Universidad Tecnológica de Panamá

Facultad de Ingeniería Eléctrica

Laboratorio de Computadores Digitales

Laboratorio #2

Fernando Guiraud
8-945-692

Profesor Elias Mendoza

Grupo: 4EE141

Semestre II 2022

1. Introducción

En esta experiencia de laboratorio se utilizará el microcontrolador ESP32. El módulo ESP32 es una solución de Wi-Fi/Bluetooth todo en uno, integrada y certificada que proporciona no solo la radio inalámbrica, sino también un procesador integrado con interfaces para conectarse con varios periféricos.

El objetivo de esta experiencia de laboratorio es generar un algoritmo que sea capaz de controlar dos leds de manera independiente por medio de un botón. Esto nos permitirá aprender a conocer el funcionamiento de la declaración de entradas y ademas la configuración por hardware de un pull up resistor.

2. Objetivos

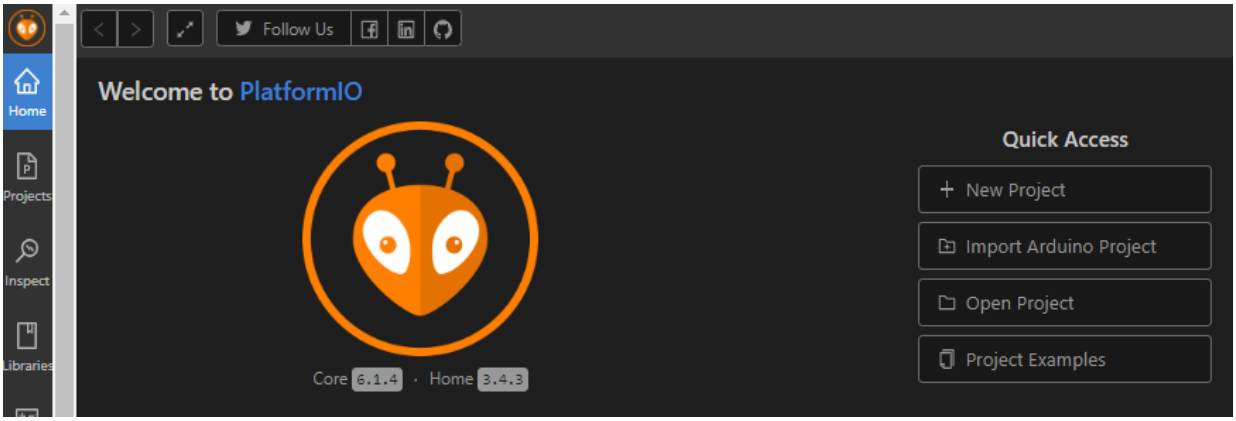
- Generar un algoritmo que sea capaz de controlar dos leds alternando su estado al presionar un botón.

3. Procedimiento

Para esta experiencia de laboratorio se utilizó el compilador de Visual Studio Code, con la extensión de PlatformIO. En esta plataforma se trabajó con el lenguaje C++.

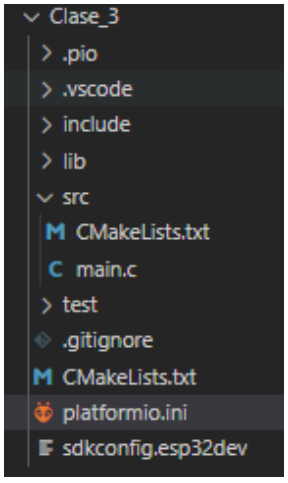
El primer paso consiste en instalar los drivers para que el computador reconozca la tarjeta a través de un puerto USB.

Una vez instalado el driver, el siguiente paso es crear un proyecto dentro de la extensión PlatformIO de Visual Studio Code.



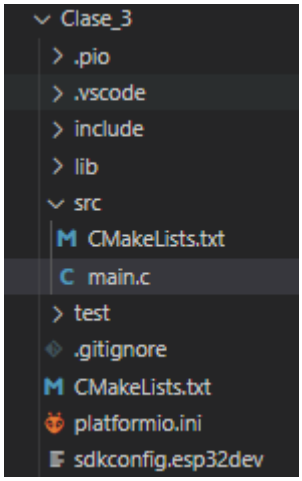
En este proyecto es necesario seleccionar el modelo de la tarjeta ESP32 y el framework que en este caso fue Espressif.

Al crear el proyecto se genera una sección llamada platformio.ini donde debemos agregar la siguiente línea para especificar la velocidad de transmisión de datos al monitor serial.



```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:esp32dev]
platform = espressif32
board = esp32dev
framework = espidf
monitor_speed = 115200
```

Posteriormente dentro de la sección scr se encuentra un archivo llamado main.c, en este archivo escribimos el cuerpo del código.



En las primeras líneas del código se declararon las librerías que contienen las funciones que fueron utilizadas posteriormente.

```
#include <stddef.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
```

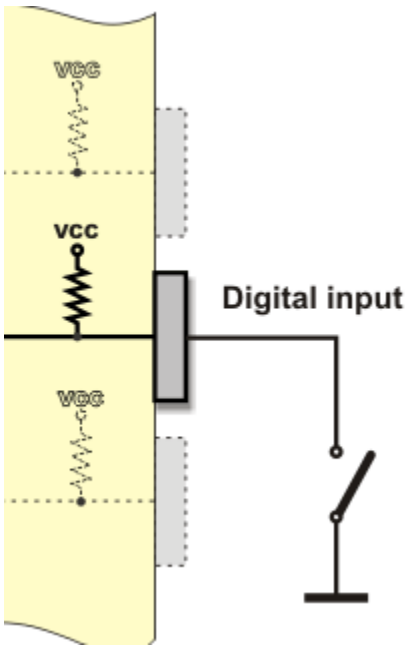
Se declararon las variables asociadas a los pines de las entradas y salidas a utilizar.

```
#define RLED 33
#define BT 25
#define GLED 26
#define GPIO_RLED_PIN_SEL (1ULL << RLED)
#define GPIO_BT_PIN_SEL (1ULL << BT)
#define GPIO_GLED_PIN_SEL (1ULL << GLED)
```

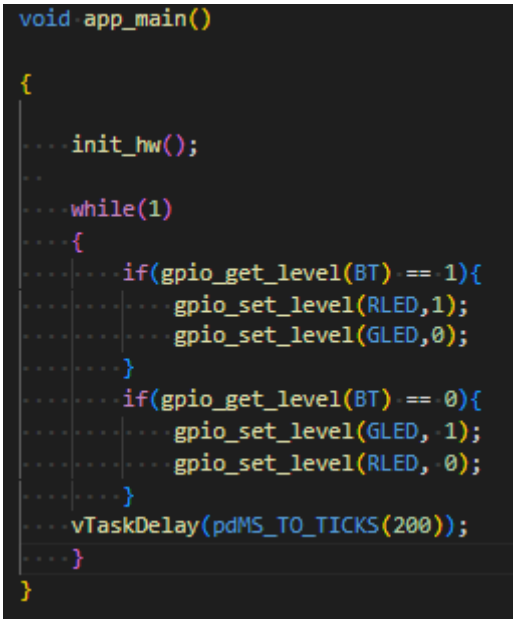
Después de esto se declararon los puertos de entrada y salida por medio de la librería GPIO. Es importante tomar en cuenta que se ha declarado una salida configurada con una resistencia de pull up, en la segunda declaración (io_config.pull_up_en = 1), asignándole un 1 como valor boleando de true.

```
static void init_hw(void){
    
    gpio_config_t io_config;

    io_config.mode = GPIO_MODE_OUTPUT;
    io_config.pin_bit_mask = GPIO_RLED_PIN_SEL|GPIO_GLED_PIN_SEL;
    io_config.pull_down_en = GPIO_PULLDOWN_DISABLE;
    io_config.pull_up_en = GPIO_PULLUP_DISABLE;
    io_config.intr_type = GPIO_INTR_DISABLE;

    gpio_config(&io_config);

    io_config.mode = GPIO_MODE_INPUT;
    io_config.pin_bit_mask = GPIO_BT_PIN_SEL;
    io_config.pull_down_en = GPIO_PULLDOWN_DISABLE;
    io_config.pull_up_en = 1;
    io_config.intr_type = GPIO_INTR_DISABLE;

    gpio_config(&io_config);

}
```

En la siguiente imagen podemos ver una representación de como sería un pull up resistor dentro del ESP32.



Por último, se estableció el cuerpo principal del algoritmo que consiste la inicialización de los puertos configurados por medio de la librería GPIO. Después, se estructuro un algoritmo repetitivo infinito donde se enciende el led rojo y se apaga el led verde siempre y cuando el pin BT tenga el estado true (1). En el caso contrario, al tener el estado false (0), se encenderá el led verde y se apagara el led rojo. Para evitar efectos de rebote de la señal en el botón, se incorpora un retardo de 200 milisegundos.

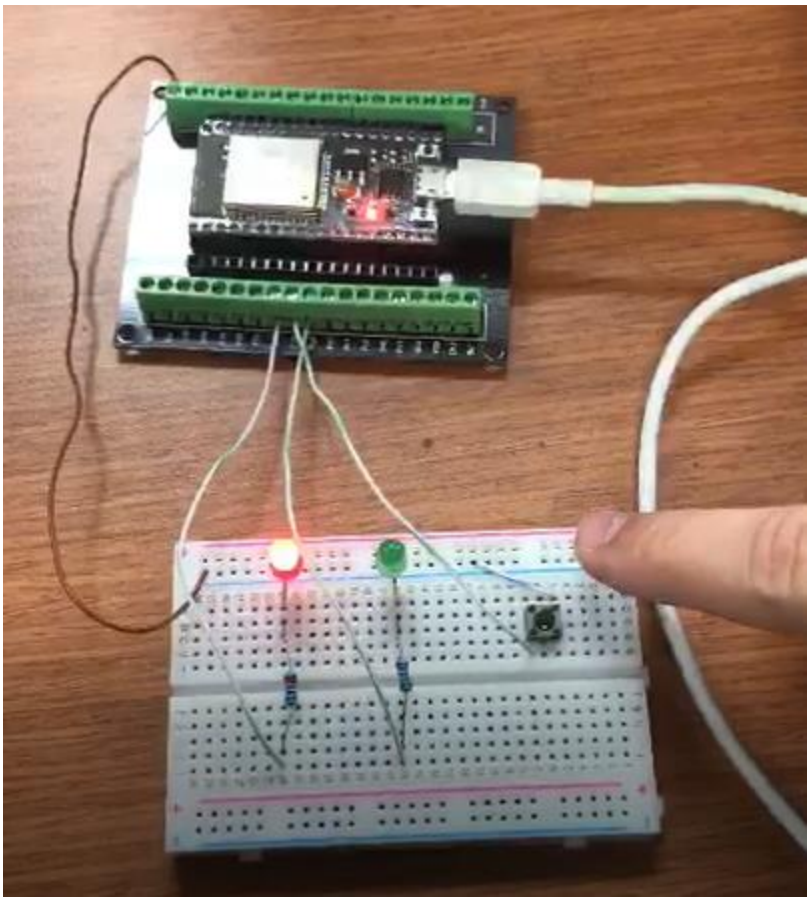


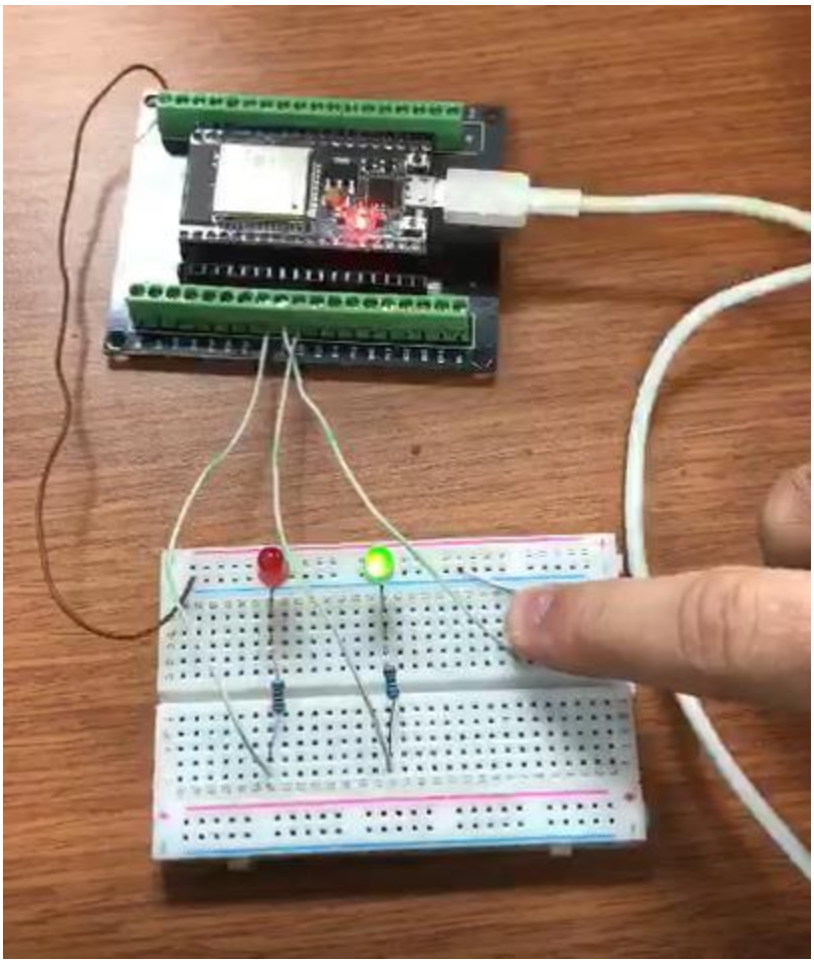
```
void app_main()

{

    init_hw();

    while(1)
    {
        if(gpio_get_level(BT) == 1){
            gpio_set_level(RLED,1);
            gpio_set_level(GLED,0);
        }
        if(gpio_get_level(BT) == 0){
            gpio_set_level(GLED, 1);
            gpio_set_level(RLED, 0);
        }
    vTaskDelay(pdMS_TO_TICKS(200));
    }
}
```

El codigo completo se ajunta en la sección de anexos.

4. Resultados

En las siguientes imágenes se mostrará la secuencia de encendido y apagado de los leds. Adicionalmente se adjuntará un video del funcionamiento en la entrega del informe.





5. Conclusiones

En esta experiencia de laboratorio se logró familiarizarse con la configuración de la declaración de variables, pudiendo entender la función de parámetros como el pull_up_en. También la estructura un poco más elaborada que en experiencias anteriores utilizando condicionales y retardos. Este procedimiento nos permitirá posteriormente crear algoritmos más complejos en futuras experiencias de laboratorio.