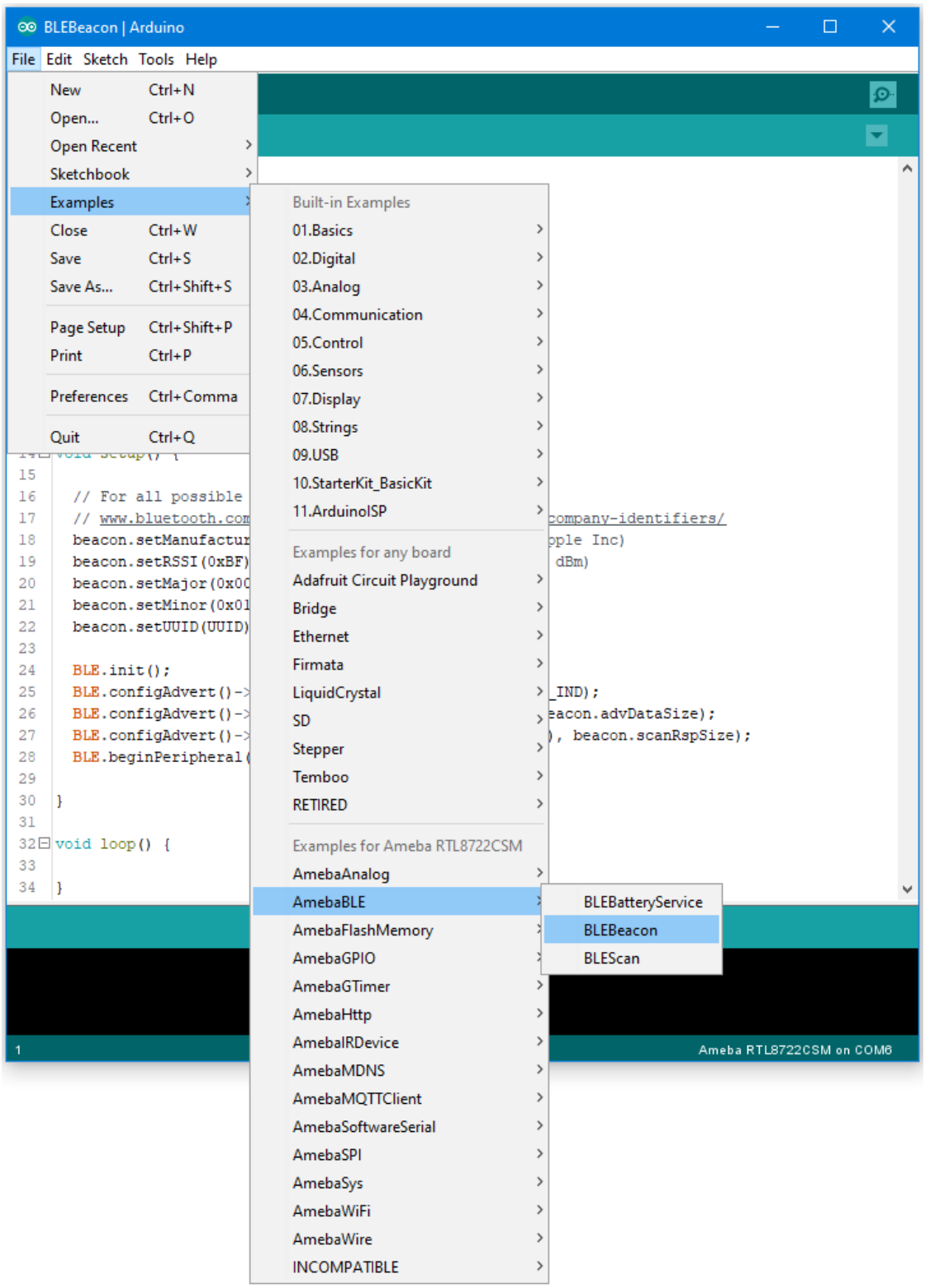
First, you need to install some Bluetooth apps on your mobile phone. These apps will show you the raw data sent by Ameba and allow you to interact with the data.
The recommended application is nRF connect, and is available at the links below:

- **Android** : https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- iOS :https://apps.apple.com/us/app/nrf-connect/id1054362403

LightBlue is an alternative application that can also be used, but has less features:

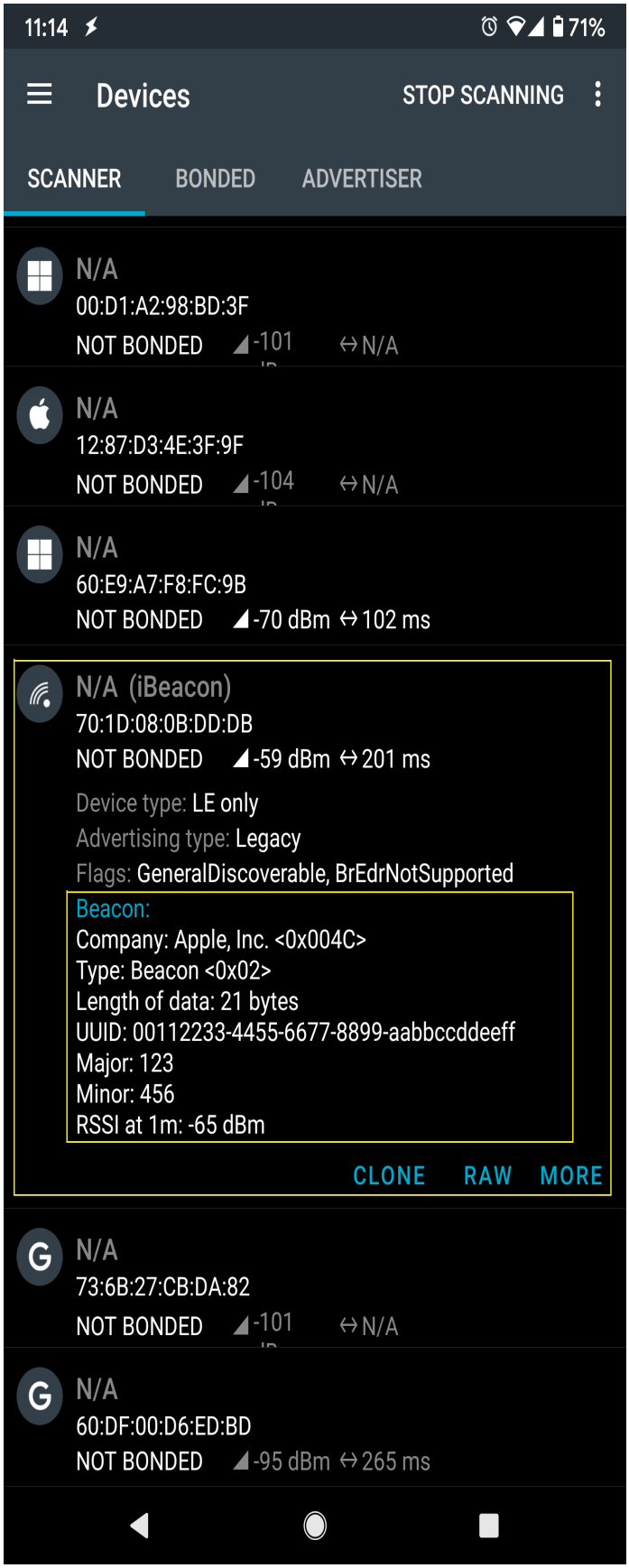
- **Android** : https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer
- iOS :https://apps.apple.com/us/app/lightblue/id557428110

Open the example, “`Files` -> “`Examples` -> “`AmebaBLE` -> “`BLEBeacon`”



Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Bluetooth app and scan for the beacon signal broadcast by Ameba.


11:14
71%
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
N/A
00:D1:A2:98:BD:3F
NOT BONDED
-101
N/A
N/A
12:87:D3:4E:3F:9F
NOT BONDED
-104
N/A
N/A
60:E9:A7:F8:FC:9B
NOT BONDED
-70 dBm
102 ms
N/A (iBeacon)
70:1D:08:0B:DD:DB
NOT BONDED
-59 dBm
201 ms
Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 00112233-4455-6677-8899-aabbccddeeff
Major: 123
Minor: 456
RSSI at 1m: -65 dBm
CLONE
RAW
MORE
N/A
73:6B:27:CB:DA:82
NOT BONDED
-101
N/A
N/A
60:DF:00:D6:ED:BD
NOT BONDED
-95 dBm
265 ms

If you happen to be in an environment with multiple BLE beacons, you can tap the entries to expand them, and verify that the beacon data is identical to the data in the sketch.

Code Reference

`setRssi()` is used to set the received signal strength indicator (rssi) data field for a beacon. The specification states that this should be the received signal strength from the beacon at a 1 meter distance. With no method to measure this, it is set to -65dBm as an estimate.

`setMajor()` and `setMinor()` are used to set the two data fields. The purpose of these data are left for the manufacturer of the beacon to define, and can be used in any way.

`setUUID()` is used to give the beacon a universally unique identifier (UUID). This is a 128-bit number usually expressed as a hexadecimal string. It is used to identify each unique beacon, and can be randomly generated for free online.

The BLEBeacon library includes both iBeacon and AltBeacon classes, replace line 6 iBeacon with altBeacon to create an AltBeacon instead. The data fields are mostly the same, with only minor changes, please look at the header files for more details.

`BLE.init()` is used to allocate memory and prepare Ameba for starting the Bluetooth stack.

`BLE.configAdvert()` is used to configure the Bluetooth advertisement settings, to which we pass the beacon data and set the device as non-connectable.

`BLE.beginPeripheral()` starts Ameba in Bluetooth peripheral mode, after which it will begin to advertise with the beacon data provided.

BLE – BLE Scan

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

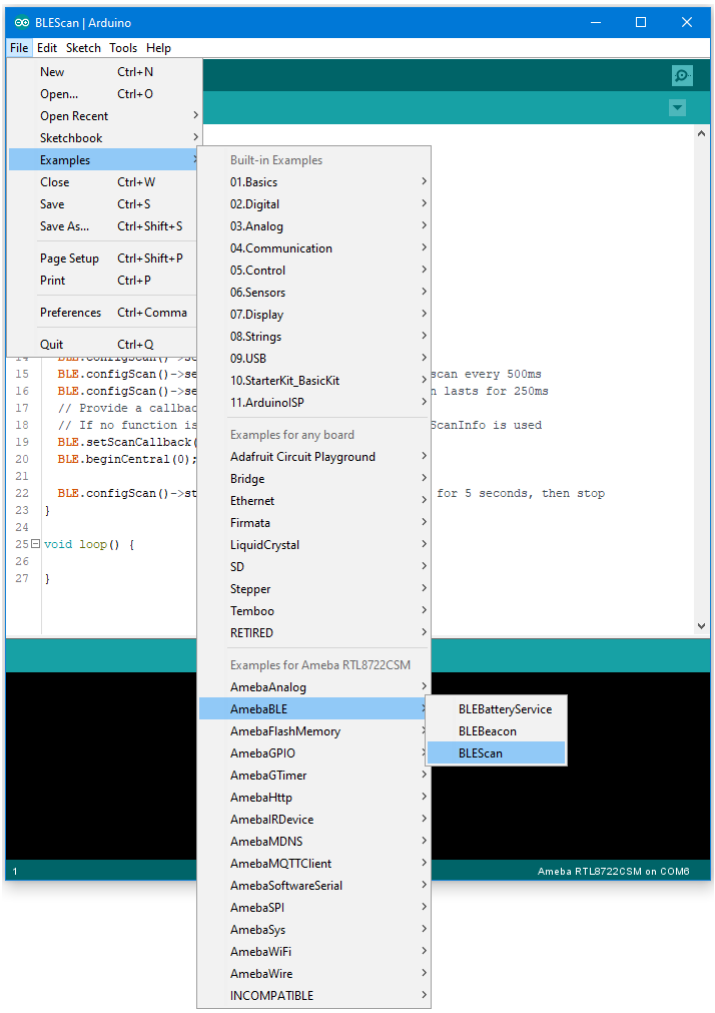
Example

Introduction

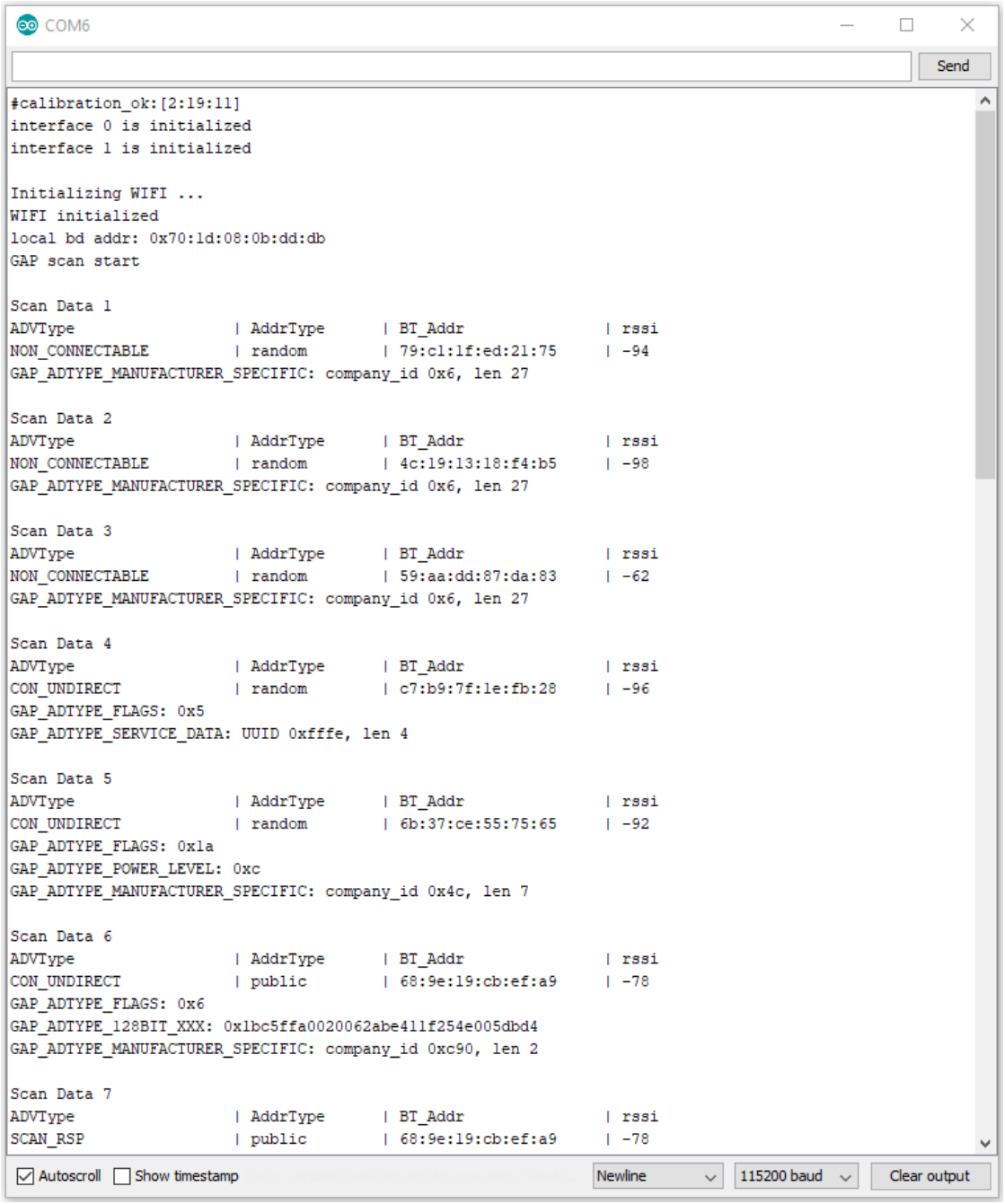
This example configures the Ameba as a Bluetooth central device, uses the scan functionality to scan for other Bluetooth devices, and prints out the results to the serial monitor.

Procedure

Open the example, "`Files`" -> "`Examples`" -> "`AmebaBLE`" -> "`BLEScan`"



Upload the code and press the reset button on Ameba once the upload is finished.
Open the Arduino serial monitor, and you should see the scan results of nearby Bluetooth devices formatted and printed out.



If you have the Bluetooth app nRF Connect installed, you can also use it to send out Bluetooth advertisements for the Ameba to pick up.

Code Reference

`setScanMode(GAP_SCAN_MODE_ACTIVE)` is used to set the scan mode. Active scanning will request for an additional scan response data packet from a device when it is found. Passive scanning will only look at the advertisement data, and not request for additional data.

`setScanInterval()` and `setScanWindow()` are used to set the frequency and duration of scans in milliseconds. A scan will start every interval duration, and each scan will last for the scan window duration. The scan window duration

should be lesser or equal to the scan interval. Set a short interval to discover devices rapidly, set a long interval to conserve power.

`setScanCallback(scanFunction)` is used to register a function to be called when scan results are received. This can be used to set a user function for additional processing of scan data, such as looking for a specific device. If no function is registered, the scan results are formatted and printed to the serial monitor by default.

`beginCentral(0)` is used to start the Bluetooth stack in Central mode. The argument 0 is used to indicate that no clients will be operating in central mode.

`startScan(5000)` is used to start the scanning process for a specified duration of 5000 milliseconds. The scan will repeat according to the set scan interval and scan window values. After 5000 milliseconds, the scan process will stop, and will be ready to be started again.

BLE – Battery Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Introduction

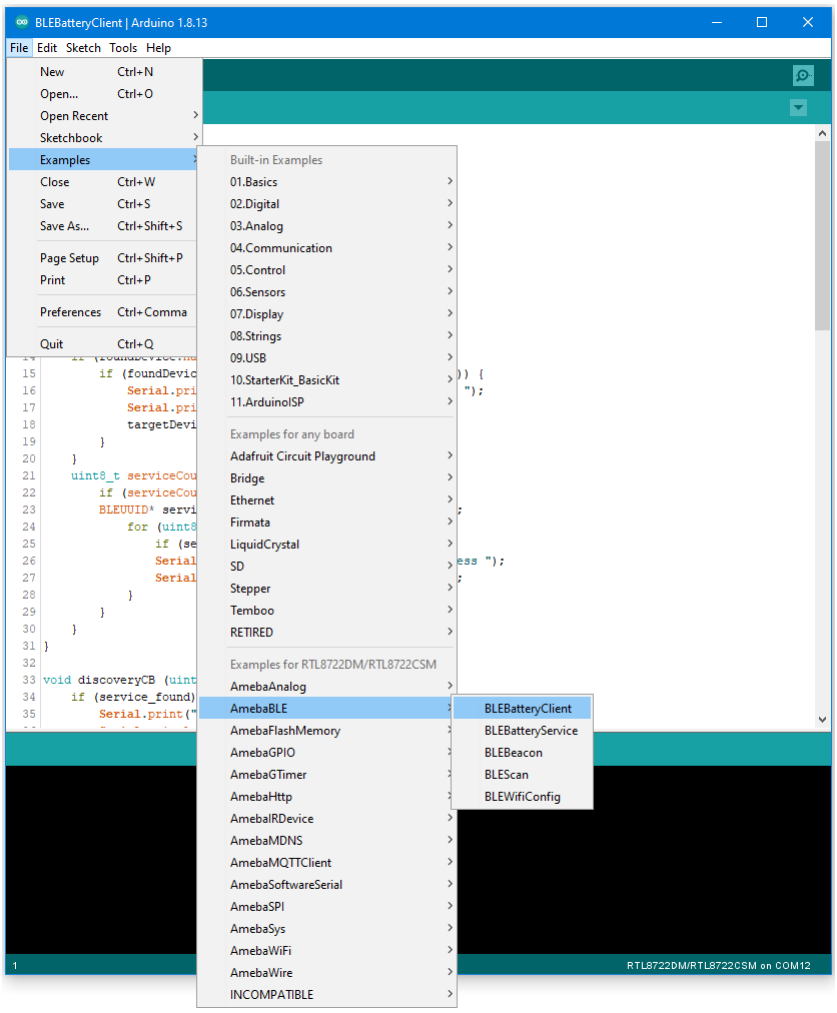
BLE connections use a server client model. The server contains the data of interest, while the client connects to the server to read the data. Commonly, a Bluetooth peripheral device acts as a server, while a Bluetooth central device acts as a client. Servers can contain many services, with each service containing a some set of data. Clients can send requests to read or write some data and can also subscribe to notifications so that the server can send data updates to a client.

In this example, a basic battery client is set up on the Ameba Bluetooth stack. The client connects to another Ameba board running the corresponding BLE battery service to read the battery level data.

Procedure

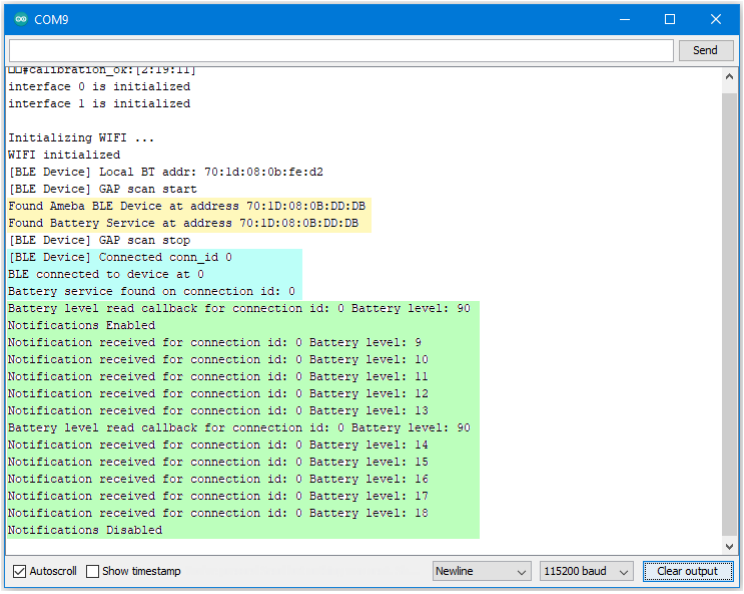
On the first Ameba board, upload the BLEBatteryService example code and let it run.

For the second Ameba board, open the example "`Files" -> "Examples" -> "AmebaBLE" -> "BLEBatteryClient`".



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor and observe the log messages as the Ameba board with the battery client scans, connects, and reads data from the Ameba board with the battery service.



Highlighted in yellow, the Ameba board with the battery client first scans for advertising BLE devices with the advertised device name "AMEBA_BLE_DEV" and the advertised service UUID of 0x180F representing the battery service.

After finding the target device, the Ameba board with the battery client forms a BLE connection and searches for a battery service on the connected device, highlighted in blue.

With the client connected to the service, the battery client begins to read data using both regular data reads and notifications, highlighted in green.

Code Reference

BLEClient is used to create a client object to discover services and characteristics on the connected device.

- `setNotifyCallback()` is used to register a function that will be called when a battery level notification is received.
- `BLE.configClient()` is used to configure the Bluetooth stack for client operation.
- `addClient(connID)` creates a new BLEClient object that corresponds to the connected device.

BLE – WiFi Configuration Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS mobile phone

Example

Introduction

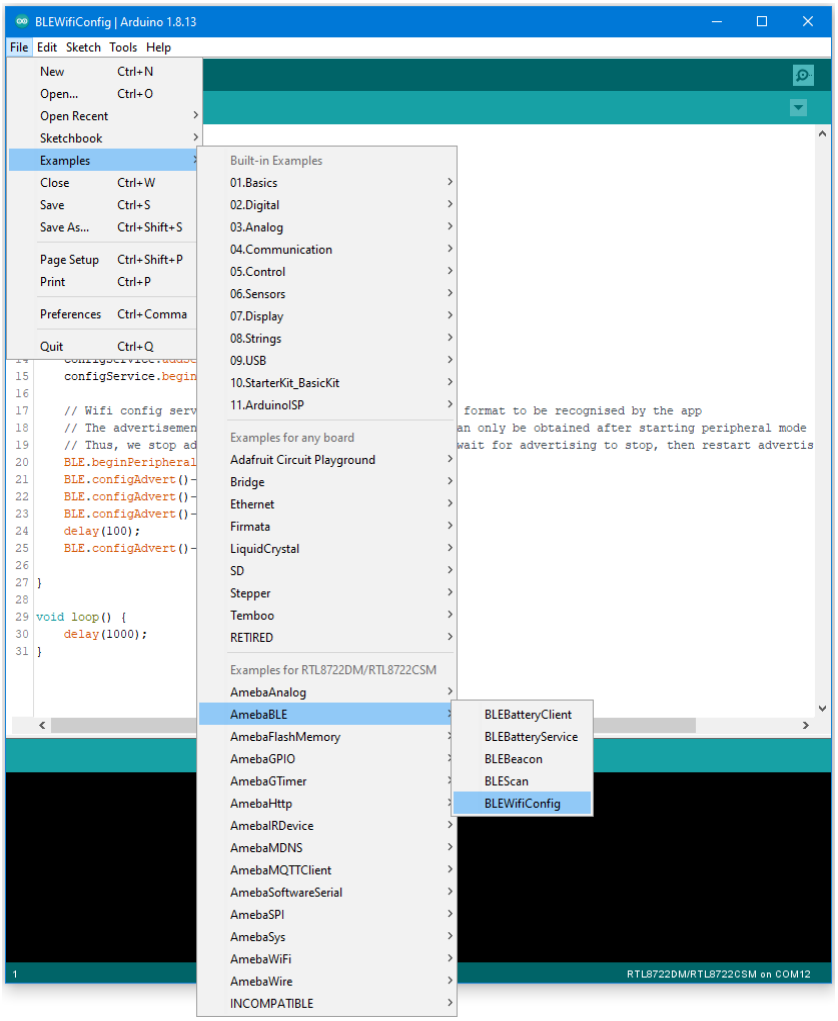
In this example, a WiFi configuration service is set up on the Ameba Bluetooth stack. A mobile phone with the configuration app connects to the Ameba device using BLE and configures the Ameba to connect to the correct WiFi access point.

Procedure

Ensure that the Realtek WiFi configuration app is installed on your mobile phone, it is available at:

– Google Play Store: https://play.google.com/store/apps/details?id=com.rtk.btconfig
– Apple App Store: https://apps.apple.com/sg/app/easy-wifi-config/id1194919510

Open the example, "`Files` -> "`Examples`" -> "`AmebaBLE`" -> "`BLEWifiConfigService`".

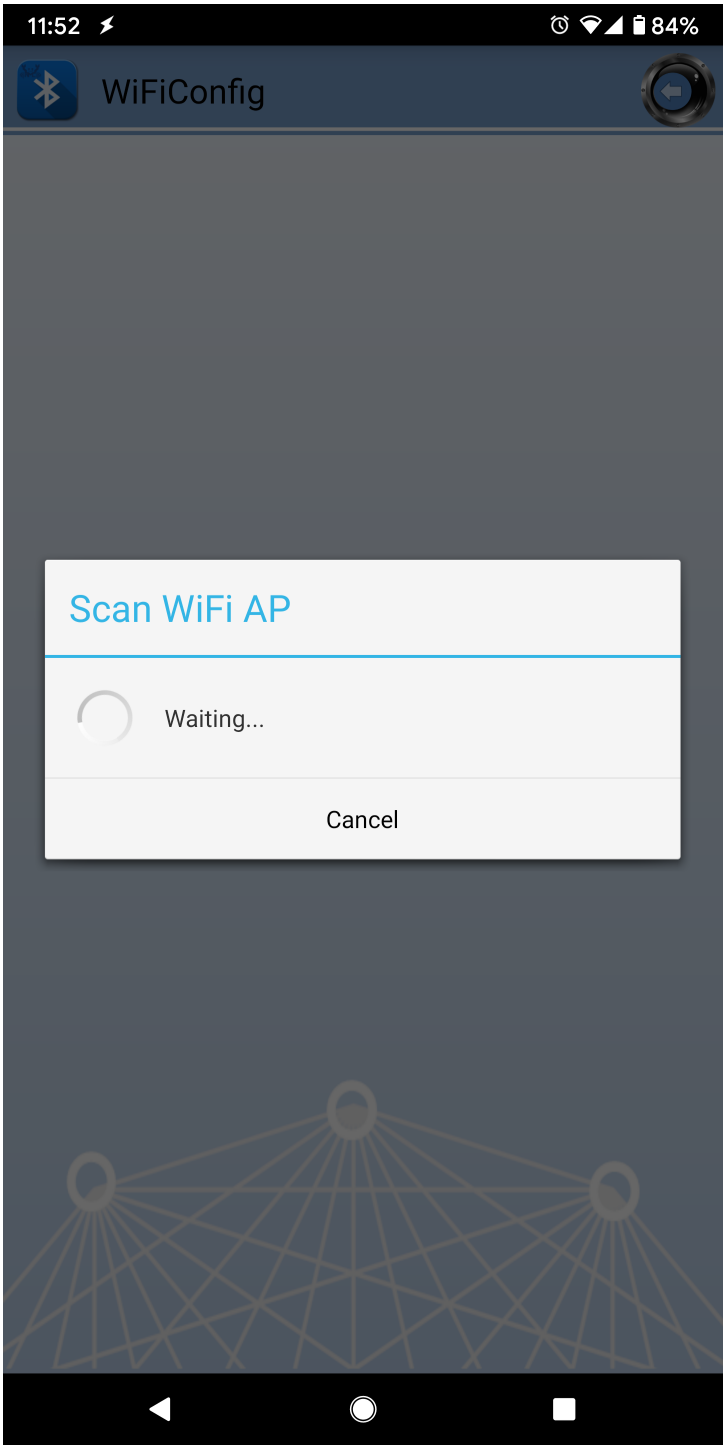


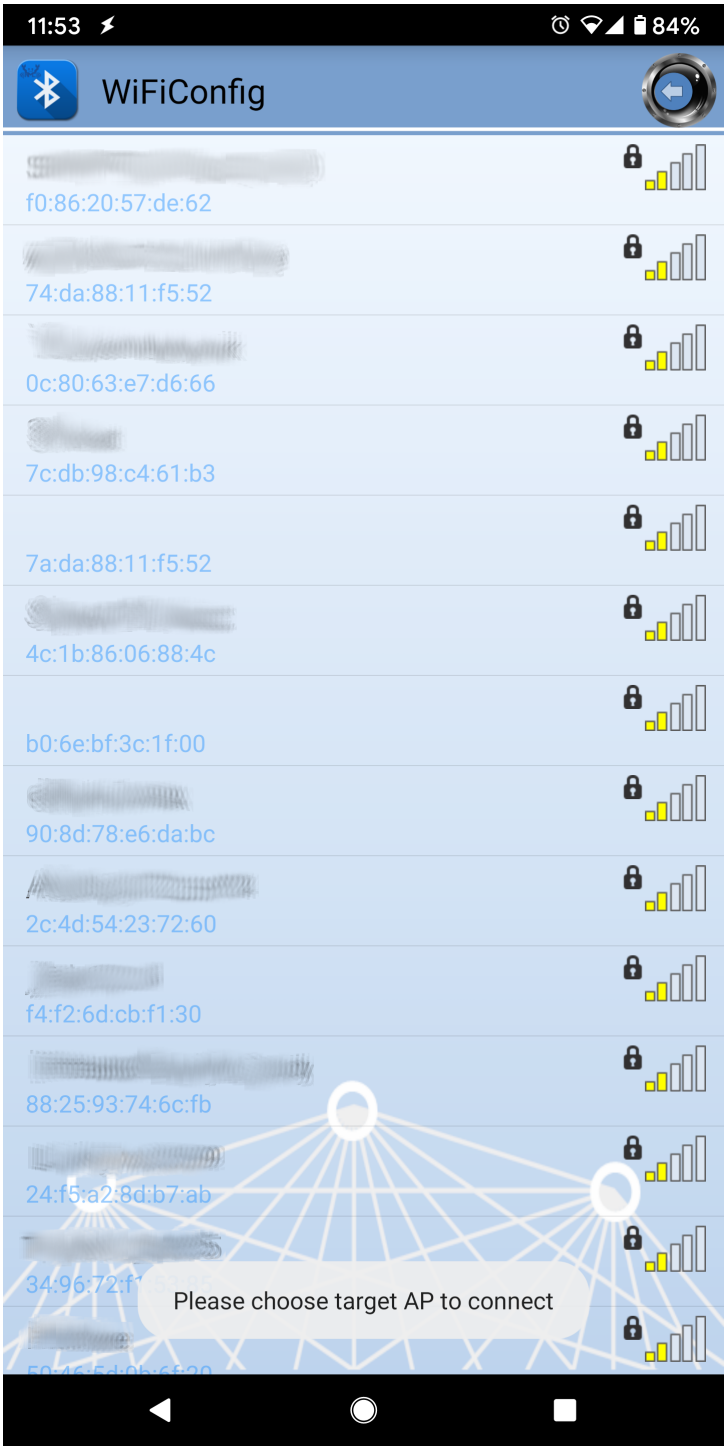
Upload the code and press the reset button on Ameba once the upload is finished.

On your mobile phone, open the Realtek WiFiConfig app and tap the round button to scan for Ameba boards.

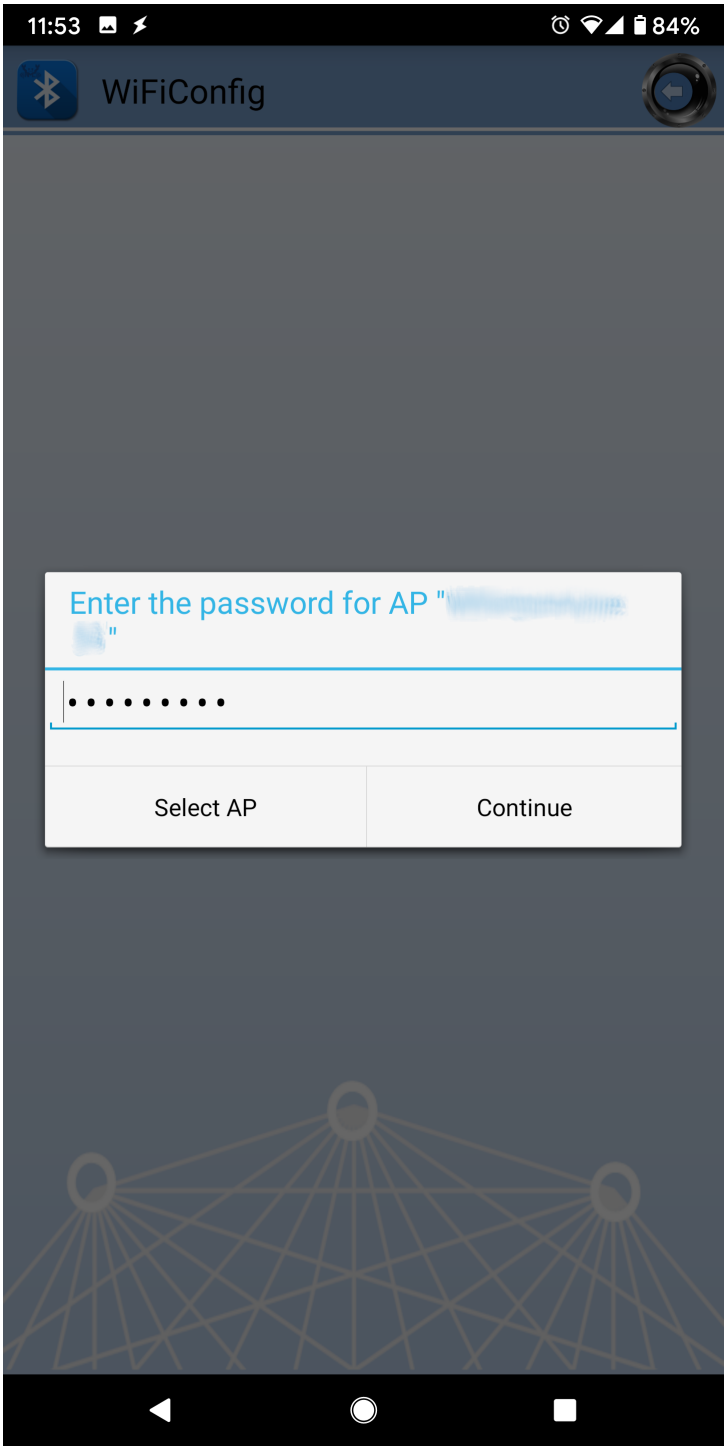
Select the correct Ameba board from the scan results. The app will connect to the Ameba board and ask the board to scan for WiFi networks and send the scan results back to the app using BLE.


4:16
82%
WiFiConfig
Choose device to configure
Ameba_0BFED2
70:1D:08:0B:FE:D2
Cancel

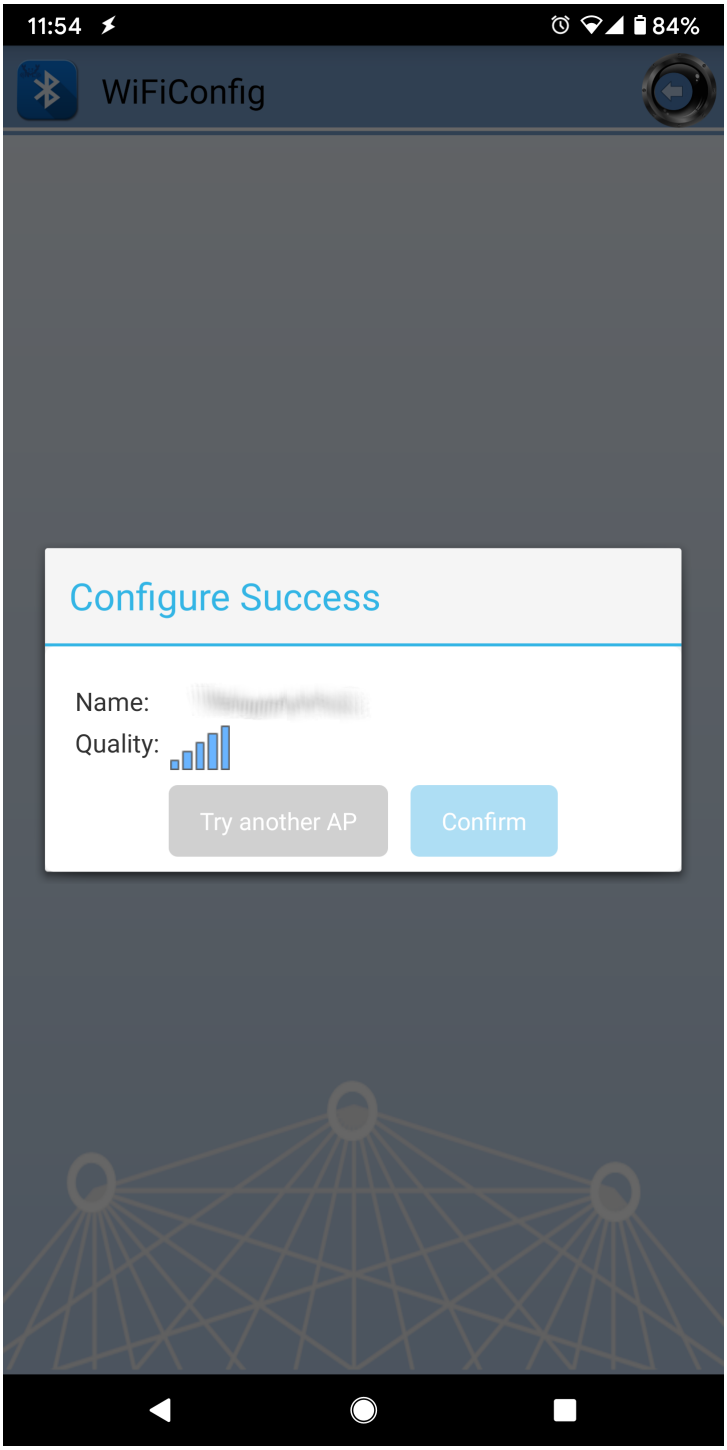

11:52
84%
WiFiConfig
Scan WiFi AP
Waiting...
Cancel



If your phone is currently connected to a WiFi network, the app will ask for the WiFi password to connect the Ameba board to the same WiFi network. Tap “Select AP” to choose another WiFi network, or enter the password and tap continue to connect Ameba to the selected WiFi network.



After the Ameba board connects to the WiFi network, the following message will be shown. Tap “Try another AP” to connect to another WiFi network or tap “Confirm” to keep the current WiFi network and disconnect BLE from the Ameba board.



Code Reference

BLEWifiConfigService is used to create an instance of the WiFi configuration service to run on the Bluetooth device.

`BLE.configAdvert()->setAdvType(configService.advData())` is used to set the correct advertisement data necessary for the phone app to find the Ameba Bluetooth device.

BLE – BLE UART Client

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

Example

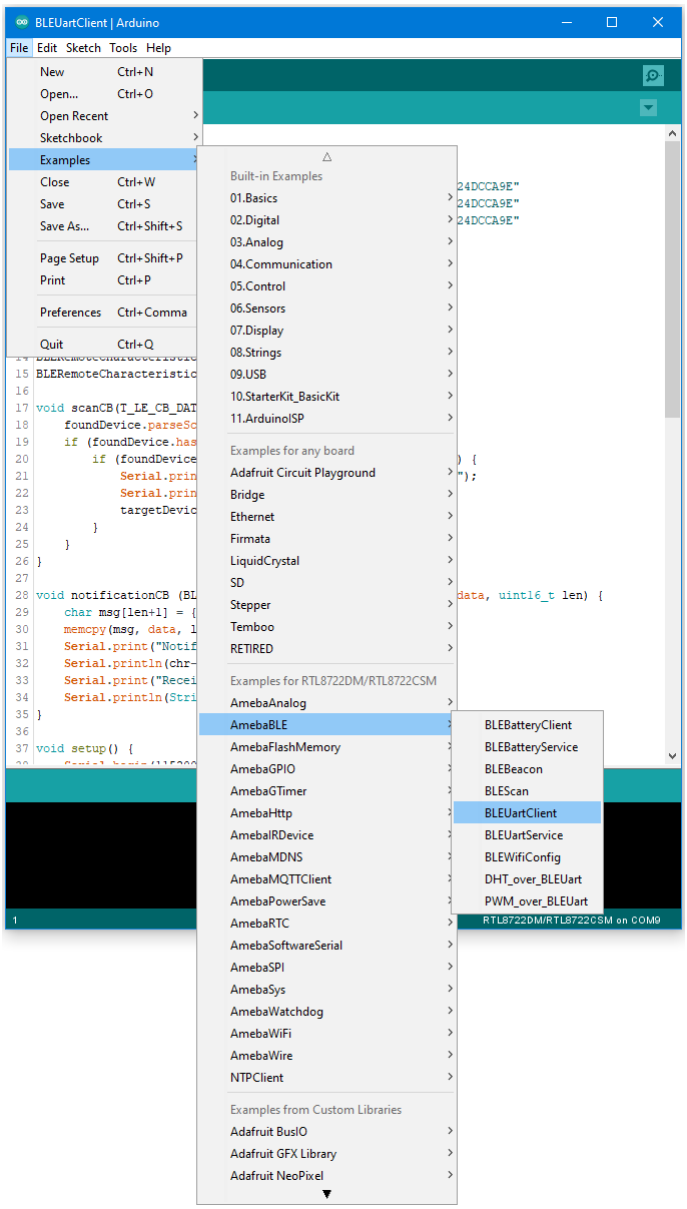
Introduction

In this example, two RTL8722 boards are connected using BLE. One board runs a BLE UART service, while the other connects to the service using a client and both boards are able to communicate with text messages over the UART service.

Procedure

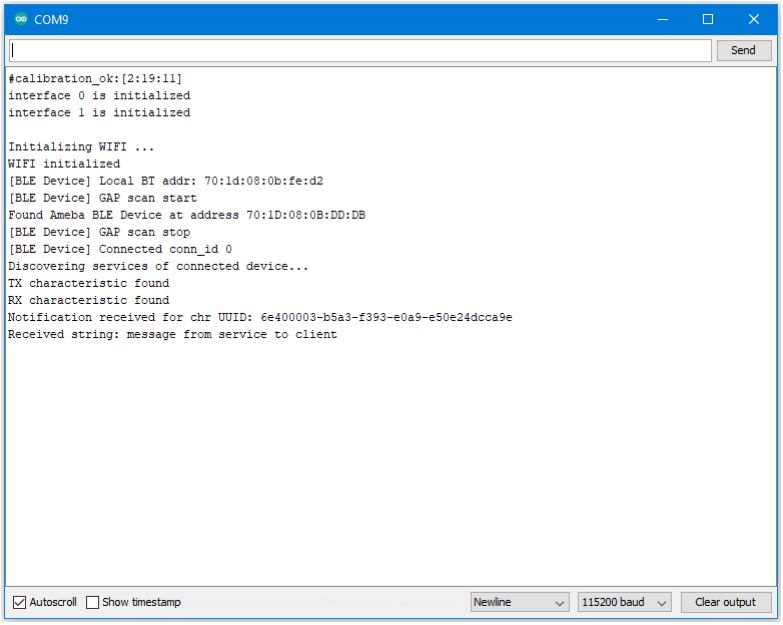
On the first board, upload the BLE UART service example code. Refer to the example guide for detailed instructions.

For the second board, open the example, "`Files" -> "Examples" -> "AmebaBLE" -> "BLEUart-Client`".



Upload the code and press the reset button on Ameba once the upload is finished.

Reset the UART service board first, wait for the BLE advertisement process to begin, and reset the UART client board. The client board should scan, discover, and connect to the service board. After connecting, the client board will verify that the correct UART service exists on the service board, before enabling notifications on the TX characteristic. Any message typed in the serial terminal will be sent to the other board using the UART service.



Code Reference

The BLEClient class is used to discover the services that exist on a connected BLE device. The discovery process will create BLERemoteService, BLERemoteCharacteristic and BLERemoteDescriptor objects corresponding to the services, characteristics and descriptors that exist on the connected device. These objects can then be used to read and write data to the connected device.

BLE – BLE UART Service

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Android / iOS smartphone

Example

Introduction

With BLE, application data is sent and received using the GATT system. GATT uses services, characteristics, and attributes to organise data and control how the data can be read from and written to. The Bluetooth SIG specification for BLE includes several predefined services for common applications, but users are free to implement custom services and characteristics to best fit their data structure and application needs

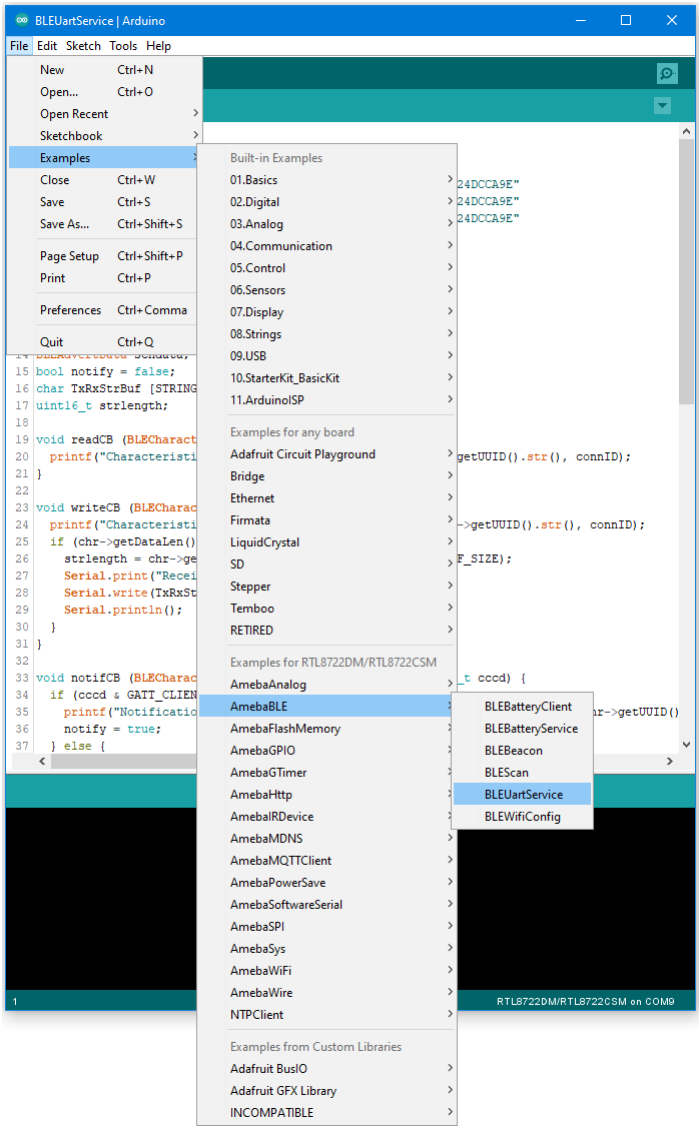
In this example, the BLEService and BLECharacteristic classes are used to implement a custom service for transmitting ASCII characters similar to regular UART. This custom service is the Nordic UART Service, which is supported in several smartphone apps.

Procedure

Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

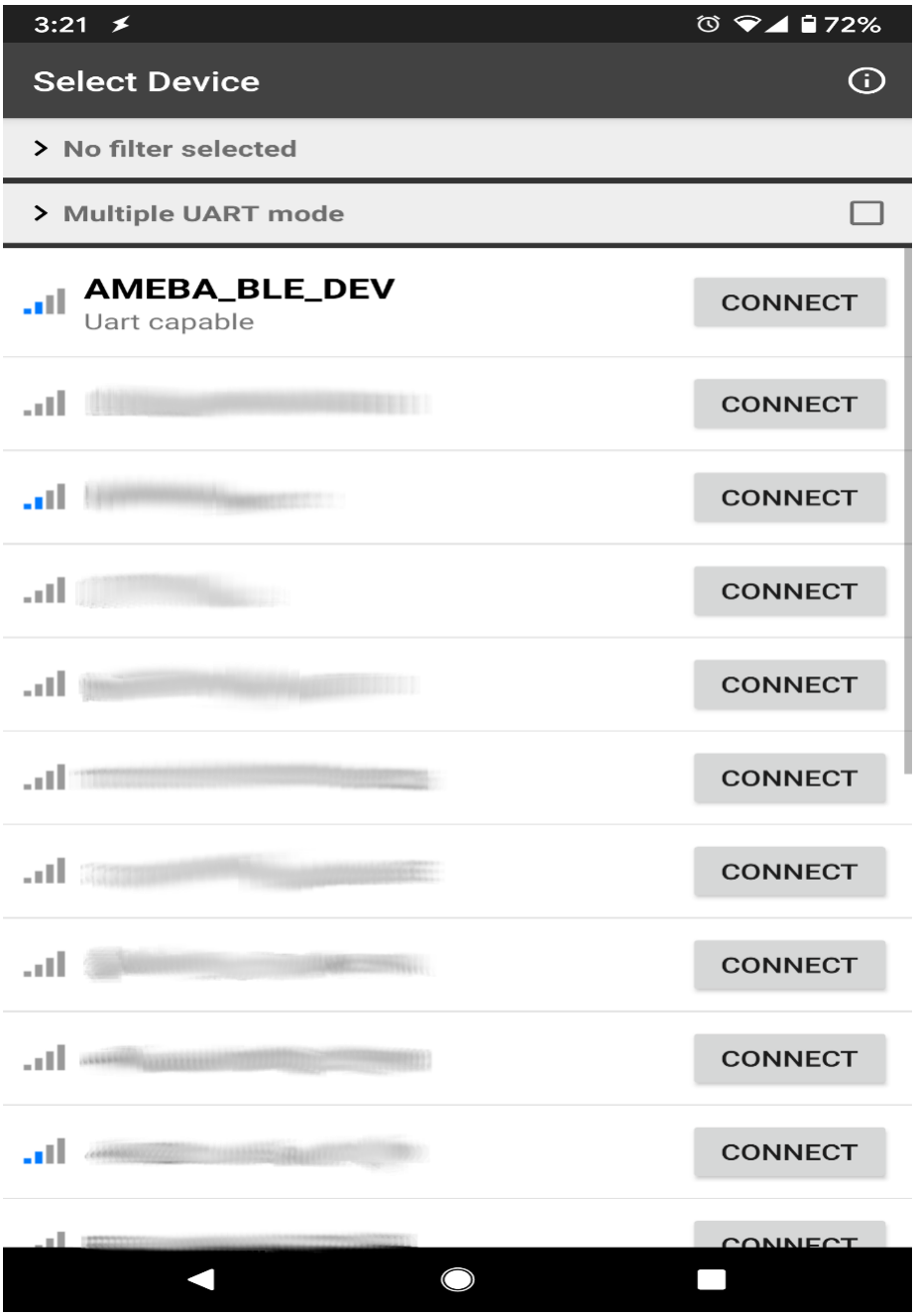
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

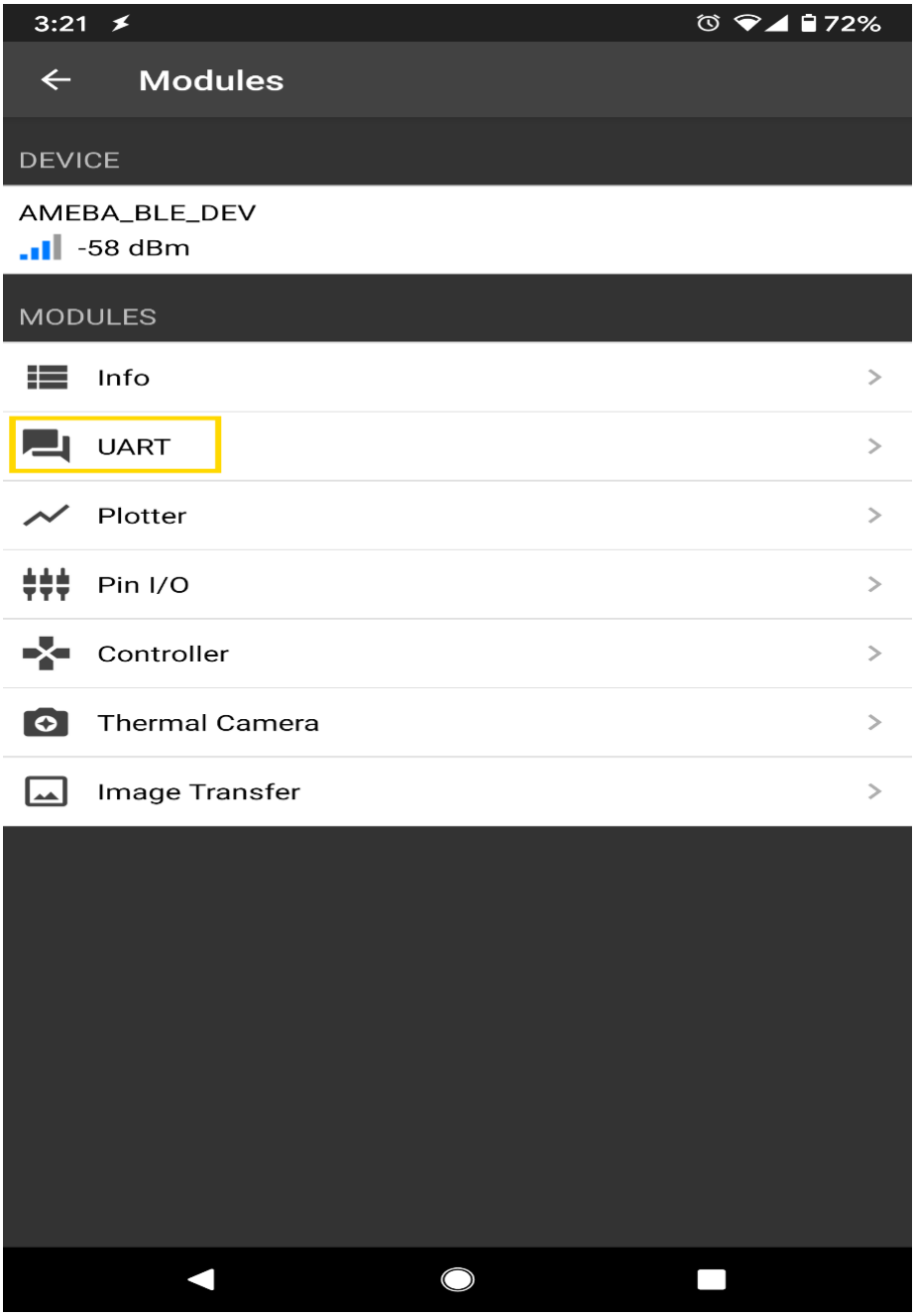
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`BLEUartService`”.



Upload the code and press the reset button on Ameba once the upload is finished.
Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app. Note that the BLE UART service on the Ameba board will only work with the UART and Plotter functions in the Bluefruit Connect app, other functions (Pin I/O, Image Transfer) require other BLE services

that are not included in this example.

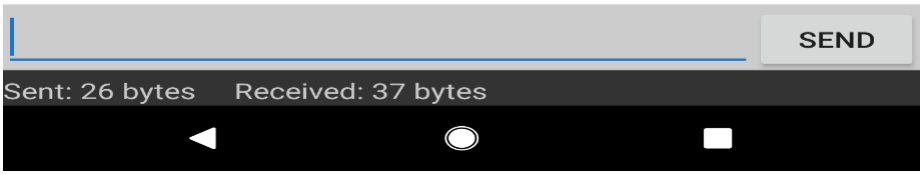


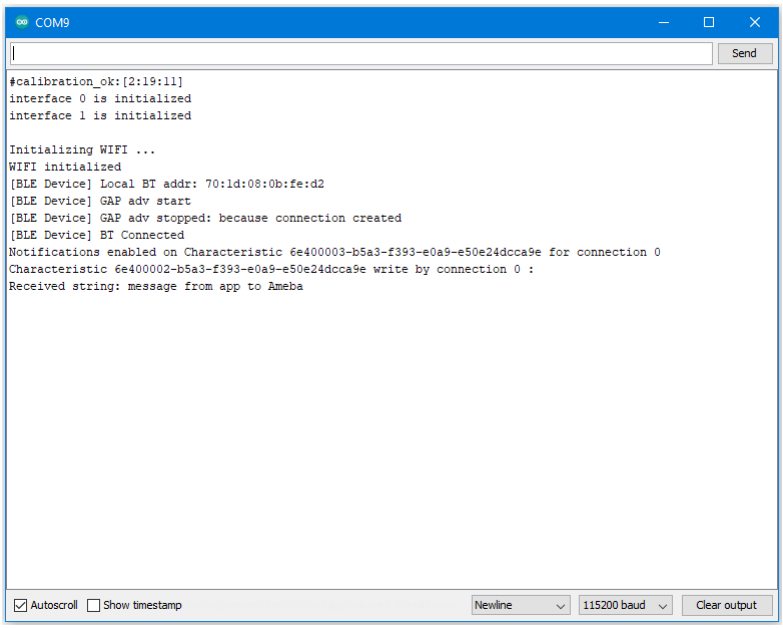


In the UART terminal section of the app, enter a message and click send. You should see the message appear in the Arduino serial monitor.

In the Arduino serial monitor, enter a message and click send. The message will appear in the smartphone app.


3:22
72%
UART
MQTT
message from app to Ameba
message from Ameba to smartphone app


SEND
Sent: 26 bytes
Received: 37 bytes



Code Reference

The BLECharacteristic class is used to create two characteristics, one for receive (Rx) and one for transmit (Tx), and added to a service created with the BLEService class.

The required read/write/notify properties are set for each characteristic using the `set__Property()` methods, and callback functions are registered using the `set__Callback()` methods. The required buffer size is also set for each characteristic so that it has enough memory to store a complete string.

When data is written to the receive characteristic, the registered callback function is called, which prints out the received data as a string to the serial monitor.

When data is received on the serial port, it is copied into the transmit characteristic buffer, and the `notify()` method is used to inform the connected device of the new data.

BLE – DHT over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21
- Android / iOS smartphone

Example

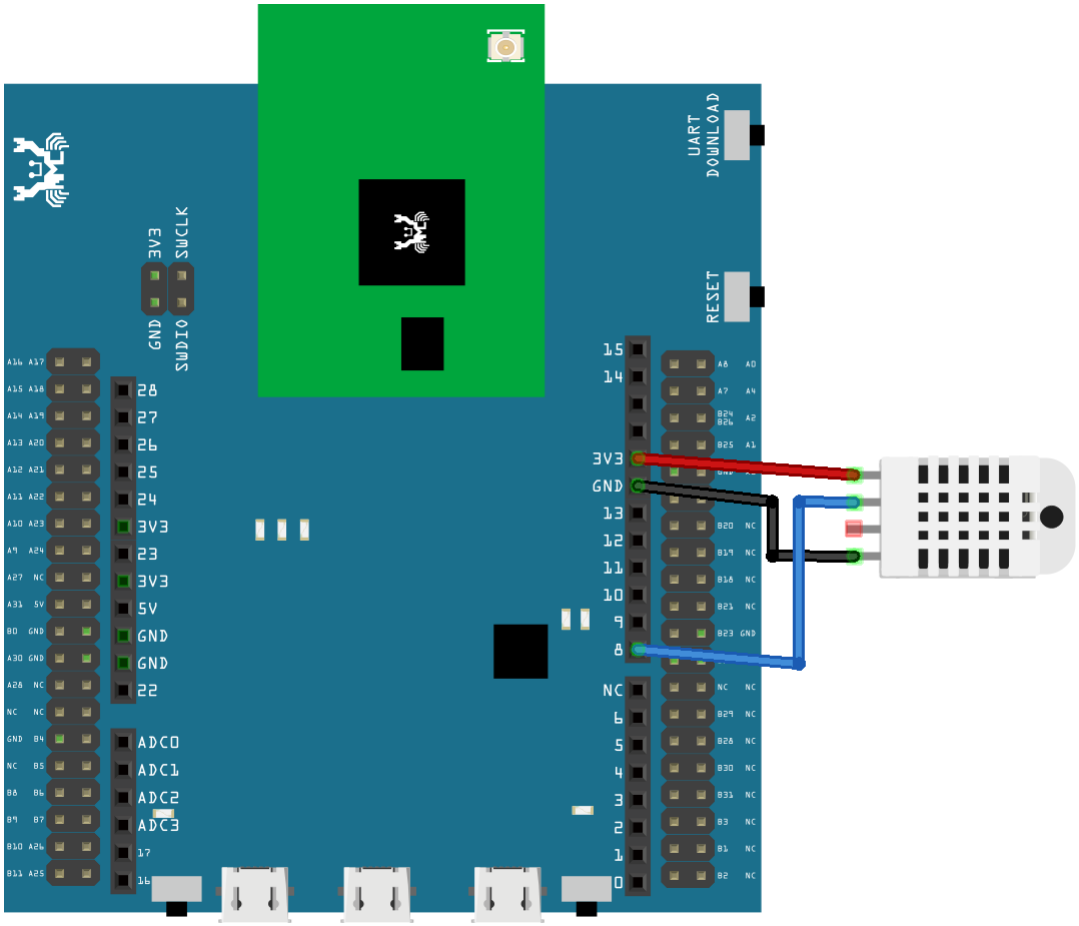
Introduction

In this example, the data obtained from a DHT temperature and humidity sensor are transmitted over a BLE UART service to a smartphone. Refer to the other examples for detailed explanations of using the DHT sensor and the BLE UART service.

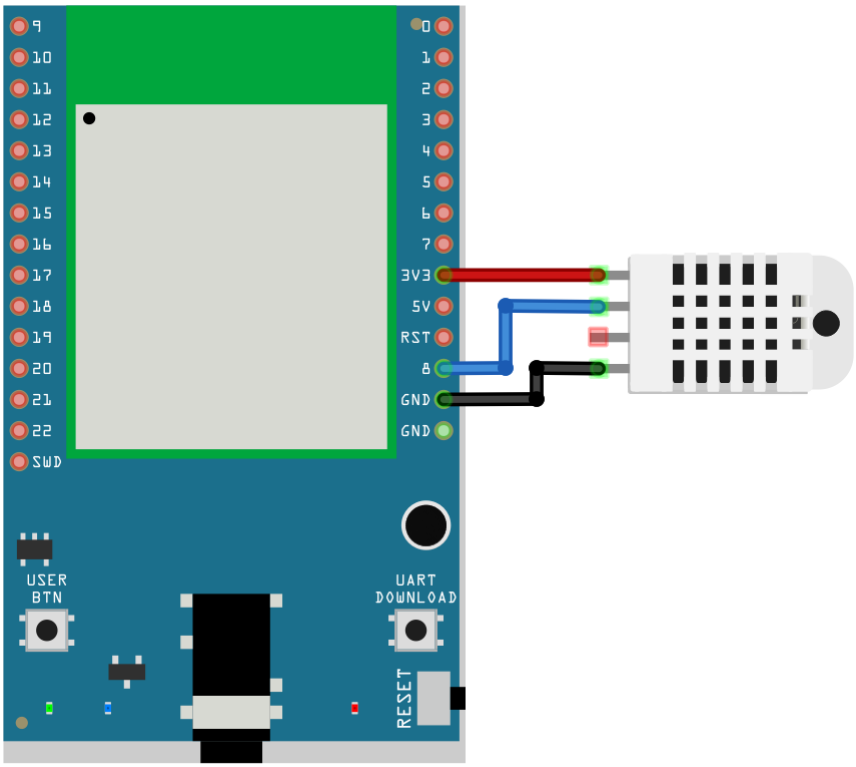
Procedure

Connect the DHT sensor to the Ameba board following the diagram.

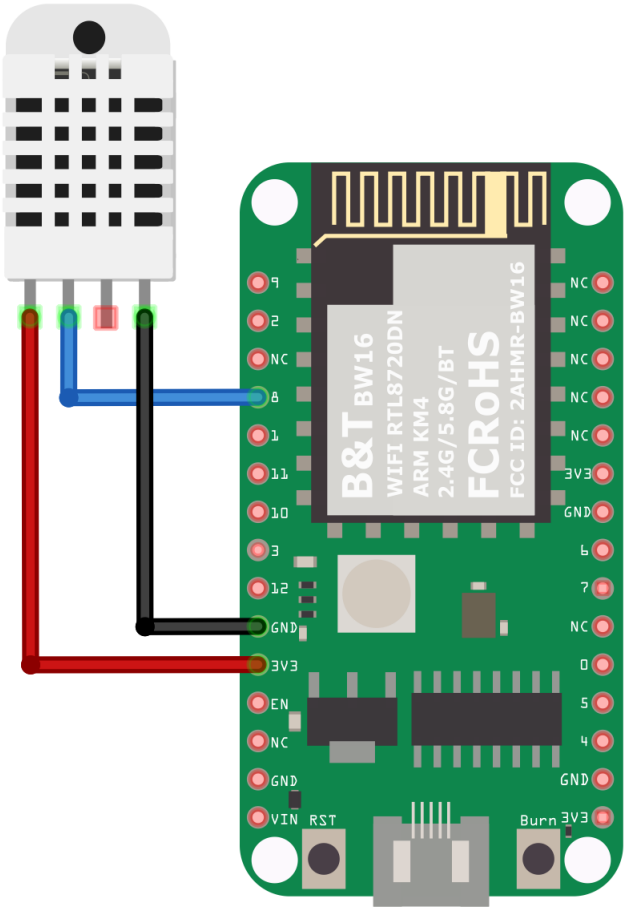
AMB21 / AMB22:



AMB23:


9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

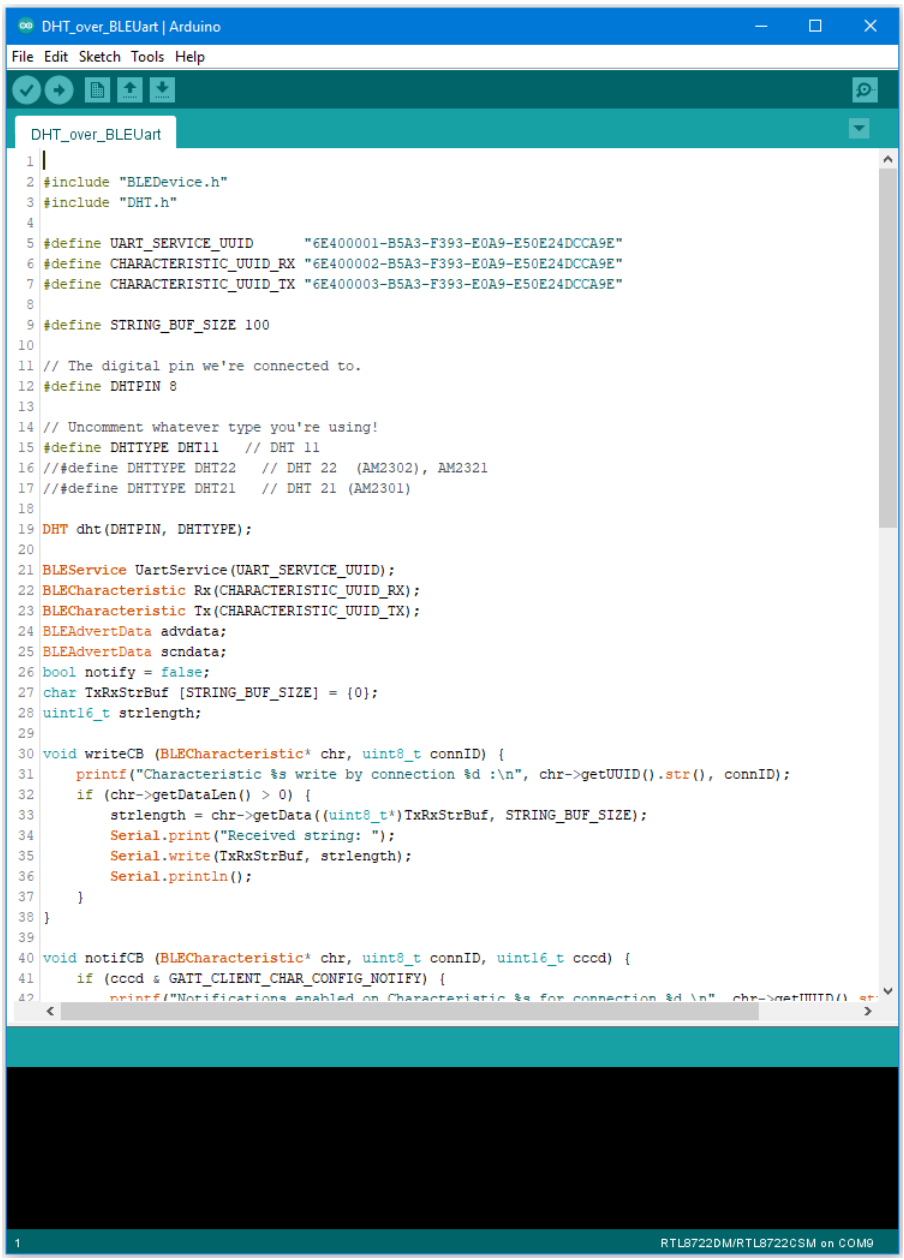
BW16:



Ensure that a compatible BLE UART app is installed on your smartphone, it is available at:
- Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connecta>https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

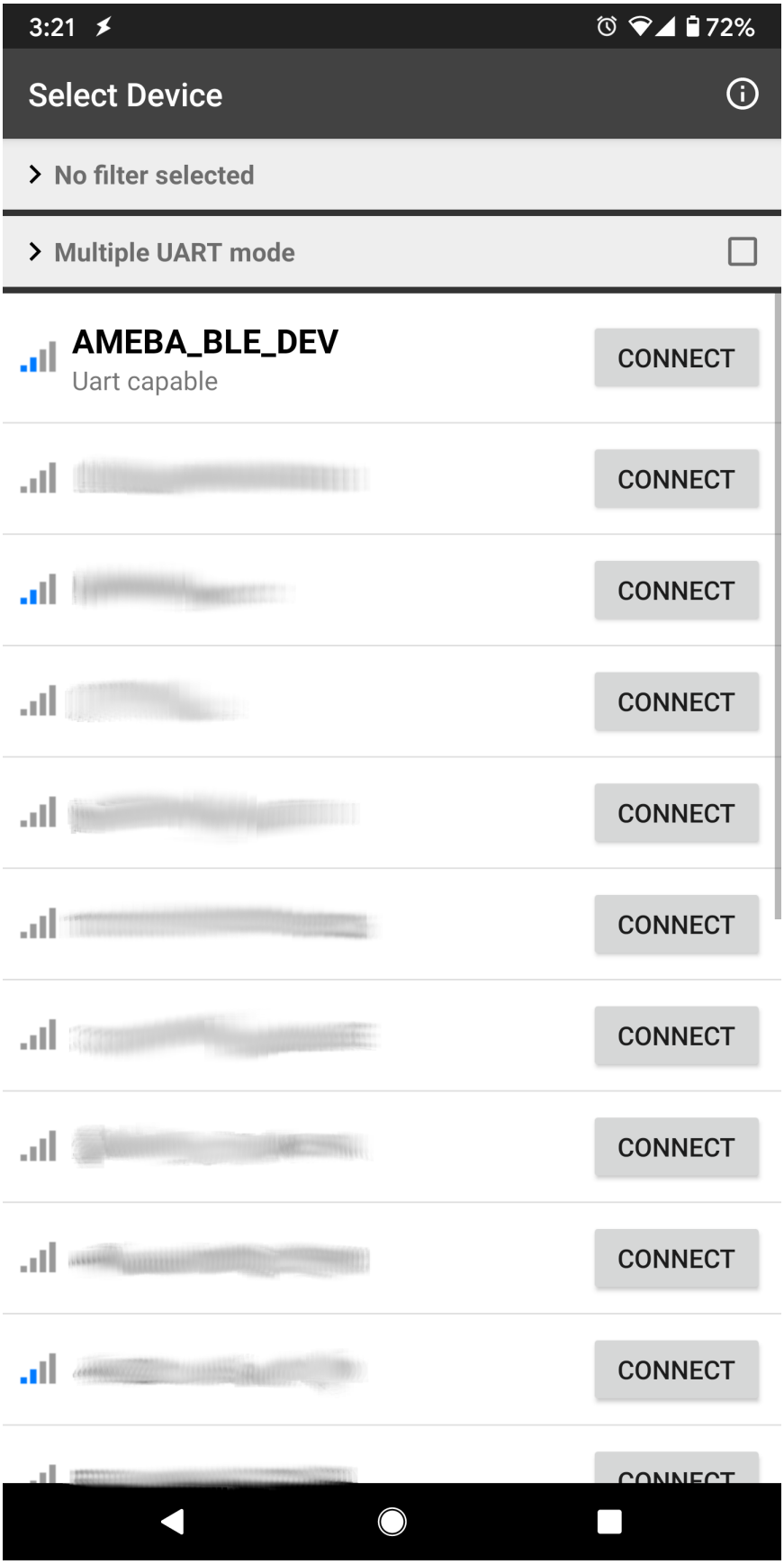
- Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

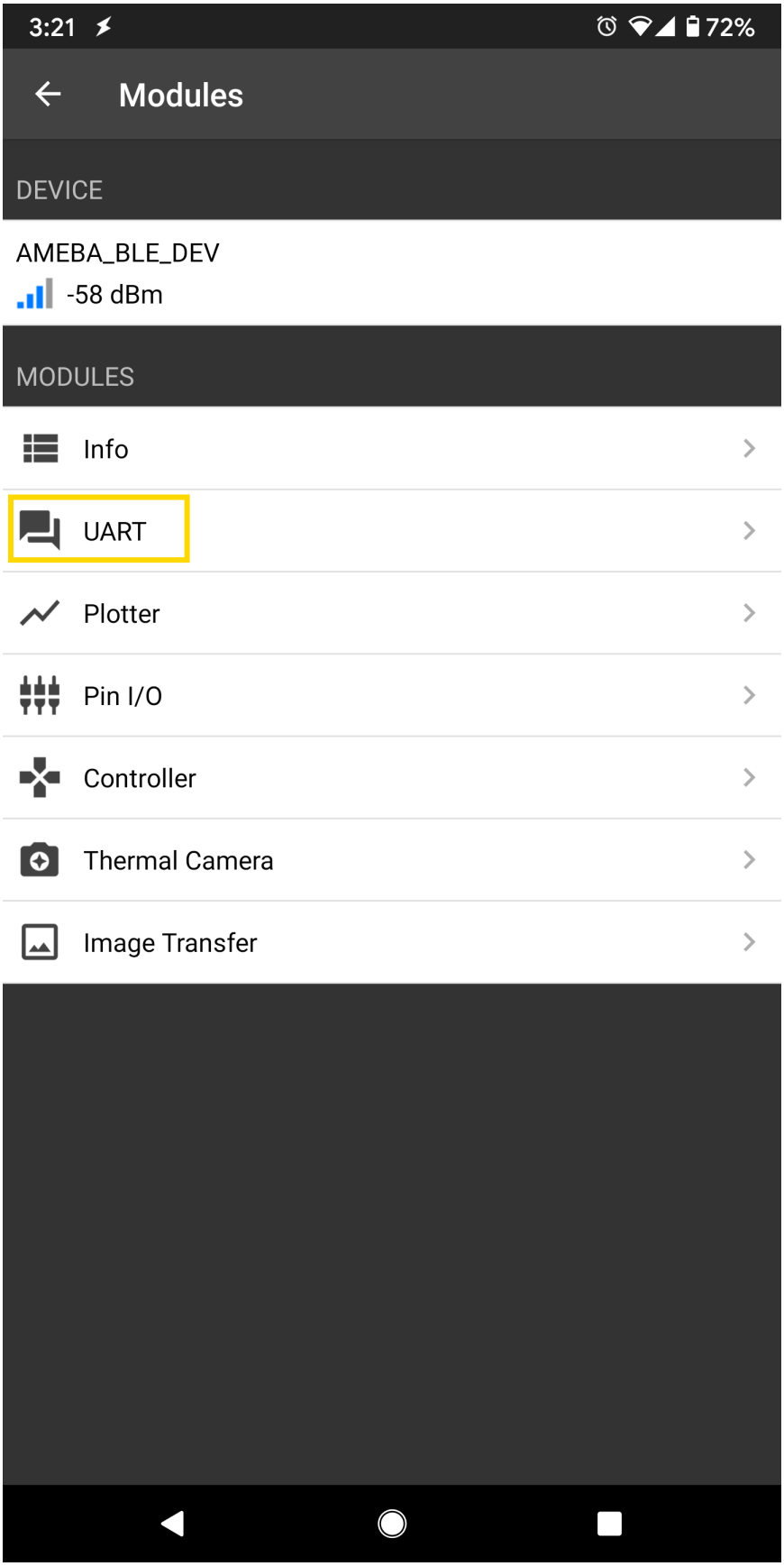
Open the example, “`Files`” -> “`Examples`” -> “`AmebaBLE`” -> “`DHT_over_BLEUart`”.



Upload the code and press the reset button on Ameba once the upload is finished.

Open the app on your smartphone, scan and connect to the Ameba board shown as “AMEBA_BLE_DEV” and choose the UART function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer

After starting the UART function, notifications should be received every 5 seconds containing the measured temperature and humidity.


4:20
81%
UART
MQTT
Humidity: 59% Temperature: 27°C
Humidity: 59% Temperature: 27°C
Humidity: 62% Temperature: 27°C
Humidity: 67% Temperature: 27°C
Humidity: 69% Temperature: 27°C
Humidity: 88% Temperature: 27°C
SEND
Sent: 0 bytes
Received: 198 bytes

BLE – PWM over BLE UART

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- RGB LED
- Android / iOS smartphone

Example

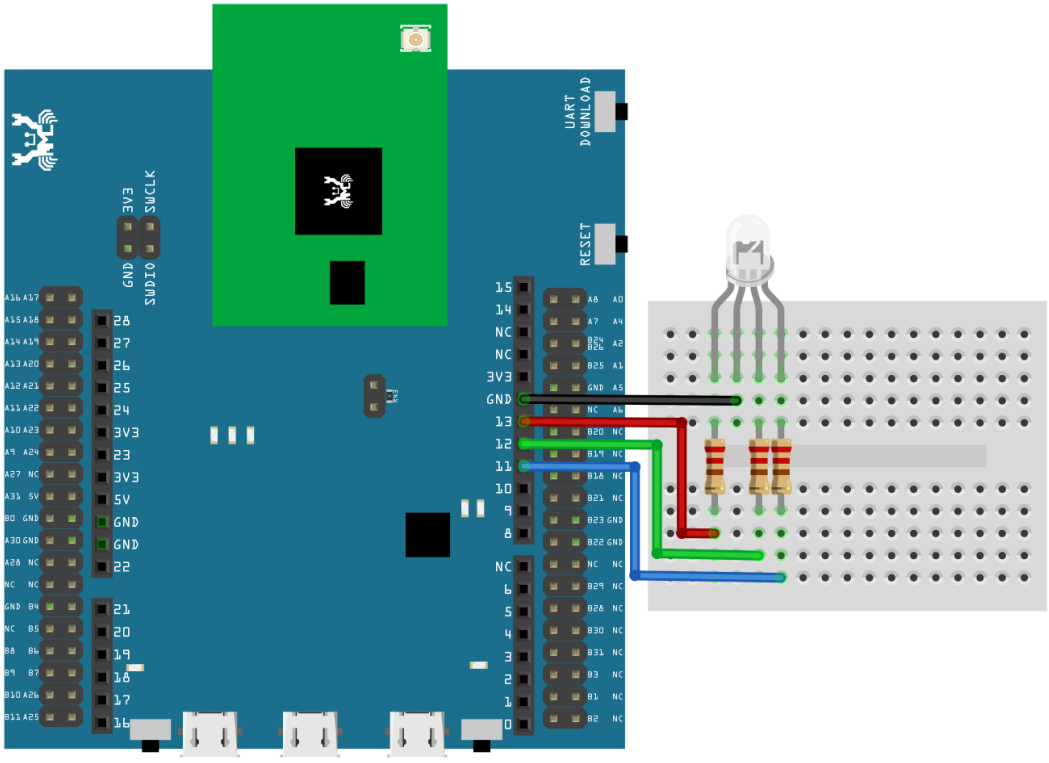
Introduction

In this example, a smartphone app is used to transmit commands over BLE UART to control the PWM outputs and change the color of a RGB LED. Refer to the other example guides for detailed explanations of the BLE UART service.

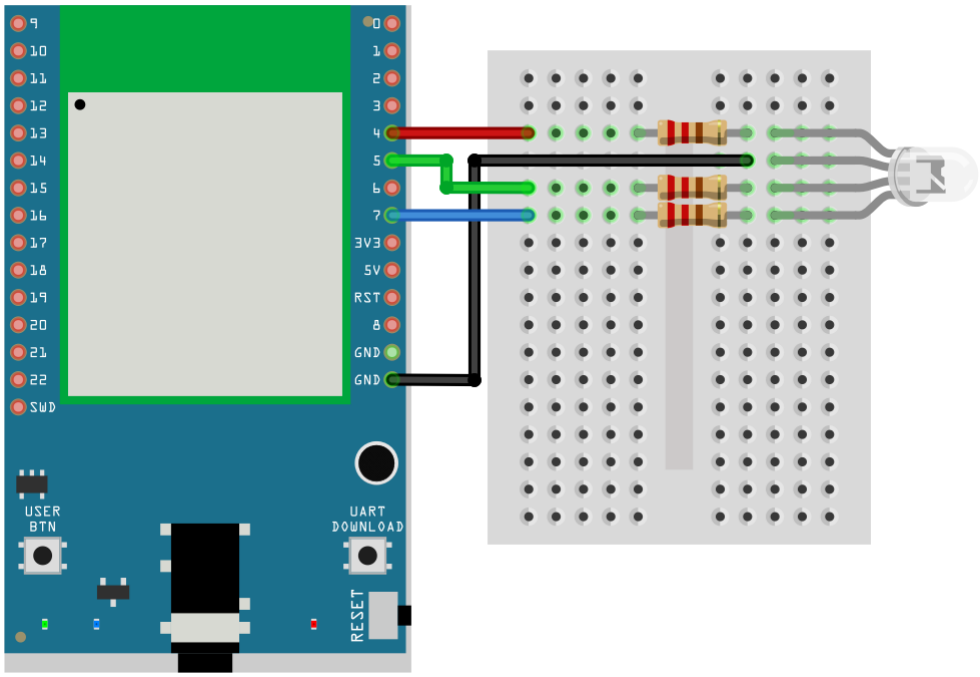
Procedure

Connect the RGB LED to the RTL8722 board following the diagram, the common LED pin may need to connect to 3.3V or GND depending on the type of LED (common anode / common cathode).

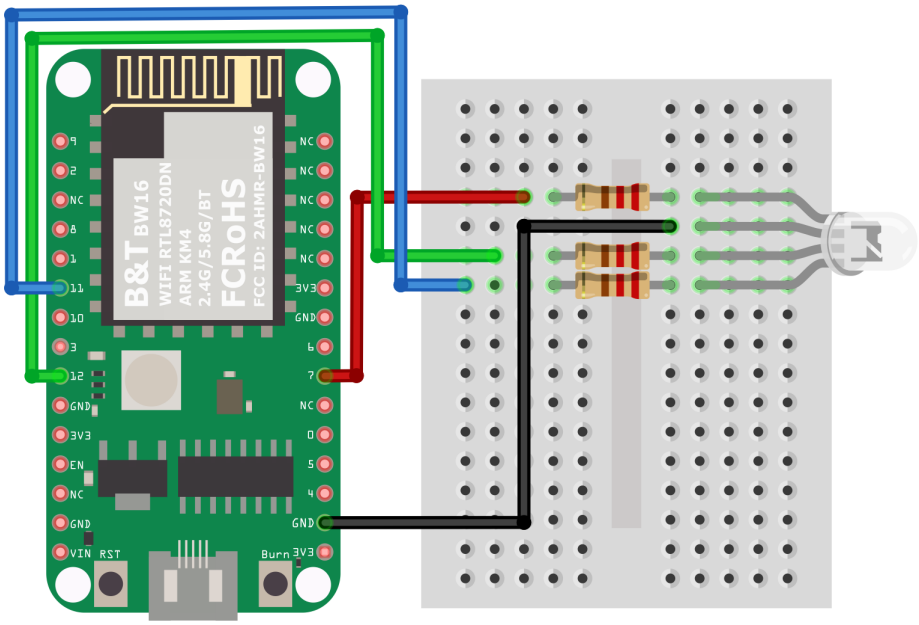
AMB21 /AMB22:



AMB23:



BW16:

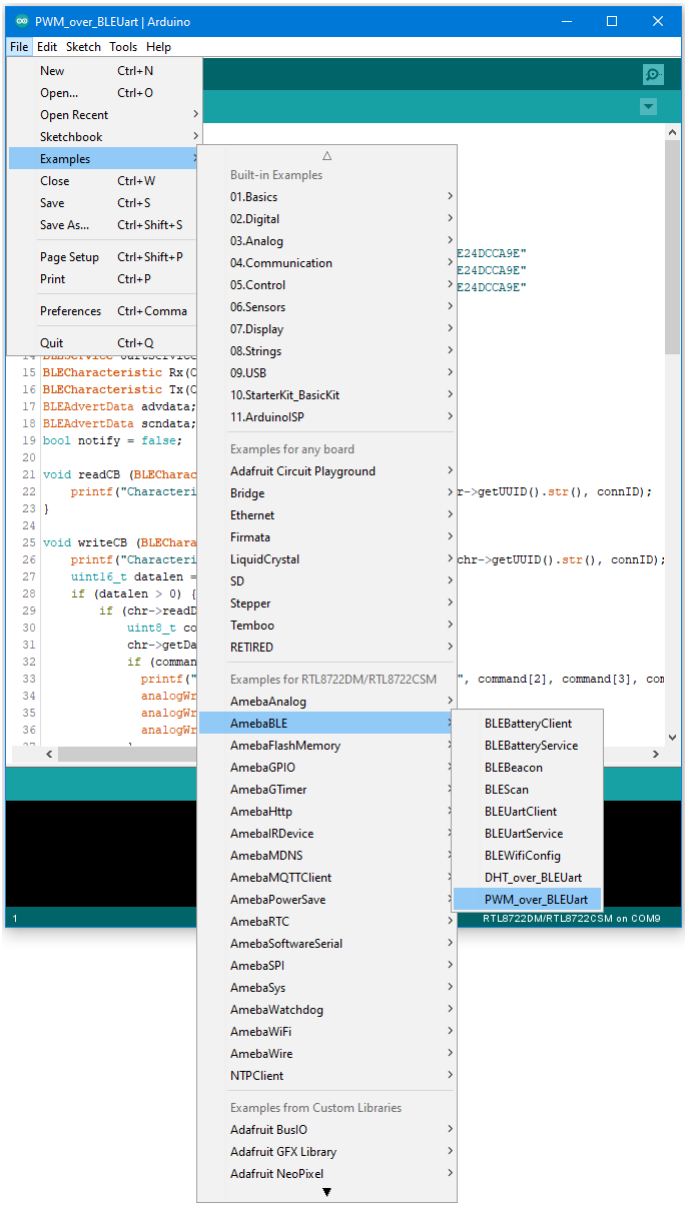


Ensure that the required app is installed on your smartphone, it is available at:
– Google Play Store:
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect

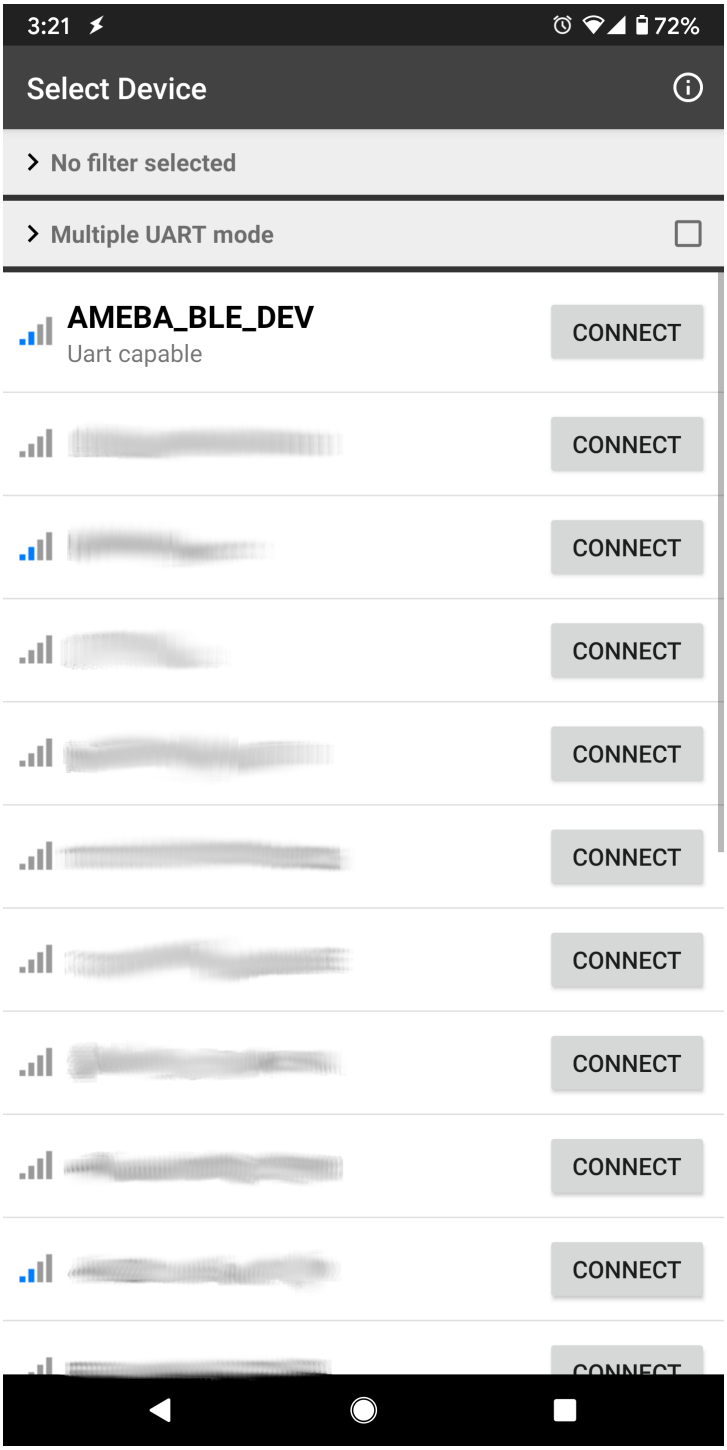
– Apple App Store:
https://apps.apple.com/us/app/bluefruit-connect/id830125974

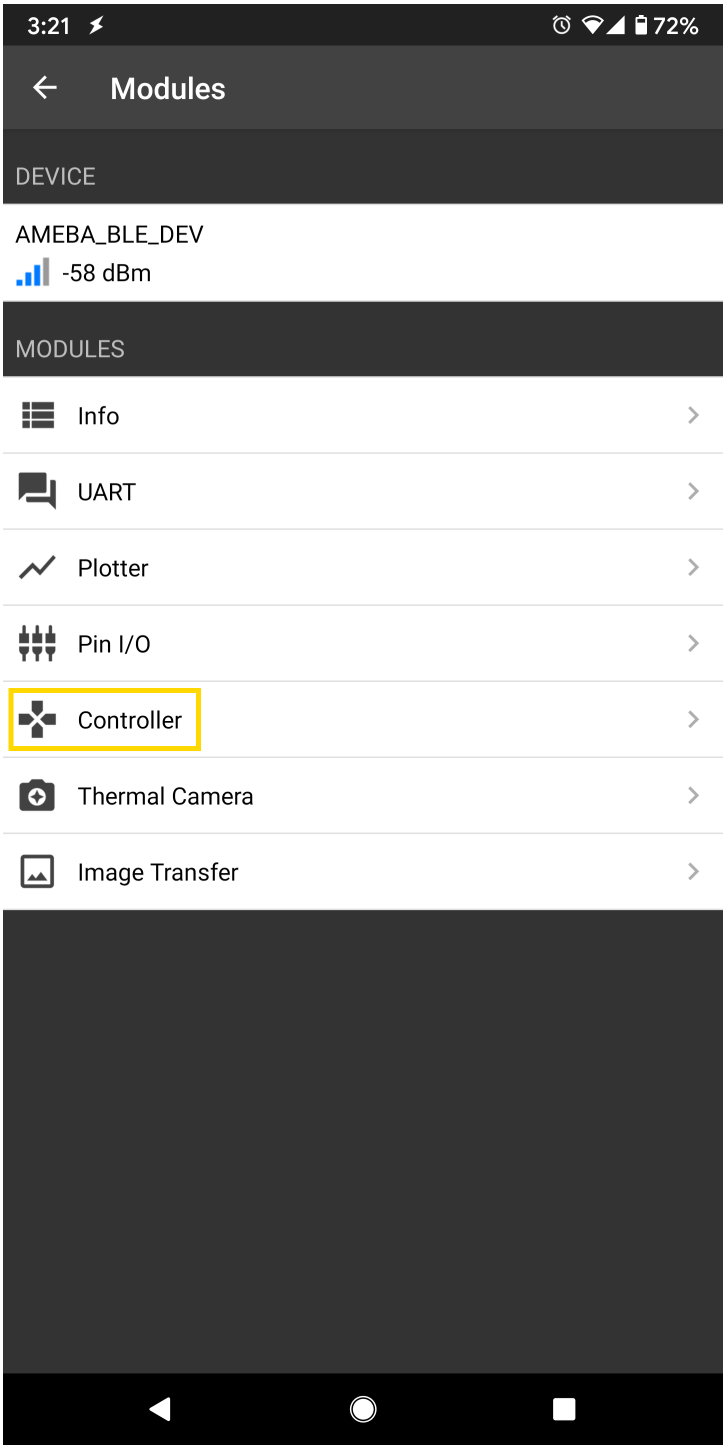
Open the example, “`Files` -> “`Examples`” -> “`AmebaBLE`” -> “`PWM_over_BLEUart`”.

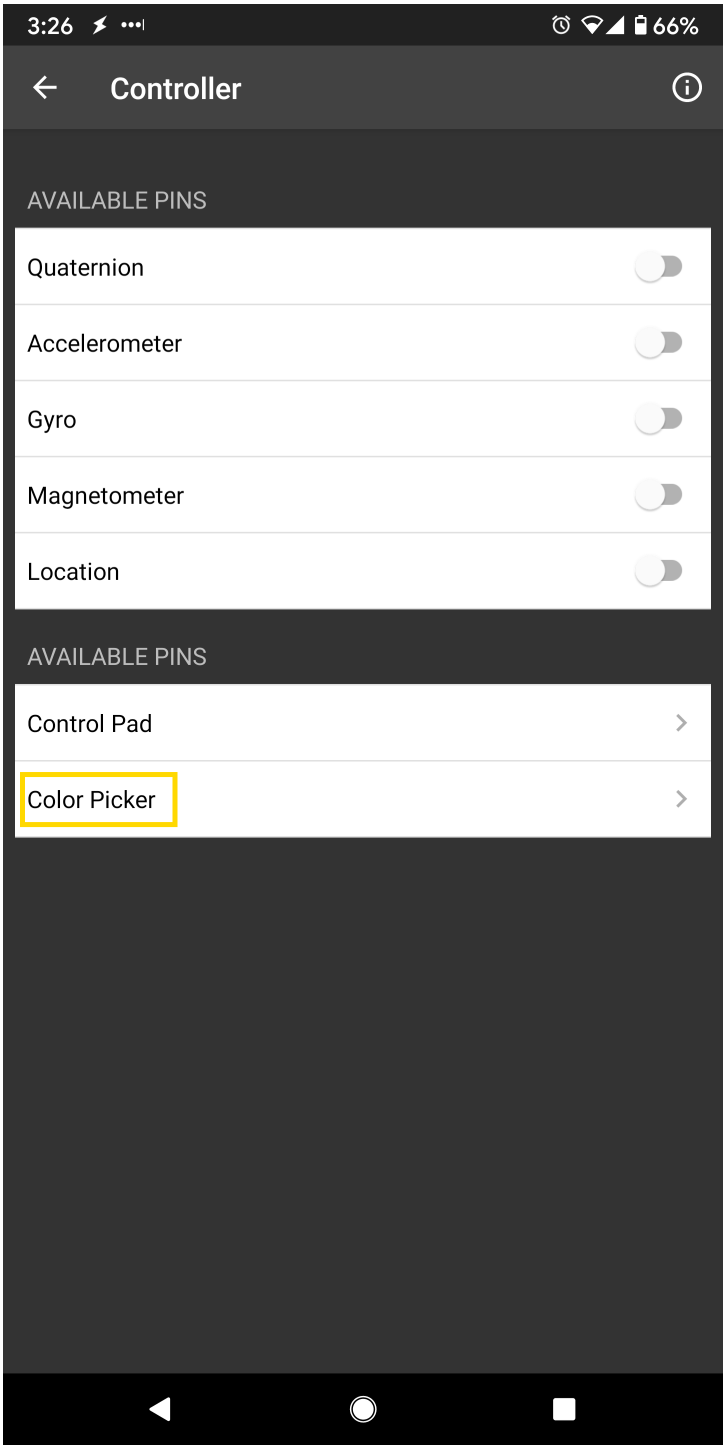
Upload the code and press the reset button on Ameba once the upload is finished.



Open the app on your smartphone, scan and connect to the board shown as “AMEBA_BLE_DEV” and choose the controller -> color picker function in the app.


3:21
72%
Select Device
No filter selected
Multiple UART mode
AMEBA_BLE_DEV
Uart capable
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT
CONNECT


3:21
72%
Modules
DEVICE
AMEBA_BLE_DEV
-58 dBm
MODULES
Info
UART
Plotter
Pin I/O
Controller
Thermal Camera
Image Transfer



Using the color selection wheel, saturation, and brightness sliders, choose a desired color and click select to send the RGB values to the board. You should see the RGB LED change to the matching color.

Code Reference

The RGB values are sent as three consecutive bytes prefixed by "!C" characters. The "!" exclamation mark is used to indicate that the following data is a command, and the "C" character is used to indicate that the data is RGB values. The received UART message is checked in the callback function for "!C" first, otherwise it is treated as a regular message and printed to the serial terminal.

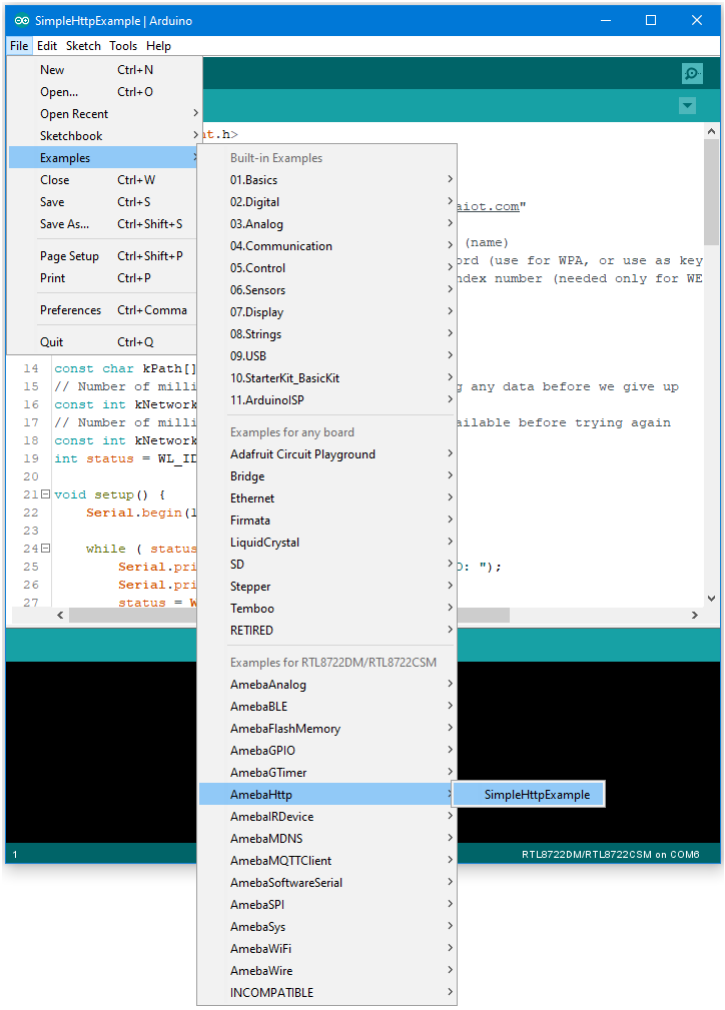
HTTP - Retrieve HTTP websites from the Internet

Materials

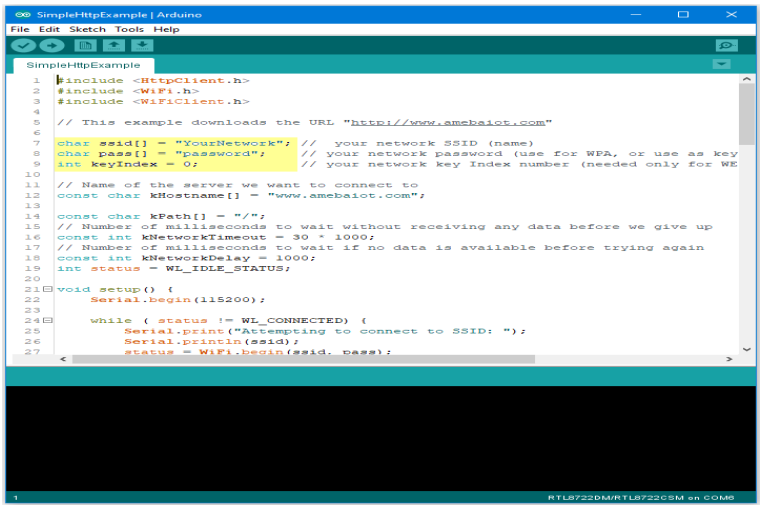
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

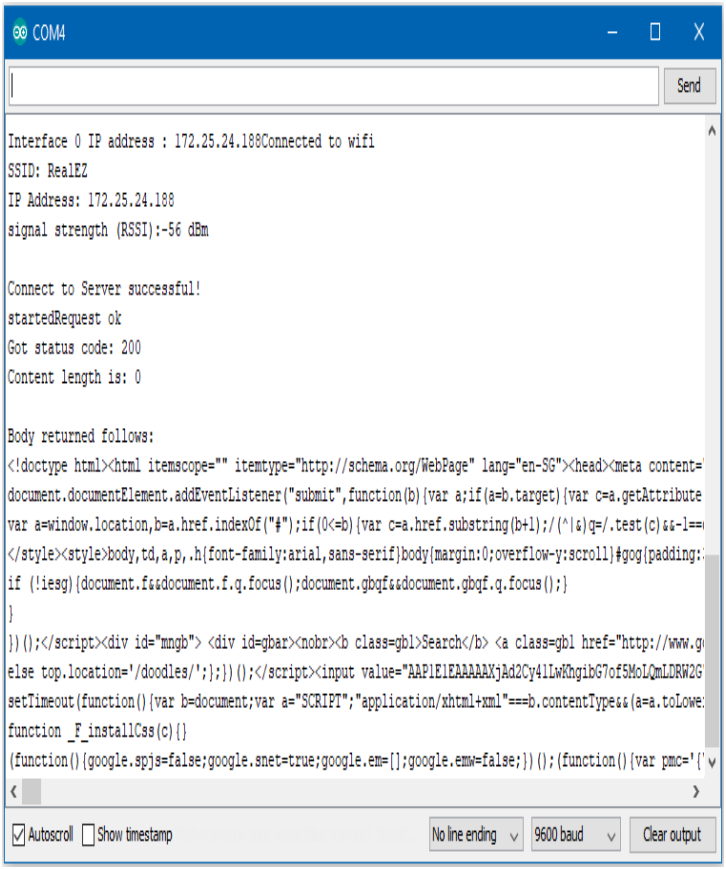
In this example, the HttpClient library is used to retrieve a webpage using the HTTP protocol.
First, make sure that the correct Ameba development board is selected in “Tools” -> “Board”
Then open “`File`” -> “`Examples`” -> “`AmebaHttp`” -> “`SimpleHttpExample`”



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in the Arduino IDE and you can see the information retrieved from the website.



Code Reference

Use `WiFi.begin()` to establish WiFi connection:
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use WiFiClient to create a client to handle the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiClient
Use HTTPClient to create a client to handle the HTTP connection.

Use `http.get()` to send a GET request to the website.

HTTP - Set up Server to Control LED

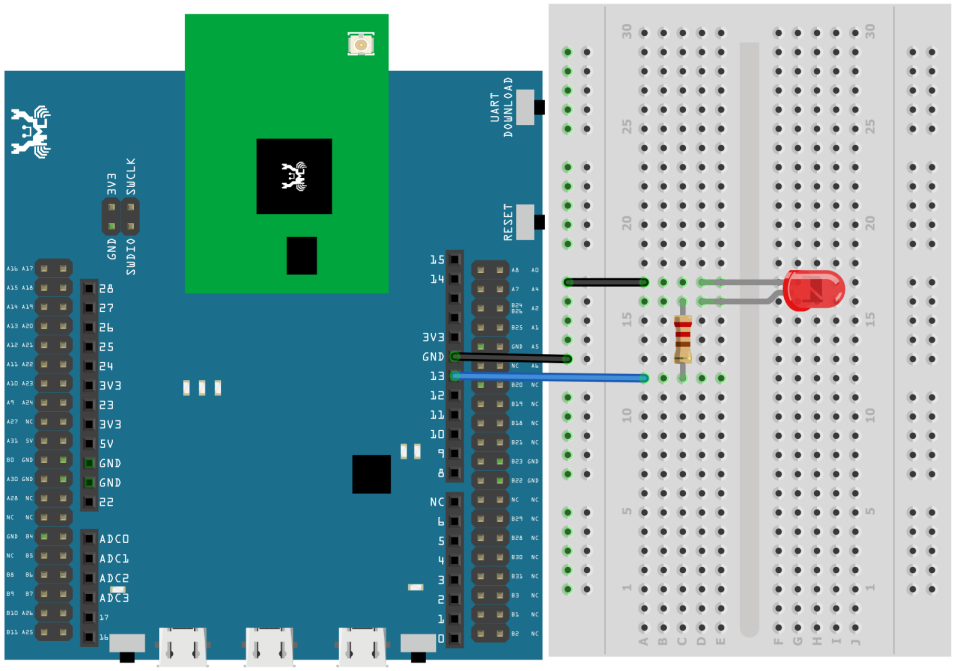
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Breadboard x 1
- LED x 1
- 1KΩ Resistor x 1

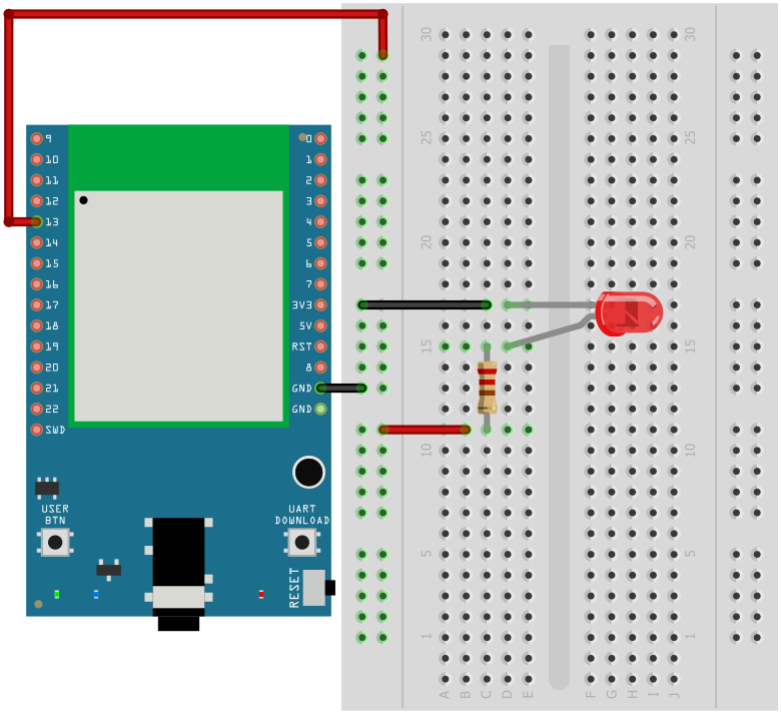
Procedure

In this example, we connect Ameba to WiFi and use Ameba as server, the user can control the LED on/off through a webpage.
First, connect Ameba with the LED.
In a LED, the longer pin is the positive pole, and the shorter pin is the negative pole. So, we connect the shorter pin to GND and connect the longer pin to D13. Additionally, to avoid the electric current exceeds the tolerance of the LED and causes damage, we connect a resistance on the positive pole.

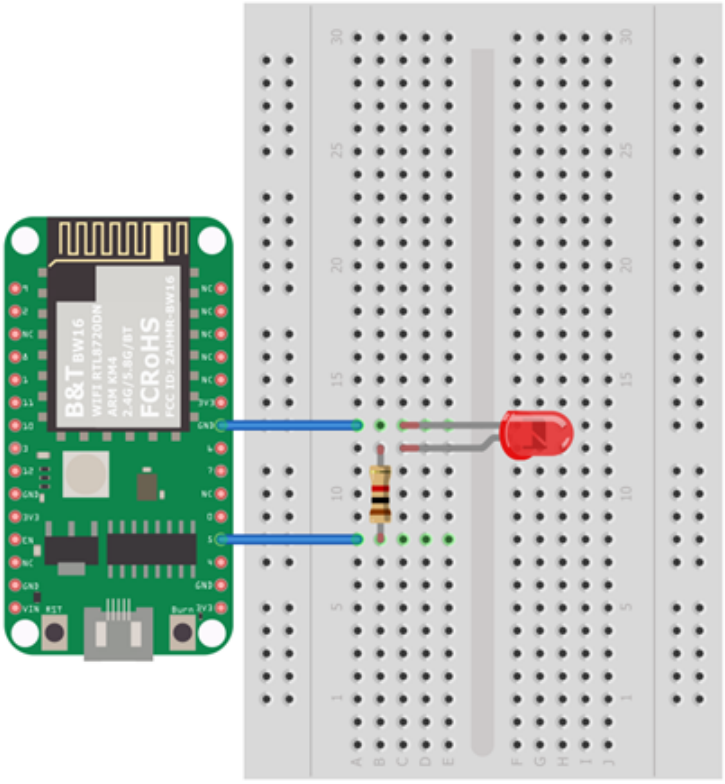
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

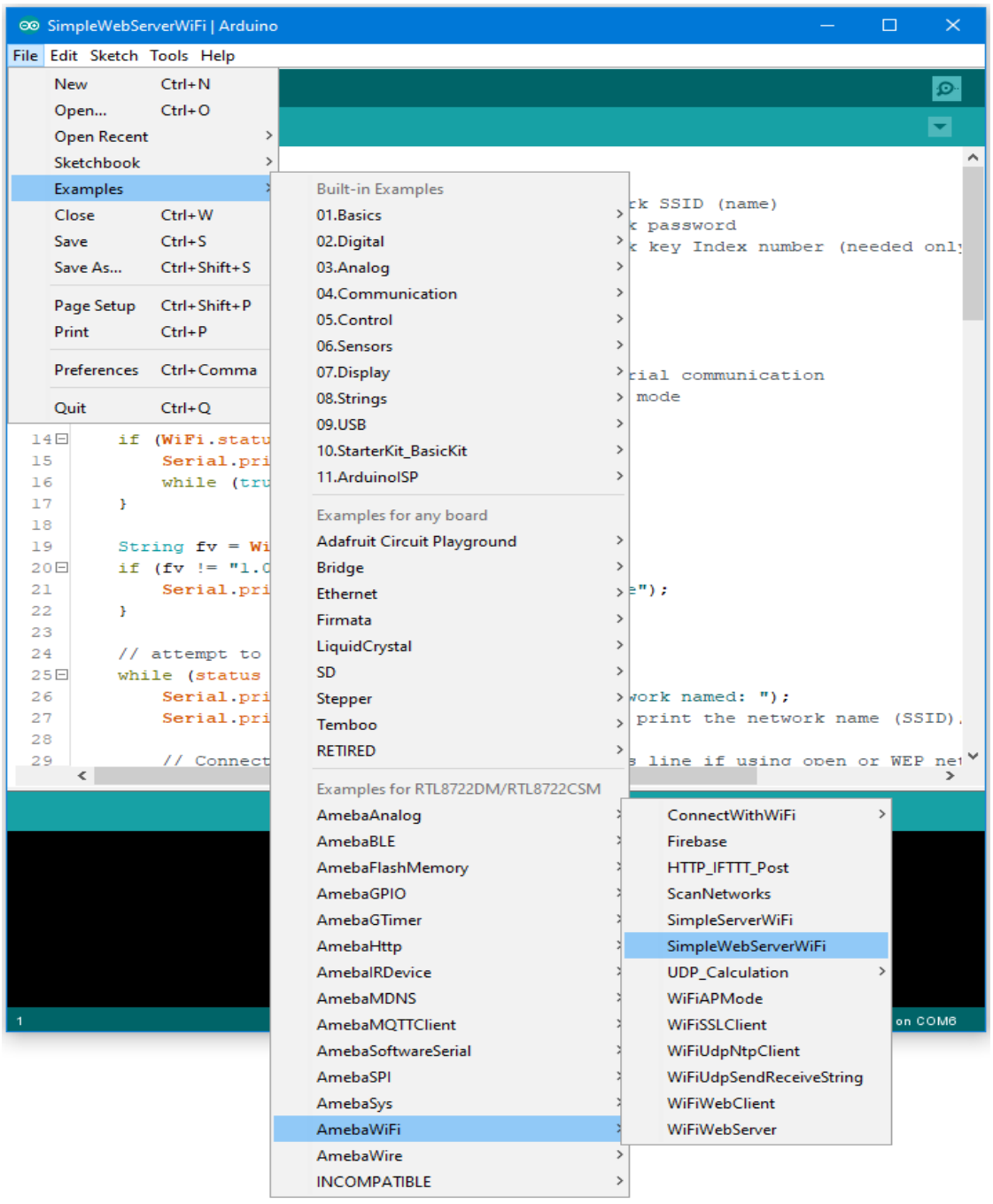


BW16 Wiring Diagram:

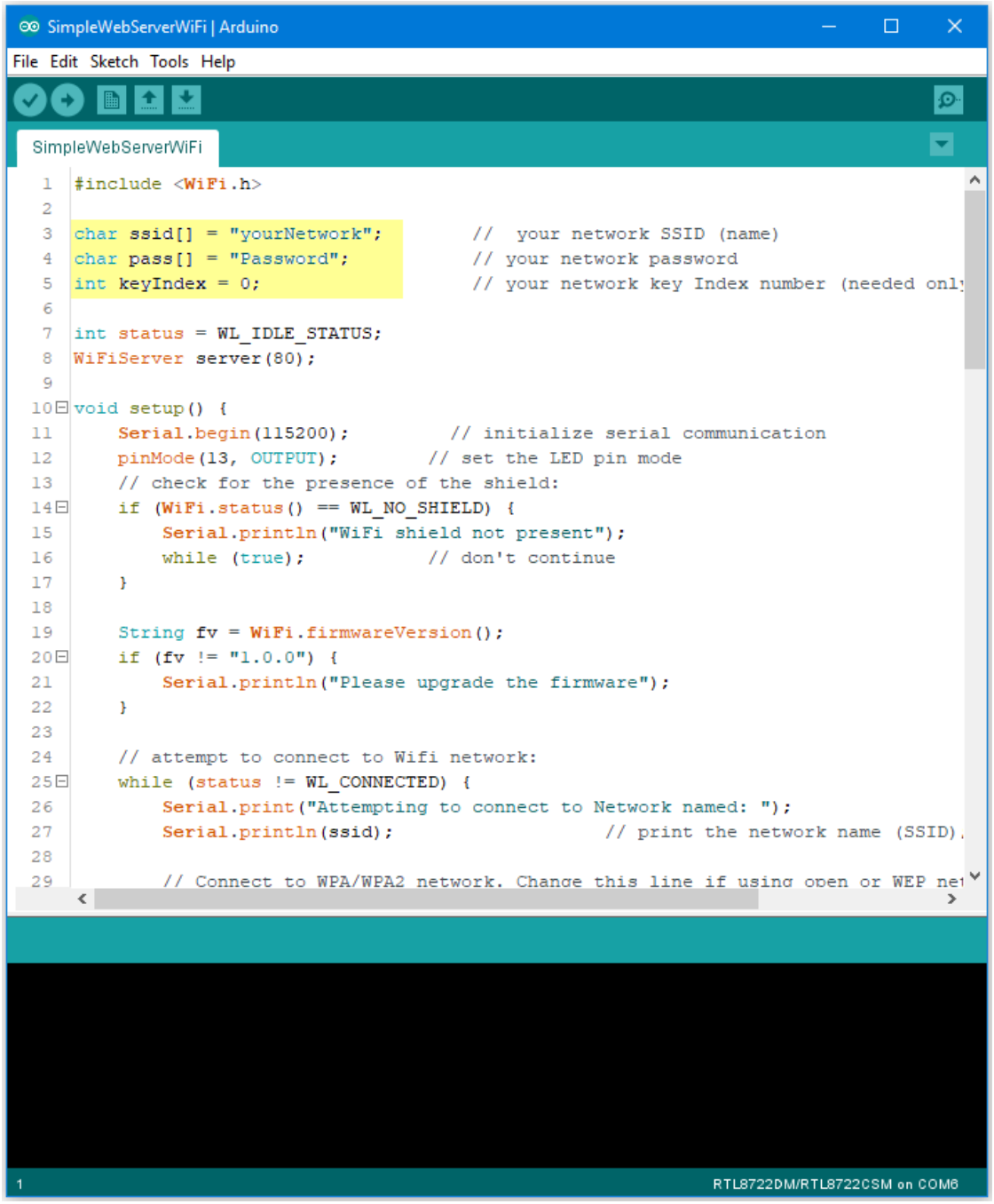


Note:
For BW16 board, you may consider to re-define "LED_PIN" macro to **10** for built-in green LED, or **11** for blue built-in LED, or **12** for red built-in LED to avoid using extra components.

Then open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`SimpleWebServerWiFi`"



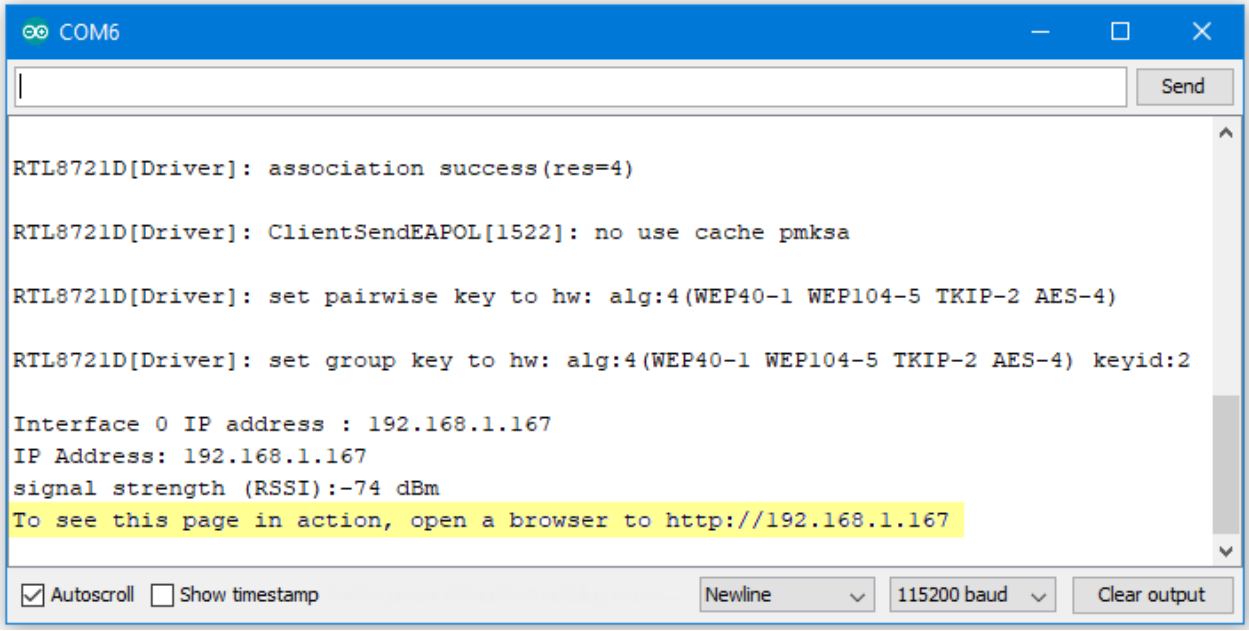
In the sample code, modify the highlighted snippet to corresponding information.



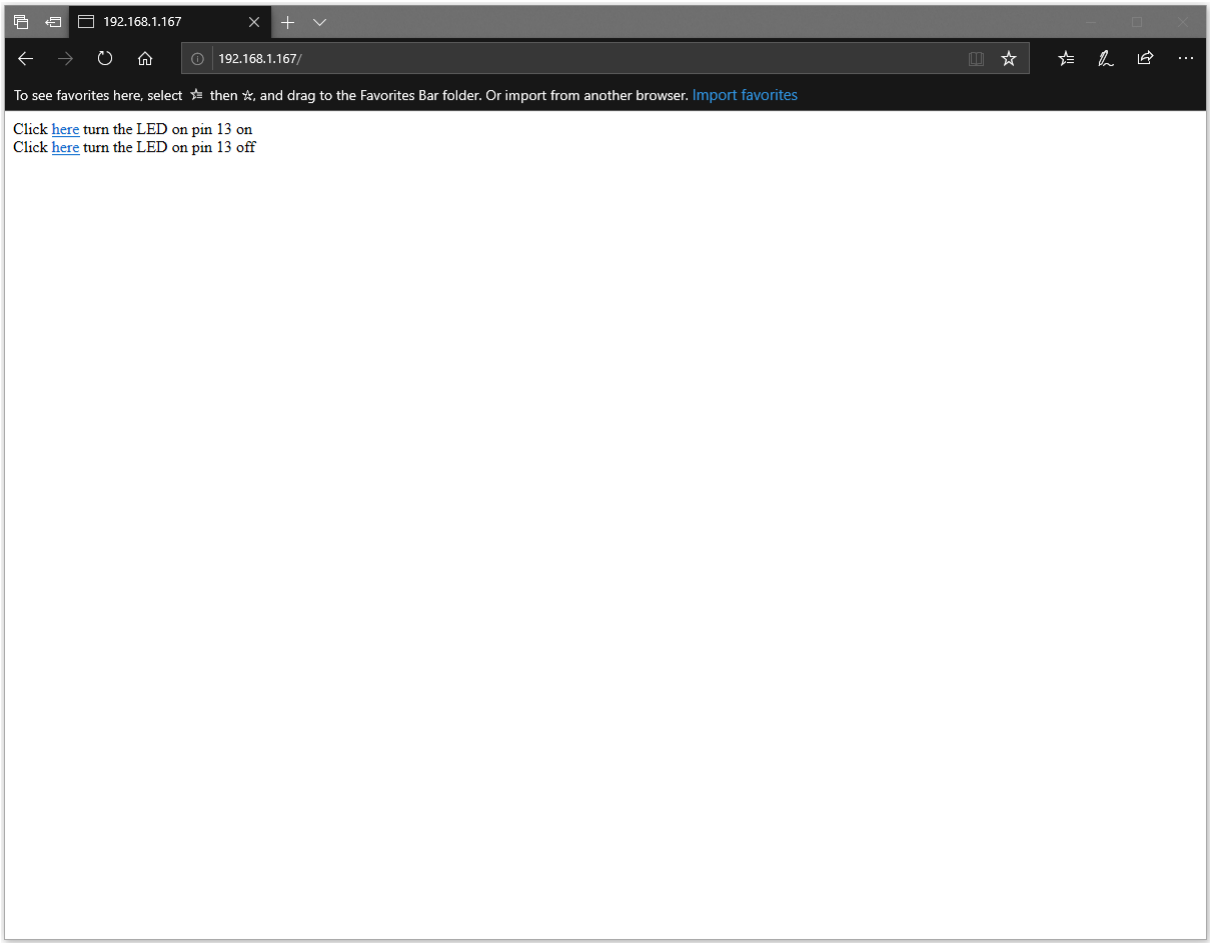
Upload the code and press the reset button on Ameba. When the connection is established, you will see the message:

```
"To see this page in action, open a browser to http://xxx.xxx.xxx.xxx"
```

in the Arduino IDE as shown in the figure:



Next, open the browser of a computer or a cell phone under the same WiFi domain, enter the address in the message.



In the webpage, you can turn on/off the LED.

Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use WiFi.RSSI() to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to get whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

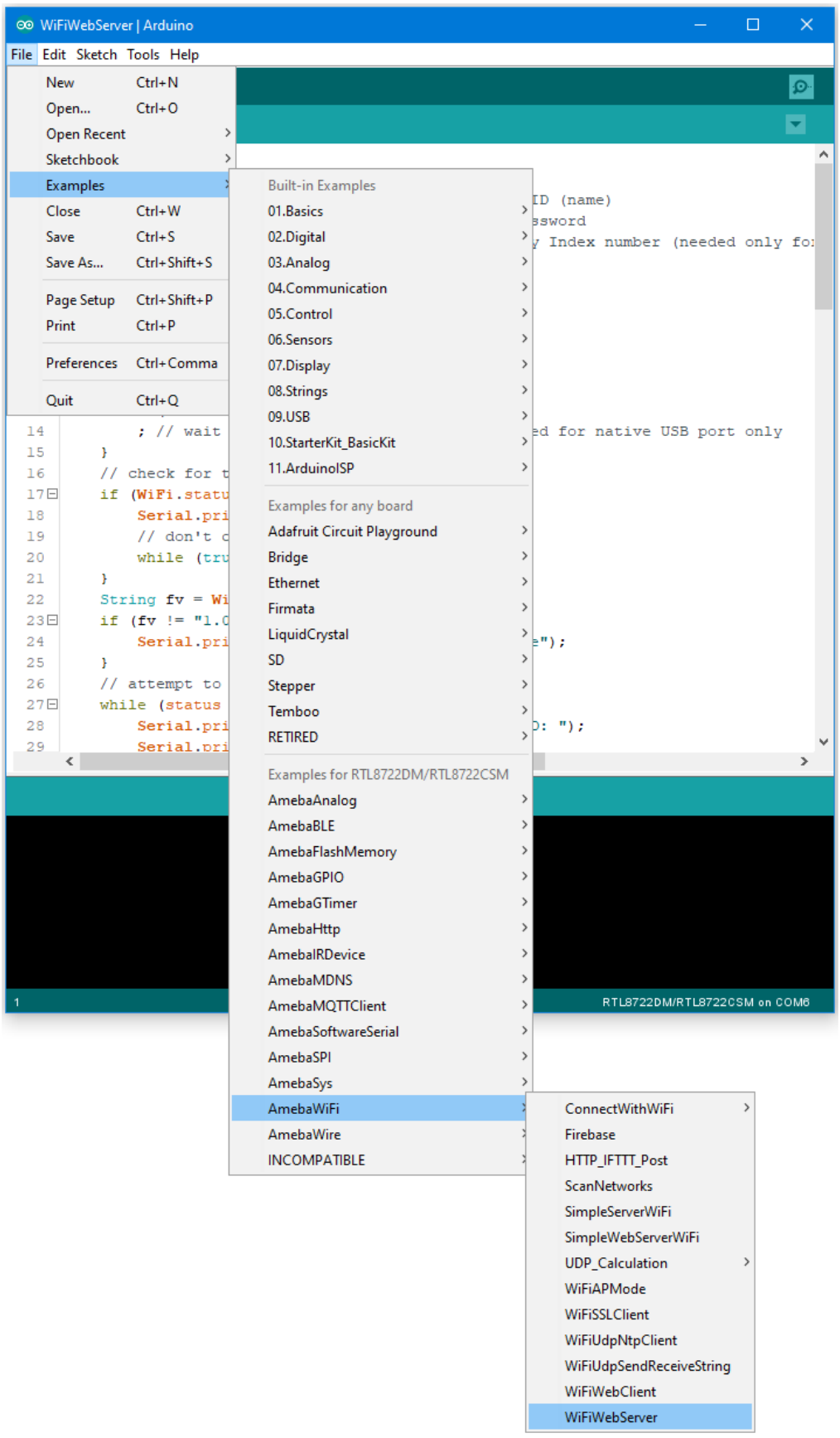
HTTP - Set up Server to Get the Ameba Status

Materials

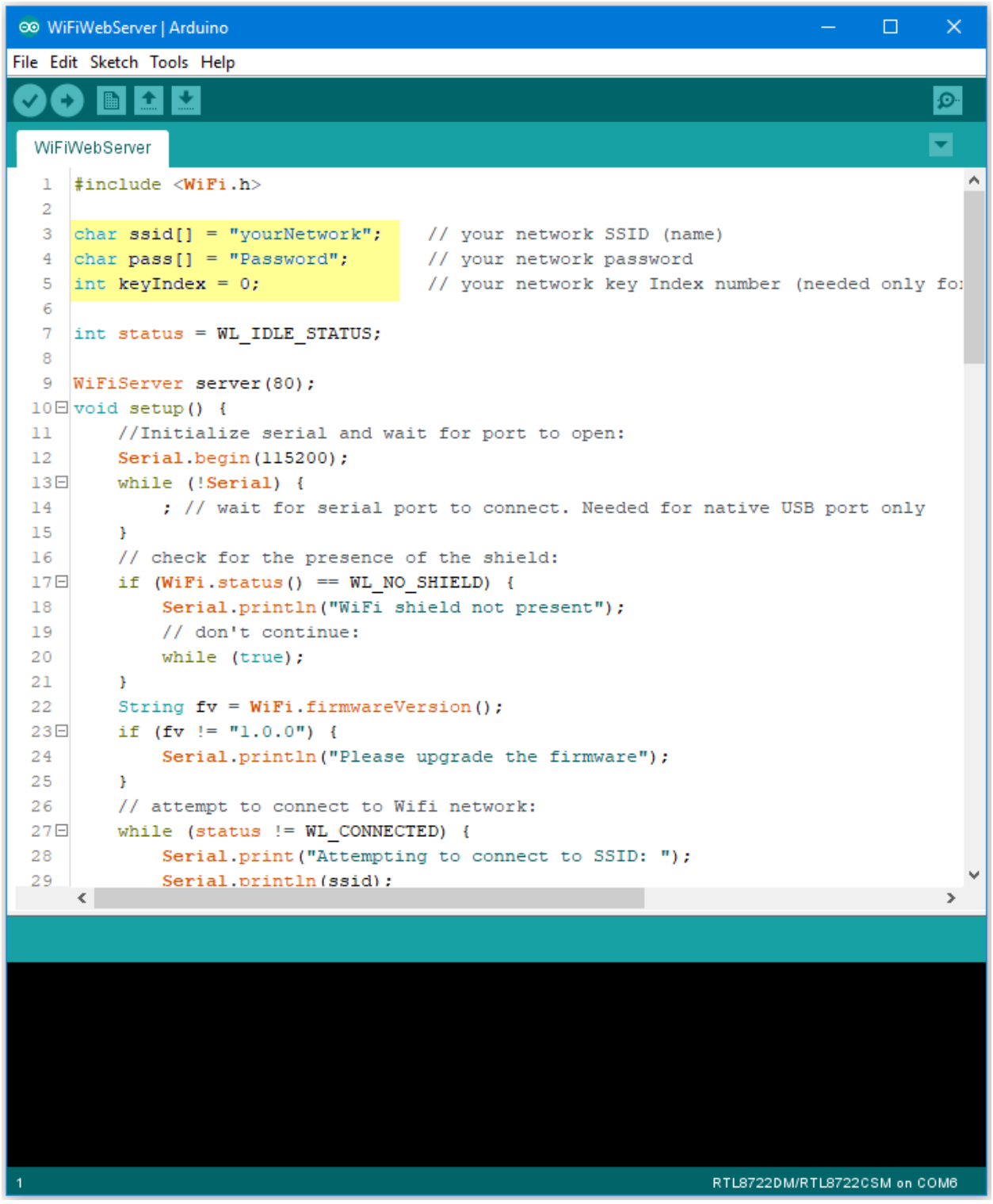
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

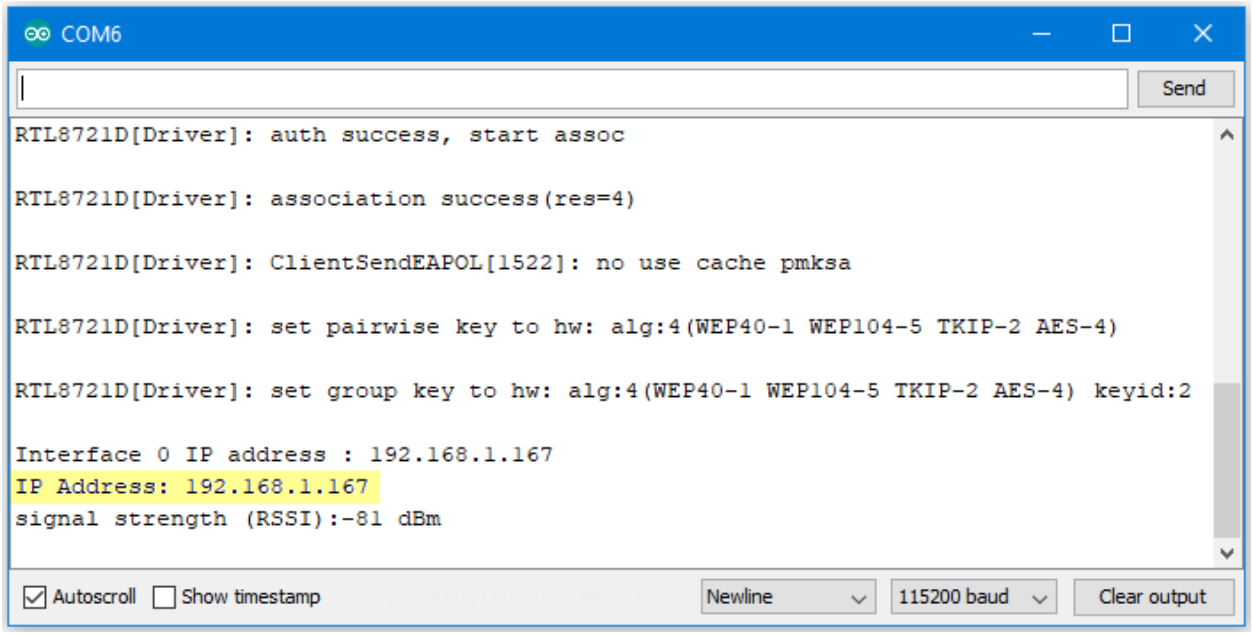
In this example, we connect Ameba to WiFi and use Ameba as server to send message to connected client.
First, open "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiWebServer`"


WiFiWebServer | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
ConnectWithWiFi
Firebase
HTTP_IFTTT_Post
ScanNetworks
SimpleServerWiFi
SimpleWebServerWiFi
UDP_Calculation
WiFiAPMode
WiFiSSLClient
WiFiUdpNtpClient
WiFiUdpSendReceiveString
WiFiWebClient
WiFiWebServer
RTL8722DM/RTL8722CSM on COM6

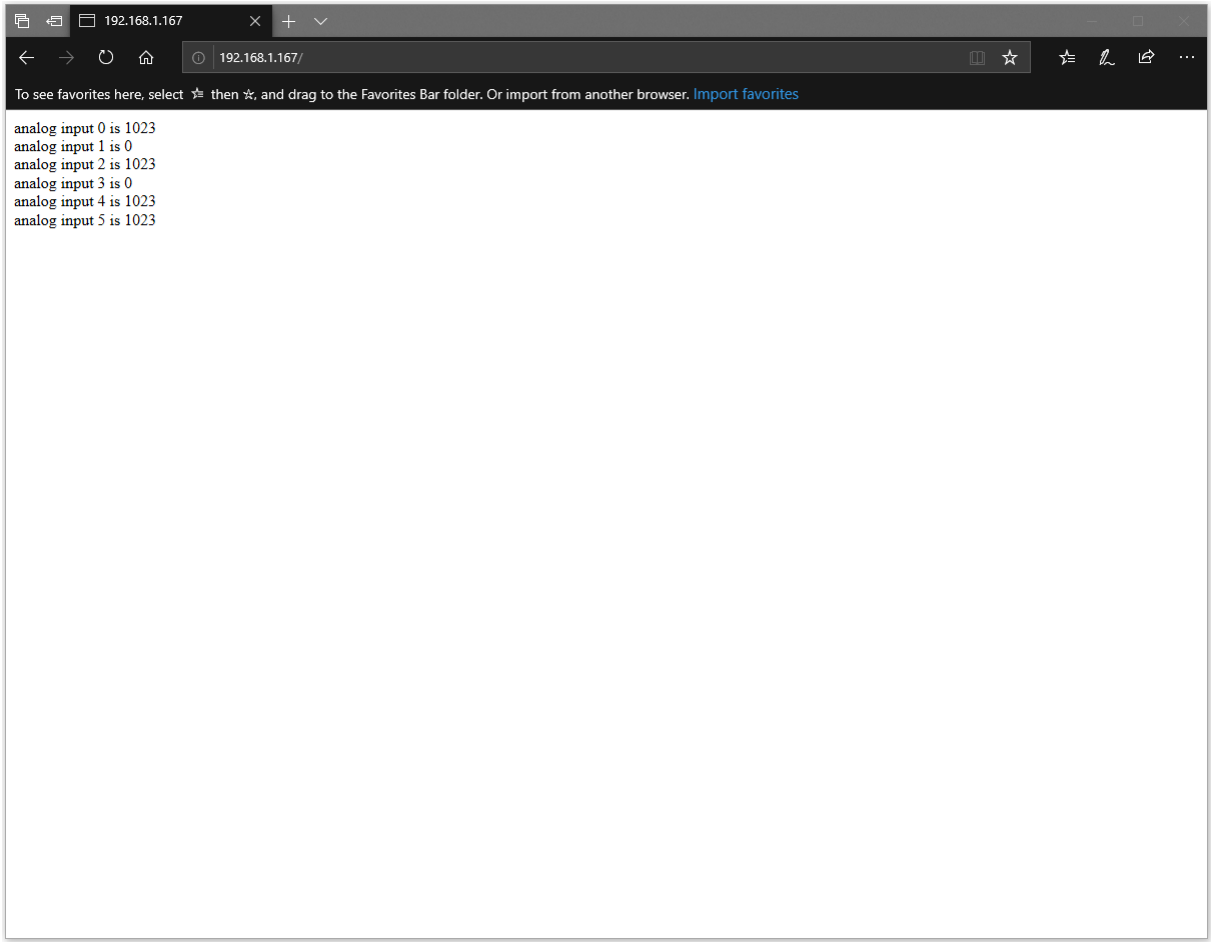
In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. After connecting to WiFi, Ameba starts to run as server. The IP of the server is shown in the serial monitor, and port is 80.



We connect to the server in a browser, and we can see the data sent from the server.



Code Reference

Use `WiFi.begin()` to establish WiFi connection.
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
se `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiServer server()` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.connected()` to check whether or not the client is connected.
https://www.arduino.cc/en/Reference/WiFiClientConnected
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.print()` to print data to the server that a client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientPrint
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

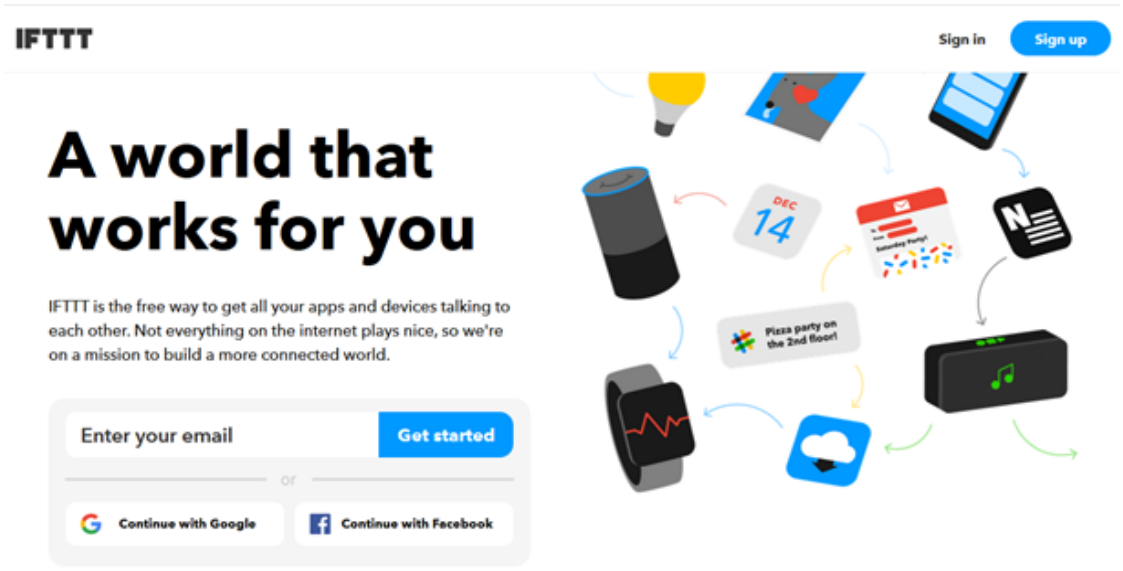
HTTP - Use IFTTT for Web Service

Introduction to IFTTT

IFTTT, known as If This Then That, is a website and mobile app and free web-based service to create the applets, or the chains of simple conditional statements. The applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, Pinterest etc.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- An account from https://ifttt.com/ , in order to access IFTTT service*



Note: Upon log in, there are several cloud and online services that are integrated with IFTTT platforms.

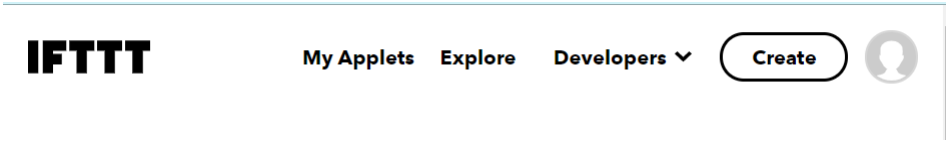
Example

- Generate Applet from IFTTT

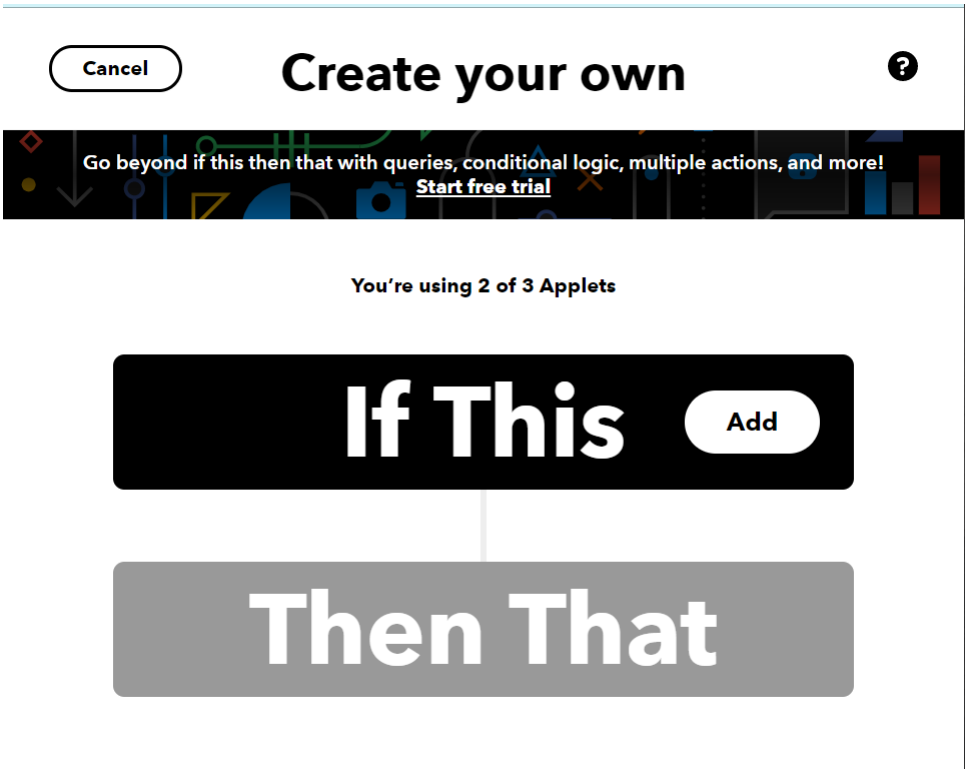
  In this example, we obtain an example of IFTTT Applet to send email to specified recipient.

  To run the example, HTTP POST feature of the Ameba is used to post a simple webhook service that is received by IFTTT platform and in turn be used to trigger a response (sending an email).

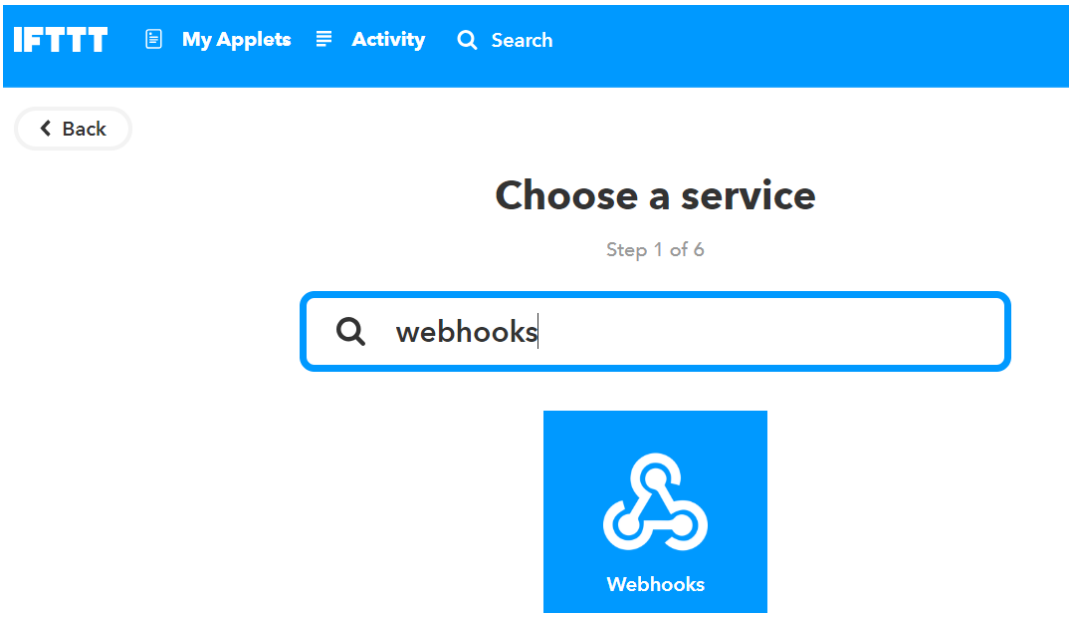
  After logging in https://ifttt.com/, click **Create** from the top bar.

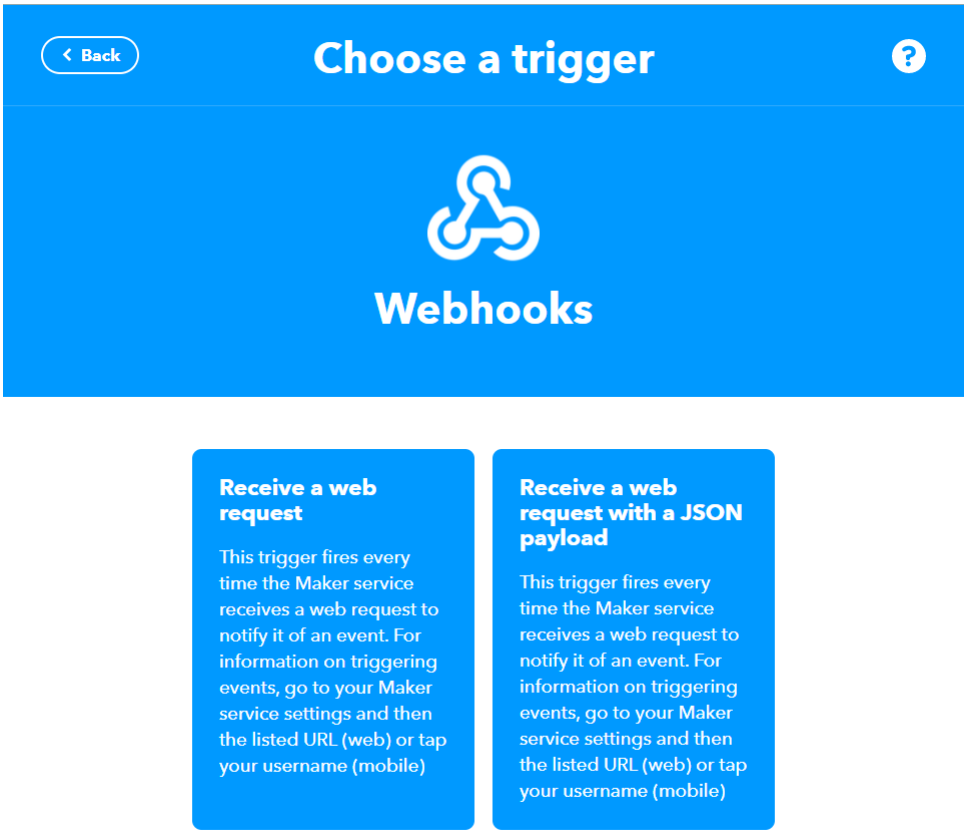
  Click **"Add"** to add the trigger.



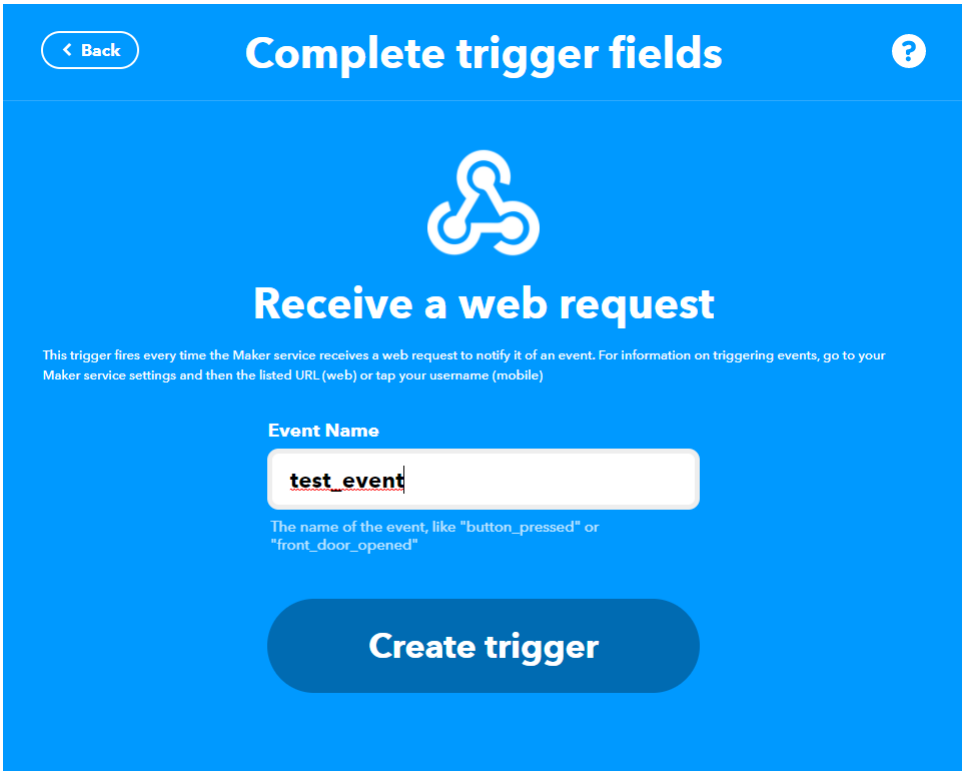
Choose Webhooks service as shown below. Alternatively, search the service by typing into the search bar.



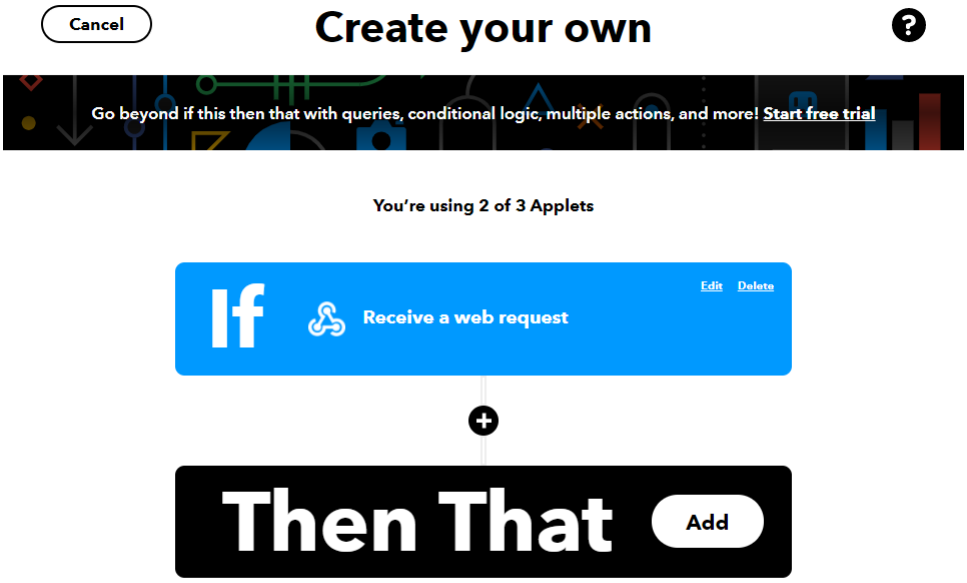
After that, the available triggers will appear. Choose Receive a Web request.



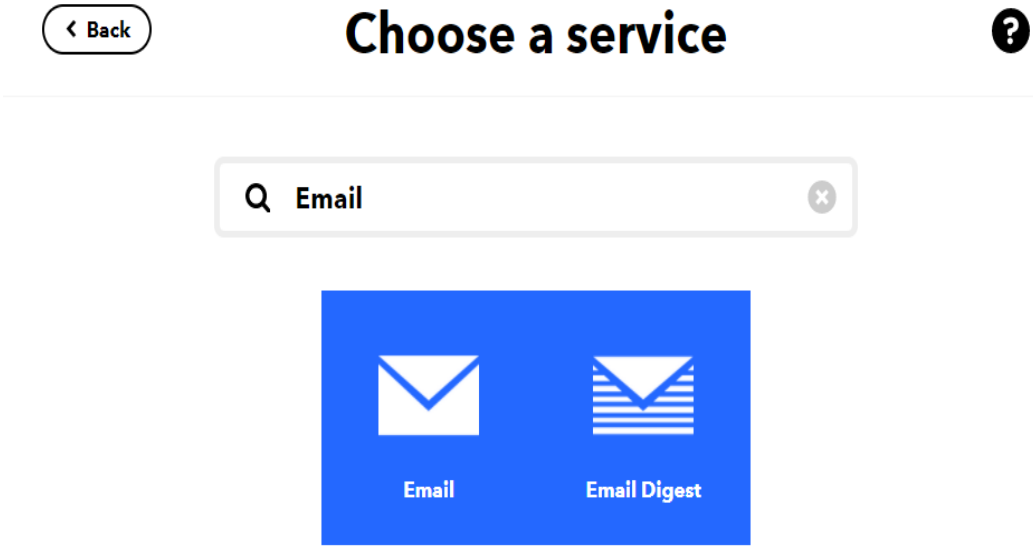
Next, an Event Name is required to identify the trigger successfully. In this example, set the Event name as “test_event”.



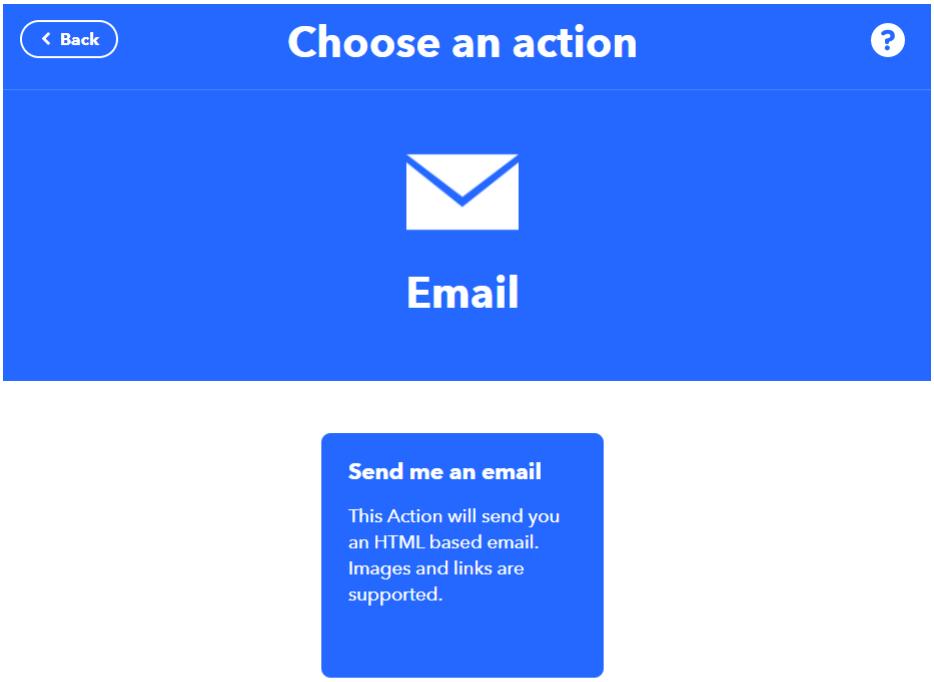
Next, click **Add** in Then That field to create the action service taken in response to the last trigger.



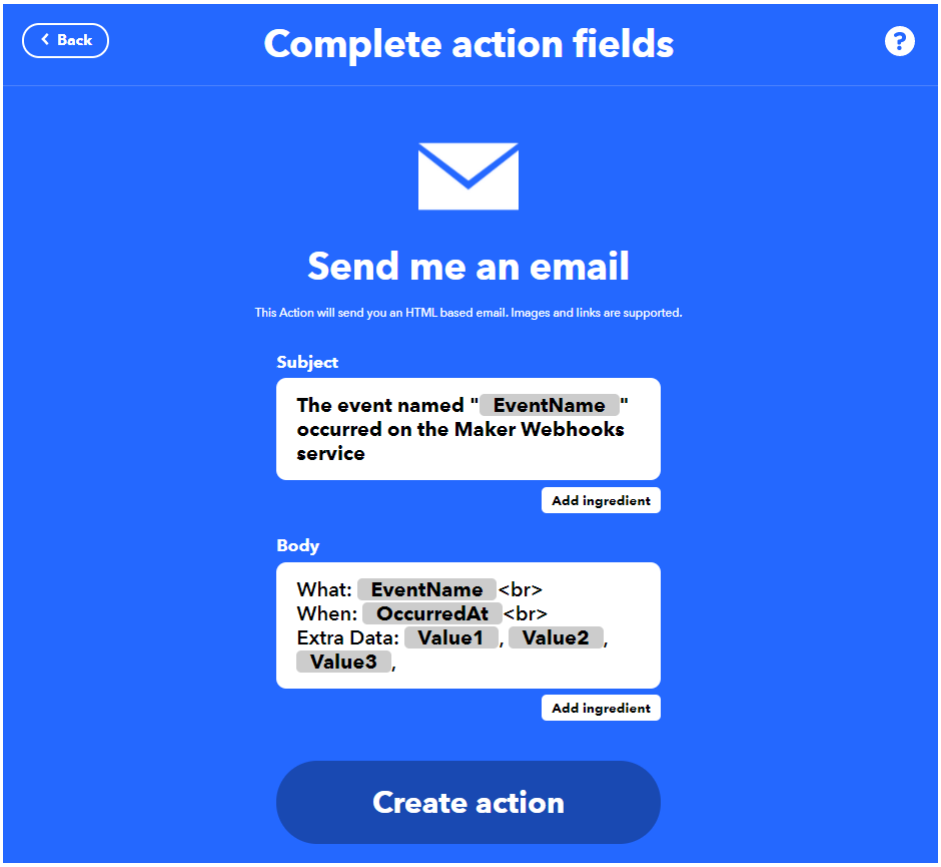
Choose Email as the action service.



Click on Send me an email.



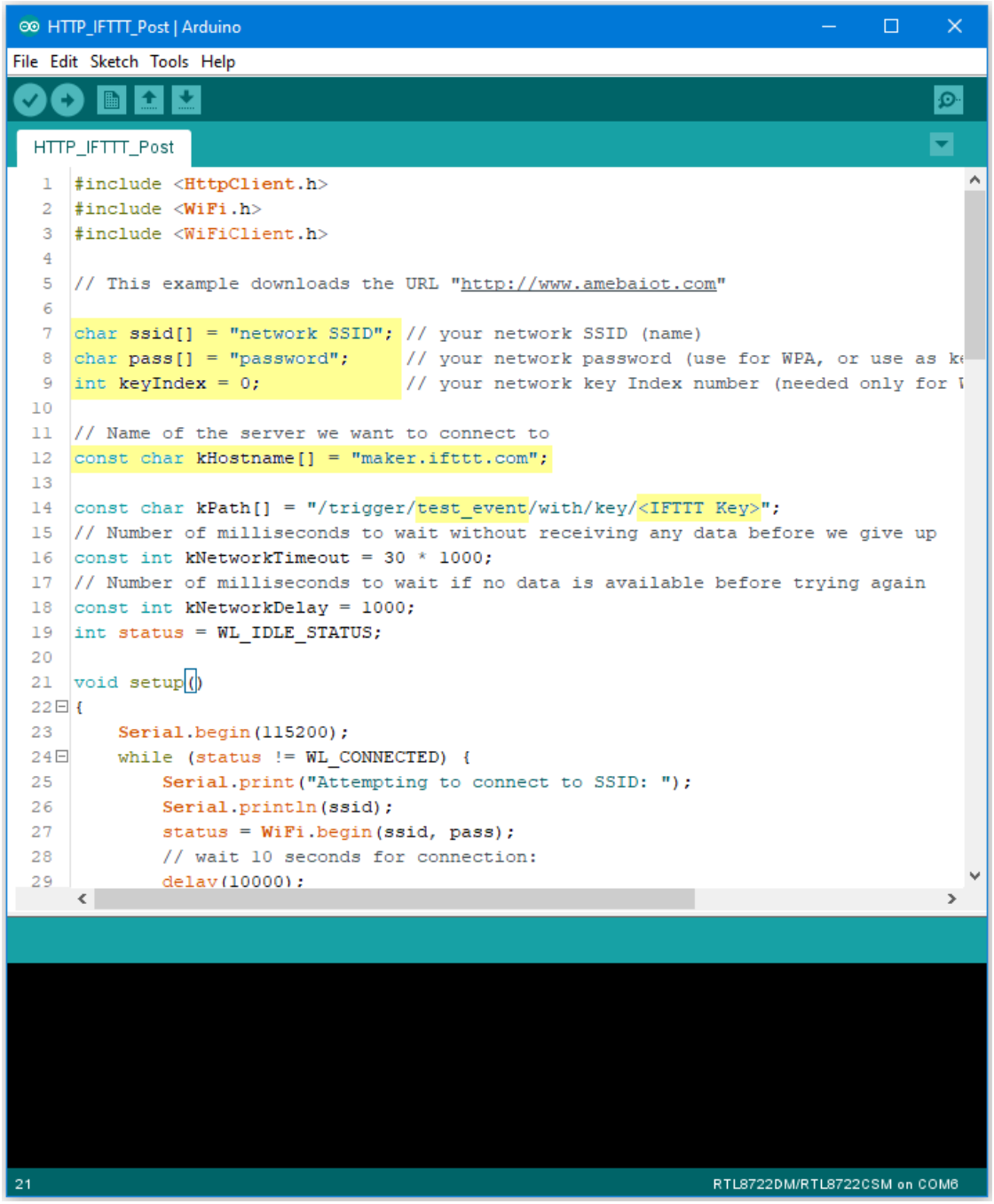
Under the template of **Send me an Email**, the contents of the email, such as subject and body is editable. Click **Create Action** to complete the action. Take note that **Email service** is offered to the email address registered under IFTTT account.



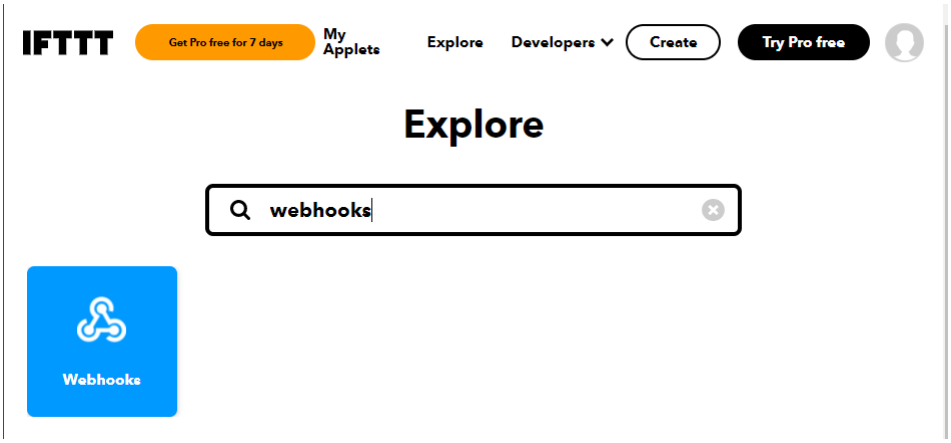
- Post the Trigger via Ameba

Once the Applet is ready in the IFTTT dashboard, the example program can be flashed onto the Ameba board to post the HTTP request.
Open the example code in "File" -> "Examples" -> "AmebaWiFi" -> "HTTP_IFTTT_Post"
In the example program, edit the following 3 items inside the code to make the program work.

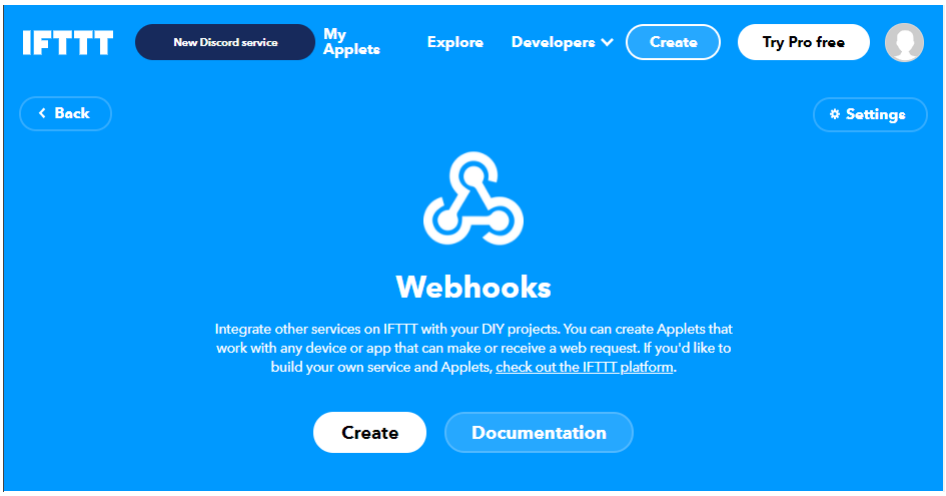
1. The WiFi credentials to connect to the Wi-Fi hotspot or access point of desirable choice.
2. Under the Host name field, enter the host name of the IFTTT service "maker.ifttt.com".
3. Under the Path name field, enter the Event name and key field "/trigger/Event name/with/key/Key Field"
   - Event name: The event name should be the same as the one specified in the IFTTT applet. In this example, the event name is "test_event".
   - Key Field: Available under webhook service in individual IFTTT account. See the next step for the steps to obtain the Key Field.



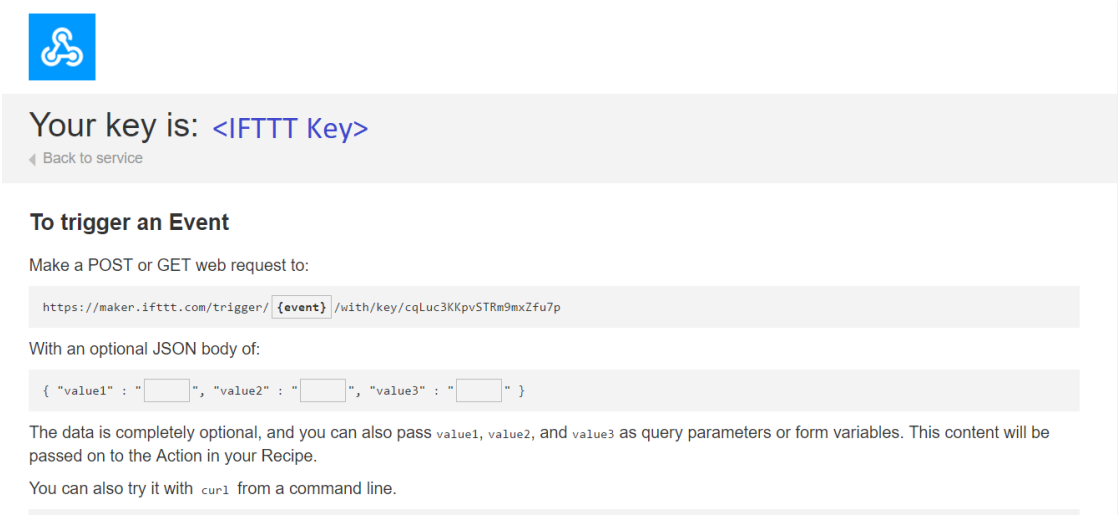
To obtain a key from documentation tab of the Webhooks, find the webhook service in the Explore tab.



On the Webhooks service page, click on the Documentation tab.

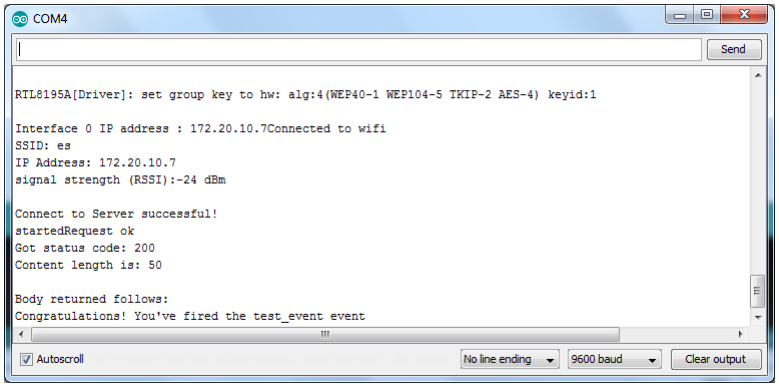


The key can be found in the documentation page. Also, information on how HTTP request can be used.

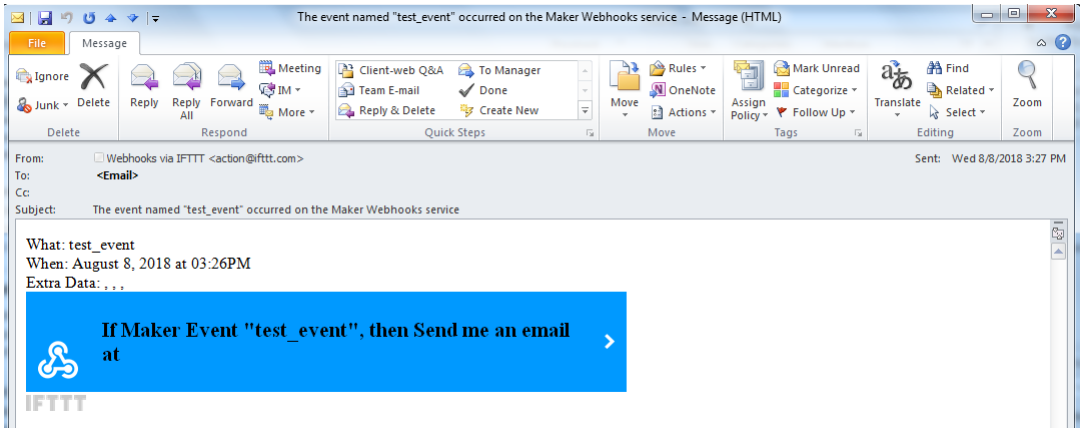


Once the example is ready, connect to Ameba board via USB Cable.

On the Arduino IDE, compile the code and upload the code onto Ameba and press the reset button. After the event has been successfully fired, “Congratulations! You have fired the test_event event” can be seen on the serial monitor and an email reminder for this event will be delivered.



Thereafter an email is sent to recipient email account registered at IFTTT Applet and email notification will be received.



IPv6 – Ameba as IPv6 Server/Client over TCP

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

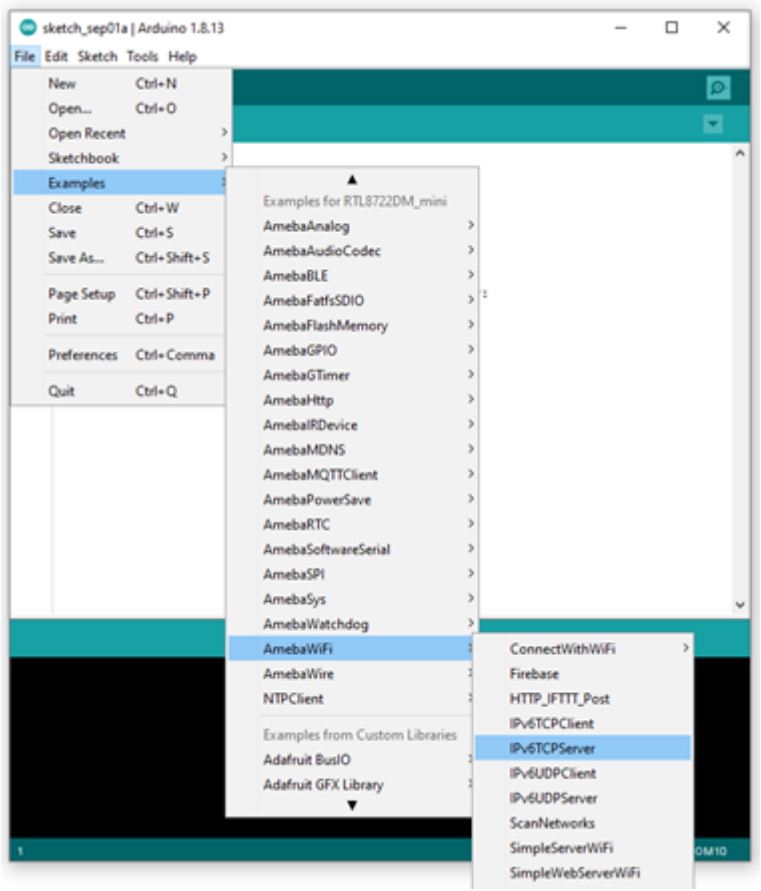
Example

Introduction

This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over TCP. Note that this example only works after you have set up the server and then configure the client accordingly.

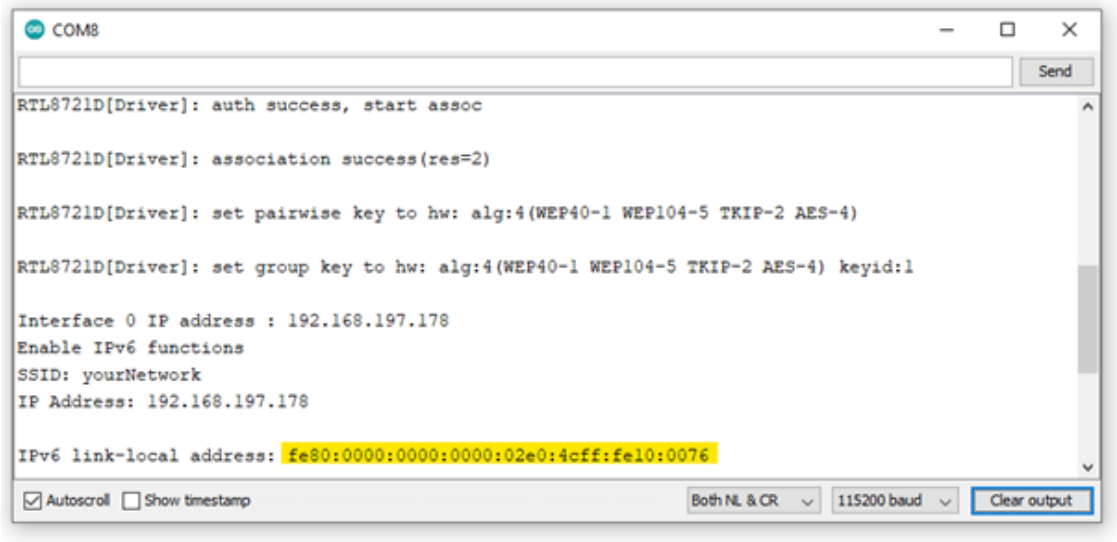
Procedure

Step 1. IPv6TCPServer Open the example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6TCPServer`”.

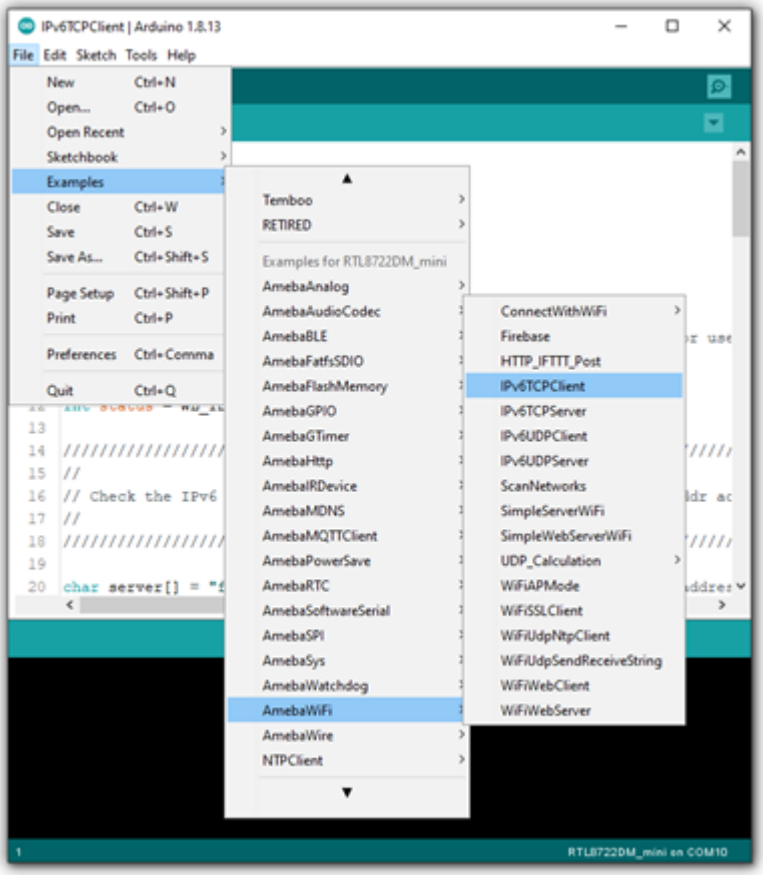


In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

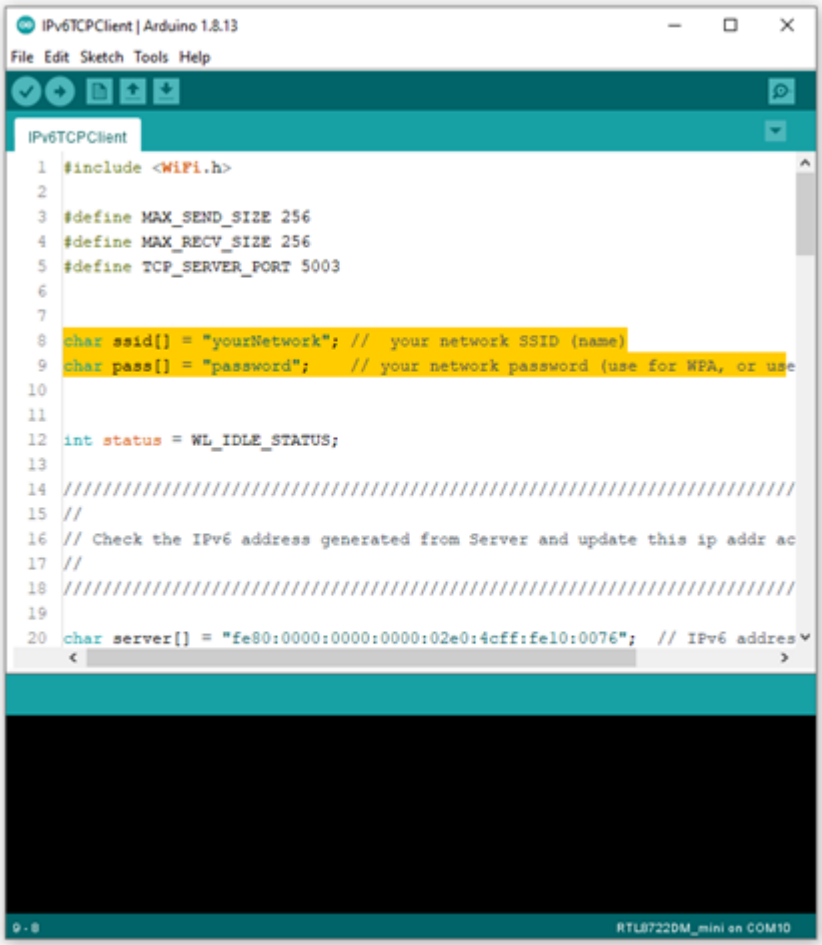
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



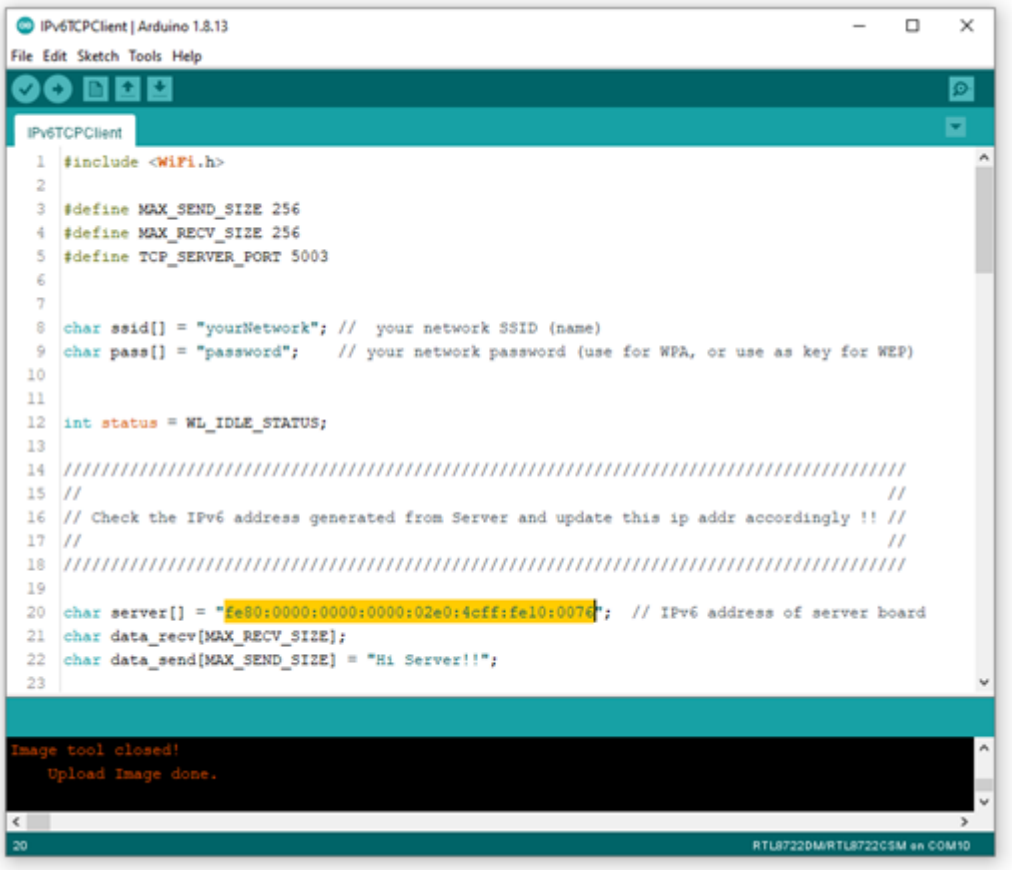
Step 2. IPv6TCPClient Now take the second Ameba D and open another example, "`Files` -> "`Examples` -> "`AmebaWiFi` -> "`IPv6TCPClient`".



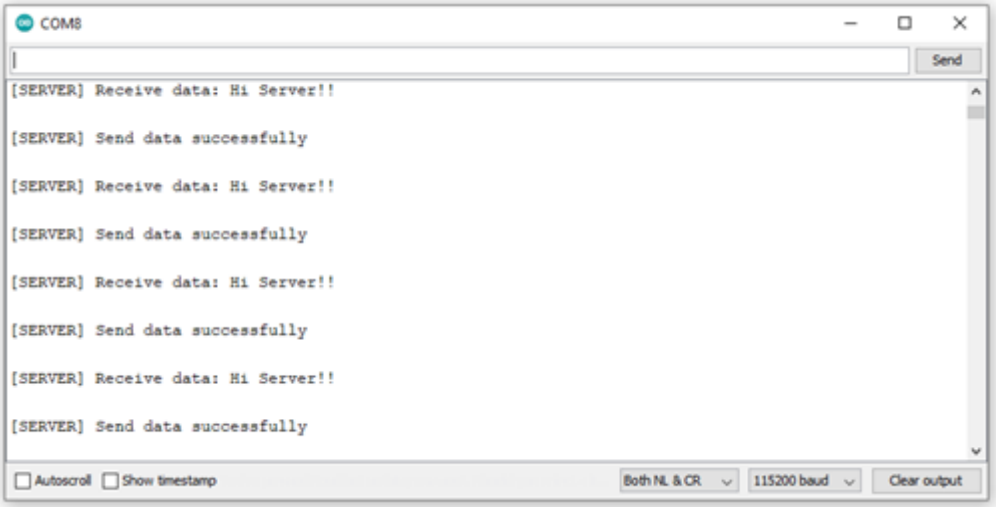
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

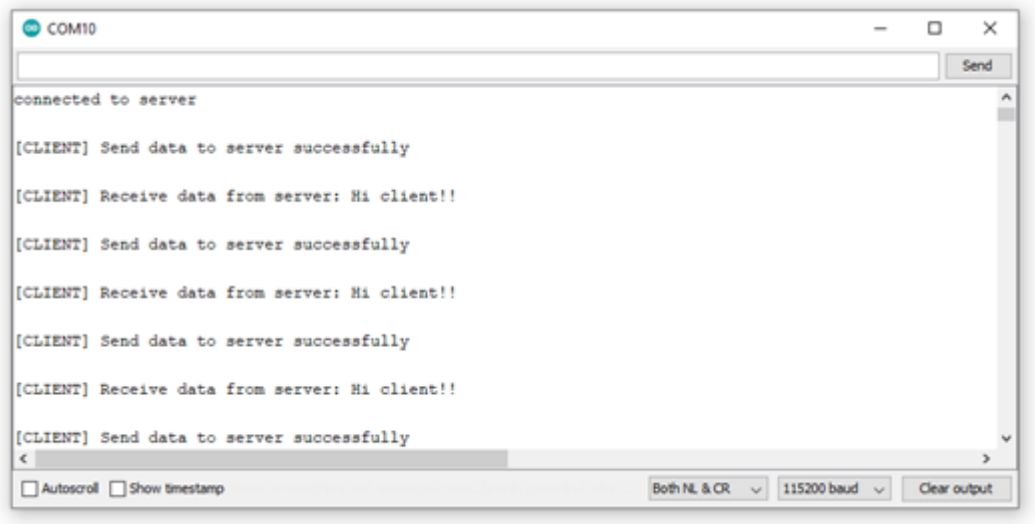


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6TCPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





IPv6 – Ameba as IPv6 Server/Client over UDP

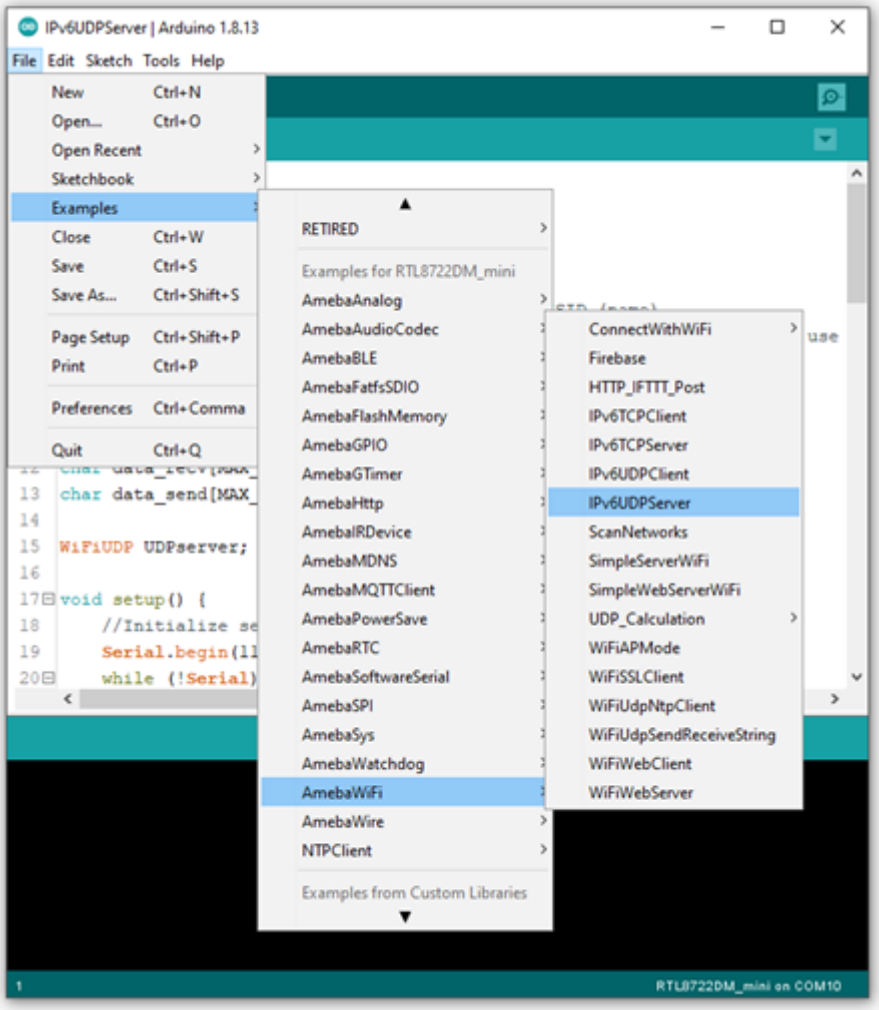
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2

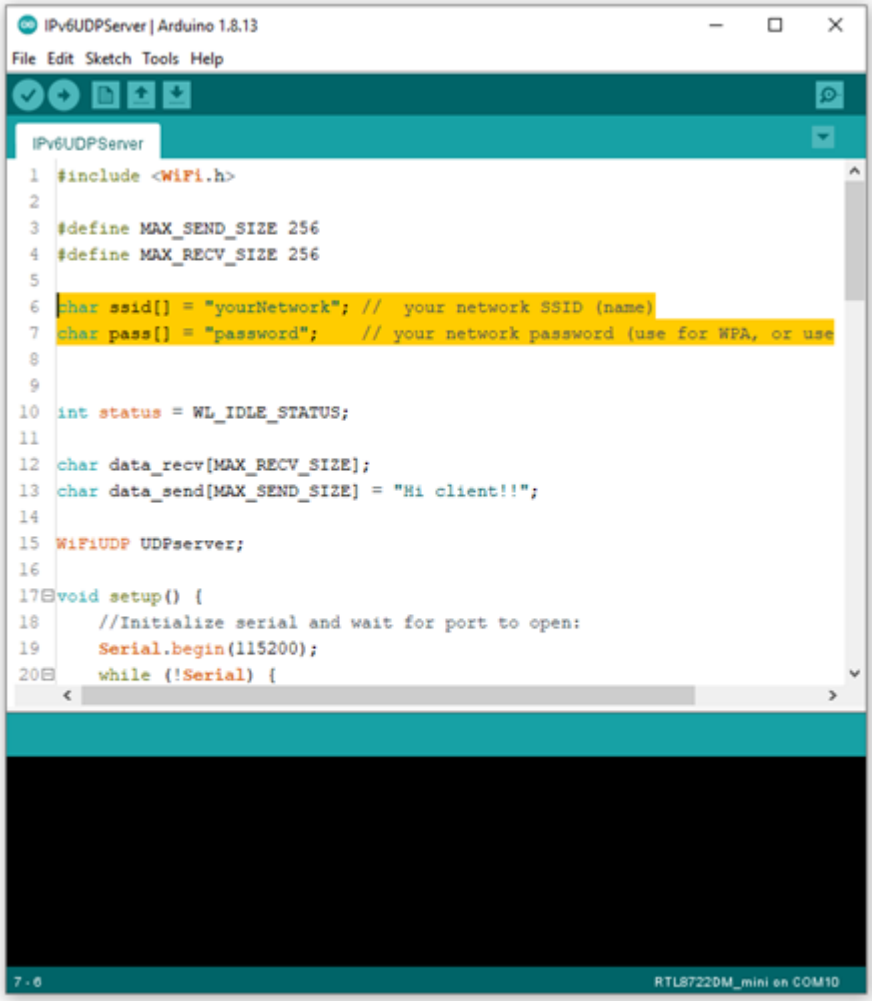
Example

Introduction

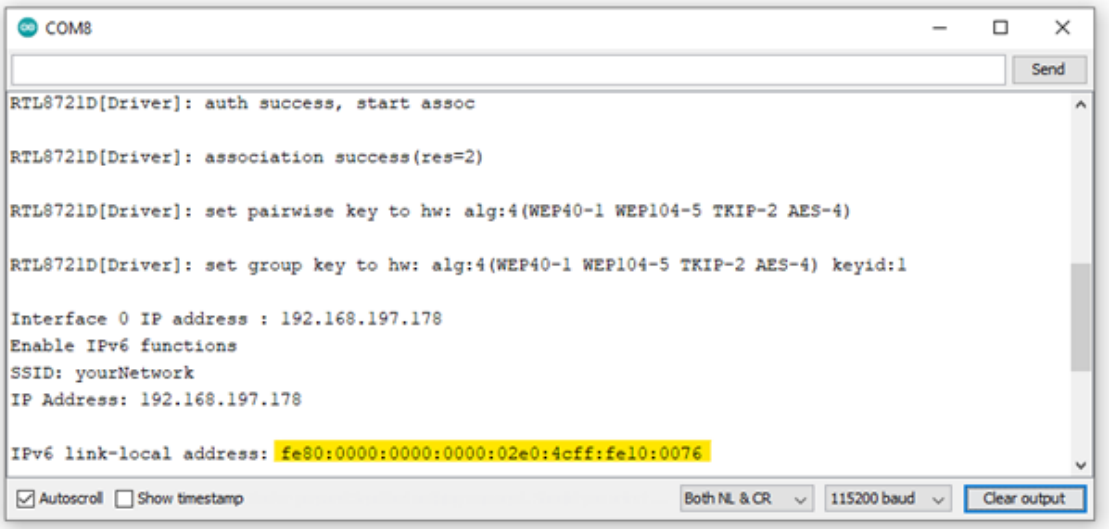
This example shows how Ameba can communicate on the local network using Internet Protocol version 6 over UDP. Note that this example only works after you have set up the server and then configure the client accordingly.



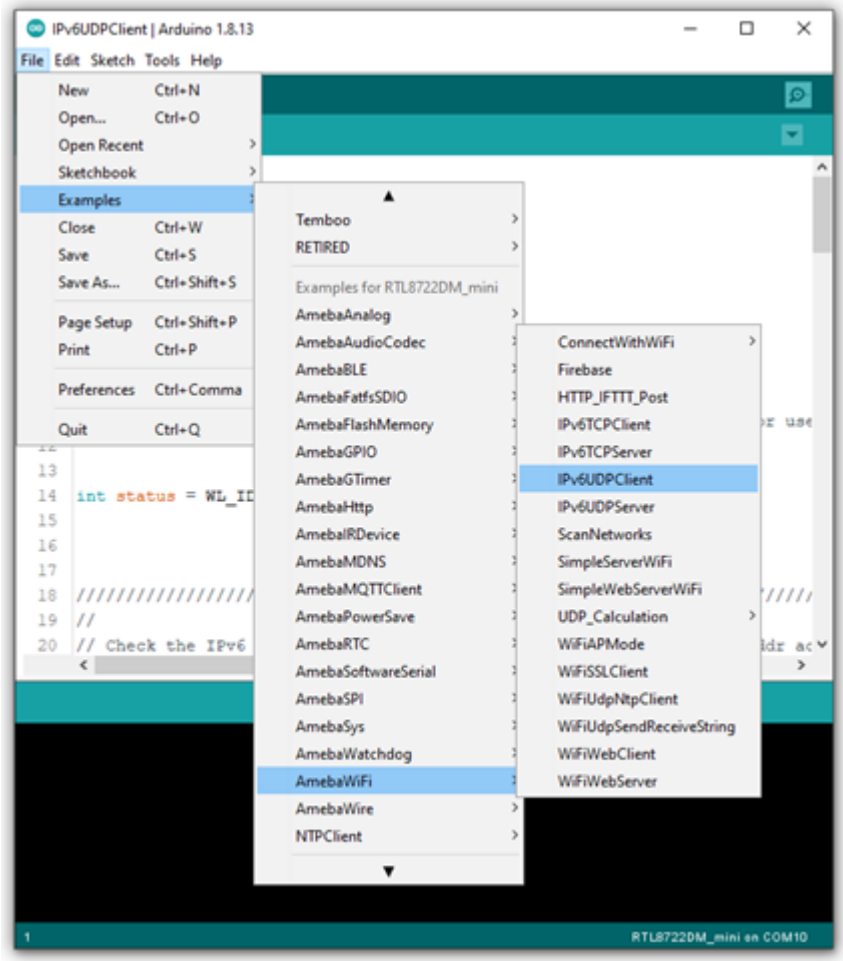
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.



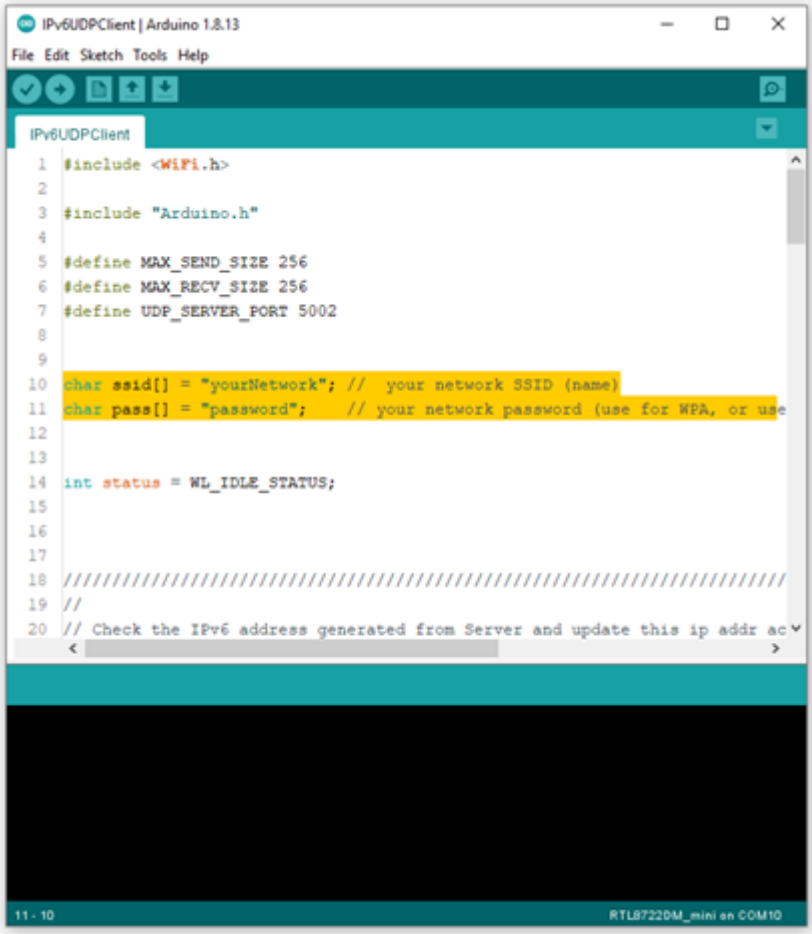
Next, upload the code and press the reset button on Ameba once the upload is finished. Open Serial Monitor and copy the IPv6 address of the Server (the highlighted area) for later use,



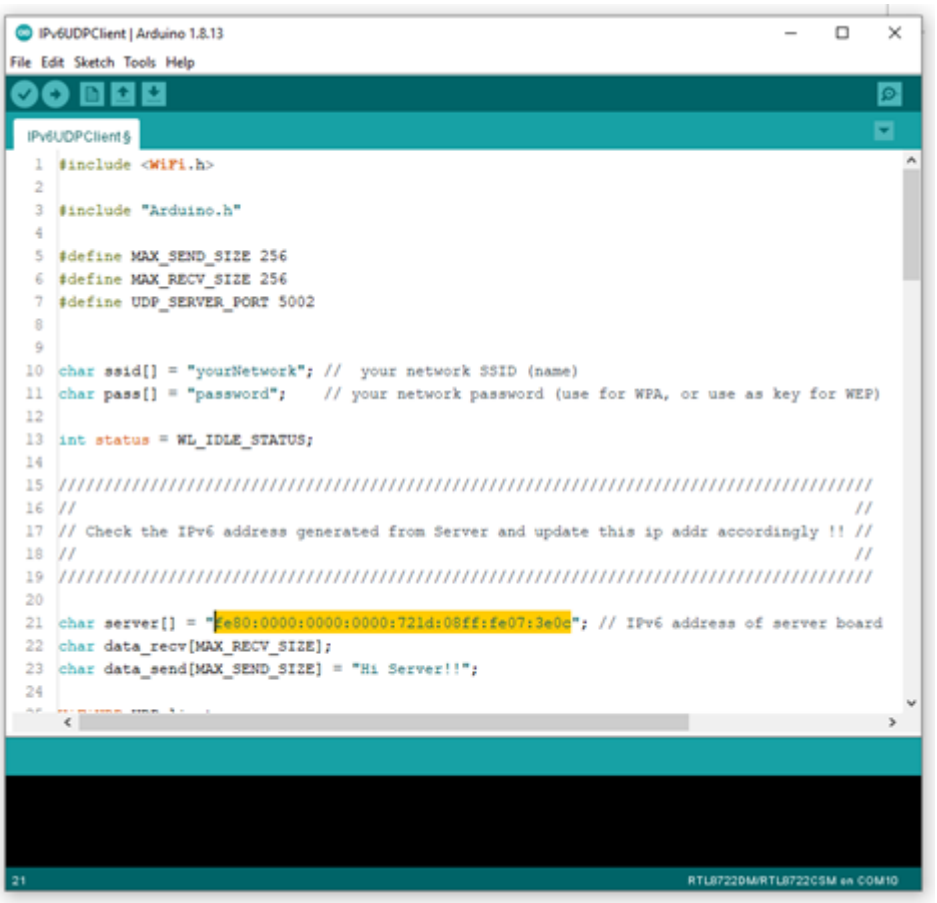
Step 2. IPv6UDPClient Now take the second Ameba D and open another example, “`Files`” -> “`Examples`” -> “`AmebaWiFi`” -> “`IPv6UDPClient`”.



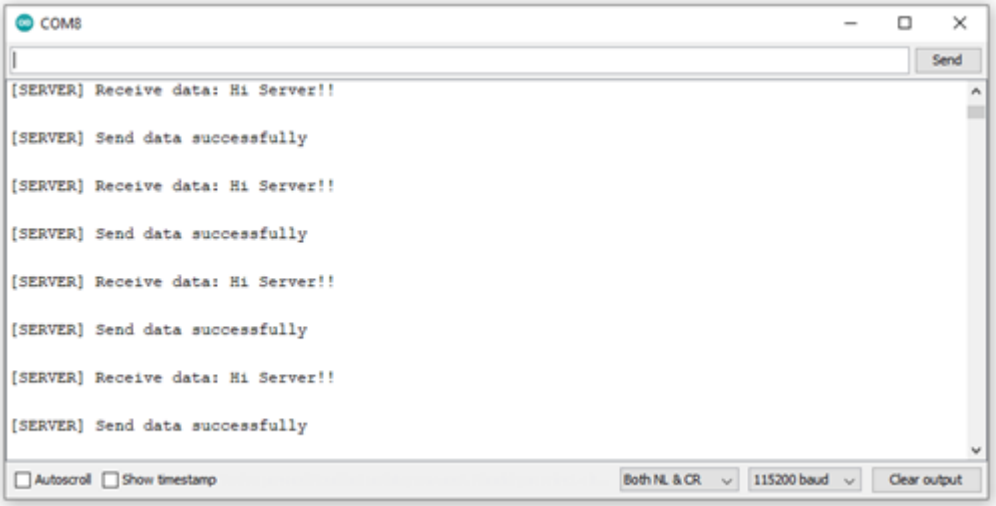
In the sample code, modify the highlighted section to enter the information required (ssid, password) to connect to your WiFi network.

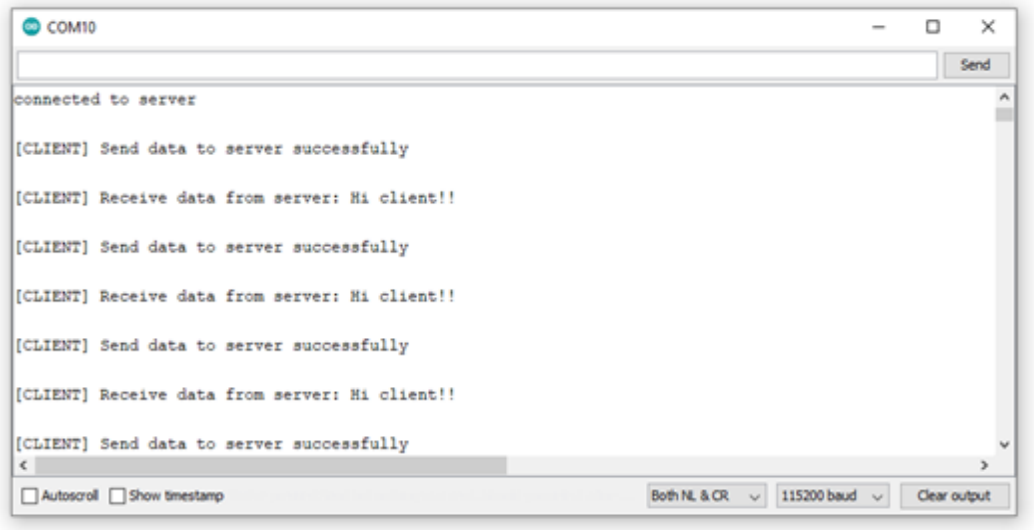


From the previous step, we have obtained the Server's IPv6 address, now we copy the server's IPv6 address to "IPv6UDPClient" example in the highlighted area below,



Next, upload the code and press the reset button on Ameba once the upload is finished.
Open Serial Monitor on the port to the second Ameba D, you should see server and client are sending text message to each other at the same time.





MDNS - Set up mDNS Client on Arduino IDE

Preparation

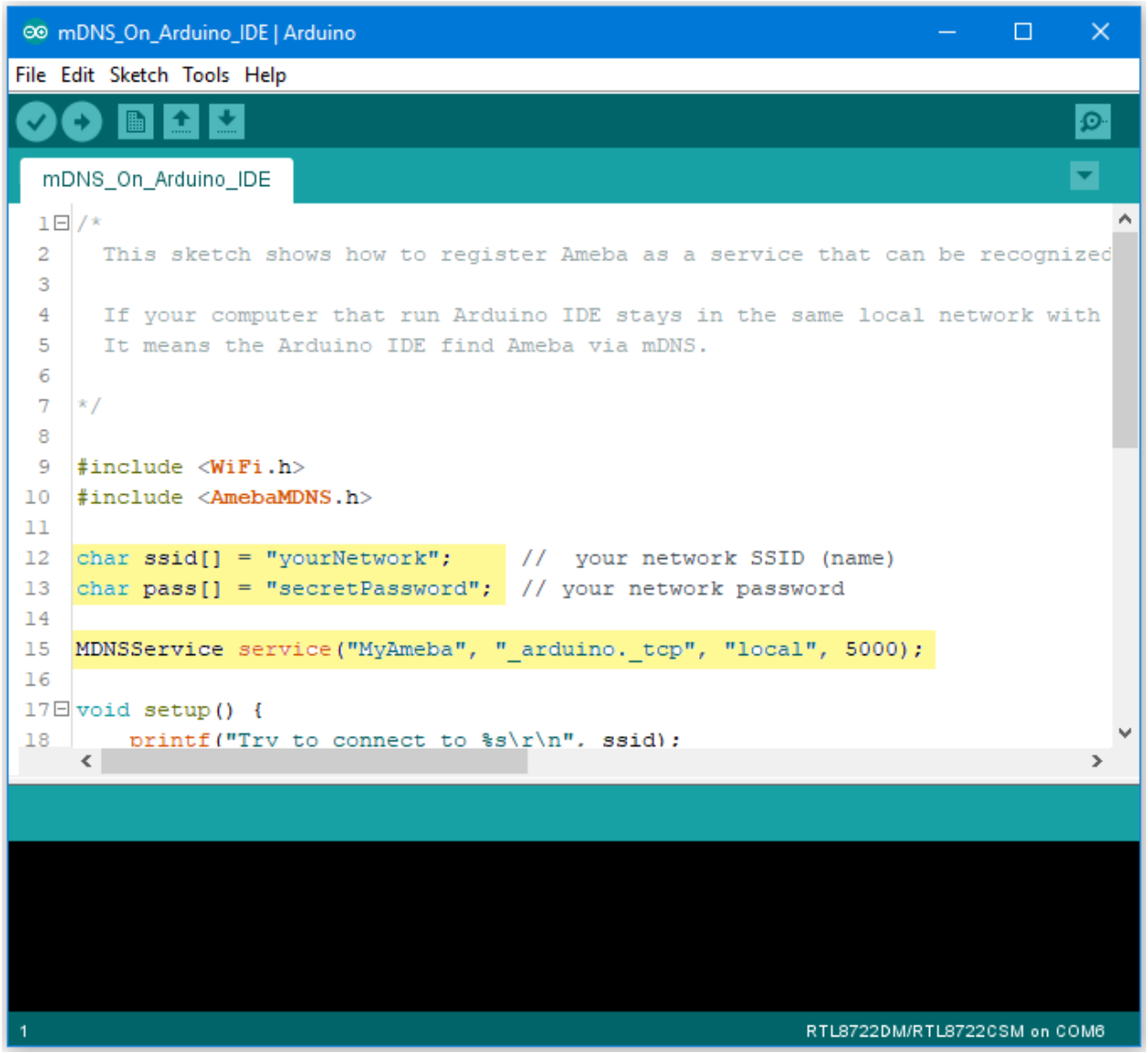
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

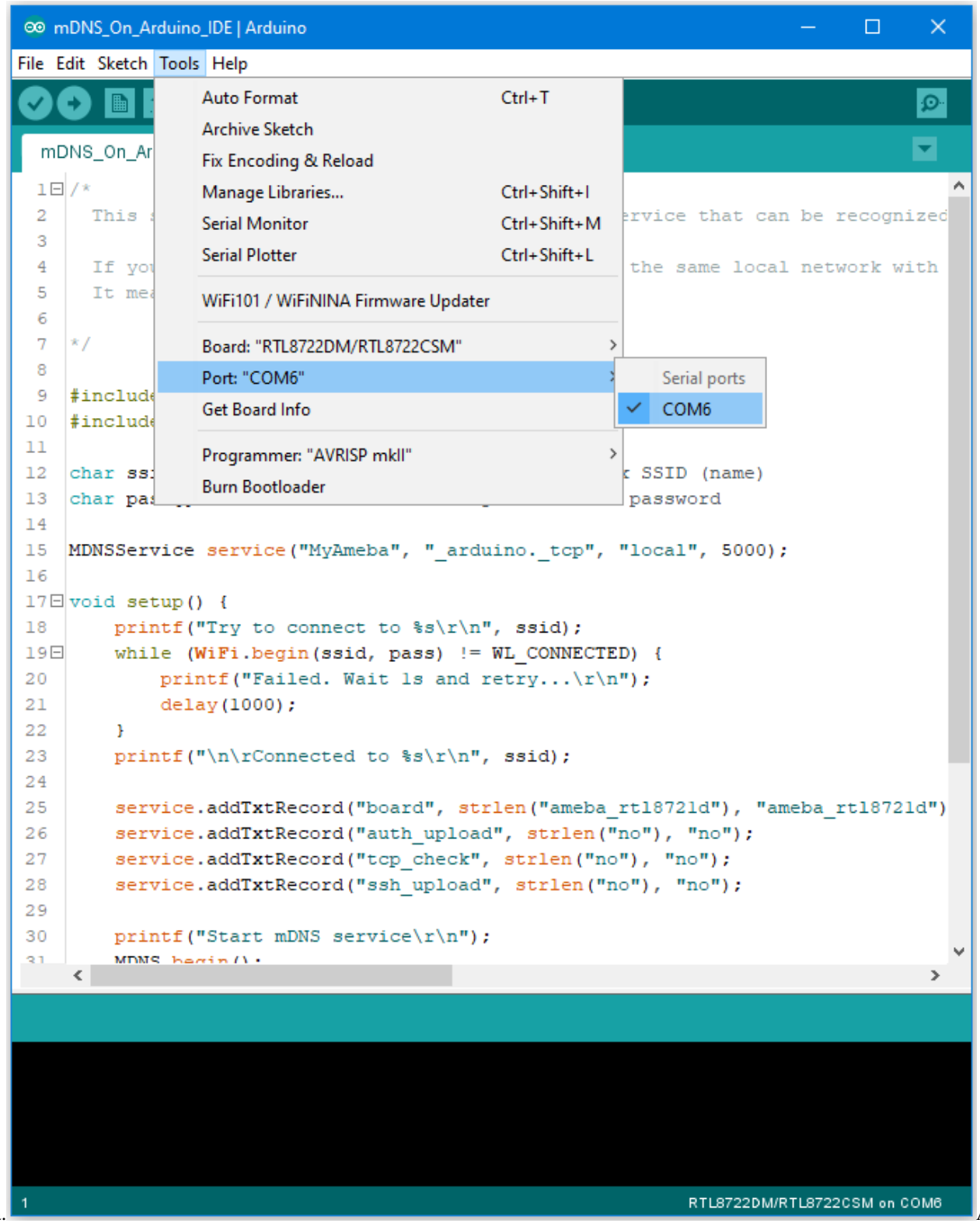
mDNS (Multicast DNS) is a protocol used in the local area network. It delivers the network information like IP address and provided services to others. mDNS is based on the UDP protocol, and it sends packets to 224.0.0.251 with port 5353 under IPv4 address. The naming style for the service follows the format: **{Instance Name}.{Protocol Name}.{Domain}**

- Instance Name: used to identify the name of the service
- Protocol Name: Divided into two parts, the front end is in regard to the name of the service, and it adds baseline as a prefix. The rear end is in regard to the transport protocol name it used, and it also adds baseline as a prefix
- Domain: Local area network in normal cases

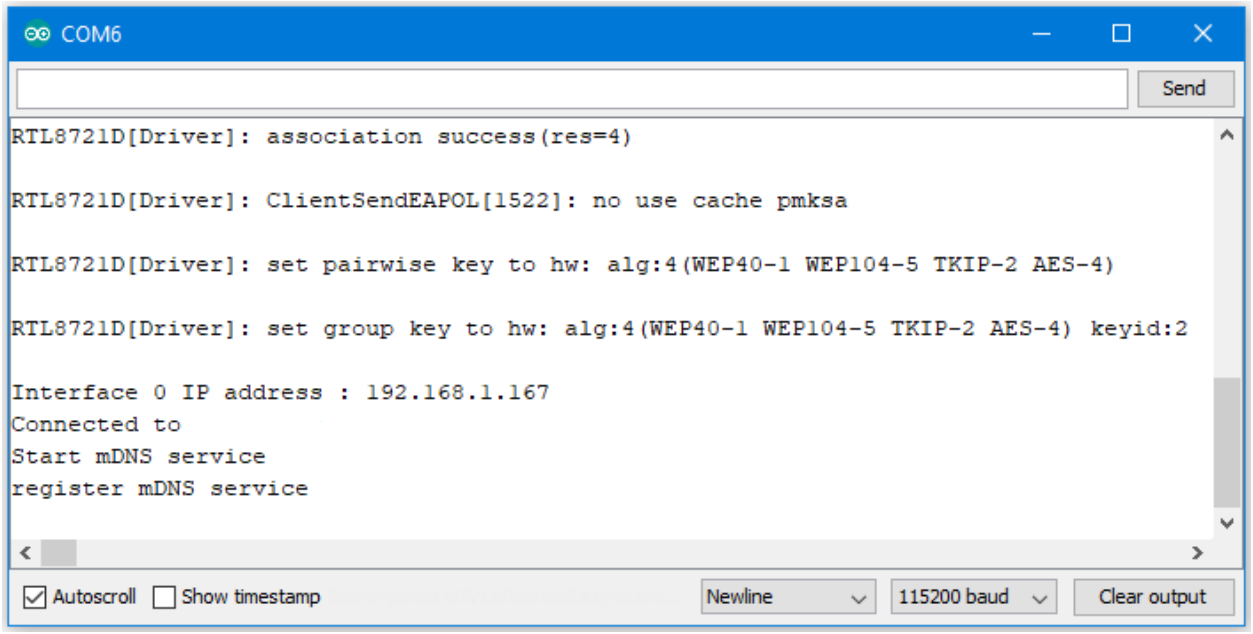
For example, Arduino IDE adopts the naming for the mDNS service which is used in OTA as following: **MyAmeba._arduino._tcp.local**
Among the naming example, “MyAmeba” can identify the Ameba device name and the name “MyAmeba” is changeable. “_arduino._tcp” is the protocol that Arduino IDE adopts, and the Domain is set as local in common.
Open the example, “`File`” -> “`Examples`” -> “`AmebaMDNS`” -> “`mdns_on_arduino_ide`”
You need to input ssid and password of the AP because the example will use WiFi connection.
And you can find out the naming of the service at the place where it declares MDNS Service. The example uses the default name “MyAmeba” and the name is changeable.



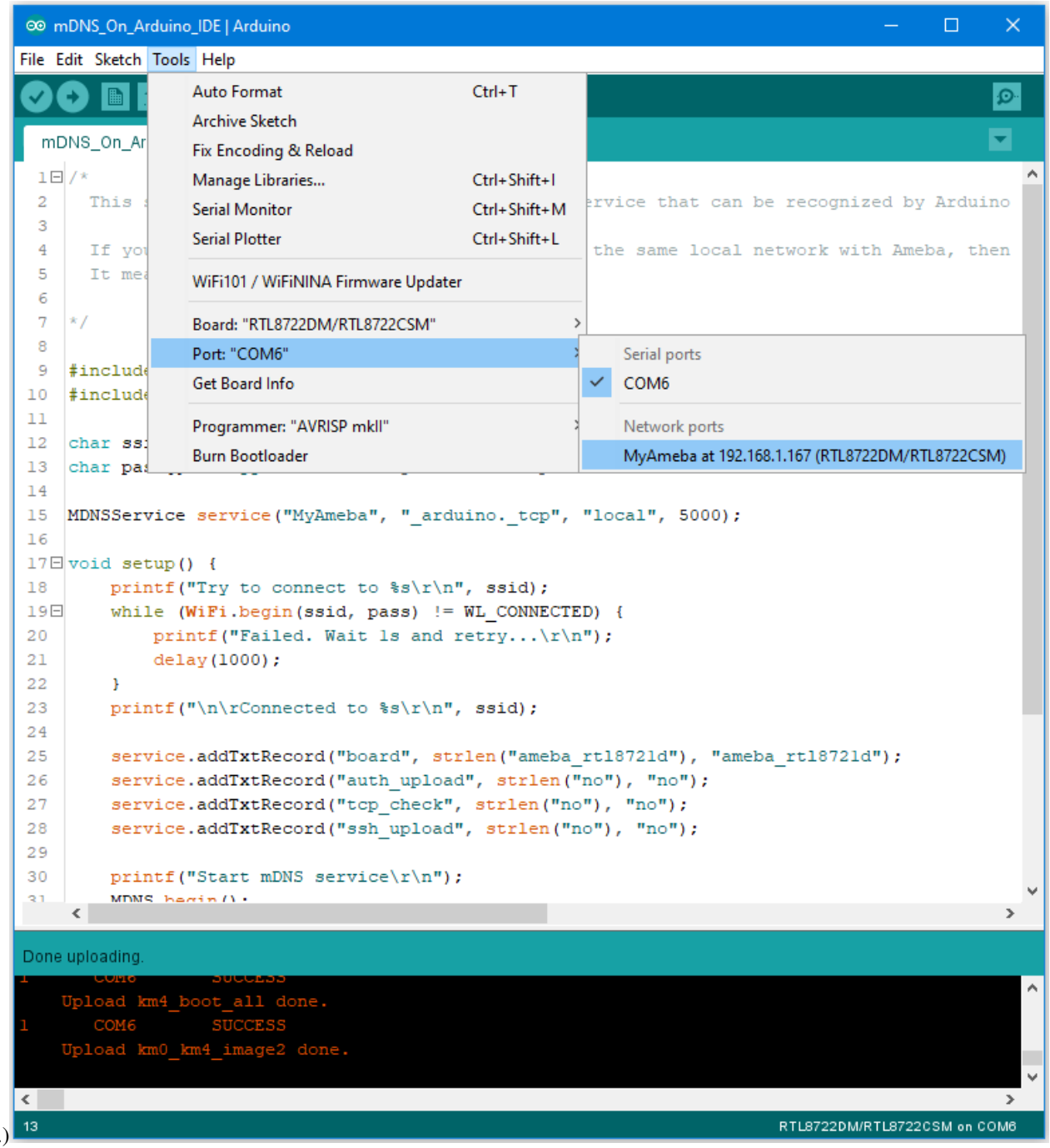
Next, go to ("Tools" -> "Port"), and you can find out at least one Serial Port. This port is simulated by Ameba board via USB. Choose this port and upload the compiled code to



Ameba. After uploading the code, press the reset button on Ameba and waiting for Ameba to connect with AP and activate the mDNS service after a while. You can see the Log at the bottom of the Serial Monitor.



Then you can find out the added item "Network Ports" **"MyAmeba at 192.168.1.167 (Ameba RTL8722DM/RTL8722CSM)"**, "MyAmeba" is the device name we set up, and "IP" is the IP address that AP assigned to Ameba, the IP address should be the same with the IP shown in the Serial Monitor. Last, "Ameba RTL8722DM/RTL8722CSM" is the type name of the board, and it means that Ameba can let Arduino IDE identify the mDNS service successfully.(We still can not use the Internet to upload the code, and we will explain this part in the OTA



example.)

If you cannot find the Network ports on your Arduino IDE, please check

- Does your computer in the same local area network with the Ameba?
- Restart the Arduino IDE, and it will find the mDNS service again
- Check the Log in Serial Monitor if the Ameba connects to the AP and activate mDNS service successfully

Code Reference

The program set up the mDNS service in the beginning, the first parameter is Instance Name, and it is changeable in this example. The second parameter is the protocol that the service used, and it would be "_arduino._tcp" for Arduino IDE. The third parameter is Domain, and it would be "local" in common. The fourth parameter is the port number for the service, it is 5000 here and we doesn't use it in the example.

```
MDNSService service("MyAmeba", "_arduino._tcp", "local", 5000);
```

After connected to the network, we set up some text fields for the service. For the following example, "board" is the name of the field, "ameba_rtl8721d" is the value of the field. "board" is used to let Arduino IDE check installed SDK to see if it exists known device or not. We will use the name of the device if there is known device, users can change "ameba_rtl8721d" to "yun" or other names to find out what's the difference if interested.

```
service.addTxtRecord("board", strlen("ameba_rtl8721d"),"ameba_rtl8721d");
```

Then we add three text fields "auth_upload", "tcp_check", and "ssh_upload", this example does not activate these services.

```
service.addTxtRecord("auth_upload", strlen("no"), "no");
service.addTxtRecord("tcp_check", strlen("no"), "no");
service.addTxtRecord("ssh_upload", strlen("no"), "no");
```

Next we activate MDNS

```
MDNS.begin();
```

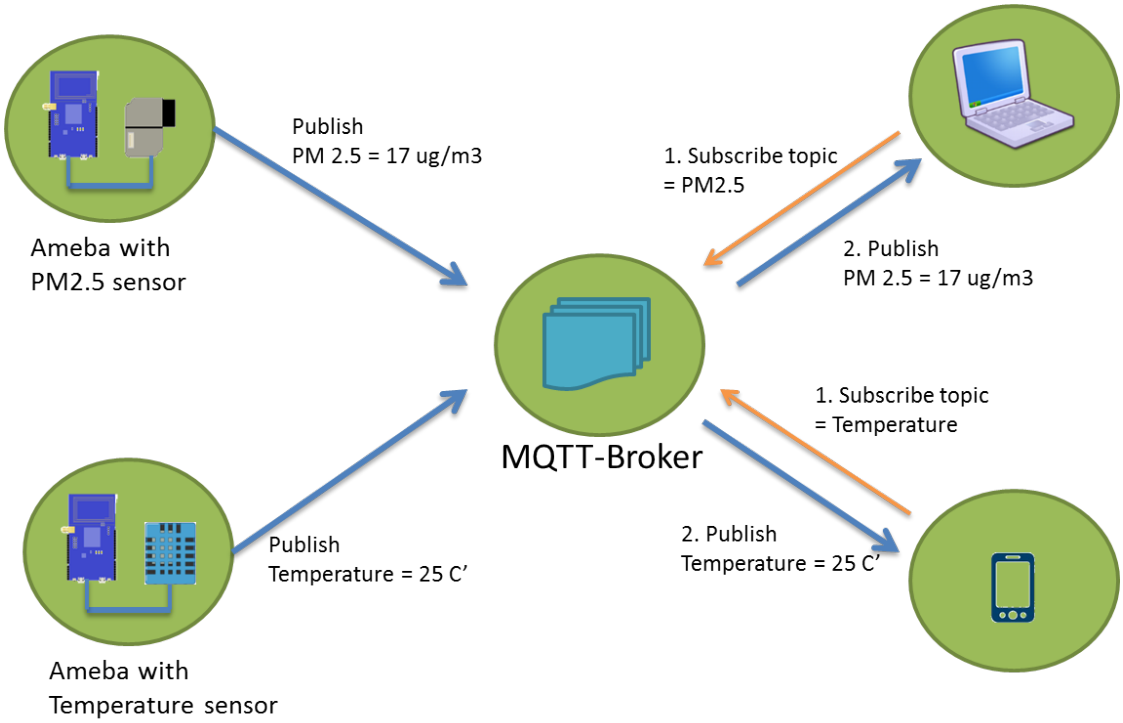
and register to the mDNS service.

```
MDNS.registerService(service);
```

MQTT - Set up MQTT Client to Communicate with Broker

Intro to MQTT

MQTT (Message Queuing Telemetry Transport) is a protocol proposed by IBM and Eurotech. The introduction in MQTT Official Website: MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. We can say MQTT is a protocol designed for IoT. MQTT is based on TCP/IP and transmits/receives data via publish/subscribe. Please refer to the figure below:



In the operation of MQTT, there are several roles:

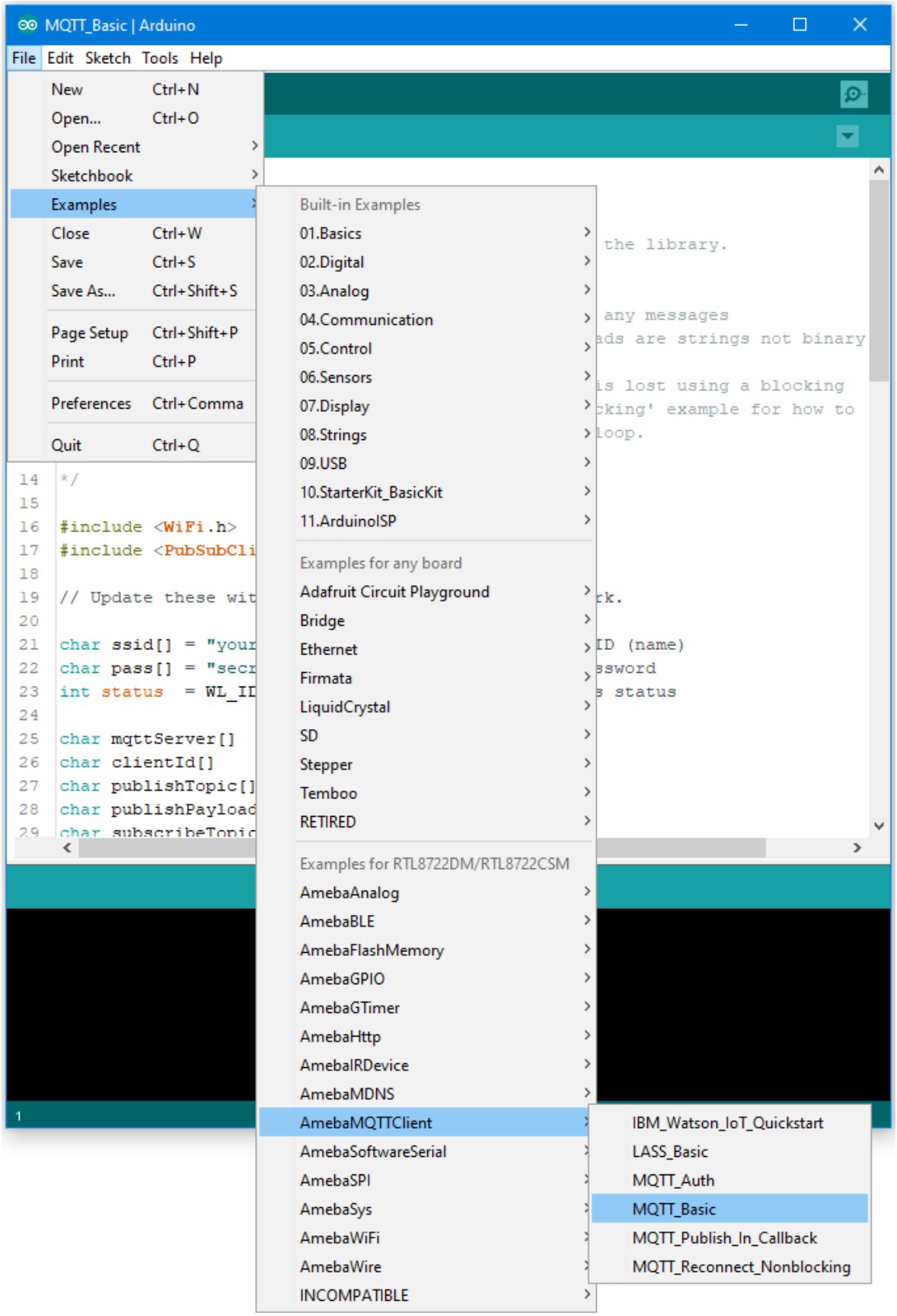
- **Publisher**: Usually publishers are the devices equipped with sensors (ex. Ameba). Publishers uploads the data of the sensors to MQTT-Broker, which serves as a database with MQTT service.
- **Subscriber**: Subscribers are referred to the devices which receive and observe messages, such as a laptop or a mobile phone.
- **Topic**: Topic is used to categorized the messages, for example the topic of a message can be "PM2.5" or "Temperature". Subscribers can choose messages of which topics they want to receive.

Preparation

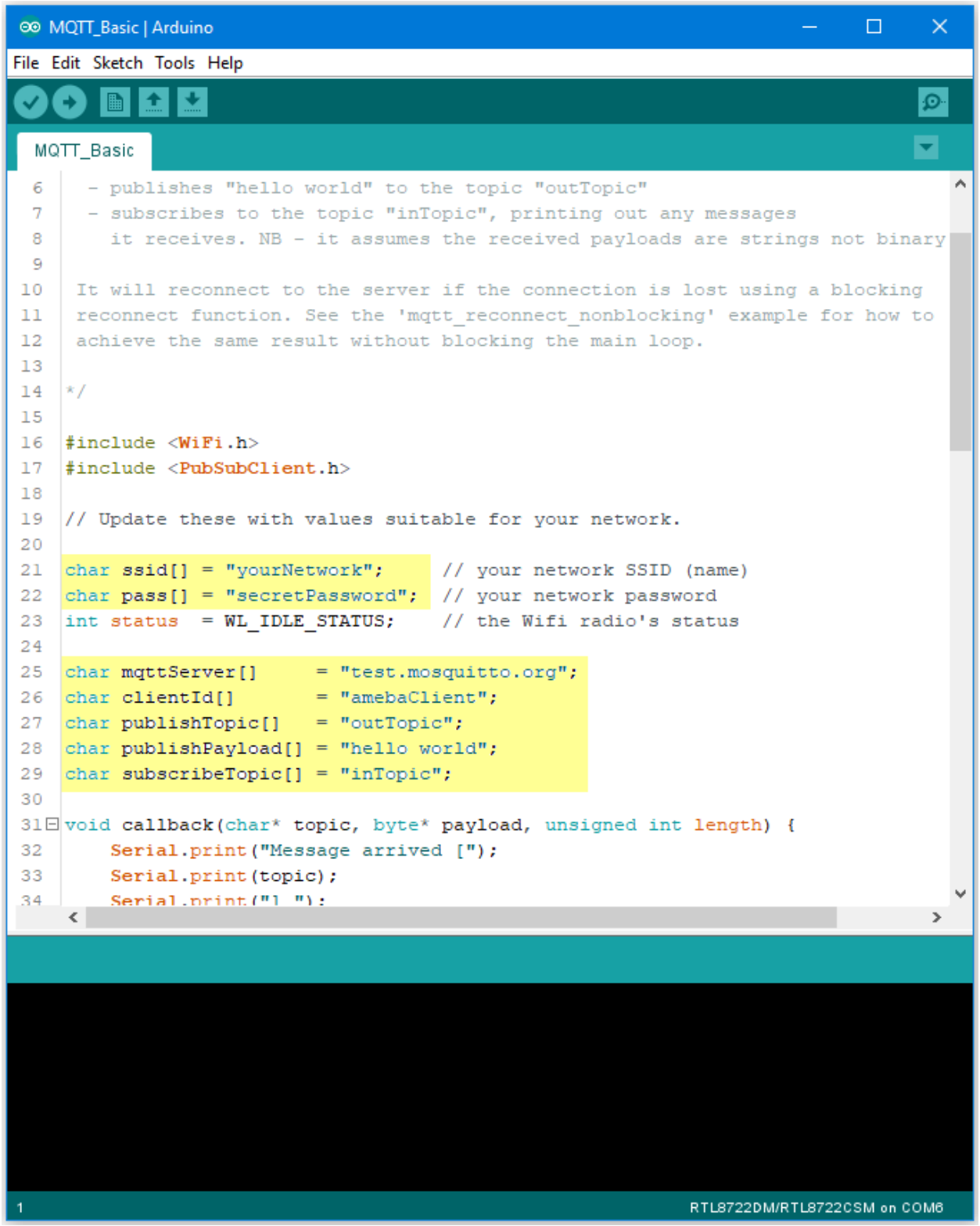
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to MQTT-Broker. Then send messages as publisher and receive messages from MQTT-Broker as subscriber.
Open the MQTT example "`File" -> "Examples" -> "AmebaMQTTClient" -> "MQTT_Basic`"


MQTT_Basic | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking

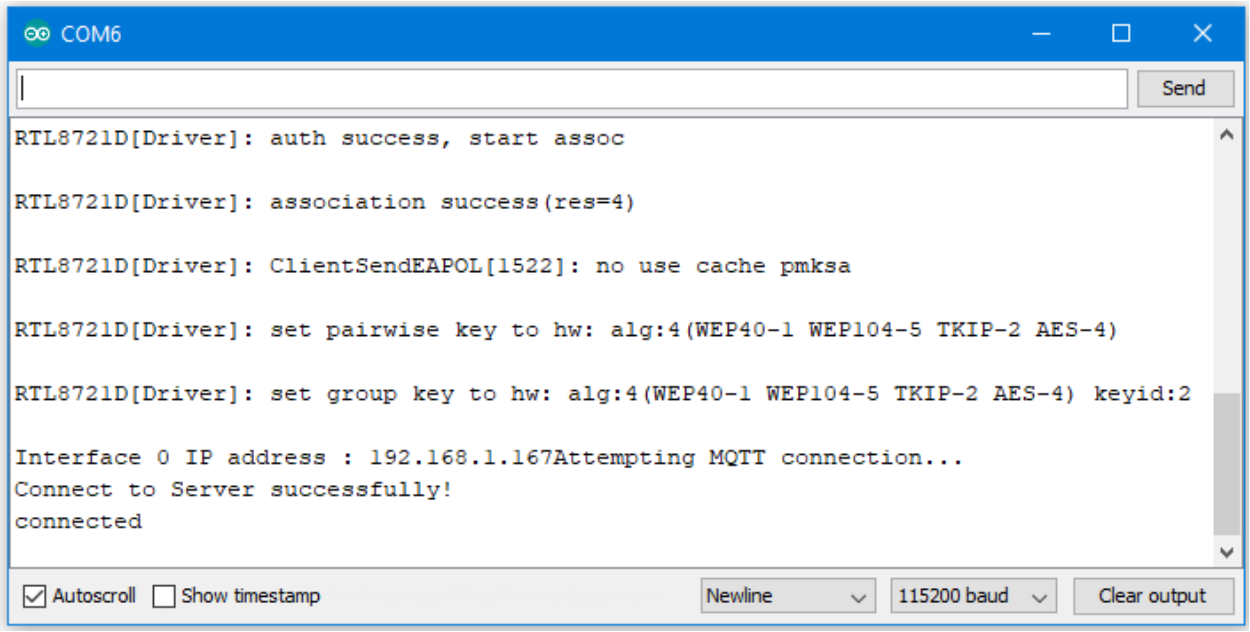
Please modify some WiFi-related parameters.
And some information related to MQTT:



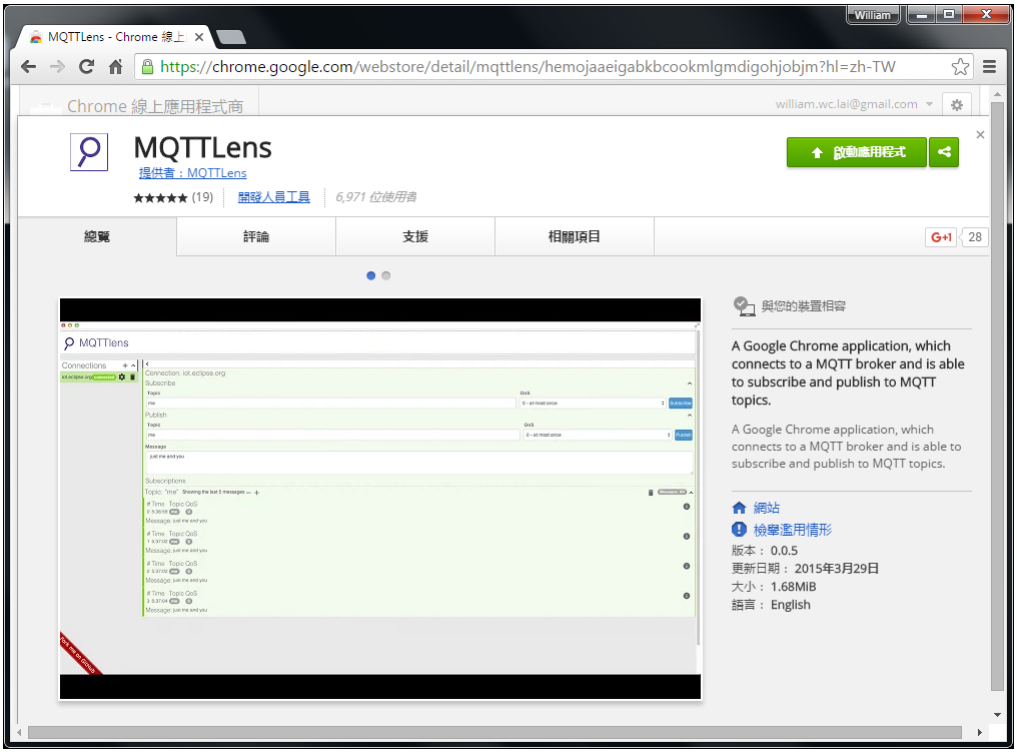
The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.

- “clientId” is an identifier for MQTT-Broker to identify the connected device.
- “publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
- “publishPayload” is the content to be published.
- “subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor



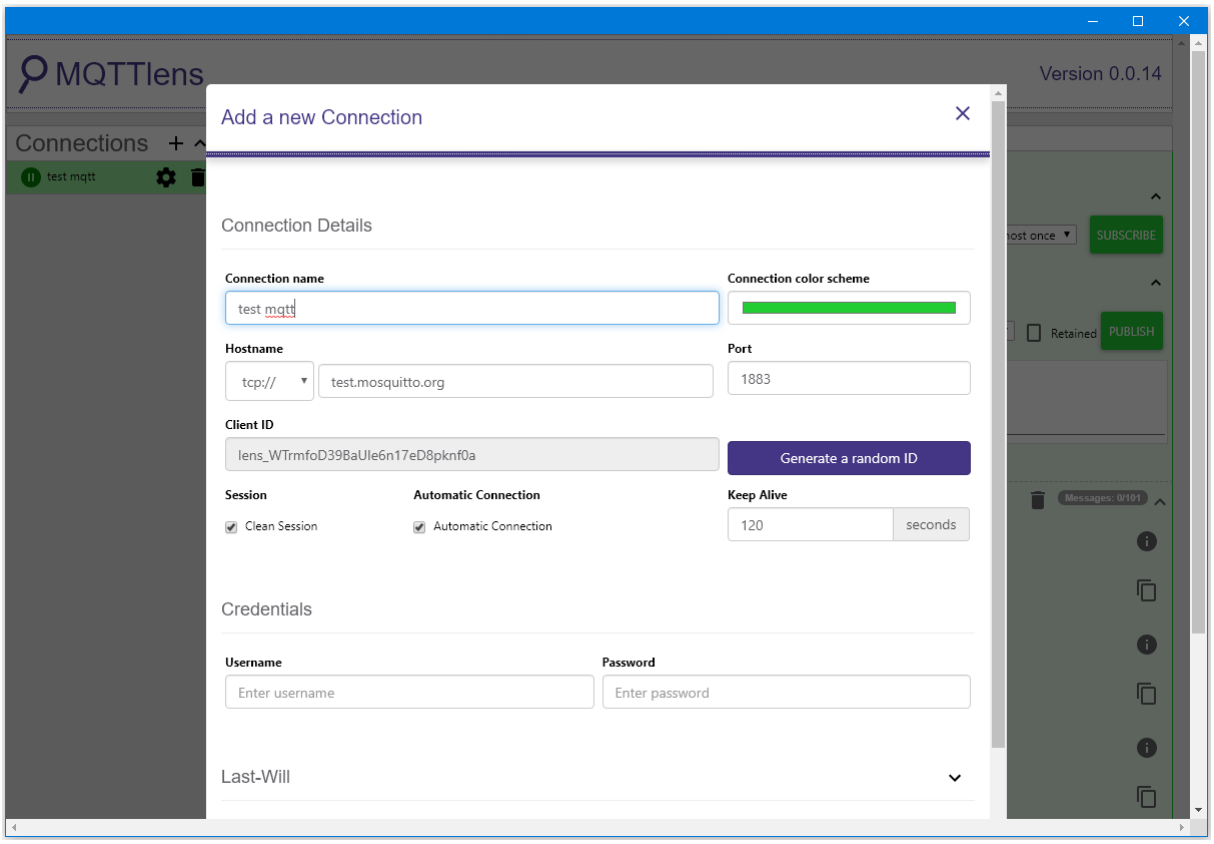
After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”.
To see the message, we need another MQTT client.
Here we use a chrome plugin “MQTTLens” to be the MQTT client. You can find it in google webstore.



Install and open the MQTTLens, click “+” next to “Connection” on the left, and fill in the required information

- **Connection Name**: Used to identify the connection, you can choose a name you like.
- **Hostname**: The MQTT-Broker server, here we use “iot.eclipse.org”
- **Client ID**: We use the default randomly generated ID.

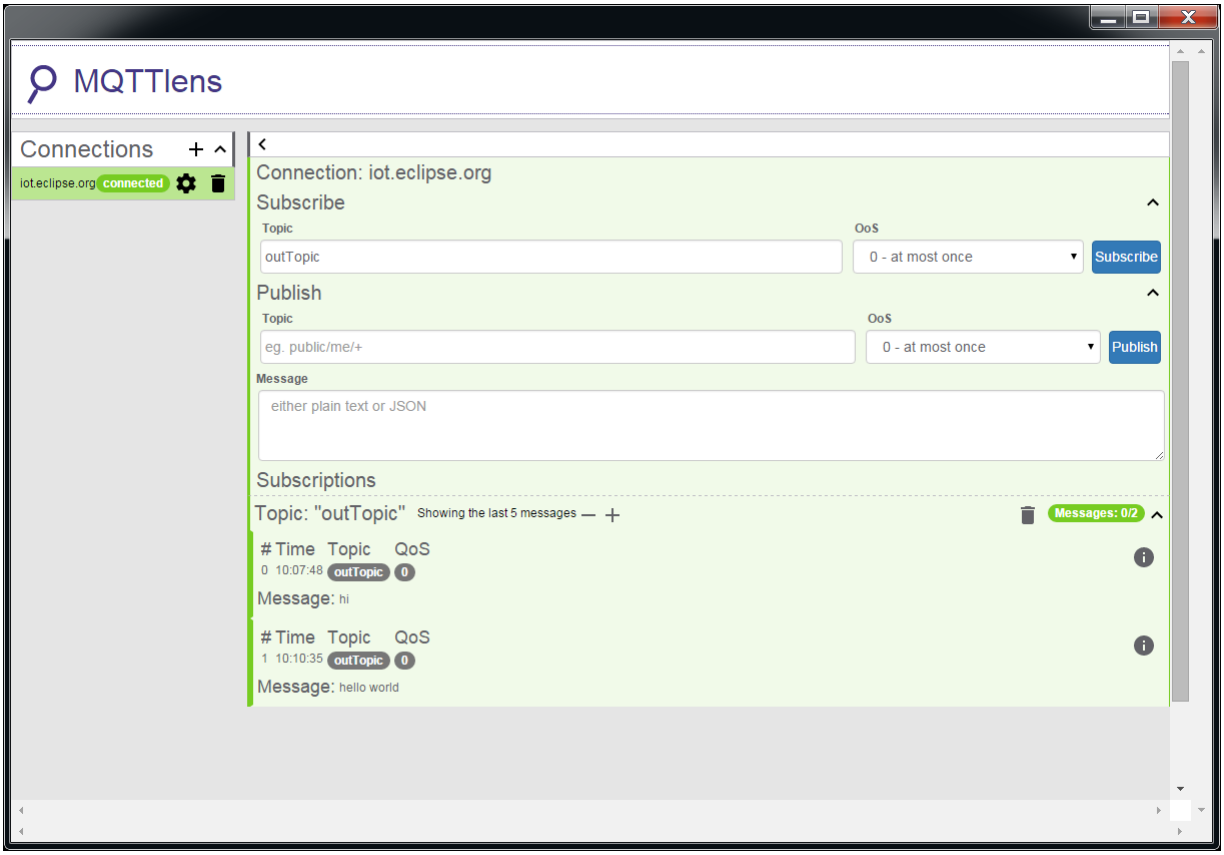
Then click “CREATE CONNECTION”.



Since we have not registered the topic we want to listen to, we would not receive any messages now.

Fill in “outTopic” in the “Topic” field and click “Subscribe”.

Wait for Ameba to send next message (or you can press the reset button). Then you can see the “hello world” message show up.



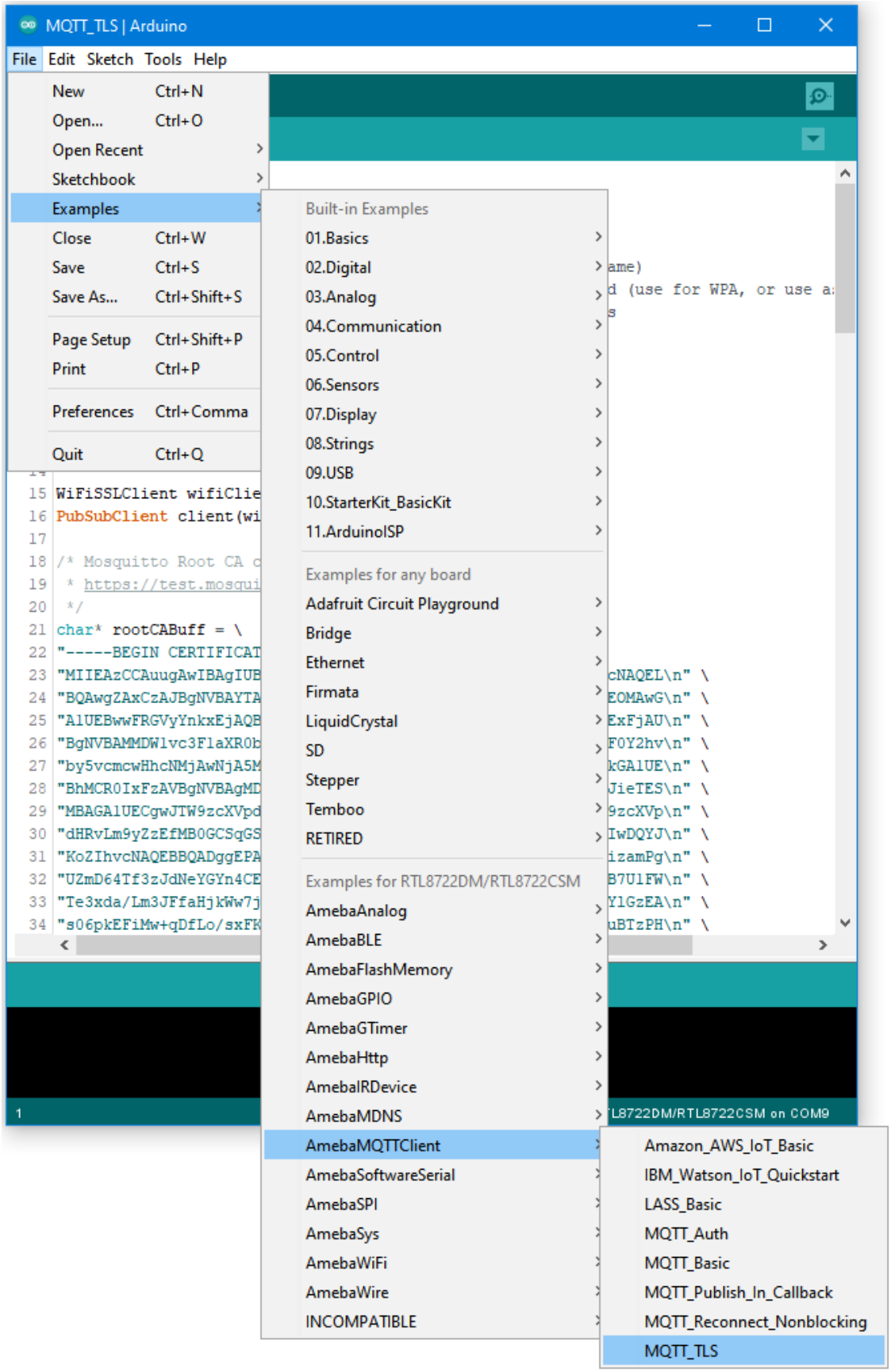
MQTT - Set up MQTT Client over TLS

Preparation

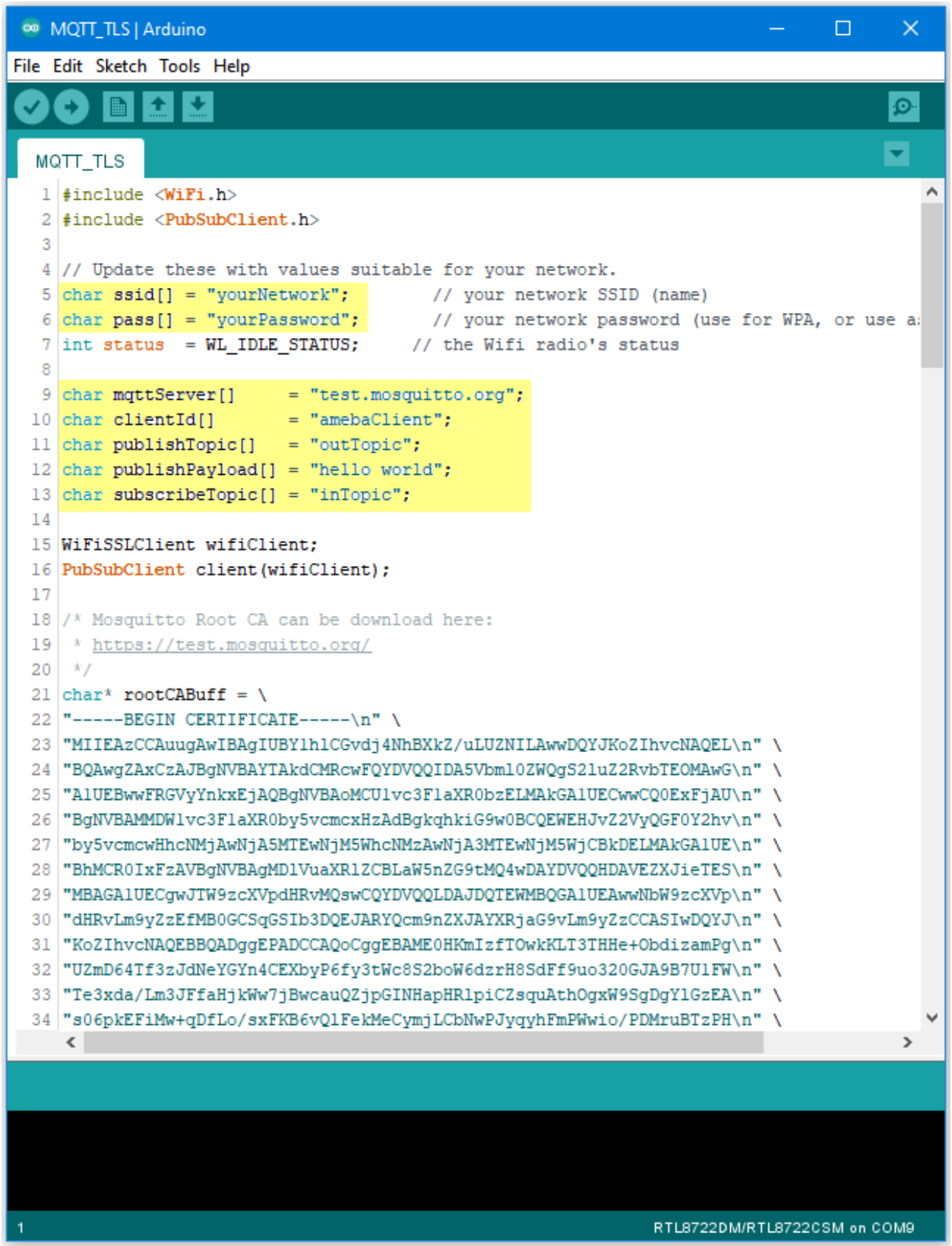
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to a MQTT broker using TLS authentication. Then send messages as a publisher and receive messages from as a subscriber. Open the MQTT example "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`MQTT_TLS`"

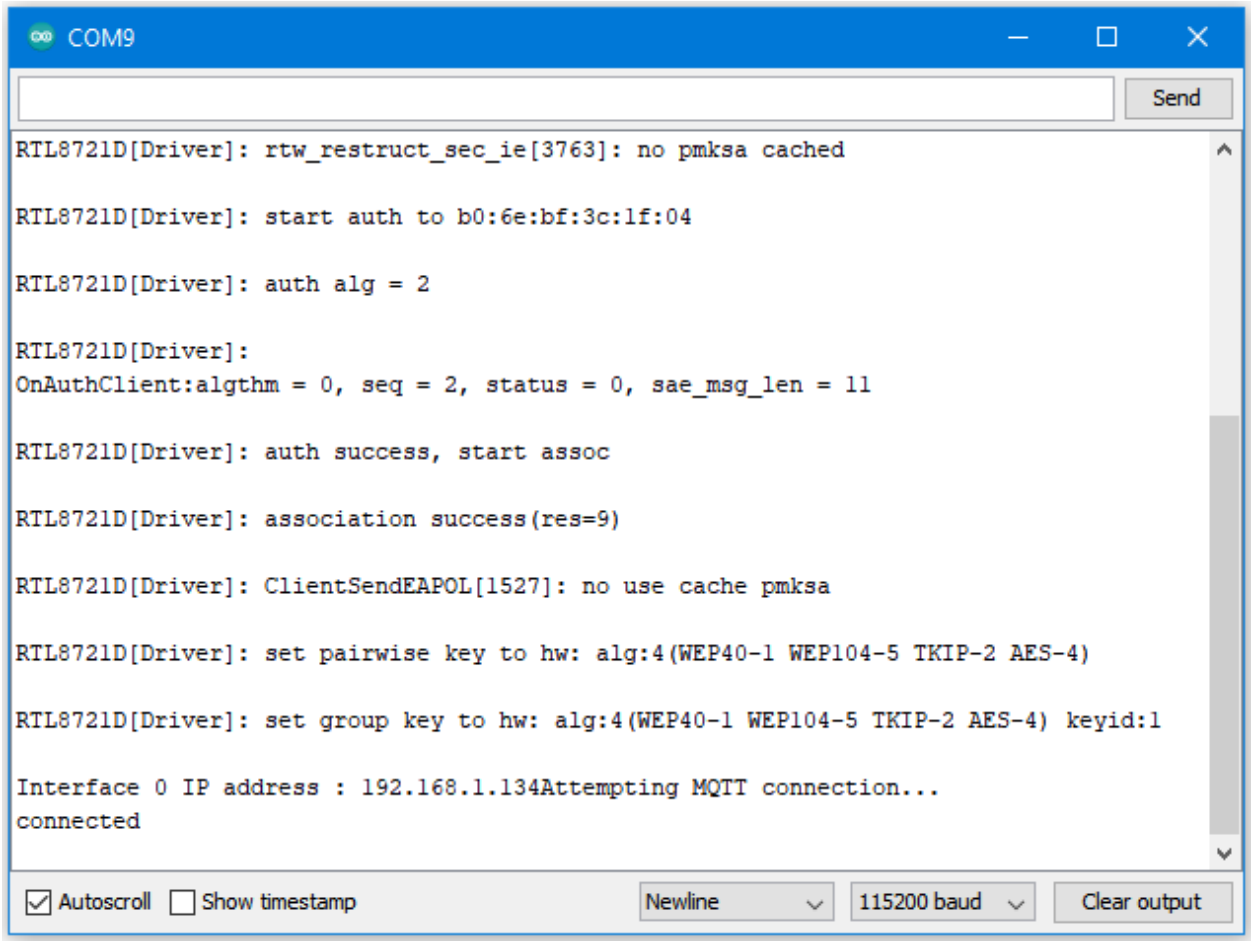

MQTT_TLS | Arduino
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for RTL8722DM/RTL8722CSM
AmebaAnalog
AmebaBLE
AmebaFlashMemory
AmebaGPIO
AmebaGTimer
AmebaHttp
AmebaIRDevice
AmebaMDNS
AmebaMQTTClient
AmebaSoftwareSerial
AmebaSPI
AmebaSys
AmebaWiFi
AmebaWire
INCOMPATIBLE
Amazon_AWS_IoT_Basic
IBM_Watson_IoT_Quickstart
LASS_Basic
MQTT_Auth
MQTT_Basic
MQTT_Publish_In_Callback
MQTT_Reconnect_Nonblocking
MQTT_TLS

Please modify the WiFi-related parameters to connect to your WiFi network.
Modify the MQTT parameters to fit your application:



The “mqttServer” refers to the MQTT-Broker, we use the free MQTT sandbox “test.mosquitto.org” for testing.
“clientId” is an identifier for MQTT-Broker to identify the connected device.
“publishTopic” is the topic of the published message, we use “outTopic” in the example. The devices subscribe to “outTopic” will receive the message.
“publishPayload” is the content to be published.
“subscribeTopic” is to tell MQTT-broker which topic we want to subscribe to.

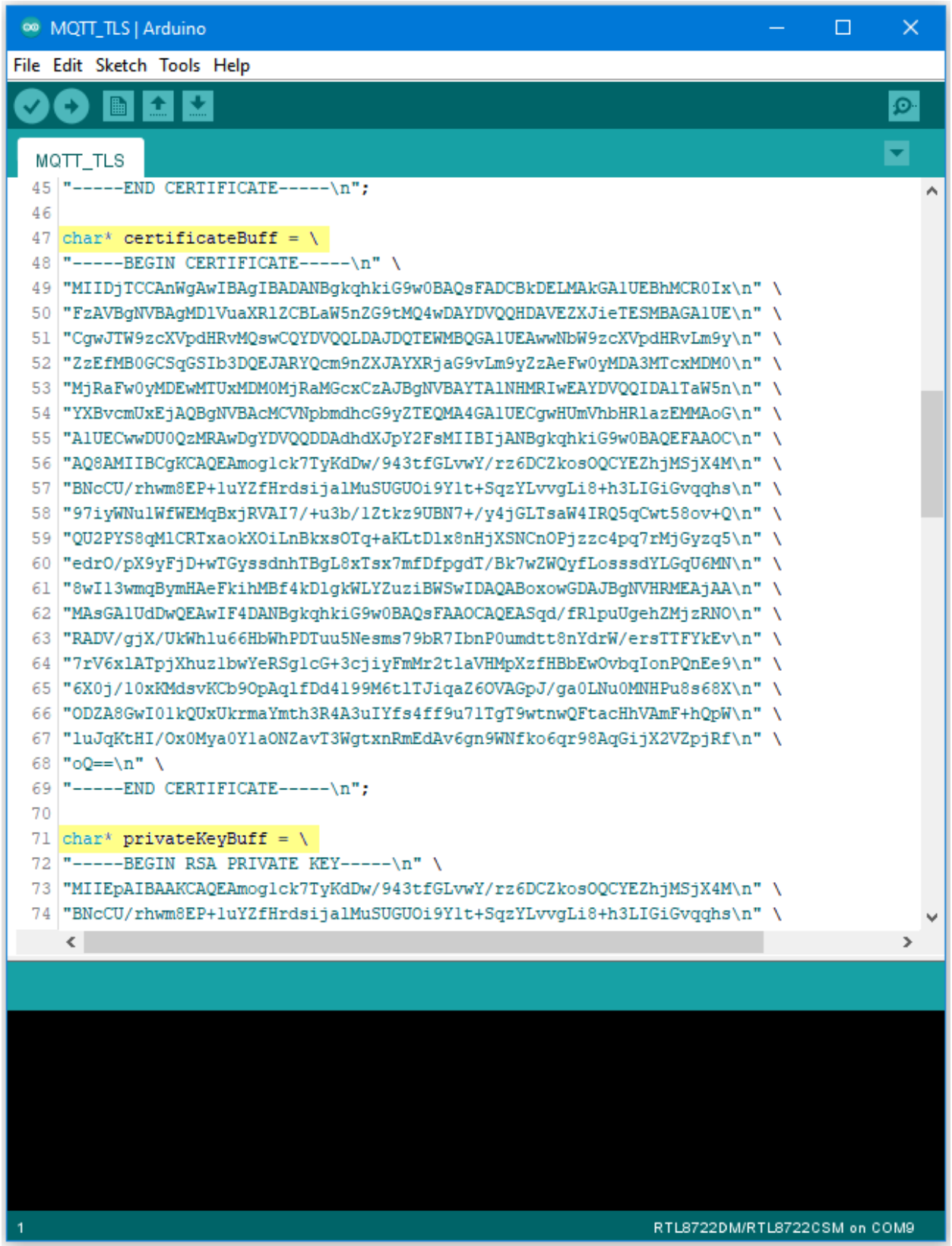
Next, compile the code and upload it to Ameba. Press the reset button, then open the serial monitor

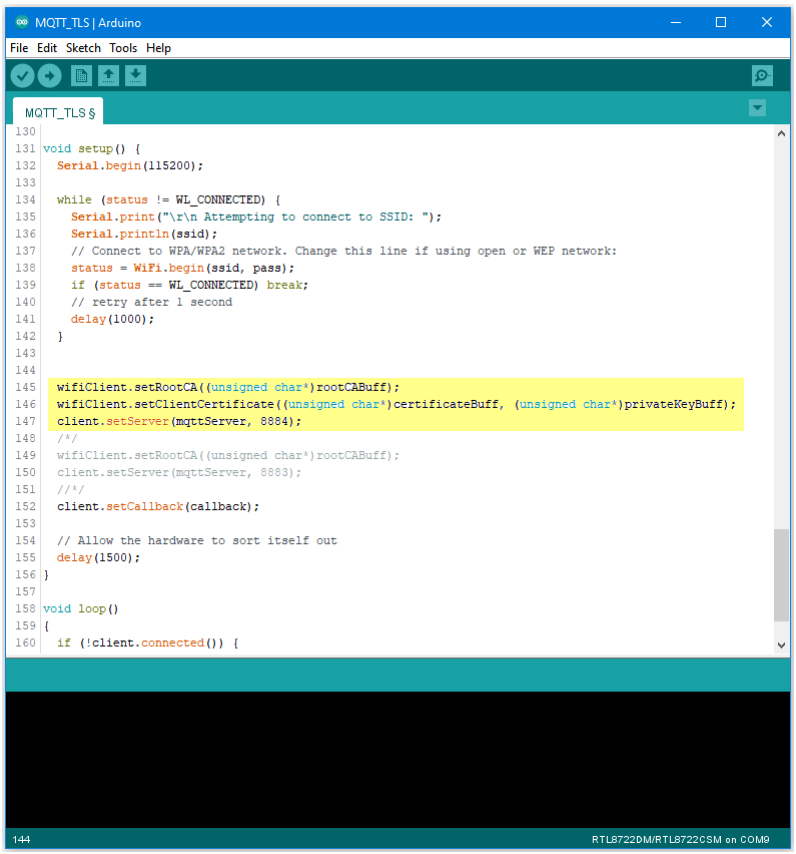


After Ameba is connected to MQTT server, it sends the message “hello world” to “outTopic”. To see the message, use another MQTT client. Refer to the MQTT_Basic example guide on how to setup a PC-based MQTT client.

If you wish to use TLS client authentication in addition to server authentication, you will need to generate an OpenSSL private key and obtain a signed certificate from the server. For testing purposes, signed certificates can be obtained from test.mosquitto.org by following the guide at https://test.mosquitto.org/ssl/.

Replace the character strings “certificateBuff” and “privateKeyBuff” with your signed certificate and OpenSSL private key, ensuring that they are formatted the same way as the shown in the example code. Also uncomment the highlighted code to enable client authentication, and to change the MQTT port number.


MQTT_TLS | Arduino
File Edit Sketch Tools Help
MQTT_TLS
"-----END CERTIFICATE-----\n";
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDjTCCAnWgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix\n" \
"FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE\n" \
"CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y\n" \
"ZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzAeFw0yMDA3MTcxMDM0\n" \
"MjRaFw0yMDEwMTUxMDM0MjRaMGcxCzAJBgNVBAYTAlNHMRIwEAYDVQQIDAlTaW5n\n" \
"YXBvcmUxEjAQBgNVBAcMCVNpbmdhcG9yZTEQMA4GA1UECgwHUmVhbHRlazEMMAoG\n" \
"A1UECwwDU0QzMRAwDgYDVQQDDAdhdXJpY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOC\n" \
"AQ8AMIIBCgKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
"97iyWNulWfWEMqBxjRVAI7/+u3b/lZtkz9UBN7+/y4jGLTsaW4IRQ5qCwt58ov+Q\n" \
"QU2PYS8qMlCRTxaokXOiLnBkxsOTq+aKLtDlx8nHjXSNCnOPjzzc4pq7rMjGyzq5\n" \
"edrO/pX9yFjD+wTGyssdnhTBgL8xTsx7mfDfpgdT/Bk7wZWQyfLosssdYLGqU6MN\n" \
"8wIl3wmqBymHAeFkihMBf4kDlgkWLYZuziBWSwIDAQABoxowGDAJBgNVHRMEAjAA\n" \
"MAsGA1UdDwQEAwIF4DANBgkqhkiG9w0BAQsFAAOCAQEASqd/fRlpuUgehZMjzRNO\n" \
"RADV/gjX/UkWhlu66HbWhPDTuu5Nesms79bR7IbnP0umdtt8nYdrW/ersTTFYkEv\n" \
"7rV6xlATpjXhuzlbwYeRSglcG+3cjiyFmMr2tlaVHMpXzfHBbEwOvbqIonPQnEe9\n" \
"6X0j/l0xKMdsvKCb9OpAqlfDd4199M6tlTJiqaZ6OVAGpJ/ga0LNu0MNHPu8s68X\n" \
"ODZA8GwI0lkQUxUkrmaYmth3R4A3uIYfs4ff9u7lTgT9wtnwQFtacHhVAmF+hQpW\n" \
"luJqKtHI/Ox0Mya0YlaONZavT3WgtxnRmEdAv6gn9WNfko6qr98AqGijX2VZpjRf\n" \
"oQ==\n" \
"-----END CERTIFICATE-----\n";
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEpAIBAAKCAQEAmoglck7TyKdDw/943tfGLvwY/rz6DCZkosOQCYEZhjMSjX4M\n" \
"BNcCU/rhwm8EP+luYZfHrdsijalMuSUGUOi9Ylt+SqzYLvvgLi8+h3LIGiGvqqhs\n" \
1
RTL8722DM/RTL8722CSM on COM9



MQTT - Use Amazon AWS IoT Shadow Service

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

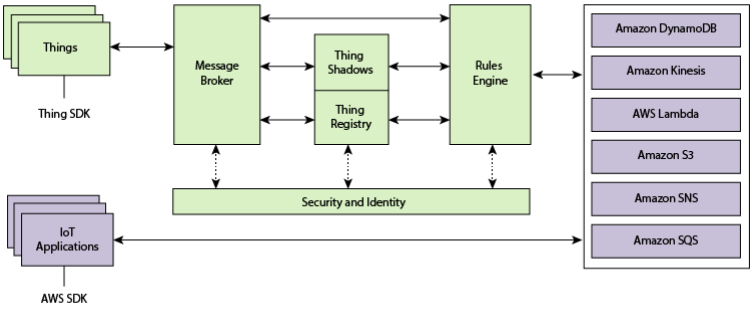
Example

Introduction

Amazon AWS IoT is a cloud IoT service platform:

Amazon AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline. (https://aws.amazon.com/iot/how-it-works/)

The service architecture of AWS IoT:

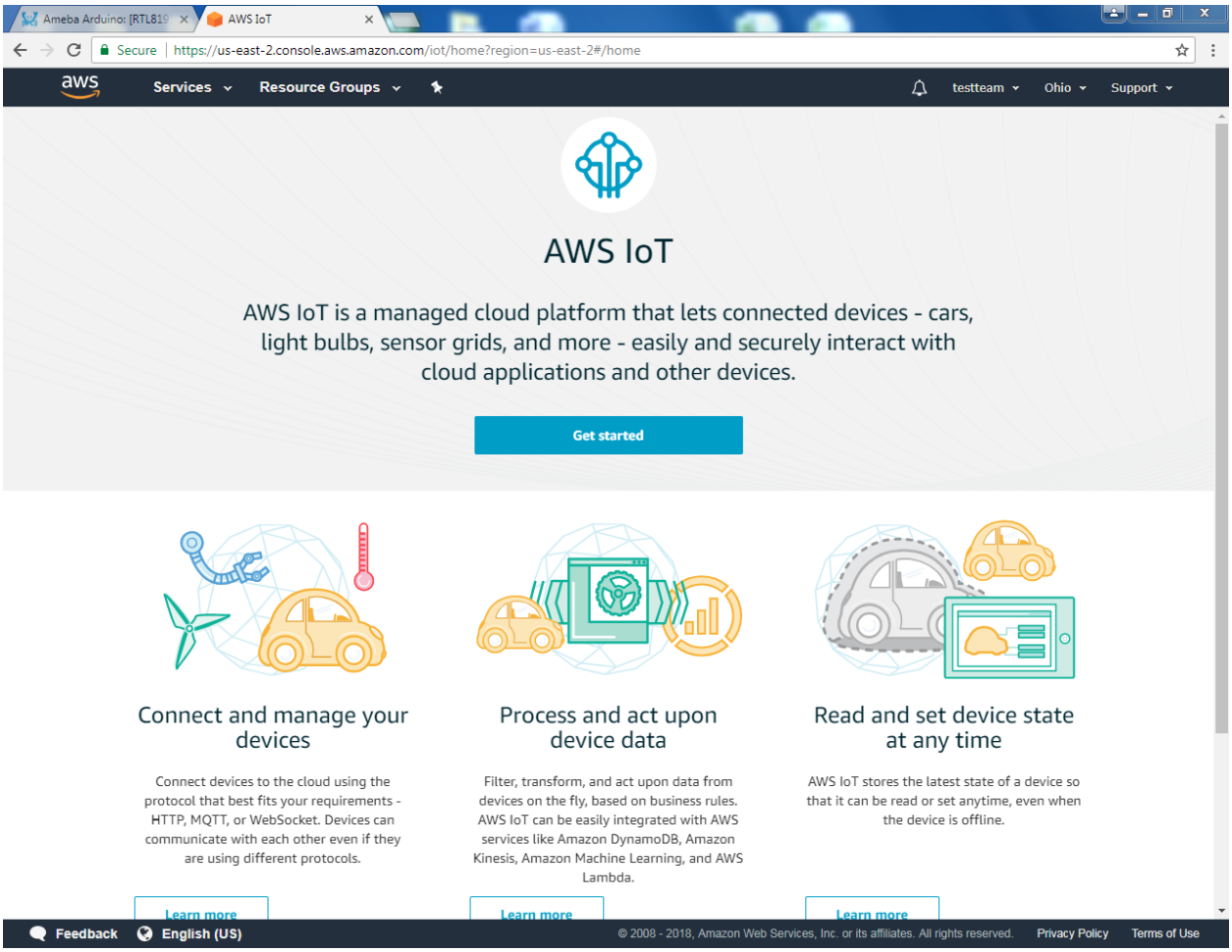


(Picture from http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html)
In the architecture, Ameba belongs to the upper-left “Things” block. A TLS secure channel will be established between “Things” and the MQTT Message Broker. Afterwards, “Things” and “Message Broker” communicate using MQTT Protocol via this secure channel. Behind the “Message Broker”, the “Thing Shadows” keeps messages temporarily when Ameba is offline, and sends the control message to Ameba next time it is connected. The “Rules Engine” allows you to place restrictions to the behavior of Things or to connect Things to other services of Amazon.

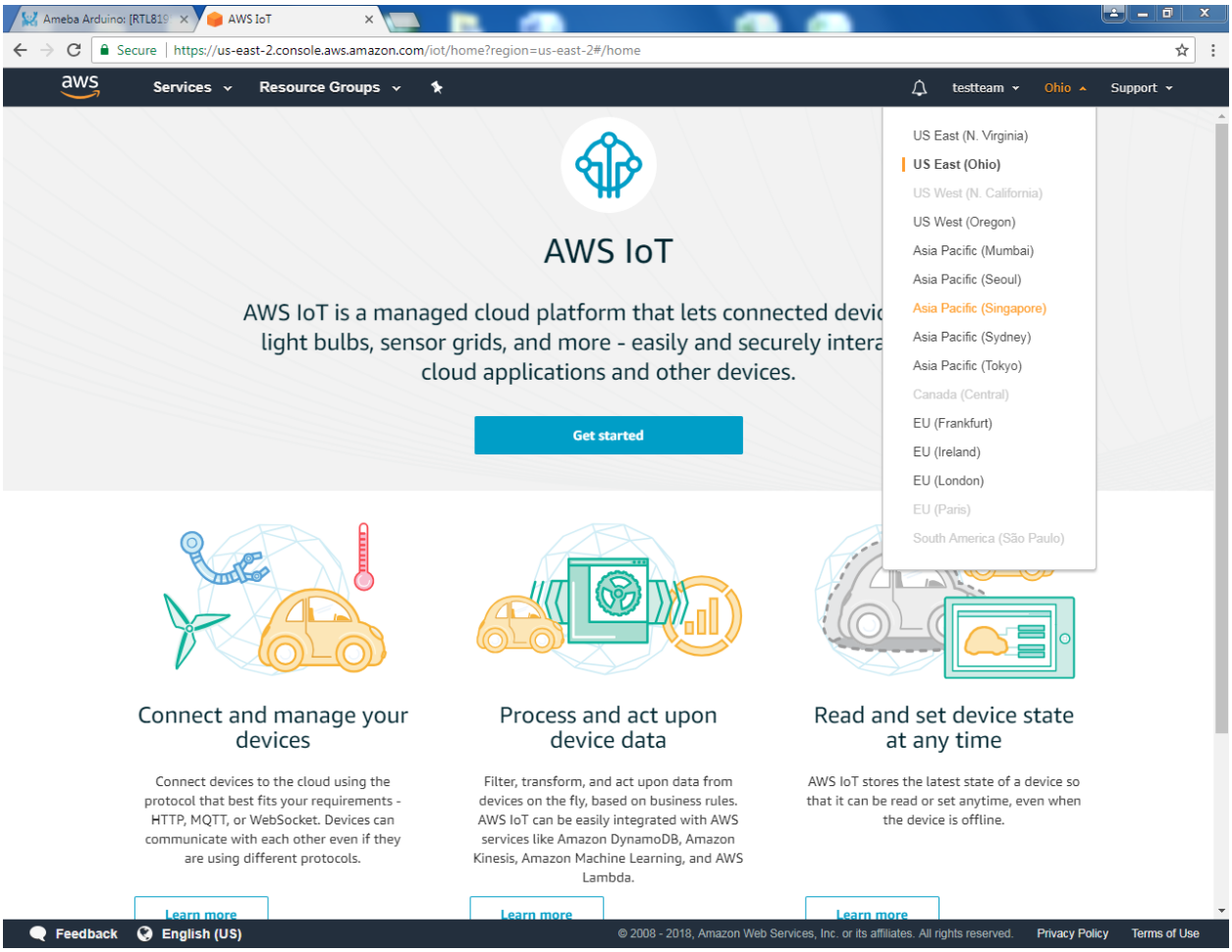
AWS Management Console

First, create an account and sign up for AWS IoT service:https://aws.amazon.com/
Afterwards, log in to the Amazon Management Console and click “IoT Core” found under services -> Internet of Things.

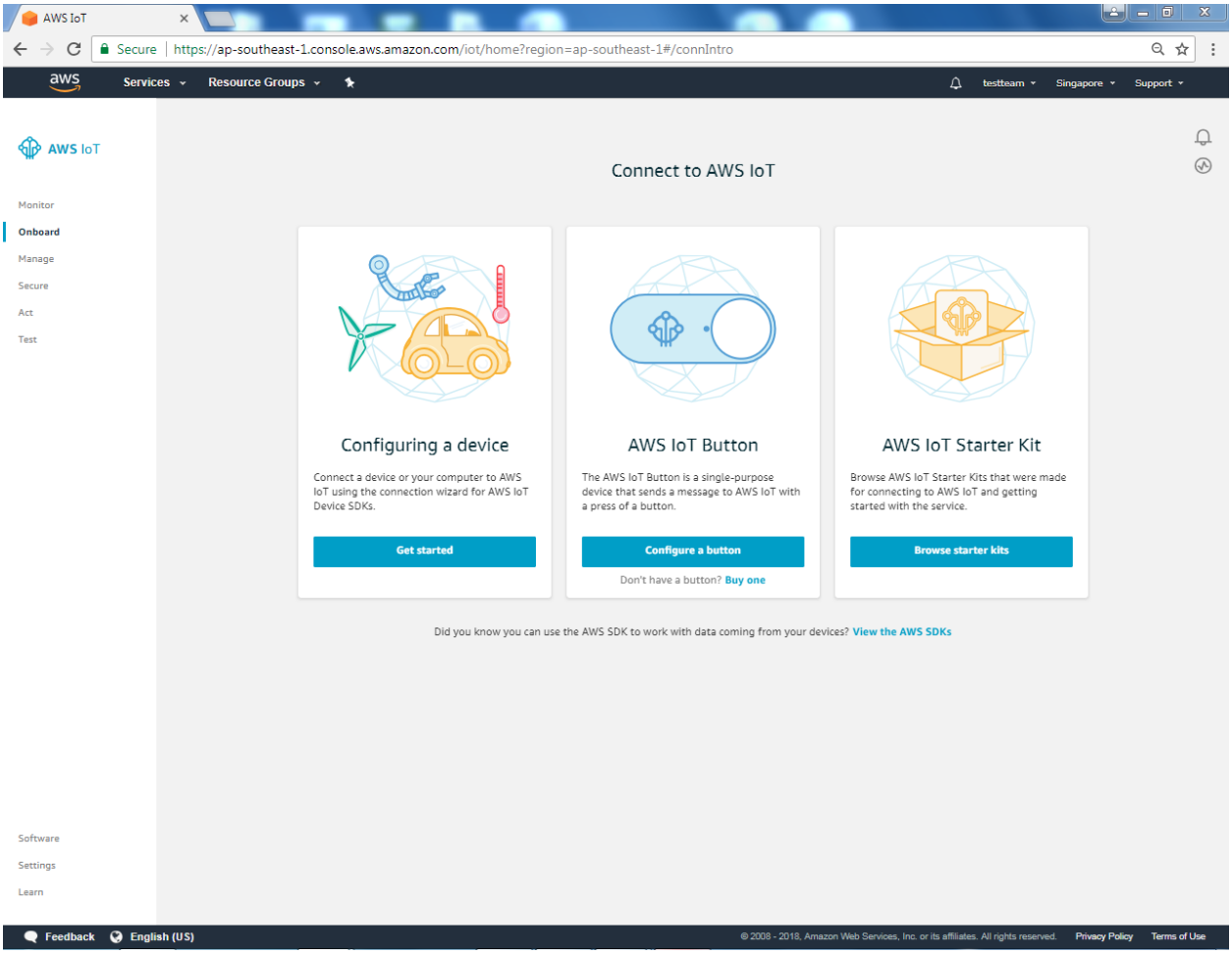
Then you will enter the home page of AWS IoT. To offer the best service quality, Amazon offers servers in different regions for users to choose from.
Click the region dropdown menu at the upper-right:



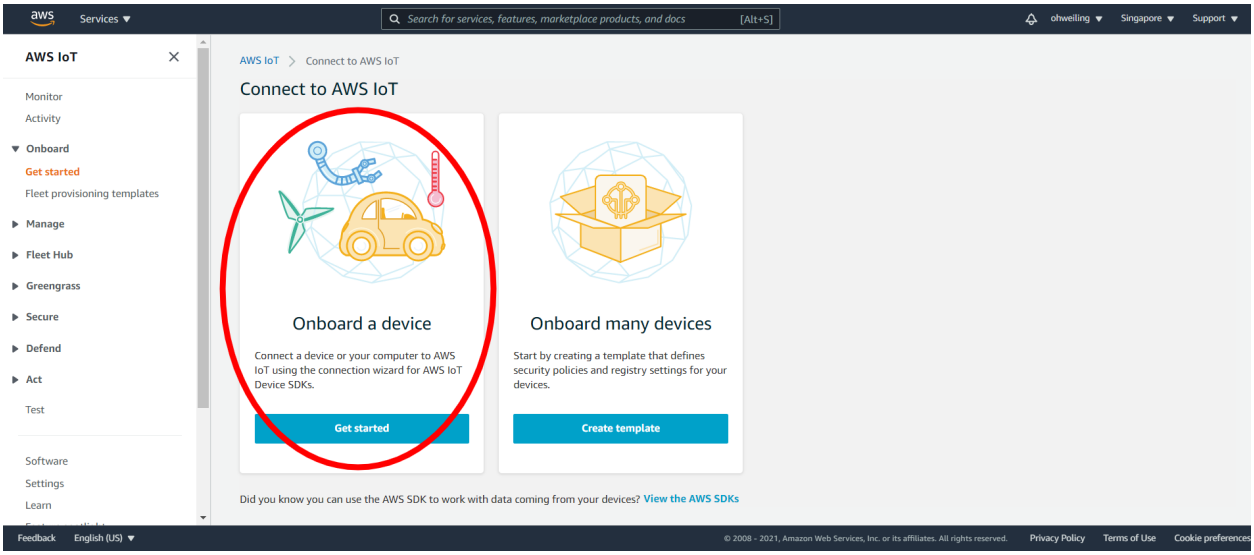
Choose a nearby region.



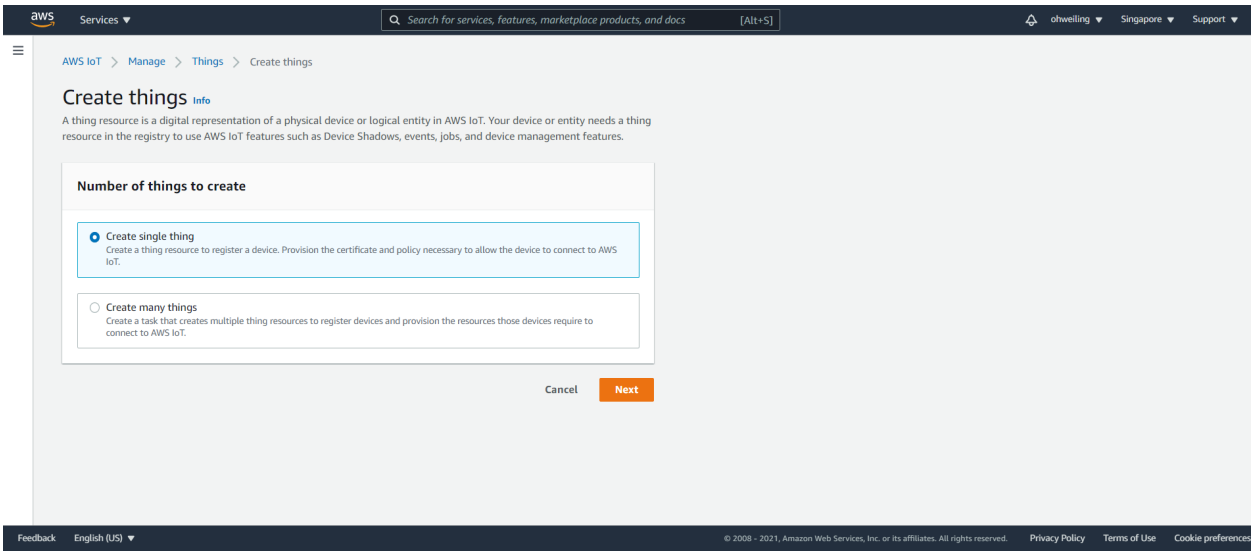
Then from Services, go to Onboard then Get Started.



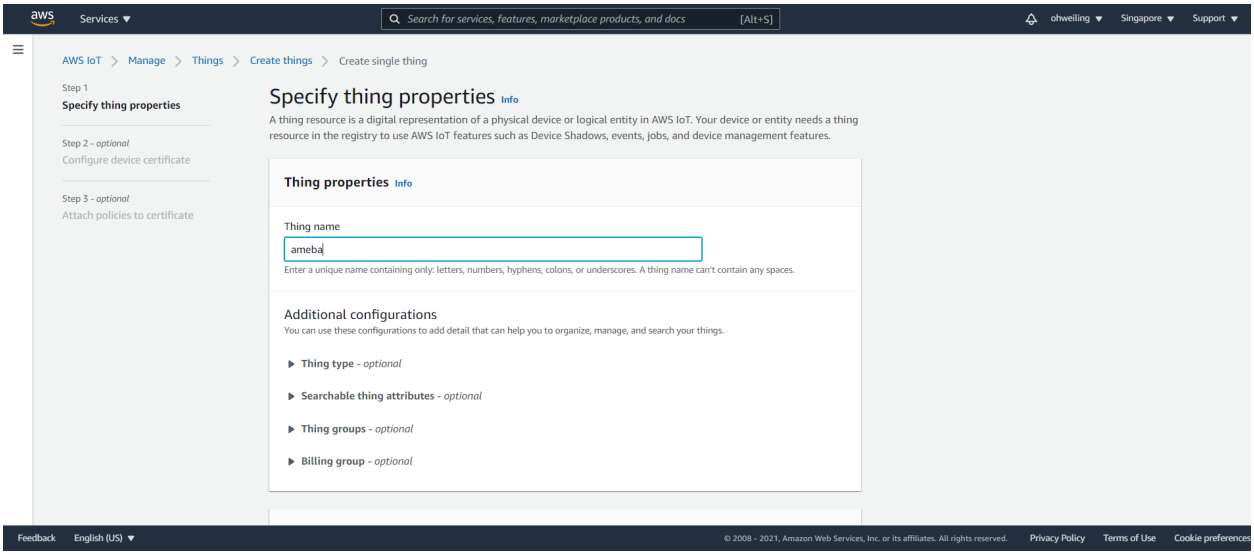
Enter the main page of AWS IoT. Under the Onboard a device, click Get started.



Click Create single thing

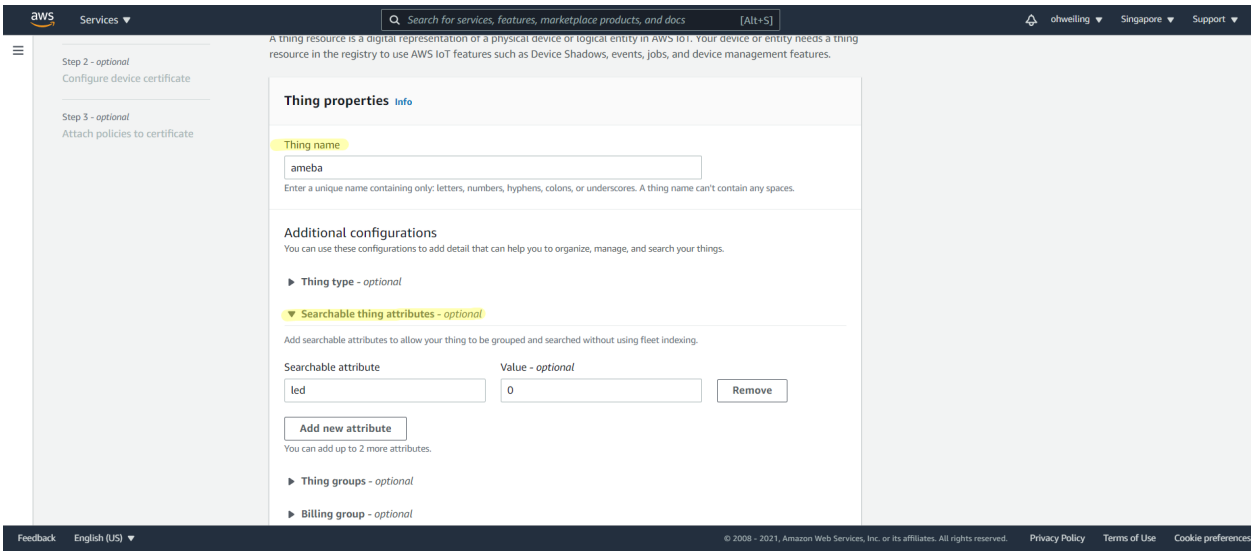


Fill in “ameba” on the name field. Attributes represent the status of Ameba.

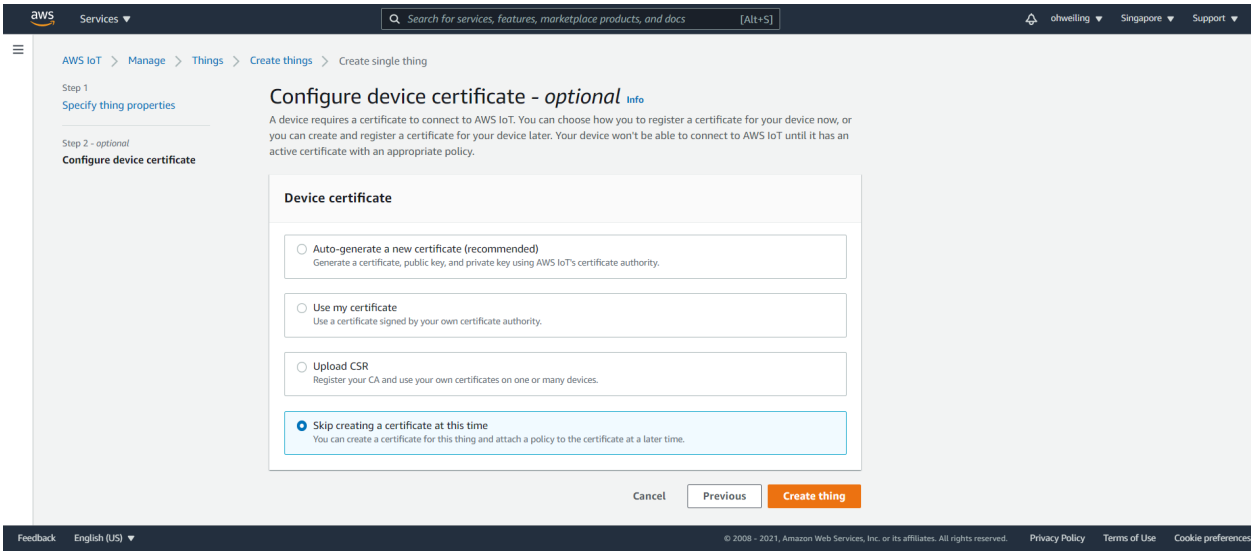


Under the searchable thing attributes. The value of the attributes can be updated directly by Ameba or by the control side and control side can request Ameba to set the attribute to desired value.

Here we add an attribute named “led” with value “0” and click “Next”.



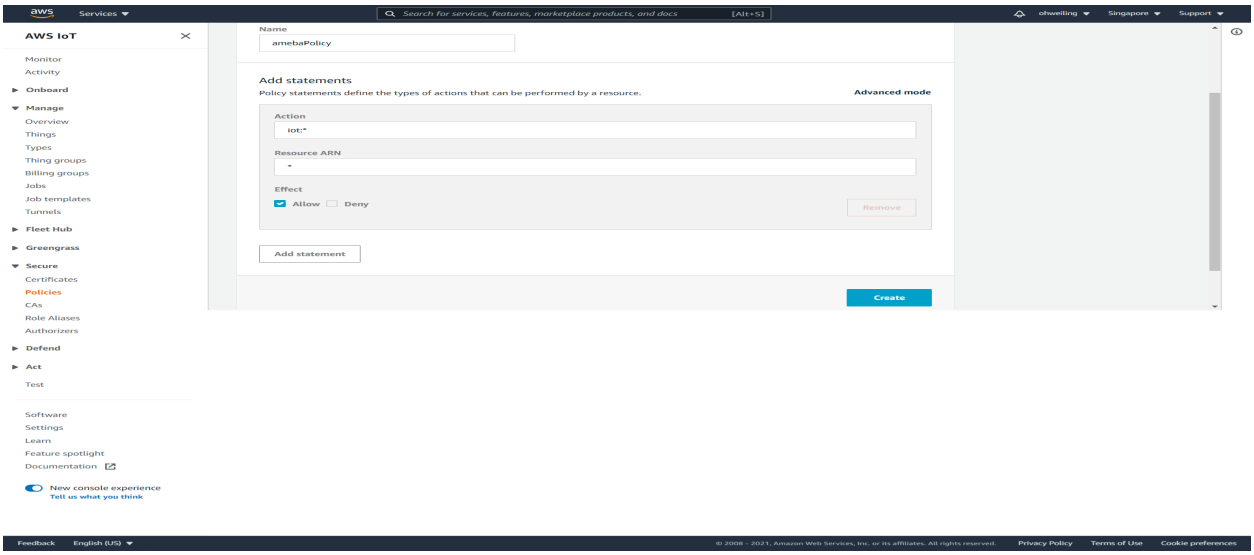
Click Skip creating a certificate at this time and then Create thing



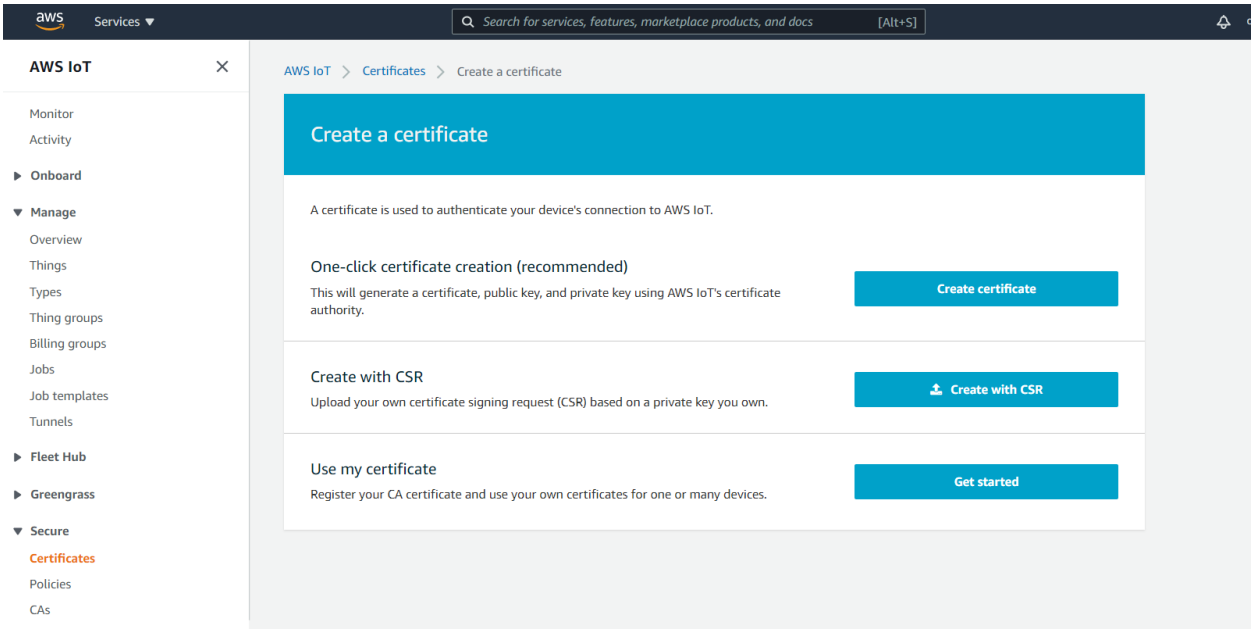
Next, click Policy¸ and create a policy. Policy is used to restrict the functions that a “thing” can do, it can limit the MQTT actions or specific topic that can be performed. Learn more about policy:

http://docs.aws.amazon.com/iot/latest/developerguide/authorization.html

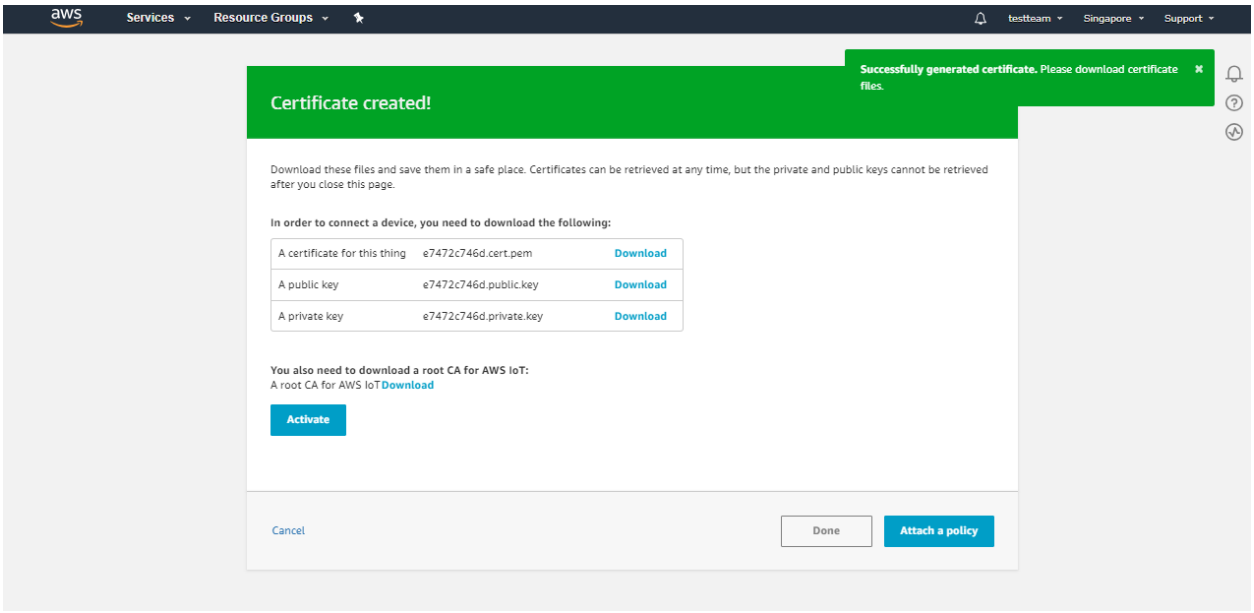
Here we do not place policy on Ameba. Fill in “amebaPolicy” in the Name field, “iot:” *in Action field and* “” in resources field. Then “Allow”. Finally, click “Create”.



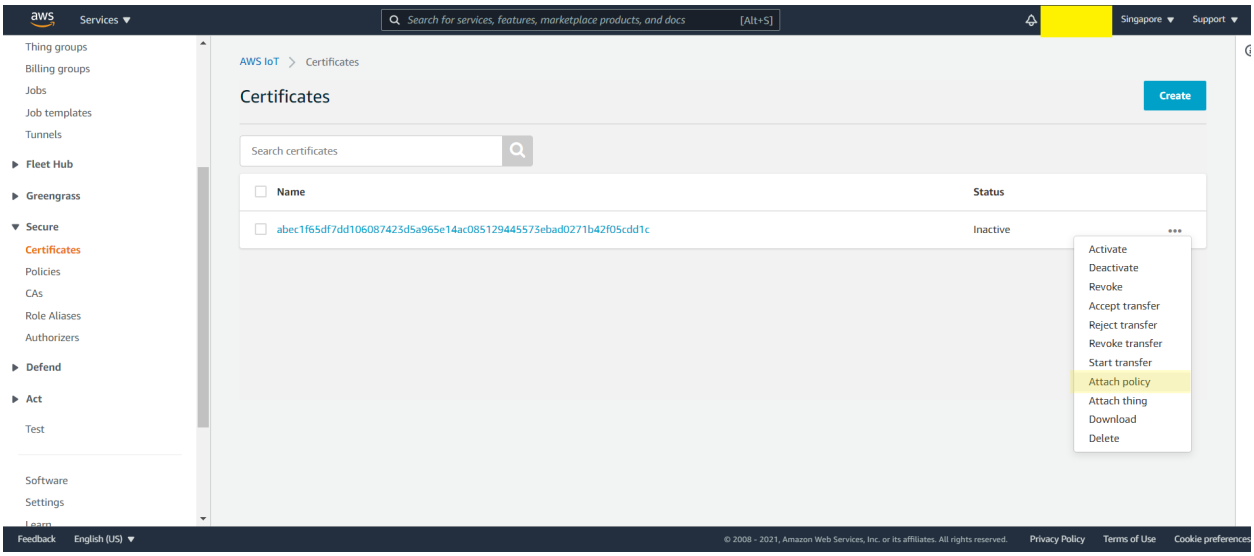
Next, we have to setup the TLS certificate. You can choose to user-defined or generate a certificate by AWS IoT. In this example we click Create Certificate to generate a TLS certificate.



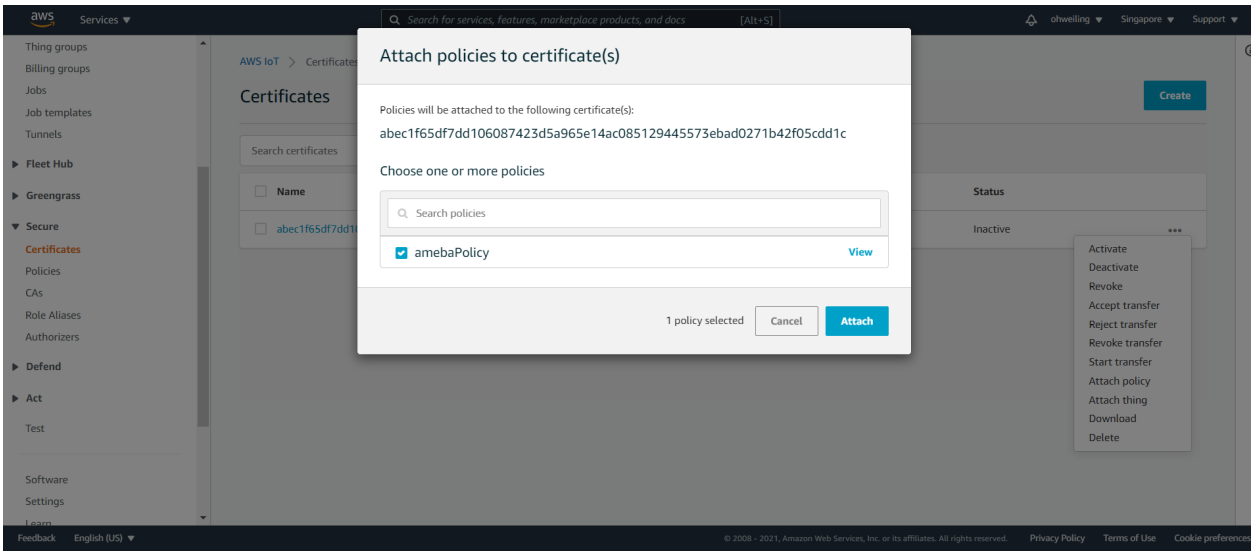
You can see 4 Links. Please download each of the link, “public key”, “private key”, “Certificate” and “rootCA”. After downloading the 4 files, click Done and go back to the certificate main page.



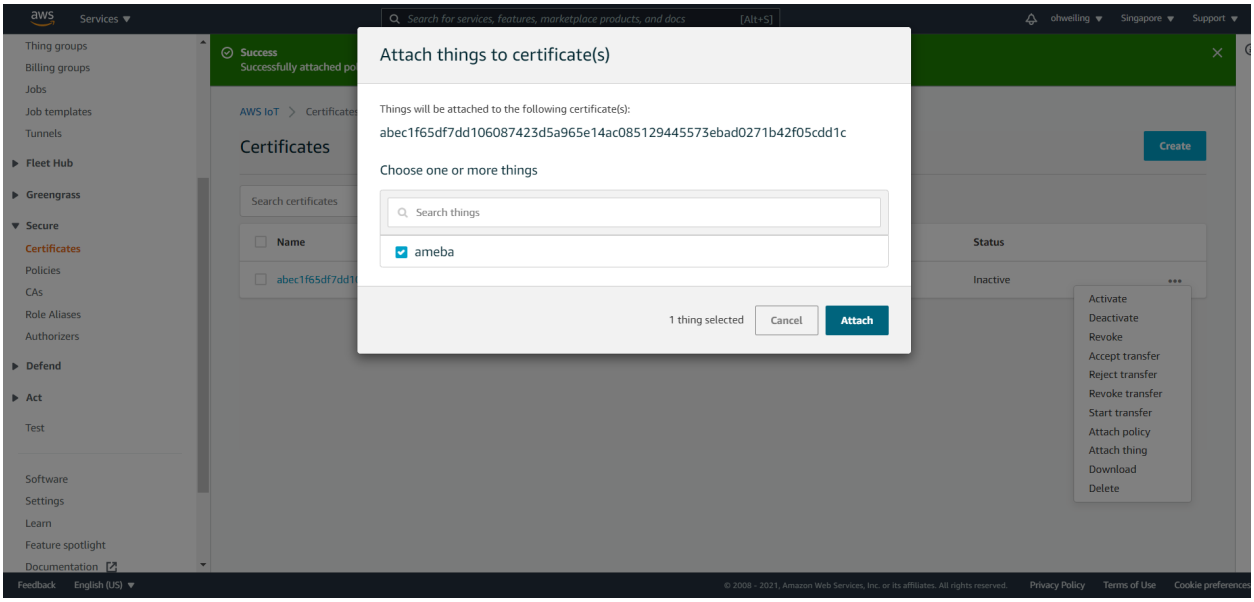
Click Attach a policy in the Actions dropdown menu.



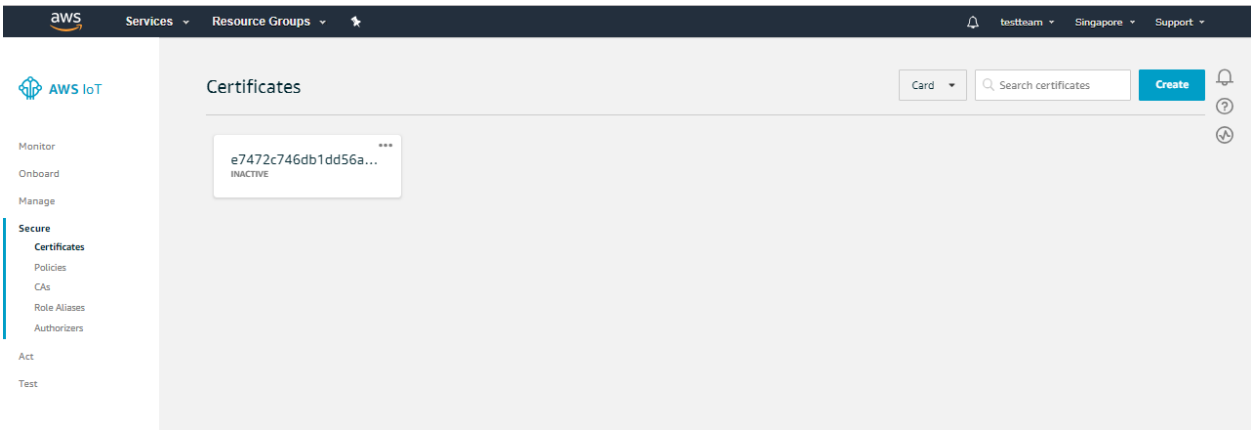
Choose amebaPolicy and click attach.



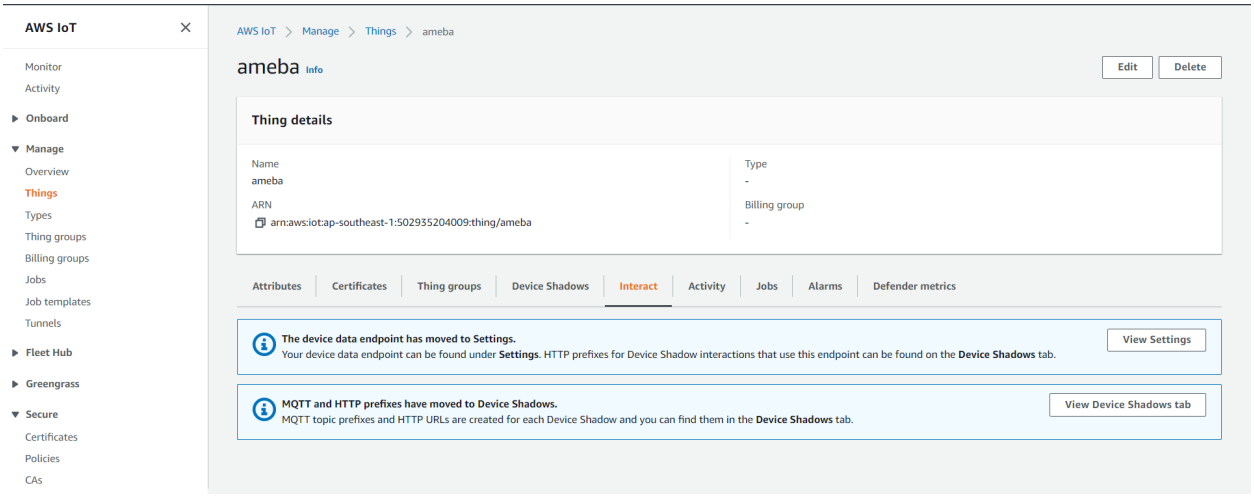
Then go back to the “Actions” drop-down menu at the top right of the certificates homepage, click on “Attach thing”, select the thing “ameba” you just created when the window below appears, then click on “Attach”



Go back to certificate main page and click Certificate and click Activate in the Actions drop down menu.



Next, click Manage, and click Things, then click “ameba” the thing we created just now.
Click on Interact and View settings.

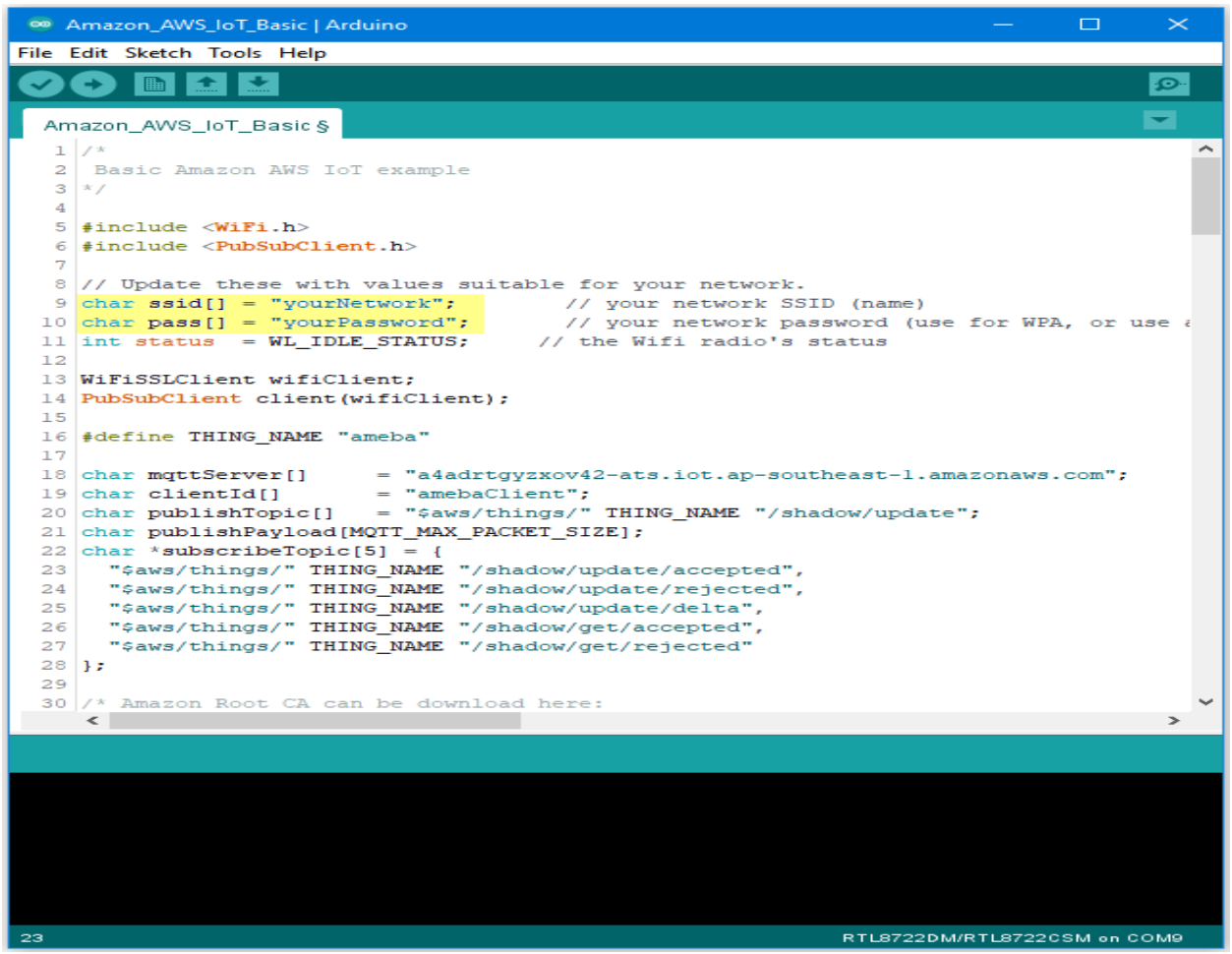


Find out the information of Rest API Endpoint to set Amazon Alexa:

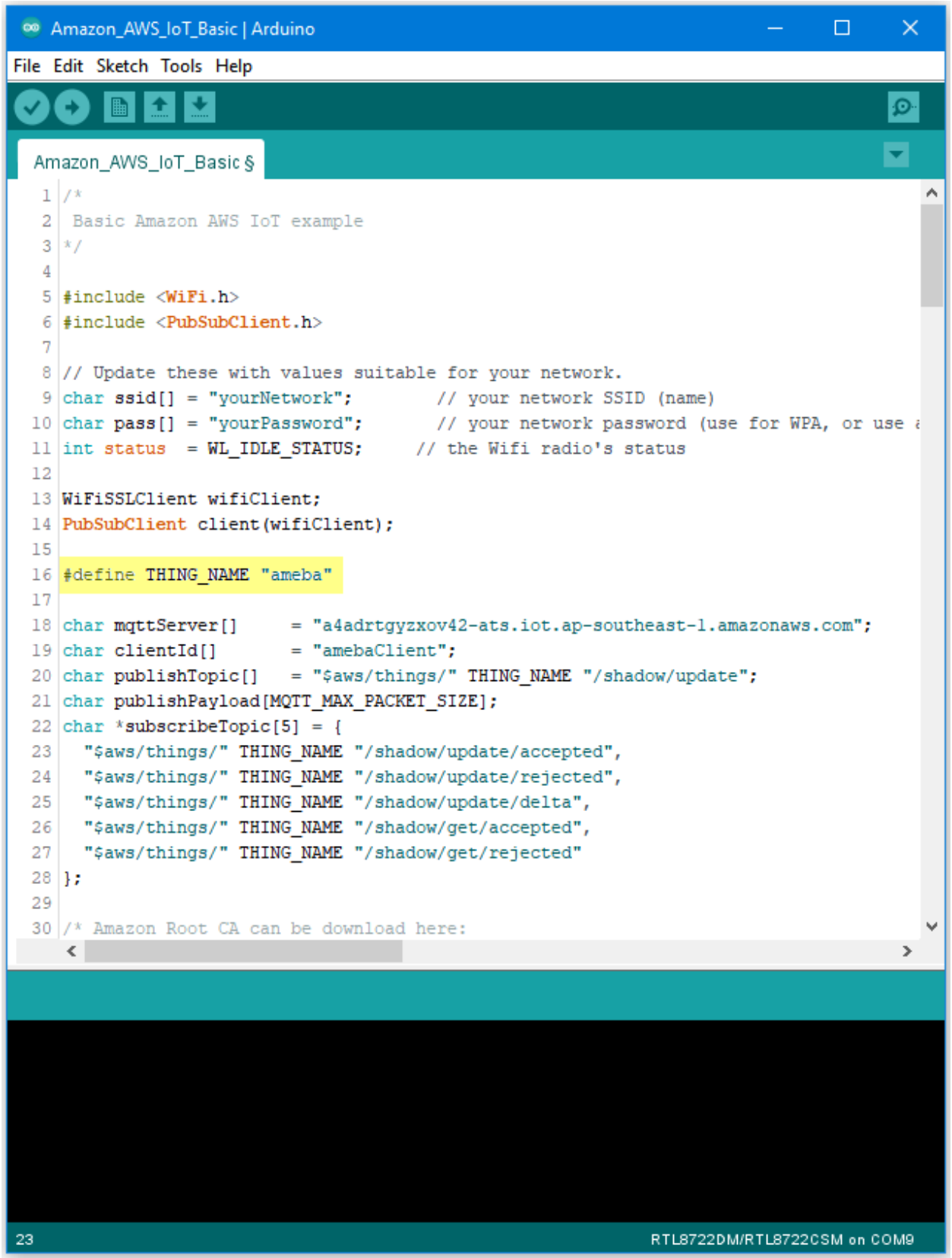
- REST API endpoint: In the value “https://a1a7oo4baosgyy.iot.us-east-1.amazonaws.com/things/ameba/shadow”, the part “a1a7oo4baosgyy.iot.us-east-1.amazonaws.com” is the MQTT Broker server address.
- MQTT topic The value “$aws/things/ameba/shadow/update” represents the MQTT topic we will use in the AWS IoT Shadow service (if we use MQTT only, without AWS IoT Shadow service, then we can specify other topic name). It is recommended to use “$aws/things/ameba/shadow/update” here.

Ameba setting

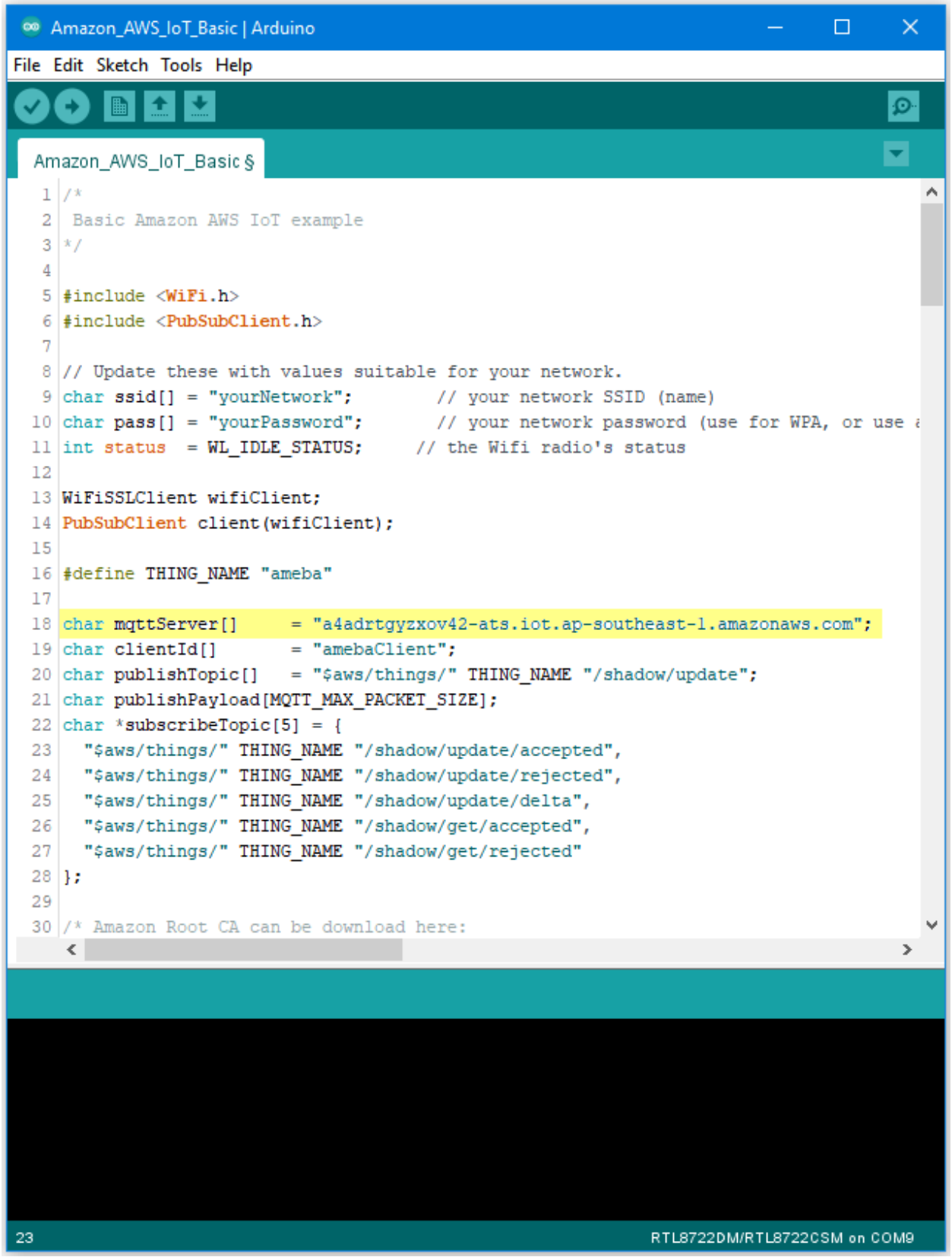
Open “`File` -> “`Examples`” -> “`AmebaMQTTClient`” -> “`Amazon_AWS_IoT_Basic`”
In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



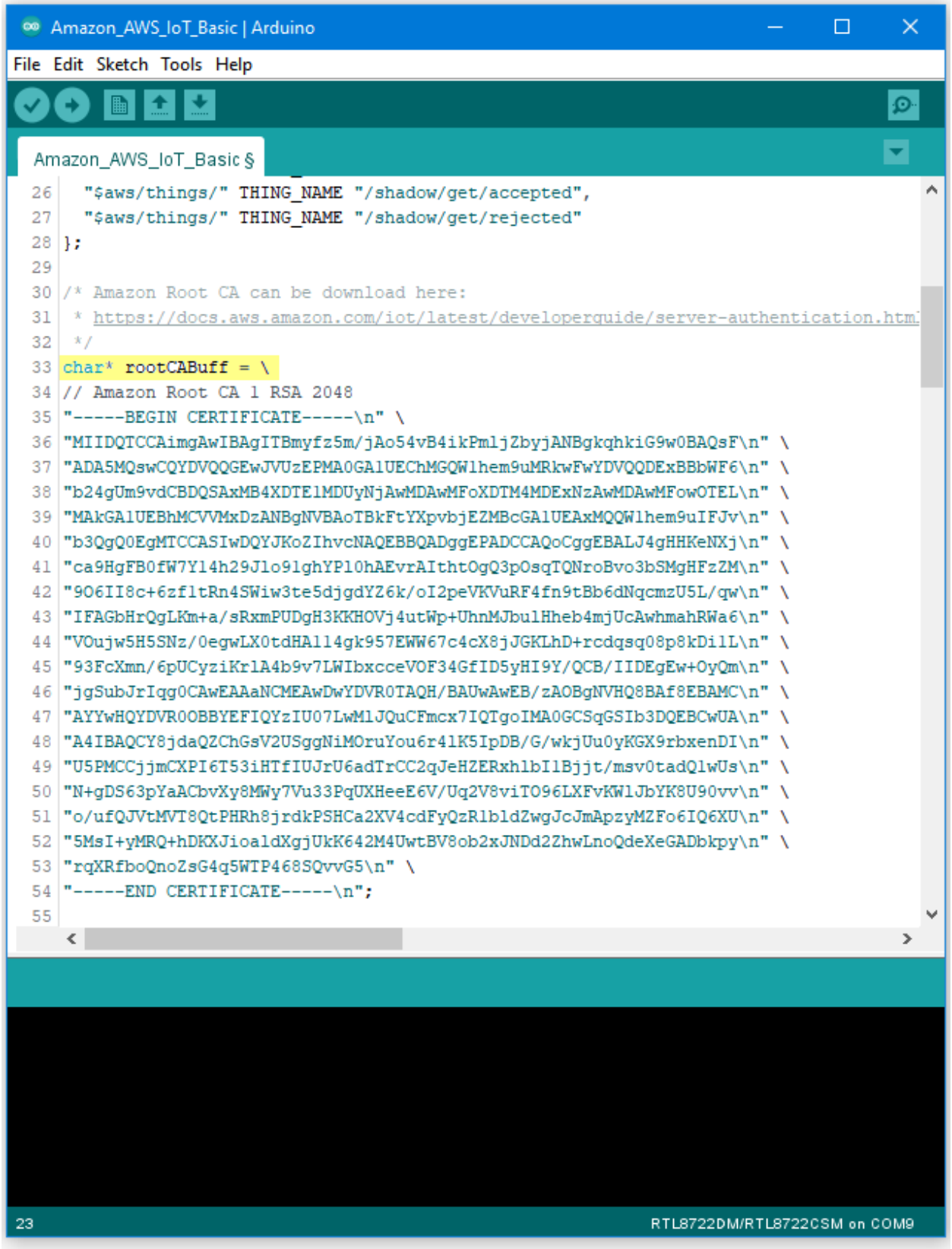
Then fill in the “thing” name “ameba”.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/*
Basic Amazon AWS IoT example
*/
#include <WiFi.h>
#include <PubSubClient.h>
// Update these with values suitable for your network.
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "yourPassword"; // your network password (use for WPA, or use
int status = WL_IDLE_STATUS; // the Wifi radio's status
WiFiSSLClient wifiClient;
PubSubClient client(wifiClient);
#define THING_NAME "ameba"
char mqttServer[] = "a4adrtgyzxov42-ats.iot.ap-southeast-1.amazonaws.com";
char clientId[] = "amebaClient";
char publishTopic[] = "$aws/things/" THING_NAME "/shadow/update";
char publishPayload[MQTT_MAX_PACKET_SIZE];
char *subscribeTopic[5] = {
"$aws/things/" THING_NAME "/shadow/update/accepted",
"$aws/things/" THING_NAME "/shadow/update/rejected",
"$aws/things/" THING_NAME "/shadow/update/delta",
"$aws/things/" THING_NAME "/shadow/get/accepted",
"$aws/things/" THING_NAME "/shadow/get/rejected"
};
/* Amazon Root CA can be download here:
23
RTL8722DM/RTL8722CSM on COM9

And the MQTT Broker server address we found earlier in AWS IoT.



Next, fill in the root CA used in TLS. Download and make sure the downloaded root CA contents conforms to the root CA used in the sketch.



Next, fill in the certificate we created in the AWS IoT Console (i.e., client certificate), usually its file name ends with "-certificate.pem.crt" (e.g., "efae24a533-certificate.pem.crt"). Open the certificate with a text editor, and adjust its format as follows to use in the sketch:

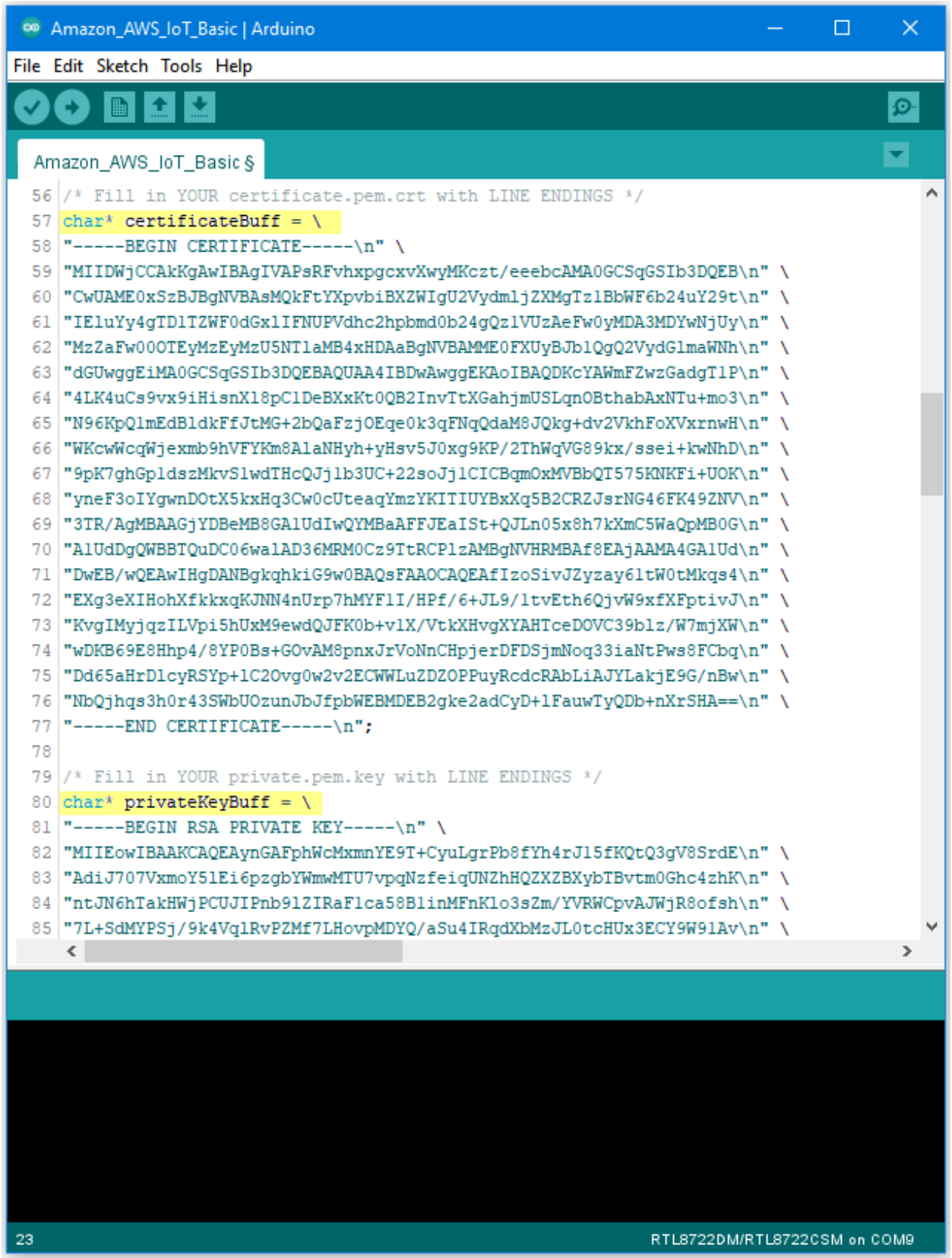
– Add the new line character "n" at the end of each line.

– Add double-quote at the beginning and the end of each line.

– To concatenate each line as a string, add "" at the end of each line.

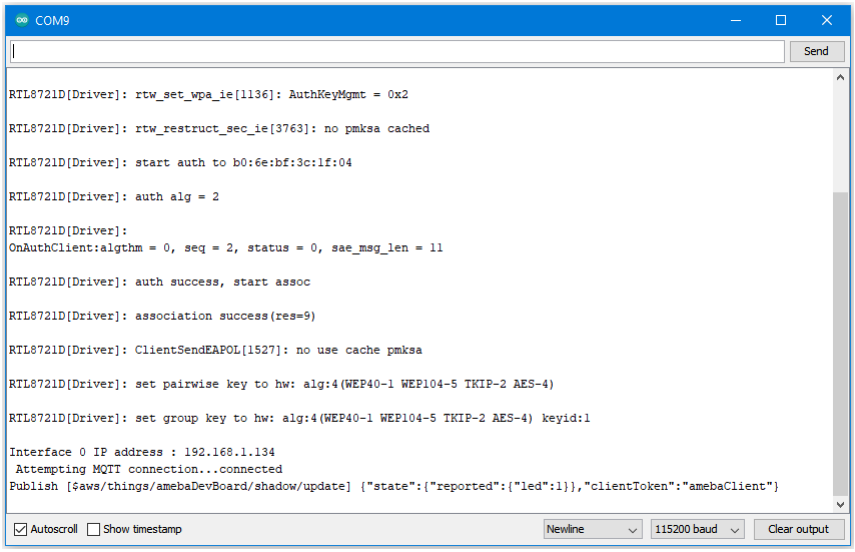
– The last line ends with semicolon.

Adjust the format of the private key in the same way and add it to privateKeyBuff.


Amazon_AWS_IoT_Basic | Arduino
File Edit Sketch Tools Help
Amazon_AWS_IoT_Basic §
/* Fill in YOUR certificate.pem.crt with LINE ENDINGS */
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDWjCCAkKgAwIBAgIVAPsRFvhxpgcxvXwyMKczt/eeebcAMA0GCSqGSIb3DQEB\n" \
"CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTzlBbWF6b24uY29t\n" \
"IEluYy4gTDlTZWF0dGxlIFNUPVdhc2hpbmd0b24gQzlVUzAeFw0yMDA3MDYwNjUy\n" \
"MzZaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJblQgQ2VydGlmaWNh\n" \
"dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDKcYAWmFZwzGadgTlP\n" \
"4LK4uCs9vx9iHisnXl8pClDeBXxKt0QB2InvTtXGahjmUSLqnOBthabAxNTu+mo3\n" \
"N96KpQlmEdBldkFfJtMG+2bQaFzjOEqe0k3qFNqQdaM8JQkg+dv2VkhFoXVxrnwH\n" \
"WKcwWcqWjexmb9hVFYKm8AlaNHyh+yHsv5J0xg9KP/2ThWqVG89kx/ssei+kwNhD\n" \
"9pK7ghGpldszMkvSlwdTHcQJjlb3UC+22soJjlCICBqmOxMVBbQT575KNKFi+UOK\n" \
"yneF3oIYgwnDOtX5kxHq3Cw0cUteaqYmzYKITIUYBxXq5B2CRZJsrNG46FK49ZNV\n" \
"3TR/AgMBAAGjYDBeMB8GA1UdIwQYMBaAFFJEaISt+QJLn05x8h7kXmC5WaQpMB0G\n" \
"A1UdDgQWBBTQuDC06walAD36MRM0Cz9TtRCPlzAMBgNVHRMBAf8EAjAAMA4GA1Ud\n" \
"DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAfIzoSivJZyzay6ltW0tMkqs4\n" \
"EXg3eXIHohXfkkxqKJNN4nUrp7hMYFlI/HPf/6+JL9/ltvEth6QjvW9xfXFptivJ\n" \
"KvgIMyjqzILVpi5hUxM9ewdQJFK0b+vlX/VtkXHvgXYAHTceDOVC39blz/W7mjXW\n" \
"wDKB69E8Hhp4/8YP0Bs+GOvAM8pnxJrVoNnCHpjerDFDSjmNoq33iaNtPws8FCbq\n" \
"Dd65aHrDlcyRSYp+lC2Ovg0w2v2ECWWLuZDZOPPuyRcdcRAbLiAJYLakjE9G/nBw\n" \
"NbQjhqs3h0r43SWbUOzunJbJfpbWEBMDEB2gke2adCyD+lFauwTyQDb+nXrSHA==\n" \
"-----END CERTIFICATE-----\n";
/* Fill in YOUR private.pem.key with LINE ENDINGS */
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEowIBAAKCAQEAynGAFphWcMxmnYE9T+CyuLgrPb8fYh4rJ15fKQtQ3gV8SrdE\n" \
"AdiJ707VxmoY5lEi6pzgbYWmwMTU7vpqNzfeiqUNZhHQZXZBXybTBvtm0Ghc4zhK\n" \
"ntJN6hTakHWjPCUJIPnb9lZIRaF1ca58BlinMFnKlo3sZm/YVRWCpvAJWjR8ofsh\n" \
"7L+SdMYPSj/9k4VqlRvPZMf7LHovpMDYQ/aSu4IRqdXbMzJL0tcHUx3ECY9W9lAv\n"
23
RTL8722DM/RTL8722CSM on COM9

Compile and run

Upload the code and press the reset button on Ameba once the upload is finished.
Open the serial monitor in the Arduino IDE and observe as Ameba connects to the AWS IoT server and sends updates on the LED state variable.



Alternatives

Ameba can also retrieve the current LED status variable from the AWS shadow. This is done by sending a message to the "shadow/get" topic. Refer to the Amazon_AWS_IoT_with_ACK example code for more information.

Code Reference

Change led state:
In this example, we use GPIO interface to control the led. We set led_pin to 10 and led_state to 1 by default in the sample code.

```
pinMode(led_pin, OUTPUT);
digitalWrite(led_pin, led_state);
```

Set up certificate:
Note that we use the WiFiSSLClient type of wifiClient.

```
WiFiSSLClient wifiClient;
```

WiFiSSLClient inherits Client, so it can be passed as the parameter of PubSubClient constructor.
Next, set up TLS certificate required in connection.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
wifiClient.setClientCertificate((unsigned char*)certificateBuff,(unsigned
↪char*)privateKeyBuff);
```

Configure MQTT Broker server
Then MQTT PubClient set MQTT Broker server to connect

```
client.setServer(mqttServer, 8883);
client.setCallback(callback);
```

Connect to MQTT Broker server:
In `loop()`, call `reconnect()` function and try to connect to MQTT Broker server and do the certificate verification.

```
while (!client.connected()) {
```

Subscribe & Publish
Next, subscribe to topics.

```
for (int i=0; i<5; i++) {
  client.subscribe(subscribeTopic[i]);
}
```

There are some common topics:
“$aws/things/ameba/shadow/update/accepted”,
“$aws/things/ameba/shadow/update/rejected”,
“$aws/things/ameba/shadow/update/delta”,
“$aws/things/ameba/shadow/get/accepted”,
“$aws/things/ameba/shadow/get/rejected”
Related documentation:
http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-data-flow.html

Then publish current status::

```
sprintf(publishPayload,
"{\"state\":{\"reported\":{\"led\":%d}},\"clientToken\":\"%s\"}",
led_state, clientId);
```

```
client.publish(publishTopic, publishPayload);
```

Listen to topic and make response:
In the callback function, we listen to the 5 subscribed topics and check if there are messages of "/shadow/get/accepted":

```
if (strstr(topic, "/shadow/get/accepted") != NULL) {
```

If there is, the message is from the control side. If the attribute state in the message is different from current state, publish the new state.

```
updateLedState(desired_led_state);
```

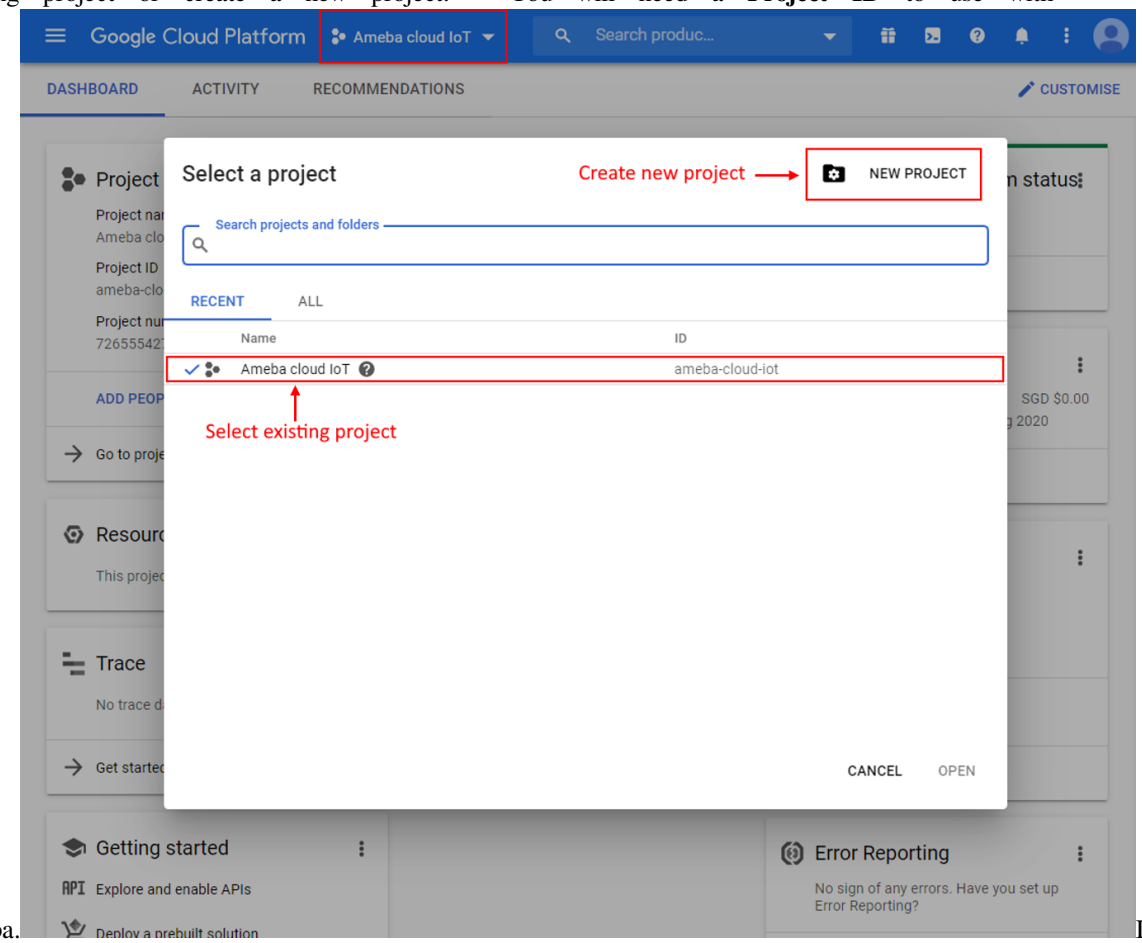
MQTT - Use Google Cloud IoT

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

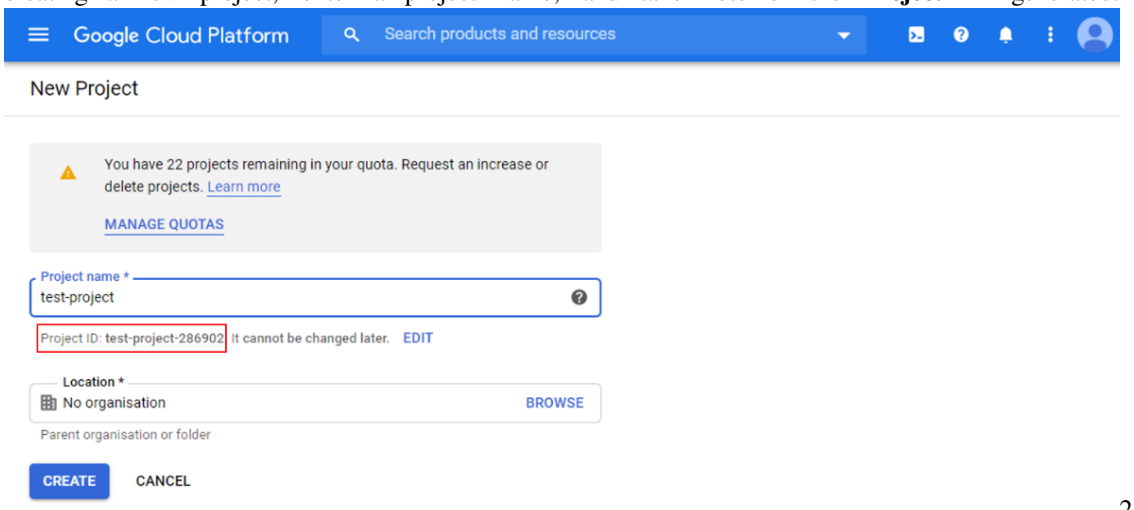
Google Cloud IoT Configuration

1. Select or create a Cloud Platform project In the Google Cloud Console, select an existing project or create a new project. You will need a **Project ID** to use with Ameba.

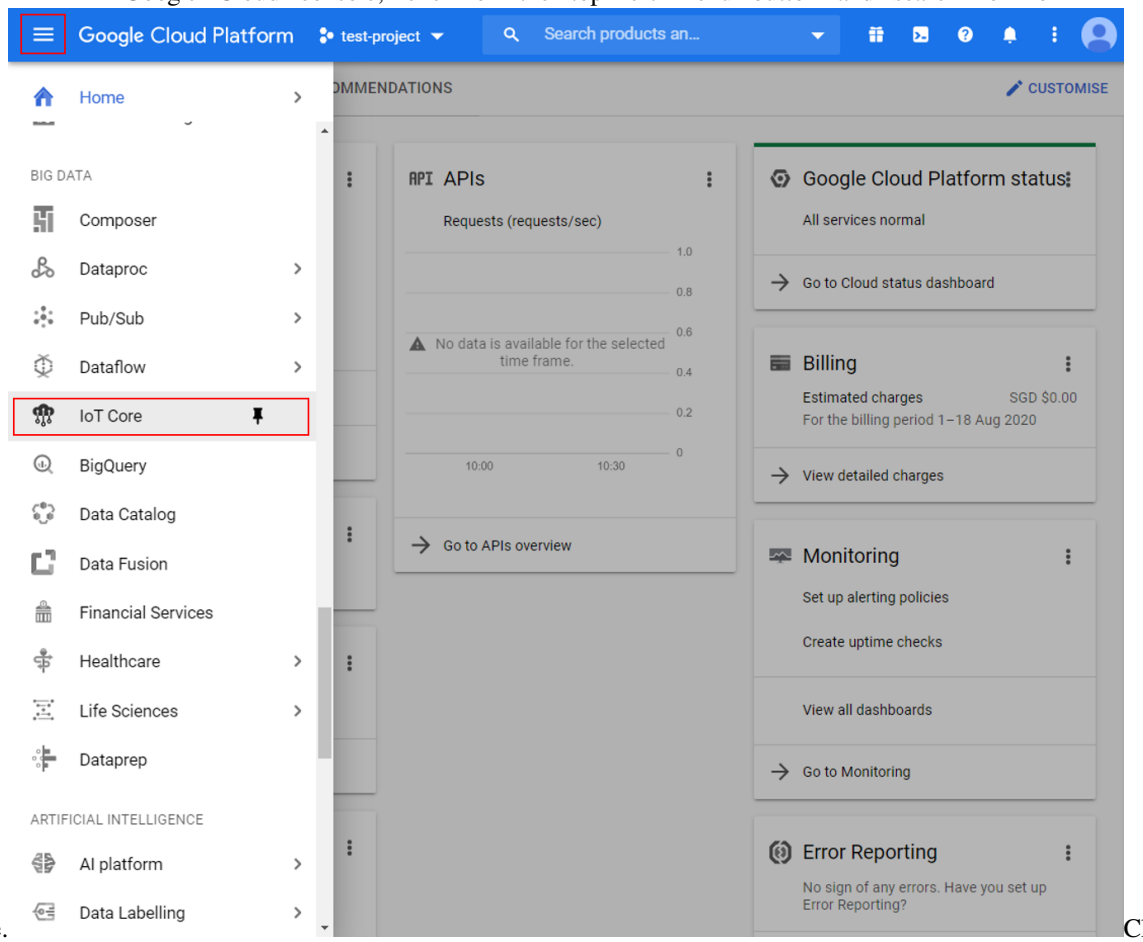


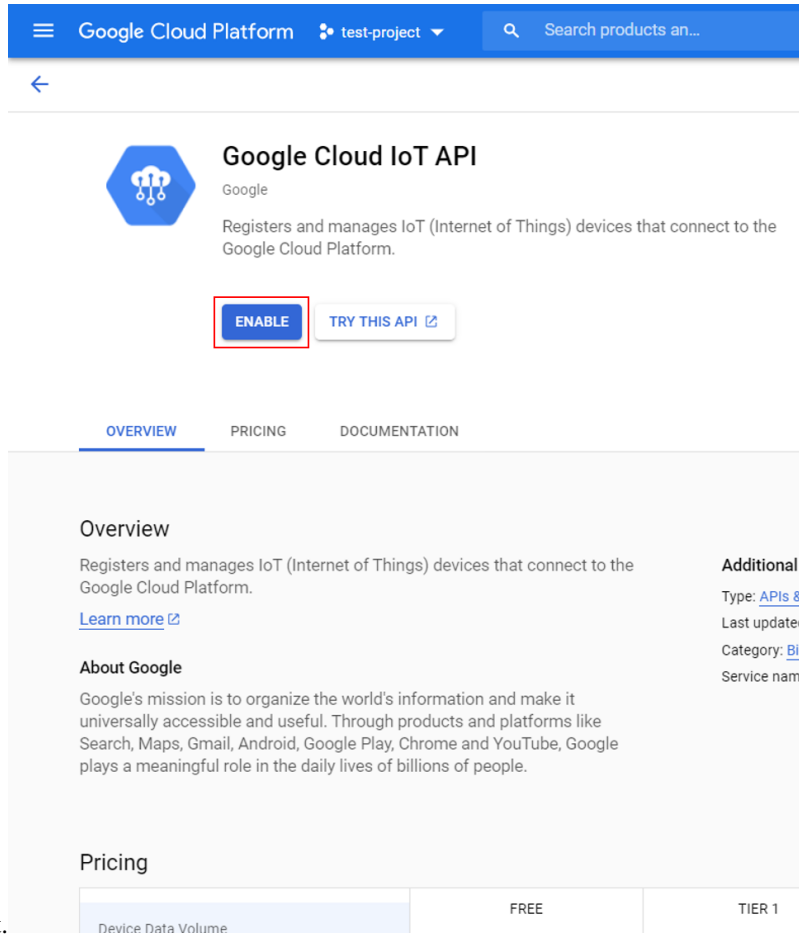
If

creating a new project, enter a project name, and take note of the **Project ID** generated.



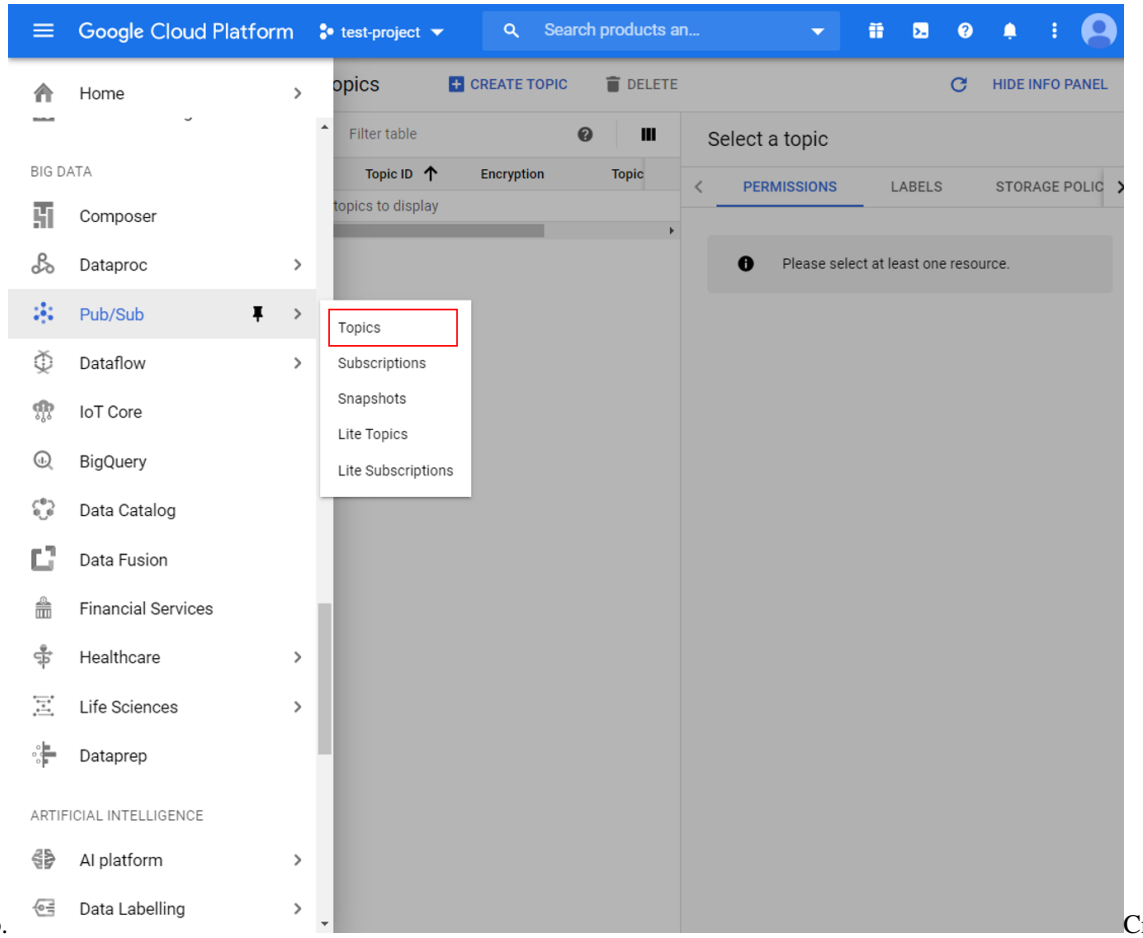
2. Enable billing for your project Billing needs to be enabled for your project to use Google Cloud Platform features. Follow the guide in Google cloud documentation to enable billing. https://cloud.google.com/billing/docs/how-to/modify-project 3. Enable the Cloud IoT Core API In Google Cloud console, click on the top left menu button and search for IoT Core. Click



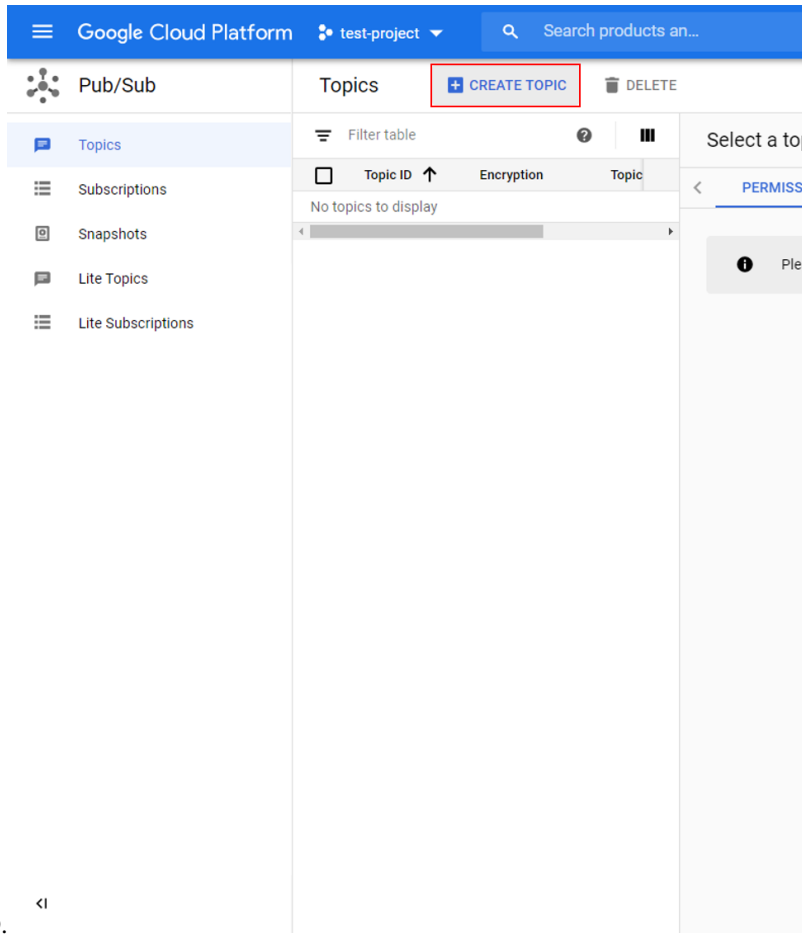


enable to activate Google Cloud IoT API for your project.

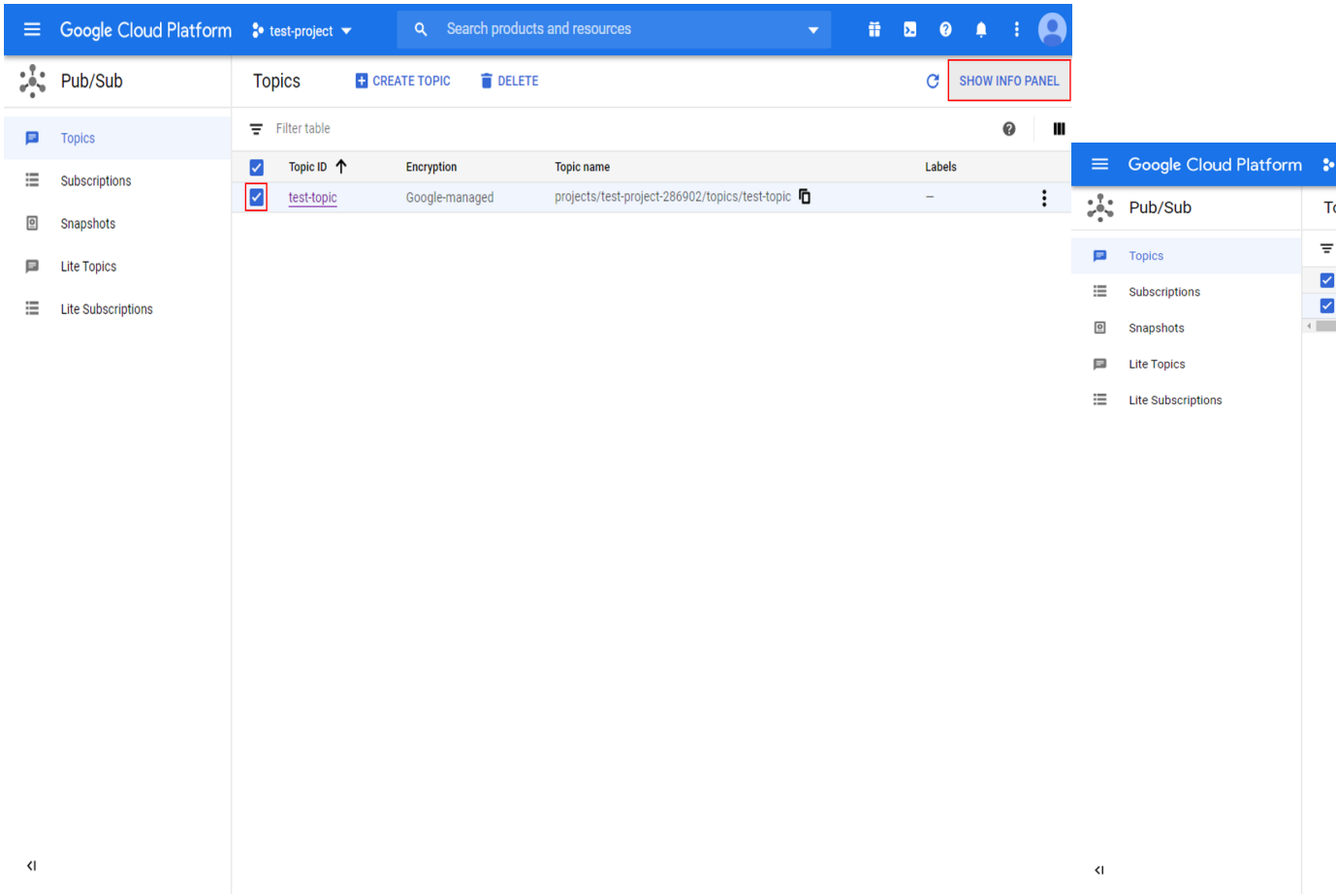
Create a Cloud Pub/Sub topic In Google Cloud console, click on the top left menu button and search for



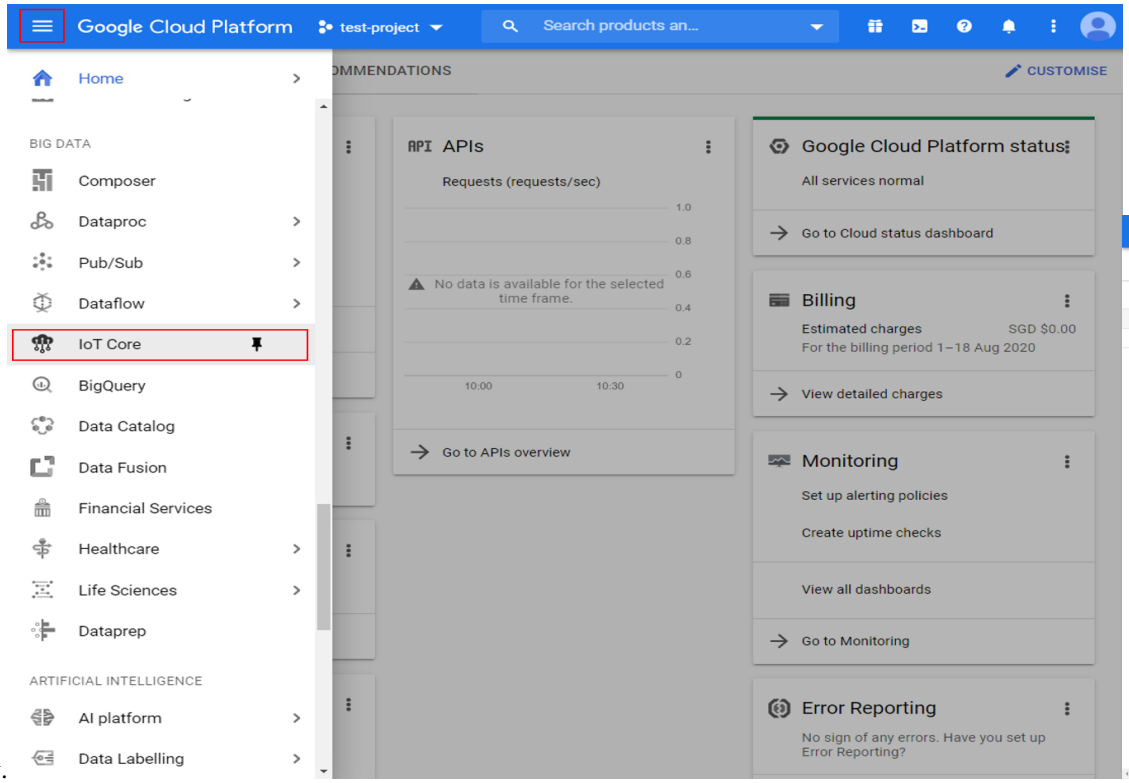
Pub/Sub. Create



a new topic for your project and give it a suitable topic ID. the topic is created, go to the permissions tab of the info panel, and add "cloud-iot@system.gserviceaccount.com" with the role of "Pub/Sub Publisher".

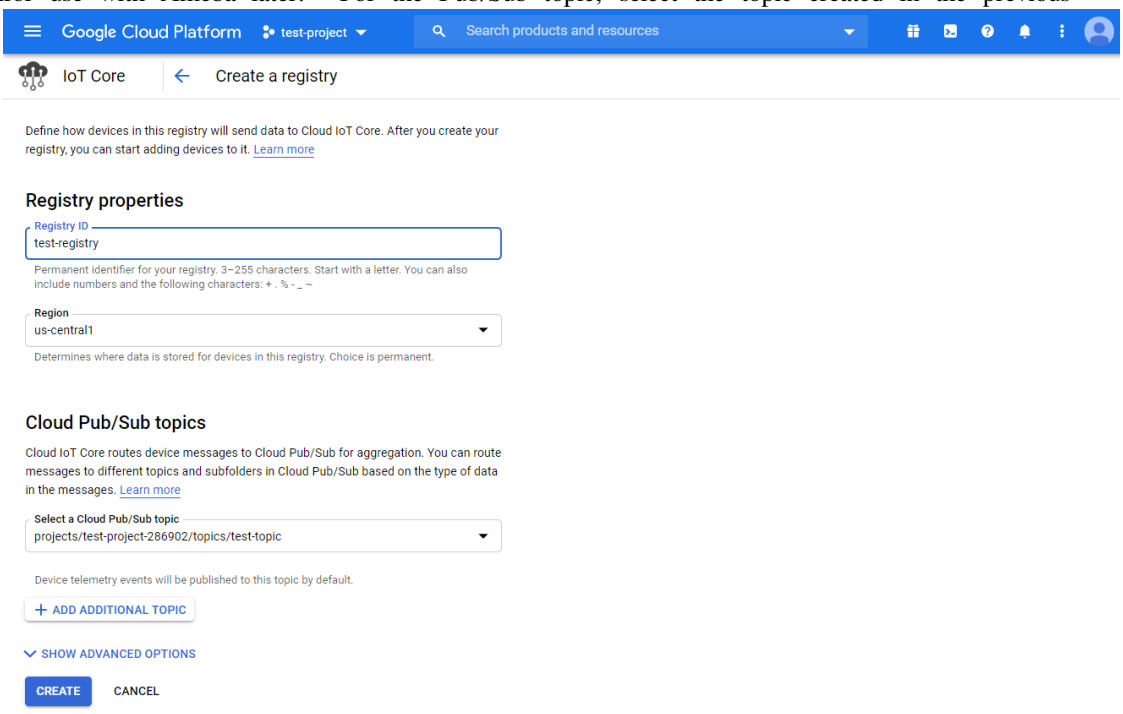


a device registry Go back to the IoT Core settings page and create a new



registry.

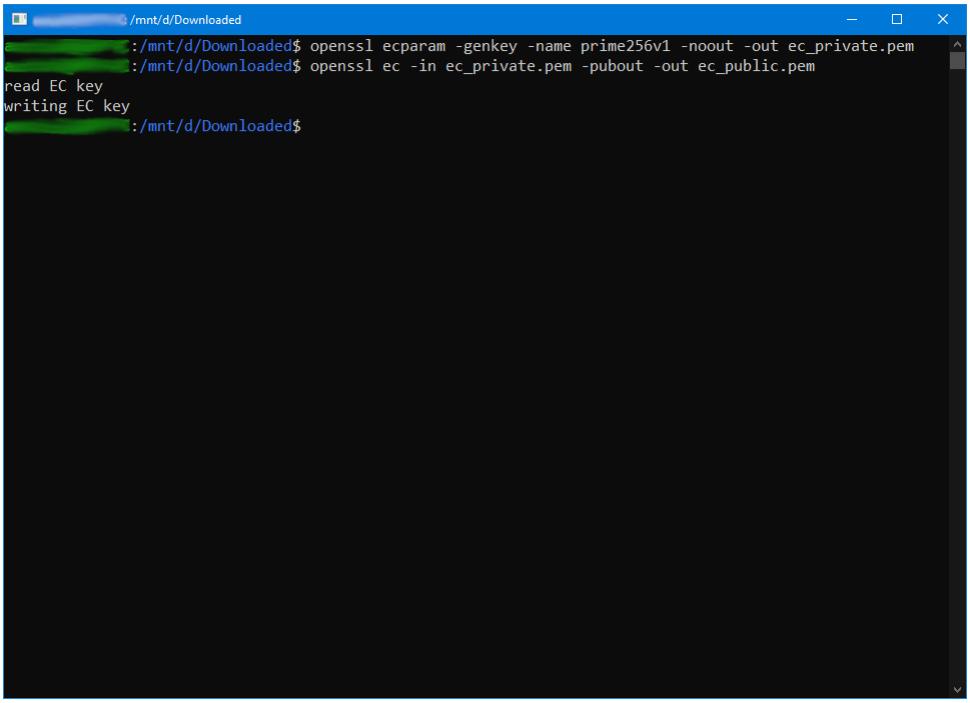
a suitable **Registry ID** and **in which to store data.** **Remember the **Registry ID** and **Region**for use with Ameba later. For the Pub/Sub topic, select the topic created in the previous



step.

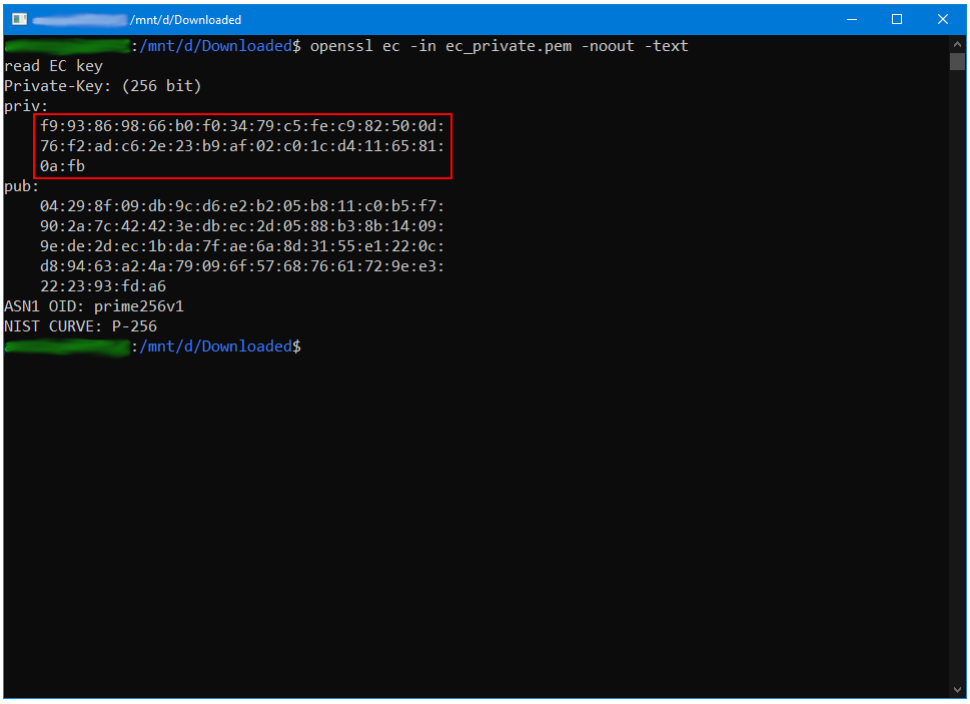
6. Create a public/private key pair Using Openssl in a terminal in Windows/Linux/MacOs, run the following commands to generate a private and public key pair. Two files will be created by these commands, “ec_private.pem” containing the private key, and “ec_public.pem” containing the public key.

```
$ openssl ecparam -genkey -name prime256v1 -noout -out ec_private.pem
$ openssl ec -in ec_private.pem -pubout -out ec_public.pem
```

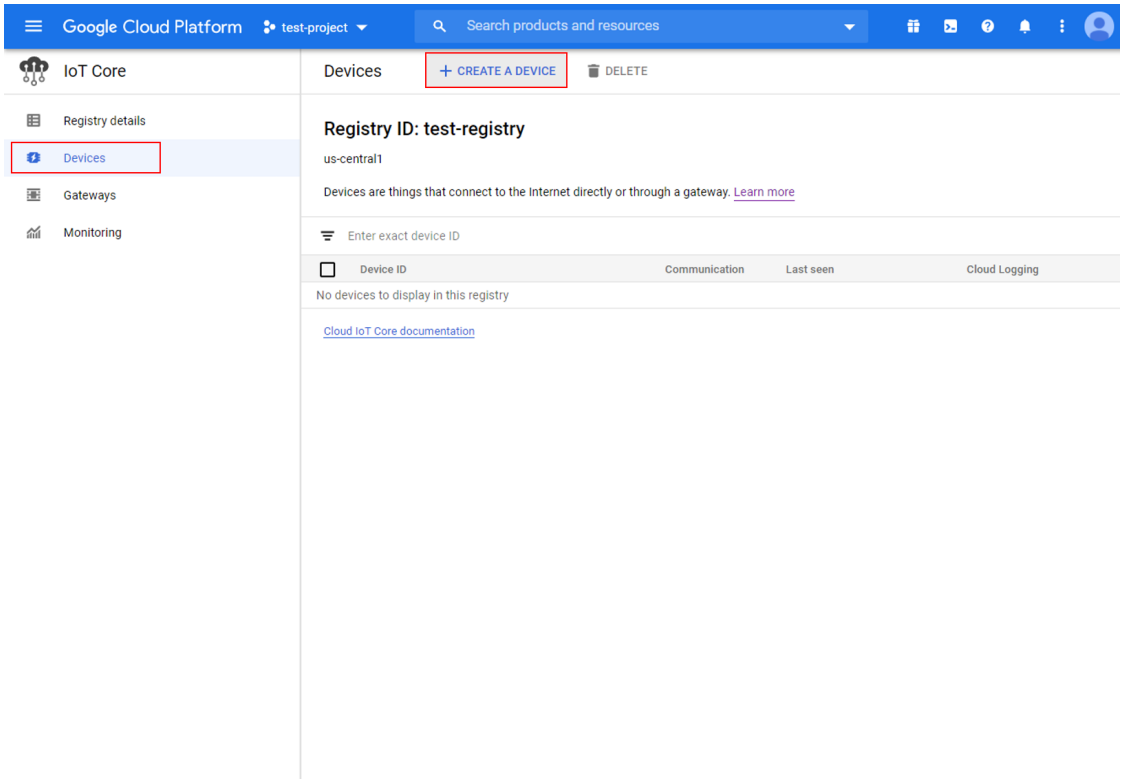


Run the next command to extract out the private key, and remember the highlighted string of hexadecimal numbers for use with Ameba later.

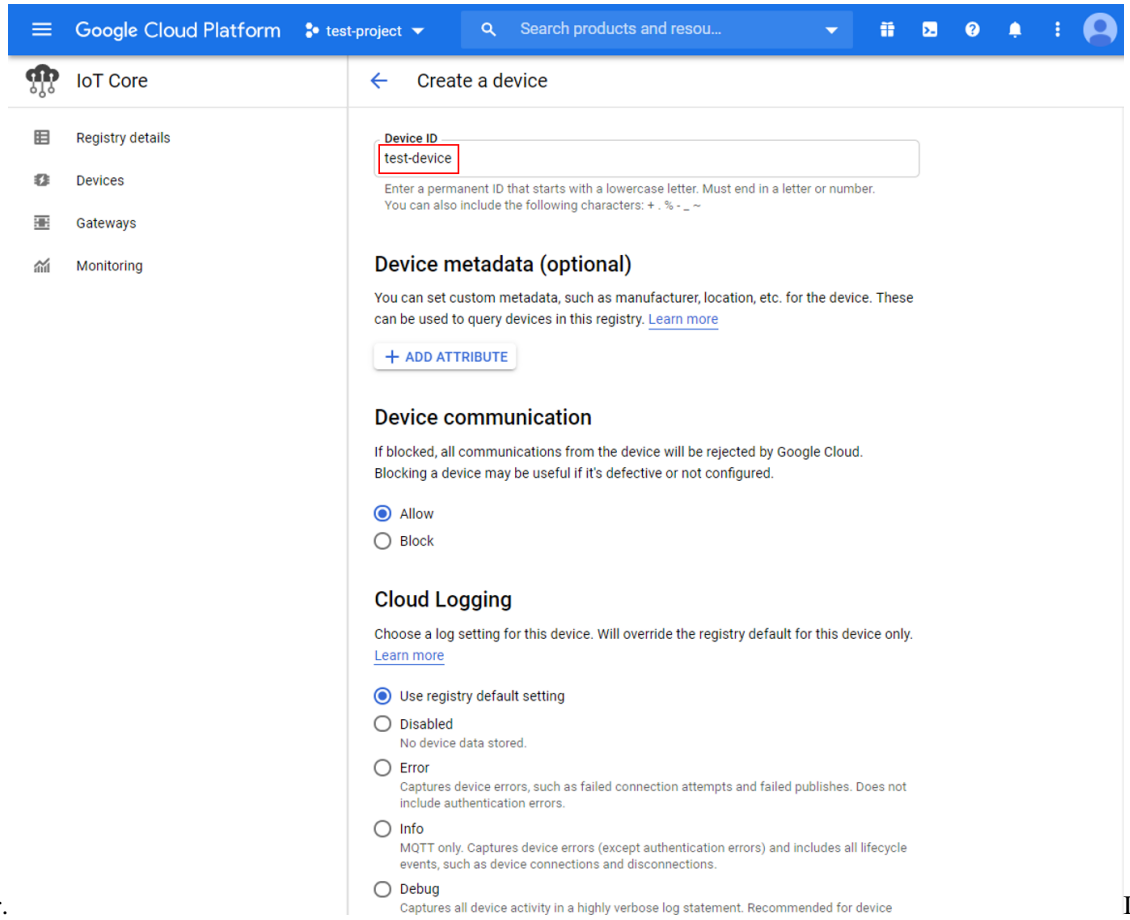
```
$ openssl ec -in ec_private.pem -noout -text
```



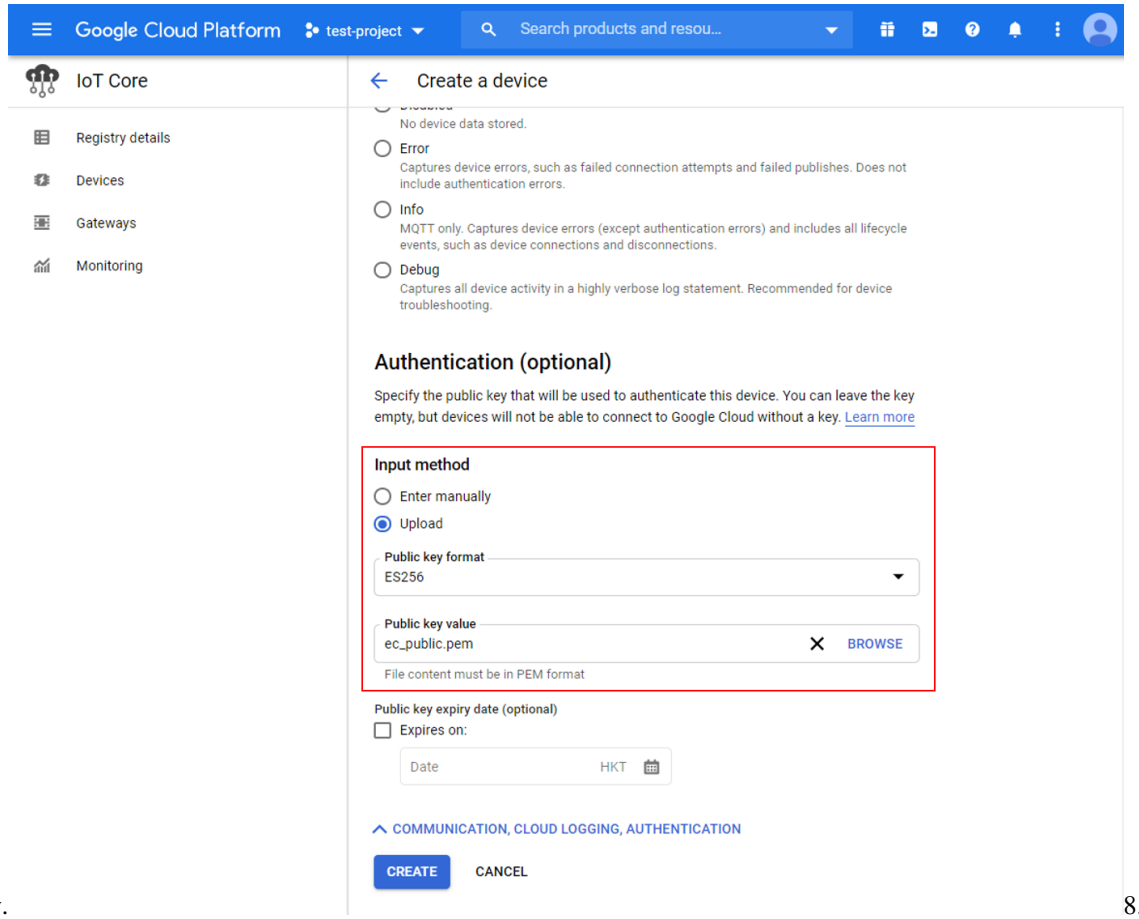
7. Create a device Go back to the IoT Core settings page and create a new device.



Give the device a suitable **Device ID** and remember it for use with Ameba

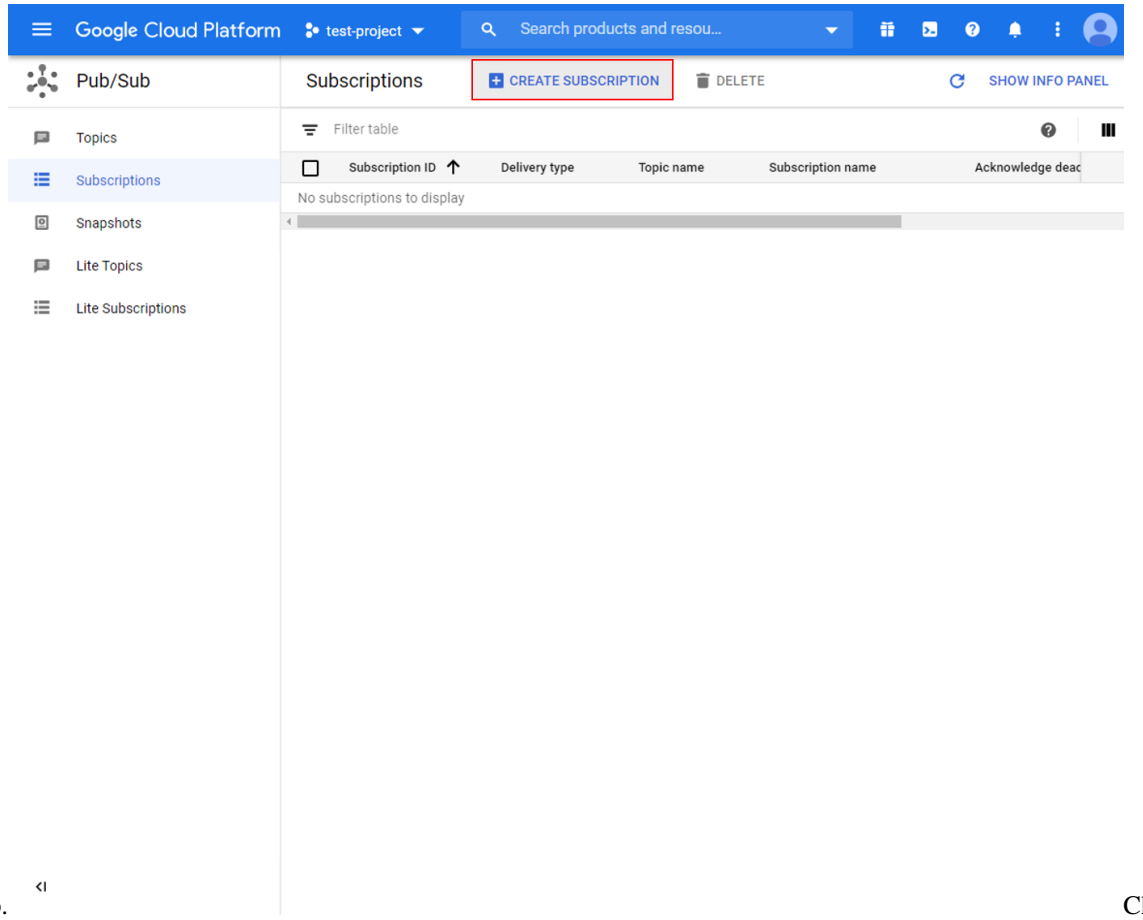


later. In the authentication section of the additional options, upload the previously generated “ec_public.pem” public

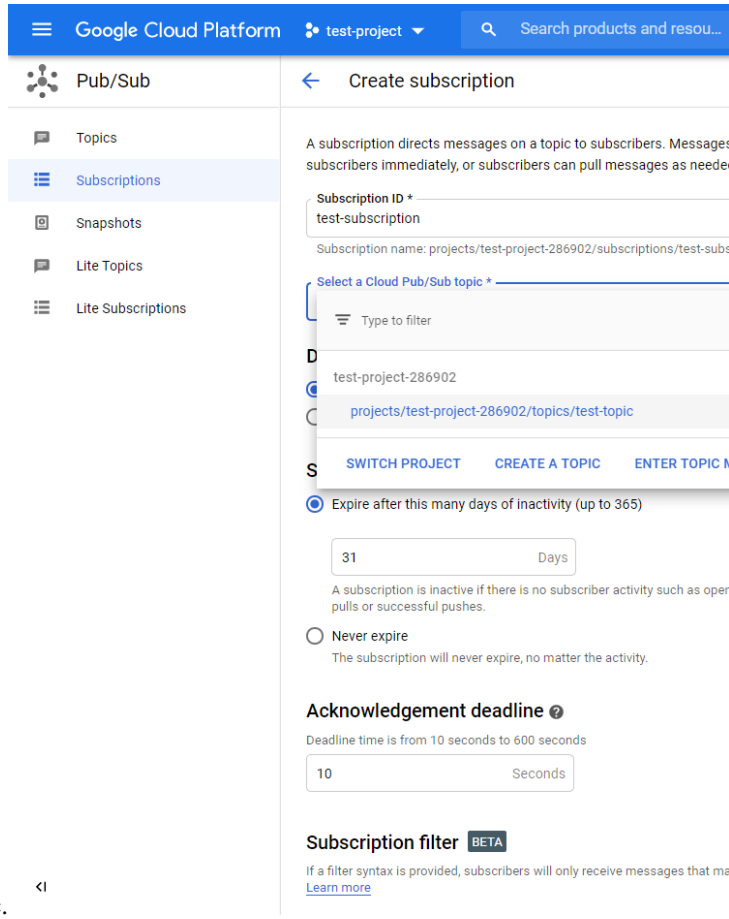


key.

8. Create a Cloud Pub/Sub subscription To observe messages sent by Ameba, create a subscription in



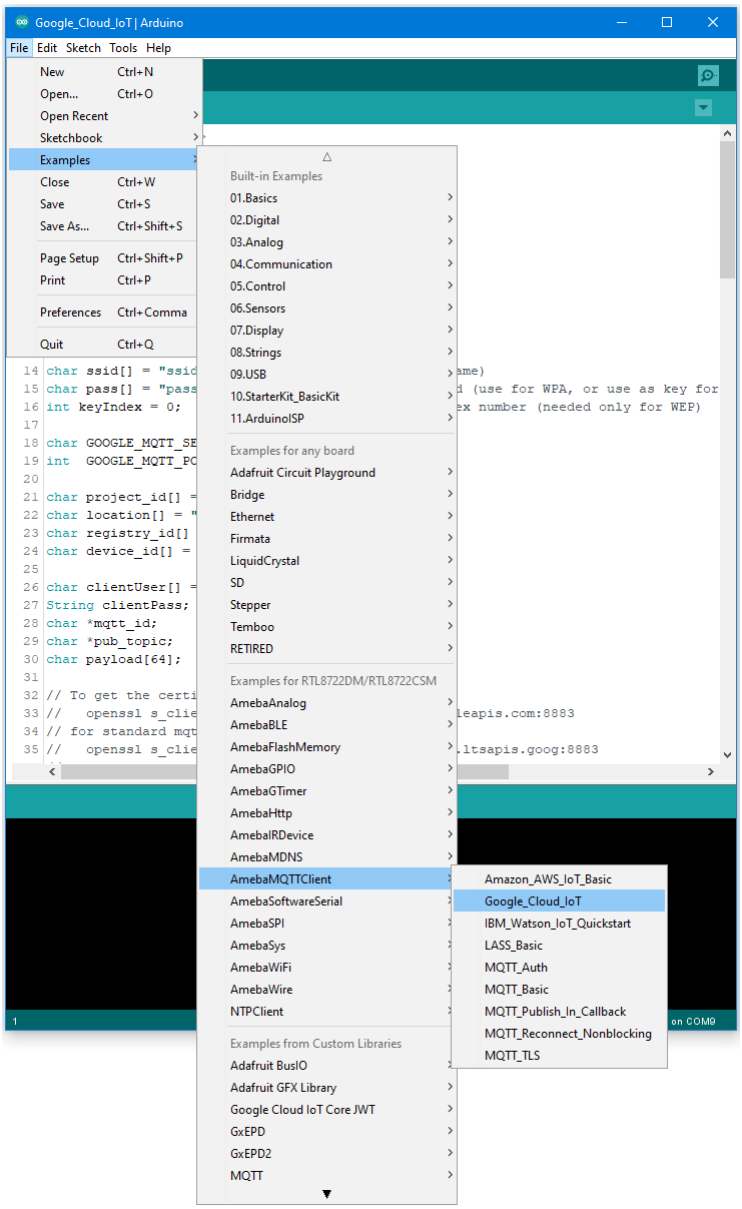
Pub/Sub. Choose



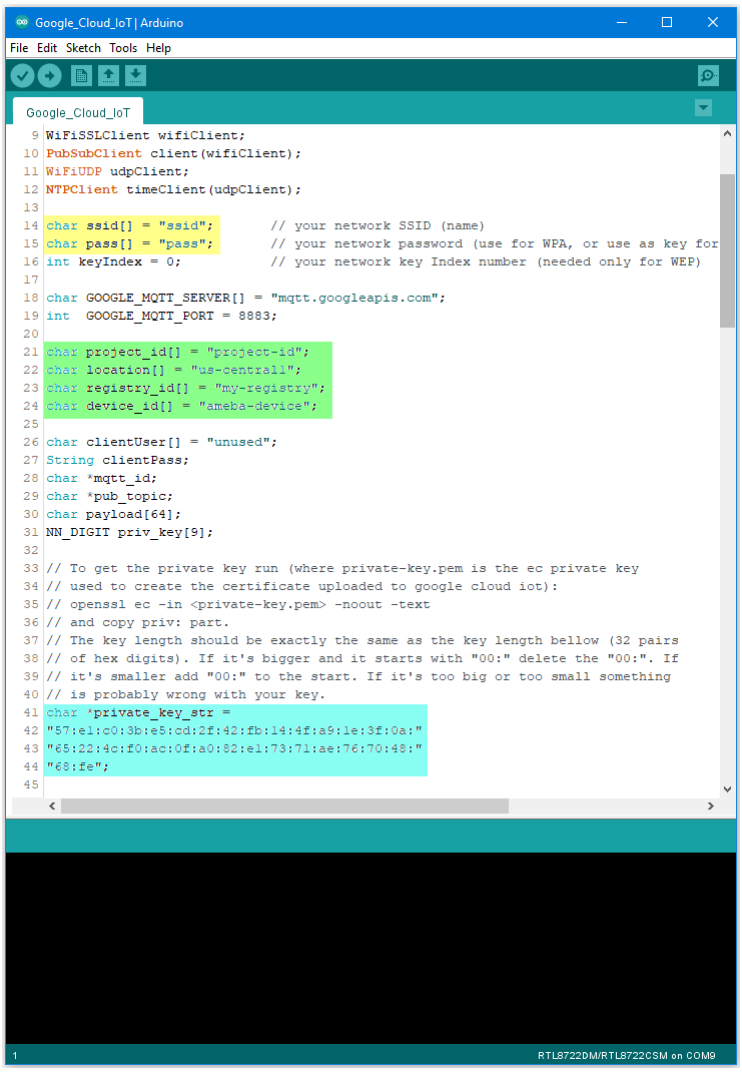
a suitable subscription ID and select the previously created topic.

Example

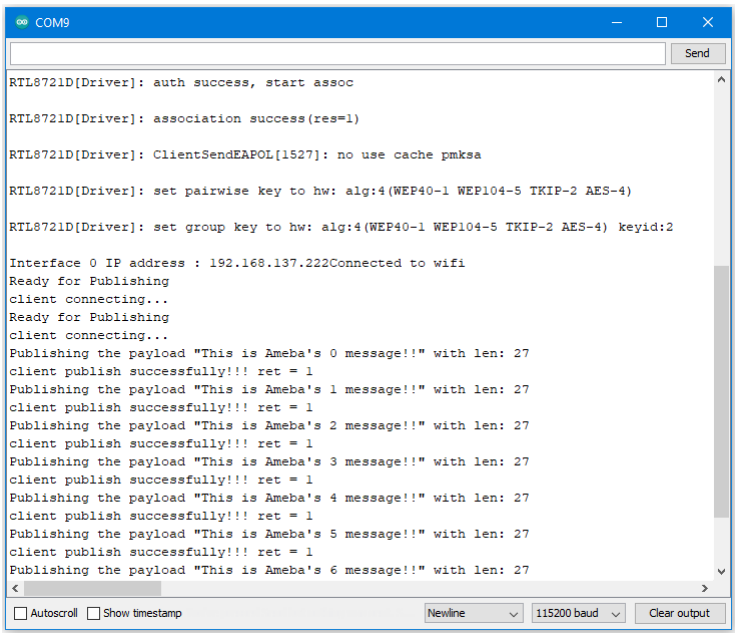
Open the example in "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`Google_Cloud_IoT`".



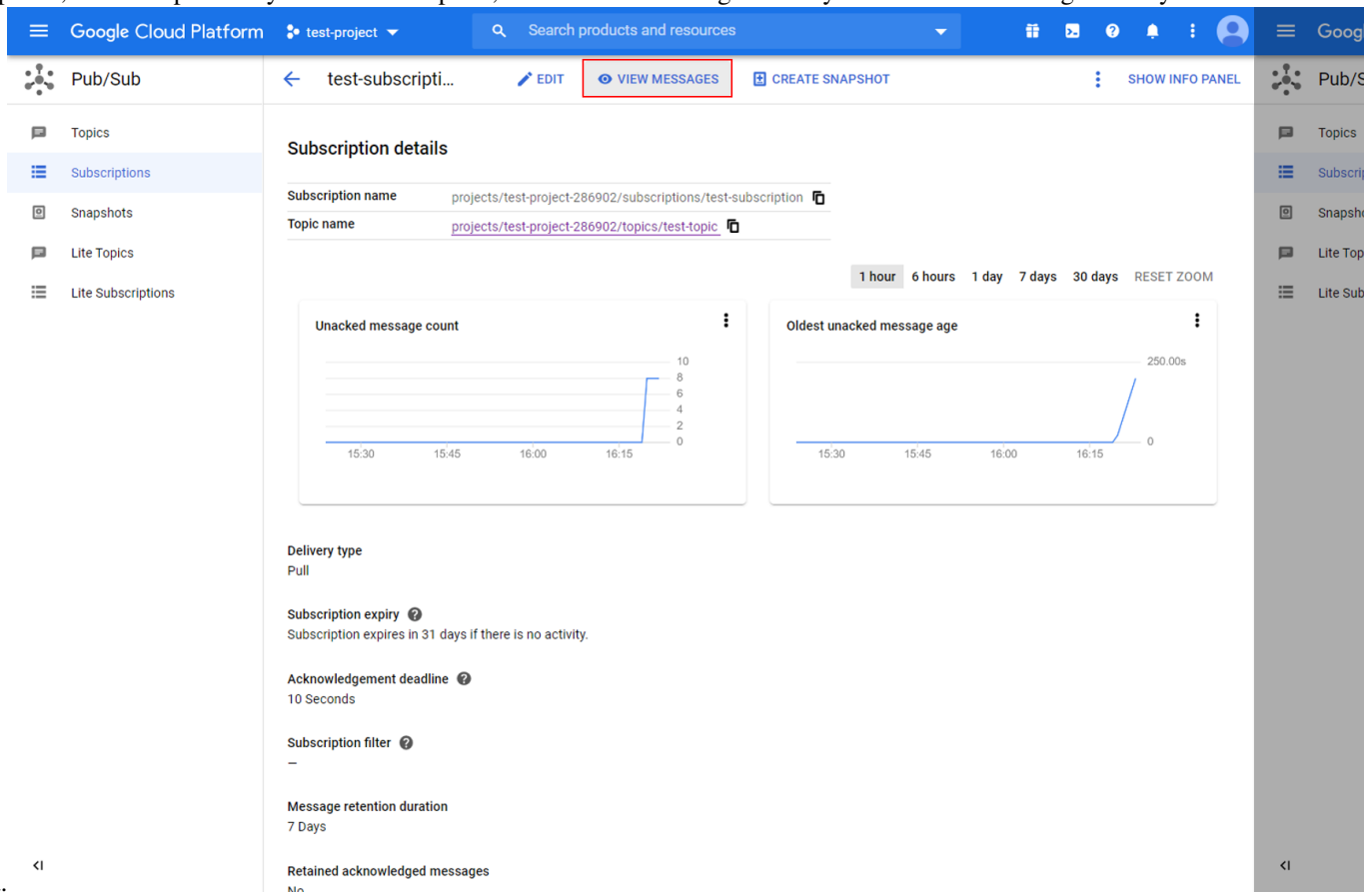
Enter the required information in the highlighted sections below.



In the yellow section, enter the SSID and password required to connect to your WiFi network. In the green section, enter the Project ID, server Region, Registry ID and Device ID previously configured in Google Cloud console. In the blue section, enter the hexadecimal string previously extracted from the private key. Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor and observe as Ameba connects and sends messages to Google Cloud IoT.



In Google Cloud console, go to Pub/Sub subscriptions, select the previously created subscription, and click view messages. Here you can view the messages sent by Ameba.



Code Reference

In `setup()`, we set up RootCA which is required to form a TLS connection with Google's servers.

```
wifiClient.setRootCA((unsigned char*)rootCABuff);
```

In `loop()`, each loop checks the Internet status and re-connect to it when the environment has a problem.

```
if (WiFi.status() != WL_CONNECTED) {
   while (WiFi.begin(ssid, pass) != WL_CONNECTED)
   {
      delay(1000);
   }
   Serial.println("Connected to wifi");
}
```

To publish messages, mqtt_id , clientPass and pub_topic are required. mqtt_id is generated by printing the project ID, server location, registry ID and device ID in the required format:

```
mqtt_id = (char *)malloc(strlen("projects/") + strlen(project_id) + strlen("/
↪locations/us-central1/registries/") + strlen(registry_id) + strlen("/devices/") +␣
↪strlen(device_id) + 1);
sprintf(mqtt_id, "projects/%s/locations/us-central1/registries/%s/devices/%s",␣
↪project_id, registry_id, device_id);
```

`clientPass` is generated using a JSON web token (JWT) generator function, which requires the project ID and current time, and signs it with the private key:

```
clientPass = CreateJwt(project_id, timeClient.getEpochTime(), priv_key);
```

`pub_topic` is generated by printing the project ID and topic in the required format:

```
pub_topic = (char *)malloc(strlen("/devices/") + strlen(device_id) + strlen("/events
↪") + 1);
sprintf(pub_topic, "/devices/%s/events", device_id);
```

MQTT Server setting:

```
client.setServer(GOOGLE_MQTT_SERVER, GOOGLE_MQTT_PORT);
client.setPublishQos(MQTTQOS1);
client.waitForAck(true);
```

Connect to google cloud and publish messages:

```
if (client.connect(mqtt_id, clientUser, clientPass.c_str())){
   // ...
        for(int i = 0; i < count; i++){
      // ...
      sprintf(payload, "This is Ameba's %d message!!", i);
      ret = client.publish(pub_topic, payload);
      // ...
   }
   // ...
   client.disconnect();
}
free(mqtt_id);
free(pub_topic);
```

MQTT - Upload PM2.5 Data to LASS System

Intro to LASS

The LASS stands for "Location Aware Sensor System". It is an open project and was started only for the interest of public welfare. Find detailed introduction here.
Practically, LASS is based on MQTT protocol to collect all kinds of uploaded data, and for those who need these data can subscribe top as well.
Find more LASS information at their official hackpad.

Preparation

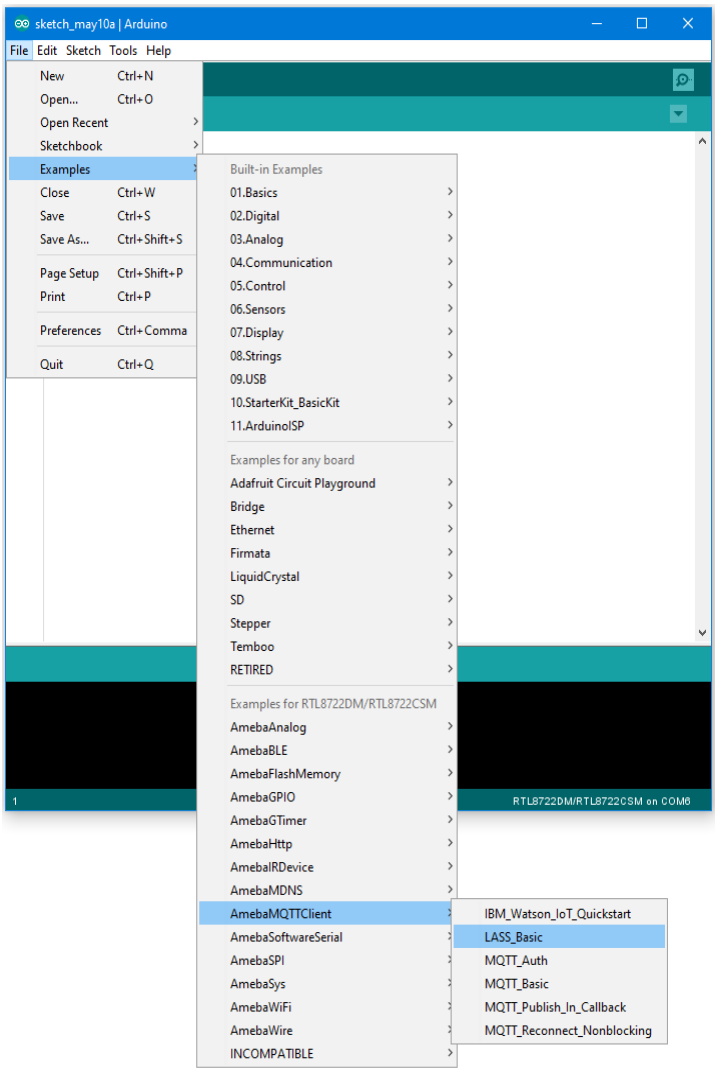
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x1

Example

In this example, we use applications mentioned at our website, including:

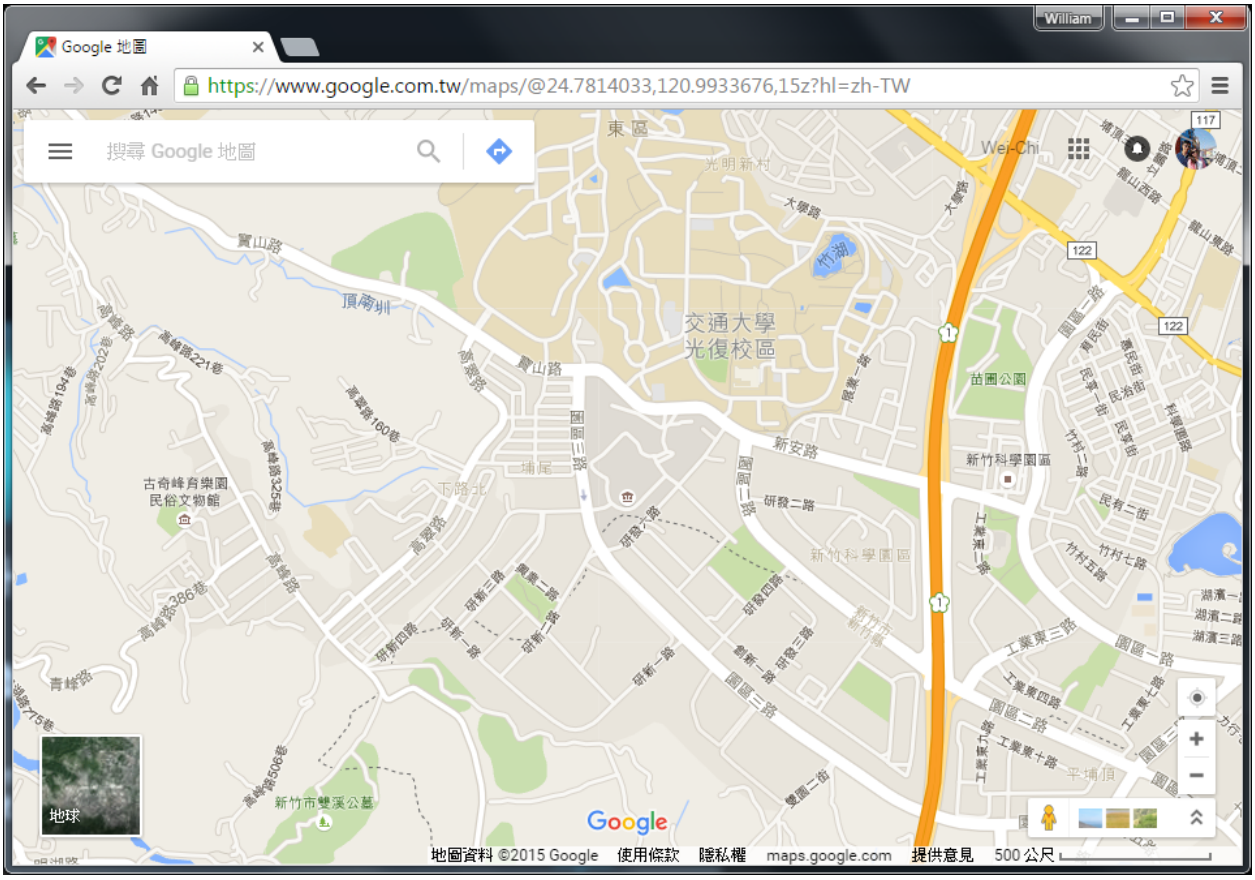
- MQTT: a MQTT-Broker to connect to LASS. The Client is "FT1_0XXXX", the XXXX are the four last digits of Ameba's Wi-Fi MAC, and the outTopic is "LASS/Test/Pm25Ameba/**clientID**", where clientID is the actual Ameba's MQTT client ID.
- NTP: uploaded data must have time notation
- PM2.5: uploaded data includes PM2.5 information

Open the example. "`File`" -> "`Examples`" -> "`AmebaMQTTClient`" -> "`lass_basic`"

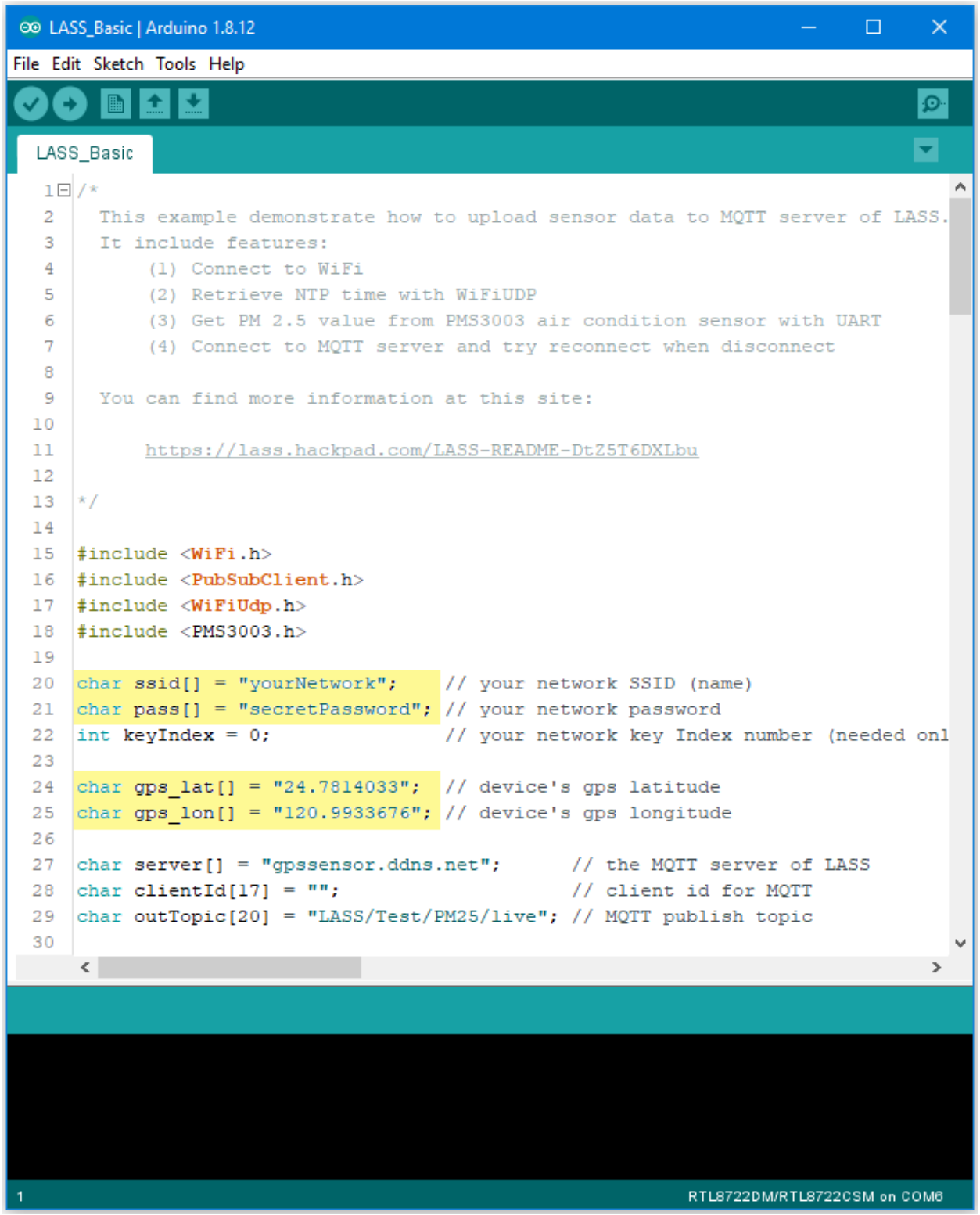


This example requires internet connection, so make sure you fill in SSID and PASS into AP information that you wish to connect.

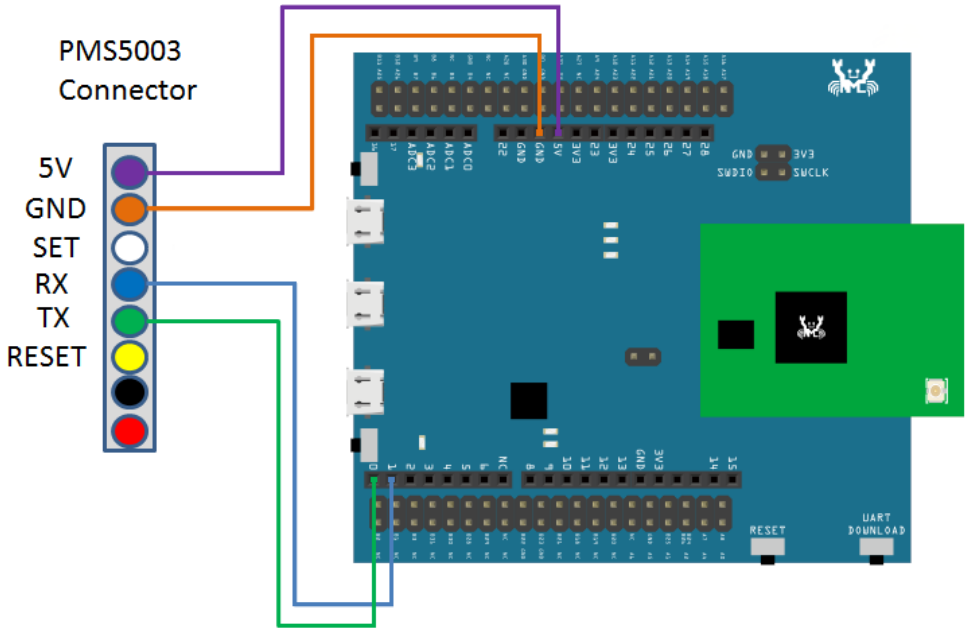
Also, LASS requires GPS information. There is no GPS sensor included in this example, so you must manually provide GPS information. Use Google Map to find the coordinates you plan to place your Ameba. You can see in this example that the latitude is 24.7814033, and the longitude is 120.9933676



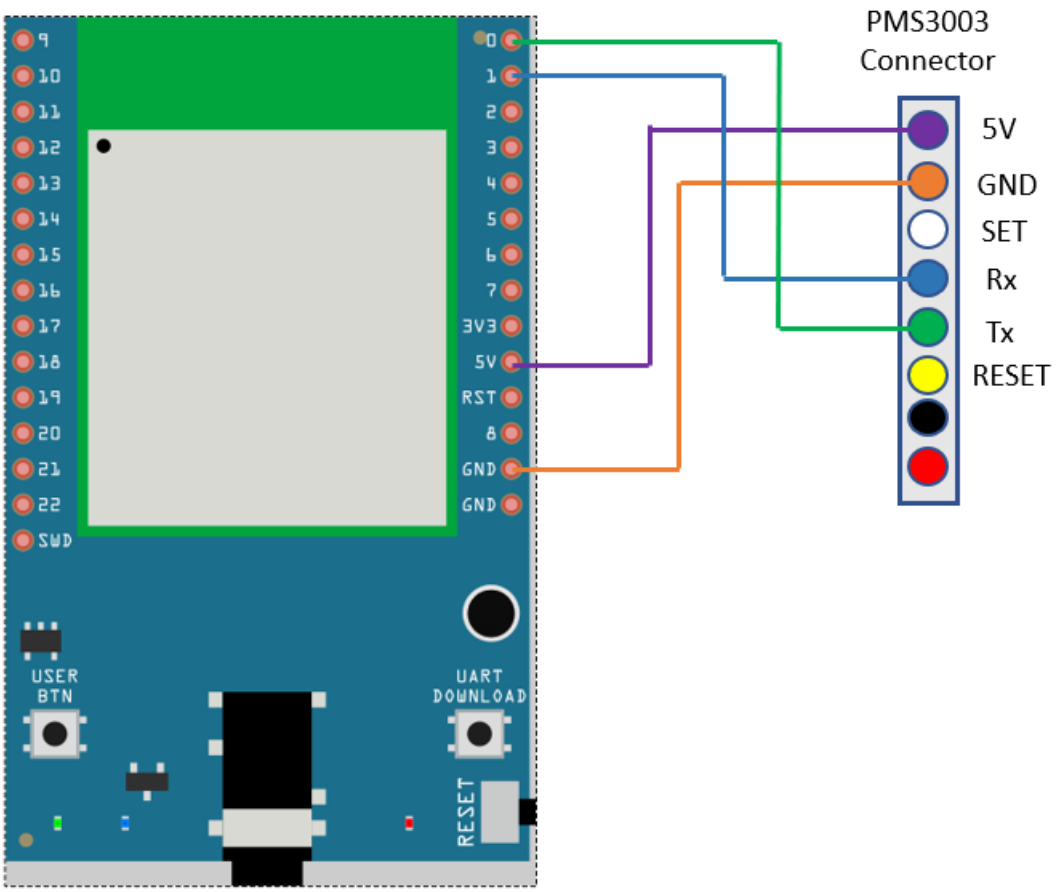
Fill in GPS info at `gps_lat` and `gps_lon`.



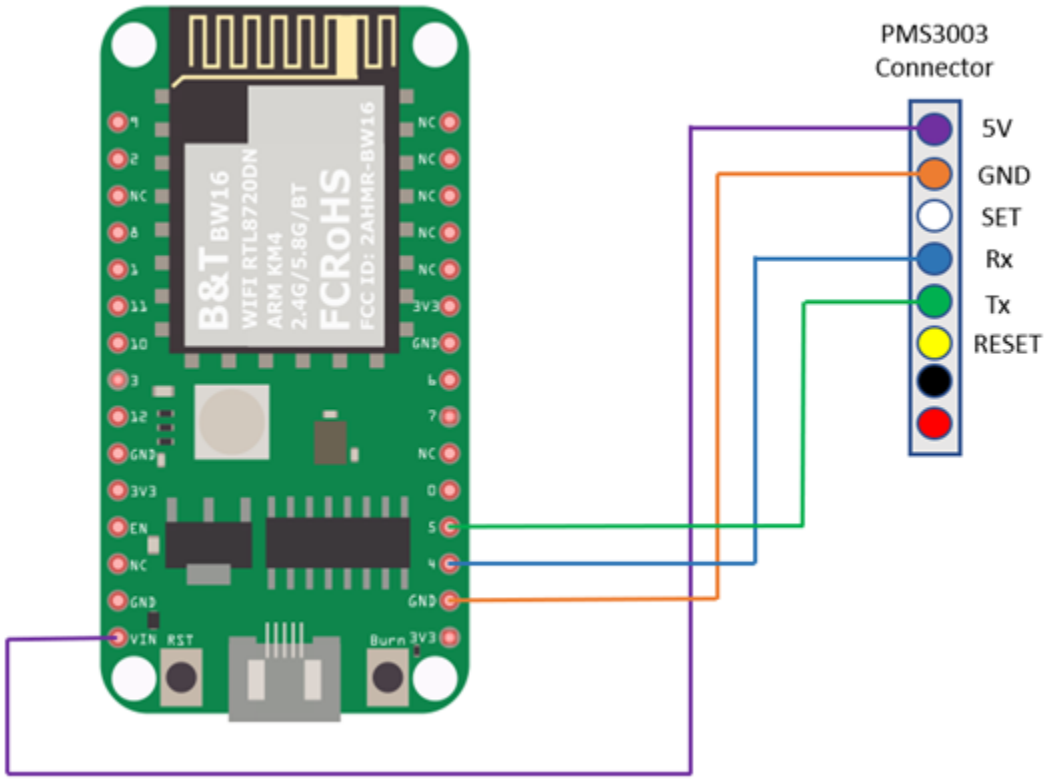
Then connect sensors according to UART-PlanTower PMS3003 wiring example.
AMB21 / AMB22:


PMS5003
Connector
5V
GND
SET
RX
TX
RESET
RESET
UART
DOWNLOAD

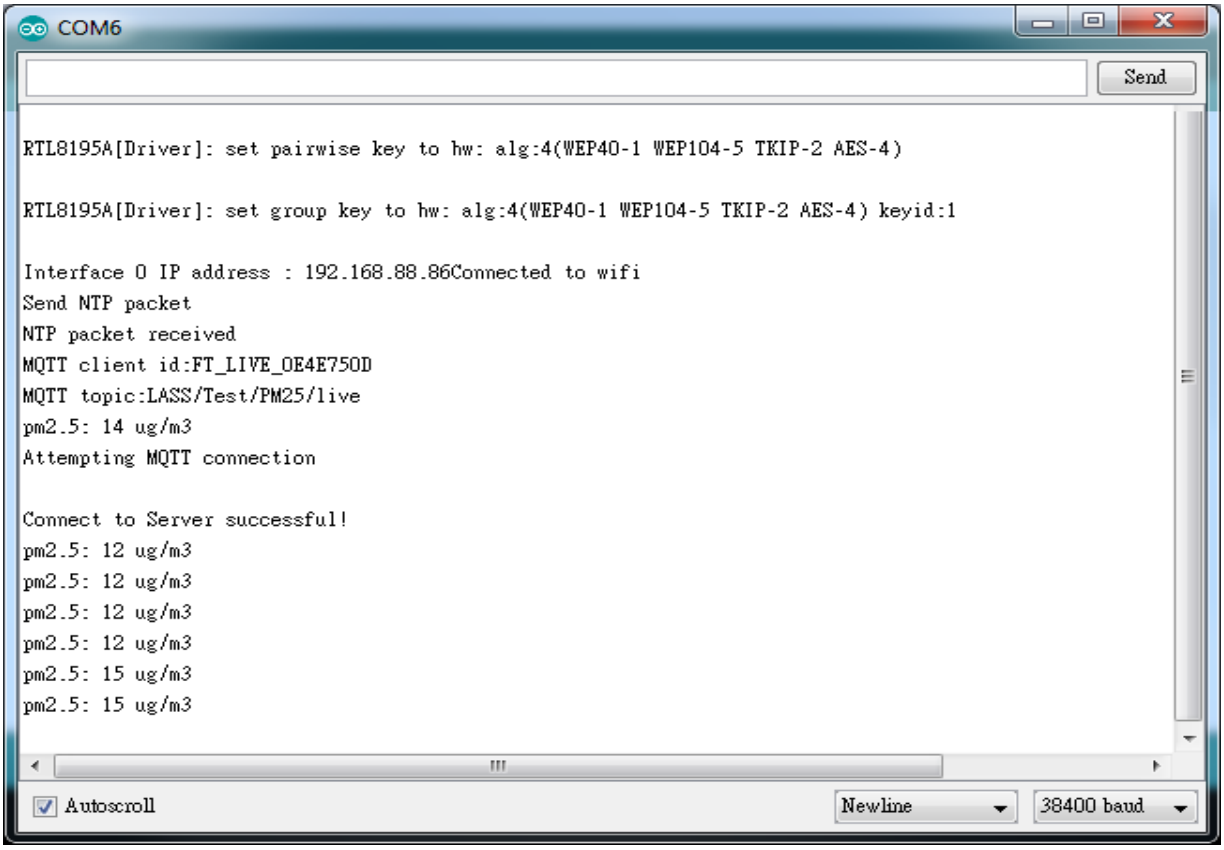
AMB23:


PMS3003
Connector
5V
GND
SET
Rx
Tx
RESET
9
10
11
12
13
14
15
16
17
18
19
20
21
22
SWD
0
1
2
3
4
5
6
7
3V3
5V
RST
8
GND
GND
USER
BTN
UART
DOWNLOAD
RESET

BW16:



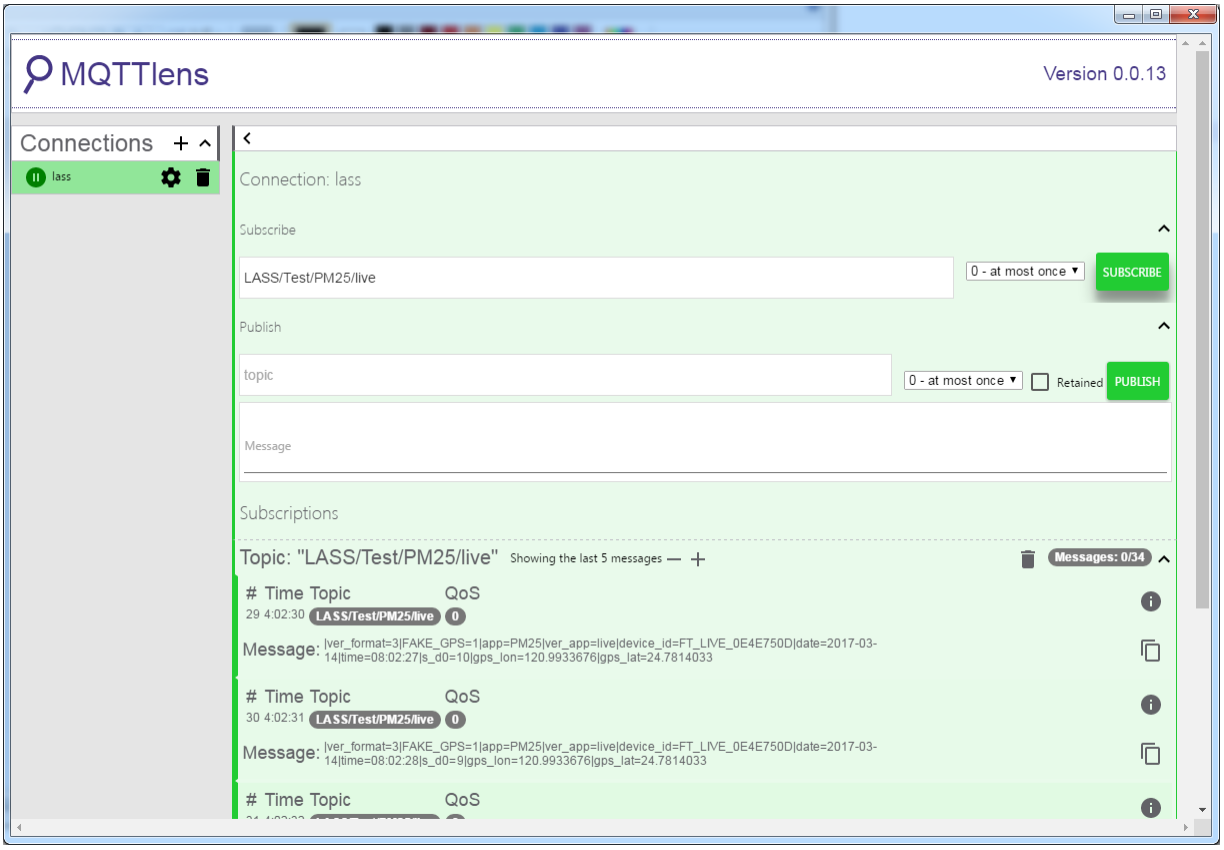
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba will attempt to read PM2.5 data every minute and upload it to LASS MQTT-Broker. Open Serial Monitor to see the uploaded data, including client id, topic, and current PM2.5 status.



We can also use MQTTlens to verify if the data is properly uploaded.

Enter “gpssensor.ddns.net” as the MQTT-Broker server and “LASS/Test/PM25/live” as the subscribe topic to receive data.

The time uses UTC format, and the PM2.5 data stores in s-d0. In the figure, s_d0 = 9 represents that the PM2.5 is 9, meaning that the entire publish/subscribe process is working successfully.



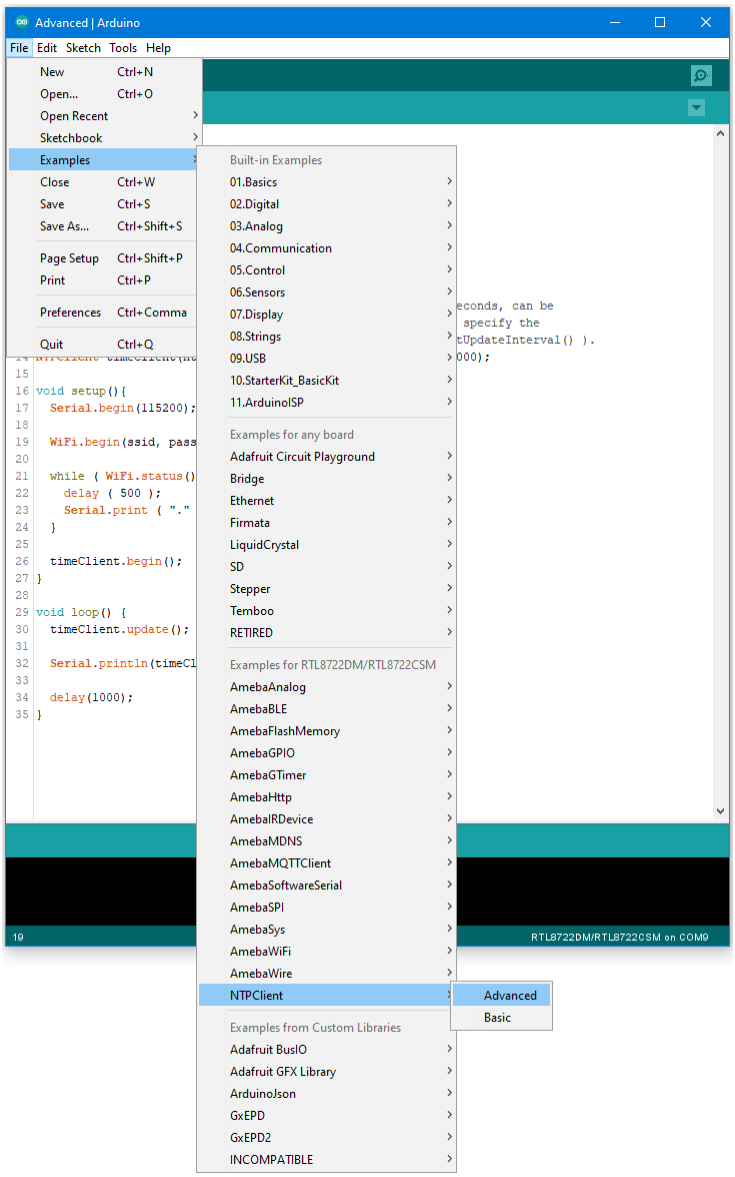
NTP - Retrieve Universal Time (UTC) by NTPClient library

Preparation

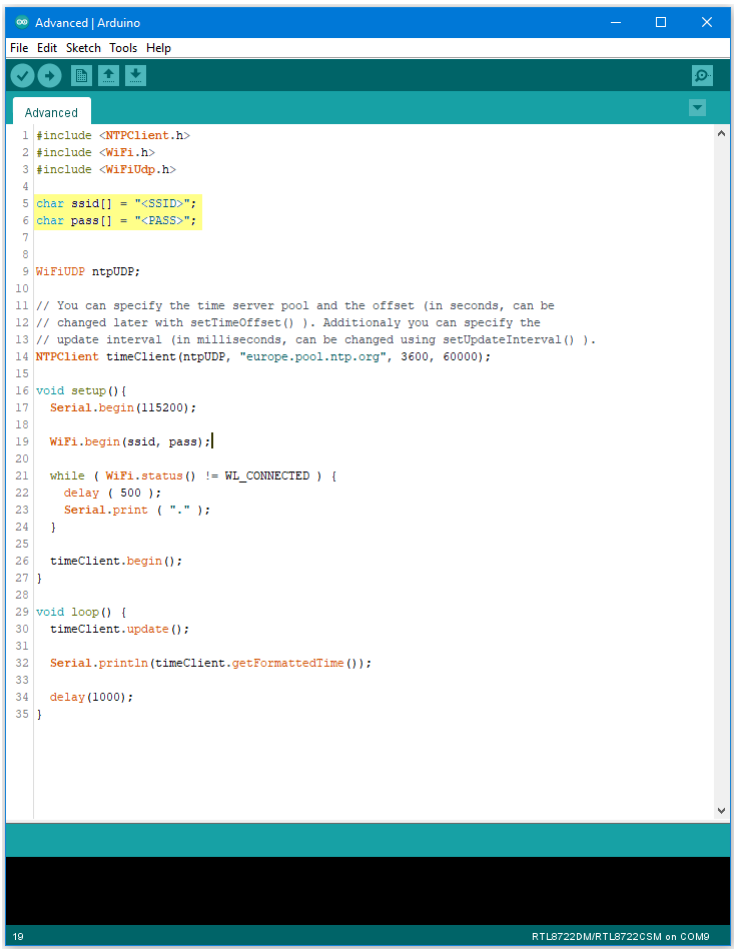
- AmebaD [RTL8722DM / RTL8722CSM / RTL8722DM MINI] x 1

Example

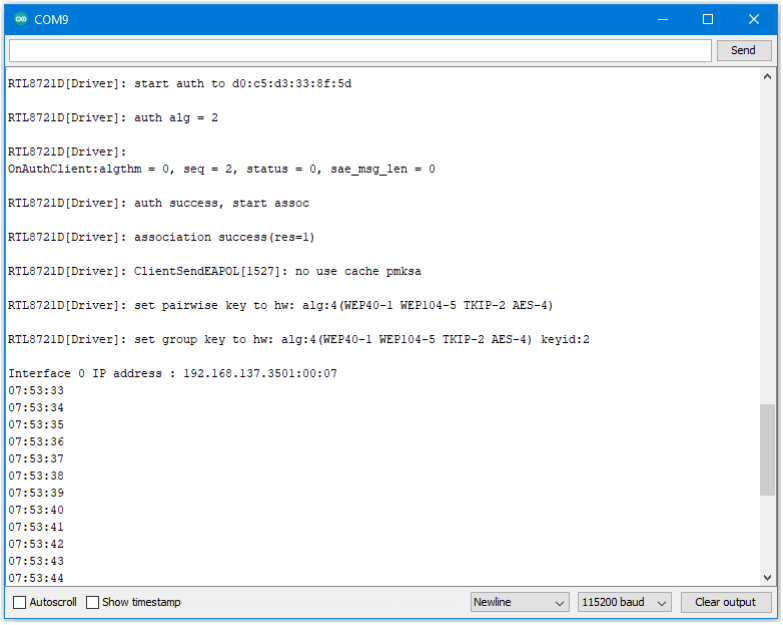
In this example, we use an NTP client to sync with NTP servers using UDP and keep track of time locally. Open the example. “`File` -> “`Examples`”-> “`NTPClient`” -> “`Advanced`”



Modify the highlighted code section (ssid, password) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi, gets the UTC time from the NTP server, and prints out the current time with time zone offset to the serial monitor.



Code Reference

Configure NTP client:

The NTPClient needs to use a UDP client for communications. A WiFiUDP client is declared and passed to the NTPClient constructor, along with an NTP server address, time zone offset in seconds, and update interval in milliseconds. If detailed configuration is not needed, just passing in the UDP client is also sufficient, refer to the "NTPClient" -> "Basic" example.

```
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 3600, 60000);
```

Start NTP client:

After connecting to WiFi, the NTPClient is started using the `begin()` function, which causes the client to sync with the NTP server and get the UTC time.

```
WiFiUDP ntpUDP;
timeClient.begin();
```

Get local time:

`getFormattedTime()` is used to format the received UTC time into the local time zone. `update()` is called every loop so that the NTPClient will sync with the NTP server once every update interval.

```
timeClient.update();
timeClient.getFormattedTime();
```

WiFi - Approximate UDP Receive Delay

Materials

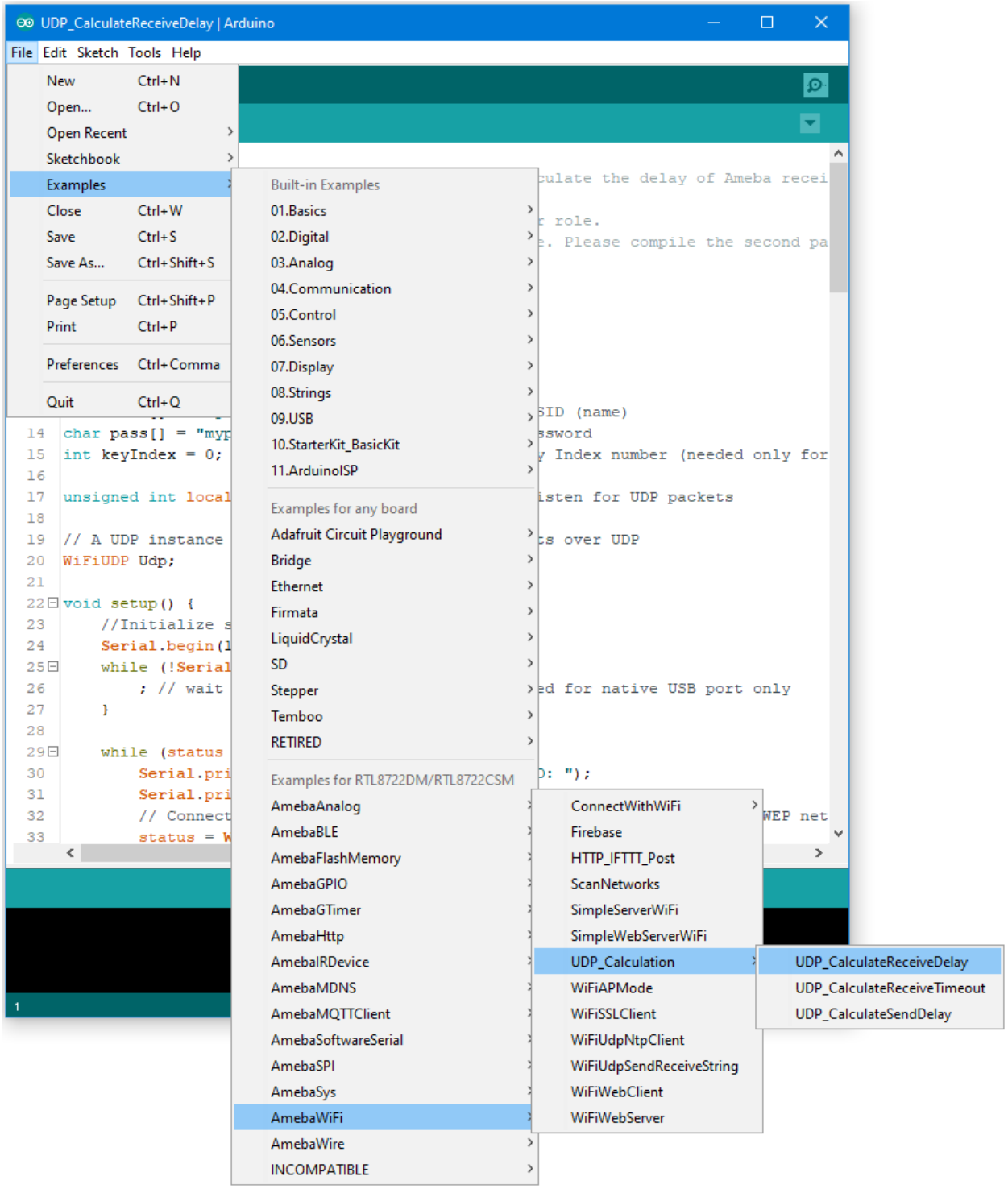
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

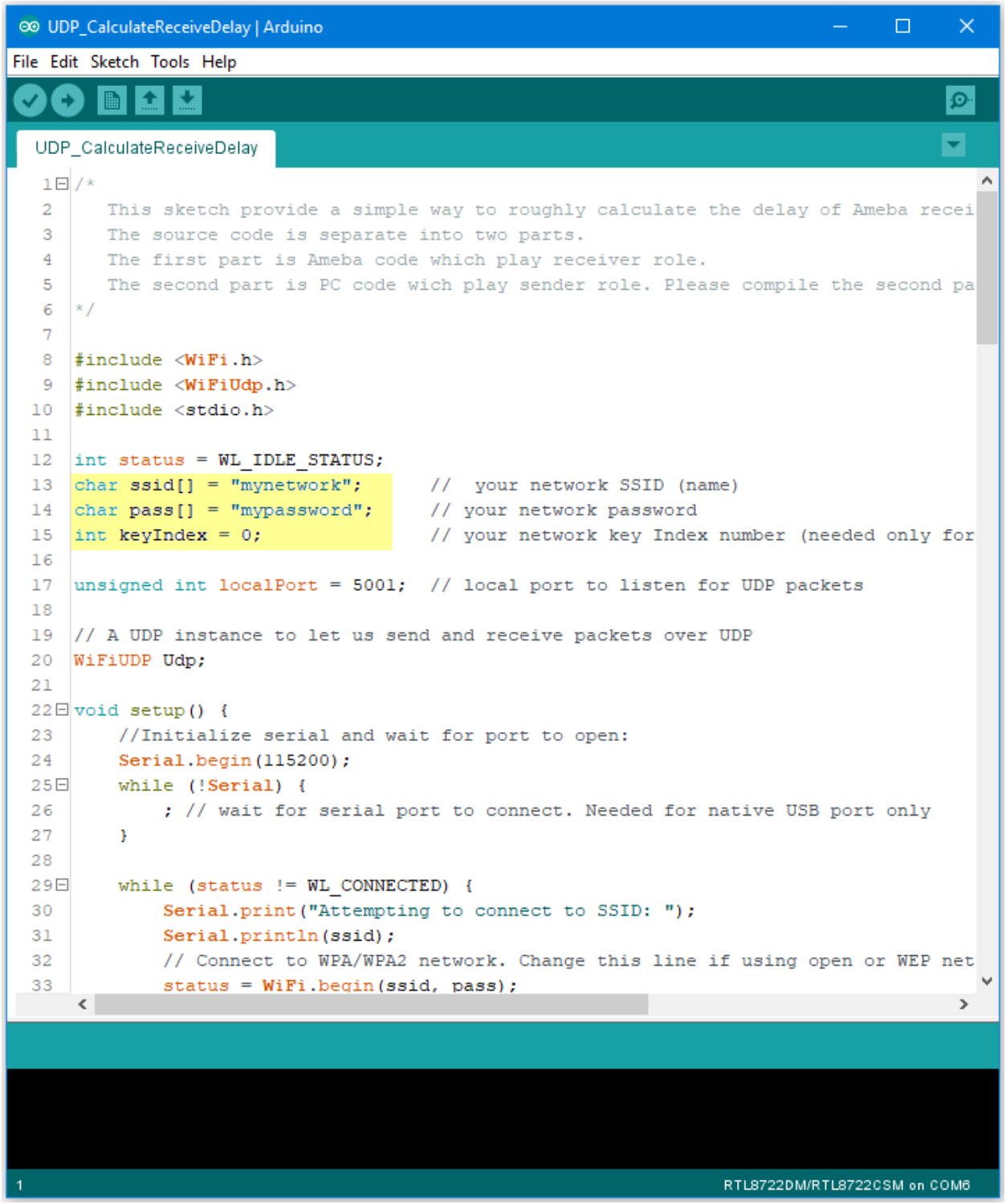
This example uses Ameba to receive UDP packets from a computer and calculates the UDP receive delay.

Ameba Preparation

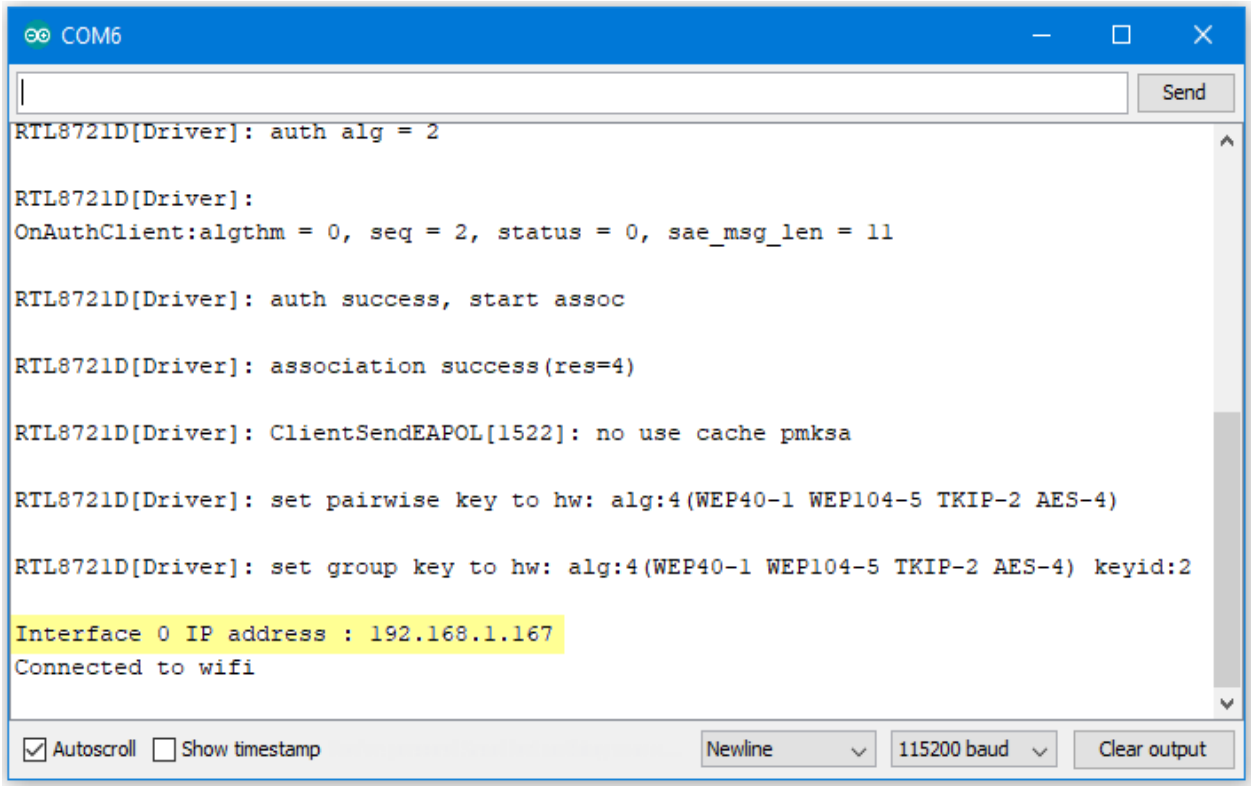
Open the "CalculateUdpReceiveDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation`" -> "`CalculateUdpReceiveDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



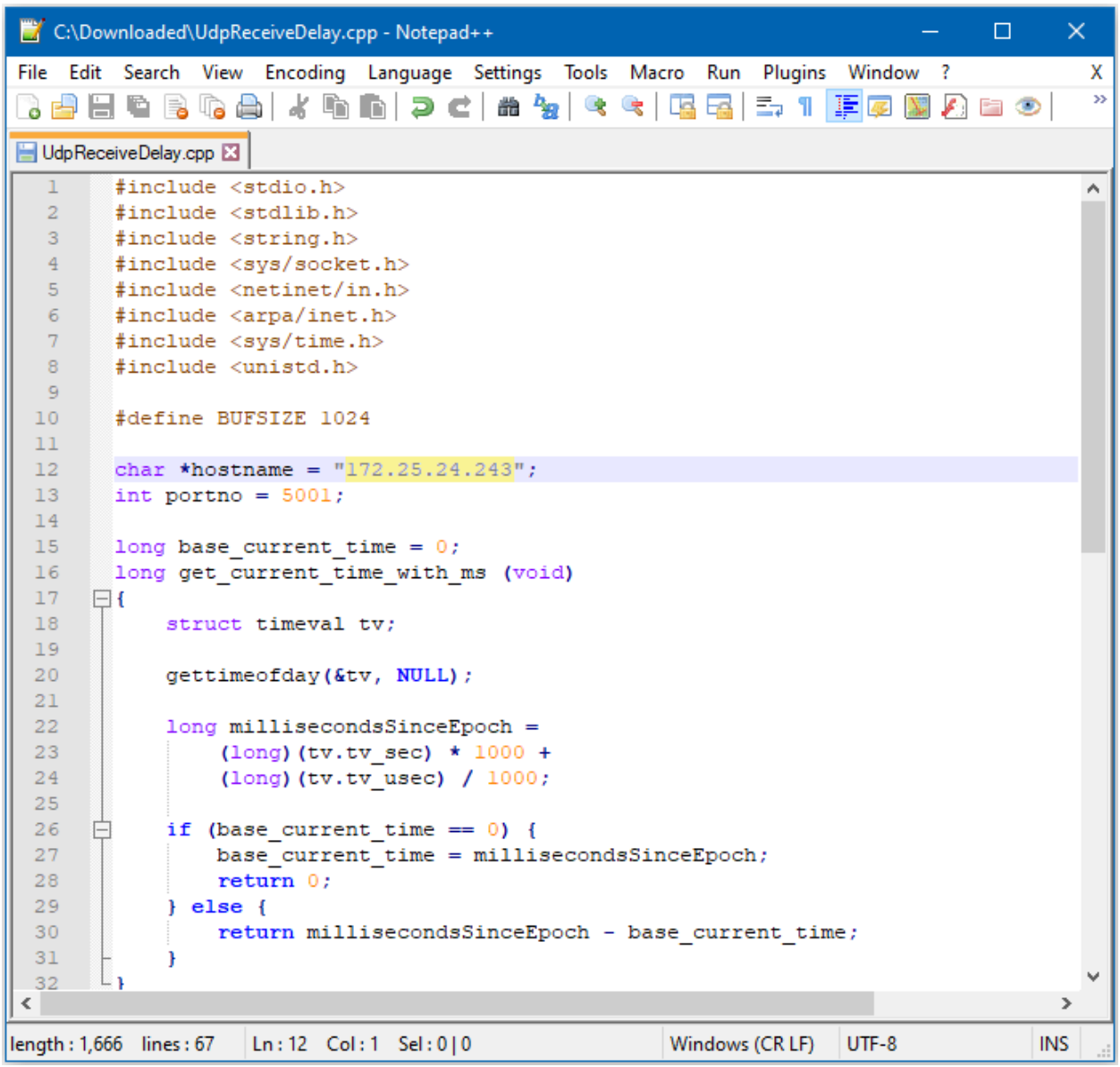
Upload the code and press the reset button on Ameba once the upload is finished. Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

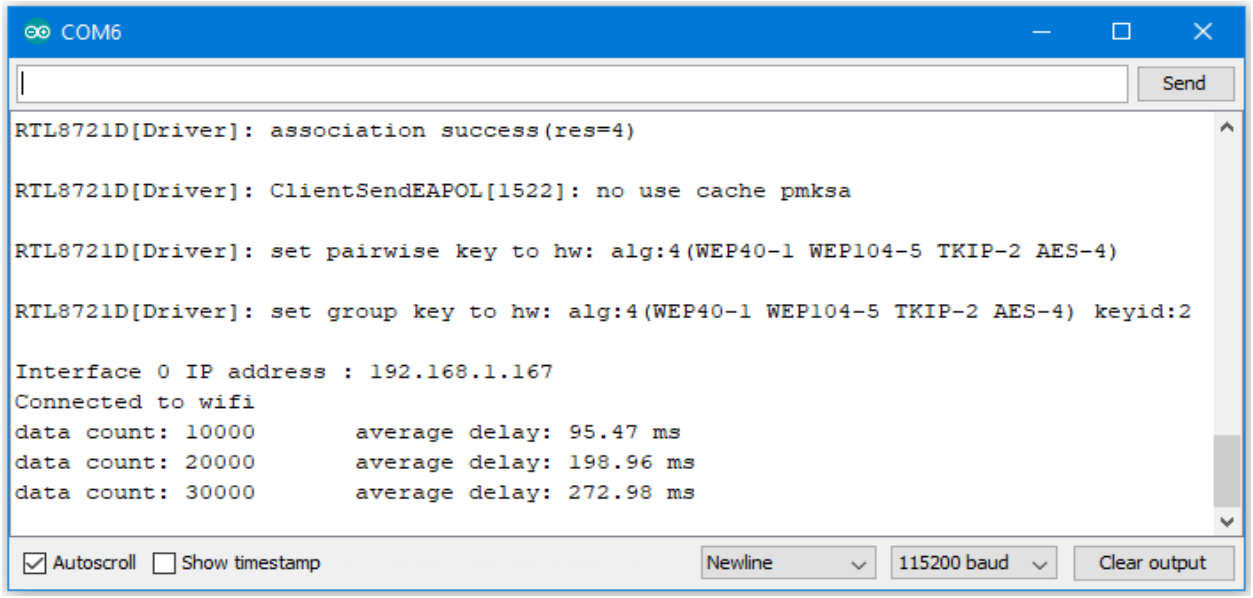
Follow the instructions there to install it. Next, from the "CalculateUdpReceiveDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveDelay.cpp", and use the command "g++ UdpReceiveDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpDelay.exe file, and the computer will begin to send packets to Ameba. Once 10000 packets have been received, Ameba will calculate the average delay and print out the result to the serial monitor. It may take up to a few minutes for 10000 packets to be sent.



WiFi - Approximate UDP Sending Delay

Materials

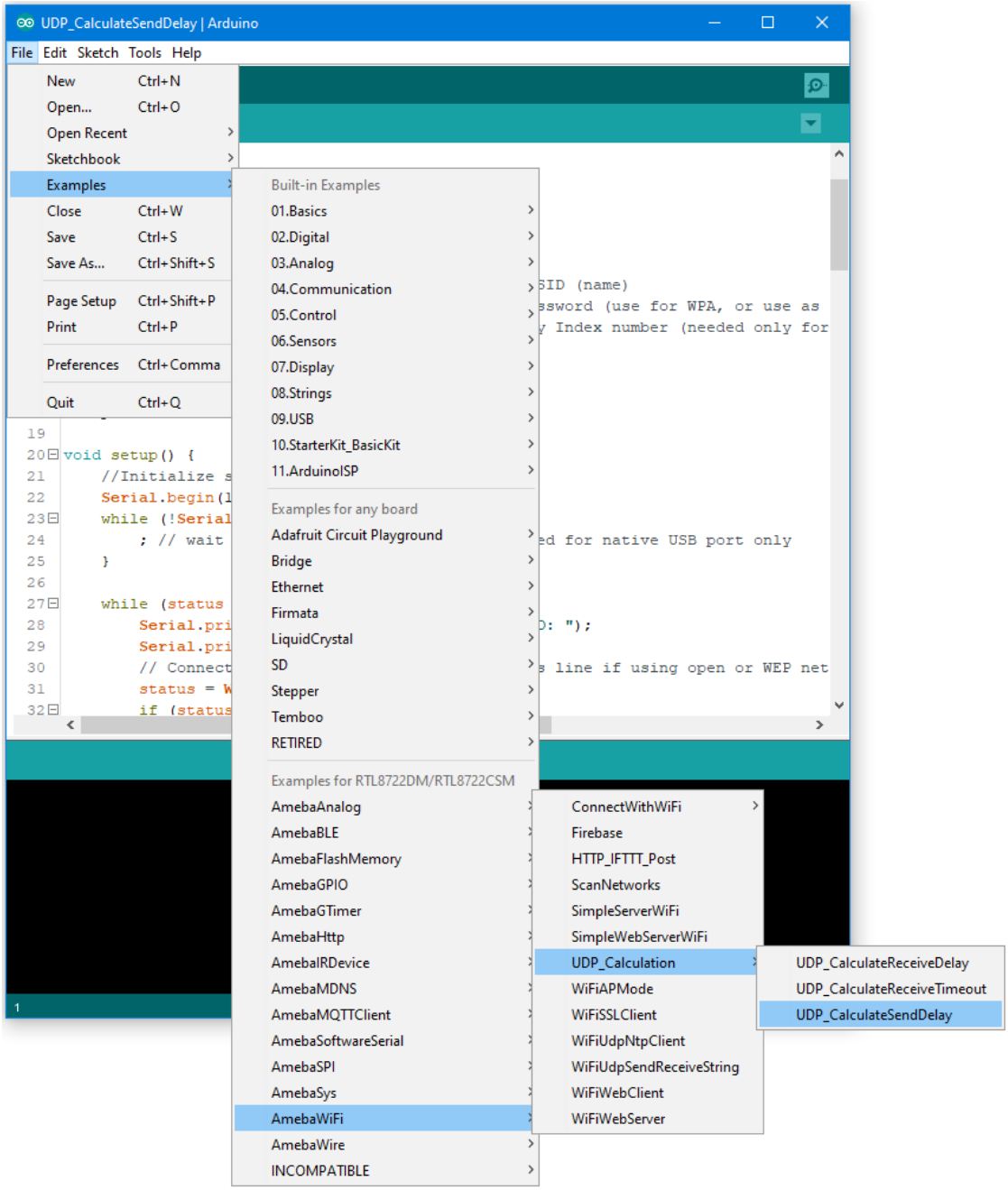
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

This example uses Ameba to send UDP packets to a computer and calculates the UDP sending delay.

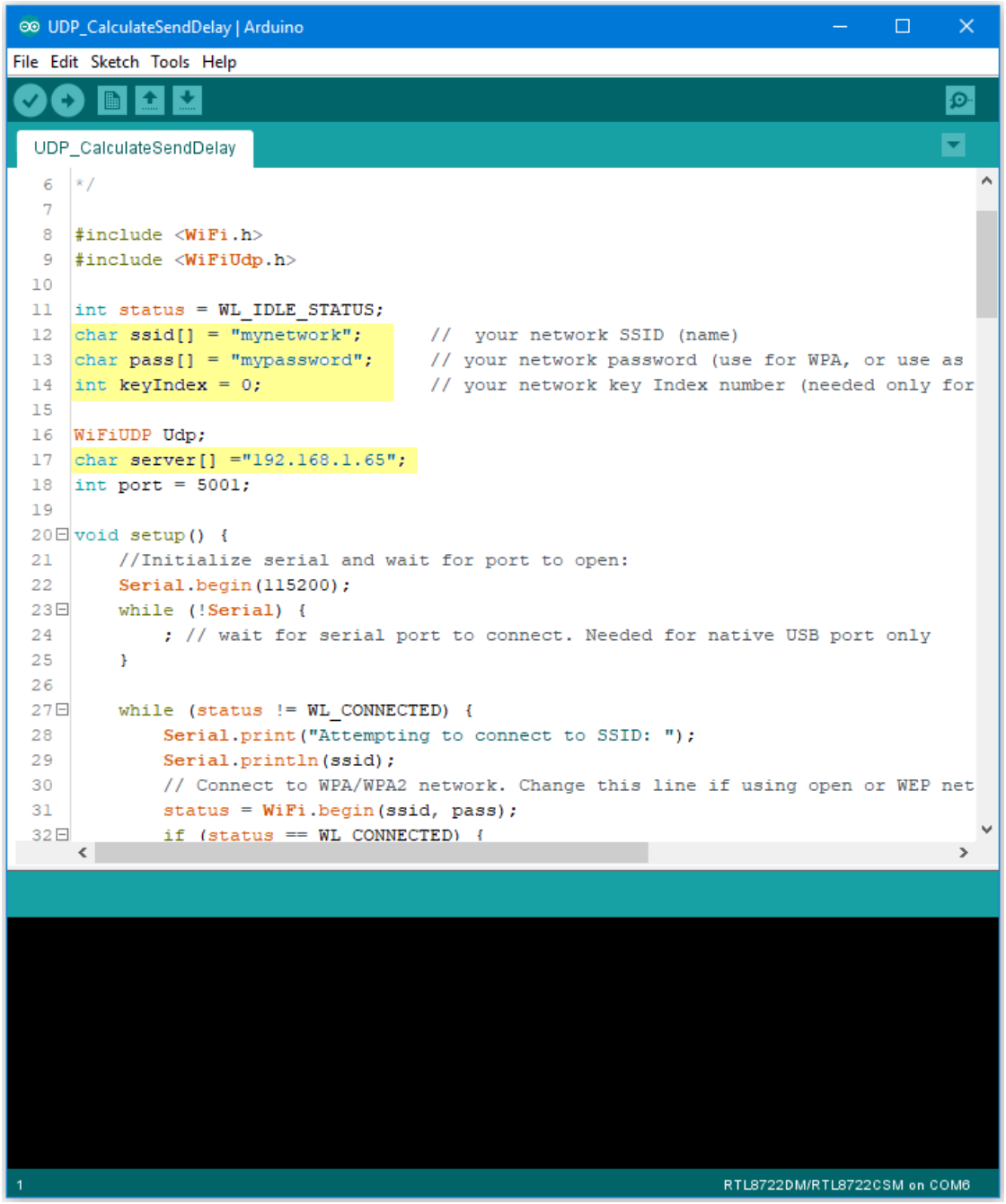
Ameba Preparation

Open the "CalculateUdpSendDelay" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation`" -> "`CalculateUdpSendDelay`".



In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.

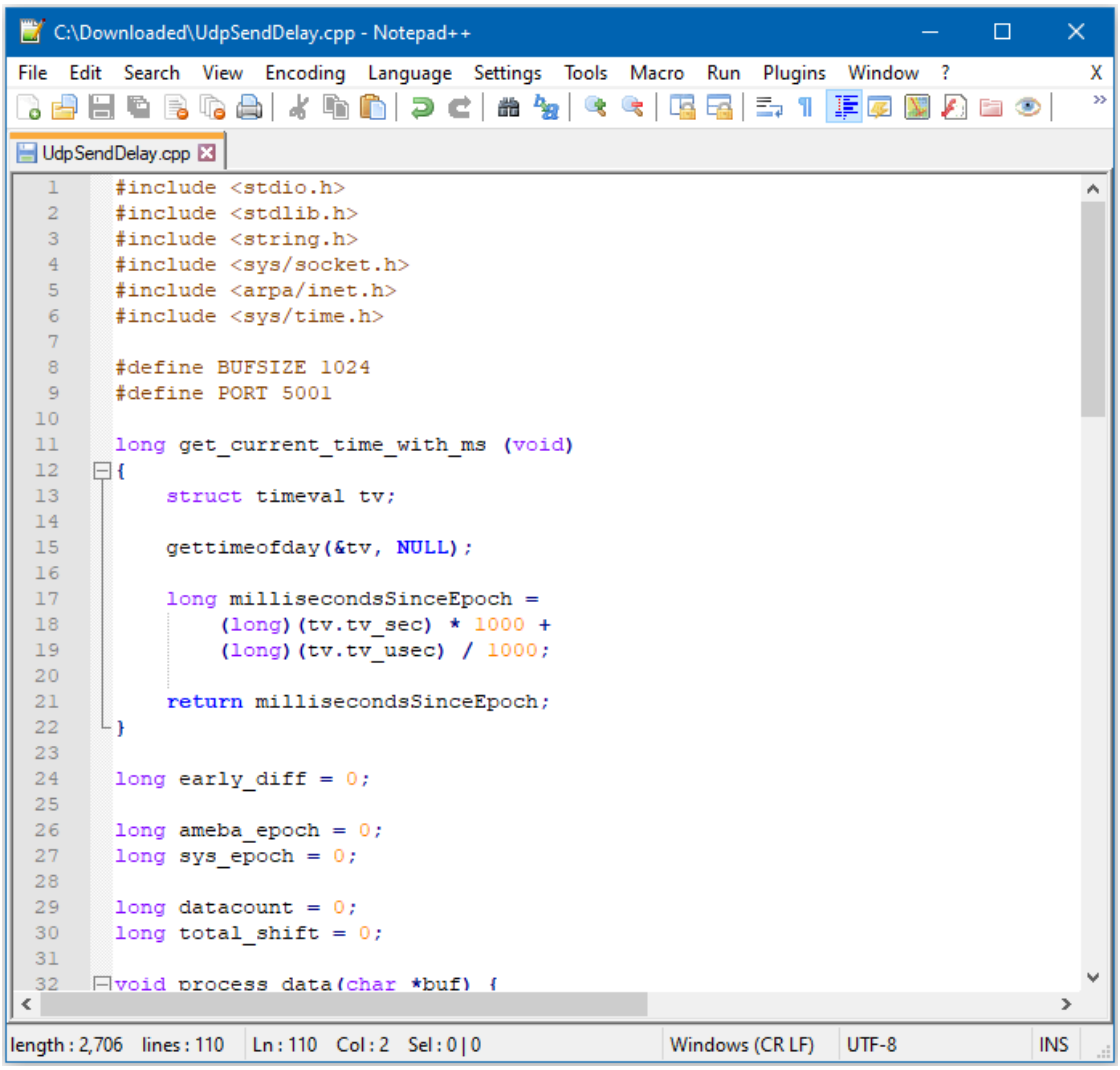
The server variable also needs to be changed to match the IP address of your computer. You can find the IP address using the “ipconfig” command in a terminal window.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpSendDelay" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file and rename the file to "UdpSendDelay.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpSendDelay.cpp", and use the command "g++ UdpSendDelay.cpp -o UdpDelay" to compile the code. A file named "UdpDelay.exe" will be created in the same directory.

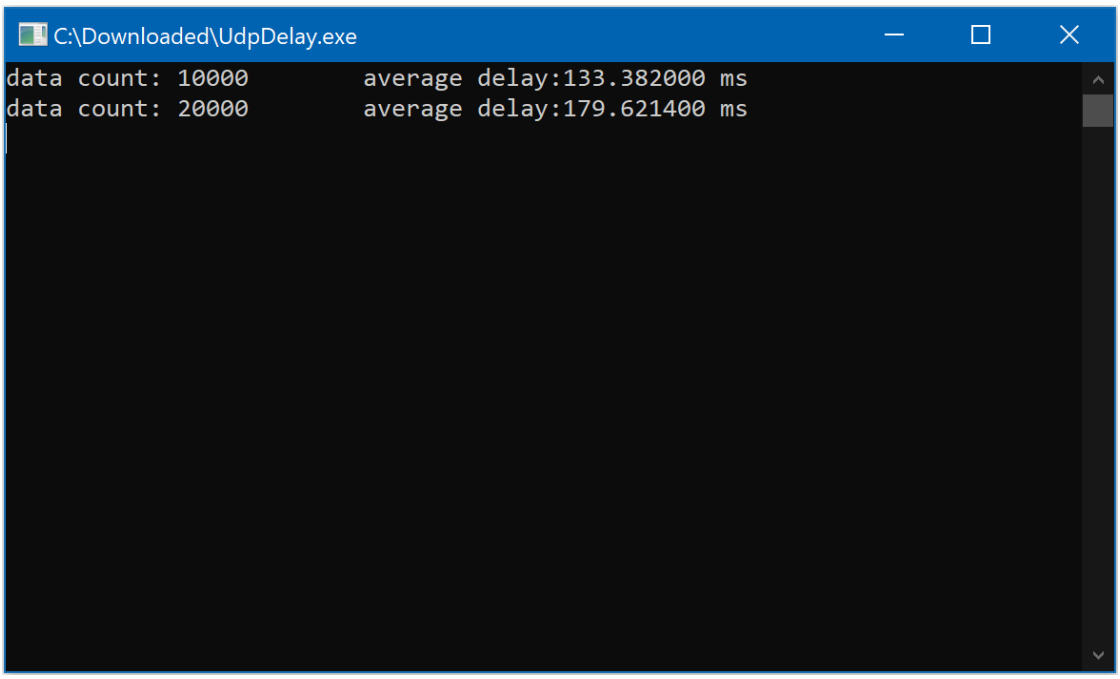
Running the Example

First, on the computer, run the UdpDelay.exe file, and the computer will begin to listen for packets from Ameba.

Next, compile and upload the code from the Arduino IDE to Ameba and press the reset button when the upload is complete.

The Ameba will begin to send UDP packets to the computer. Once 10000 packets have been received, the computer will calculate the average delay and print out the result.

It will take some time for 10000 packets to be sent.



WiFi - Approximate UDP Receive Timeout

Materials

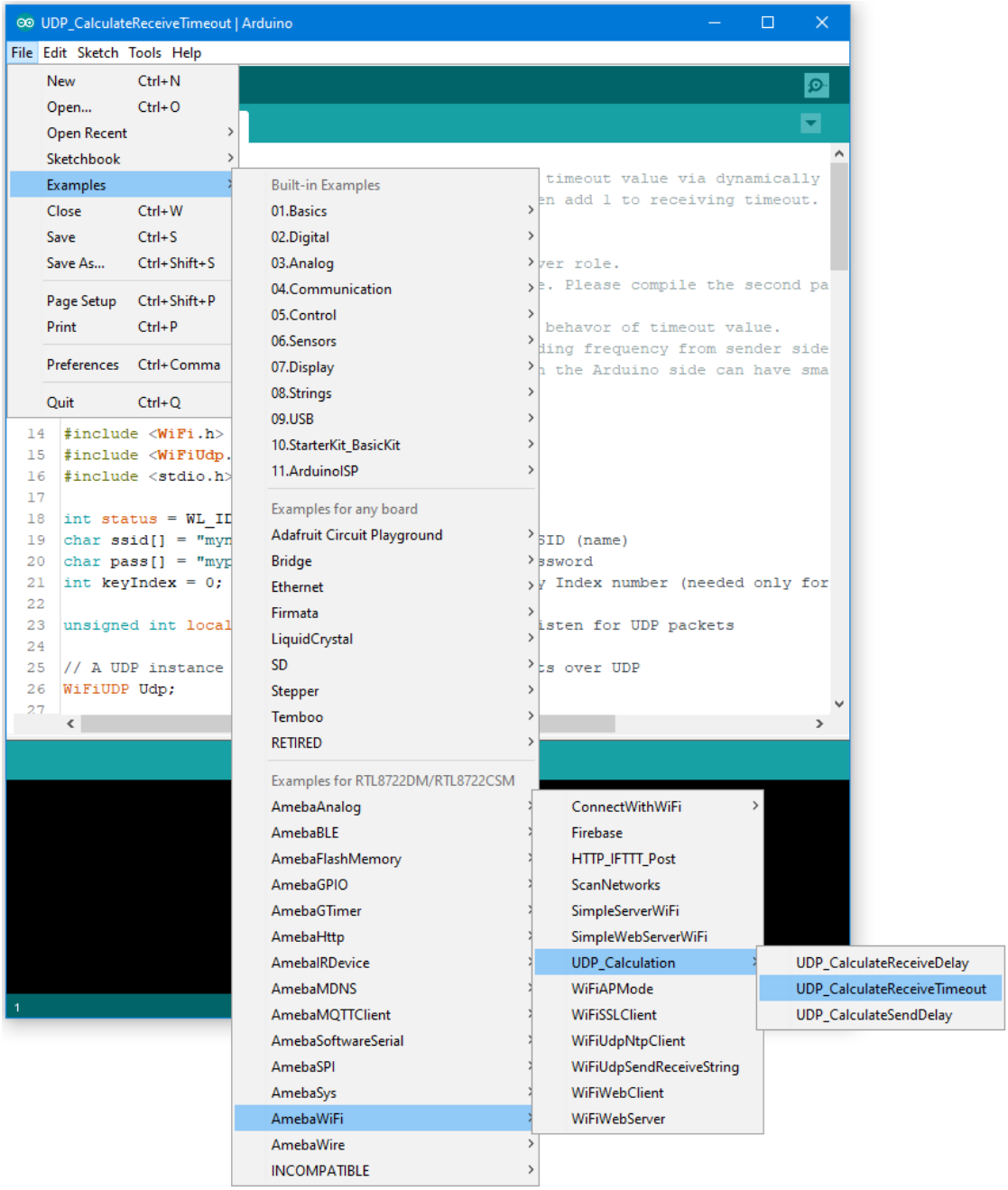
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Windows computer connected to same network

Example

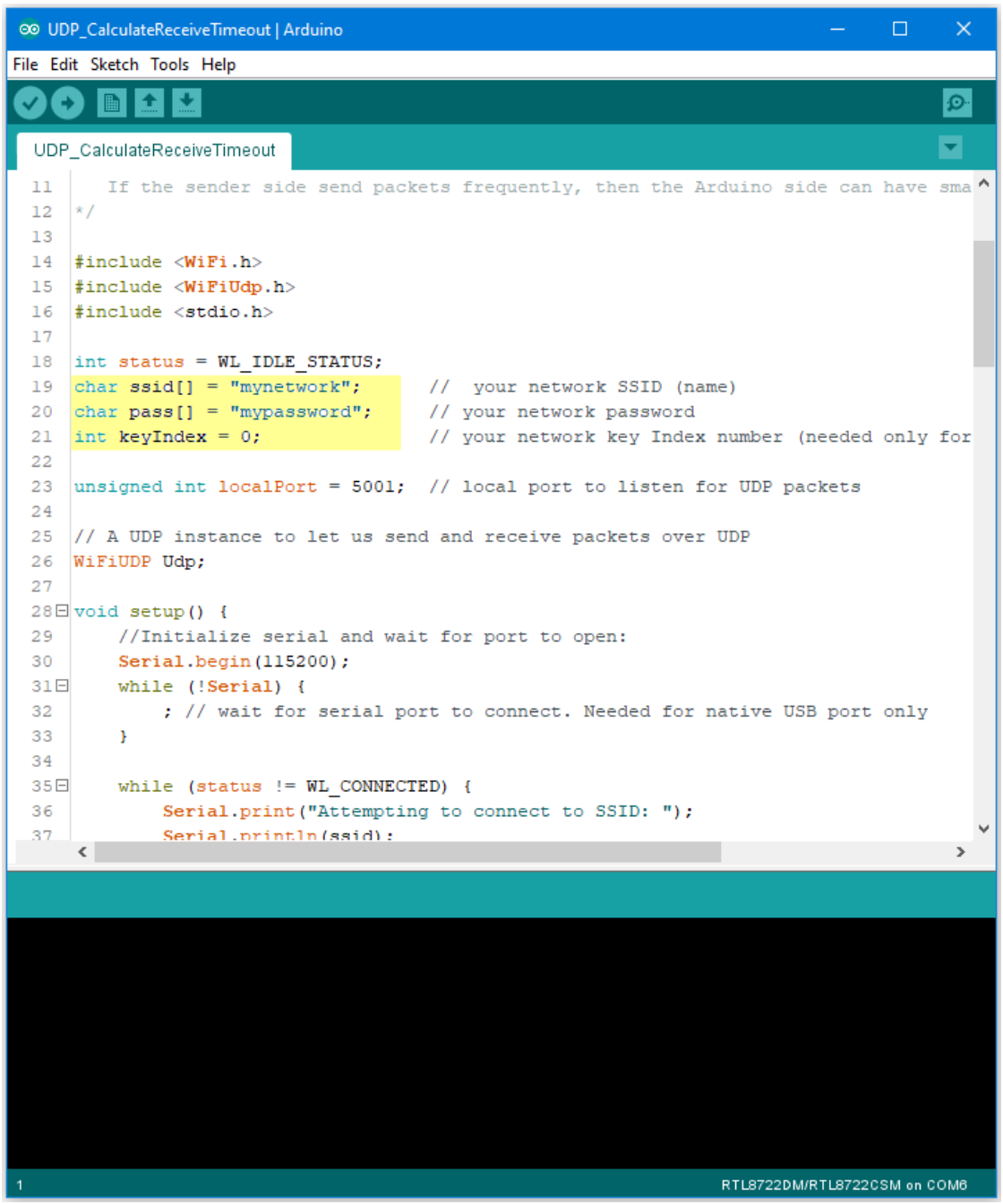
This example uses Ameba to receive UDP packets from a computer and calculates the allowed UDP receive timeout setting.

Ameba Preparation

Open the "CalculateUdpReceiveTimeout" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`UDP_Calculation `" -> "`CalculateUdpReceiveTimeout`".

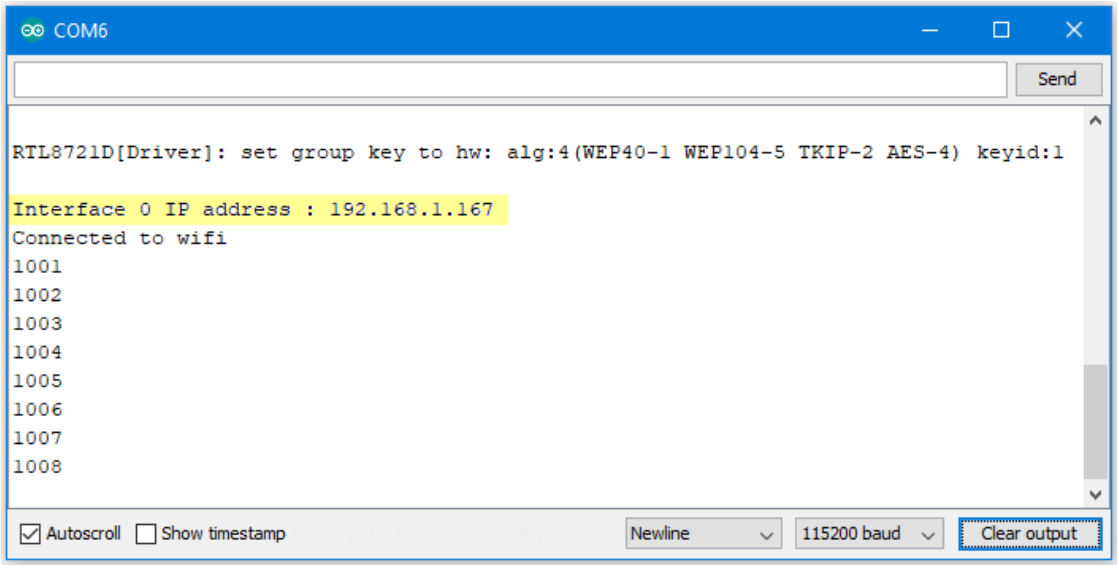


In the sample code, modify the highlighted section to enter the information required (ssid, password, key index) to connect to your WiFi network.



Upload the code and press the reset button on Ameba once the upload is finished.

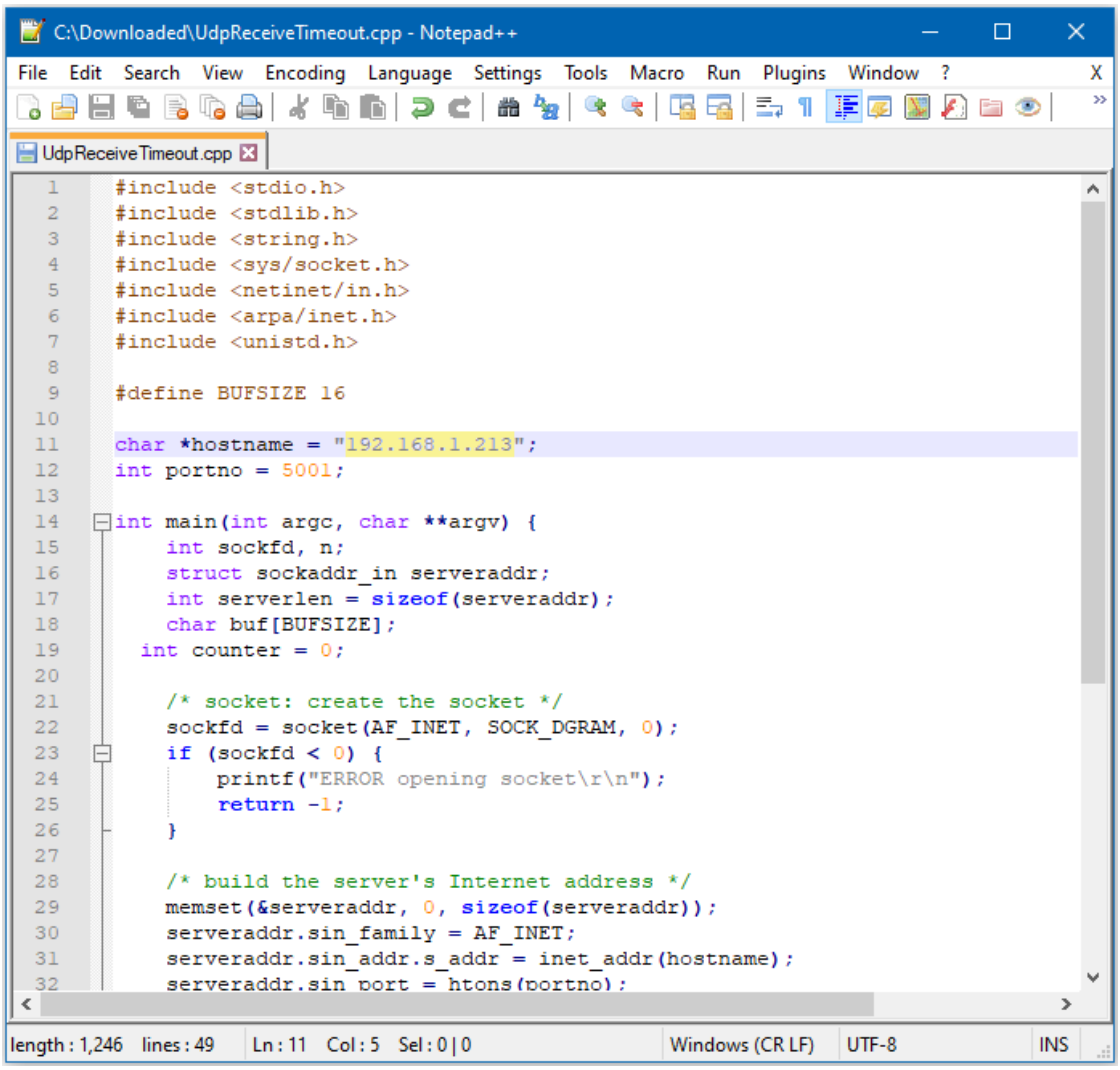
Open the serial monitor in Arduino IDE and take note of the IP address assigned to Ameba.



Computer Preparation

On the computer, Cygwin will be required to compile the code to send the UDP packets. Cygwin can be downloaded from https://www.cygwin.com/

Follow the instructions there to install it. Next, from the "CalculateUdpReceiveTimeout" Arduino example, copy the code from the bottom between "#if 0" and "#endif", into a new text file, change the hostname to the IP address assigned to Ameba, and rename the file to "UdpReceiveTimeout.cpp".



Next, open a Cygwin terminal, change the working directory to the location of "UdpReceiveTimeout.cpp", and use the command "g++ UdpReceiveTimeout.cpp -o UdpTimeout" to compile the code. A file named "UdpTimeout.exe" will be created in the same directory.

Running the Example

Reset the Ameba, wait for the WiFi to connect, and check that the IP address remains the same. On the computer, run the UdpTimeout.exe file, and the computer will begin to send packets continuously to Ameba.

The timeout value is set to 1000ms initially. For each packet received successfully, Ameba decreases the timeout value. The next packet must be received within the timeout period, otherwise Ameba registers a failed packet and increases the timeout value. Open the serial monitor and observe the timeout value converge to a minimum value.

WiFi - Connect to WiFi networks

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Procedure

There three common encryption type in WiFi connection. The first one is “OPEN”, which means there is no password needed to connect to this network. The second type of encryption is WPA, which requires the correct password to access. The third type is WEP, which requires a hexadecimal password and a keyindex.

In the following, we will give a brief introduction on how to establish WiFi connection with these three types of encryption on Ameba.

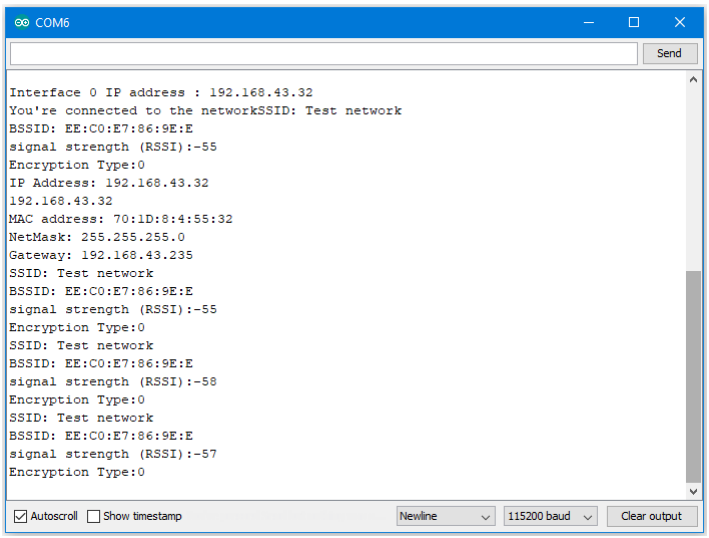
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”.

- Open (WiFi connection without password)

  Open the “ConnectNoEncryption” example in “File” -> “Examples” -> “AmebaWiFi” -> “ConnectWithWiFi” -> “ConnectNoEncryption”

  In the sample code, modify “ssid” to be the same as the WiFi SSID to be connected to.

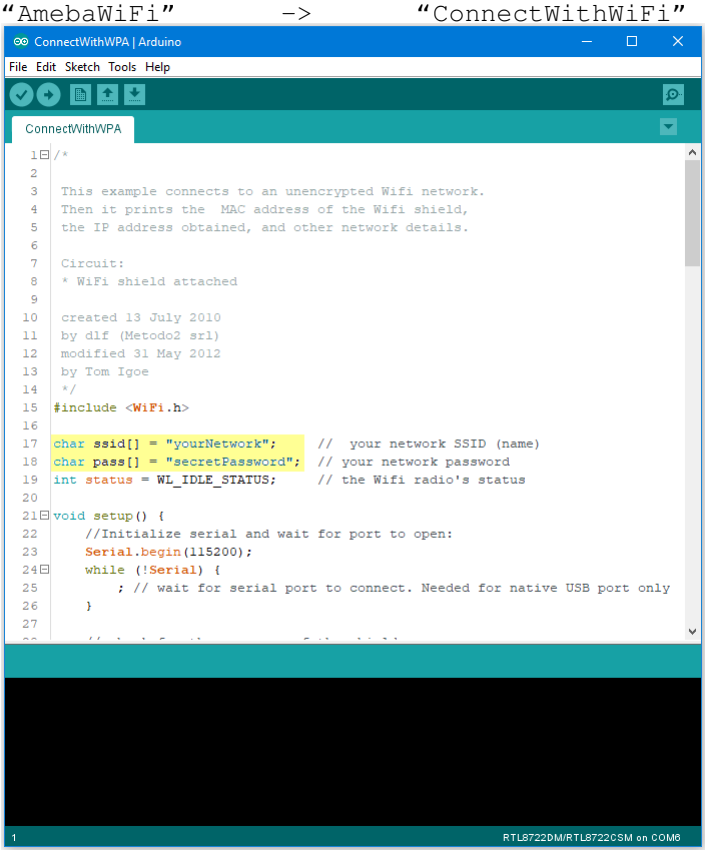
  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message “You’re connected to the networkSSID: XXXXX”, and the information of this WiFi connection is printed in the



serial monitor every 10 seconds.

- WiFi connection with WPA encryption

  Open the "ConnectWithWPA" example in "File" -> "Examples" -> "AmebaWiFi" -> "ConnectWithWiFi" -> "ConnectWithWPA"



  In the sample code, modify "ssid" to the WiFi SSID to be connected to and "pass" to the network password.

  Next, upload the sample code, and press the reset button on Ameba. Then you will see a message "You're connected to the networkSSID: XXXXX", and the information of this WiFi connection is printed in the serial monitor every 10 seconds.

- WiFi connection with WEP encryption

Open the “ConnectWithWEP” example in “File” -> “Examples” -> “AmebaWiFi” -> “ConnectWithWiFi” -> “ConnectWithWEP”

In the sample code, modify “ssid” to the SSID to be connected, “key” to the hexadecimal password, “keyIndex” to your key index number.

Next, upload the sample code, and press the reset button on Ameba. Then you will see a message “You’re connected to the networkSSID: XXXXX”, and the information of this WiFi connection is printed in the IDE every 10 seconds.

Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.encryptionType()` to get the encryption type of the WiFi connection.
https://www.arduino.cc/en/Reference/WiFiEncryptionType
Use `WiFi.BSSID()` to get the MAC address of the router you are connected to.
https://www.arduino.cc/en/Reference/WiFiBSSID
To get the information of Ameba:
Use `WiFi.macAddress()` to get the MAC address of Ameba.
https://www.arduino.cc/en/Reference/WiFiMACAddress

Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFi.subnetMask()` to get the subnet mask.
https://www.arduino.cc/en/Reference/WiFiSubnetMask
Use `WiFi.gatewayIP()` to get the WiFi shield's gateway IP address.
https://www.arduino.cc/en/Reference/WiFiGatewayIP

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

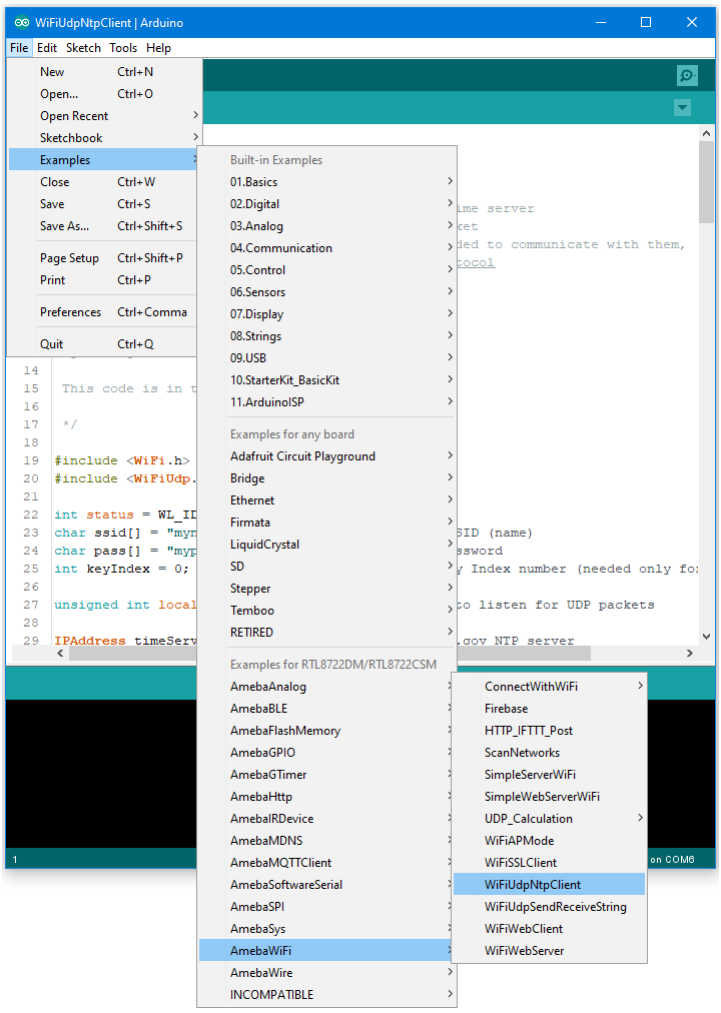
WiFi - Retrieve Universal Time (UTC) by UDP

Preparation

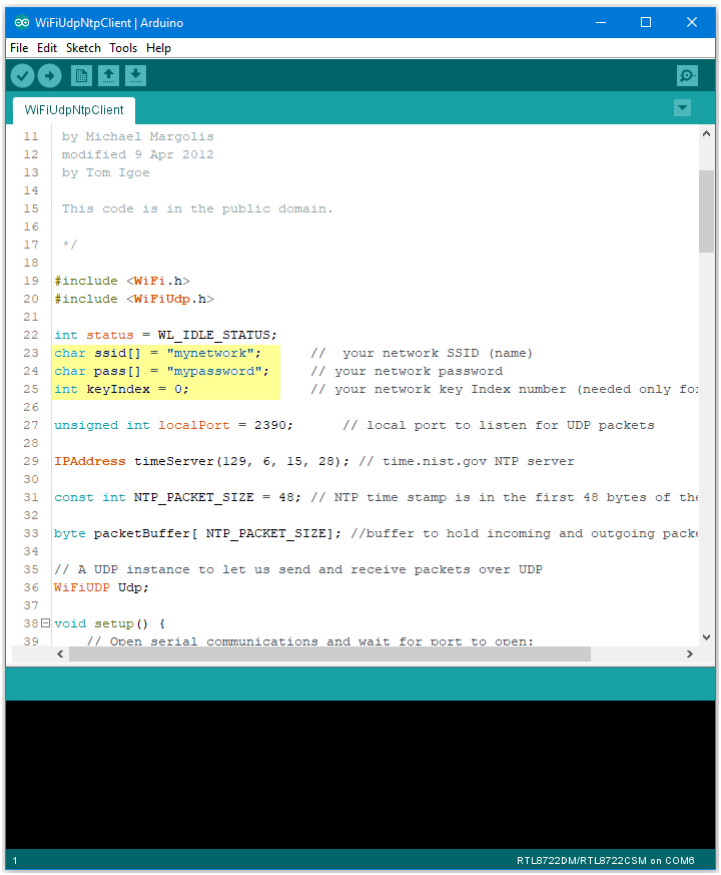
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we connect Ameba to WiFi. Then send NTP (Network Time Protocol, RFC 1305) request to NTP server using UDP. After receiving the NTP request, the NTP server replies current UTC (Coordinated Universal Time) packet. We will parse the UTC packet to show current UTC time in the serial monitor.
Open the example: `"File" -> "Examples" -> "AmebaWiFi" -> "WiFiUdpNtpClient"`

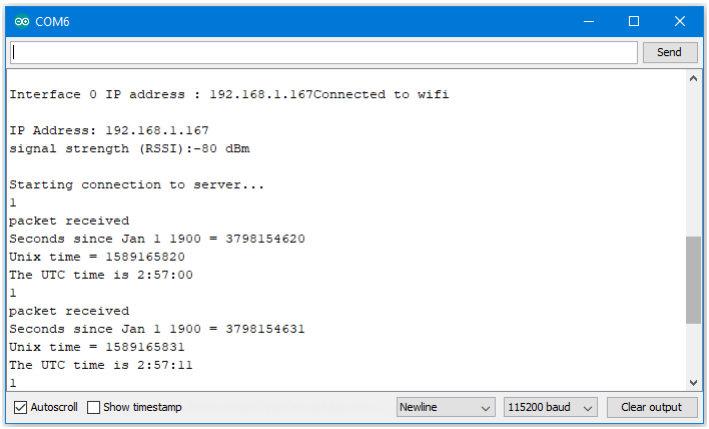


Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and sends NTP request packet to NTP server “129.6.15.28”.

We parse the replied packet and show UTC time in serial monitor:



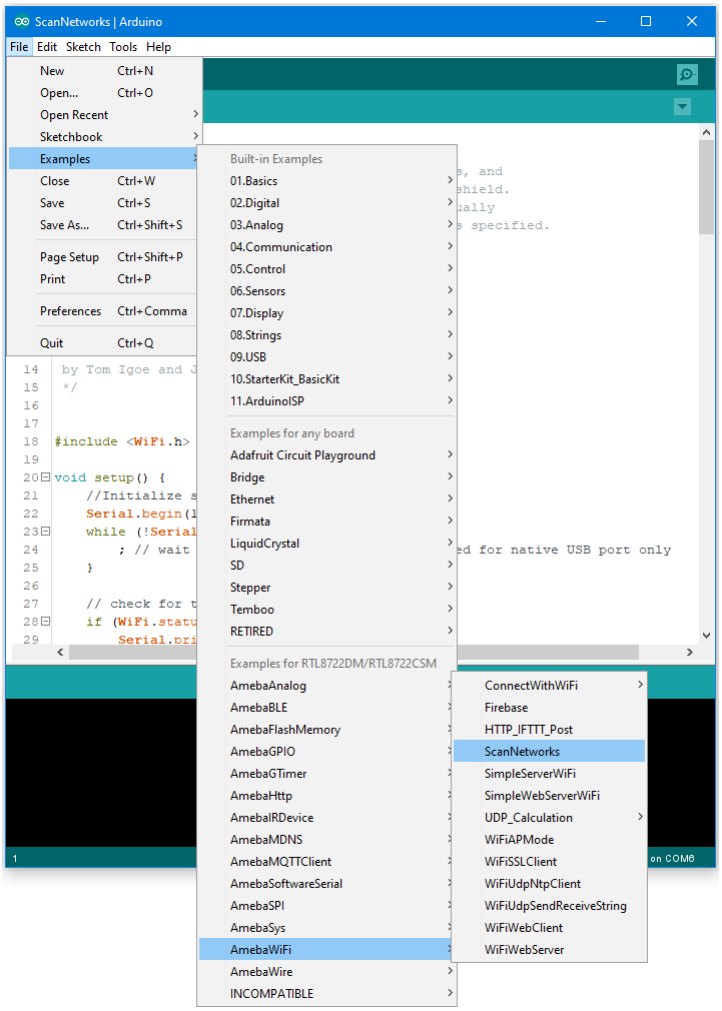
WiFi - Scan the surrounding WiFi networks

Materials

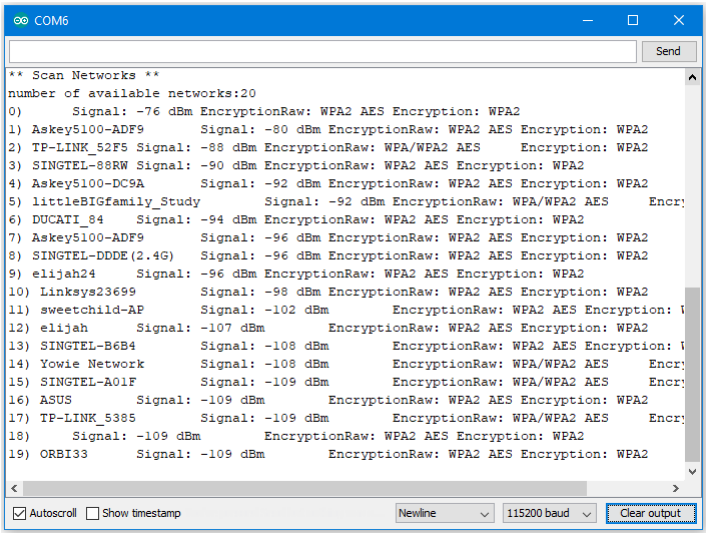
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Antenna x 1
- **AmebaD [:raw-html:`<p style="color:#1A76B4;">AMB21(RTL8722DM/CSM)</p>` / AMB23(RTL8722DM_MINI) / BW16(RTL8729DN)] x 1**

Example

In this example, we use Ameba to scan available WiFi hotspots in the surroundings, and prints the SSID, encryption type, signal strength information of each detected hotspot.
First, make sure the correct Ameba development board is selected in Arduino IDE: "`Tools`" -> "`Board`"
Open the "ScanNetworks" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`ScanNetworks`":



Then upload the sample code and press the reset button on Ameba. Afterwards, you can see "**Scan Networks**" message appears, with the detected WiFi hotspots and the information of each hotspot.



Code Reference

- First we use `WiFi.macAddress(mac)` to get the MAC address of Ameba: https://www.arduino.cc/en/Reference/WiFiMACAddress
- Then we use `WiFi.scanNetworks()` to detect WiFi hotspots: https://www.arduino.cc/en/Reference/WiFiScanNetworks
- To get information of detected WiFi hotspot: We use `WiFi.SSID(thisNet)` to retrieve SSID of a network: https://www.arduino.cc/en/Reference/WiFiSSID We use `WiFi.RSSI(thisNet)` to get the signal strength of the connection to the router: https://www.arduino.cc/en/Reference/WiFiRSSI
- We use `WiFi.encryptionType(thisNet)` to get the encryption type of the network: https://www.arduino.cc/en/Reference/WiFiEncryptionType

Comparison with Arduino

In the Arduino platform, we need to add an extra WiFi shield to be the WiFi module to realize the WiFi connection. And we must `#include` to use SPI to communicate with WiFi module.
However, Ameba is already equipped with WiFi module. Therefore, `#include` is not needed.

WiFi - Set up Server to communicate

Materials

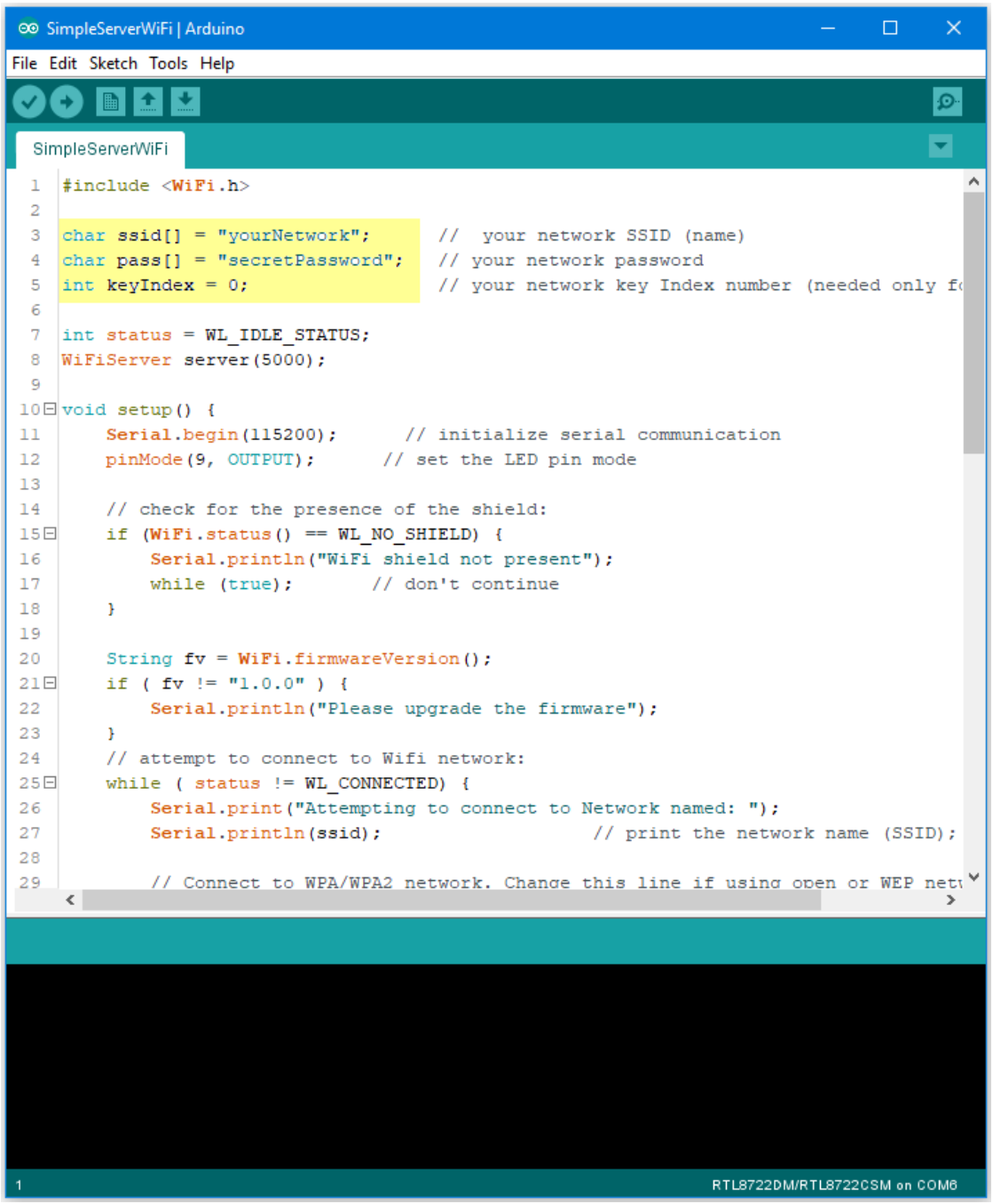
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Laptop Make sure it is connected to the same network domain as Ameba, and tcp tools are installed.

Example

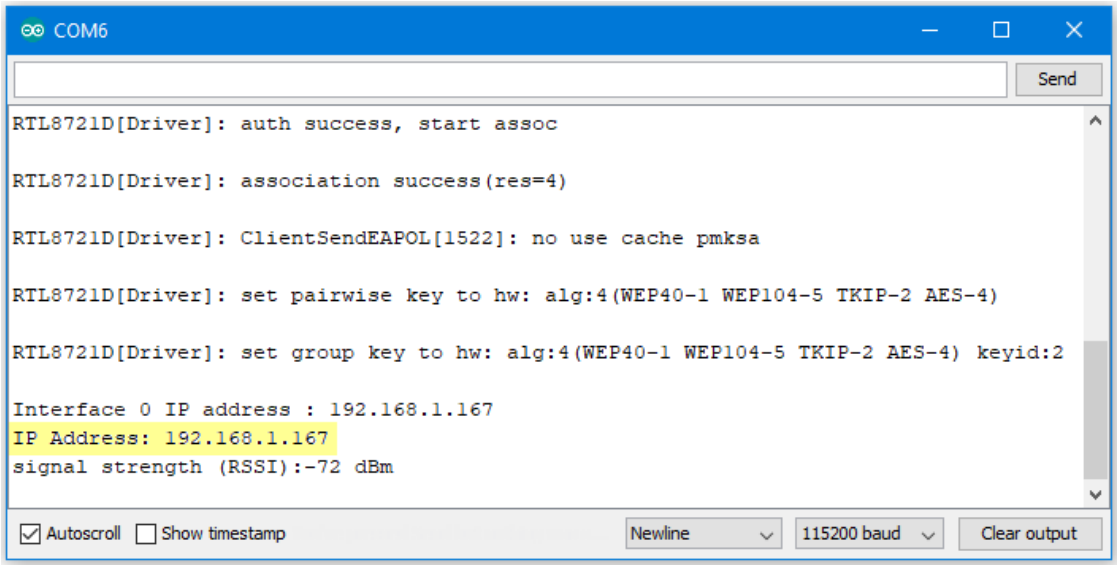
In this example, we first connect Ameba to WiFi, then we use Ameba as server to communicate with client.

First, we make sure the correct Ameba development board is set in “Tools” -> “Board”

Then, open the Simple WiFi Server example in “File” -> “Examples” -> “AmebaWiFi” -> “SimpleServerWiFi”

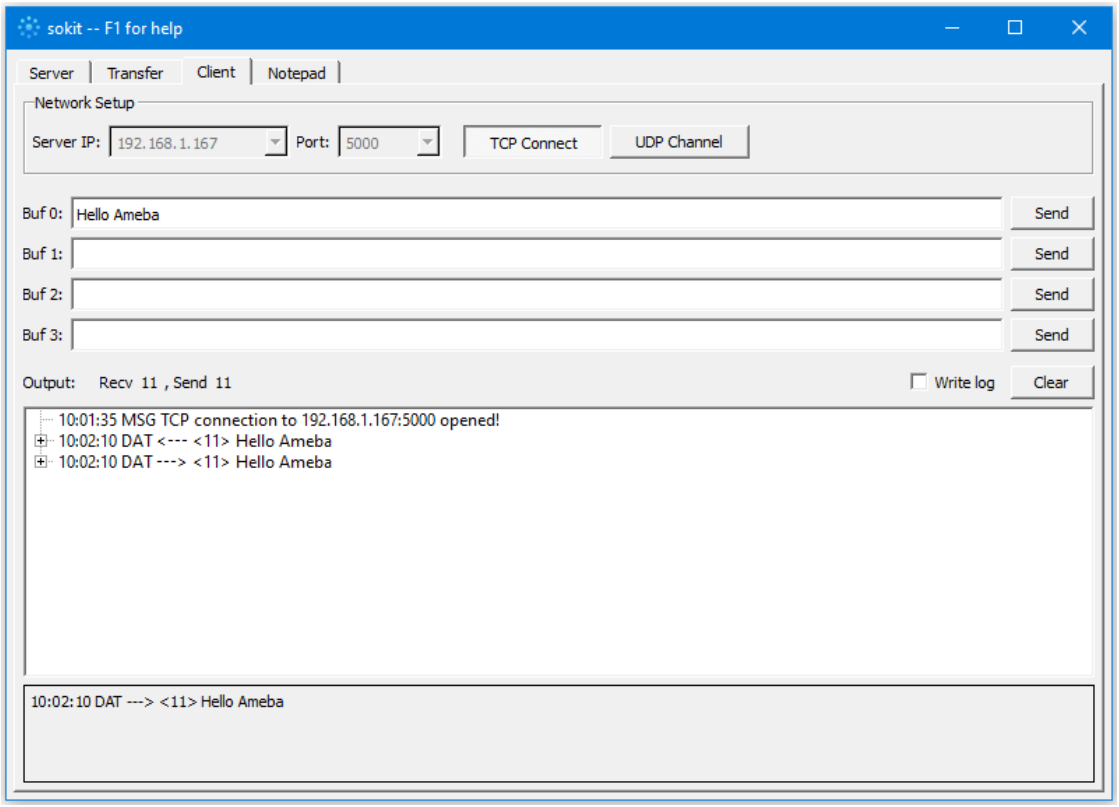


In the sample code, modify the highlighted parameters and enter the ssid and password for your WiFi connection.



Next, upload the code, then press the reset button on Ameba. At this moment, you will see the connection information is displayed in the console.

Next, we use the socket tool in the laptop to be the client and connect to the IP address of the Ameba board shown in the connection information at port 5000. (Note: The socket tool we used in this example is “sokit”)

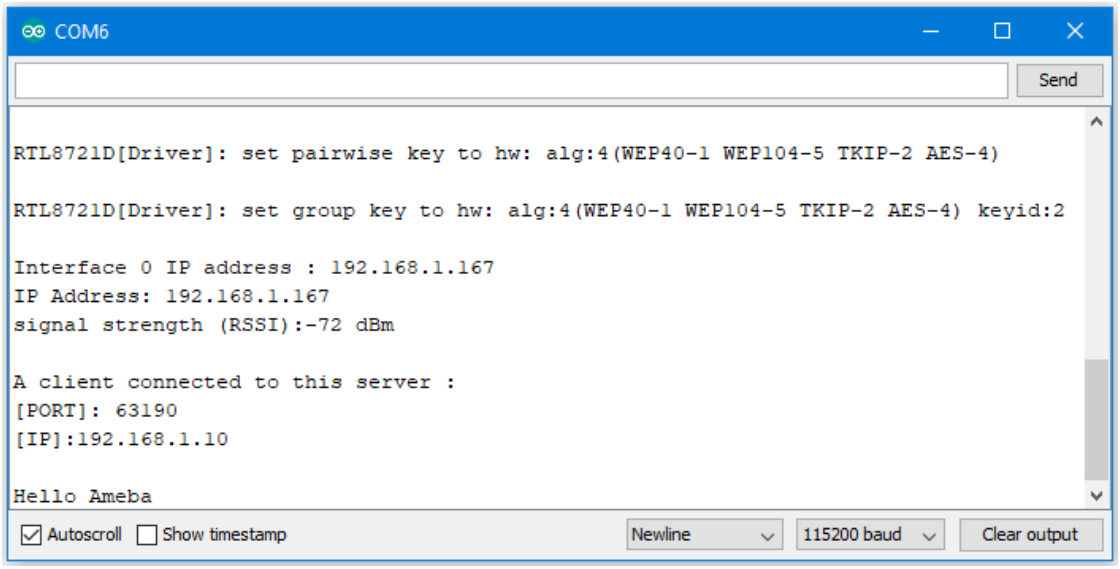


Click on the “Client” tab to choose the client mode, specify the IP and port of the server, then click “TCP Connect”.

If the connection is established successfully, the server shows a message: “A client connected to this Server”, and the IP and port of the connected client.

In this example, when the client and server are connected and the client sends a string to Ameba server, the Ameba server

returns the identical string back to the client.



The string sent to server is returned and showed at the client side.

Code Reference

Use `WiFi.begin()` to establish WiFi connection;
https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection:
Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the Ameba WiFi shield's IP address.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Create server and transmitting data:
Use `Server(port)` to create a server that listens on the specified port.
https://www.arduino.cc/en/Reference/WiFiServer
Use `server.begin()` to tell the server to begin listening for incoming connections.
https://www.arduino.cc/en/Reference/WiFiServerBegin
Use `server.available()` to get a client that is connected to the server and has data available for reading.
https://www.arduino.cc/en/Reference/WiFiServerAvailable
Use `client.read()` to read the next byte received from the server.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.write()` to write data to the server.
https://www.arduino.cc/en/Reference/WiFiClientWrite
Use `client.stop()` to disconnect from the server.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up Client to Retrieve Google Search Information

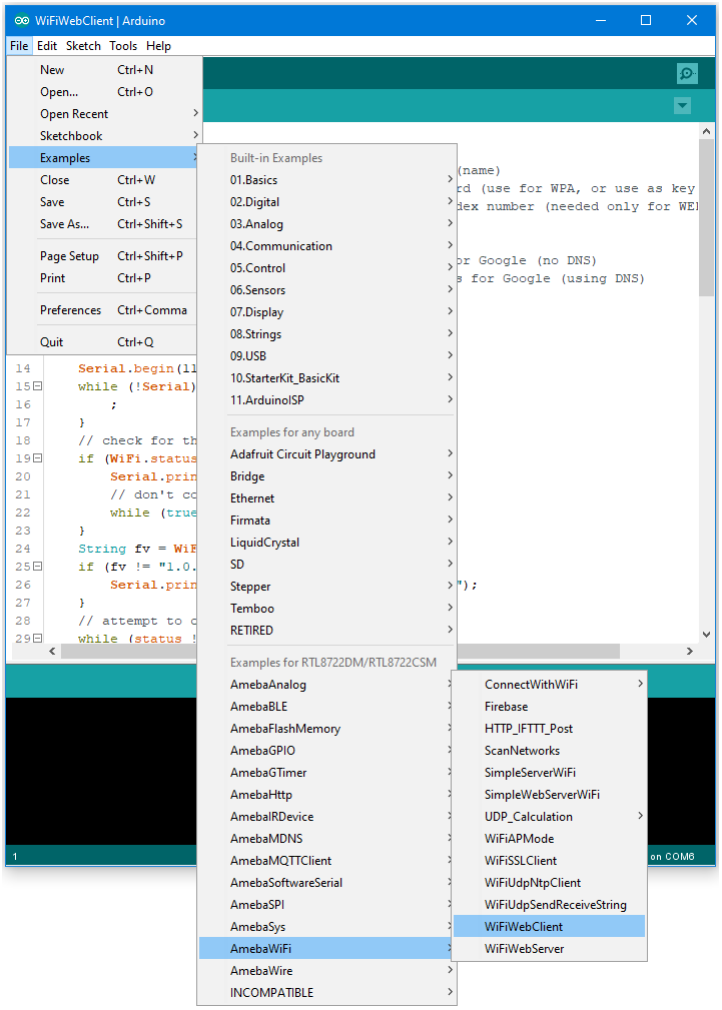
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

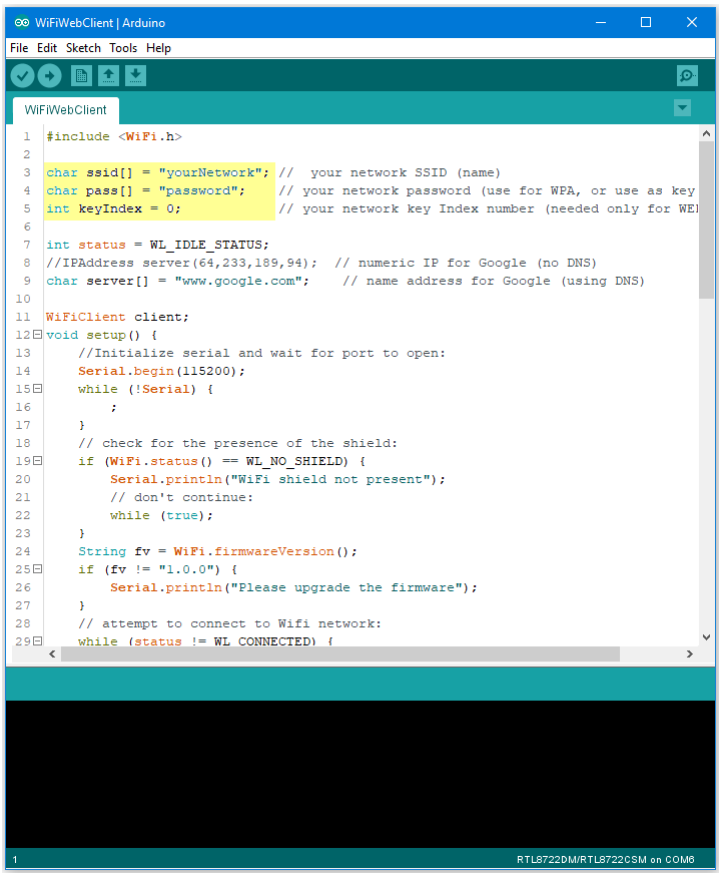
Example

In this example, we use Ameba to be a web client to retrieve information from the Internet.
First, make sure the correct Ameba development board is selected in “Tools” -> “Board”

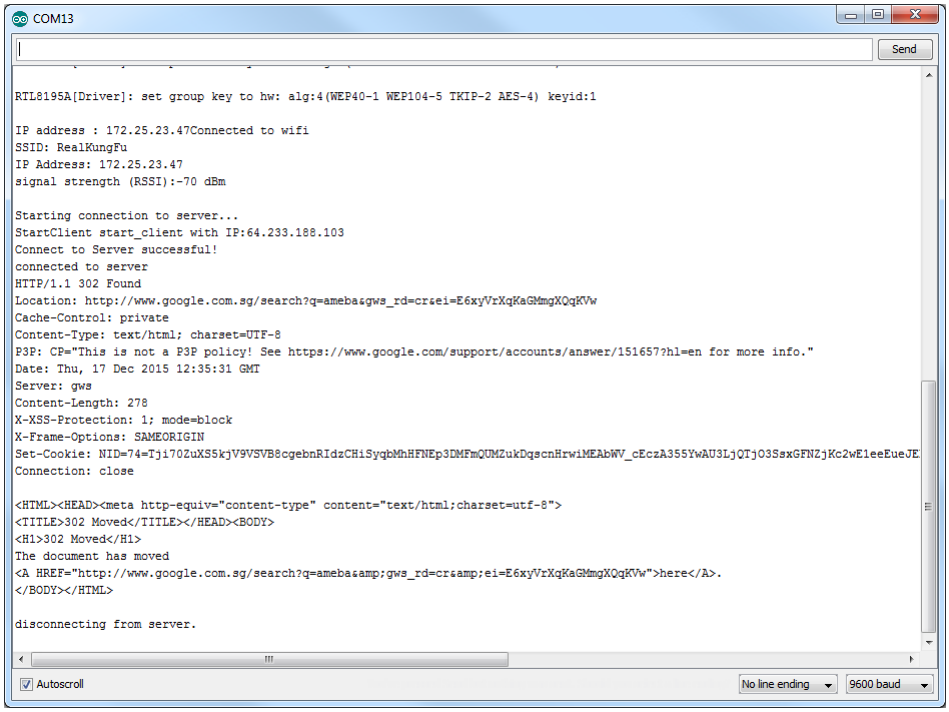
Then open “`File`” -> “`Examples`” -> “`AmebaWiFi`” -> “`WiFiWebClient`”



In the sample code, modify the highlighted snippet and enter the required information (ssid, password, key index) required to connect to your WiFi network.



Upload the code and press the reset button on Ameba. Then you can see the information retrieved from Google is shown in the Arduino serial monitor.



Code Reference

https://www.arduino.cc/en/Reference/WiFiBegin
To get the information of a WiFi connection: Use `WiFi.SSID()` to get SSID of the current connected network.
https://www.arduino.cc/en/Reference/WiFiSSID
Use `WiFi.RSSI()` to get the signal strength of the connection.
https://www.arduino.cc/en/Reference/WiFiRSSI
Use `WiFi.localIP()` to get the IP address of Ameba.
https://www.arduino.cc/en/Reference/WiFiLocalIP
Use `WiFiClient()` to create a client.
https://www.arduino.cc/en/Reference/WiFiClient
Use `client.connect()` to connect to the IP address and port specified.
https://www.arduino.cc/en/Reference/WiFiClientConnect
Use `client.println()` to print data followed by a carriage return and newline.
https://www.arduino.cc/en/Reference/WiFiClientPrintln
Use `client.available()` to return the number of bytes available for reading.
https://www.arduino.cc/en/Reference/WiFiClientAvailable
Use `client.read()` to read the next byte received from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFiClientRead
Use `client.stop()` to disconnect from the server the client is connected to.
https://www.arduino.cc/en/Reference/WiFIClientStop

WiFi - Set up UDP Server to Communicate

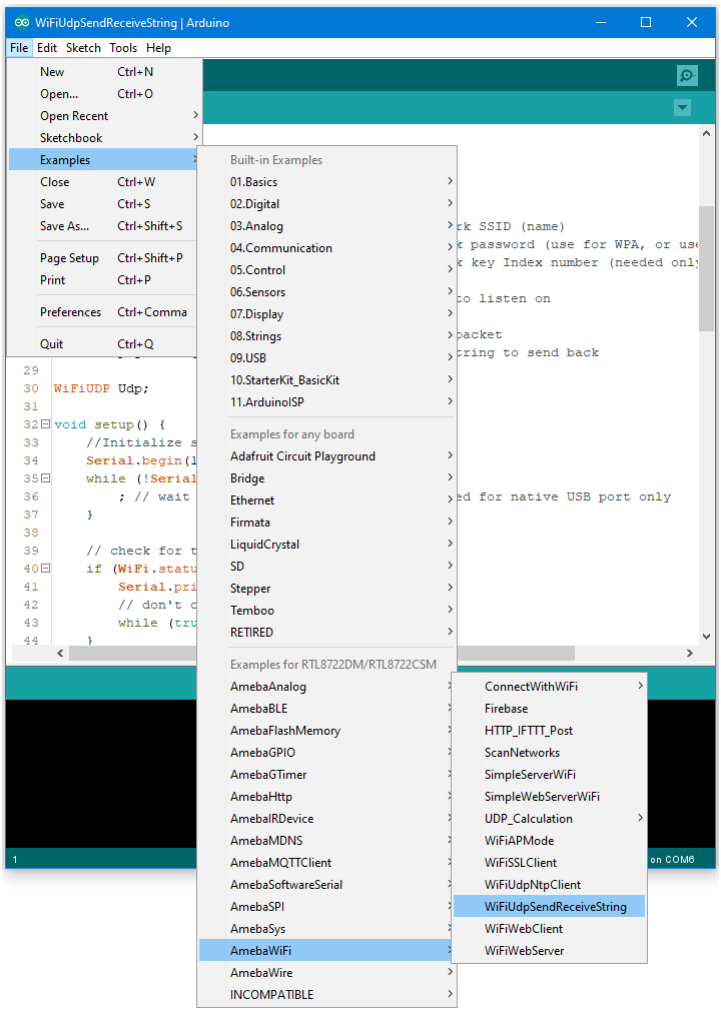
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

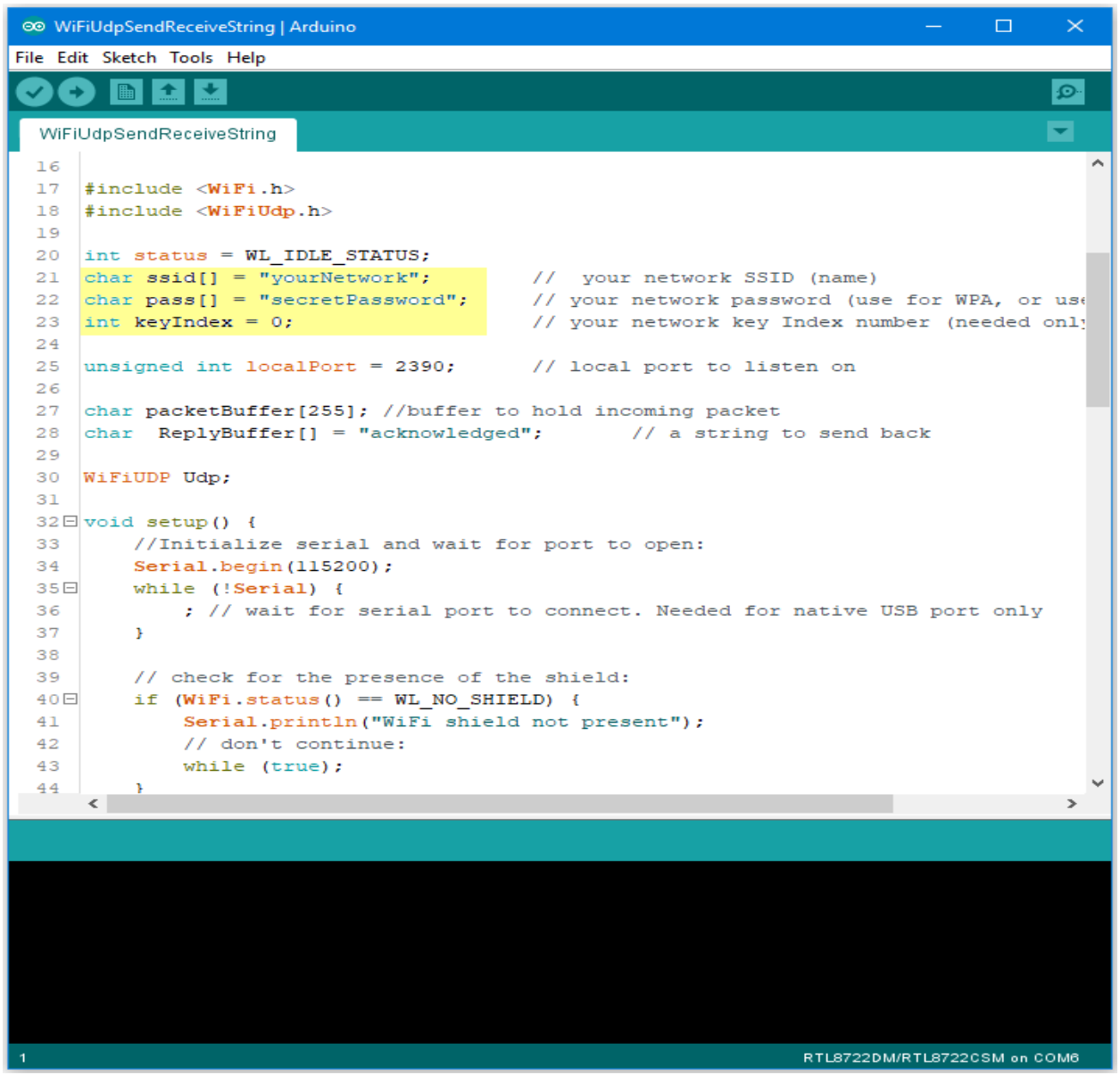
Example

In this example, we connect Ameba to WiFi and use Ameba to be an UDP server. When Ameba receives a message from UDP client, it replies "acknowledged" message to client.

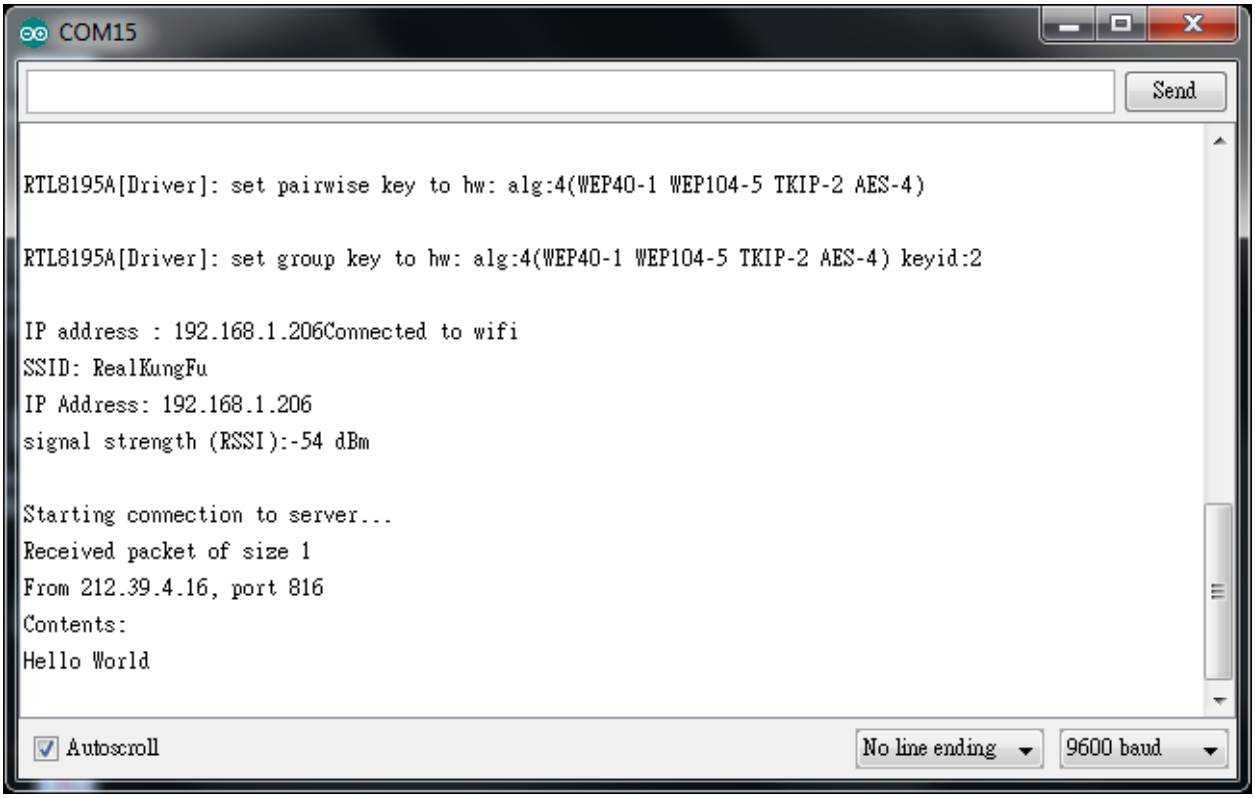
Open the WiFi Web Server example. "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiUdpSendReceiveString`"



Modify the highlighted code section (ssid, password, keyindex) to connect to your WiFi network.



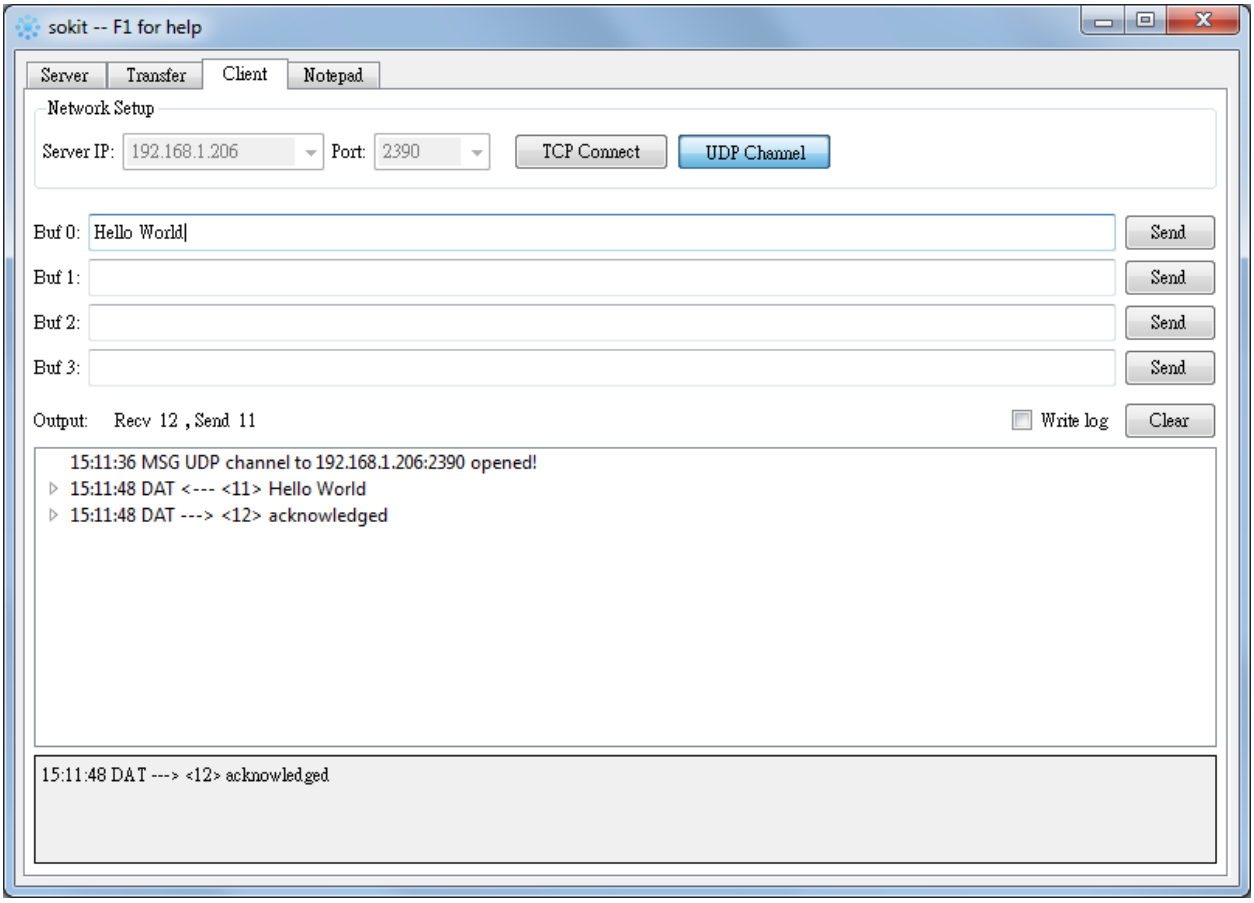
Compile the code and upload it to Ameba. After pressing the Reset button, Ameba connects to WiFi and starts the UDP server with port 2390. After the UDP server starts service, Ameba prints the “Starting connection to server” message and waits for client connection.



As to the UDP client, we use “sokit” program in the computer to connect to UDP server.

Choose client mode and fill in the IP of UDP server (which is the IP of Ameba) and port 2390, then click “UDP Connect”.

After the connection is established, fill in “Hello World” in the Buf 0 field in sokit and click “Send”. Then you can see the Ameba UDP server replies “acknowledged”.



Code Reference

Refer to the Arduino tutorial for detailed information about this example.
https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString

First, use `begin()` to open an UDP port on Ameba.
https://www.arduino.cc/en/Reference/WiFiUDPBegin

Use `parsePacket()` to wait for data from client.
https://www.arduino.cc/en/Reference/WiFiUDPParsePacket

When a connection is established, use `remoteIP()` and `remotePort()` to get the IP and port of the client.
https://www.arduino.cc/en/Reference/WiFiUDPRemoteIP

Then use `read()` to read the data sent by client.
https://www.arduino.cc/en/Reference/WiFiUDPRead

To send reply, use `beginPacket()`, `write()`, `end()`.
https://www.arduino.cc/en/Reference/WiFiUDPBeginPacket
https://www.arduino.cc/en/Reference/WiFiUDPWrite
https://www.arduino.cc/en/Reference/WiFiUDPEndPacket

WiFi - Set up WiFi AP Mode

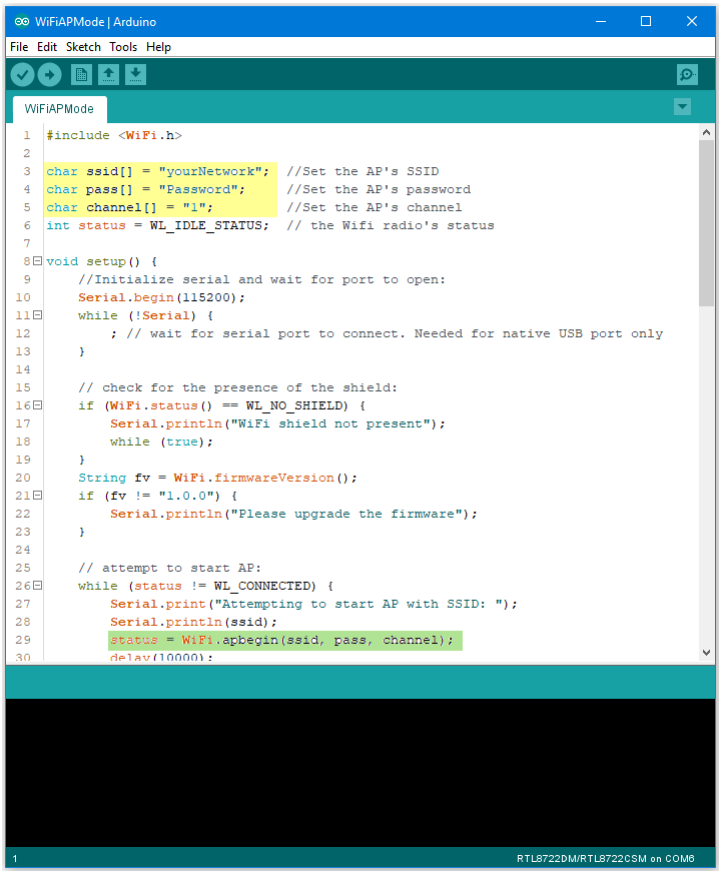
In AP mode, Ameba can accept at most 3 station connections, and can be set to open mode or WPA2 mode.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we turn on the AP mode of Ameba and connect station to Ameba.
Open the WiFi AP example, "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiAPMode`"



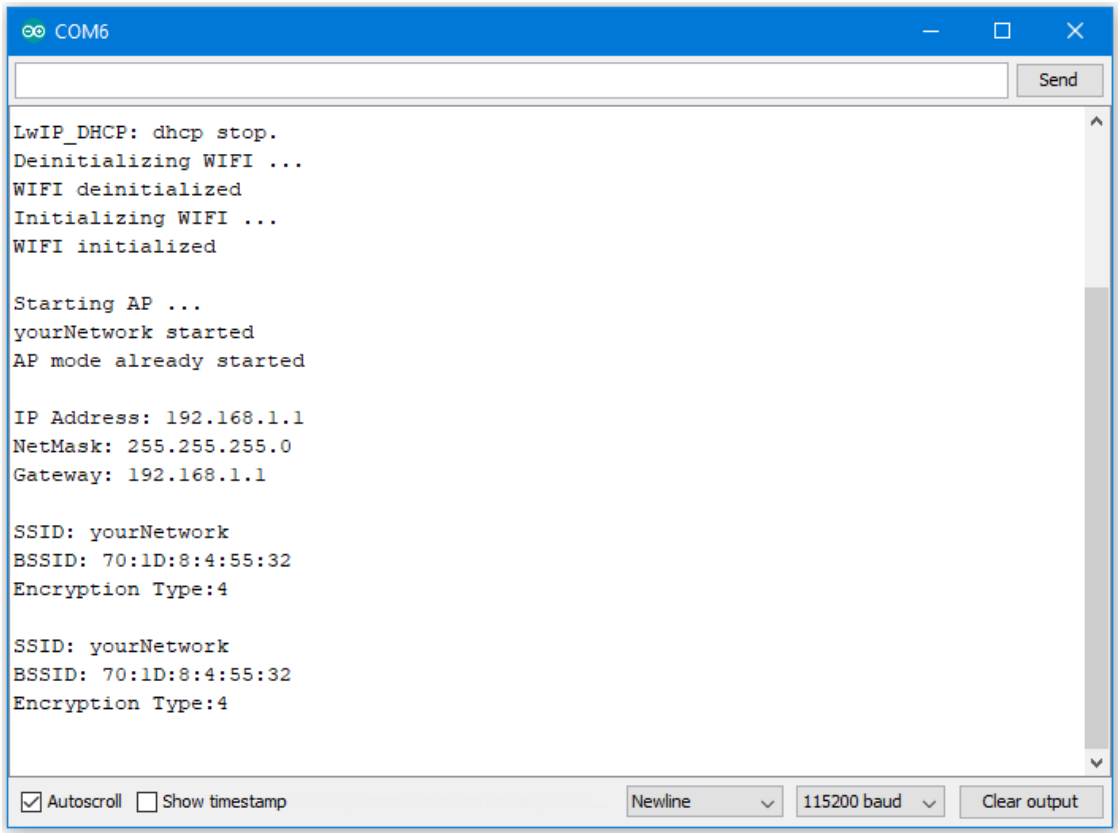
In the highlighted code snippet, fill in your SSID, PASSWORD and CHANNEL.

The code highlighted in green is the API we used to turn on the AP mode in security mode.

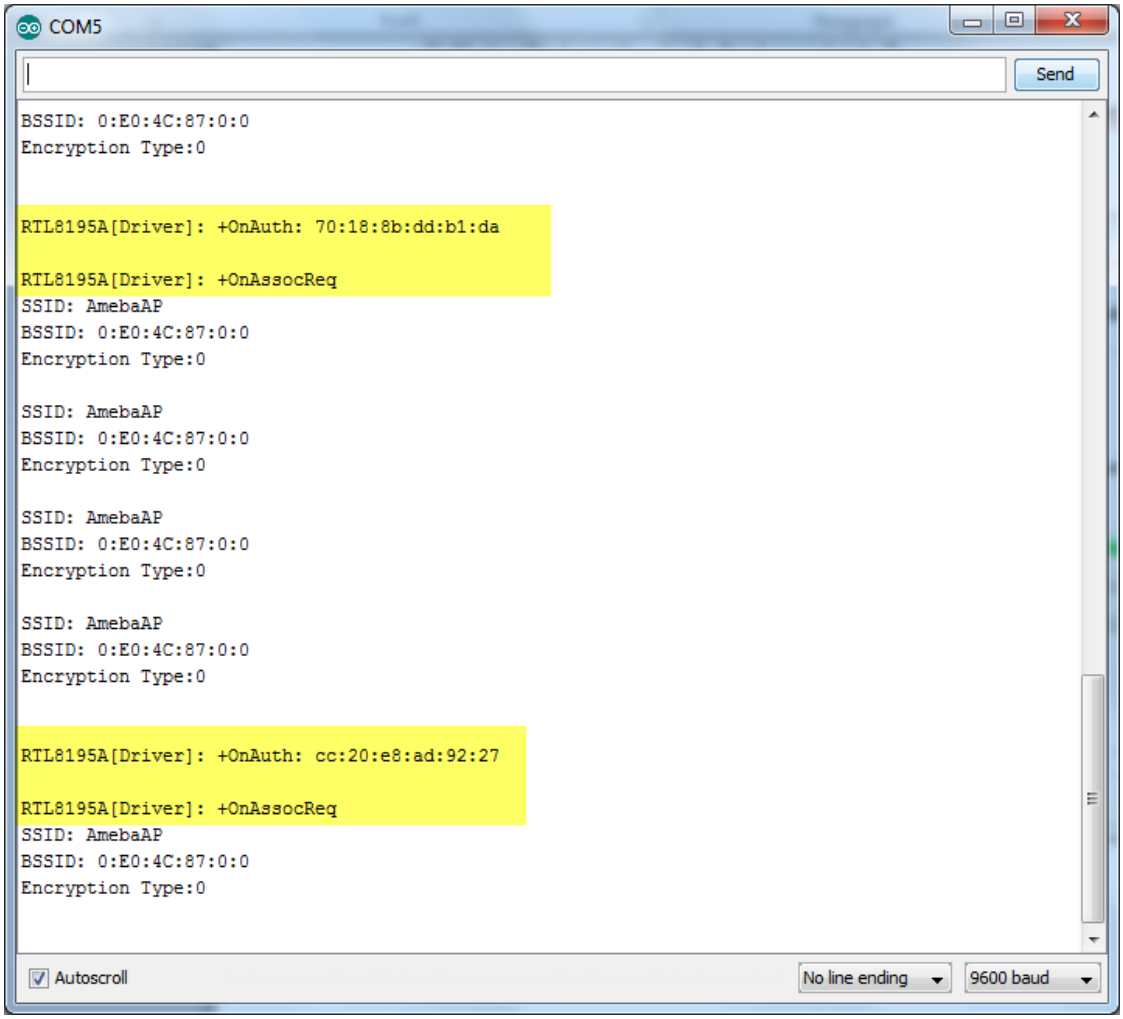
If you want to turn on the AP mode in open mode, please modify the code to

```
status = WiFi.apbegin(ssid, channel);
```

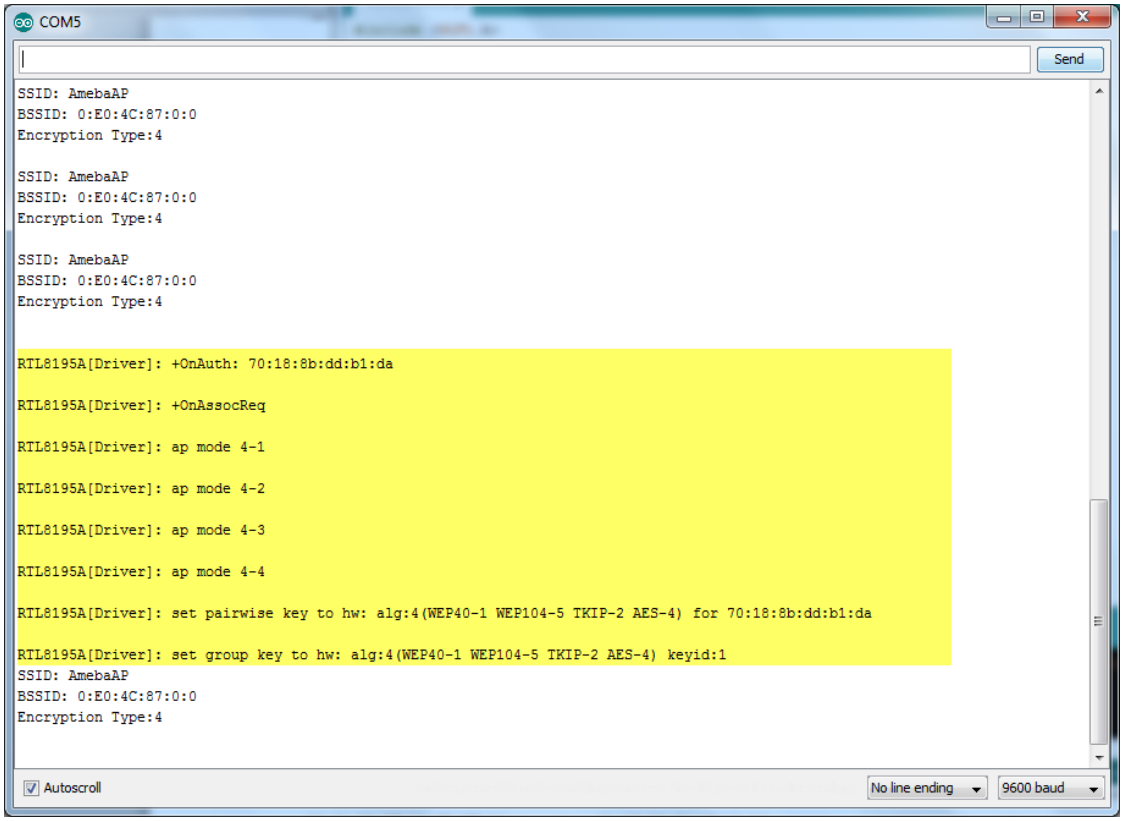
Then upload the sample code and press reset, and you can see related information shown in serial monitor.



In the figure below, we show the messages shown in serial monitor when two stations connect to Ameba AP in open mode:



In the figure below, we show the messages shown in serial monitor when a station connects to Ameba AP in security mode:



WiFi - Set up SSL Client for HTTPS Communication

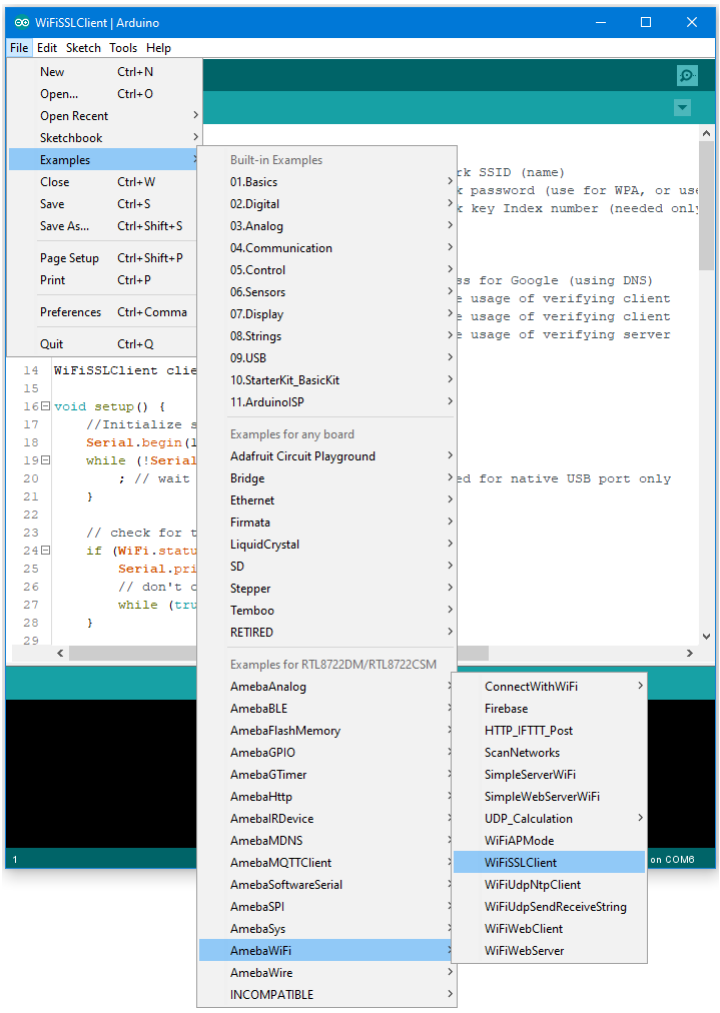
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

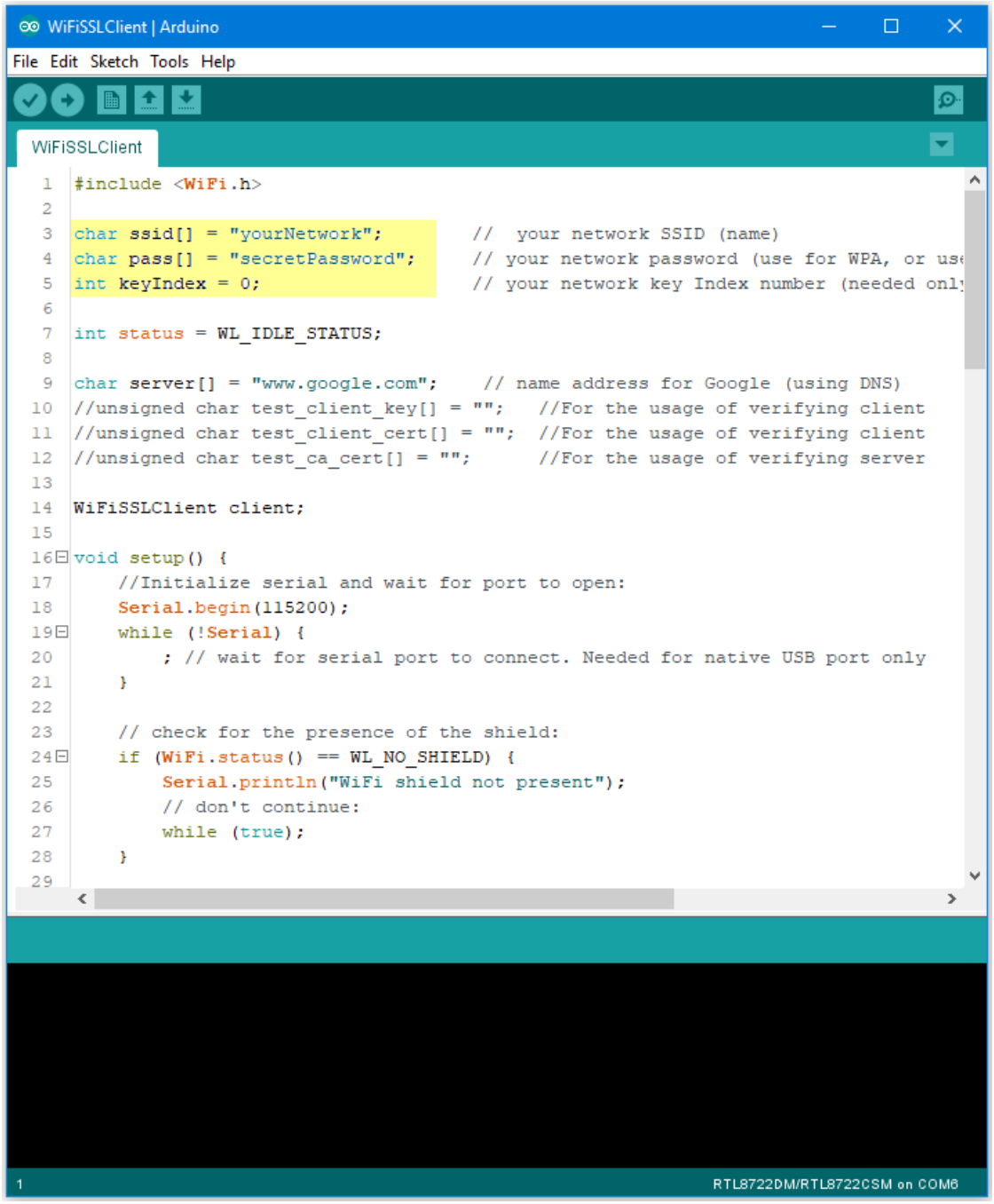
Example

This example uses Ameba to securely retrieve information from the internet using SSL. SSL is an acronym for Secure Sockets Layer. It is a cryptographic protocol designed to provide communications security over a computer network, by encrypting the messages passed between server and client.

Open the "WiFiSSLClient" example in "`File`" -> "`Examples`" -> "`AmebaWiFi`" -> "`WiFiSSLClient`".

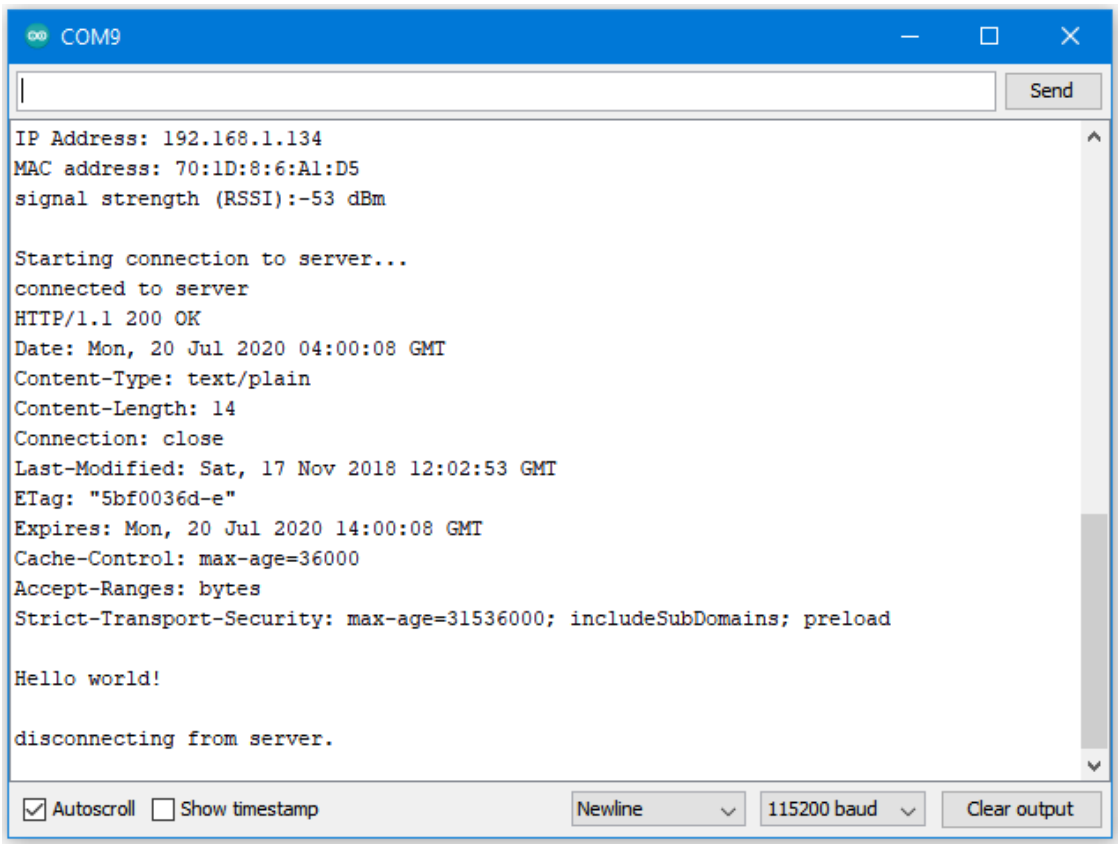


In the sample code, modify the highlighted snippet to reflect your WiFi network settings.



Upload the code and press the reset button on Ameba once the upload is finished.

Open the serial monitor in the Arduino IDE and observe as Ameba retrieves a text file from os.mbed.com.

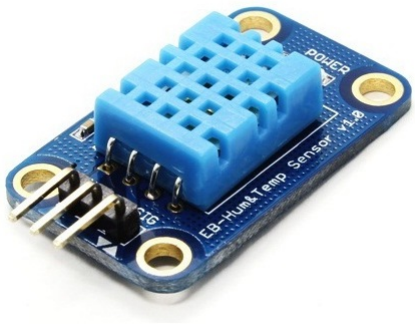
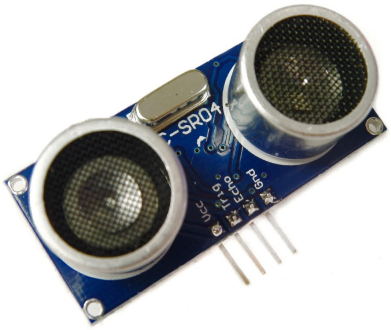
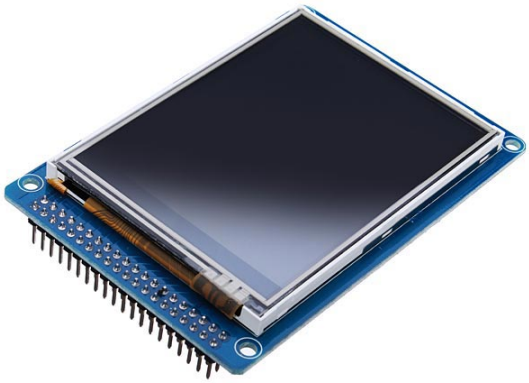
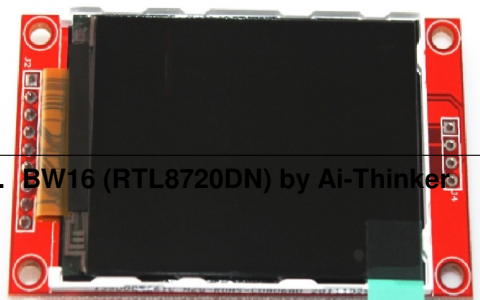
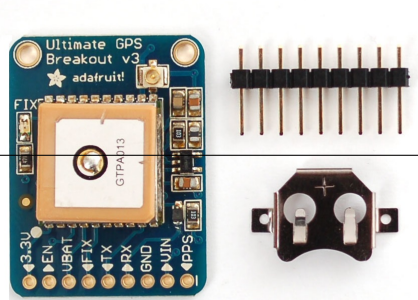


Code Reference

Use “WiFiSSLClient client;” to create a client that uses SSL. After creation, the client can be used in the same way as a regular client.

Components Used

Table 8: Components used in Examples

	
DHT11_DHT22 Humidity & temperature sensor	HC_SR04 Distance measurement function
	
ILI9341 TFT LCD TFT LCD display with SPI interface	PMS3003/5003 Air quality sensor that detects concentration of micro particulate matters
	

Peripheral Examples

E-Paper - Display Images

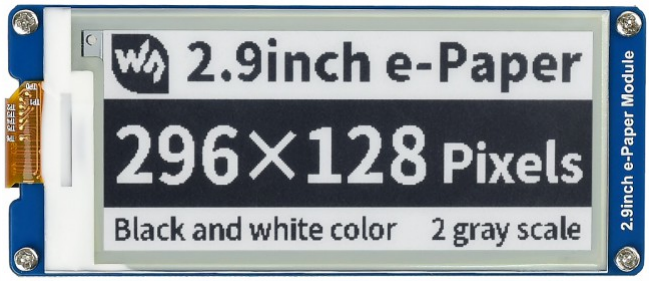
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

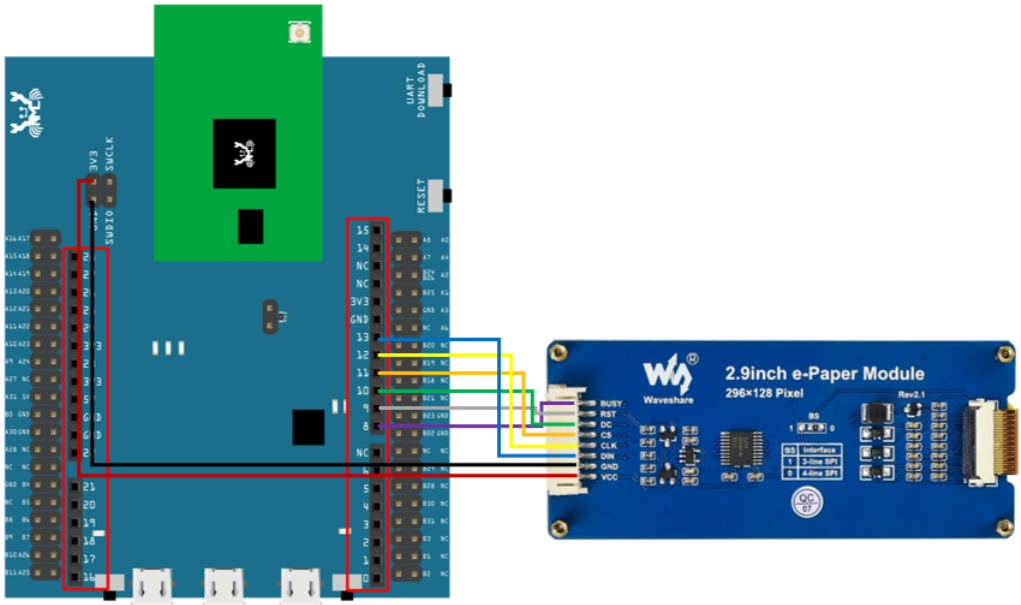
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

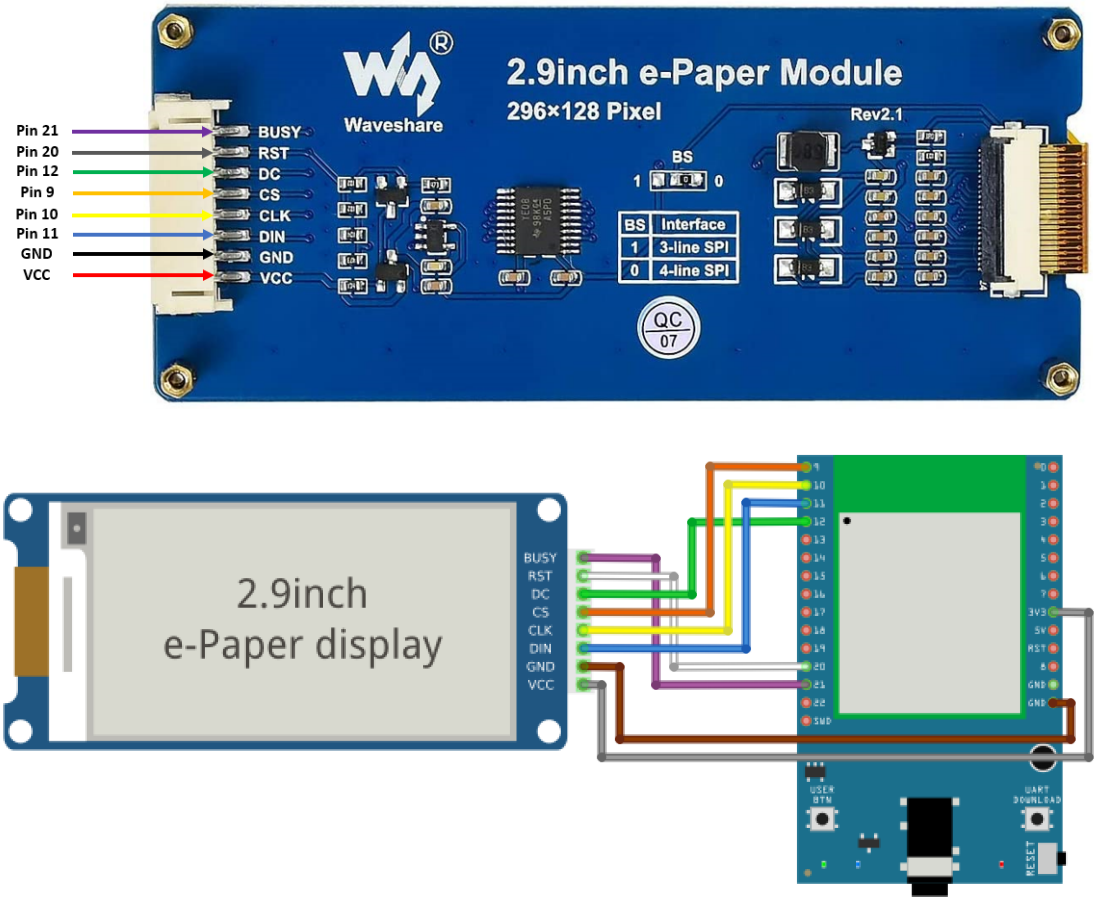
Front view of the e-Paper Module:



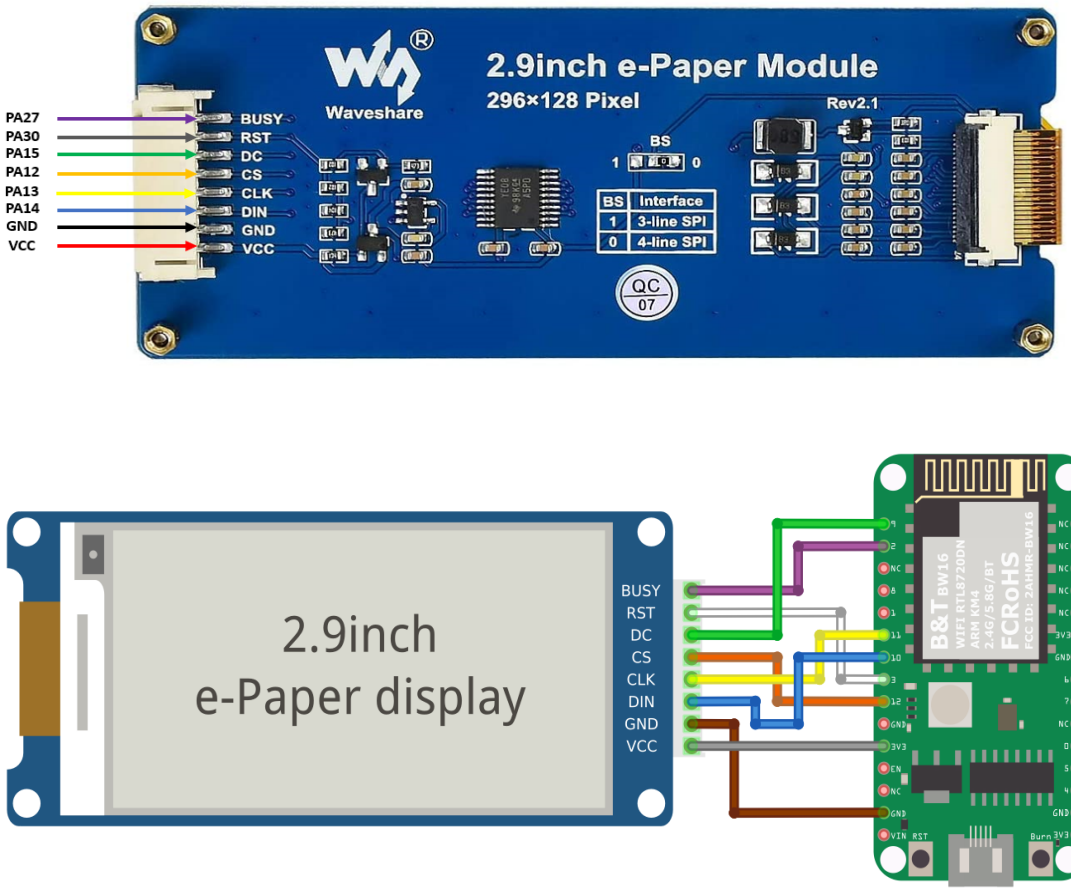
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:

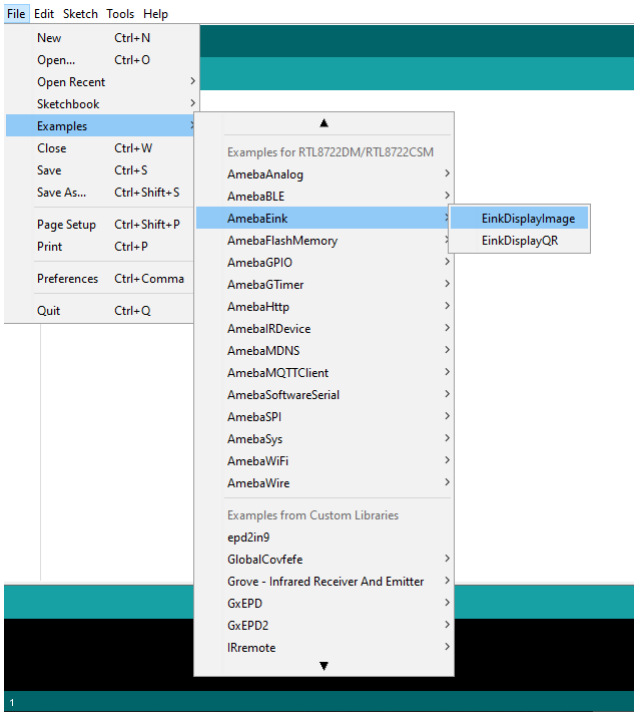


Firstly, you need to prepare a picture/photo in the format of 296×128 pixels. We can easily find a photo resizing tool online, for example, the Online Image Resizer.

Following the instructions on the website, then download the generated image in JPG format.

Secondly, we use the Image2LCD tool to transfer the downloaded 296×128 image into hexadecimal codes. You can visit this YouTube link to get detailed instructions.

Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`EinkDisplayImage`":



Press the reset button after uploading the sample code, you will need to wait for around 1-2 seconds for the e-Paper module to fresh its screen. Then the screen will start to display an image for 5 seconds first, then 3 different QR codes will be displayed every 5 seconds (showing in the screenshot below, you may scan the QR codes and find out more information if you wish to). Lastly, a gif which comes in form of 3 frames will be displayed for a few seconds.

Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html

E-Paper - Display Text

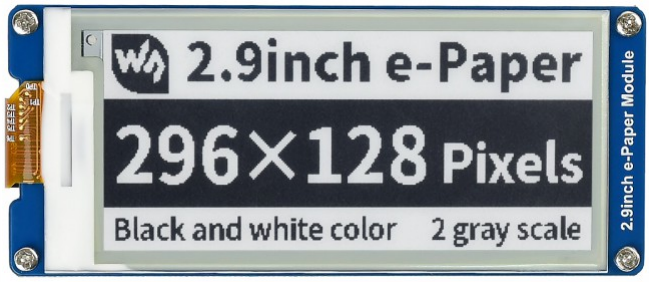
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

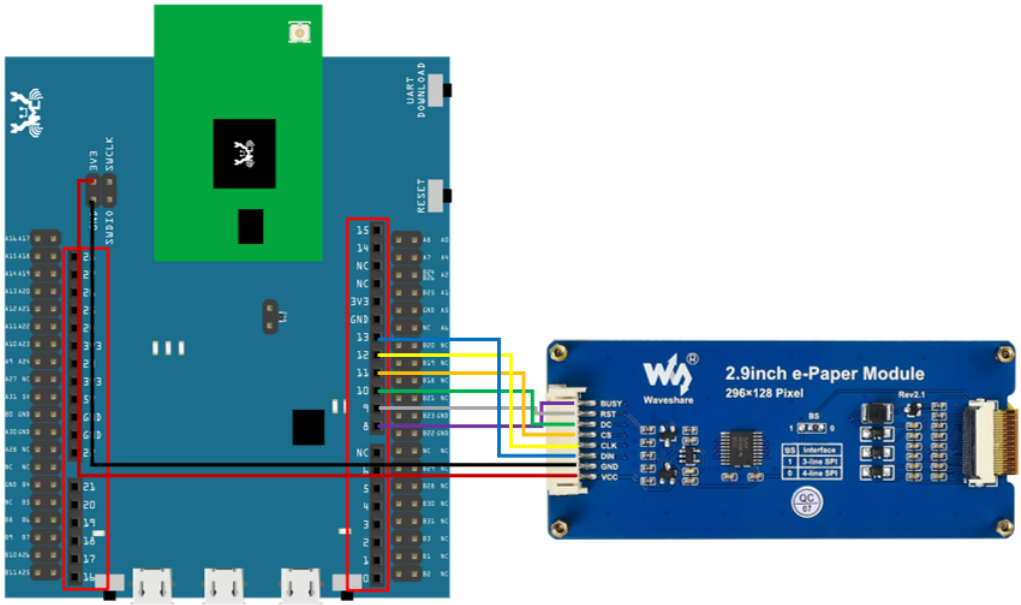
Example

In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.
The 2.9" active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

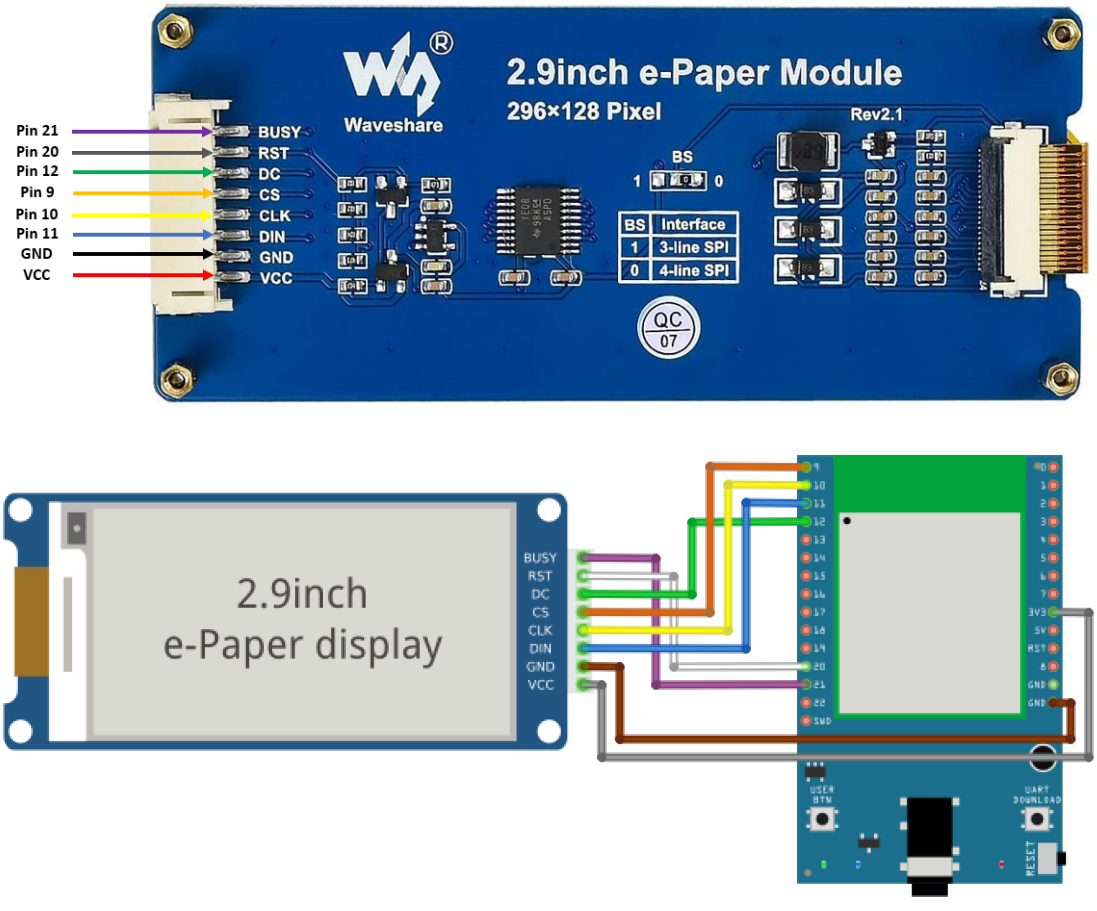
Front view of the e-Paper Module:



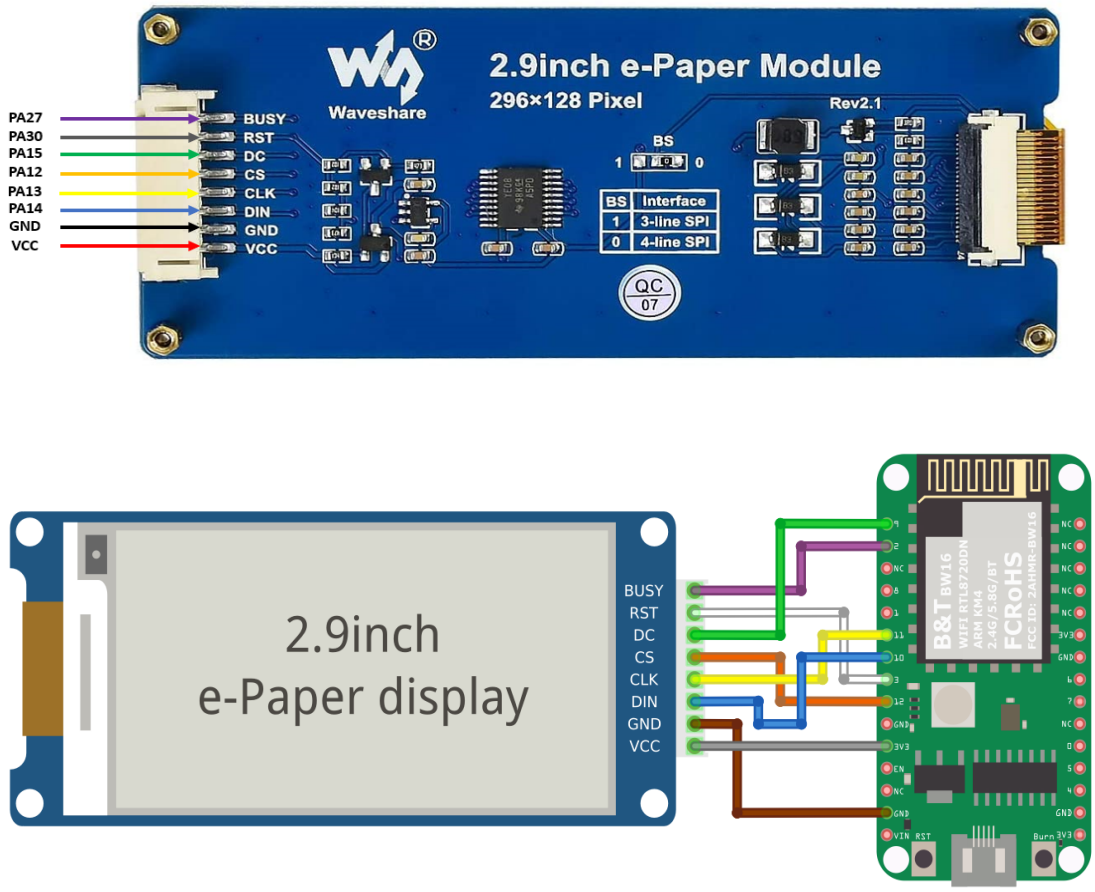
AMB21 / AMB22 Wiring Diagram:



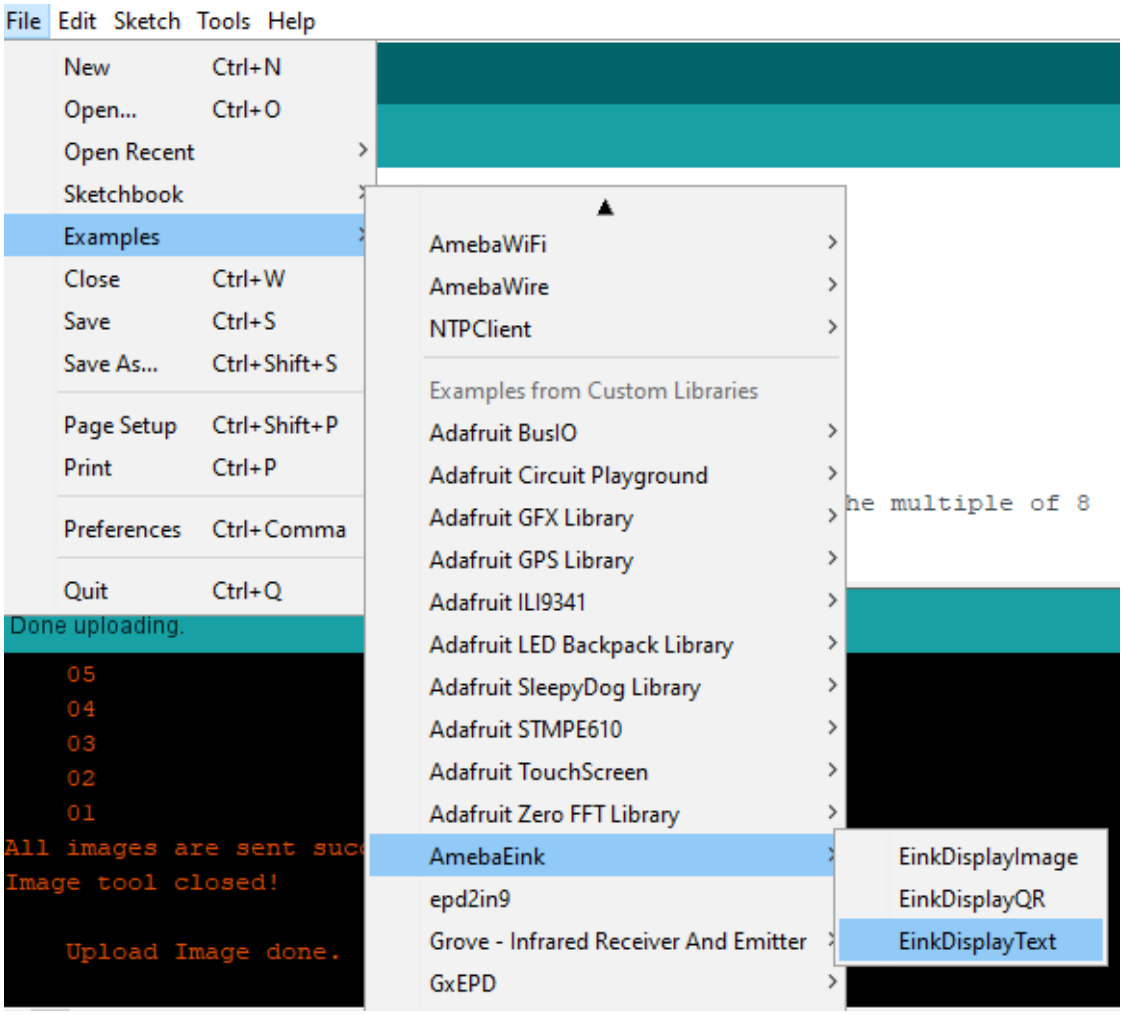
AMB23 Wiring Diagram:



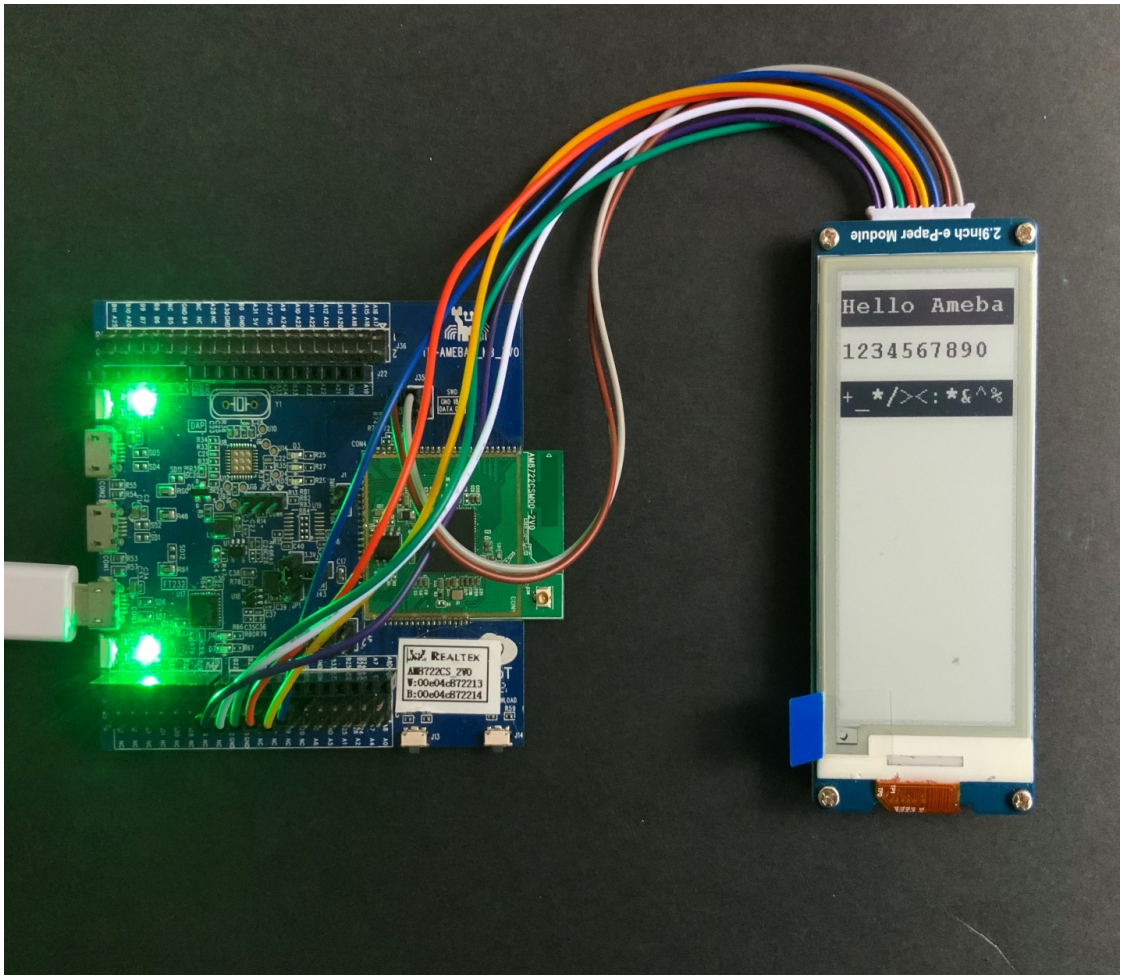
BW16 Wiring Diagram:



Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" -> "`Examples`" -> "`AmebaEink`" -> "`EinkDisplayText`":



Upload the code to the board and press the Reset button after the uploading is done. You will find these texts displayed on the board:



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html

E-Paper - Display User-generated QR Code

Materials

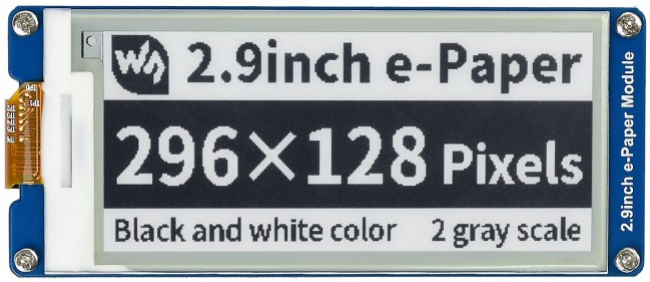
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Waveshare 2.9inch e-Paper HAT (D) x 1

Example

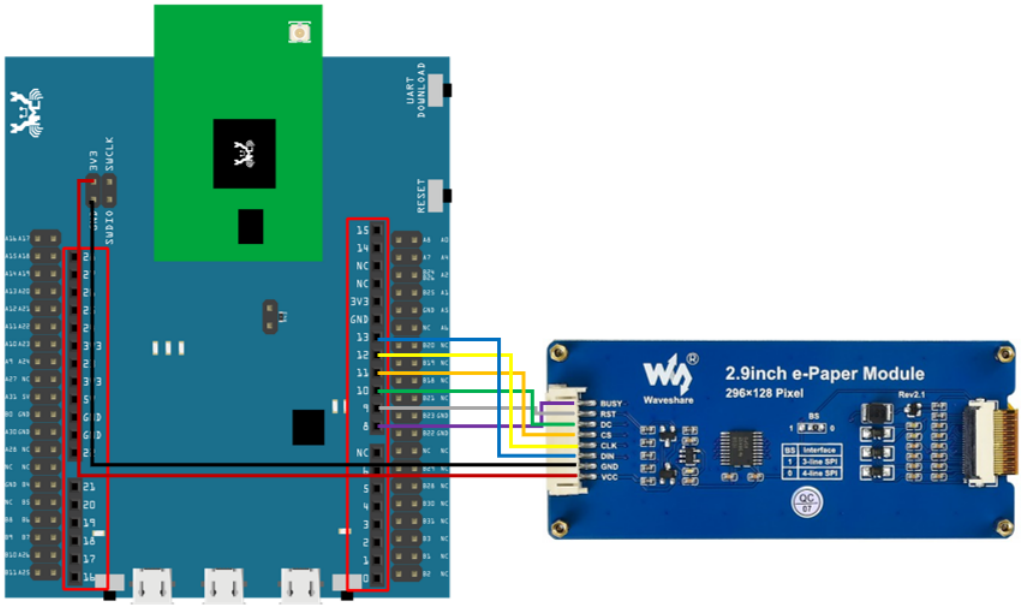
In this example, we use the Ameba RTL8722 module connects to a Waveshare 2.9inch e-Paper module to display a few QR codes. The display uses the flexible substrate as a base plate, with an interface and a reference system design.

The 2.9” active area contains 296×128 pixels and has 1-bit white/black full display capabilities. An integrated circuit contains gate buffer, source buffer, interface, timing control logic, oscillator, etc… are supplied with each panel.
You may refer to the official 2.9inch e-Paper HAT (D) datasheet to know more information about this module.

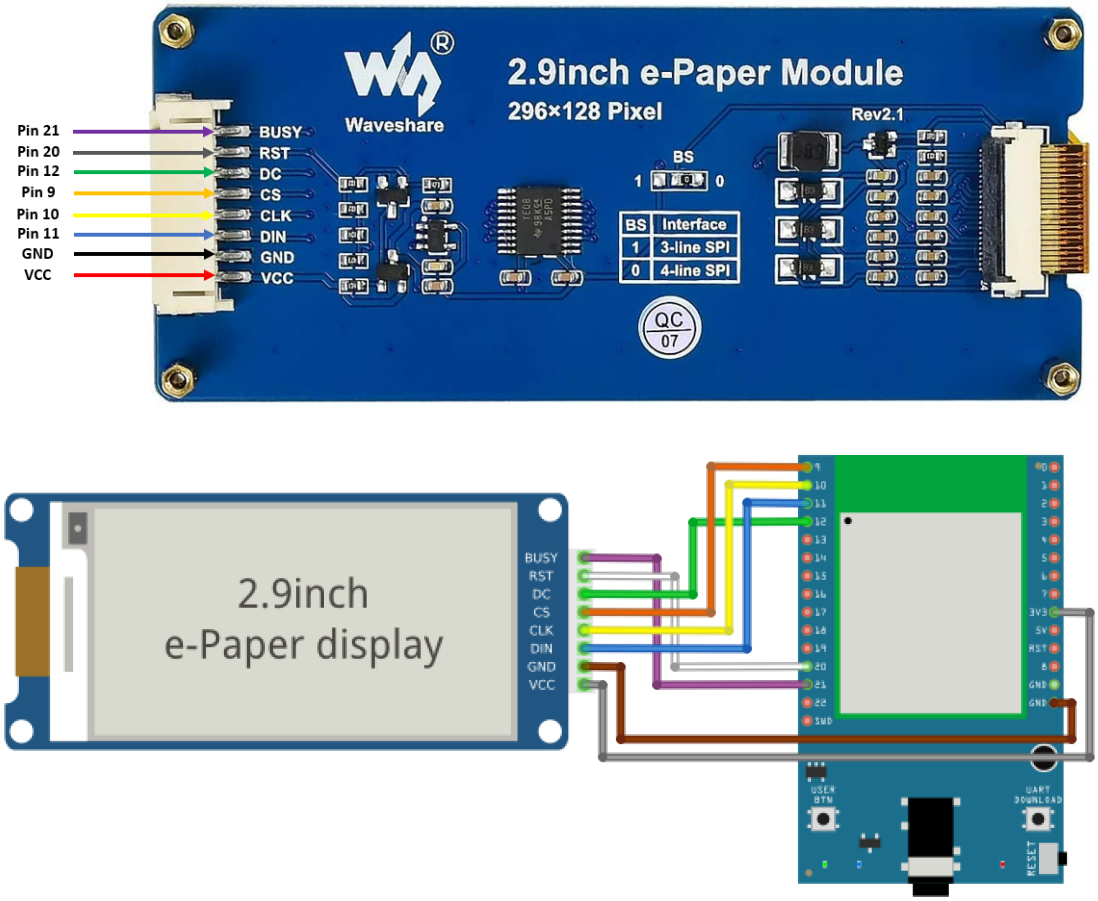
Front view of the e-Paper Module:



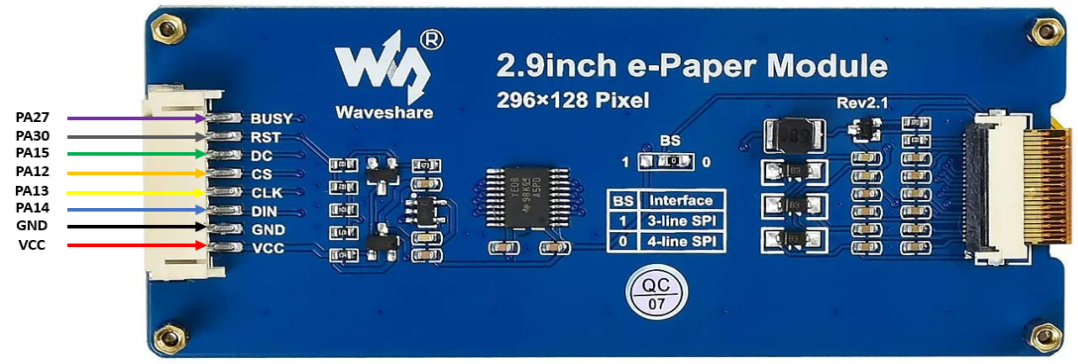
AMB21 / AMB22 Wiring Diagram:

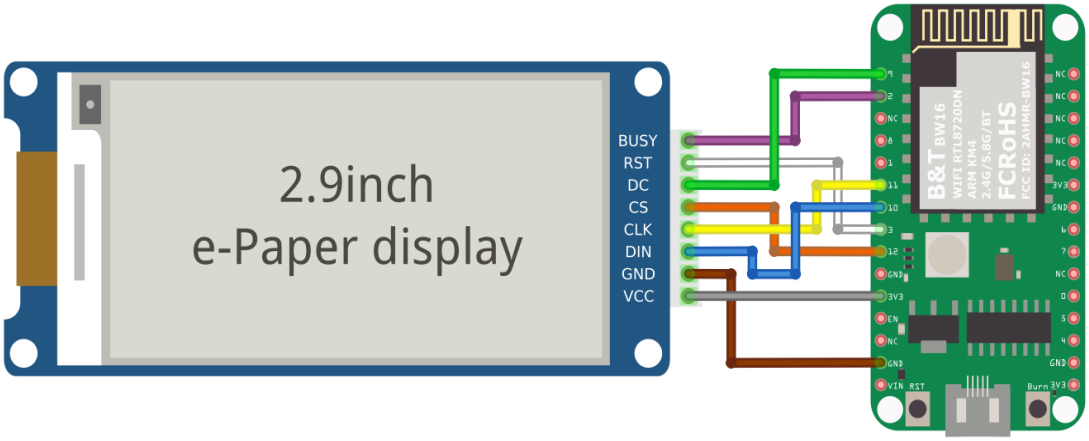


AMB23 Wiring Diagram:

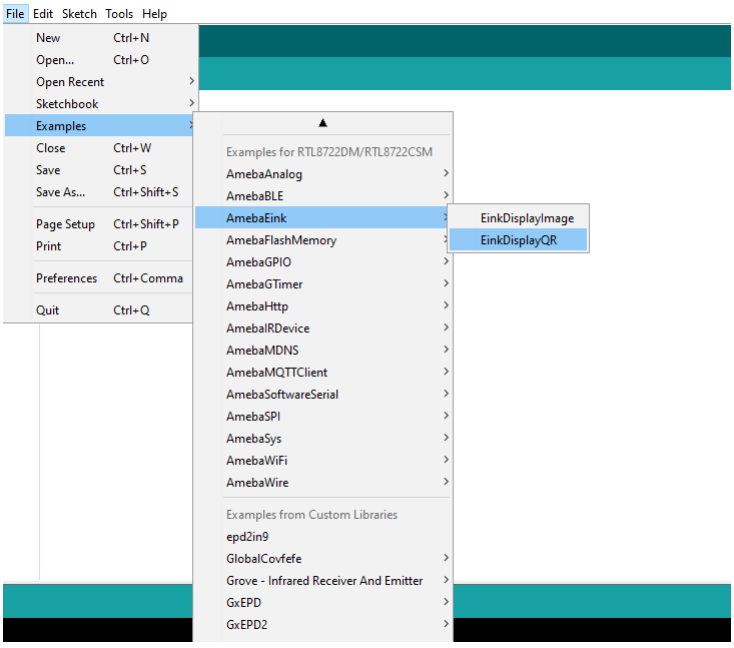


BW16 Wiring Diagram:

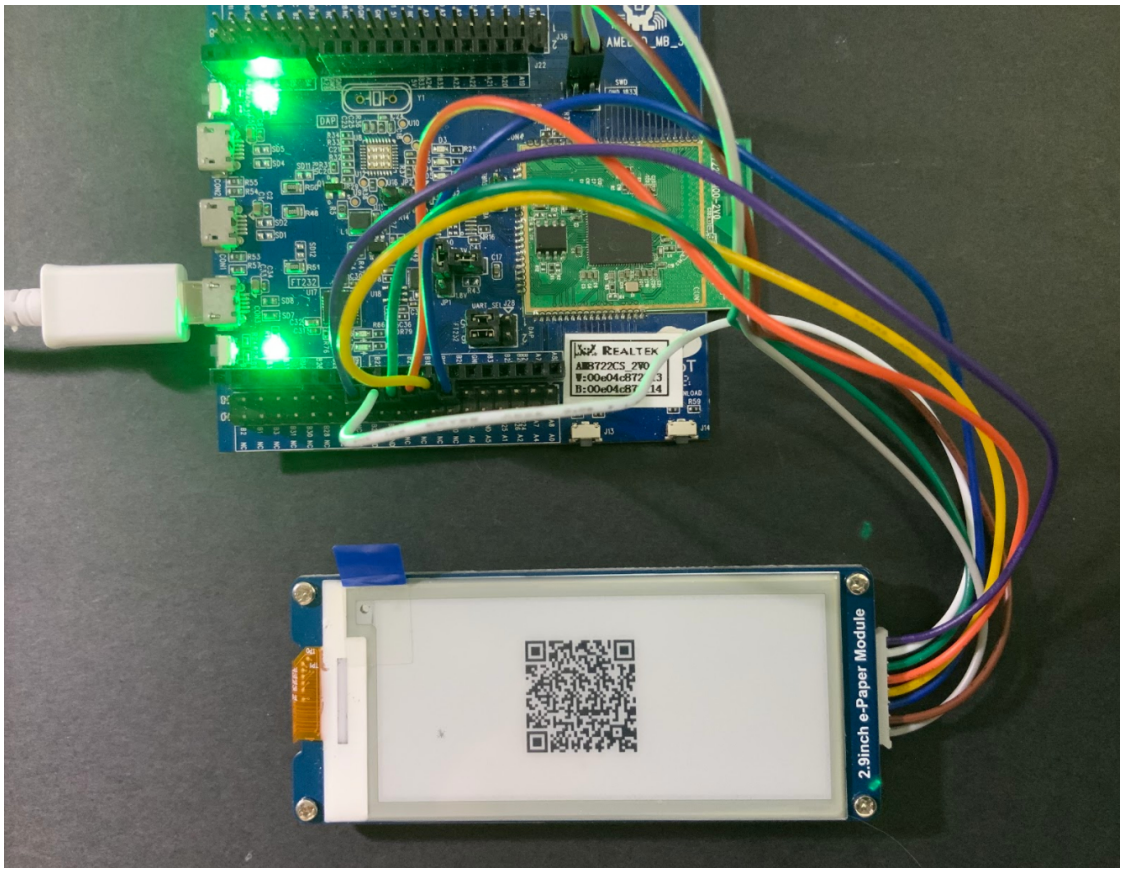




Download the Eink zip library, AmebaEink.zip, at
https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries
Then install the AmebaEink.zip. Open the "DisplayQR" example in "`File`" → "`Examples`" → "`AmebaEink`" → "`DisplayQR`":



Modify the URL in the loop() section as your wish, after that, verify and upload the code to the Ameba board. Upon successfully upload the sample code and press the reset button, a QR code generated based on the URL of your input will be shown on the E-Paper module. The QR code showing below leads to our Ameba IoT official website: Ameba ARDUINO



Code Reference

[1] We use Good Display GDEH029A1 2.9 Inch / 296×128 Resolution / Partial Refresh Arduino Sample Code to get the e-Paper successfully Display: http://www.good-display.com/product/201.html
[2] Provide the link to how to generate a QR code on the E-paper module: https://eugeniopace.org/qrcode/arduino/eink/2019/07/01/qrcode-on-arduino.html
[3] A simple library for generating QR codes in C, optimized for processing and memory-constrained systems: https://github.com/ricmoo/QRCode#data-capacities

Flash Memory - Store data in FlashEEProm

Preparation

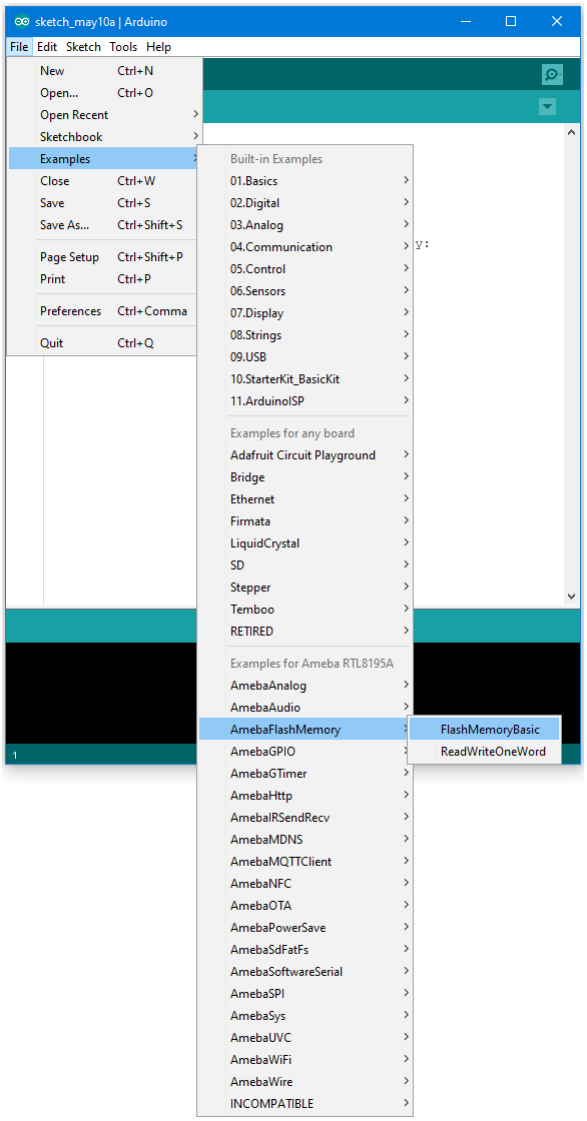
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

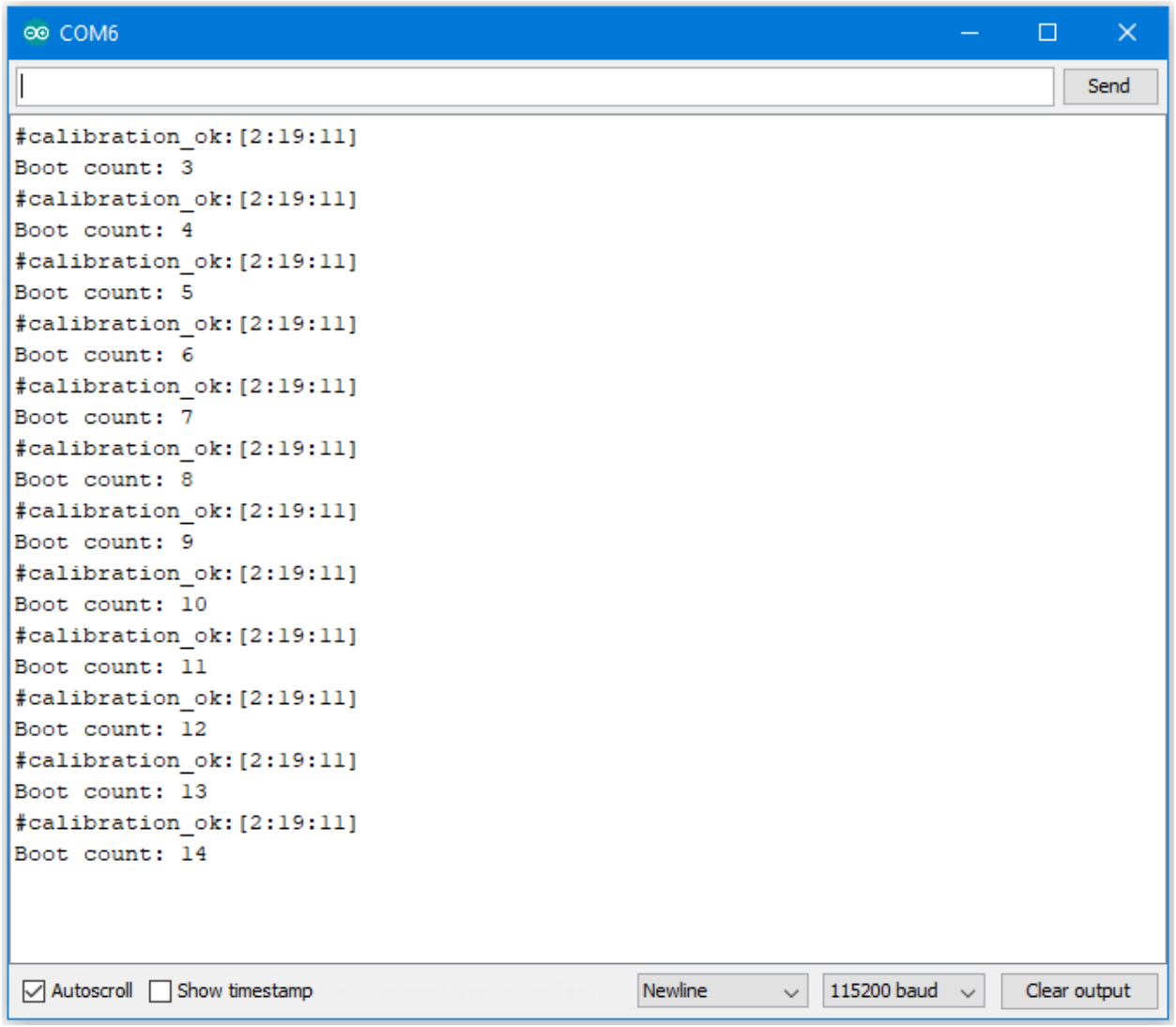
Ameba provides Flash Memory component for data storage and the data can be preserved when the power is off if necessary, e.g., compiled program. To avoid the memory space overlapped with the program on Ameba, the Flash API

uses the tail part of the address space, with sector size 4K.
In this example, we store the value of boot times in flash memory. Every time Ameba reboots, it reads the boot times from flash, increases the value by 1, and writes it back to flash memory.

First open the example, "`File` -> "`Example`" -> "`AmebaFlashMemory`" -> "`FlashMemoryBasic`"



Compile and upload to Ameba, then press the reset button.
Open the Serial Monitor, press the reset button for a few times. Then you can see the boot times value increases.



Code Reference

By default, the Flash Memory API uses address 0xFF000~0xFFFFF to store data.

There is limitation when writing to flash memory. That is, you can not directly write data to the same address you used in last write. To do that correctly, you need erase the sector first. The Flash API of Ameba uses a 4K SRAM to record the user modification and do the erase/write task together.

Use `FlashMemory.read()` to read from Flash memory.
Use `FlashMemory.buf[0] = 0x00;` to manipulate the 4K buf.
Use `FlashMemory.update();` to update the data in buf to Flash Memory.

Flash Memory - Use Flash Memory Larger Than 4K

Preparation

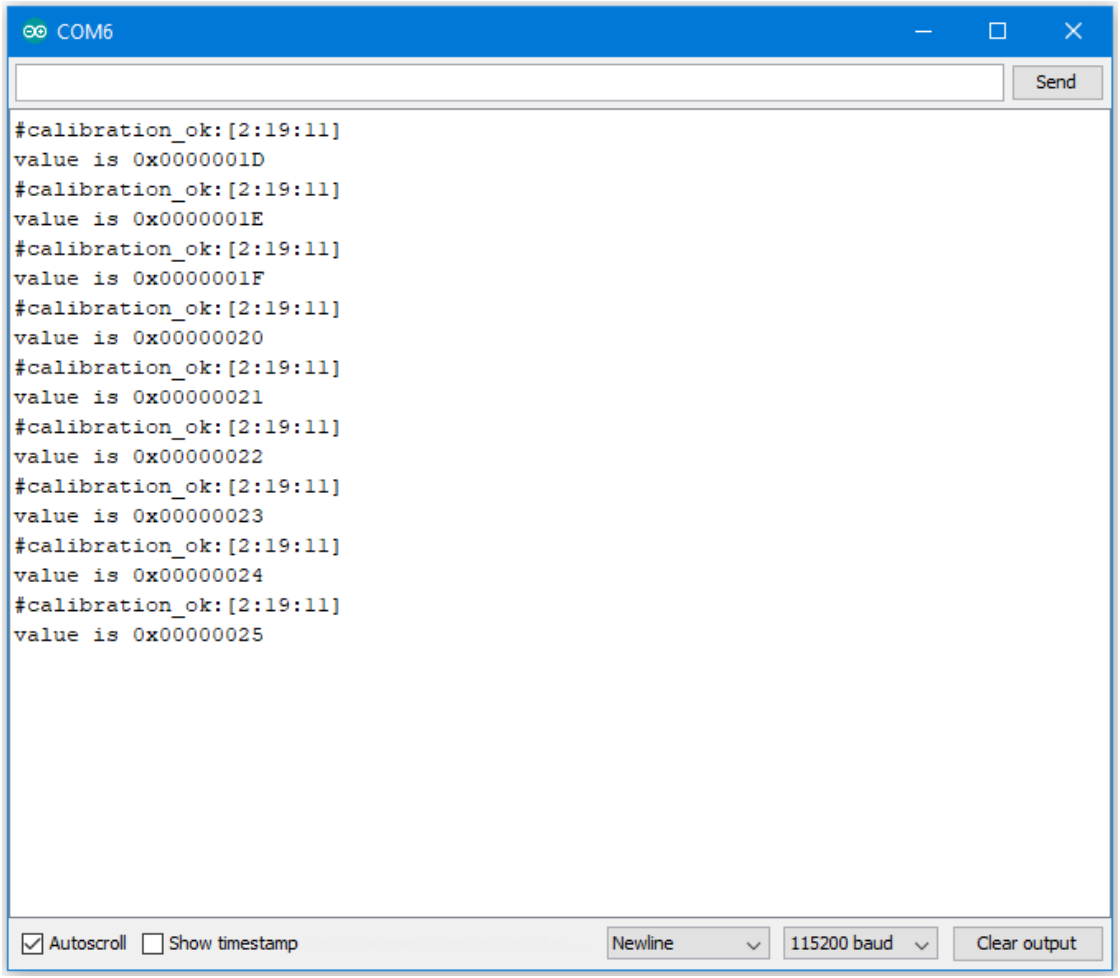
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Flash Memory API uses memory of 4K bytes, which is normally sufficient for most application. However, larger memory can be provided by specifying a specific memory address and required size.

First, open the sample code in "`File`" -> "`Examples`" -> "`AmebaFlashMemory`" -> "`ReadWriteOneWord`"

In this example, we specify the starting address of flash memory is 0xFC000 and size is 0x4000 (The default starting address is 0xFF000 and size is 0x1000).
Then calculate correct address according to the specified offset and perform read/write operation. In the sample code we use offset 0x3F00, that is, 0xFC000 + 0x3F00 = 0xFFF00 in flash. We set the value to 0 at first, then increase by 1 every time Ameba reboots.



Code Reference

We can use the flash api we used in previous flash memory example, but we need to use `begin()` function to specify the desired starting address and memory size.

```
FlashMemory.begin(0xFC000, 0x4000);
```

Use `readWord()` to read the value stored in a memory address. In the example, we read the value stored in memory offset 0x3F00, that is 0xFC000 + 0x3F00 = 0xFFF00. `readWord()` function reads a 32-bit value and returns it.

```
value = FlashMemory.readWord(0x3F00);
```

Use `writeWord()` to write to a memory address. The first argument is the memory offset, the second argument is the value to write to memory.

```
FlashMemory.writeWord(0x3F0C, value);
```

GPIO - Measure Distance By Ultrasound Module

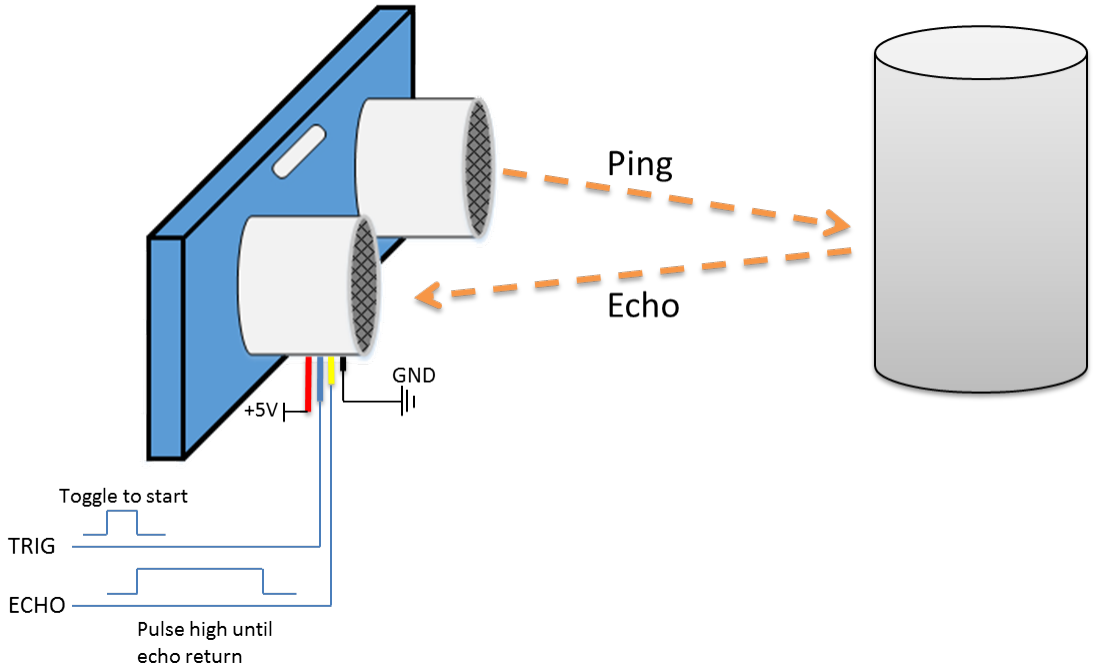
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- HC-SR04 Ultrasonic x 1
- Dropping resistor or Level converter

Example

HC-SR04 is a module that uses ultrasound to measure the distance. It looks like a pair of eyes in its appearance, therefore it's often installed onto robot-vehicle or mechanical bugs to be their eyes.

The way it works is that first we "toggle high" the TRIG pin (that is to pull high then pull low). The HC-SR04 would send eight 40kHz sound wave signal and pull high the ECHO pin. When the sound wave returns back, it pull low the ECHO pin.

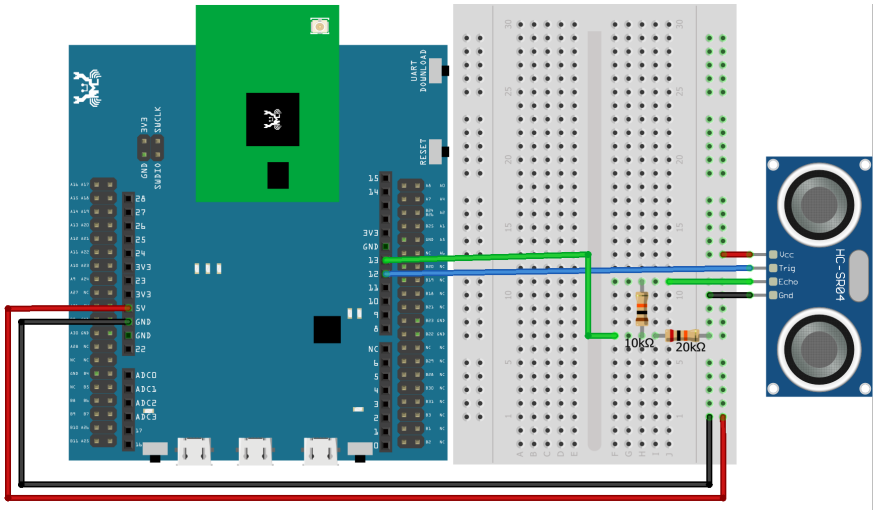


Assume the velocity of sound is 340 m/s, the time it takes for the sound to advance 1 cm in the air is 340*100*10^-6 = 29 us.

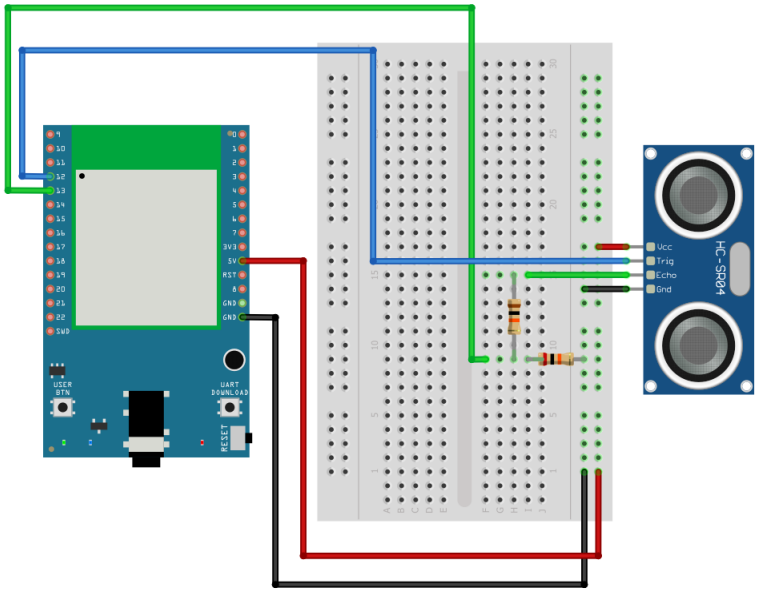
The sound wave actually travels twice the distance between HC-SR04 and the object, therefore the distance can be calculated by (time/29) / 2 = time / 58.

The working voltage of HC-SR04 is 5V. When we pull high the ECHO pin to 5V, the voltage might cause damage to the GPIO pin of Ameba. To avoid this situation, we need to drop the voltage as follows:

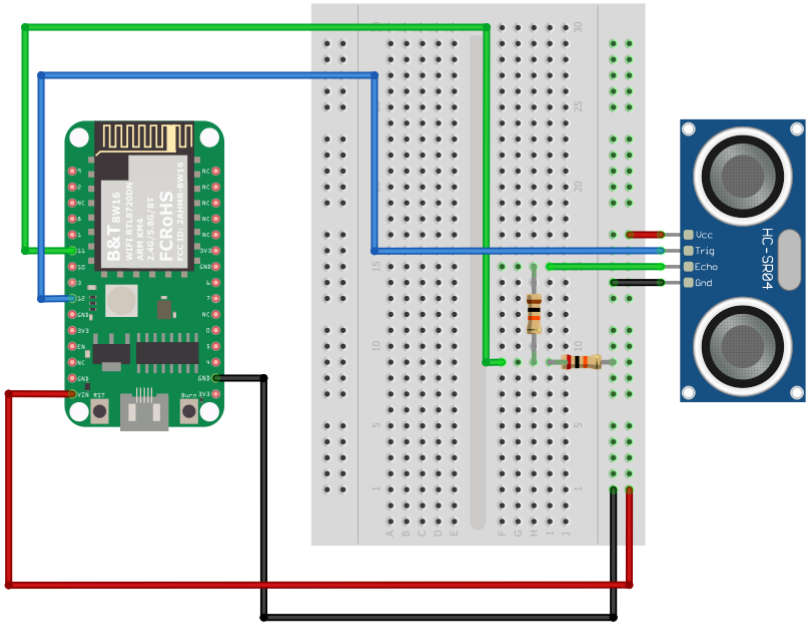
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

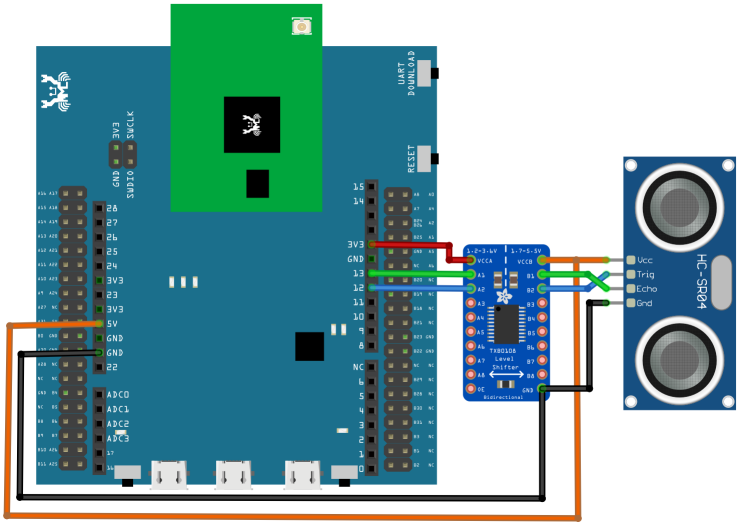


BW16 Wiring Diagram:

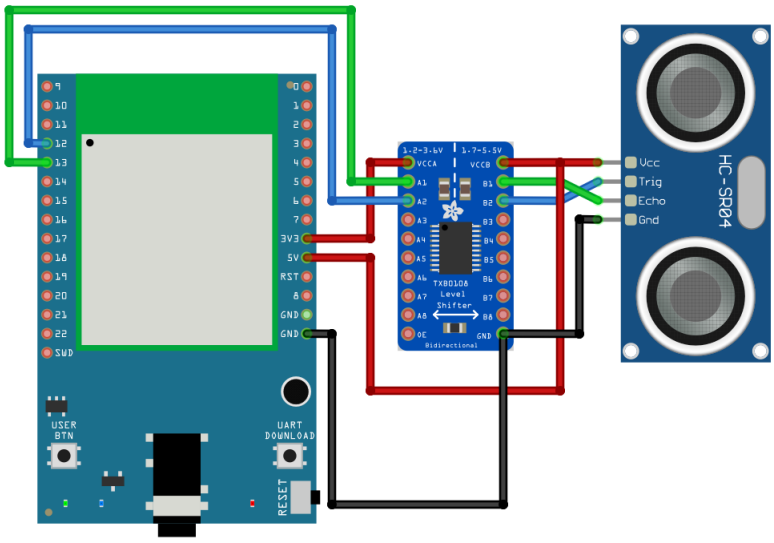


We pick the resistors with resistance 1:2, in the example we use 10kΩ and 20kΩ.
If you do not have resistors in hand, you can use level converter instead.The TXB0108 8 channel level converter is a suitable example:

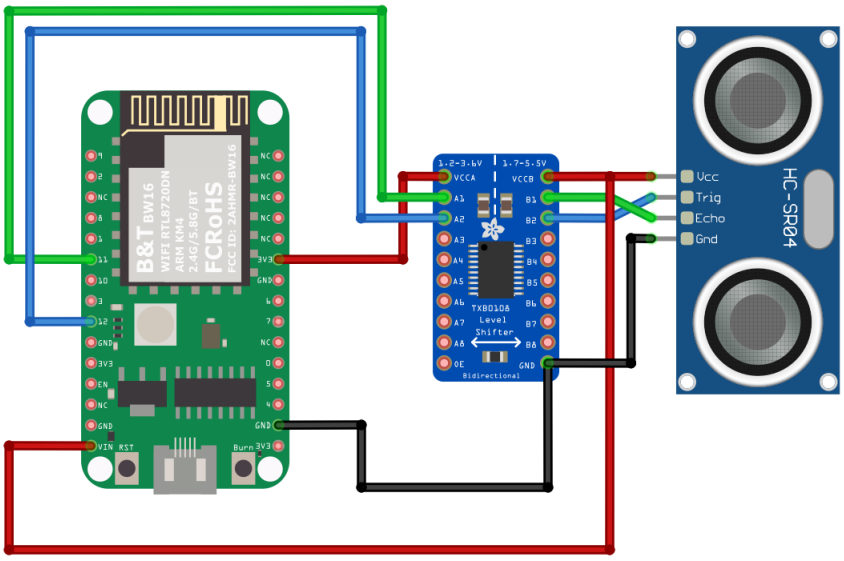
AMB21 / AMB22 Wiring Diagram:



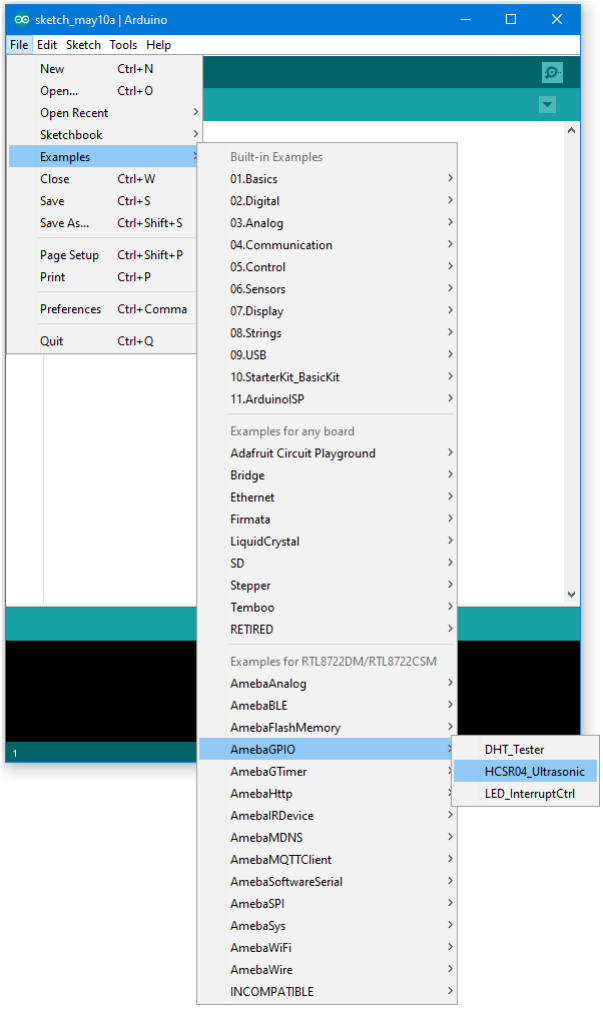
AMB23 Wiring Diagram:



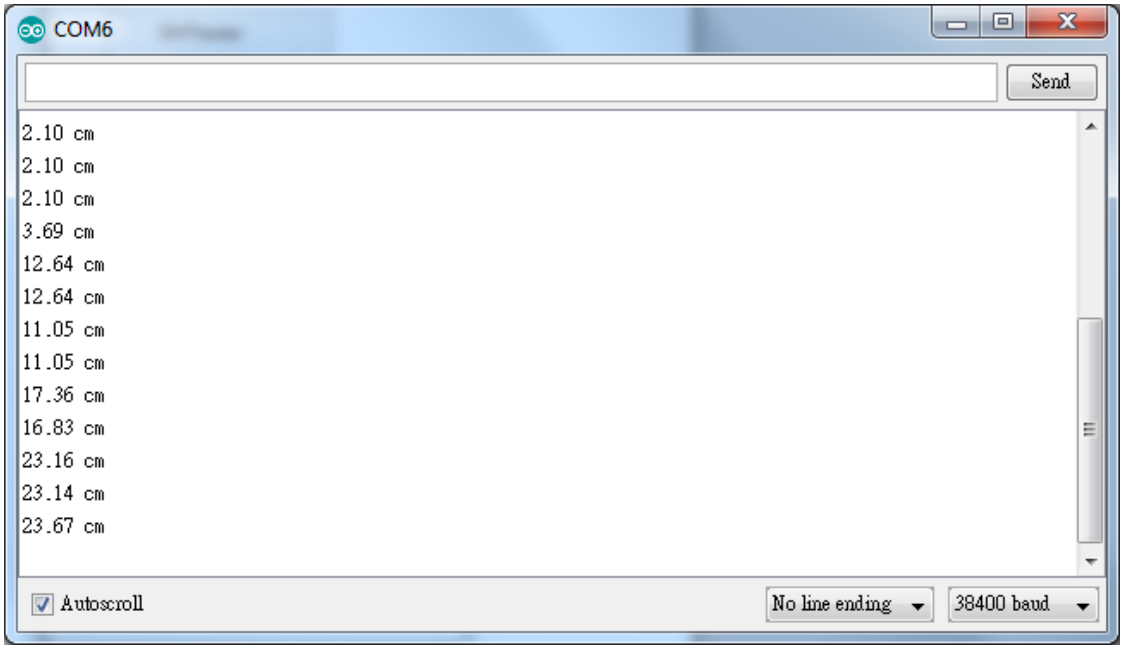
BW16 Wiring Diagram:



Next, open the sample code in "`File` -> "`Examples`" -> "`AmebaGPIO`" -> "`HCSR04_Ultrasonic`"



Compile and upload to Ameba, then press the reset button. Open the Serial Monitor, the calculated result is output to serial monitor every 2 seconds.



Note that the HCSR04 module uses the reflection of sound wave to calculate the distance, thus the result can be affected by the surface material of the object (e.g., harsh surface tends to cause scattering of sound wave, and soft surface may cause the sound wave to be absorbed).

Code Reference

Before the measurement starts, we need to pull high the TRIG pin for 10us and then pull low. By doing this, we are telling the HC-SR04 that we are about to start the measurement:

```
digitalWrite(trigger_pin, HIGH);
delayMicroseconds(10);
digitalWrite(trigger_pin, LOW);
```

Next, use pulseIn to measure the time when the ECHO pin is pulled high.

```
duration = pulseIn (echo_pin, HIGH);
```

Finally, use the formula to calculate the distance.

```
distance = duration / 58;
```

GPIO - Measure Temperature and Humidity

Preparation

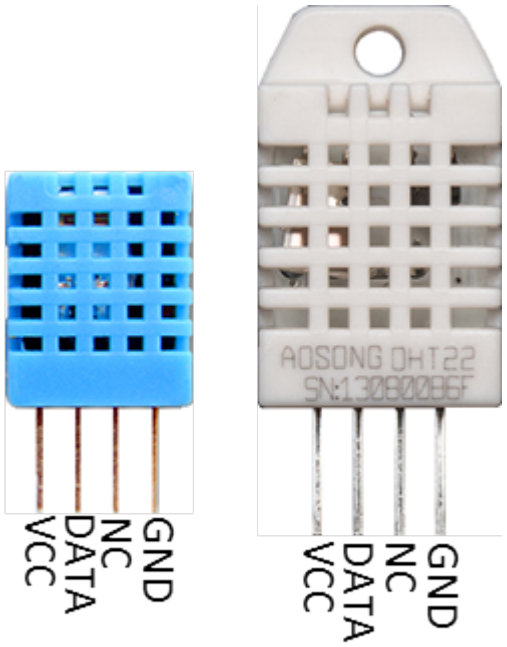
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21

Example

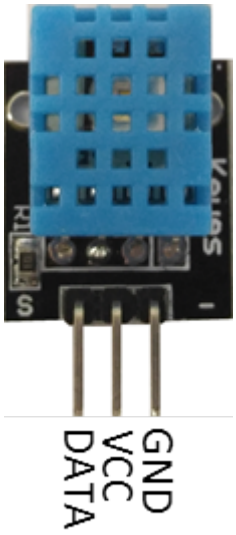
DHT11 is a temperature and humidity sensor which operates at voltage 3.3V~5V. At room temperature, the measurable range of the humidity is 20% ~ 90%RH with ±5%RH precision, the measurable range of the temperature is 0 ~ 50℃ with ±2℃ precision.

Another choice of temperature and humidity sensor is DHT22 sensor, which has better precision. Its measurable range of the humidity is 0%~100%RH with ±5%RH precision, the measurable range of the temperature is -40~125 ℃ with ±0.2℃ precision.

There are 4 pins on the sensor:



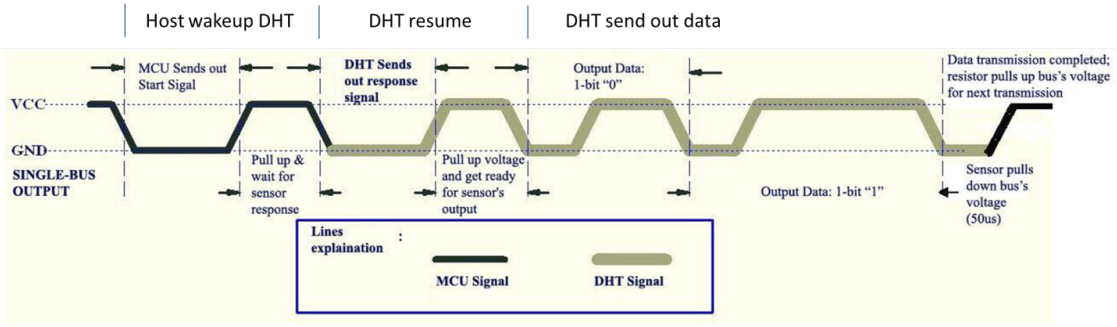
Since one of the 4 pins has no function, there are temperature/humidity sensors with only 3 pins on the market:



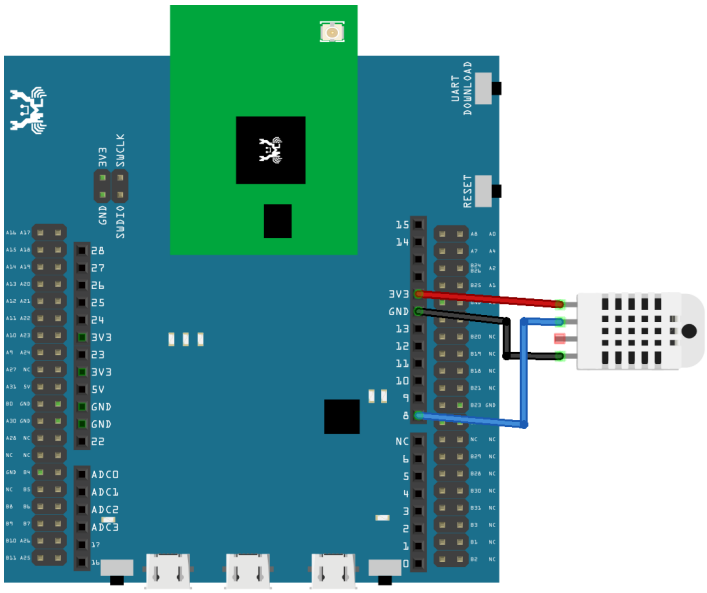
DHT is normally in the sleeping mode. To get the temperature/humidity data, please follow the steps:

1. Awake DHT: Ameba toggles low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital out to Ameba.

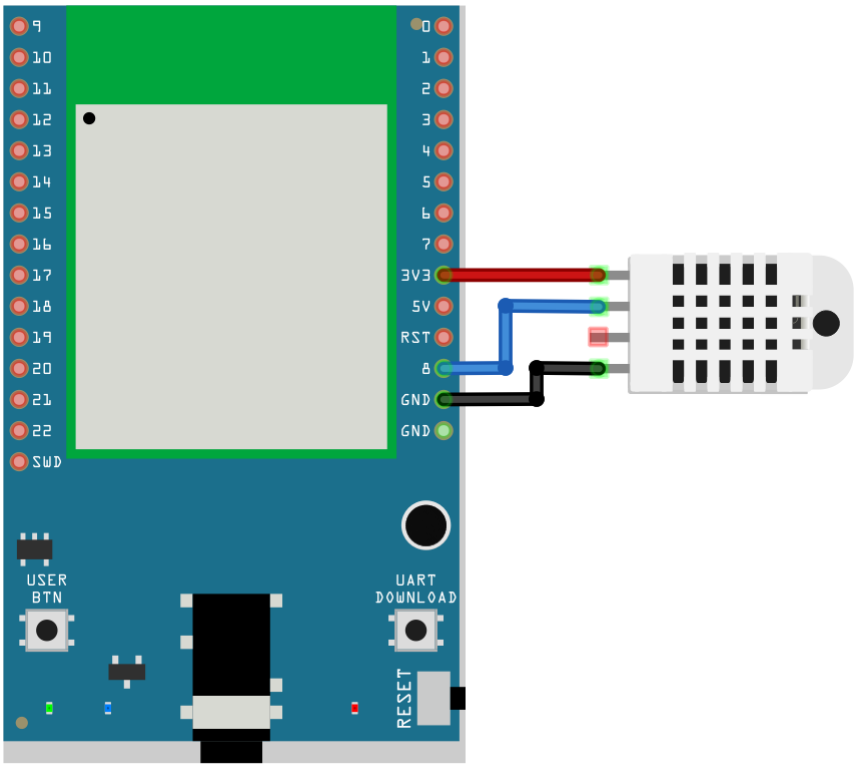
2. DHT response: DHT also toggle low its DATA pin of GPIO. Now the DATA pin of GPIO serves as digital in for Ameba.
3. DHT sends data: DHT sends out the temperature/humidity data (which has size 5 bytes) in a bit by bit manner. To represent each bit, DHT first pull low the DATA GPIO pin for a while and then pull high. If the duration of high is smaller than low, it stands for bit 0. Otherwise it stands for bit 1.



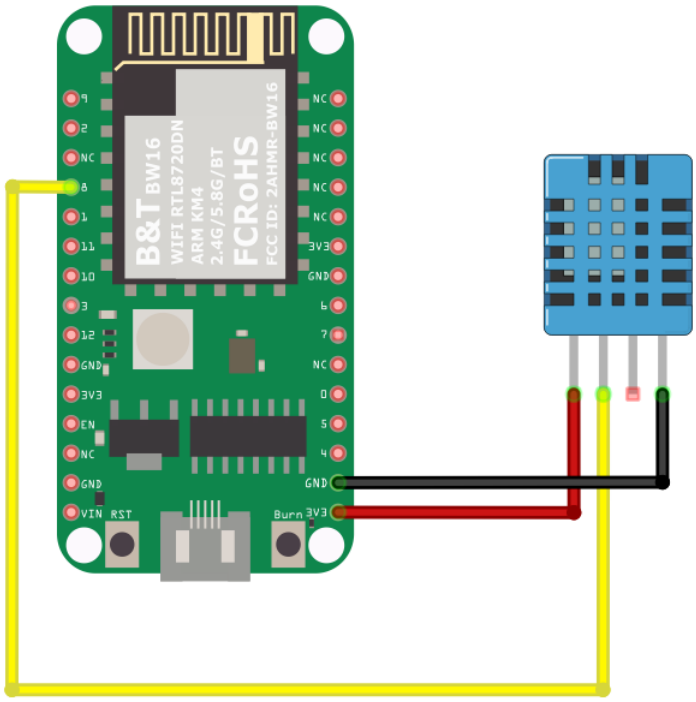
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

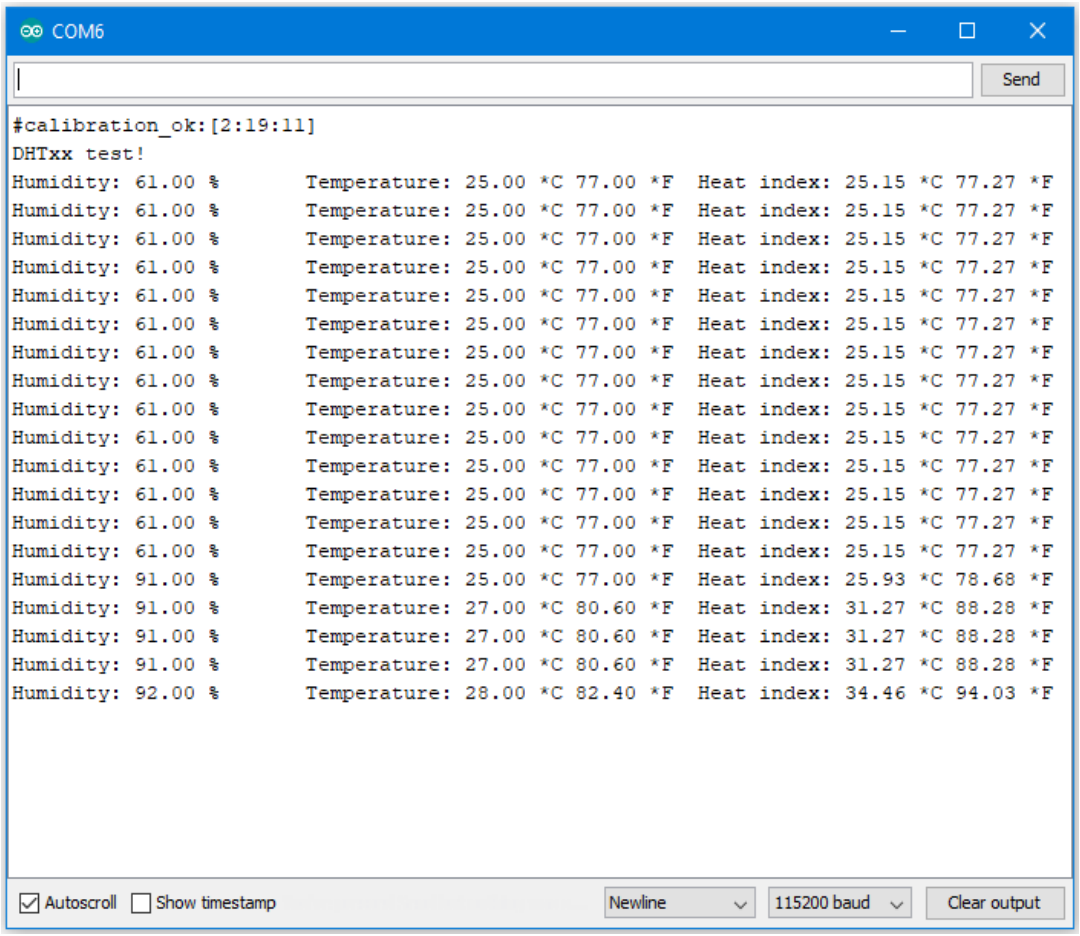


BW16 Wiring Diagram:



Open the sample code in "`Files`" -> "`Examples`" -> "`AmebaGPIO`" -> "`DHT_Tester`". Compile and

upload to Ameba, then press the reset button. The result would be shown on the Serial Monitor.



```
#calibration_ok:[2:19:11]
DHTxx test!
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 61.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.15 *C 77.27 *F
Humidity: 91.00 %	Temperature: 25.00 *C 77.00 *F	Heat index: 25.93 *C 78.68 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 91.00 %	Temperature: 27.00 *C 80.60 *F	Heat index: 31.27 *C 88.28 *F
Humidity: 92.00 %	Temperature: 28.00 *C 82.40 *F	Heat index: 34.46 *C 94.03 *F
```

Code Reference

Use `dht.readHumidity()` read the humidity value, and use `dht.readTemperature()` to read the temperature value.

Every time we read the temperature/humidity data, Ameba uses the buffered temperature/humidity data unless it found the data has expired (i.e., has not been updated for over 2 seconds). If the data is expired, Ameba issues a request to DHT to read the latest data.

GPIO - Use GPIO Interrupt To Control LED

Preparation

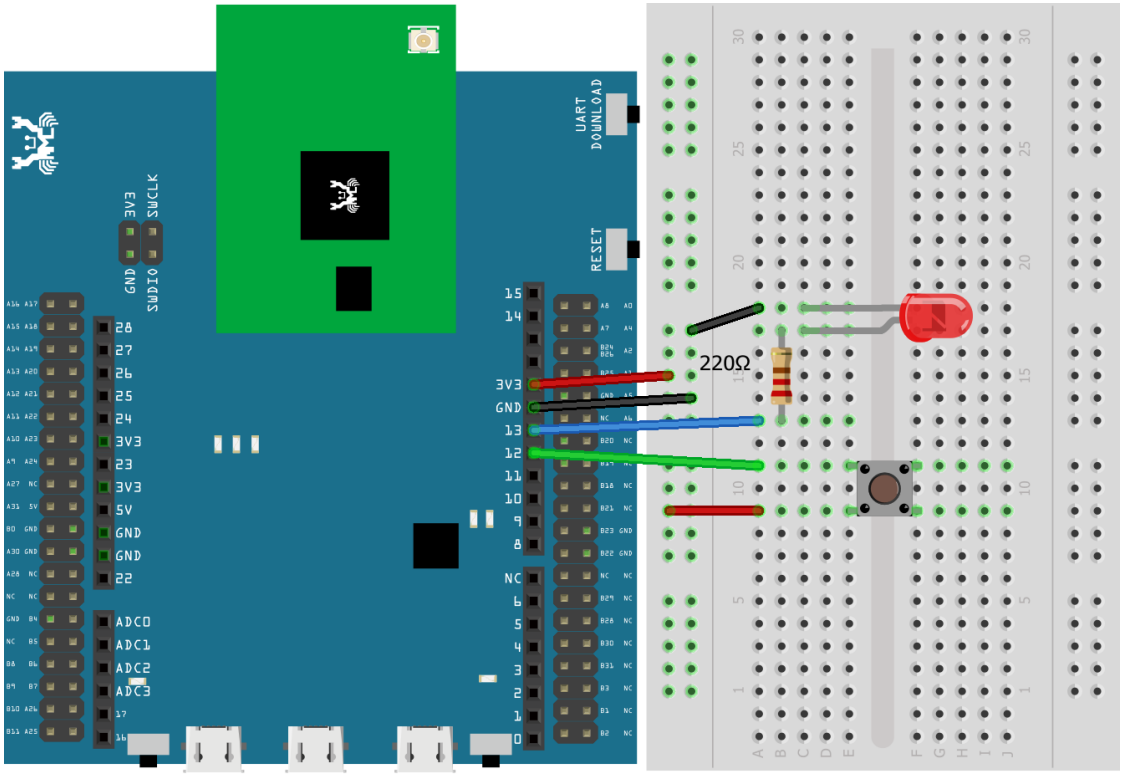
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- LED x 1
- Button x 1

Example

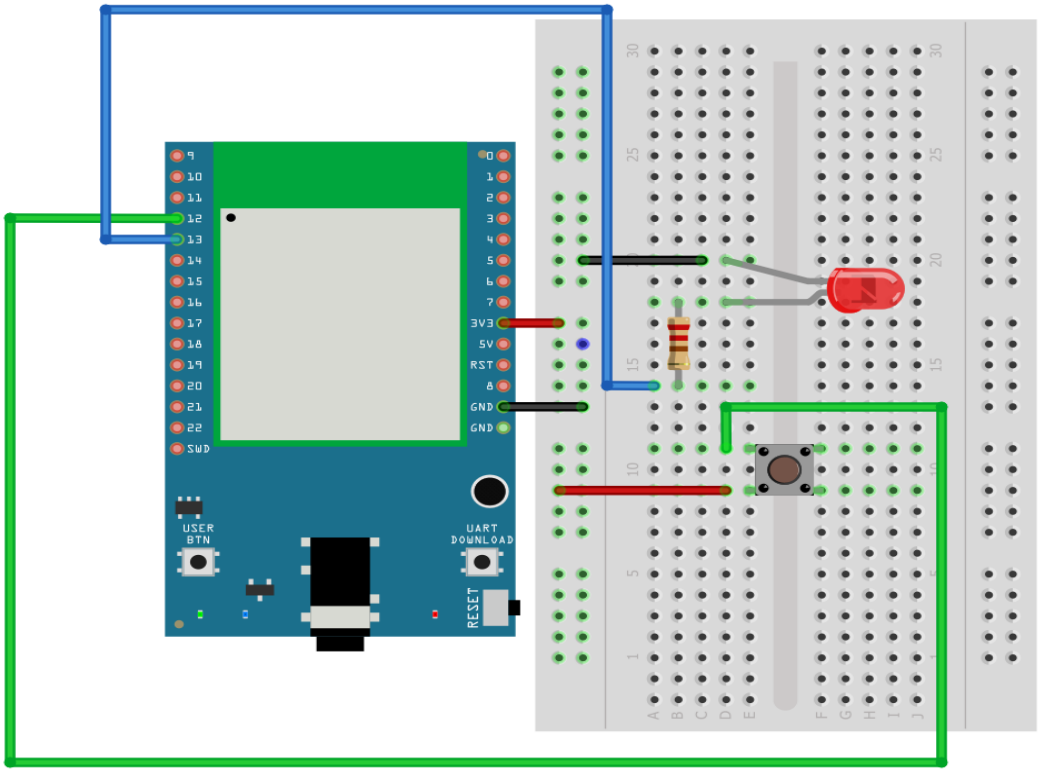
In this example, we use a button to trigger interrupt and control the LED. When we press and release the button, the LED dims, press and release the button again, and the LED lights.Note that in the Arduino example “Button and LED”, LED only lights when the button is pressed and hold, when we release the button, the LED dims.

Open the example, “`Files`” -> “`Examples`” -> “`AmebaGPIO`” -> “`LED_InterruptCtrl`”

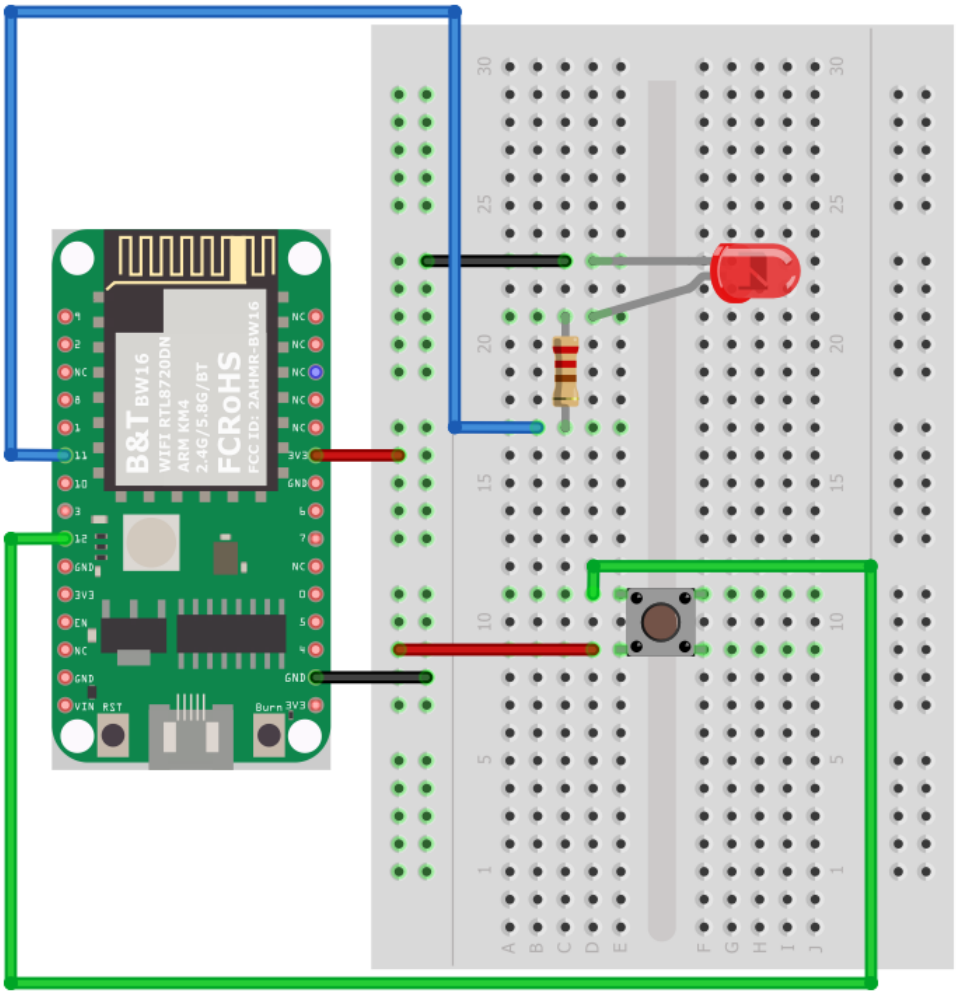
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Compile and upload the program, press reset.
The LED lights at first. Press and release the button, then the LED should dim. Press again, then the LED should light.

Code Reference

In

```
setup()
```

we set Pin 12 to

```
INPUT_IRQ_RISE
```

, this means that an interrupt occurs when the voltage of this pin changes from GND to 3V3. Therefore, we connect the other side of the button to 3V3, so as to trigger interrupt event when the button is pressed.

```
pinMode(button, INPUT_IRQ_RISE);
```

On the other hand, we can set pin 12 to

```
INPUT_IRQ_FALL
```

, this means that an interrupt occurs when the voltage of this pin changes from 3V3 to GND. In this case, the other side of the button is connected to GND.Next, we need to specify the funtion to be execute to handle the interrupt:

```
digitalSetIrqHandler(button, button_handler);
```

The second parameter is a function pointer, with prototype:

```
void button_handler(uint32_t id, uint32_t event)
```

In this handler, every time we press and release the button, we trigger an interrupt, and change the status of the LED.

GTimer - Using The Periodic GTimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

Ameba provides 4 hardware GTimer for users to use. The timers' resolutions are at microseconds scale.
The timer can be set to be periodic or for single use. The periodic timers reset periodically, and the single-use timers do not.

Open the example, "`File" -> "Examples" -> "AmebaGTimer" -> "TimerPeriodical`". Compile and upload to Ameba, and press reset.
In the Serial Monitor, you can see the counter value is increased periodically.

Code Reference

The first argument of begin() is the timer id (0~3).
The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s.
The third argument specifies the function to call when the time is up. In the example, we call the "myhandler" function to increase the counter value by 1 and print the counter value to serial monitor.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler);
```

The GTimer is periodic by default, therefore "myhandler" function is called every second. When we want to stop the GTimer, use `stop()`:

```
GTimer.stop(0);
```

GTimer - Using the One-Time Gtimer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

In this example, we will use 4 One-Time GTimer, and pass user data to each timer.

Open the example "File" -> "Examples" -> "AmebaGTimer" -> "TimerOneshot". Compile and upload to Ameba, and press reset. Then you can see the 4 timer log printed to the serial monitor in series.

Code Reference

The first argument of begin() is the Timer ID (0~3). The second argument is the value of the timer (in microseconds). In the example, we fill in 1000000us = 1s. The third argument specifies the function to call when the time is up. The fourth argument is to set whether this timer is a periodic timer, we use "false" here to begin a single-use timer. The fifth argument is the user data, we give 0 here to represent that this is timer 0.

```
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
```

Next, we set up the second timer, which has timer value 2 seconds, and user data 1. And other timers are set similarly.

```
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
```

In myhandler function, we print the user data to serial monitor. Since the 4 timers are separately set to count for 1, 2, 3, 4 seconds, from 1 second to 4 second, the user data of each timer are printed on the serial monitor in order. After 4 second, no log will be printed.

I2C - Send Data to Arduino UNO

Introduction of I2C

> There are two roles in the operation of I2C, one is "master", the other is "slave". Only one master is allowed and can be connected to many slaves. Each slave has its unique address, which is used in the communication between master and the slave. I2C uses two pins, one is for data transmission (SDA), the other is for the clock (SCL). Master uses the SCL to inform slave of the upcoming data transmission, and the data is transmitted through SDA. The I2C example was named "Wire" in the Arduino example.

Materials

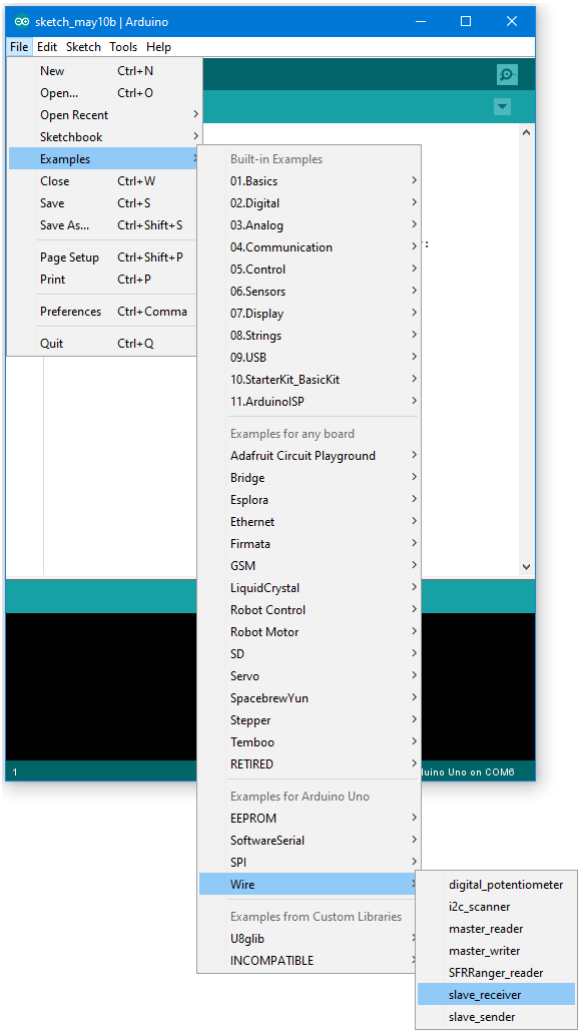
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In this example, we use Ameba as the I2C master writer, and use Arduino as the I2C slave receiver.
When the I2C slave receives string sent from I2C master, it prints the received string.

- Setting up Arduino Uno to be I2C Slave

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`"
Open the "Slave Receiver" example in "`Examples`" -> "`Wire`" -> "`slave_receiver`":

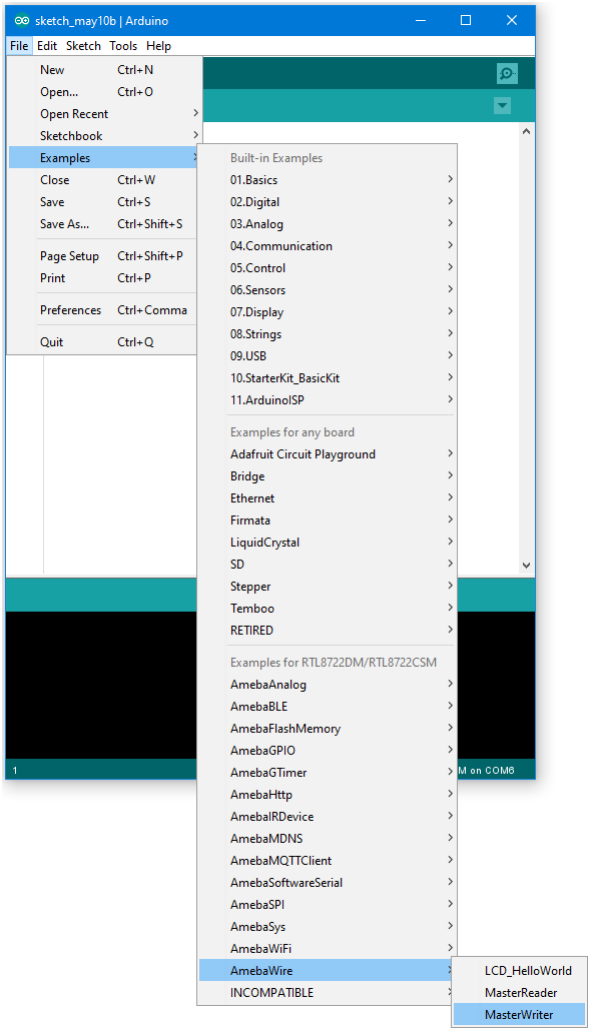


Then click "`Sketch`" -> "`Upload`" to compile and upload the example to Arduino Uno.

- Setting up Ameba to be I2C Master

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE:
"`Tools`" -> "`Board`"

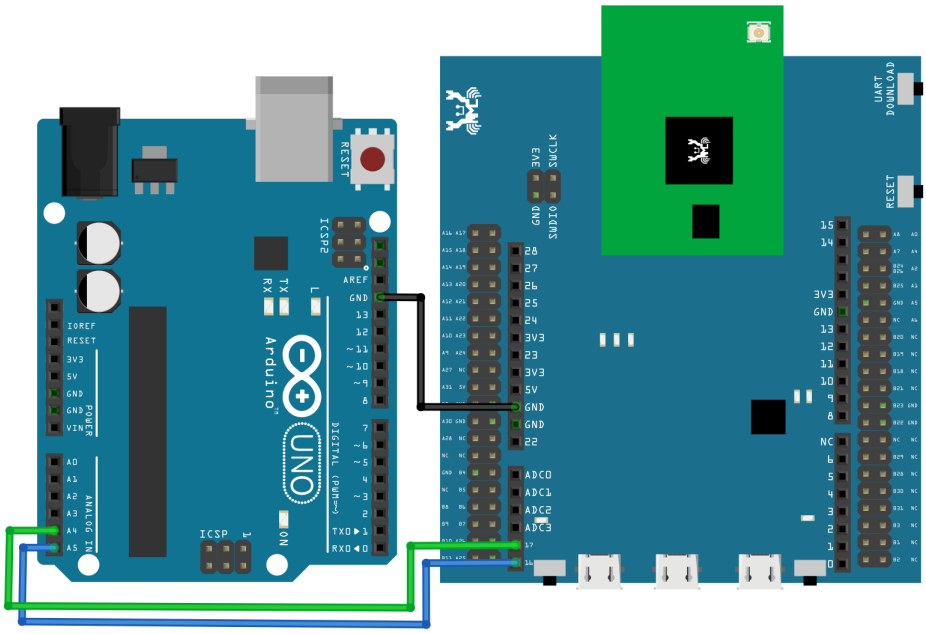
Then open the "Master Writer" example in "`File`" -> "`Examples`" -> "`AmebaWire`" -> "`MasterWriter`"



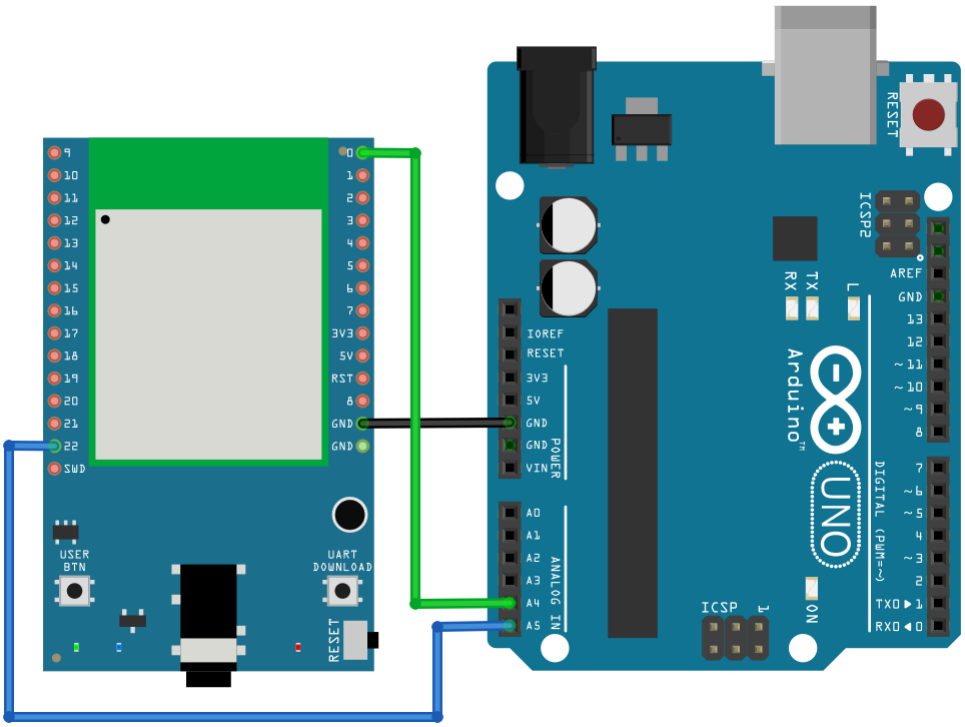
- Wiring

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

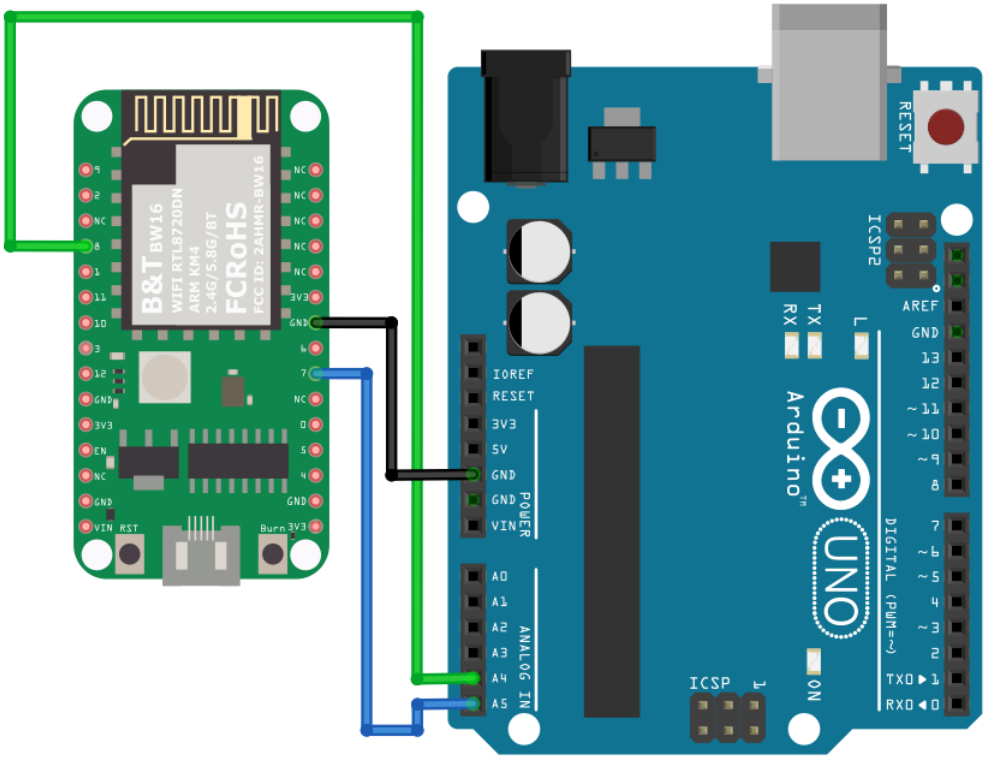
AMB21/ AMB22 Wiring Diagram:



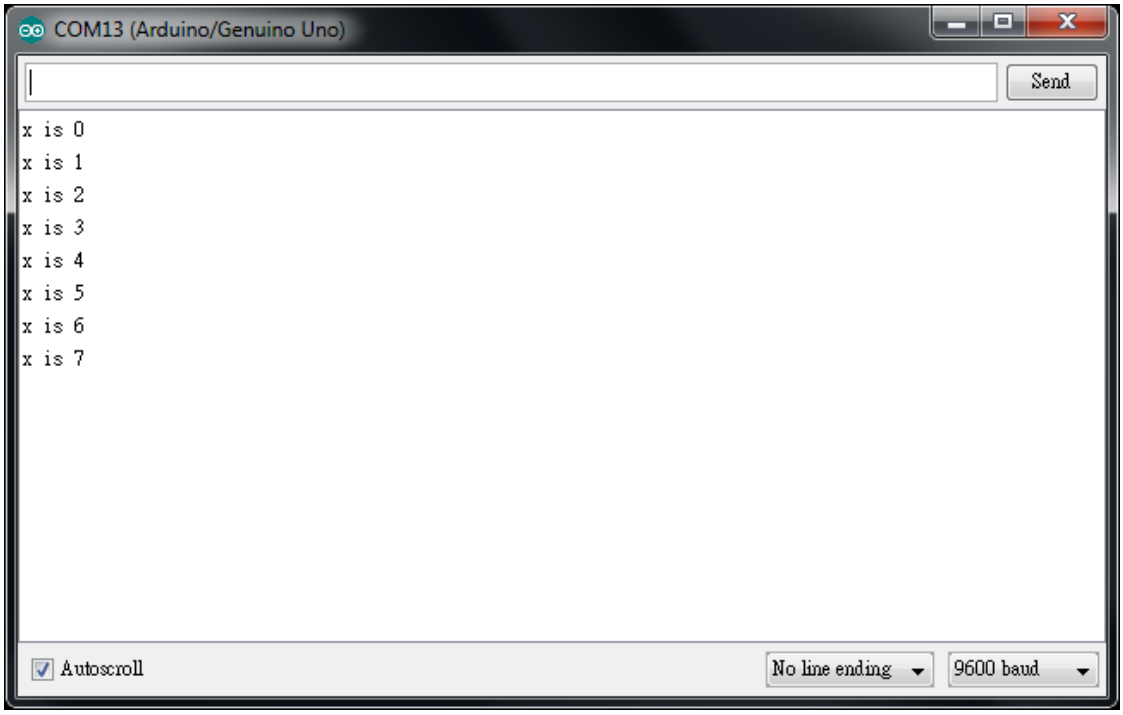
AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the Arduino IDE of the Arduino Uno and open the serial monitor ("Tools" -> "Serial Monitor").
In the Serial Monitor, you can see the messages printed from Arduino Uno.
Next, press the reset button on Arduino Uno. Now the Arduino Uno is waiting for the connection from I2C Master.
We press the reset button on Ameba to start to send messages. Then observe the serial monitor, you can see the messages show up every half second.



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterWriter

First use Wire.begin()/Wire.begin(address) to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses Wire.beginTransmission(address) to begin a transmission to the I2C slave with the given address:
https://www.arduino.cc/en/Reference/WireBeginTransmission

Uses Wire.write() to send data, and finally use Wire.endTransmission() to end a transmission to a Slave and transmits the bytes that were queued:
https://www.arduino.cc/en/Reference/WireEndTransmission

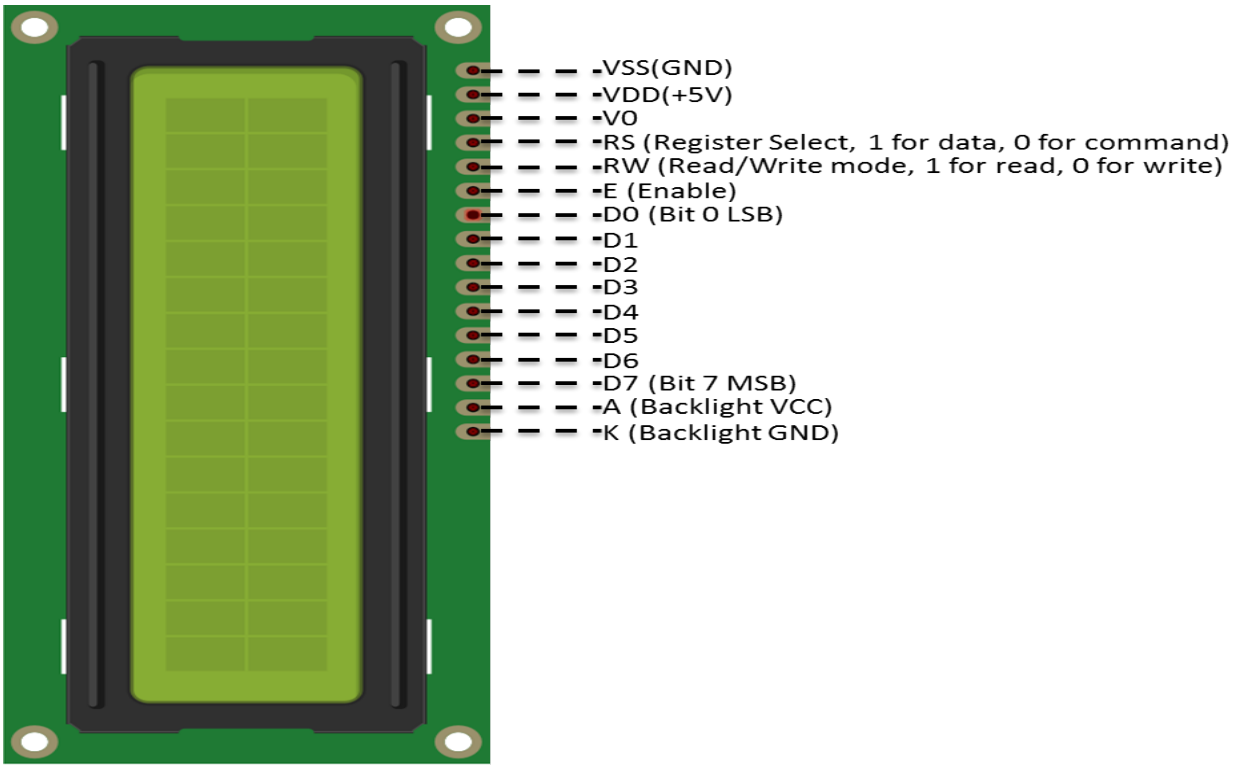
I2C - Display Data On LCD Screen

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- I2C 2×16 LCD

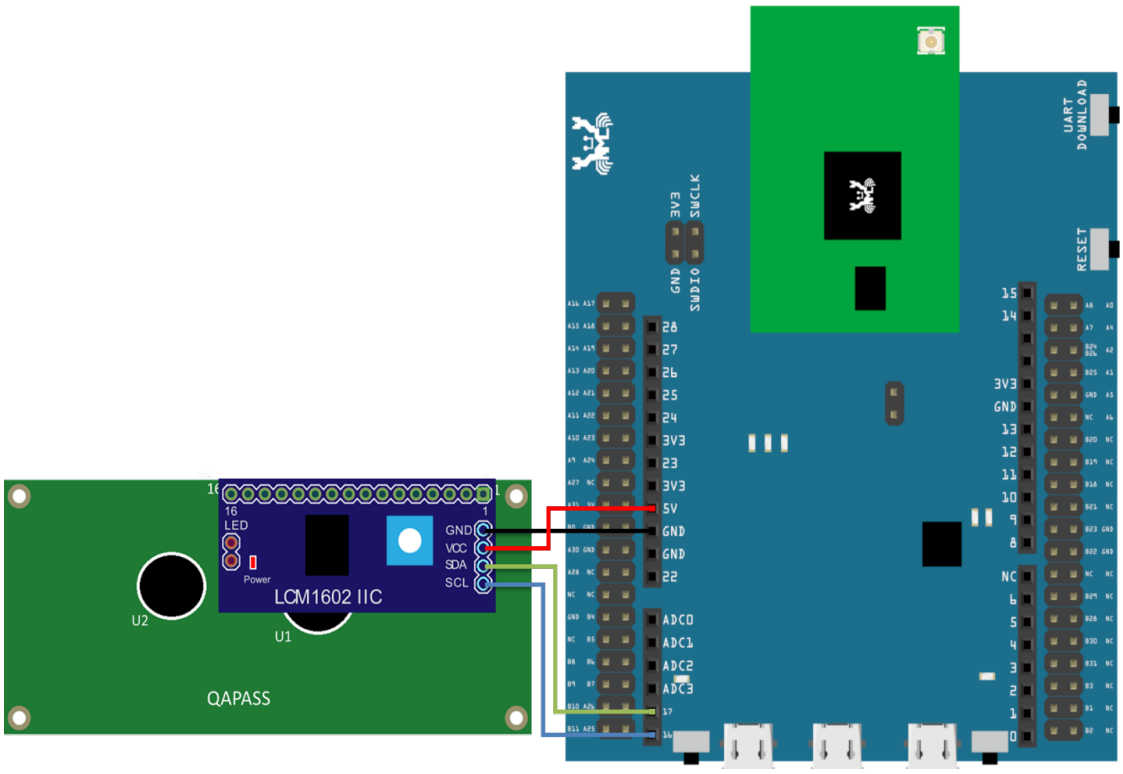
Example

Normally there are many pins on an LCD display, as shown in below figure.

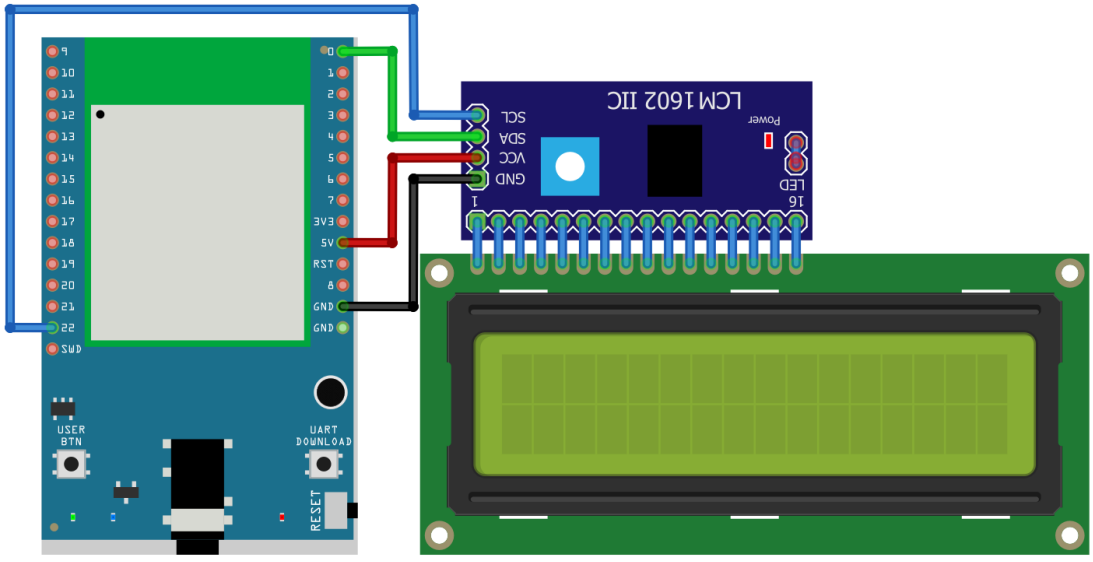


An LCD display can be equipped with an additional processing chip to process the data. The processing chip can connect to a microcontroller using the I2C interface.

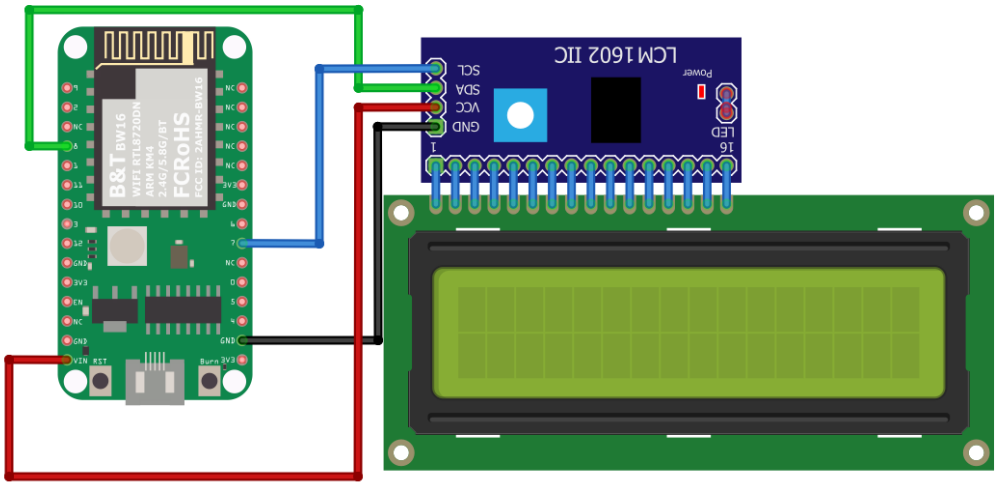
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

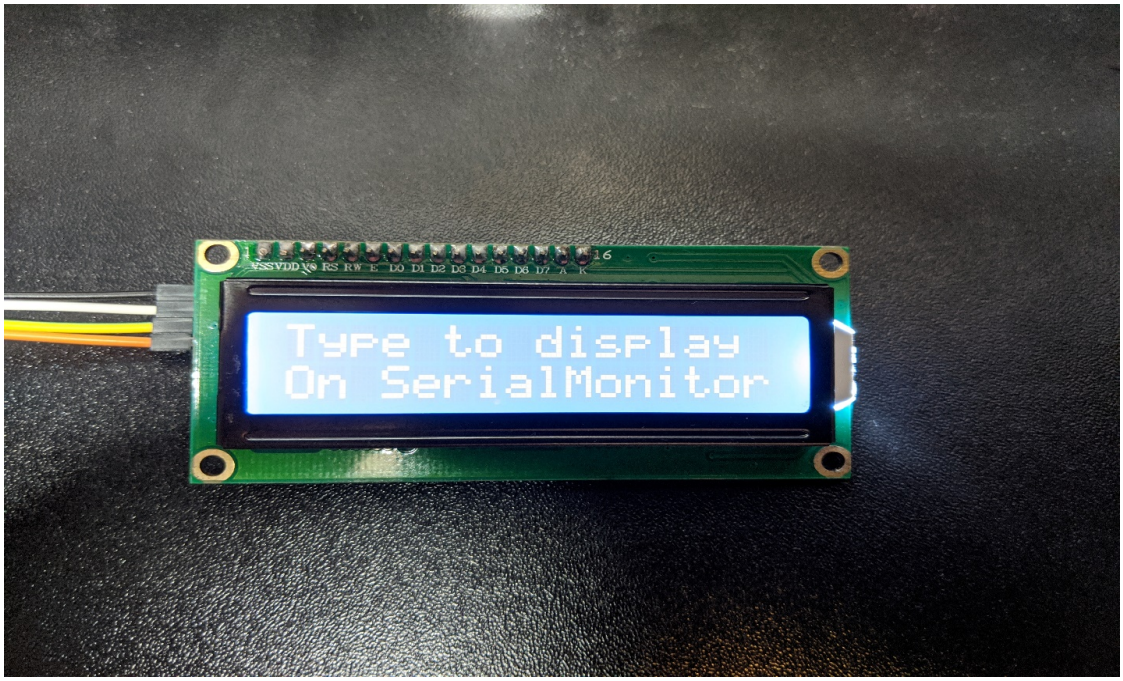


BW16 Wiring Diagram:



Open the example in "`File` -> "`Examples`" -> "`AmebaWire`" -> "`LCD_HelloWorld`".
Compile and upload to Ameba, then press the reset button.
Then you can see "Hello World" in the first line, and "Ameba" in the second line displayed on the LCD screen.

After 8 seconds, you can input to the Serial Monitor the string you would like to display on the LCD.



For example, we enter “123456789” and press “Send”:

Code Reference

The required settings of each model of LCD might be different, the constructor we use in this example is:

```
LiquidCrystal_I2C(uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs,
                  uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7,
                  uint8_t backlighPin, t_backlighPol pol);
```

And the setting parameters are as follows:

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C␣
↪address
```

The first parameter 0x27 is the address of I2C. Each of the following 8 parameters represents the meaning of each bit in a byte, i.e., En is bit 2, Rw is bit 1, Rs is bit 0, d4 is bit 4, and so forth.

Call `backlight()` to light the screen,
Call `setCursor(0, 0)` to set the position of the cursor.
LCD inherits the Print class, so we can use `lcd.print()` to output string on the screen.

I2C - Receive Data from Arduino UNO

Materials

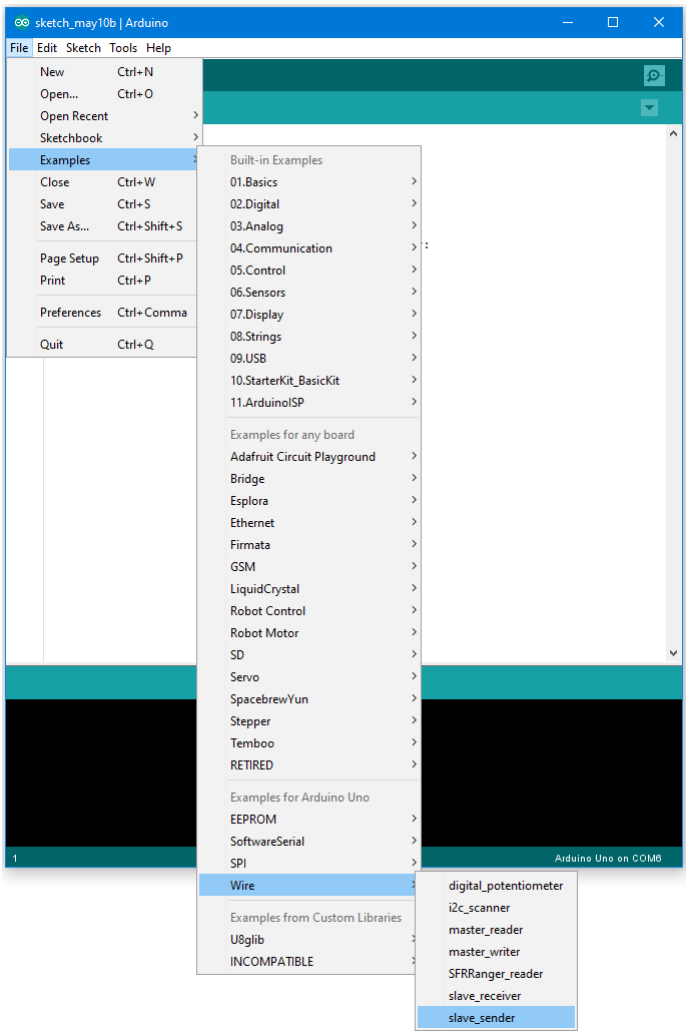
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Arduino UNO x 1

Example

In the previous example "I2C – Communicate with Arduino UNO via I2C" , Ameba, the I2C master, transmits data to the Arduino UNO, the I2C slave.
As to this example, Ameba is the I2C master, and receives data from the Arduino UNO, which is the I2C slave.

- **Setting up Arduino Uno to be I2C Slave**

First, select Arduino in the Arduino IDE in "`Tools`" -> "`Board`" -> "`Arduino Uno`":
Open "`Examples`" -> "`Wire`" -> "`slave_sender`"

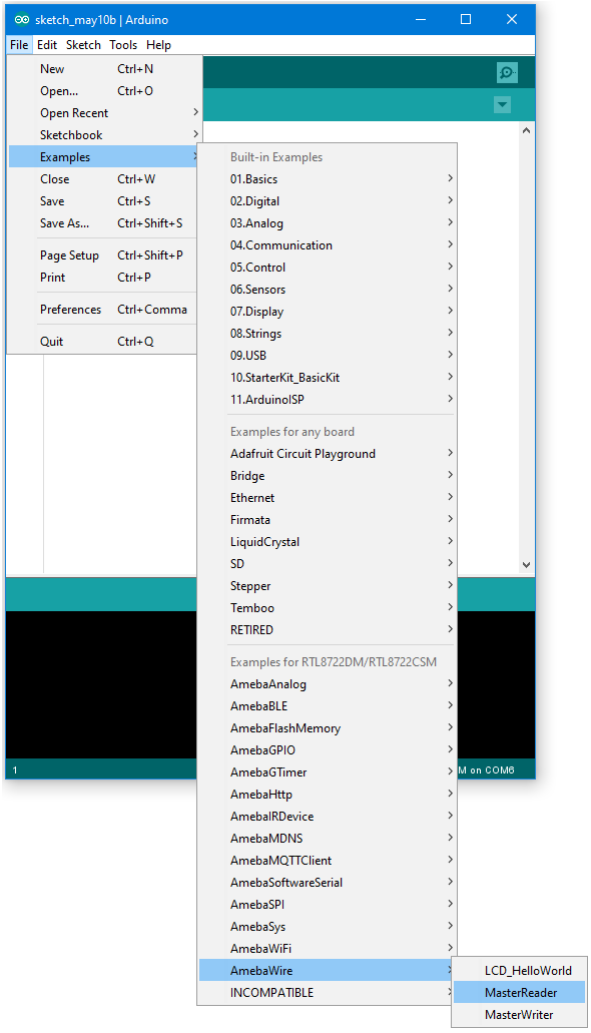


Then click “Sketch” -> “Upload” to compile and upload the example to Arduino Uno.

- **Setting up Ameba to be I2C Master**

Next, open another window of Arduino IDE, make sure to choose your Ameba development board in the IDE: “Tools” -> “Board”

Open “`File`” -> “`Examples`” -> “`AmebaWire`” -> “`MasterReader`”

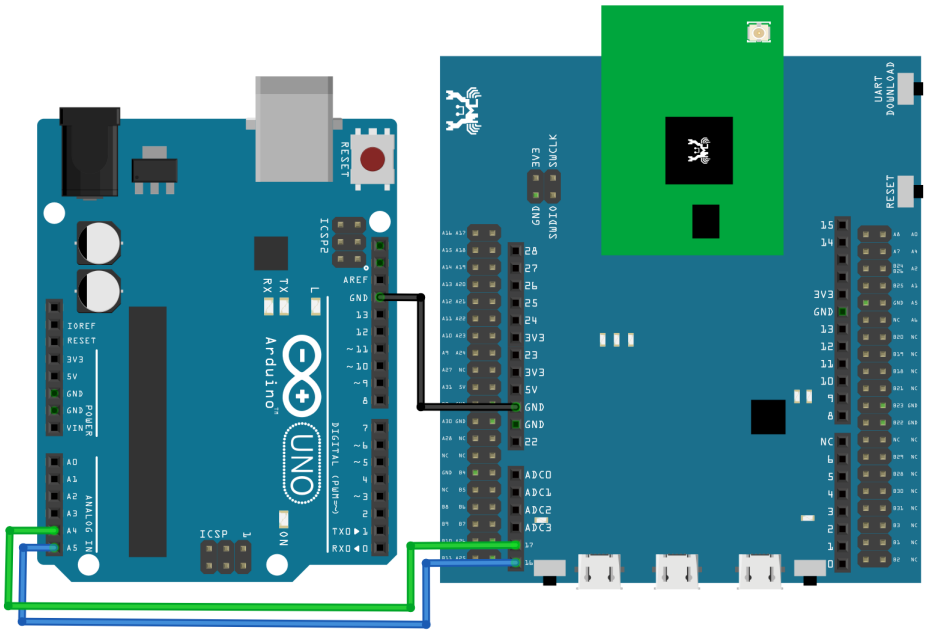


Click “Sketch” -> “Upload” to compile and upload the example to Ameba.

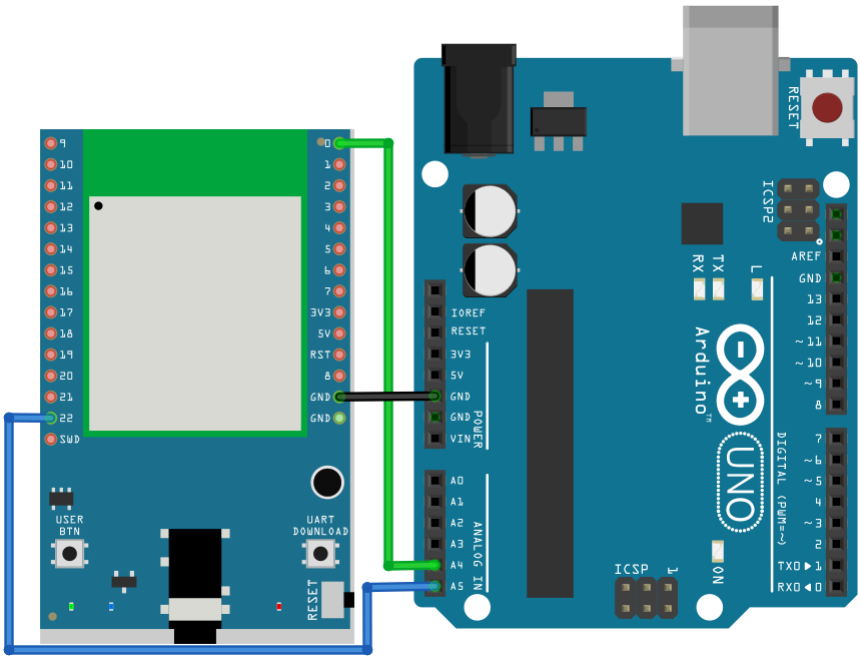
- **Wiring**

The Arduino example uses A4 as the I2C SDA and A5 as the I2C SCL.
Another important thing is that the GND pins of Arduino and Ameba should be connected to each other.

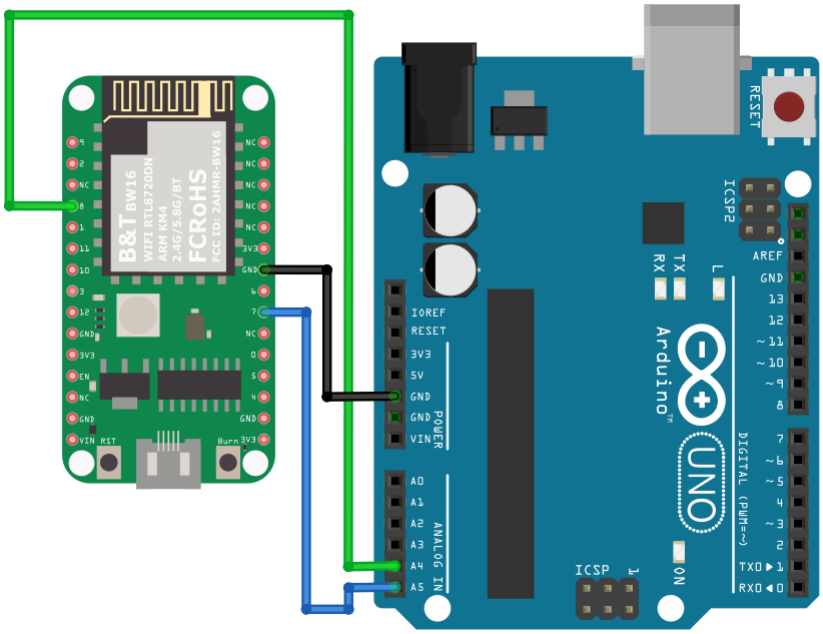
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

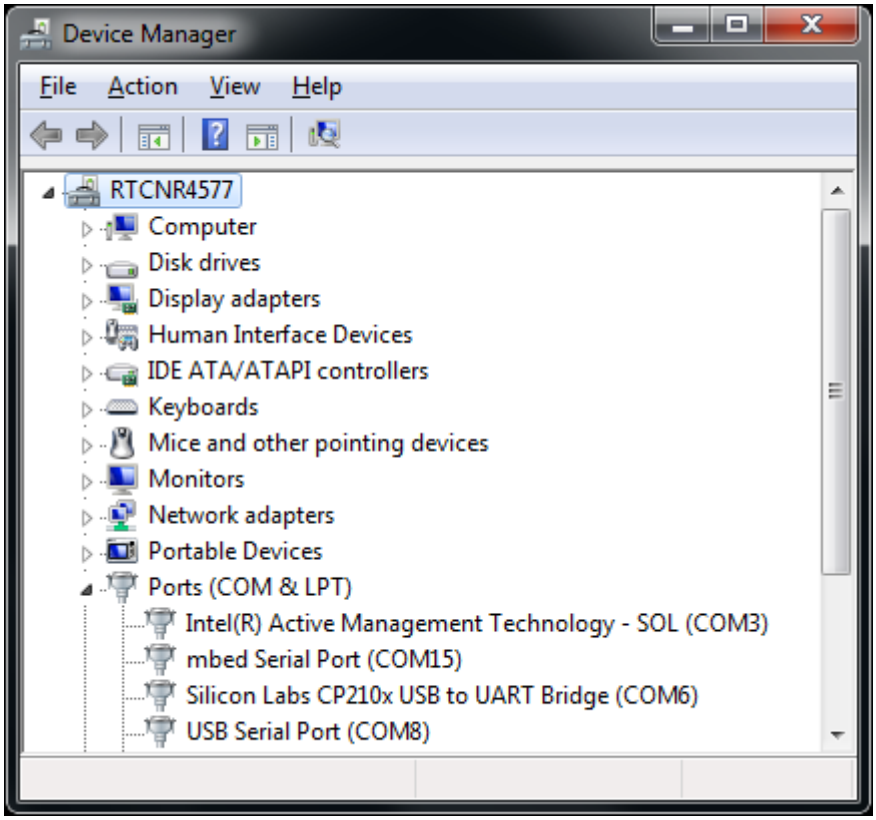


BW16 Wiring Diagram:



Next, we will observe the data receive by Ameba in the Serial Monitor.

(Note: If you do not know which port the Ameba development board is connected to, please find it in the Device Manager of Windows first. Ameba is connected as “mbed Serial Port”. For example, if you find mbed Serial Port (COM15) means Ameba is connected to port COM15.)



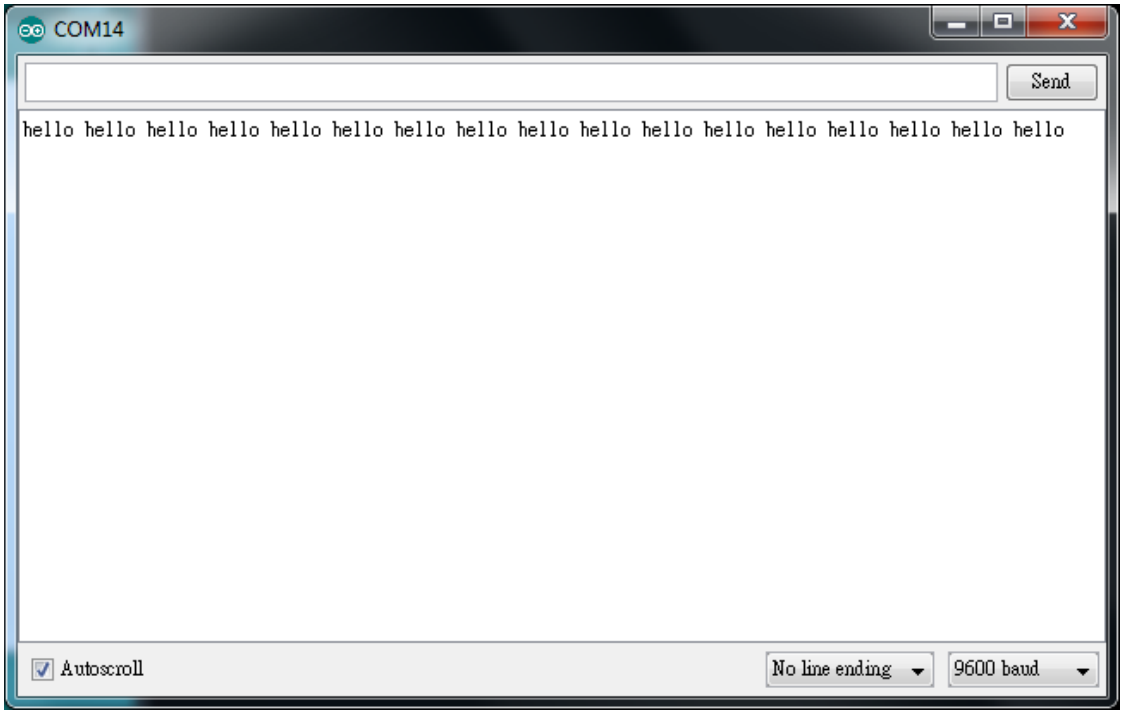
We select the port in “Tools” -> “Port” -> “COM15” (the port connected to Ameba)

Open the Arduino IDE window of the Ameba, go to “Tools” -> “Serial Monitor” to display the messages printed by Ameba.

Press the reset button on Arduino Uno, Arduino Uno now waits for connection from I2C master.

Then press the reset button on Ameba, Ameba will start to receive messages from Arduino Uno. And you can see the “hello ” message printed every half second in serial monitor.

(NOTE: If the message does not show in the Serial Monitor of Ameba, please close and open the serial monitor again.)



Code Reference

You can find detailed information of this example in the documentation of Arduino:
https://www.arduino.cc/en/Tutorial/MasterReader

First use `Wire.begin()` / `Wire.begin(address)` to join the I2C bus as a master or slave, in the Master case the address is not required.
https://www.arduino.cc/en/Reference/WireBegin

Next, the Master uses `Wire.requestFrom()` to specify from which Slave to request data.
https://www.arduino.cc/en/Reference/WireRequestFrom

IR - Transmit IR NEC Raw Data And Decode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 2
- Grove – Infrared Emitter x1 (Figure 1)
- Grove – Infrared Receiver x1 (Figure 2)

Example

In this example, we use two Ameba RTL8722 modules that connecting with an infrared (IR) Emitter and an IR Receiver separately to transmit and receive IR NEC Raw data.

Figure 1: Grove – Infrared Receiver

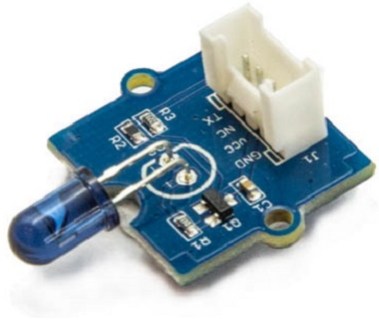


Figure 2: Grove – Infrared Emitter

On the transmission side, the transmitter will send IR NEC raw data. The raw data can be seen as consecutive durations of “marks” and “spaces” (Figure 3) in microseconds (us).

- Mark: a specific period of sending pulses
- Space: a specific period of sending nothing

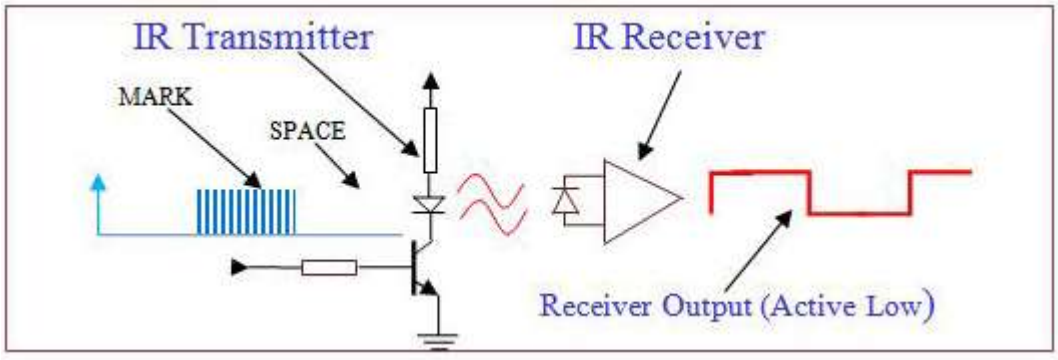


Figure 3: A typical IR transmission and reception setup implementation

For more details, please refer to SB-Projects’ topic of IR Remote Control Theory to learn the theory of IR remote controls operation and a collection of IR protocol descriptions. In this example, we are going to use NEC (Now Renesas, also known as Japanese Format) as the transmission protocol.

NEC Features

- 8-bit address and 8-bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- The carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

Modulation

NEC protocol uses Pulse Distance Encoding of the bits for data communication (Figure 4). A logical “1” is represented by total duration of 2250us, with 560us of “marks” and (2250-560) us of “spaces”. While logical ”0” is represented by total duration of 1120us, with 560us “marks” and (1120-560) us of “spaces”.

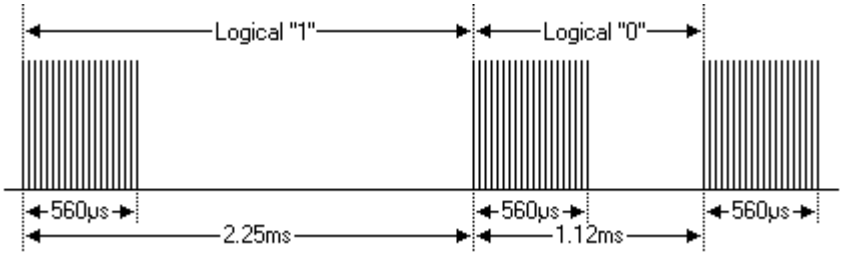


Figure 4: Modulation of NEC

Since a total number of 32-bit data together with the header and the end-bit will be transferred (Figure 5). If we separate the data in the time-frame (in us), there will be (2 + 32) x 2 + 1 = 69 “marks” / “spaces” to be transmitted (Figure 6), which forms the raw NEC data we would like to transmit in our Arduino “*.ino” file. This part of the code can be modified by users. Details of how to obtain raw data code for your remote devices, you may refer to Ken Shirriff’s blog, where it provides multiple libraries provided online.

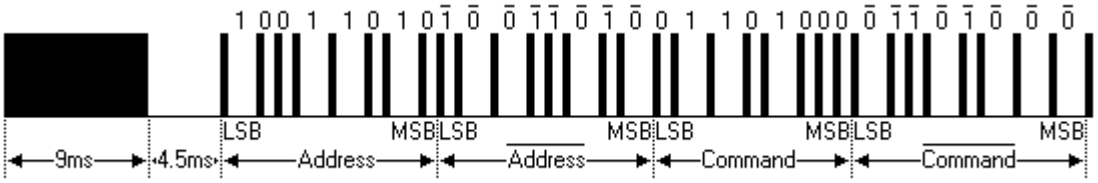


Figure 5: Sample of a Full NEC Data (in logic1 or 0)

```
9000, 4500,                                                                   // starting bit
560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560,       // address
560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~address
560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560,       // data
560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~data
560                                                                           // stoping bit
```

Figure 6: Sample of a Full NEC RAW Data (in us)

Figure 7 and 8 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722 board.

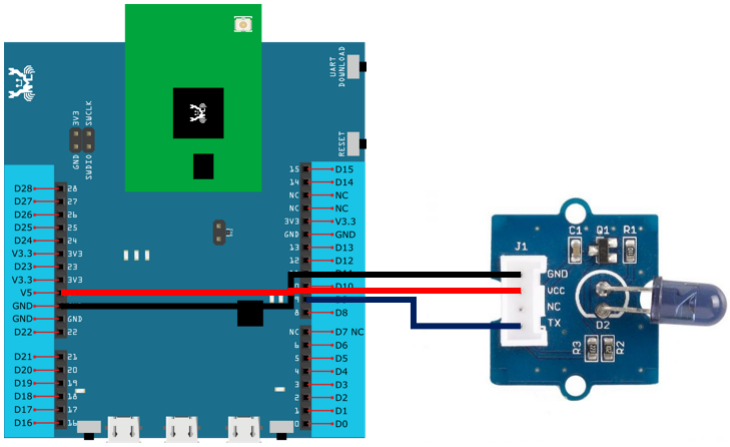


Figure 7: Pin configuration of IR Emitter and AMB21/AMB22

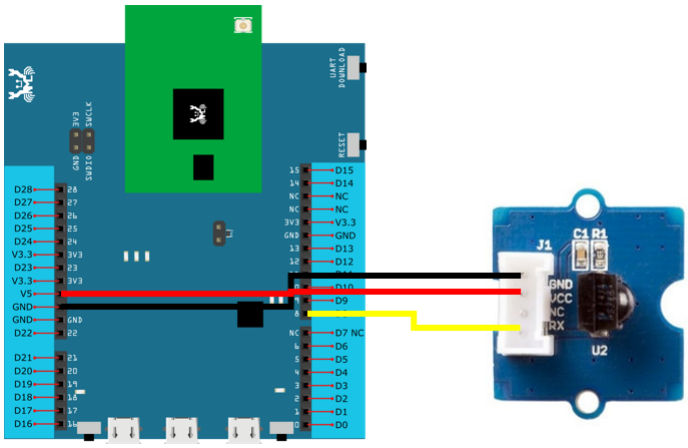


Figure 8: Pin configuration of the IR Receiver and Ameba RTL8722

Figure 9 and Figure 10 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8722DM MINI.

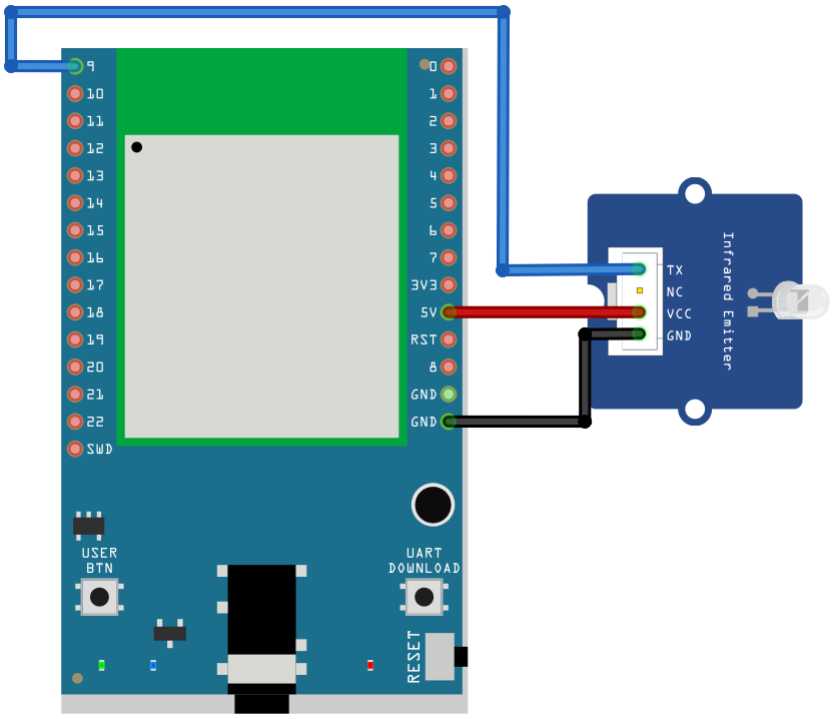


Figure 9: Pin configuration of IR Emitter and AMB23

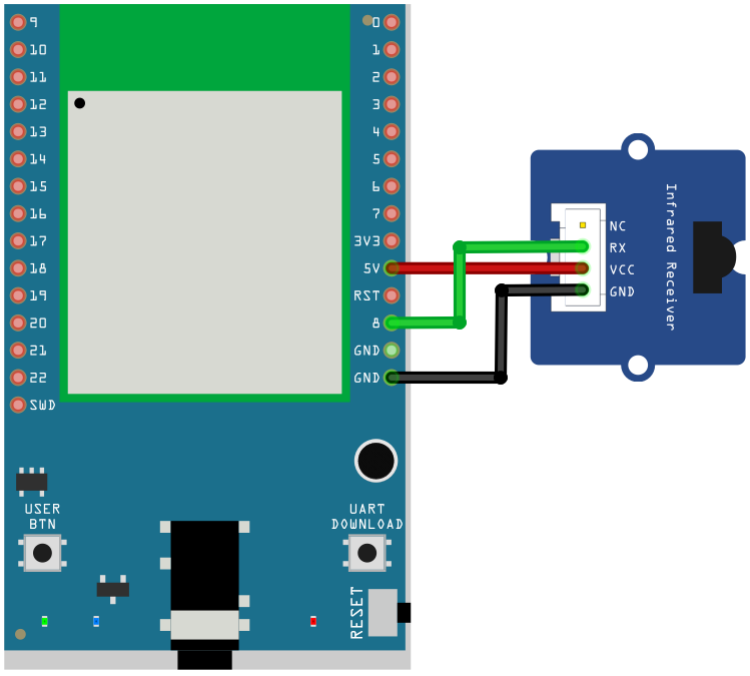


Figure 10: Pin configuration of the IR Receiver and AMB23

Figure 11 and Figure 12 shows the pin configuration of IR Emitter and Receiver with Ameba RTL8720DN (BW16).

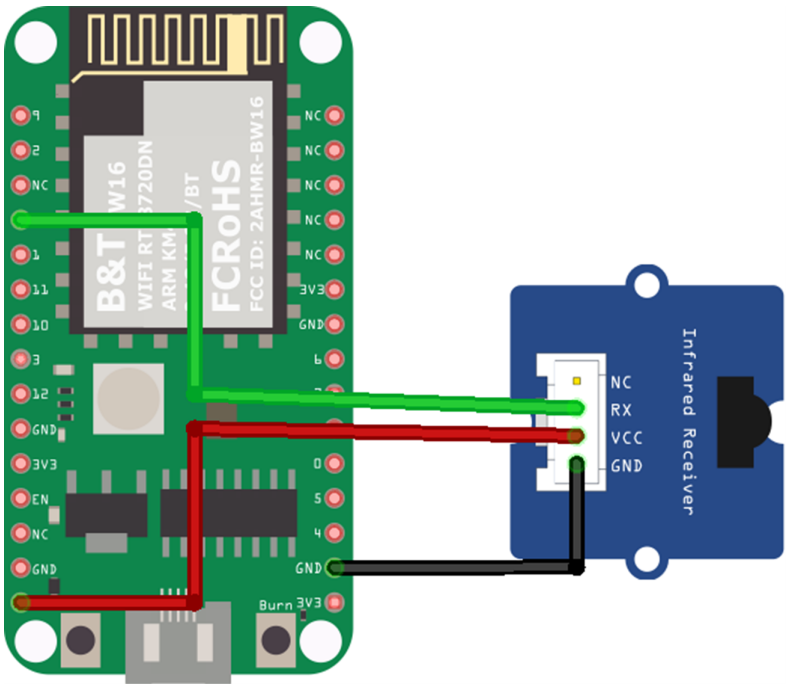


Figure 11: Pin configuration of IR Emitter and BW16

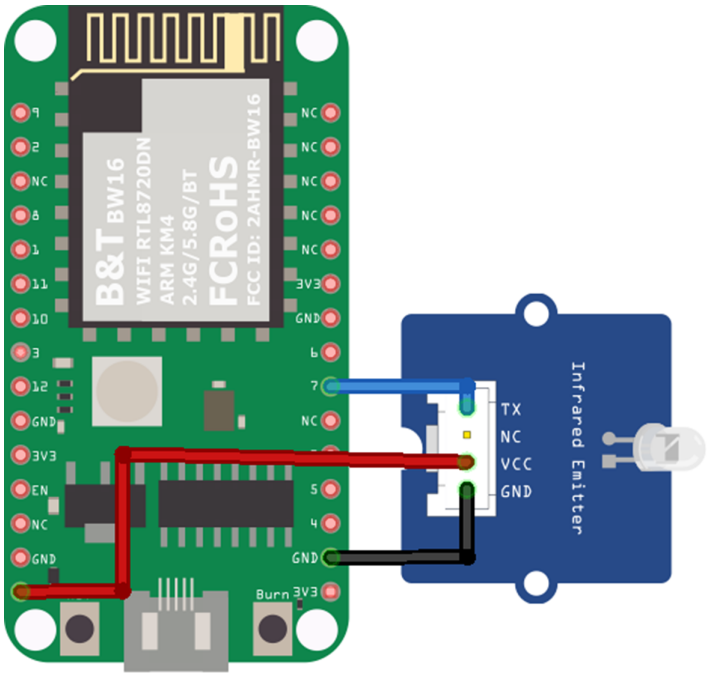


Figure 12: Pin configuration of IR Receiver and BW16

After the connection is being set up correctly, we will move to the coding part for this example. First, make sure the correct Ameba development board is selected in Arduino IDE: “Tools” -> “Board”.

Open the “IRSendRAW” example in “`File` -> “`Examples`” -> “`AmebaIRDevice`” -> “`IRSendRAW`” (Figure 11) and upload to 1st board connected with IR Emitter:

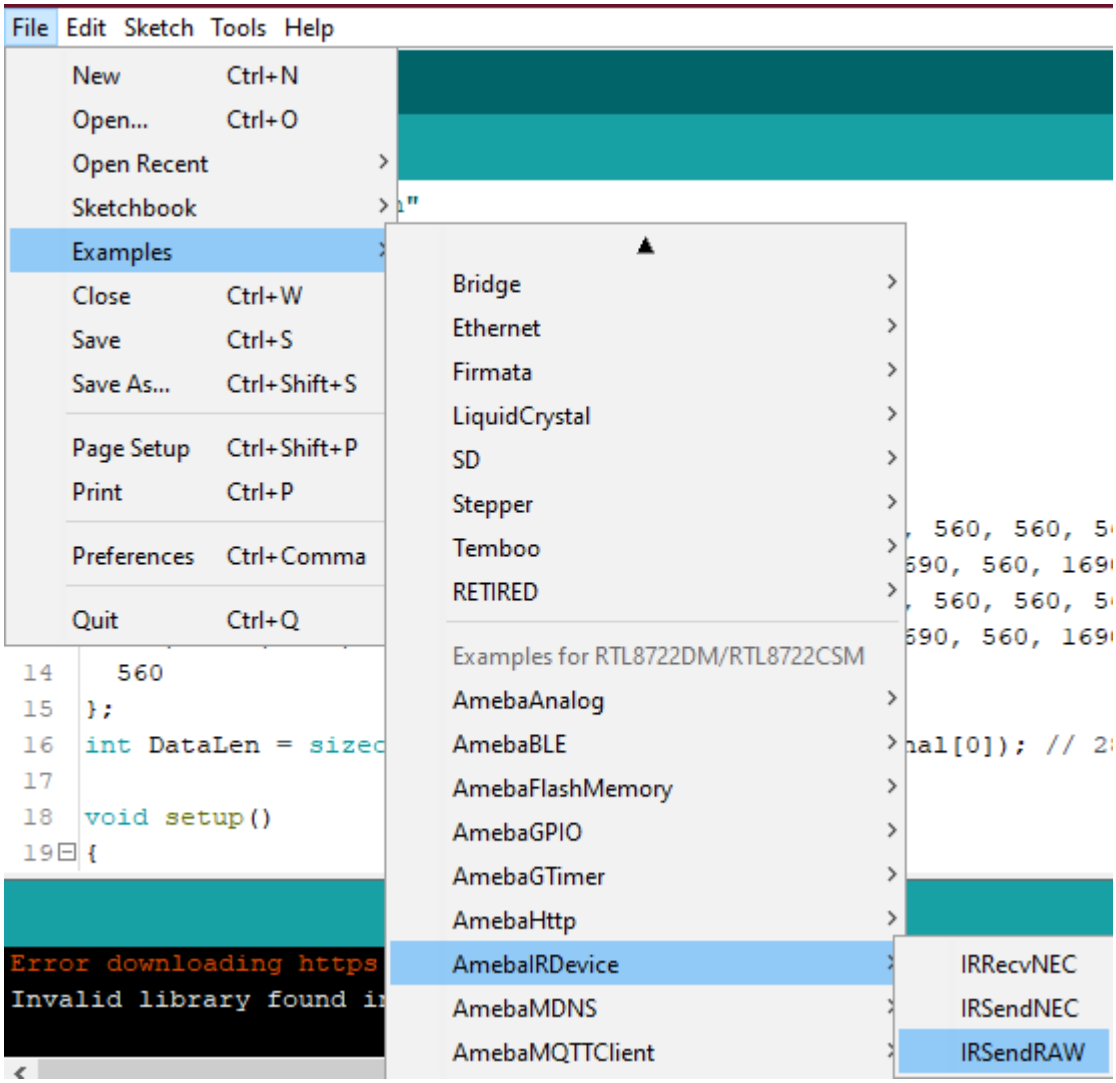


Figure 13: Example Location of IRSendRaw and IRRecvNEC

After successfully upload the sample code for IRSendRaw, you might need to upload the IRRecvNEC example for the 2nd board connected with IR Receiver from "`File`" -> "`Examples`" -> "`AmebaIRDevice`" -> "`IRRecvNEC`".

After opening the serial monitor on the IR Receiver side and press the reset buttons on two boards, the data "48" will be received every 3 seconds (due to the delays () function, not compulsory to wait). After decoding the signal from the receiving Pin D8 and transmitting Pin D9 with Logic Analyser and Pulse View (Figure 10), the result is also shown as "48" after decoding the receiving data with IR NEC Protocol.

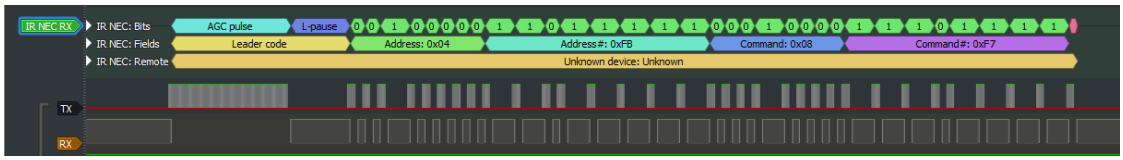


Figure 14: Pulse View results from sending and receiving pin

Code Reference

[1] Seeed Official website for Grove – Infrared Receiver
https://wiki.seeedstudio.com/Grove-Infrared_Receiver/

[2] Seed Official website for Grove – Infrared Emitter
https://wiki.seeedstudio.com/Grove-Infrared_Emitter/

[3] Ken SHirriff's blog on A Multi-Protocol Infrared Remote Library for the Arduino
http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

[4] SB-Projects: IR Remote Control Project
https://www.sbprojects.net/knowledge/ir/index.php

Power Save - Deep Sleep Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

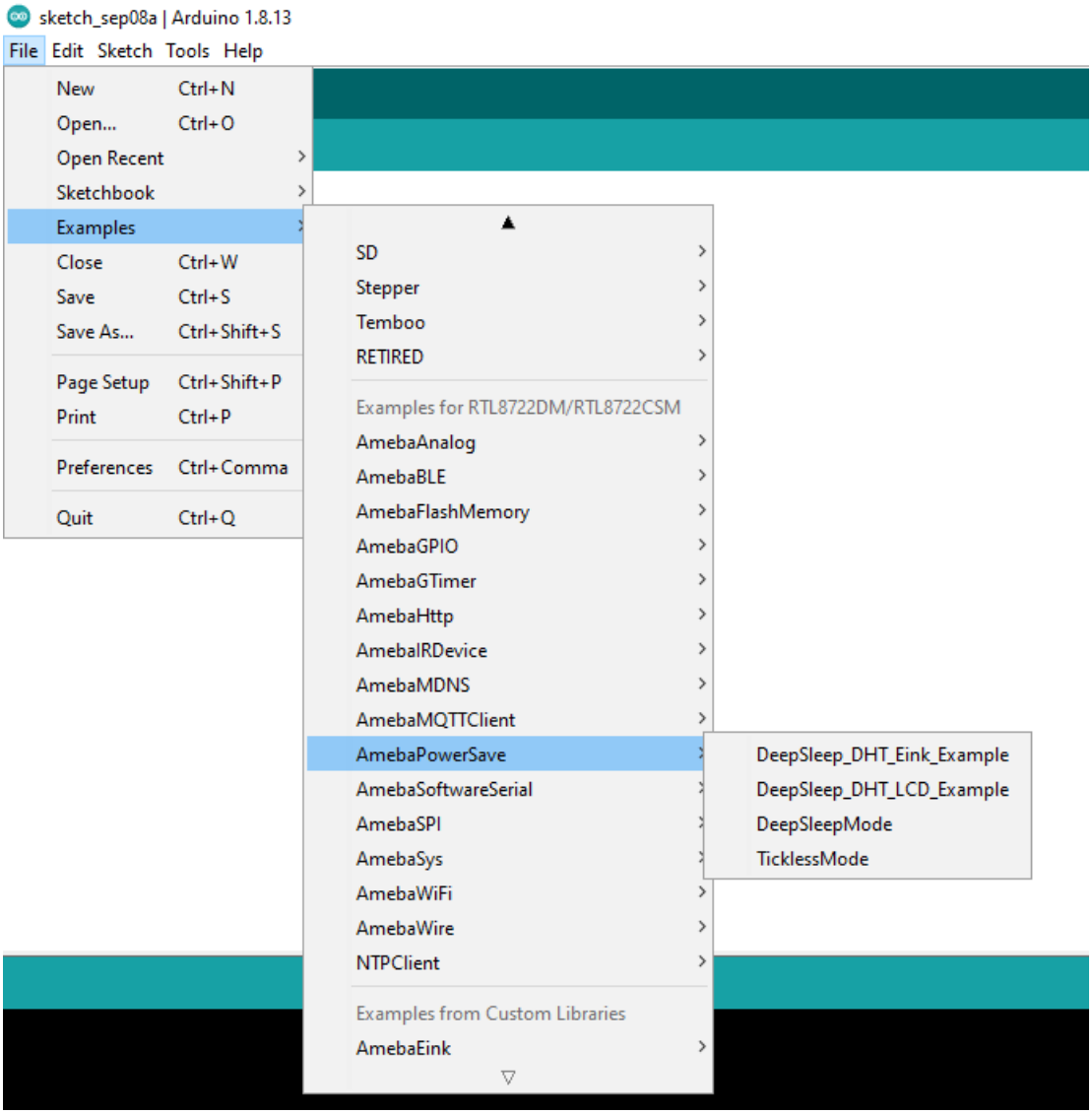
Example

Introduction

Ameba-D supports 2 low power modes which are deepsleep mode and sleep mode. Deep Sleep mode turns off more power domain than sleep mode. The power consumption of Deep Sleep mode is around 7μA to 8μA as compared to normal state which is around 22mA. This example describes how to enter Deep Sleep mode and configure the wakeup source

Procedure

Open "`File`" -> "`Examples`" -> "`AmebaPowerSave`" -> "`DeepSleepMode`"



Set condition values as picture below.
"DS_WAKEUP_SOURCE" is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON Timer(SET_DS_AON_TIMER_WAKEUP);
AON GPIO pins(SET_AONWAKEPIN_WAKEUP);
RTC Timer(SET_DS_RTC_WAKEUP);
```

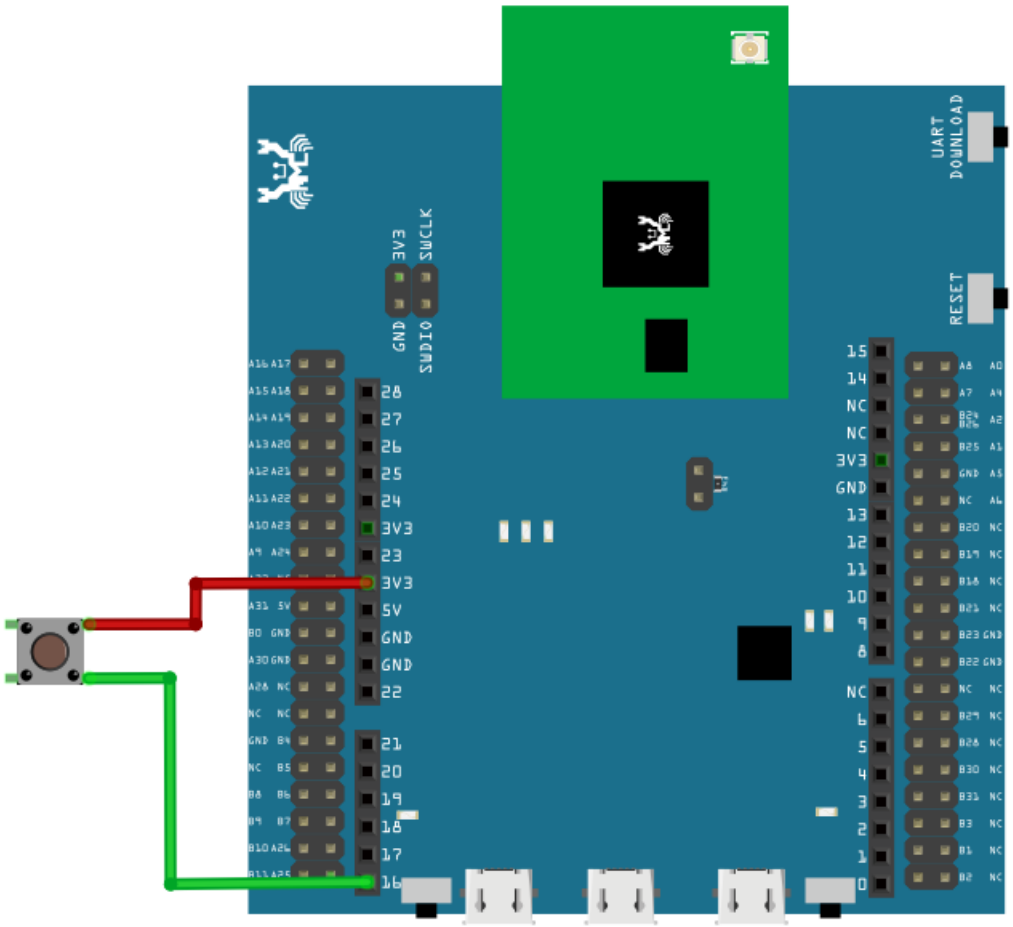
Using AON Timer as wakeup source

AON Timer can be set from 0 to 32760000ms range by `AON_TIMER_SLEEP_DURATION`

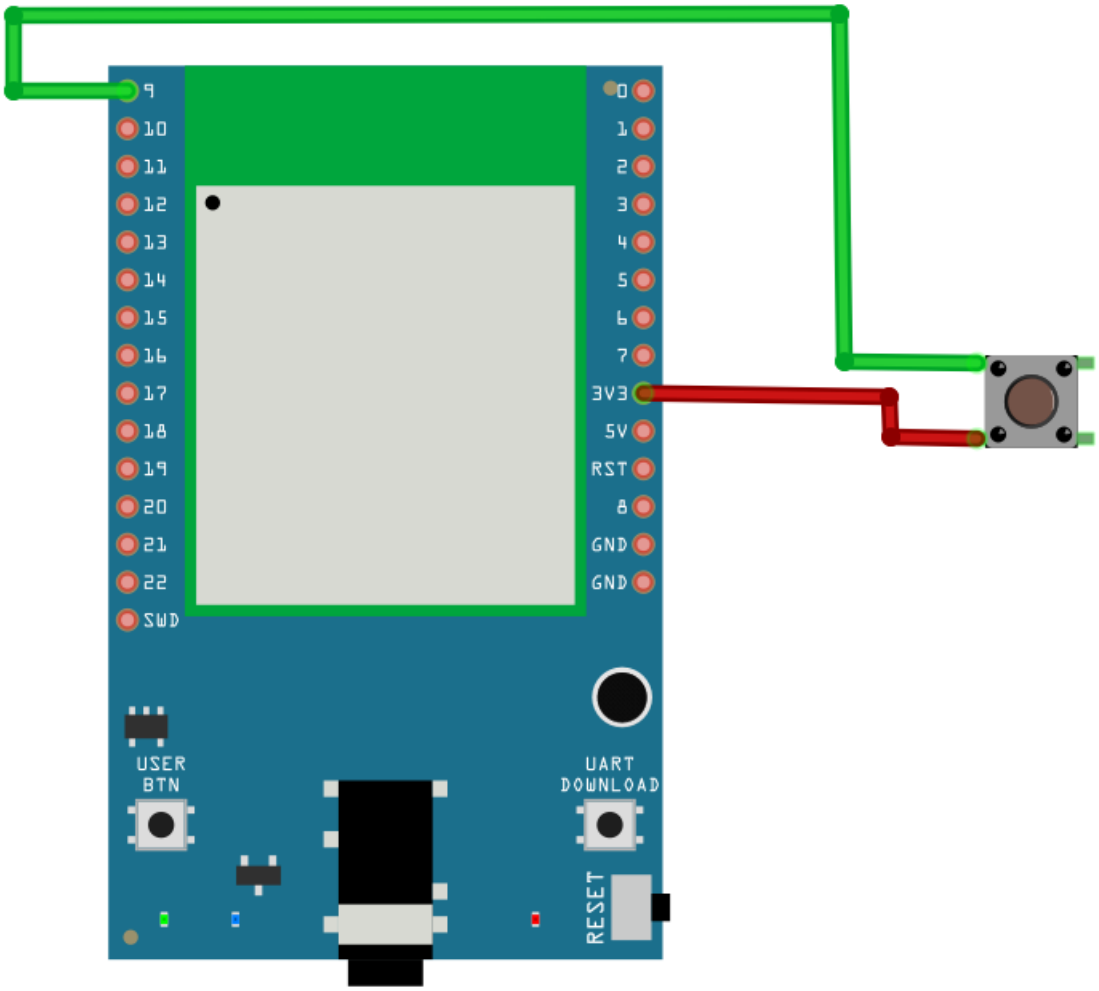
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16),

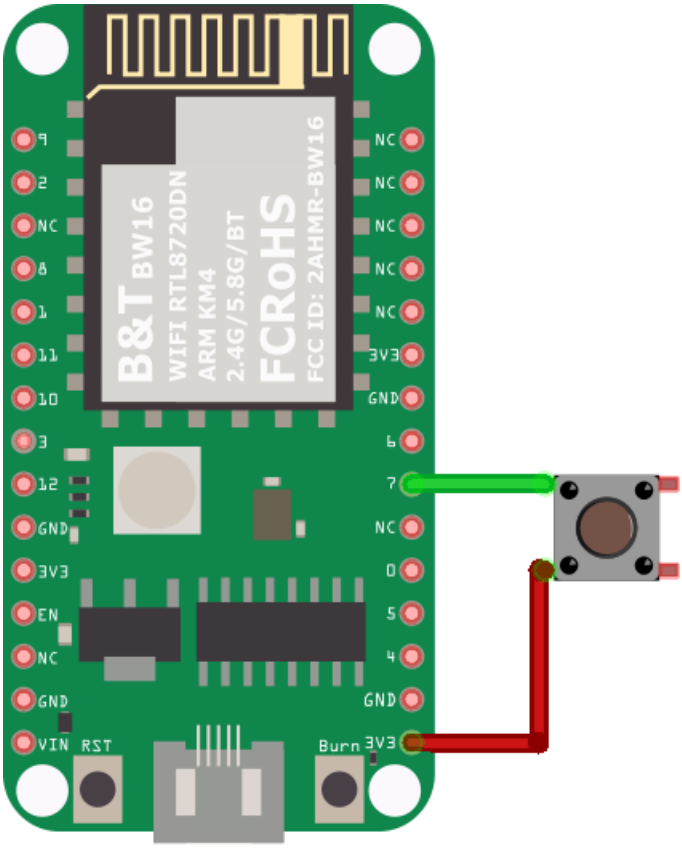
GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).



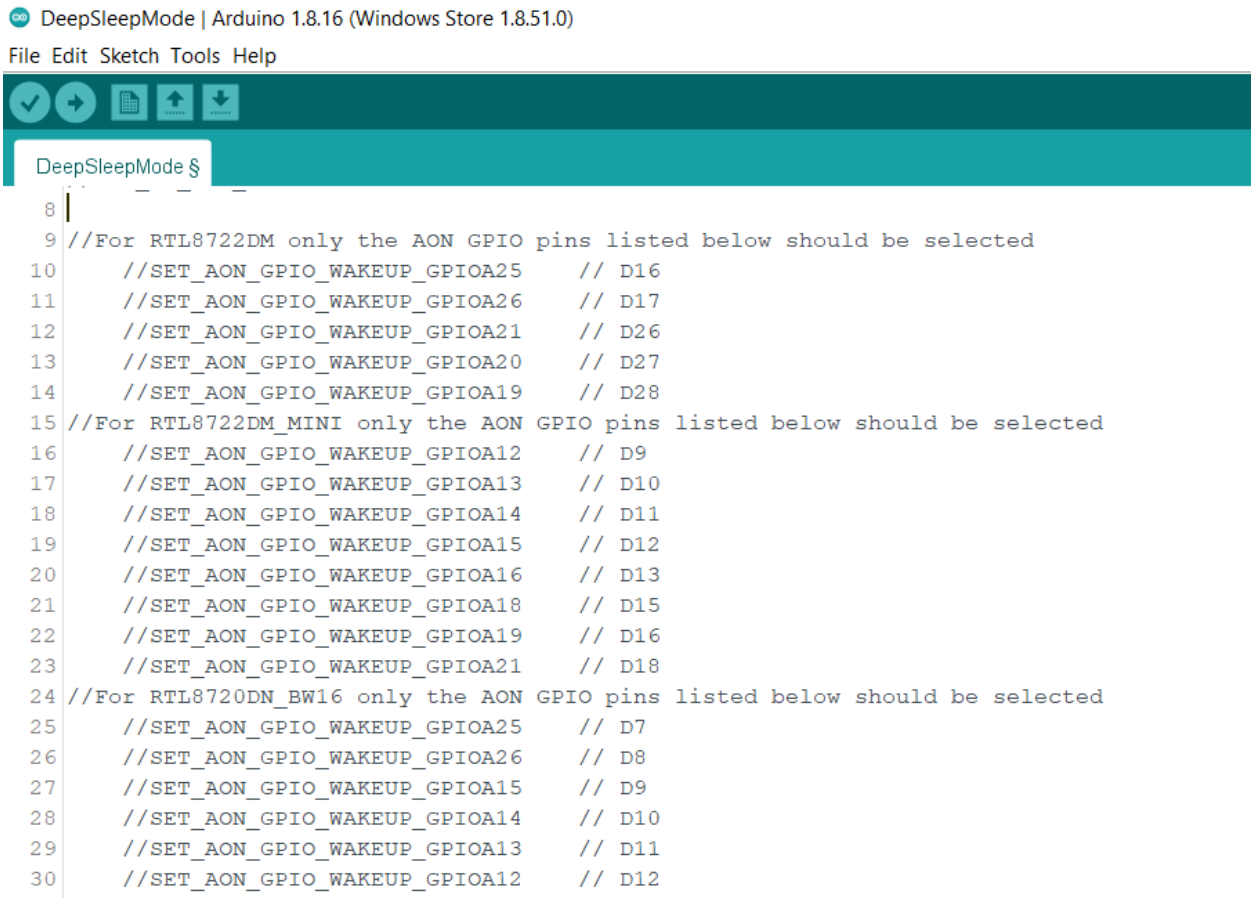
For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

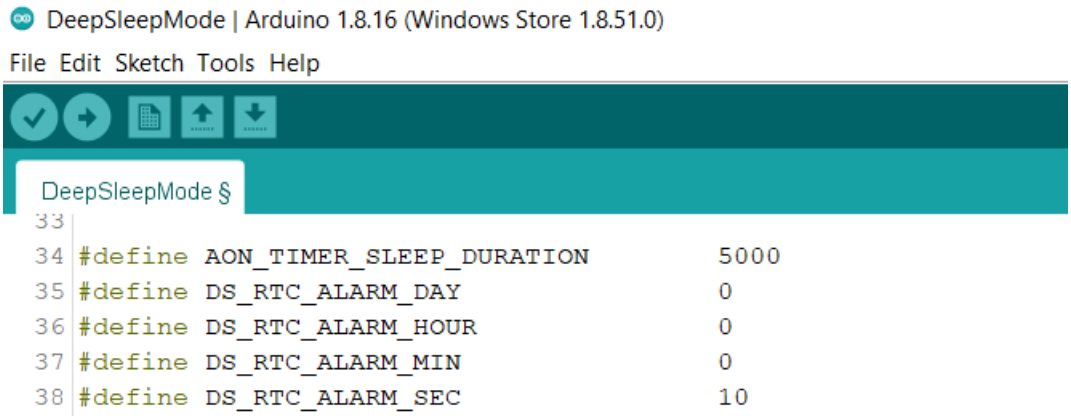


These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below



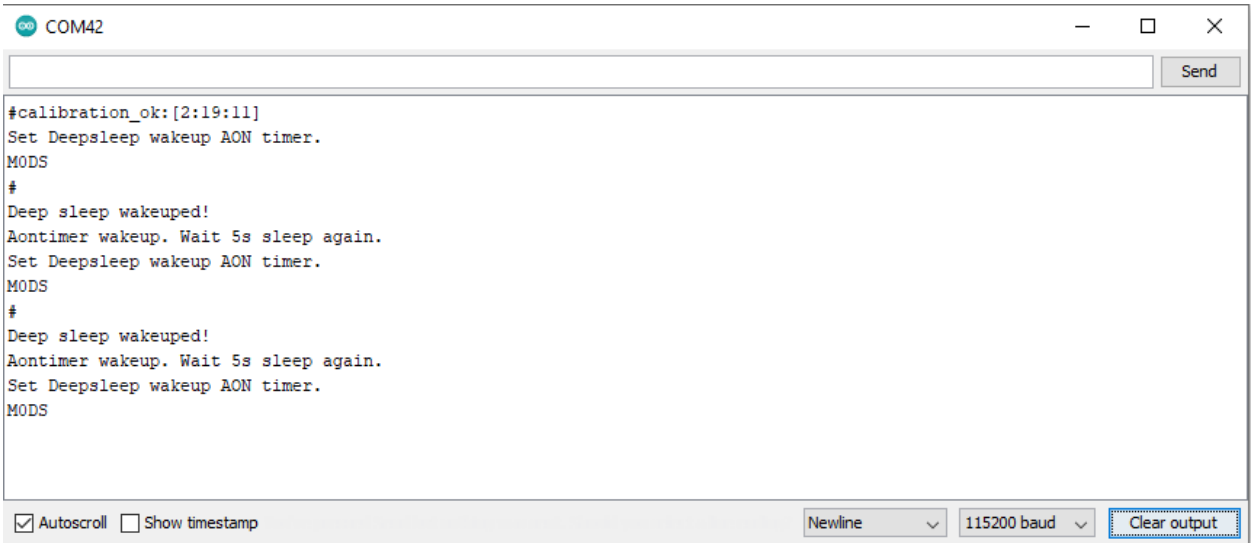
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

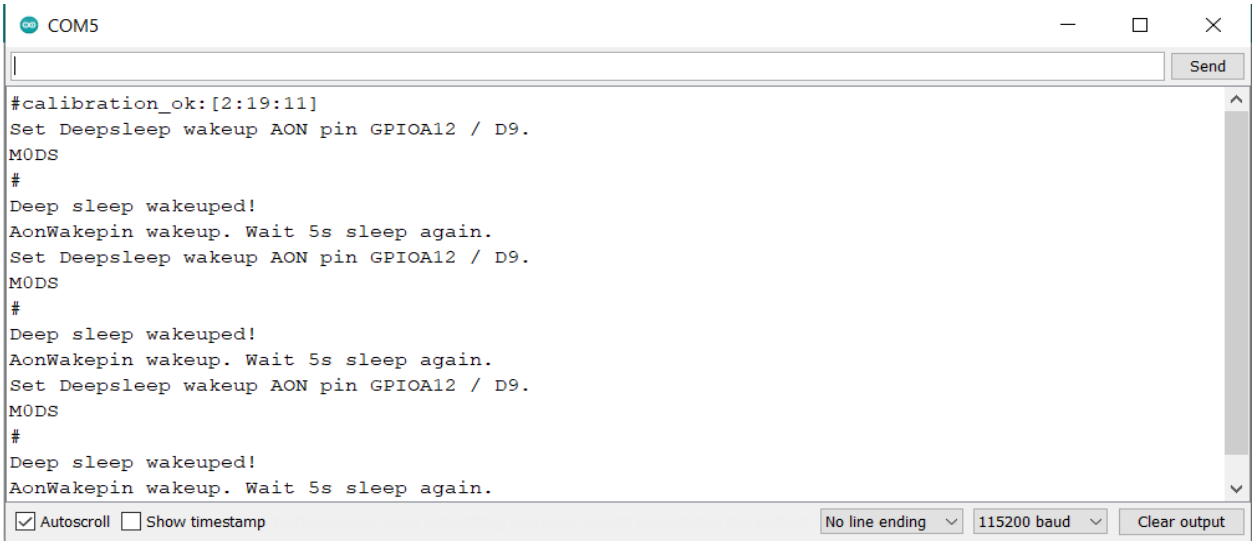


When all the condition values are set, the system will run and switch between normal and deep sleep mode which is controlled by the wakeup source. The serial monitor will display the switching log as shown below.

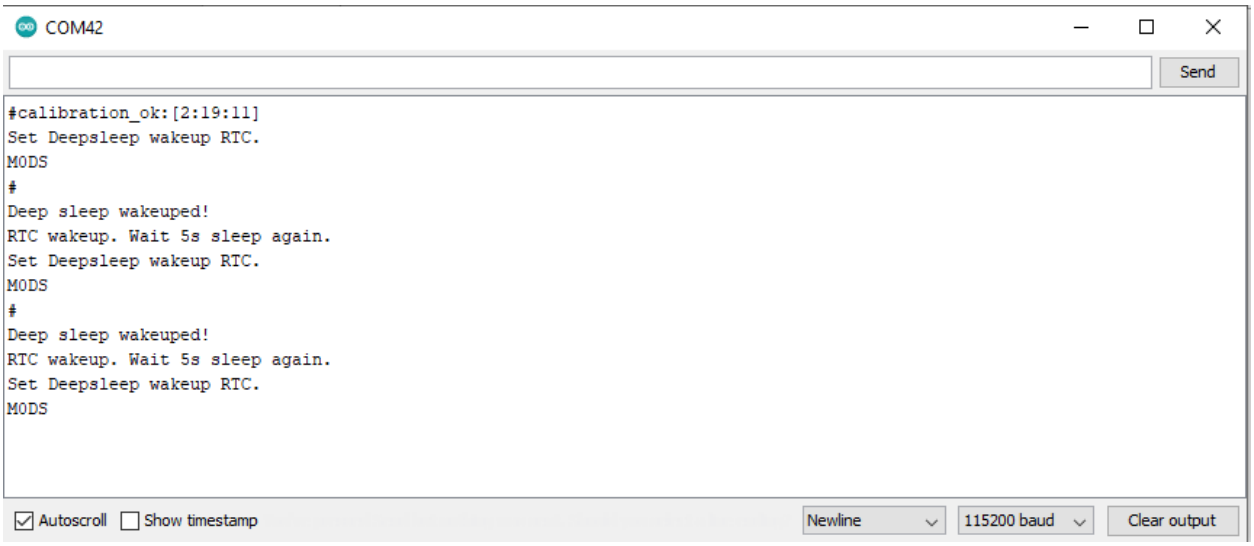
AON Timer



AON GPIO Pin



RTC Timer



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and E-paper

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

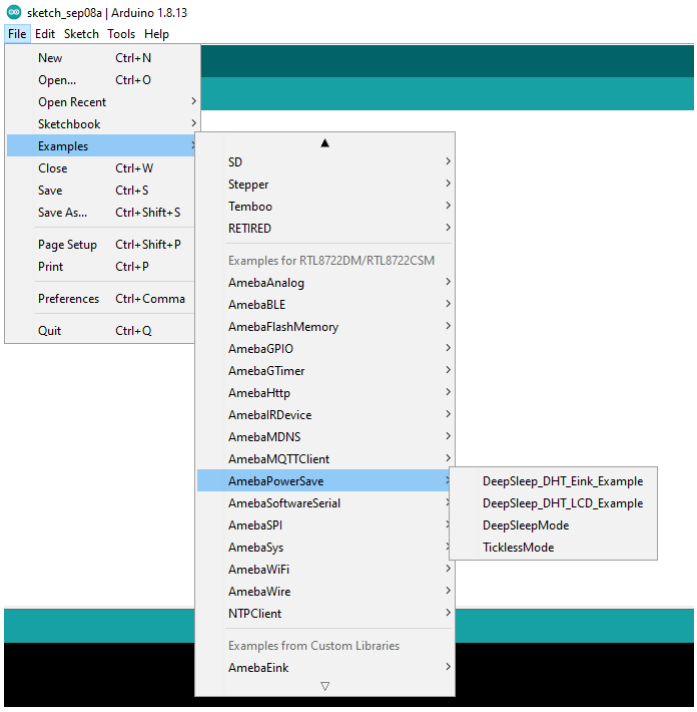
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between “working” and “sleep”(power save).Using DHT sensor to read data and display on Eink screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Download the Eink zip library, AmebaEink.zip, at https://github.com/ambiot/ambd_arduino/tree/master/Arduino_zip_libraries. Then install the AmebaEink.zip.
Open “`File” -> “Examples” -> “AmebaPowerSave” -> “DeepSleep_DHT_Eink_Example`”



Set condition values as picture below.
`DS_WAKEUP_SOURCE` is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

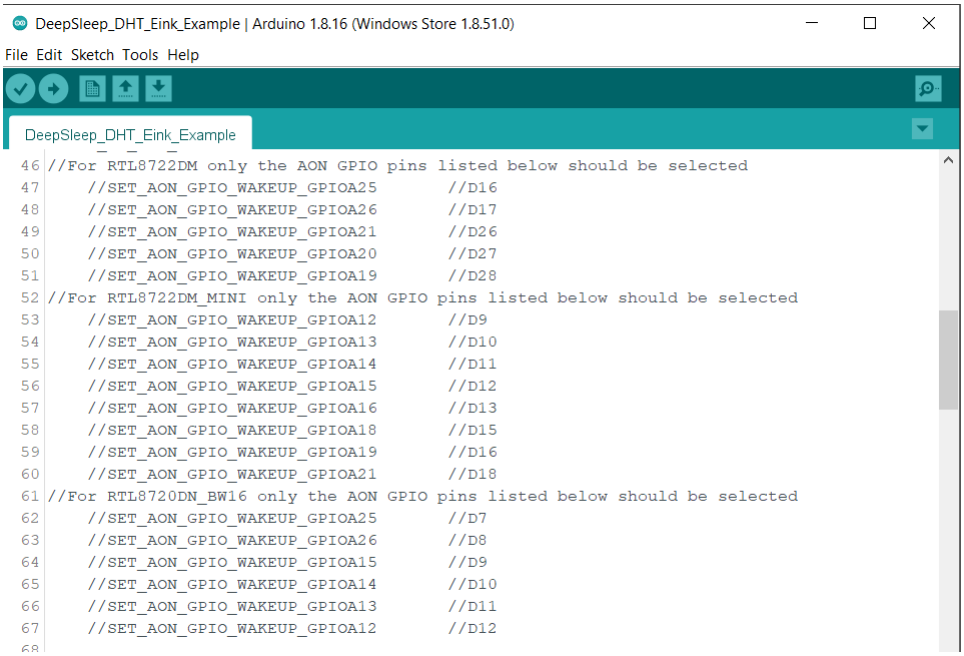
Using AON Timer as wakeup source
AON timer can be set from 0 to 32760000 range (unit ms) by `AON_TIMER_SLEEP_DURATION`

Using AON GPIO pins as wakeup source
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

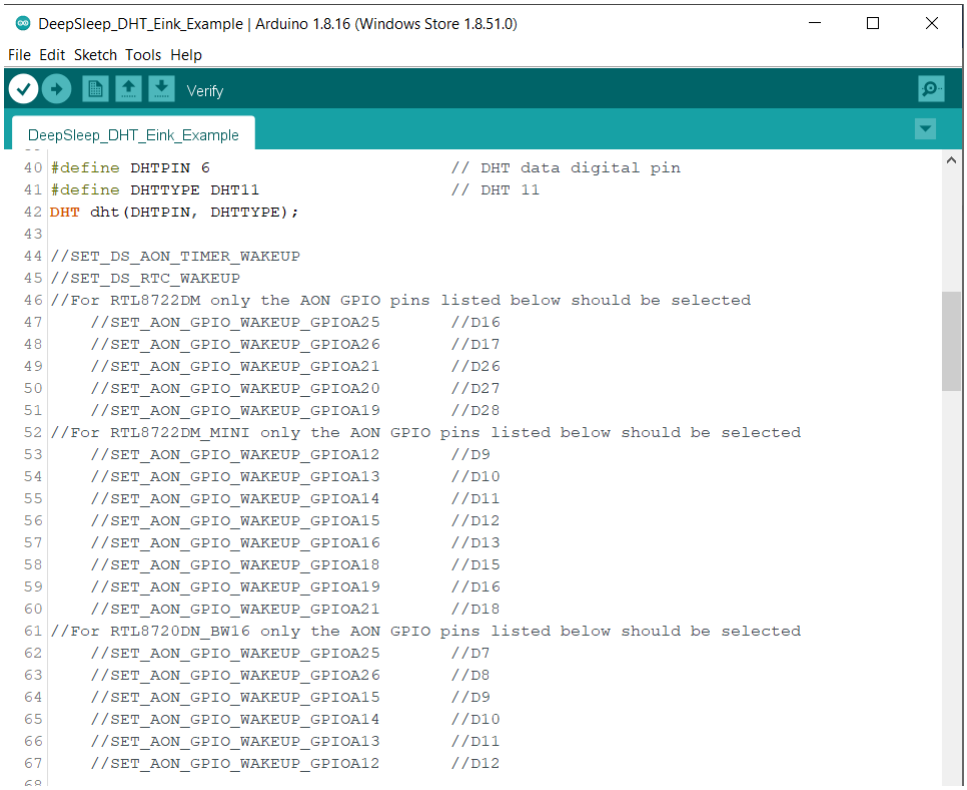
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. Eink screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Deep Sleep for DHT and LCD

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- DHT11 or DHT22 or DHT21 x 1
- LCD I2C screen x 1

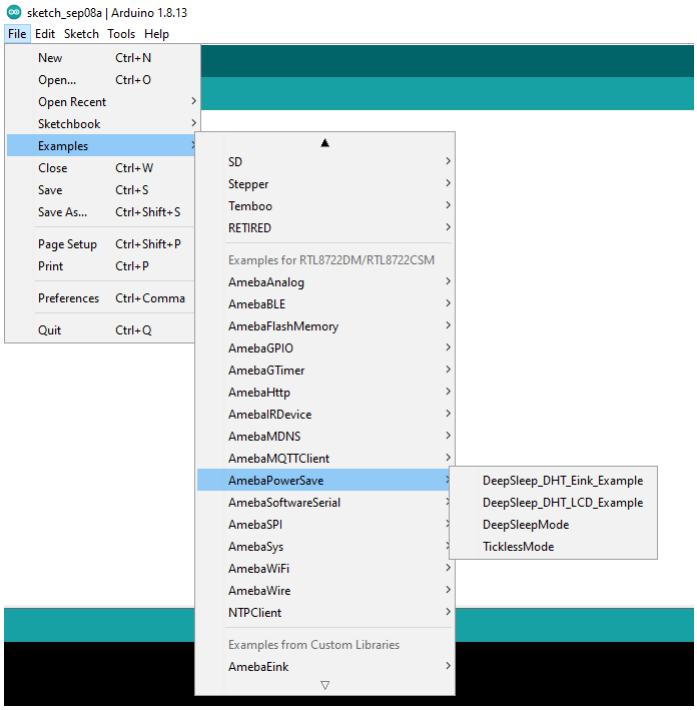
Example

Introduction

Ameba-D supports low power modes which are deepsleep mode. Deepsleep mode turns off most of the system power domain. The power consumptions of core module in DeepSleep Mode is around 7uA to 8uA compare to normal state around 22mA. This example gives demo of system switch between “working” and “sleep”(power save).Using DHT sensor to read data and display on LCD screen when system is awake. After 5 seconds system auto enter DeepSleep Mode for power save. System will wake up by wakeup source.(Aon timer, Aon Pins or RTC timer).

Procedure

Open “File” -> “Examples” -> “AmebaPowerSave” -> “DeepSleep_DHT_LCD_Example”



Set condition values as picture below.

`DS_WAKEUP_SOURCE` is used to set the wake-up source, user can chose 3 wake up sources now,

```
AON timer (SET_DS_AON_TIMER_WAKEUP);
AON pins (SET_AON_WAKEPIN_WAKEUP);
RTC timer (SET_DS_RTC_WAKEUP);
```

Using AON Timer as wakeup source

AON timer can be set from 0 to 32760000 range (unit ms) by `AON_TIMER_SLEEP_DURATION`

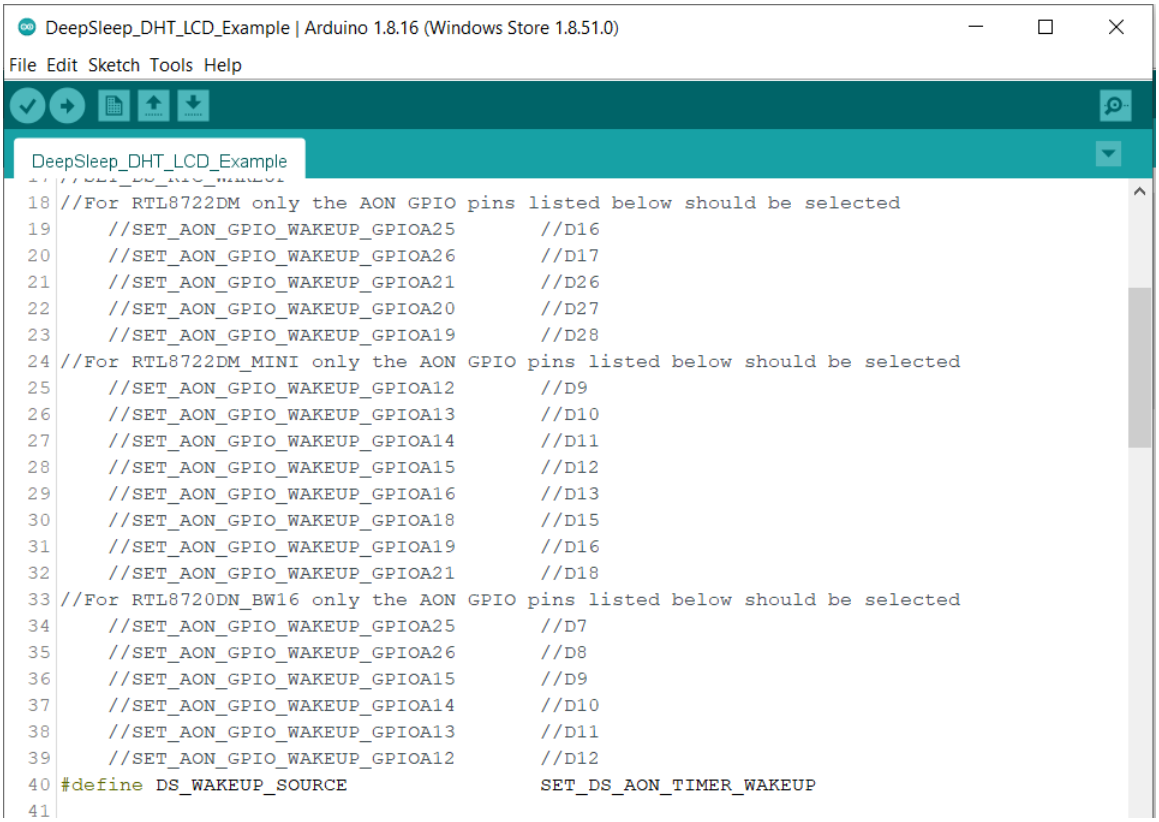
Using AON GPIO pins as wakeup source

For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).

For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).

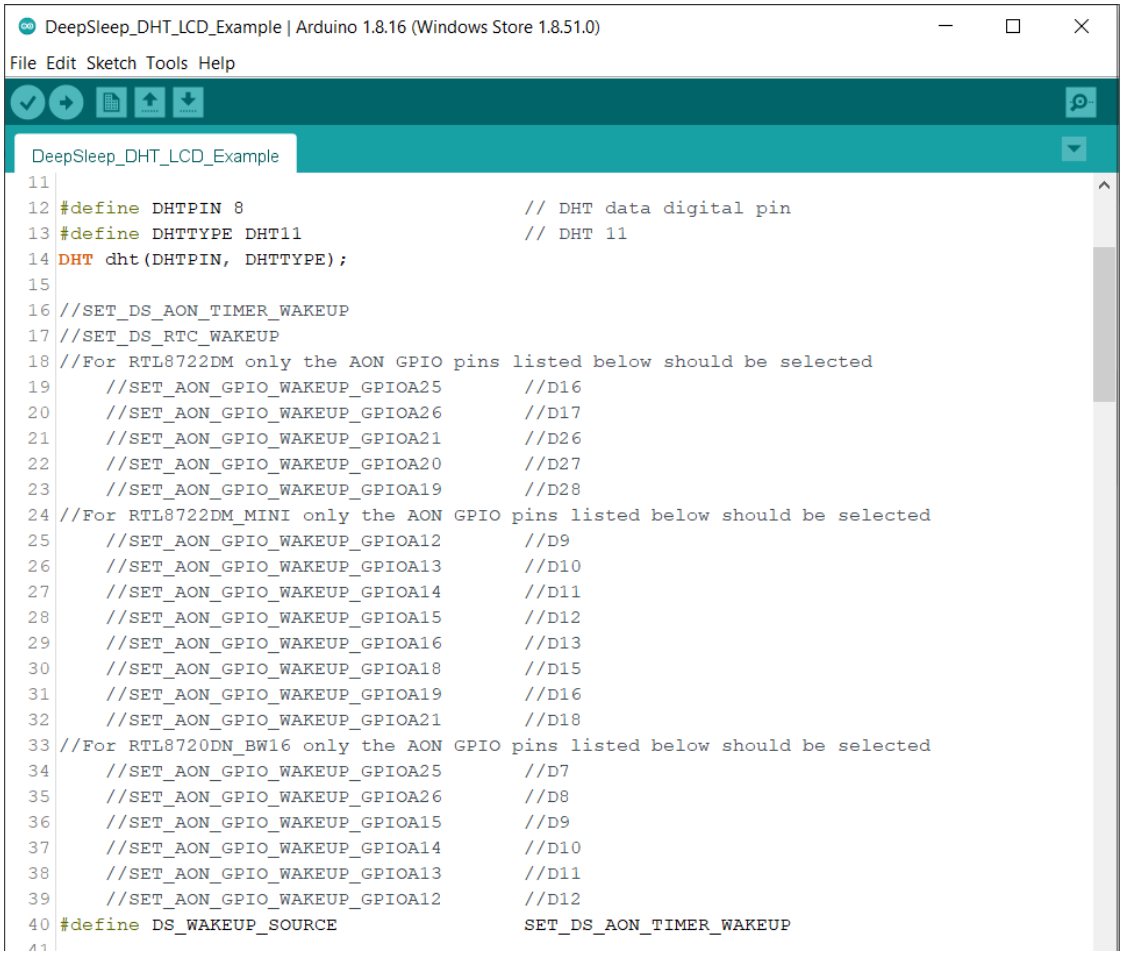
These AON pins can be set by using `SET_AON_GPIO_WAKEUP_GPIOA25` or the pin that you want to use as shown in the picture below.



Using RTC Timer as wakeup source

RTC timer wakeup system is by setting alarm. The alarm has 4 values, day, hour, min and sec. All 4 values can be set by `DS_RTC_ALARM_DAY`, `DS_RTC_ALARM_HOUR`, `DS_RTC_ALARM_MIN`, and `DS_RTC_ALARM_SEC`

`DHTPIN` is used to set DHT sensor data pin. User can choose any GPIO pins.
`DHTTYPE` is used to set DHT sensor type. (DHT11, DHT22 and DHT33)



When finished the condition values setting, system will run and switch between normal working mode and deepsleep mode controlled by wakeup source. LCD screen will display the temperature and humidity data measured from DHT sensor when system is awake.

Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

Power Save - Tickless Mode

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

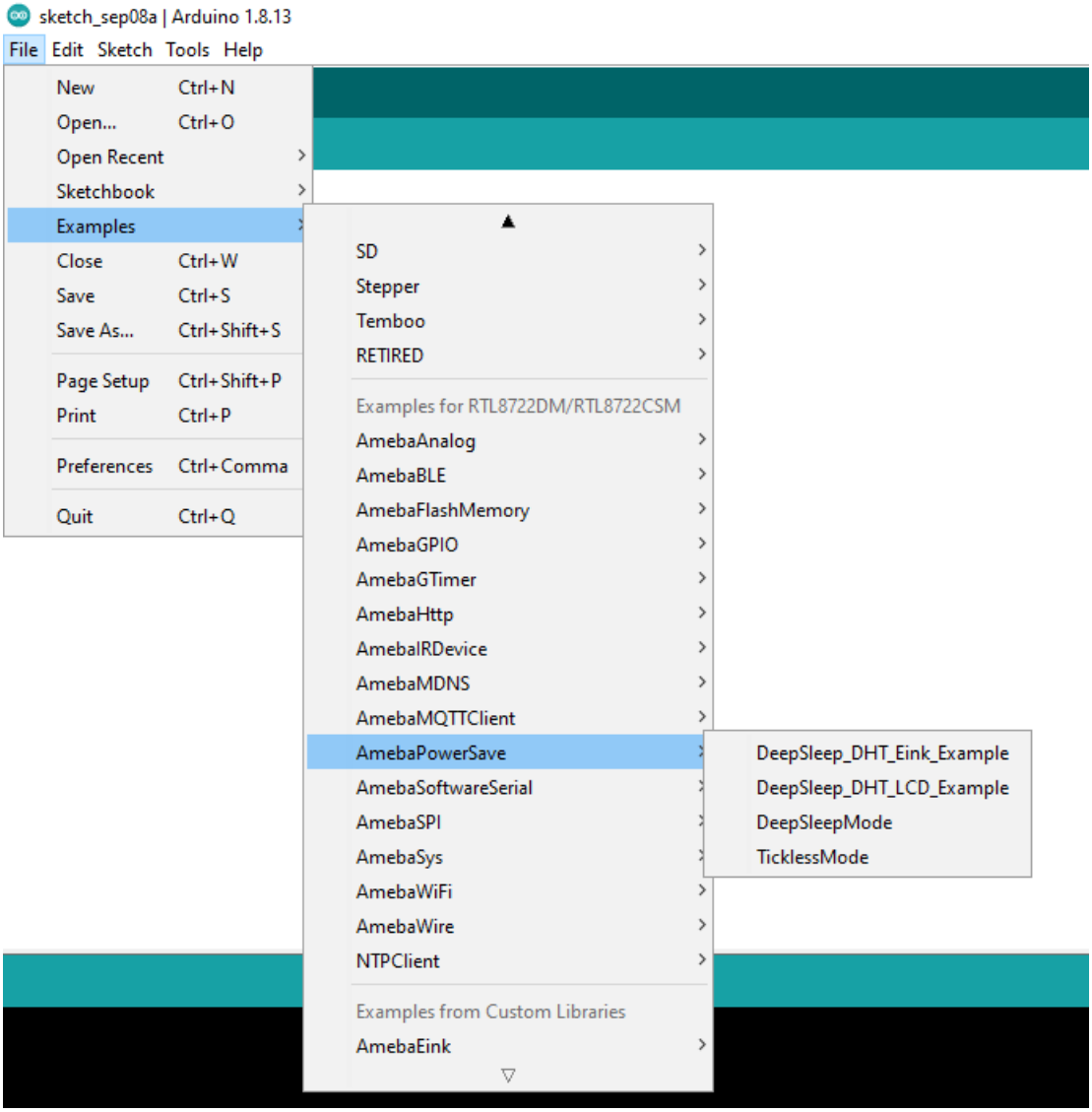
Example

Introduction

Ameba-D supports two low power modes which are deepsleep mode and sleep mode. The power consumptions of Tickless Sleep Mode is around 28uA to 30uA compare to normal state around 15mA. This example describes how to use freertos tickless with uart interruptable interface.

Procedure

Open “File” -> “Examples” -> “AmebaPowerSave” -> “TicklessMode”



Set condition values as picture below.

“TL_WAKEUP_SOURCE” is used to set the wake-up source, user can chose 3 wake up sources now,

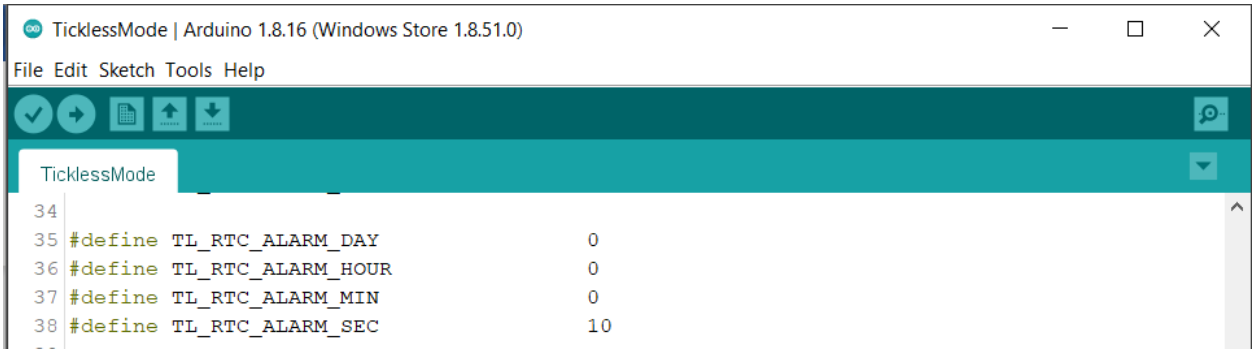
```
LOGUART(SET_TL_UART_WAKEUP);
RTC Timer(SET_TL_RTC_WAKEUP);
AON pins(SET_AON_WAKEPIN_WAKEUP);
```

Using LOGUART as wakeup source

When the LOGUART is selected as the wakeup source, the “TL_Suspend_function” will enter sleep mode. And then it is kept alive for 13s then enter sleep mode. To wakeup, press enter.

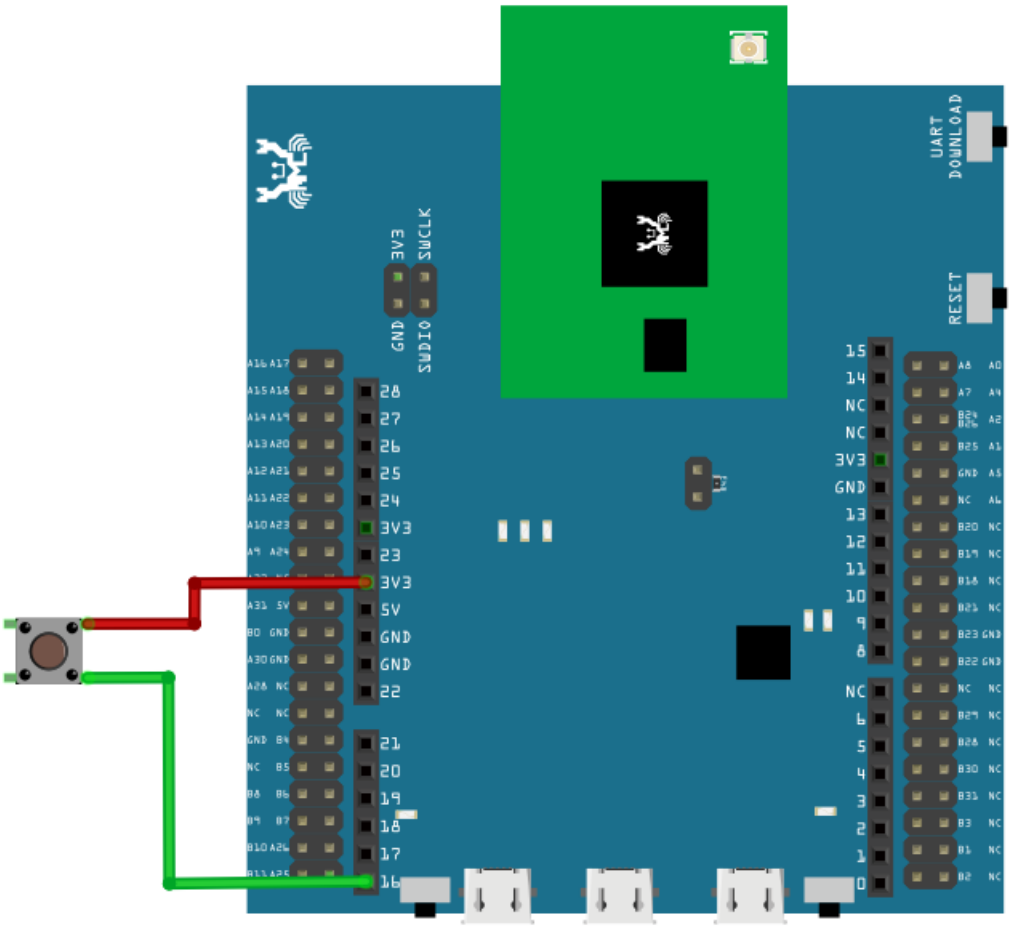
Using RTC Timer as wakeup source

RTC Timer wakeup system is by setting alarm. The alarm has 4 values to be set, day, hour, min and sec. All 4 values can be set by DS_RTC_ALARM_DAY, DS_RTC_ALARM_HOUR, DS_RTC_ALARM_MIN, and DS_RTC_ALARM_SEC.

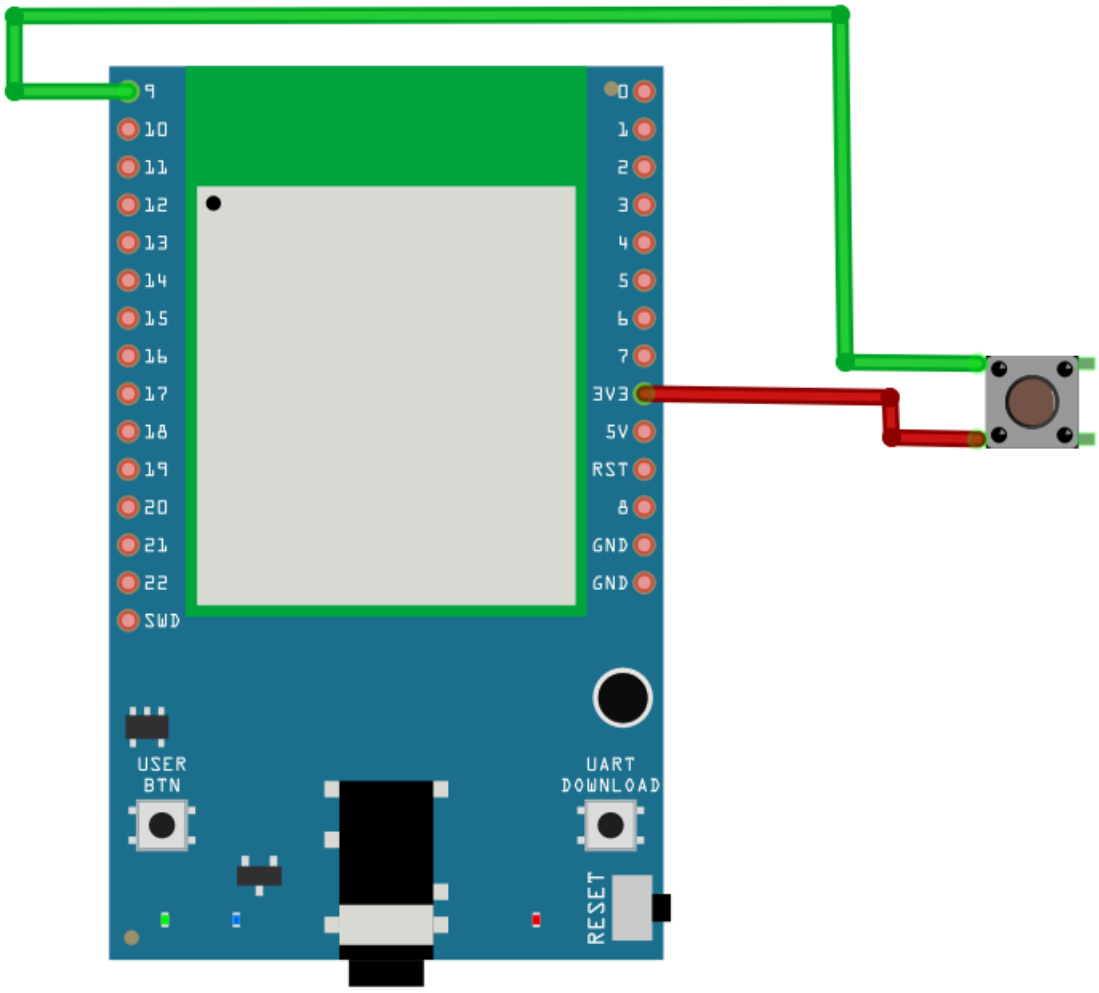


Using AON GPIO pins as wakeup source

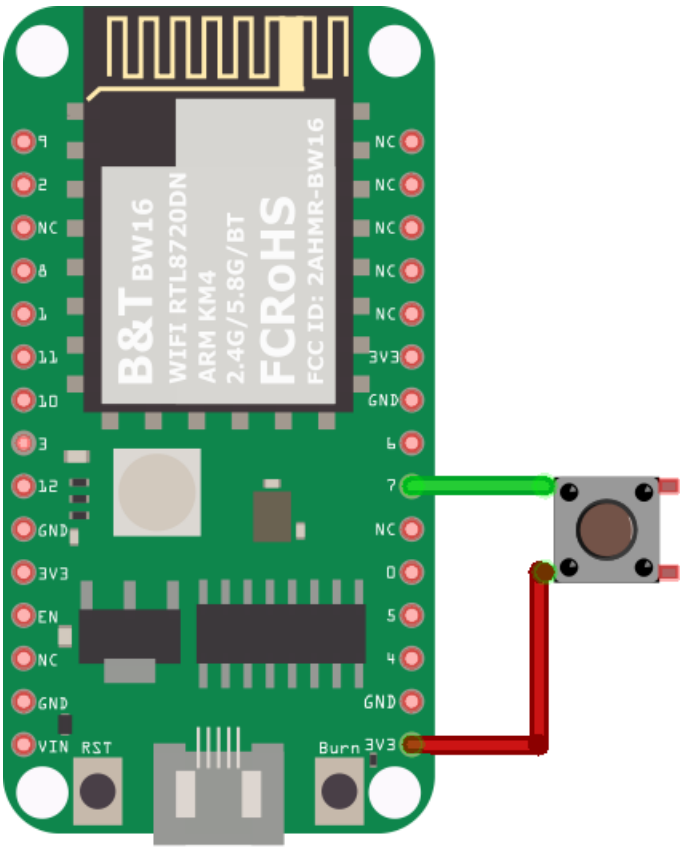
For AMB21, there are 5 pins that can be set as AON pins and active high for wakeup, GPIOA25(D16), GPIOA26(D17), GPIOA21(D26), GPIOA20(D27), GPIOA(D28).

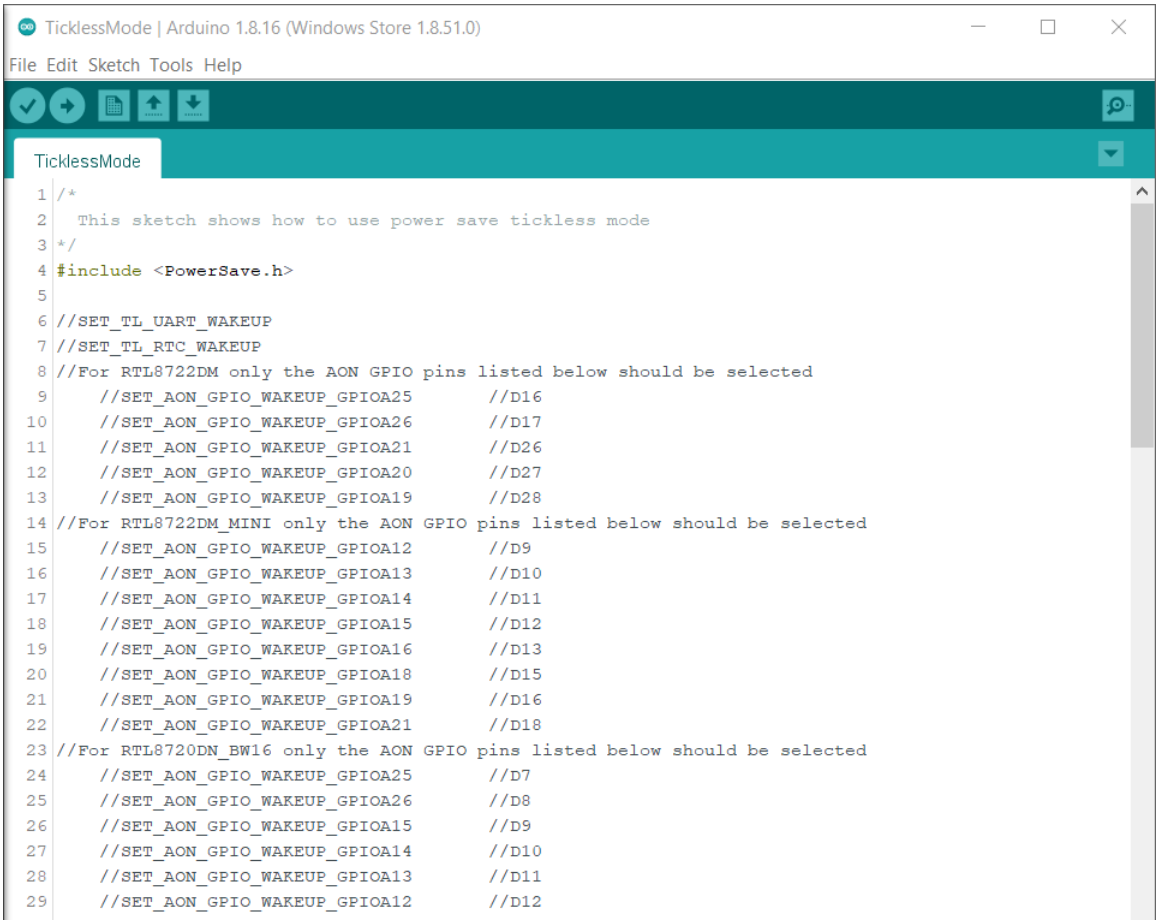


For AMB23, there are 8 pins that can be set as AON pins and active high for wakeup, GPIOA12(D9), GPIOA13(D10), GPIOA14(D11), GPIOA15(D12), GPIOA16(D13), GPIOA18(D15), GPIOA19(D16), GPIOA21(D18).



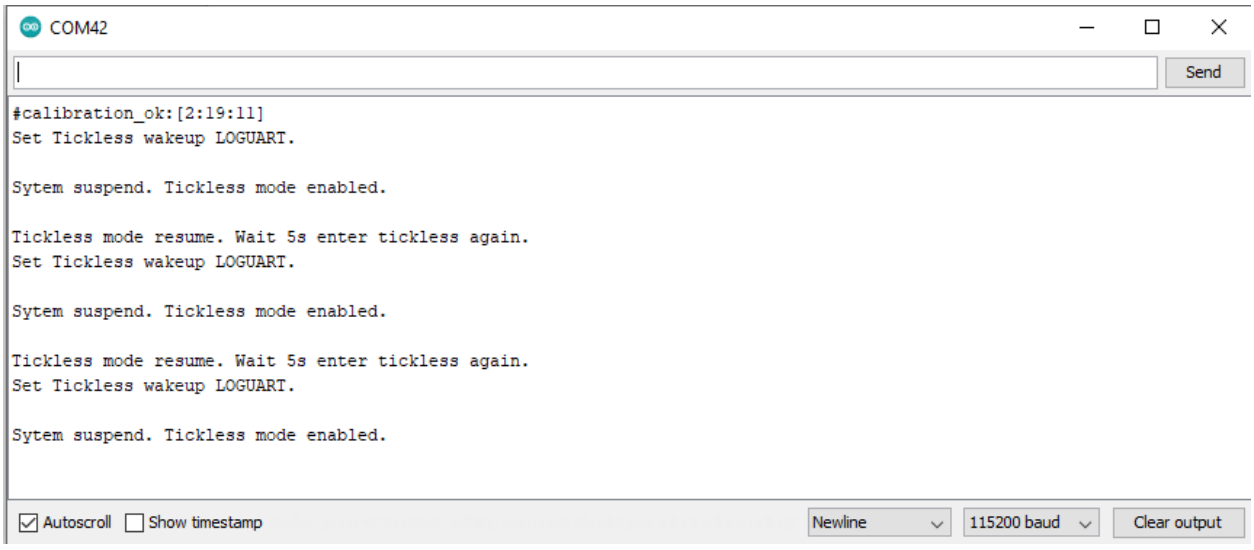
For BW16, there is only 6 pins that can be set as AON pin and active high for wakeup, GPIOA_25 (D7), GPIOA_26 (D8), GPIOA_15 (D9), GPIOA_14 (D10), GPIOA_13 (D11), GPIOA_12 (D12).


B&T BW16
WIFI RTL8720DN
ARM KM4
2.4G/5.8G/BT
FCC ID: 2AHMR-BW16
NC
NC
NC
NC
NC
3V3
GND
6
7
NC
0
5
4
GND
3V3
Burn
9
2
NC
8
1
11
10
3
12
GND
3V3
EN
NC
GND
VIN
RST

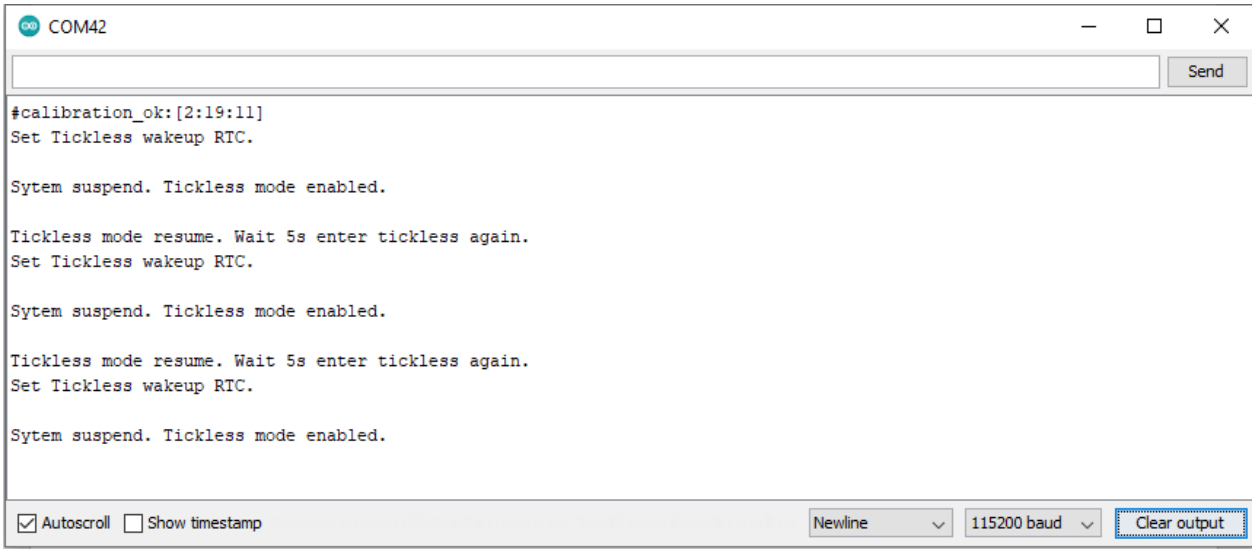


TL_SYSACTIVE_TIME is for setting time duration of the system to keep alive. (Unit ms)

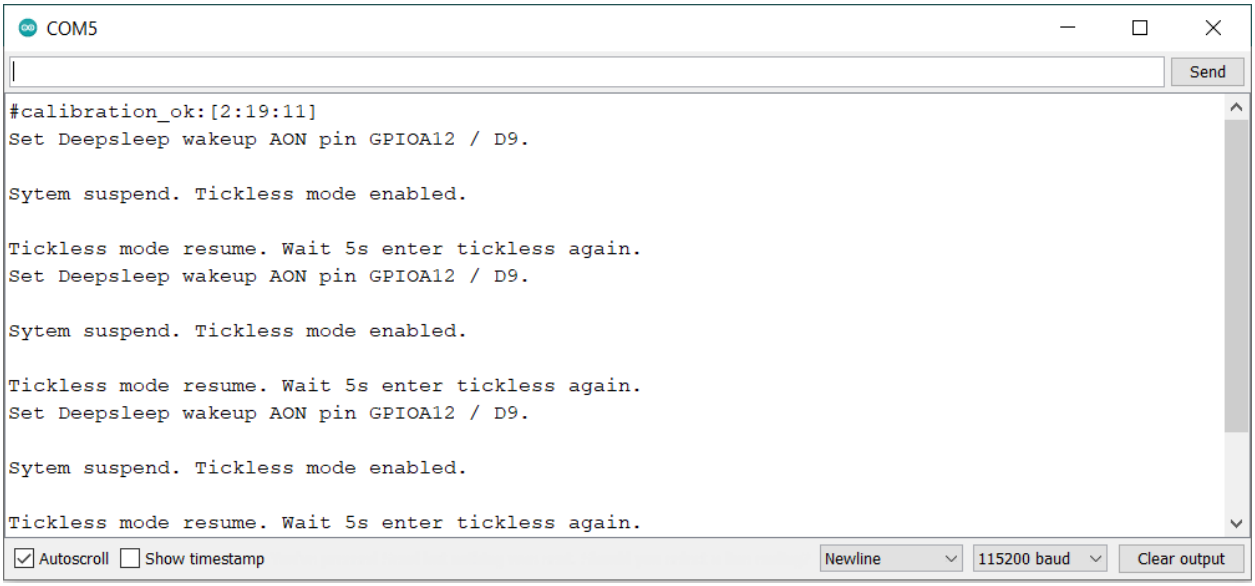
LOGUART



RTC Timer



AON GPIO Pins



Code Reference

Please refer to the API Documents PowerSave section for detail description of all API.

PWM - Play Music by Buzzer

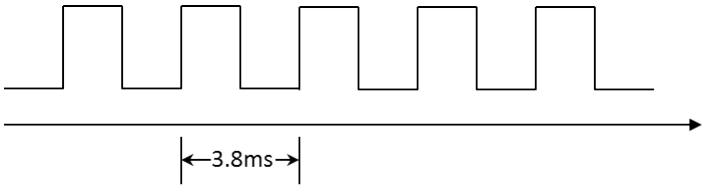
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Buzzer x 1

Example

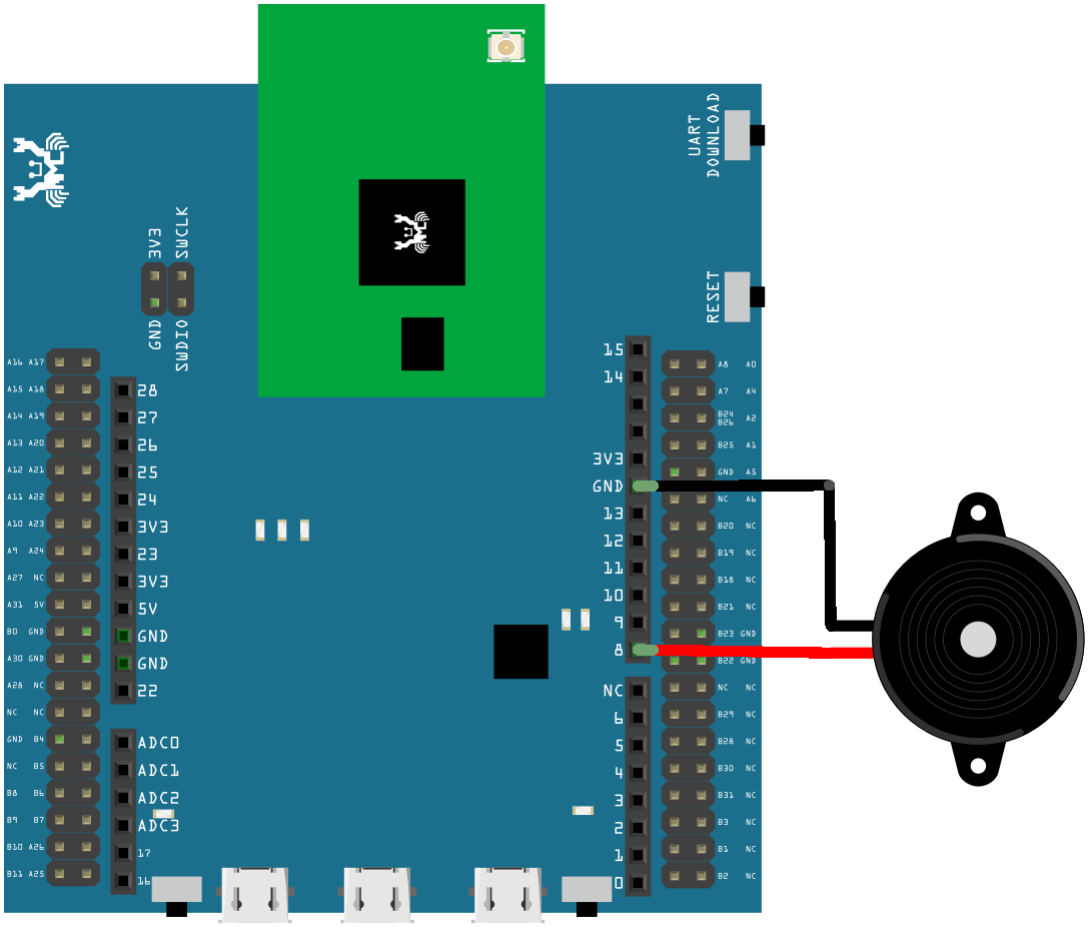
A sound is composed of volume, tone and timbre. Volume is determined by the amplitude of the sound wave. Tone is determined by the frequency of the sound wave. Timbre is determined by the waveform of the sound wave.

In this example, we use PWM to control the buzzer to emit sound with desired tone. As PWM outputs square wave, if we wish to emit tone C4 (frequency=262Hz), we have to make PWM to output square wave with wavelength 1/262 = 3.8ms:

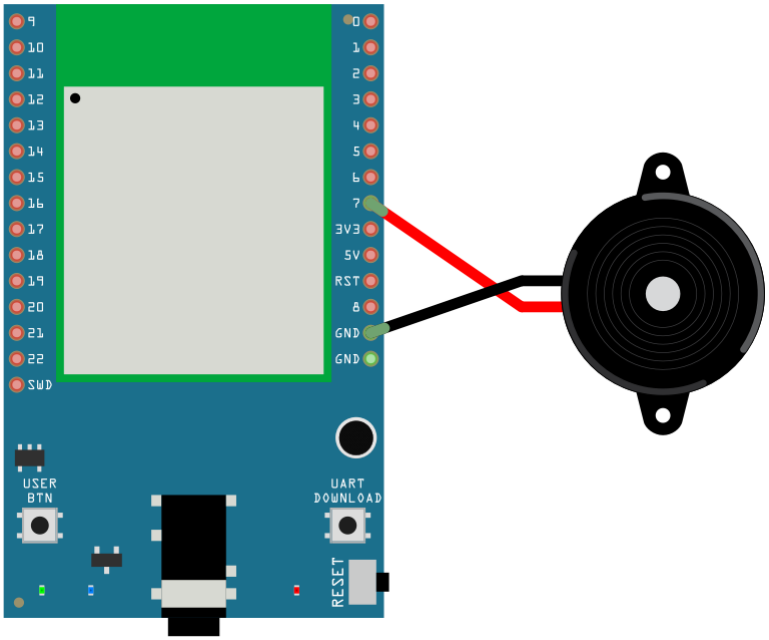


We use PWM to output sound wave with different frequency, so as to play music by the buzzer.
Connect the buzzer to the PWM output pin shown in the following diagrams.

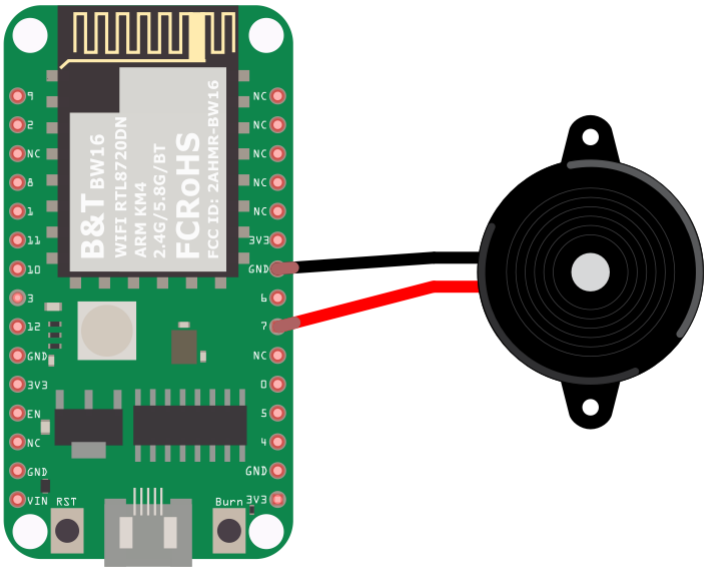
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example code in "`Examples`" -> "`AmebaAnalog`" -> "`TonePlayMelody`"
Compile and upload to Ameba, press the reset button. Then you can hear the buzzer playing music.

Code Reference

Ameba implement the tone() and noTone() API of Arduino:
https://www.arduino.cc/en/Reference/Tone
https://www.arduino.cc/en/Reference/NoTone

In the sample code, we initiate a melody array, which stores the tones to make. Another array, noteDurations, contains the length of each tone, 4 represents quarter note (equals to 3000ms/4 = 750ms, and plus an extra 30% time pause), 8 represents eighth note.

PWM - Servo Control

Preparation

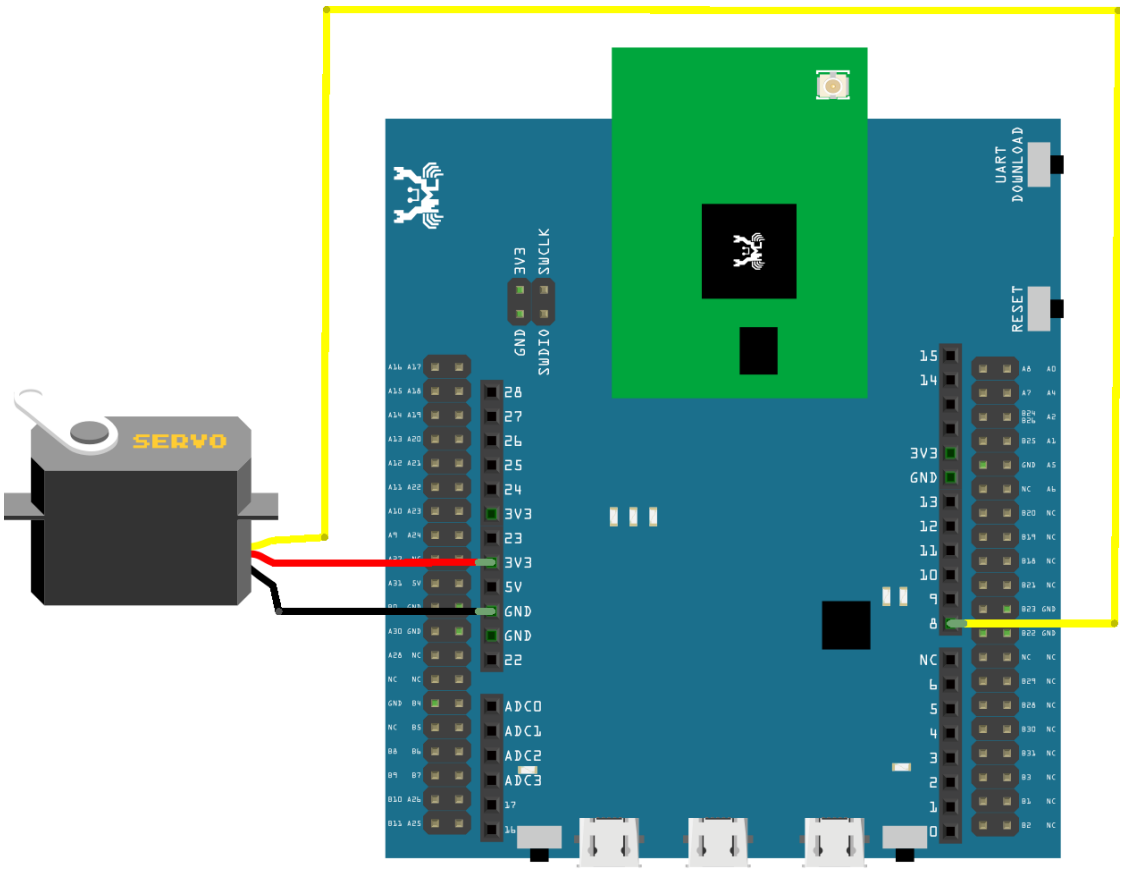
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Servo x 1 (Ex. Tower Pro SG90)

Example

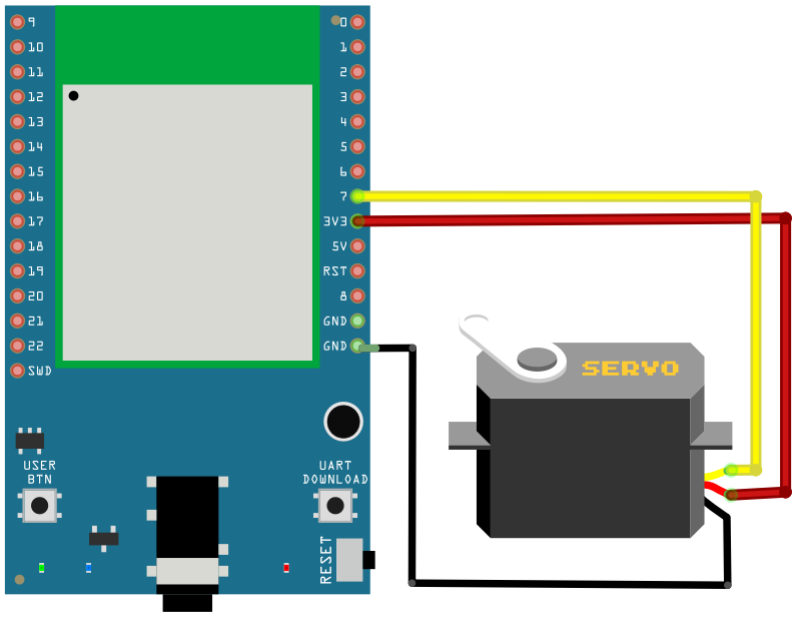
A typical servo has 3 wires, the red wire is for power, black or brown one should be connected to GND, and the other one is for signal data. We use PWM signal to control the rotation angle of the axis of the servo. The frequency of the signal is 50Hz, that is length 20ms. Each servo defines its pulse bandwidth, which is usually 1ms~2ms.

To control the rotation angle, for example if 1ms-length pulse rotates the axis to degree 0, then 1.5 ms pulse rotates the axis to 90 degrees, and 2 ms pulse rotates the axis to 180 degrees. Furthermore, a servo defines the “dead bandwidth”, which stands for the required minimum difference of the length of two consecutive pulse for the servo to work.

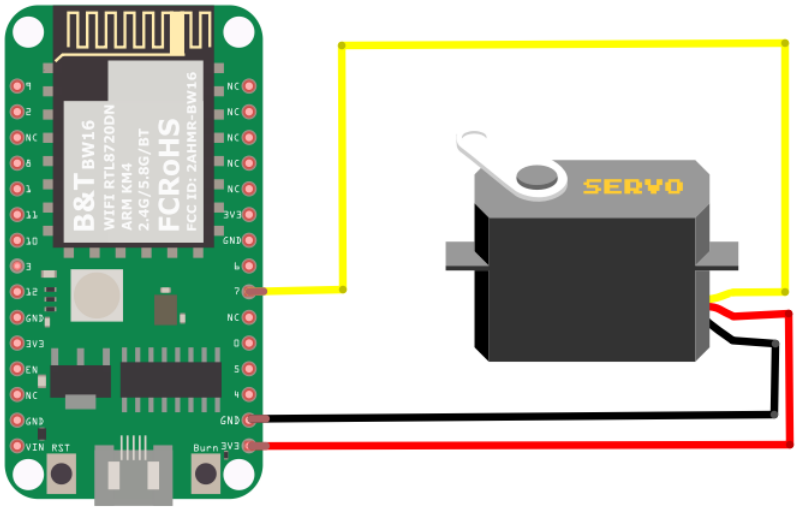
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:



BW16 Wiring Diagram:



Open the example, “File” -> “Examples” -> “AmebaAnalog” -> “ServoSweep”
This example makes the servo to rotate from degree 0 to 180, and then rotate back to degree 0.

Code Reference

The Servo API of Ameba is similar to the API of Arduino. To distinguish from the original API of Arduino, we name the header file “AmebaServo.h” and the Class “AmebaServo”, the usage is identical to the Arduino API.

The default pulse bandwidth of Arduino Servo is 0.5ms~2.4ms, which is the same as Tower Pro SG90. Therefore, we set the attached pin directly:

```
myservo.attach(9);
```

Next, rotate the axis to desired position:

```
myservo.write(pos);
```

RTC - Simple RTC

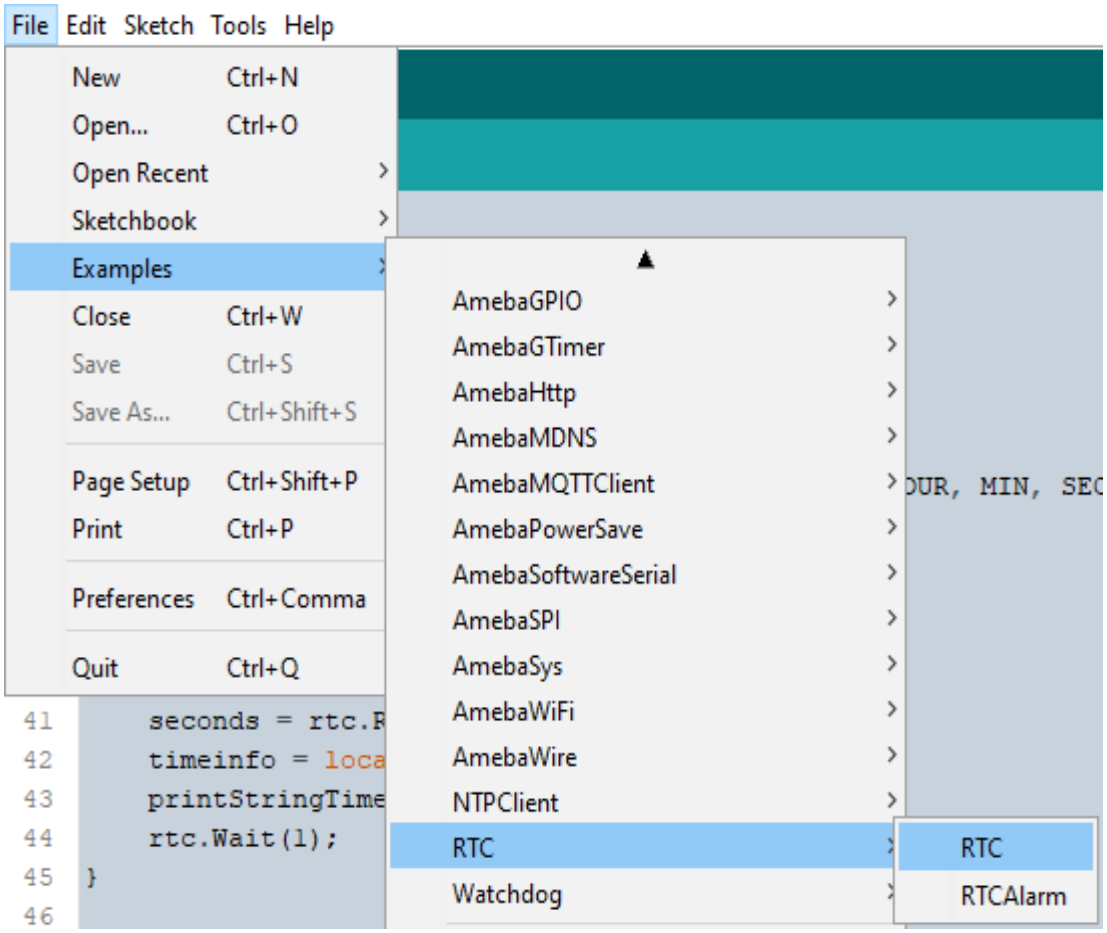
Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

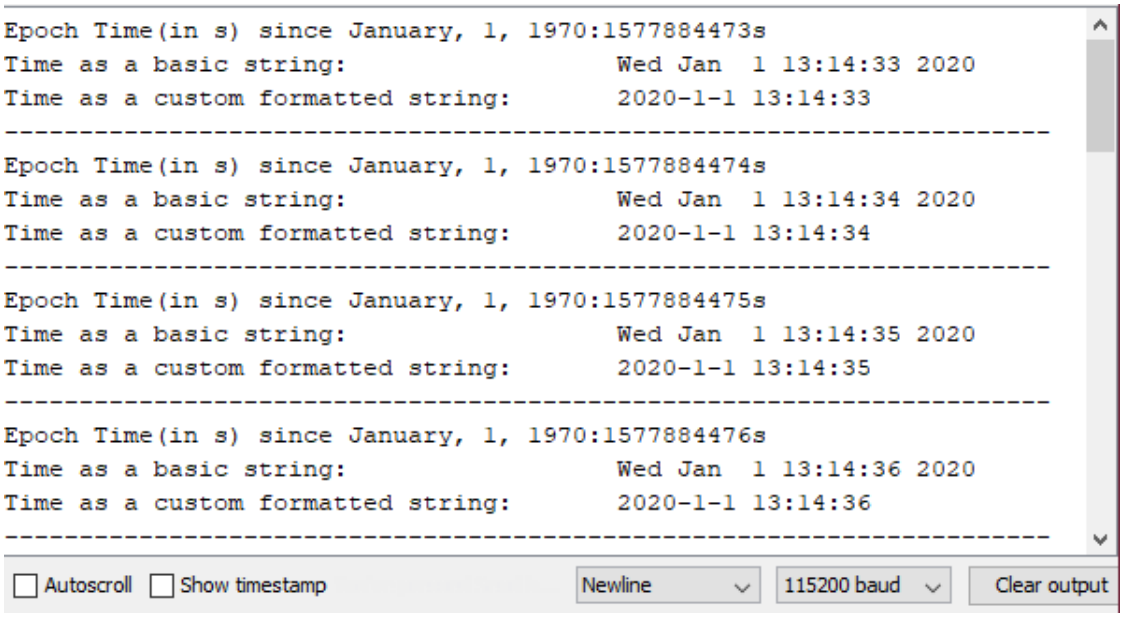
Example

This example demonstrates how to use the RTC library methods. This function describes how to use the RTC API. The RTC function is implemented by an independent BCD timer/counter.

Select the correct Ameba development board from the Arduino IDE: “Tools” -> “Board”.
Then open the “RTC” example from:
`"File" -> "Examples" -> "AmebaRTC" -> "RTC":`



Upon successfully upload the sample code and press the reset button, this example will print out time information since the user initialized time every second in the Serial Monitor.



Code Reference

[1] Simple RTC example from Arduino Tutorials:
https://www.arduino.cc/en/Tutorial/SimpleRTC

RTC - Simple RTC Alarm

Materials

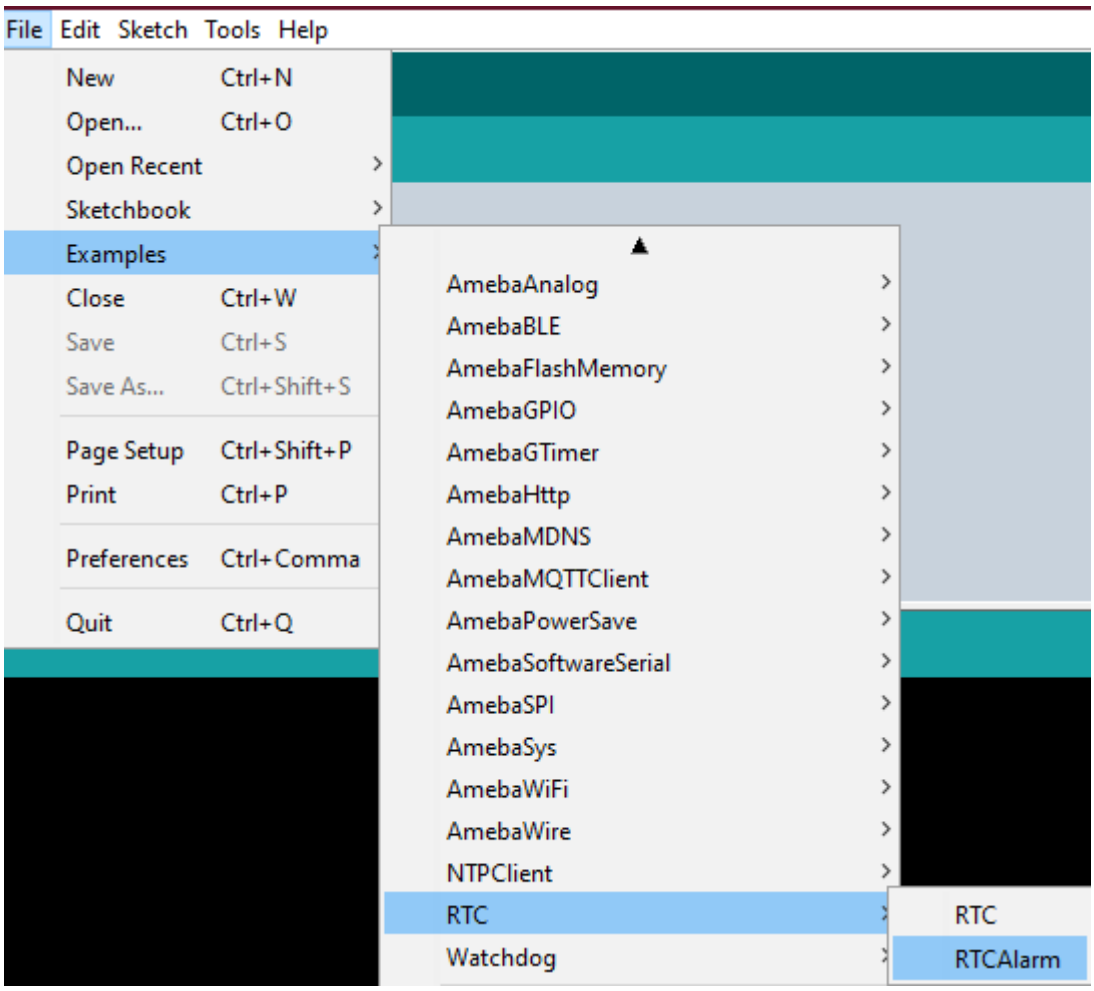
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

Example

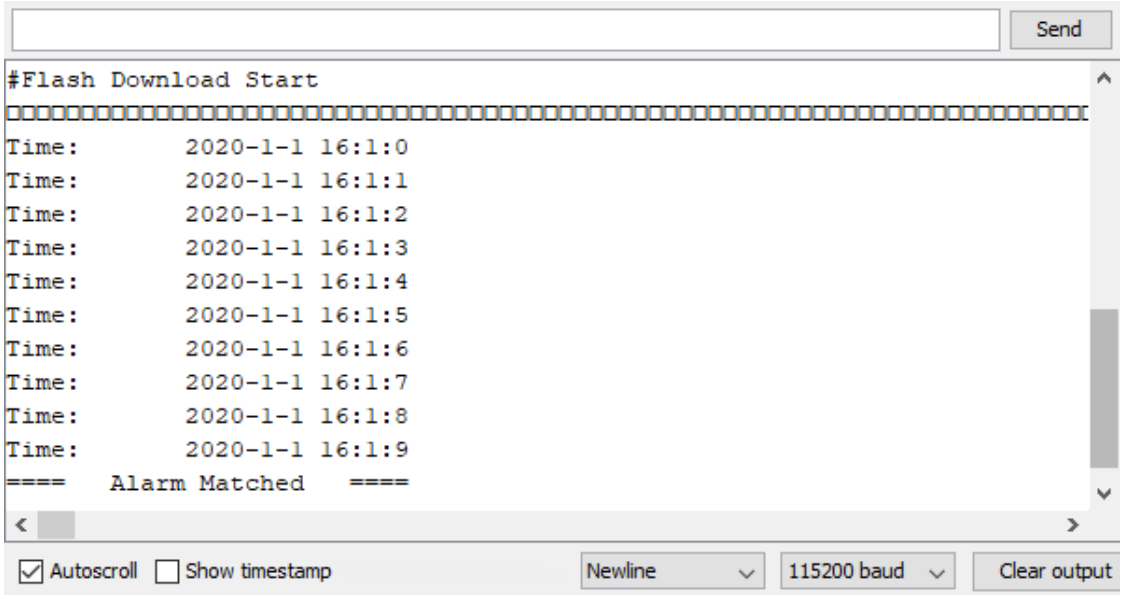
This example demonstrates how to use the RTC library methods to create a RTC Alarm, so that to do some tasks when an alarm is matched. In particular, the RTC time is set at 16:00:00 and an alarm at 16:00:10. When the time matches, "Alarm Match" information will be printed on the serial monitor.

First, select the correct Ameba development board from the Arduino IDE: "Tools" -> "Board".

Then open the "RTCAlarm" example from: `"File" -> "Examples" -> "RTC" -> "RTCAlarm"`:



In the example, the RTC time is set at 16:00:00 and an alarm is set at 16:00:10. Upon successfully upload the sample code and press the reset button. When the alarm time (10 seconds) is reached the attached interrupt function will print the following information: "Alarm Matched!" showing in this figure below.



SPI – Print Image And Text On LCD Screen

If you are not familiar with SPI, please read Introduction to SPI first.

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1

Example

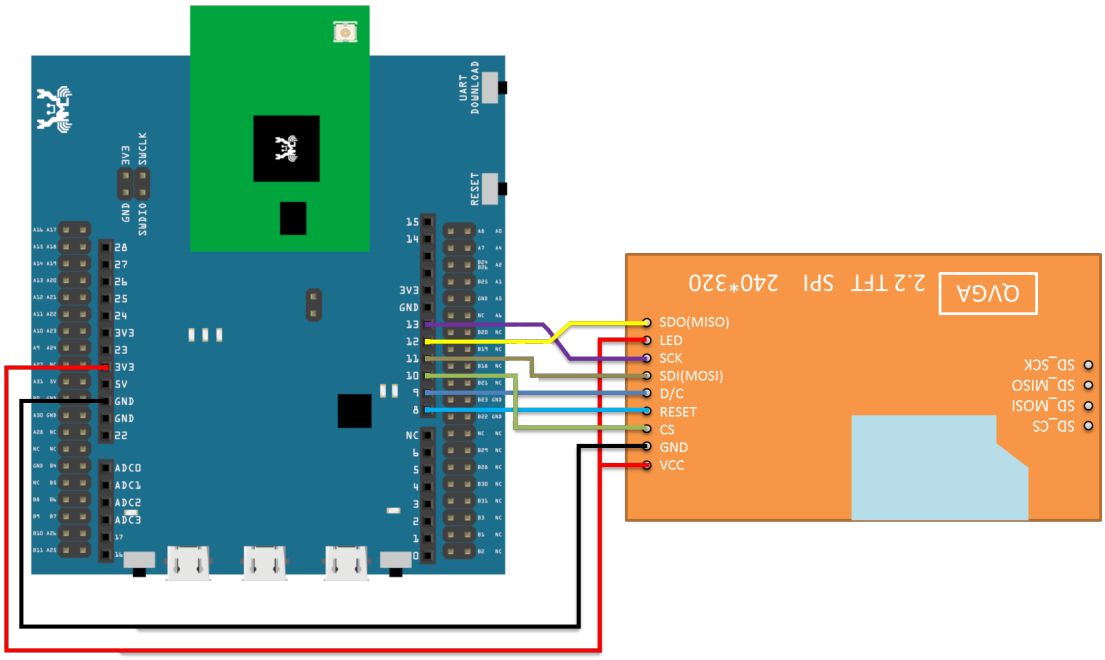
We have tested the following two models of ILI9341 TFT LCD with SPI interface:

- Adafruit 2.8″ TFT LCD (with touch screen)
  - https://www.adafruit.com/products/1651
  - https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2?view=all
- QVGA 2.2″ TFT LCD
  - http://www.lcdwiki.com/2.2inch_SPI_Module_ILI9341_SKU:MSP2202

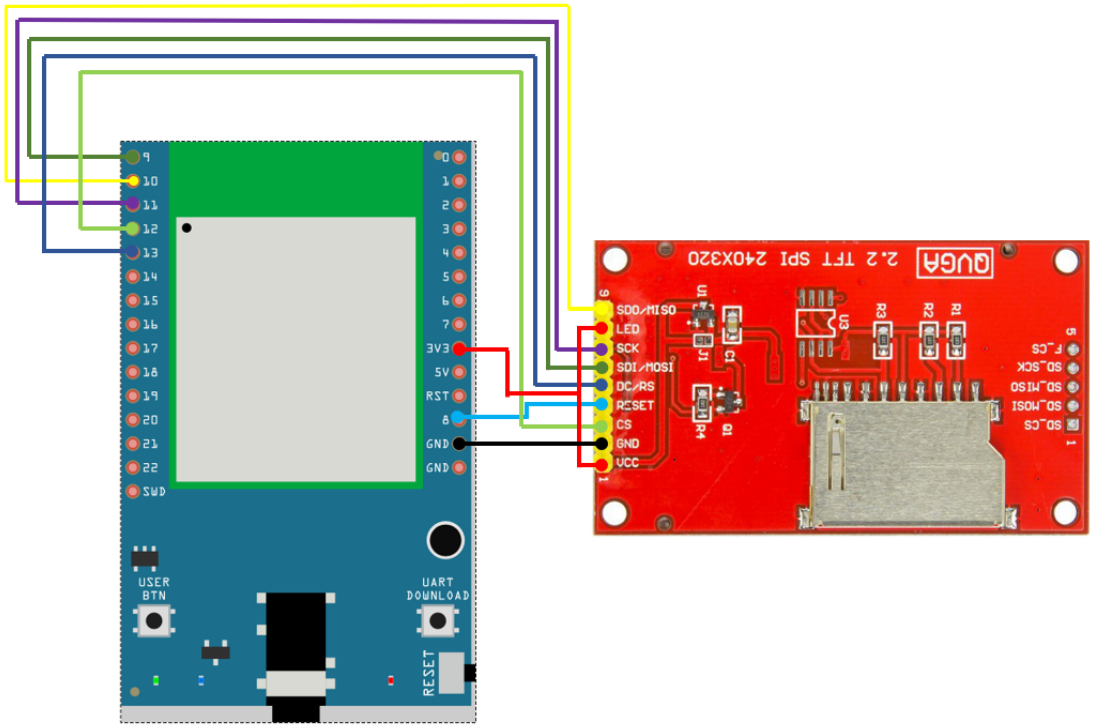
Common pins in ILI9341 TFT LCD with SPI interface:

- MOSI: Standard SPI Pin
- MISO: Standard SPI Pin
- SLK: Standard SPI Pin
- CS: Standard SPI Pin
- RESET: Used to reboot LCD.
- D/C: Data/Command. When it is at Low, the signal transmitted are commands, otherwise the data transmitted are data.
- LED (or BL): Adapt the screen backlight. Can be controlled by PWM or connected to VCC for 100% backlight.
- VCC: Connected to 3V or 5V, depends on its spec.
- GND: Connected to GND.

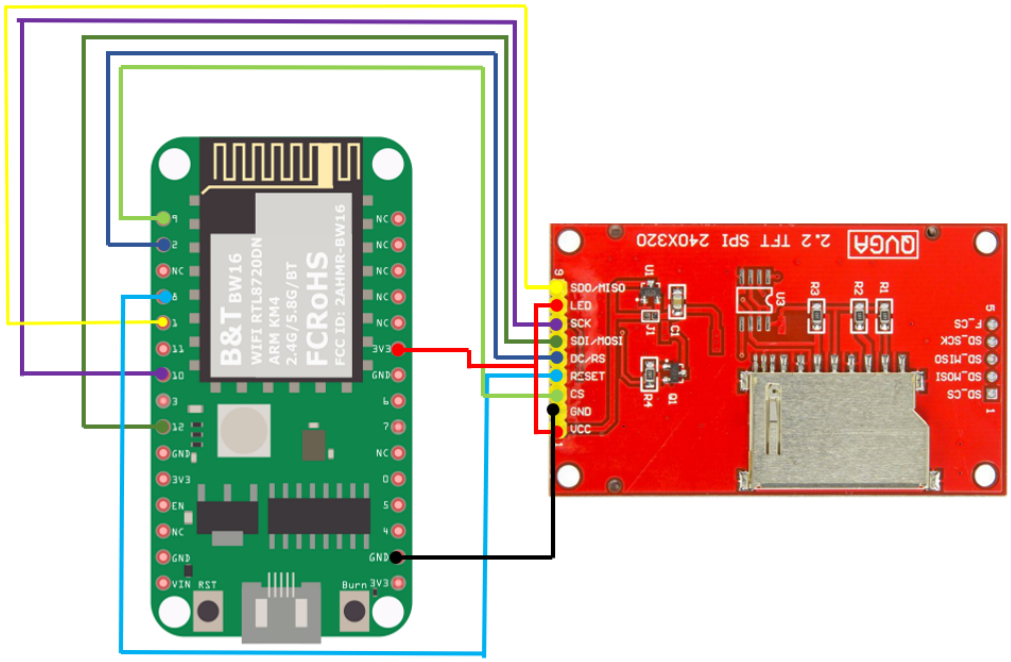
AMB21/ AMB22 and QVGA TFT LCD Wiring Diagram:



AMB23 and QVGA TFT LCD Wiring Diagram:



BW16 and QVGA TFT LCD Wiring Diagram:

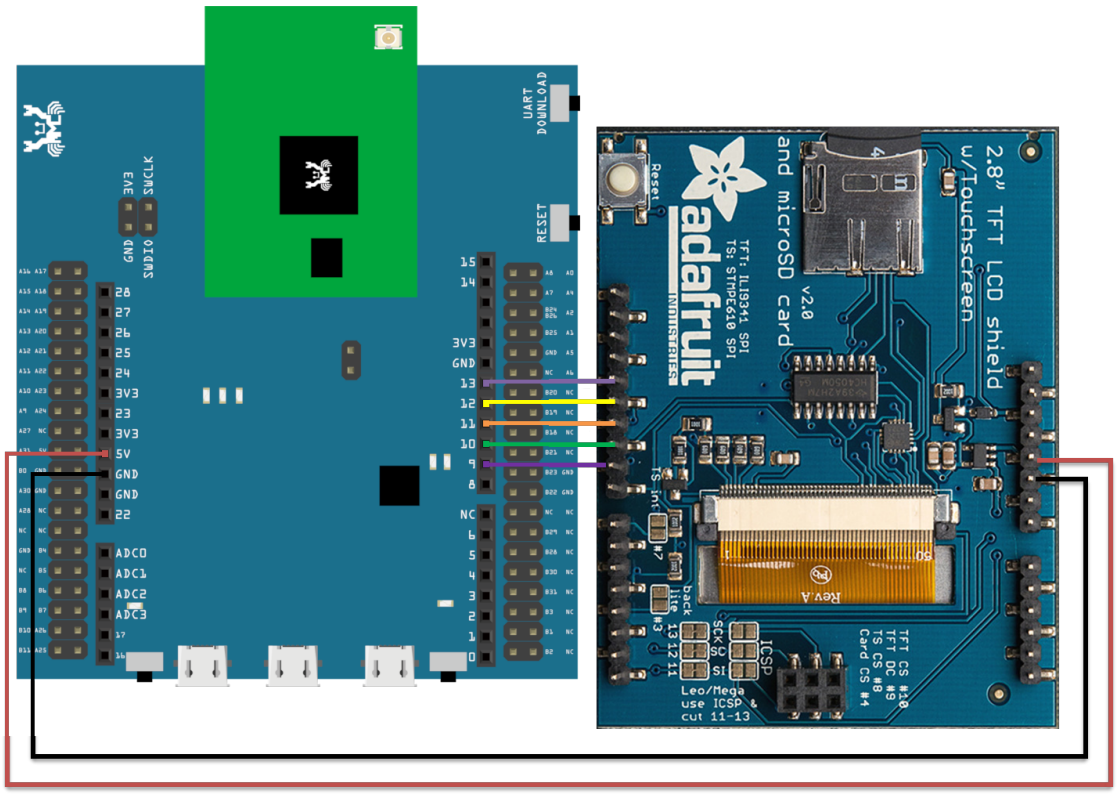


Wiring example of Adafruit 2.8” TFT LCD touch shield:

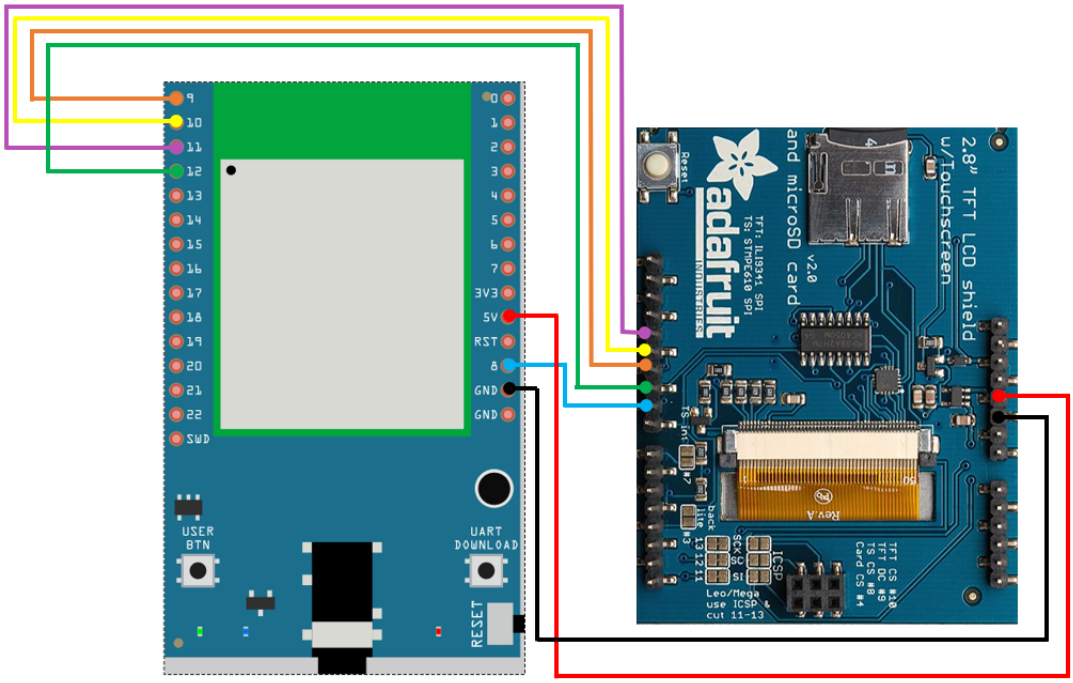
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.

AMB21 / AMB22 and Adafruit 2.8” TFT LCD touch shield Wiring Diagram:

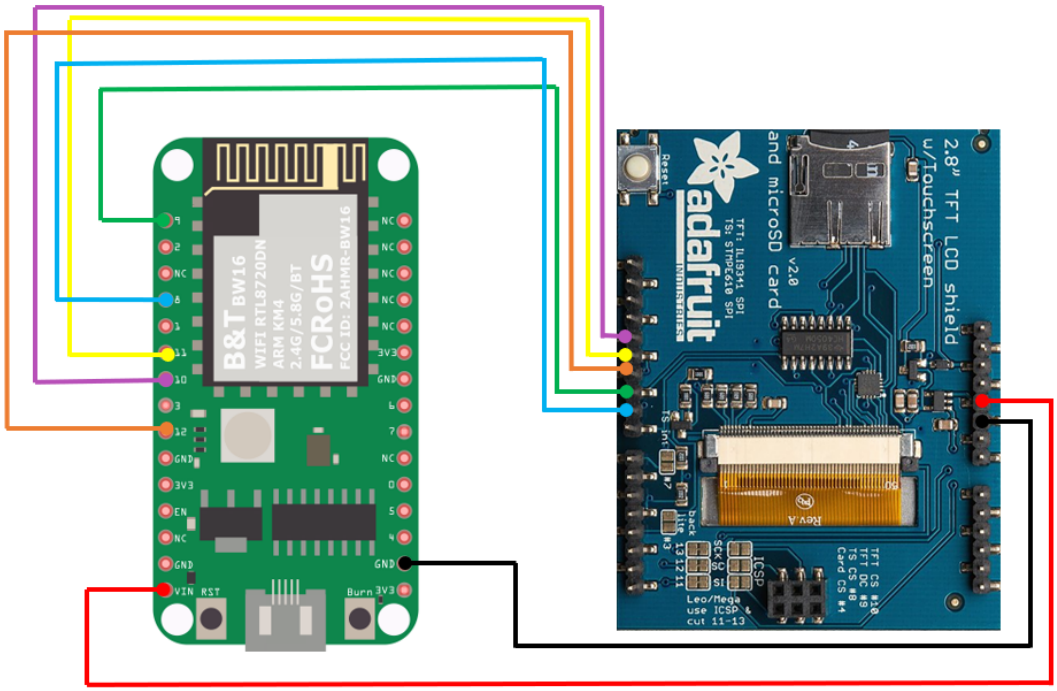
Please note that this shield model enables the backlight by default and pin 8 is not for backlight, and the VCC should be connected to 5V.



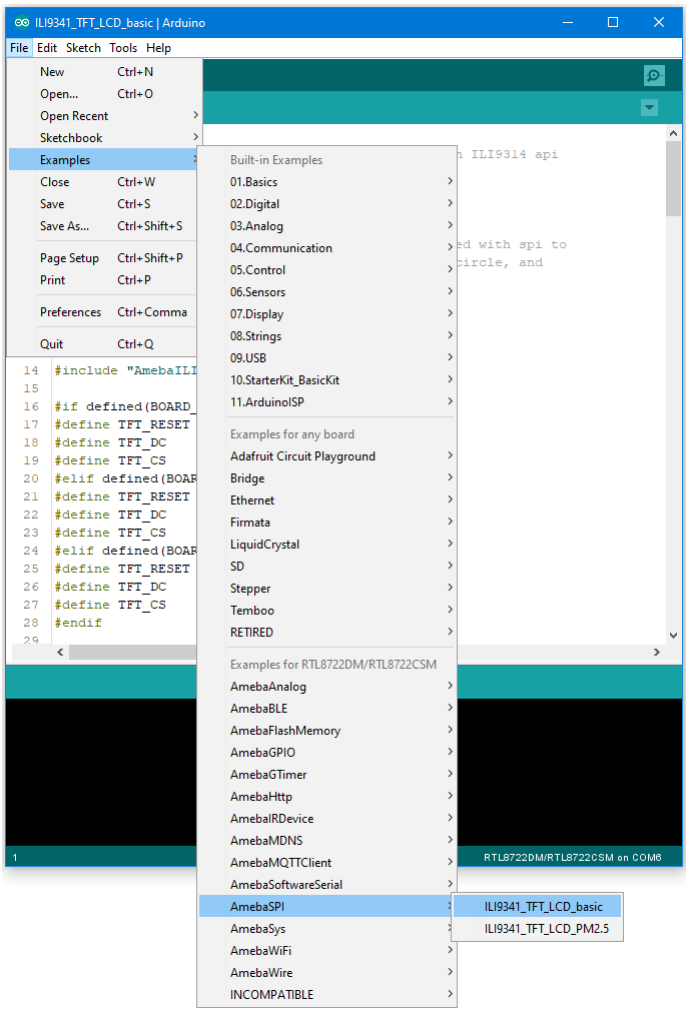
AMB23 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:



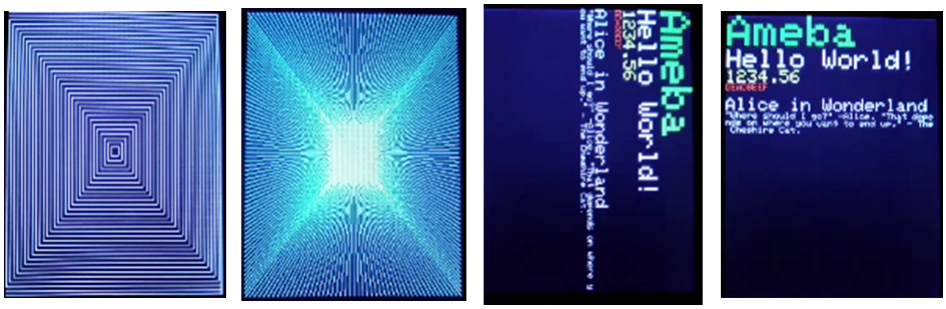
BW16 and Adafruit 2.8" TFT LCD touch shield Wiring Diagram:



Open the example, “Files” -> “Examples” -> “AmebaSPI” -> “ILI9341_TFT_LCD_basic”



Compile and upload to Ameba, then press the reset button.
Then you can see some display tests appear on the LCD screen, such as displaying different colors, drawing vertical and horizontal lines, drawing circles, etc.…



Code Reference

- **RGB 16-bit**

  ILI9341 uses RGB 16-bit to display colors. Different from RGB 24-bit, it uses 5 bits for red, 6 bits for green, 5

bits for blue. For example, the RGB 24-bit representation of sky blue is 0x87CEFF, that is in binary:

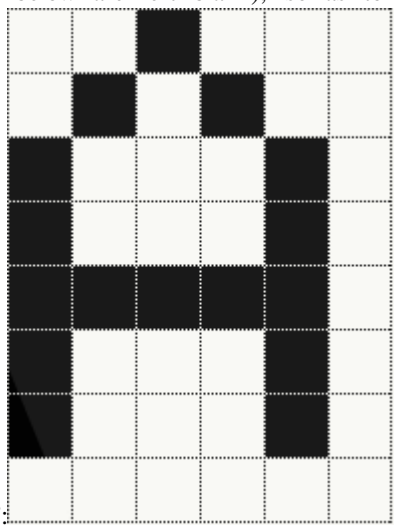
  - Red: 0x87 = B10000111
  - Green: 0xCE = B11001110
  - Blue: 0xFF = B11111111

and converted to RGB 16-bit:

  - Red: B10000
  - Green: B110011
  - Blue: B11111

Then concatenate them, which forms B1000011001111111 = 0x867F

- **Drawing of ILI9341**
  - First you must specify the range of the rectangle to draw, then pass the 2-byte RGB 16-bit color to ILI9341 corresponding to each pixel one by one, in this way ILI9341 fills each color to each pixel.
  - You still must specify the drawing range even though the range covers only one pixel.
  - From the rules we mentioned above, we can conclude that drawing vertical or horizontal lines are faster than diagonal lines.
- **Printing text on ILI9341**
  - In our API, each character is 5×7 but each character is printed to size 6×8 (its right side and below are left blank), so as to separate from next character. For example, the character "A":



  - The font size represents the dot size. For example, if the font size is 2, each dot in the character is a 2×2 rectangle
- **Screen rotation**
  - ILI9341 provides 0, 90, 180, 270 degrees screen rotation.
  - If the original width is 240 and original height is 320, when the screen rotates 90 degrees, the width becomes 320 and the height becomes 240.

SPI – Show PM2.5 Concentration On ILI9341 TFT LCD

If you are not familiar with SPI, please read Introduction to SPI first.

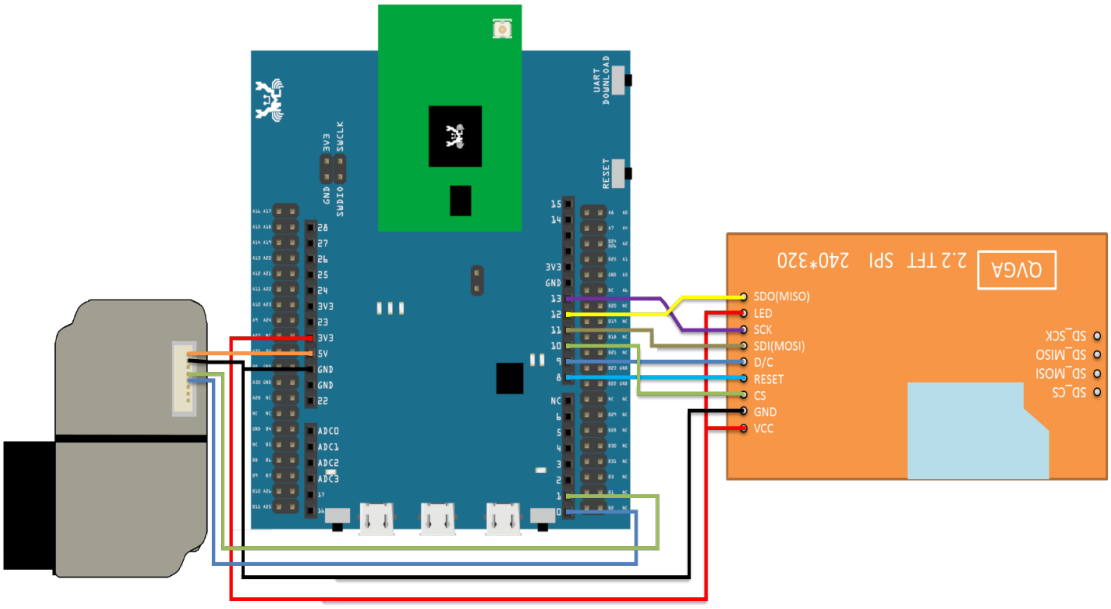
Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- ILI9341 TFT LCD with SPI interface x 1
- Plantower PMS3003 or PMS5003 x 1

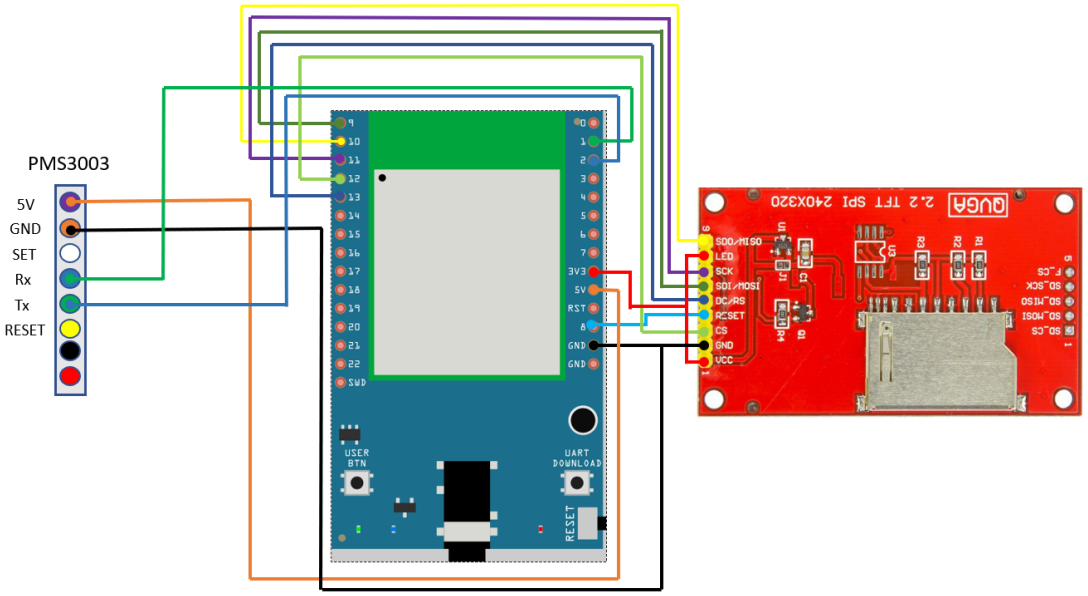
Example

This example extends previous PM2.5 example to show the PM2.5 concentration on the LCD.

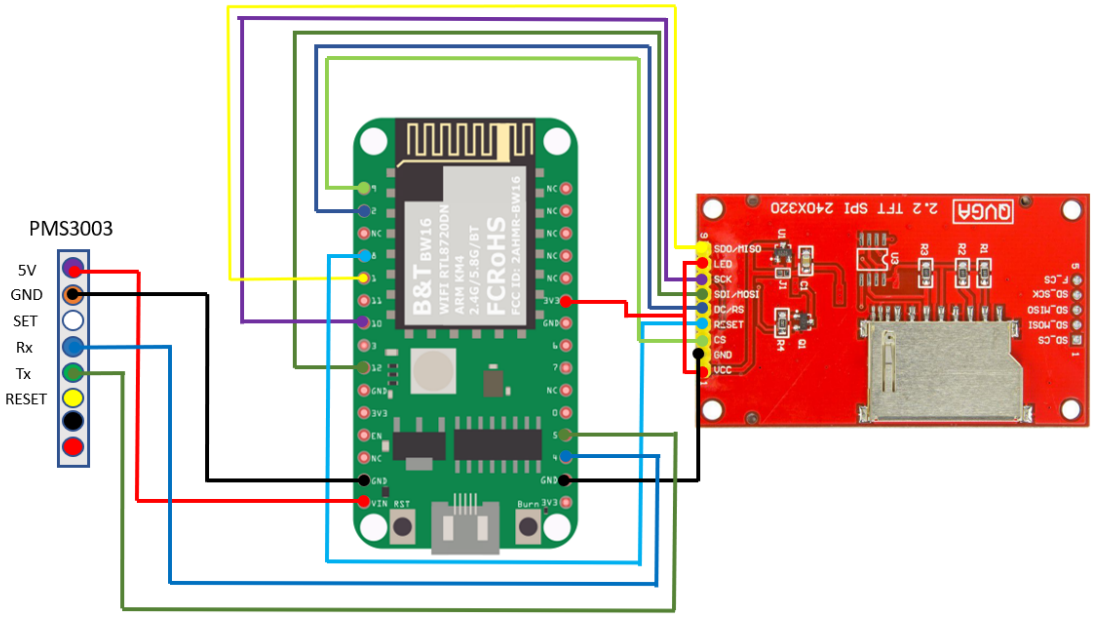
AMB21 / AMB22 and QVGA TFT LCD Wiring Diagram:
(Note: PMS3003/PMS5003 sensor requires 5V voltage)



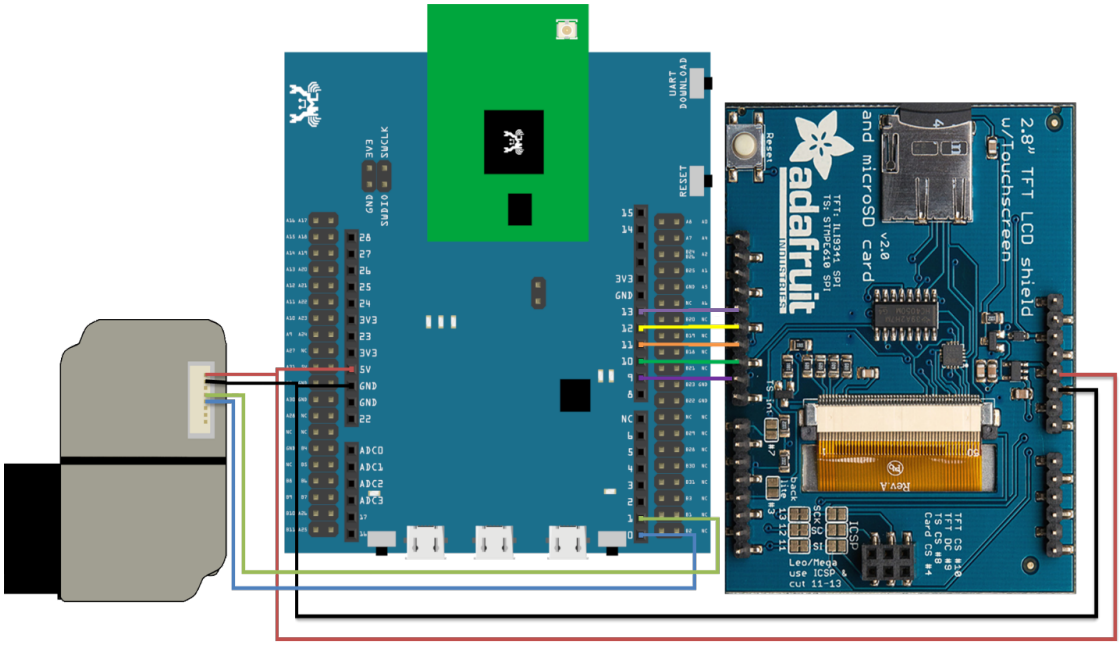
AMB23 and QVGA TFT LCD Wiring Diagram:



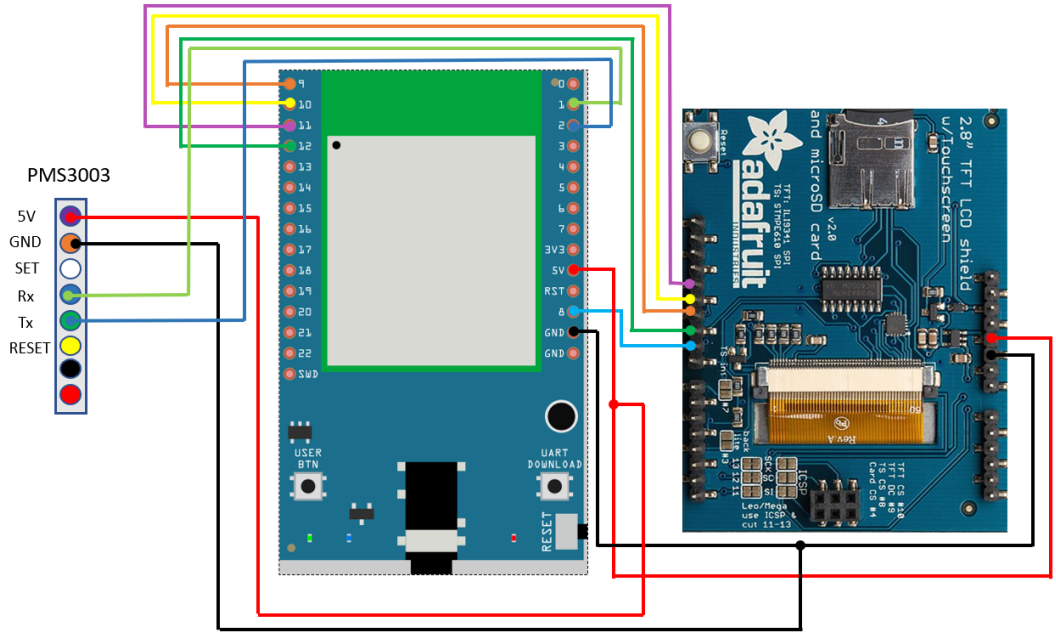
BW16 and QVGA TFT LCD Wiring Diagram:



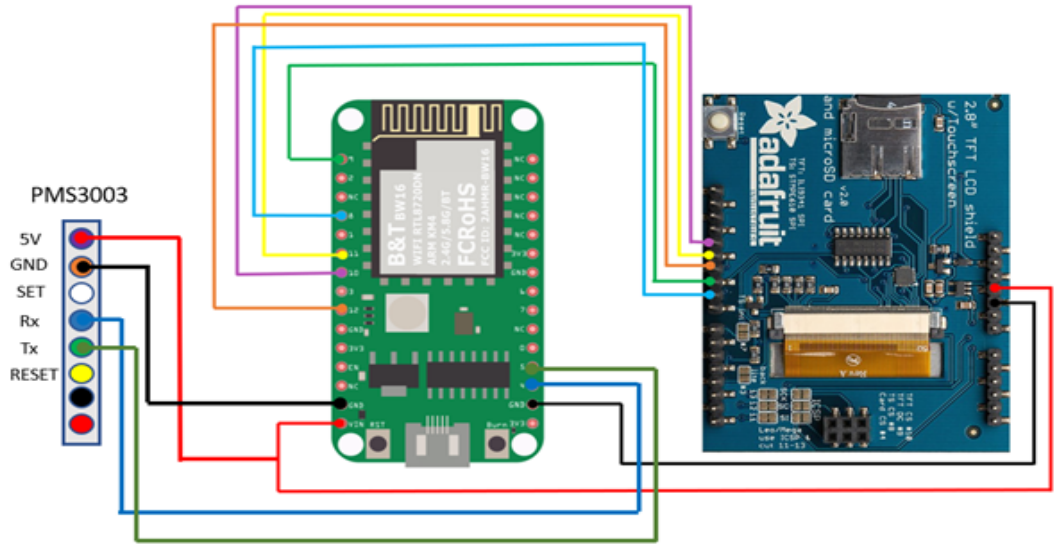
AMB21 / AMB22 and Adafruit 2.8" TFT LCD Wiring Diagram:



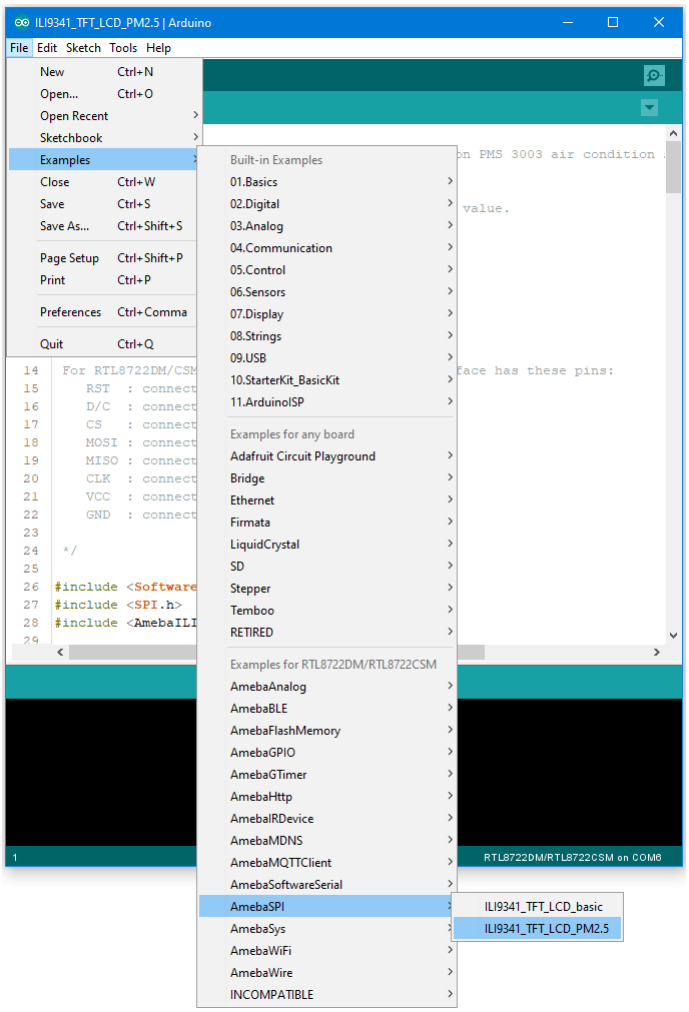
AMB23 and and Adafruit 2.8" TFT LCD Wiring Diagram:



BW16 and and Adafruit 2.8" TFT LCD Wiring Diagram:

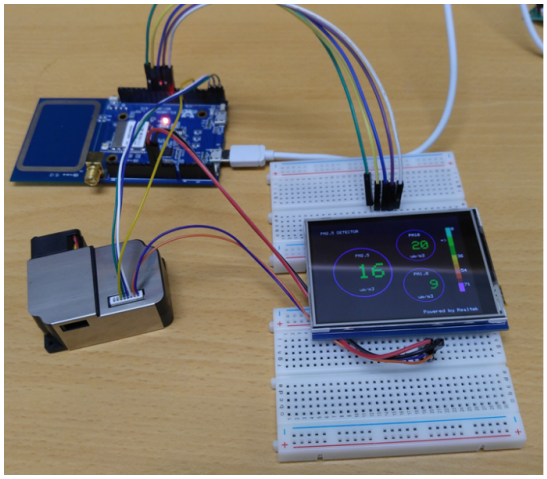
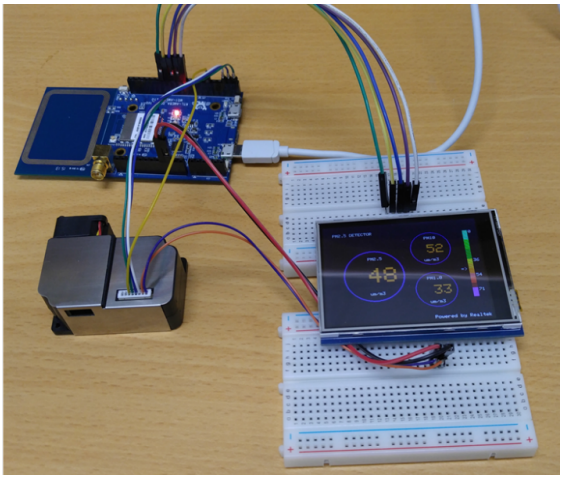


Open the example, "`Files` -> "`Examples`" -> "`AmebaSPI`" -> "`PM25_on_ILI9341_TFT_LCD`"



Compile and upload to Ameba, then press the reset button.

Then you can see the concentration value of PM1.0, PM2.5 and PM10 on the LCD.

Code Reference

In this example, first rotate the screen by 90 degrees, and draw the static components such as the circles, the measuring scale, and the title text. After the concentration value is detected, it is printed inside the circle.

UART - Communicate with PC over USB to Serial Module

Introduction of UART

UART uses two wire, one for transmitting and the other one for receiving, so the data transmission is bidirectional. The communication uses a predefined frequency (baud rate) to transmit data. In Arduino, UART is called "Serial". There is only one hardware UART on Arduino Uno and it is primarily used to read the log and messages printed by Arduino (so it is also called "Log UART"). If we use the hardware UART for other purposes, the Log UART does not have resources to function. To provide more UART connections, it is possible to use a GPIO pin to simulate the behavior of UART with a software approach, this is called Software Serial. Ameba is equipped with several hardware UART ports, but it is also compatible with the Software Serial library.

Materials

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

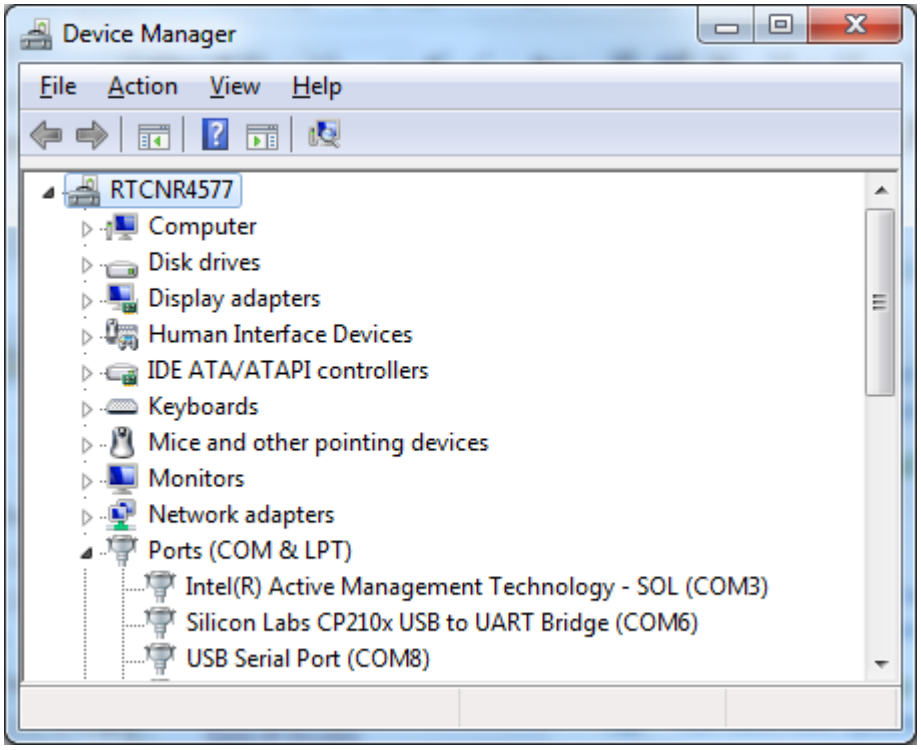
Example

In this example, we use UART to connect USB to TTL adapter to Ameba.
USB to TTL adapter sends data to Ameba, the data would be returned by Ameba, and showed on the screen.

- **Install USB to TTL Adapter**

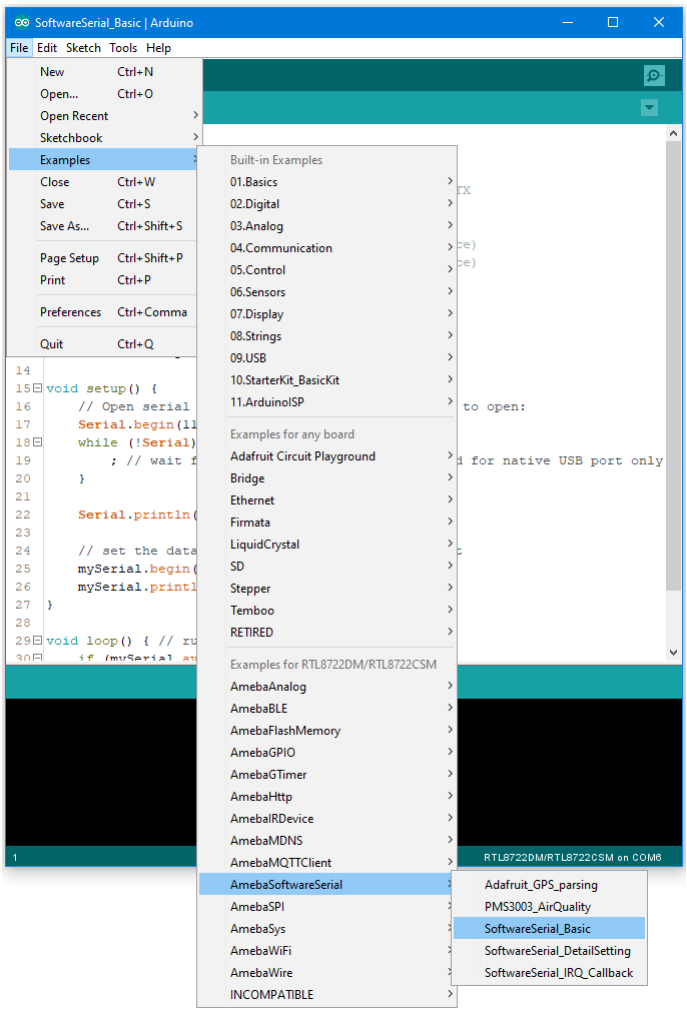
USB to TTL adapter converts USB to serial interface. Normally, there are at least 4 pins on the adapter, that is 3V3 (or 5V), GND, TX and RX. Generally, installing the driver for the USB to TTL adapter would be required before using it. If the adapter uses the chip of FTDI, Windows will search and install the driver automatically, otherwise, you may need to install corresponding driver yourself.

Afterwards, open device manager. You can find corresponding serial port number of the USB to TTL adapter:



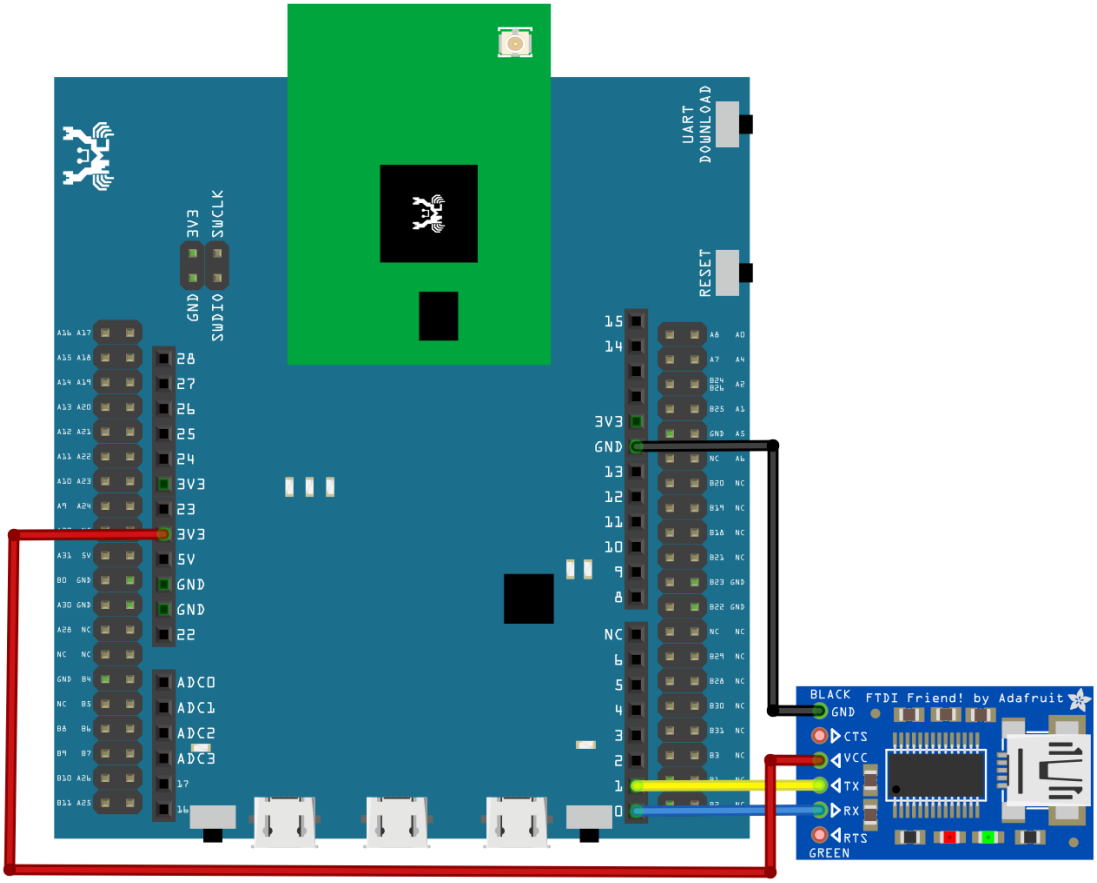
- Executing the Example

Open the "SoftwareSerialExample" example in "`File`" -> "`Examples`" -> "`AmebaSoftwareSerial`" -> "`SoftwareSerial_Basic`":

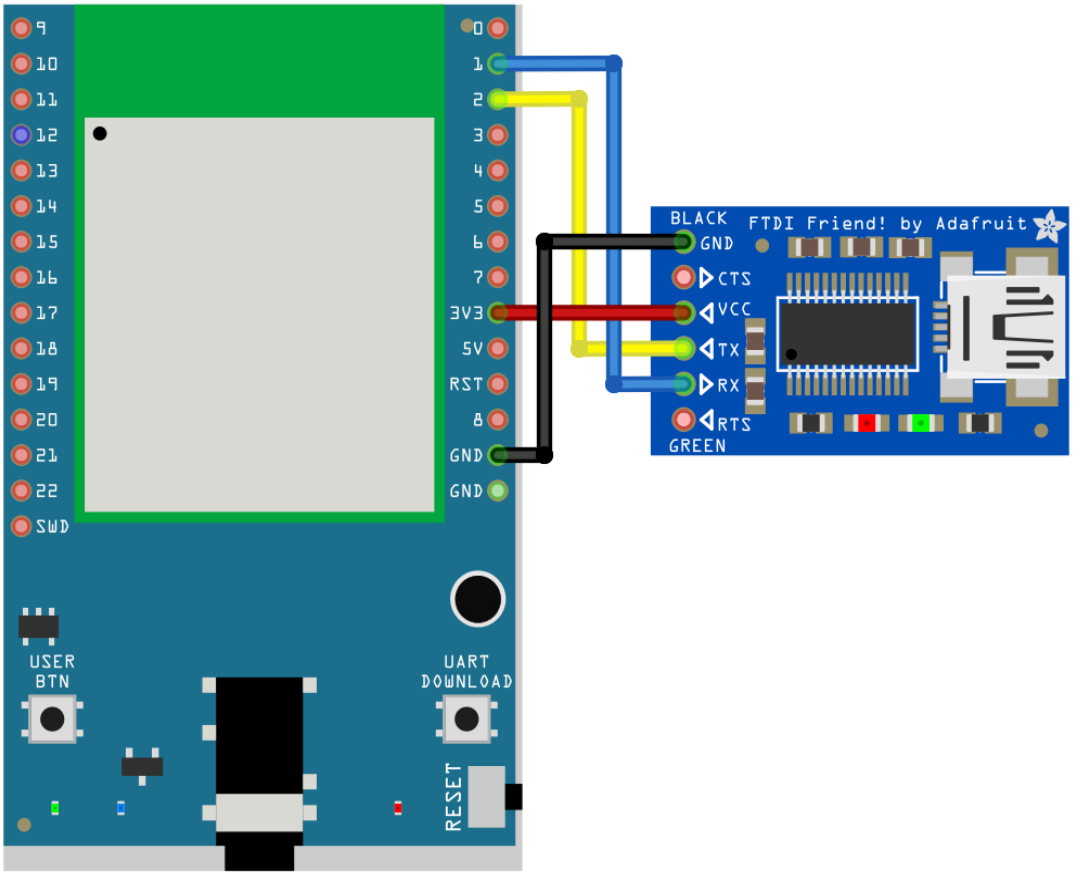


Connect the wire as the following diagrams show. The TX pin of USB to TTL adapter is connected to the RX of Ameba, and the RX pin of USB to TTL adapter is connected to the TX of Ameba.

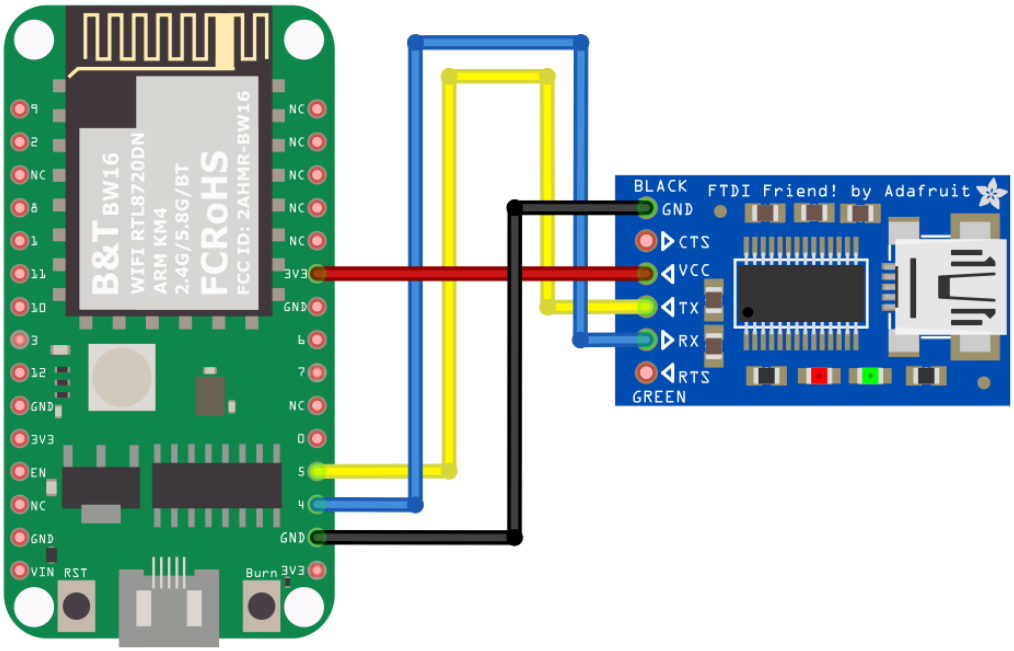
AMB21 / AMB22 Wiring Diagram:



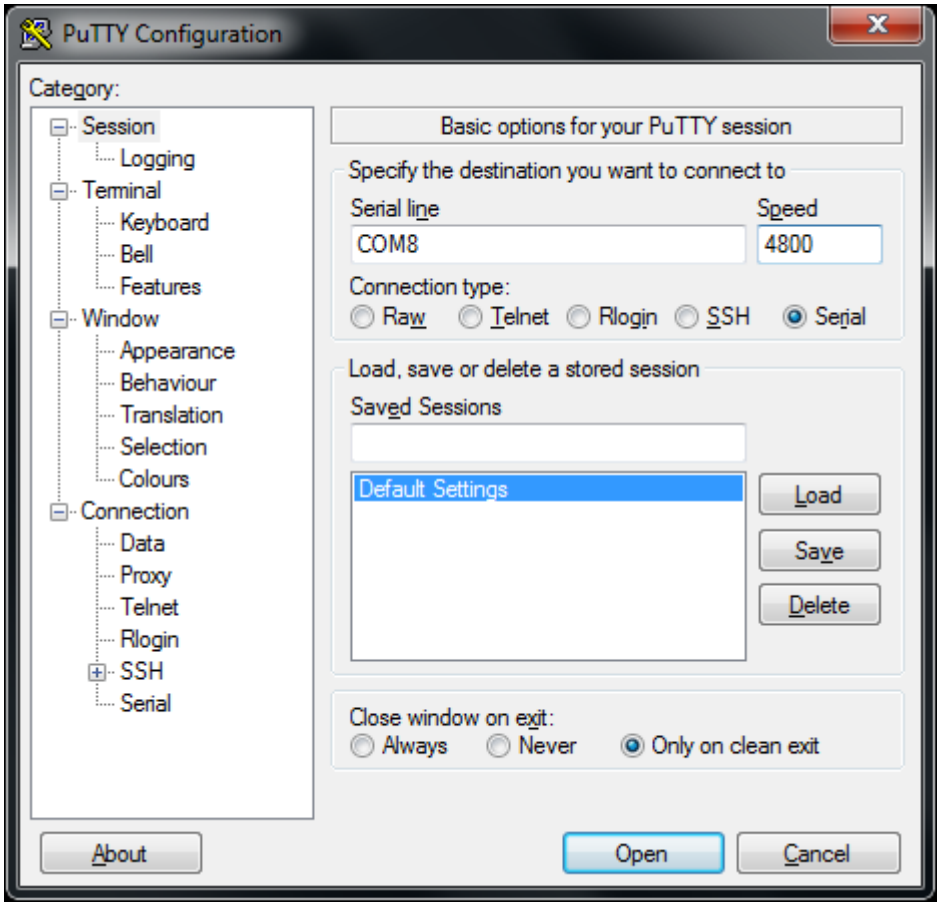
AMB23 Wiring Diagram:



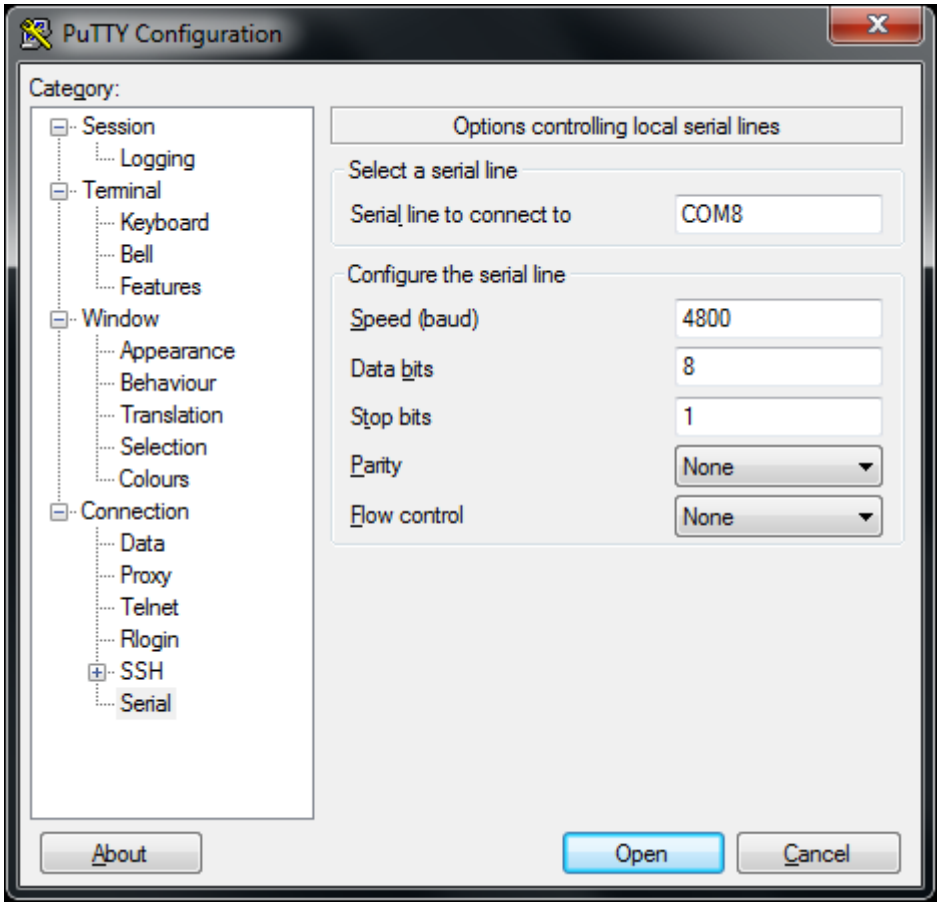
BW16 Wiring Diagram:



Next, open a serial port terminal, such as Putty or Tera Term. (Putty is used in this example). Open the Putty window, choose "Serial" in connection type, and specify the port number of the USB to TTL adapter (e.g. COM8). In the speed field, fill in the baud rate of this connection. Note that both sides of the connection should use the same baud rate. In this example we set baud rate 4800.



Next, select “Serial” on the left side. Set data bits to 8, stop bits to 1, parity to none, and flow control to none.



Then click Open and press the reset button on Ameba. You can see the "Hello, world?" message appears in Putty. If characters are typed into Putty, the input characters would be sent to Serial RX of Ameba by TX of USB to TTL Adapter, and returned by Serial TX of Ameba. Finally, RX of USB to TTL Adapter receives the returned characters and prints them in Putty. Therefore, if you insert "I am fine", you will get something like this:

Code Reference

First, use `SoftwareSerial:begin(speed)` to set the baud rate for the serial communication:
https://www.arduino.cc/en/Reference/SoftwareSerialBegin

Use `write()` to send data, and use `SoftwareSerial:available()` to get the number of bytes available for reading from a software serial port:

https://www.arduino.cc/en/Reference/SoftwareSerialAvailable
If there are data available to read, use read() to read from serial port.

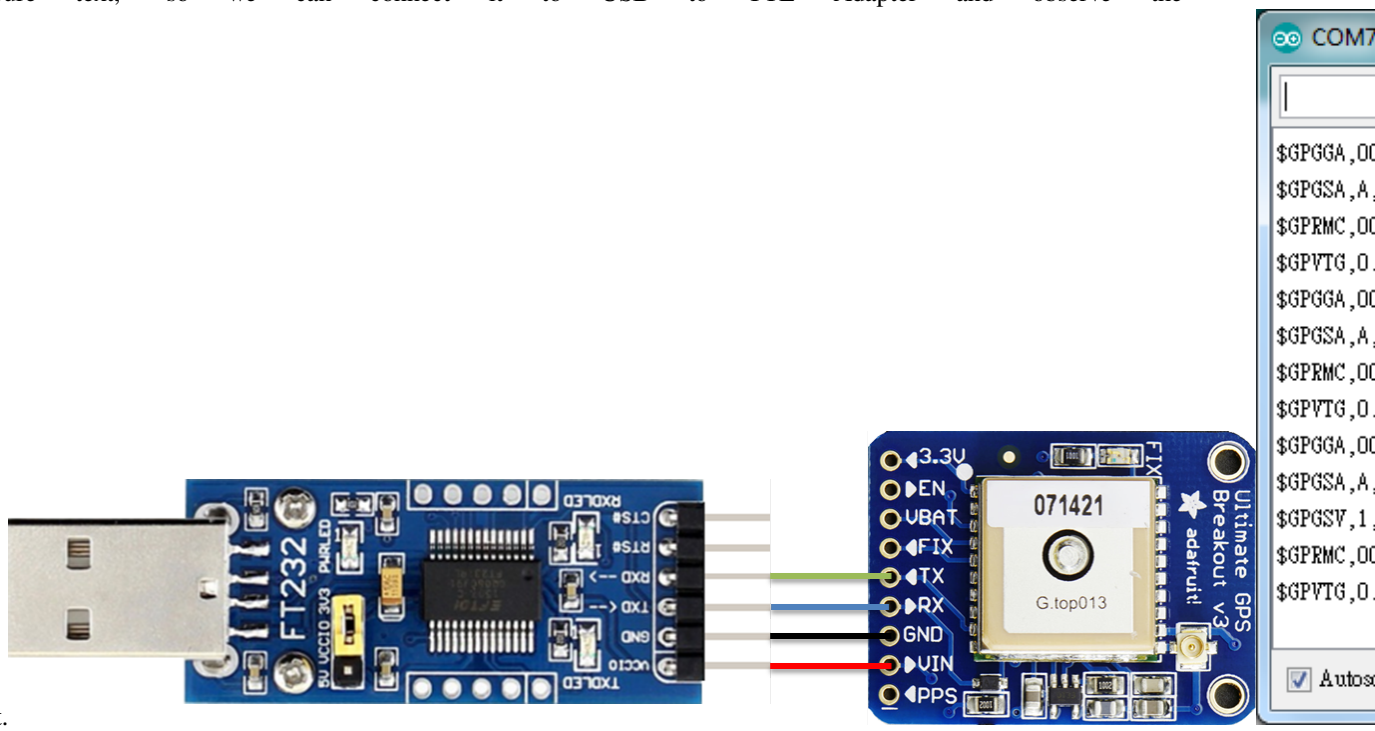
UART - Retrieve GPS Position

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- Adafruit Ultimate GPS Breakout x 1 (Refer to official document)

Example

In this example, we use Adafruit Ultimate GPS Breakout. Its data format is pure text, so we can connect it to USB to TTL Adapter and observe the output.



It follows the NMEA sentence format (refer to http://aprs.gids.nl/nmea/) The GPS signal is weak in indoor environment. The status that the GPS signal is not received is called “not fix”. Bring the GPS module outdoors, when the GPS signal is “fix”, you would get message similar to the figure

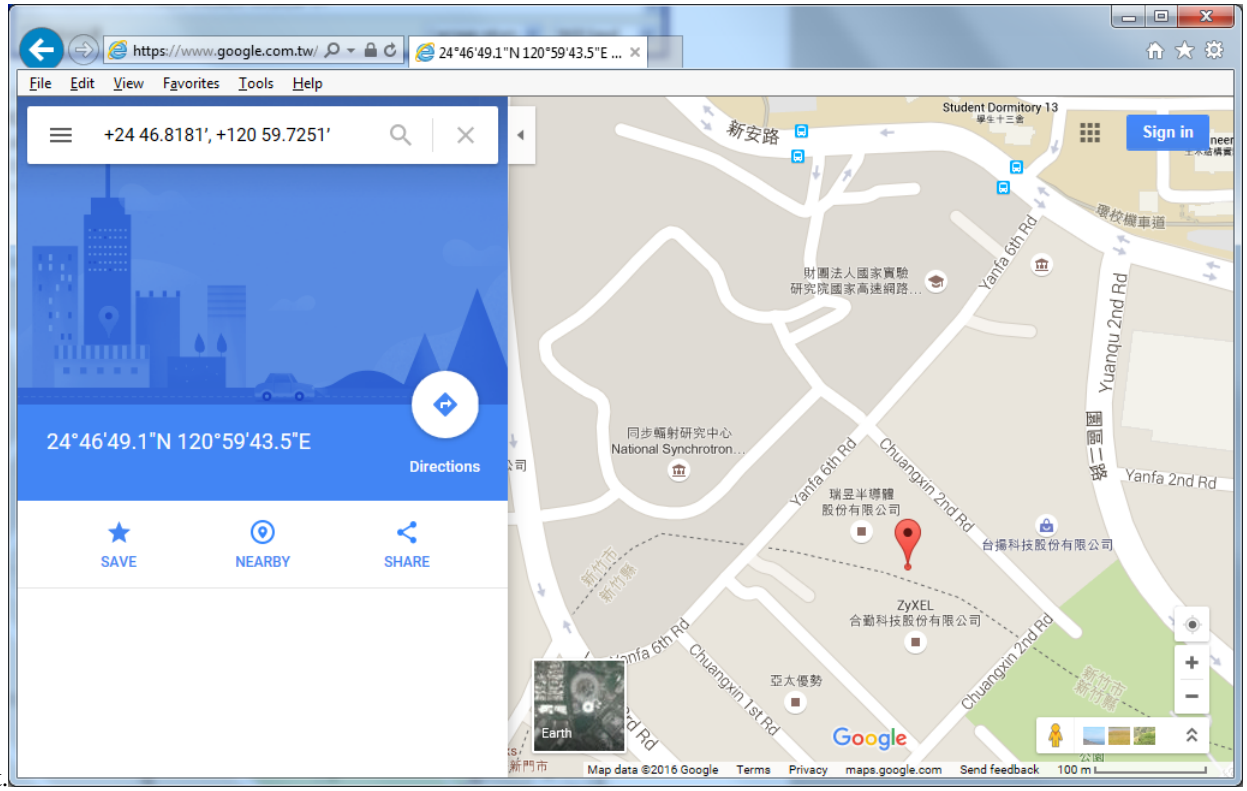
below.

In this example we are only interested in the "$GPRMC (Global Positioning Recommended Minimum Coordinates)":
$GPRMC,032122.000,A,2446.8181,N,12059.7251,E,0.39,78.89,270116,,,A*53
Each field is separated by a comma.

- First field is the GMT time (Greenwich Mean Time), that is 032122.000 in this example. The time format is HH:MM:SS.SSS, i.e., 03:21:22.000. Note that the time zone and the daylight-saving time adjustment should be handled on your own.
- Second field represents the status code
  - V: Void (Invalid)
  - A: Active, meaning the GPS signal is fix.
- The third to sixth fields represent the geolocation

In this example, 2446.8181,N represents 24 degrees 46.8181 minutes north latitude, and 12059.7251,E represents 120 degrees 59.7251 minutes east longitude.

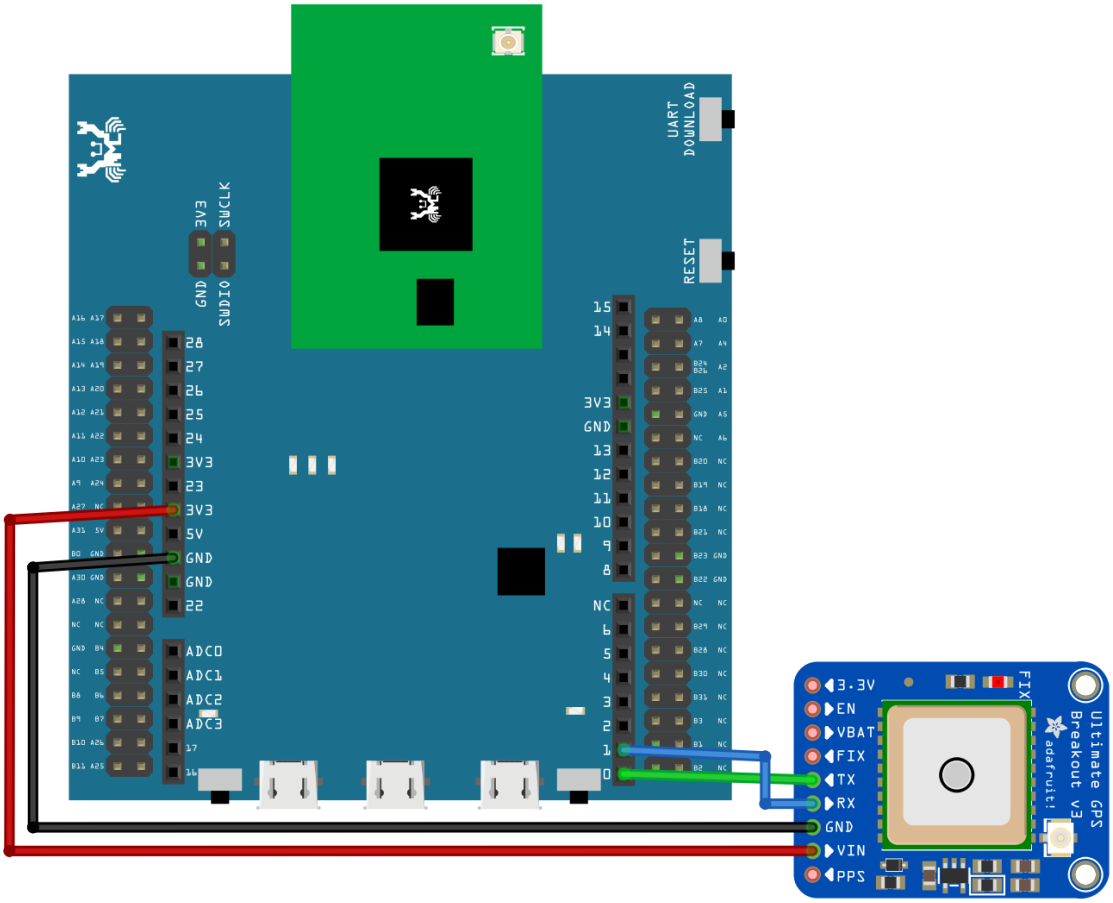
We can search **+24 46.8181', +120 59.7251'** in Google map to check whether the position is



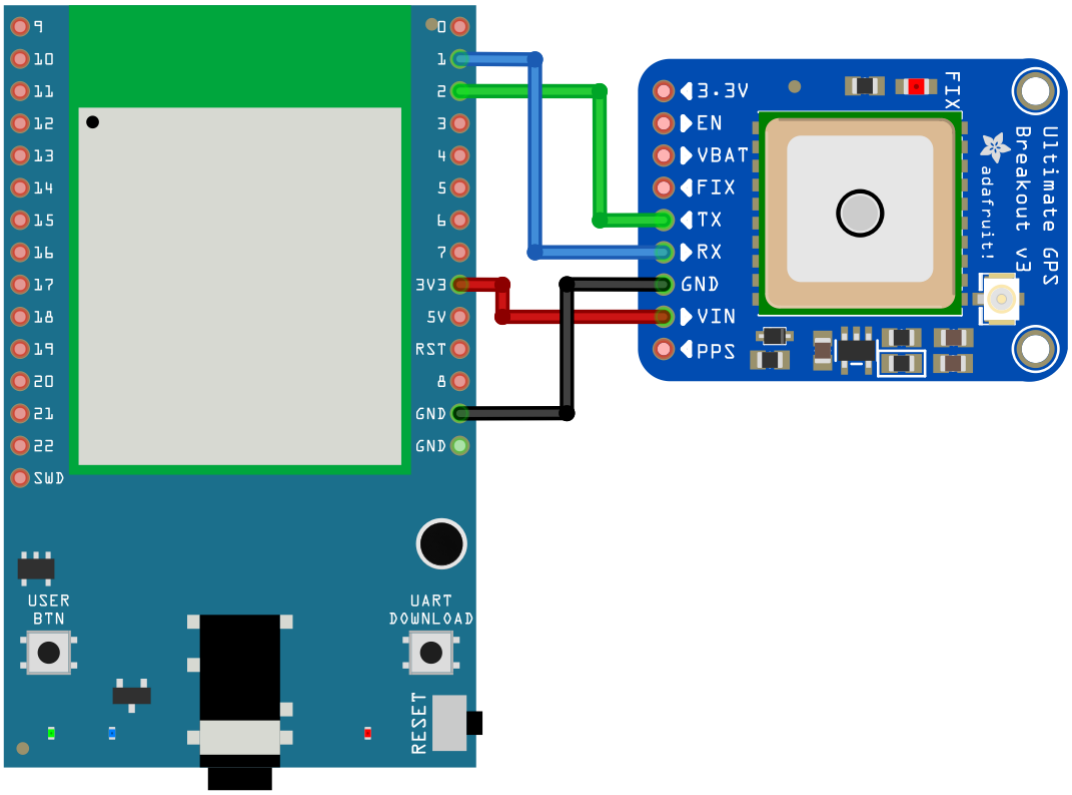
correct.

- The seventh field is relative speed(knot). 1 knot = 1.852km/hr, in this example the relative speed is 0.39 knot.
- The eighth field is the moving angle, which is calculated by its moving orbit.
- The ninth field is the date with format ddMMyy. In this example, "270116" stands for day 27, January, year 2016.
- The last field is checksum. In the example we have *53 as checksum.

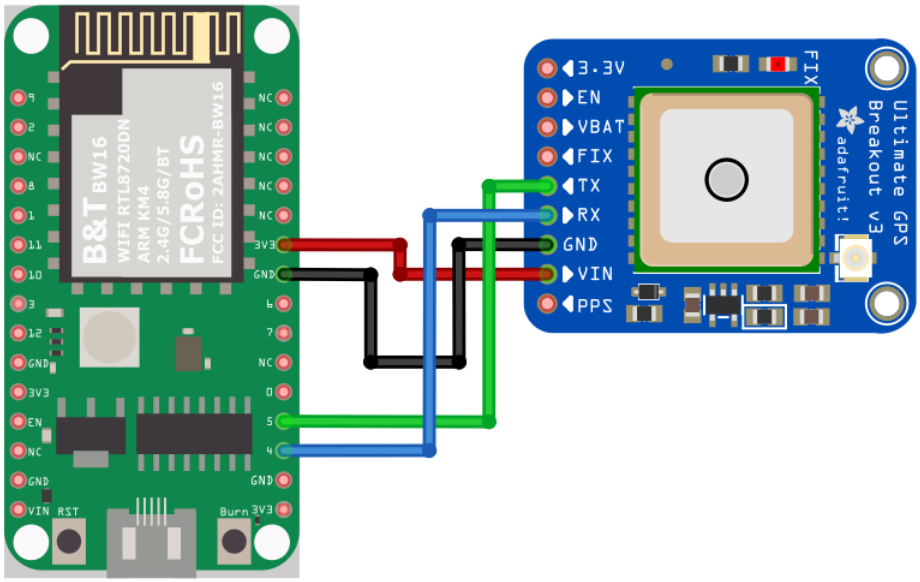
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

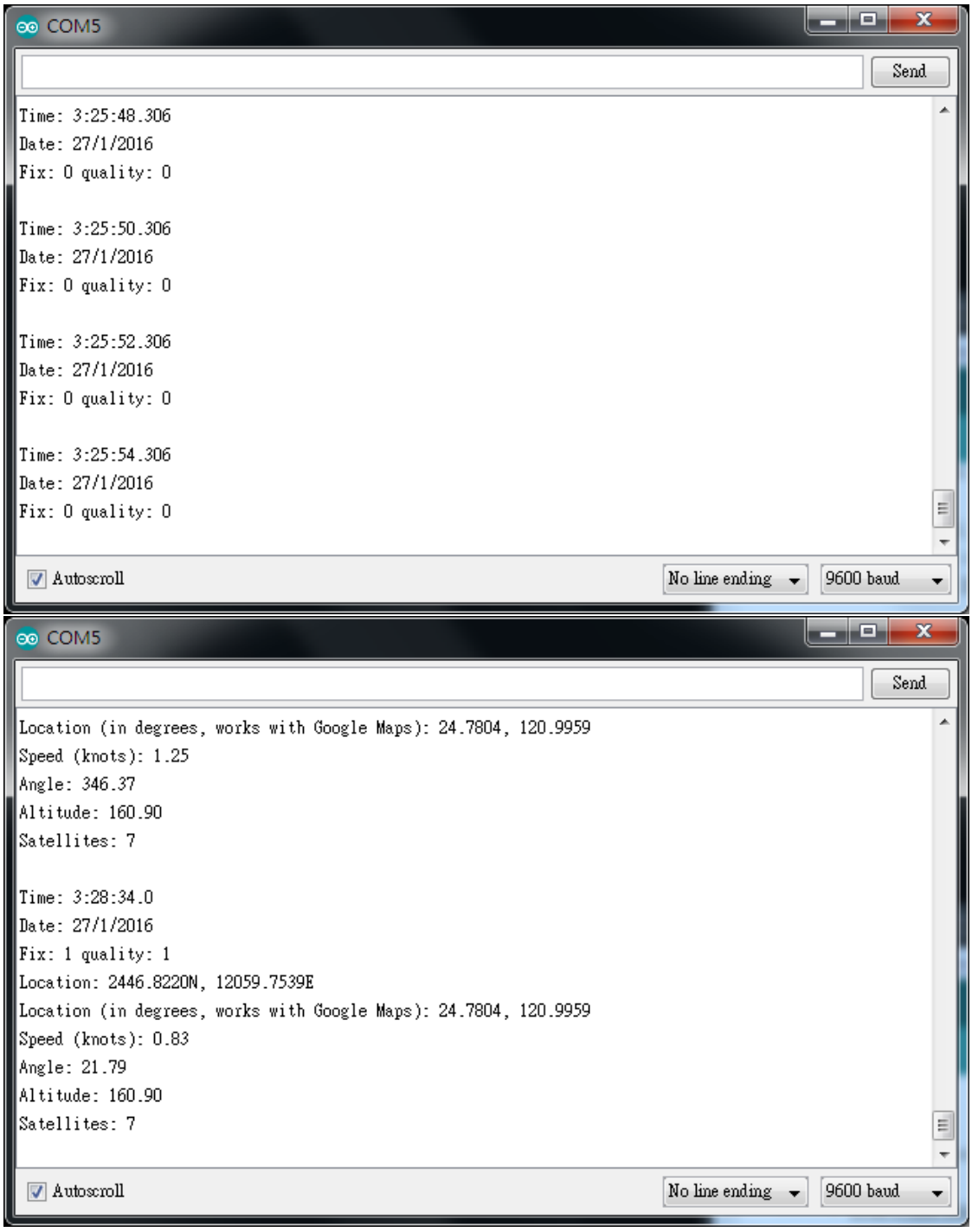


AMB23 Wiring Diagram:



Open the example in "Files" -> "Examples" -> "AmebaSoftwareSerial" -> "Adafruit_GPS_parsing".

Compile and upload to Ameba, then press the reset button.
The result will be output to Serial Monitor:



UART – Set Callback Function For UART Communications

Materials

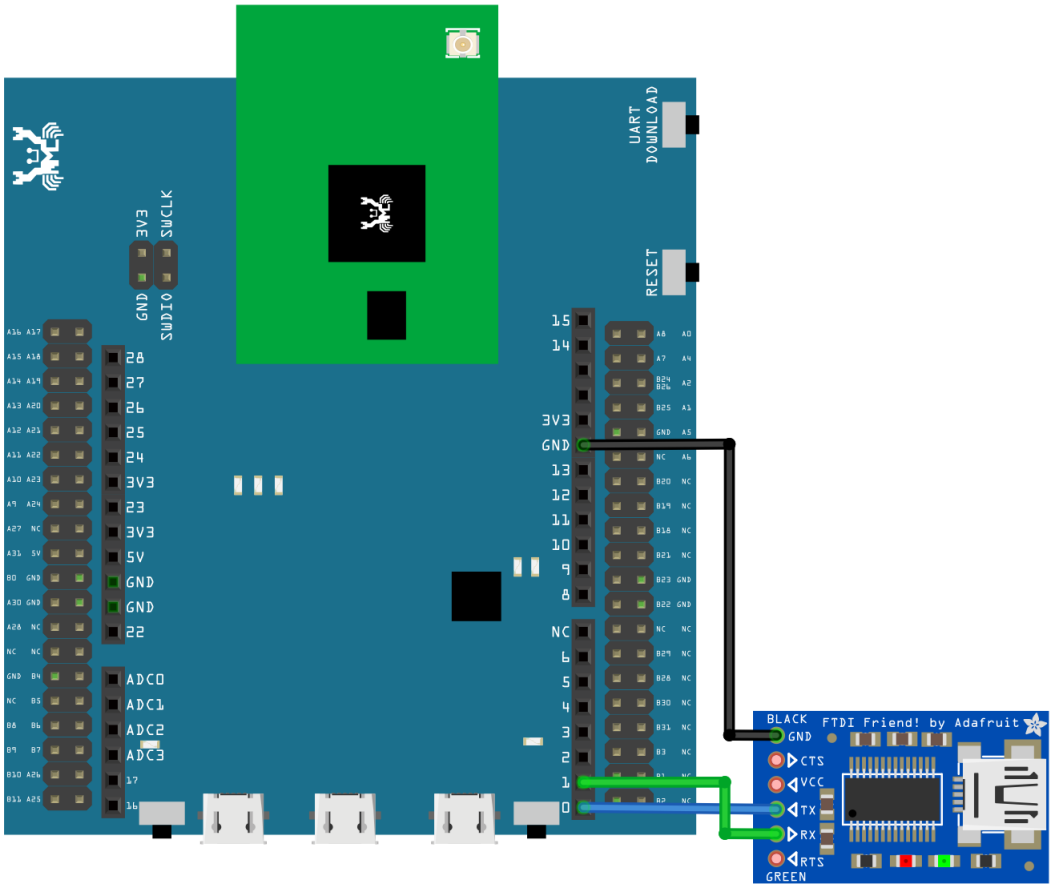
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- USB to TTL Adapter x 1

Example

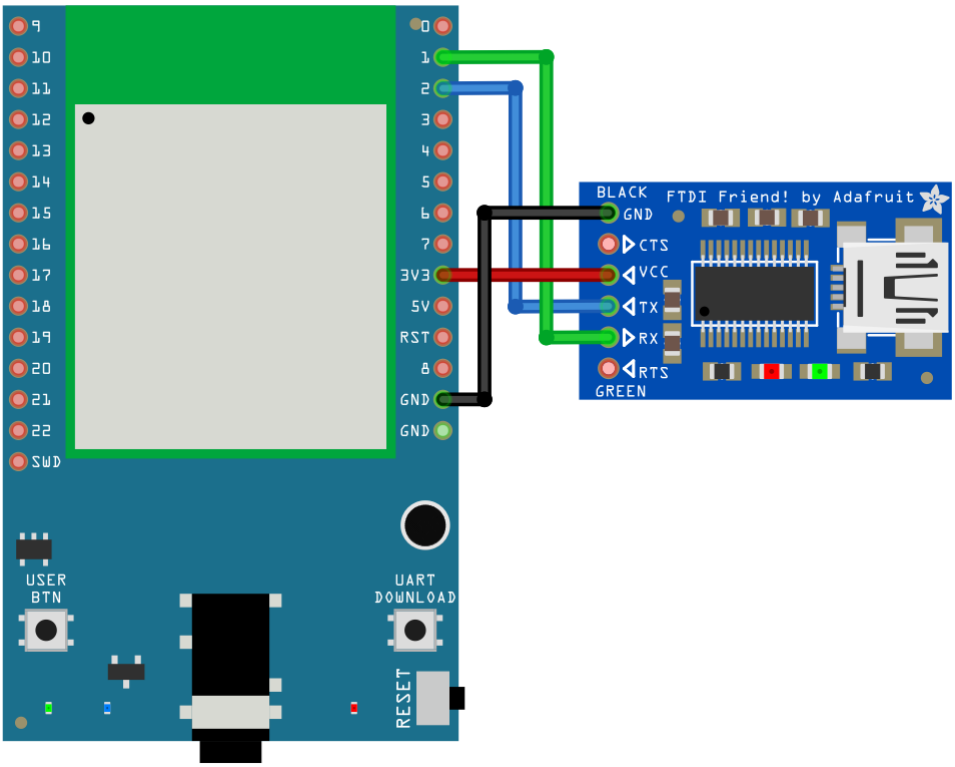
This example shows how to set a callback function for UART communication to process the UART data.

A USB to TTL adapter is required for this example. Ensure that you have the driver installed and connect it to the Ameba board as shown.

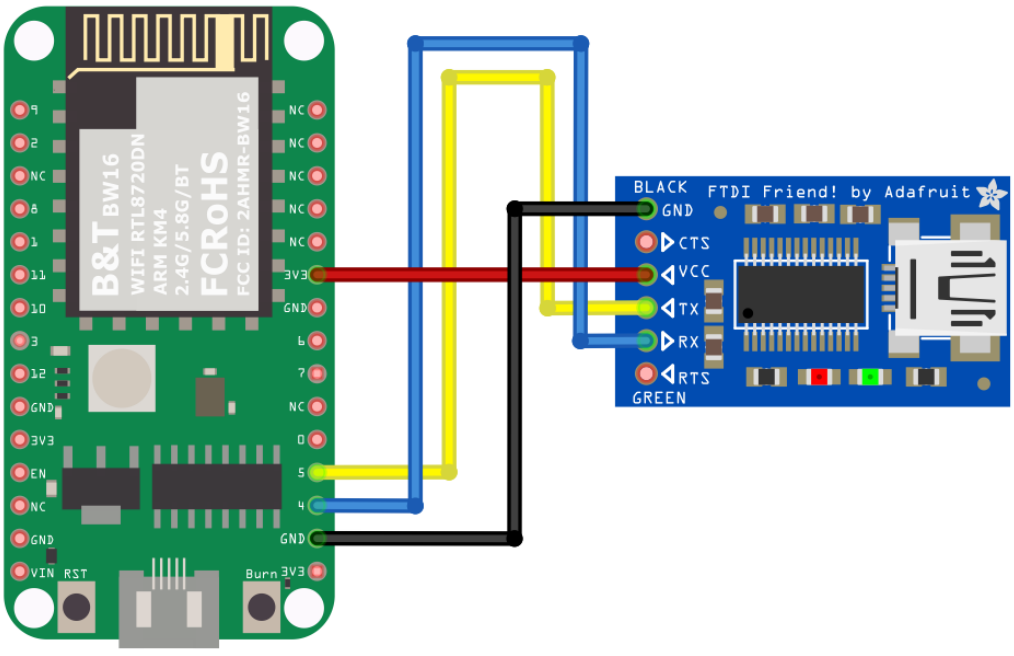
AMB21 / AMB22 Wiring Diagram:



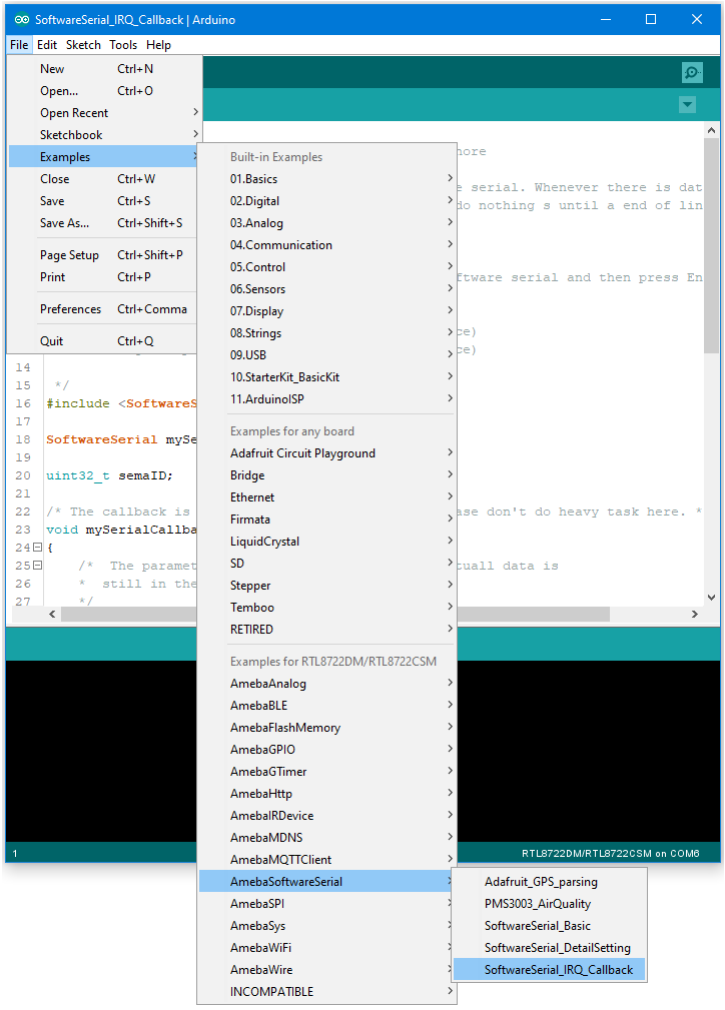
AMB23 Wiring Diagram:



BW16 Wiring Diagram:

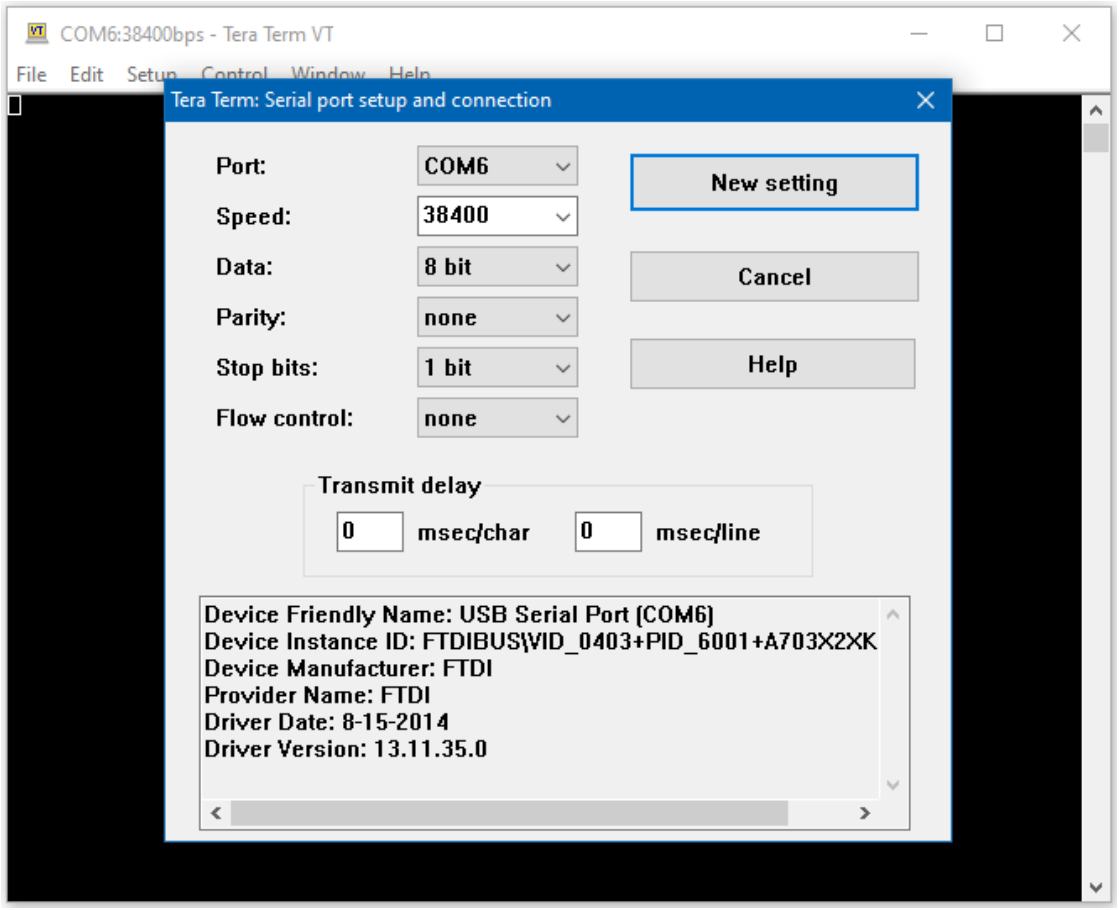


Open the example in "`File`" -> "`Examples`" -> "`AmebaSoftwareSerial`" -> "`SoftwareSerial_Irq_Callback`"

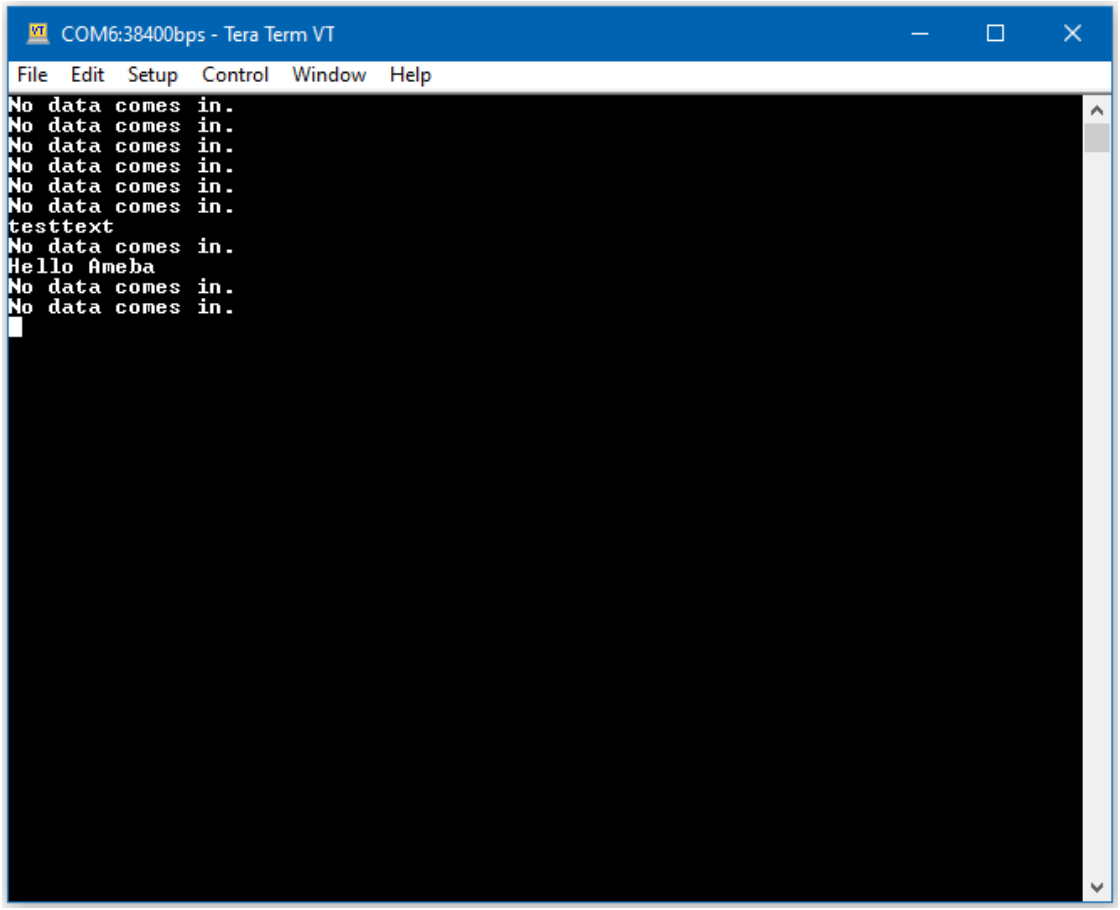


Upload the code and press the reset button on Ameba once the upload is finished.
Next, using a terminal program, such as TeraTerm or PuTTY, open a serial port and configure it according to the settings. Make sure the serial port number corresponds to the USB to TTL adapter.

- Speed: 38400
- Data: 8 bit
- Parity: none
- Stop bits: 1 bit
- Flow control: none



Once the serial port is open, type in the terminal and press the enter key, and you will see the corresponding output.



Code Reference

`mySerial.setAvailableCallback(mySerialCallback);` is used to set the function mySerialCallback as a callback function for software serial. When a new character is received, the callback function checks if the character corresponds to the enter key, and releases the semaphore if it is true, which in turn allows the main loop to print out all the previously received characters.

UART - PM2.5 Concentration in The Air

Preparation

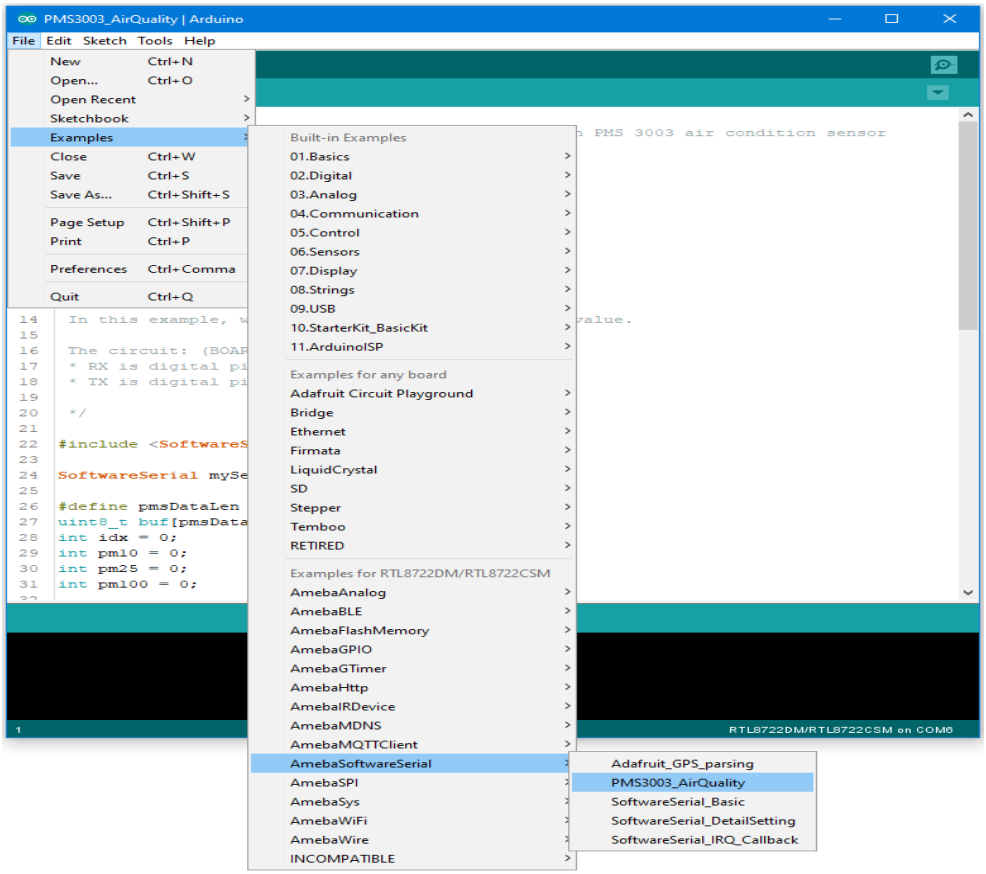
- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1
- PlanTower PMS3003 or PMS5003 x 1

Example

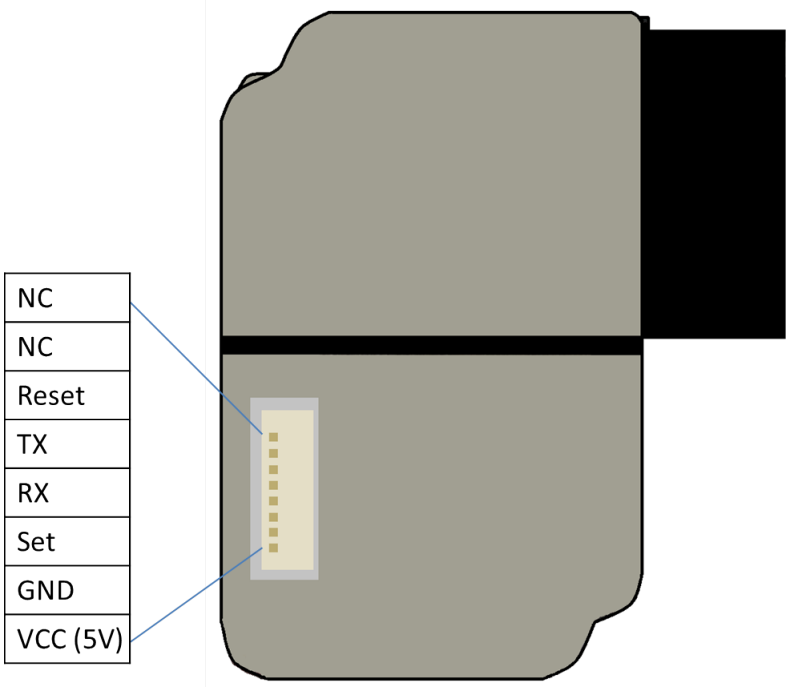
PMS3003 (or PMS5003) is a sensor of air quality, it can detect the concentration of those 0.3 to 10 micrometer particulate matters in the air. The sensor output its data via UART.

The PMS3003 (or PMS5003) sensor detects the concentration value of PM 1.0, PM 2.5, PM 10. Take PM 2.5 for example, it stands for the fine particles with a diameter of 2.5 micrometers or less.

Open the example in “File” -> “Examples” -> “AmebaSoftwareSerial” -> “PMS3003_AirQuality”



There are 8 pins in PMS3003:

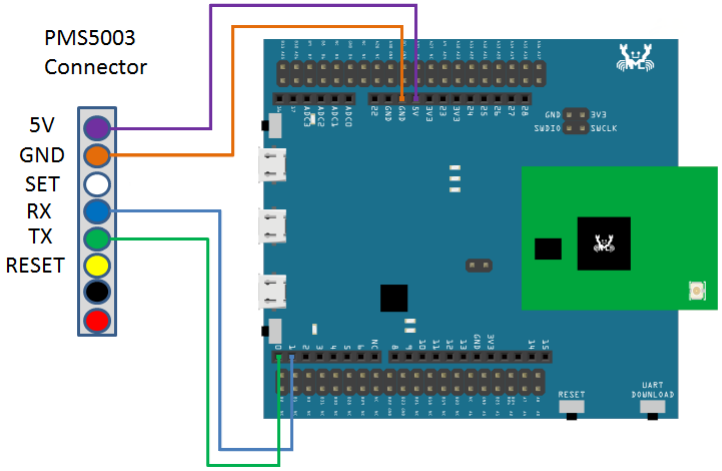


PMS3003 requires 5V power, but the working voltage of its IC is 3.3C. Therefore, the working voltage of Reset, TX, RX, Set are 3.3 as well. If the "Set" pin is pulled to high, the PMS3003 is put to operating mode. If the "Set" pin is pulled low, the PMS3003 is put to standby mode.

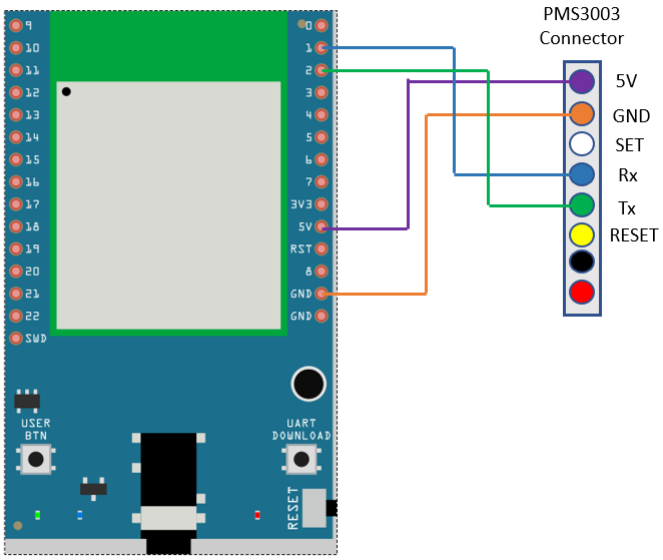
TX/RX pins are for UART connection. Under operating mode, PMS3003 output the data it reads continuously. Each data is of 32 byte, please refer to the following article for detailed data format information:

https://www.dfrobot.com/wiki/index.php?title=PM2.5_laser_dust_sensor_SKU:SEN0177 RTL8722

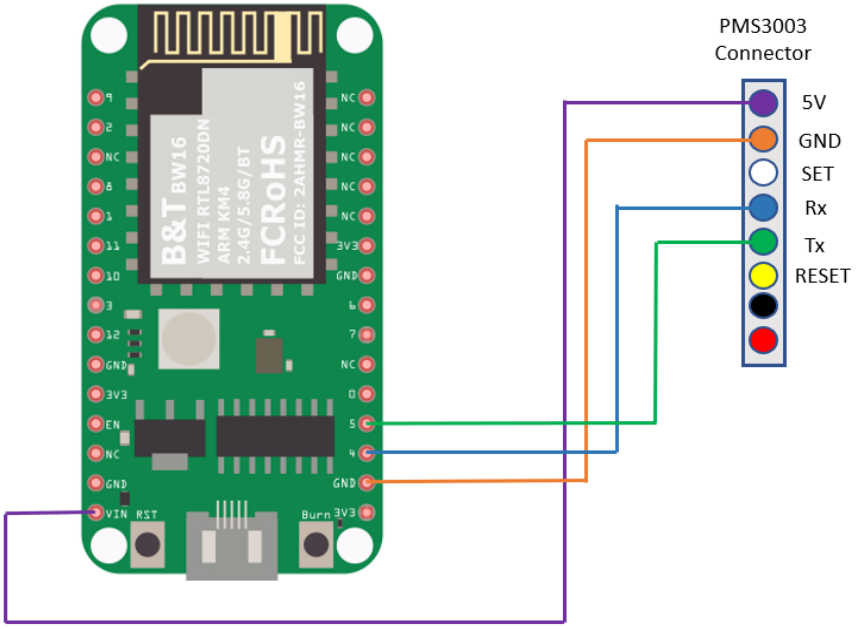
AMB21 / AMB22 Wiring Diagram:



AMB23 Wiring Diagram:

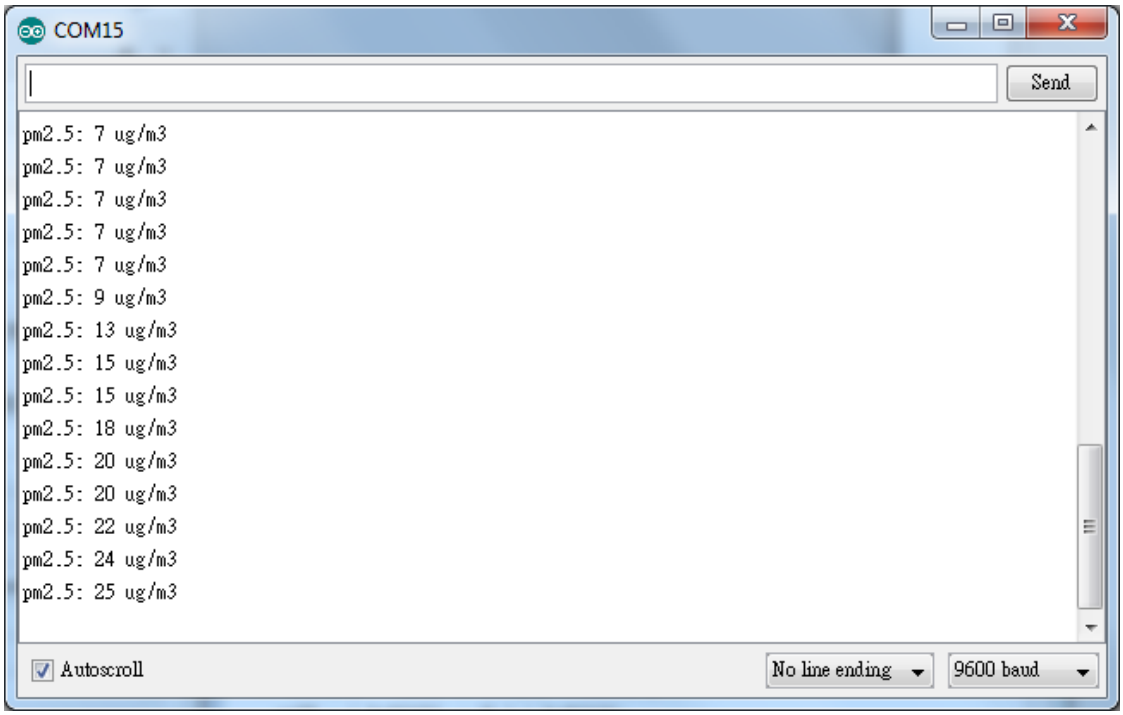


BW16 Wiring Diagram:



In this example, we do not use the “Set” and “Reset” pins.

Compile the code and upload it to Ameba. After pressing the Reset button, Ameba starts to output the PM 2.5 data to serial monitor.



Watchdog - Simple WDG Timer

Preparation

- AmebaD [AMB21 / AMB22 / AMB23 / BW16] x 1

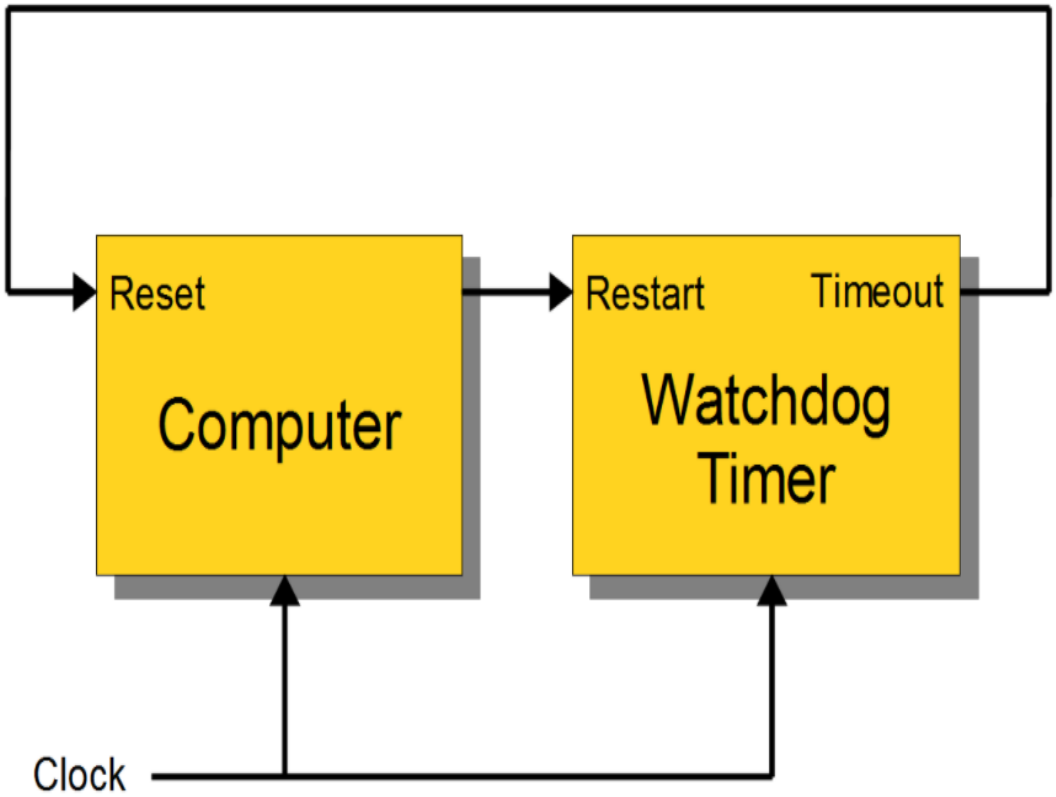
Example

In this example, we will use this simple watchdog timer example runs on the Ameba RTL8722 module to illustrate how to use the watchdog API. Before we get into the details of the example, let's briefly go through the definition of Watchdog as well as it's working principles.

Watchdog

Watchdog Timer (WDT) is a hardware timer that is used to detect the occurrence of a software fault, then automatically generates a system reset or a watchdog interrupt on the expiry of a programmed period.

In layman terms, imagine in the situation while your micro-controller is confused in an infinity loop, or any case like the micro-controller hang while performing some tasks. The normal troubleshooting method would be to press the reset button and jump out of the infinity loop. However, is it practically impossible to do press on the button all time, therefore, the watchdog timer that embedded inside the micro-controller would help with this situation.



Feed the Dog

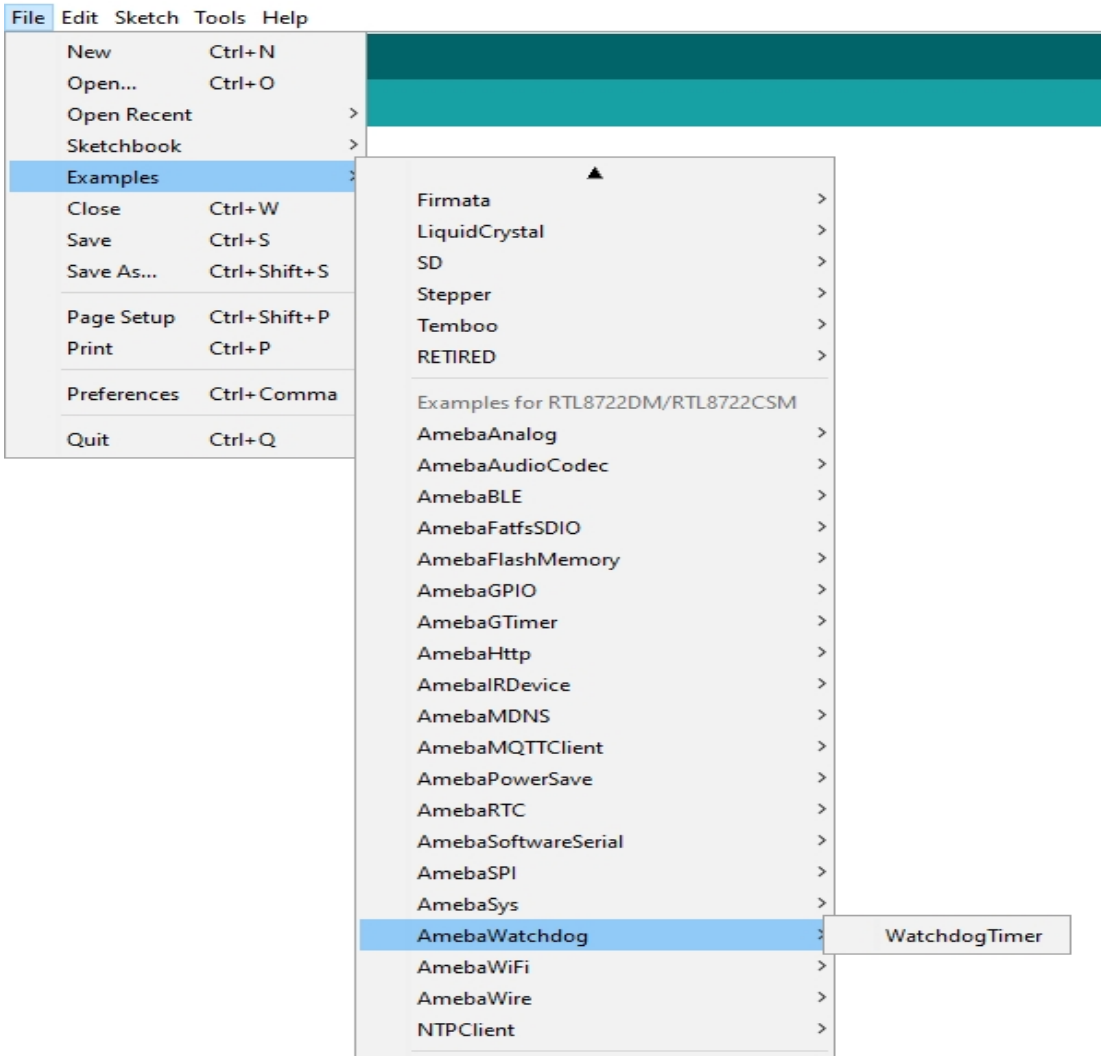
If you have a dog in your home. You need to feed that dog at a regular interval. if you can't feed one day, it will bite

you! And likewise, this is the working logic behind the watchdog timer.

In our example, we created 2 tasks that contain some loop that runs repeatedly, one is called “Small_Task” and the other is called “Big_Task”. We are enabling the watchdog timer is loaded with an initial value (5 seconds) greater than the total delay in the “Small_Task”, but shorter than the “Big_Task”.
For the successful case, the watchdog is being refreshed/feed within 5 seconds, however, for the failed case, the loop is under processing and the watchdog is not being fresh after 5 seconds, which triggers the watchdog (dog barks), an interrupt is generated to reset the processor. Likewise, the watchdog timer protects the micro-controller from the hanging case.

Then we move to the coding part for this example, for this example, you will only need the RTL8722CSM/RTL8722DM/RTL8722DM MINI Board itself.

Firstly, make sure the correct Ameba development board is selected in Arduino IDE: “`Tools`” -> “`Board`” -> “`RTL8722CSM/RTL8722DM`” (or “`RTL8722DM MINI`”). Then open the “Watchdog Timer” example in “`File`” -> “`Examples`” -> “`AmebaWatchdog`” -> “`Watchdog Timer`”:



Upon successfully upload the sample code, open the serial monitor, and press the reset button. You will find that the "Small_Task" can refresh the watchdog within the 5 seconds (initialized in the watchdog timer). However, the "Big_Task" will not be able to refresh the watchdog within 5 seconds, which the watchdog "barks" then the microcontroller reset.


```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
```

COM10

```
#calibration_ok:[2:19:11]
......doing small task......
Small_Task finished refresh watchdog.
......doing big task, up to 10......
doing dummy task #1
doing dummy task #2
doing dummy task #3
doing dummy task #4
doing dummy task #5
watchdog barks!!!
doing dummy task #6
doing dummy task #7
doing dummy task #8
doing dummy task #9
doing dummy task #10
Big_Task finished refresh watchdog.
```

Community Examples

Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards.
To make contributions, please visit our official GitHub Wiki page.

Tip: Welcome to share your examples under the **Community Examples** section if you have completed a project using the Ameba boards

1.3.3 API Documents

RTL8722DM ARDUINO Online API Documents

Analog

Class AmebaServo

AmebaServo Class

Description

Defines a class of manipulating servo motors connected to Arduino pins.

Syntax

class AmebaServo

Members

Public Constructors	
AmebaServo::AmebaServo	Constructs an AmebaServo object.
Public Methods	
AmebaServo::attach	Attach the given pin to the next free channel.
AmebaServo::detach	Detach the servo.
AmebaServo::write	Write value, if the value is < 200 it's treated as an angle, otherwise as pulse-width in microseconds.
AmebaServo::writeMicroseconds	Write pulse width in microseconds.
AmebaServo::read	Output current pulse width as an angle between 0 and 180 degrees.
AmebaServo::readMicroseconds	Output current pulse width in microseconds for this servo.
AmebaServo::attached	Check if the servo is attached.

AmebaServo::attach

Description

Attach the given pin to the next free channel, sets pinMode (including minimum and maximum values for writes), returns channel number, or 0 if failure.

Syntax

```
uint8_t attach(int pin);

uint8_t attach(int pin, int min, int max);
```

Parameters

pin: The Arduino pin number to be attached.

min: Minimum values for writes.

max: Maximum values for writes.

Returns

The function returns channel number or 0

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree.

Listing 3: ServoSweep.ino

```
/* Sweep
by BARRAGAN < http://barraganstudio.com >
This example code is in the public domain.
modified 8 Nov 2013
by Scott Fitzgerald
http://www.arduino.cc/en/Tutorial/Sweep
refined 2016/03/18 by Realtek
*/

#include "AmebaServo.h"

// create servo object to control a servo
// 4 servo objects can be created correspond to PWM pins

AmebaServo myservo;

// variable to store the servo position
int pos = 0;

void setup() {
#if defined(BOARD_RTL8195A)
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#elif defined(BOARD_RTL8710)
    // attaches the servo on pin 13 to the servo object
    myservo.attach(13);
#elif defined(BOARD_RTL8721D)
    // attaches the servo on pin 8 to the servo object
    myservo.attach(8);
#else
    // attaches the servo on pin 9 to the servo object
    myservo.attach(9);
#endif
}

void loop() {
    // goes from 0 degrees to 180 degrees in steps of 1 degree
    for (pos = 0; pos <= 180; pos += 1) {
    // tell servo to go to position in variable 'pos'
    myservo.write(pos);
    // waits 15ms for the servo to reach the position
    delay(15);
    }
    // goes from 180 degrees to 0 degrees
    for (pos = 180; pos >= 0; pos -= 1) {
        // tell servo to go to position in variable 'pos'
        myservo.write(pos);
        // waits 15ms for the servo to reach the position
```

(continues on next page)

(continued from previous page)

```
49          delay(15);
50      }
51  }
```

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::detach

Description

Detach the servo.

Syntax

void AmebaServo::detach(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::write

Description

Write an integer value to the function, if the value is < 200, it's being treated as an angle, otherwise as pulse-width in microseconds.

Syntax

void AmebaServo::write(int value);

Parameters

value: The value < 200 its treated as an angle; otherwise as pulse width in microseconds.

Returns

The function returns nothing.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::writeMicroseconds

Description

Write pulse width to the servo in microseconds.

Syntax

void AmebaServo::writeMicroseconds(int value);

Parameters

value: Write value the pulse width in microseconds.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::read

Description

The function reads current pulse width and returns as an angle between 0 and 180 degrees.

Syntax

int AmebaServo::read(void);

Parameters

The function requires no input parameter.

Returns

The pulse width as an angle between 0 ~ 180 degrees.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::readMicroseconds

Description

The function returns a Boolean value "true" if this servo is attached, otherwise returns "false".

Syntax

int AmebaServo::readMicroseconds(void);

Parameters

The function requires no input parameter.

Returns

The function returns current servo pulse width in microseconds.

Example Code

NA

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AmebaServo::attached

Description

It returns true if this servo is attached, otherwise false.

Syntax

bool AmebaServo::attached(void);

Parameters

The function requires no input parameter.

Returns

The function returns a Boolean value as true or false.

Example Code

Example: ServoSweep

The code demos servo motor sweeping from 0 degrees to 180 degrees then sweep back to 0 degrees in the step of 1 degree. Please refer to code in "AmebaServo:: attach" section.

Notes and Warnings

Every time must include the header file "AmebaServo.h" in front of the project to use the class function.

AudioCodec

Class AudioCodec

Description

A class used for general control and management of the hardware audio codec functions.

Syntax

class AudioCodec

Members

Public Constructors

The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named Codec.

Public Methods

AudioCodec::begin	Configure and start the audio codec for transmit and receive operation
AudioCodec::end	Stop all audio codec operation
AudioCodec::getBufferPageSize	Get the byte size of a single page of the audio codec buffer
AudioCodec::setSampleRate	Configure the audio codec transmit and receive sampling rate
AudioCodec::setBitDepth	Configure the audio codec transmit and receive bit depth (bits per sample)
AudioCodec::setChannelCount	Configure the audio codec transmit and receive channel count
AudioCodec::setInputMicType	Configure for analog or digital input microphone type
AudioCodec::setInputLRMux	Configure input left right channel multiplexing
AudioCodec::setDMicBoost	Configure boost gain for digital microphone input
AudioCodec::setAMicBoost	Configure boost gain for analog microphone input
AudioCodec::setADCGain	Configure gain of ADC used to acquire analog input
AudioCodec::muteInput	Mute input audio data stream
AudioCodec::setOutputVolume	Configure output audio volume
AudioCodec::muteOutput	Mute output audio
AudioCodec::writeAvaliable	Check for free buffer page available for data write
AudioCodec::writeDataPage	Write audio data to an available buffer page
AudioCodec::readAvaliable	Check for buffer page with new data available for read
AudioCodec::readDataPage	Read audio data from a ready buffer page
AudioCodec::setWriteCallback	Set a callback function to be notified when a free buffer page is available for write
AudioCodec::setReadCallback	Set a callback function to be notified when a buffer page with new data is available for read

AudioCodec::begin

Description

Configure and start the audio codec for transmit and receive operation.

Syntax

void begin(bool input, bool output);

Parameters

input: enable audio codec data input

output: enable audio codec data output

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::end

Description

Stop all audio codec operation.

Syntax

void end();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::getBufferPageSize

Description

Get the byte size of a single page of the audio codec buffer.

Syntax

uint32_t getBufferPageSize();

Parameters

The function requires no input parameter.

Returns

The size of a audio codec buffer page, in number of bytes.

Example Code

NA

Notes and Warnings

The AudioCodec class includes a transmit and receive buffer to store audio sample data while transferring to and from the DAC output and ADC input. The buffer is divided into pages of fixed size, and audio data can be read and written one page at a time. Depending on the configured bit depth (bits per audio sample) and channel count, a buffer page may contain a different number of audio samples.

AudioCodec::setSampleRate

Description

Configure the audio codec transmit and receive sampling rate.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: desired audio codec sampling rate in Hz. Default value of 48000. Supported values: 8000, 16000, 32000, 44100, 48000, 88200, 96000.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

High sample rates above 48000Hz will require frequent buffer reads and writes to keep up with the large amount of data input and output. If there is insufficient processing time dedicated to this task, audio quality will be degraded.

AudioCodec::setBitDepth

Description

Configure the audio codec transmit and receive bit depth (bits per sample).

Syntax

void setBitDepth(uint8_t bitDepth);

Parameters

bitDepth: desired number of bits per sample. Default value of 16. Supported values: 8, 16, 24.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Setting a bit depth of 24 bits per sample will require 32 bits (4 bytes) of buffer space for storing each sample, with the most significant byte ignored.

AudioCodec::setChannelCount

Description

Configure the audio codec transmit and receive channel count.

Syntax

void setChannelCount(uint8_t monoStereo);

Parameters

monoStereo: number of channels. Default value of 1. Supported values: 1, 2.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setInputMicType

Description

Configure for analog or digital input microphone type.

Syntax

Void setInputMicType(Mic_Type micType);

Parameters

micType: Input microphone type. Default value ANALOGMIC. Valid values:

- ANALOGMIC – microphone with an analog output

- PDMMIC – digital microphone with a PDM output

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

For analog single-ended output, connect to PA_4 for the left channel and PA_2 for the right channel.

For digital PDM output, connect the PDM clock to PB_1 and PDM data to PB_2.

AudioCodec::setInputLRMux

Description

Configure input left right channel multiplexing.

Syntax

void setInputLRMux(uint32_t mux);

Parameters

mux: desired left right audio channel multiplexing setting. Default value RX_CH_LR. Valid values:

- RX_CH_LR
- RX_CH_RL
- RX_CH_LL
- RX_CH_RR

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In mono channel mode, both RX_CH_LR and RX_CH_LL will result in the audio codec sampling input data from the left channel microphone. Similarly, both RX_CH_RL and RX_CH_RR will result in the audio codec sampling input data from the right channel microphone.

In stereo channel mode, RX_CH_RL will switch the positions of input data sampled from the microphones. RX_CH_RR and RX_CH_LL will result in duplicated samples from the right and left microphones respectively.** **

AudioCodec::setDMicBoost

Description

Configure boost gain for digital microphone input.

Syntax

void setDMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel digital microphone input

rightBoost: boost gain for right channel digital microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 12dB
- 2 : 24dB
- 3 : 36dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setAMicBoost

Description

Configure boost gain for analog microphone input.

Syntax

void setAMicBoost(uint32_t leftBoost, uint32_t rightBoost);

Parameters

leftBoost: boost gain for left channel analog microphone input

rightBoost: boost gain for right channel analog microphone input

Valid boost gain values:

- 0 : 0dB
- 1 : 20dB
- 2 : 30dB
- 3 : 40dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this function if additional gain is required after using setADCGain function.

AudioCodec::setADCGain

Description

Configure gain of ADC used to acquire analog input.

Syntax

void setADCGain(uint32_t leftGain, uint32_t rightGain);

Parameters

leftGain: Gain for left channel ADC

rightGain: Gain for right channel ADC

Valid value range is from 0x00 to 0x7f. Gain increases by 0.375dB for every increment in value:

- 0x00 : -17.625dB
- 0x01 : -17.25dB
- 0x2f : 0dB
- 0x30 : 0.375dB
- 0x7f : 30dB

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::muteInput

Description

Mute input audio data stream.

Syntax

void muteInput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel input, 0 to unmute

rightMute: 1 to mute right channel input, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::setOutputVolume

Description

Configure output audio volume.

Syntax

void setOutputVolume(uint8_t leftVol, uint8_t rightVol);

Parameters

leftVol: left channel output volume

rightVol: right channel output volume

Valid value ranges from 0 to 100, corresponding to a volume of -65.625dB to 0dB.

Returns

The function returns nothing.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::muteOutput

Description

Mute output audio.

Syntax

void muteOutput(uint8_t leftMute, uint8_t rightMute);

Parameters

leftMute: 1 to mute left channel output, 0 to unmute

rightMute: 1 to mute right channel output, 0 to unmute

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

AudioCodec::writeAvaliable

Description

Check for free buffer page available for data write.

Syntax

bool writeAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page that is available for writing data into. Returns false if all buffer pages are full.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::writeDataPage

Description

Write audio data to an available buffer page.

Syntax

uint32_t writeDataPage(int8_t* src, uint32_t len);

uint32_t writeDataPage(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing audio samples to write to audio codec.

len: number of audio samples in array.

Returns

The function returns the number of audio samples written to the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readAvaliable

Description

Check for buffer page with new data available for read.

Syntax

bool readAvaliable();

Parameters

The function requires no input parameter.

Returns

Returns true if there is a buffer page with new data that is ready for reading data from. Returns false if all buffer pages are empty.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::readDataPage

Description

Read audio data from a ready buffer page.

Syntax

uint32_t readDataPage(int8_t* dst, uint32_t len);

uint32_t readDataPage(int16_t* dst, uint32_t len);

Parameters

dst: pointer to array to contain audio samples read from audio codec.

len: number of audio samples to read.

Returns

The function returns the number of audio samples read from the audio codec.

Example Code

Example: BasicInputOutput

Notes and Warnings

AudioCodec::setWriteCallback

Description

Set a callback function to be notified when a free buffer page is available for write.

Syntax

void setWriteCallback(**void** (*writeCB)(**void*));

Parameters

writeCB: function to be called when a buffer page becomes available for data write. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec finishes outputting the data in a buffer page.

AudioCodec::setReadCallback

Description

Set a callback function to be notified when a buffer page with new data is available for read.

Syntax

void setReadCallback(**void** (*readCB)(**void*));

Parameters

readCB: function to be called when a buffer page with new data becomes available for data read. Takes no arguments and returns nothing

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

After starting the audio codec with AudioCodec::begin(), the callback function will be called each time the audio codec fills up a buffer page with newly acquired audio samples.

Class FFT

Description

A class used for performing FFT calculations with real-number inputs and outputs.

Syntax

class FFT

Members

Public Constructors

FFT::FFT	Create an instance of the FFT class

Public Methods

FFT::setWindow	Configure the window function used in FFT calculations
FFT::calculate	Calculate FFT for an input array of values
FFT::getFrequencyBins	Get the FFT output frequency bins
FFT::getFFTSize	Get the size of FFT output for a given input size

FFT::FFT

Description

Create a FFT class object.

Syntax

void FFT();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

FFT::setWindow

Description

Configure the window function used in FFT calculations.

Syntax

void setWindow(FFTWindow_t window, uint16_t sampleCount);

Parameters

window: The window function to be used in FFT calculations. Valid values: None, Hann, Hamming.

sampleCount: Number of sample datapoints in the input.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings

The window function is used to reduce the effects of discontinuities that occur when the input signal has frequencies that do not fit an integer number of periods in the sample datapoints.

More information on FFTs and window functions can be seen at:

https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf

https://en.wikipedia.org/wiki/Window_function

FFT::Calculate

Description

Calculate FFT for an input array of values.

Syntax

void calculate(float* inputBuf, float* outputBuf, uint16_t sampleCount);

void calculate(int16_t* inputBuf, float* outputBuf, uint16_t sampleCount);

Parameters

inputBuf: pointer to an array of sampleCount size, containing input sample datapoints, in float or uint16_t format.

outputBuf: pointer to a float array of sampleCount/2 size, for containing FFT output.

sampleCount: number of sample datapoints in the input array, valid values: 16, 32, 64, 128, 256, 512, 1024, 2048.

Returns

The function returns nothing.

Example Code

Example:FFT

Notes and Warnings

Large sample counts will require a longer time for FFT calculations, but will also return a result with higher frequency resolution.

FFT::getFrequencyBins

Description

Get the FFT output frequency bins.

Syntax

void getFrequencyBins(uint16_t* outputBuf, uint16_t sampleCount, uint32_t sampleRate);

Parameters

outputBuf: pointer to a uint16_t array of sampleCount/2 size, for containing the calculated center frequency of each FFT output element.

Returns

The function returns nothing.

Example Code

Example: FFT

Notes and Warnings NA

—

FFT::getFFTSize

Description

Get the size of FFT output for a given input size.

Syntax

uint16_t getFFTSize(uint16_t sampleCount);

Parameters

sampleCount: number of input sample datapoints.

Returns

The function returns the FFT output size for the given sampleCount, which is sampleCount/2.

Example Code

NA

Notes and Warnings NA

Class PlaybackWav

Description

A class used for control and playback of .wav file format audio data.

Syntax

class PlaybackWav

Members

Public Constructors

PlaybackWav::PlaybackWav	Create an instance of the PlaybackWav class

Public Methods

PlaybackWav::openFile	Open a .wav file for playback
PlaybackWav::closeFile	Close a previously opened file
PlaybackWav::fileOpened	Check if a .wav file is already opened
PlaybackWav::getSampleRate	Get the sample rate of the .wav file
PlaybackWav::getChannelCount	Get the number of audio channels in the .wav file
PlaybackWav::getBitDepth	Get the bit depth of each sample in the .wav file
PlaybackWav::getLengthMillis	Get the playback length of the .wav file in milliseconds
PlaybackWav::getPositionMillis	Get the current playback position in milliseconds
PlaybackWav::setPositionMillis	Set the current playback position in milliseconds
PlaybackWav::millisToBytes	Convert a playback duration to equivalent number of bytes
PlaybackWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
PlaybackWav::readAudioData	Read audio data from the .wav file

---

PlaybackWav::PlaybackWav

Description

Create a PlaybackWav class object.

Syntax

void PlaybackWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordPlaybackWav

Notes and Warnings

NA

PlaybackWav::getSampleRate

Description

Get the sample rate of the .wav file.

Syntax

uint32_t getSampleRate(void);

Parameters

The function requires no input parameter.

Returns

The function returns sampling rate encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getChannelCount

Description

Get the number of audio channels in the .wav file.

Syntax

uint16_t getChannelCount(void);

Parameters

The function requires no input parameter.

Returns

The function returns channel count encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getBitDepth

Description

Get the bit depth of each sample in the .wav file.

Syntax

uint16_t getBitDepth(void);

Parameters

The function requires no input parameter.

Returns

The function returns bit depth encoded in the .wav file header.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getLengthMillis

Description

Get the playback length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the total playback length of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::getPositionMillis

Description

Get the current playback position in milliseconds.

Syntax

uint32_t getPositionMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current playback position of the currently open .wav file in milliseconds.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

PlaybackWav::setPositionMillis

Description

Set the current playback position in milliseconds.

Syntax

void setPositionMillis(uint32_t pos);

Parameters

pos: The desired playback position expressed in milliseconds.

Returns

The function returns nothing.

Example Code

Example: PlaybackWavFile

Notes and Warnings

Any changes to playback position will only take effect on the next call to PlaybackWav::readAudioData. If the desired playback position is beyond the total playback length of the file, the playback position will be set to the end of file, and no audio data will be output on subsequent data reads.

PlaybackWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

PlaybackWav::readAudioData

Description

Read audio data from the .wav file.

Syntax

- `uint32_t readAudioData(int8_t* dst, uint32_t len);`
- `uint32_t readAudioData(int16_t* dst, uint32_t len);`

Parameters

- `dst`: pointer to array to store data read from .wav file.
- `len`: number of audio samples to read from .wav file.

Returns

The function returns number of audio samples read.

Example Code

Example: PlaybackWavFile

Notes and Warnings

NA

Class RecordWav

Description

A class used for control and recording of .wav file format audio data.

Syntax

class RecordWav

Members

Public Constructors

RecordWav:: RecordWav	Create an instance of the RecordWav class

Public Methods

RecordWav::openFile	Open a .wav file for playback
RecordWav::closeFile	Close a previously opened file
RecordWav::fileOpened	Check if a .wav file is already opened
RecordWav::setSampleRate	Get the sample rate of the .wav file
RecordWav::setChannelCount	Set the number of audio channels in the .wav file
RecordWav::setBitDepth	Set the bit depth of each sample in the .wav file
RecordWav::getLengthMillis	Get the current record length of the .wav file in milliseconds
RecordWav::millisToBytes	Convert a playback duration to equivalent number of bytes
RecordWav::bytesToMillis	Convert number of bytes to an equivalent playback duration
RecordWav::writeAudioData	Write audio data to the .wav file

RecordWav::RecordWav

Description

Create a RecordWav class object.

Syntax

void RecordWav(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::openFile

Description

Open a .wav file for recording.

Syntax

void openFile(const char* absFilepath);

Parameters

absFilepath: the filepath of the .wav file to open.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::closeFile

Description

Close a previously opened file.

Syntax

void closeFile(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

Any open .wav files should be closed after recording is complete, otherwise, loss of recorded audio data may occur.

RecordWav::fileOpened

Description

Check if a .wav file is already opened.

Syntax

bool fileOpened(void);

Parameters

The function requires no input parameter.

Returns

The function returns true if a .wav file is already open, false otherwise.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setSampleRate

Description

Set the recording sample rate of the .wav file.

Syntax

void setSampleRate(uint32_t sampleRate);

Parameters

sampleRate: The desired recording sample rate.

Returns

The function returns nothing.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

RecordWav::setChannelCount

Description

Set the number of recording audio channels in the .wav file.

Syntax

void setChannelCount(uint16_t channelCount);

Parameters

channelCount: number of recording audio channels.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::setBitDepth

Description

Set the recording bit depth of each sample in the .wav file.

Syntax

void setBitDepth(uint16_t bitDepth);

Parameters

bitDepth: number of bits per sample.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

RecordWav::getLengthMillis

Description

Get the current recorded length of the .wav file in milliseconds.

Syntax

uint32_t getLengthMillis(void);

Parameters

The function requires no input parameter.

Returns

The function returns the current recorded length of the currently open .wav file in milliseconds.

Example Code

NA

Notes and Warnings

NA

RecordWav::millisToBytes

Description

Convert a playback duration to equivalent number of bytes.

Syntax

uint32_t millisToBytes(uint32_t ms);

Parameters

ms: playback duration in milliseconds.

Returns

The function returns the number of bytes that is equivalent to the input playback duration, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::bytesToMillis

Description

Convert number of bytes to an equivalent playback duration.

Syntax

uint32_t bytesToMillis(uint32_t bytes);

Parameters

bytes: playback duration in number of bytes.

Returns

The function returns the time duration in milliseconds that is equivalent to the input number of bytes, converted using the current sample rate, number of channels and bit depth.

Example Code

NA

Notes and Warnings

NA

RecordWav::writeAudioData

Description

Write audio data to the .wav file.

Syntax

uint32_t writeAudioData(int8_t* src, uint32_t len); uint32_t writeAudioData(int16_t* src, uint32_t len);

Parameters

src: pointer to array containing data to write to .wav file. len: number of audio samples to write to .wav file.

Returns

The function returns number of audio samples written.

Example Code

Example: RecordWavFile

Notes and Warnings

NA

BLE

Class BLEAddr

BLEAddr Class

Description

A class used for managing Bluetooth addresses.

Syntax
class BLEAddr

Members

Public Constructors	
BLEAddr::BLEAddr	Constructs a BLEAddr object
Public Methods	
BLEAddr::str	Get the Bluetooth address represented as a formatted string
BLEAddr::data	Get the Bluetooth address represented as an integer array

BLEAddr::BLEAddr

Description
Constructs a BLEAddr object.

Syntax
BLEAddr::BLEAddr(void);
BLEAddr::BLEAddr(uint8_t (&addr)[6]);
BLEAddr::BLEAddr(const char* str);

Parameters
addr: An array of 6 bytes containing the desired Bluetooth address.
str: A character string representing the desired Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
When expressed as a string, the Bluetooth address should be written as 6 bytes in hexadecimal format, using a colon ":" to separate the bytes is acceptable (example – 00:11:22:33:EE:FF).

BLEAddr::str

Description
Get the Bluetooth address represented as a formatted string.

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a character string containing the hexadecimal representation of the Bluetooth address.

Example Code
Example: BLEScan

Notes and Warnings
The Bluetooth address expressed as a string will be written as 6 bytes in hexadecimal format, with a colon ":" separating the bytes (example – 00:11:22:33:EE:FF).

BLEAddr::data

Description
Get the Bluetooth address represented as an integer array.

Syntax
uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to a 6 byte array containing the Bluetooth address.

Example Code
NA

Notes and Warnings

The Bluetooth address is stored with MSB at array index [5].

Class BLEAdvert

BLEAdvert Class

Description
A class used for managing BLE advertising settings.

Syntax
class BLEAdvert

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configAdvert().

Public Methods	
BLEAdvert::updateAdvertParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEAdvert::startAdv	Start BLE advertising
BLEAdvert::stopAdv	Stop BLE advertising
BLEAdvert::setAdvType	Set the BLE advertising type
BLEAdvert::setMinInterval	Set the BLE advertising minimum interval
BLEAdvert::setMaxInterval	Set the BLE advertising maximum interval
BLEAdvert::setAdvData	Set BLE advertising data
BLEAdvert::setScanRspData	Set BLE scan response data

BLEAdvert::updateAdvertParams

Description
Update the lower Bluetooth stack with the current advertising settings.

Syntax
void updateAdvertParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Please use the other class member functions to set the BLE advertising parameters first before using this function.

BLEAdvert::startAdv

Description
Start BLE advertising.

Syntax
void startAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly start the BLE advertising process without first registering the necessary callback and handler functions. Call BLEDevice::beginPeripheral() to register the necessary functions and start advertising for the first time.

BLEAdvert::stopAdv

Description
Stop BLE advertising.

Syntax
void stopAdv(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in controlling and updating BLE advertising parameters. You should avoid using this function to directly stop the BLE advertising process. Call BLEDevice::end() to stop advertising and free up used resources.

BLEAdvert::setAdvType

Description
Set the BLE advertising type.

Syntax
void setAdvType(uint8_t advType);

Parameters
advType: the desired advertisement type. Valid values:
– 0 = GAP_ADTYPE_ADV_IND : connectable undirected advertisement
– 1 = GAP_ADTYPE_ADV_HDC_DIRECT_IND : connectable high duty cycle directed
– 2 = GAP_ADTYPE_ADV_SCAN_IND : scannable undirected advertisement
– 3 = GAP_ADTYPE_ADV_NONCONN_IND : Non-connectable undirected advertisement
– 4 = GAP_ADTYPE_ADV_LDC_DIRECT_IND : connectable low duty cycle directed advertisement

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this function with the GAP_ADTYPE_ADV_IND argument if connection requests should be allowed, and GAP_ADTYPE_ADV_NONCONN_IND if all connection requests should be rejected.

BLEAdvert::setMinInterval

Description
Set the minimum BLE advertising interval.

Syntax
void setMinInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement minimum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setMaxInterval

Description
Set the maximum BLE advertising interval.

Syntax
void setMaxInterval(uint16_t minInt_ms);

Parameters
minInt_ms: the desired advertisement maximum interval, expressed in milliseconds. The valid values for the interval are from 20ms to 10240ms.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLE advertisements will repeat with an interval between the set minimum and maximum intervals. Set a shorter interval for the BLE device to be discovered rapidly and set a longer interval to conserve power.

BLEAdvert::setAdvData

Description
Set BLE advertising data.

Syntax
void setAdvData(BLEAdvertData adData);
void setAdvData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.
size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

BLEAdvert::setScanRspData

Description
Set BLE scan response data.

Syntax
void setScanRspData(BLEAdvertData adData);
void setScanRspData(uint8_t* pData, uint8_t size);

Parameters
adData: scan response data formatted in a BLEAdvertData class object
pData: pointer to a byte array containing the required scan response data.

size: number of bytes the scan response data contains, maximum of 31 bytes.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
N/A

Class BLEAdvertData

BLEAdvertData Class

Description
A class used for managing BLE advertising data.

Syntax
class BLEAdvertData

Members

Public Constructors	
BLEAdvertData::BLEAdvertData	Constructs a BLEAdvertData object

Public Methods	
BLEAdvertData::clear	Clear all advertising data
BLEAdvertData::addData	Add binary advertising data
BLEAdvertData::addFlags	Add flags to advertising data
B LEAdvertData::addPartialServices	Add partial services to advertising data
BL EAdvertData::addCompleteServices	Add complete services to advertising data
BLEAdvertData::addAppearance	Add device appearance to advertising data
BLEAdvertData::addShortName	Add short device name to advertising data
BLEAdvertData::addCompleteName	Add complete device name to advertising data
BLEAdvertData::parseScanInfo	Parse advertising data received from a scan
BLEAdvertData::hasFlags	Check if received data includes advertising flags
BLEAdvertData::hasUUID	Check if received data includes UUIDs
BLEAdvertData::hasName	Check if received data includes device name
BLEAdvertData::hasManufacturer	Check if received data includes manufacturer data
BLEAdvertData::getAdvType	Get advertising type of received data
BLEAdvertData::getAddrType	Get Bluetooth address type of received data
BLEAdvertData::getAddr	Get Bluetooth address of received data
BLEAdvertData::getRSSI	Get RSSI of received data
BLEAdvertData::getFlags	Get advertising flags of received data
BLEAdvertData::getServiceCount	Get number of advertised services in received data
BLEAdvertData::getServiceList	Get array of advertised services in received data
BLEAdvertData::getName	Get advertised device name in received data
BLEAdvertData::getTxPower	Get advertised transmission power in received data
BLEAdvertData::getAppearance	Get advertised device appearance in received data
BLEAdvertData::getManufacturer	Get advertised manufacturer in received data
BLEAdver tData::getManufacturerDataLength	Get length of manufacturer data in received data
BL EAdvertData::getManufacturerData	Get advertised manufacturer data in received data

BLEAdvertData::BLEAdvertData

Description
Constructs a BLEAdvertData object.

Syntax
BLEAdvertData::BLEAdvertData(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class is used for managing BLE advertising data for two primary uses. First is to assemble advertising data for broadcasting as advertising packets. Second is to process and split up the advertising data received from a scan into separate types.

BLEAdvertData::clear

Description
Clear all advertising data currently saved in class object.

Syntax
void clear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addData

Description
Add binary advertising data.

Syntax
void addData(const uint8_t* data, uint8_t size);

Parameters
data: pointer to array containing desired advertising data.
size: number of bytes in array.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
This function is provided for flexibility in adding BLE advertising data. Other functions should be used for adding advertising data if possible, as this function does not perform any checks on the validity of the data.

BLEAdvertData::addFlags

Description
Add flags to advertising data.

Syntax
uint8_t addFlags(uint8_t flags);

Parameters
flags: desired flags to add to advertising data. Valid values:
– GAP_ADTYPE_FLAGS_LIMITED
– GAP_ADTYPE_FLAGS_GENERAL
– GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_CONTROLLER
– GAP_ADTYPE_FLAGS_SIMULTANEOUS_LE_BREDR_HOST

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::addPartialServices

Description
Add partial list of service UUIDs to advertising data.

Syntax
uint8_t addPartialServices(BLEUUID uuid);

Parameters
uuid: the desired UUID contained in BLEUUID class object.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::addCompleteServices

Description
Add complete list of service UUIDs to advertising data.

Syntax
uint8_t addCompleteServices(BLEUUID uuid);
uint8_t addCompleteServices(uint8_t uuidBitLength);

Parameters
uuid: the desired UUID contained in BLEUUID class object.
uuidBitLength: UUID bit length for which a blank entry is to be added. Valid values: 16, 32, 128.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
uuidBitLength is used when it is desired to add a blank entry to the advertisement data, used to indicate that no services with UUIDs of a certain length are available.

BLEAdvertData::addAppearance

Description
Add device appearance to advertising data.

Syntax
uint8_t addAppearance(uint16_t appearance);

Parameters
appearance: the desired device appearance.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for a full list of device appearance values.

BLEAdvertData::addShortName

Description
Add shortened device name to advertising data.

Syntax
uint8_t addShortName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
NA

Notes and Warnings

NA

BLEAdvertData::addCompleteName

Description
Add complete device name to advertising data.

Syntax
uint8_t addCompleteName(const char* str);

Parameters
str: character string containing desired device name.

Returns
Current total size of advertising data.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLEAdvertData::parseScanInfo

Description
Parse advertising data received from a scan.

Syntax
void parseScanInfo(T_LE_CB_DATA *p_data);

Parameters
p_data: pointer to advertising data received from a Bluetooth scan.

Returns
The function returns nothing.

Example Code

Example: BLEBatteryClient

Notes and Warnings
Advertising data fields of parsed receive data can be access using member functions starting with “has” and “get”.

BLEAdvertData::hasFlags

Description
Check if received data includes advertising flags.

Syntax
bool hasFlags(void);

Parameters
The function requires no input parameter.

Returns
True if flags are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasUUID

Description
Check if received data includes service UUIDs.

Syntax
bool hasUUID(void);

Parameters
The function requires no input parameter.

Returns

True if service UUIDs are present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::hasName

Description
Check if received data includes device name.

Syntax
bool hasName(void);

Parameters
The function requires no input parameter.

Returns
True if device name is present in received advertising data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::hasManufacturer

Description
Check if received data includes manufacturer specific data.

Syntax
bool hasManufacturer(void);

Parameters

The function requires no input parameter.

Returns
True if manufacturer specific data is present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAdvType

Description
Get advertising type of received data.

Syntax
T_GAP_ADV_EVT_TYPE getAdvType(void);

Parameters
The function requires no input parameter.

Returns
Advertising type of received advertising data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_ADV_EVT_TYPE_UNDIRECTED
– GAP_ADV_EVT_TYPE_DIRECTED
– GAP_ADV_EVT_TYPE_SCANNABLE
– GAP_ADV_EVT_TYPE_NON_CONNECTABEL
– GAP_ADV_EVT_TYPE_SCAN_RSP

BLEAdvertData::getAddrType

Description
Get Bluetooth address type of received data.

Syntax
T_GAP_REMOTE_ADDR_TYPE getAddrType(void);

Parameters
The function requires no input parameter.

Returns
Bluetooth address type of received data.

Example Code
NA

Notes and Warnings
Possible types:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

BLEAdvertData::getRSSI

Description
Get received signal strength indicator (RSSI) of received data.

Syntax
Int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
Received signal strength.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getFlags

Description
Get advertising flags of received data.

Syntax
uint8_t getFlags(void);

Parameters
The function requires no input parameter.

Returns
Advertising flags present in received advertising data, expressed as a single byte.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getServiceCount

Description
Get number of advertised services in received data.

Syntax
uint8_t getServiceCount(void);

Parameters
The function requires no input parameter.

Returns
Number of advertised service UUIDs in received data.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getServiceList

Description
Get list of advertised service UUIDs in received data.

Syntax
BLEUUID* getServiceList(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a BLEUUID array containing all advertised service UUIDs.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getName

Description
Get advertised device name in received data.

Syntax
String getName(void);

Parameters
The function requires no input parameter.

Returns
Advertised device name contained in a String class object.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEAdvertData::getTxPower

Description
Get advertised transmission power in received data.

Syntax
int8_t getTxPower(void);

Parameters
The function requires no input parameter.

Returns
Advertised transmission power.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getAppearance

Description
Get advertised device appearance in received data.

Syntax
uint16_t getAppearance(void);

Parameters
The function requires no input parameter.

Returns
Advertised device appearance.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of device appearance values.

BLEAdvertData::getManufacturer

Description
Get advertised manufacturer in received data.

Syntax
uint16_t getManufacturer(void);

Parameters
The function requires no input parameter.

Returns
Advertised manufacturer.

Example Code
NA

Notes and Warnings
Refer to Bluetooth specifications for full list of manufacturer codes.

BLEAdvertData::getManufacturerDataLength

Description
Get length of manufacturer data in received data.

Syntax
uint8_t getManufacturerDataLength(void);

Parameters
The function requires no input parameter.

Returns
Number of bytes of manufacturer data present in received advertising data.

Example Code
NA

Notes and Warnings
NA

BLEAdvertData::getManufacturerData

Description
Get manufacturer data in received data.

Syntax
uint8_t* getManufacturerData(void);

Parameters
The function requires no input parameter.

Returns
Pointer to array containing manufacturer data.

Example Code
NA

Notes and Warnings
NA

Class BLEBeacon

iBeacon Class

Description
A class used for managing iBeacon BLE advertising data.

Syntax
class iBeacon

Members

Public Constructors	
iBeacon::iBeacon	Create an instance of iBeacon advertising data
Public Methods	
iBeacon::getManufacturerId	Get current manufacturer ID value
iBeacon::getUUID	Get current UUID value
iBeacon::getMajor	Get current Major value
iBeacon::getMinor	Get current Minor value
iBeacon::getRSSI	Get current RSSI value
iBeacon::setManufacturerId	Set manufacturer ID value
iBeacon::setUUID	Set UUID value
iBeacon::setMajor	Set Major value
iBeacon::setMinor	Set Minor value
iBeacon::setRSSI	Set RSSI value
iBeacon::getAdvData	Get current advertising data
iBeacon::getScanRsp	Get current scan response data

altBeacon Class

Description
A class used for managing altBeacon BLE advertising data.

Syntax
class altBeacon

Members

Public Constructors	
altBeacon::altBeacon	Create an instance of altBeacon advertising data
Public Methods	
altBeacon::getManufacturerId	Get current manufacturer ID value
altBeacon::getUUID	Get current UUID value
altBeacon::getMajor	Get current Major value
altBeacon::getMinor	Get current Minor value
altBeacon::getRSSI	Get current RSSI value
altBeacon::getRSVD	Get current Reserved value
altBeacon::setManufacturerId	Set manufacturer ID value
altBeacon::setUUID	Set UUID value
altBeacon::setMajor	Set Major value
altBeacon::setMinor	Set Minor value
altBeacon::setRSSI	Set RSSI value
altBeacon::setRSVD	Set Reserved value
altBeacon::getAdvData	Get current advertising data
altBeacon::getScanRsp	Get current scan response data

iBeacon::iBeacon

Description
Create an iBeacon object.

Syntax
void iBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

altBeacon::altBeacon

Description
Create an altBeacon object.

Syntax
void altBeacon(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “BLEBeacon.h” to use this class function.

iBeacon::getManufacturerId

altBeacon::getManufacturerId

Description
Get current Manufacturer ID value.

Syntax
uint16_t getManufacturerId(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Company ID.

Example Code
NA

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::getUUID

altBeacon::getUUID

Description
Get the current UUID value.

Syntax
void getUUID(uint8_t* UUID);

Parameters
UUID: pointer to a 16 element uint8_t array, current UUID will be copied into the array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::getMajor

altBeacon::getMajor

Description
Get current Major value.

Syntax
uint16_t getMajor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Major value.

Example Code

NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getMinor

altBeacon::getMinor

Description
Get current Minor value.

Syntax
uint16_t getMinor(void);

Parameters
The function requires no input parameter.

Returns
A 16-bit unsigned integer containing the current Minor value.

Example Code
NA

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getRSSI

altBeacon::getRSSI

Description
Get the current RSSI value.

Syntax
int8_t getRSSI(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit signed integer containing the currently set RSSI value.

Example Code
NA

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setManufacturerId

altBeacon::setManufacturerId

Description
Set Manufacturer ID value.

Syntax
void setManufacturerId(uint16_t id);

Parameters
id: desired Manufacturer ID

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Refer to https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/ for the full list of assigned Bluetooth company identifiers.

iBeacon::setUUID

altBeacon::setUUID

Description
Set UUID value.

Syntax
void setUUID(uint8_t* UUID);
void setUUID(const char* UUID);

Parameters
uint8_t* UUID: pointer to a 16 element uint8_t array containing the desired UUID
const char* UUID: desired UUID expressed as a character string

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
UUID is a 128-bit number used to uniquely identify a beacon. It is commonly expressed as a 32-character hexadecimal string. UUIDs can be generated at https://www.uuidgenerator.net/.

iBeacon::setMajor

altBeacon::setMajor

Description
Set Major value.

Syntax
void setMajor(uint16_t major);

Parameters
major: desired Major value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setMinor

altBeacon::setMinor

Description
Set Minor value.

Syntax
void setMinor(uint16_t minor);

Parameters
minor: desired Minor value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
Major and Minor are values used for customizing beacons. These can be set to any value. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::setRSSI

altBeacon::setRSSI

Description
Set RSSI value.

Syntax
void setRSSI(int8_t RSSI);

Parameters
RSSI: desired RSSI value

Returns
The function returns nothing.

Example Code
Example: BLEBeacon

Notes and Warnings
The beacon RSSI value is the received signal strength at 1 meter. This can be used to estimate the distance to the beacon. Refer to https://developer.apple.com/ibeacon/ or https://altbeacon.org/ for more information.

iBeacon::getAdvData

altBeacon::getAdvData

Description
Get current beacon advertising data.

Syntax
uint8_t* getAdvData(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

iBeacon::getScanRsp

altBeacon::getScanRsp

Description
Get current beacon advertising scan response data.

Syntax
uint8_t* getScanRsp(void);

Parameters
The function requires no input parameter.

Returns
A uint8_t pointer to the structure containing beacon advertising scan response data.

Example Code
NA

Notes and Warnings
Avoid changing the beacon data through the returned pointer, use the member functions instead.

altBeacon::getRSVD

Description
Get current Reserved value.

Syntax
uint8_t getRSVD(void);

Parameters
The function requires no input parameter.

Returns
An 8-bit unsigned integer containing the current Reserved value.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

altBeacon::setRSVD

Description
Set Reserved value.

Syntax
void setRSVD(uint8_t rsvd);

Parameters
rsvd: desired Reserved value

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Reserved for use by the manufacturer to implement special features. The interpretation of this value is to be defined by the manufacturer and is to be evaluated based on the MFG ID value. Refer to https://altbeacon.org/ for more information.

Class BLECharacteristic

BLECharacteristic Class

Description

A class used for creating and managing BLE GATT characteristics.

Syntax
class BLECharacteristic

Members

Public Constructors	
BLEC haracteristic::BLECharacteristic	Constructs a BLECharacteristic object
Public Methods	
BLECharacteristic::setUUID	Set the characteristic UUID
BLECharacteristic::getUUID	Get the characteristic UUID
BLECharacteristic::setBufferLen	Set the size of the internal data buffer
BLECharacteristic::getBufferLen	Get the current size of the internal data buffer
BL ECharacteristic::setReadProperty	Get the current size of the internal data bufferSet the characteristic r
BLE Characteristic::setWriteProperty	Set the characteristic write property
BLEC haracteristic::setNotifyProperty	Set the characteristic notify property
BLECha racteristic::setIndicateProperty	Set the characteristic indicate property

continues o

Table 9 – continued from previous page

Public Constructors	
BLECharacteristic::setProperties	Set the characteristic properties
BLECharacteristic::getProperties	Get the characteristic properties
BLECharacteristic::readString	Read the characteristic data buffer as a String object
BLECharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLECharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLECharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLECharacteristic::writeString	Write data to the characteristic data buffer as a String object or chara
BLECharacteristic::writeData8	Write data to the characteristic data buffer as an unsigned 8-bit integ
BLECharacteristic::writeData16	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::writeData32	Write data to the characteristic data buffer as an unsigned 16-bit inte
BLECharacteristic::setData	Write data to the characteristic data buffer
BLECharacteristic::getData	Read data from the characteristic data buffer
BLECharacteristic::getDataBuff	Get a pointer to the characteristic data buffer
BLECharacteristic::getDataLen	Get the number of bytes of data in the characteristic data buffer
BLECharacteristic::notify	Send a notification to a connected device
BLECharacteristic::indicate	Send an indication to a connected device
BLEC haracteristic::setUserDescriptor	Add a user description descriptor to characteristic
BLECha racteristic::setFormatDescriptor	Add a data format descriptor to characteristic
BLECharacteristic::Add a data format descriptor to characteristic	Set a user function as a read callback
BLE Characteristic::setWriteCallback	Set a user function as a write callback
BL ECharacteristic::setCCCDCallback	Set a user function as a CCCD write callback

BLECharacteristic::BLECharacteristic

Description

Constructs a BLECharacteristic object.

Syntax

BLECharacteristic::BLECharacteristic(BLEUUID uuid);
BLECharacteristic::BLECharacteristic(const char* uuid);

Parameters

uuid: characteristic UUID, expressed as a BLEUUID class object or a character array

Returns

The function returns nothing.

Example Code

Example: BLEUartService

Notes and Warnings

NA

BLECharacteristic::setUUID

Description
Set the characteristic UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: the new characteristic UUID, expressed with a BLEUUID class object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLECharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadProperty

Description
Set the characteristic read property.

Syntax
void setReadProperty(bool value);

Parameters
value: TRUE to allow connected devices to read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteProperty

Description
Set the characteristic write property.

Syntax
void setWriteProperty(bool value);

Parameters
value: TRUE to allow connected devices to write characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setNotifyProperty

Description
Set the characteristic notify property.

Syntax
void setNotifyProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data notifications.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setIndicateProperty

Description
Set the characteristic indicate property.

Syntax
void setIndicateProperty(bool value);

Parameters
value: TRUE to allow connected devices to enable receiving characteristic data indications.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling this property will add a CCCD descriptor to the characteristic.

BLECharacteristic::setProperties

Description
Set the characteristic properties.

Syntax
void setProperties(uint8_t value);

Parameters
value: desired characteristic properties

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getProperties

Description
Get the currently set characteristic properties.

Syntax
uint8_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set characteristic properties expressed as an unsigned 8-bit integer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readString

Description
Read the data in the characteristic internal buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a String class object.

Example Code
Example: BLEUartService

Notes and Warnings
Non-ASCII data may result in unexpected characters in the string.

BLECharacteristic::readData8

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData16

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Read the data in the characteristic internal buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic internal buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::readData32

Description
Write data to the characteristic data buffer as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the characteristic buffer, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::writeData8

Description
Write data to the characteristic data buffer as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
Example: BLEBatteryService

Notes and Warnings

NA

BLECharacteristic::writeData16

Description
Write data to the characteristic data buffer as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::writeData32

Description
Write data to the characteristic data buffer as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setData

Description
Write data to the characteristic data buffer.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getData

Description
Read data from the characteristic data buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLECharacteristic::getDataBuff

Description
Get a pointer to the characteristic data buffer.

Syntax
uint8_t* getDataBuff();

Parameters
The function requires no input parameter.

Returns
The function returns a pointer to the uint8_t array used as the characteristic internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::getDataLen

Description
Get the number of bytes of data in the characteristic data buffer.

Syntax
uint16_t getDataLen

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes of data in the internal buffer.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::notify

Description
Send a notification to a connected device.

Syntax
void notify(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send a notification to.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::indicate

Description
Send an indication to a connected device.

Syntax
void indicate(uint8_t conn_id);

Parameters
conn_id: the connection ID for the device to send an indication to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setUserDescriptor

Description
Add a user description descriptor attribute (UUID 0x2901) to the characteristic.

Syntax
void setUserDescriptor(const char* description);

Parameters
description: the desired user description string expressed in a char array.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setFormatDescriptor

Description
Add a data format descriptor attribute (UUID 0x2904) to the characteristic.

Syntax
void setFormatDescriptor(uint8_t format, uint8_t exponent, uint16_t unit, uint16_t description);

Parameters
format: refer to https://www.bluetooth.com/specifications/assigned-numbers/format-types/ for the valid values and associated format types.
exponent: base-10 exponent to be applied to characteristic data value.

unit: refer to https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf for the valid values and associated units.
descriptor: refer to https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-descriptors/ for the valid values and associated descriptors.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLECharacteristic::setReadCallback

Description
Set a user function to be called when the characteristic data is read by a connected device.

Syntax
void setReadCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing data read
conn_id: connection ID of connected device that read characteristic data

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
NA

BLECharacteristic::setWriteCallback

Description

Set a user function to be called when the characteristic data is written by a connected device.

Syntax
void setWriteCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLECharacteristic::setCCCDCallback

Description
Set a user function to be called when a connected device modifies the characteristic CCCD to enable or disable notifications or indications.

Syntax
void setCCCDCallback(void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id, uint16_t ccc_bits));

Parameters
fCallback: A user callback function that returns void and takes two arguments.
chr: pointer to BLECharacteristic object containing written data.
conn_id: connection ID of connected device that wrote characteristic data.
ccc_bits: the new CCCD data bits after modification by the connected device

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

Class BLEClient

BLEClient Class

Description
A class used for discovering and accessing BLE GATT services on a connected remote device.

Syntax
class BLEClient

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLEDevice::addClient().

Public Methods	
BLEClient::connected	Check if the corresponding remote device for the client is connected
BLEClient::discoverServices	Start service discovery process for connected device
BLEClient::discoveryDone	Determine if service discovery process has been completed
BLEClient::printServices	Format and print discovered services to serial port
BLEClient::getService	Get a specific service on the remote device
BLEClient::getConnId	
BLEClient::getClientId	Get corresponding client ID
BLEClient::setDisconnectCallback	Set a user function to be called when the remote device is disconnected

BLEClient::connected

Description
Check if the remote device associated with the client is still connected.

Syntax
bool connected();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the remote device is connected.

Example Code
NA

Notes and Warnings
NA

BLEClient::discoverServices

Description
Start the service discovery process for the connected remote device.

Syntax
void discoverServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::discoveryDone

Description
Check if the service discovery process has been completed.

Syntax
bool discoveryDone();

Parameters
The function requires no input parameter.

Returns
TThe function returns TRUE if the service discovery process has been completed successfully, FALSE if the service discovery process failed, is still in progress, or has yet to start.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::printServices

Description
Print out a formatted list of discovered services to the serial port.

Syntax
void printServices();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEClient::getService

Description
Get a service with the specified UUID on the remote device.

Syntax
BLERemoteService* getService(const char* uuid);
BLERemoteService* getService(BLEUUID uuid);

Parameters
uuid: the desired service UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found service as a BLERemoteService object pointer, otherwise nullptr is returned if a service with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLEClient::getConnId

Description
Get the connection ID associated with the remote device.

Syntax
uint8_t getConnId;

Parameters
The function requires no input parameter.

Returns
The function returns the connection ID for the connected remote device.

Example Code
NA

Notes and Warnings
NA

BLEClient::getClientId

Description
Get the client ID for the BLEClient object.

Syntax
T_CLIENT_ID getClientId();;

Parameters
The function requires no input parameter.

Returns
The function returns the BLEClient object's client ID.

Example Code
NA

Notes and Warnings
The client ID is used when calling internal GATT client API.

BLEClient::setDisconnectCallback

Description
Set a user function as a callback function when the remote device is disconnected.

Syntax
void setDisconnectCallback(void (*fCallback) (BLEClient* client));

Parameters
fCallback: A user callback function that returns void and takes one argument.
client: A pointer to the BLEClient object corresponding to the disconnected remote device

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The user callback function will be called after the remote device has disconnected, before the characteristics, services and client associated with the remote device are deleted.

Class BLEConnect

BLEConnect Class

Description
A class used for managing BLE connection settings.

Syntax
class BLEConnect

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configConnection.

Public Methods	
BLEConnect::connect	Connect to a target BLE device
BLEConnect::disconnect	Disconnect from a target BLE device
BLEConnect::setScanInterval	Set the BLE scanning interval when connecting
BLEConnect::setScanWindow	Set the BLE scanning window when connecting
BLEConnect::setConnInterval	Set the BLE connection interval duration
BLEConnect::setConnLatency	Set the BLE connection slave latency
BLEConnect::setConnTimeout	Set the BLE connection timeout value
BLEConnect::updateConnParams	Send new BLE connection parameters to a connected device
BLEConnect::getConnInfo	Get connection information
BLEConnect::getConnAddr	Get the Bluetooth address for a certain connection
BLEConnect::getConnId	Get the connection ID for a certain device

BLEConnect::connect

Description
Connect to a target BLE device.

Syntax
bool connect(char* btAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(uint8_t (&btAddr)[6], T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);
bool connect(BLEAdvertData targetDevice, uint16_t scanTimeout);
bool connect(BLEAddr destAddr, T_GAP_REMOTE_ADDR_TYPE destAddrType, uint16_t scanTimeout);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t (&btAddr): target device Bluetooth address contained in a 6 byte array.
destAddr: target device Bluetooth address contained in BLEAddr class object.

targetDevice: advertising data packet scanned from target device.
destAddrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM
scan timeout: duration in milliseconds for which to look for target device before giving up.

Returns
True if connection successful, false if connection failed.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLEConnect::disconnect

Description
Disconnect from a target BLE device.

Syntax
bool disconnect(uint8_t connId);

Parameters
connId: connection ID for target device.

Returns
True if operation successful, false if otherwise.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanInterval

Description

Set the BLE scan interval when searching for a target device to connect to.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setScanWindow

Description
Set the BLE scan window when searching for a target device to connect to.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEConnect::setConnInterval

Description
Set the BLE connection interval value.

Syntax
void setConnInterval(uint16_t min_ms, uint16_t max_ms);

Parameters
min_ms: minimum acceptable connection interval in milliseconds. Value range of 8 to 4000.
max_ms: maximum acceptable connection interval in milliseconds. Value range of 8 to 4000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection interval defines the period between successive connection events between a connected central and peripheral device. Even if there is no data to exchange, a connection event is required to maintain the connection. max_ms should be larger than or equal to min_ms.

BLEConnect::setConnLatency

Description
Set the BLE connection slave latency value.

Syntax
void setConnLatency(uint16_t latency);

Parameters
latency: Connection slave latency value. Value range of 0 to 499.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection slave latency defines the number of successive connection events a connected peripheral device can ignore without being considered as disconnected by the central device.

BLEConnect::setConnTimeout

Description
Set the BLE connection timeout value.

Syntax
void setConnTimeout(uint16_t timeout_ms);

Parameters
timeout_ms: connection timeout in milliseconds. Value range of 100 to 32000.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The BLE connection timeout defines the duration after a failed connection events before a peripheral or central device considers the connection broken.

BLEConnect::updateConnParams

Description
Update a connected device with new connection parameters.

Syntax
void updateConnParams(uint8_t conn_id);

Parameters
conn_id: connection ID of target device to update connection parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Update a connected device with previously set connection interval, slave latency and timeout values. The connected device may reject the new values if it is unable to conform to them.

BLEConnect::getConnInfo

Description
Get connection information.

Syntax
bool getConnInfo(uint8_t connId, T_GAP_CONN_INFO *pConnInfo);

Parameters
connId: connection ID to get connection information from.
pConnInfo: pointer to T_GAP_CONN_INFO structure to store obtained connection information.

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnAddr

Description
Get the Bluetooth address for a certain connection.

Syntax
bool getConnAddr(uint8_t connId, uint8_t* addr, uint8_t* addrType);

Parameters
connId: connection ID to get address information for

addr: pointer to 6 byte array to store retrieved Bluetooth address
addrType: pointer to uint8_t variable to store retrieved Bluetooth address type

Returns
True if operation success, false if operation failed.

Example Code
NA

Notes and Warnings
NA

BLEConnect::getConnId

Description
Get the connection ID for a certain device.

Syntax
int8_t getConnId(char* btAddr, uint8_t addrType);
int8_t getConnId(uint8_t* btAddr, uint8_t addrType);
int8_t getConnId(BLEAdvertData targetDevice);

Parameters
char* btAddr: target device Bluetooth address expressed as a character string.
uint8_t* btAddr: pointer to a 6 byte array containing target device Bluetooth address.
targetDevice: advertising data packet scanned from target device.
addrType: Bluetooth address type of target device. Valid values:
– GAP_REMOTE_ADDR_LE_PUBLIC
– GAP_REMOTE_ADDR_LE_RANDOM

Returns
The function returns the requested connection ID. Returns -1 if failed to obtain connection ID.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

Class BLEDevice

BLEDevice Class

Description
A class used for general control and management of BLE functions.

Syntax
class BLEDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLE.

Public Methods	
BLEDevice::init	Allocate resources required for BLE functionality
BLEDevice::deinit	Free resources used by BLE functionality
BLEDevice::connected	Check if a BLE device is connected
BLEDevice::setDeviceName	Set BLE GAP device name
BLEDevice::setDeviceAppearance	Set BLE GAP device appearance
BLEDevice::configAdvert	Configure BLE advertising parameters
BLEDevice::configScan	Configure BLE scan parameters
BLEDevice::setScanCallback	Set callback function for BLE scans
BLEDevice::beginCentral	Start BLE stack in central mode
BLEDevice::beginPeripheral	Start BLE stack in peripheral mode
BLEDevice::end	Stop BLE stack
BLEDevice::configServer	Configure BLE stack for services
BLEDevice::addService	Add a service to the BLE stack
BLEDevice::configClient	Configure BLE stack for clients
BLEDevice::addClient	Add a client to the BLE stack
BLEDevice::getLocalAddr	Get local device Bluetooth address

BLEDevice::init

Description
Allocate resources required for BLE functionality.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Call this member function first before using any other member functions in the BLEDevice class.

BLEDevice::deinit

Description
Free up resources used for BLE functionality.

Syntax
void deinit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Call this member function last after all other BLE operations are stopped.

BLEDevice::connected

Description
Check if a BLE device is connected.

Syntax
bool connected(void);

Parameters
The function requires no input parameter.

Returns
TRUE if another BLE device is connected, FALSE if no BLE device is connected.

Example Code
NA

Notes and Warnings
NA

BLEDevice::setDeviceName

Description
Set the BLE GAP device name.

Syntax
void setDeviceName(String devName);

Parameters
devName: desired device name contained in an Arduino String object

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The GAP device name has a maximum length of 39 characters. Other devices can see this name after a BLE connection is established. This name is separate and different from the device name sent in a BLE advertisement, the names should be the same but are not required.

BLEDevice::setDeviceAppearance

Description
Set the BLE GAP device appearance.

Syntax
void setDeviceAppearance(uint16_t devAppearance);

Parameters
devAppearance: desired device appearance expressed as a 16-bit unsigned integer.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to Bluetooth SIG assigned device appearances at https://www.bluetooth.com/specifications/gatt/characteristics/.

BLEDevice::configAdvert

Description
Configure BLE advertising parameters.

Syntax
BLEAdvert* configAdvert(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEAdvert class instance for configuring BLE advertising parameters.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function instead of creating a BLEAdvert class instance manually.

BLEDevice::configScan

Description
Configure BLE scanning parameters.

Syntax
BLEScan* configScan(void);

Parameters
The function requires no input parameter.

Returns
A pointer to a BLEScan class instance for configuring BLE scanning parameters.

Example Code
Example: BLEScan

```
#include “BLEDevice.h”
#include “BLEScan.h”
int dataCount = 0;
void scanFunction(T_LE_CB_DATA* p_data) {
printf(”rnScan Data %drn”, ++dataCount);
BLE.configScan()->printScanInfo(p_data);
}
void setup() {
BLE.init();
BLE.configScan()->setScanMode(GAP_SCAN_MODE_ACTIVE);
BLE.configScan()->setScanInterval(500); // Start a scan every 500ms
BLE.configScan()->setScanWindow(250); // Each scan lasts for 250ms
// Provide a callback function to process scan data.
// If no function is provided, default BLEScan::printScanInfo is used
BLE.setScanCallback(scanFunction);
BLE.beginCentral(0);
BLE.configScan()->startScan(5000); // Repeat scans for 5 seconds, then stop
}
void loop() {
}
```

Notes and Warnings

Use this member function instead of creating a BLEScan class instance manually.

BLEDevice::setScanCallback

Description
Set a callback function for processing BLE scan results.

Syntax
void setScanCallback(void (*scanCB)(T_LE_CB_DATA));

Parameters
scanCB: a function that returns nothing and takes in a scan data pointer of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to set a callback function that will be called for each BLE device scan result found.

BLEDevice::beginCentral

Description
Start the BLE stack in central mode.

Syntax
void beginCentral(uint8_t connCount);

Parameters
connCount: maximum number of allowed connected devices. If no argument is provided, default to maximum allowed connected devices for specific board.

Returns
The function returns nothing.

Example Code
Example: BLEScan
The function returns nothing.

Notes and Warnings
Use this member function to start the device in BLE central mode, after other BLE parameters are set correctly.

BLEDevice::beginPeripheral

Description
Start the BLE stack in peripheral mode.

Syntax
void beginPeripheral(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function to start the device in BLE peripheral mode, after other BLE parameters are set correctly.

BLEDevice::end

Description
Stop the BLE stack.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Use this member function to stop the device operating in either BLE peripheral mode or BLE central mode.

BLEDevice::configServer

Description
Configure the BLE stack for services.

Syntax
void configServer(uint8_t maxServiceCount);

Parameters
maxServiceCount: Maximum number of services that will run on the device

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
Use this member function before adding any service to the BLE stack.

BLEDevice::addService

Description
Add a new service to the BLE stack.

Syntax
void addService(BLEService& newService);

Parameters
newService: the service to be added, defined using a BLEService class object.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryService

Notes and Warnings
N/A

BLEDevice::configClient

Description
Configure the BLE stack for clients.

Syntax
void configClient();

Parameters
The function requries no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Use this member function before adding any client to the BLE stack.

BLEDevice::addClient

Description
Add a new client to the BLE stack.

Syntax
BLEClient* addClient(uint8_t connId);

Parameters
connId: the connection ID of the connected device to create a client for.

Returns
The function returns a pointer to a BLEClient class object, corresponding to the device with the specified connection ID, which can be used to access the services and characteristics on the connected device.

Example Code
Example: BLEBatteryClient

Notes and Warnings
Only one client should be added per connected device.
The BLEClient object and any service, characteristic, descriptor associated with the connected device will be deleted when the device is disconnected.

BLEDevice::getLocalAddr

Description
Get local device Bluetooth address.

Syntax
void getLocalAddr(uint8_t (&addr)[GAP_BD_ADDR_LEN]);

Parameters
addr: 6 byte array to store local device Bluetooth address.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Local device address is only available after starting in central or peripheral mode. This function will return all zeros for the address if central or peripheral mode is not in operation.

Class BLEHIDDevice

BLEHIDDevice Class

Description

A class used for creating and managing HID over GATT Profile (HOGP) services.

Syntax
class BLEHIDDevice

Members

Public Constructors
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named BLEHIDDev.

Public Methods	
BLEHIDDevice::init	Initialize the HID Device Profile by creating the required services
BLEHIDD evice::setNumOutputReport	Configure the number of HID output reports
BLEHID Device::setNumInputReport	Configure the number of HID input reports
B LEHIDDevice::setReportMap	Configure the HID report map
BLEHIDDevice::inputReport	Send a HID input report
BLEHIDDevice ::setOutputReportCallback	Set a user callback function for receiving HID output reports
BLEHIDD evice::bootKeyboardReport	Send a HID boot keyboard input report
BLEHIDDevice::setHidInfo	Set HID info of the HID service
B LEHIDDevice::setBattLevel	Set battery level info of the Battery service
BLEHIDDevice::setPNPInfo	Set PNP information of the Device Information service
BLEHIDDevi ce::setManufacturerString	Set manufacturer information of the Device Information service
BLE HIDDevice::setModelString	Set model information of the Device Information service
BLEHIDDevice::hidService	Get reference to HID service
BLE HIDDevice::devInfoService	Get reference to Device Information service
BLEHIDDevice::battService	Get reference to Battery service

BLEHIDDevice::init

Description
Initialize the HID Device profile by creating the required services.

Syntax
void init(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID Device object should be initialized before any HID reports can be sent.

BLEHIDDevice::setNumOutputReport

Description
Configure the number of HID output reports.

Syntax
void setNumOutputReport (uint8_t numOutputReports);

Parameters
numOutputReports: number of output reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of output reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setNumInputReport

Description
Configure the number of HID input reports.

Syntax
void setNumInputReport (uint8_t numInputReports);

Parameters
numInputReports: number of input reports

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
The number of input reports should be configured before BLEHIDDevice init() function is called.

BLEHIDDevice::setReportMap

Description
Configure the HID report map characteristic with a HID report descriptor.

Syntax
void setReportMap (uint8_t* report_map, uint16_t len);

Parameters
report_map: pointer to HID report descriptor
len: HID report descriptor length in bytes

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
The HID report map characteristic can only be configured after BLEHIDDevice init() function is called.

BLEHIDDevice::inputReport

Description
Send a HID input report.

Syntax
void inputReport (uint8_t reportID, uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters

reportID: HID report ID of input report
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
HID input reports can only be sent after BLEHIDDevice init() function has been called.

BLEHIDDevice::setOutputReportCallback

Description
Set a user callback function for receiving HID output report data.

Syntax
void setOutputReportCallback (uint8_t reportID, void (*fCallback) (BLECharacteristic* chr, uint8_t conn_id));

Parameters
reportID: HID report ID of output report to link callback function with
chr: BLECharacteristic class object containing received HID output report data
conn_id: connection ID of device which sent HID report data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Setting a user callback function for output reports can only occur after BLEHIDDevice init() function has been called.

BLEHIDDevice::bootKeyboardReport

Description
Send a HID boot keyboard input report.

Syntax
void bootKeyboardReport (uint8_t* data, uint16_t len, uint8_t conn_id);

Parameters
data: pointer to HID input report data to send
len: length of HID input report data in bytes
conn_id: connection ID of device to send HID report to

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, the HID service Protocol Mode characteristic has boot mode disabled. To send boot keyboard input reports, the Protocol Mode characteristic needs to have boot mode enabled.

BLEHIDDevice::setHidInfo

Description
Set data of the HID Info characteristic of the HID service.

Syntax
void setHidInfo (uint16_t bcd, uint8_t country, uint8_t flags);

Parameters
bcd: 16-bit unsigned integer representing version number of base USB HID Specification implemented by HID Device
country: 8-bit integer identifying country HID Device hardware is localized for. Most hardware is not localized (value 0x00).
flags: Bit flags indicating remote-wake capability and advertising when bonded but not connected.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
For detailed information on the characteristic, refer to Bluetooth SIG HID Service specifications.

BLEHIDDevice::setBattLevel

Description
Set battery level data of the Battery service.

Syntax
void setBattLevel (uint8_t level);

Parameters
level: battery level expressed as % of full charge

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Battery level is set to 100% by default. For detailed information refer to Bluetooth SIG Battery service specifications.

BLEHIDDevice::setPNPInfo

Description
Set PNP data of the Device Information service.

Syntax
void setPNPInfo (uint8_t sig, uint16_t vid, uint16_t pid, uint16_t version);

Parameters
sig: The Vendor ID Source field designates which organization assigned the value used in the Vendor ID field value.
vid: The Vendor ID field is intended to uniquely identify the vendor of the device.
pid: The Product ID field is intended to distinguish between different products made by the vendor.
version: The Product Version field is a numeric expression identifying the device release number in Binary-Coded Decimal.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
By default, sig and vid are configured to indicate Realtek as the vendor. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setManufacturerString

Description
Set manufacturer information of the Device Information service.

Syntax
void setManufacturerString (const char* manufacturer);

Parameters
manufacturer: pointer to character string containing manufacturer name info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Manufacturer is set to “Realtek” by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::setModelString

Description
Set model information of the Device Information service.

Syntax
void setModelString (const char* model);

Parameters
model: pointer to character string containing device model info.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Model is set to “Ameba_BLE_HID” by default. For detailed information refer to Bluetooth SIG Device Information service specifications.

BLEHIDDevice::hidService

Description
Get reference to HID service

Syntax
BLEService& hidService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the HID service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::devInfoService

Description
Get reference to Device Information service

Syntax
BLEService& devInfoService ();

Parameters
The function requires no input parameter.

Returns

The function returns a reference to the BLEService class object for the Device Information service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDDevice::battService

Description
Get reference to Battery service

Syntax
BLEService& battService ();

Parameters
The function requires no input parameter.

Returns
The function returns a reference to the BLEService class object for the Battery service.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLEHIDGamepad

BLEHIDGamepad Class

Description

A class used for creating and managing a BLE HID Gamepad.

Syntax
class BLEHIDGamepad

Members

Public Constructors	
BLEHIDGame pad::BLEHIDGamepad	Constructs a BLEHIDGamepad object
Public Methods	
BLEHIDGa mepad::setReportID	Set HID report ID for the HID Gamepad
BLEHIDGame pad::gamepadReport	Send a HID Gamepad report
BLEHIDGa mepad::buttonPress	Send a HID Gamepad report indicating buttons pressed
BLEHIDGame pad::buttonRelease	Send a HID Gamepad report indicating buttons released
BLEHIDGamepad ::buttonReleaseAll	Send a HID Gamepad report indicating no buttons pressed
BLE HIDGamepad::setHat	Send a HID Gamepad report indicating hat switch position
BLEH IDGamepad::setAxes	Send a HID Gamepad report indicating position of all axes
BLEHIDGam epad::setLeftStick	Send a HID Gamepad report indicating position of axes corresponding to left analog stick
BLEHIDGame pad::setRightStick	Send a HID Gamepad report indicating position of axes corresponding to right analog stick
BLEHIDGa mepad::setTriggers	Send a HID Gamepad report indicating position of axes corresponding to triggers

BLEHIDGamepad::BLEHIDGamepad

Description
Constructs a BLE object

Syntax
BLEHIDGamepad::BLEHIDGamepad();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
By default, the BLEHIDGamepad class assumes the HID report descriptor implements a gamepad device with 16 buttons, 6 16-bit axes and an 8-direction hat switch. This class will not work if a different gamepad report descriptor is implemented.

BLEHIDGamepad::setReportID

Description
Set HID report ID for the HID Gamepad.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the gamepad device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
HID report ID should start at 1. Some systems may consider a report ID of 0 as invalid.

BLEHIDGamepad::gamepadReport

Description
Send a HID Gamepad report.

Syntax
void gamepadReport (hid_gamepad_report_t* report);
void gamepadReport (uint16_t buttons, uint8_t hat, int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
report: pointer to gamepad report structure containing data on all inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::buttonPress

Description
Send a HID Gamepad report indicating buttons pressed.

Syntax
void buttonPress (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonRelease

Description
Send a HID Gamepad report indicating buttons released.

Syntax
void buttonRelease (uint16_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::buttonReleaseAll

Description
Send a HID Gamepad report indicating no buttons pressed.

Syntax
void buttonReleaseAll (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setHat

Description
Send a HID Gamepad report indicating hat switch position.

Syntax
void setHat (uint8_t hat);

Parameters
hat: position of hat switch. Valid values:
– GAMEPAD_HAT_CENTERED = 0
– GAMEPAD_HAT_UP = 1
– GAMEPAD_HAT_UP_RIGHT = 2
– GAMEPAD_HAT_RIGHT = 3
– GAMEPAD_HAT_DOWN_RIGHT = 4
– GAMEPAD_HAT_DOWN = 5
– GAMEPAD_HAT_DOWN_LEFT = 6
– GAMEPAD_HAT_LEFT = 7
– GAMEPAD_HAT_UP_LEFT = 8

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setAxes

Description
Send a HID Gamepad report indicating position of all axes.

Syntax
void setAxes (int16_t x, int16_t y, int16_t z, int16_t Rz, int16_t Rx, int16_t Ry);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
Example: BLEHIDGamepad

Notes and Warnings
NA

BLEHIDGamepad::setLeftStick

Description
Send a HID Gamepad report indicating position of axes corresponding to left analog stick.

Syntax
void setLeftStick (int16_t x, int16_t y);

Parameters
x: position of x axis. Integer value from -32767 to 32767.
y: position of y axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setRightStick

Description
Send a HID Gamepad report indicating position of axes corresponding to right analog stick.

Syntax
void setLeftStick (int16_t z, int16_t Rz);

Parameters
z: position of z axis. Integer value from -32767 to 32767.
Rz: position of Rz axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDGamepad::setTriggers

Description
Send a HID Gamepad report indicating position of axes corresponding to triggers.

Syntax
void setTriggers (int16_t Rx, int16_t Ry);

Parameters
Rx: position of Rx axis. Integer value from -32767 to 32767.
Ry: position of Ry axis. Integer value from -32767 to 32767.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

Class BLEHIDKeyboard

BLEHIDKeyboard Class

Description

A class used for creating and managing a BLE HID Keyboard.

Syntax

class BLEHIDKeyboard

Members

Public Constructors	
BLEHIDKeybo ard::BLEHIDKeyboard	Constructs a BLEHIDKeyboard object
Public Methods	
BLEHIDKe yboard::setReportID	Set HID report ID for the HID Keyboard and HID consumer control
BLEHIDKeybo ard::consumerReport	Send a HID Consumer report
BLEHIDKeybo ard::keyboardReport	Send a HID Keyboard report
BLEHIDKeyb oard::consumerPress	Send a HID Consumer report indicating button pressed
BLEHIDKeyboa rd::consumerRelease	Send a HID Consumer report indicating button released
BLEHI DKeyboard::keypress	Send a HID Keyboard report indicating keys pressed
BLEHIDK eyboard::keyRelease	Send a HID Keyboard report indicating keys released
BLEHIDKeyb oard::keyReleaseAll	Send a HID Keyboard report indicating no keys pressed
BLEHIDKey board::keyCharPress	Send a HID Keyboard report indicating keys pressed to output an ASCII character
BLEHIDKe yboard::keySequence	Send a HID Keyboard report indicating keys pressed to output an ASCII string

BLEHIDKeyboard::BLEHIDKeyboard

Description

Constructs a BLEHIDKeyboard object.

Syntax

BLEHIDKeyboard::BLEHIDKeyboard();

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::setReportID

Description
Set HID report ID for the HID Keyboard and HID consumer control.

Syntax
void setReportID (uint8_t reportIDKeyboard, uint8_t reportIDConsumer);

Parameters
reportIDKeyboard: The report ID for the HID keyboard device, corresponding to the HID report descriptor.
reportIDConsumer: The report ID for the HID consumer control device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerReport

Description
Send a HID Consumer report.

Syntax
void consumerReport (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyboardReport

Description
Send a HID Keyboard report.

Syntax
void keyboardReport (void);
void keyboardReport (uint8_t modifiers, uint8_t keycode[6]);

Parameters
modifiers: bitmap indicating key modifiers pressed (CTRL, ALT, SHIFT).
keycode: byte array indicating keys pressed.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerPress

Description
Send a HID Consumer report indicating button pressed.

Syntax
void consumerPress (uint16_t usage_code);

Parameters
usage_code: HID consumer control usage code for the button pressed.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::consumerRelease

Description
Send a HID Consumer report indicating button released.

Syntax
void consumerRelease (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keypress

Description
Send a HID Keyboard report indicating keys pressed.

Syntax
void keyPress (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyRelease

Description
Send a HID Keyboard report indicating keys released.

Syntax
void keyRelease (uint16_t key);

Parameters
key: HID keycode for key pressed, value ranges from 0x00 to 0xE7.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keyReleaseAll

Description
Send a HID Keyboard report indicating no keys pressed.

Syntax
void keyReleaseAll(void);

Parameters
The function requires no input parameter.

Returns

The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

BLEHIDKeyboard::keyCharPress

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII character.

Syntax
void keyCharPress (char ch);

Parameters
ch: ASCII character to output.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDKeyboard::keySequence

Description
Send a HID Keyboard report indicating keys pressed to output an ASCII string.

Syntax
void keySequence (const char* str, uint16_t delayTime);
void keySequence (String str, uint16_t delayTime);

Parameters
str: pointer to character string to output
str: String object containing character string to output

delayTime: time delay between key press and release, in milliseconds. Default value of 5.

Returns
The function returns nothing.

Example Code
Example: BLEHIDKeyboard

Notes and Warnings
NA

Class BLEHIDMouse

BLEHIDMouse Class

Description

A class used for creating and managing a BLE HID Mouse.

Syntax
class BLEHIDMouse

Members

Public Constructors	
BLE HIDMouse::BLEHIDMouse	Constructs a BLEHIDMouse object
Public Methods	
BLE HIDMouse::setReportID	Set HID report ID for the HID Mouse
BLE HIDMouse::mouseReport	Send a HID Mouse report
BL EHIDMouse::mousePress	Send a HID Mouse report indicating buttons pressed
BLEH IDMouse::mouseRelease	Send a HID Mouse report indicating buttons released
BLEHIDM ouse::mouseReleaseAll	Send a HID Mouse report indicating no buttons pressed
B LEHIDMouse::mouseMove	Send a HID Mouse report indicating mouse movement
BLE HIDMouse::mouseScroll	Send a HID Mouse report indicating mouse scroll wheel movement

BLEHIDMouse::BLEHIDMouse

Description
Constructs a BLEHIDMouse object.

Syntax
BLEHIDMouse::BLEHIDMouse();

Parameters

The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::setReportID

Description
Set HID report ID for the HID Mouse.

Syntax
void setReportID (uint8_t reportID);

Parameters
reportID: The report ID for the HID mouse device, corresponding to the HID report descriptor.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseReport

Description
Send a HID Mouse report.

Syntax
void mouseReport (hid_mouse_report_t* report);
void mouseReport (uint8_t buttons, int8_t x, int8_t y, int8_t scroll);

Parameters
report: pointer to mouse report structure containing data on mouse inputs
buttons: bitmap indicating state of each button. 1 = pressed, 0 = released.
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mousePress

Description
Send a HID Mouse report indicating buttons pressed.

Syntax
void mousePress (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons pressed. 1 = pressed.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseRelease

Description
Send a HID Mouse report indicating buttons released.

Syntax
void mouseRelease (uint8_t buttons);

Parameters
buttons: bitmap indicating buttons released. 1 = released.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseReleaseAll

Description
Send a HID Mouse report indicating no buttons pressed.

Syntax
void mouseReleaseAll(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEHIDMouse::mouseMove

Description
Send a HID Mouse report indicating mouse movement.

Syntax
void mouseMove (int8_t x, int8_t y);

Parameters
x: mouse x-axis movement. Integer value from -127 to 127.
y: mouse y-axis movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

BLEHIDMouse::mouseScroll

Description
Send a HID Mouse report indicating mouse scroll wheel movement.

Syntax
void mouseScroll (int8_t scroll);

Parameters
scroll: mouse scroll wheel movement. Integer value from -127 to 127.

Returns
The function returns nothing.

Example Code
Example: BLEHIDMouse

Notes and Warnings
NA

Class BLERemoteCharacteristic

BLERemoteCharacteristic Class

Description
A class used for managing BLE GATT characteristics on connected remote devices.

Syntax
class BLERemoteCharacteristic

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteService::getCharacteristic().

Public Methods	
BLERem oteCharacteristic::getDescriptor	Get a specific descriptor on the remote device
BLERemoteCharacteristic::getUUID	Get the characteristic UUID
BLERe moteCharacteristic::setBufferLen	Set the size of the internal data buffer
BLERe moteCharacteristic::getBufferLen	Get the current size of the internal data buffer
BLERemoteCharacteristic::canRead	Determine if characteristic has read property enabled
B LERemoteCharacteristic::canWrite	Determine if characteristic has write property enabled
BL ERemoteCharacteristic::canNotify	Determine if characteristic has notify property enabled
BLER emoteCharacteristic::canIndicate	Determine if characteristic has indicate property enabled
BLERem oteCharacteristic::getProperties	Get the characteristic properties
BLE RemoteCharacteristic::readString	Read the characteristic data buffer as a String object
BL ERemoteCharacteristic::readData8	Read the characteristic data buffer as an unsigned 8-bit integer
BLE RemoteCharacteristic::readData16	Read the characteristic data buffer as an unsigned 16-bit integer
BLE RemoteCharacteristic::readData32	Read the characteristic data buffer as an unsigned 32-bit integer
BLER emoteCharacteristic::writeString	Write data to the characteristic as a String object or character array
BLE RemoteCharacteristic::writeData8	Write data to the characteristic as an unsigned 8-bit integer
BLER emoteCharacteristic::writeData16	Write data to the characteristic as an unsigned 16-bit integer
BLER emoteCharacteristic::writeData32	Write data to the characteristic as an unsigned 16-bit integer
BLERemoteCharacteristic::setData	Write data to the characteristic
BLERemoteCharacteristic::getData	Read data from the characteristic
BLERemoteChar acteristic::enableNotifyIndicate	Enable notification or indication for the characteristic
BLERemoteChara cteristic::disableNotifyIndicate	Disable notification and indication for the characteristic
BLERemoteC haracteristic::setNotifyCallback	Set a user function as a notification callback

BLERemoteCharacteristic::getDescriptor

Description
Get a descriptor with the specified UUID on the remote device.

Syntax
BLERemoteDescriptor* getDescriptor(const char* uuid);
BLERemoteDescriptor* getDescriptor(BLEUUID uuid);

Parameters
uuid: the desired descriptor UUID, expressed as a character array or a BLEUUID object

Returns
The function returns the found descriptor as a BLERemoteDescriptor object pointer, otherwise nullptr is returned if a descriptor with the UUID is not found.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getUUID

Description
Get the characteristic UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setBufferLen

Description
Set the size of the internal data buffer of the characteristic.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
Characteristic data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteCharacteristic::getBufferLen

Description
Get the size of the characteristic internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canRead

Description
Determine if characteristic has read property enabled.

Syntax
bool canRead();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the read property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canWrite

Description
Determine if characteristic has write property enabled.

Syntax
bool canWrite();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the write property for the characteristic is enabled.

Example Code
NA

Notes and Warnings

NA

BLERemoteCharacteristic::canNotify

Description
Determine if characteristic has notify property enabled.

Syntax
bool canNotify();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the notify property for the characteristic is enabled.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::canIndicate

Description
Determine if characteristic has indicate property enabled.

Syntax
bool canIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns TRUE if the indicate property for the characteristic is enabled.

Example Code

NA

Notes and Warnings
NA

BLERemoteCharacteristic::getProperties

Description
Get the characteristic properties.

Syntax
uint16_t getProperties();

Parameters
The function requires no input parameter.

Returns
The function returns the characteristic properties.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readString

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns

The function returns the data in the characteristic buffer expressed as a String class object.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData8

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint8_t value.

Example Code
Example: BLEBatteryClient

Notes and Warnings
NA

BLERemoteCharacteristic::readData16

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::readData32

Description
Request for characteristic data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the characteristic buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeString

Description
Write data to the remote device characteristic as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote characteristic, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData8

Description
Write data to the remote device characteristic as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData16

Description
Write data to the remote device characteristic as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the characteristic buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::writeData32

Description
Write data to the remote device characteristic as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the characteristic buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setData

Description
Write data to the remote device characteristic.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::getData

Description
Request for characteristic data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

BLERemoteCharacteristic::enableNotifyIndicate

Description
Enable the remote device to send notifications or indications for the characteristic.

Syntax
void enableNotifyIndicate(bool notify = 1);

Parameters
notify: TRUE to enable notifications, FALSE to enable indications.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

BLERemoteCharacteristic::disableNotifyIndicate

Description
Disable receiving notifications and indications for the characteristic from the remote device.

Syntax
void disableNotifyIndicate();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLERemoteCharacteristic::setNotifyCallback

Description
Set a user function to be called when the characteristic receives a notification from the remote device.

Syntax
void setNotifyCallback(void (*fCallback) (BLERemoteCharacteristic* chr, uint8_t* data, uint16_t length));

Parameters
fCallback: A user callback function that returns void and takes three arguments.
chr: pointer to BLERemoteCharacteristic object associated with notification.
data: pointer to byte array containing notification data.
length: number of bytes of notification data in array.

Returns
The function returns nothing.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLERemoteDescriptor

BLERemoteDescriptor Class

Description
A class used for managing BLE GATT descriptors on connected remote devices.

Syntax
class BLERemoteDescriptor

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLERemoteCharacteristic::getDescriptor().

Public Methods	
BLERemoteDescriptor::getUUID	Get the descriptor UUID
B LERemoteDescriptor::setBufferLen	Set the size of the internal data buffer
B LERemoteDescriptor::getBufferLen	Get the current size of the internal data buffer
BLERemoteDescriptor::readString	Read the descriptor data buffer as a String object
BLERemoteDescriptor::readData8	Read the descriptor data buffer as an unsigned 8-bit integer
BLERemoteDescriptor::readData16	Read the descriptor data buffer as an unsigned 16-bit integer
BLERemoteDescriptor::readData32	Read the descriptor data buffer as an unsigned 32-bit integer
BLERemoteDescriptor::writeString	Write data to the descriptor as a String object or character array
BLERemoteDescriptor::writeData8	Write data to the descriptor as an unsigned 8-bit integer
BLERemoteDescriptor::writeData16	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::writeData32	Write data to the descriptor as an unsigned 16-bit integer
BLERemoteDescriptor::setData	Write data to the descriptor
BLERemoteDescriptor::getData	Read data from the descriptor

BLERemoteDescriptor::getUUID

Description
Get the descriptor UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the descriptor UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setBufferLen

Description
Set the size of the internal data buffer of the descriptor.

Syntax
void setBufferLen(uint16_t max_len);

Parameters
max_len: number of bytes to resize the internal buffer to.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Descriptor data buffer has a default size of 20 bytes and can be increased up to 230 bytes.

BLERemoteDescriptor::getBufferLen

Description
Get the size of the descriptor internal buffer.

Syntax
uint16_t getBufferLen();

Parameters
The function requires no input parameter.

Returns
The function returns the currently set internal buffer size.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readString

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as a String class object.

Syntax
String readString();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a String class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData8

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 8-bit integer.

Syntax
uint8_t readData8();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint8_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData16

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 16-bit integer.

Syntax
uint16_t readData16();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint16_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::readData32

Description
Request for descriptor data from the remote device and read the data in the buffer, expressed as an unsigned 32-bit integer.

Syntax
uint32_t readData32();

Parameters
The function requires no input parameter.

Returns
The function returns the data in the descriptor buffer expressed as a uint32_t value.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeString

Description
Write data to the remote device descriptor as a String object or character array.

Syntax
bool writeString(String str);
bool writeString(const char* str);

Parameters
str: the data to write to the remote descriptor, expressed as a String class object or a char array.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData8

Description
Write data to the remote device descriptor as an unsigned 8-bit integer.

Syntax
bool writeData8(uint8_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 8-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData16

Description
Write data to the remote device descriptor as an unsigned 16-bit integer.

Syntax
bool writeData16(uint16_t num);

Parameters
num: the data to write to the descriptor buffer expressed as an unsigned 16-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::writeData32

Description
Write data to the remote device descriptor as a 32-bit integer.

Syntax
bool writeData32(uint32_t num);
bool writeData32(int num);

Parameters
num: the data to write to the descriptor buffer expressed as a 32-bit integer.

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::setData

Description
Write data to the remote device descriptor.

Syntax
bool setData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array containing desired data
datalen: number of bytes of data to write

Returns
The function returns TRUE if write data is successful.

Example Code
NA

Notes and Warnings
NA

BLERemoteDescriptor::getData

Description
Request for descriptor data from the remote device and read the data in the buffer.

Syntax
uint16_t getData(uint8_t* data, uint16_t datalen);

Parameters
data: pointer to byte array to save data read from buffer
datalen: number of bytes of data to read

Returns
The function returns the number of bytes read.

Example Code
NA

Notes and Warnings
If the data buffer contains less data than requested, it will only read the available number of bytes of data.

Class BLERemoteService

BLERemoteService Class

Description
A class used for managing BLE GATT services on connected remote devices.

Syntax
class BLERemoteService

Members

Public Constructors
No public constructor is available for this class. You can get a pointer to an instance of this class using BLEClient::getService().

Public Methods	
BLERemoteService::getUUID	Get the service UUID
BLE RemoteService::getCharacteristic	Get a specific characteristic on the remote device

BLERemoteService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID as a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLERemoteService::getCharacteristic

Description
Get a characteristic with the specified UUID on the remote device.

Syntax
BLERemoteCharacteristic* getCharacteristic (const char* uuid);
BLERemoteCharacteristic* getCharacteristic (BLEUUID uuid);

Parameters
uuid: the desired characteristic UUID, expressed as a character array or a BLEUUID object.

Returns
The function returns the found characteristic as a BLERemoteCharacteristic object pointer, otherwise nullptr is returned if a characteristic with the UUID is not found.

Example Code
Example: BLEUartClient

Notes and Warnings
NA

Class BLEScan

BLEScan Class

Description
A class used for managing BLE scanning settings.

Syntax
class BLEScan

Members

Public Constructors
No public constructor is available as this class is intended to be a singleton class. You can get a pointer to this class using BLEDevice::configScan

Public Methods	
BLEScan::updateScanParams	Update the current BLE advertisement settings to the lower Bluetooth stack
BLEScan::startScan	Start a BLE scan
BLEScan::stopScan	Stop a BLE scan
BLEScan::setScanMode	Set the BLE scanning mode
BLEScan::setScanInterval	Set the BLE scanning interval
BLEScan::setScanWindow	Set the BLE scanning window
BLEScan::setScanDuplicateFilter	Set the BLE scan duplicate filter
BLEScan::scanInProgress	Check if a scan is currently in progress
BLEScan::printScanInfo	Print out scanned information

BLEScan::updateScanParams

Description
Update the lower Bluetooth stack with the current scan settings.

Syntax
void updateScanParams(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Stop any scans in progress first before using this function.

BLEScan::startScan

Description
Start BLE scanning.

Syntax
void startScan();
void startScan(uint32_t scanDuration_ms);

Parameters
scanDuration: BLE scan will stop after scanDuration milliseconds.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Set the scan parameters first before starting a scan. BLE scans will occur continuously for the duration set with BLEDevice::setScanWindow() and will repeat with a time interval set with BLEDevice::setScanInterval(). Call this member function without an argument to start scanning until BLEDevice::stopScan() is called.

BLEScan::stopScan

Description
Stop BLE scanning.

Syntax
void stopScan(void);

Parameters
The function requires no input paramter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEScan::setScanMode

Description
Set the BLE scan mode.

Syntax
void setScanMode(uint8_t scanMode);

Parameters
scanMode: GAP_SCAN_MODE_PASSIVE for passive scanning, GAP_SCAN_MODE_ACTIVE for active scanning

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Active scanning will request for scan response packets after discovering an advertising device. Passive scanning will only capture advertising data packets.

BLEScan::setScanInterval

Description
Set the BLE scan interval.

Syntax
void setScanInterval(uint16_t scanInt_ms);

Parameters
scanInt_ms: scan interval in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will repeat with a time interval set with this member function.

BLEScan::setScanWindow

Description
Set the BLE scan window.

Syntax
void setScanWindow(uint16_t scanWindow_ms);

Parameters
scanWindow_ms: scan window in milliseconds. Value range of 3 to 10240.

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
A BLE scan will scan continuously for a window duration set with this member function. The scan window should be less than or equal to the scan interval.

BLEScan::setScanDuplicateFilter

Description
Set the scan duplicate filter.

Syntax
void setScanDuplicateFilter(bool dupeFilter);

Parameters
dupeFilter: TRUE to enable duplicate filtering.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Enabling duplicate filters will ignore scan results for devices already discovered previously.

BLEScan::scanInProgress

Description
Set the scan duplicate filter.

Syntax
bool scanInProgress(void);

Parameters
The function requires no input paramter.

Returns
TRUE if BLE scanning is in progress.

Example Code
NA

Notes and Warnings
NA

BLEScan::printScanInfo

Description
Parse and print out scanned information.

Syntax
void printScanInfo(T_LE_CB_DATA* p_data);

Parameters
p_data: pointer to scan data of type T_LE_CB_DATA*

Returns
The function returns nothing.

Example Code
Example: BLEScan

Notes and Warnings
Use this member function to parse the various fields of received advertisement data packets and print the results out to the serial monitor.

Class BLEService

BLEService Class

Description
A class used for creating and managing BLE GATT services.

Syntax
class BLEService

Members

Public Constructors	
BLEService::BLEService	Constructs a BLEService object
Public Methods	
BLEService::setUUID	Set service UUID
BLEService::getUUID	Get service UUID
BLEService::addCharacteristic	Add a characteristic to service
BLEService::getCharacteristic	Get a previously added characteristic

BLEService::BLEService

Description

Constructs a BLEService object.

Syntax
BLEService::BLEService(BLEUUID uuid);
BLEService::BLEService(const char* uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object or a character array

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings
NA

BLEService::setUUID

Description
Set the service UUID.

Syntax
void setUUID(BLEUUID uuid);

Parameters
uuid: service UUID, expressed as a BLEUUID class object.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEService::getUUID

Description
Get the service UUID.

Syntax
BLEUUID getUUID();

Parameters
The function requires no input parameter.

Returns
The function returns the service UUID in a BLEUUID class object.

Example Code
NA

Notes and Warnings
NA

BLEService::addCharacteristic

Description
Add a characteristic to the service.

Syntax
void addCharacteristic(BLECharacteristic& newChar);

Parameters
newChar: the BLECharacteristic to add to the service.

Returns
The function returns nothing.

Example Code
Example: BLEUartService

Notes and Warnings

NA

BLEService::getCharacteristic

Description
Get a previously added characteristic.

Syntax
BLECharacteristic* getCharacteristic(uint8_t charIndex);

Parameters
charIndex: position index of characteristic.

Returns
The function returns a pointer to the BLECharacteristic at the requested position index.

Example Code
NA

Notes and Warnings
NA

Class BLEUUID

BLEUUID Class

Description
A class used for creating and managing UUIDs.

Syntax
class BLEUUID

Members

Public Constructors	
BLEUUID::BLEUUID	Create a UUID object
Public Methods	
BLEUUID::str	Get the character string representation of UUID
BLEUUID::data	Get the binary representation of UUID
BLEUUID::length	Get the length of UUID

BLEUUID::BLEUUID

Description
Create a UUID object from a UUID character string

Syntax
BLEUUID();
BLEUUID(const char* str);
BLEUUID(uint8_t* data, uint8_t length);

Parameters
str: UUID character string used to created object
data: pointer to byte array containing the desired UUID
length: number of bytes in array containing the desired UUID. Valid values of 2, 4 or 16

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

BLEUUID::str

Description
Get the character string representation of UUID

Syntax
const char* str(void);

Parameters
The function requires no input parameter.

Returns
Pointer to a character string representation of the UUID

Example Code
NA

Notes and Warnings

BLEUUID::data

Description
Get the binary representation of UUID

Syntax
const uint8_t* data(void);

Parameters
The function requires no input parameter.

Returns
Pointer to an unsigned 8-bit integer array containing the UUID expressed in binary form

Example Code
NA

Notes and Warnings
Returned pointer is of const uint8_t* type and will not allow changing of the data.

BLEUUID::length

Description
Get the length of UUID

Syntax
uint8_t length(void);

Parameters
The function requires no input parameter.

Returns
Length of the UUID, in terms of bytes

Example Code

NA

Notes and Warnings
A 4-character UUID will be 16 bits / 2 bytes long.
A 32-character UUID will be 128 bits / 16 bytes long.

Class BLEWifiConfigService

BLEWifiConfigService Class

Description
A class used for managing a BLE WiFi configuration service running on the device.

Syntax
class BLEWifiConfigService

Members

Public Constructors	
BLEWifiCon figService::BLEWifiConfigService	Only one instance of this class should be created

Public Methods	
BLEWifiConfigService::begin	Start background thread to process WiFi configuration commands
BLEWifiConfigService::end	Stop background thread processing WiFi configuration commands
BLEWifiConfigService::addService	Add the service to the BLE stack
BLEWifiConfigService::advData	Get advertising data correctly formatted for WiFi configuration service

BLEWifiConfigService::BLEWifiConfigService

Description
Create an instance of the BLEWifiConfigService object.

Syntax
void BLEWifiConfigService ();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
Only one instance of this class / service should be created.

BLEWifiConfigService::begin

Description
Start background thread to process WiFi configuration commands.

Syntax
void begin();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::end

Description
Stop background thread processing WiFi configuration commands.

Syntax
void end();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

BLEWifiConfigService::addService

Description
Add the WiFi configuration service to the BLE stack.

Syntax
void addService();

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: BLEWifiConfig

Notes and Warnings
NA

BLEWifiConfigService::advData

Description
Get advertising data correctly formatted for WiFi configuration service.

Syntax
BLEAdvertData advData();

Parameters
The function requires no input parameter.

Returns
The function returns a BLEAdvertData object that contains the required advertising data fields for the WiFi configuration service to work.

Example Code
Example: BLEWifiConfig

Notes and Warnings
The advertisement data needs to be correctly formatted for the corresponding smartphone app to recognise the device. WiFi configuration service advertisement data requires the local BT address, and should be called only after peripheral mode is started and may also require stopping and restarting the advertising process.

EPDIF

Class EpdIF

EpdIf Class

Description
A class used to control the electronic paper display internal functions.

Syntax
class EpdIf

Members

Public Constructors
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named EpdIf.

Public Methods	
EpdIf::EPD_Dis_Part	Put an image buffer to the frame memory, but not updating the display
EpdIf::EPD_SetFrame	Put display data to the frame memory, usually used for setup text display functions
EpdIf::EPD_SetRAMValue_BaseMap	To read image data stored in the RAM, but not display on the screen
EpdIf::EPD_SetFrameMemory	To read image data stored in the buffer, but not display on the screen
EpdIf::EPD_UpdateDisplay	Update the display
EpdIf::EPD_ClearScreen_White	Clear the frame memory with the White color, but not updating the display
EpdIf::EPD_ClearScreen_Black	Clear the frame memory with the Black color, but not updating the display
EpdIf::EPD_Busy	Wait until the Busy pin goes to low, which is the idle state
EpdIf::EPD_Reset	Used for the Epaper module reset. Often used to awaken the module in deep sleep
EpdIf::EPD_Sleep	After this command is transmitted, the chip would enter the deep-sleep mode to save power

EpdIf:: EPD_Dis_Part

Description

Put an image buffer to the frame memory, but not updating the display.

Syntax

void EPD_Dis_Part(unsigned int x_start, unsigned int y_start, const unsigned char* datas, unsigned int PART_COLUMN, unsigned int PART_LINE);

Parameters

x_start: starting position of the x-axis
y_start: starting position of the y-axis
datas: data to be displayed on the e-paper module
PART_COLUMN: height of the display area
PART_LINE: width of the display area

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

EpdIf:: EPD_SetFrame

Description

Put display data to the frame memory, usually used for setup text display functions.

Syntax
void EPD_SetFrame(const unsigned char* image_buffer, int x, int y, int image_width, int image_height);

Parameters
image_buffer: the buffer which stores the data to be displayed on the e-paper module, usually used to display texts.
x: starting position of the x-axis
y: starting position of the y-axis
image_width: width of the display area
image_height: height of the display area

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetRAMValue_BaseMap

Description
To read image data stored in the RAM, but not display on the screen.

Syntax
void EPD_SetRAMValue_BaseMap(const unsigned char* datas);

Parameters
datas: contains the black and white information that forms the image stored in RAM

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_SetFrameMemory

Description
To read image data stored in the buffer but not display on the screen.

Syntax
void EPD_SetFrameMemory(const unsigned char* image_buffer);

Parameters
image_buffer: the buffer where stores the image data in hexadecimal numbers

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf:: EPD_UpdateDisplay

Description
Update the ePaper display module. Always combined used with functions set the frames.

Syntax
void EPD_UpdateDisplay(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

There are 2 memory areas embedded in the e-paper display but once this function is called, then the next action of SetFrameMemory or ClearScreen will set the other memory area.

EpdIf:: EPD_ClearScreen_White

Description
Clear the frame memory with the White color.

Syntax
void EpdIf::EPD_ClearScreen_White(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_ClearScreen_Black

Description
Clear the frame memory with the Black color.

Syntax
void EpdIf::EPD_ClearScreen_Black(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Busy

Description
Wait until the busy_pin goes to low, which is the idle state.

Syntax
void EpdIf::EPD_Busy(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
If the users want to see the actual display on the e-paper screen, the function EPD_UpdateDisplay() is required to be added behind this code.

EpdIf:: EPD_Reset

Description
This command will let the E-paper module reset, it is often used to awaken the module in while it's in the deep sleep mode, you will find more details in the function EpdIf:: EPD_Sleep().

Syntax
void EpdIf::EPD_Reset(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

EpdIf::EPD_Sleep

Description
After this command is transmitted, the chip would enter the deep-sleep mode to save power. The deep sleep mode would return to standby by hardware reset. You can use EPD:: Init() to awaken the E-paper module.

Syntax
void EpdIf::EPD_Sleep(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

FatfsSDCard

Class SdFatFs

Description

Defines a class of SD FAT File system.

Syntax

class SdFatFs

Members

Public Constructors

SdFatFs::SdFatFs Constructs a SdFatFs object

SdFatFs::~SdFatFs Destructs a SdFatFs object

Public Methods

SdFatFs::begin	Initialize SD FAT File System
SdFatFs::end	Deinitialize SD FAT File System
SdFatFs::*getRootPath	Get the root path of the SD FAT File System
SdFatFs::readDir	List items under a specific folder
SdFatFs::mkdir	Create folder
SdFatFs::rm	Remove folder or file
SdFatFs::isDir	Check if a specific path is a directory
SdFatFs::isFile	Check if a specific path is a file
SdFatFs::getLastModTime	Get the last modified time for a file or directory
SdFatFs::setLastModTime	Set the last modified time for a file or directory
SdFatFs::status	Return the current status of SD
SdFatFs::open	Open a file

SdFatFs::begin

Description
Initialize SD FAT File System.

Syntax
int SdFatFs::begin(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::end

Description
De-initialize SD FAT File System.

Syntax
int SdFatFs::end(void);

Parameters
The function requires no input parameter.

Returns
Returns “0” if success, else returns a negative value.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::*getRootPath

Description
Get the root path of the SD FAT File System. The logical volume character is starting from ‘0’, so the root path would like “0:/”.

Syntax
char *SdFatFs::getRootPath(void);

Parameters
The function requires no input parameter.

Returns
The function returns the root path.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::readDir

Description
List items under a specific folder. List items under a specific folder and store the result in the buffer that user specified. Each item is separated by '0'.

Syntax
int SdFatFs::readDir(char *path, char *result_buf, unsigned int bufsize);

Parameters
path: The absolute directory path to be listed.
result_buf: The buffer to be stored results.
bufsize: The size of result_buf. If results exceed this size, then the results larger than this size would be discarded.

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: get_file_attribute; list_root_files

Notes and Warnings
Include "SdFatFs.h" to use the class function.

SdFatFs::mkdir

Description
Create folder.

Syntax
int SdFatFs::mkdir(char *absolute_path);

Parameters
absolute_path: The absolute directory path to be created

Returns
Returns "0" if success, else returns a negative value.

Example Code
Example: create_folder

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::rm

Description
Remove folder or file.

Syntax
int SdFatFs::rm(char *absolute_path);

Parameters
absolute_path: The absolute directory or file path to be deleted

Returns
Returns “0” if success, else returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isDir

Description
Check if a specific path is a directory.

Syntax
unsigned char SdFatFs::isDir(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::isFile

Description
Check if a specific path is a file.

Syntax
unsigned char SdFatFs::isFile(char *absolute_path);

Parameters
absolute_path: The absolute path to be queried

Returns
The function returns “1” if it is a directory, else returns “0”.

Example Code
Example: get_file_attribute

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::getLastModTime

Description
Get the last modified time for a file or directory.

Syntax
int SdFatFs::getLastModTime(char *absolute_path, uint16_t *year, uint16_t *month, uint16_t *date, uint16_t *hour, uint16_t *minute, uint16_t *second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.

month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::setLastModTime

Description
Set the last modified time for a file or directory. Ameba doesn’t have built-in RTC. So we manually change file/directory last modified time.

Syntax
int SdFatFs::setLastModTime(char *absolute_path, uint16_t year, uint16_t month, uint16_t date, uint16_t hour, uint16_t minute, uint16_t second);

Parameters
absolute_path: The absolute path to be queried.
year: The value of the year.
month: The value of the month.
date: The value of the date.
hour: The value of an hour.
minute: The value of a minute.
second: field “second” contains no valid information in the current version.

Returns
The function returns “0” if success, otherwise returns a negative value for failure.

Example Code
Example: last_modified_time

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::open

Description
Open a file.

Syntax
SdFatFile SdFatFs::open(char *absolute_path);

Parameters
absolute_path: The path to a file.

Returns
The file object is an instance of SdFatFile.

Example Code
Example: create_folder; file_read_write; get_file_attribute; last_modified_time; list_root_files.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

SdFatFs::status

Description
Return the current status of SD.

Syntax
int SdFatFs::status(void);

Parameters
The function requires no input parameter.

Returns
Function returns “1” if ready to use, else return “0” if the status is inactivating or abnormal.

Example Code
NA.

Notes and Warnings
Include “SdFatFs.h” to use the class function.

Class SdFatFile

Description

Defines a class of SD FAT File.

Syntax
class SdFatFile

Members

Public Constructors	
SdFatFile::SdFatFile	Constructs a SdFatFile object
SdFatFile::~SdFatFile	Destructs a SdFatFile object
Public Methods	
SdFatFile::write	Write 1 byte/bytes to file
SdFatFile::read	Read 1 byte/bytes from the file
SdFatFile::peek	Read 1 byte from file without move curser
SdFatFile::available	Check if the cursor is at EOF (End-Of-File)
SdFatFile::bool	Check if file is opened
SdFatFile::seek	Change cursor to a specific position
SdFatFile::close	Close file

SdFatFile::write

Description
Write 1 byte or bytes to the file.

Syntax
size_t SdFatFile::write(uint8_t c);
size_t SdFatFile::write(const uint8_t *buf, size_t size);

Parameters
c: The character to be written.
buf: The buffer to be written.
size: The length of buffer to be written.

Returns

The function returns the number of byte count that has been successfully written to the file.

Example Code
NA.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: read

Description
Read 1 byte or bytes from the file.

Syntax
int SdFatFile::read(void);
int SdFatFile::read(void *buf, uint16_t nbyte);

Parameters
buf: The buffer to store the content.
nbyte: The buffer size. (Or can be regarded as the desired length to read).

Returns
The function returns a read character or the read size of the buffer.

Example Code

```
1. #include “FatFs_SD.h”
2.
3. char dirname[] = “testdir”;
4. char filename[] = “test.txt”;
5. char write_content[] = “hello world!”;
6.
7. FatFsSD fs;
8.
9. void setup() {
10. char buf[128];
11. char absolute_filename[128];
12.
13. fs.begin();
```

```
14.
15. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), dirname);
16. fs.mkdir(absolute_filename);
17. printf(“create dir at \”%s"rn”, absolute_filename);
18.
19. sprintf(absolute_filename, “%s%s/%s”, fs.getRootPath(), dirname, filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.println(write_content);
22. file.close();
23. printf(“create file at \”%s"rn”, absolute_filename);
24.
25. printf(“read back from \”%s"rn”, absolute_filename);
26. file = fs.open(absolute_filename);
27.
28. memset(buf, 0, sizeof(buf));
29. file.read(buf, sizeof(buf));
30.
31. file.close();
32. printf(“==== content ====rn”);
33. printf(“%s”, buf);
34. printf(“==== end ====rn”);
35.
36. fs.end();
37. }
38.
39. void loop() {
40. delay(1000);
41. }
42.
```

Example: create_folder;
This example shows how to create a folder and open a file under it.

```
1. #include “FatFs_SD.h”
2.
3. char filename[] = “test.txt”;
4. char write_content[] = “hello world!”;
```

```
5.
6.  FatFsSD fs;
7.
8.  void setup() {
9.  char buf[128];
10. char absolute_filename[128];
11.
12. fs.begin();
13.
14. printf("write something to \"%s"rn", filename);
15. sprintf(absolute_filename, "%s%s", fs.getRootPath(), filename);
16. SdFatFile file = fs.open(absolute_filename);
17.
18. file.println(write_content);
19.
20. file.close();
21. printf("write finishrnrn");
22.
23. printf("read back from \"%s"rn", filename);
24. file = fs.open(absolute_filename);
25.
26. memset(buf, 0, sizeof(buf));
27. file.read(buf, sizeof(buf));
28.
29. file.close();
30. printf("==== content ====rn");
31. printf("%s", buf);
32. printf("==== end ====rn");
33.
34. fs.end();
35. }
36.
37. void loop() {
38. delay(1000);
39. }
40.
```

Example: file_read_write;
This example shows how to open/close files and perform read/write to it.

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: peek

Description
Read one byte from the file without moving the curser.

Syntax
int SdFatFile::peek(void);

Parameters
The function requires no input parameter.

Returns
The function returns the read character as an integer number.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: available

Description
Check if the cursor is at EOF.

Syntax
int SdFatFile::available(void);

Parameters
The function requires no input parameter.

Returns
The function returns “0” if the cursor is at EOF, else returns “1”.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: flush

Description
It is a nop. This is an inherited function from class Stream. And it does not affect SD File.

Syntax
void SdFatFile::flush(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” to use the class function.

SdFatFile:: seek

Description
Change cursor to a specific position.

Syntax
int SdFatFile::seek(uint32_t pos);

Parameters
pos: The desired position.

Returns
The function returns 0 if success otherwise returns a negative value.

Example Code
NA

Notes and Warnings
Include “SdFatFile.h” in order to use the class function.

SdFatFile:: close

Description
Close file.

Syntax
int SdFatFile::close(void);

Parameters
The function requires no input parameter.

Returns
The function returns 0 if runs successfully otherwise it returns a negative value.

Example Code
Example: last_modified_time;
The example shows how to get and set last modified time of a file.

Example: create_folder;
This example shows how to create a folder and open a file under it. The details of the code can be found in the section of SdFatFile:: read.
Example: file_read_write;
This example shows how to open/close files and perform read/write to it. The details of the code can be found in the section of SdFatFile:: read.

1. #include <FatFs_SD.h>
2.
3. FatFsSD fs;
4.
5. char filename[] = “test.txt”;
6.
7. void setup() {
8. char absolute_filename[128];
9.

```
10. uint16_t year = 2021;
11. uint16_t month = 4;
12. uint16_t date = 4;
13. uint16_t hour = 12;
14. uint16_t minute = 12;
15. uint16_t second = 12;
16.
17. fs.begin();
18.
19. sprintf(absolute_filename, “%s%s”, fs.getRootPath(), filename);
20. SdFatFile file = fs.open(absolute_filename);
21. file.close();
22.
23. fs.setLastModTime(absolute_filename, year, month, date, hour, minute, second);
24.
25. fs.getLastModTime(absolute_filename, &year, &month, &date, &hour, &minute, &second);
26. printf(“filename:"%s"rn”, absolute_filename);
27. printf(“time mod:%04d/%02d/%02d %02d:%02d:%02drn”, year, month, date, hour, minute, second);
28.
29. fs.end();
30. }
31.
32. void loop() {
33. delay(1000);
34. }
35.
```

Notes and Warnings

Include “SdFatFile.h” in order to use the class function.

FlashMemory

Class EpdIF

FlashMemoryClass Class

Description
Defines a class of Flash memory API

Syntax
class FlashMemoryClass

Members

Public Constructors	
Fl ashMemoryClass::FlashMemoryClass	Constructs a FlashMemoryClass object
Fla shMemoryClass::~FlashMemoryClass	Deconstructs a FlashMemoryClass object
Public Methods	
FlashMemoryClass::begin	Initialize/Re-initialize the base address and size
FlashMemoryClass::read	Read the content to buf
FlashMemoryClass::update	Write buf back to flash memory
FlashMemoryClass::readWord	Read 4 bytes from flash memory
FlashMemoryClass::writeWord	Write 4 bytes into flash memory
FlashMemoryClass::buf_size	The buf size
FlashMemoryClass::*buf	The buf to be operated

FlashMemoryClass::FlashMemoryClass

Description
Constructs a FlashMemoryClass object.

Syntax
FlashMemoryClass(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address to operate.
_buf_size: The buf size for mirror a copy to reduce flash memory operation

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Direct read from flash memory is allowed. To write data into flash memory, each bit on flash memory can only change from '1' to '0' and it cannot change from '0' to '1'. To make sure the data are correctly written we do erase the flash memory sector before write data on it.

```
#include <FlashMemory.h>
void setup() {
FlashMemory.read();
if (FlashMemory.buf[0] == 0xFF) {
FlashMemory.buf[0] = 0x00;
FlashMemory.update();
Serial.println("write count to 0");
} else {
FlashMemory.buf[0]++;
FlashMemory.update();
Serial.print("Boot count: ");
Serial.println(FlashMemory.buf[0]);
}
}
void loop() {
delay(1000);
}
```

Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

```
#include <FlashMemory.h>
void setup() {
unsigned int value;
/* request flash size 0x4000 from 0xFC000 */
FlashMemory.begin(0xFC000, 0x4000);
/* read one word (32-bit) from 0xFC000 plus offset 0x3F00 */
value = FlashMemory.readWord(0x3F00);
printf("value is 0x%08Xrn", value);
if (value == 0xFFFFFFFF) {
value = 0;
```

```
} else {
value++;
}
/* write one word (32-bit) to 0xFC000 plus offset 0x3F00 */
FlashMemory.writeWord(0x3F00, value);
}
void loop() {
// put your main code here, to run repeatedly:
}
```

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::begin

Description
Initialize/Re-initialize the base address and size. The base address shell aligns with the size of 0x1000. And the size shell is multiple of 0x1000.

Syntax
void begin(unsigned int _base_address, unsigned int _buf_size);

Parameters
_base_address: The base address
_buf_size: The desired work size

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include “FlashMemory.h” to use the class function.

FlashMemoryClass::read

Description
Read the content to buf. Read flash memory into the buf. The size would be 0x1000.

Syntax
void read(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: FleshMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba’s flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of “FlashMemoryClass:: FlashMemoryClass”.

Notes and Warnings
Include “FlashMemory.h” to use the class function.

FlashMemoryClass::update

Description
Write buf back to flash memory. Write flash memory with the content of the buffer. The size is 0x1000.

Syntax
void update(bool erase = true);

Parameters
erase: By default, it is true and erases flash memory before writing to it

Returns

The function returns nothing.

Example Code

Example: FleshMemory_Basic

This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::readWord

Description

Read 4 bytes from flash memory. Read 4 byte from specific offset based on base address.

Syntax

unsigned int readWord(unsigned int offset);

Parameters

offset: The offset according to the base address

Returns

The read data with a size of 4 bytes

Example Code

Example: ReadWriteOneWord

This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).

Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings

Include "FlashMemory.h" to use the class function.

FlashMemoryClass::writeWord

Description

Write 4 bytes into flash memory. It will try to write 4 bytes first. If the read data differ from the write data, then we buffer the sector of flash memory, erase it, and write correct data back to it.

Syntax
void writeWord(unsigned int offset, unsigned int data);

Parameters
offset: The offset according to the base address
data: The data to be written

Returns
The function returns nothing.

Example Code
Example: ReadWriteOneWord
This example shows how to request flash memory larger than default 0x4000, and read/write one specific word (32-bit).
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::buf_size

Description
The buf size (It can be regarded as work size).

Syntax
unsigned int buf_size;

Example Code
Example: FlashMemory_Basic
This example demonstrates the basic use of flash memory. Since boot count is stored in flash, each time upon device boot up, the boot count will be read from the flash, add one, then write back to the flash. Ameba's flash memory can be edit in a unit of a sector which has the size of 4K bytes.
Details of the example codes can be found in the previous section of "FlashMemoryClass:: FlashMemoryClass".

Notes and Warnings
Include "FlashMemory.h" to use the class function.

FlashMemoryClass::*buf

Description
The buf to be operated. Modify buf won't change the content of the buf. It needs an update to write back to flash memory.

Syntax
unsigned char *buf;

Example Code
NA

Notes and Warnings
Include "FlashMemory.h" to use the class function.

GPIO

Class DHT

DHT Class

Description
Defines a class of using DHT temperature & humidity sensors

Syntax
class DHT

Members

Public Constructors	
DHT::DHT	Constructs a DHT object
Public Methods	
DHT::begin	Initialize the DHT sensor
DHT::readTemperature	Read temperature(Fahrenheit or Celcius) from the DHT sensor
DHT::convertCtoF	Convert a value from Celcius to Fahrenheit
DHT::convertFtoC	Convert a value from Fahrenheit to Celcius
DHT::readHumidity	Read humidity(%) from the DHT sensor
DHT::computeHeatIndex	Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations)
DHT::read	Check if the sensor is readable

DHT::DHT

Description
Constructs a DHT object.

Syntax
DHT::DHT(uint8_t pin, uint8_t type, uint8_t count)

Parameters
pin: The Arduino digital PIN connected
type: The DHT sensor type(DHT11, DHT22, or DHT21)
count: The count is now ignored as the DHT reading algorithm adjusts itself based on the speed of the processor

Returns
The function returns nothing.

Example Code
Example:DHTTester
The code demos basic testing for various DHT humidity & temperature sensors.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

// Example testing sketch for various DHT humidity/temperature sensors

// Written by ladyada, public domain

#include "DHT.h"

// The digital pin we're connected to.

#define DHTPIN 8

// Uncomment whatever type you're using!

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V

// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1

// to 3.3V instead of 5V!

// Connect pin 2 of the sensor to whatever your DHTPIN is

// Connect pin 4 (on the right) of the sensor to GROUND

// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.

// Note that older versions of this library took an optional third parameter to

```
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(115200);
Serial.println("DHTxx test!");
dht.begin();
}
void loop() {
// Wait a few seconds between measurements.
delay(2000);
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *Ft");
Serial.print("Heat index: ");
```

```
Serial.print(hic);
Serial.print(" *C ");
Serial.print(hif);
Serial.println(" *F");
}
```

DHT::begin

Description
Initialize the DHT sensor.

Syntax
void DHT::begin(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::readTemperature

Description
Read temperature(Fahrenheit or Celcius) from the DHT sensor.

Syntax
float DHT::readTemperature(bool S, bool force);

Parameters
S: Temperature scale, True is Fahrenheit and False is Celcius
force: Index of checking sensor readability, default is False

Returns
The function returns the current temperature as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertCtoF

Description
Convert a value from Celcius to Fahrenheit.

Syntax
float DHT::convertCtoF(float c);

Parameters
c: The value in Celcius

Returns
The function returns the temperature in Fahrenheit as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::convertFtoC

Description
Convert a value from Fahrenheit to Celcius.

Syntax
float DHT::convertFtoC(float f);

Parameters
f: The value in Fahrenheit

Returns
The function returns the temperature in Celcius as a float number.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

DHT::computeHeatIndex

Description
Compute the HeatIndex from the readings (Using both Rothfusz and Steadman's equations). More details refer to http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml .

Syntax
float DHT::computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit);

Parameters
temperature: The temperature value
percentHumidity: The humidity percent value
isFahrenheit: True, temperature value in Fahrenheit (Default); False, temperature value in Celcius

Returns
The function returns the heat index in Fahrenheit or Celsius as a float value.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings

Every time must include the header file “DHT.h” in front of the project to use the class function.

DHT::readHumidity

Description
Reading temperature or humidity from the DHT sensor and return as a float value(%).

Syntax
float DHT::readHumidity(bool force);

Parameters
force: Ignored.

Returns
The function returns current humidity in a float number (in %).

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the “DHT: DHT” section.

Notes and Warnings
Every time must include the header file “DHT.h” in front of the project to use the class function. Reading temperature or humidity takes about 250 milliseconds! Sensor readings may also be up to 2 seconds.

DHT::read

Description
Check if the sensor is readable.

Syntax
boolean DHT::read(bool force);

Parameters
force: Index of whether checking the sensor was read less than two seconds ago or not. False, checking; True, not checking.

Returns

Return the last correct measurement of the sensor. False, low means not readable; True, high means readable.

Example Code
Example: DHTTester
The code demos basic testing for various DHT humidity & temperature sensors. Please refer to code in the "DHT: DHT" section.

Notes and Warnings
Every time must include the header file "DHT.h" in front of the project to use the class function.

Class HttpClient

InterruptLock Class

Description
Defines a class of turning off/on interrupts temporarily

Syntax
class InterruptLock

Members

Public Constructors	
InterruptLock::InterruptLock	Constructs a InterruptLock object
InterruptLock::~ InterruptLock	Deconstructs a InterruptLock object

---

GTimer

Class EpdIF

GTimerClass Class

Description
GTimer is a hardware timer and this class is to operate it. The GTimer occupy same resource as PWM. Please make sure the timer is not conflict with you PWM index.

Syntax
class GTimerClass

Members

Public Constructors	
GTimerClass::GTimerClass	Constructs a GTimerClass object
Public Methods	
GTimerClass::begin	Initialize a timer and start it immediately
GTimerClass::stop	Stop a specific timer
GTimerClass::reload	Reload a specific timer
GTimerClass::read_us	Read current countdown value

GTimerClass::begin

Description
Initialize a timer and start it immediately.

Syntax
void GTimerClass::begin(uint32_t timerid, uint32_t duration_us, void (*handler)(uint32_t), bool periodical, uint32_t userdata);

Parameters
timerid: There are 5 valid GTimer with timer id 0~4.
duration_us: The duration of the timer. The time unit is microsecond and the precision is 32768Hz.
periodical: By default, the timer would keep periodically countdown and reload which means the handler would periodically be invoked.
userdate: The user data brings to the handler.

Returns
The function returns nothing.

Example Code
Example: TimerOneshot

/*

This sketch shows how to use several hardware timers in invoke handler only once for each timer.

*/

#include <GTimer.h>

void myhandler(**uint32_t** data) {

Serial.print("I am timer!");

Serial.println(data);

}

void setup() {

```
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// timerid 0, period 1s, invoke myhandler, invoke only once, user data is 0
GTimer.begin(0, 1 * 1000 * 1000, myhandler, false, 0);
// timerid 1, period 2s, invoke myhandler, invoke only once, user data is 1
GTimer.begin(1, 2 * 1000 * 1000, myhandler, false, 1);
GTimer.begin(2, 3 * 1000 * 1000, myhandler, false, 2);
GTimer.begin(3, 4 * 1000 * 1000, myhandler, false, 3);
}
void loop() {
delay(1000);
}
```

Example: TimerPeriodical

```
/*
This sketch shows how to use hardware timer and invoke interrupt handler periodically
*/
#include <GTimer.h>
int counter = 0;
void myhandler(uint32_t data) {
counter++;
Serial.print("counter: ");
Serial.println(counter);
if (counter >= 10) {
Serial.println("stop timer");
GTimer.stop(0);
}
}
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(115200);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
```

```
// timerid 0, period 1s, invoke myhander
GTimer.begin(0, (1 * 1000 * 1000), myhandler);
}
void loop() {
delay(1000);
}
```

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::stop

Description
Stop a specific timer

Syntax
void GTimerClass::stop(uint32_t timerid);

Parameters
timerid: Stop the timer with this timer id

Returns
The function returns nothing.

Example Code
Example: TimerPeriodical, please refer to GTimerClass:: begin for more details.

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::reload

Description
Reload a specific timer. The GTimer is a countdown timer. Reload it would make it discard the current countdown value and restart countdown based on the duration.

Syntax

void GTimerClass::reload(uint32_t timerid, uint32_t duration_us);

Parameters
timerid: The timer to be modified
duration_us: The updated duration in unit of microseconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

GTimerClass::read_us

Description
Read the current countdown value

Syntax
uint64_t GTimerClass::read_us(uint32_t timerid);

Parameters
timerid: The timer to be read

Returns
The function returns the current countdown value.

Example Code
NA

Notes and Warnings
Include “GTimer.h” to use the class function.

Http

Class HttpClient

HttpClient Class

Description
Defines a class of using HttpClient

Syntax
class HttpClient

Members

Public Constructors	
HttpClient::HttpClient	Constructs a HttpClient object
Public Methods	
HttpClient::beginRequest	Start a more complex request
HttpClient::endRequest	End a more complex request
HttpClient::get	Connect to the server and start to send a GET request
HttpClient::post	Connect to the server and start to send a POST request
HttpClient::put	Connect to the server and start to send a PUT request
HttpClient::startRequest	Connect to the server and start to send the request
HttpClient::sendHeader	Send an additional header line
HttpClient::sendBasicAuth	Send a basic authentication header
HttpClient::finishRequest	Finish sending the HTTP request
HttpClient::responseStatusCode	Get the HTTP status code contained in the response
HttpClient::readHeader	Read the next character of the response headers
HttpClient::skipResponseHeaders	Skip any response headers to get to the body
HttpClient::endOfHeadersReached	Test whether all of the response headers have been consumed
HttpClient::endOfBodyReached	Test whether the end of the body has been reached
HttpClient::contentLength	Return the length of the body

HttpClient::HttpClient

Description
Constructs a HttpClient object. If Marco “PROXY_ENABLED” is defined, currently disabled as introduces a dependency on DNS.h in Ethernet.

Syntax
HttpClient::HttpClient(Client& aClient, const char* aProxy = NULL, uint16_t aProxyPort = 0);
HttpClient::HttpClient(Client& aClient);

Parameters
aClient: The object of class WiFiClient.

aProxy: The proxy name. The default proxy name is “NULL”.
aProxyPort: The proxy port. The default value for the proxy port is 0.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrate how to download the content from URL indicated in kHostname[].

```
#include <HttpClient.h>
#include <WiFi.h>
#include <WiFiClient.h>
char ssid[] = “YourNetwork”; // your network SSID (name)
char pass[] = “password”; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
// Name of the server we want to connect to
const char kHostname[] = “www.google.com”;
const char kPath[] = “/”;
// Number of milliseconds to wait without receiving any data before we give up
const int kNetworkTimeout = 30*1000;
// Number of milliseconds to wait if no data is available before trying again
const int kNetworkDelay = 1000;
int status = WL_IDLE_STATUS;
void setup() {
Serial.begin(9600);
while ( status != WL_CONNECTED) {
Serial.print(“Attempting to connect to SSID: “);
Serial.println(ssid);
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println(“Connected to wifi”);
printWifiStatus();
}
void loop() {
```

```
int err =0;
WiFiClient c;
HttpClient http(c);
err = http.get(kHostname, kPath);
if (err == 0)
{
Serial.println("startedRequest ok");
err = http.responseStatusCode();
if (err >= 0)
{
Serial.print("Got status code: ");
Serial.println(err);
// Usually you'd check that the response code is 200 or a
// similar "success" code (200-299) before carrying on,
// but we'll print out whatever response we get
err = http.skipResponseHeaders();
if (err >= 0)
{
int bodyLen = http.contentLength();
Serial.print("Content length is: ");
Serial.println(bodyLen);
Serial.println();
Serial.println("Body returned follows:");
// Now we've got to the body, so we can print it out
unsigned long timeoutStart = millis();
char c;
// Whilst we haven't timed out & haven't reached the end of the body
while ( (http.connected() || http.available()) &&
((millis() - timeoutStart) < kNetworkTimeout) )
{
if (http.available())
{
c = http.read();
// Print out this character
Serial.print(c);
bodyLen--;
```

```
// We read something, reset the timeout counter
timeoutStart = millis();
}
else
{
// We haven't got any data, so let's pause to allow some to arrive
delay(kNetworkDelay);
}
}
}
else
{
Serial.print("Failed to skip response headers: ");
Serial.println(err);
}
}
else
{
Serial.print("Getting response failed: ");
Serial.println(err);
}
}
else
{
Serial.print("Connect failed: ");
Serial.println(err);
}
http.stop();
// And just stop, now that we've tried a download
while(1);
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
```

```
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::beginRequest

Description
Start a more complex request. Use this when you need to send additional headers in the request, but you will also need to call endRequest() when you are finished.

Syntax
void HttpClient::beginRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::endRequest

Description

End a more complex request. Use this when you need to have sent additional headers in the request, but you will also need to call beginRequest() at the start.

Syntax

void HttpClient::endRequest(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::get

Description

Connect to the server and start to send a "GET" request. If the input parameter contains "aServerAddress", the connection will not perform a DNS lookup and just purely connect to the given IP address.

Syntax

int HttpClient::get(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::get(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: The name of the server being connected to. If aServerName is "NULL", the "Host" header line will not be sent.

aServerPort: The port on which server connected.

aURLPath: The URL to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::post

Description

Connect to the server and start to send a "POST" request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::post(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::post(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::put

Description

Connect to the server and start to send a PUT request. If the input parameter has "aServerAddress", connects doesn't perform a DNS lookup and just connects to the given IP address.

Syntax

int HttpClient::put(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aUserAgent = NULL);

int HttpClient::put(const IPAddress& aServerAddress, const char* aServerName, const char* aURLPath, const char* aUserAgent = NULL);

Parameters

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aUserAgent: User-Agent string to be sent. If aUserAgent indicated as "NULL", the default user-agent kUserAgent will be sent.

aServerAddress: IP address of the server to connect to.

Returns

Return 0 if successful, otherwise indicates an error occurs.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::startRequest

Description

Connect to the server and start to send the request.

Syntax

int HttpClient::startRequest(const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

int HttpClient::startRequest(const IPAddress& aServerAddress, const char* aServerName, uint16_t aServerPort, const char* aURLPath, const char* aHttpMethod, const char* aUserAgent);

Parameters

aServerAddress: IP address of the server to connect to.

aServerName: Name of the server being connected to. If NULL, the "Host" header line won't be sent.

aServerPort: Port to connect to on the server.

aURLPath: Url to request.

aHttpMethod: Type of HTTP request to make, e.g. "GET", "POST", etc.

aUserAgent: User-Agent string to send. If NULL the default user-agent kUserAgent will be sent.

Returns

Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include "HttpClient.h" to use the class function.

HttpClient::sendHeader

Description

The function sends an additional header line.

The function void HttpClient:: sendHeader(const char* aHeader);can only be called in between the calls to startRequest and finishRequest.

The other 2 functions void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue); and void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue); are alternate form the previous one, which takes the header name and content as separately (as strings or integer). For example, to send an XXXXXX header, user might call sendHeader("XXXXX", "Something") or sendHeader("XXXXX", 123).And the call will add the ": " in the log to separate different header in the case of multiple headers.

Syntax

void HttpClient::sendHeader(const char* aHeader);

void HttpClient::sendHeader(const char* aHeaderName, const char* aHeaderValue);

void HttpClient::sendHeader(const char* aHeaderName, const int aHeaderValue);

Parameters
aHeader: Header line to send, in its entirety (but without the trailing CRLF. E.g. "Authorization: Basic YQDDCAIGES".
aHeaderName: Type of header being sent.
aHeaderValue: Value for that header.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include "HttpClient.h" to use the class function.

HttpClient::sendBasicAuth

Description
The function sends a basic authentication header which will encode the given username and password, and send them in a suitable header line for doing Basic Authentication.

Syntax
void HttpClient::sendBasicAuth(const char* aUser, const char* aPassword);

Parameters
aUser: Username for the authorization.
aPassword: Password for the user aUser.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include “HttpClient.h” to use the class function.

HttpClient::finishRequest

Description
Finish sending the HTTP request. The function sends a blank line to signify the end of the request.

Syntax
void HttpClient::finishRequest(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::responseStatusCode

Description
Get the HTTP status code contained in the response. For example, “200” for successful requests, “404” for file not found, etc.

Syntax
int HttpClient::responseStatusCode(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include “HttpClient.h” to use the class function.

HttpClient::readHeader

Description

The function reads the next character of the response headers. This functions the same as read() but to be used when reading through the headers which are slightly less efficient. The user might check whether the end of the headers has been reached by calling endOfHeadersReached(), although after that point this will still return data as read() would.

Syntax

int HttpClient::readHeader(void);

Parameters

The function requires no input parameter.

Returns

Return the next character of the response headers.

Example Code

Example: SimpleHttpExample

The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings

Include “HttpClient.h” to use the class function.

HttpClient::skipResponseHeaders

Description

Skip any response headers to get to the body. Use this if you don’t want to do any special processing of the headers returned in the response. You can also use it after you’ve found all of the headers you’re interested in, and just want to get on with processing the body.

Syntax

int HttpClient::skipResponseHeaders(void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfHeadersReached

Description
Test whether all of the response headers have been consumed.

Syntax
bool HttpClient::endOfHeadersReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now processing the response body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::endOfBodyReached

Description

Test whether the end of the body has been reached. It only works if the Content-Length header was returned by the server.

Syntax
bool HttpClient::endOfBodyReached(void);

Parameters
The function requires no input parameter.

Returns
Return true if we are now at the end of the body, else false.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

HttpClient::contentLength

Description
The function returns the length of the body.

Syntax
int HttpClient::contentLength(void);

Parameters
The function requires no input parameter.

Returns
Return Length of the body, in bytes, or kNoContentLengthHeader if no Content-Length header was returned by the server.

Example Code
Example: SimpleHttpExample
The example demonstrates how to download the content from the URL indicated in kHostname[]. Details of the code can be found in the previous section of HttpClient:: HttpClient.

Notes and Warnings
Include “HttpClient.h” to use the class function.

IRDevice

Class HttpClient

IRDevice Class

Description
A class used for managing, sending, and receiving data using IR.

Syntax
class IRDevice

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named IR.	

Public Methods	
IRDevice::getFreq	Get the current IR modulation frequency
IRDevice::begin	Allocate resources and start the IR device with a custom frequency
IRDevice::end	Stop the IR device operations and free up resources
IRDevice::send	Send IR raw data
IRDevice::beginNEC	Allocate resources and start the IR device with a frequency suitable for the NEC protocol
IRDevice::sendNEC	Send data using the NEC protocol
IRDevice::recvNEC	Receive data using the NEC protocol

IRDevice::getFreq

Description
Get the current IR modulation frequency.

Syntax
uint32_t getFreq(void);

Parameters
The function requires no input parameter.

Returns
Currently set IR modulation frequency in Hertz.

Example Code
NA

Notes and Warnings
NA

IRDevice::begin

Description
Allocate resources and start the IR device with a custom frequency.

Syntax
void begin(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode, uint32_t freq);

Parameters
receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX
freq: IR modulation frequency in Hertz

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
IR device can only operate in either transmit or receive mode.

IRDevice::end

Description
Stop the IR device operations and free up resources.

Syntax

void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

IRDevice::send

Description
Send IR raw data.

Syntax
void send(const unsigned int buf[ ] , uint16_t len);

Parameters
buf[ ] : IR raw signals (in us) in an array form.
len: total length of the IR raw signal array.

Returns
The function returns nothing.

Example Code

#include "IRDevice.h"

// User defined txPin, rxPin and carrier frequency

#define IR_RX_PIN 8

#define IR_TX_PIN 9

#define CARRIER_FREQ 38000

unsigned **int** irRawSignal[] = {

9000, 4500, // starting bit

```
560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, 560, 560, // address 00100000 ? 4
560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, 560, 1690, // ~ address 11011111
560, 560, 560, 560, 560, 560, 560, 1690, 560, 560, 560, 560, 560, 560, 560, 560, // data 00010000 ? 8
560, 1690, 560, 1690, 560, 1690, 560, 560, 560, 1690, 560, 1690, 560, 1690, 560, 1690, //~ data 11101111
560 // stoping bit
};
int DataLen = sizeof(irRawSignal) / sizeof(irRawSignal[0]); // 284/ 4 = 71
void setup()
{
Serial.begin(115200);
IR.begin(IR_RX_PIN, IR_TX_PIN, IR_MODE_TX, CARRIER_FREQ);
}
void loop()
{
IR.send(irRawSignal, DataLen);
Serial.println("Finished Sending NEC Raw Data….");
delay(3000);
}
```

Notes and Warnings
IR Raw Data array contains information in the form of consecutive microseconds (us). For more details, please refer to: http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html.

IRDevice::beginNEC

Description
Allocate resources and start the IR device with a frequency suitable for the NEC protocol.

Syntax
void beginNEC(uint8_t receivePin, uint8_t transmitPin, uint32_t irMode);

Parameters
receivePin: pin on which IR sensor is connected. Hardware IR receiver is available at pins 3, 8, 17.
transmitPin: pin on which IR LED is connected. Hardware IR transmitter is available at pins 6, 9, 16.
irMode: transmit or receive mode. Valid values: IR_MODE_TX, IR_MODE_RX

Returns

The function returns nothing.

Example Code
Example: IRRecvNEC

#include "IRDevice.h"

uint8_t adr = 0;

uint8_t cmd = 0;

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(115200);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

IR.beginNEC(8, 9, IR_MODE_RX); // configure for NEC IR protocol

}

void loop() {

if (IR.recvNEC(adr, cmd, 1000)) {

Serial.print("Received ");

Serial.print(adr);

Serial.print(cmd);

Serial.println();

} **else** {

Serial.println("Received nothing, timed out");

}

//IR.end();

}

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::sendNEC

Description
Send data using the NEC protocol.

Syntax
void sendNEC(uint8_t adr, uint8_t cmd);

Parameters
adr: 8-bit address to transmit
cmd: 8-bit command to transmit

Returns
The function returns nothing.

Example Code
Example: IRSendNEC

#include "IRDevice.h"

uint8_t adr = 0;

uint8_t cmd = 0;

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(115200);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

IR.beginNEC(8, 9, IR_MODE_TX); // configure for NEC IR protocol

}

void loop() {

if (cmd++ >=255) {

adr++;

}

IR.sendNEC(adr, cmd);

Serial.print("Sent ");

Serial.print(adr);

Serial.print(cmd);

Serial.println();

//IR.end(); // Call this method to stop IR device and free up the pins for other uses

}

Notes and Warnings

IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

IRDevice::recvNEC

Description
Receive data using the NEC protocol.

Syntax
void recvNEC(uint8_t& adr, uint8_t& cmd uint32_t timeout);

Parameters
adr: variable to store received NEC address
cmd: variable to store received NEC command
timeout: time duration to wait for an incoming transmission

Returns
The function returns "1" if data has been received, returns "0" if no data has been received.

Example Code
Example: IRRecvNEC
Details of the code can be found in the previous section of IRDevice::beginNEC.

Notes and Warnings
IR device can only operate in either transmit or receive mode. Refer
to https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol for the NEC protocol.

MDNS

Class HttpClient

MDNSClass Class

Description
A class used for registering and removing MDNS service records.

Syntax
class MDNSClass

Members

Public Constructors	
The public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named MDNS.	

Public Methods	
MDNSClass::begin	Start MDNS operations
MDNSClass::end	Stop MDNS operations
MDNSClass::registerService	Add a service record
MDNSClass::deregisterService	Remove service record
MDNSClass::updateService	Update service record

MDNSClass::begin

Description

Start MDNS operations to begin responding to MDNS queries.

Syntax

void begin(void);

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: mDNS_On_Arduino_IDE

This example shows how to register Ameba as a service that can be recognized by Arduino IDE. If both of the PC runs Arduino IDE and the Ameba board are connecting to the same local network. Then you can find Ameba in “Tools” -> “Port” -> “Arduino at 192.168.1.238 (Ameba RTL8195A), which means the Arduino IDE find Ameba via mDNS.

```
#include <WiFi.h>

#include <AmebaMDNS.h>

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

MDNSService service(“MyAmeba”, “_arduino._tcp”, “local”, 5000);

void setup() {

printf(“Try to connect to %srn”, ssid);

while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
```

```
printf("Failed. Wait 1s and retry…rn");
delay(1000);
}
printf("Connected to %srn", ssid);
service.addTxtRecord("board", strlen("ameba_rtl8195a"), "ameba_rtl8195a");
service.addTxtRecord("auth_upload", strlen("no"), "no");
service.addTxtRecord("tcp_check", strlen("no"), "no");
service.addTxtRecord("ssh_upload", strlen("no"), "no");
printf("Start mDNS servicern");
MDNS.begin();
printf("register mDNS servicern");
MDNS.registerService(service);
}
void loop() {
// put your main code here, to run repeatedly:
delay(1000);
}
```

Notes and Warnings
Include "AmebaMDNS.h" to use the class function.

MDNSClass::end

Description
Stop MDNS operations and stop responding to MDNS queries.

Syntax
void end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::registerService

Description
Add a service record to be included in MDNS responses.

Syntax
void register service(MDNSService service);

Parameters
service: MDNSService class object with required MDNS service data

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::deregisterService

Description
Remove a service record from MDNS responses.

Syntax
void deregisterService(MDNSService service);

Parameters
service: MDNSService class object to be removed

Returns
The function returns nothing.

Example Code

Example: mDNS_On_Arduino_IDE
Details of the code can be found in the previous section of MDNSClass:: begin.

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSClass::updateService

Description
Update a service record.

Syntax
void updateService(MDNSService service, unsigned int ttl);

Parameters
service: MDNSService class object to be updated
ttl: time-to-live(TTL) for service

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

Class HttpClient

MDNSService Class

Description
A class used for creating MDNS service records.

Syntax
class MDNSService

Members

Public Constructors	
MDNSService::MDNSService	Create a MDNS service record
Public Methods	
MDNSService::addTxtRecord	Add text to MDNS service record

MDNSService::MDNSService

Description
Create a MDNS service record.

Syntax
MDNSService(char* name, char* service_type, char* domain, unsigned short port, int bufsize);

Parameters
name: device name
service_type: MDNS service type
domain: host domain
port: network port
bufsize: size of buffer for MDNS text record

Returns
The function returns nothing.

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MDNSService::addTxtRecord

Description
Add text to MDNS service record.

Syntax
int addTextRecord(char* key, int value_len, char* value);

Parameters
key: record type expressed as character string
value_len: length of value string

value: record value expressed as character string

Returns
0 if add record successful

Example Code
Example: mDNS_On_Arduino_IDE

Notes and Warnings
Include “AmebaMDNS.h” to use the class function.

MQTTClient

Class PMUClass

PubSubClient Class

Description
Defines a class of MQTT implementation for Ameba.

Syntax
class PubSubClient

Members

Public Constructors	
PubSubClient::PubSubClient	Constructs a PubSubClient object
Public Methods	
PubSubClient::setServer	Set MQTT server address and port
PubSubClient::setCallback	Set callback function
PubSubClient::setClient	Set WiFi client
PubSubClient::setStream	Set data stream
PubSubClient::connect	Attempt to connect to server
PubSubClient::disconnect	Disconnect from current session
PubSubClient::publish	Publish a message to server
PubSubClient::publish_P	Same as above
PubSubClient::subscribe	Subscribe to a topic
PubSubClient::unsubscribe	Unsubscribe to a topic
PubSubClient::loop	Keep MQTT session alive and process any queuing tasks
PubSubClient::connected	Check if client still connected
PubSubClient::state	Return connection state

PubSubClient::PubSubClient

Description
Constructs a PubSubClient object and, if applicable, sets server address, port, callback function, data stream and wifi client.

Syntax
PubSubClient::PubSubClient();
PubSubClient::PubSubClient(Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, Client& client, Stream&);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
PubSubClient::PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);

Parameters
client: the network client to use, for example WiFiClient
IPAddress: MQTT server address
port: port for MQTT, usually 1883 for unencrypted connection
MQTT_CALLBACK_SIGNATURE: callback function for MQTT
Stream: a stream to write received messages to

Returns
The function returns nothing.

Example Code
Example: MQTT_Basic

#include <WiFi.h>

#include <PubSubClient.h>

// Update these with values suitable for your network.

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int status = WL_IDLE_STATUS; // the Wifi radio’s status

char mqttServer[] = “test.mosquitto.org”;

```
char clientId[] = "amebaClient";
char publishTopic[] = "outTopic";
char publishPayload[] = "hello world";
char subscribeTopic[] = "inTopic";
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i=0;i<length;i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}
WiFiClient wifiClient;
PubSubClient client(wifiClient);
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection…");
// Attempt to connect
if (client.connect(clientId)) {
Serial.println("connected");
// Once connected, publish an announcement…
client.publish(publishTopic, publishPayload);
// … and resubscribe
client.subscribe(subscribeTopic);
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}
void setup()
```

```
{
Serial.begin(38400);
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
client.setServer(mqttServer, 1883);
client.setCallback(callback);
// Allow the hardware to sort itself out
delay(1500);
}
void loop()
{
if (!client.connected()) {
reconnect();
}
client.loop();
}
```

Notes and Warnings
PubSubClient::PubSubClient(Client& client) would suffice for normal MQTT connection

PubSubClient::setServer

Description
Sets the server details.

Syntax
PubSubClient& PubSubClient::setServer(uint8_t * ip, uint16_t port)
PubSubClient& PubSubClient::setServer(IPAddress ip, uint16_t port)
PubSubClient& PubSubClient::setServer(const char * domain, uint16_t port)

Parameters

ip: the address of the server
port: the port to connect to, default 1883
domain: the address of the server

Returns
The client instance, allowing the function to be chained

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setCallback

Description
Sets the message callback function.

Syntax
PubSubClient& PubSubClient::setCallback(MQTT_CALLBACK_SIGNATURE)

Parameters
MQTT_CALLBACK_SIGNATURE: a pointer to a message callback function called when a message arrives for a subscription created by this client.

Returns
The client instance, allowing the function to be chained.

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::setClient

Description
Sets the network client instance to use.

Syntax
PubSubClient& PubSubClient::setClient(Client& client)

Parameters
client: the network client to use, for example WiFiClient

Returns
The client instance, allowing the function to be chained

Example Code
NA

Notes and Warnings
NA

PubSubClient::setStream

Description
Sets the stream to write received messages to.

Syntax
PubSubClient& PubSubClient::setStream(Stream& stream)

Parameters
stream: a stream to write received messages to

Returns
The client instance, allowing the function to be chained.

Example Code
NA

Notes and Warnings
NA

PubSubClient::connect

Description
Connects the client to the server.

Syntax

boolean PubSubClient::connect(const char *id)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass)

boolean PubSubClient::connect(const char *id, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

boolean PubSubClient::connect(const char *id, const char *user, const char *pass, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage)

Parameters

id: Client ID, a unique string identifier

user: Username for authentication, default NULL

pass: Password for authentication, default NULL

willTopic: the topic to be used by the will message

willQoS: the quality of service to be used by the will message

willRetain: whether the will should be published with the retain flag

willMessage: the payload of the will message

Returns

True – connection succeeded

False – connection failed

Example Code

Example: MQTT_Basic

Notes and Warnings

Client ID is required and should always be unique else connection might be rejected by the server.

PubSubClient::disconnect

Description

Disconnect the client

Syntax

void PubSubClient::disconnect(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PubSubClient::publish

Description
Publishes a message to the specified topic.

Syntax
boolean PubSubClient::publish(const char* topic, const char* payload)
boolean PubSubClient::publish(const char* topic, const char* payload, boolean retained)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength)
boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
Default max packet size is 128 bytes.

PubSubClient::publish_P

Description
Publishes a message stored in PROGMEM to the specified topic.

Syntax
boolean PubSubClient::publish_P(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained)

Parameters
topic: the topic to publish to
payload: the message to publish
plength: the length of the payload. Required if payload is a byte[]
retained: whether the message should be retained
– false – not retained
– true – retained

Returns
False – publish failed, either connection lost or message too large
True – publish succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::subscribe

Description
Subscribes to messages published to the specified topic.

Syntax
boolean PubSubClient::subscribe(const char* topic)
boolean PubSubClient::subscribe(const char* topic, uint8_t qos)

Parameters
topic: the topic to subscribe to
qos: the qos to subscribe at

Returns
False – sending the subscribe failed, either connection lost or message too large
True – sending the subscribe succeeded

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::unsubscribe

Description
Unsubscribes from the specified topic.

Syntax
boolean PubSubClient::unsubscribe(const char* topic)

Parameters
topic: the topic to unsubscribe to

Returns
False – sending the unsubscribe failed, either connection lost or message too large
True – sending the unsubscribe succeeded

Example Code
NA

Notes and Warnings
NA

PubSubClient::loop

Description
A must method called regularly to allow the client to process incoming messages and maintain its connection to the server.

Syntax
boolean PubSubClient::loop(void)

Parameters
The function requires no input parameter.

Returns
False – the client is no longer connected
True – the client is still connected

Example Code
Example: MQTT_Basic

Notes and Warnings
A required method that should not be blocked for too long.

PubSubClient::connected

Description
Checks whether the client is connected to the server.

Syntax
boolean PubSubClient::connected(void)

Parameters
The function requires no input parameter.

Returns
False – the client is not connected
True – the client is connected

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

PubSubClient::state

Description
Returns the current state of the client. If a connection attempt fails, this can be used to get more information about the failure.
All of the values have corresponding constants defined in PubSubClient.h.

Syntax
int PubSubClient::state(void)

Parameters
The function requires no input parameter.

Returns
-4 : MQTT_CONNECTION_TIMEOUT – the server didn't respond within the keepalive time
-3 : MQTT_CONNECTION_LOST – the network connection was broken
-2 : MQTT_CONNECT_FAILED – the network connection failed
-1 : MQTT_DISCONNECTED – the client is disconnected cleanly
0 : MQTT_CONNECTED – the client is connected
1 : MQTT_CONNECT_BAD_PROTOCOL – the server doesn't support the requested version of MQTT
2 : MQTT_CONNECT_BAD_CLIENT_ID – the server rejected the client identifier
3 : MQTT_CONNECT_UNAVAILABLE – the server was unable to accept the connection
4 : MQTT_CONNECT_BAD_CREDENTIALS – the username/password were rejected
5 : MQTT_CONNECT_UNAUTHORIZED – the client was not authorized to connect

Example Code
Example: MQTT_Basic

Notes and Warnings
NA

Readme

The Ameba MQTT related APIs and examples are works based on the PubSubClient libraries written by Nicholas O'Leary
These include,

- `PubSubClient.cpp`
- `PubSubClient.h`

These libraries are under MIT License.

NTPClient

Readme

The NTPClient library is based on the NTPClient library written by Fabrice Weinberg, which can be found at https://github.com/arduino-libraries/NTPClient.
These include,

- `NTPClient.cpp`
- `NTPClient.h`

These libraries are licensed under MIT License.

PowerSave

Class PMUClass

PMUClass Class

Description
Defines a class of using Power Save API

Syntax
class PMUClass

Members

Public Constructors	
PMUClass::PMUClass	Constructs a PMUClass object
Public Methods	
PMUCLASS::begin	Initialize the PMUCLASS and select sleep mode
PMUCLASS::AONTimerDuration	Set the duration of AON Timer
PMUCLASS::AONTimerCmd	Disable the AON Timer for power save usage
PMUCLASS::RTCWakeSetup	Set up RTC Timer for power save usage
PMUCLASS::enable	Enable power save deep sleep mode
PMUCLASS::AONWakeReason	Check AON wakeup source
PMUCLASS::WakePinCheck	Check AON GPIO pin wakeup source
PMUCLASS::AONWakeClear	Clear all the AON wakeup source
PMUCLASS::DsleepWakeStatusGet	Check if deepsleep mode is set
PMUCLASS::TL_sysactive_time	Tickless mode system active time
PMUCLASS::TL_wakelock	Tickless mode wake lock, select acquire of release
PMUCLASS::DS_AON_TIMER_WAKEUP	Return the Wakeup source
PMUCLASS::DS_RTC_WAKEUP	Return the Wakeup source
PMUCLASS::TL_UART_WAKEUP	Return the Wakeup source
PMUCLASS::TL_RTC_WAKEUP	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA12	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA13	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA14	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA15	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA16	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA17	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA18	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA19	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA20	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA21	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA25	Return the Wakeup source
PMUC LASS::AON_WAKEPIN_WAKEUP_GPIOA26	Return the Wakeup source

PMUCLASS::PMUCLASS

Description
Constructs a PMUCLASS object.

Syntax
PMUCLASS::PMUCLASS(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::begin

Description
Initialize the PMUCLASS and select sleep mode.

Syntax
void PMUClass::begin(uint32_t sleep_mode);

Parameters
sleep_mode: Selection value, “11” enters the DeepSleep Mode, “22” enters the Tickless Mode

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerDuration

Description
Set the duration of AON Timer

Syntax
void PMUClass::AONTimerDuration(uint32_t duration_ms);

Parameters
duration_ms: Timer duration between 0 to 32760000ms.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONTimerCmd

Description
Disable the AON timer for power save usage.

Syntax
void PMUClass::AONTimerCmd(void);

Parameters
c: The value in Celcius.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::RTCWakeSetup

Description
Set up the RTC timer for power save usage.

Syntax
void PMUClass::RTCWakeSetu(uint32_t duration_d, unit32_t duration_h, uint32_t duration_m, uint32_t duration_s);

Parameters
duration_d: Set alarm for number of days from 0.
duration_h: Set alarm for number of hours from 0.
duration_m: Set alarm for number of minutes from 0.
duration_s: Set alarm for number of seconds from0.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::enable

Description
Enable power save deep sleep mode

Syntax
void PMUClass::enable(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeReason

Description
Check the AON wakeup source

Syntax
uint32_t PMUClass::AONWakeReason(void);

Parameters
The function requires no input parameter.

Returns
Returns the value of wakeup deepsleep source. “11” for AON pin, “22” for AON timer, “33” for RTC timer and “0” for none.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::WakePinCheck

Description
Check which AON GPIO pins are the wakeup source

Syntax
int PMUClass::WakePinCheck(void);

Parameters
The function requires no input parameter.

Returns
Return the pin number for indicating Arduino pin names.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AONWakeClear

Description
Clear all AON Wakeup source.

Syntax
void PMUClass::AONWakeClear(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DsleepWakeStatusGet

Description
Check if deepsleep mode is set.

Syntax
bool PMUClass::DsleepWakeStatusGet(void);

Parameters
The function requires no input parameter.

Returns
Return TRUE when enter DeepSleep Mode or FALSE for negative.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_sysactive_time

Description
Tickless mode system active time.

Syntax
void PMUClass::TL_sysactive_time(uint32_t duration_ms);

Parameters
duration_ms: Set the duration of system active time. The unit is in milliseconds.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_wakelock

Description
Tickless mode wake lock, select acquire or release.

Syntax
void PMUClass::TL_wakelock(uint32_t select_lock);

Parameters
select_lock: Wake lock selection value, “1” for acquire or “0” for release.

Returns
The function returns nothing.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_AON_TIMER_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_AON_TIMER_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON Timer as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::DS_RTC_WAKEUP

Description
Return the Wakeup source for DeepSleep Mode.

Syntax
void PMUClass::DS_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_UART_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_UART_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns LOGUART as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::TL_RTC_WAKEUP

Description
Return the Wakeup source for Tickless Mode.

Syntax
void PMUClass::TL_RTC_WAKEUP(void);

Parameters
The function requires no input parameter.

Returns
This function returns RTC as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA12

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA12(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA12 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA13

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA13(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA13 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA14

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA14(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA14 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA15

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA15(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA15 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA16

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA16(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA16 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA17

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA17(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA17 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA18

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA18(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA18 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA19

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA19(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA19 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA20

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA20(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA20 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA21

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA21(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA21 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA25

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA25(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA25 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

PMUCLASS::AON_WAKEPIN_WAKEUP_GPIOA26

Description
Return the Wakeup source.

Syntax
void PMUClass::AON_WAKEPIN_WAKEUP_GPIOA26(void);

Parameters
The function requires no input parameter.

Returns
This function returns AON GPIOA26 pin as the wakeup source and output it on the Serial monitor.

Example Code
Example: DeepSleep_DHT_Eink_Example; DeepSleep_DHT_LCD_Example; DeepSleepMode; TicklessMode;

Notes and Warnings
Include “PMUCLASS.h” in order to use the class function.

RTC

Class RTC

RTC Class

Description
A class used for displaying date and time and alarm configuration using RTC, the independent BCD (Binary-Coded-Decimal) timer.

Syntax
class RTC

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named RTC.	

Public Methods	
RTC:: Init	Initializes the RTC device, including the Clock, the RTC registers, and other functions
RTC:: DeInit	Deinitialize the RTC device
RTC:: Write	Set the specified timestamp in seconds to RTC
RTC:: Read	Get the current timestamp in seconds from RTC
RTC:: Wait	Wait for 1 second
RTC:: SetEpoch	Convert human-readable time to epoch time

RTC::Init

Description
Initializes the RTC device, including the Clock, the RTC registers, and other functions.

Syntax
void RTC::Init(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC

```
/*
* This function describes how to use the RTC API.
* The RTC function is implemented by an independent BCD timer/counter.
* This example will print out the time information every second.
*/
#include <stdio.h>
#include <time.h>
#include "rtc.h"
#define YEAR 2020
#define MONTH 9
#define DAY 10
#define HOUR 20
#define MIN 30
#define SEC 40
/* Create an rtc object */
RTC rtc;
int32_t seconds;
struct tm *timeinfo;
void setup() {
Serial.begin(115200);
rtc.Init(); // initialize RTC
}
void loop() {
// step 1: convert user time to epoch
int epochTime = humanReadableToEpoch(YEAR, MONTH, DAY, HOUR, MIN, SEC);
// step 2: write epoch time to rtc
rtc.Write(epochTime);
while (1) {
seconds = rtc.Read();
printf("Epoch Time (in s) since January 1, 1970 = %dsn", seconds);
printf("Time as a basic string = %s", ctime(&seconds));
```

```
timeinfo = localtime(&seconds);
printf("Time as a custom formatted string = %d-%d-%d %d:%d:%dn",
(timeinfo->tm_year + 1900), (timeinfo->tm_mon + 1), timeinfo->tm_mday, timeinfo->tm_hour,
timeinfo->tm_min, timeinfo->tm_sec);
Serial.println();
rtc.wait(1);
}
}
// convert human readable time to epoch time
int humanReadableToEpoch(int year, int month, int day, int hour, int min, int sec) {
struct tm t;
time_t t_of_day;
t.tm_year = year - 1900; // Year - 1970
t.tm_mon = month - 1; // Month, where 0 = jan
t.tm_mday = day; // Day of the month
t.tm_hour = hour;
t.tm_min = min;
t.tm_sec = sec;
t.tm_isdst = -1; // Is DST on? 1 = yes, 0 = no, -1 = unknown
t_of_day = mktime(&t);
// printf("seconds since the Epoch: %dn", (long)t_of_day);
return t_of_day;
}
```

Notes and Warnings
NA

RTC::DeInit

Description
Deinitializes the RTC device.

Syntax
void RTC::DeInit(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Write

Description
Set the specified timestamp in seconds to RTC. Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Syntax
void RTC::Write(int t);

Parameters
Parameters
t: Seconds from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS) to specified date and time which is to be set.

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC::Read

Description
Get the current timestamp in seconds from RTC. The current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Syntax
int32_t RTC::Read(void);

Parameters
The function requires no input parameter.

Returns
The function returns the current timestamp in seconds which is calculated from 1970.1.1 00:00:00 (YEAR.MONTH.DAY, HOUR: MIN: SECONDS).

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: Wait

Description
Send IR raw data.

Syntax
void RTC::wait(float s);

Parameters
s: unit microseconds (1 us)

Returns
The function returns nothing.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

RTC:: SetEpoch

Description
Convert human-readable time to epoch time

Syntax
int RTC:: SetEpoch(int year, int month, int day, int hour, int min, int sec);

Parameters
year: user input year
month: user input month
day: user input day
hour: user input hour
min: user input minutes
sec: user input seconds

Returns
The function returns epoch time in seconds for RTC use.

Example Code
Example: RTC
Details of the code can be found in the previous section of RTC:: Init.

Notes and Warnings
NA

SoftwareSerial

Class Adafruit_GPS

Adafruit_GPS Class

Description
Defines a class to use GPS module on Ameba.

Syntax
class Adafruit_GPS

Members

Public Constructors	
Adafruit_GPS::Adafruit_GPS	Constructs an Adafruit_GPS object
Public Methods	
Adafruit_GPS::begin	Initialize serial communication
*Adafruit_GPS:: lastNMEA	Returns the last NMEA line received and unsets the received flag
Adafruit_GPS:: newNMEAreceived	Check to see if a new NMEA line has been received
Adafruit_GPS:: common_init	Initialization code used by all constructor types
Adafruit_GPS:: sendCommand	Send a command to the GPS device
Adafruit_GPS:: pause	Pause/unpause receiving new data
Adafruit_GPS:: parseHex	Read a Hex value and return the decimal equivalent
Adafruit_GPS:: read	Read one character from the GPS device
Adafruit_GPS:: parse	Parse an NMEA string
Adafruit_GPS:: wakeup	Wake the sensor up
Adafruit_GPS:: standby	Standby Mode Switches
Adafruit_GPS::waitForSentence	Wait for a specified sentence from the device
Adafruit_GPS::LOCUS_StartLogger	Start the LOCUS logger
Adafruit_GPS::LOCUS_StopLogger	Stop the LOCUS logger
Adafruit_GPS::LOCUS_ReadStatus	Read the logger status

Adafruit_GPS::Adafruit_GPS

Description

Constructs an Adafruit_GPS object and initialize serial using a SoftSerial object.

Syntax

Adafruit_GPS::Adafruit_GPS(SoftwareSerial *ser)

Adafruit_GPS::Adafruit_GPS(HardwareSerial *ser)

Parameters

ser: a Serial instance

Returns

The function returns nothing.

Example Code

Example: Adafruit_GPS_parsing

This example code from Adafruit demonstrates GPS modules using MTK3329/MTK3339 driver. This code shows how to listen to the GPS module in an interrupt which allows the program to have more 'freedom' – just parse when a new NMEA sentence is available! Then access data when desired.

#include <Adafruit_GPS.h>

#include <SoftwareSerial.h>

// If you're using a GPS module:

// Connect the GPS Power pin to 3.3V

```
// Connect the GPS Ground pin to ground
// Connect the GPS TX (transmit) pin to Digital 0
// Connect the GPS RX (receive) pin to Digital 1
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1);
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RTL8710 need change GPS TX/RX to pin 17 and 5
#else
SoftwareSerial mySerial(0, 1);
#endif
Adafruit_GPS GPS(&mySerial);
// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences.
#define GPSECHO false
void setup()
{
Serial.begin(38400);
Serial.println("Adafruit GPS library basic test!");
// 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
GPS.begin(9600);
// uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitude
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
// uncomment this line to turn on only the "minimum recommended" data
//GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
// For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
// the parser doesn't care about other sentences at this time
// Set the update rate
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
// For the parsing code to work nicely and have time to sort thru the data, and
// print it out we don't suggest using anything higher than 1 Hz
// Request updates on antenna status, comment out to keep quiet
GPS.sendCommand(PGCMD_ANTENNA);
delay(1000);
// Ask for firmware version
mySerial.println(PMTK_Q_RELEASE);
}
```

```
uint32_t timer = millis();
void loop() // run over and over again
{
// in case you are not using the interrupt above, you'll
// need to 'hand query' the GPS, not suggested :(
// read data from the GPS in the 'main loop'
char c = GPS.read();
// if you want to debug, this is a good time to do it!
if (GPSECHO)
if (c) Serial.print(c);
// if a sentence is received, we can check the checksum, parse it…
if (GPS.newNMEAreceived()) {
// a tricky thing here is if we print the NMEA sentence, or data
// we end up not listening and catching other sentences!
// so be very wary if using OUTPUT_ALLDATA and trytng to print out data
//Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false
if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to false
return; // we can fail to parse a sentence in which case we should just wait for another
}
// if millis() or timer wraps around, we'll just reset it
if (timer > millis()) timer = millis();
// approximately every 2 seconds or so, print out the current stats
if (millis() - timer > 2000) {
timer = millis(); // reset the timer
Serial.print("nTime: ");
Serial.print(GPS.hour, DEC); Serial.print(':');
Serial.print(GPS.minute, DEC); Serial.print(':');
Serial.print(GPS.seconds, DEC); Serial.print('.');
Serial.println(GPS.milliseconds);
Serial.print("Date: ");
Serial.print(GPS.day, DEC); Serial.print('/');
Serial.print(GPS.month, DEC); Serial.print("/20");
Serial.println(GPS.year, DEC);
Serial.print("Fix: "); Serial.print((int)GPS.fix);
Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
if (GPS.fix) {
```

Serial.print(“Location: “);

Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);

Serial.print(”, “);

Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);

Serial.print(“Location (in degrees, works with Google Maps): “);

Serial.print(GPS.latitudeDegrees, 4);

Serial.print(”, “);

Serial.println(GPS.longitudeDegrees, 4);

Serial.print(“Speed (knots): “); Serial.println(GPS.speed);

Serial.print(“Angle: “); Serial.println(GPS.angle);

Serial.print(“Altitude: “); Serial.println(GPS.altitude);

Serial.print(“Satellites: “); Serial.println((**int**)GPS.satellites);

}

}

}

Notes and Warnings
IMPORTANT: SoftSerial is using hardware serial so pin mapping cannot be altered.

Adafruit_GPS::begin

Description
Initialize serial communication

Syntax
void Adafruit_GPS::begin(uint16_t baud)

Parameters
baud: serial baud rate

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

***Adafruit_GPS::lastNMEA**

Description
Returns the last NMEA line received and unsets the received flag

Syntax
char *Adafruit_GPS::lastNMEA(void)

Parameters
The function requires no input parameter.

Returns
Pointer to the last line string

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::newNMEAreceived

Description
Check to see if a new NMEA line has been received

Syntax
boolean Adafruit_GPS::newNMEAreceived(void)

Parameters
The function requires no input parameter.

Returns
True if received, false if not

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::common_init

Description
Initialization code used by all constructor types

Syntax
void Adafruit_GPS::common_init(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::sendCommand

Description
Send a command to the GPS device

Syntax
void Adafruit_GPS::sendCommand(const char *str)

Parameters
str: Pointer to a string holding the command to send

Returns
The function returns nothing.

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::pause

Description
Pause/unpause receiving new data

Syntax
void Adafruit_GPS::pause(boolean p)

Parameters
p: True = pause, false = unpause

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::parseHex

Description
Read a Hex value and return the decimal equivalent

Syntax
uint8_t Adafruit_GPS::parseHex(char c)

Parameters
c: Hex value

Returns
The decimal equivalent of the Hex value

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::read

Description
Read one character from the GPS device

Syntax
char Adafruit_GPS::read(void)

Parameters
The function requires no input parameter.

Returns
The character that we received, or 0 if nothing was available

Example Code
Example: Adafruit_GPS_parsing
The details of the code can be found in the previous section of Adafruit_GPS:: Adafruit_GPS.

Notes and Warnings
NA

Adafruit_GPS::parse

Description
Parse an NMEA string

Syntax
boolean Adafruit_GPS::parse(char *nmea)

Parameters
nmea: an NMEA string

Returns
True if we parsed it, false if it has invalid data

Example Code
Example: Adafruit_GPS_parsing

Notes and Warnings
NA

Adafruit_GPS::wakeup

Description
Wake the sensor up

Syntax
boolean Adafruit_GPS::wakeup(void)

Parameters
The function requires no input parameter.

Returns
True if woken up, false if not in standby or failed to wake

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::standby

Description
Standby Mode Switches

Syntax
boolean Adafruit_GPS::standby(void)

Parameters
The function requires no input parameter.

Returns
False if already in standby, true if it entered standby

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::waitForSentence

Description
Wait for a specified sentence from the device

Syntax
boolean Adafruit_GPS::waitForSentence(const char *wait4me, uint8_t max)

Parameters
wait4me: Pointer to a string holding the desired response
max: How long to wait, default is MAXWAITSENTENCE

Returns
True if we got what we wanted, false otherwise

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StartLogger

Description
Start the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StartLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings
NA

Adafruit_GPS::LOCUS_StopLogger

Description
Stop the LOCUS logger

Syntax
boolean Adafruit_GPS::LOCUS_StopLogger(void)

Parameters
The function requires no input parameter.

Returns
True on success, false if it failed

Example Code
NA

Notes and Warnings

NA

Adafruit_GPS::LOCUS_ReadStatus

Description
Read the logger status

Syntax
boolean Adafruit_GPS::LOCUS_ReadStatus(void)

Parameters
The function requires no input parameter.

Returns
True if we read the data, false if there was no response

Example Code
NA

Notes and Warnings
NA

Class HttpClient

PMS3003 Class

Description
Defines a class to work with PMS3003 air quality sensor on Ameba.

Syntax
class PMS3003

Members

Public Constructors	
PMS3003::PMS3003	Constructs a PMS3003 object
Public Methods	
PMS3003::begin	Initialize hardware UART
PMS3003::end	Free allocated space thus stopping UART
PMS3003::get_pm1p0_cf1	Get PM1.0 under correction factor = 1
PMS3003:: get_pm2p5_cf1	Get PM2.5 under correction factor = 1
PMS3003:: get_pm10_cf1	Get PM10 under correction factor = 1
PMS3003:: get_pm1p0_air	Get PM1.0 air quality
PMS3003:: get_pm2p5_air	Get PM2.5 air quality
PMS3003:: get_pm10_air	Get PM10 air quality
PMS3003:update_cache	Updates the cache memory
PMS3003::pms3003_handle_interrupt	Set up the serial event handler

PMS3003::PMS3003

Description
Constructs a PMS3003 object and initialize the pin mapping.

Syntax
PMS3003::PMS3003(int _rx, int _tx, int _set, int _reset)

Parameters
_rx: RX pin of UART
_tx: TX pin of UART
_set: Set pin
_reset: Reset pin

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::begin

Description
Initialize hardware UART and allocate space for serial buffer

Syntax
void PMS3003::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::end

Description
Free serial buffer space and stop UART

Syntax
void PMS3003::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_cf1

Description
Get PM1.0 under correction factor = 1

Syntax
int PMS3003::get_pm1p0_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value “pm1p0_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm2p5_cf1

Description
Get PM2.5 under correction factor = 1

Syntax
int PMS3003::get_pm2p5_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_cf1

Description
Get PM10 under correction factor = 1

Syntax
int PMS3003::get_pm10_cf1(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_cf1” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm1p0_air

Description
Get PM1.0 air quality

Syntax
int PMS3003::get_pm1p0_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm1p0_air” as an integer.

Example Code
NA

Notes and Warnings

NA

PMS3003::get_pm2p5_air

Description
Get PM2.5 air quality

Syntax
int PMS3003::get_pm2p5_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm2p5_air” as an integer.

Example Code
NA

Notes and Warnings
NA

PMS3003::get_pm10_air

Description
Get PM10 air quality

Syntax
int PMS3003::get_pm10_air(void)

Parameters
The function requires no input parameter.

Returns
The function returns the value of “pm10_air” as an integer.

Example Code

NA

Notes and Warnings
NA

PMS3003::pms3003_handle_interrupt

Description
Set up the serial event handler

Syntax
void pms3003_handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: device identifier
event: Serial event for handling incoming data

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

PMS3003::update_cache

Description
Serves the function of updating cache memory. One package has 32 bytes. Illustrate the formate by using below raw data: 42 4d 00 1c 00 1b 00 21 00 29 00 1a 00 21 00 29 2b fb 04 be 00 6b 00 10 00 04 00 04 67 00 04 46
42 4d : header signature
00 1c : frame length, 0x001c = 28 bytes (not include header and this field)
00 1b : PM1.0 under CF=1
00 21 : PM2.5 under CF=1
00 29 : PM10 under CF=1
00 1a : PM1.0 under air
00 21 : PM2.5 under air
00 29 : PM10 under air

2b fb : number of pariticle, diameter size 0.3 um in 0.1 liter air
04 be : number of pariticle, diameter size 0.5 um in 0.1 liter air
00 6b : number of pariticle, diameter size 1.0 um in 0.1 liter air
00 10 : number of pariticle, diameter size 2.5 um in 0.1 liter air
00 04 : number of pariticle, diameter size 5.0 um in 0.1 liter air
00 04 : number of pariticle, diameter size 10 um in 0.1 liter air
67 : serial number
00 : error code
04 46 :
checksum,0x42+0x4d+0x00+0x1c+0x00+0x1b+0x00+0x21+0x00+0x29+0x00+0x1a+0x00+0x21+0x00+0x29+
0x2b+0xfb+0x04+0xbe+0x00+0x6b+0x00+0x10+0x00+0x04+0x00+0x04+0x67+0x00 = 0x0446

Syntax
void PMS3003::update_cache(void)

Parameters
The function requires no input parameters.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Class SoftwareSerial

SoftwareSerial Class

Description
Defines a class of software serial implementation for Ameba.

Syntax
class SoftwareSerial

Members

Public Constructors	
SoftwareSerial::SoftwareSerial	Constructs a SoftwareSerial object
Public Methods	
SoftwareSerial::begin	Sets the speed (baud rate) for the serial communication
SoftwareSerial::listen	Enables the selected software serial port to listen
SoftwareSerial::end	Same as stopListening
SoftwareSerial::stopListening	Stop listening on the port
SoftwareSerial::peek	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::write	Prints data to the transmit pin of the software serial port as raw bytes
SoftwareSerial::read	Return a character that was received on the RX pin of the software serial port
SoftwareSerial::available	Get the number of bytes (characters) available for reading from a software serial port
SoftwareSerial::flush	Flush the received buffer
SoftwareSerial::setBufferSize	Set buffer size
SoftwareSerial::setAvailableCallback	Set available callback
SoftwareSerial::handle_interrupt	Private methods handles interrupt

SoftwareSerial::SoftwareSerial

Description
Constructs a SoftwareSerial object and sets RX and TX pin, and inverse logic.

Syntax
SoftwareSerial::SoftwareSerial(uint8_t receivePin, uint8_t transmitPin, bool inverse_logic /* = false */)

Parameters
receivePin: the pin on which to receive serial data
transmitPin: the pin on which to transmit serial data
inverse_logic: is used to invert the sense of incoming bits

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX.

/*

The circuit: (BOARD RTL8195A)

* RX is digital pin 0 (connect to TX of other devices)

* TX is digital pin 1 (connect to RX of other devices)

*/

```
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(57600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
Serial.println("Goodnight moon!");
// set the data rate for the SoftwareSerial port
mySerial.begin(4800);
mySerial.println("Hello, world?");
}
void loop() { // run over and over
if (mySerial.available()) {
mySerial.write(mySerial.read());
}
}
```

Notes and Warnings
Software Serial is using hardware serial thus DO NOT change the default pins

SoftwareSerial::begin

Description
Sets the speed (baud rate) for the serial communication

Syntax
void SoftwareSerial::begin(long speed)
void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits)

void SoftwareSerial::begin(long speed, int data_bits, int parity, int stop_bits, int flowctrl, int rtsPin, int ctsPin)

Parameters
speed: the baud rate
data_bits: number of data bits, 8 bits(default) or 7 bits
stop_bits: number of stop bits, 1 bit(default), 1.5 bits or 2 bits
flowctrl: flow control pin
rtsPin: request to send pin
ctsPin: clear to send pin

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial_Basic:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::listen

Description
Enables the selected software serial port to listen

Syntax
bool SoftwareSerial::listen(void)

Parameters
The function requires no input parameter.

Returns
Returns true if it replaces another

Example Code
NA

Notes and Warnings

NA

SoftwareSerial::end

Description
Same as stopListening

Syntax
void SoftwareSerial::end(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::isListening

Description
Tests to see if requested software serial port is actively listening

Syntax
bool SoftwareSerial::isListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if the port is listening.

Example Code

NA

Notes and Warnings
NA

SoftwareSerial::stopListening

Description
Stop listening on the port

Syntax
bool SoftwareSerial::stopListening(void)

Parameters
The function requires no input parameter.

Returns
The function returns “True” if listening on the port is stopped.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::peek

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::peek(void)

Parameters
The function requires no input parameter.

Returns

The function returns the character read, or returns "-1" if none is available.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::write

Description
Prints data to the transmit pin of the software serial port as raw bytes

Syntax
size_t SoftwareSerial::write(uint8_t b)

Parameters
b: byte to be written

Returns
The function returns the number of bytes written.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::read

Description
Return a character that was received on the RX pin of the software serial port

Syntax
int SoftwareSerial::read(void)

Parameters
The function requires no input parameter.

Returns
The function returns the character read, or -1 if none is available.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::available

Description
Get the number of bytes available for reading from a software serial port

Syntax
int SoftwareSerial::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available to read.

Example Code
Example: SoftwareSerialExample
The example demonstrates a software serial test, it receives from serial RX and sends it to serial TX. Details of the code can be found in the previous section of SoftwareSerial:: SoftwareSerial.

Notes and Warnings
NA

SoftwareSerial::flush

Description
Flush the received buffer

Syntax
void SoftwareSerial::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setBufferSize

Description
Set buffer size

Syntax
void SoftwareSerial::setBufferSize(uint32_t buffer_size)

Parameters
buffer_size: the size of the serial buffer

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SoftwareSerial::setAvailableCallback

Description
Set available callback

Syntax
void SoftwareSerial::setAvailableCallback(void (*callback)(char c))

Parameters
*callback: user-defined serial callback function

Returns
The function returns nothing.

Example Code
Example: SoftwareSerialIrqCallback
This example demonstrates the software serial testing using IRQ callback and semaphore. Set callback function "mySerialCalback" to software serial. Whenever there is data comes in, "mySerialCallback" is invoked. In this sketch, it does nothing until the end of the line. And then it sends a semaphore. The loop() uses a non-busy loop to wait for the semaphore. To test this sketch, you need to type something on software serial and then press Enter.

```
/*
The circuit: (BOARD RTL8195A)
RX is digital pin 0 (connect to TX of other devices)
TX is digital pin 1 (connect to RX of other devices)
*/
#include <SoftwareSerial.h>
#if defined(BOARD_RTL8195A)
SoftwareSerial mySerial(0, 1); // RX, TX
#elif defined(BOARD_RTL8710)
SoftwareSerial mySerial(17, 5); // RX, TX
#else
SoftwareSerial mySerial(0, 1); // RX, TX
#endif
uint32_t semaID;
// The callback is hooking at UART IRQ handler and please don't do heavy task here.
void mySerialCallback(char c)
{
```

```
/* The parameter c is only for peeking. The actual data is
* still in the buffer of SoftwareSerial.
*/
if (c == 'r' || c == 'n') {
os_semaphore_release(semaID);
}
}
void setup() {
// use 1 count for binary semaphore
semaID = os_semaphore_create(1);
// There is a token in the semaphore, clear it.
os_semaphore_wait(semaID, 0xFFFFFFFF);
// set the data rate for the SoftwareSerial port
mySerial.begin(38400);
mySerial.setAvailableCallback(mySerialCallback);
}
void loop() { // run over and over
// wait semaphore for 5s timeout
if (os_semaphore_wait(semaID, 5 * 1000)) {
// we got data before timeout
while(mySerial.available()) {
mySerial.print((char)mySerial.read());
}
mySerial.println();
} else {
mySerial.println("No data comes in.");
}
}
```

Notes and Warnings
NA

SoftwareSerial::handle_interrupt

Description
A private method handles the interrupt

Syntax
void handle_interrupt(uint32_t id, uint32_t event)

Parameters
id: the interupt id
event: interrupt event

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba Software Serial related APIs and examples are works based on libraries formerly known as `NewSoftSerial.h` by Mikal Hart (http://arduiniana.org/libraries/newsoftserial).
These include,

- `SoftwareSerial.cpp`
- `SoftwareSerial.h`

These libraries are under GNU Lesser General Public License.

The Ameba GPS related APIs and examples are works based on Adafruit GPS library written by Limor Fried/Ladyada for Adafruit Industries (http://www.adafruit.com/products/746).
These include,

- `Adafruit_GPS.cpp`
- `Adafruit_GPS.h`

These libraries are under BSD License.

SPI

Class AmebaILI9341

AmebaILI9341 Class

Description

Defines a class to use ILI9341 TFT SPI display for Ameba.

Syntax

class AmebaILI9341

Members

Public Constructors	
AmebaILI9341::AmebaILI9341	Constructs an AmebaILI9341 object
Public Methods	
AmebaILI9341::begin	Initialize SPI, pin mapping and screen configuration
AmebaILI9341::setAddress	Initialize image size and position
AmebaILI9341::writecommand	SPI transfer a command
AmebaILI9341::writedata	SPI transfer a piece of data
AmebaILI9341::setRotation	Set screen orientation
AmebaILI9341::fillScreen	Fill the screen with a color
AmebaILI9341::clr	Clear screen
AmebaILI9341::fillRectangle	Fill a rectangular space with a color
AmebaILI9341::drawPixel	Turn on a pixel on the screen
AmebaILI9341::drawChar	To print a character on the screen
AmebaILI9341::drawLine	Draw line on the screen
AmebaILI9341::drawRectangle	Draw a rectangle on the screen
AmebaILI9341::drawCircle	Draw a circle on the screen
AmebaILI9341::write	Same as drawChar
AmebaILI9341::getWidth	Return the width 240
AmebaILI9341::getHeight	Return the height 320
AmebaILI9341::setCursor	Set cursor to the desired position
AmebaILI9341::setForeground	Set foreground color
AmebaILI9341::setBackground	Set background color
AmebaILI9341::setFontSize	Set character font size
AmebaILI9341::reset	Reset pin to High or Low

AmebaILI9341::AmebaILI9341

Description

Constructs an AmebaILI9341 object and set CS, DC and RESET pins .

Syntax

AmebaILI9341::AmebaILI9341(int csPin, int dcPin, int resetPin)

Parameters

csPin: pin for Chip Select dcPin: pin for Data/Command resetPin: pin for Reset

Returns

The function returns nothing.

Example Code

Example: : PM25_ON_ILI9341_TFT_LCD

This example demonstrates how to read pm2.5 value on PMS 3003 air-condition sensor and display it on ILI9341 TFT LCD.

/*

PMS 3003 pin map is as follow:

PIN1 :VCC, connect to 5V

PIN2 :GND

PIN3 :SET, 0:Standby mode, 1:operating mode

PIN4 :RXD :Serial RX

PIN5 :TXD :Serial TX

PIN6 :RESET

PIN7 :NC

PIN8 :NC

In this example, we only use Serial to get PM 2.5 value.

The circuit:

* RX is digital pin 0 (connect to TX of PMS 3003)

* TX is digital pin 1 (connect to RX of PMS 3003)

For RTL8195A ILI9341 TFT LCD with SPI interface has these pins:

D/C : connect to pin 9

CS : connect to pin 10

MOSI : connect to pin 11

MISO : connect to pin 12

CLK : connect to pin 13

VCC : connect to 3V3

GND : connect to GND

*/

#include "SoftwareSerial.h"

#include "SPI.h"

#include "AmebaILI9341.h"

#if defined(BOARD_RTL8195A)

SoftwareSerial mySerial(0, 1); // RX, TX

#define TFT_RESET 8

#define TFT_DC 9

#define TFT_CS 10

#elif defined(BOARD_RTL8710)

```
SoftwareSerial mySerial(17, 5); // RX, TX
// IMPORTANT: Due to limit pin, we do not connect TFT_RESET pin.
#define TFT_RESET 0xFFFFFFFF
#define TFT_DC 2
#define TFT_CS 10
#endif
AmebaILI9341 tft = AmebaILI9341(TFT_CS, TFT_DC, TFT_RESET);
#define ILI9341_SPI_FREQUENCY 20000000
#define pmsDataLen 32
uint8_t buf[pmsDataLen];
int idx = 0;
int pm10 = 0;
int last_pm25 = 0;
int pm25 = 0;
int pm100 = 0;
uint16_t pm25color[] = {
0x9FF3,
0x37E0,
0x3660,
0xFFE0,
0xFE60,
0xFCC0,
0xFB2C,
0xF800,
0x9800,
0xC99F
};
void setup() {
Serial.begin(57600);
mySerial.begin(9600); // PMS 3003 UART has baud rate 9600
SPI.setDefaultFrequency(ILI9341_SPI_FREQUENCY);
tft.begin();
drawPictureFrames();
}
void loop() { // run over and over
uint8_t c;
```

```
idx = 0;
memset(buf, 0, pmsDataLen);
while (true) {
while (c != 0x42) {
while (!mySerial.available());
c = mySerial.read();
}
while (!mySerial.available());
c = mySerial.read();
if (c == 0x4d) {
// now we got a correct header)
buf[idx++] = 0x42;
buf[idx++] = 0x4d;
break;
}
}
while (idx != pmsDataLen) {
while(!mySerial.available());
buf[idx++] = mySerial.read();
}
pm10 = ( buf[10] << 8 ) | buf[11];
last_pm25 = pm25;
pm25 = ( buf[12] << 8 ) | buf[13];
pm100 = ( buf[14] << 8 ) | buf[15];
updateValueToTftScreen();
}
void drawPictureFrames() {
tft.setRotation(1);
tft.clr();
tft.setFontSize(1);
// Upper title
tft.setFontSize(1);
tft.setCursor(20,20);
tft.print("PM2.5 DETECTOR");
// PM 2.5 Circle Frame
tft.drawCircle(100,130,60, ILI9341_BLUE);
```

```
tft.drawCircle(100,130,61, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(90,85);
tft.print("PM2.5");
tft.setFontSize(1);
tft.setCursor(90,170);
tft.print("um/m3");
// PM 10 Circle Frame
tft.drawCircle(220,70,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(210,40);
tft.print("PM10");
tft.setFontSize(1);
tft.setCursor(205,95);
tft.print("um/m3");
// PM 1.0 Circle Frame
tft.drawCircle(220,170,40, ILI9341_BLUE);
tft.setFontSize(1);
tft.setCursor(205,140);
tft.print("PM1.0");
tft.setFontSize(1);
tft.setCursor(205,195);
tft.print("um/m3");
// right side bar, referenced from: http://taqm.epa.gov.tw/taqm/tw/
tft.fillRectangle(290, 30+ 0*2, 10, 12*2, pm25color[0]); // 0~11
tft.fillRectangle(290, 30+12*2, 10, 12*2, pm25color[1]); // 12-23
tft.fillRectangle(290, 30+24*2, 10, 12*2, pm25color[2]); // 24-35
tft.fillRectangle(290, 30+36*2, 10, 6*2, pm25color[3]); // 36-41
tft.fillRectangle(290, 30+42*2, 10, 6*2, pm25color[4]); // 42-47
tft.fillRectangle(290, 30+48*2, 10, 6*2, pm25color[5]); // 48-53
tft.fillRectangle(290, 30+54*2, 10, 6*2, pm25color[6]); // 54-58
tft.fillRectangle(290, 30+59*2, 10, 6*2, pm25color[7]); // 59-64
tft.fillRectangle(290, 30+65*2, 10, 6*2, pm25color[8]); // 65-70
tft.fillRectangle(290, 30+71*2, 10, 10*2, pm25color[9]); // >=71
tft.setCursor(302, 30);
tft.setFontSize(1);
```

```
tft.print("0");
tft.setCursor(302, 30+36*2);
tft.print("36");
tft.setCursor(302, 30+54*2);
tft.print("54");
tft.setCursor(302, 30+71*2);
tft.print("71");
// bottom right text
tft.setCursor(210,230);
tft.setFontSize(1);
tft.print("Powered by Realtek");
updateValueToTftScreen();
}
void updateValueToTftScreen() {
tft.setCursor(60, 111);
tft.setFontSize(5);
tft.setForeground( getPm25Color(pm25) );
if (pm25 < 10) {
tft.print(" ");
} else if (pm25 < 100) {
tft.print(" ");
}
tft.print(pm25);
tft.setCursor(195,60);
tft.setFontSize(3);
if (pm100 < 10) {
tft.print(" ");
} else if (pm100 < 100) {
tft.print(" ");
}
tft.print(pm100);
tft.setCursor(198,160);
if (pm10 < 10) {
tft.print(" ");
} else if (pm10 < 100) {
tft.print(" ");
```

```
}
tft.print(pm10);
tft.setFontSize(1);
tft.setForeground(ILI9341_WHITE);
if (last_pm25 > 80) {
tft.fillRectangle(275, 80*2+30-3, 12, 8, ILI9341_BLACK);
} else {
tft.fillRectangle(275, last_pm25*2+30-3, 12, 8, ILI9341_BLACK);
}
if (pm25 > 80) {
tft.setCursor(275, 80*2+30-3);
} else {
tft.setCursor(275, pm25*2+30-3);
}
tft.print("=>");
}
uint16_t getPm25Color(int v) {
if (v < 12) {
return pm25color[0];
} else if (v < 24) {
return pm25color[1];
} else if (v < 36) {
return pm25color[2];
} else if (v < 42) {
return pm25color[3];
} else if (v < 48) {
return pm25color[4];
} else if (v < 54) {
return pm25color[5];
} else if (v < 59) {
return pm25color[6];
} else if (v < 65) {
return pm25color[7];
} else if (v < 71) {
return pm25color[8];
} else {
```

return pm25color[9];

}

}

Notes and Warnings

NA

AmebaILI9341::begin

Description

Initialize hardware SPI, pin mapping and screen configuration

Syntax

void AmebaILI9341::begin(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

This method is required to run first before other operations on the display.

AmebaILI9341::setAddress

Description

Initialize image size and positioning on the display

Syntax

void AmebaILI9341::setAddress(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: rightmost coordinate of the image y1: bottom coordinate of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Do not use this to set the cursor, use the “setCursor” method instead.

AmebaILI9341::writecommand

Description

Write a single-byte command to display

Syntax

void AmebaILI9341::writecommand(uint8_t command)

Parameters

command: a single byte command

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::writedata

Description

Write 1 byte of data to display

Syntax

void AmebaILI9341::writedata(uint8_t data)

Parameters

data: 1 byte data

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Only use this method to write 1 byte at a time.

AmebaILI9341::setRotation

Description

Setting screen orientation, “0” for no rotation, “1” for 90 degrees rotation and so on so forth.

Syntax

void AmebaILI9341::setRotation(uint8_t m)/span> **Parameters**

m: one of the 4 rotation modes -> “0” for no rotation, “1” for 90^0, “2” for 180^0, “3” for 270^0

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

if m=4, it’s equivalent to mode 0, and m=5 for mode 1, m=6 for mode 2 so on so forth.

AmebaILI9341::fillScreen

Description

Fill the entire screen with one color

Syntax

void AmebaILI9341::fillScreen(uint16_t color)

Parameters

color: a 16-bit color reference defined in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Refer to AmebaILI9341.h for available colors.

AmebaILI9341::clr

Description

Fill the entire screen with a certain background-color

Syntax

void AmebaILI9341::clr(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341

Notes and Warnings

background-color can be set by calling setBackground method.

AmebaILI9341::fillRectangle

Description

Fill a rectangular space with a color on the screen

Syntax

void AmebaILI9341::fillRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::drawPixel

Description

Turn on a pixel on the screen

Syntax

void AmebaILI9341::drawPixel(int16_t x, int16_t y, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawChar

Description

Draw character on the screen

Syntax

void AmebaILI9341::drawChar(unsigned char c) void AmebaILI9341::drawChar(int16_t x, int16_t y, unsigned char c, uint16_t _fontcolor, uint16_t _background, uint8_t _fontsize)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image c: a character _fontcolor: font color _background: background color _fontsize: font size

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

In the actual example, the Print method is used to print a string of character on the screen instead of using this method.

AmebaILI9341::drawLine

Description

Draw a straight line on the screen

Syntax

void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1) void AmebaILI9341::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image x1: leftmost coordinate of the image y1: top coordinate of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawRectangle

Description

Draw a rectangular shape on the screen

Syntax

void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h) void AmebaILI9341::drawRectangle(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

Parameters

x: leftmost coordinate of the image y: top coordinate of the image w: width of the image h: height of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::drawCircle

Description

Draw a circular shape on the screen

Syntax

void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r) void AmebaILI9341::drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)

Parameters

x0: leftmost coordinate of the image y0: top coordinate of the image r: radius of the image color: the color of the image

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

Include “AmebaServo.h” to use the class function.

AmebaILI9341::write

Description

Same as drawChar, write a character on the screen

Syntax

size_t AmebaILI9341::write(uint8_t c)

Parameters

c: a character to be written on the screen

Returns

Number of bytes written

Example Code

NA

Notes and Warnings

This an inherited method from Print class and is seldom used.

AmebaILI9341::getWidth

Description

Get the width of the image

Syntax

int16_t AmebaILI9341::getWidth(void)

Parameters

The function requires no input parameter.

Returns

Width of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::getHeight

Description

Get the height of the image

Syntax

int16_t AmebaILI9341::getHeight(void)

Parameters

The function requires no input parameter.

Returns

Height of the image

Example Code

NA

Notes and Warnings

NA

AmebaILI9341::setCursor

Description

Set the cursor to a specific position on the screen

Syntax

void AmebaILI9341::setCursor(int16_t x, int16_t y)

Parameters

x: coordinate on the x-axis y: coordinate on the y-axis

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setForeground

Description

Set foreground color

Syntax

void AmebaILI9341::setForeground(uint16_t color)

Parameters

color: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setBackground

Description

Set background color

Syntax

void AmebaILI9341::setBackground(uint16_t _background)

Parameters

_background: one of the colors available in AmebaILI9341.h

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::setFontSize

Description

Set the font size of the characters printed on the screen.

Syntax

void AmebaILI9341::setFontSize(uint8_t size)

Parameters

size: font size, default 1 for smallest, 5 for largest font size

Returns

The function returns nothing.

Example Code

Example: PM25_ON_ILI9341_TFT_LCD

Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings

NA

AmebaILI9341::reset

Description

Reset the pin to High or Low

Syntax

void AmebaILI9341::reset(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings

NA

Class SPISettings_SPIClass

SPISettings Class

Description
Defines a class to set SPI parameters.

Syntax
class SPISettings

Members

Public Constructors	
SPISettings::SPISettings	Create a SPISettings object and set SPI clock speed, bit order and data mode

SPISettings::SPISettings

Description
Construct an object and configure SPI parameters — clock speed, bit order and data model to the preferred default value.

Syntax
SPISettings YourObject(uint32_t clock, BitOrder bitOrder, uint8_t dataMode);

Parameters
clock: SPI clock speed, default is 4000000
bitOrder: order of bit stream, MSB first or LSB first, default is MSBFIRST
dataMode: There are 4 modes -> SPI_MODE0~3, default is SPI_MODE0

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This class seldom used alone, it is always used with beginTransaction() as a parameter in SPIClass.

SPIClass Class

Description
Defines a class of SPI implementation for Ameba.

Syntax
class SPIClass

Members

Public Constructors	
SPIClass::SPIClass	Constructs an SPI object
Public Methods	
SPIClass::transfer	Transfer data through SPI
SPIClass::transfer16	Transfer a 16-bits data through SPI
SPIClass::beginTransaction	Set slave select pin and SPI initial settings
SPIClass::endTransaction	Stop SPI transaction
SPIClass::begin	Associate each SPI pin to Ameba pin using ameba HAL APIs
SPIClass::end	Stop SPI master mode
SPIClass::setBitOrder	Set MSB first or LSB first
SPIClass::setDataMode	Set to one of the four data modes
SPIClass::setClockDivider	Set to correct clock speed (no effect on Ameba)
SPIClass::setDefaultFrequency	Set default SPI frequency

SPIClass::SPIClass

Description
Construct an SPI object, create a pointer to the object, and attach “MOSI, MISO, CLK, and SS” to each pin on Ameba.

Syntax
SPIClass(void *pSpiObj, int mosi, int miso, int clk, int ss);

Parameters
pSpiObj: SPI pointer to the object
mosi: master out slave in
miso: master in slave out
clk: clock
ss: slave select

Returns
The function returns nothing.

Example Code

SPIClass SPI((void *)(&spi_obj0), 11, 12, 13, 10);

Notes and Warnings
2 SPI objects are created in the library for 2 different hardware SPI on Ameba (if applicable), use “SPI” for first hardware SPI and “SPI1” for the second.

SPIClass::transfer

Description
Calling HAL API to send data in the buffer to the slave

Syntax
byte SPIClass::transfer (byte _pin, uint8_t _data, SPITransferMode _mode);
byte SPIClass::transfer (uint8_t _data, SPITransferMode _mode);
void SPIClass::transfer (byte _pin, void *_buf, size_t _count, SPITransferMode _mode);
void SPIClass::transfer (void *_buf, size_t _count, SPITransferMode _mode);

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode
_count: number of bytes of data
_buf: data buffer

Returns
Void or “0” in case of error, “d” in case success

Example Code
NA

Notes and Warnings
NA

SPIClass::transfer16

Description
Same as “transfer” method above except data being of 16-bits.

Syntax
uint16_t SPIClass::transfer16(byte _pin, uint16_t _data, SPITransferMode _mode)
uint16_t SPIClass::transfer16(uint16_t _data, SPITransferMode _mode)

Parameters
_pin: Slave select pin
_data: Actual data being sent over
_mode: SPI transfer mode

Returns
The data being transferred

Example Code
NA

Notes and Warnings
NA

SPIClass::beginTransaction

Description
Set slave select pin and initialize SPI with default settings using SPISettings class.

Syntax
void SPIClass::beginTransaction(uint8_t pin, SPISettings settings)
void SPIClass::beginTransaction(SPISettings settings)

Parameters
pin: slave select pin
settings: an object of SPISettings class

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Refer to SPISettings class for details of the initial settings.

SPIClass::endTransaction

Description
Set slave select pin to 1 and stop SPI transaction.

Syntax
void SPIClass::endTransaction(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::begin

Description
Calling HAL APIs to initialize SPI pins to physical Ameba pins and set SPI format and frequency

Syntax
void SPIClass::begin(void)
void SPIClass::begin(int ss)

Parameters
void or ss: slave select

Returns
The function returns nothing.

Example Code

NA

Notes and Warnings
This is a required method to use SPI on Ameba.

SPIClass::end

Description
Free hardware SPI from any activity.

Syntax
void SPIClass::end(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setBitOrder

Description
A specific method to set bit order to either MSB first or LSB first and set slave select pin.

Syntax
void SPIClass::setBitOrder(uint8_t _pin, BitOrder _bitOrder)
void SPIClass::setBitOrder(BitOrder _order)

Parameters
_pin: slave select
_bitOrder: bit order -> either MSB first or LSB first
_order: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setDataMode

Description
A specific method to set data mode to one of the 4 modes (default: SPI_MODE0) and set slave lave select pin.

Syntax
void SPIClass::setDataMode(uint8_t _pin, uint8_t _mode)
void SPIClass::setDataMode(uint8_t _mode)

Parameters
_pin: slave select
_mode: one of the 4 modes (default: SPI_MODE0)

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

SPIClass::setClockDivider

Description
A specific method to set to divider in order to get correct clock speed

Syntax
void SPIClass::setClockDivider(uint8_t _pin, uint8_t _divider)

void SPIClass::setClockDivider(uint8_t _div)

Parameters
_pin: slave select
_divider: clock divider
_div: same as above

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This function does not affect the Ameba board.

SPIClass::setDefaultFrequency

Description
A specific method to set default SPI frequency

Syntax
void SPIClass::setDefaultFrequency(int _frequency)

Parameters
_frequency: the default SPI frequency

Returns
The function returns nothing.

Example Code
Example: PM25_on_ILI9341_TFT_LCD
Details of the code are given in the previous section of AmebaILI9341:: AmebaILI9341.

Notes and Warnings
Take note that defaultFrequency = _frequency.

Readme

The Ameba SPI related APIs and examples are works based on SPI Master library for arduino written by Cristian Maglie <c.maglie@arduino.cc> and Paul Stoffregen <paul@pjrc.com> (Transaction API).

These include,
SPI.cpp
SPI.h

These libraries are under GNU Lesser General Public License, version 2.1.

Sys

Wiring_OS_API

Wiring OS API

Description
A wrapper to CMSIS (Cortex Microcontroller Software Interface Standard) OS API which serve as a RTOS to create multi-threaded application with real-time behaviour.

Syntax
NA

Members

Public Methods	
os_thread_create_arduino	Create a thread and add it to Active Threads and set it to state READY
os_thread_get_id_arduino	Return the thread ID of the current running thread
os_thread_terminate_arduino	Terminate execution of a thread and remove it from Active Threads
os_thread_yield_arduino	Pass control to next thread that is in state READY
os_thread_set_priority_arduino	Change priority of an active thread
os_thread_get_priority_arduino	Get current priority of an active thread
os_signal_set_arduino	Set the specified Signal Flags of an active thread
os_signal_clear_arduino	Clear the specified Signal Flags of an active thread
os_signal_wait_arduino	Wait for one or more Signal Flags to become signaled for the current RUNNING thread
os_timer_create_arduino	Create a timer
os_timer_start_arduino	Start or restart a timer
os_timer_stop_arduino	Stop the timer
os_timer_delete_arduino	Delete a timer that was created by os_timer_create
os_semaphore_create_arduino	Create and Initialize a Semaphore object used for managing resources
os_semaphore_wait_arduino	Wait until a Semaphore token becomes available
os_semaphore_release_arduino	Release a Semaphore token
os_semaphore_delete_arduino	Delete a Semaphore that was created by os_semaphore_create
os_get_free_heap_size_arduino	Return the available heap memory space when called

os_thread_create_arduino

Description

Create a thread and add it to Active Threads and set it to state READY.

Syntax

uint32_t os_thread_create_arduino (void (*task)(const void *argument), void *argument, int priority, uint32_t stack_size);

Parameters

task: task Function pointer which is the thread body. It should not run into the end of function unless os_thread_terminate is invoked

argument: the data pointer which brings to task

priority: The underlying os is FreeRTOS. It executes tasks with highest priority which are not in idle state.

stack_size: The stack_size is used as memory heap only for this task.

Returns

The thread id which is used in thread operation and signalling.

Example Code

NA

Notes and Warnings

NA

os_thread_get_id_arduino

Description
Return the thread ID of the current running thread

Syntax
uint32_t os_thread_get_id_arduino (void);

Parameters
The function requires no input parameter.

Returns
Current thread id which calls os_thread_get_id.

Example Code
NA

Notes and Warnings
NA

os_thread_terminate_arduino

Description
Terminate execution of a thread and remove it from Active Threads

Syntax
uint32_t os_thread_terminate_arduino (uint32_t thread_id);

Parameters
thread_id: Terminate the thread with specific thread_id

Returns
os_status code

Example Code
NA

Notes and Warnings
Thread should not ended without terminate first.

os_thread_yield_arduino

Description
Pass control to next thread that is in state READY

Syntax
uint32_t os_thread_yield_arduino (void);

Parameters
The function requires no input parameter.

Returns
os_status code

Example Code
NA

Notes and Warnings
By default, the minimal execution unit is 1 millisecond. In a scenario that if a thread with smaller want to handout execution right to a thread with higher priority immediately without waiting for the ending of current 1 millisecond, then invoke os_thread_yield can transfer exection right to OS's idle task and check which is the next execution thread.

os_thread_set_priority_arduino

Description
Change priority of an active thread

Syntax
uint32_t os_thread_set_priority_arduino (uint32_t thread_id, int priority);

Parameters
thread_id: The target thread with the thread id to be changed
priority: The updated priority

Returns

The function returns nothing.

Example Code
NA

Notes and Warnings
NA

os_thread_get_priority_arduino

Description
Get current priority of an active thread

Syntax
uint32_t os_thread_get_priority_arduino (uint32_t thread_id);

Parameters
thread_id: The target thread with the thread id to be searched

Returns
os_priority

Example Code
NA

Notes and Warnings
NA

os_signal_set_arduino

Description
Set the specified Signal Flags of an active thread

Syntax
int32_t os_signal_set_arduino (uint32_t thread_id, int32_t signals);

Parameters

thread_id: Send signal to a thread with the thread id
signals: the signals to be send

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_clear_arduino

Description
Clear the specified Signal Flags of an active thread

Syntax
int32_t os_signal_clear_arduino (uint32_t thread_id, int32_t signals);

Parameters
thread_id: Clear signal to a thread with the thread id
signals: The signals to be clear

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_signal_wait_arduino

Description
Wait for one or more Signal Flags to become signaled for the current RUNNING thread

Syntax
os_event_t os_signal_wait_arduino (int32_t signals, uint32_t millisec);

Parameters
signals: the signals to be wait
millisec: the timeout value if no signal comes in. Fill in 0xFFFFFFFF for infinite wait

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_create_arduino

Description
Create a timer

Syntax
uint32_t os_timer_create_arduino (void (*callback)(void const *argument), uint8_t isPeriodic, void *argument);

Parameters
callback: The function to be invoke when timer timeout
isPeriodic: OS_TIMER_ONCE or OS_TIMER_PERIODIC
argument: The argument that is bring into callback function

Returns
timer id

Example Code
NA

Notes and Warnings
NA

os_timer_start_arduino

Description
Start or restart a timer

Syntax
uint32_t os_timer_start_arduino (uint32_t timer_id, uint32_t millisec);

Parameters
timer_id: The timer id obtained from by os_timer_create
millisec: The delays after timer starts

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_stop_arduino

Description
Stop the timer

Syntax
uint32_t os_timer_stop_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_timer_delete_arduino

Description
Delete a timer that was created by os_timer_create

Syntax
uint32_t os_timer_delete_arduino (uint32_t timer_id);

Parameters
timer_id: The timer id obtained from by os_timer_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_create_arduino

Description
Create and Initialize a Semaphore object used for managing resources

Syntax
uint32_t os_semaphore_create_arduino (int32_t count);

Parameters
count: The number of available resources

Returns
semaphore ID

Example Code
NA

Notes and Warnings
NA

os_semaphore_wait_arduino

Description
Wait until a Semaphore token becomes available

Syntax
int32_t os_semaphore_wait_arduino (uint32_t semaphore_id, uint32_t millisec);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create
millisec: timeout value

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_release_arduino

Description
Release a Semaphore token

Syntax
uint32_t os_semaphore_release_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_semaphore_delete_arduino

Description
Delete a Semaphore that was created by os_semaphore_create

Syntax
uint32_t os_semaphore_delete_arduino (uint32_t semaphore_id);

Parameters
semaphore_id: semaphore id obtained from os_semaphore_create

Returns
os_status code

Example Code
NA

Notes and Warnings
NA

os_get_free_heap_size_arduino

Description
Return the available heap memory space when called

Syntax
size_t os_get_free_heap_size_arduino(void);

Parameters
The function requires no input parameter.

Returns
current free heap size

Example Code
Example: MemInfo

Notes and Warnings
NA

WDT

Class WDT

WDT Class

Description
A class used for initializing, starting, stopping watchdog timer.

Syntax
class WDT

Members

Public Constructors	
A public constructor should not be used as this class is intended to be a singleton class. Access member functions using the object instance named WDT.	

Public Methods	
WDT:: InitWatchdog	Initializes the watchdog, include time setting, and mode register
WDT:: StartWatchdog	Start the watchdog counting
WDT:: StopWatchdog	Stop the watchdog counting
WDT:: RefreshWatchdog	Refresh the watchdog counting to prevent WDT timeout
WDT:: InitWatchdogIRQ	Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler

WDT:: InitWatchdog

Description
Initializes the watchdog, include time setting, and mode register.

Syntax
void InitWatchdog(uint32_t timeout_ms);

Parameters
timeout_ms: the watch-dog timer timeout value in millisecond (ms). The default action after watchdog timer timeout is to reset the whole system.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer

```
/*
* This example describes how to use watchdog api.
* In this example, watchdog is setup to 5s timeout.
* Watchdog won't bark if we refresh it before timeout in smallTask.
* The timer is also reloaded after refresh.
* Otherwise, while running bigTask, watchdog will restart system in default or call callback function if registered.
*/
#include "wdt.h"
#define RUN_CALLBACK_IF_WATCHDOG_BARKS (0)
WDT wdt;
void setup() {
Serial.begin(115200);
wdt.InitWatchdog(5000); // setup 5s watchdog
#if RUN_CALLBACK_IF_WATCHDOG_BARKS
wdt.InitWatchdogIRQ(my_watchdog_irq_handler, 0);
#else
// system would restart in default when watchdog barks
#endif
wdt.StartWatchdog(); // enable watchdog timer
successfulTask();
failedTask();
while (1)
```

```
;
}
void loop() {
}
void successfulTask(void) {
Serial.println("……doing small task……");
for (int i = 0; i < 50000000; i++) // dummy task
asm(" nop");
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
/*
* Doing this task will lead to failed refresh the
* watchdog timer within the time limits of 5 seconds
*/
void failedTask(void) {
Serial.println("……doing big task……");
for (int i = 0; i < 10; i++) {
Serial.print("doing dummy task #");
Serial.println(i, DEC);
for (int j = 0; j < 50000000; j++) // dummy task
asm(" nop");
}
Serial.println("refresh watchdogrn");
wdt.RefreshWatchdog();
}
void my_watchdog_irq_handler(uint32_t id) {
printf("watchdog barks!!!rn");
WDG_Cmd(DISABLE);
}
```

Notes and Warnings
NA

WDT:: StartWatchdog

Description
Start the watchdog counting.

Syntax
void StartWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: StopWatchdog

Description
Stop the watchdog counting.

Syntax
void StopWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: RefreshWatchdog

Description
Refresh the watchdog counting to prevent WDT timeout.

Syntax
void RefreshWatchdog(void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WDT:: InitWatchdogIRQ

Description
Switch the watchdog timer to interrupt mode and register a watchdog timer timeout interrupt handler. The interrupt handler will be called when the watchdog timer is timeout.

Syntax
void WDT::InitWatchdogIRQ(wdt_irq_handler handler, uint32_t id)

Parameters
handler: the callback function for WDT timeout interrupt.
id: the parameter for the callback function

Returns
The function returns nothing.

Example Code
Example: WatchdogTimer
You may refer to the code in previous section of WDT::InitWatchdog.

Notes and Warnings
NA

WiFi

Class WiFi

WiFiClass Class

Description
Defines a class of WiFi and network implementation for Ameba.

Syntax
class WiFiClass

Members

Public Constructors	
WiFiClass::WiFiClass	Constructs a WiFiClass object and initializes the WiFi libraries and network settings
Public Methods	
WiFi-Class::firmwareVersion	Get firmware version
WiFiClass:: begin	Start Wifi connection for OPEN networks
WiFiClass:: config	Configure network IP settings
WiFiClass:: setDNS	Set the DNS server IP address to use
WiFiClass:: disconnect	Disconnect from the network
WiFiClass:: macAddress	Get the interface MAC address
WiFiClass:: localIP	Get the interface IP address
WiFiClass:: subnetMask	Get the interface subnet mask address
WiFiClass:: gatewayIP	Get the gateway IP address
WiFiClass:: SSID	Return the current SSID associated with the network
WiFiClass:: BSSID	Return the current BSSID associated with the network
WiFiClass:: RSSI	Return the current RSSI (Received Signal Strength in dBm) associated with the network
WiFiClass:: encryption-Type	Return the Encryption Type associated with the network
WiFiClass:: scanNetworks	Start scan WiFi networks available
WiFiClass:: SSID	Return the SSID discovered during the network scan
WiFiClass:: encryption-Type	Return the encryption type of the networks discovered during the scanNetworks
WiFiClass:: encryption-TypeEx	Return the security type and encryption type of the networks discovered during the scanNetworks
WiFiClass:: RSSI	Return the RSSI of the networks discovered during the scanNetworks
WiFiClass:: status	Return Connection status
WiFiClass:: hostByName	Resolve the given hostname to an IP address
WiFiClass:: apbegin	Start AP mode
WiFiClass:: disablePower-Save	Disable power-saving mode

WiFiClass::WiFiClass

Description

Constructs a WiFiClass object and initializes the WiFi libraries and network settings.

Syntax

WiFiClass::WiFiClass()

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

NA

Notes and Warnings
An instance of WiFiClass is created as WiFi inside WiFi.h and is extern for direct use.

WiFiClass::firmwareVersion

Description
Get firmware version

Syntax
char* WiFiClass::firmwareVersion()

Parameters
The function requires no input parameter.

Returns
WiFi firmware version

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFI network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>

// char ssid[] = "yourNetwork"; // your network SSID (name)

// char pass[] = "secretPassword"; // your network password

char ssid[] = "SINGTEL-D45F"; // your network SSID (name)

char pass[] = "mooxuteeth"; // your network key

int status = WL_IDLE_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {
```

```
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to WPA SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
```

```
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
```

```
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::begin

Description
Start Wifi connection for OPEN networks

Syntax
int WiFiClass::begin(char* ssid)
int WiFiClass::begin(char* ssid, uint8_t key_idx, const char *key)
int WiFiClass::begin(char* ssid, const char *passphrase)

Parameters
ssid: Pointer to the SSID string
key_idx: The key index to set. Valid values are 0-3.
key: Key input buffer.
passphrase: Passphrase. Valid characters in a passphrase must be between ASCII 32-126 (decimal).

Returns
WiFi status

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::config

Description
Configure network settings for the WiFi network

Syntax
void WiFiClass::config(IPAddress local_ip)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway)
void WiFiClass::config(IPAddress local_ip, IPAddress dns_server, IPAddress gateway, IPAddress subnet)

Parameters
local_ip: Local device IP address to use on the network
dns_server: IP address of the DNS server to use
gateway: IP address of the gateway device on the network
subnet: Subnet mask for the network, expressed as a IP address

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
This will disable the DHCP client when connecting to a network, and will require the network accepts a static IP. The configured IP addresses will also apply to AP mode, but the DHCP server will not be disabled in AP mode.

WiFiClass::setDNS

Description
Configure the IP address of the DNS server to use

Syntax
void WiFiClass::setDNS(IPAddress dns_server1)
void WiFiClass::setDNS(IPAddress dns_server1, IPAddress dns_server2)

Parameters
dns_server1: IP address of DNS server to use
dns_server2: IP address of DNS server to use

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClass::disconnect

Description
Disconnect from the network

Syntax
int WiFiClass::disconnect()

Parameters
The function requires no input parameter.

Returns
The function returns one value of wl_status_t enum as an integer.

Example Code
NA

Notes and Warnings
NA

WiFiClass::macAddress

Description
Get the interface MAC address

Syntax
uint8_t* WiFiClass::macAddress(uint8_t* mac)

Parameters
mac: an array to store MAC address

Returns
The function returns a pointer to uint8_t array with length WL_MAC_ADDR_LENGTH.

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::localIP

Description

Get the interface IP address

Syntax

IPAddress WiFiClass::localIP()

Parameters

The function requires no input parameter.

Returns

Ip address value

Example Code

Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings

NA

WiFiClass::subnetMask

Description

Get the interface subnet mask address

Syntax

IPAddress WiFiClass::subnetMask()

Parameters
The function requires no input parameter.

Returns
subnet mask address value

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details.

```
#include <WiFi.h>
char ssid[] = "SINGTEL-D45F_5G"; // the name of your network
int status = WL_IDLE_STATUS; // the Wifi radio's status
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to open SSID: ");
Serial.println(ssid);
status = WiFi.begin(ssid);
// wait 10 seconds for connection:
delay(10000);
```

```
}
// you're connected now, so print out the data:
Serial.print("You're connected to the network");
printCurrentNet();
printWifiData();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
```

```
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
Serial.print("Gateway: ");
Serial.println(gateway);
}
void printCurrentNet() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of the router you're attached to:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.println(rssi);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
}
```

Notes and Warnings

NA

WiFiClass::gatewayIP

Description
Get the gateway IP address

Syntax
IPAddress WiFiClass::gatewayIP()

Parameters
The function requires no input parameter.

Returns
The function returns the value of the gateway IP address.

Example Code
Example: ConnectNoEncryption
This example demonstrates how to connect to an unencrypted WiFi network and prints the MAC address of the WiFi shield, the IP address obtained, and other network details. Details of the code can be found in the section of WiFiClass:: subnetMask.

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the current SSID associated with the network

Syntax
char* WiFiClass::SSID()

Parameters
The function requires no input parameter.

Returns
The function returns current SSID associate with the network.

Example Code
Example: ConnectWithWPA

This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::BSSID

Description
Return the current BSSID associated with the network

Syntax
uint8_t* WiFiClass::BSSID(uint8_t* bssid)

Parameters
bssid: an array to store bssid

Returns
pointer to uint8_t array with length WL_MAC_ADDR_LENGTH

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the current RSSI (Received Signal Strength in dBm) associated with the network

Syntax
int32_t WiFiClass::RSSI()

Parameters
The function requires no input parameter.

Returns
The function returns a signed-value signal strength

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the Encryption Type associated with the network

Syntax
uint8_t WiFiClass::encryptionType()

Parameters
The function requires no input parameter.

Returns
The function returns one unsigned integer value of wl_enc_type enum.

Example Code
Example: ConnectWithWPA

Notes and Warnings
NA

WiFiClass::scanNetworks

Description
Start scan WiFi networks available

Syntax
int8_t WiFiClass::scanNetworks()

Parameters
The function requires no input parameter.

Returns
The function returns the number of discovered networks as an integer.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified.

```
#include <WiFi.h>
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// Print WiFi MAC address:
printMacAddress();
}
void loop() {
// scan for existing networks:
Serial.println("Scanning available networks…");
listNetworks();
delay(10000);
}
```

```
void printMacAddress() {
// the MAC address of your Wifi shield
byte mac[6];
// print your MAC address:
WiFi.macAddress(mac);
Serial.print("MAC: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
}
void listNetworks() {
// scan for nearby networks:
Serial.println("* Scan Networks *");
int numSsid = WiFi.scanNetworks();
if (numSsid == -1) {
Serial.println("Couldn't get a wifi connection");
while (true);
}
// print the list of networks seen:
Serial.print("number of available networks:");
Serial.println(numSsid);
// print the network number and name for each network found:
for (int thisNet = 0; thisNet < numSsid; thisNet++) {
Serial.print(thisNet);
Serial.print(") ");
Serial.print(WiFi.SSID(thisNet));
Serial.print("tSignal: ");
Serial.print(WiFi.RSSI(thisNet));
```

```
Serial.print(" dBm");
Serial.print("tEncryptionRaw: ");
printEncryptionTypeEx(WiFi.encryptionTypeEx(thisNet));
Serial.print("tEncryption: ");
printEncryptionType(WiFi.encryptionType(thisNet));
}
}
void printEncryptionTypeEx(uint32_t thisType) {
/* Arduino wifi api use encryption type to mapping to security type.
* This function demonstrate how to get more richful information of security type.
*/
switch (thisType) {
case SECURITY_OPEN:
Serial.print("Open");
break;
case SECURITY_WEP_PSK:
Serial.print("WEP");
break;
case SECURITY_WPA_TKIP_PSK:
Serial.print("WPA TKIP");
break;
case SECURITY_WPA_AES_PSK:
Serial.print("WPA AES");
break;
case SECURITY_WPA2_AES_PSK:
Serial.print("WPA2 AES");
break;
case SECURITY_WPA2_TKIP_PSK:
Serial.print("WPA2 TKIP");
break;
case SECURITY_WPA2_MIXED_PSK:
Serial.print("WPA2 Mixed");
break;
case SECURITY_WPA_WPA2_MIXED:
Serial.print("WPA/WPA2 AES");
break;
```

```
}
}
void printEncryptionType(int thisType) {
// read the encryption type and print out the name:
switch (thisType) {
case ENC_TYPE_WEP:
Serial.println("WEP");
break;
case ENC_TYPE_TKIP:
Serial.println("WPA");
break;
case ENC_TYPE_CCMP:
Serial.println("WPA2");
break;
case ENC_TYPE_NONE:
Serial.println("None");
break;
case ENC_TYPE_AUTO:
Serial.println("Auto");
break;
}
}
```

Notes and Warnings
NA

WiFiClass::SSID

Description
Return the SSID discovered during the network scan

Syntax
char* WiFiClass::SSID(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
The function returns ssid string of the specified item on the networks scanned a list.

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionType

Description
Return the encryption type of the networks discovered during the scanNetworks

Syntax
uint8_t WiFiClass::encryptionType(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
encryption type (enum wl_enc_type) of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::encryptionTypeEx

Description
Return the security type and encryption type of the networks discovered during the scanNetworks

Syntax
uint32_t WiFiClass::encryptionTypeEx(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
security and encryption type of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::RSSI

Description
Return the RSSI of the networks discovered during the scanNetworks

Syntax
int32_t WiFiClass::RSSI(uint8_t networkItem)

Parameters
networkItem: specify from which network item want to get the information

Returns
signed value of RSSI of the specified item on the networks scanned a list

Example Code
Example: ScanNetworks
This example prints the Wifi shield's MAC address, and scans for available Wifi networks using the Wifi shield. Every ten seconds, it scans again. It doesn't connect to any network, so no encryption scheme is specified. The details of the code can be found in the previous section of WiFiClass:: scanNetworks.

Notes and Warnings
NA

WiFiClass::status

Description
Return Connection status

Syntax
uint8_t WiFiClass::status()

Parameters
The function requires no input parameter.

Returns
The function returns one of the values defined in wl_status_t as an unsigned integer.

Example Code
Example: ConnectWithWPA
This example demos how to connect to an unencrypted WiFi network, and prints the MAC address of the Wifi shield, the IP address obtained, and other network details. The details of the code can be found in the previous section of WiFiClass:: firmwareVersion.

Notes and Warnings
NA

WiFiClass::hostByName

Description
Resolve the given hostname to an IP address

Syntax
int WiFiClass::hostByName(const char* aHostname, IPAddress& aResult)

Parameters
aHostname: Name to be resolved
aResult: IPAddress structure to store the returned IP address

Returns
The function returns “1” if aIPAddrString was successfully converted to an IP address,else otherwise, it will return as an error code.

Example Code

NA

Notes and Warnings
NA

WiFiClass::apbegin

Description
Start AP mode

Syntax
int WiFiClass::apbegin(char* ssid, char* channel)
int WiFiClass::apbegin(char* ssid, char* password, char* channel)

Parameters
ssid: SSID of the AP network
channel: AP's channel, default 1
password: AP's password

Returns
The function will return the WiFi status.

Example Code
Example: WiFiAPMode

#include

char ssid[] = "yourNetwork"; //Set the AP's SSID

char pass[] = "Password"; //Set the AP's password

char channel[] = "1"; //Set the AP's channel

int status = WL_IDLE_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL_NO_SHIELD) {

```
Serial.println("WiFi shield not present");
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to start AP:
while (status != WL_CONNECTED) {
Serial.print("Attempting to start AP with SSID: ");
Serial.println(ssid);
status = WiFi.apbegin(ssid, pass, channel);
delay(10000);
}
//AP MODE already started:
Serial.println("AP mode already started");
Serial.println();
printWifiData();
printCurrentNet();
}
void loop() {
// check the network connection once every 10 seconds:
delay(10000);
printCurrentNet();
}
void printWifiData() {
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your subnet mask:
IPAddress subnet = WiFi.subnetMask();
Serial.print("NetMask: ");
Serial.println(subnet);
// print your gateway address:
IPAddress gateway = WiFi.gatewayIP();
```

```
Serial.print("Gateway: ");
Serial.println(gateway);
Serial.println();
}
void printCurrentNet() {
// print the SSID of the AP:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print the MAC address of AP:
byte bssid[6];
WiFi.BSSID(bssid);
Serial.print("BSSID: ");
Serial.print(bssid[0], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.println(bssid[5], HEX);
// print the encryption type:
byte encryption = WiFi.encryptionType();
Serial.print("Encryption Type:");
Serial.println(encryption, HEX);
Serial.println();
}
```

Notes and Warnings
NA

WiFiClass::disablePowerSave

Description
Disable power-saving mode

Syntax
int WiFiClass::disablePowerSave()

Parameters
The function requires no input parameter.

Returns
1 if disable success, 0 if failed

Example Code
NA

Notes and Warnings
NA

Class WiFiClient

WiFiClient Class

Description
Defines a class of WiFi Client implementation for Ameba.

Syntax
class WiFiClient

Members

Public Constructors	
WiFiClient::WiFiClient	Constructs a WiFiClient instance that connects to the specified IP address and port.
Public Methods	
WiFiClient::connect	Connect to the IP address and port
WiFiClient::write	Write a single byte into the packet
WiFiClient::available	Number of bytes remaining in the current packet
WiFiClient::read	Read a single byte from the current packet
WiFiClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiClient:: flush	Finish reading the current packet
WiFiClient::stop	Stop client connection
WiFiClient::connected	Check if client is connected, return 1 if connected, 0 if not
WiFiClient::setRecvTimeout	Set receiving timeout

WiFiClient::WiFiClient

Description

Constructs a WiFiClient instance that connects to the specified IP address and port.

Syntax
WiFiClient::WiFiClient()
WiFiClient::WiFiClient(uint8_t sock)

Parameters
sock: socket state, default -1.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient

```
#include <WiFi.h>
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "password"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
//IPAddress server(64,233,189,94); // numeric IP for Google (no DNS)
char server[] = "www.google.com"; // name address for Google (using DNS)
WiFiClient client;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
;
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
```

```
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 80)) {
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=ameba HTTP/1.1");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
```

```
// do nothing forevermore:
while (true);
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiClient::connect

Description
Connect to the IP address and port

Syntax
int WiFiClient::connect(IPAddress ip, uint16_t port)
int WiFiClient::connect(const char *host, uint16_t port)

Parameters
ip: IP address
host: Host name
port: the port to listen on

Returns

Returns “1”: if successful
Returns “0”: if failed

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiClient::write(uint8_t byte)
size_t WiFiClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message
size: the size of the buffer

Returns
The function returns single byte into the packet or returns bytes size from buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiClient::available(void)

Parameters
The function requires no input parameter.

Returns
• Function returns the number of bytes available in the current packet
Function returns 0: if no data available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiClient::read()
int WiFiClient::read(unsigned char* buf, size_t size)
int WiFiClient::read(char *buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiClient::flush

Description
Finish reading the current packet

Syntax
void WiFiClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiClient::stop

Description
Disconnect from the server. Stop client connection

Syntax
void WiFiClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::connected

Description
Check if client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiWebClient
The details of the example are explained in the previous section of WiFiClient:: WiFiClient.

Notes and Warnings
NA

WiFiClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
0

Example Code
NA

Notes and Warnings
NA

Class WiFiServer

WiFiServer Class

Description
Defines a class of WiFi server implementation for Ameba.

Syntax
class WiFiServer

Members

Public Constructors	
WiFiServer::WiFiServer	Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port
Public Methods	
WiFiServer::available	Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop()
WiFiServer::begin	Tells the server to begin listening for incoming connections
WiFiServer::write	Write data to all the clients connected to a server

WiFiServer::WiFiServer

Description
Constructs a WiFiServer object and creates a server that listens for incoming connections on the specified port.

Syntax
WiFiServer::WiFiServer(uint16_t port)

Parameters
port: The port number being connected to.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi

#include <WiFi.h>

char ssid[] = “yourNetwork”; // your network SSID (name)

char pass[] = “secretPassword”; // your network password

int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;

WiFiServer server(5000);

void setup() {

Serial.begin(9600); // initialize serial communication

pinMode(9, OUTPUT); // set the LED pin mode

```
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
while (true); // don't continue
}
String fv = WiFi.firmwareVersion();
if ( fv != "1.1.0" )
Serial.println("Please upgrade the firmware");
// attempt to connect to Wifi network:
while ( status != WL_CONNECTED) {
Serial.print("Attempting to connect to Network named: ");
Serial.println(ssid); // print the network name (SSID);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid, pass);
// wait 10 seconds for connection:
delay(10000);
}
server.begin(); // start the tcp server on port 5000
printWifiStatus(); // you're connected now, so print out the status
}
char buffer[256];
void loop() {
WiFiClient client = server.available();
while (client.connected()) {
memset(buffer, 0, 256);
int n = client.read((uint8_t*)(&buffer[0]), sizeof(buffer));
if (n > 0) {
for (int i=0; i<n; i++) {
Serial.print(buffer[i]);
}
n = client.write(buffer, n);
if (n <= 0) break;
}
}
client.stop();
}
```

```
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiServer::available

Description
Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; you can close it by calling the client.stop().

Syntax
WiFiClient WiFiServer::available(uint8_t* status)

Parameters
status: WiFi availability status

Returns
A Client object; if no Client has data available for reading, this object will evaluate to false in an if-statement

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::begin

Description
Tells the server to begin listening for incoming connections

Syntax
void WiFiServer::begin(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

WiFiServer::write

Description
Write data to all the clients connected to a server

Syntax
size_t WiFiServer::write(uint8_t b)
size_t WiFiServer::write(const uint8_t *buf, size_t size)

Parameters
b: byte to be written
buf: data buffer
size: Size of the data in the buffer

Returns
The function returns the number of bytes written. It is not necessary to read this.

Example Code
Example: SimpleServerWiFi
Details of the code can be found in the previous section of WiFiServer:: WiFiServer.

Notes and Warnings
NA

Class WiFiSSLClient

WiFiSSLClient Class

Description
Defines a class of WiFi Secure Socket Layer Client implementation for Ameba.

Syntax
class WiFiSSLClient

Members

Public Constructors	
WiFiSSLClient::WiFiSSLClient	Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port
Public Methods	
WiFiSSLClient::connect	Connect to the IP address and port
WiFiSSLClient::write	Write a single byte into the packet
WiFiSSLClient::available	Number of bytes remaining in the current packet
WiFiSSLClient::read	Read a single byte from the current packet
WiFiSSLClient:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiSSLClient:: flush	Finish reading the current packet
WiFiSSLClient::stop	Stop SSL client connection
WiFiSSLClient::connected	Check if SSL client is connected, return 1 if connected, 0 if not
WiFiSSLClient:: setRootCA	Set Root CA for authentication
WiFiSSLClient:: setClientCertificate	Set certificate of the client
WiFiSSLClient::setRecvTimeout	Set receiving timeout
WiFiSSLClient::setPreSharedKey	Set the pre shared key (PSK) to use for authentication

WiFiSSLClient::WiFiSSLClient

Description
Constructs a WiFiSSLClient instance that always connects in SSL to the specified IP address and port.

Syntax
WiFiSSLClient::WiFiSSLClient(void)
WiFiSSLClient::WiFiSSLClient(uint8_t sock)

Parameters
sock: socket state, default -1

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient

```
#include
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword";// your network password (use for WPA, or WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
char server[] = "www.google.com"; // name address for Google (using DNS)
//unsigned char test_client_key[] = ""; //For the usage of verifying client
//unsigned char test_client_cert[] = ""; //For the usage of verifying client
//unsigned char test_ca_cert[] = ""; //For the usage of verifying server
WiFiSSLClient client;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
```

```
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
// if you get a connection, report back via serial:
if (client.connect(server, 443)) { //client.connect(server, 443, test_ca_cert, test_client_cert, test_client_key)
Serial.println("connected to server");
// Make a HTTP request:
client.println("GET /search?q=realtek HTTP/1.0");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
else
Serial.println("connected to server failed");
}
void loop() {
// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
char c = client.read();
Serial.write(c);
}
// if the server's disconnected, stop the client:
if (!client.connected()) {
Serial.println();
Serial.println("disconnecting from server.");
client.stop();
// do nothing forevermore:
while (true);
```

```
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[0], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.println(mac[5], HEX);
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
NA

WiFiSSLClient::connect

Description
Connect to the IP address and port.

Syntax
int WiFiSSLClient::connect(IPAddress ip, uint16_t port)
int WiFiSSLClient::connect(const char *host, uint16_t port)
int WiFiSSLClient::connect(const char* host, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)
int WiFiSSLClient::connect(IPAddress ip, uint16_t port, unsigned char* rootCABuff, unsigned char* cli_cert, unsigned char* cli_key)

Parameters
ip: IP address
host: Host name
port: the port to listen on
rootCABuff: buffer that store root CA
cli_cert: buffer that store client certificate
cli_key buffer that store client key pair

Returns
1: if successful
0: if failed

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::write

Description
Write a single byte into the packet

Syntax
size_t WiFiSSLClient::write(uint8_t byte)
size_t WiFiSSLClient::write(const uint8_t *buf, size_t size)

Parameters
byte: the outgoing byte
buf: the outgoing message

size: the size of the buffer

Returns
The function returns single -byte into the packet or turns bytes size from the buffer into the packet.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::available

Description
Number of bytes remaining in the current packet

Syntax
int WiFiSSLClient::available(void)

Parameters
The function requires no input parameter.

Returns
The function returns the number of bytes available in the current packet; else return "0:" if no data available.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::read

Description
Read a single byte from the current packet

Syntax
int WiFiSSLClient::read()

int WiFiSSLClient::read(unsigned char* buf, size_t size)

Parameters
buf: buffer to hold incoming packets (char*)
size: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::peek

Description
Return the next byte from the current packet without moving on to the next byte.

Syntax
int WiFiSSLClient::peek(void)

Parameters
The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::flush

Description
Finish reading the current packet

Syntax
void WiFiSSLClient::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::stop

Description
Disconnect from the server. Stop SSL client connection

Syntax
void WiFiSSLClient::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::connected

Description
Check if SSL client is connected, return 1 if connected, 0 if not.

Syntax
uint8_t WiFiSSLClient::connected(void)

Parameters
The function requires no input parameter.

Returns
The function returns “1” if connected, returns “0” if not connected.

Example Code
Example: WiFiSSLClient
Details of the code can be found in the previous section of WiFiSSLClient:: WiFiSSLClient.

Notes and Warnings
NA

WiFiSSLClient::setRootCA

Description
Set Root CA for authentication

Syntax
void WiFiSSLClient::setRootCA(unsigned char *rootCA)

Parameters
rootCA: a string of rootCA

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings

NA

WiFiSSLClient::setClientCertificate

Description
Set certificate of client

Syntax
void WiFiSSLClient::setClientCertificate(unsigned char *client_ca, unsigned char *private_key)

Parameters
client_ca: Client certificate
private_key: client's private key pair

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setRecvTimeout

Description
Set receiving timeout

Syntax
int WiFiSSLClient::setRecvTimeout(int timeout)

Parameters
timeout: timeout in seconds

Returns
The function returns "0".

Example Code
NA

Notes and Warnings
NA

WiFiSSLClient::setPreSharedKey

Description
Set the pre shared key (PSK) to use for authentication

Syntax
void WiFiSSLClient::setPreSharedKey(unsigned char *pskIdent, unsigned char *psKey)

Parameters
pskIdent: identity for PSK
psKey: Pre shared key

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
Do not set a root CA and client certificate if PSK should be used for authentication. If root CA, client certificate and PSK are all set, certificate based authentication will be used.

Class WiFiUdp

WiFiUDP Class

Description
Defines a class of WiFi UDP implementation for Ameba.

Syntax
class WiFiUDP

Members

Public Constructors	
WiFiUDP::WiFiUDP	Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages
Public Methods	
WiFiUDP:: begin	initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use
WiFiUDP:: stop	Finish with the UDP socket
WiFiUDP:: beginPacket	Start building up a packet to send to the remote host-specific in IP and port
WiFiUDP:: endPacket	Finish off this packet and send it
WiFiUDP:: write	Write a single byte into the packet
WiFiUDP:: writeImmediately	Send packet immediately from the buffer
WiFiUDP:: parsePacket	Start processing the next available incoming packet
WiFiUDP::available	Number of bytes remaining in the current packet
WiFiUDP::read	Read a single byte from the current packet
WiFiUDP:: peek	Return the next byte from the current packet without moving on to the next byte
WiFiUDP:: flush	Finish reading the current packet
WiFiUDP:: remoteIP	Return the IP address of the host who sent the current incoming packet
WiFiUDP:: remotePort	Return the port of the host who sent the current incoming packet
WiFiUDP:: setRecvTimeout	Set receiving timeout

WiFiUDP::WiFiUDP

Description

Constructs a WiFiUDP instance of the WiFi UDP class that can send and receive UDP messages.

Syntax

WiFiUDP::WiFiUDP(void)

Parameters

The function requires no input parameter.

Returns

The function returns nothing.

Example Code

Example: WiFiUdpSendReceiveString

This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort.

#include <WiFi.h>

#include <WiFiUdp.h>

```
int status = WL_IDLE_STATUS;
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)
unsigned int localPort = 2390; // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back
WiFiUDP Udp;
void setup() {
//Initialize serial and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
// check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
Serial.println("WiFi shield not present");
// don't continue:
while (true);
}
String fv = WiFi.firmwareVersion();
if (fv != "1.1.0") {
Serial.println("Please upgrade the firmware");
}
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
// Connect to WPA/WPA2 network. Change this line if using open or WEP network:
status = WiFi.begin(ssid,pass);
// wait 10 seconds for connection:
delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
Serial.println("nStarting connection to server…");
```

```
// if you get a connection, report back via serial:
Udp.begin(localPort);
}
void loop() {
// if there's data available, read a packet
int packetSize = Udp.parsePacket();
if (packetSize) {
Serial.print("Received packet of size ");
Serial.println(packetSize);
Serial.print("From ");
IPAddress remoteIp = Udp.remoteIP();
Serial.print(remoteIp);
Serial.print(", port ");
Serial.println(Udp.remotePort());
// read the packet into packetBufffer
int len = Udp.read(packetBuffer, 255);
if (len > 0) {
packetBuffer[len] = 0;
}
Serial.println("Contents:");
Serial.println(packetBuffer);
// send a reply, to the IP address and port that sent us the packet we received
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
Udp.write(ReplyBuffer);
Udp.endPacket();
}
}
void printWifiStatus() {
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
// print the received signal strength:
```

```
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
}
```

Notes and Warnings
This constructor does not take in any parameter, thus use another method to set up the IP address and port number.

WiFiUDP::begin

Description
Initialize, start listening on the specified port. Returns 1 if successful, 0 if there are no sockets available to use.

Syntax
uint8_t WiFiUDP::begin(uint16_t port)

Parameters
port: the local port to listen on

Returns
1: if successful
0: if there are no sockets available to use

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::stop

Description
Disconnect from the server. Release any resource being used during the UDP session.

Syntax
void WiFiUDP::stop(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::beginPacket

Description
Start building up a packet to send to the remote host-specific in IP and port.

Syntax
int WiFiUDP::beginPacket(const char *host, uint16_t port)
int WiFiUDP::beginPacket(IPAddress ip, uint16_t port)

Parameters
host: hostname
port: port number
ip: IP address

Returns
1: if successful
0: if there was a problem with the supplied IP address or port

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::endPacket

Description
Finish off this packet and send it

Syntax
int WiFiUDP::endPacket(void)

Parameters
The function requires no input parameter.

Returns
1: if the packet was sent successfully
0: if there was an error

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::write

Description
Write a single byte into the packet.

Syntax
size_t WiFiUDP::write(uint8_t byte)
size_t WiFiUDP::write(const uint8_t *buffer, size_t size)

Parameters
byte: the outgoing byte

buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::writeImmediately

Description
Send packet immediately from the buffer

Syntax
size_t WiFiUDP::writeImmediately(const uint8_t *buffer, size_t size)

Parameters
buffer: the outgoing message
size: the size of the buffer

Returns
single-byte into the packet
bytes size from the buffer into the packet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::parsePacket

Description
Start processing the next available incoming packet

Syntax
int WiFiUDP::parsePacket(void)

Parameters
The function requires no input parameter.

Returns
The function returns the size of the packet in bytes or returns "0:" if no packets are available.

Example Code
Example: WiFiUdpSendReceiveString

Notes and Warnings
NA

WiFiUDP::available

Description
Number of bytes remaining in the current packet.

Syntax
int WiFiUDP::available(void)

Parameters
The function requires no input parameter.

Returns
the number of bytes available in the current packet
0: if parsePacket hasn't been called yet

Example Code
NA

Notes and Warnings
NA

WiFiUDP::read

Description
Read a single byte from the current packet

Syntax
int WiFiUDP::read()
int WiFiUDP::read(unsigned char* buffer, size_t len)

Parameters
buffer: buffer to hold incoming packets (char*)
len: maximum size of the buffer (int)

Returns
size: the size of the buffer
-1: if no buffer is available

Example Code
Example: WiFiUdpSendReceiveString
his example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::peek

Description
Return the next byte from the current packet without moving on to the next byte

Syntax
int WiFiUDP::peek(void)

Parameters

The function requires no input parameter.

Returns
b: the next byte or character
-1: if none is available

Example Code
NA

Notes and Warnings
NA

WiFiUDP::flush

Description
Finish reading the current packet

Syntax
void WiFiUDP::flush(void)

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

WiFiUDP::remoteIP

Description
Return the IP address of the host who sent the current incoming packet

Syntax
IPAddress WiFiUDP::remoteIP(void)

Parameters
The function requires no input parameter.

Returns
IP address connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::remotePort

Description
Return the port of the host who sent the current incoming packet

Syntax
uint16_t WiFiUDP::remotePort(void)

Parameters
The function requires no input parameter.

Returns
The remote port connecting to

Example Code
Example: WiFiUdpSendReceiveString
This example demonstrates WiFi UDP send and receive string. This sketch waits for a UDP packet on a local port using a WiFi shield. When a packet is received an Acknowledge packet is sent to the client on port remotePort. The detail of the code can be found in WiFiUDP:: WiFiUDP.

Notes and Warnings
NA

WiFiUDP::setRecvTimeout

Description
Set receiving timeout

Syntax
void WiFiUDP::setRecvTimeout(int timeout)

Parameters
timeout in seconds

Returns
The function returns nothing.

Example Code
NA

Notes and Warnings
NA

Readme

The Ameba WiFi related APIs and examples are works based on Arduino WiFI shield libraries (https://www.arduino.cc/en/Reference/WiFi).
These include,

- `WiFi.cpp`
- `WiFi.h`
- `WiFiServer.cpp`
- `WiFiServer.h`
- `WiFiUdp.cpp`
- `WiFiUdp.h`

Wire

Class TwoWire

TwoWire Class

Description
Defines a class of I2C API

Syntax
class TwoWire

Members

Public Constructors	
TwoWire::TwoWire	Constructs a TwoWire object
Public Methods	
TwoWire::begin	Initialize I2C master/slave
TwoWire::setClock	Set I2C frequency
TwoWire::beginTransmission	Begin I2C transmission
TwoWire::endTransmission	End I2C transmission
TwoWire::requestFrom	Set I2C requestFrom
TwoWire::write	Write data to I2C
TwoWire::available	Check if I2C is available
TwoWire::read	Read data from I2C
TwoWire::peek	Read peek from I2C
TwoWire::flush	Do nothing, use, and transmission(..) to force data transfer
TwoWire::onReceive	Callback function when I2C on receive
TwoWire::onRequest	Callback function when I2C on request

TwoWire::TwoWire

Description
Constructs a TwoWire object.

Syntax
TwoWire::TwoWire (uint32_t dwSDAPin, uint32_t dwSCLPin);

Parameters
dwSDAPin: The Arduino PIN to be set as an SDA pin.
dwSCLPin: The Arduino PIN to be set as an SCL pin.

Returns
The function returns nothing.

Example Code

Example: MasterWriter

This example demonstrates the use of the wire library writes to an I2C/TWI slave device.

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
}
byte x = 0;
void loop() {
Wire.beginTransmission(8); // transmit to device #8
Wire.write("x is "); // sends five bytes
Wire.write(x); // sends one byte
Wire.endTransmission(); // stop transmitting
x++;
delay(500);
}
```

Example: MasterReader

```
#include <Wire.h>
void setup() {
Wire.begin(); // join i2c bus (address optional for master)
Serial.begin(9600); // start serial for output
}
void loop() {
Wire.requestFrom(8, 6); // request 6 bytes from slave device #8
while (Wire.available()) { // slave may send less than requested
char c = Wire.read(); // receive a byte as character
Serial.print(c); // print the character
}
delay(500);
}
```

This example demonstrates the use of the wire library reads data from an I2C/TWI slave device.

Notes and Warnings

Include "Wire.h" to use the class function.

TwoWire::begin

Description
Initialize I2C master/slave.

Syntax
void TwoWire::begin (void);
void TwoWire::begin (uint8_t address = 0);
void TwoWire::begin (int address);

Parameters
void: Set the I2C master mode.
address: Set the I2C master mode with slave address value.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::setClock

Description
Set I2C frequency.

Syntax
void TwoWire::setClock(uint32_t frequency);

Parameters
frequency: The frequency values.

Returns
The function returns nothing.

Example Code

Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::beginTransmission

Description
Begin I2C transmission.

Syntax
void TwoWire::beginTransmission (uint8_t address);
void TwoWire::beginTransmission (int address);

Parameters
address: The transmission address.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::endTransmission

Description
End I2C transmission. Originally, ‘endTransmission’ was an f(void) function. It has been modified to take one parameter indicating whether or not a STOP should be performed on the bus. Calling endTransmission(false) allows a sketch to perform a repeated start.

WARNING: Nothing in the library keeps track of whether the bus tenure has been properly ended with a STOP. It is very possible to leave the bus in a hung state if no call to endTransmission(true) is made. Some I2C devices will behave oddly if they do not see a STOP.

If the input parameter is void, this provides backward compatibility with the original definition, and expected behavior, of endTransmission.

Syntax
uint8_t TwoWire::endTransmission (uint8_t sendStop);
uint8_t TwoWire::endTransmission (void);

Parameters
sendStop: True to end the transmission

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::requestFrom

Description
Set I2C requestFrom.

Syntax
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity, uint8_t sendStop);
uint8_t TwoWire::requestFrom (uint8_t address, uint8_t quantity);
uint8_t TwoWire::requestFrom(int address, int quantity);
uint8_t TwoWire::requestFrom (int address, int quantity, int sendStop);

Parameters
address: I2C read address.
quantity: I2C read quantity.
sendStop: True to end the transmission.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::write

Description
Write data to I2C.

Syntax
size_t TwoWire::write (uint8_t data);
size_t TwoWire::write (const uint8_t *data, size_t quantity);

Parameters
data: The data to be transmitted.
quantity: The quantity of data.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::available

Description
Check if I2C is available.

Syntax
int TwoWire::available (void);

Parameters
The function requires no input parameter.

Returns
Return 0 if successful, else error.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::read

Description
Read data from I2C

Syntax
int TwoWire::read (void);

Parameters
The function requires no input parameter.

Returns
The read data from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::peek

Description
Read peek from I2C.

Syntax
int TwoWire::peek (void);

Parameters
The function requires no input parameter.

Returns
The peek data read from the receive buffer.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::flush

Description
Do nothing, use endTransmission(..) to force data transfer.

Syntax
void TwoWire::flush (void);

Parameters
The function requires no input parameter.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter

Notes and Warnings
Include “Wire.h” in order to use the class function.

TwoWire::onReceive

Description
Callback function when I2C on receive.

Syntax
void TwoWire::onReceive (void(*function)(int));

Parameters
function: The callback function.

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

TwoWire::onRequest

Description
Callback function when I2C on request.

Syntax
void TwoWire::onRequest (void(*function)(void));

Parameters
function: The callback function

Returns
The function returns nothing.

Example Code
Example: MasterReader; MasterWriter
The details of the code can be found in the previous section of TwoWire:: TwoWire.

Notes and Warnings
Include “Wire.h” to use the class function.

Wire_Readme

The Ameba LCD related api and example are works based on "New LiquidCrystal library" (https://bitbucket.org/fmalpartida/new-liquidcrystal/).

These include,

LCD.h
LCD.cpp
I2CIO.h
I2CIO.cpp
LiquidCrystal_I2C.h
LiquidCrystal_I2C.cpp
examples/LcdHelloWorld/LcdHelloWorld.ino

These files inherit the licence of "New LiquidCrystal Library" which are under a Creative Commons Attribution-ShareAlike 3.0 Unported License. CC BY-SA 3.0.

1.3.4 Resources

Links

- AmebaD Arduino Github
- Arduino Website

1.3.5 Support

FAQ

Where to buy Ameba RTL8722DM Board?

Refer to Purchase link.

Which Bluetooth standards are supported by RTL8722CSM/RTL8722DM?

Both boards support BLE 5.0. Classic Bluetooth (BR/EDR) is not supported.

Which BLE roles are supported?

RTL8722CSM/RTL8722DM can operate as either a BLE Central or BLE Peripheral device.

Are all pins on RTL8722CSM/RTL8722DM usable?

No, those marked `NC` are not connected to any pin and thus unusable.

Is XIP (execute in place) supported on RTL8722CSM/RTL8722DM?

Yes, it is supported.

Does RTL8722CSM support 5G WiFi?

No. Only RTL8722DM supports dual band 2.4G + 5G WiFi. RTL8722CSM only supports single band 2.4G WiFi.

How to enter the download mode?

Press and hold the UART DOWNLOAD button. Then Press the RESET button and release both UART DOWNLOAD and RESET buttons.

Trouble shooting

RTL8722CSM/RTL8722DM cannot be found as a Bluetooth device.

Please make sure the antenna is connected properly. Check your code for the correct Bluetooth configurations.

My code is not behaving as I expected.

Try to debug your program using `printf()` and `Serial.print()` statements. If the issue persists, you can ask for help at Forums

Why is there no output on my serial terminal after connecting to RTL8722CSM/RTL8722DM UART?

RTL8722CSM/RTL8722DM is by default configured at 115200 baudrate, please check if your serial terminal is configured to 115200.

My program is not being downloaded into RTL8722CSM/RTL8722DM?

Please follow the procedure for the correct downloading?

1. Enter the download mode. The on-board Green LED will blink when entered download mode.
2. When downloading the image into board the on-board Red LED will blink
3. After a successful download, you will see log like this “All images sent successfully”.

Sometimes WiFi signal is weak?

The default antenna for RTL8722CSM/RTL8722DM uses the I-Pex Connector. Please change/connect the I-Pex Connector antenna.

Why is my board not powering up?

Please make sure the connector J38 beside resistor R43 is connected. The connector is used to link the power to IC.

If you have driver issue to connect board to your computer?

Please go to https://ftdichip.com/drivers/ for USB driver.

CHAPTER

TWO

MICROPYTHON SDK

Welcome to Ameba MicroPython online documentation.

2.1 Getting Started

2.1.1 Ameba MicroPython: Getting Started with RTL8722

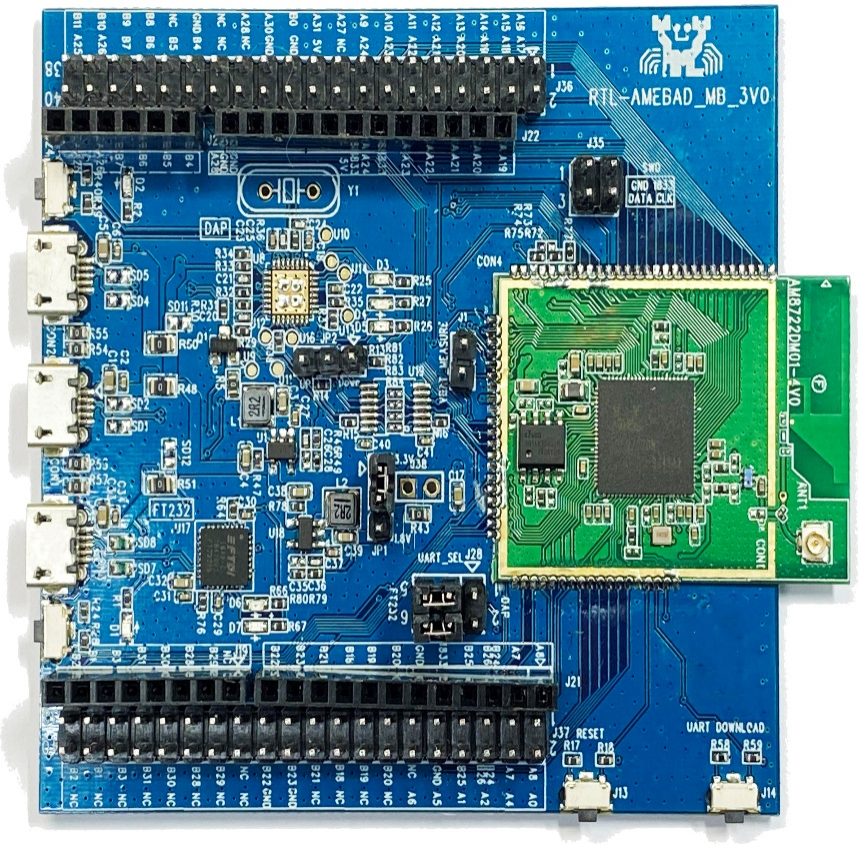
Required Environment

AmebaD RTL8722CSM/RTL8722DM MicroPython SDK currently supports Windows 10 and Linux operating systems.

Introduction to AmebaD RTL8722CSM/RTL8722DM

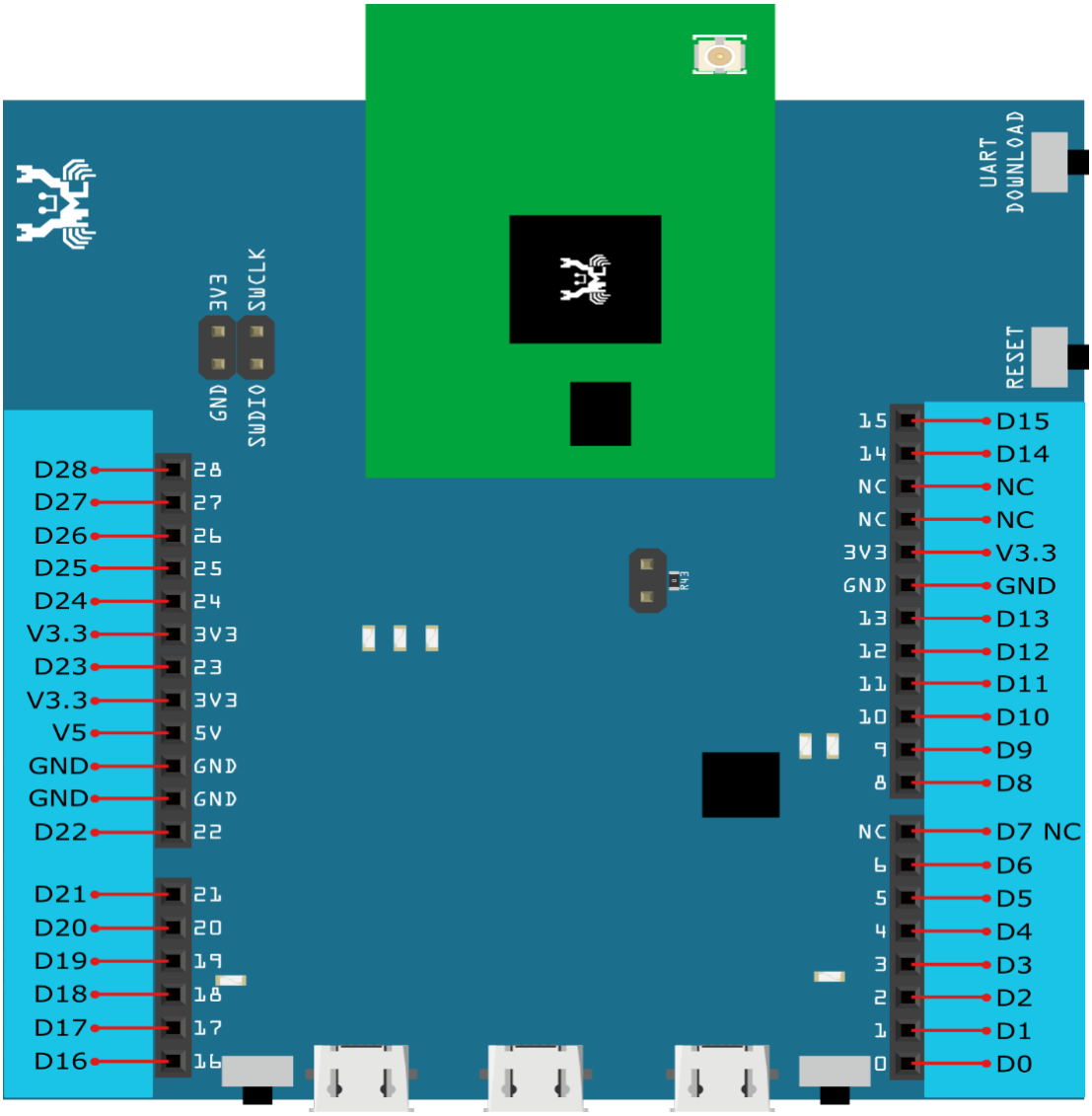
Ameba is an easy-to-program platform for developing all kind of IoT applications. AmebaD is equipped with various peripheral interfaces, including WiFi, BLE, GPIO, I2C, UART, SPI, PWM, ADC and so on. Through these interfaces, AmebaD can connect with electronic components such as LED, switches, manometer, hygrometer, PM2.5 dust sensors, …etc.

The collected data can be uploaded via WiFi and be utilized by applications on smart devices to realize IoT implementation.

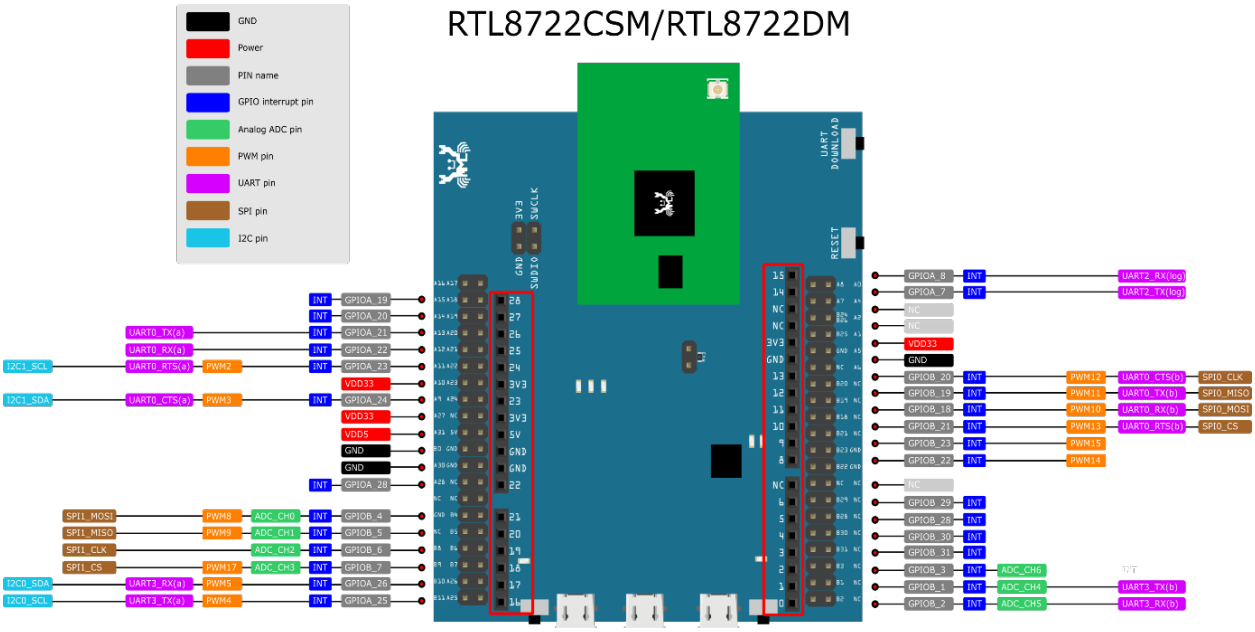


AmebaD and Arduino Uno have similar size, as shown in the above figure, and the pins on AmebaD are compatible with Arduino Uno.

AmebaD uses Micro USB to supply power, which is common in many smart devices.
Please refer to the following figure and table for the pin diagram and function of AmebaD.


UART DOWNLOAD
RESET
3V3
SWCLK
GND
SWDIO
D28 28
D27 27
D26 26
D25 25
D24 24
V3.3 3V3
D23 23
V3.3 3V3
V5 5V
GND GND
GND GND
D22 22
D21 21
D20 20
D19 19
D18 18
D17 17
D16 16
15 D15
14 D14
NC NC
NC NC
3V3 V3.3
GND GND
13 D13
12 D12
11 D11
10 D10
9 D9
8 D8
NC D7 NC
6 D6
5 D5
4 D4
3 D3
2 D2
1 D1
0 D0

	PIN name	GPIO	ADC	PWM	UART	SPI	I2C
D00	GPIOB_2	✓	ADC5		UART3_RX(b)		
D01	GPIOB_1	✓	ADC4		UART3_TX(b)		
D02	GPIOB_3	✓	ADC6				
D03	GPIOB_31	✓					
D04	GPIOB_30	✓					
D05	GPIOB_28	✓					
D06	GPIOB_29	✓					
D07	NC						
D08	GPIOB_22	✓		PWM14			
D09	GPIOB_23	✓		PWM15			
D10	GPIOB_21	✓		PWM13	UART0_RTS(b)	SPI0_CS	
D11	GPIOB_18	✓		PWM10	UART0_RX(b)	SPI0_MOSI	
D12	GPIOB_19	✓		PWM11	UART0_TX(b)	SPI0_MISO	
D13	GPIOB_20	✓		PWM12	UART0_CTS(b)	SPI0_CLK	
D14	GPIOA_7	✓			UART2_TX(log)		
D15	GPIOA_8	✓			UART2_RX(log)		
D16	GPIOA_25	✓		PWM4	UART3_RX(a)		I2C0_SCL
D17	GPIOA_26	✓		PWM5	UART3_TX(a)		I2C0_SDA
D18	GPIOB_7	✓	ADC3	PWM17		SPI1_CS	
D19	GPIOB_6	✓	ADC2			SPI1_CLK	
D20	GPIOB_5	✓	ADC1	PWM9		SPI1_MISO	
D21	GPIOB_4	✓	ADC0	PWM8		SPI1_MOSI	
D22	GPIOA_28	✓					
D23	GPIOA_24	✓		PWM3	UART0_CTS(a)		I2C1_SDA
D24	GPIOA_23	✓		PWM2	UART0_RTS(a)		I2C1_SCL
D25	GPIOA_22	✓			UART0_RX(a)		
D26	GPIOA_21	✓			UART0_TX(a)		
D27	GPIOA_20	✓					
D28	GPIOA_19	✓					



Note:

Not all sets of peripherals shown on the picture/table above are available on MicroPython, please refer to "*API Documents*" section for more information.

Introduction to RTL8722 MicroPython port

Background Information

REPL stands for Read-Evaluation-Print-Loop, it is an interactive prompt that you can use to access and control your microcontroller.

REPL has been equipped with other powerful features such as tab completion, line editing, auto-indentation, input history and more. It basically functions like the classic Python IDLE but running on microcontroller.

To use REPL, simply open any serial terminal software (most common ones are teraterm, putty etc.) on your PC and connect to your microcontroller's serial port, then set baudrate to `115200` before manually reset the board, then you will see >>> MicroPython prompt appear on the terminal. Now you may type in any Python script on REPL as long as it's support by MicroPython and your microcontroller's MicroPython port.

Most importantly, try to abuse "`help()`" function as much as possible to gain more information. For example, upon microcontroller power up and REPL shown, just type

```
>>> help()
```

You will see a help page giving you more details about this port; also if you type

```
>>> help(modules)
```

it will list out all available builtin modules that are at your disposal

Furthermore, if you want to learn more about a module, such as its API and CONSTANT available, simply type the following code and details of that module will be returned to you,

```
>>> help(the module of your interest)
```

Let's take Pin module (GPIO) as an example:

```
>>> help(Pin)

object <class 'Pin'> is of type type
id -- <function>
init -- <function>
value -- <function>
off -- <function>
on -- <function>
toggle -- <function>
board -- <class 'board'>
IN -- 0
OUT -- 1
PULL_NONE -- 0
PULL_UP -- 1
PULL_DOWN -- 2
```

REPL Hotkeys

- `Ctrl + d`

  Soft reboot MicroPython will perform software reboot, this is useful when your microcontroller is behaving abnormally. This will also run scripts in 'boot.py' once again. Note that this will only reset the MicroPython interpreter not the hardware, all your previously configured hardware will stay the way it is until you manually hard-reset the board.

- `Ctrl + e`

  Paste mode Paste mode allow you to perform pasting a large trunk of code into REPL at once without executing code line by line. This is useful when you have found a MicroPython library and wish to test it out immediately by copy and paste

- `Ctrl + b`

  Normal mode This hotkey will set REPL back to normal mode. This is useful if you are stuck in certain mode and can not get out.

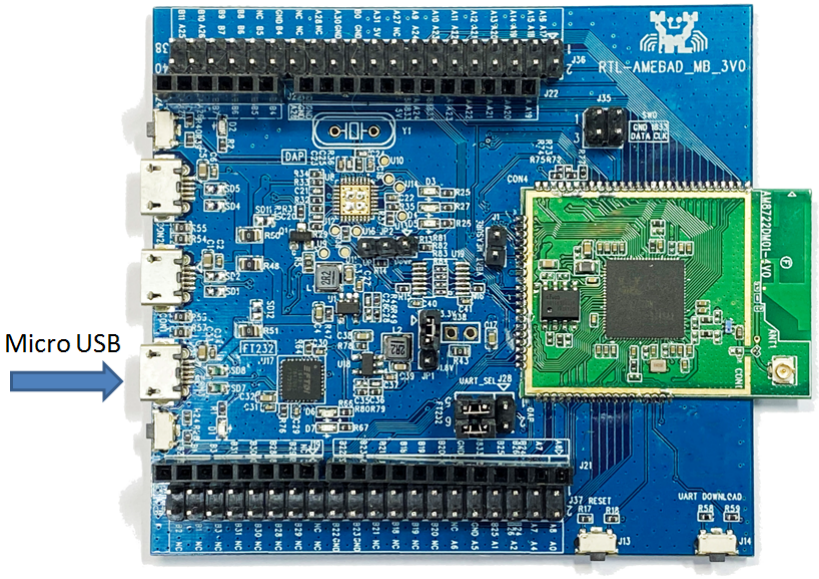
- `Ctrl + c`

  Quick cancel This hotkey help you to cancel any input and return a new line

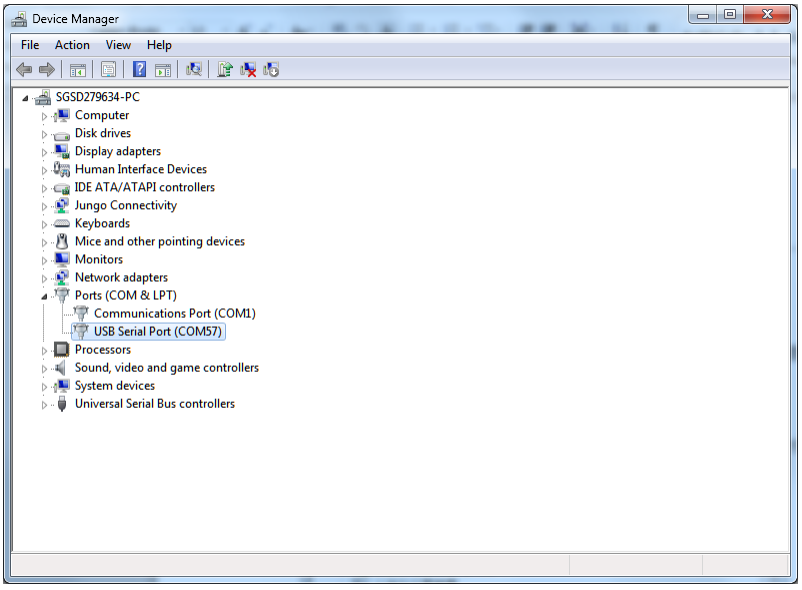
Setting up Development Environment

Step 1. Installing the Driver

First, connect AmebaD to the computer via Micro USB:



If this is the first time you connect AmebaD to your computer, the USB driver for AmebaD will be automatic installed. If you have driver issue of connect board to your computer please go to https://ftdichip.com/drivers/ for USB driver. You can check the COM port number in Device Manager of your computer:

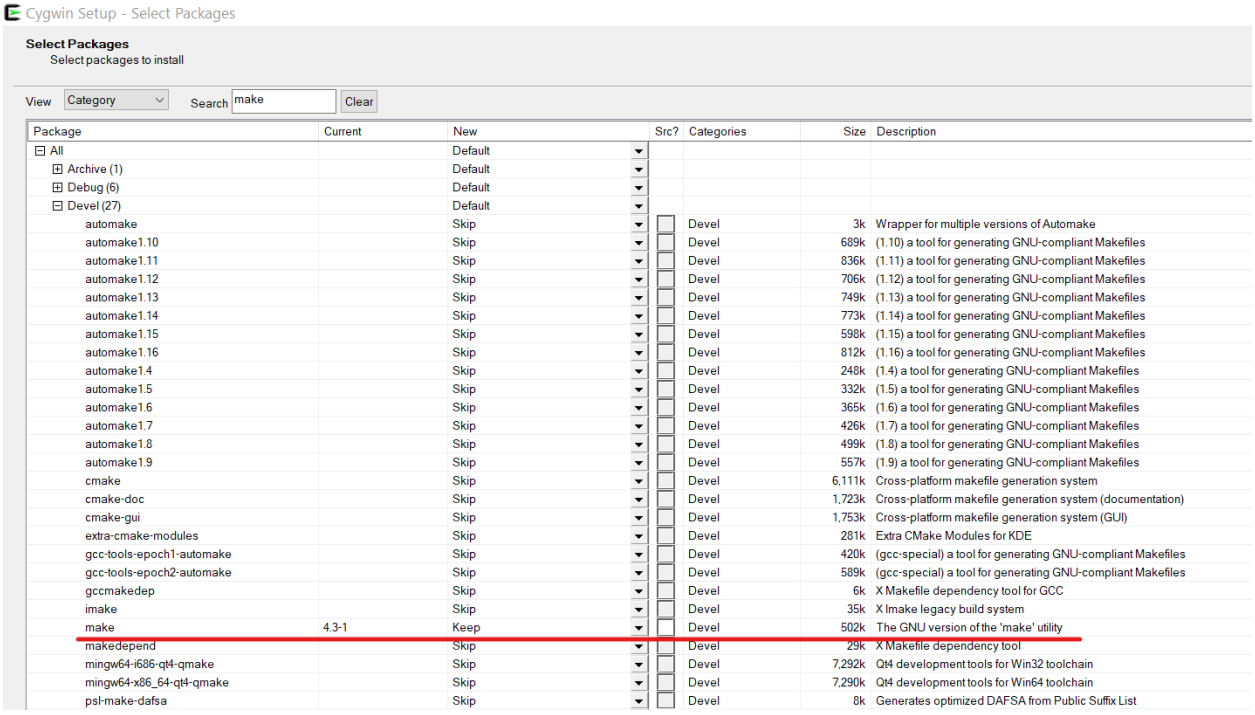


Step 2. Installing the necessary tools

On Windows

For Windows users, please install a serial terminal software to interact with MicroPython. The most common serial terminals are `Tera Term` and `Putty`, here we recommend using `Tera Term`, which can be downloaded from internet.

For advanced developer who wish to compile MicroPython firmware from scratch, then please be sure to install `Cygwin`, which is a Linux-like environment running on Windows system. When selecting the `Cygwin` installer, we recommend using the `Cygwin` 32-bit version. During `Cygwin` installation, installer will prompt user if wish to install other software, please make sure to select the GNU version of `make` from the `Devel` category (see picture below) and pick the latest edition.



Package	Current	New	Src?	Categories	Size	Description
All		Default				
Archive (1)		Default				
Debug (6)		Default				
Devel (27)		Default				
automake		Skip		Devel	3k	Wrapper for multiple versions of Automake
automake1.10		Skip		Devel	689k	(1.10) a tool for generating GNU-compliant Makefiles
automake1.11		Skip		Devel	836k	(1.11) a tool for generating GNU-compliant Makefiles
automake1.12		Skip		Devel	706k	(1.12) a tool for generating GNU-compliant Makefiles
automake1.13		Skip		Devel	749k	(1.13) a tool for generating GNU-compliant Makefiles
automake1.14		Skip		Devel	773k	(1.14) a tool for generating GNU-compliant Makefiles
automake1.15		Skip		Devel	598k	(1.15) a tool for generating GNU-compliant Makefiles
automake1.16		Skip		Devel	812k	(1.16) a tool for generating GNU-compliant Makefiles
automake1.4		Skip		Devel	248k	(1.4) a tool for generating GNU-compliant Makefiles
automake1.5		Skip		Devel	332k	(1.5) a tool for generating GNU-compliant Makefiles
automake1.6		Skip		Devel	365k	(1.6) a tool for generating GNU-compliant Makefiles
automake1.7		Skip		Devel	426k	(1.7) a tool for generating GNU-compliant Makefiles
automake1.8		Skip		Devel	499k	(1.8) a tool for generating GNU-compliant Makefiles
automake1.9		Skip		Devel	557k	(1.9) a tool for generating GNU-compliant Makefiles
cmake		Skip		Devel	6,111k	Cross-platform makefile generation system
cmake-doc		Skip		Devel	1,723k	Cross-platform makefile generation system (documentation)
cmake-gui		Skip		Devel	1,753k	Cross-platform makefile generation system (GUI)
extra-cmake-modules		Skip		Devel	281k	Extra CMake Modules for KDE
gcc-tools-epoch1-automake		Skip		Devel	420k	(gcc-special) a tool for generating GNU-compliant Makefiles
gcc-tools-epoch2-automake		Skip		Devel	589k	(gcc-special) a tool for generating GNU-compliant Makefiles
gccmakedep		Skip		Devel	6k	X Makefile dependency tool for GCC
imake		Skip		Devel	35k	X Imake legacy build system
make	4.3-1	Keep		Devel	502k	The GNU version of the 'make' utility
makedepend		Skip		Devel	29k	X Makefile dependency tool
mingw64-i686-qt4-qmake		Skip		Devel	7,292k	Qt4 development tools for Win32 toolchain
mingw64-x86_64-qt4-qmake		Skip		Devel	7,290k	Qt4 development tools for Win64 toolchain
psl-make-dafsa		Skip		Devel	8k	Generates optimized DAFSA from Public Suffix List

Also, Python3 is required during firmware compilation, so be sure to download the latest Python3 from its official website and have it added as environment variable when asked during installation.

On Linux

For Linux user, please install a serial terminal software of your choice using `apt-get` install command. Here we recommend using `picocom` for its lightweight.

For advanced developer interested in developing MicroPython module in C, please make sure the GNU make of at least version 3.82 or newer and Python3 are installed and can be found using terminal.

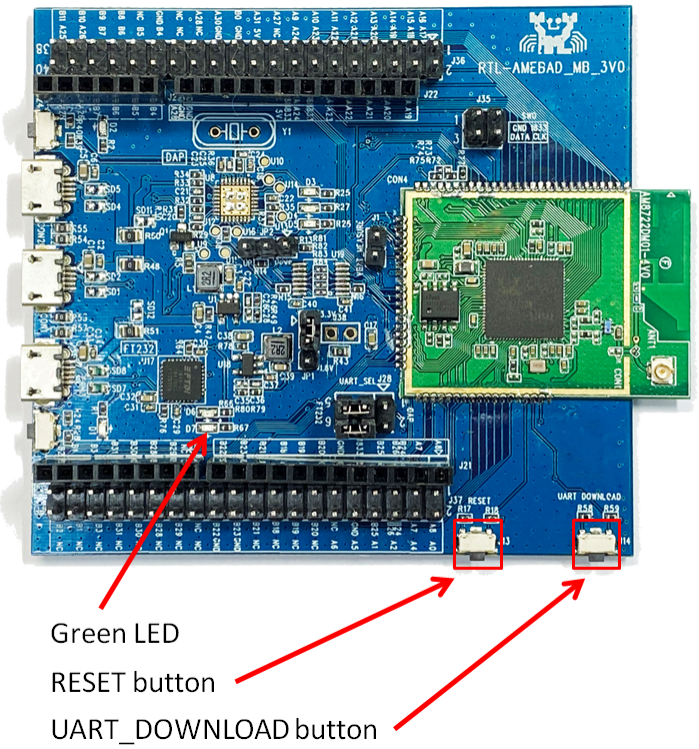
2.1.2 Upload Firmware into Ameba

Step 1. Navigate to "Release" folder

After downloading the MicroPython repository from GitHub, you will notice a "`Release`" folder in the root directory of this repository, enter this folder and locate a tool named "`Double-Click-Me-to-Upload`".

Step 2. Enter UART Download mode

To do this, first press and hold the `UART_DOWNLOAD` button, then press the `RESET` button. If success, you should see a green LED flashing on your ameba.



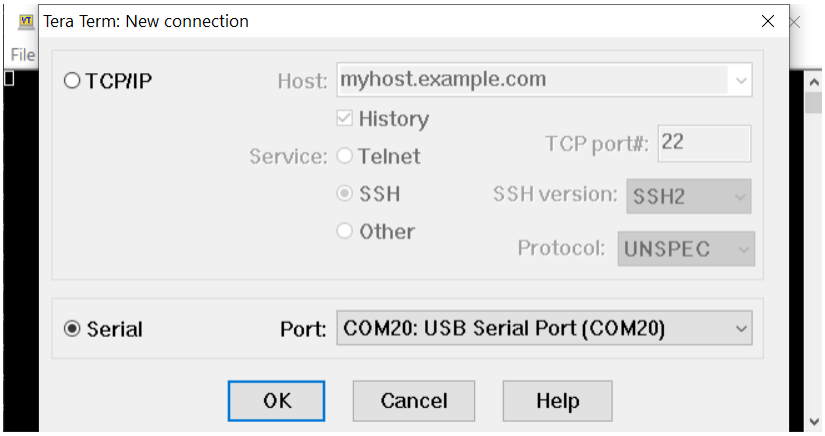
Step 3. Run "Double-Click-Me-to-Upload"

As the name suggested, double click on the file to run it, follow instructions printed on the screen to update the ameba's serial COM port (this is known to us during the driver installation step mentioned above) so the uploading can be carried out successfully. Once the uploading is successful, you will see a line of log printed on the screen – "All images are sent successfully"

2.1.3 Try the First Example

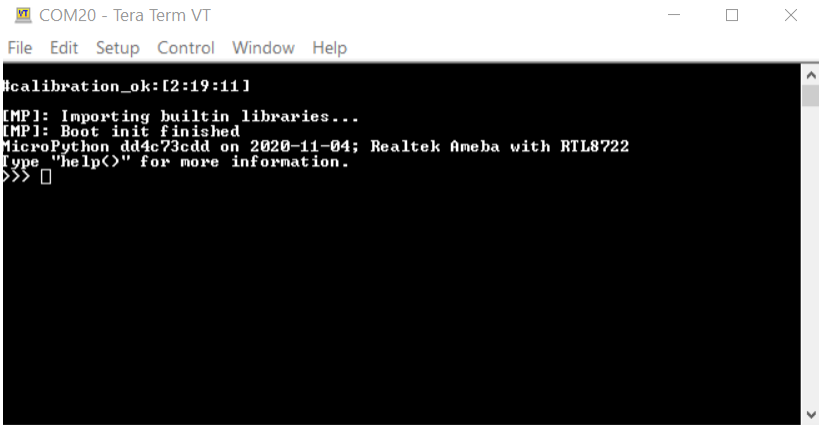
Step 1. Open REPL

REPL stands for Read, Evaluate, Print and Loop, it is the MicroPython's terminal for user to control the microcontroller. REPL is running on LOG UART, thus we need to open our serial terminal software, in this case, `Tera Term` to see REPL.



Once `Tera Term` is opened, select "Serial" like in the picture above and choose your Ameba's serial port using the dropdown list, after that, hit "OK". If your serial terminal is not configured to `115200` baud rate, now is the time to change it to `115200` and leave the rest of settings as default.

Now that the serial port is connected, press the RESET button once on your ameba and you should see the MicroPython's welcome page as shown below.



What happened here was that your Ameba first check its calibration data and then boot into MicroPython's firmware, MicroPython then run the "`boot.py`" python script and imported builtin libraries.

Now, you can simply type

```
>>> help()
```

to see more information, and type

```
>>> help(modules)
```

to check all readily available libraries.

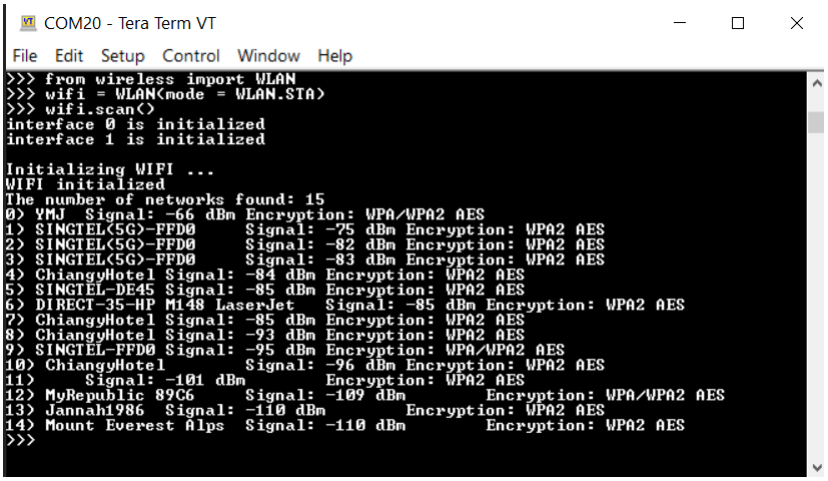
Step 2. Run WiFi Scan example

As most of peripherals' examples requires additional hardware to show the example is working, we will just use WiFi Scan example as our first example and to see how easy it is to control WiFi using MicroPython.

Now, please follow along by copy+paste the following code or manually typing them out into `Tera Term` and hit "Enter"

```
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.scan()
```

You should be able to see the returned result with all discovered wireless network in your surrounding



(End)

Note: If you face any issue, please refer to the *FAQ* and *Trouble-shooting* page.

2.2 Release History

Version 1.0.2 release - 2021/10/14

- Feature:
  - Add MacOS toolchain to support building on MacOS
  - Pre-test passed for RTL8722DM MINI
- API Updates:

  - Update PWM module with new API
  - Re-structure SDFS module and remove warning when no SD card
- Misc
  - Update WLAN and other libraries
  - Update readme documentation

Version 1.0.1 release - 2021/06/07

- Feature:
  - Added MacOS Support for firmware uploading (not compilation)
  - Fixed PWM API issue with loop
  - Implemented SDFS (SD FileSystem) module [Currently only support RTL8722DM_mini]
  - Update welcome message and help message
  - Update Ameba SDK and libraries
  - Fixed network and WLAN security issues
  - Fix bugs related to WiFi
  - Update Readme
  - Provide examples for new module

Version 1.0.0 release - 2020/11/11

- Feature:
  - OS Support Windows and Linux
  - WiFi
  - Socket
  - ADC
  - builtin help
  - Example and online API

Version 0.0.1 alpha release - 2020/09/29

- Feature:
  - Ported basic MicroPython functions
  - Implemented REPL and basic terminal functions
  - Added Pin Mapping for RTL8722
  - Added peripheral helper modules:
  - GPIO
  - RTC
  - Time and Delay

  - PWM
  - Timer
  - UART
  - I2C
  - SPI

2.3 Examples

2.3.1 Peripheral Examples

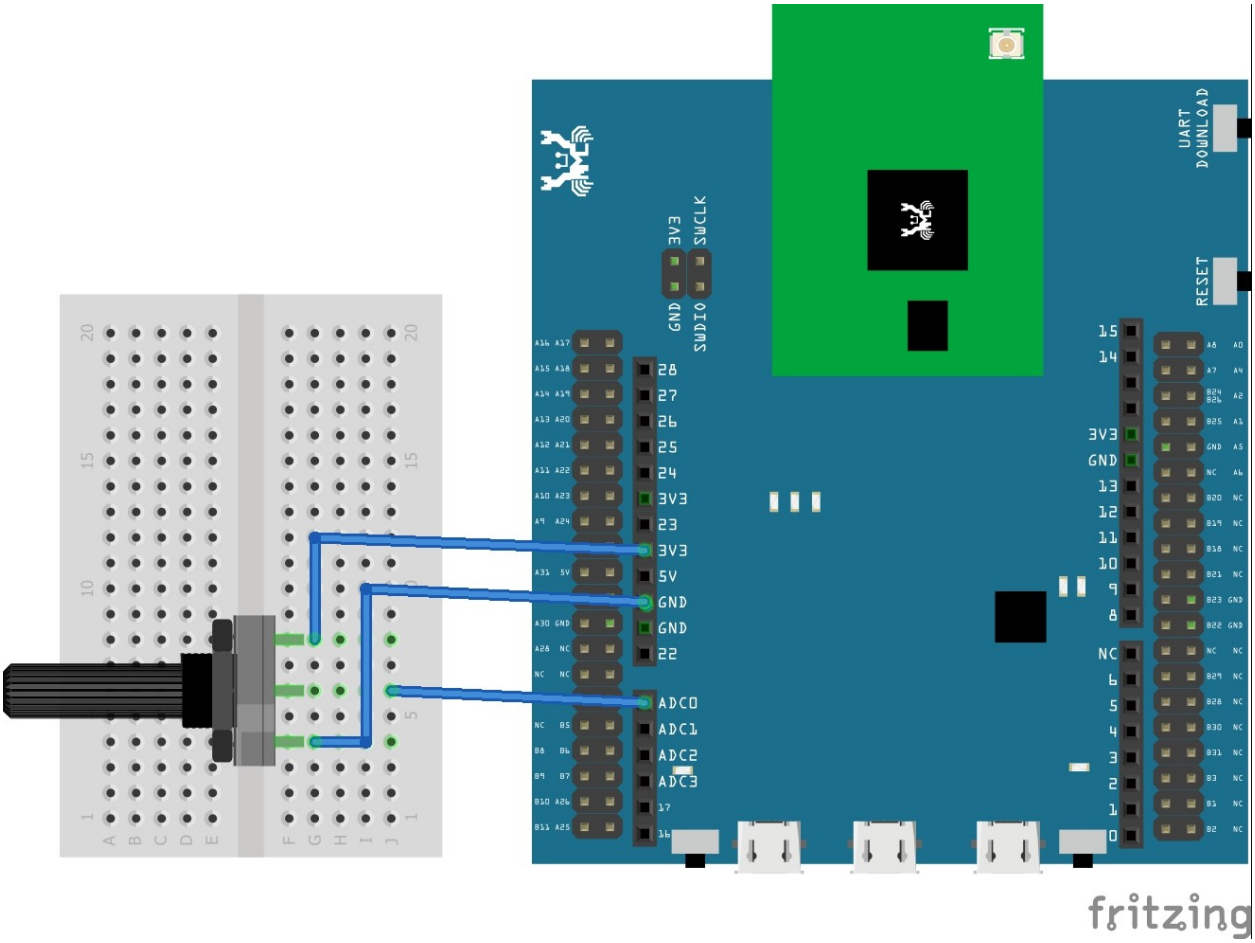
[RTL8722CSM] [RTL8722DM] ADC - Read potentiometer

Materials

- Ameba x 1
- Potentiometerx 1

Steps

Here we connect ameba to a potentiometer to measure its analogue value, the connection is as follows.



Copy and paste the following code into REPL.

```
import socket
a = ADC(0)
a.read()
```

[RTL8722CSM] [RTL8722DM] GPIO - Blink

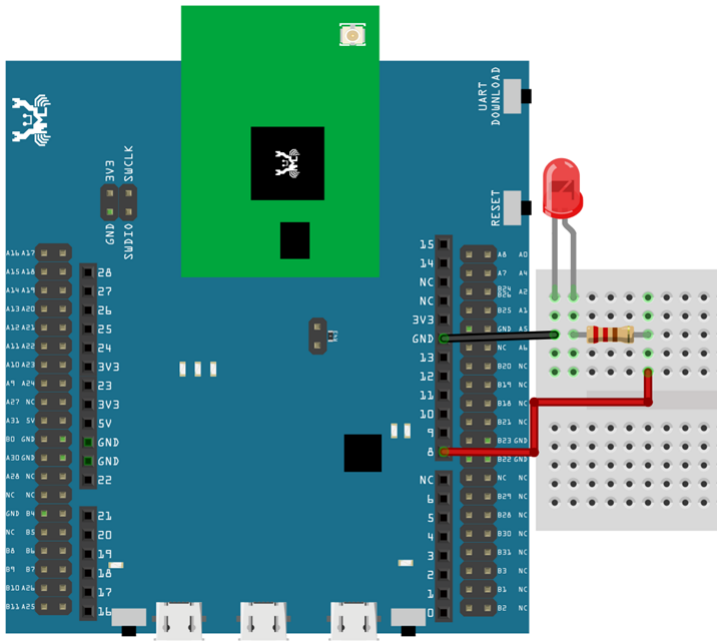
Materials

- Ameba x 1
- LED x 1
- Resistor(220ohm) x 1

Steps

Blink is one of the best examples to get started with MicroPython.

Let us connect pin PB_22 to the anode leg of an LED which in series with a current limiting resistor and GND to cathode of the LED as shown below,



Then, copy the following code and press `Ctrl + e` in `REPL` to enter the paste mode (for more information about `REPL` and paste mode, check "Getting Started" page). If you are using `Tera Term`, simply right click on any blank space of the terminal and paste the code into `REPL`, then press `Ctrl + d` to execute the code. If everything is order, you should be able to see the LED blink for 3 times in 3 seconds.

```
from machine import Pin
a = Pin("PB_22", Pin.OUT)
a.value(1)
time.sleep_ms(500)
a.value(0)
time.sleep_ms(500)
a.on()
time.sleep_ms(500)
a.off()
time.sleep_ms(500)
a.toggle()
time.sleep_ms(500)
a.toggle()
```

[RTL8722CSM] [RTL8722DM] I2C - Send and Receive

Materials

- Ameba x 1
- Arduino UNO x 1

Steps

I2C is a very common module on microcontrollers, it only takes 2 wire and able to achieve data rate at up to 3.4Mbps. It works in master-slave model and a master can simultaneously connect to up to 128 slaves, making it a very versatile communication protocol between microcontroller and sensor.

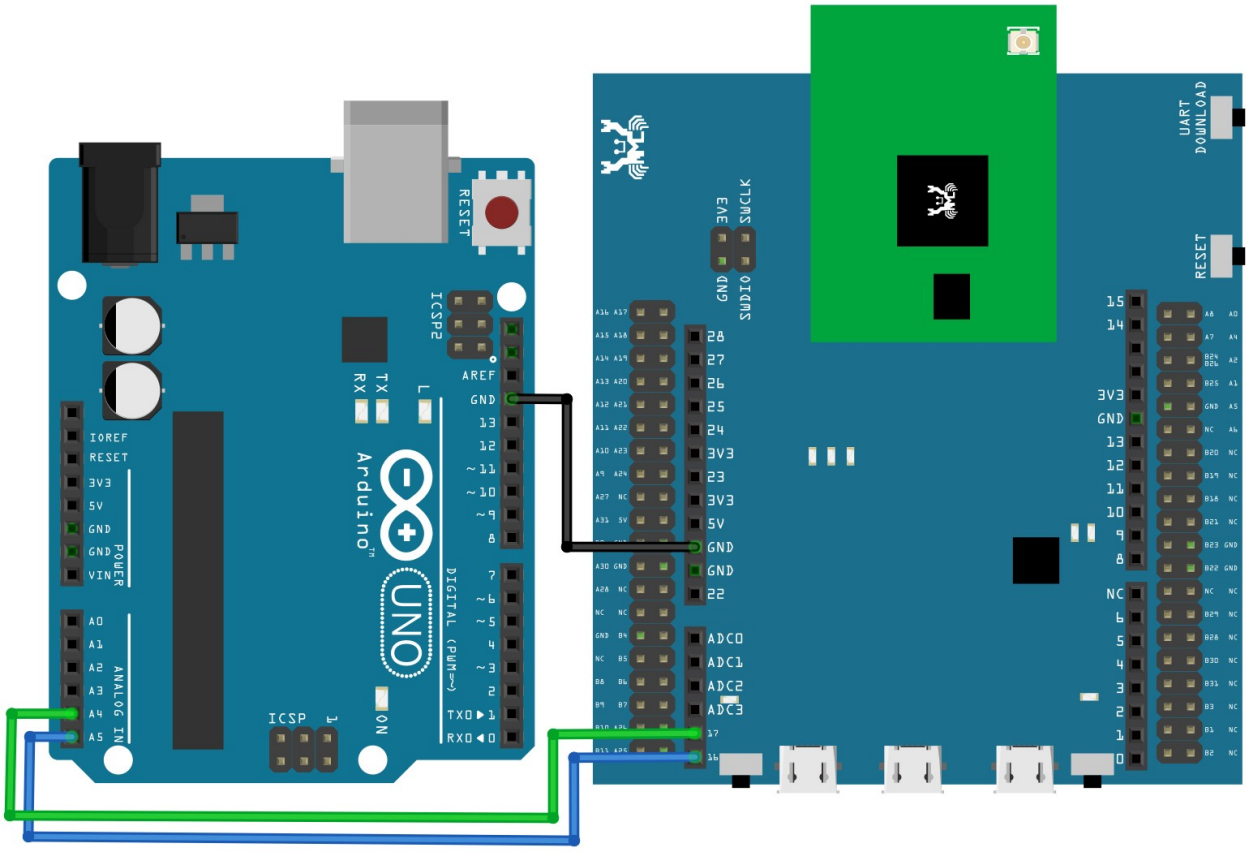
Here we are going to use Ameba as an I2C master and Arduino UNO as a slave to achieve I2C send and recv.

Before connection, make sure to upload the *"Examples -> Wire -> Slave_receiver"* example code to Arduino UNO.

Connection is shown as follows, here we are using PA_26 as SDA pin and PA_25 as SCL.

Note: There is currently 1 set of I2C available to MicroPython user, they are

Unit	SDA	SCL
0	PA_26	PA_25



Then copy and paste the following code line by line into REPL to see their effects.

```
from machine import Pin, I2C
i2c = I2C(scl = "PA_25", sda = "PA_26", freq=100000) # configure I2C with pins and␣
↪freq. of 100KHz
i2c.scan()
i2c.writeto(8, 123) # send 1 byte to slave with address 8
i2c.readfrom(8, 6) # receive 6 bytes from slave
```

[RTL8722CSM] [RTL8722DM] PWM - LED fade

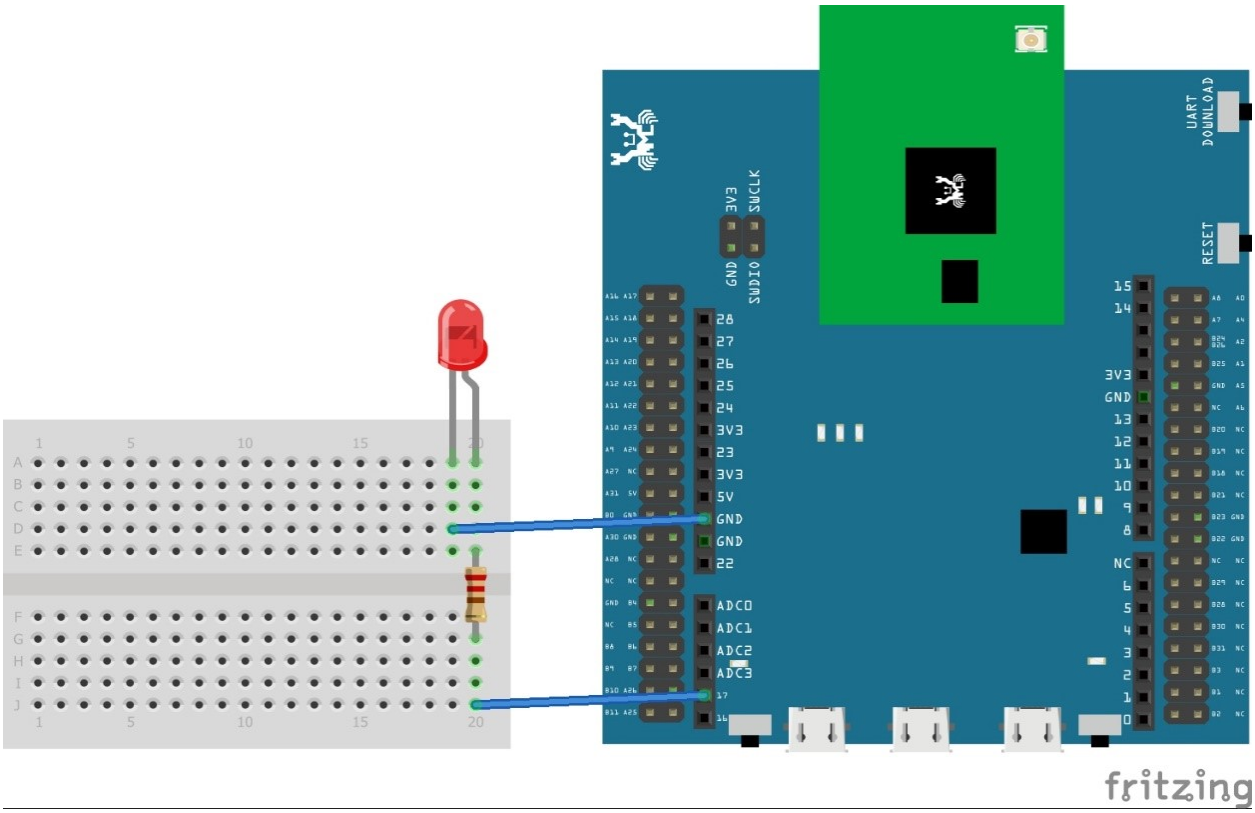
Materials

- Ameba x 1
- LED x 1
- Resistor(220ohm) x 1

Steps

PWM use pulse width modulation to control output duty cycle and is widely used to control LED brightness and motor. Here we are using an LED to demonstrate how PWM works.

Let us connect pin PA_26 to the anode leg of an LED which in series with a current limiting resistor and GND to cathode of the LED as shown below,



Then, copy and paste the following code line by line into REPL and hit Enter. If everything is in order, you should be able to see the LED slowly become brighter as you paste another line of code.

```
from machine import Pin, PWM
import time
p = PWM(pin = "PA_26")
# 0 duty cycle thus output 0
p.write(0.0)
# 10% duty cycle
p.write(0.1)
# 50% duty cycle
```

(continues on next page)

(continued from previous page)

```
p.write(0.5)
# 100% duty cycle
p.write(1.0)
```

[RTL8722CSM] [RTL8722DM] RTC - Get time

Materials

- Ameba x 1

Steps

RTC module help microcontroller to keep track of time and is essential to our time module. Here we an example to demonstrate how to get local time and update the time.

Copy and paste the following code line by line into REPL to see its effect.

```
rtc = RTC()
rtc.datetime() # get date and time
rtc.datetime((2020, 12, 31, 4, 23, 58, 59, 0)) # set a specific date and time (year,␣
↪month, day, weekday(0 for Monday), hour, minute, second, total seconds)
rtc.datetime() # check the updated date and time
```

[RTL8722CSM] [RTL8722DM] Socket - Echo Server and Client

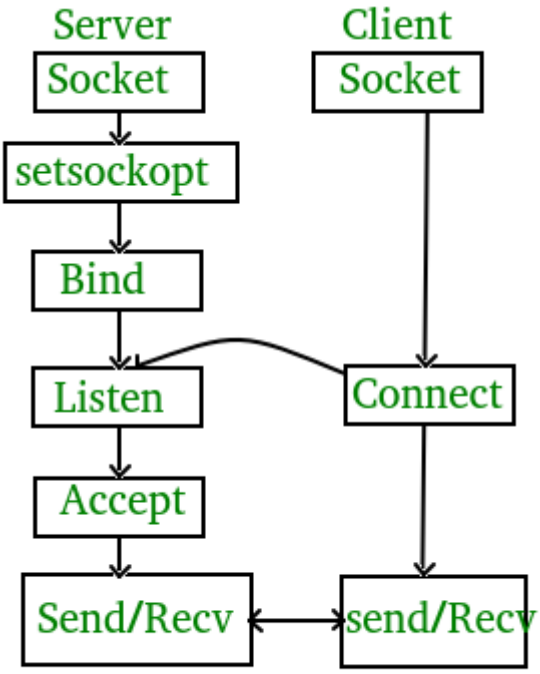
Materials

- Ameba x 2

Steps

After WiFi is set up, the best way to access the internet is to use socket. Socket is like an imaginary ethernet socket by which you use to connect your PC to some server on the internet like Google or Github.

Application layer protocol like HTTP are also built on top of socket. Once you are given an IP address and a port number, you can start to connect to the remote device and talk to it.



Here is an example of letting a server socket and a client socket to echo each other's message, to use this example, you need 2 ameba RTL8722 running MicroPython, copy and paste the following code to 2 ameba respectively under REPL paste mode.

This is the **server** code,

```
import socket
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.connect(ssid = "YourWiFiSSID", pswd = "YourWiFiPassword") # change the ssid and pswd to yours
s = socket.SOCK()
port = 5000
s.bind(port)
s.listen()
conn, addr = s.accept()
while True:
  data = conn.recv(1024)
  conn.send(data+"from server")
```

This is the **client** code,

```
import socket
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.connect(ssid = "YourWiFiSSID", pswd = "YourWiFiPassword") # change the ssid and pswd to yours
c = socket.SOCK()
# make sure to check the server IP address and update in the next line of code
c.connect("your server IP address", 5000)
c.send("hello world")
data = c.recv(1024)
```

(continues on next page)

(continued from previous page)

```
print(data)
```

[RTL8722CSM] [RTL8722DM] Socket - Get information from HTTP website

Materials

- Ameba x 1

Steps

With socket created, we can visit an HTTP website and get information from it.

Copy and paste the following code into REPL under paste mode.

```
import socket
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.connect(ssid = "YourWiFiSSID", pswd = "YourPassword") # change the ssid and pswd to yours
def http_get(url):
    _, _, host, path = url.split('/', 3)
    c = socket.SOCK()
    # We are visiting MicroPython official website's test page
    c.connect(host, 80)
    c.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host), 'utf8'))
    while True:
            data = c.recv(100)
            if data:
            print(str(data,'utf8'), end='')
            else:
            break
http_get('http://micropython.org/ks/test.html')
```

[RTL8722CSM] [RTL8722DM] SPI - Slave Receive

Materials

- Ameba x 1
- Arduino UNO x 1

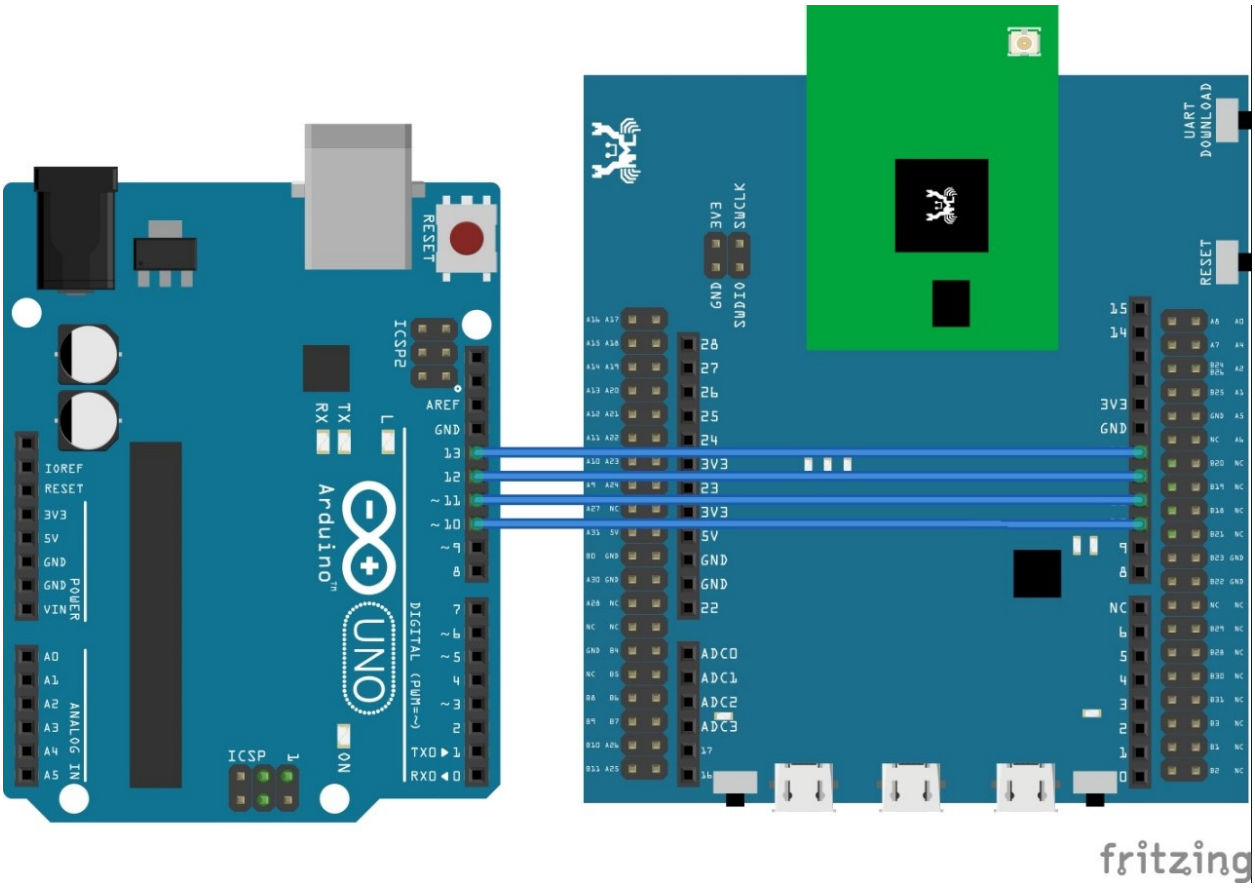
Steps

SPI is a fast and robust communication protocol that are commonly found on many microcontrollers and is often used to retrieve sensor data or output image signal to a display. Ameba support SPI in both master and slave mode. Here we are going to see an example demonstrating how ameba receive data in slave mode on MicroPython.

Before connection, make sure to upload the following code to your Arduino UNO.

```
//////////////////////
// SPI Master Write //
//////////////////////
#include <SPI.h>
void setup (void) {
              Serial.begin(115200); //set baud rate to 115200 for usart
              digitalWrite(SS, HIGH); // disable Slave Select
              SPI.begin ();
}
void loop (void) {
              char c;
              digitalWrite(SS, LOW); // enable Slave Select
              // send test string
              for (const char * p = "Hello, world!\r" ; c = *p; p++) {
              SPI.transfer(c);
              Serial.print(c);
                      }
                 Serial.println();
              digitalWrite(SS, HIGH); // disable Slave Select
              delay(2000);
}
```

Connection is shown as follows, here we are using unit 0 as SPI slave, and Arduino UNO as SPI master,



Then copy and paste the following code into REPL under paste mode to see their effects.

```
from machine import SPI
s1= SPI(0 , mode = SPI.SLAVE)
for i in range(14):
chr(s1.read())
```

[RTL8722CSM] [RTL8722DM] Time - Delay and Timing

Materials

- Ameba x 1

Steps

MicroPython has provided rich functions to deal with time and delay, here are some examples.

Copy and paste the following code line by line into REPL to see its effect.

```
import time
time.sleep(1) # sleep for 1 second
time.sleep_ms(500) # sleep for 500 milliseconds
time.sleep_us(10) # sleep for 10 microseconds
start = time.ticks_ms() # get millisecond counter
```

[RTL8722CSM] [RTL8722DM] Timer - Periodical timer

Materials

- Ameba x 1

Steps

There are 3 sets of general timers available to user, each at 32KHz, they are timer 1/2/3. Here we use timer 1 as example to demonstrate how a periodical timer works.

Copy and paste the first 3 lines of code into REPL to see its effect.

```
from machine import Timer
t = Timer(1) # Use Timer 1/2/3 only
t.start(2000000, t.PERIODICAL) # Set GTimer fired periodically at duration of 2␣
↪seconds, printing text on the terminal
# To stop the periodical timer, type
t.stop()
```

A text of `-timer triggered. to stop: type t.stop()-` will be printed on the terminal every 2 seconds. To stop the timer, simply type `t.stop()`.

[RTL8722CSM] [RTL8722DM] UART - Send and Receive

Materials

- Ameba x 1
- TTL USB to Serial module x 1

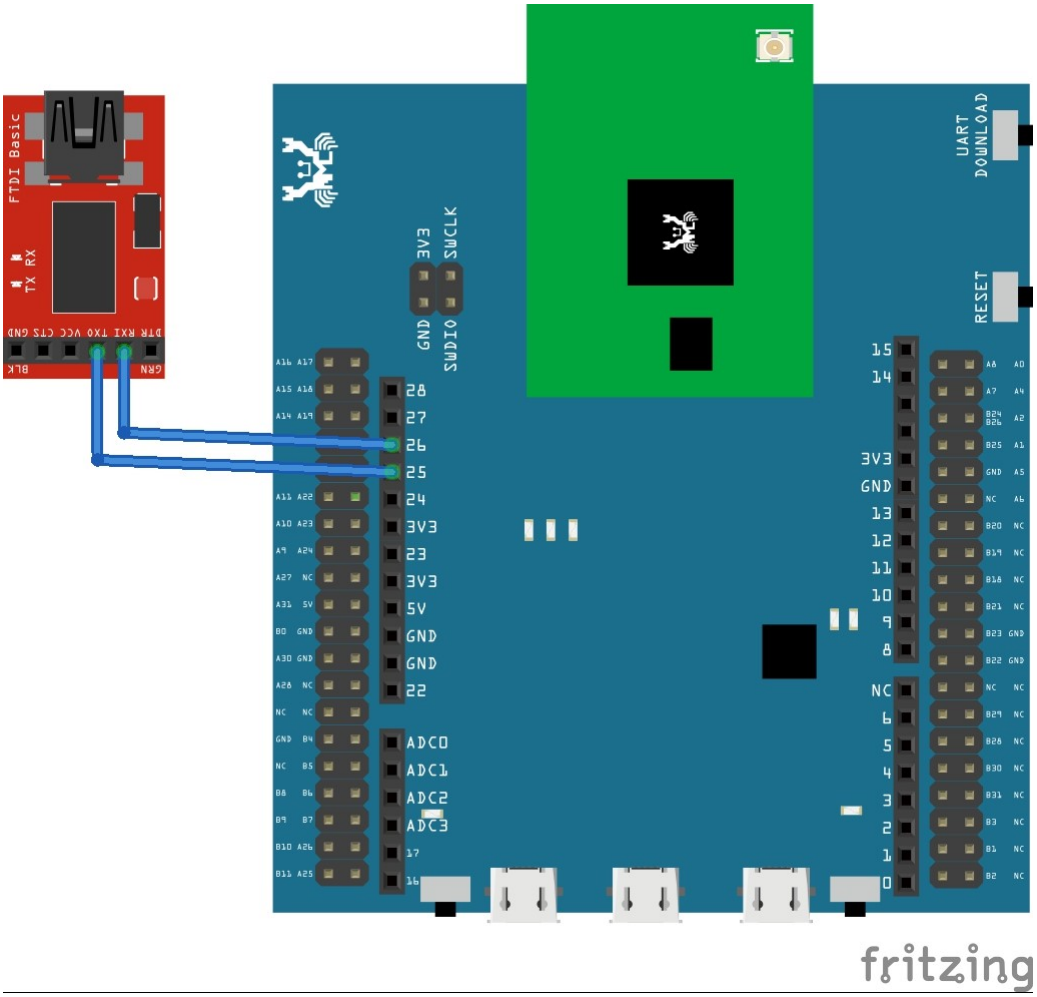
Steps

UART is a very versatile communication protocol and almost an essential part of a microcontroller. A TTL USB to Serial module is an IC that helps to translate UART signal to USB signal so that we can see uart log printed on our PC. This module is often found on many development boards, including ameba. However, the module on Ameba is reserved for LOG UART and Firmware uploading, that is why we need a separate module to communicate between ameba and PC.

There are currently 2 sets of UART available to MicroPython users and they are:

Unit	tx	RX
0	PA_21	PA_22
3	PA_26	PA_25

Here we are using unit 0 to demonstrate how uart works on ameba.
Connect TTL module to `PA_21` and `PA_22` as shown below,



Then, copy and paste the following code line by line into REPL to see its effect.

```
from machine import UART
uart = UART(tx="PA_21", rx= "PA_22")
uart.init()
uart.write('hello')
uart.read(5) # read up to 5 bytes
```

[RTL8722DM_MINI] SDFS - Data Editing

Materials

- Ameba RTL8722DM_MINI x 1
- MicroSD Card x 1 (SD card must be < 32GB with format set to fatfs)

Steps

SD File System module supports SD card data manipulation in the form of file. With it, you can control and inspect files as you like and keep them on non-volatile memory.

Copy and paste the following code line by line into REPL to see its effect.

```
from machine import SDFS
s=SDFS()                # create a short form
s.create("ameba.txt")   # create a file named "ameba.txt"
s.write("ameba.txt", "ameba supports sd card file system!") # write a string to the␣
↪file just created
s.read("ameba.txt")     # read the content from the same file
s.rm("ameba.txt")       # delete the file
```

Note: No file open or close is needed, the API does that automatically for you.

[RTL8722DM_MINI] SDFS - Directory Navigation

Materials

- Ameba RTL8722DM_MINI x 1
- MicroSD Card x 1 (SD card must be < 32GB with format set to fatfs)

Steps

SD File System is supported on MicroPython RTL8722 port through importing the SDFS module from the machine module. This module is a simplified file system with the aim to highlight SD card manipulation, thus it does not support virtual file system as well as virtual file object.

Copy and paste the following code line by line into REPL to see its effect.

```
from machine import SDFS
s=SDFS()           # create an instance and mount on file system on SD card
s.listdir()        # listing the files and folders under current path
s.mkdir("test")    # create a folder named "test" under current path
s.chdir("test")    # change directory to test folder
s.pwd()            # print out present working directory(current path)
s.chdir("/")       # change directory bach to root directory
s.rm("test")       # delete the test folder
```

Note: No file open or close is needed, the API does that automatically for you.

2.3.2 Network Examples

[RTL8722CSM] [RTL8722DM] WiFi - WiFi Connect

Materials

- Ameba x 1

Steps

Ameba can connect to WiFi access point with open security or WPA2 security type, which is the most common security type used in household wireless routers.

Here we are going to connect to a WiFi access point using code below, copy and paste the following code line by line into REPL to see their effects.

```
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.connect(ssid = "YourWiFiName", pswd = "YourWiFiPassword")
```

[RTL8722CSM] [RTL8722DM] WiFi - WiFi Scan

Materials

- Ameba x 1

Steps

WiFi Scan function can help us quickly discover what WiFi networks are available in our surrounding.

This example does not require any additional hardware, thus simply copy, and paste the following code into REPL to see its effect.

```
from wireless import WLAN
wifi = WLAN(mode = WLAN.STA)
wifi.scan()
```

2.4 Board HDK

2.4.1 EVB

- **HDK-AMEBAD_MB_4V2**
  - Layout
  - Schematic

- **HDK-AMEBAD_MB_4V0**
  - Layout
  - Schematic

2.4.2 RTL8722DM Module

- **HDK-AM8722DM01_6V2_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_6V1_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_4V1**
  - Layout
  - Schematic

2.5 API Documents

RTL8722DM MicroPython Online API Documents

2.5.1 ADC

API Documents

Constructors

ADC(*unit* [required])
Create an ADC object associated with the given unit ID. This allows you to then read analog values on the pin assigned to the unit ID.

- `unit`: unit number is tied to a specific pin. Refer to table below for more information,

Unit	Pin
0	PB_4
1	PB_5
2	PB_6
3	PB_7
4	PB_1
5	PB_2
6	PB_3

Methods

ADC.read()
Read the value on the analog pin and return it.

2.5.2 I2C

API Documents

Constructors

I2C(*unit_id* [optional], *"sda_pin"* [required], *"scl_pin"* [required], *frequency* [optional])
Create a I2C object associated with the given pin name and configure it using other parameters.
This allows you to then read/write data on the I2C bus

- `unit_id` : The unit ID of the hardware I2C, assume default value if left blank
- `"sda_pin"`: The pin name of SDA
- `"scl_pin"`: The pin name of SCL
- `frequency`: The frequency at which I2C operates at, assume default value if left blank.

Note: All optional parameters have default values as follows:

Parameter	Default
Unit_id	0
Frequency	100000 (Hz)

Methods

I2C.reset()
This method de-initializes the I2C device.

I2C.scan()
This method scans and return the available I2C addresses.

I2C.readinto(*buf* [required], *flag*[optional])
This method reads the data received at I2C buffer into a user-declared buffer

- **buf**: a buffer of string / array /byte array type
- **flag**: a Boolean flag, if True then send a NACK at the end, vice versa

I2C.write(*buf*[required]**)**
This method sends data stored in the buffer.

- **buf:** a buffer of string / array /byte array type

I2C.readfrom(*addr*[required], *len*[required], *stop*[optional]**)**
This method reads len bytes of data from given address, if stop is True, then send a STOP bit at the end of the transmission.

- **addr**: the address to read from
- **len**: the number of bytes to expect
- **stop**: a Boolean flag whether or not to send a STOP bit at the end of transmission

I2C.readfrom_into(*addr*[required], *buf*[required], *stop*[optional]**)**
This method reads data from given address into the user-declared buffer provided, if stop is True, then send a STOP bit at the end of the transmission.

- **addr**: the address to read from
- **buf**: a data buffer of string / array/ byte array type
- **stop**: a Boolean flag, if True then send a STOP bit at the end of transmission, vice versa

I2C.writeto(*addr*[required], *value*[required], *stop*[optional]**)**
This method sends an integer value to the given address, if stop is True, then send a STOP bit at the end of the transmission.

- **addr**: the address to write to
- **value**: an integer value to be sent over
- **stop**: a Boolean flag, if True then send a STOP bit at the end of transmission, vice versa

2.5.3 Pin

API Documents

Constructors

Pin(*"pin_name"*[required]*, direction*[required]*, pull_mode*[optional]*, value*[optional])
Create a Pin object associated with the given gpio pin name and configure it using other parameters. This allows you to then read/write digital values on the pin.

- `"pin_name"`: The name of the pin, must be in string format, use help(Pin.board) to check all pin names

- `direction`:
  - `Pin.IN` – for input
  - `Pin.OUT` - for output
- `pull_mode`:
  - `Pin.PULL_NONE` – no pull-up or down resistor
  - `Pin.PULL_UP` – enable pull-up resistor
  - `Pin.PULL_DOWN` – enable pull-down resistor
  - default value – `Pin.PULL_NONE`
- `value`: Initial value, only applicable to OUTPUT, for example `value = 1`. Default `value = 0`.

Methods

Pin.id()
This method will return the associated GPIO pin name after declaring a Pin object.

Pin.init(*"pin_name"*[required]*, direction*[required]*, pull_mode*[optional]*, value*[optional])
Identical function as the Constructor, it creates and initializes a Pin object using parameter typed in.

Pin.value(*number*[optional])
This method can be used in 2 ways,

1. Output *number* keyed in

   number can only be either 0 or 1 , indicating logic 0 or logic 1

2. Check the status of the pin

   When left blank, this method will check the status (logic 0 /1) of the Pin, regardless of which direction this Pin is configured as.

Pin.on()
This method sends a logic 1 signal to the associated pin

Pin.off()
This method sends a logic 0 signal to the associated pin

Pin.toggle()
This method toggles the logic signal of the associated pin

2.5.4 PWM

API Documents

Constructors

PWM(*unit*[optional]*, "pin_name"*[required]**)**
Create a PWM object associated with the given pin name. This allows you to then write PWM signal on the pin.

- `unit`: unit ID of the hardware PWM, will use default unit 0 if leave blank
- `"pin_name"`: The name of the pin, must be in string format. See below for PWM supported pins.

Note:

PWM is currently only supported on the following pins,
PA_23 , PA_24 , PA_25 , PA_26

Methods

PWM.write(*dutycycle_float*[required]**)**
This method will output a PWM signal with given duty cycle on the associated GPIO pin.

- **dutycycle_float**: a floating point duty cycle value, can be from 0.0 (0%) to 1.0 (100%)

2.5.5 RTC

API Documents

Constructors

RTC()
Create a RTC object.

Methods

RTC.datetime(*array_8*[optional]**)**
This method works in 2 ways

- Return the local date and time if NOT passing any argument into it.
  The returned format is as follows:
  (year, month, date, hour, minute, second, weekday[0-6 for Mon to Sun], yearday[1-366])
- Update the local date and time if passing an eight-elements array into it, the array format is same as above

2.5.6 SDFS

API Documents

Constructors

SDFS()
Create a `sdfs` object and configure it to the given mode. This then allows you to navigate through the SD card and read/write files as you see.

Note: No parameter is required

Methods

sdfs.listdir()
This method listing the files and folders under current path.

sdfs.mkdir(*"folder name"*[required] **)**
This method attempts to create a folder under current path.

- **folder name**: the name of the new folder/directory you wish to create, it must be a string less than 128 characters

sdfs.chdir(*"folder name"*[required] **)**
This method change directory to the one given in the parameter.

- **folder name**: the name of the folder/directory you wish to navigate to, it must be a string less than 128 characters

---

Note: Key in "/" as the parameter to this API would navigate back to the root directory.

---

sdfs.pwd()
This method is to print out present working directory (current path).

sdfs.rm(*"folder name or file name*[required]" **)**
This method is to delete a file or a folder. Note that, the folder to be deleted has to be empty before it can be deleted successfully.

- **folder name or file name**: the name of the folder or file you wish to delete, it must be a string less than 128 characters

sdfs.create(*"file name*[required]" **)**
This method is to create a file.

- **file name**: The name of the file you wish to create.

sdfs.write(*"file name*[required]" , *"string*[required]"**)**
This method is to write your input as a string to a file specified.

- **file name**: The name of the file you wish to read.
- **string**: The data you wish to write.

sdfs.read(*"file name*[required]"**)**
This method is to read the content from a file.

- **file name**: The name of the file you wish to read.

2.5.7 Socket

API Documents

Constructors

socket.SOCK(*domain*[optional], *type*[optional]**)**
Create a SOCK object and configure it with the given parameters. SOCK class is under socket class and is the main class we use for all socket level communications.

- `domain`: domain address family type. Default is `AF_INET`
  - `AF_INET`: IPv4, classic IP address with dot-notation that is slowly being replaced by IPv6 due to shortage.
  - `AF_INET6`: IPv6, IP address with colon-notation
- `type`: socket type, default is `SOCK_STREAM`
  - `SOCK_STREAM`: TCP type
  - `SOCK_DGRAM`: UDP type

Methods

socket.SOCK.connect(*host*[required], *port*[required]**)**
This method connects to a remote server as client.

- **host:** a website address in string
- **port:** port number in integer

socket.SOCK.bind(*port*[required]**)**
This method creates a server socket and binds it to the given port number.

- **port:** port number in integer

socket.SOCK.listen()
This method set the server to listening state, waiting for client connection at the given port.

socket.SOCK.accept()
This method accepts a client connection and return a new socket object for subsequent communication and client's address.

socket.SOCK.recv(*length*[required]**)**
This method receive data with given length

- **length:** the length of data expected to receive

socket.SOCK.send(*buffer*[required]**)**
This method sends data stored in the buffer

- **buffer:** a data buffer in format of array/bytearray/string

socket.SOCK.settimeout(*seconds*[required]**)**
This method set socket's timeout to the given value

- **seconds:** new timeout in seconds

socket.SOCK.close()
This method close the socket.

2.5.8 SPI

API Documents

Constructors

SPI(*unit_id*[required], *baudrate*[optional], *polarity*[optional], *phase*[optional], *databits*[optional], *firstbit*[optional], *miso*[optional], *mosi*[optional], *sck*[optional], *mode*[optional])
Create a SPI object and configure it using other parameters. This allows you to then read/write data on the SPI bus.

- `unit_id`: The unit ID of the hardware SPI, assume default value if left blank
- `baudrate`: The speed of SPI
- `polarity`: one of factor determining SPI mode. (deprecated)
- `phase`: one of factor determining SPI mode. (deprecated)
- `databits`: number of data bits
- `Firstbit`: this determine whether first bit is MSB or LSB
- `miso`: miso pin. (deprecated)
- `mosi`: mosi pin. (deprecated)
- `sck`: clock pin. (deprecated)
- `mode`: either MASTER mode or SLAVE mode

Note: All optional parameters have default values as follows:

Parameter	Default
Baudrate	2000000 Hz
Polarity	Inactive_low
Phase	Toggle_middle
Databits	8
Firstbit	MSB
Miso	N.A.
Mosi	N.A.
Sck	N.A.
Mode	MASTER

There is currently 2 set of SPI, they are:

unit	MOSI	MISO	SCK	CS
0	PB_18	PB_19	PB_20	PB_21
1	PB_4	PB_5	PB_6	PB_7

Note: both unit support master mode, but only `unit 0` supports slave mode.

Methods

SPI.read()
This method waits and read data received in SPI buffer, then return the data received. Works in both master and slave mode.

SPI.write(*value*[required])
This method writes an integer value to SPI bus. Works in both master and slave mode.

- **value**: an integer value to be sent on SPI bus

2.5.9 Time

API Documents

Constructors

N.A.

Methods

time.sleep(*seconds*[required])
This method will stop the microcontroller from what it is doing and delay for the given time.

- **seconds**: number of seconds, must be an integer

time.sleep_ms(*milliseconds*[required])
This method will stop the microcontroller from what it is doing and delay for the given time.

- **milliseconds**: number of milliseconds, must be an integer

time.sleep_us(*microseconds*[required])
This method will stop the microcontroller from what it is doing and delay for the given time.

- **microseconds**: number of microseconds, must be an integer

time.time()
This method will return the total number of seconds elapsed since Epoch (1970-01-01).

time.localtime()
This method will return RTC's local time in the following format,
(year, month, date, hour, minute, second, weekday[0-6 for Mon to Sun], yearday[1-366])

time.mktime(*tuple*[required])
This is inverse function of localtime. Its argument is a full 8-tuple which expresses a time as per localtime. It returns an integer which is the number of seconds since Jan 1, 2000.

- **tuple**: an 8-element tuple

time.ticks_ms()
This method returns an increasing millisecond counter with an arbitrary reference point. Normally used together with **ticks_add()** and **ticks_diff()**

time.ticks_add(*starting_ticks*[required], *ticks_added*[required])
This method add given number of ticks to the starting_ticks.

- **starting_ticks**: millisecond counter obtained from ticks_ms()
- **ticks_added**: number of ticks to add

time.ticks_diff(*end_ticks*[required], starting_*ticks*[required])
This method perform subtraction on parameters given and return the difference of end_ticks minus starting_ticks.

- **end_ticks**: millisecond counter obtained from ticks_ms()
- **starting_ticks**: millisecond counter obtained from ticks_ms()

2.5.10 Timer

API Documents

Constructors

Timer(*unit*[optional])
Create a timer object with given unit ID.

- `unit`: can be 1 / 2 / 3 for timer 1 / 2 / 3

Methods

Timer.start(*microseconds*[required], *type*[required])
This method will start a given type of timer, either one-shot or periodical, at duration of given microseconds.

- **microseconds**: number of microseconds interval, must be an integer
- **type**: either Timer. PERIODICAL or Timer.ONESHOT

Timer.deinit()
This method will de-initialize the Timer object created and stop the timer.

Timer.stop()
This method will stop the timer and its timer interrupt handler.

Timer.us ()
This method will return the current timer tick in microsecond.

Timer.tick ()
This method will return the current timer tick in Gtimer clock(0~32768).

Timer.reload (*duration_us*[required])
This method will reload the timer with given duration in microsecond.

- **duration_us**: duration in microsecond

2.5.11 UART

API Documents

Constructors

UART(*unit*[optional], *baudrate*[optional], *databits*[optional], *stopbit*[optional], *paritybit*[optional], *timeout*[optional], *tx_pin*[required], *rx_pin*[required])
Create a UART object associated with the given tx and rx pins and configure it using other parameters. This allows you to then read/write uart signal on the pins.

- `unit`: The unit ID, either 0 or 3
- `baudrate`: 115200 is the recommended baudrate on ameba
- `databits`: the number of bits for data bits, usually 7 or 8 bits
- `stopbits`: the number of stop bits, usually 1 or 1.5 or 2 bits
- `paritybit`: for parity check, usually none, odd or even

- `timeout`: how long uart wait before its timeout (in milliseconds)
- `tx_pin`: the transmitter pin, connect the rx pin of the receiver
- `rx_pin`: the receiver pin, connect to tx pin of the transmitter

---

Note: Not all parameters are required, thus MicroPython will assume its default value once left blank, here are the default values for each optional parameter:

Parameter	Default Value
Unit	0
Baudrate	115200
Databits	8
Stopbits	1
Paritybit	0
Timeout	10 (ms)

Methods

UART.init()
This method initializes and configures the UART.

UART.read(*length*[optional])
This method reads the data received in UART buffer.

- **length:** the length of the data to receive

UART.readline()
This method is similar to read(), but read a line ending with a newline character.

UART.write(*buffer*[require])
This method sends the buffer of bytes to the bus and returns the number of bytes written.

- **buffer**: data buffer that can be a string, an integer or other data types

UART.irq_enable(*bool*[optional])
This method works in 2 way:

1. Check the status of uart irq when NOT passing any argument, and it will return True if irq is enabled, False if disabled
2. Enable/disable uart irq handler by passing True or False as bool

UART.irq_handler(*function*[required])

Passing the python handler to uart irq so that it will be triggered when an UART event occurs.

- **function:** a function defined in python or a lambda function

2.5.12 WiFi

API Documents

Constructors

WLAN(*mode*[required]**)**
Create a WLAN object and configure it to the given mode. This then allows you to control WiFi and check its status.

- `mode`: use WLAN.STA for station mode

Methods

WLAN.scan()
This method scan and list out all available WiFi network in the surroundings.

WLAN.connect(*ssid*[required], *pswd*[optional], *security*[optional]**)**
This method attempts to establish a connection to a WiFi access point.

- **ssid**: The name of your WiFi network
- **pswd:** The password of your WiFi network
- **security:** The security type of your WiFi network

Note: Leaving optional parameters blank will assume taking default values, which are:

Parameter	Default value
pswd	NULL
security	WPA2_AES_PSK

Note:

Connecting to an **OPEN** network is also supported, just omit
`pswd` parameter and type in `security = WLAN.OPEN` followed by ssid.

WLAN.get_ip()

This method returns the IP address of the current WLAN interface. Only works after successful connection to an AP.

WLAN.disconnect()
This method disconnect ameba from current WiFi AP, but still keep WiFi module on.

WLAN.on()
This method turns on the WiFi device.

WLAN.off()
This method shut down WiFi device and suspend all connections.

WLAN.wifi_is_running()
This method returns the WiFi status. True when WiFi is on, and False when off.

WLAN.is_connect_to_ap()
This method returns the connection status. True if ameba is connected to an AP, False if ameba is not connected to anything.

2.6 Resources

2.7 Support

2.7.1 FAQ

What is MicroPython and how to use it?

Please refer to MicroPython official website for more information.

Can I use all Python libraries available online?

No, MicroPython only support a small section of the classic Python standard library. However, this can be done by porting the classic python library to MicroPython.

Are all pins on RTL8722CSM/RTL8722DM usable?

No, those marked “NC” are not connected to any pin and thus unusable.

Does RTL8722CSM support 5G WiFi?

No. Only RTL8722DM supports dual band 2.4G + 5G WiFi. RTL8722CSM only supports single band 2.4G WiFi.

How to enter the download mode?

Press and hold the UART DOWNLOAD button. Then Press the RESET button and release both UART DOWNLOAD and RESET buttons.

2.7.2 Trouble-shooting

Compilation of MicroPython firmware failed

During the building process, some user may encounter error that suspend the process, this is due to missing system environment setup and can be fixed as follows,

1. **Error related to Python**

   By default, MicroPython uses *Python3* to run building scripts for the MicroPython kernals, if you encounter error related to python, it may be because the path of the *Python3* executable is not added to system environment variable.

   However, if environment variable is already added but the build could not be completed, you may try,

   1) Re-start your PC
   2) Type "python" on your terminal, if the python shown is Python3, then please add
      `PYTHON = python`
      to the second line of the *Makefile* under *port/rtl8722* folder

2. **Error related to MPY-CROSS**

   If building process stop when mpy-cross shown as error, there is a step to be done as follows,

   1) navigate to "MicroPython_RTL8722/mpy-cross" folder
   2) Open your Cygwin/Linux terminal and just type
      `make`
      Wait for make finish building the MicroPython Cross Compiler, then this should fix the error

My code is not behaving as I expected

Try to debug your program using **print( )** function and learn more about each API used through the API page.

Why am I constantly getting "syntax error" from REPL?

Please note that MicroPython only support **Python3** syntax.

How to upload my python script into Ameba?

There are 3 ways of uploading your python code into Ameba,

1. **via REPL normal mode**

   In the normal REPL mode, you can paste your into REPL code line by line and have them executed sequentially, but note that syntax will be automatically indented when using condition checking or loop, like "if" or "while", incorrect indenting will crash your input script

2. **via REPL paste mode**

   When in normal REPL mode, press "Ctrl"+ "e" will enter paste mode, paste mode only allow pasting a large chunk of a complete code, incomplete code or editing after pasting will mess up your syntax and cause error

3. **via mp_frozenmodules**

   By placing your python script into the "mp_frozenmodules" folder under "rtl8722" folder, your code will be embedded into the MicroPython firmware and uploaded to Ameba, after that you can use it by simply importing the name of your python script. If you get syntax error using this method, you better check your python code syntax again.

Why is there no output on my serial terminal after connecting to RTL8722CSM/RTL8722DM UART?

RTL8722CSM/RTL8722DM is by default configured at 115200 baudrate, please check if your serial terminal is configured to 115200.

My program is not being downloaded into RTL8722CSM/RTL8722DM?

Please follow the procedure for the correct downloading

1. Enter the download mode. The on-board Green LED will blink when entered download mode.
2. When downloading the image into board the on-board Red LED will blink
3. After a successful download, you will see log like this "All images sent successfully".

Sometimes WiFi signal is weak?

The default antenna for RTL8722CSM/RTL8722DM uses the I-Pex Connector. Please change/connect the I-Pex Connector antenna.

Why is my board not powering up?

Please make sure the connector J38 beside resistor R43 is connected. The connector is used to link the power to IC.

If you have driver issue to connect board to your computer?

Please go to https://ftdichip.com/drivers/ for USB driver.

CHAPTER

THREE

STANDARD SDK

Welcome to Ameba Standard SDK online documentation

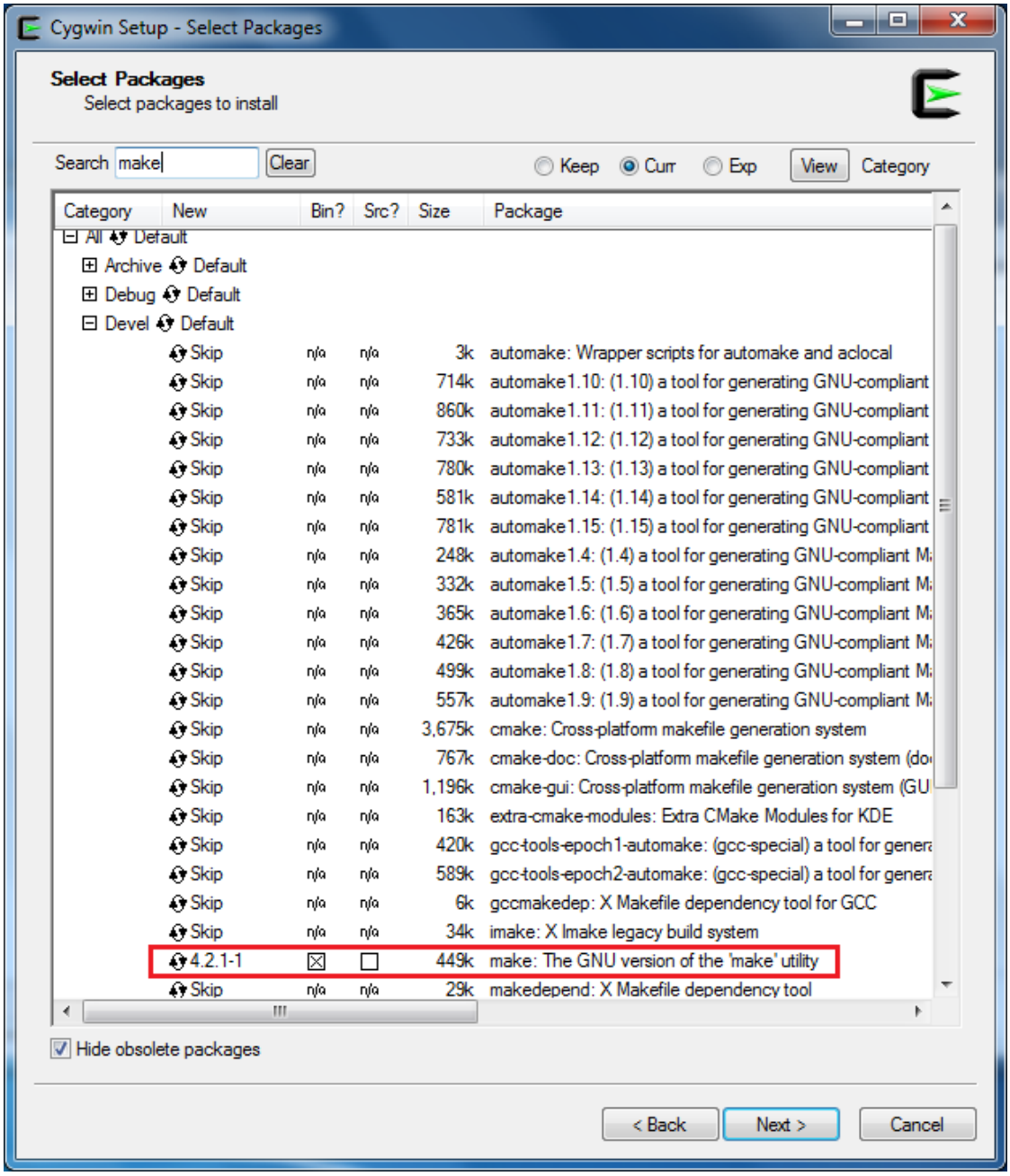
Ameba RTL8722DM (AMB 21)

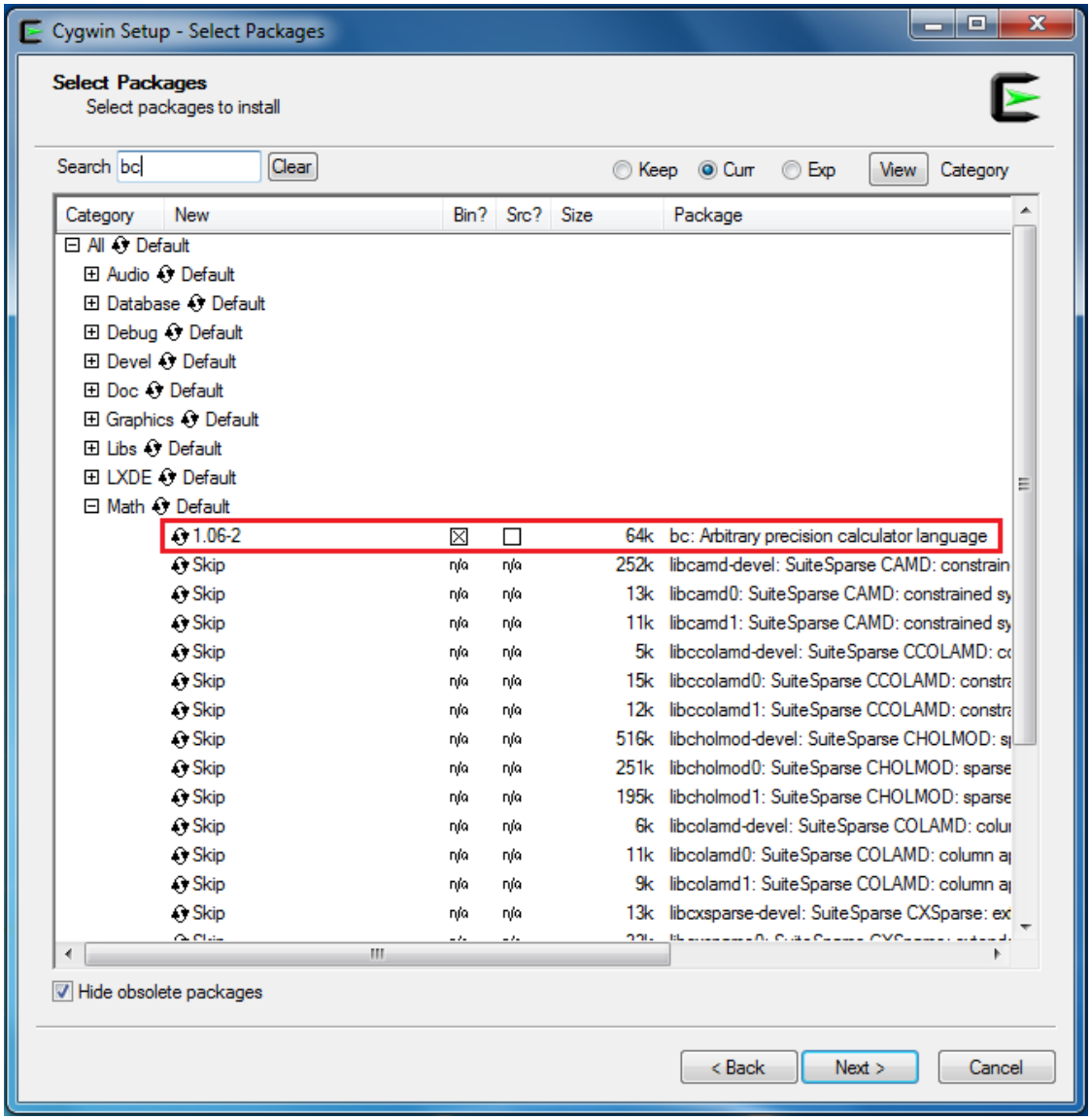
3.1 Getting Started

3.1.1 Setup of the GCC Development Environment

On Windows, you can use `Cygwin` as the GCC development environment. `Cygwin` is a large collection of GNU and open source tools which provide functionality similar to a Linux distribution on Windows.
Click http://cygwin.com/ and download the `Cygwin` package setup-x86.exe for your Windows platform.

1. 32-bit `Cygwin` is supported both for 32-bit Windows and 64-bit Windows.
2. During the installation of `Cygwin` package, include 'Devel -> make' and 'Math -> bc' utilities on the Select Packages page, as below shows.


Cygwin Setup - Select Packages
Select Packages
Select packages to install
Search make
Clear
Keep
Curr
Exp
View
Category
Category New Bin? Src? Size Package
All Default
Archive Default
Debug Default
Devel Default
Skip n/a n/a 3k automake: Wrapper scripts for automake and aclocal
Skip n/a n/a 714k automake1.10: (1.10) a tool for generating GNU-compliant
Skip n/a n/a 860k automake1.11: (1.11) a tool for generating GNU-compliant
Skip n/a n/a 733k automake1.12: (1.12) a tool for generating GNU-compliant
Skip n/a n/a 780k automake1.13: (1.13) a tool for generating GNU-compliant
Skip n/a n/a 581k automake1.14: (1.14) a tool for generating GNU-compliant
Skip n/a n/a 781k automake1.15: (1.15) a tool for generating GNU-compliant
Skip n/a n/a 248k automake1.4: (1.4) a tool for generating GNU-compliant M
Skip n/a n/a 332k automake1.5: (1.5) a tool for generating GNU-compliant M
Skip n/a n/a 365k automake1.6: (1.6) a tool for generating GNU-compliant M
Skip n/a n/a 426k automake1.7: (1.7) a tool for generating GNU-compliant M
Skip n/a n/a 499k automake1.8: (1.8) a tool for generating GNU-compliant M
Skip n/a n/a 557k automake1.9: (1.9) a tool for generating GNU-compliant M
Skip n/a n/a 3,675k cmake: Cross-platform makefile generation system
Skip n/a n/a 767k cmake-doc: Cross-platform makefile generation system (do
Skip n/a n/a 1,196k cmake-gui: Cross-platform makefile generation system (GU
Skip n/a n/a 163k extra-cmake-modules: Extra CMake Modules for KDE
Skip n/a n/a 420k gcc-tools-epoch1-automake: (gcc-special) a tool for genera
Skip n/a n/a 589k gcc-tools-epoch2-automake: (gcc-special) a tool for genera
Skip n/a n/a 6k gccmakedep: X Makefile dependency tool for GCC
Skip n/a n/a 34k imake: X Imake legacy build system
4.2.1-1 n/a 449k make: The GNU version of the 'make' utility
Skip n/a n/a 29k makedepend: X Makefile dependency tool
Hide obsolete packages
< Back
Next >
Cancel



Note: For Linux, refer to AN0400 Ameba-D Application Note v12.pdf to build the GCC development environment.

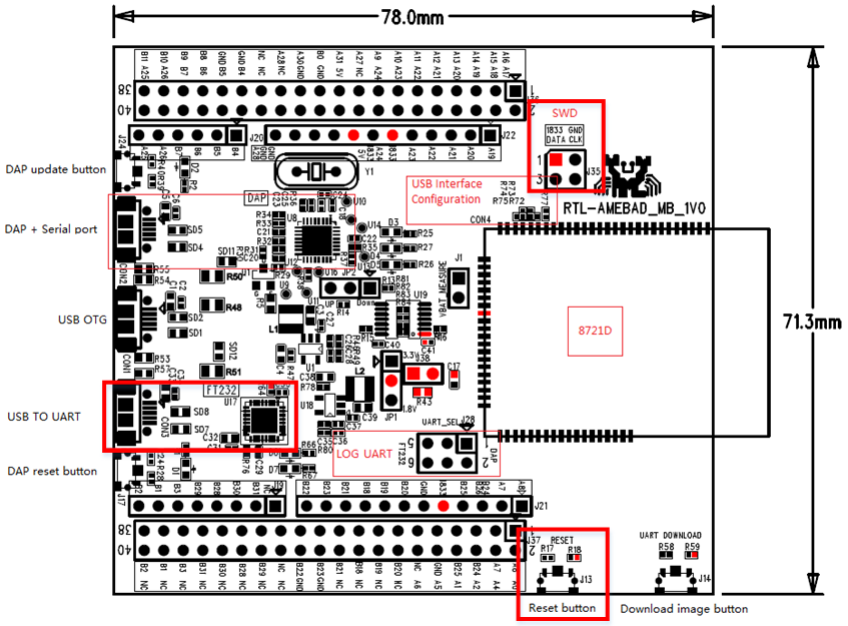
3.1.2 Knowledge about Ameba-D Demo Board

For Ameba-D, there are many types of chipsets available, such as RTL8720CS, RTL8721CSM, RTL8722CSM, RTL8720DN, RTL8720DM, RTL8721DM, and RTL8722DM.
In addition, the chipsets can be embedded on Ameba-D DEV demo board, which is extended to various I/O interfaces. The corresponding HDK (Hardware Development Kit) documents are available, please contact RTK for further details.

The hardware block diagram of Ameba-D demo board is shown below.

- USB TO UART: power supply and log print.
- The baudrate is 115200bps

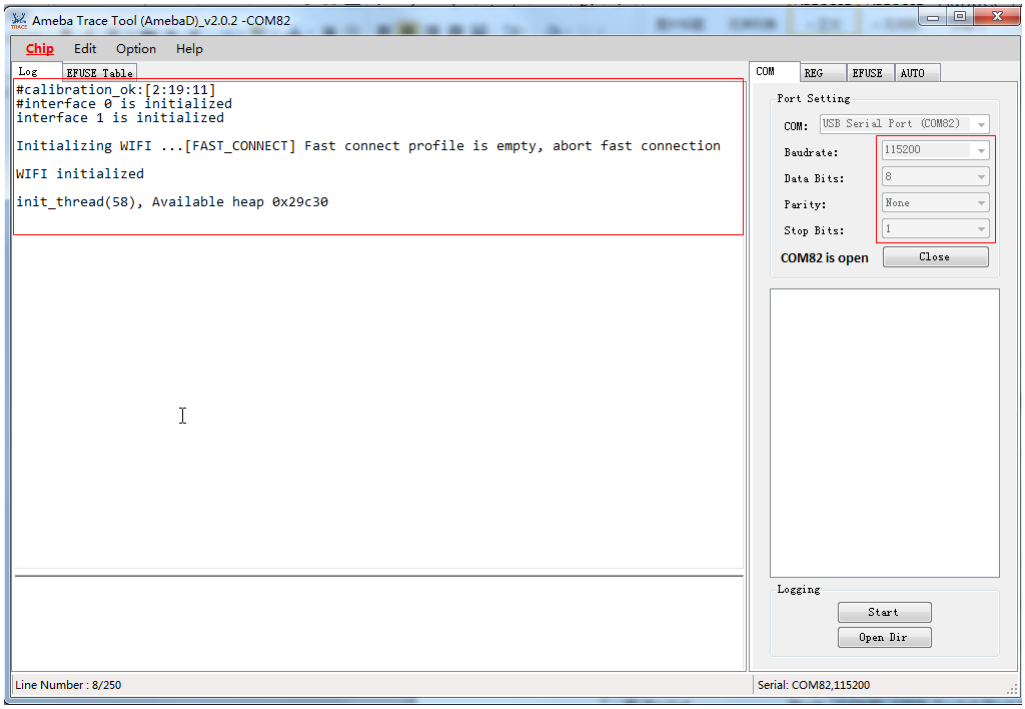
- SWD: SWD interface, used to download images and debug with IAR.
- Reset button: reset Ameba-D to run firmware after IAR completes download.



3.1.3 Connection to Log Console

On Ameba-D board, FTDI Chip and FT232 can be used for the log console and debugger. To view the log console, make use of the terminal tool, such as `SecureCRT`/`teraterm`/`putty` and etc. We will take our internal tool as an example.

1) Select the corresponding serial uart configure communicate parameter and then open it.
2) Press the Reset button on Ameba-D board. Some messages can be found in the terminal.



3.1.4 Building the First GCC Project on Ameba-D

The following steps are for first-time developer to build GCC project, under existing RTK SDK.

Building Code This section illustrates how to build SDK.

First, you need to switch to GCC project directory. For Windows, open `Cygwin` terminal and use `$ cd` command to change directory to KM0 or KM4 project directory of Ameba-D SDK.

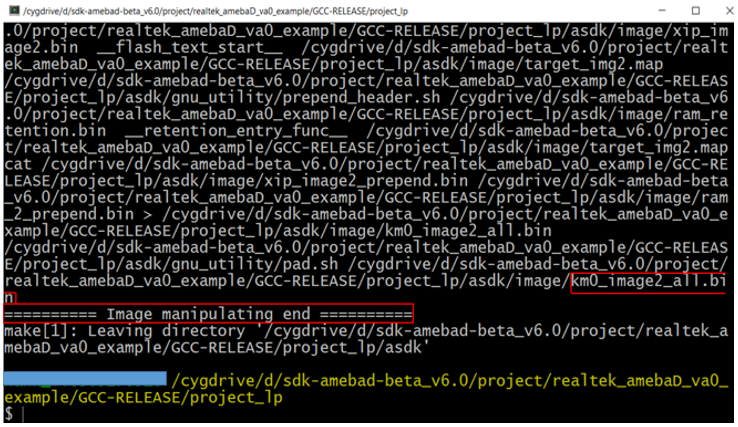
Note: You need to replace the `{path}` to your own SDK location, and add "`cygdrive`" prefix in front of the SDK location, so that Cygwin can access your file system.

- `$ cd /cygdrive/{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp`
- `$ cd /cygdrive/{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp`

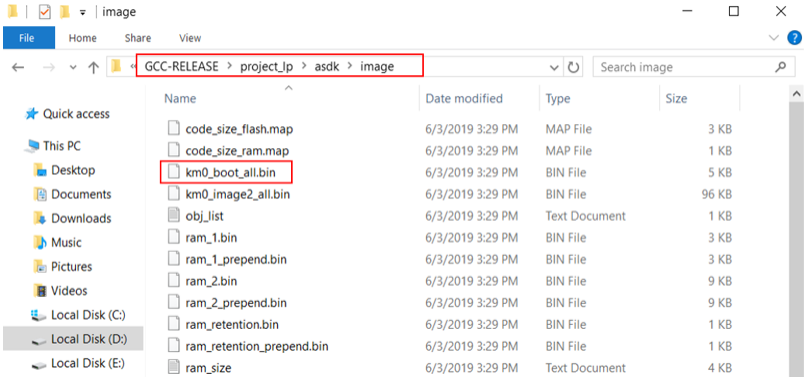
Linux, open its own terminal and use $ cd command to change directory to KM0 or KM4 project directory of Ameba-D SDK.

- `$ cd /{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp`
- `$ cd /{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp`

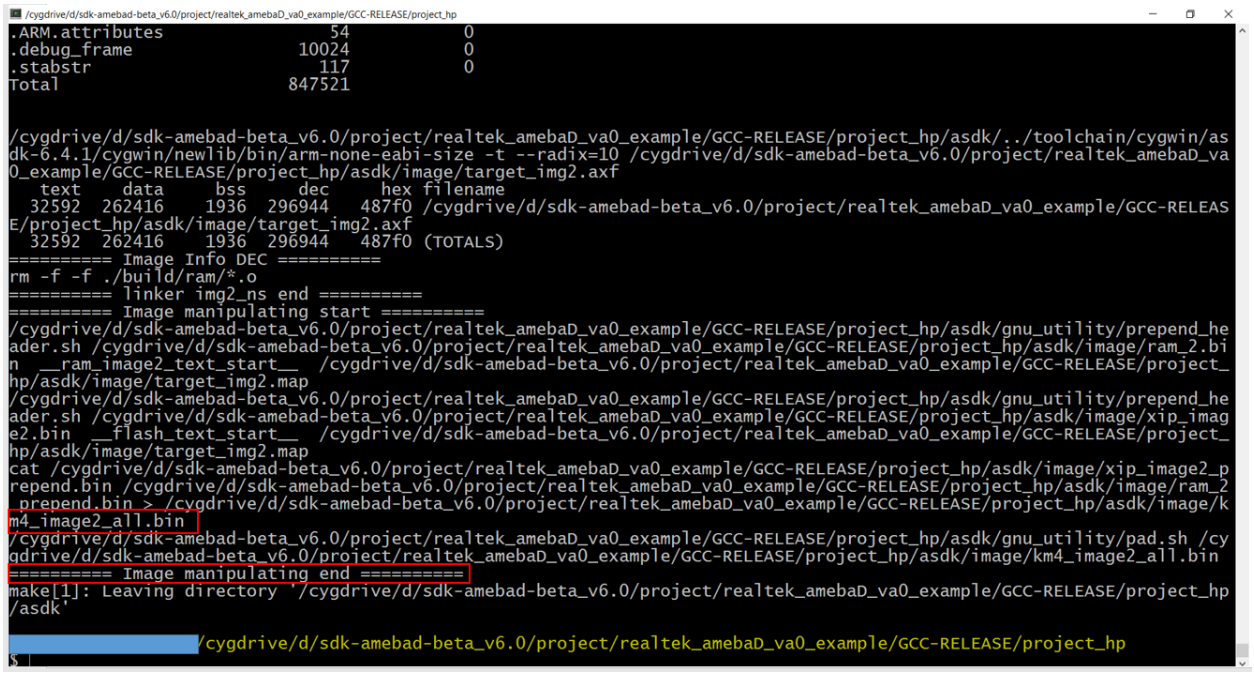
To build SDK for normal image, simply use `$ make all` command under the corresponding project directories on Cygwin (Windows) or terminal (Linux). KM0 project For KM0 project, if the terminal contains "`km0_image2_all.bin`" and "`Image manipulating end`" output message, it means that the image has been built successfully, as below shows.



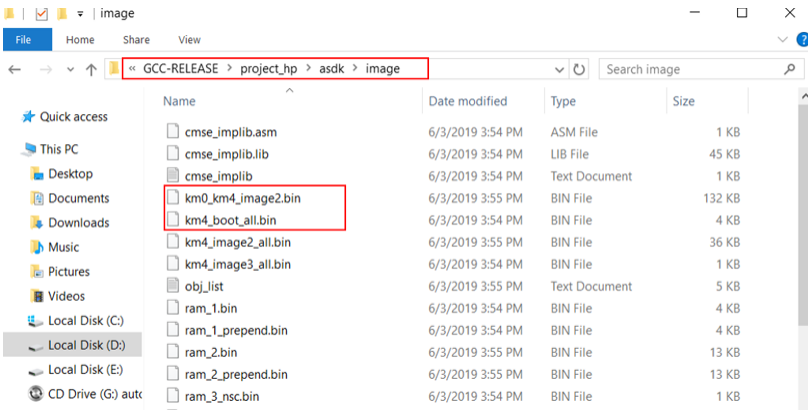
If somehow it is built failed, type `$ make clean` to clean and then redo the make procedure.
After successfully built, the image file is located in `project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image` , as below shows.



KM4 project For KM4 project, if the terminal contains "`km0_image2_all.bin`" and "`Image manipulating end`" output message, it means that the image has been built successfully, as below shows.



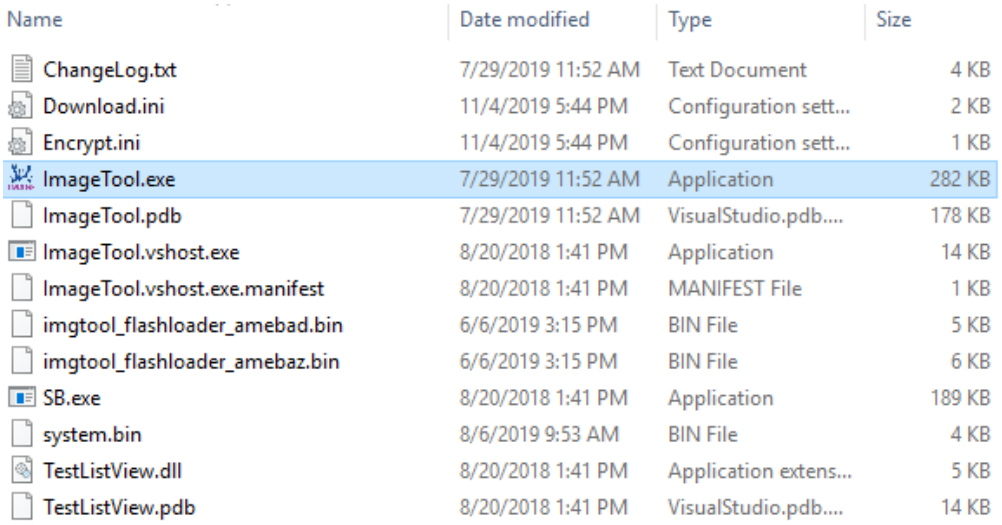
If somehow it built failed, type `$ make clean` to clean and then redo the make procedure.
After built successfully, the image file is located in `project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/image`, as below shows.



3.1.5 Downloading Images to Ameba-D

Realtek provides an image tool to download images on windows.

- Environment Requirements: EX. WinXP, Win 7 Above, Microsoft .NET Framework 3.5
- `ImageTool.exe` Location: `SDK\tools\AmebaD\Image_Tool\ImageTool.exe`



Name	Date modified	Type	Size
ChangeLog.txt	7/29/2019 11:52 AM	Text Document	4 KB
Download.ini	11/4/2019 5:44 PM	Configuration sett...	2 KB
Encrypt.ini	11/4/2019 5:44 PM	Configuration sett...	1 KB
ImageTool.exe	7/29/2019 11:52 AM	Application	282 KB
ImageTool.pdb	7/29/2019 11:52 AM	VisualStudio.pdb....	178 KB
ImageTool.vshost.exe	8/20/2018 1:41 PM	Application	14 KB
ImageTool.vshost.exe.manifest	8/20/2018 1:41 PM	MANIFEST File	1 KB
imgtool_flashloader_amebad.bin	6/6/2019 3:15 PM	BIN File	5 KB
imgtool_flashloader_amebaz.bin	6/6/2019 3:15 PM	BIN File	6 KB
SB.exe	8/20/2018 1:41 PM	Application	189 KB
system.bin	8/6/2019 9:53 AM	BIN File	4 KB
TestListView.dll	8/20/2018 1:41 PM	Application extens...	5 KB
TestListView.pdb	8/20/2018 1:41 PM	VisualStudio.pdb....	14 KB

Assuming that the ImageTool on PC is a server, it sends images files to Ameba (client) through UART. To download image from server to client, the client must enter uart download first.

1) **Enter into UART_DOWNLOAD mode.**

   - Push the UART DOWNLOAD button and keep it pressed.
   - Re-power on the board or press the Reset button.
   - Release the UART DOWNLOAD button.

   Now, Ameba board gets into UART_DOWNLOAD mode and is ready to receive data.

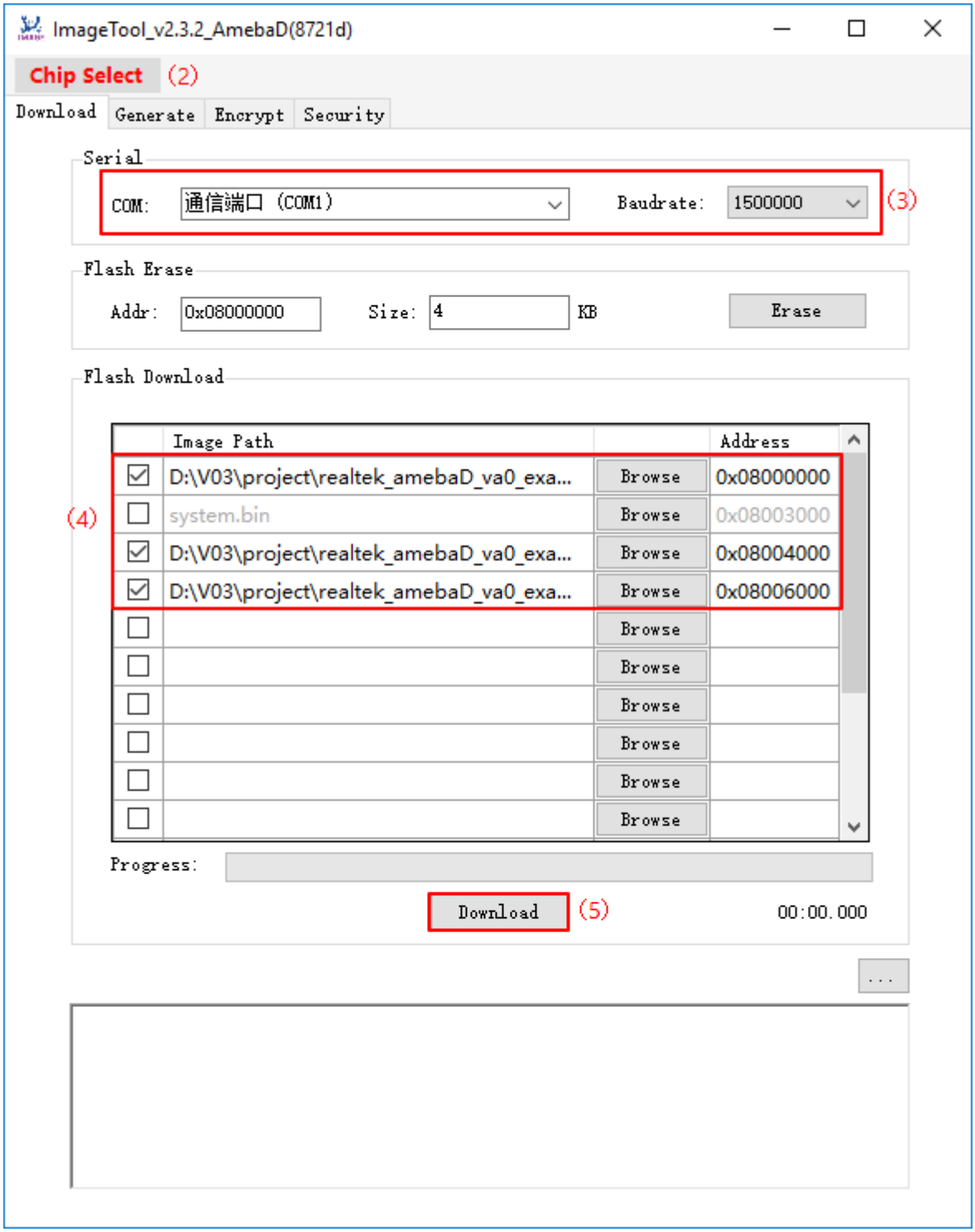
2) **Click Chip Select(in red) on UI and select chip (AmebaD or AmebaZ).**

3) **Select the corresponding serial port and transmission baud rate. The default baudrate is 1.5Mbps (recommended).**

4) **Click the Browse button to select the images (km0_boot_all.bin/km4_boot_all.bin/km0_km4_image2.bin) to be programmed and input addresses.**

   - The image path is located in:
     `{path}\project\realtek_amebaD_va0_example\GCC-RELEASE\project_lp\asdk\image`
     and
     `{path}\project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\image`,
     where {path} is the location of the project on your own computer.
   - The default target address is the SDK default image address, you can use it directly.

5) **Click Download button to start. The progress bar will show the transmit progress of each image. You can also get the message of operation successfully or errors from the log window.**


ImageTool_v2.3.2_AmebaD(8721d)
Chip Select
(2)
Download
Generate
Encrypt
Security
Serial
COM:
通信端口 (COM1)
Baudrate:
1500000
(3)
Flash Erase
Addr:
0x08000000
Size:
4
KB
Erase
Flash Download
Image Path
Address
(4)
D:\V03\project\realtek_amebaD_va0_exa...
Browse
0x08000000
system.bin
0x08003000
0x08004000
0x08006000
Progress:
Download
(5)
00:00.000
...

3.2 Release History

Version 1.0.0 – 2021/10/12

- Hardware information:
  - CPU. 32-bit KM4 (Arm Cortex-M33 compatible) and 32-bit KM0 (Arm Cortex-M23 compatible)
  - MEMORY. 512KB SRAM + 4MB PSRAM
- Feature:
  - Integrated 802.11a/n Wi-Fi SoC
  - USB Host/Device
  - SD Host
  - BLE5.0
  - Codec
  - LCDC
  - Key Matrix
  - 1 PCM interface
  - 4 UART interface
  - 1 I2S Interface
  - 2 I2C interface
  - 7 ADC
  - 17 PWM
  - Max 54 GPIO

3.3 Board HDK

AmebaD HDK Downlaod

3.3.1 EVB

- **HDK-AMEBAD_MB_4V2**
  - `Layout`
  - `Schematic`

- **HDK-AMEBAD_MB_4V0**
  - `Layout`
  - `Schematic`

3.3.2 RTL8722DM Module

- **HDK-AM8722DM01_6V2_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_6V1_WI LPF**
  - Layout
  - Schematic

- **HDK-AM8722DM01_4V1**
  - Layout
  - Schematic

3.4 Support

3.4.1 FAQ

Where to buy Ameba RTL8722DM Board?

Refer to Purchase link.

3.4.2 Trouble shooting

If you have driver issue of connect board to your computer?

Please go to https://ftdichip.com/drivers/ for USB driver.

CHAPTER

FOUR

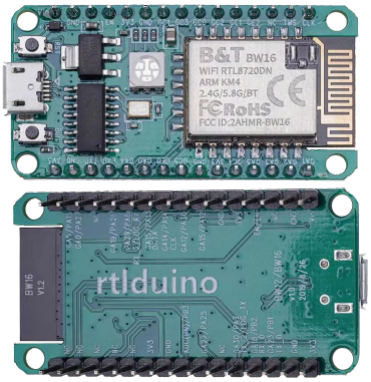
DOWNLOAD

4.1 AMB21/AMB22 (RTL8722DM)

4.2 AMB23 (RTL8722DM MINI)

4.3 BW16 (RTL8720DN) by Ai-Thinker

Table 1: Development / Evaluation Boards

		
AMB21(RTL8722DM)	*AMB23(RTL8722DM MINI)*	*BW16(RTL8720DN) by Ai-Thinker*

Tip:

We welcome contributions to the our SDKs documentations.
To make contributions, please visit our official GitHub Wiki page.