



## UM0166 Realtek RTL8710AF MQTT Example Guide

---

---

## Table of Contents

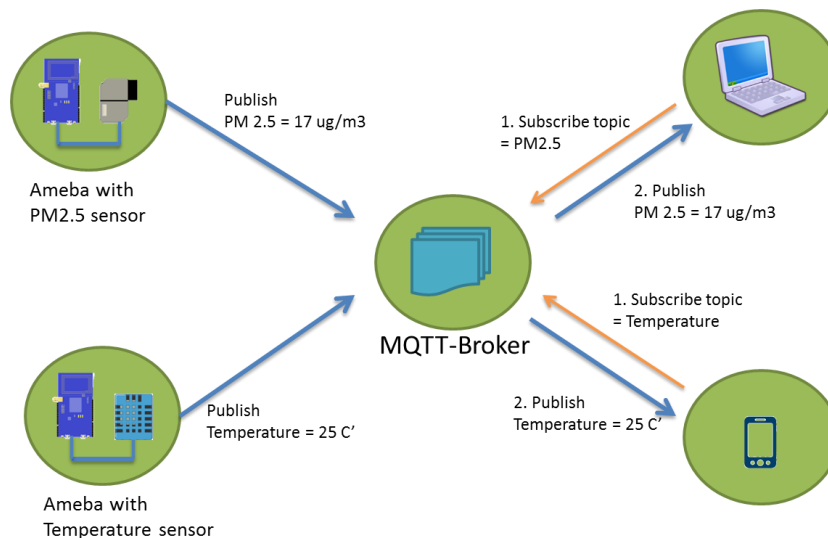
1	Introduction to MQTT .....	3
2	Example 1: Basic MQTT connect .....	4
2.1	Building and Running the Basic Example .....	4
2.1.1	Testing the MQTT functionality .....	7
3	Example 2: DHT Sensor with MQTT .....	11

# 1 Introduction 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.



In the operation of MQTT, there are several roles:

- **Publisher:** Usually publishers are the devices equipped with sensors (ex. Ameba). Publishers upload 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 categorize 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.

In order to demonstrate the working of MQTT we have prepared 2 examples.

1. Simple MQTT cloud connection to subscribe and publish messages using specific topics.
2. Connect the DHT11 sensor to the board and transmitting ambient temperature and humidity values from the sensor to the MQTT broker.

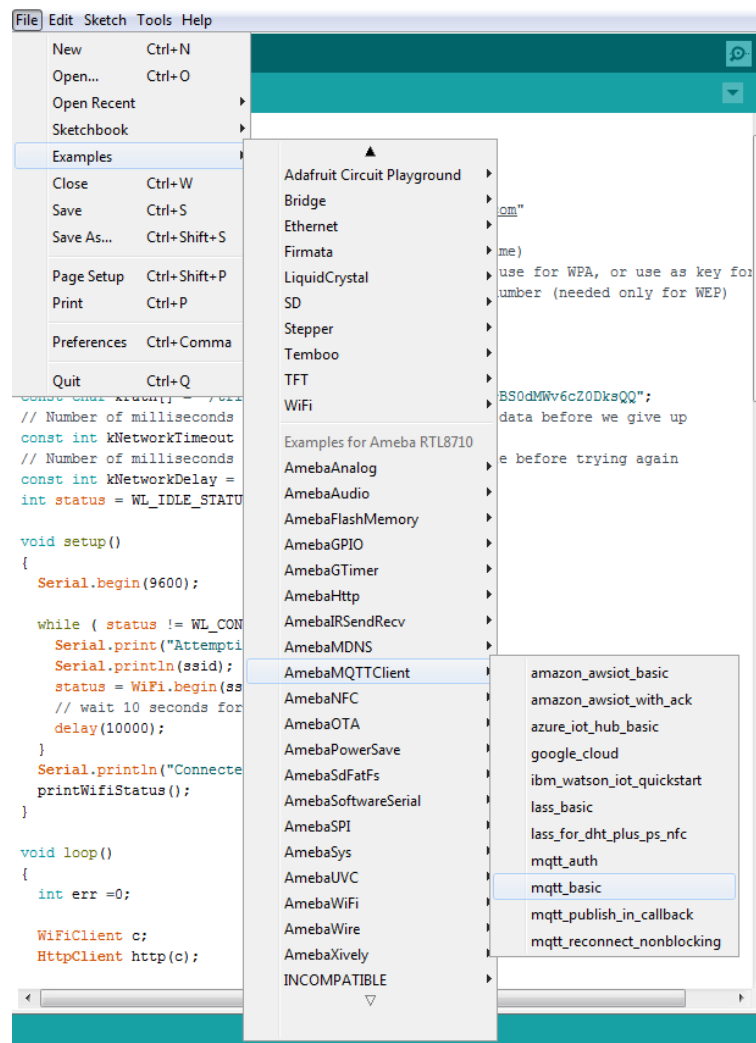
## 2 Example 1: Basic MQTT connect

In this example we demonstrate the basic functionality of how the example can communicate to an MQTT broker and publish and subscribe to topics and send data.

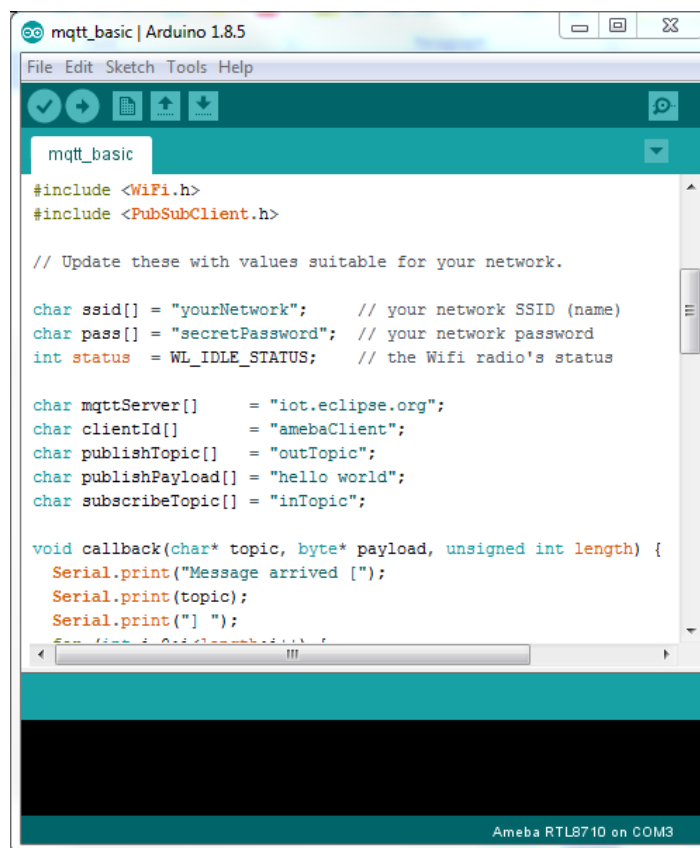
### 2.1 Building and Running the Basic Example

- This MQTT example can be found in the inbuilt suite of examples for the Realtek boards' package within the Arduino IDE as shown below.

Open the MQTT example: "File" -> "Examples" -> "Ameba MQTT Client" -> "mqtt\_basic"



- Please edit the Wi-Fi SSID and password in the example by filling in the SSID and password of the access point or the hotspot appropriately.
- It is to be noted that in the code there are certain fields for the MQTT application as shown below:
  - “mqttServer[]” : This is to specify the mqtt broker that we intend to connect to. In this case, we are connecting to the free mqtt broker provided by “iot.eclipse.org”
  - “clientId[]” : This is just a unique identifier for our mqtt client which is the board. This is an optional parameter.
  - “publishTopic[]” : This is the name of the topic that is being published from the device.
  - “publishPayload[]”: This is the payload data that is sent under the publish topic to the mqtt broker. This is the data that is sent with the publish topic.
  - “subscribeTopic[]”: This is the topic to which the board subscribes to. All messages that are published to the mqtt server with this topic are routed to the device automatically if it has subscribed to that particular topic.

A screenshot of the Arduino IDE interface. The title bar shows 'mqtt\_basic | Arduino 1.8.5'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening, saving, and running sketches. The main text area displays the code for 'mqtt\_basic'. The code includes headers for WiFi and PubSubClient, defines network credentials (ssid, pass, status), sets MQTT server, client ID, publish topic, publish payload, and subscribe topic, and defines a callback function for handling incoming messages. The status bar at the bottom indicates 'Ameba RTL8710 on COM3'.

```
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

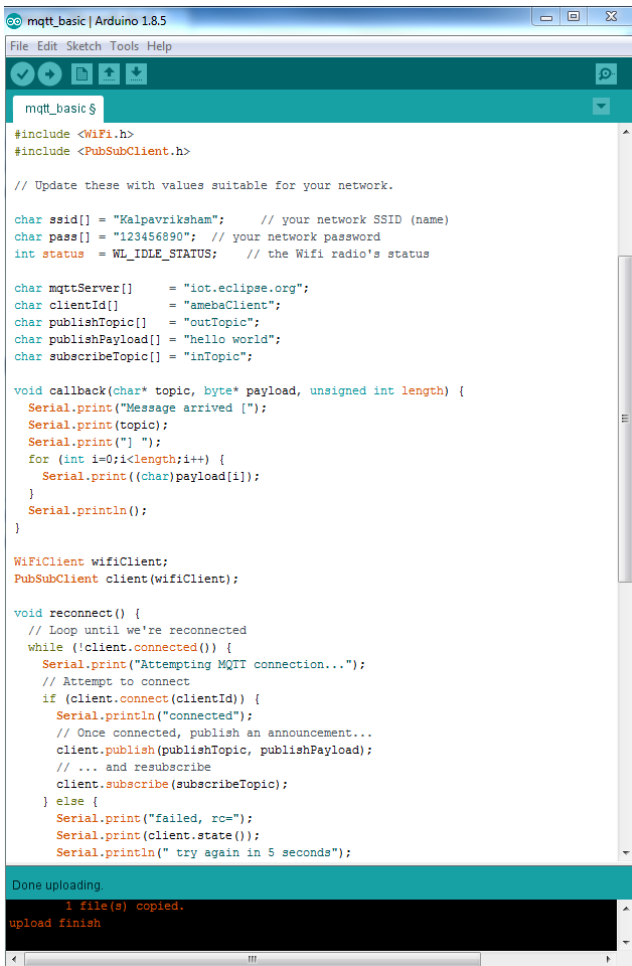
char mqttServer[] = "iot.eclipse.org";
char clientId[] = "amebaClient";
char publishTopic[] = "outTopic";
char publishPayload[] = "hello world";
char subscribeTopic[] = "inTopic";

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();
}
```

In this example we are using the standard “iot.eclipse.org” which is provided for free from eclipse for all to use. The server has both encrypted and unencrypted ports for different applications. (In

case you do not wish to use this server, the example can be configured to use any other MQTT broker.)

- Once the example is setup, compile the code. Click on “Sketch” -> “Verify/Compile” on Arduino. Upload (flashed) the code onto Ameba (During the flashing, component D3 which will be blinking on the board)
- Once the flashing is completed (as seen that the component D3 on the board will stop flashing), “upload finish” will be reflected as shown below.



The screenshot shows the Arduino IDE interface with the 'mqtt\_basic' sketch loaded. The code includes headers for WiFi and PubSubClient, defines network credentials and MQTT server details, and implements a callback function and a reconnect loop. The status bar at the bottom indicates 'Done uploading' and 'upload finish'.

```
mqtt_basic $  
#include <WiFi.h>  
#include <PubSubClient.h>  
  
// Update these with values suitable for your network.  
  
char ssid[] = "Kalpavriksham"; // your network SSID (name)  
char pass[] = "123456890"; // your network password  
int status = WL_IDLE_STATUS; // the Wifi radio's status  
  
char mqttServer[] = "iot.eclipse.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");  
    }  
  }  
}
```

Done uploading.  
1 file(s) copied.  
upload finish

- Press the “Reset” button on the board and use the serial monitor to see the output logs. A similar log will be seen as the one shown below stating “Attempting MQTT connection.. Connect to Server successful! connected.”

```

COM3
Message arrived [inTopic] Can you hear me?
<RTL8195A>=====
ROM Version: 0.3
Build ToolChain Version: gcc version 4.8.3 (Realtek ASDK-4.8.3pi Build 2003)
=====
Check boot type form eFuse
SPI Initial
Image1 length: 0x3a88, Image Addr: 0x10000bc8
Image1 Validate OK, Going jump to Image1
BOOT from Flash: YES
SPI calibration
Find the available window
Baud:2; auto_length:0; Delay start:0; Delay end:63
[SPIF Err]SpicNVMeCalStore: The flash memory(0x9088 = 0x0) is not able to be write, Err
SPI calibration
Find the available window
Baud:1; auto_length:11; Delay start:0; Delay end:63
[SPIF Err]SpicNVMeCalStore: The flash memory(0x90b8 = 0x0) is not able to be write, Err
OTA addr 0x0 INVALID

Load NEW fw 0
Flash Image2:Addr 0xb000, Len 239012, Load to SRAM 0x10006000
No Image3
Img2 Sign: RIKWin, InfaStart @ 0x10006049
===== Enter Image 2 =====
Attempting to connect to SSID: Kalpavriksham
Interface 0 is initialized
Interface 1 is initialized

Initializing WIFI ...
WIFI initialized

RTL8195A[Driver]: set ssid [Kalpavriksham]

RTL8195A[Driver]: start auth to 2e:0e:3d:83:fe:a4

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=1)

RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

RTL8195A[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1

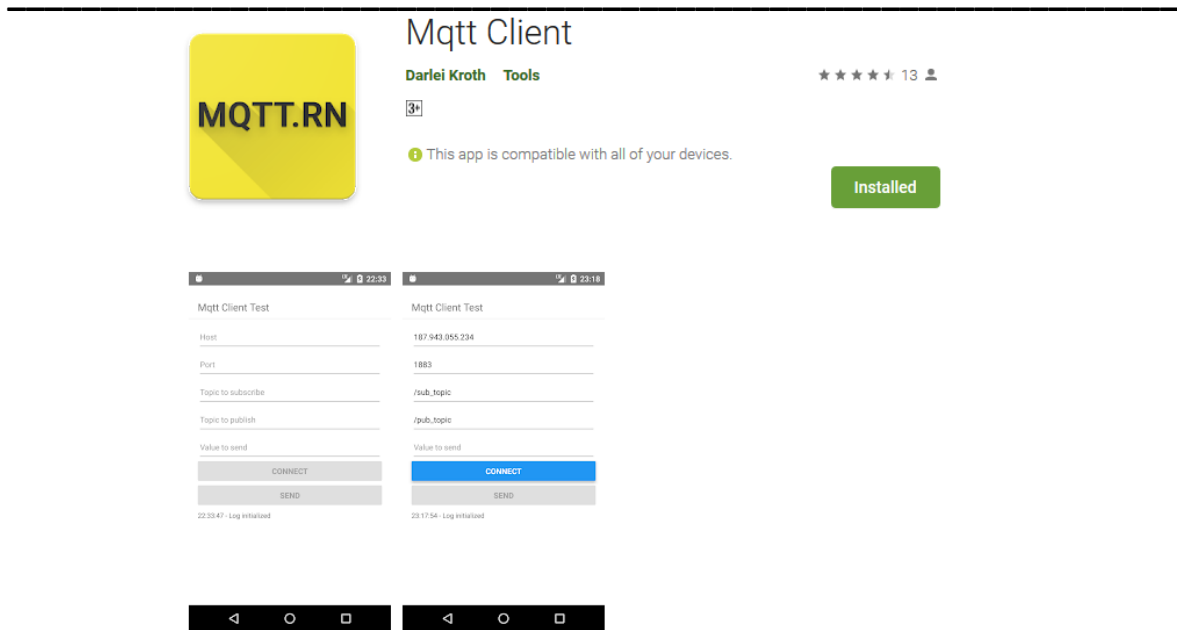
Interface 0 IP address : 192.168.43.247Attempting MQTT connection...
Connect to Server successful!
connected

```

Now it is to be noted that since we are connecting to a public MQTT server, there is a possibility that other people on the internet might publish messages with the same topic name as we have subscribed to and hence it is common to see stray messages appear on the console. In case you do not wish to get stray messages it is advised to use a private server.

### 2.1.1 Testing the MQTT functionality

Once the MQTT example is flashed, we need to test it with another MQTT client. Since MQTT is an M2M(Machine to Machine) protocol, another client needs to be in place to test the subscribing and publishing of the messages. In case you have another MQTT capable device, it can be used to publish and subscribe to the topics used in this example and test. In case there is no MQTT capable device available, the MQTT functionality can be tested using a mobile app. There are many such apps available for both Android as well as IOS. For simplicity we shall be using the “MqttClient” app available on the play store as shown below:

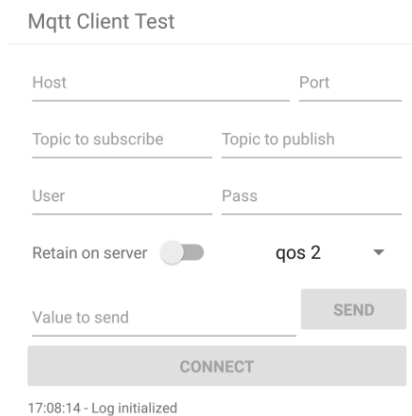


(For IOS users, one can use the app called “Mqttt” on the App Store made by “Chenhsin Wong” which can be downloaded from the link: <https://itunes.apple.com/in/app/mqttt/id1217080708?mt=8>)

Disclaimer: It is to be noted that these mobile apps are 3<sup>rd</sup> party softwares and Realtek takes no responsibility to the availability/reliability and/or accuracy of these services. It is the user’s responsibility to find a suitable client to test his/her application and the 3<sup>rd</sup> party softwares referred to in this document are only a reference.

In order to test the MQTT example using the MQTTClient app, the following needs to be done.

- Download and install the application from the play store in android phones.
- Once the app is downloaded, the interface looks as follows:



- The MQTT broker URL needs to be entered in the “Host” field and the port needs to be specified. In this case the “Topic to subscribe” needs to be the topic that is being published from the board



and the “Topic to publish” needs to be the topic that the board subscribes to, only then can an 2 way communication via MQTT be setup.

- Enter the following to the respective fields
  - Host : iot.eclipse.org
  - Port : 1883
  - Topic to subscribe : outTopic
  - Topic to publish : inTopic
- The configuration for this example is as shown below.

Mqtt Client Test

iot.eclipse.org 1883

outTopic inTopic

User Pass

Retain on server ☐ qos 2 ▼

Value to send

17:40:49 - Log initialized

- The username and password fields can be left empty as we are using an open server to test the MQTT functionality. In case a closed server is being used, the username and password can be used.
- In case only one action either publish or subscribe is being done, the app doesn't allow us to connect, in this scenario it is advised to just fill in a random topic for the feature that isn't being used and enable the connection.
- Once all the configurations are done, press “CONNECT” to initiate a connection to the server.
- It is to be noted here that some infrastructure Wi-Fi connections and some Access Points deliberately block MQTT servers, it is essential to use an internet connection that allows you to connect to MQTT servers in order to test using these types of applications. It is advised to use mobile data in order to perform these tests.
- Once all configurations are set and the app connects to the MQTT server, the connect icon changes to “DISCONNECT” as shown below.

### Mqtt Client Test

iot.eclipse.org	1883
outTopic	inTopic
User	Pass
Retain on server <input type="checkbox"/>	qos 2 ▼
Value to send	<b>SEND</b>
<b>DISCONNECT</b>	

17:41:43 - connected  
17:41:41 - Log initialized

- As seen in the code in the example, the publish payload from the board is “Hello World” and hence when the board publishes the topic, we are able to see “Hello World” printed on the console.

### Mqtt Client Test

iot.eclipse.org	1883
outTopic	inTopic
User	Pass
Retain on server <input type="checkbox"/>	qos 2 ▼
Value to send	<b>SEND</b>
<b>DISCONNECT</b>	

17:41:47 - hello world

- In order to send any data to the board we can enter the text we wish to send in the “Value to send” field and press the “Send” button as shown below.

### Mqtt Client Test

iot.eclipse.org	1883
outTopic	inTopic
User	Pass
Retain on server <input type="checkbox"/>	qos 2 ▼
hello ameba	<b>SEND</b>
<b>DISCONNECT</b>	

17:50:58 - hello world

- Once you hit send, the text you type(in this case hello ameba is types) is sent as the payload along with the topic that specified and since the same topic is being subscribed by the board it will be received and printed on the serial monitor as shown on the screen below: “message arrived[inTopic] hello ameba”

```
Attempting to connect to SSID: IOT_MERCURY_59E0
interface 0 is initialized
interface 1 is initialized

Initializing WIFI ...
WIFI initialized

RTL8195A[Driver]: set ssid [IOT_MERCURY_59E0]

RTL8195A[Driver]: start auth to e4:f3:f5:22:59:e0

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=0)

RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

RTL8195A[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:

Interface 0 IP address : 192.168.90.103Attempting MQTT connection...
Connect to Server successful!
connected
Message arrived [inTopic] Don't be sad. Just improve yourself!
Message arrived [inTopic] hello ameba
```

- Press “Disconnect” on the mqtt mobile app to stop the connection activity

### 3 Example 2: DHT Sensor with MQTT

This example details how to connect the DHT sensor to the Realtek board and read the temperature and humidity data and transmit the results to an MQTT broker.

The code to run this particular example is placed inside the folder of the Github repository: “MQTT\_DHT\_POST”. Load the example on the Arduino IDE.

- Disconnect the Ameba board from the laptop/computer
- Open the Example and edit the SSID and password to the Wi-Fi connection that you are connecting to.

```

MQTT_DHT_POST
#include <PubSubClient.h>
#include "DHT.h"
#include <avr/dtostrf.h>

#define DHTPIN 2 // what digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11

#define BOARD_ID "ameba_4000"

// Update these with values suitable for your network.

char ssid[] = "IOT_MERCURY_S9E0"; // your network SSID (name)
char pass[] = "1234567890"; // your network password
int status = WL_IDLE_STATUS; // the Wifi radio's status

char mqttServer[] = "iot.eclipse.org";

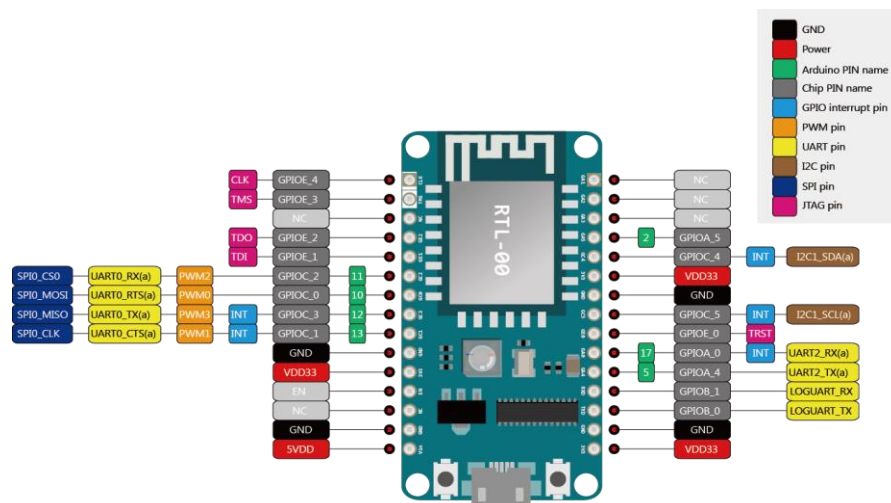
char clientId[] = BOARD_ID;

WiFiClient wifiClient;
PubSubClient client(wifiClient);

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
  }
}

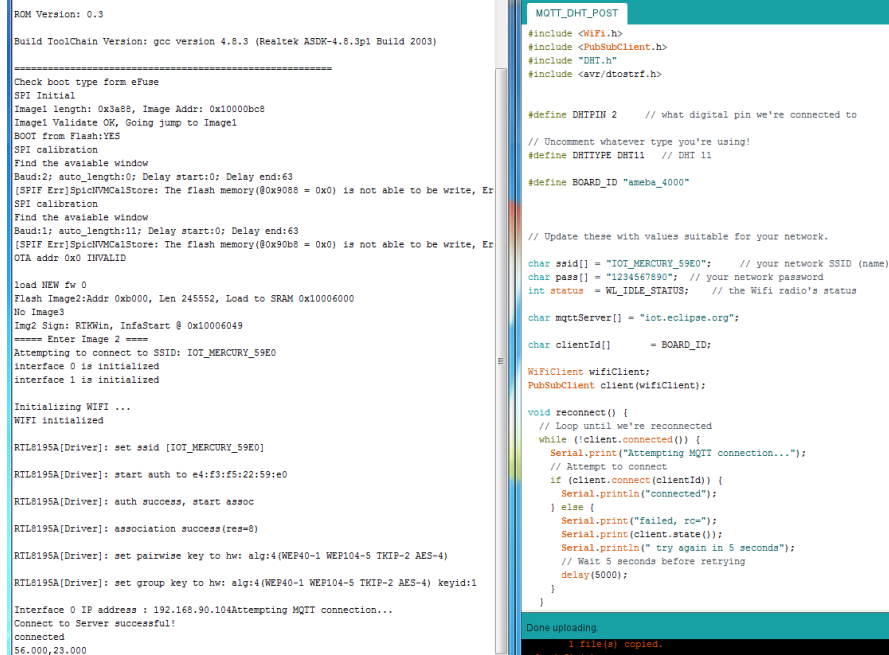
```

- Once this is done connect the DHT sensor to the physical pins on the board.
- It is to be noted that the 3 pins of the DHT sensor should be connected to the Realtek board as detailed below:
  - VCC→3V3
  - GND→GND
  - DAT→GA5
- It is to be noted that inside the code we have defined the DHTPIN as 2. This is because in the actual pinout of the RTL8710AF board, the GPIOA\_5 is mapped to the Arduino pin 2 as shown in the pinout diagram below.



- In this example we are using the DHT 11 sensor and hence the DHTYPE define is set to "DHT11"
- Connect the Ameba board to the computer via Micro USB. Verify and upload (flashed) the code onto Ameba.

- Reset the board. Once the board resets it will connect to the Wi-Fi and start sending the temperature and humidity values over MQTT to the broker at an interval of 10 seconds. It can be observed in the serial monitor as shown below.



The left screenshot shows the serial monitor output of the sketch. It displays the boot process, including SPI initialization, flash memory checks, and Wi-Fi connection attempts. The output shows the board successfully connecting to the Wi-Fi network 'IOT\_MERCURY\_59E0' and attempting to connect to the MQTT broker at 192.168.90.104.

```
ROM Version: 0.3
Build ToolChain Version: gcc version 4.8.3 (Realtek ASMC-4.8.3pl Build 2003)

=====
Check boot type from eFuse
SPI Initial
Image1 length: 0x3a88, Image Addr: 0x10000bc8
Image1 Validate OK, Going jump to Image1
BOOT from Flash: YES
SPI calibration
Find the available window
Baud1: auto_length:0; Delay start:0; Delay end:63
(SPIF Err)SpiCVMCalStore: The flash memory(0x90b8 = 0x0) is not able to be write, Er
SPI calibration
Find the available window
Baud1: auto_length:11; Delay start:0; Delay end:63
(SPIF Err)SpiCVMCalStore: The flash memory(0x90b8 = 0x0) is not able to be write, Er
OTA addr 0x0 INVALID

load NEW fw 0
Flash Image2:Addr 0xb000, Len 245552, Load to SRAM 0x10006000
No Image3
Img2 Sign: RTNWin, InfoStart @ 0x10006049
===== Enter Image 2 =====
Attempting to connect to SSID: IOT_MERCURY_59E0
Interface 0 is initialized
Interface 1 is initialized

Initializing WIFI ...
WIFI initialized

RTL8195A[Driver]: set ssid [IOT_MERCURY_59E0]
RTL8195A[Driver]: start auth to e4:f3:ff:22:59:e0
RTL8195A[Driver]: auth success, start assoc
RTL8195A[Driver]: association success(res=0)
RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)
RTL8195A[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1
Interface 0 IP address : 192.168.90.104Attempting MQTT connection...
Connect to Server successful!
connected
56.000,23.000
```

The right screenshot shows the source code for the `MQTT_DHT_POST` sketch. It includes headers for `WiFi`, `PubSubClient`, and `DHT`. It defines the digital pin for the DHT sensor and the DHT type. The code sets up the Wi-Fi network and the MQTT broker. It defines the MQTT client ID as `BOARD_ID`, which is set to `"ameba_4000"`. The `reconnect` function is defined to loop until the client is connected. The `void setup` function initializes the Wi-Fi and the MQTT client. The `void loop` function publishes the temperature and humidity values to the MQTT broker at an interval of 10 seconds.

```
MQTT_DHT_POST

#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <avr/dtostrf.h>

#define DHTPIN 2 // what digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11

#define BOARD_ID "ameba_4000"

// Update these with values suitable for your network.

char ssid[] = "IOT_MERCURY_59E0"; // your network SSID (name)
char pass[] = "1234567890"; // your network password
int status = WL_IDLE_STATUS; // the Wifi radio's status

char mqttServer[] = "iot.eclipse.org";

char clientID[] = BOARD_ID;

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");
    } 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() {
  // Initialize serial port
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect
  }
  // Initialize Wi-Fi
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(5000);
    Serial.print(".");
  }
  // Initialize MQTT client
  client.setServer(mqttServer, 1883);
  reconnect();
}

void loop() {
  // Publish temperature and humidity values
  float temp = dht.temperature();
  float hum = dht.humidity();
  String msg = String(temp) + "," + String(hum);
  client.publish(clientID, msg);
  delay(10000);
}
```

- In order to reduce the amount of data being sent over MQTT, the Humidity in Percentage(%) and the Temperature in Celsius(degC) are transmitted as comma separated values. The data can be formatted as per requirement in different applications within the code.
- In order to view the MQTT results, the `MqttClient` software can be used as specified in 2.1.1.
- It is to be noted in this example that the topic to be subscribed to is the clientID, since the data is being published with the clientID as the publish topic in this case the clientID is mapped to the `"BOARD_ID"` which is the string `"ameba_4000"`.