#include <ESP8266WiFi.h>

#include <PubSubClient.h>

// Wi-Fi credentials

const char\* ssid = "Cmf";

const char\* password = "03101992";

// MQTT broker details

const char\* mqtt\_server = "broker.hivemq.com"; // Public broker

const int mqtt\_port = 1883;

const char\* mqtt\_topic\_subscribe = "home/devices/control/#"; // Topic to receive control commands (wildcard for multiple devices)

const char\* mqtt\_topic\_publish\_prefix = "home/device/status"; // Prefix for device-specific publish topics

// Define the pins for the devices

const int devicePins[4] = {D0, D1, D2, D3}; // Replace with actual pins

// Wi-Fi and MQTT clients

WiFiClient espClient;

PubSubClient client(espClient);

// Function to handle incoming MQTT messages

void callback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Message received on topic: ");

Serial.println(topic);

String message;

for (int i = 0; i < length; i++) {

message += (char)payload[i];

}

//message.trim();

Serial.print("Message length: ");

Serial.println(message.length());

// Extract device ID from the topic

int deviceIndex = -1;

String deviceTopic = String(topic);

int separatorIndex = deviceTopic.indexOf('/');

if (separatorIndex > 0) {

String deviceId = deviceTopic.substring(separatorIndex + 1);

for (int i = 0; i < 4; i++) {

if (deviceId == String(i)) {

deviceIndex = i;

break;

}

}

}

// Control relay based on device ID and message

/\*if (deviceIndex >= 0 && (message == "ON" || message == "OFF")) {

digitalWrite(devicePins[deviceIndex], message == "ON" ? LOW : HIGH);

Serial.print("Device ");

Serial.print(deviceIndex);

Serial.println(message == "ON" ? " turned ON" : " turned OFF");

client.publish((mqtt\_topic\_publish\_prefix + String(deviceIndex)).c\_str(), message.c\_str());

} else {

Serial.println("Unknown command or invalid device ID");

client.publish((mqtt\_topic\_publish\_prefix + String("all")).c\_str(), "Unknown command or invalid device ID received");

}\*/

if (deviceIndex >= 0 && message == "ON" ){

digitalWrite(devicePins[deviceIndex], LOW); // Turn relay ON

Serial.println("Relay turned ON");

client.publish(mqtt\_topic\_publish\_prefix, "Device is ON");

} else if (deviceIndex >= 0 && message == "OFF") {

digitalWrite(devicePins[deviceIndex], HIGH); // Turn relay OFF

Serial.println("Relay turned OFF");

client.publish(mqtt\_topic\_publish\_prefix, "Device is OFF");

} else {

Serial.println("Unknown command");

client.publish((mqtt\_topic\_publish\_prefix + String("all")).c\_str(), "Unknown command or invalid device ID received" );

}

}

void setup() {

Serial.begin(115200);

// Initialize device pins

for (int pin : devicePins) {

pinMode(pin, OUTPUT);

digitalWrite(pin, HIGH); // Ensure relays are OFF initially

}

// Connect to Wi-Fi

Serial.print("Connecting to Wi-Fi");

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWi-Fi connected!");

// Set MQTT server and callback

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

// Connect to MQTT broker and subscribe to wildcard topic

while (!client.connected()) {

Serial.print("Connecting to MQTT broker...");

if (client.connect("ESP8266Client2075")) {

Serial.println("Connected to MQTT broker!");

if (client.subscribe(mqtt\_topic\_subscribe)) {

Serial.println("Subscribed to topic: home/devices/control/#");

} else {

Serial.println("Failed to subscribe");

}

} else {

Serial.print("Failed to connect, rc=");

Serial.println(client.state());

Serial.println("Retrying in 5 seconds...");

delay(5000);

}

}

}

void loop() {

// Maintain MQTT connection

if (!client.connected()) {

while (!client.connected()) {

Serial.print("Reconnecting to MQTT broker...");

if (client.connect("ESP8266Client2075")) {

Serial.println("Reconnected!");

client.publish((mqtt\_topic\_publish\_prefix + String("all")).c\_str(), "Connected");

client.subscribe(mqtt\_topic\_subscribe);

} else {

Serial.print("Failed, rc=");

Serial.println(client.state());

delay(5000);

}

}

}

client.loop();

}