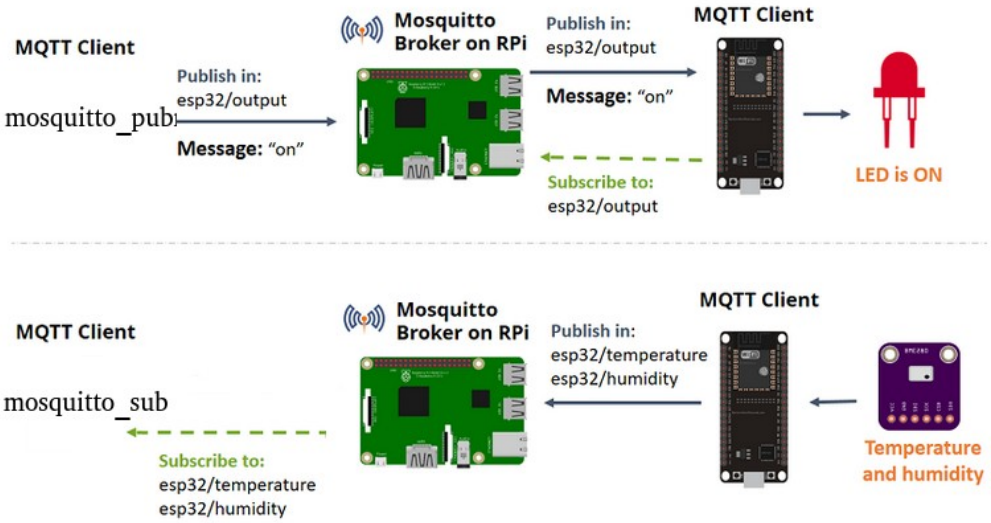
Client MQTT sur ESP32

Sommaire :

I - Contexte

Le dialogue **MQTT** sera conforme au schéma ci-dessous :



Le broker **MQTT** sera un Raspberry Pi d'adresse 172.17.20.X comportant le broker mosquitto. Le broker disposera d'un compte : login esp32 et mot de passe esp32.

Le client MQTT constitué d'un esp32 publiera des valeurs de température et d'humidité sur les topics **"esp32/temperature"** et **"esp32/humidity"**. Il s'abonnera aussi au topic **"esp32/output"**.

Afin de valider cette publication et cet abonnement on utilisera aussi un client MQTT sous Linux (**mosquitto-client**).

II - Publication MQTT

L'exemple de programme ci-dessous, publie cycliquement une **valeur d'humidité** sur le topic **"esp32/humidity"** et affiche les valeurs publiées :

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

#define HUMIDITE_TOPIC "esp32/humidity"  //Topic humidite
#define COMMAND_TOPIC "esp32/output"  //Topic output

const char* ssid = "AP-BTS-SNIR";
const char* password = "12345678";
const char* mqtt_server = "172.17.20.X";
```

```
const char* mqtt_user = "esp32";  // s'il a été configuré sur Mosquitto
const char* mqtt_password = "esp32"; // idem

void callback(char*, byte*, unsigned int);
void reconnect();

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

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    }
  Serial.println("");
  Serial.println("WiFi connexion OK ");
  Serial.print("=> Addresse IP : ");
  Serial.print(WiFi.localIP());
  Serial.println();
  Serial.print("=> Addresse MAC : ");
  Serial.println(WiFi.macAddress());
  Serial.println();

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  // put your main code here, to run repeatedly:
if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) { // toutes les 5s
    lastMsg = now;

    // Génération d'une humidité comprise entre 0 et 100
    humidite = (float)((rand() % 10000))/100;

    Serial.print("Humidite = ");
    Serial.println(humidite);
    //client.publish("esp32/humidity", String(humidite).c_str());
```

```
    client.publish(HUMIDITE_TOPIC, String(humidite).c_str());
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
      // Subscribe
      client.subscribe("esp32/output");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  if (String(topic) == COMMAND_TOPIC) {
    Serial.print("Changing output to ");
    if(messageTemp == "on"){
      Serial.println("on");
    }
    else if(messageTemp == "off"){
      Serial.println("off");
    }
  }
}
```

Remarque : Lors de la connexion au broker **mqtt**, appel de la méthode **client.connect()**, le client est identifié avec un **nom** : dans cet exemple "**ESP32Client**". Il **faudra impérativement** modifier ce nom sur chaque **client MQTT** développé sur **ESP32**.

Si on veut envoyer dans un **seul topic** plusieurs informations, par exemple une valeur de **température** et une valeur d'**humidité**, il faut créer un **objet JSON** contenant ces 2 grandeurs que nous publierons sur le **topic** correspondant. Cet objet JSON aura la syntaxe suivante :

{"temperature":"20.5","humidite":"52.5"}

Afin de coder la génération de cet objet JSON, on utilisera la portion de code suivant :

```
  String json = "{";
  json+= ",\"temperature\":\""+String(temperature)+"\"";
  json+= ",\"humidite\":\""+String(humidite)+"\"";
  json+="}";
Serial.println("JSON = " + json);
```

On peut valider le programme en s'abonnant au topic **"esp32/humidity"** à l'aide de la commande Linux :

```
mosquitto_sub -u esp32 -P esp32 -h 172.17.20.X -t "esp32/humidity"
```