Raspberry Pi MQTT setup:

URL: https://iot4beginners.com/mosquitto-mqtt-broker-on-raspberry-pi/

Installing

Installing Mosquitto MQTT on Raspberry Pi

First update the operating system on your Raspberry Pi:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

Now, open a terminal and type the following command:

```
sudo apt install -y mosquitto mosquitto-clients
```

You can test your installation using the following command:

```
mosquitto -v
```

This will return the version of Mosquitto MQTT installed on your Raspberry Pi.

For the rest of this tutorial, you shall need the IP address of your Raspberry Pi. Type the following command and note down the IP address:

```
hostname -I
```

Setup Broker

Creating an MQTT Broker on Raspberry Pi

The easiest way to understand this protocol is to create a broker on Raspberry Pi and use it to publish and subscribe to topics. Mosquitto MQTT provides a layer of security that authorizes only specific clients to publish or subscribe to topics. For this, we need to set up a username and password. This step is optional, however, it is recommended to use it in all your projects.

Type the following command:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

The last line in the file will be:

```
include_dir /etc/mosquitto/conf.d
```

Remove this line and add the following lines at the end of the file:

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883
```

The above three lines will tell the broker, listening on port 1883, to prevent any communications from devices that do not have a valid username and password.

Outcomment

#***allow_anonymous*** and #***password_file***

if you don’t use usernames and password for login.

```
sudo mosquitto_passwd -c /etc/mosquitto/pwfile username
```

Type the above command in a terminal window. Replace "**username**" with your username. You will be prompted to enter a password. Type a password and press Enter.

Finally, reboot the Pi for the changes to take effect.

```
sudo reboot
```

sudo mosquitto_passwd -c /etc/mosquitto/pwfile kokojuice2

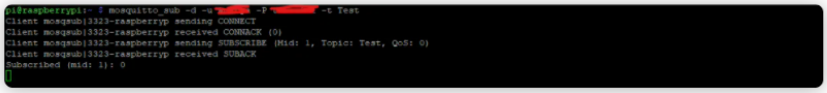
Password: kokojuice123
Reenter password: kokojuice123

Subscribe to topic

Subscribe to a Topic

Open a terminal and type the following command:

```
mosquitto_sub -d -u username -P password -t Test
```



In the command, replace **username** and **password** with the username and password you created before. In case that step was skipped, just type the following command:

```
mosquitto_sub -d -t Test
```

We have successfully subscribed to our **Test** topic. Now we have to publish a message to this topic.

mosquitto_sub -d -u kokojuice2 -P kokojuice123 -t Test

Publish to topic

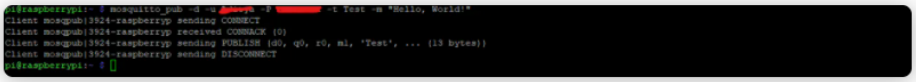
Publish a message to a Topic

To publish a message to a topic, type the following command:

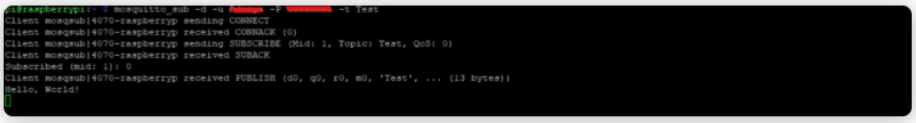
```
mosquitto_pub -d -u username -P password -t Test -m "Hello, World!"
```

Replace **username** and **password** with you username and password. In case that step was skipped, simply type the following command:

```
mosquitto_pub -d -t Test -m "Hello, World!"
```



Publish Window



Subscribe Window

mosquitto_pub -d -u kokojuice2 -P kokojuice123 -t Test
-m "This is a test if MQTT works!"

Created by Joachim Pham

Get it working with ESP32

More info : https://joy-it.net/files/files/Produkte/SBC-NodeMCU-ESP32/SBC-NodeMCU-ESP32-Manual-20200320.pdf

Next up we want Temperature sent through MQTT, with ESP32 and receiving with Raspberry Pi.

Using the **DHT11** to get Temperature and humidity data

Libraries needed:

- ***Adafruit BME280*** Library in Library Manager **Arduino IDE**

- ***Adafruit Sensor*** Library

Tutorial for other DHT's:

https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/

Problem with uploading?

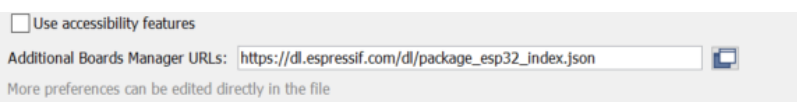
Push **BOOT**-button at uploading program

Check **Port**

Check **Board → to add ESP32:**

File – Preference – Additional Boards –

https://dl.espressif.com/dl/package_esp32_index.json



Restart IDE

Check **Baudrate** = 115200

Attention! After the initial installation, tha board rate may have changed to *921600*. This could cause problems. In such a case, set the baud rate to *115200* to avoid any problems.

With this code, ESP can read DHT.

Next, combine MQTT with DHT and ESP:

https://randomnerdtutorials.com/esp32-mqtt-publish-bme280-arduino/

First download Libraries:

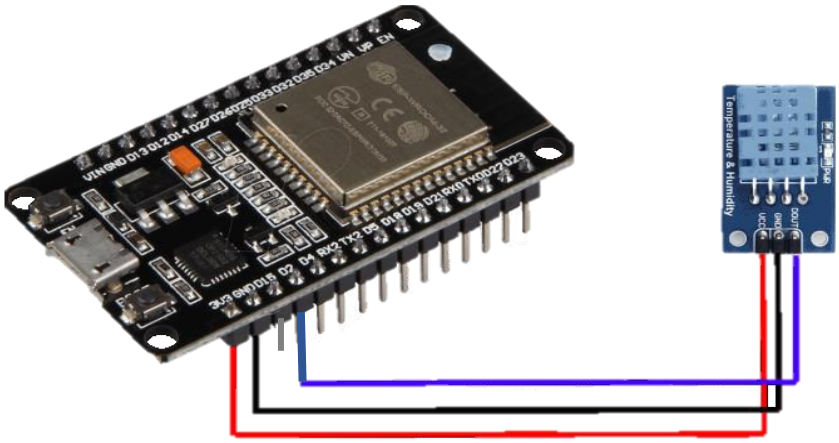
- ***Async MQTT Client*** Library

- ***Async TCP*** Library

ESP32_DHT11

```
#include <DHT.h>
#include <DHT_U.h>
#include <Adafruit_Sensor.h>



#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
Serial.begin(9600);
Serial.println(F("DHT11 test!"));

dht.begin();
}

void loop() {
delay(2000);

float h = dht.readHumidity();
float t = dht.readTemperature();
float f = dht.readTemperature(true);   // in Fahrenheit

if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("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 (isFahrenheit = false)
float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

Wiring DHT to ESP32:



Raw code accessible on:

https://github.com/Kokojuice2/MQTT-esp32-DHT.git

FullyFeatured-ESP32_DHT_MQTT

```
/*
This example uses FreeRTOS softwaretimers as there is no built-in Ticker library
*/

    // Requirements:

// DHT --------------------------
#include <DHT.h>
#include <DHT_U.h>
#include <Adafruit_Sensor.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float h = dht.readHumidity();
float t = dht.readTemperature();
//float f = dht.readTemperature(true);   // in Fahrenheit

float temp, hum;


// WIFI -------------------------
#include <WiFi.h>
extern "C" {
  #include "freertos/FreeRTOS.h"
  #include "freertos/timers.h"
}

#define WIFI_SSID "G Phone"                    // ENTER your wifi SSID
#define WIFI_PASSWORD "Easy1234*"                // ENTER your wifi password


// MQTT --------------------------
#include <AsyncMqttClient.h>

#define MQTT_HOST IPAddress(172, 20, 10, 5)
#define MQTT_PORT 1883

AsyncMqttClient mqttClient;
TimerHandle_t mqttReconnectTimer;
TimerHandle_t wifiReconnectTimer;

// Temperature MQTT Topics
#define MQTT_PUB_TEMP "esp32/temperature"
#define MQTT_PUB_HUM "esp32/humidity"
//#define MQTT_PUB_PRES "esp32/pressure"


//-----------------------------------

unsigned long previousMillis = 0;   // Stores last time temperature was published
const long interval = 10000;        // Interval at which to publish sensor readings

//_______________void's_________________________________

void connectToWifi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}

void connectToMqtt() {
  Serial.println("Connecting to MQTT...");
  mqttClient.connect();
}

void WiFiEvent(WiFiEvent_t event) {
    Serial.printf("[WiFi-event] event: %d\n", event);
    switch(event) {
    case SYSTEM_EVENT_STA_GOT_IP:
        Serial.println("WiFi connected");
        Serial.println("IP address: ");
        Serial.println(WiFi.localIP());
        connectToMqtt();
        break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
        Serial.println("WiFi lost connection");
        xTimerStop(mqttReconnectTimer, 0); // ensure we don't reconnect to MQTT while reconnecting to Wi-Fi
    xTimerStart(wifiReconnectTimer, 0);
        break;
    }
}

void onMqttConnect(bool sessionPresent) {
  Serial.println("Connected to MQTT.");
  Serial.print("Session present: ");
  Serial.println(sessionPresent);

  uint16_t packetIdSub = mqttClient.subscribe("test/lol", 2);
  Serial.print("Subscribing at QoS 2, packetId: ");
  Serial.println(packetIdSub);

  mqttClient.publish("test/lol", 0, true, "test 1");
  Serial.println("Publishing at QoS 0");

  // 1st publish
  uint16_t packetIdPub1 = mqttClient.publish("test/lol", 1, true, "test 2");
```

Created by Joachim Pham

```
  Serial.print("Publishing at QoS 1, packetId: ");
  Serial.println(packetIdPub1);
  // 2th publish
  uint16_t packetIdPub2 = mqttClient.publish("test/lol", 2, true, "test 3");
  Serial.print("Publishing at QoS 2, packetId: ");
  Serial.println(packetIdPub2);

}

void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
  Serial.println("Disconnected from MQTT.");

  if (WiFi.isConnected()) {
    xTimerStart(mqttReconnectTimer, 0);
  }
}

void onMqttSubscribe(uint16_t packetId, uint8_t qos) {
  Serial.println("Subscribe acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
  Serial.print("  qos: ");
  Serial.println(qos);
}

void onMqttUnsubscribe(uint16_t packetId) {
  Serial.println("Unsubscribe acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
}

void onMqttMessage(char* topic, char* payload, AsyncMqttClientMessageProperties properties, size_t len, size_t index, size_t total) {
  Serial.println("Publish received.");
  Serial.print("  topic: ");
  Serial.println(topic);
  Serial.print("  qos: ");
  Serial.println(properties.qos);
  Serial.print("  dup: ");
  Serial.println(properties.dup);
  Serial.print("  retain: ");
  Serial.println(properties.retain);
  Serial.print("  len: ");
  Serial.println(len);
  Serial.print("  index: ");
  Serial.println(index);
  Serial.print("  total: ");
  Serial.println(total);
}

void onMqttPublish(uint16_t packetId) {
  Serial.println("Publish acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
}

void setup() {
  Serial.begin(115200);
  Serial.println();

  dht.begin();

  connectToWifi();
  Serial.println("Wifi ready?");

  mqttReconnectTimer = xTimerCreate("mqttTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToMqtt));
  wifiReconnectTimer = xTimerCreate("wifiTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToWifi));

  WiFi.onEvent(WiFiEvent);

  mqttClient.onConnect(onMqttConnect);
  mqttClient.onDisconnect(onMqttDisconnect);
  mqttClient.onSubscribe(onMqttSubscribe);
  mqttClient.onUnsubscribe(onMqttUnsubscribe);
  mqttClient.onMessage(onMqttMessage);
  mqttClient.onPublish(onMqttPublish);
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  // If your broker requires authentication (username and password), set them below
  mqttClient.setCredentials("kokojuice2", "kokojuice123");        //<-- Username and pswd MQTT

}

void loop() {


  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);   // in Fahrenheit


  if (isnan(h) || isnan(t)) {
  Serial.println(F("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 (isFahrenheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));

  delay(2000);

  unsigned long currentMillis = millis()+ 2000;
  // Every X number of seconds (interval = 10 seconds)
  // it publishes a new MQTT message
  if (currentMillis - previousMillis >= interval) {
    // Save the last time a new reading was published
    previousMillis = currentMillis;
    // New BME280 sensor readings
    temp = dht.readTemperature();
    //temp = 1.8*bme.readTemperature() + 32;
    hum = dht.readHumidity();
    //pres = bme.readPressure()/100.0F;

    // Publish an MQTT message on topic esp32/BME2800/temperature
    uint16_t packetIdPub1 = mqttClient.publish(MQTT_PUB_TEMP, 1, true, String(temp).c_str());
    Serial.printf("Publishing on topic %s at QoS 1, packetId: %i", MQTT_PUB_TEMP, packetIdPub1);
    Serial.print("");
    Serial.printf("Message: %.2f \n", temp);

    // Publish an MQTT message on topic esp32/BME2800/altitude
    uint16_t packetIdPub2 = mqttClient.publish(MQTT_PUB_HUM, 1, true, String(hum).c_str());
    Serial.printf("Publishing on topic %s at QoS 1, packetId %i: ", MQTT_PUB_HUM, packetIdPub2);
    Serial.print("");
    Serial.printf("Message: %.2f \n", hum);

    // Publish an MQTT message on topic esp32/BME2800/pressure

    //uint16_t packetIdPub3 = mqttClient.publish(MQTT_PUB_PRES, 1, true, String(pres).c_str());
    //Serial.printf("Publishing on topic %s at QoS 1, packetId: %i", MQTT_PUB_PRES, packetIdPub3);
    //Serial.printf("Message: %.3f \n", pres);
  }
}
```

and get this in **Serial Monitor**:

```
13:57:58.214 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:00.255 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:02.295 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:04.294 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:04.327 -> Publishing on topic esp32/bme280/temperature at QoS 1, packetId: 0Message: 23.10
13:58:04.327 -> Publishing on topic esp32/bme280/humidity at QoS 1, packetId 0: Message: 62.00
13:58:06.328 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:08.377 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:10.375 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:12.410 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:14.448 -> Humidity: 90.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.82°C 74.87°F
13:58:14.448 -> Publishing on topic esp32/bme280/temperature at QoS 1, packetId: 0Message: 23.10
13:58:14.448 -> Publishing on topic esp32/bme280/humidity at QoS 1, packetId 0: Message: 90.00
13:58:16.484 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
13:58:18.502 -> Humidity: 62.00%  Temperature: 23.10°C 73.58°F  Heat index: 23.08°C 73.55°F
```

Subscribe Window Temperature:

```
pi@RasPhamily:~ $ mosquitto_sub -d -u kokojuice2 -P kokojuice123 -t esp32/temper
ature
Client mosqsub|2228-RasPhamily sending CONNECT
Client mosqsub|2228-RasPhamily received CONNACK (0)
Client mosqsub|2228-RasPhamily sending SUBSCRIBE (Mid: 1, Topic: esp32/temperatu
re, QoS: 0)
Client mosqsub|2228-RasPhamily received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|2228-RasPhamily received PUBLISH (d0, q0, r1, m0, 'esp32/temperat
ure', ... (5 bytes))
22.00
```

```
mosquitto_sub -d -u kokojuice2 -P kokojuice123 -t esp32/temperature
```

Subscribe Window Humidity :

```
^C
pi@RasPhamily:~ $ mosquitto_sub -d -u kokojuice2 -P kokojuice123 -t e
sp32/humidity
Client mosqsub|2237-RasPhamily sending CONNECT
Client mosqsub|2237-RasPhamily received CONNACK (0)
Client mosqsub|2237-RasPhamily sending SUBSCRIBE (Mid: 1, Topic: esp3
2/humidity, QoS: 0)
Client mosqsub|2237-RasPhamily received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|2237-RasPhamily received PUBLISH (d0, q0, r1, m0, 'esp
32/humidity', ... (5 bytes))
65.00
Client mosqsub|2237-RasPhamily received PUBLISH (d0, q0, r0, m0, 'esp32/hum
idity', ... (5 bytes))
66.00
```

```
mosquitto_sub -d -u kokojuice2 -P kokojuice123 -t esp32/humidity
```

Full window :