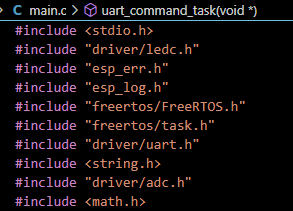
**DECLARACION DE LIBRERIAS:**



stdio.h: Controla la entrada/salida estándar.

driver/ledc.h: Controla LED y PWM.

esp\_err.h: Maneja códigos de error.

esp\_log.h: Registra mensajes de registro.

freertos/FreeRTOS.h y freertos/task.h: Administra tareas en tiempo real.

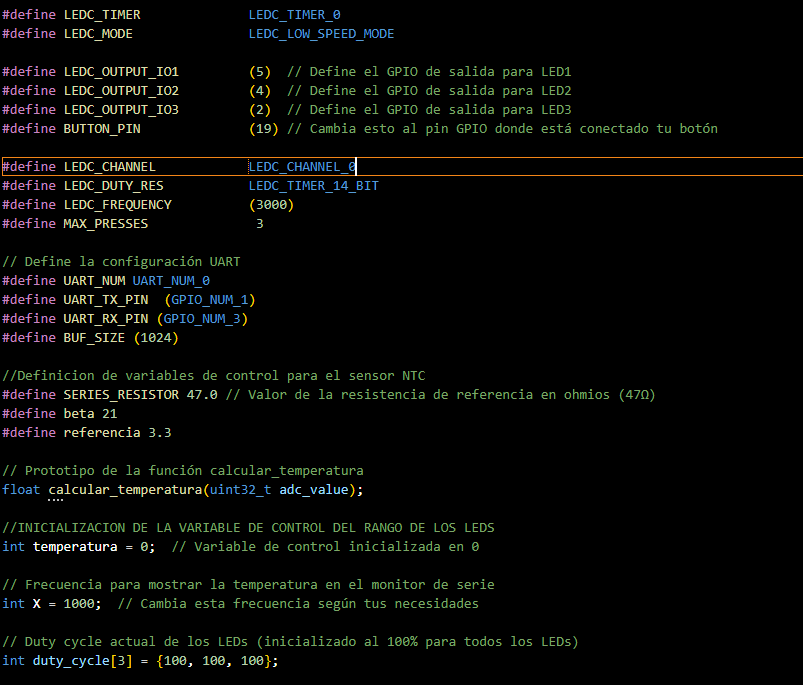
driver/uart.h: Controla la comunicación UART.

<string.h>: Manipula cadenas de caracteres.

driver/adc.h: Controla el convertidor analógico a digital (ADC).

**<math.h>**: Realiza operaciones matemáticas.

**Definición de variables y configuración:**

****

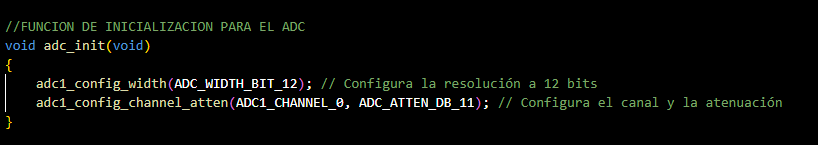
* Se definen configuraciones para el módulo LEDC:

1. LEDC\_TIMER y LEDC\_MODE especifican el temporizador y el modo de velocidad baja para controlar los LEDs.
2. Se definen tres pines de salida (LEDC\_OUTPUT\_IO1, LEDC\_OUTPUT\_IO2, LEDC\_OUTPUT\_IO3) para conectar tres LEDs.
3. LEDC\_CHANNEL se establece en LEDC\_CHANNEL\_0.
4. LEDC\_DUTY\_RES se establece en LEDC\_TIMER\_14\_BIT.
5. LEDC\_FREQUENCY establece la frecuencia de PWM en 3000 Hz.

* Se define un pin para un botón (BUTTON\_PIN) que se utiliza para controlar el programa.
* Se configura UART para la comunicación serie. Se utiliza el UART 0 (UART\_NUM\_0) con pines de transmisión (UART\_TX\_PIN) y recepción (UART\_RX\_PIN) específicos.
* Se definen constantes y variables para la medición de temperatura utilizando un sensor NTC. Esto incluye la resistencia de referencia, el valor beta y la referencia de voltaje.
* Se declara el prototipo de una función llamada calcular\_temperatura, que se utilizará para calcular la temperatura a partir de valores de ADC (convertidor analógico a digital).
* Se inicializa una variable de control temperatura en 0, que se utilizará para controlar el rango de los LEDs.
* Se establece la frecuencia (X) para mostrar la temperatura en el monitor de serie.
* Se inicializa un arreglo duty\_cycle con tres valores de 100, que representan la intensidad de luz (duty cycle) inicial de los LEDs, todos al 100%.

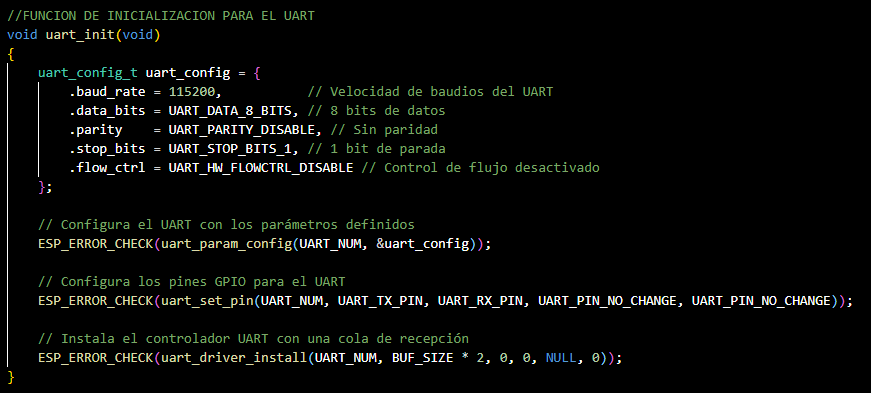
**FUNCIONES E INTERRUPCIONES:**

**Función ADC:**

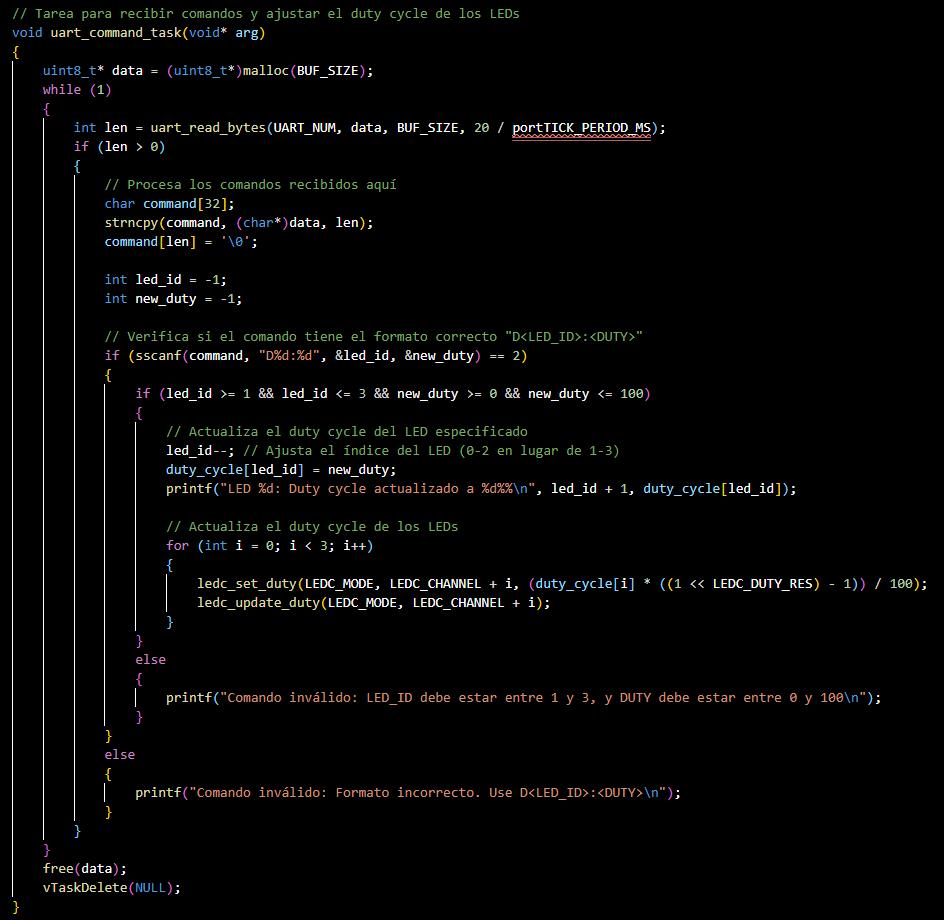
****

La función `adc\_init` inicializa el convertidor analógico a digital (ADC) en un microcontrolador. Configura la resolución del ADC en 12 bits para una alta precisión y el canal 0 del ADC con una atenuación de -11 dB, adaptando la señal de entrada a un rango de 0-3.9 V. Esto prepara el ADC para realizar mediciones precisas en el canal 0 con la configuración adecuada.

**Función UART:**

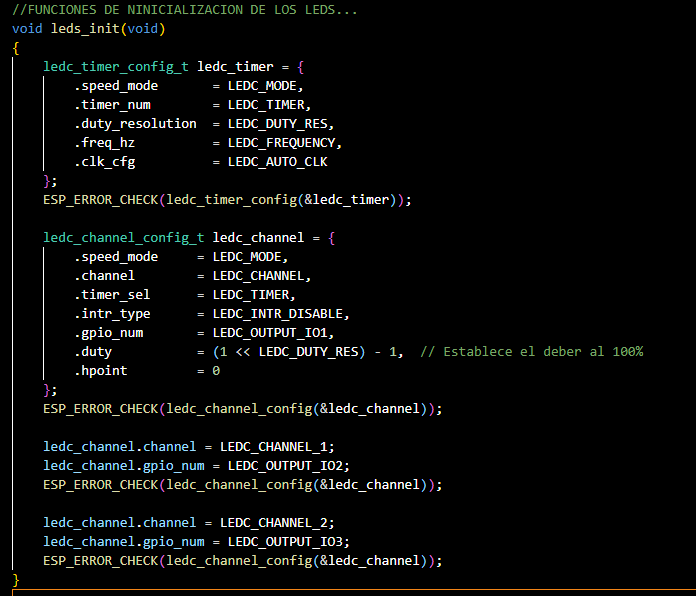
****

La función `uart\_init` se encarga de la inicialización del módulo UART en el microcontrolador. Establece la configuración del UART, incluyendo la velocidad de baudios, el número de bits de datos, la paridad, los bits de parada y el control de flujo. Luego, configura los pines GPIO para el UART y crea una cola de recepción para manejar los datos entrantes. Una vez que esta función se ejecuta, el UART estará listo para la comunicación serie, con la configuración especificada, y será capaz de recibir y transmitir datos de manera efectiva.



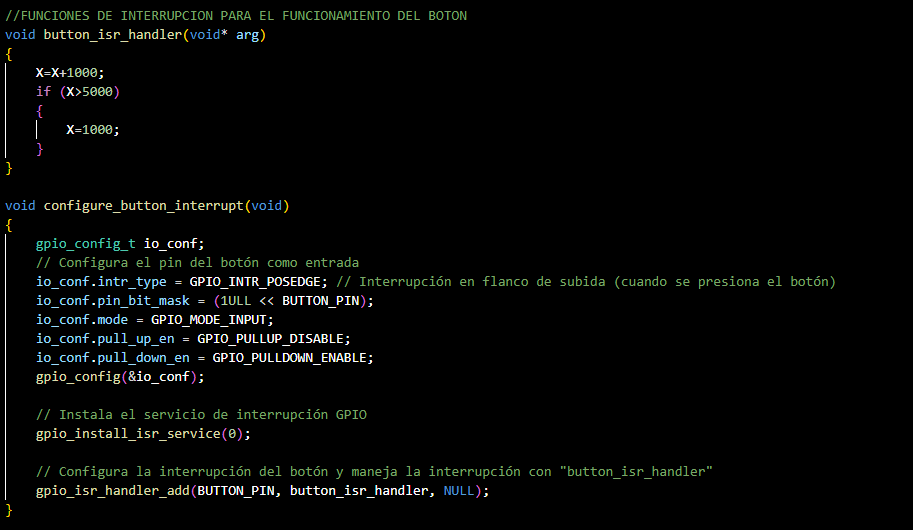
La tarea `uart\_command\_task` opera de manera continua, escuchando comandos enviados a través de la comunicación UART del microcontrolador. Cuando se recibe un comando válido en el formato "D<LED\_ID>:<DUTY>", donde `LED\_ID` debe estar entre 1 y 3 y `DUTY` entre 0 y 100, la tarea ajusta el ciclo de trabajo de los LEDs según el `LED\_ID` especificado, un ejemplo de comando seria D1:50, lo que cambia la intensidad del led 1 al 50%. Luego, actualiza el ciclo de trabajo de los LEDs y proporciona un mensaje de confirmación. Si el comando es inválido debido a un formato incorrecto o valores fuera de rango, se emite un mensaje de error correspondiente. Esto permite controlar los LEDs de manera remota a través de UART y recibir retroalimentación sobre los ajustes realizados.

**Función LEDS:**

****

La función leds\_init desencadena la inicialización de los LEDs en el microcontrolador, aprovechando el módulo LEDC (Controlador de Luz de Bajo Consumo). Primero, configura un temporizador LEDC, estableciendo la velocidad de operación, el número de temporizador, la resolución del ciclo de trabajo, la frecuencia de PWM y la configuración del reloj automático. Esta configuración permite una gestión precisa de la iluminación de los LEDs. A continuación, se configuran canales LEDC para cada uno de los tres LEDs. Cada canal se asocia a la velocidad de operación LEDC, se especifica el número de canal, se selecciona el temporizador correspondiente, se desactiva cualquier tipo de interrupción, se designa el pin GPIO al que está conectado el LED y se establece un ciclo de trabajo inicial al 100%, lo que significa que los LEDs se encienden a plena potencia al inicio.

**Función Botón:**

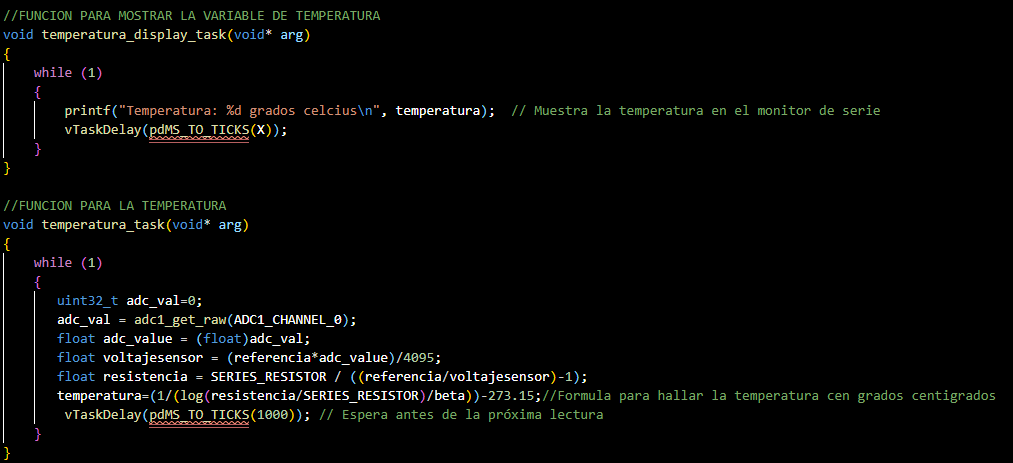
****

Las funciones `button\_isr\_handler` y `configure\_button\_interrupt` se utilizan para gestionar las interrupciones generadas por un botón en un microcontrolador. Aquí está una breve descripción de cada una:

* `button\_isr\_handler(void\* arg)`: Esta función se encarga de manejar la interrupción generada por el botón. Cuando se ejecuta, aumenta el valor de `X` en 1s y luego verifica si `X` supera 5s. Si es así, reinicia `X` a 1s. En resumen, esta función incrementa `X` en pasos de 1s cada vez que se presiona el botón, y si alcanza 5s, lo reinicia.
* `configure\_button\_interrupt()`: Esta función se encarga de configurar la interrupción generada por el botón. Primero, establece la configuración del pin del botón como entrada y especifica que la interrupción se debe generar en el flanco de subida (cuando se presiona el botón). Luego, instala el servicio de interrupción GPIO y configura la interrupción del botón para que sea manejada por la función `button\_isr\_handler`. En resumen, esta función configura el entorno para detectar cuando se presiona el botón y ejecutar la función `button\_isr\_handler` en respuesta a esa acción.

En conjunto, estas funciones permiten detectar y gestionar las pulsaciones del botón, lo que, en este caso, modifica el valor de `X` en el microcontrolador cuando se presiona el botón, lo que ajusta la frecuencia con la que se actualiza la temperatura.

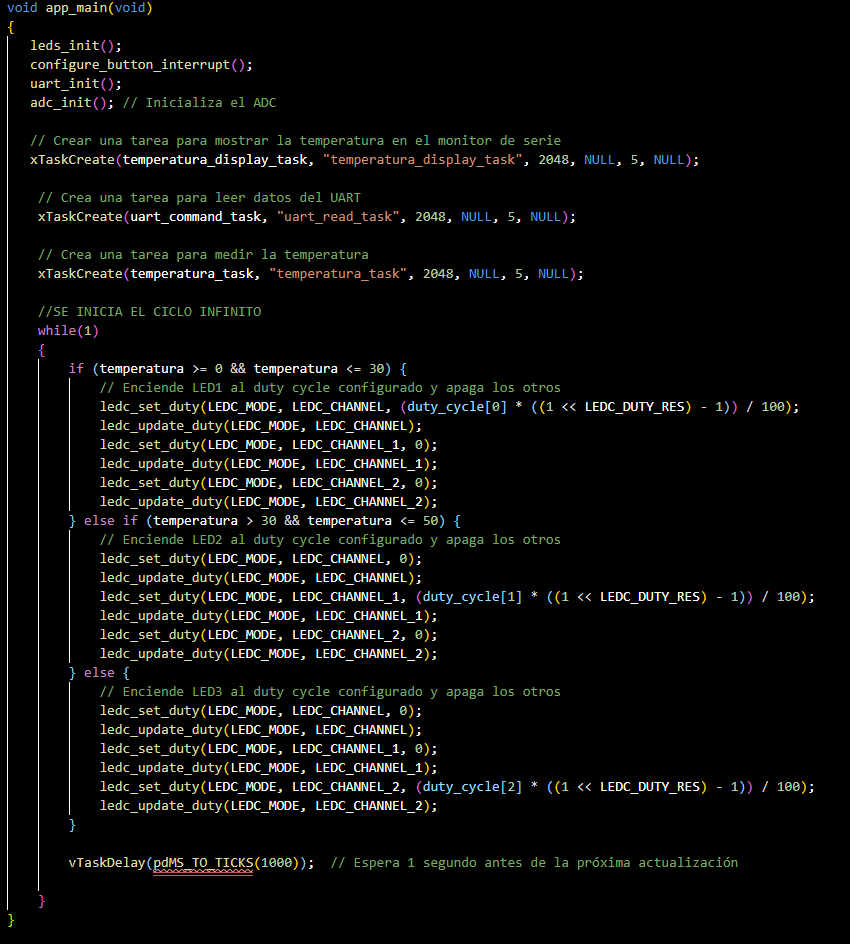
**Función NTC:**

****

Estas dos funciones trabajan en conjunto para medir y mostrar la temperatura utilizando un sensor NTC (Termistor Negativo de Coeficiente de Temperatura) en el microcontrolador. Aquí está una descripción concisa de cada función:

* temperatura\_display\_task(void\* arg): Esta tarea se encarga de mostrar la temperatura en grados Celsius en el monitor de serie de manera continua. Utiliza la variable global temperatura y la imprime en el formato "Temperatura: X grados Celsius", donde "X" es el valor de temperatura actual. La tarea se ejecuta cada X milisegundos, lo que permite actualizar y mostrar la temperatura periódicamente.
* temperatura\_task(void\* arg): Esta tarea mide la temperatura utilizando un sensor NTC. Primero, lee un valor crudo del convertidor analógico a digital (ADC) en el canal 0. Luego, calcula la tensión del sensor y, a partir de esa tensión y la resistencia de referencia, calcula la resistencia del sensor. Finalmente, utiliza una fórmula para convertir la resistencia en grados Celsius y almacena el resultado en la variable global temperatura. La tarea se ejecuta cada segundo (1000 milisegundos), lo que proporciona lecturas de temperatura regulares.

**CICLO PRINCIPAL:**

****

El app\_main es la función principal de la aplicación. Aquí está un resumen de lo que hace:

* Se inicializan los LEDs llamando a la función leds\_init(), se configura la interrupción del botón mediante configure\_button\_interrupt(), se inicia el UART con uart\_init(), y se inicializa el ADC utilizando adc\_init().
* Se crean tres tareas de FreeRTOS:

1. temperatura\_display\_task muestra la temperatura en el monitor de serie.
2. uart\_command\_task espera y procesa comandos a través del UART para ajustar el ciclo de trabajo de los LEDs.
3. temperatura\_task mide la temperatura utilizando un sensor NTC y actualiza la variable global temperatura con lecturas regulares.

* En un bucle infinito, se verifica el valor de temperatura y se ajusta el ciclo de trabajo de los LEDs en función de la temperatura actual. Si la temperatura está entre 0 y 30 grados Celsius, se enciende el LED1 y se apagan los otros. Si está entre 30 y 50 grados Celsius, se enciende el LED2 y se apagan los demás. Si es superior a 50 grados Celsius, se enciende el LED3 y se apagan los otros dos.
* Después de cada ajuste del ciclo de trabajo, el programa espera 1 segundo antes de volver a verificar la temperatura y realizar nuevos ajustes.