

IoT Capstone Project Report

on

“LED Control Using ESP32 and MQTT via EMQX Broker”

Submitted by

Rakhi Yadav

Student at

Madhav Institute of Technology & Science, Gwalior

Submitted to

YHills IoT Capstone Program



IoT May Batch 2025

Chapter 1

Introduction

In this IoT capstone project, we demonstrate a simple and effective method to remotely control an LED using an ESP32 development board, the MQTT protocol, and an EMQX MQTT broker. The ESP32 connects to a Wi-Fi network and subscribes to a specific MQTT topic. A user can publish messages like "ON" or "OFF" to this topic from an MQTT client (such as MQTTX or a mobile app). When the ESP32 receives these messages, it toggles the LED accordingly.

This setup showcases the real-time communication capabilities of the MQTT protocol, which is ideal for IoT applications due to its lightweight design and low bandwidth usage. The project highlights the core principles of device-to-device communication, cloud-based messaging, and IoT automation.

By using EMQX, a high-performance MQTT broker, the system ensures reliable message delivery and fast response times. This kind of architecture is commonly used in smart home systems, remote device control, and sensor networks, where real-time responsiveness and low power consumption are critical.

Chapter 2

Objective

The main objective of this IoT project is to design and implement a system that allows **remote control of an LED** using an **ESP32 microcontroller** and the **MQTT protocol**, facilitated through an **EMQX MQTT broker**.

This project aims to:

1. **Demonstrate practical use of MQTT** for lightweight and real-time device communication.
2. **Use the ESP32** as a low-cost, Wi-Fi-enabled microcontroller to act as an MQTT subscriber.
3. **Implement a basic IoT control system** that receives messages and performs corresponding actions (turning an LED ON or OFF).
4. **Enable real-time communication** between a user-operated MQTT client (e.g., MQTTX or mobile app) and an ESP32-based IoT device.
5. Provide a **foundational understanding** of cloud-connected device control, suitable for scaling into smart home systems or larger IoT networks.

Chapter 3

Hardware and Software Used

1. Hardware Components

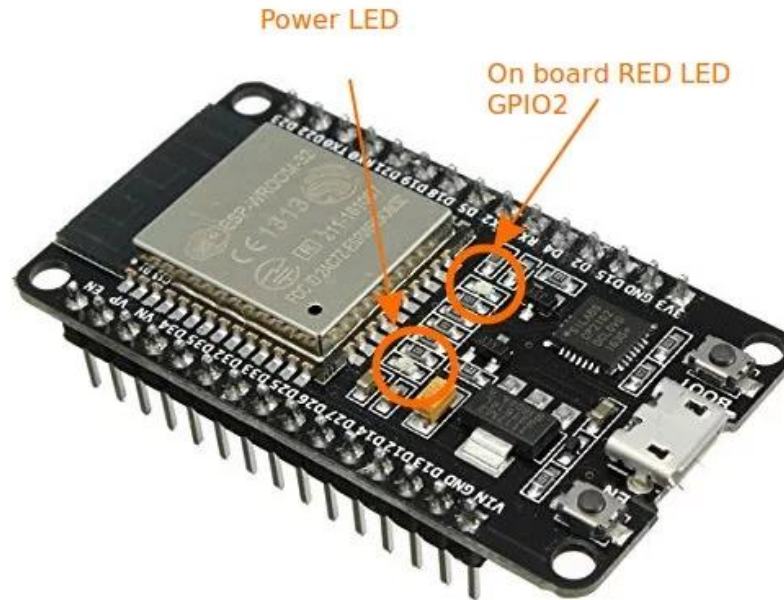
| Component | Description |
|----------------------------|--|
| ESP32 Development Board | A Wi-Fi-enabled microcontroller used to run the MQTT client and control the LED. |
| LED (Light Emitting Diode) | Indicates ON/OFF state based on MQTT messages. |
| 220-ohm Resistor | Limits current to prevent damage to the LED. |
| Breadboard | For non-permanent prototyping of the circuit. |
| Jumper Wires | Used to connect components on the breadboard. |
| Micro-USB Cable | For uploading code to ESP32 and power supply. |

2. Software Tools

| Software | Purpose |
|--|--|
| Arduino IDE | Used to write and upload code to the ESP32. |
| WiFi.h & PubSubClient.h Libraries | Enable Wi-Fi connectivity and MQTT communication. |
| EMQX MQTT Broker | Handles message exchange between clients and ESP32. |
| MQTTX (<i>or MQTT mobile app</i>) | Used as an MQTT client to send commands to the broker. |
| Serial Monitor (<i>in Arduino IDE</i>) | To debug and monitor ESP32 output. |

Chapter 4

Circuit Diagram



Chapter 5

How it works

1. **ESP32 connects to Wi-Fi:**

Upon powering up, the ESP32 initializes and connects to the configured Wi-Fi network using stored SSID and password.

2. **Connects to EMQX MQTT Broker:**

After Wi-Fi is connected, the ESP32 establishes an MQTT connection to the EMQX broker using the broker's IP address or URL and port (typically 1883).

3. **Subscribes to MQTT Topic:**

The ESP32 subscribes to a designated MQTT topic, for example, `/esp32/led`. This means it listens for any messages published on that topic.

4. **Waits for Incoming Messages:**

The ESP32 stays connected, waiting for new MQTT messages on the subscribed topic.

5. **Receives Message & Executes Callback:**

When a message arrives, a callback function is triggered. This function reads the message payload.

6. **Controls the LED Based on Message:**

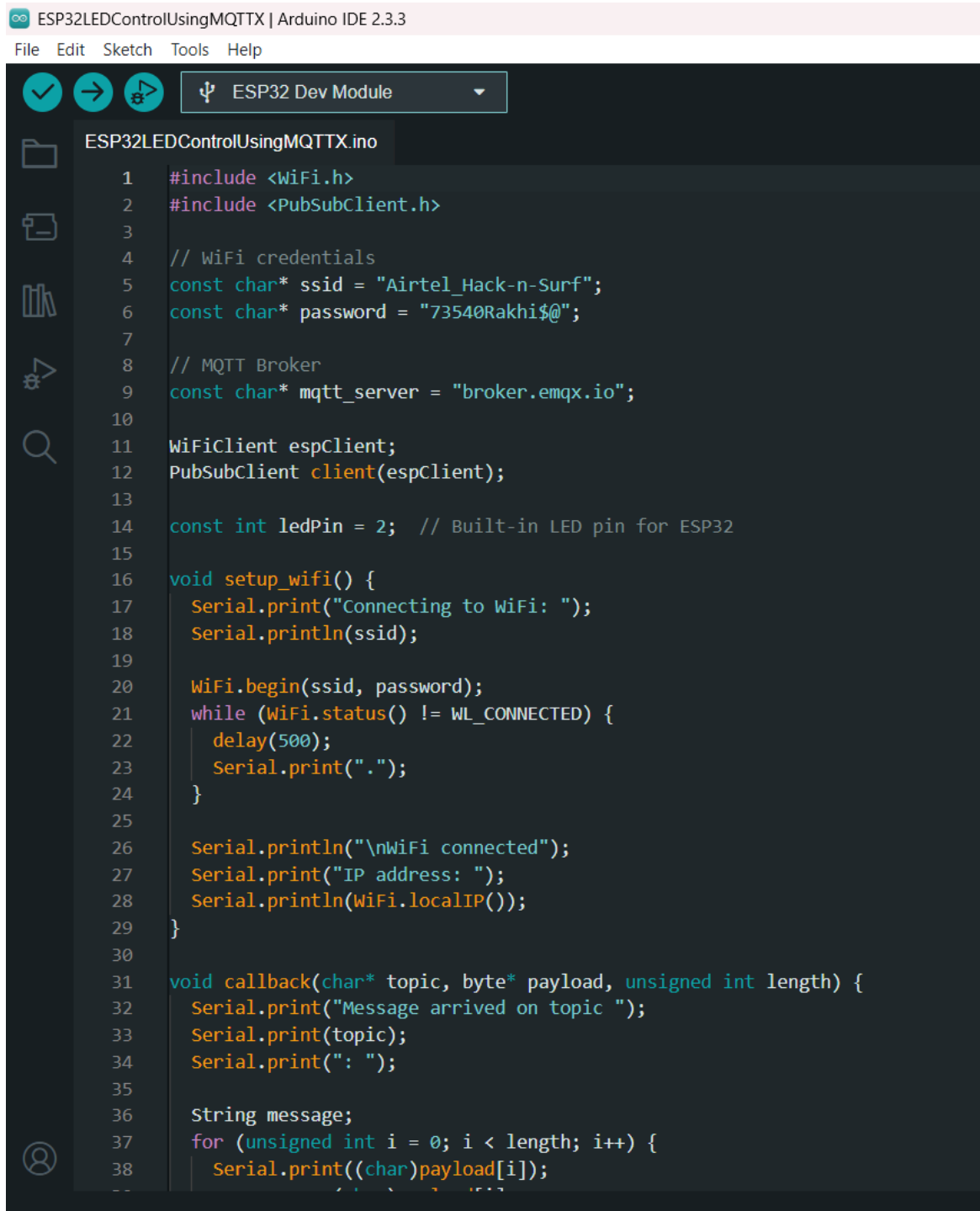
- If the message payload is "ON" or "1", the ESP32 sets the GPIO pin connected to the LED to HIGH, turning the LED ON.
- If the message payload is "OFF" or "2", the ESP32 sets the GPIO pin to LOW, turning the LED OFF.

7. **Loop Continues:**

The ESP32 continuously listens for new MQTT messages, enabling real-time control.

Chapter 6

Code Overview



```
ESP32LEDControlUsingMQTTX | Arduino IDE 2.3.3
File Edit Sketch Tools Help

ESP32LEDControlUsingMQTTX.ino
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  // WiFi credentials
5  const char* ssid = "Airtel_Hack-n-Surf";
6  const char* password = "73540Rakhi$@";
7
8  // MQTT Broker
9  const char* mqtt_server = "broker.emqx.io";
10
11  WiFiClient espClient;
12  PubSubClient client(espClient);
13
14  const int ledPin = 2; // Built-in LED pin for ESP32
15
16  void setup_wifi() {
17    Serial.print("Connecting to WiFi: ");
18    Serial.println(ssid);
19
20    WiFi.begin(ssid, password);
21    while (WiFi.status() != WL_CONNECTED) {
22      delay(500);
23      Serial.print(".");
24    }
25
26    Serial.println("\nWiFi connected");
27    Serial.print("IP address: ");
28    Serial.println(WiFi.localIP());
29  }
30
31  void callback(char* topic, byte* payload, unsigned int length) {
32    Serial.print("Message arrived on topic ");
33    Serial.print(topic);
34    Serial.print(": ");
35
36    String message;
37    for (unsigned int i = 0; i < length; i++) {
38      Serial.print((char)payload[i]);
39    }
40    Serial.println();
41  }
42
43  void loop() {
44    client.loop();
45  }
```


Chapter 7

Result & Output

1. Project Outcome

The project successfully demonstrated remote control of an LED connected to the ESP32 using MQTT messages sent via the EMQX broker. The system responded in real-time to commands sent from an MQTT client, confirming the correct functioning of the IoT communication setup.

2. Detailed Observations

- When the MQTT client published the message "ON" or "1" to the topic `/esp32/led`, the LED turned **ON** immediately.
- When the MQTT client published "OFF" or "2", the LED turned **OFF**.
- The ESP32 successfully maintained its connection to both the Wi-Fi network and the MQTT broker over extended periods.
- Message transmission and LED response latency were minimal, making it suitable for real-time applications.
- The MQTT broker (EMQX) handled message routing smoothly with no disconnections observed.

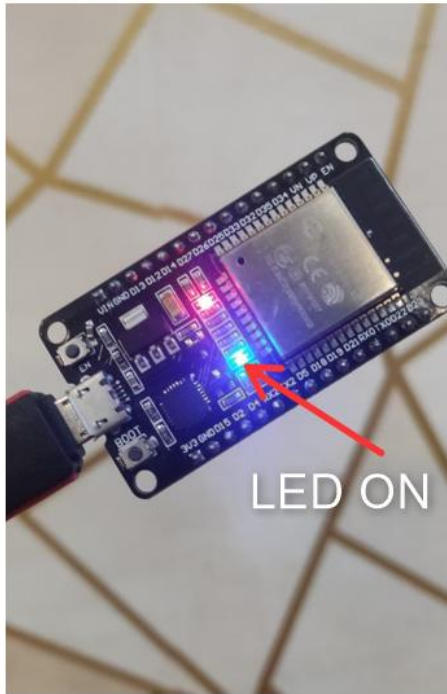
3. Summary Table

| MQTT Message Sent | LED Status | Expected Behavior |
|-------------------|---------------|---------------------------|
| ON or 1 | ON (LED lit) | LED turns on immediately |
| OFF or 2 | OFF (LED off) | LED turns off immediately |

4. Conclusion on Results

The project met its objectives by proving the concept of MQTT-based device control using ESP32 and EMQX. The system is reliable, responsive, and scalable for future enhancements.

5. Visuals



Chapter 8

Conclusion

This project successfully demonstrated the core principles of IoT communication using the ESP32, MQTT protocol, and EMQX broker to control an LED remotely. The implementation shows how lightweight messaging protocols like MQTT enable real-time and reliable device control over the internet.

The ESP32 was able to connect seamlessly to Wi-Fi and maintain an MQTT subscription to listen for control commands. Upon receiving valid messages, it executed the corresponding action of turning the LED ON or OFF, illustrating efficient two-way communication between devices and users.

This project highlights the practical aspects of:

- Setting up MQTT communication for IoT devices
- Using EMQX as a powerful, open-source MQTT broker
- Implementing real-time control of hardware using cloud-based messaging

The setup serves as a foundational model that can be extended for complex IoT systems such as smart home automation, remote sensor control, and data monitoring platforms.

Final Remarks:

- The project is reliable and easy to scale.
- Future enhancements could include sensor integration, multiple device control, and web-based dashboards.
- The use of MQTT ensures low bandwidth and power consumption, which are critical in IoT environments.