
# Laboratorio 1

## Diseño IOT y Sistemas Embebidos

DIEGO BERGARA
CARLOS GONZALEZ
FELIPE CABRERA

DOCENTES:

PABLO ALONSO
AGUSTÍN DERRIGIBUS

# Primera Parte

## 1-C. Estructura

- Carpeta "main": Contiene el archivo principal del proyecto (main.c o main.cpp) que es el punto de entrada del programa. También puede contener otros archivos fuente y de encabezado utilizados por el programa principal.

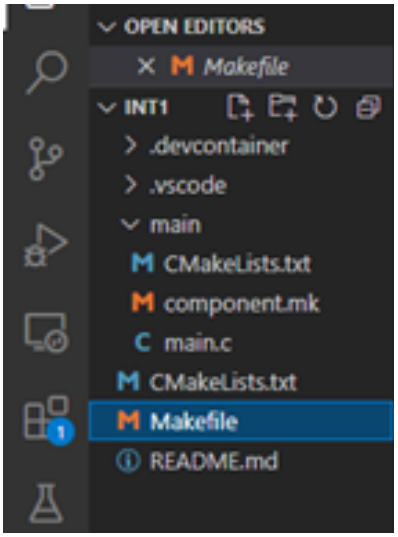
- Carpeta "components": Contiene componentes reutilizables que se pueden usar en el proyecto, como controladores de pantalla, controladores de sensores, controladores de red, etc. Estos componentes están organizados en subcarpetas, y cada subcarpeta representa un componente diferente.

- Carpeta "build": Esta carpeta contiene los archivos de construcción generados por la herramienta de construcción CMake. Estos archivos se generan cuando se compila el proyecto y se utilizan para crear el binario final que se carga en la placa de desarrollo.

- Archivo "CMakeLists.txt": Este archivo es el archivo de configuración de CMake utilizado para compilar el proyecto. Contiene información sobre los archivos fuente, los componentes, las bibliotecas, las opciones de compilación, etc.

- Archivo "sdkconfig": Este archivo contiene la configuración de la placa y las opciones de compilación utilizadas para compilar el proyecto. Puede ser modificado manualmente para personalizar la configuración del proyecto.

- Carpeta .vscode: En esta carpeta se encuentran los JSON correspondientes a la configuración para el desarrollo en Visual Studio Code



## 2-D Opciones se pueden modificar en el sdkconfig

- Configuración de la placa: Es posible configurar parámetros específicos de la placa, como el tipo de microcontrolador, el tamaño de la memoria, el pinout, la velocidad de reloj, entre otros.

- Opciones de compilación: Es posible especificar opciones de compilación específicas, como la optimización del código, el tipo de herramienta de compilación, el tipo de depuración, entre otros.

- Configuración de la biblioteca: Es posible configurar opciones específicas de la biblioteca utilizada en el proyecto, como el tipo de biblioteca (estática o dinámica), el tipo de biblioteca de red, entre otros.

- Características del sistema operativo: Es posible configurar opciones específicas del sistema operativo utilizado en el proyecto, como el tamaño de la pila, el tamaño del búfer de red, el tipo de planificador, entre otros.

- Configuración de la conectividad: Es posible configurar opciones específicas de la conectividad utilizada en el proyecto, como el tipo de protocolo de red, el tipo de seguridad utilizado, entre otros.

Al cambiar el build en la interfaz grafica del sdkconfig se puede ver el siguiente cambio entre el old y el nuevo:

```
18  #
19  # Build type
20  #
21  CONFIG_APP_BUILD_TYPE_APP_2NDBOOT=y
22  # CONFIG_APP_BUILD_TYPE_ELF_RAM is not set
23  CONFIG_APP_BUILD_GENERATE_BINARIES=y
24  CONFIG_APP_BUILD_BOOTLOADER=y
25  CONFIG_APP_BUILD_USE_FLASH_SECTIONS=y
26  # end of Build type
```

El nuevo:

```
18  #
19  # Build type
20  #
21  # CONFIG_APP_BUILD_TYPE_APP_2NDBOOT is not set
22  CONFIG_APP_BUILD_TYPE_ELF_RAM=y
23  # end of Build type
24
```

## 3-B Compilación

- El archivo "flasher_args.json" contiene los argumentos que se deben pasar a "esptool.py" para cargar el firmware en la placa. Algunos de los parámetros que se pueden encontrar en este archivo incluyen la velocidad de baudios, el puerto COM utilizado para la conexión serial, el tipo de dispositivo, entre otros.

- El archivo "project_description.json" contiene información detallada sobre el proyecto, como el nombre del proyecto, la descripción, la versión del SDK utilizado,

la versión del firmware, entre otros. Esta información se utiliza para fines de documentación y seguimiento de versiones del proyecto.

- En resumen, son archivos  generados por el SDK de Espressif para facilitar la carga del firmware generado en la placa de desarrollo. Por esto, no es recomendado editarlos manualmente, ya que son generados automáticamente en el proceso de construcción y carga del firmware.

### 3-C Buscar las variables antes declaradas en el build

Dentro del archivo .map en la carpeta build  podemos ver que:

- exampleArray ocupa 12 bytes de memoria y se encuentra en la dirección 0x000000003ffbf200.

- exampleData ocupa 4 bytes de memoria y se encuentra en la dirección 0x000000003ffbf20c

### 3-D Sustituir ARRAY_SIZE por 10

Tras sustituír ARRAY_SIZE observamos el siguiente cambio

- Cambia el espacio de memoria de exampleArray, de 12 a 10 bytes, pero la dirección de memoria se mantiene, 0x000000003ffbf200

# Segunda Parte

### 1-B Cómo controlar LED RGB

External GPIO45 es un dispositivo de entrada y salida de propósito general.
Se debe conectar el GPIO45 con el pin RGB_CTRL a través del jumper.
Por lo que, se enviaran 24 bits de datos que representarán un color (RGB) al led (WS2812).

El pseudocódigo que utilizaremos es el siguiente:

```
led_init()
{
    led_rgb_init() // Configures GPIO45
}
```

```
set_color(color){

   if (color == RED) {
        set_pixel(255,0,0)
        refresh()
    } else if (color == GREEN) {
        set_pixel(0,255,0)
        refresh()
    } else if(color == BLUE) {
        set_pixel(0,0,255)
        refresh()
    }
}


main()
{
    led_init()

    while (1)
    {
        set_color("RED")
        delay
        set_color("GREEN")
        delay
        set_color("BLUE")
        delay
    }

}
```