MQTT with Arduino

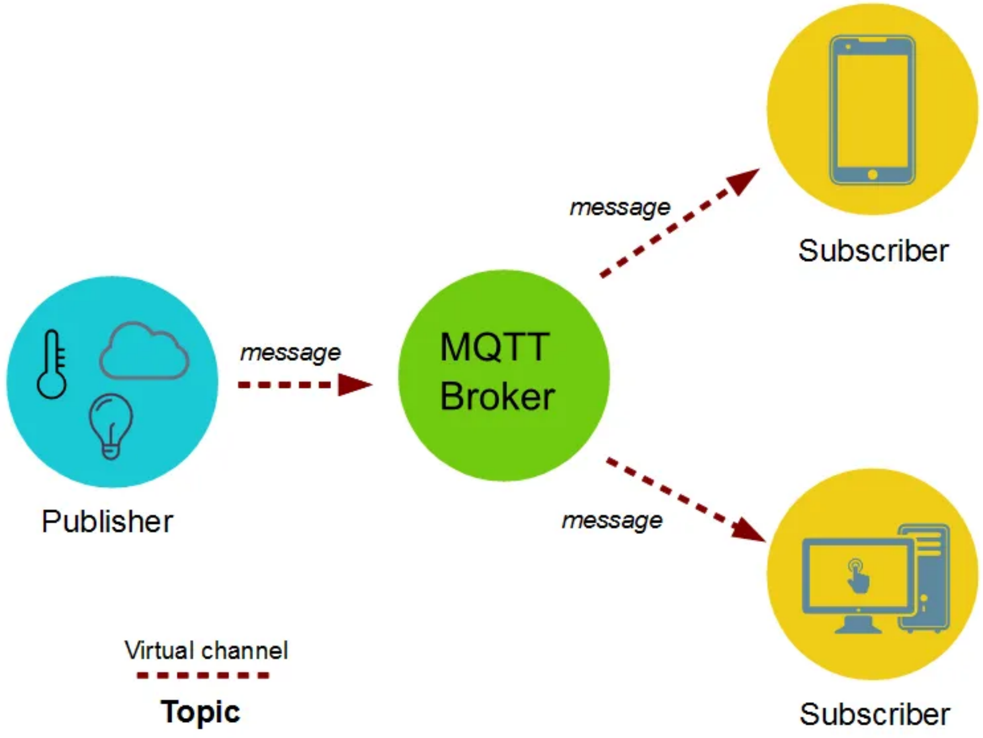
MQTT was developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech; now Cirrus Link) in 1999 for the monitoring of an oil pipeline through the desert.

The goals were to have a protocol, which is bandwidth-efficient and uses little battery power, because the devices were connected via satellite link and this was extremely expensive at that time.

The protocol uses a publish/subscribe architecture wich is event-driven and enables messages to be pushed to clients. The central communication point is the MQTT broker, it is in charge of dispatching all messages between the senders and the rightful receivers. Each client that publishes a message to the broker, includes a topic into the message.

The topic is the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client. Therefore the clients don’t have to know each other, they only communicate over the topic.

This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.
source (https://www.hivemq.com/blog/how-to-get-started-with-mqtt)



A topic is a simple string defined by the user that can have more hierarchy levels, which are separated by a slash.

```
mdef/input/team1/temperature
mdef/ouput/team2/motor
```

Wilcards can also be used in sigle leves ej. `mdef/input/+/temperature` will return **temperatures of all teams**.
Or in multilevels: `mdef/output/#` will return **all** outputs from **all teams**.

OUR BROKER

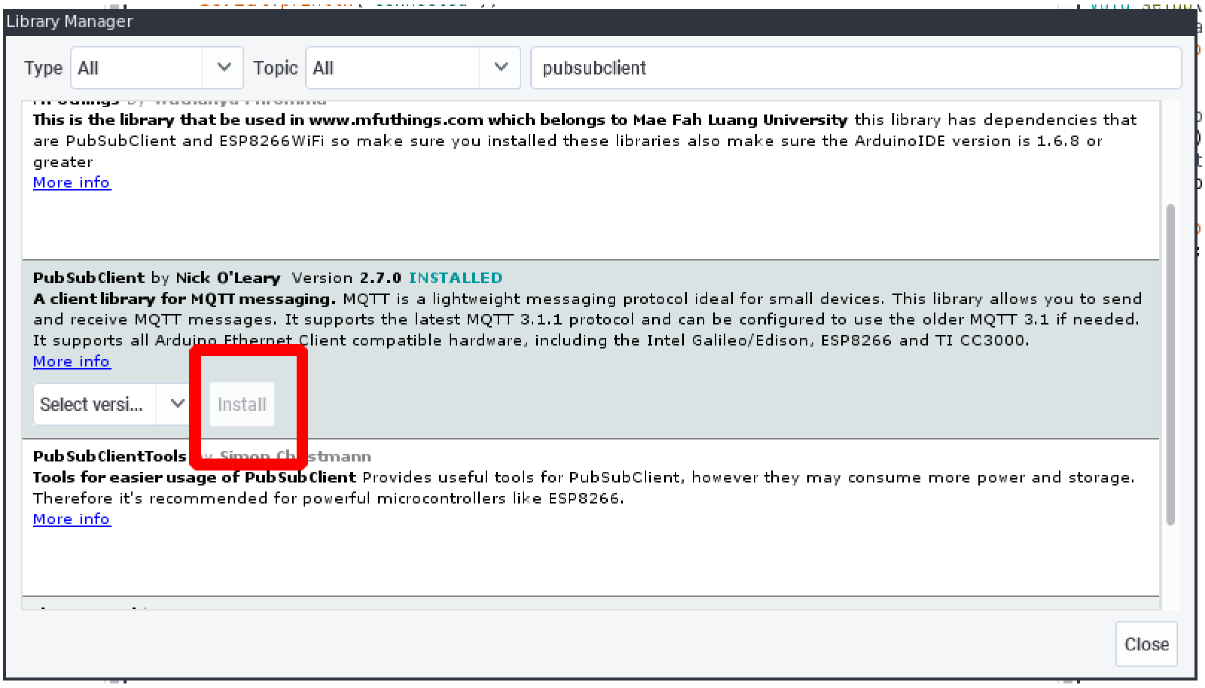
Today we will use mosquitto (https://mosquitto.org/) an open-source MQTT broker running in Ubuntu Linux (https://ubuntu.com/) on a Digital Ocean (https://www.digitalocean.com/) instance in their Frankfurt datacenter.

IP: `159.89.105.9`
DOMAIN: patch.pral2a.com (http://patch.pral2a.com)

`1883`	MQTT, unencrypted
`8883`	MQTT, encrypted
`8884`	~~MQTT, encrypted, client certificate required~~
`8887`	~~MQTT, encrypted, server certificate deliberately expired~~
`8080`	~~MQTT over WebSockets, unencrypted~~
`8081`	MQTT over WebSockets, encrypted

MQTT on Arduino IDE

- Install MQTT libray:
  Open the **Library manager** in Arduino menu *Sketch* -> *Include Library* -> *Manage Libraries* and search for the **PubSubClient** library, install it.



This is the library that be used in www.mfuthings.com which belongs to Mae Fah Luang University this library has dependencies that are PubSubClient and ESP8266WiFi so make sure you installed these libraries also make sure the ArduinoIDE version is 1.6.8 or greater
More info

PubSubClient by Nick O'Leary Version 2.7.0 INSTALLED
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
More info

PubSubClientTools by Simon Christmann
Tools for easier usage of PubSubClient Provides useful tools for PubSubClient, however they may consume more power and storage. Therefore it's recommended for powerful microcontrollers like ESP8266.
More info

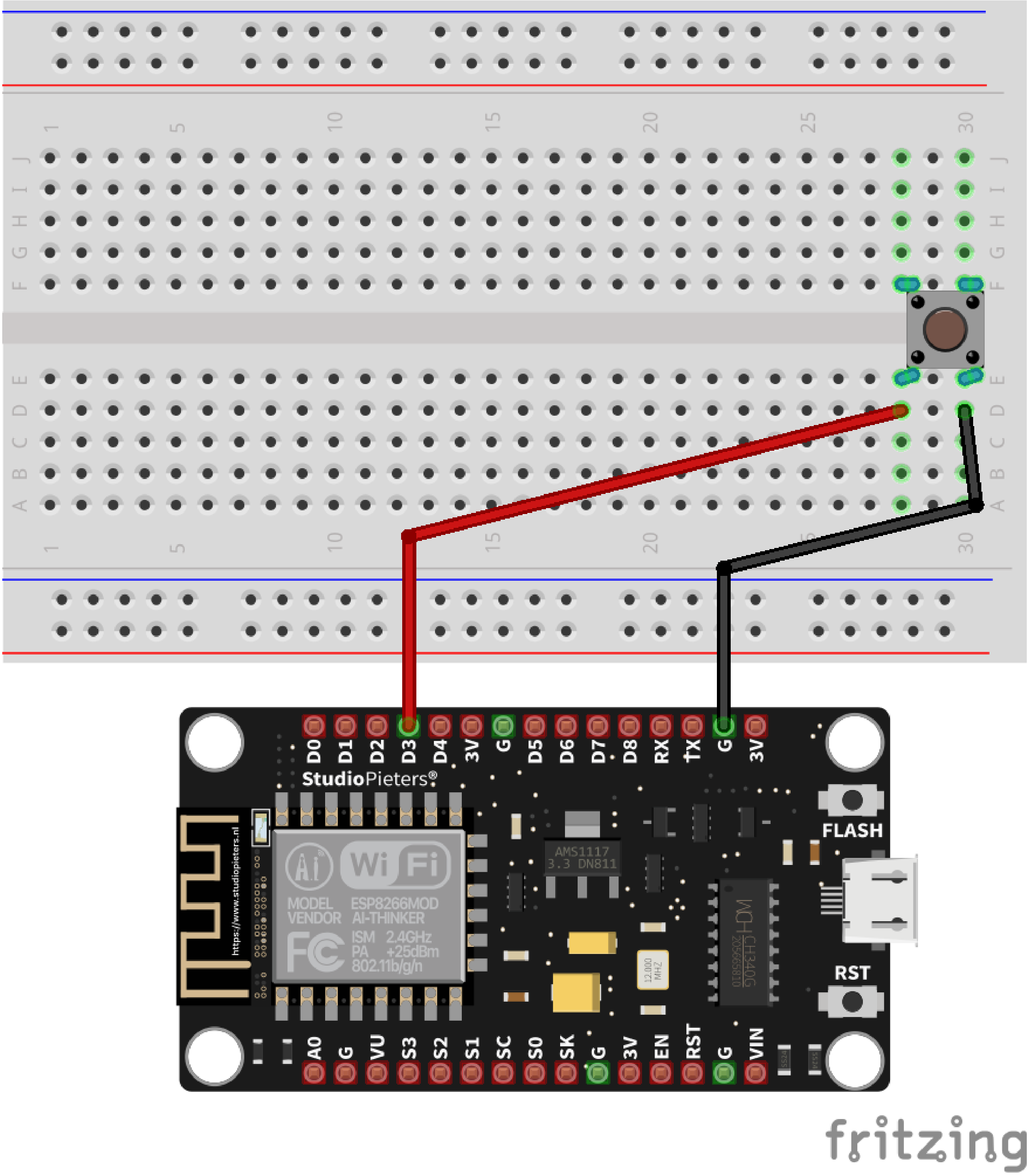
You can find the full API documentation for the *PubSubClient* library here (https://pubsubclient.knolleary.net/api.html)

For the first test you can copy/paste the code example in this document.

Code Example

Minimal Wiring

You will need to wire a button to fully test the code example.



You can also wire an external LED or use the default in the NODE MCU board.

▶ Full Code

Global setup

Include Libraries

```
#include <WiFi.h>
#include <PubSubClient.h>
```

Wifi

Change the WiFi **ssid** and **password** to fit your's.

```
const char* ssid = "myWiFiName";
const char* password = "myWifiPassword";
WiFiClient wifiClient;
```

MQTT

Set the **IP address or name of your MQTT broker**, the name that will identify your device and the names of a topic(s) to subscribe and/or publish.

```
const char* ssid = "Brillant";
const char* password = "estelarhelenabrillant";
const char* mqtt_server = "patch.pral2a.com";
const char* topicToSub = "teamx/in";
const char* topicToPub = "teamx/out";
PubSubClient mqttClient(wifiClient);
```

> Adding **variables for topic names is optional** but in this way if you need to change them you can do it in one point instead of tracking all the places where they are used.

Helper functions

MQTT connect function

This function is called on the main loop, it checks if the connection with the MQTT broker is alive and reconnects if necessary.
Here **we subscribe to the topics we want to receive.**

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(topicToPub, "hello world");
      // ... and resubscribe
      client.subscribe(topicToSub);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

Subscription callback

This function is executed every time a message is received. In this example if we receive the words *on* or *off* in a specific topic we change the led state.

```
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();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
  }

}
```

Subscribing to a new topic is done in three points of the code:

1. Near the top create a variable with the **topic name** you want to subscribe to.
2. In the `mqttConnect()` function add a line **subscribing to the new topic**: `mqttClient.subscribe(newName)`
3. In the `callback()` function check if the new topic is **receiving** something, **evaluate** the message and **take some actions!**

Main functions

Setup

Start Serial communication, WiFi, MQTT and the pin to control internal led.

```
void setup() {
  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUTTON pin as an output
  pinMode(buttonPin, INPUT_PULLUP);     // Initialize the BUILTIN_LED pin as an output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

Loop

In the main loop we **check if the MQTT connection is still alive** and call the `mqttConnect()` function if needed.
We need to call the `mqttClient.loop()` function every cycle so the *PubSubClient* library has time to do his inner works.

```
// Declare needed variables
long lastMsg = 0;
char msg[50];

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
    int buttonState = !digitalRead(buttonPin);
    snprintf (msg, 50, "%ld", buttonState);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topicToPub, msg);
  }
}
```

Now make sure your broker is running and it is on the right address and power your feather!

Publishing a MQTT message can be done just adding a call to the `publish(topic, message)` function of the MQTT client.

Troubleshooting

- Check the output of the Feather **Serial port**
  - Is WiFi connection successfull?
  - Is your Feather finding the MQTT broker?
- To **interact with MQTT from your computer** you can use mosquitto (https://mosquitto.org/) software.
  - Try sending a MQTT message to a subscribed topic and check if the feather receives it.
  - Subscribe to a topic whre your Feather is posting and check if you receive something.
- **Log in** your Raspberry Pi MQTT broker.
  - Check the logs of the Mosquitto broker (`/var/log/mosquitto/mosquitto.log`)

  - Stop the broker `sudo systemctl stop mosquitto` and run it manually (just type `mosquitto` and hit enter) to see the the console output.

> Printing **debug messages to the Serial port** can make your code big and more complicated, but when something doesn’t work you will be gratefull of those lines!

Extra

▼ Full Code

```

/*
 Basic ESP8266 MQTT example

 This sketch demonstrates the capabilities of the pubsub library in combination
 with the ESP8266 board/library.

 It connects to an MQTT server then:
  - publishes "hello world" to the topic "outTopic" every two seconds
  - subscribes to the topic "inTopic", printing out any messages
    it receives. NB - it assumes the received payloads are strings not binary
  - If the first character of the topic "inTopic" is an 1, switch ON the ESP Le
    else switch it off

 It will reconnect to the server if the connection is lost using a blocking
 reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
 achieve the same result without blocking the main loop.

 To install the ESP8266 board, (using Arduino 1.6.4+):
  - Add the following 3rd party board manager under "File -> Preferences -> Add
       http://arduino.esp8266.com/stable/package_esp8266com_index.json
  - Open the "Tools -> Board -> Board Manager" and click install for the ESP826
  - Select your ESP8266 in "Tools -> Board"

*/

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.

const char* ssid = "Brillant";
const char* password = "estelarhelenabrillant";
const char* mqtt_server = "patch.pral2a.com";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

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();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("/hello", "hello world");
      // ... and resubscribe
      client.subscribe("/arduino_in");
    } 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() {
  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an out
  pinMode(D3, INPUT_PULLUP);     // Initialize the BUILTIN_LED pin as an output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
    int buttonState = !digitalRead(D3);
    snprintf (msg, 50, "%ld", buttonState);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("/arduino_out", msg);
  }
}

```