Bibliotecas Incluidas

- `<string>`:
    - Proporciona funcionalidades para el manejo de cadenas de texto en C++. Es útil para operaciones como concatenación, comparación, y otras manipulaciones de strings que puedas necesitar.
- `<ESP8266WiFi.h>`:
    - Esencial para conectar el ESP8266 a una red WiFi. Permite realizar tareas como conectarse a una red, manejar conexiones, obtener información de la red, etc. Es la base para cualquier comunicación de red que hagas con el ESP8266.
- `<PubSubClient.h>`:
    - Esta biblioteca permite al ESP8266 comunicarse usando el protocolo MQTT, que es muy popular en IoT para el envío de mensajes entre dispositivos. Facilita publicar y suscribirse a topics MQTT, lo que es crucial para la interacción entre tu dispositivo y un broker MQTT.
- `<ArduinoJson.h>`:
    - Proporciona un manejo eficiente de objetos JSON, lo cual es muy útil para el intercambio de datos en proyectos IoT. Puedes usarla para parsear datos recibidos en formato JSON o para crear JSONs para enviar datos.
- `<ESP8266httpUpdate.h>`:
    - Facilita la actualización del firmware a través de HTTP (OTA - Over The Air). Esto permite que tu dispositivo ESP8266 se actualice remotamente sin necesidad de una conexión física al ordenador.
- `"Button2.h"`:
    - Una biblioteca para manejar interacciones con botones. Permite detectar clics simples, dobles, largos, etc., y es muy útil para interfaces de usuario sencillas.

Definiciones y Constantes

- **`__HTTPS__`**:
    - Se usa para habilitar o deshabilitar características específicas de HTTPS.
- **`BUTTON_PIN`**:
    - Define el pin del GPIO al que está conectado el botón, en este caso, el GPIO 0.
- **`OTA_URL`**:
    - Define la URL del servidor de actualizaciones OTA. Es la dirección a la que el ESP8266 se conectará para buscar y descargar actualizaciones de firmware. Esta url es https://iot.ac.uma.es:1880/esp8266-ota/update. Si queremos actualizar el

dispositivo mediante OTA podemos acerlo publicando el archivo binario compilado en la direccion: https://iot.ac.uma.es:1880/esp8266-ota

- **`FW_BOARD_NAME`**:
    - Para versiones del IDE < 2.0 poner en FW_BOARD_NAME el identificador de la placa, por ejemplo ".nodemcu", ver nombre del binario generado para conocer identificador utilizado. Para versiones del IDE >= 2.0 dejar vacío.
- **`HTTP_OTA_VERSION`**:
    - Se usa para identificar la versión actual del firmware, para el proceso de actualización OTA.
- **`DEBUG_STRING`**:
    - Una macro útil para el debug. Agrega el nombre de la función y el número de línea al mensaje de depuración, lo que facilita rastrear dónde ocurren problemas o eventos en el código.

Instancias y Objetos

`WiFiClient`

- **Biblioteca:** ESP8266WiFi
- **Propósito:** Crear un cliente que puede conectarse a un servidor a través de WiFi.
- **Descripción:** `WiFiClient` es una clase que se utiliza para manejar la conexión de red. El objeto `wClient` actúa como un cliente de red en este contexto.

`PubSubClient`

- **Biblioteca:** PubSubClient
- **Propósito:** Utilizado para la comunicación MQTT.
- **Descripción:** `PubSubClient`, inicializado con `wClient`, conecta el ESP8266 a un broker MQTT. Permite publicar mensajes a topics MQTT y suscribirse para recibir mensajes.

`Button2`

- **Biblioteca:** Button2
- **Propósito:** Manejar interacciones con botones físicos.
- **Descripción:** `Button2` gestiona un botón físico conectado al ESP8266. Permite detectar y manejar distintos tipos de pulsaciones como clics simples, dobles, y largos.

`ADC_MODE(ADC_VCC)`

Configura el ADC para que en lugar de leer de un pin analógico externo, lea el voltaje de alimentación del ESP8266.

Función `conecta_wifi`

Descripción

Esta función establece la conexión WiFi del dispositivo ESP8266 a una red especificada. Se utiliza para conectar el dispositivo a internet o a una red local.

Función `conecta_wifi`

Descripción

Esta función establece la conexión WiFi del dispositivo ESP8266 a una red especificada. Se utiliza para conectar el dispositivo a internet o a una red local.

Detalles del Proceso

1. **Inicio de Conexión:**
   Primero, se imprime un mensaje en el puerto serie (`Serial.println`) para indicar que se está iniciando el proceso de conexión a la red WiFi.
2. **Configuración del Modo WiFi:**
   Se configura el módulo WiFi en modo estación (`WIFI_STA`). Este modo permite que el ESP8266 funcione como un cliente WiFi, conectándose a una red WiFi existente.
3. **Establecimiento de la Conexión:**
   Se inicia la conexión WiFi utilizando las credenciales de la red (`ssid` y `password`) mediante `WiFi.begin`.
4. **Bucle de Espera:**
   Se entra en un bucle que se ejecuta mientras el estado de la conexión WiFi no sea `WL_CONNECTED`. Durante este bucle, se realiza una pausa de 200 milisegundos entre cada verificación del estado de la conexión, para evitar la saturación del procesador y proporcionar tiempo para que la conexión se establezca. Mientras espera, imprime puntos en el puerto serie como indicador de progreso.
5. **Conexión Establecida:**
   Una vez que el dispositivo se ha conectado con éxito a la red WiFi, sale del bucle y finaliza la función, indicado por un salto de línea en el puerto serie (`Serial.println`).

Función `conecta_mqtt`

Descripción

Establece y mantiene la conexión del dispositivo ESP8266 con un broker MQTT. Esta función intenta conectarse al broker MQTT y se suscribe a varios topics para la comunicación.

Proceso de la Función

1. **Bucle de Conexión:**

- La función entra en un bucle que se repite hasta que se establece una conexión con el broker MQTT.

2. **Intento de Conexión:**
   - En cada iteración del bucle, intenta establecer una conexión con el broker MQTT, utilizando el ID de la placa (`ID_PLACA`) como identificador del cliente MQTT.
3. **Suscripciones MQTT:**
   - Una vez establecida la conexión, el cliente MQTT se suscribe a una serie de topics, que incluyen topics para la publicación y recepción de datos, comandos, y actualizaciones de estado.
4. **Publicación de Estado de Conexión:**
   - Después de conectarse y suscribirse a los topics necesarios, publica un mensaje en un topic específico para confirmar la conexión exitosa.
5. **Manejo de Errores:**
   - Si la conexión falla, se imprime un mensaje de error en el puerto serie y se espera 5 segundos antes de reintentar, para evitar intentos continuos de reconexión.

Función `procesa_mensaje`

Descripción

Maneja los mensajes recibidos a través de MQTT, ejecutando acciones basadas en el topic y el contenido del mensaje.

Funcionamiento

- **Parámetros:**
  - `char *topic`: Topic MQTT donde se publicó el mensaje.
  - `byte *payload`: Cuerpo del mensaje.
  - `unsigned int length`: Longitud del mensaje.
- **Proceso:**
  1. **Conversión del Mensaje:**
     - Convierte el `payload` en una cadena `String` para facilitar su procesamiento.
  2. **Impresión del Mensaje:**
     - Imprime el mensaje recibido y el topic correspondiente en el puerto serie.
  3. **Manejo de Diferentes Topics:**
     - Utiliza condicionales para procesar el mensaje según el topic.
     - **Control del Motor (`topic_SUB_motor_cmd`):**
       - Deserializa el JSON del mensaje.
       - Establece la intensidad del motor basándose en el valor recibido.

        - Si hay un error en la deserialización, imprime un mensaje de error.
    - **Control del Switch (`topic_SUB_switch_cmd`):**
        - Deserializa el JSON y obtiene el estado deseado del switch.
        - Cambia el estado de el GPIO al que esta conectado el LED.
        - Publica el estado actual del switch en el topic correspondiente.
        - En caso de error en la deserialización, informa mediante un mensaje de error.
    - **Control de la Alarma (`topic_SUB_alarm_cmd`):**
        - Deserializa el mensaje para determinar el estado de la alarma.
        - Activa o desactiva la alarma basándose en el mensaje.
        - Publica el estado de la alarma en el topic correspondiente.
        - Informa sobre errores en la deserialización si los hay.
    - **Actualización FOTA (`topic_SUB_FOTA`):**
        - Deserializa el mensaje para obtener la orden de actualización FOTA.
        - Actualiza el valor de la variable booleana para iniciar la actualización FOTA si es necesario.
        - Informa sobre errores en la deserialización.

Función `intenta_OTA`

Descripción

Esta función maneja el proceso de actualización Over-The-Air (OTA) del firmware en el dispositivo ESP8266.

Detalles del Proceso

1. **Inicio de la Actualización:**
    - La función inicia informando a través del puerto serie que se está comprobando si hay actualizaciones disponibles, mostrando la URL de donde se descargará la actualización.
2. **Configuración de Callbacks:**
    - Configura funciones específicas (`callbacks`) para diferentes eventos durante el proceso OTA, como el inicio, error, progreso y finalización de la actualización.
3. **Configuración del Cliente WiFi:**
    - Dependiendo de si se está utilizando HTTPS o no, se configura un `WiFiClientSecure` o un `WiFiClient` respectivamente.
4. **Proceso de Actualización:**
    - Utiliza la biblioteca `ESPhttpUpdate` para intentar descargar y actualizar el firmware.
    - Toma como parámetros el cliente WiFi, la URL de actualización y la versión actual del firmware.

5. **Manejo de Resultados de la Actualización:**
    - Gestiona los diferentes resultados del intento de actualización:
        - `HTTP_UPDATE_FAILED`: Informa si la actualización ha fallado, mostrando el error correspondiente.
        - `HTTP_UPDATE_NO_UPDATES`: Indica que el dispositivo ya tiene la última versión del firmware.
        - `HTTP_UPDATE_OK`: Confirma que la actualización se ha completado con éxito.

Función `singleClick`

Descripción

Gestiona la acción de un clic simple en el botón de flash.

Funcionamiento

- Al detectar un clic simple, la función imprime un mensaje de confirmación.
- Cambia el estado de `intensidadDeseada`:
    - Si `intensidadDeseada` es 0 (indicando que está apagado), lo establece al valor de `intensidadPrevia` (encendiendo el dispositivo).
    - Si `intensidadDeseada` no es 0 (indicando que está encendido), guarda el valor actual en `intensidadPrevia` y lo establece a 0 (apagando el dispositivo).

Función `longClick`

Descripción

Maneja la acción de un clic largo en el botón de flash.

Funcionamiento

- Al detectar un clic largo, imprime un mensaje "long click" en el puerto serie para indicar la acción.
- Llama a la función `intenta_OTA()`, iniciando así el proceso de actualización del firmware Over-The-Air (OTA).

Función `doubleClick`

Descripción

Responde a un doble clic en un botón.

Funcionamiento

- Al detectar un doble clic, la función imprime "double click" en el puerto serie.

- Establece `intensidadDeseada` en 100, lo que representa la activación del motor al 100% de intensidad.

Función `loop`

Descripción

Esta función es el núcleo del programa, ejecutándose repetidamente para manejar conexiones MQTT, interacciones de botones, y actualizar datos y estados.

Funcionamiento

1. **Gestión de la Conexión MQTT:**
    - Verifica si el cliente MQTT está conectado.
    - Si no está conectado, intenta reconectar utilizando la función `conecta_mqtt`.
2. **Manejo de Eventos de Botones:**
    - Llama a `button.loop()` para que la librería Button2 procese cualquier evento de botón ocurrido.
3. **Envío de Datos:**
    - Envía datos periódicamente (cada 2000 milisegundos).
    - Enciende un LED para indicar que está enviando datos.
    - Recopila y envía información como tiempo de funcionamiento, voltaje, estado de sensores, y datos de conexión WiFi.
    - Apaga el LED tras enviar los datos.
4. **Control de Intensidad del Motor:**
    - Ajusta gradualmente la intensidad del motor a una velocidad de 1% cada 10 milisegundos.
    - Publica el nuevo estado del motor en un topic MQTT cuando se alcanza la intensidad deseada.