```
// PARCIAL OPCIONAL (b) - 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_WiFi.h>

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

// Configuración del MQTT Broker:
const char* mqtt_server = "1e3162d2609342fdbfc6c92c89371f04.s1.eu.hivemq.cloud"; // Dirección
del servidor MQTT
const int mqtt_port = 8883; // Puerto seguro MQTT
const char* mqtt_username = "dagd - UIS"; // Usuario MQTT
const char* mqtt_password = "Uis2024-1"; // Contraseña MQTT

const char* CONTROL_LED = "LED_Control"; // Canal para controlar el LED
const char* HUMIDITY = "Humedad"; // Canal para enviar datos de humedad simulada

const int LED_1 = 26; // Pin donde está conectado el LED
const int IN_Pin = 27; // Pin analógico donde se leería la humedad (simulada en este caso)

// Certificado raíz para conexión segura MQTT
static const char* root_ca PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu
ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY
MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc
h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+
0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U
A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW
T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH
B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC
B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv
KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn
OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn
jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw
qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI
rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV
HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq
hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILlaS/V9lZL
ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ
3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK
```

```
NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5
ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur
TkXWStAmzOVyyghqpZXjFaH3pO3JLF+l+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC
jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc
oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1lvh+wjChP4kqKOJ2qxq
4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA
mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d
emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iItreGCc=
-----END CERTIFICATE-----
)EOF";

WiFiClientSecure espClient;
PubSubClient client(espClient);

unsigned long last_MSG = 0;
char MSG[50];

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
}

float readHumidity() {
  static int counter = 0; // Variable estática para contar llamadas a la función
  float humidity; // Variable para almacenar el valor simulado de humedad

  // Generar valores simulados diferentes cada vez que se llama la función
  switch (counter % 3) { // Selecciona un caso basado en el contador módulo 3
    case 0:
      humidity = 42.0; // Valor simulado de humedad para el caso 0
      break;
    case 1:
      humidity = 66.9; // Valor simulado de humedad para el caso 1
      break;
    case 2:
      humidity = 58.2; // Valor simulado de humedad para el caso 2
      break;
    default:
```

```
      humidity = 45.5; // Valor por defecto si no coincide ningún caso
      break;
  }

  counter++; // Incrementa el contador para la próxima llamada
  return humidity; // Devuelve el valor simulado de humedad
}

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(LED_1, 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)
  }
  else if (strcmp(Topic, "HUMIDITY") == 0) { // Comprueba si el mensaje fue recibido en el
canal de humedad
    float humidity = readHumidity(); // Lee el valor simulado de humedad
    Serial.println("Valor de Humedad: " + String(humidity)); // Imprime el valor de humedad
en la consola serie
    client.publish("HUMIDITY", String(humidity).c_str(), true); // Publica el valor de
humedad en el canal MQTT
  }
}

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
      client.subscribe(HUMIDITY); // Suscribe al cliente MQTT al canal de humedad
    } 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(LED_1, OUTPUT); // Configura el pin del LED como salida
```

```
  Serial.begin(115200); // Inicia la comunicación serial a 115200 baudios
  setup_wifi(); // Configura la conexión WiFi
  espClient.setCACert(root_ca); // Configura el certificado raíz para conexión segura MQTT
  client.setServer(mqtt_server, mqtt_port); // Configura el servidor MQTT y puerto
  client.setCallback(callback); // Configura la función de callback para manejar mensajes
entrantes
}

void loop() {
  if (!client.connected()) { // Si no está conectado al servidor MQTT
    reconnect(); // Intenta reconectar
  }
  client.loop(); // Maneja la comunicación con el servidor MQTT

  // Publica el valor de humedad cada cierto tiempo
  unsigned long now = millis(); // Obtiene el tiempo actual en milisegundos
  if (now - last_MSG > 2000) { // Si han pasado más de 2 segundos desde la última publicación
    last_MSG = now; // Actualiza el tiempo de la última publicación

    float humidity = readHumidity(); // Lee el valor simulado de humedad
    Serial.println("Valor de Humedad: " + String(humidity)); // Imprime el valor de humedad
en la consola serie
    client.publish("HUMIDITY", String(humidity).c_str(), true); // Publica el valor de
humedad en el canal MQTT
  }
}
```