

# Using MQTT Protocol in IoT

**Pakistan Institute of Engineering and Applied Sciences**

## **Objectives**

- Understand the fundamentals of MQTT—a lightweight messaging protocol designed for IoT.
- Learn the publish/subscribe communication pattern.
- Configure ESP32 / WeMos D1 Mini (ESP8266) to connect to a public MQTT broker (e.g., [test.mosquitto.org](http://test.mosquitto.org) / [broker.hivemq.com](http://broker.hivemq.com) / [iot.intellichouse.com.pk](http://iot.intellichouse.com.pk)).
- Publish sensor/button data to a topic (Collaborate with another group).
- Subscribe to a topic and control actuators based on received data (Collaborate with another group).
- Implement code that publishes messages to a topic and subscribes to another topic to receive messages (single device as publisher and subscriber).
- Gain hands-on experience with IoT messaging, debugging, and integrating sensor/actuator control through MQTT.

## **Required Components**

- WeMos D1 Mini (ESP8266) / ESP32 – Optionally 2 boards (one each for publish and subscribe)
- LED (any color) plus a  $220\Omega$  current-limiting resistor
- One push button with optional pullup resistor
- Breadboard and Jumper Wires
- USB cable for programming
- Reliable WiFi connection with internet (if using public broker). Internet is not required if using local hosted broker.
- (Optional) MQTT client tool (e.g., MQTT Explorer, MQTT.fx or an online MQTT dashboard) for testing

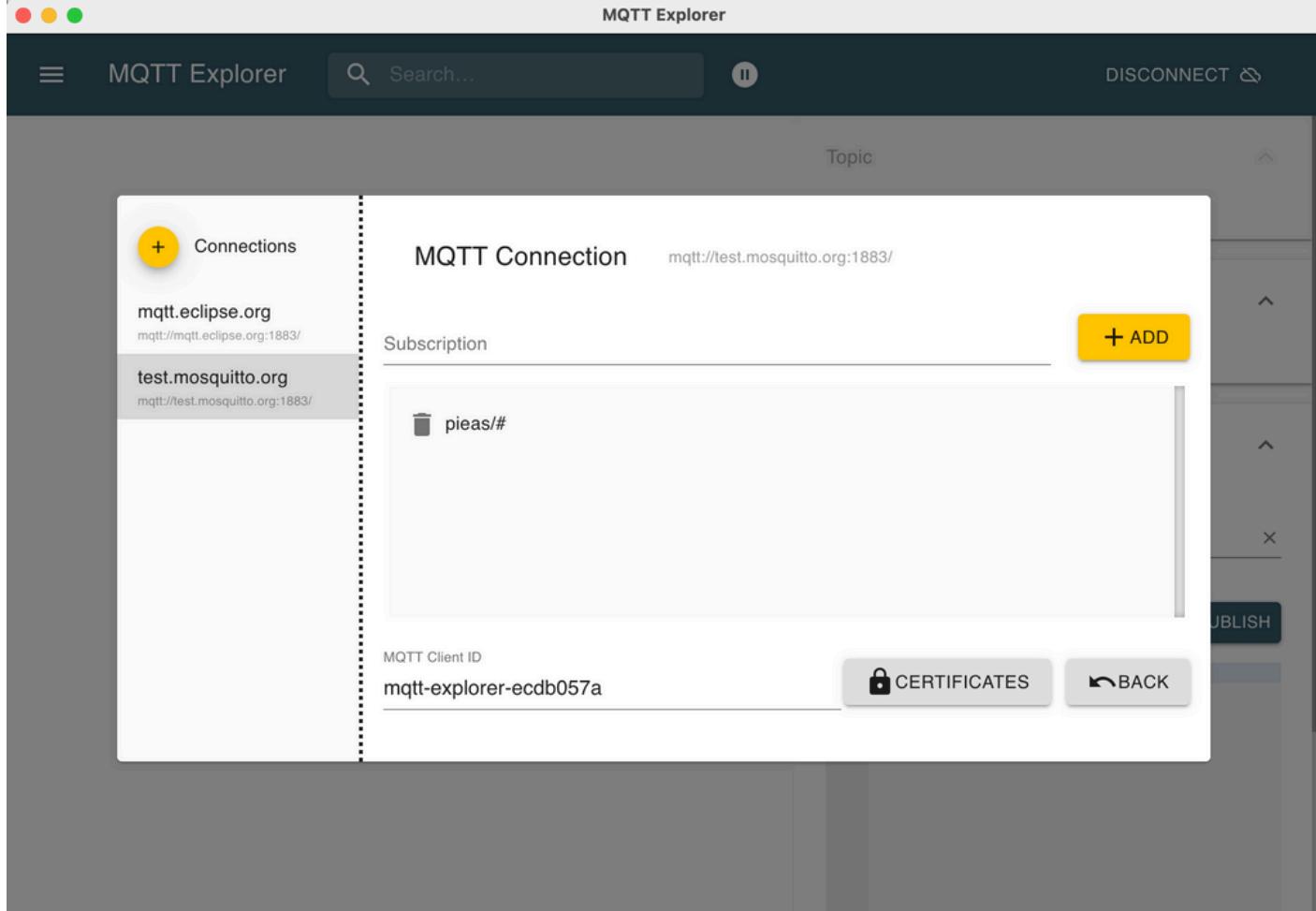
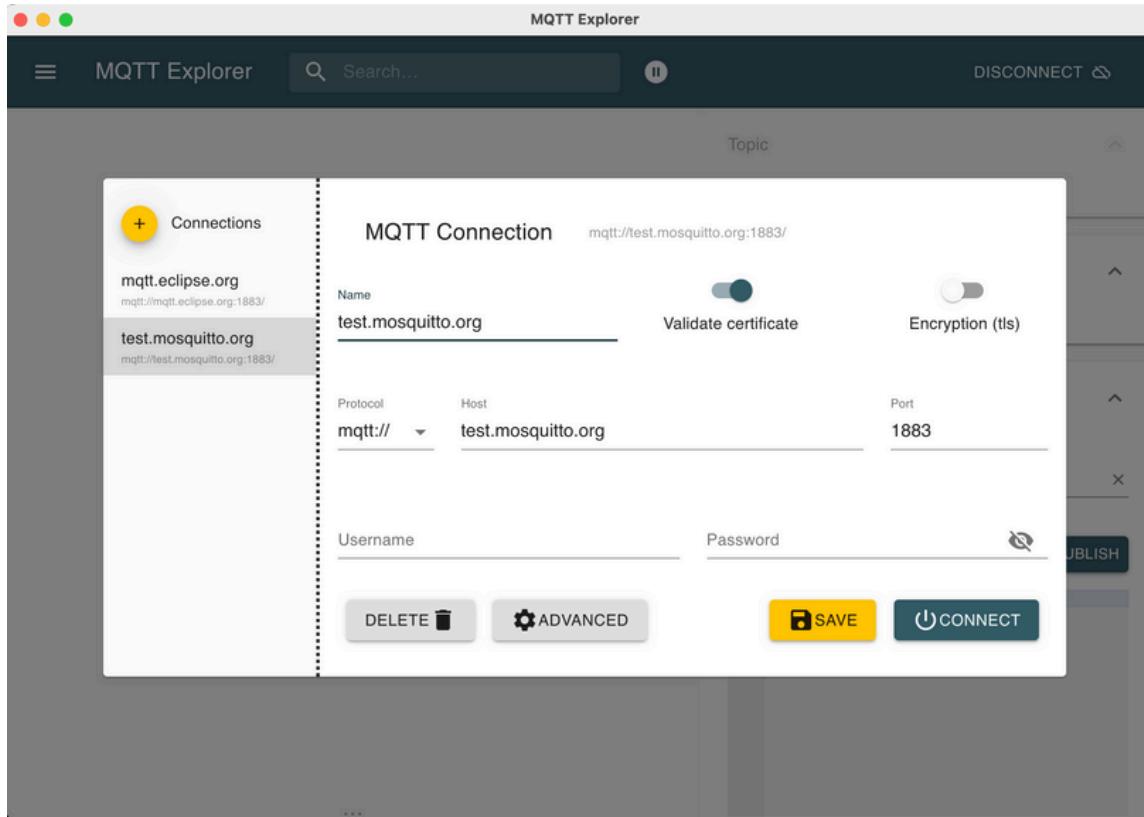
## **Publish/Subscribe Model**

- Publisher: Sends messages to a topic.
- Subscriber: Listens for messages on a topic.
- Broker: Manages topics and routes messages from publishers to subscribers.

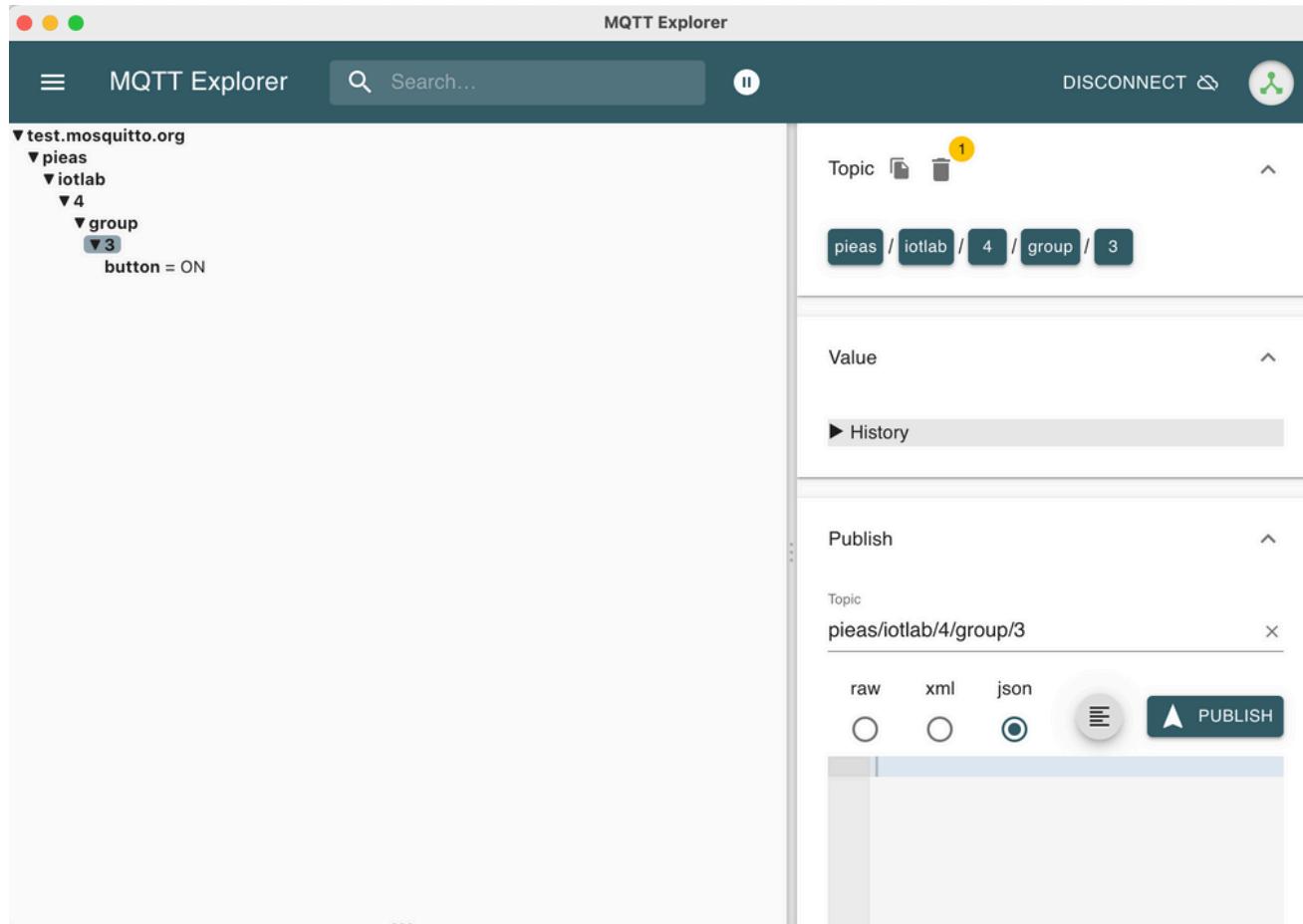
## **Public MQTT Broker**

- For this lab, we will use a public MQTT broker such as [test.mosquitto.org](http://test.mosquitto.org). No authentication is needed, which simplifies initial testing. Default MQTT broker port is 1883.

# Part 1: Setting Up the Environment



**Added the topic pieas/# and deleted other topics for a better focus on our relevant messages**



**MQTT Explorer has been installed and is in the working condition**

## Part 2: Configure the MQTT Publisher (Button)

The screenshot shows the Arduino IDE with the file 'LAB04.ino' open. The code is as follows:

```
1 //Nida Pervaiz //Urwah Rasheed //Wajeeha Jahangir // Areeb Ur Rehman
2 #include <ESP8266WiFi.h>
3 #include <PubSubClient.h>
4 // Wi-Fi Credentials
5 const char* ssid = "IoT-Class";
6 const char* password = "iotclass1";
7 // MQTT Broker
8 const char* mqtt_server = "test.mosquitto.org"; // Check Broker
9 const int mqtt_port = 1883;
10 const char* topic = "pieas/iotlab/4/group/6/button"; // Check Group No.
11 WiFiClient espClient;
12 PubSubClient client(espClient);
13 // Button Setup
14 const int buttonPin = D2;
15 bool lastButtonState = LOW;
16 void setup() {
17   Serial.begin(115200);
18   pinMode(buttonPin, INPUT_PULLUP);
19   setupWiFi();
20   client.setServer(mqtt_server, mqtt_port);
21 }
22 void setupWiFi() {
23   delay(10);
24   WiFi.begin(ssid, password);
25   while (WiFi.status() != WL_CONNECTED) {
26     delay(500);
27     Serial.print(".");
28   }
29   Serial.println("WiFi Connected");
30 }
31 void reconnect() {
32   while (!client.connected()) {
33     if (!client.connect("n_0_0_0_0_0_0_0_0")) { // Use a random unique device id
34       Serial.println("Connection failed");
35     } else {
36       Serial.println("Connected");
37     }
38   }
39 }
```

## Output

```
SEGMENT    BYTES    DESCRIPTION
└ IROM      248440   code in flash
esptool.py v3.0
Serial port /dev/cu.usbserial-14110
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: a8:48:fa:dd:60:e4
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 282240 bytes to 207293...
Writing at 0x00000000... (7 %)
```



⌚ indexing: 12/86

We(Group 3) served as publisher and group 6 served as subscriber

The screenshot shows the Arduino IDE interface. The top bar has icons for file operations (checkmark, arrow, etc.) and a dropdown menu set to "LOLIN(WEMOS) D1 R2...". The left sidebar contains icons for file, folder, library, and search. The main area displays the code for "LAB04.ino".

```
LAB04.ino
4 // Wi-Fi Credentials
5 const char* ssid = "IoT-Class";
6 const char* password = "iotclass1";
7 // MQTT Broker
8 const char* mqtt_server = "test.mosquitto.org"; // Check Broker
9 const int mqtt_port = 1883;
10 const char* topic = "pieas/iotlab/4/group/6/button"; // Check Group No.
11 WiFiClient espClient;
12 PubSubClient client(espClient);
13 // Button Setup
14 const int buttonPin = D2;
15 bool lastButtonState = LOW;
16 void setup() {
17   Serial.begin(115200);
18   pinMode(buttonPin, INPUT_PULLUP);
19   setupWiFi();
20   client.setServer(mqtt_server, mqtt_port);
21 }
22 void setupWiFi() {
23   delay(10);
24   WiFi.begin(ssid, password);
25   while (WiFi.status() != WL_CONNECTED) {
26     delay(500);
27     Serial.print(".");
28   }
29   Serial.println("WiFi Connected");
30 }
31 void reconnect() {
32   while (!client.connected()) {
33     if (client.connect("p.io.0.567456")) { // Use a random unique device id.
34       Serial.println("MQTT Connected");
35     } else {
```

Output

Output Serial Monitor X

Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-14110')

```
0x0x0x.....  
Trying to connect MQTT ...  
MQTT Connected  
Button is ON  
Button is OFF  
Button is ON  
Button is OFF  
Button is ON  
MQTT Connected
```

## CODE IMPLEMENTATION

```
//Nida Pervaiz //Urwah Rasheed //Wajeeha Jahangir // Areeb Ur Rehman  
#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
// Wi-Fi Credentials  
const char* ssid = "IoT-Class";  
const char* password = "iotclass1";  
// MQTT Broker  
const char* mqtt_server = "test.mosquitto.org"; // Check Broker  
const int mqtt_port = 1883;  
const char* topic = "pieas/iotlab/4/group/6/button"; // Check Group No.  
WiFiClient espClient;  
PubSubClient client(espClient);  
// Button Setup  
const int buttonPin = D2;  
bool lastButtonState = LOW;  
void setup() {  
    Serial.begin(115200);  
    pinMode(buttonPin, INPUT_PULLUP);  
    setupWiFi();  
    client.setServer(mqtt_server, mqtt_port);  
}  
void setupWiFi() {  
    delay(10);  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
        Serial.print(".");  
    }  
    Serial.println("WiFi Connected");  
}
```

```

void reconnect() {
  while (!client.connected()) {
    if (client.connect("p.iot.0.567456")) { // Use a random unique device id.
      Serial.println("MQTT Connected");
    } else {
      Serial.println("Trying to connect MQTT ... ");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();
  bool buttonState = digitalRead(buttonPin);
  if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      Serial.println("Button is ON");
      client.publish(topic, "ON");
    } else {
      Serial.println("Button is OFF");
      client.publish(topic, "OFF");
    }
    lastButtonState = buttonState;
  }
  delay(50);
}

```

## ON MQTT EXPLORER

The screenshot shows the MQTT Explorer application interface. At the top, there is a header bar with a menu icon (three horizontal lines), the title "MQTT Explorer", a search bar with a magnifying glass icon and placeholder "Search...", and a settings or refresh icon (two vertical bars).

The main area displays a hierarchical tree view of MQTT topics:

- ▼ test.mosquitto.org
  - ▼ pieas
  - ▼ iotlab
    - ▼ 4
      - ▼ group
        - ▼ 3
          - button = OFF**
  - ▼ 6
    - button = ON**

0x0x0x.....

Trying to connect MQTT ...

MQTT Connected

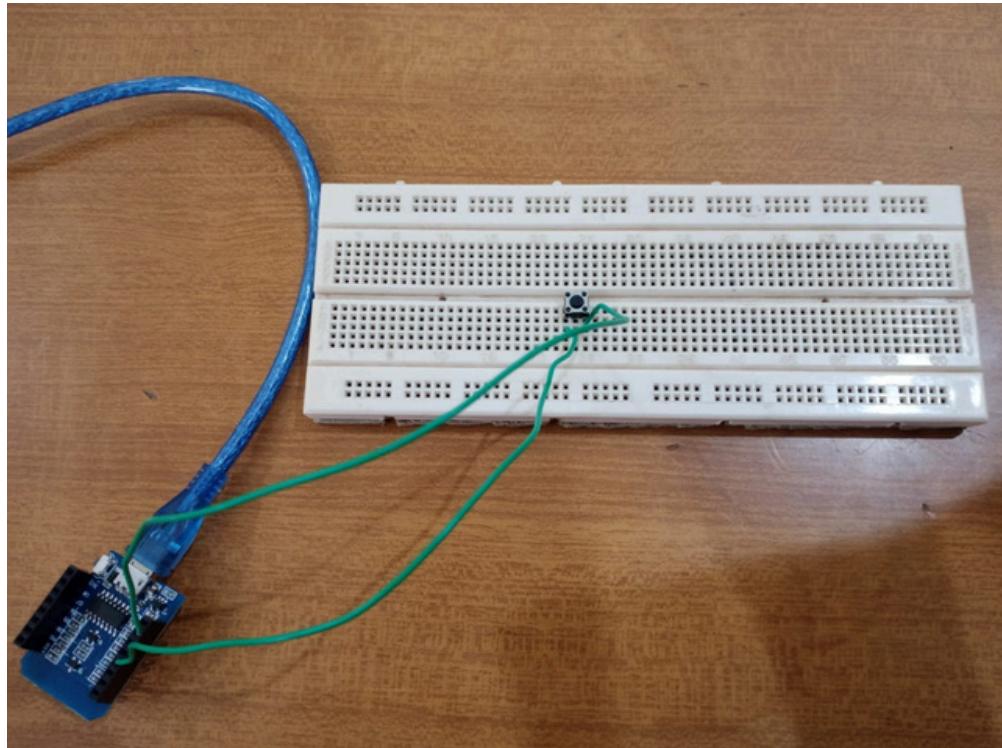
Button is ON

Button is OFF

Button is ON

Button is OFF

Button is ON



## Wiring:

One pin of button with D2 pin of ESP8266. The other pin connected with ground. We used INPUT\_PULLUP in the code instead of using a pull up resistor in hardware.

## Part 3: Configure the MQTT Subscriber (LED)

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// Wi-Fi Credentials (same as publisher)
const char* ssid = "OPPO A13";
const char* password = "12345678";
// MQTT Broker
const char* mqtt_server = "test.mosquitto.org"; // Check Broker
const int mqtt_port = 1883;
const char* topic = "pieas/iotlab/4/group/6/button"; // Match publisher
WiFiClient espClient;
PubSubClient client(espClient);
// LED Setup
const int ledPin = D4;
void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH); // Initially LED is OFF
  setupWiFi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback); // Handle incoming messages
}
```

```

void setupWiFi() {
delay(10);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("WiFi Connected");
}

void reconnect() {
while (!client.connected()) {
if (client.connect("p.iot.0.5686754")) { // Use a random unique device id.
Serial.println("MQTT Connected");
client.subscribe(topic); // Additional Line to subscribe topic
} else {
Serial.println("Trying to connect MQTT ... ");
delay(5000);
}
}
}

void callback(char* topic, byte* payload, unsigned int length) {
String message;
for (int i = 0; i < length; i++) {
message += (char)payload[i];
}

//Print received message on Serial Console
Serial.print("Got Message: ");
Serial.print(topic);
Serial.print(" > ");
Serial.println(message);
//Set LED state based on the received Message
if (message == "ON") {
digitalWrite(ledPin, LOW);
} else if (message == "OFF") {
digitalWrite(ledPin, HIGH);
}
}

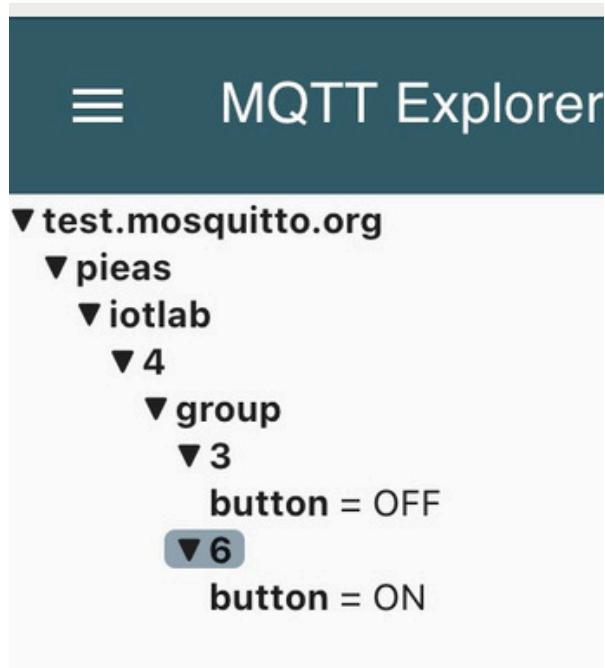
void loop() {
if (!client.connected()) reconnect();
client.subscribe(topic);
client.loop();
}

```

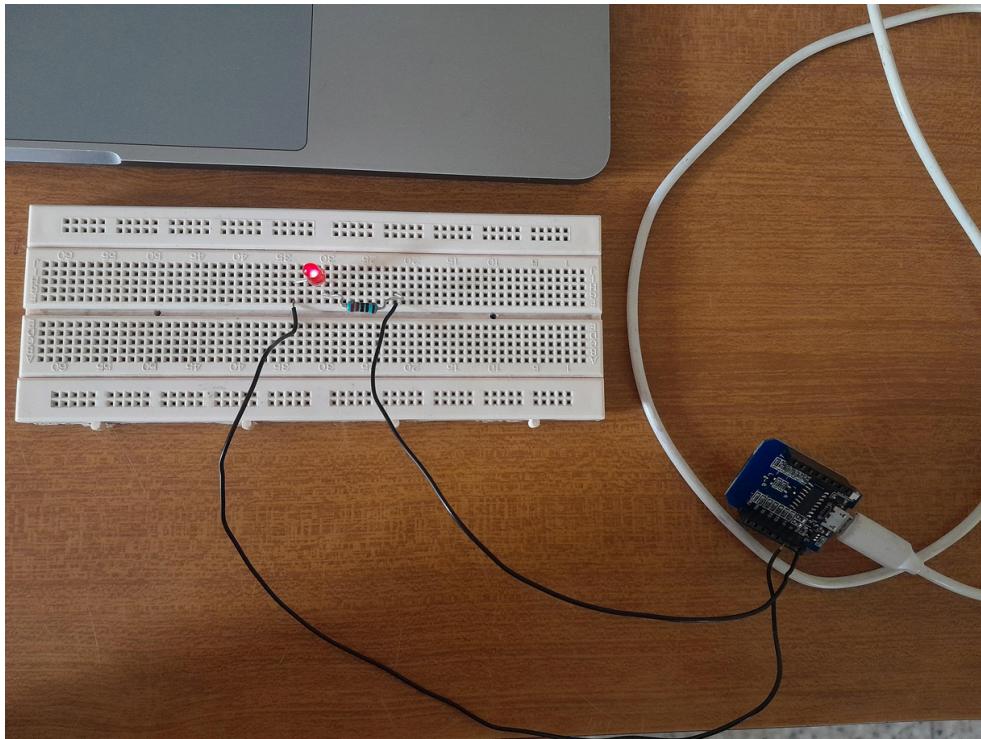
**now we(group 3) served as subscriber and group 6 served as publisher**



# On MQTT Explorer



## WIRING:



Anode (+) → 220 ohm resistor

Cathode (-) → WeMos D1 GND pin

Resistor's other terminal → WeMos D1 digital pin (D4)

The screenshot shows the MQTT Explorer application interface. On the left, a tree view displays the following topic structure under "test.mosquitto.org":

- pleas
- iotlab
- 4
  - group
    - 0
      - button = OFF
    - 4
      - task3
        - button = ON
    - 6
      - button = ON

On the right, the publishing interface is shown with the following details:

- Topic:** pleas / iotlab / 4 / group / 6
- Value:** (empty)
- History:** (empty)
- Publish:**
  - Topic:** pleas/iotlab/4/group/6
  - Format:** raw (radio button selected), xml, json
  - PUBLISH:** button

# Part 4: Both Publisher and Subscriber on Single Device

//Wajeeha Jahangir //Nida Pervaiz //Urwah Rasheed //Areeb Ur Rehman

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

// Wi-Fi Credentials (same as publisher)
const char* ssid = "OPPO A31";
const char* password = "12345678";
// MQTT Broker
const char* mqtt_server = "test.mosquitto.org"; // Check Broker
const int mqtt_port = 1883;
const char* topic = "pieas/iotlab/4/group/3/button"; // Match publisher
WiFiClient espClient;
PubSubClient client(espClient);
// Button Setup
const int buttonPin = D2;
bool lastButtonState = LOW;
// LED Setup
const int ledPin = D4;
void setup() {
    Serial.begin(115200);
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, HIGH); // Initially LED is OFF
    setupWiFi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback); // Handle incoming messages
}
void setupWiFi() {
    delay(10);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi Connected");
}
```

```
void reconnect() {
  while (!client.connected()) {
    if (client.connect("p.iot.0.56993")) { // Use a random unique device id.
      Serial.println("MQTT Connected");
      client.subscribe(topic); // Additional Line to subscribe topic
    } else {
      Serial.println("Trying to connect MQTT ... ");
      delay(5000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }

  //Print received message on Serial Console
  Serial.print("Got Message: ");
  Serial.print(topic);
  Serial.print(" > ");
  Serial.println(message);
  //Set LED state based on the received Message
  if (message == "ON") {
    digitalWrite(ledPin, LOW);
  } else if (message == "OFF") {
    digitalWrite(ledPin, HIGH);
  }
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();
  bool buttonState = digitalRead(buttonPin);
  if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      Serial.println("Button is ON");
      client.publish(topic, "ON");
    } else {
      Serial.println("Button is OFF");
      client.publish(topic, "OFF");
    }
    lastButtonState = buttonState;
  }
  delay(50);
}
```

LOLIN(WEMOS) D1 R2...

```

sketch_mar26b.ino
1 //Wajeeha Jahangir //Nida Pervaiz //Urwah Rasheed //Areeb Ur Rehman
2
3 #include <ESP8266WiFi.h>
4 #include <PubSubClient.h>
5
6 // Wi-Fi Credentials (same as publisher)
7 const char ssid = "OPPO A31";
8 const char password = "12345678";
9 // MQTT Broker
10 const char mqtt_server = "test.mosquitto.org"; // Check Broker
11 const int mqtt_port = 1883;
12 const char topic = "pies/iotlab/4/group/3/button"; // Match publisher
13 WiFiClient espClient;
14 PubSubClient client(espClient);
15 // Button Setup
16 const int buttonPin = D2;
17 bool lastButtonState = LOW;
18 // LED Setup
19 const int ledPin = D4;
20 void setup() {
21   Serial.begin(115200);
22   pinMode(buttonPin, INPUT);
23   pinMode(ledPin, OUTPUT);
24   digitalWrite(ledPin, HIGH); // Initially LED is OFF
25   setupWiFi();
26   client.setServer(mqtt_server, mqtt_port);
27   client.setCallback(callback); // Handle incoming messages
28 }
29 void setupWiFi() {
30   delay(10);
31   WiFi.begin(ssid, password);
32   while (WiFi.status() != WL_CONNECTED) {

```

Output

```

Features: WiFi
Crystal is 26MHz
MAC: a8:48:f0:dd:60:e4
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 283728 bytes to 288259...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (15 %)
Writing at 0x00008000... (23 %)
Writing at 0x0000c000... (30 %)
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)

```

Uploading...

⚠ Connection lost. Cloud sketch actions and updates won't be available.

LN 27, Col 59 LOLIN(WEMOS) D1 R2 & mini on /dev/cu.usbserial-14510 3

LOLIN(WEMOS) D1 R2...

```

sketch_mar26b.ino
18 // LED Setup
19 const int ledPin = D4;
20 void setup() {
21   Serial.begin(115200);
22   pinMode(buttonPin, INPUT);
23   pinMode(ledPin, OUTPUT);
24   digitalWrite(ledPin, HIGH); // Initially LED is OFF
25
26   function digitalWrite
27 } → void
28 } ← Parameters:
29   • uint8_t pin
30   • uint8_t val
31
32

```

Output

```

void digitalWrite(uint8_t pin, uint8_t val)

```

Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-14510')

```

10:28:50.349 ->
10:29:32.854 -> MQTT Connected

```

```
10:29:33.880 -> MQTT Connected  
10:29:33.880 -> MQTT Connected  
10:29:33.880 -> MQTT Connected  
10:29:52.438 -> 0000000x0x0x.WiFi Connected  
10:30:00.846 -> MQTT Connected  
10:30:00.846 -> Button is ON  
10:30:23.055 -> Button is OFF  
10:30:30.039 -> Button is ON  
10:30:31.010 -> Button is OFF
```



⚠ Offline ⚡ indexing: 55/87

```
10:33:09.150 -> trying to connect MQTT ...  
10:33:19.157 -> Trying to connect MQTT ...  
10:33:28.301 -> MQTT Connected  
10:33:28.301 -> Button is ON  
10:33:42.191 -> Button is OFF  
10:33:48.495 -> Got Message: pieas/iotlab/4/group/3/button > ON
```

## ON MQTT EXPLORER

The screenshot shows the MQTT Explorer application interface. The left sidebar displays a tree structure of MQTT topics under "test.mosquitto.org":

- pieas
- iotlab
- 4
- group
  - 3
    - button = OFF
  - 6 (1 topic, 1 message)

The right side of the interface has several panels:

- Topic:** A list of topics: pieas / iotlab / 4 / group / 3. The number "1" is displayed above the list.
- Value:** A placeholder for message values.
- History:** A placeholder for message history.
- Publish:** A panel for publishing messages.
  - Topic:** pieas/iotlab/4/group/3
  - Message Types:** raw, xml, json (json is selected)
  - PUBLISH:** A button to send the message.

# Wiring:

## LED Wiring:

Anode (+) → WeMos D1 digital pin (D4)

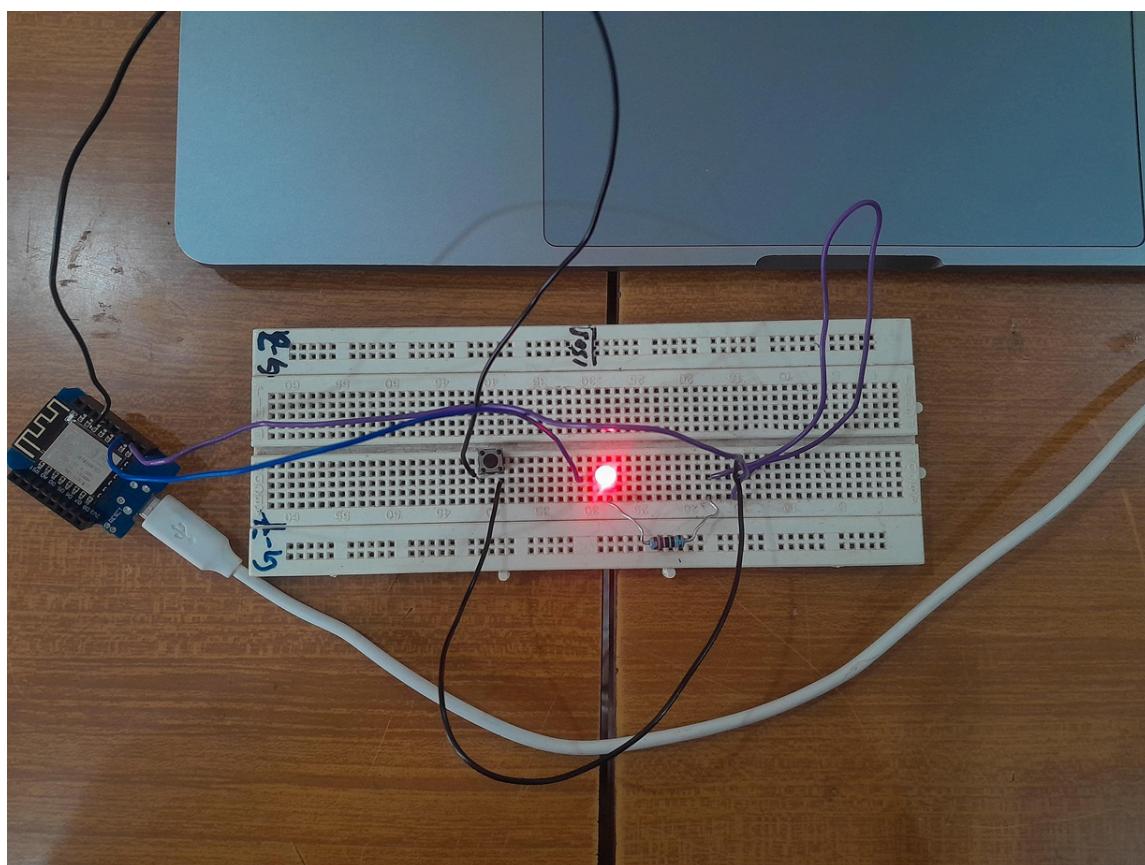
Cathode (-) → 220 ohm resistor

Resistor's other terminal → WeMos D1 GND pin

## Button Wiring:

One terminal → WeMos D1 digital pin (D2)

Other terminal → WeMos D1 GND pin



# Advanced Tasks (Optional for Practice)

## Add a Sensor:

Modify the publisher to read a DHT11 sensor and publish temperature to topic: "pieas/iotlab/4/group/[group\_id]/temperature".

**As we didn't have the sensor available so we used a function to generate random temperature and published it on MQTT explorer .**

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

// Wi-Fi Credentials
const char* ssid = "Galaxy A1334A5";
const char* password = "pmgt8561";

// MQTT Broker Details
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;
const char* temp_topic = "pieas/iotlab/4/group/3/temperature"; // Topic for simulated temperature

WiFiClient espClient;
PubSubClient client(espClient);

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

    setupWiFi();
    client.setServer(mqtt_server, mqtt_port);
}

void setupWiFi() {
    delay(10);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi Connected");
}

void reconnect() {
    while (!client.connected()) {
        if (client.connect("p.iot.0.569")) { // Unique device ID
            Serial.println("MQTT Connected");
        } else {
            Serial.println("Trying to connect to MQTT...");
            delay(5000);
        }
    }
}
```

```

void loop() {
if (!client.connected()) reconnect();
client.loop();
// Simulating DHT11 Temperature Reading
int randomTemperature = random(20, 35); // Generates a random temperature between 20°C and 35°C
char tempString[8];
sprintf(tempString, "%d", randomTemperature); // Convert integer to string
client.publish(temp_topic, tempString); // Publish the random temperature
Serial.print("Published Temperature: ");
Serial.println(tempString);
delay(5000); // Publish temperature every 5 seconds

```

#### LAB04TEMPERATURE.ino

```

1  #include <ESP8266WiFi.h>
2  #include <PubSubClient.h>
3
4  // Wi-Fi Credentials
5  const char* ssid = "Galaxy A1334A5";
6  const char* password = "pmgt8561";
7
8  // MQTT Broker Details
9  const char* mqtt_server = "test.mosquitto.org";
10 const int mqtt_port = 1883;
11 const char* temp_topic = "pieas/iotlab/4/group/3/temperature"; // Topic for simulated temperature
12
13 WiFiClient espClient;
14 PubSubClient client(espClient);
15
16 void setup() {
17     Serial.begin(115200);
18
19     setupWiFi();
20     client.setServer(mqtt_server, mqtt_port);
21 }
22
23 void setupWiFi() {
24     delay(10);
25     WiFi.begin(ssid, password);
26     while (WiFi.status() != WL_CONNECTED) {
27         delay(500);
28         Serial.print(".");
29     }
30     Serial.println("WiFi Connected");
31 }
32

```

#### Output Serial Monitor

```

. Variables and constants in RAM (global, static), used 28476 / 80192 bytes (35%)
SEGMENT  BYTES  DESCRIPTION
DATA      1512    initialized variables
RODATA    1108    constants
BSS       25856   zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 59747 / 65536 bytes (91%)
SEGMENT  BYTES  DESCRIPTION
ICACHE    32768   reserved space for flash instruction cache
IRAM      26979   code in IRAM
. Code in flash (default, ICACHE_FLASH_ATTR), used 248744 / 1048576 bytes (23%)
SEGMENT  BYTES  DESCRIPTION
IROM      248744  code in flash
esptool.py v3.0
Serial port /dev/cu.usbserial-14510
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: a8:48:fa:dc:8a:8c

```

The screenshot shows the Arduino IDE interface. The top bar displays the title "LOLIN(WEMOS) D1 R2...". The left sidebar shows a file tree with "SKETCHBOOK" at the root, followed by "LAB04", "LAB04PART02", "LAB04PART4", and "LAB04TEMPERATURE" which is selected and highlighted in blue. The main area contains the code for "LAB04TEMPERATURE.ino". The code includes #includes for WiFi and PubSubClient, defines Wi-Fi credentials, and sets up MQTT broker details. It initializes WiFiClient and PubSubClient, and defines setup() and setupWiFi() functions. The setup() function begins serial communication at 115200 bps, calls setupWiFi(), and sets the MQTT server and port. The setupWiFi() function attempts to connect to the Wi-Fi network, printing "." to the Serial Monitor until it succeeds. Once connected, it prints "WiFi Connected". The output window shows the published temperatures from the Serial Monitor.

```
1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4 // Wi-Fi Credentials
5 const char* ssid = "Galaxy A1334A5";
6 const char* password = "pmgt8561";
7
8 // MQTT Broker Details
9 const char* mqtt_server = "test.mosquitto.org";
10 const int mqtt_port = 1883;
11 const char* temp_topic = "pieas/iotlab/4/group/3/temperature"; // Topic for simulated temperature
12
13 WiFiClient espClient;
14 PubSubClient client(espClient);
15
16 void setup() {
17   Serial.begin(115200);
18
19   setupWiFi();
20   client.setServer(mqtt_server, mqtt_port);
21 }
22
23 void setupWiFi() {
24   delay(10);
25   WiFi.begin(ssid, password);
26   while (WiFi.status() != WL_CONNECTED) {
27     delay(500);
28     Serial.print(".");
29   }
30   Serial.println("WiFi Connected");
31 }
32 }
```

Output    Serial Monitor X

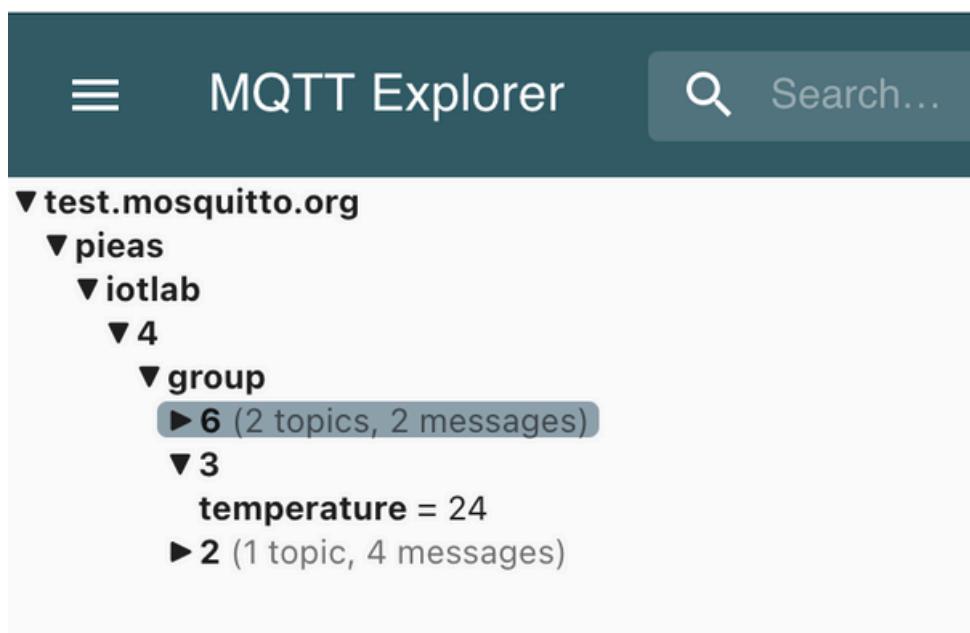
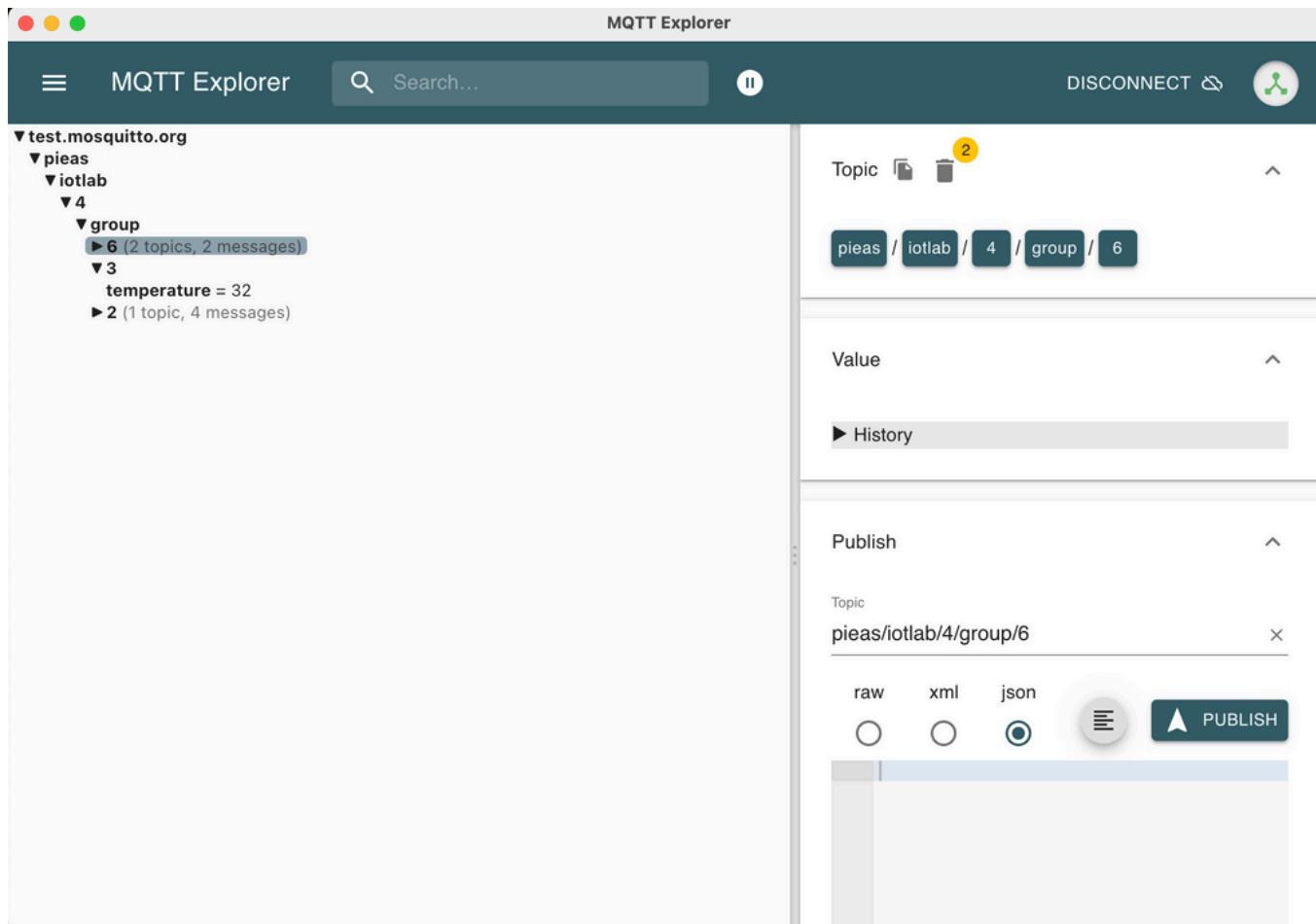
Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-14510')

Published Temperature: 33  
Published Temperature: 34  
Published Temperature: 25  
Published Temperature: 22  
Published Temperature: 31

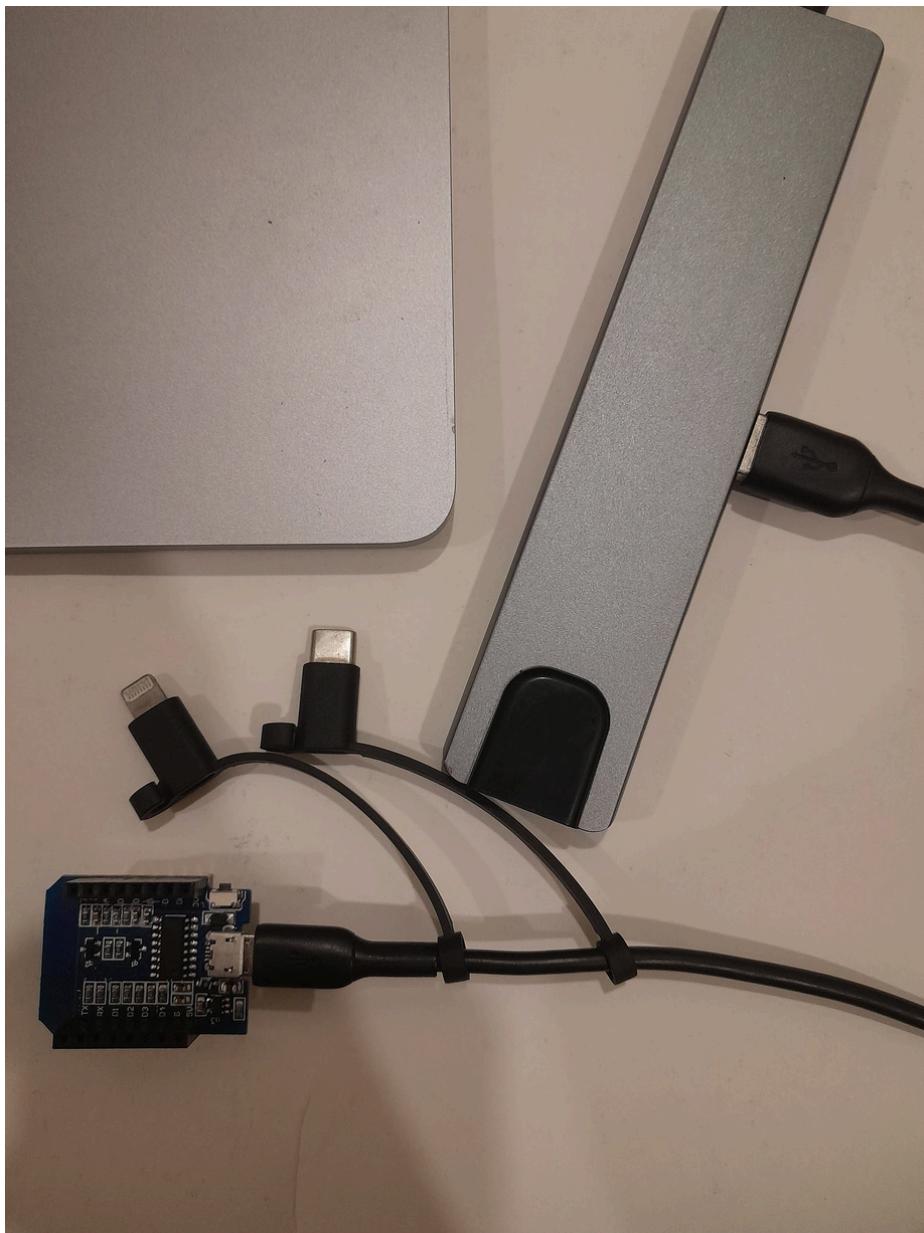
The screenshot shows the Arduino Serial Monitor window. The tabs "Output" and "Serial Monitor" are visible, with "Serial Monitor" being active. The message field says "Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-14510')". The main area displays a series of temperature values printed from the Serial Monitor:

Published Temperature: 33  
Published Temperature: 22  
Published Temperature: 24  
Published Temperature: 32  
Published Temperature: 34  
Published Temperature: 30  
Published Temperature: 34  
Published Temperature: 30  
Published Temperature: 30  
Published Temperature: 28  
Published Temperature: 24  
Published Temperature: 33  
Published Temperature: 23  
Published Temperature: 26  
Published Temperature: 25  
Published Temperature: 21

# ON MQTT EXPLORER



## WIRING:



**As we didn't have the sensor available so we used a function to generate random temperature and published it on MQTT explorer .**

---

# **Part 1: Setting Up the Environment**

## **Steps Followed:**

Installed the required PubSubClient library ,configured WiFi credentials, and installed MQTT client software for message testing.

## **Verification:**

Checked WiFi connectivity and the MQTT client could connect to the broker.

## **Challenges and solutions:**

Faced Wifi connection issues , which are resolved by verifying SSID/password and ensuring an active connection .

Initially selected wrong board ( D1 R1) in Arduino IDE, causing errors. Changing to D1 R2 and Mini to fixed issue.

---

# **Part 2: Configure the MQTT Publisher (Button)**

## **Steps Followed:**

Programmed Esp8266 to connect to WIFI , set up MQTT broker connection ,read button state and publish ON/OFF messages.

## **Verification:**

Used the Serial Monitor and MQTT explorer to confirm message publishing.

## **Challenges and solutions:**

MQTT was not connecting due to weak WIFI signals , moving outside improved the connection and it worked fine.

---

# **Part 3: Configure the MQTT Subscriber (LED)**

## **Step Followed:**

Programmed Esp8266 to connect to WIFI , subscribe to MQTT topic ,and control LED based on received messages.

## **Verification:**

Checked the serial Monitor and observed the LED turning ON/OFF upon receiving messages.

## **Challenges and solutions:**

Encountered MQTT disconnection issues, which were resolved by moving to a strong WIFI signal area and implementing a reconnection function.

## **Part 4: Both Publisher and Subscriber on Single Device**

### **Step Followed:**

Combined publishing and subscribing functionalities in one esp8266 to publish button state and control LED.

### **Verification:**

Used Serial Monitor to observed real-time button state publishing and LED response.

### **Challenges and solutions:**

Initially, the wrong board selection (D1 R1) caused errors. Once changed to D1 R2 & Mini , communication worked properly.

---

## **Advanced Tasks (Optional for Practice)**

### **Step Followed:**

Modified the publisher code to read the temperature data from a DHT11 sensor and publish it to the topic pieas/iotlab/4/group/3/temperature.

### **Verification:**

Used Serial Monitor and MQTT Explorer to check if temperature values were correctly published.

### **Challenges and solutions:**

Initially faced reading error due to incorrect wiring , fixed by making correct connections of wires.

---

**END**