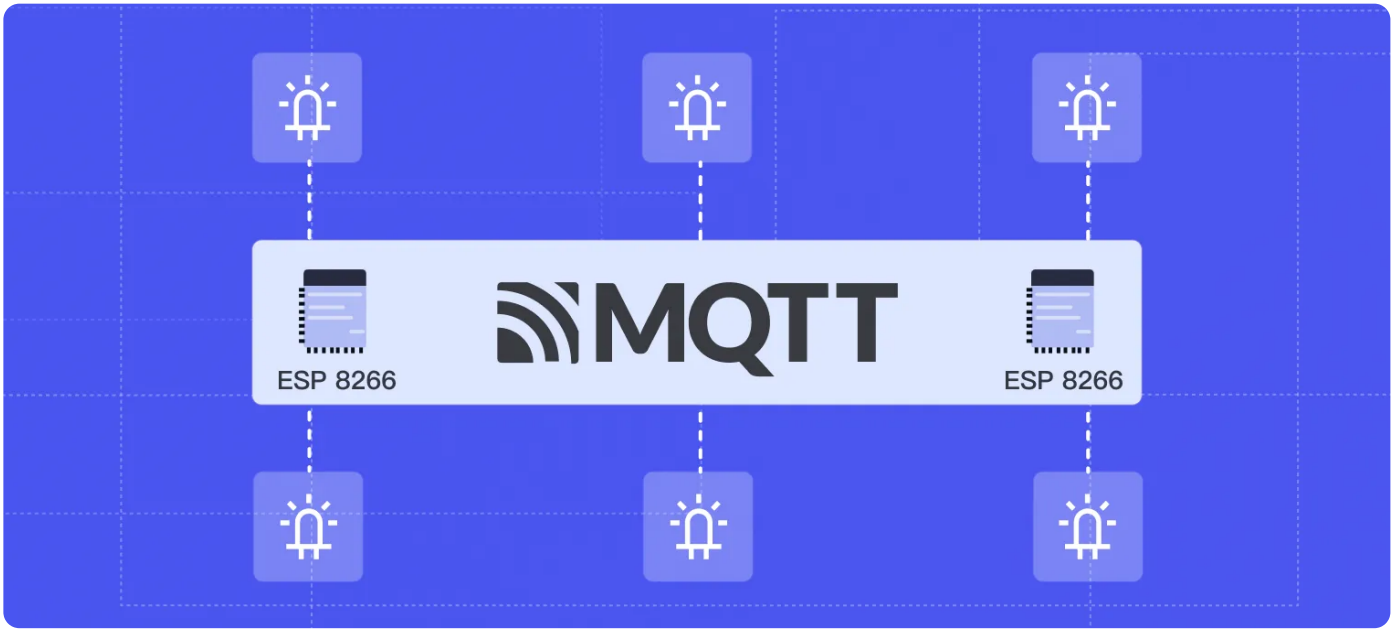
Remote control LED with ESP8266 and MQTT

By Dekun Tao 2021-03-25 Edit Feedback



MQTT is a lightweight and flexible IoT message exchanging and data delivery protocol. It dedicates to implementing a balance of flexibility and hardware/network resources for IoT developers.

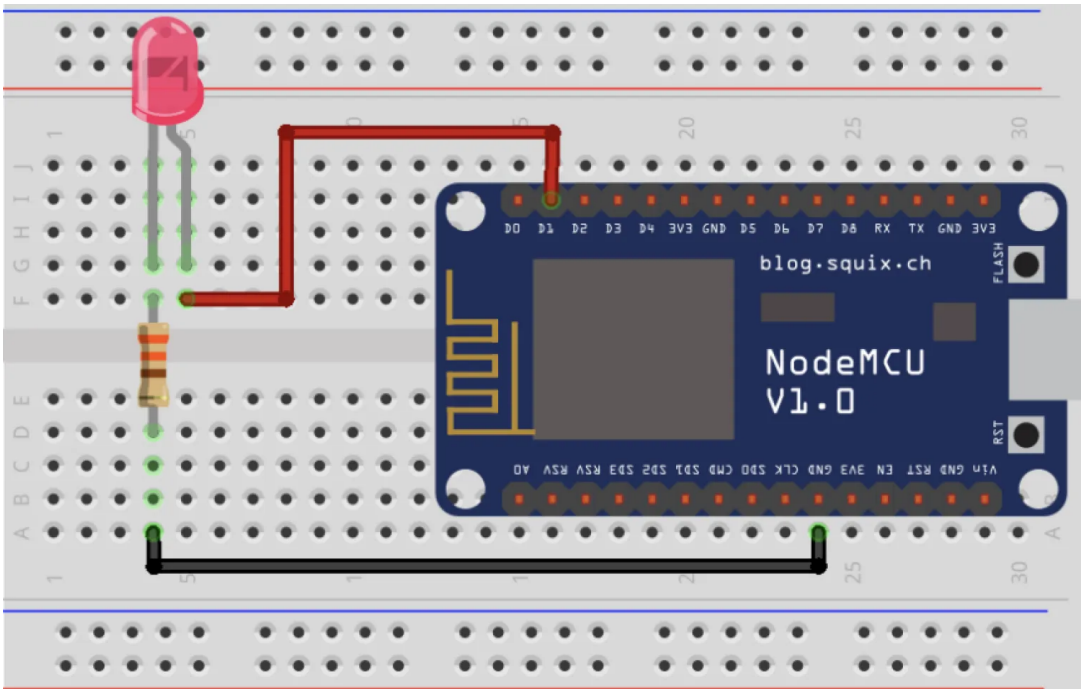
NodeMCU is an open-source IoT platform. It uses the Lua language to program. This platform is based on eLua open-source projects, and its underlying layer uses the ESP8266 SDK 0.9.5.

In this project, we will implement remote control LED lights via NodeMCU(ESP8266) and the free **public MQTT broker** which is operated and maintained by **EMQX Cloud**, and use the Arduino IDE to program NodeMCU ESP8266. EMQX Cloud is the **MQTT IoT Cloud Service Platform** launched by EMQ, which provides the **MQTT 5.0** access service with one-stop operations and maintenance managed and a unique isolated environment.

Required components

- NodeMCU
- Arduino IDE
- LED * 1， 330 Ω resistor
- **MQTT X**: Elegant cross-platform MQTT 5.0 client tool
- The free public MQTT broker
  - Broker: **broker.emqx.io**
  - TCP Port: **1883**
  - Websocket Port: **8083**

NodeMCU ESP8266 and LED connection diagram



Code writing

1. First, we will import the **ESP8266WiFi** and **PubSubClient** libraries. The ESP8266WiFi library can connect the ESP8266 to the WiFi network, and the PubSubClient library allows us to connect to the MQTT broker and publish/subscribe to topics.

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

2. We will use the **D1** pin of NodeMCU ESP8266 to connect to the LED. Actually, the inside of this pin is connected to **GPIO5** of the ESP8266 module.

```
// GPIO 5 D1
#define LED 5
```

3. Set the WIFI name and password as well as the MQTT Broker connection address and port.

```
// WiFi
const char *ssid = "mousse"; // Enter your WiFi name
const char *password = "qweqweqwe";  // Enter WiFi password

// MQTT Broker
const char *mqtt_broker = "broker.emqx.io";
const char *topic = "esp8266/led";
const char *mqtt_username = "emqx";
const char *mqtt_password = "public";
const int mqtt_port = 1883;
```

4. We have opened a serial connection to print the program's results and connect to the WiFi network.

```
// Set software serial baud to 115200;
Serial.begin(115200);
// connecting to a WiFi network
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
}
```

5. We will set the MQTT Broker and also print the connection information to the serial monitor.

```
 //connecting to a MQTT Broker
client.setServer(mqtt_broker, mqtt_port);
client.setCallback(callback);
while (!client.connected()) {
    String client_id = "esp8266-client-";
    client_id += String(WiFi.macAddress());
    Serial.println("Connecting to public emqx mqtt broker.....");
    if (client.connect(client_id, mqtt_username, mqtt_password)) {
        Serial.println("Public emqx mqtt broker connected");
    } else {
        Serial.print("failed with state ");
        Serial.print(client.state());
        delay(2000);
    }
}
```

6. After successfully connecting to the MQTT Broker, ESP8266 will publish and subscribe to the MQTT Broker.

```
// publish and subscribe
client.publish(topic, "hello emqx");
client.subscribe(topic);
```

7. Writing a callback function to read the sending commands from the serial monitor and control the LED on and off.

```
void callback(char *topic, byte *payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    String message;
```

```
    for (int i = 0; i < length; i++) {
        message = message + (char) payload[i];  // convert *byte to str
    }
    Serial.print(message);
    if (message == "on") { digitalWrite(LED, LOW); }   // LED on
    if (message == "off") { digitalWrite(LED, HIGH); } // LED off
    Serial.println();
    Serial.println("-----------------------");
}
```

8. Complete code.

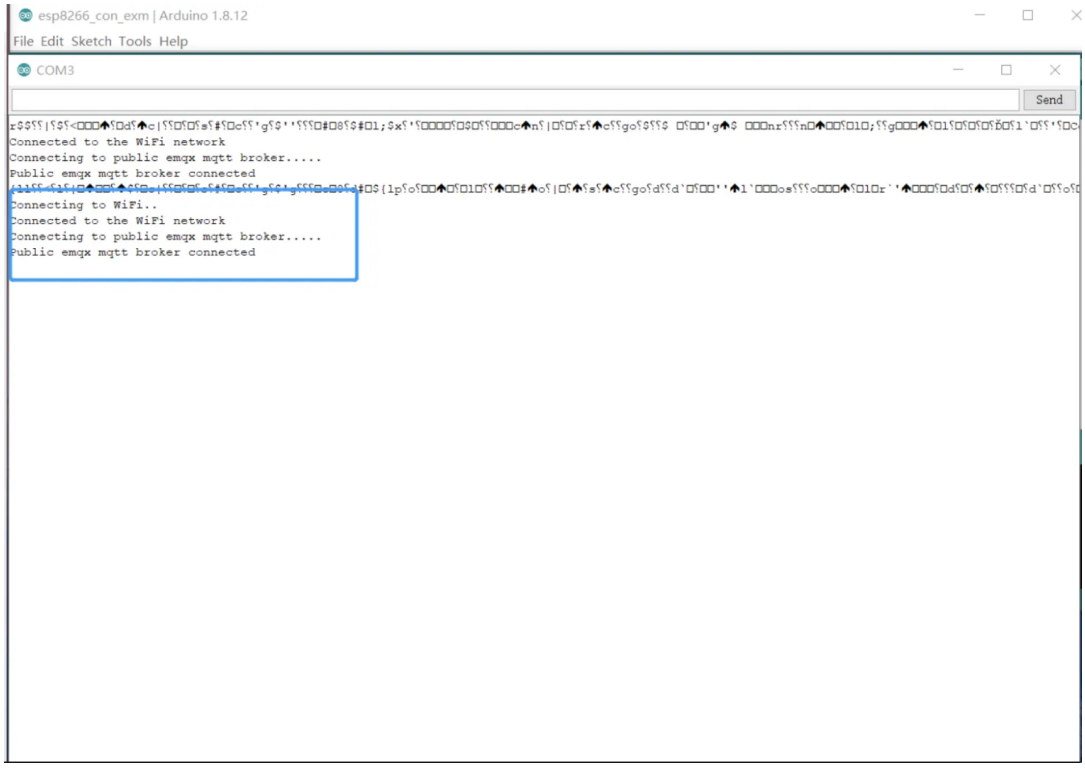
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// GPIO 5 D1
#define LED 5

// WiFi
const char *ssid = "mousse"; // Enter your WiFi name
const char *password = "qweqweqwe";  // Enter WiFi password

// MQTT Broker
const char *mqtt_broker = "broker.emqx.io";
const char *topic = "esp8266/led";
const char *mqtt_username = "emqx";
const char *mqtt_password = "public";
const int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
    // Set software serial baud to 115200;
    Serial.begin(115200);
    // connecting to a WiFi network
```

```
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.println("Connecting to WiFi..");
    }
    Serial.println("Connected to the WiFi network");
    //connecting to a mqtt broker
    client.setServer(mqtt_broker, mqtt_port);
    client.setCallback(callback);
    while (!client.connected()) {
        String client_id = "esp8266-client-";
        client_id += String(WiFi.macAddress());
        Serial.println("Connecting to public emqx mqtt broker.....");
        if (client.connect(client_id, mqtt_username, mqtt_password)) {
            Serial.println("Public emqx mqtt broker connected");
        } else {
            Serial.print("failed with state ");
            Serial.print(client.state());
            delay(2000);
        }
    }
    // publish and subscribe
    client.publish(topic, "hello emqx");
    client.subscribe(topic);
}

void callback(char *topic, byte *payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    String message;
    for (int i = 0; i < length; i++) {
        message = message + (char) payload[i];  // convert *byte to str
    }
    Serial.print(message);
    if (message == "on") { digitalWrite(LED, LOW); }   // LED on
    if (message == "off") { digitalWrite(LED, HIGH); } // LED off
```

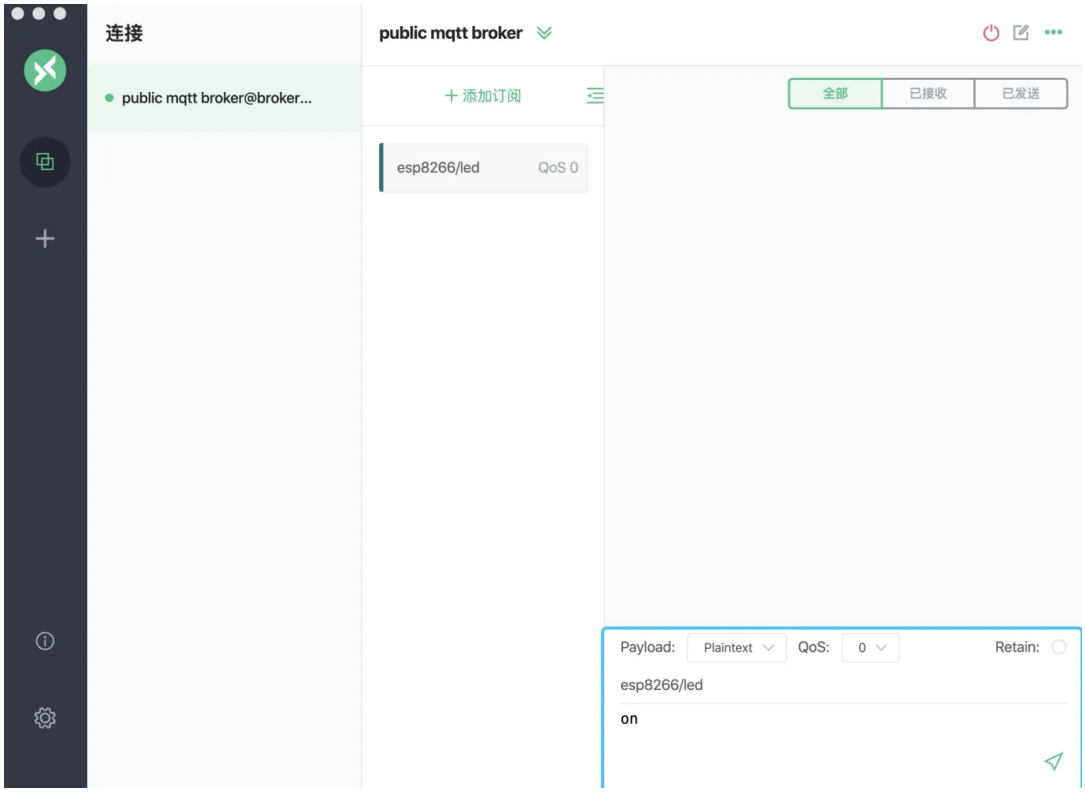
```
        Serial.println();
        Serial.println("-----------------------");
    }

    void loop() {
        client.loop();
    }
```

Connecting and testing

1. Please use **Arduino IDE** to upload the complete code to ESP8266 and open the serial monitor.



2. Establish the connection between the MQTT X client and MQTT Broker and send commands to the ESP8266.



Summary

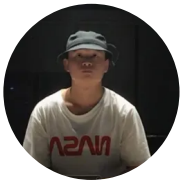
So far, we have successfully implemented remote control of the LED light using the NodeMCU ESP8266 and free public MQTT broker. This example only describes a simple scenario, while a more secure connection method and persistence of IoT data are needed in the actual projects.

Next, you can check out **The Easy-to-understand Guide to MQTT Protocol** series of articles provided by EMQ to learn about MQTT protocol features, explore more advanced applications of MQTT, and get started with MQTT application and service development.

ESP8266

Share

Article By



Dekun Tao

EMQX Cloud Engineer, programming in Python and Rust.

Related Posts

2020-05-22

ESP8266 connects to…

This project will implement connectin…

2020-10-12 Saiteng You

Comparison of Python MQTT…

This article collects three common Pytho…

2020-08-12 Saiteng You

Use MQTT with Raspberry Pi

This article introduces how to use Python to…