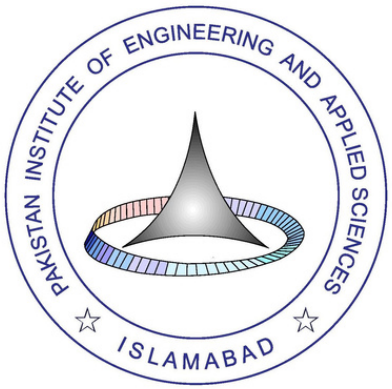


Using MQTT Protocol in IoT

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 / broker.hivemq.com / iot.intellihouse.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 220Ω 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

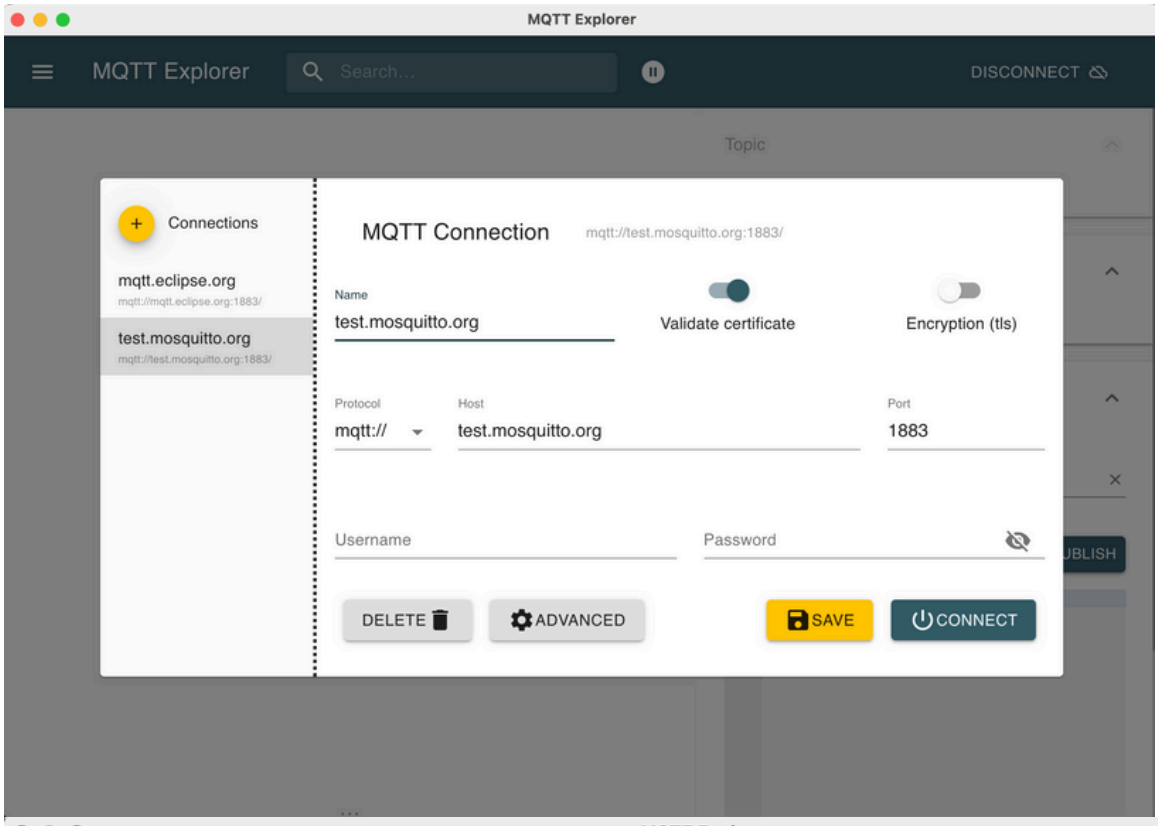
Publish/Subscribe Model

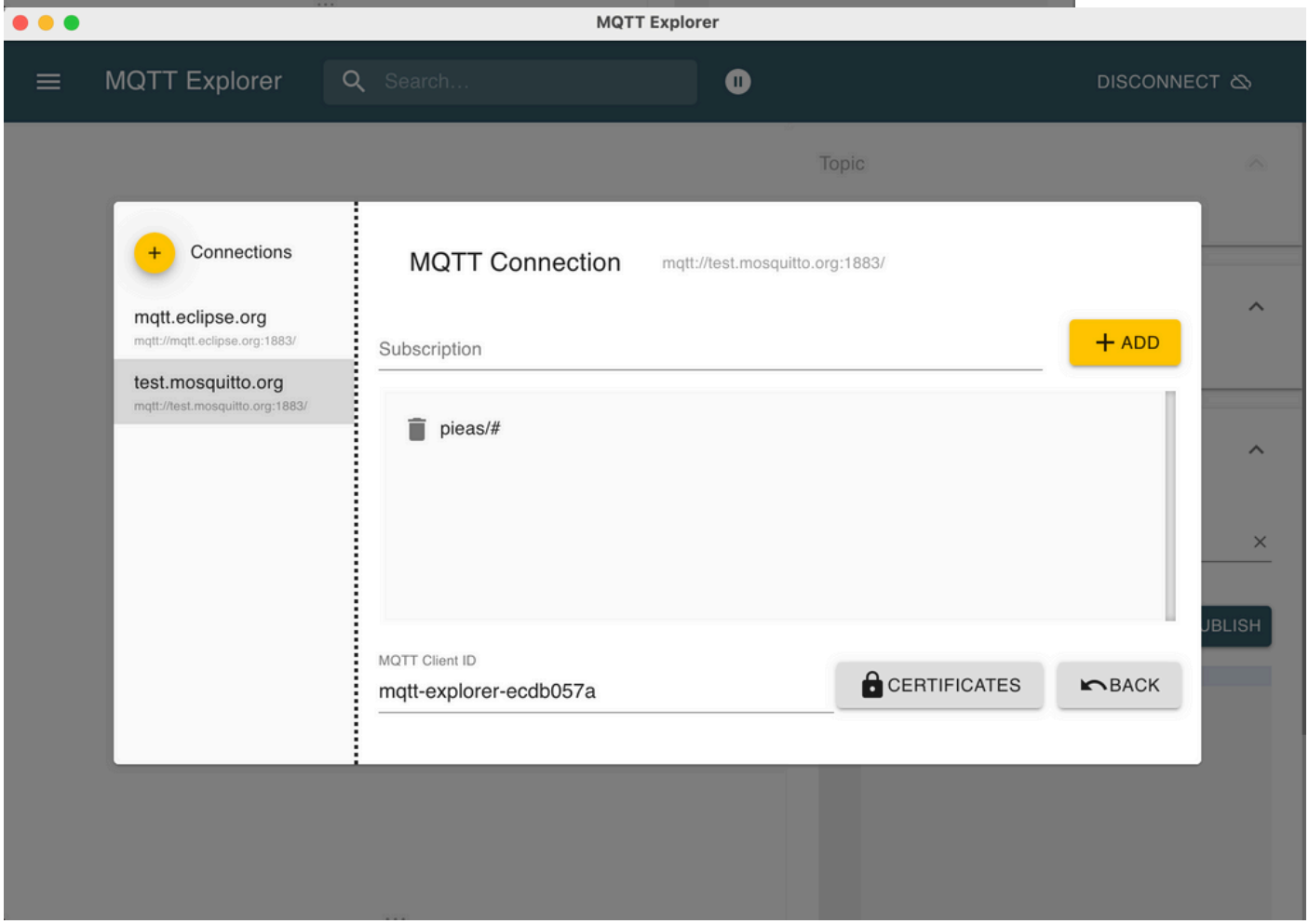
• Publisher: Sends messages to a topic.
• Subscriber: Listens for messages on a topic.
• Broker: Manages topics and routes messages from publishers to subscribers.

Public MQTT Broker

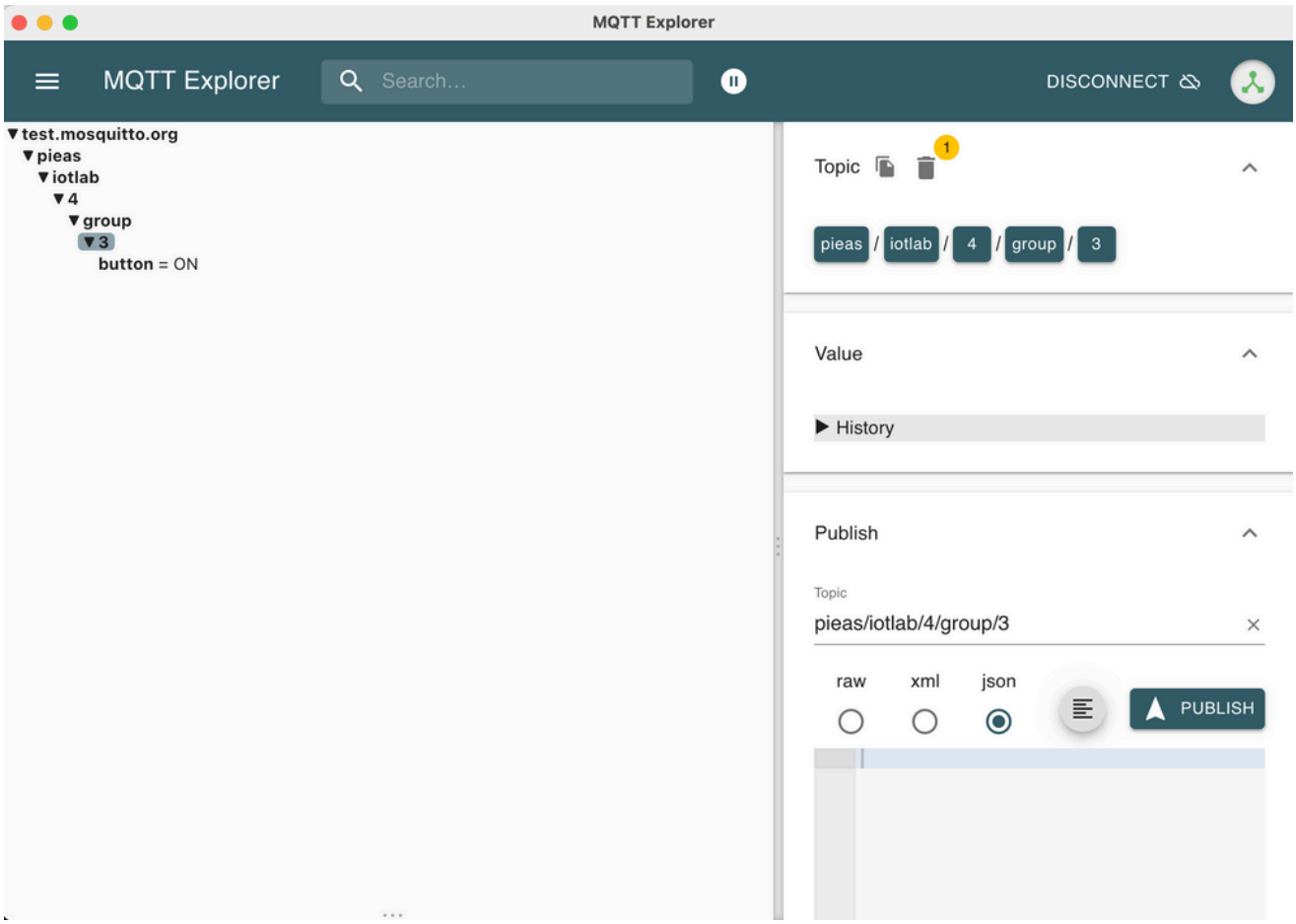
• For this lab, we will use a public MQTT broker such as test.mosquitto.org. No authentication is needed, which simplifies initial testing. Default MQTT broker port is 1883.

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)

LOLIN(WEMOS) D1 R2...

LAB04.ino

```
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("p.iot.0.567456")) { // Use a random unique device id
```

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

LOLIN(WEMOS) D1 R2...

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.iot.0.567456")) { // Use a random unique device id.
34   Serial.println("MQTT Connected");
35   } else {
```

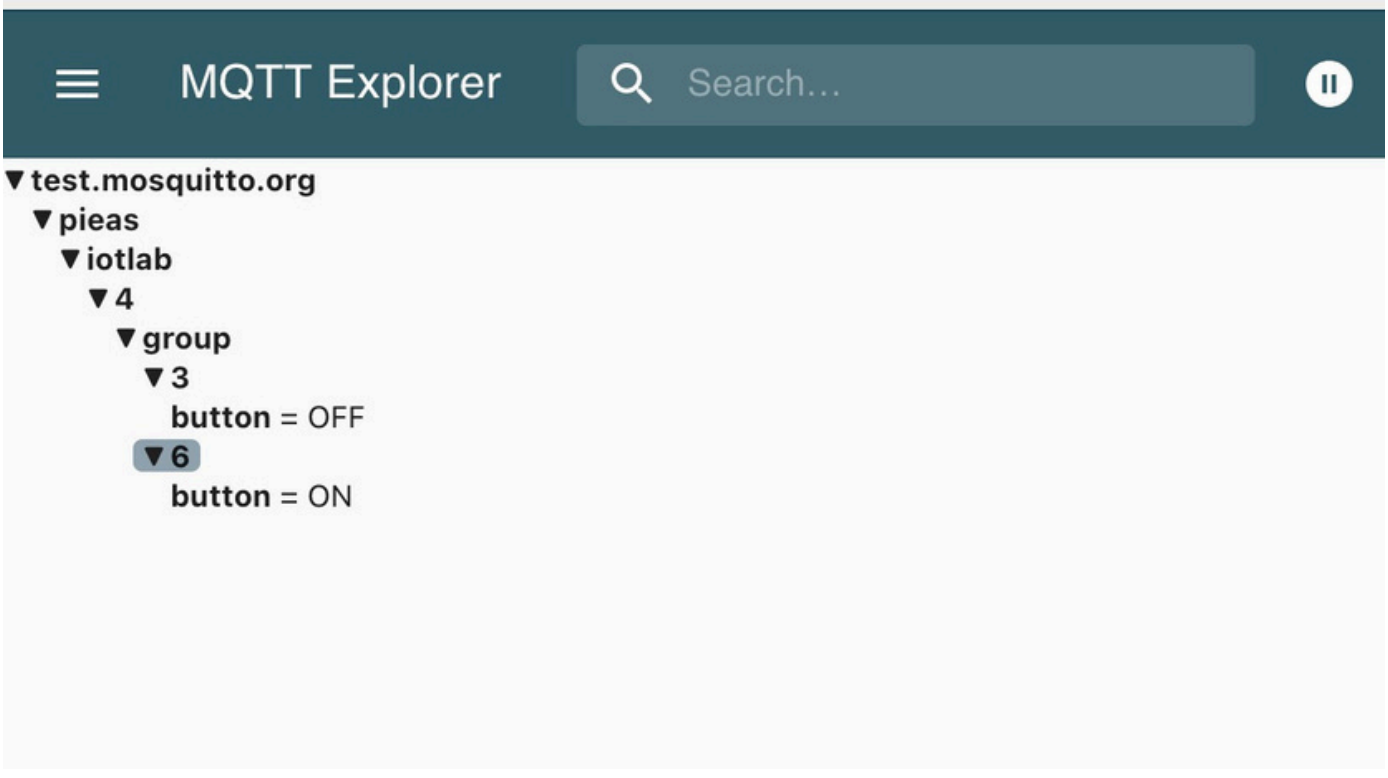
Output

CODE IMPLEMENTATION

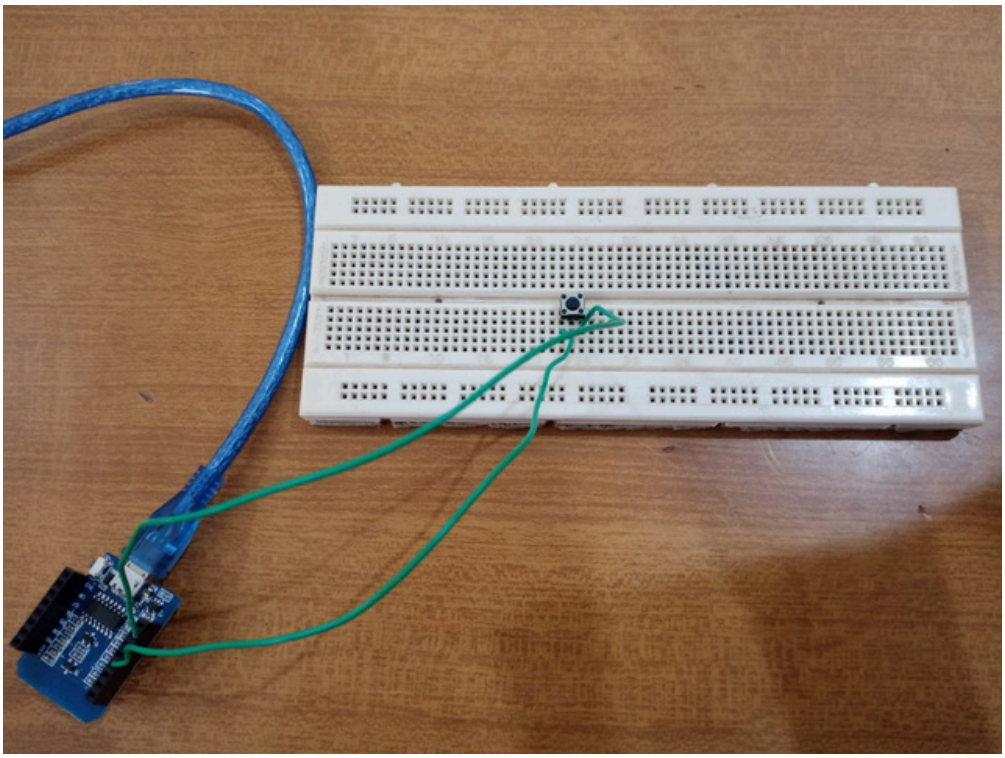
```
//Nida Pervaiz //Urwah Rasheed //Wajeeha Jahangir // Areeb Ur Rehman
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// Wi-Fi Credentials
const char* ssid = "IoT-Class";
const char* password = "iotclass1";
// MQTT Broker
const char* mqtt_server = "test.mosquitto.org"; // Check Broker
const int 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



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



Wiring:

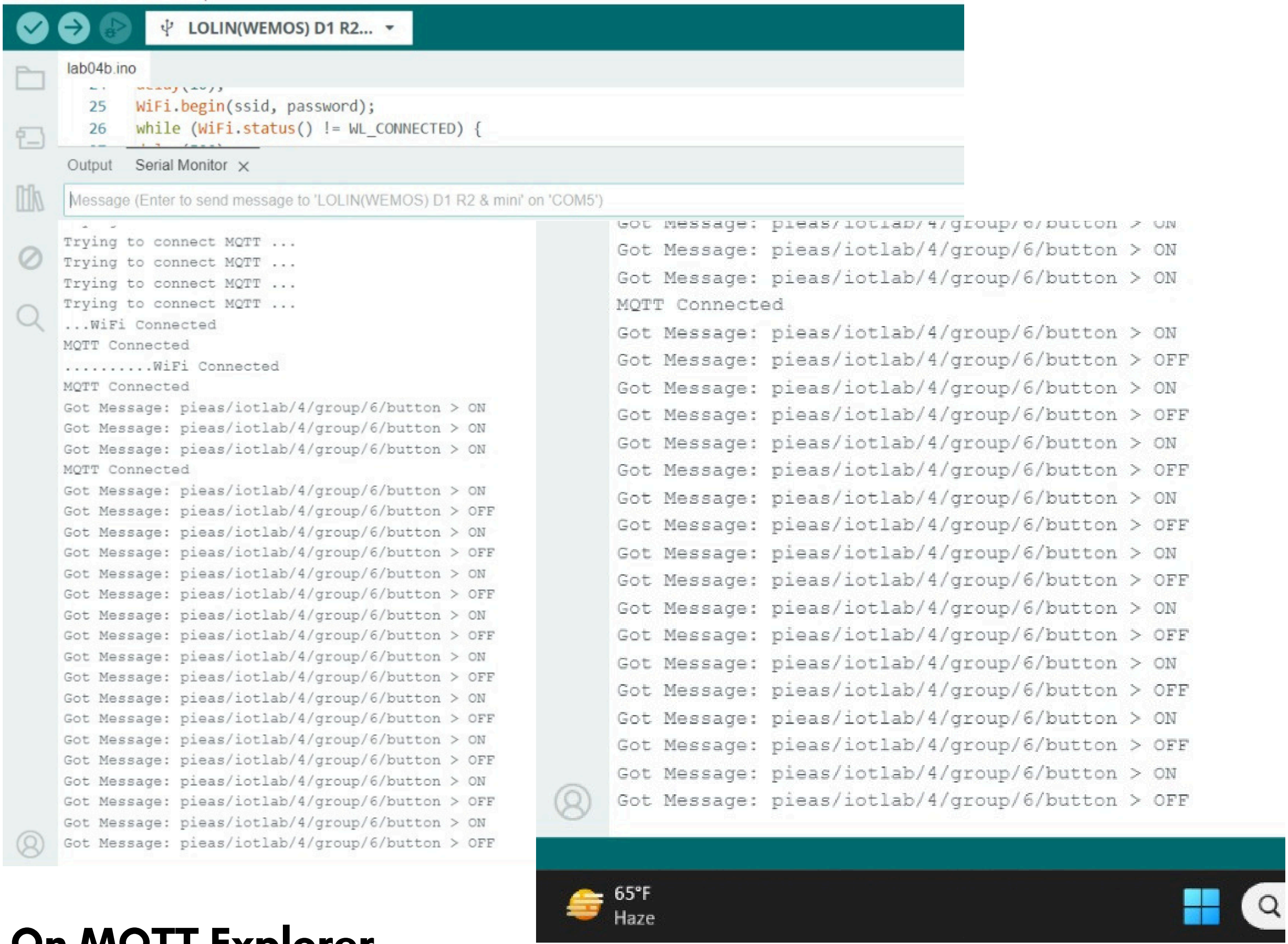
One pin of button with D2 pin of ESP8266. The other pin connected with ground. We used INPUT_PULLUP in the code instead of using a pull up resistor in hardware.

Part 3: Configure the MQTT Subscriber (LED)

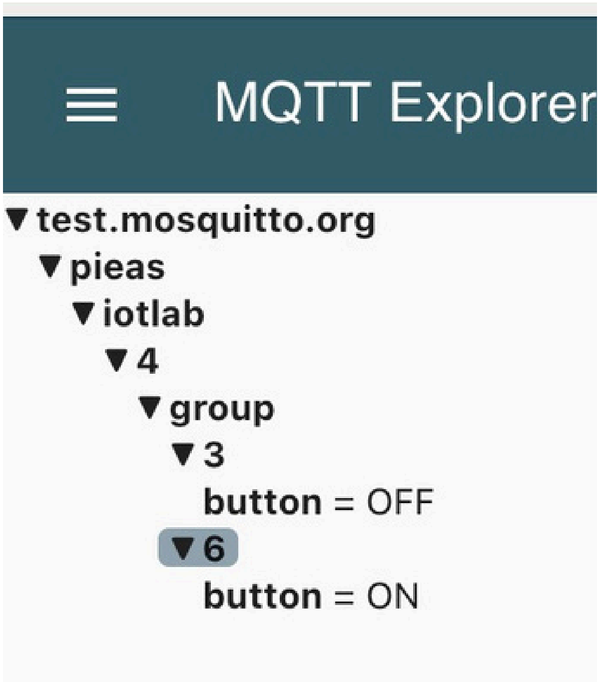
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// Wi-Fi Credentials (same as publisher)
const char* ssid = "OPPO A13";
const char* password = "12345678";
// MQTT Broker
const char* mqtt_server = "test.mosquitto.org"; // Check Broker
const int mqtt_port = 1883;
const char* topic = "pieas/iotlab/4/group/6/button"; // 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();
```

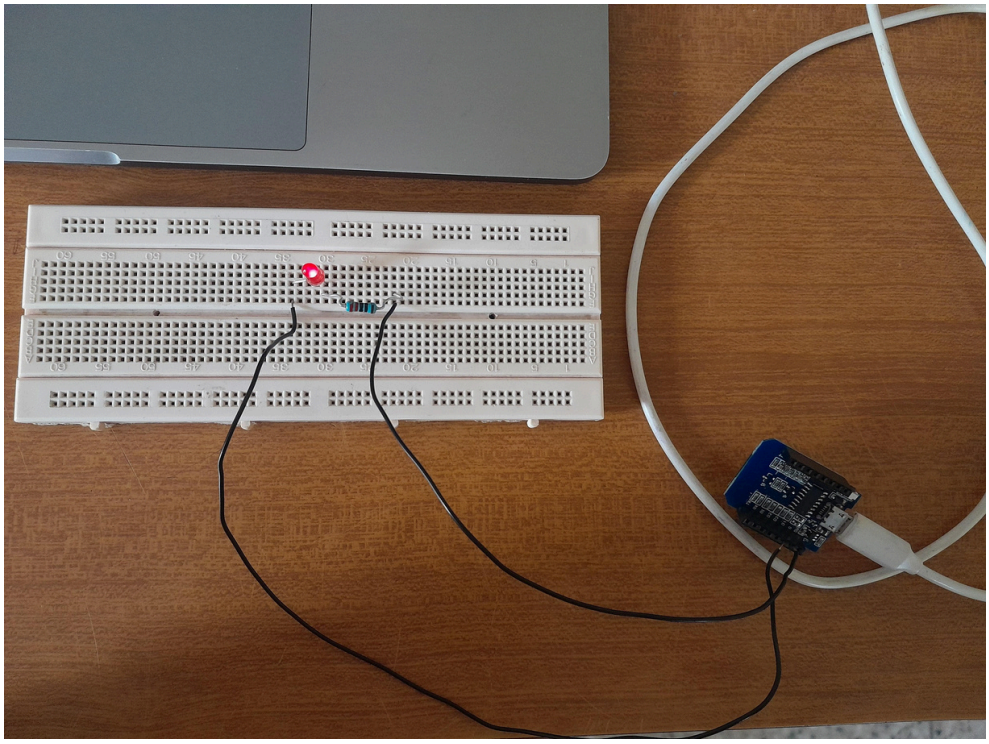
now we(group 3) served as subscriber and group 6 served as publisher



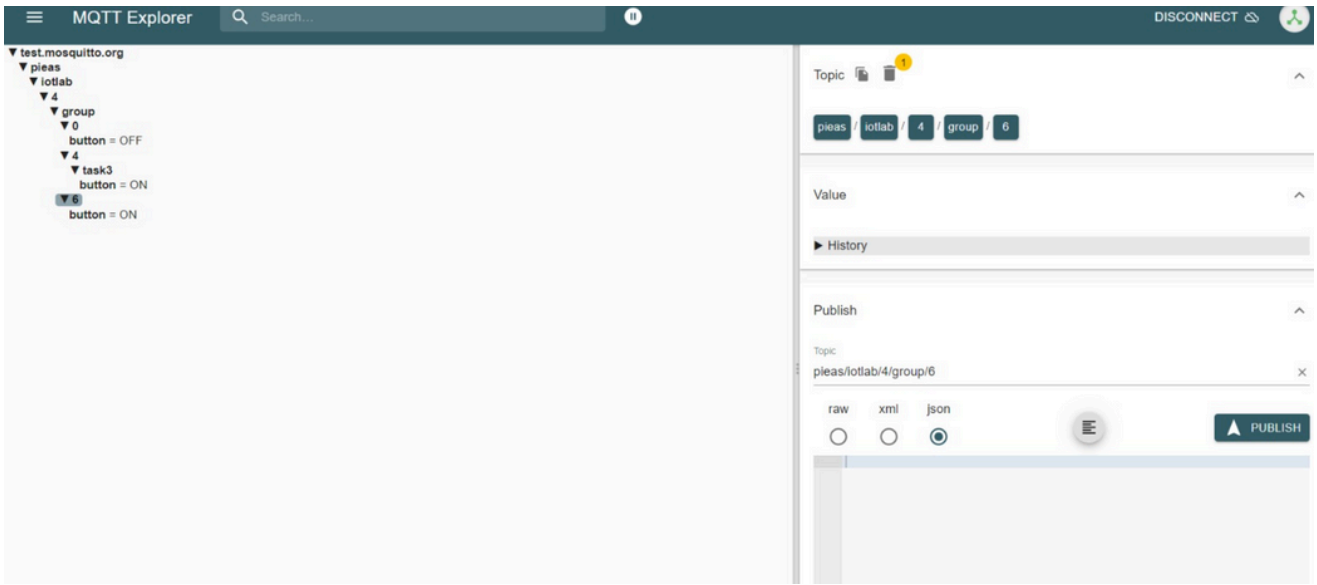
On MQTT Explorer



WIRING:



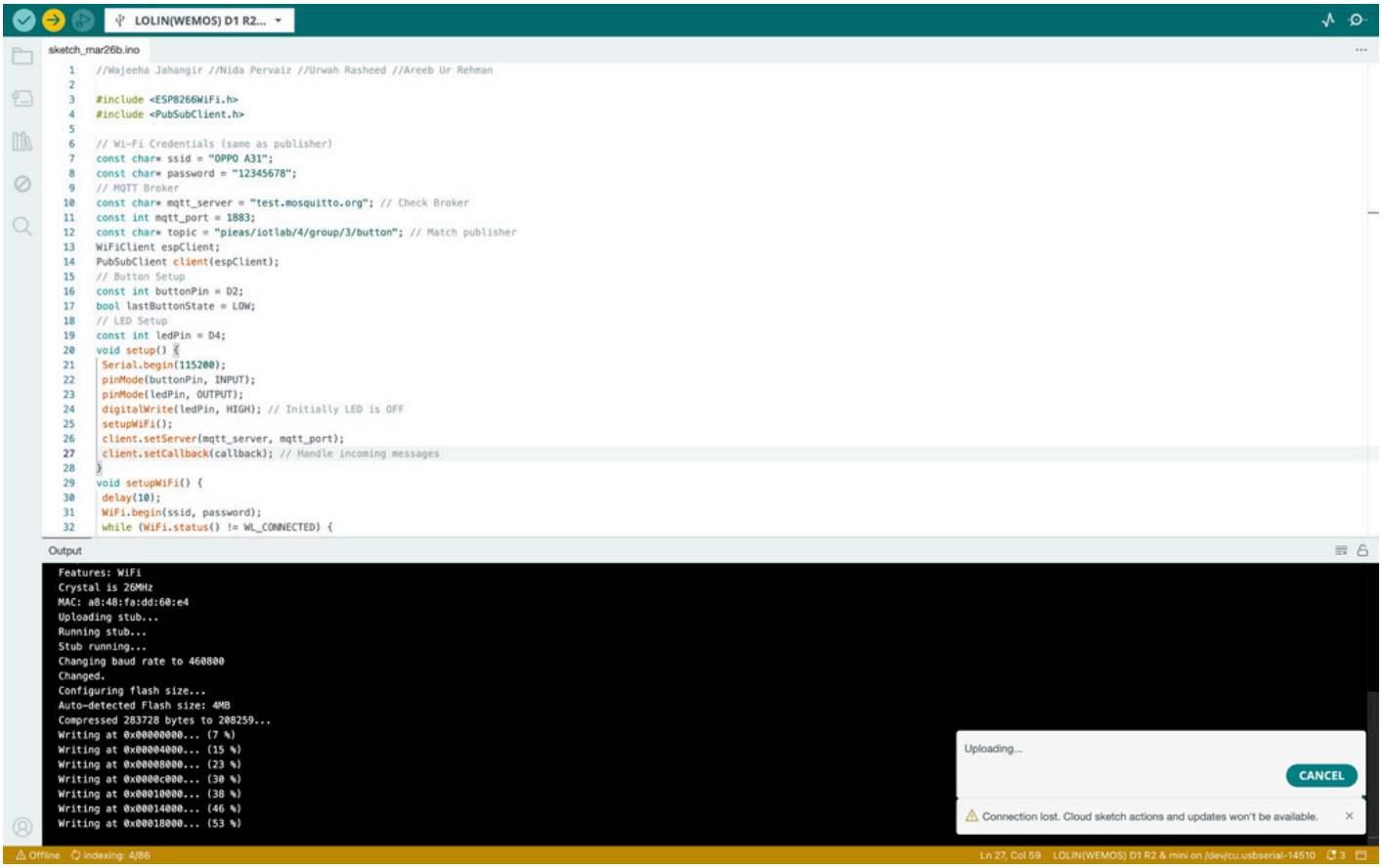
Anode (+) →220 ohm resistor
Cathode (-) → WeMos D1 GND pin
Resistor's other terminal → WeMos D1 digital pin (D4)

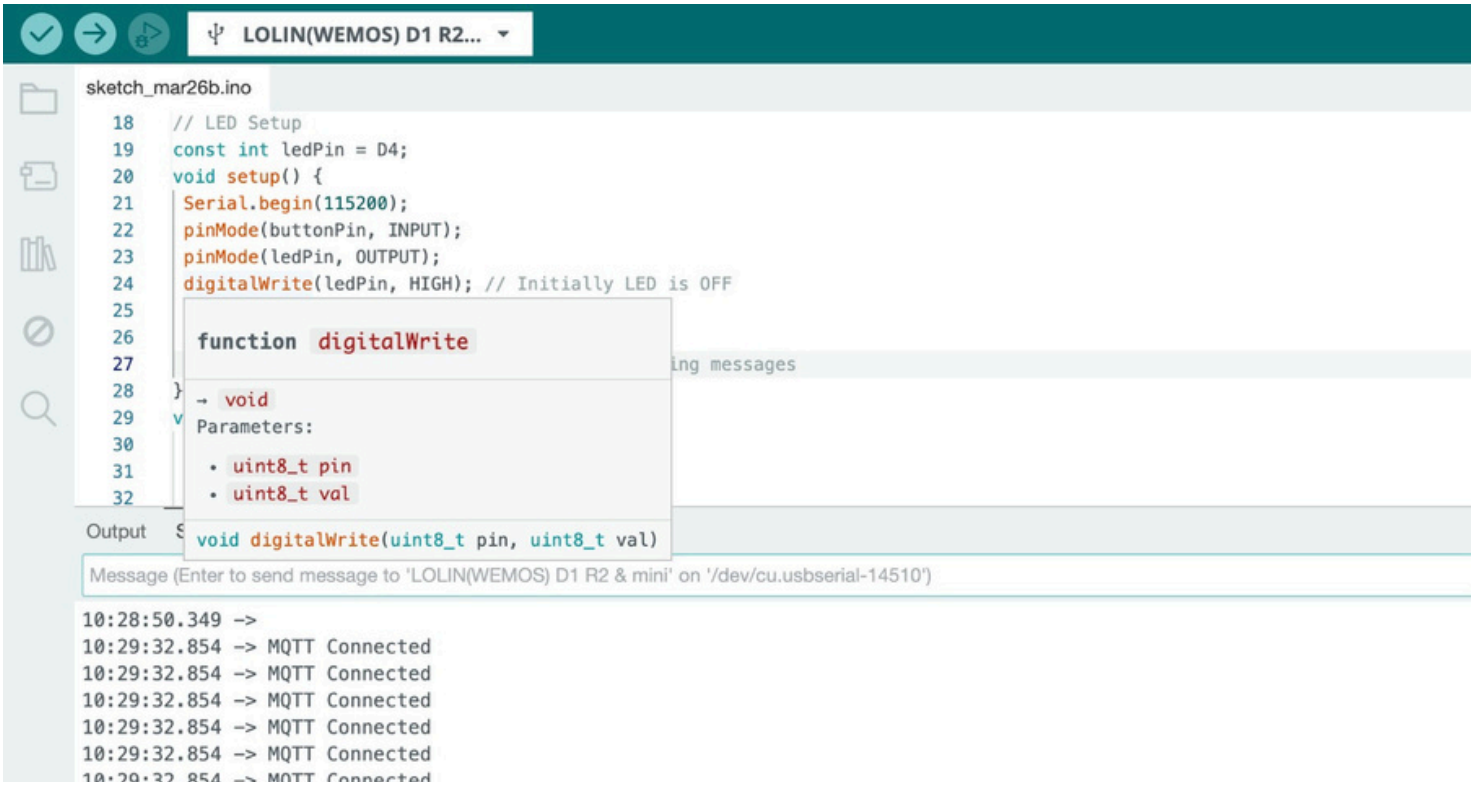


Part 4: Both Publisher and Subscriber on Single Device

```
//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
//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) {
Output
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 283728 bytes to 208259...
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...
CANCEL
Connection lost. Cloud sketch actions and updates won't be available.
Offline
indexing: 4/86
Ln 27, Col 59
LOLIN(WEMOS) D1 R2 & mini on /dev/cu.usbserial-14510

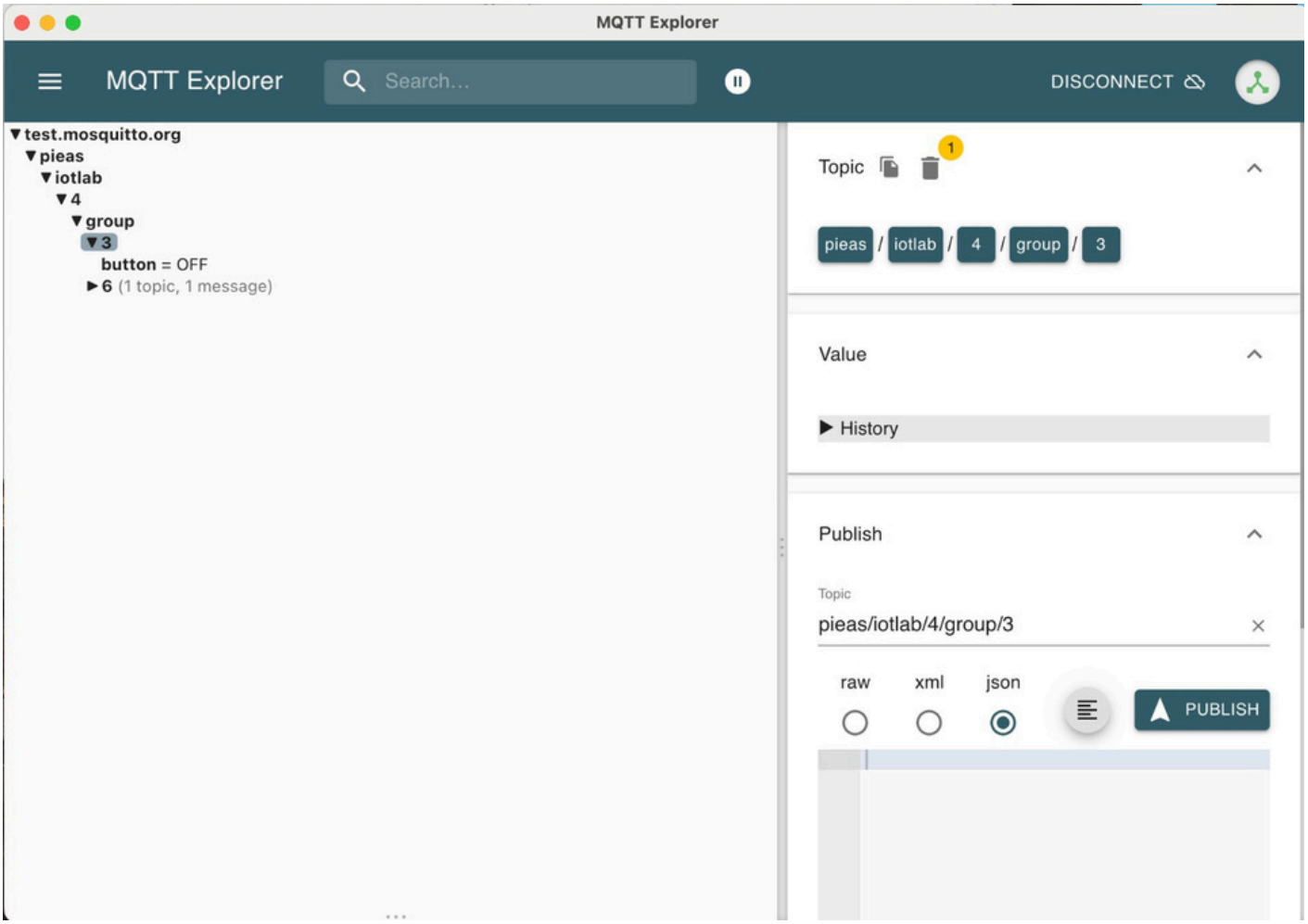

LOLIN(WEMOS) D1 R2...
sketch_mar26b.ino
// LED Setup
const int ledPin = D4;
void setup() {
Serial.begin(115200);
pinMode(buttonPin, INPUT);
pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, HIGH); // Initially LED is OFF
function digitalWrite
→ void
Parameters:
• uint8_t pin
• uint8_t val
void digitalWrite(uint8_t pin, uint8_t val)
ing messages
Output
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
10:29:32.854 -> MQTT Connected
10:29:32.854 -> MQTT Connected
10:29:32.854 -> MQTT Connected
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 -> ???????x?x?x.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

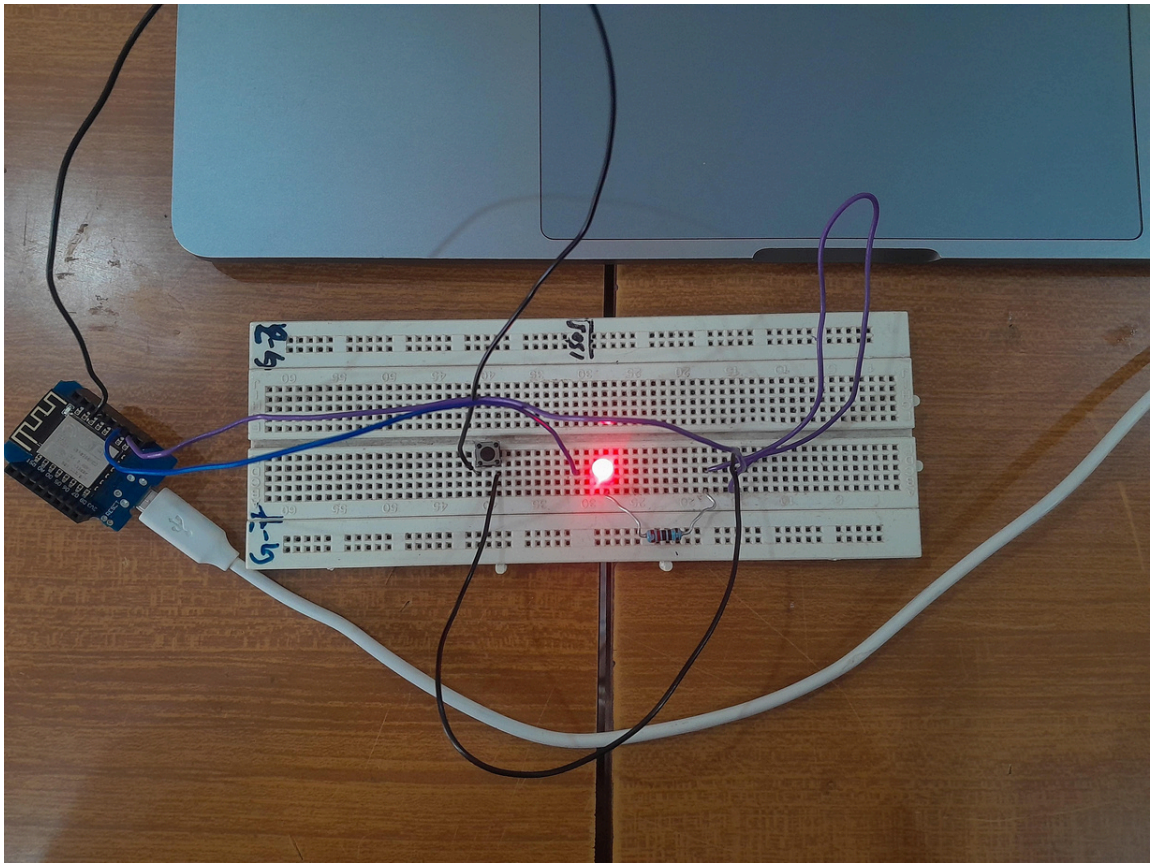
```
10:33:19.157 -> Trying to connect MQTT ...
10:33:28.301 -> MQTT Connected
10:33:28.301 -> Button is ON
10:33:42.191 -> Button is OFF
10:33:48.495 -> Got Message: pieas/iotlab/4/group/3/button > ON
```

ON MQTT EXPLORER



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);
    }
  }
}
}
```

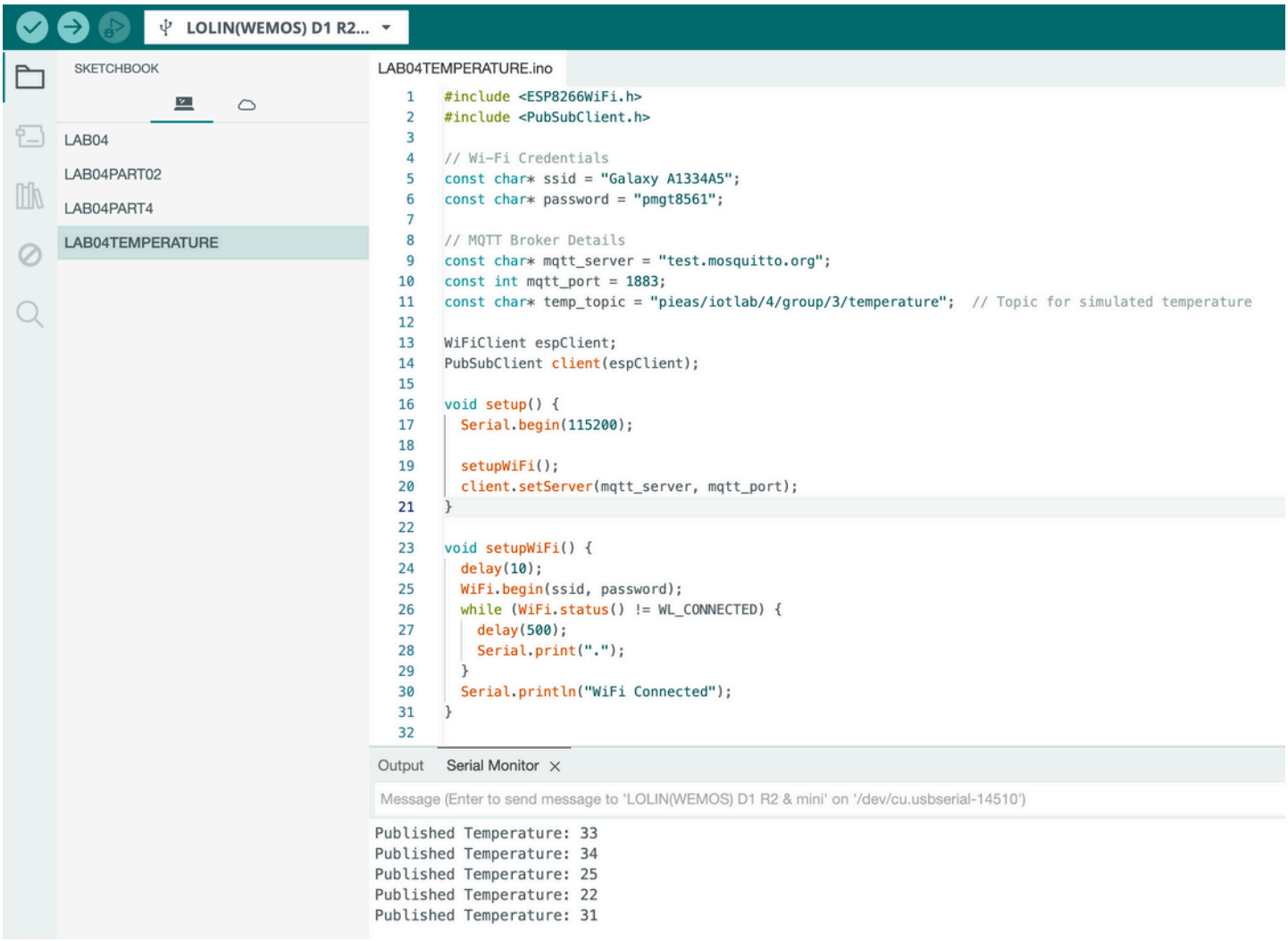
```
void loop() {
 if (!client.connected()) reconnect();
 client.loop();
 // Simulating DHT11 Temperature Reading
 int randomTemperature = random(20, 35); // Generates a random temperature between 20°C and 35°C
 char tempString[8];
 sprintf(tempString, "%d", randomTemperature); // Convert integer to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

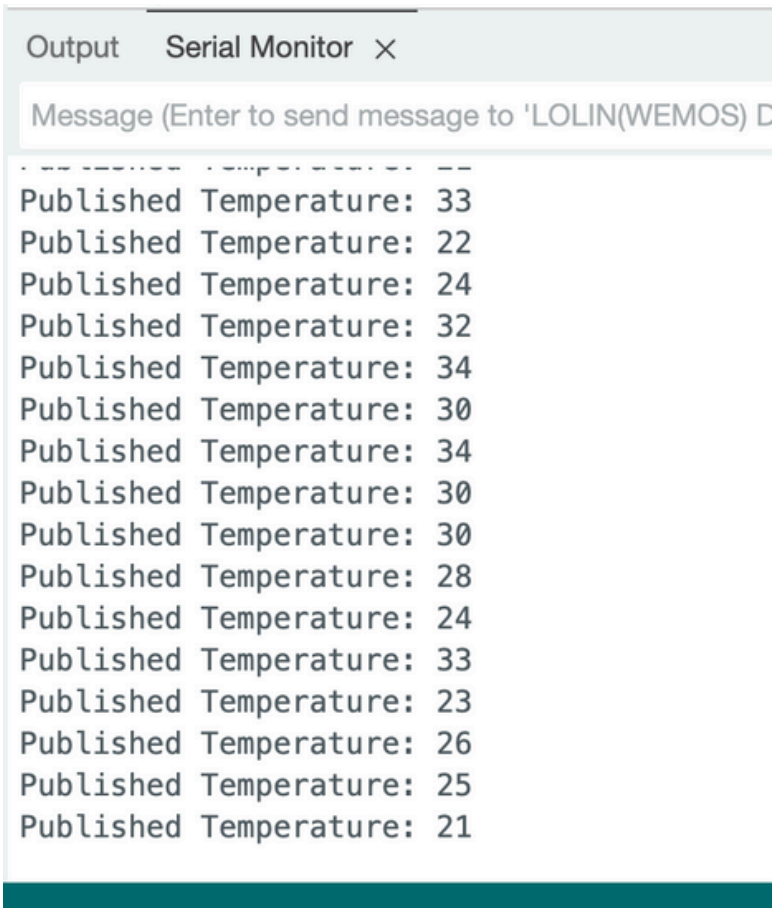
LAB04TEMPERATURE.ino

```
#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");
}

```

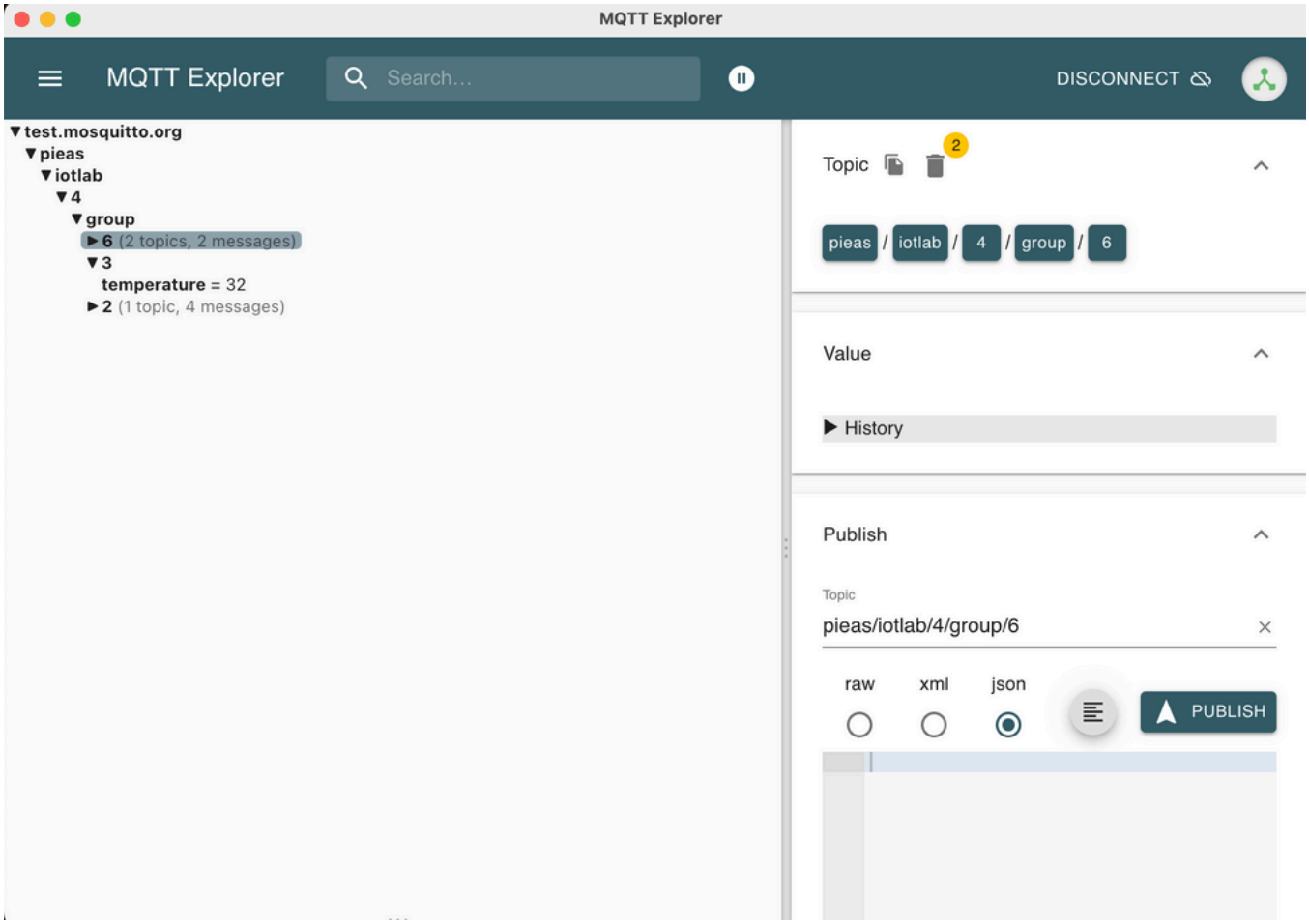
Output | Serial Monitor

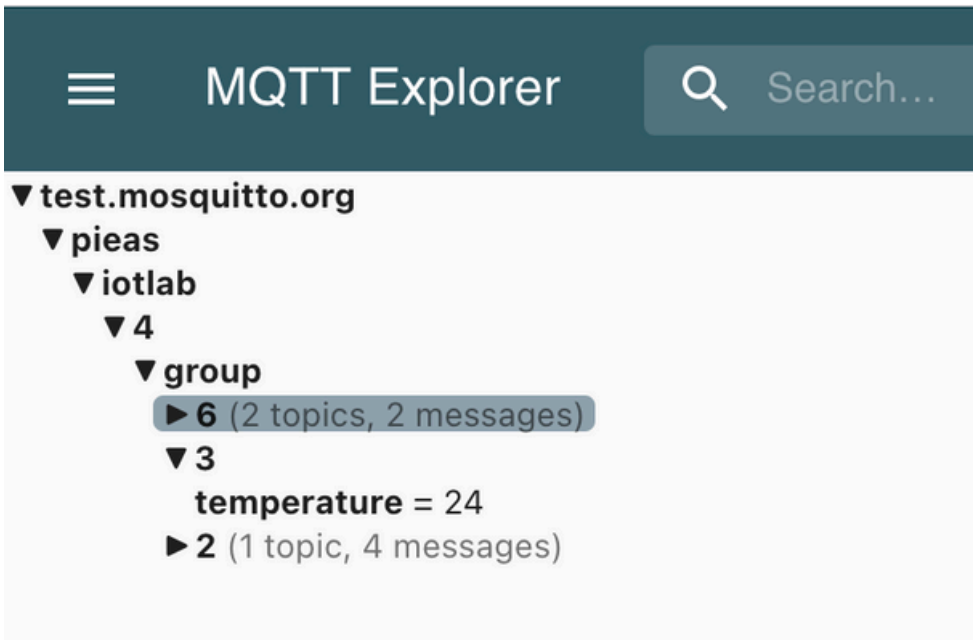
```
. Variables and constants in RAM (global, static), used 28476 / 80192 bytes (35%)
|   SEGMENT  BYTES    DESCRIPTION
|-- DATA     1512     initialized variables
|-- RODATA   1108     constants
`-- BSS      25856    zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 59747 / 65536 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
```


LOLIN(WEMOS) D1 R2...
SKETCHBOOK
LAB04
LAB04PART02
LAB04PART4
LAB04TEMPERATURE
LAB04TEMPERATURE.ino
#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");
}
Output
Serial Monitor
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

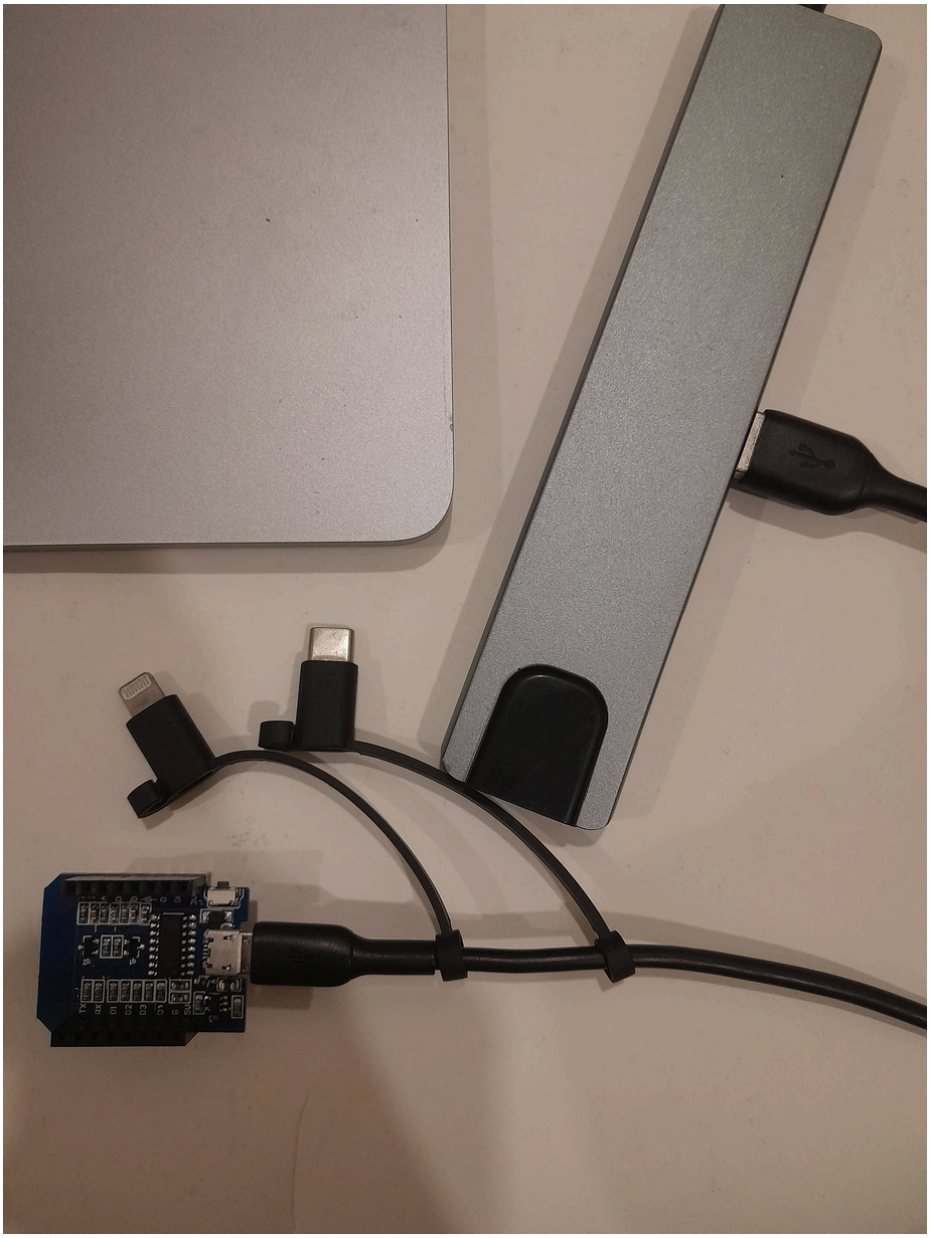

Output
Serial Monitor
Message (Enter to send message to 'LOLIN(WEMOS) D
Published Temperature: 33
Published Temperature: 22
Published Temperature: 24
Published Temperature: 32
Published Temperature: 34
Published Temperature: 30
Published Temperature: 34
Published Temperature: 30
Published Temperature: 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