**1. Introducción**

El código implementa un sistema IoT con las siguientes funcionalidades:

* **Sensor DHT11**: Mide temperatura y humedad.
* **Conexión MQTT**: Publica y recibe mensajes de un broker.
* **Actualización OTA (Over The Air)**: Permite actualizar el firmware de manera remota.
* **Control de LED RGB**: Permite ajustar brillo y color mediante MQTT o un botón.
* **Gestión WiFi**: Conexión automática a una red WiFi especificada.

**Librerías Utilizadas**

Para garantizar el correcto funcionamiento del sistema, se han utilizado las siguientes librerías:

**En el ESP32:**

* **WiFi.h**: Para establecer y gestionar la conexión WiFi.
* **PubSubClient.h**: Para la comunicación MQTT con el broker.
* **ArduinoJson.h**: Para gestionar la codificación y decodificación de datos JSON.
* **DHTesp.h**: Para interactuar con el sensor DHT de temperatura y humedad.
* **HTTPUpdate.h**: Para implementar actualizaciones OTA (Over-The-Air) del firmware en dispositivos como el ESP32 o ESP8266. Permite que el dispositivo descargue e instale un nuevo firmware desde un servidor remoto sin necesidad de conexión física.
* **HTTPClient.h**: Permite que el ESP32 se comunique con servidores HTTP/HTTPS para realizar solicitudes web como GET, POST, PUT o DELETE.

**En Node-RED:**

* **Node-RED-Dashboard**: Para crear interfaces gráficas interactivas.
* **MQTT**: Para suscribirse y publicar mensajes en el broker.
* **JSON**: Para interpretar los datos recibidos en formato JSON.
* **Debug**: Para la depuración en tiempo real.

**Descripción del Sistema**

**Arquitectura General**

El sistema consta de los siguientes componentes principales:

1. **ESP32**: Dispositivo IoT que recoge datos de sensores y los envía a través del protocolo MQTT.
2. **Broker MQTT**: Punto de interconexión para la transmisión de mensajes entre dispositivos y Node-RED.
3. **Node-RED**: Plataforma de desarrollo que permite visualizar datos y enviar comandos al ESP32.

**Funcionamiento**

1. **Inicialización**: El ESP32 conecta a una red WiFi y establece una conexión con el broker MQTT utilizando credenciales preconfiguradas.
2. **Lectura de Sensores**: El ESP32 recopila datos de sensores conectados, como temperatura, humedad, etc.
3. **Envío de Datos**: Los datos se envían al broker MQTT en formato JSON.
4. **Recepción y Visualización**: Node-RED recibe los datos y los muestra en un panel gráfico.
5. **Control Remoto**: Los usuarios pueden enviar comandos desde Node-RED al ESP32 para activar o desactivar dispositivos.

**Técnicas de Robustez Implementadas**

**Manejo de Fallos de Sensores**

* **Verificación de Datos Nulos**: Se comprueba si los sensores devuelven datos válidos antes de procesarlos. Si no, se registra un mensaje de error y se reintenta la lectura.
* **Reintentos Configurables**: Se establecen intentos múltiples para obtener datos del sensor antes de marcar un fallo.

**Aseguramiento de Transmisión de Mensajes**

* **QoS (Quality of Service)**: En MQTT, se utiliza un nivel de QoS que garantiza la entrega de los mensajes.
* **Reenvío Automático**: Si el ESP32 detecta que no se puede conectar al broker, almacena los mensajes en un buffer para reenviarlos más tarde.

**Reconexión Automática**

* **Gestión de Reconexión WiFi**: Si la conexión WiFi se pierde, el ESP32 intenta reconectarse de forma automática.
* **Reconexión al Broker MQTT**: El sistema incluye una función para detectar la pérdida de conexión con el broker y reestablecerla automáticamente.

**Configuración Inicial**

**ESP32**

1. Descargue e instale el IDE de Arduino.
2. Añada las librerías mencionadas anteriormente desde el gestor de librerías.
3. Configure el archivo de código principal con:
   * Credenciales de red WiFi (SSID y contraseña).
   * Dirección del broker MQTT.

**Node-RED**

1. Instale Node-RED en su computadora o servidor.
2. Añada las dependencias necesarias mediante el gestor de paquetes de Node-RED.
3. Configure los nodos MQTT para conectarse al mismo broker que el ESP32.

**Interacción con el Sistema**

**Visualización de Datos**

El panel de Node-RED permite visualizar gráficos en tiempo real con los datos recopilados por el ESP32.

**Control de Dispositivos**

Desde el mismo panel, puede enviar comandos para controlar dispositivos conectados al ESP32, como relés o LEDs.

**Resolución de Problemas**

**Problemas Comunes**

1. **El ESP32 no conecta al WiFi**:
   * Verifique las credenciales de red.
   * Asegúrese de que el router esté funcionando correctamente.
2. **Node-RED no recibe datos**:
   * Confirme que el broker MQTT esté activo.
   * Revise la configuración de los nodos MQTT en Node-RED.
3. **Datos inconsistentes de los sensores**:
   * Compruebe las conexiones físicas de los sensores.
   * Verifique el rango de operación de los sensores.

**2. Dependencias**

El código utiliza las siguientes bibliotecas:

* **DHTesp:** Para interactuar con el sensor DHT11.
* **WiFi y PubSubClient:** Para la gestión de conexión WiFi y MQTT.
* **ArduinoJson:** Para manejar mensajes JSON.
* **HTTPUpdate y HTTPClient:** Para realizar las actualizaciones OTA.
* **Button2:** Para gestionar eventos de un botón.

**3. Configuración Inicial**

**3.1 Definición de Constantes**

* **OTA\_URL:** URL del servidor donde se encuentra el firmware para OTA.
* **ssid y password:** Credenciales de la red WiFi.
* **server, user, pass:** Configuración del broker MQTT.
* Pines del hardware:
  + **PIN\_DHT:** Pin para el sensor DHT.
  + **BUTTON\_PIN:** Pin para el botón de control.

**3.2 Variables Globales**

Se utilizan para almacenar:

* Datos del sensor (temperatura y humedad).
* Estado del LED RGB (brillo, color y encendido/apagado).
* Periodos de envío de datos y verificación de actualizaciones.
* Temas (topics) MQTT para publicación y suscripción.

**4. Funciones Principales**

**4.1 Conexión WiFi**

**-- *void connect\_wifi()***

Esta función conecta el dispositivo a la red WiFi definida. Imprime en el monitor serie la dirección IP asignada.

**4.2 Conexión MQTT**

***-- void connect\_mqtt()***

Intenta conectar al broker MQTT. Si la conexión es exitosa:

* Se suscribe a varios temas MQTT.
* Publica un mensaje indicando que el dispositivo está en línea.

**4.3 Manejo de Mensajes MQTT**

Los mensajes recibidos son procesados según el tema correspondiente:

* **process\_mqtt\_config:** Actualiza configuraciones del dispositivo.
* **process\_mqtt\_brightness:** Ajusta el brillo del LED RGB.
* **process\_mqtt\_color:** Cambia el color del LED RGB.
* **process\_mqtt\_switch:** Enciende o apaga el LED RGB.
* **process\_mqtt\_fota:** Inicia el proceso de actualización OTA.

**4.4 Actualización OTA**

***-- void intenta\_OTA()***

Verifica si hay una nueva versión del firmware en el servidor OTA. Si hay actualización disponible, la descarga e instala. En caso de error, imprime un mensaje detallado.

**4.5 Gestor del LED RGB**

**-- void updateLED(unsigned long ahoraArg)**

Actualiza el estado del LED RGB basado en:

* Brillo y color objetivo.
* Periodo de cambio de brillo.

**4.6 Manejo del Botón**

Se gestionan los siguientes eventos del botón:

* **Simple Click**: Enciende o apaga el LED.
* **Double Click**: Ajusta el brillo al 100%.
* **Long Click**: Inicia la actualización OTA.

**4.7 Medición y Envio de Datos**

***-- void measureSendData(unsigned long ahoraArg)***

Lee la temperatura y humedad del sensor DHT y envía los datos al broker MQTT en intervalos definidos.

**5. Flujo Principal (setup y loop)**

**5.1 Setup**

Inicializa:

* La conexión WiFi y MQTT.
* El sensor DHT y el botón.
* Los temas MQTT (topics) asociados al dispositivo.

**5.2 Loop**

Ejecuta continuamente:

* **Gestor de LED RGB**: Actualiza el brillo y color del LED.
* **Medición y envío de datos**: Publica los datos del sensor.
* **Verificación de OTA**: Comprueba si hay una actualización disponible.
* **Manejo de mensajes MQTT**: Procesa mensajes recibidos.

**6. Temas MQTT**

* **II16/<CHIPID>/conexion:** Indica si el dispositivo está conectado.
* **II16/<CHIPID>/datos:** Publica los datos de temperatura, humedad y WiFi.
* **II16/<CHIPID>/led/brillo:** Ajusta el brillo del LED.
* **II16/<CHIPID>/led/color:** Cambia el color del LED.
* **II16/<CHIPID>/switch:** Enciende o apaga el LED.
* **II16/<CHIPID>/fota:** Inicia una actualización OTA.

**7. Personalización**

El usuario puede personalizar:

* **Credenciales WiFi** (ssid, password).
* **URL del servidor OTA** (OTA\_URL).
* **Pines** para el sensor DHT y el botón.
* **Periodos de operación**: Envió de datos, actualizaciones OTA, cambios de brillo.