```
// PARCIAL 8 - DISEÑO CON uP y uC. 2024-1.
// NOMBRE: Diego Andrés García Díaz.
// CÓDIGO: 2195533.
// ------------------------------------------------------------------------

#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
#include "credenciales.h"

// Configuración Wi-Fi:
// En archivo: "credenciales.h"

const int pinLED = 26;

WiFiClientSecure espClient;
PubSubClient client(espClient);

void setup_wifi() {
  delay(10); // Pequeña pausa al inicio
  Serial.println(); // Salto de línea en la consola serie
  Serial.print("Connecting to: "); // Mensaje de conexión a WiFi
  Serial.println(ssid); // Imprime el nombre de la red WiFi
  WiFi.begin(ssid, password); // Conecta a la red WiFi utilizando credenciales guardadas
  while (WiFi.status() != WL_CONNECTED) { // Espera hasta que se establezca la conexión
    delay(500); // Espera medio segundo
    Serial.print("."); // Imprime un punto cada medio segundo para indicar el intento de
conexión
  }
  Serial.println(""); // Salto de línea en la consola serie
  Serial.println("WiFi connected. "); // Mensaje de conexión exitosa a WiFi
  Serial.println("IP address: "); // Mensaje de dirección IP asignada
  Serial.println(WiFi.localIP()); // Imprime la dirección IP local asignada por el enrutador
}

void callback(char* Topic, byte* pay_Load, unsigned int Length) {
  Serial.print("Mensaje recibido en el canal: "); // Mensaje de recepción de mensaje MQTT
  Serial.println(Topic); // Imprime el nombre del canal MQTT donde se recibió el mensaje

  if (strcmp(Topic, CONTROL_LED) == 0) { // Comprueba si el mensaje fue recibido en el canal
de control LED
    int estado = pay_Load[0] - '0'; // Convierte el primer byte del mensaje a entero
    digitalWrite(pinLED, estado); // Enciende o apaga el LED según el estado recibido
    Serial.print("Estado LED: "); // Mensaje de estado del LED
    Serial.println(estado); // Imprime el estado del LED (1 o 0)
  }

  // Convierte a una cadena de caracteres para facilitar la comparación
  String mensaje = "";
  for (int i = 0; i < Length; i++) {
```

```
    mensaje += (char)pay_Load[i];
  }
  Serial.print("Mensaje: ");
  Serial.println(mensaje);

  // Compara el mensaje recibido y controla el LED en consecuencia
  if (mensaje.equals("abrir puerta")) {
    digitalWrite(pinLED, HIGH);  // Enciende el LED
    Serial.println("LED ON.");
  } else if (mensaje.equals("cerrar puerta")) {
    digitalWrite(pinLED, LOW);   // Apaga el LED
    Serial.println("LED OFF.");
  }
}

void reconnect() {
  while (!client.connected()) { // Loop hasta que se establezca la conexión con el servidor
MQTT
    Serial.print("Intentando conexión a MQTT..."); // Mensaje de intento de conexión MQTT
    if (client.connect("Cliente ESP32", mqtt_username, mqtt_password)) { // Intenta conectar
al servidor MQTT con usuario y contraseña
      Serial.println(""); // Salto de línea en la consola serie
      Serial.println("Conexión Exitosa."); // Mensaje de conexión exitosa a MQTT
      client.subscribe(CONTROL_LED); // Suscribe al cliente MQTT al canal de control LED
    } else {
      Serial.print("Conexión Fallida, rc = "); // Mensaje de falla de conexión MQTT
      Serial.print(client.state()); // Imprime el estado de conexión MQTT
      Serial.println("Intentando nuevamente... "); // Mensaje de reintento de conexión
      delay(5000); // Espera 5 segundos antes de intentar nuevamente
    }
  }
}
void setup() {
  pinMode(pinLED, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  espClient.setCACert(root_ca);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

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