Universidad Tecnológica de Panamá

Facultad de Ingeniería Eléctrica

Laboratorio de Computadores Digitales

Laboratorio #1

Fernando Guiraud
8-945-692

Profesor Elias Mendoza

Grupo: 4EE141

Semestre II 2022

1. Introducción

En esta experiencia de laboratorio se utilizara 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 procesador en realidad tiene dos núcleos de procesamiento cuyas frecuencias operativas pueden controlarse independientemente entre 80 megahercios (MHz) y 240 MHz. Los periféricos del procesador facilitan la conexión a una variedad de interfaces externas como:

Interfaz periférica serial (SPI)
I2C
Transmisor receptor asíncrono universal (UART)
I2S
Ethernet
Tarjetas SD
Interfaces táctiles y capacitivas

Hay varios módulos ESP32 diferentes que un desarrollador puede seleccionar según sus necesidades de aplicación. El primer módulo ESP32 y el más popular es el ESP32-WROOM-32D, que funciona hasta 240 MHz. El módulo incluye una antena de rastreo de placa de CI, que simplifica la implementación. También evita tener que agregar el hardware adicional y la complejidad de diseño asociada con una antena conectada IPEX. Sin embargo, si se selecciona la opción de conector IPEX, hay muchas buenas opciones de antenas, como la W24P-U de Inventek Systems.

2. Objetivos

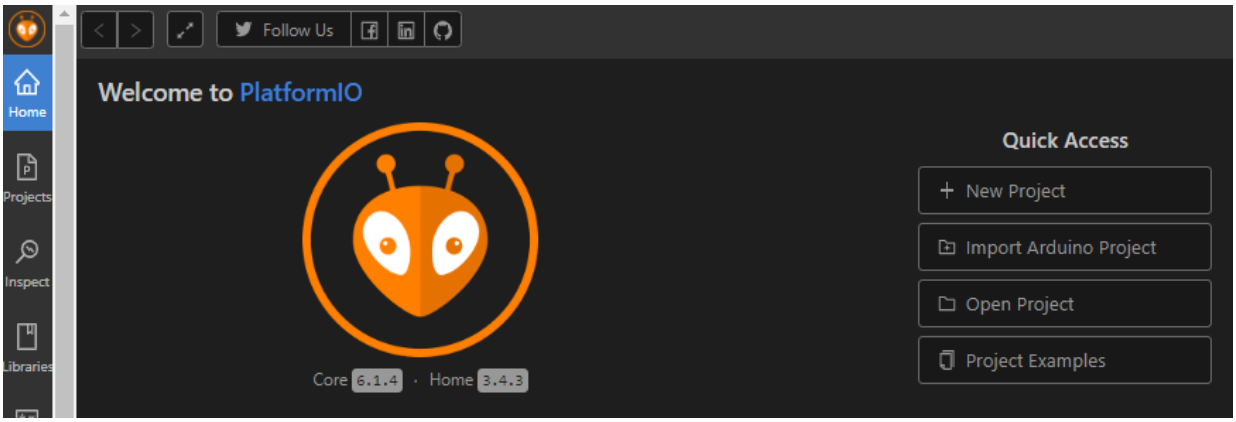
- Generar un algoritmo que sea capaz de encender tres leds de manera independiente y secuencial utilizando retardos.

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++.

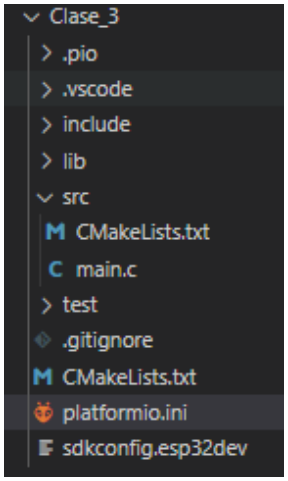
El primer paso consiste en instalar los drivers para que el computador reconozca la tarjeta a través de un puerto USB.

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



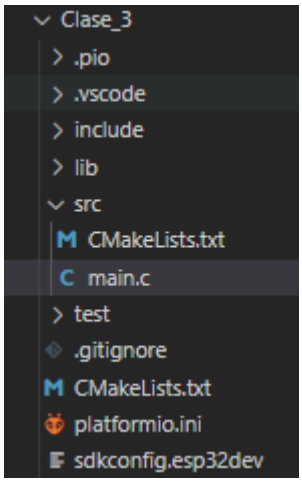
En este proyecto es necesario seleccionar el modelo de la tarjeta ESP32 y el framework que en este caso fue Espressif.

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.

```
1
2   #include <stddef.h>
3   #include "freertos/FreeRTOS.h"
4   #include "freertos/task.h"
5   #include "driver/gpio.h"
6
```

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

```
#define RLED 33
#define BLED 25
#define GLED 26

#define GPIO_RLED_PIN_SEL (1ULL << RLED)
#define GPIO_BLED_PIN_SEL (1ULL << BLED)
#define GPIO_GLED_PIN_SEL (1ULL << GLED)

uint8_t set_level = 0;
```

Después de esto se declararon los puertos de entrada y salida por medio de la librería GPIO.

```
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_BLED_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);
}
```

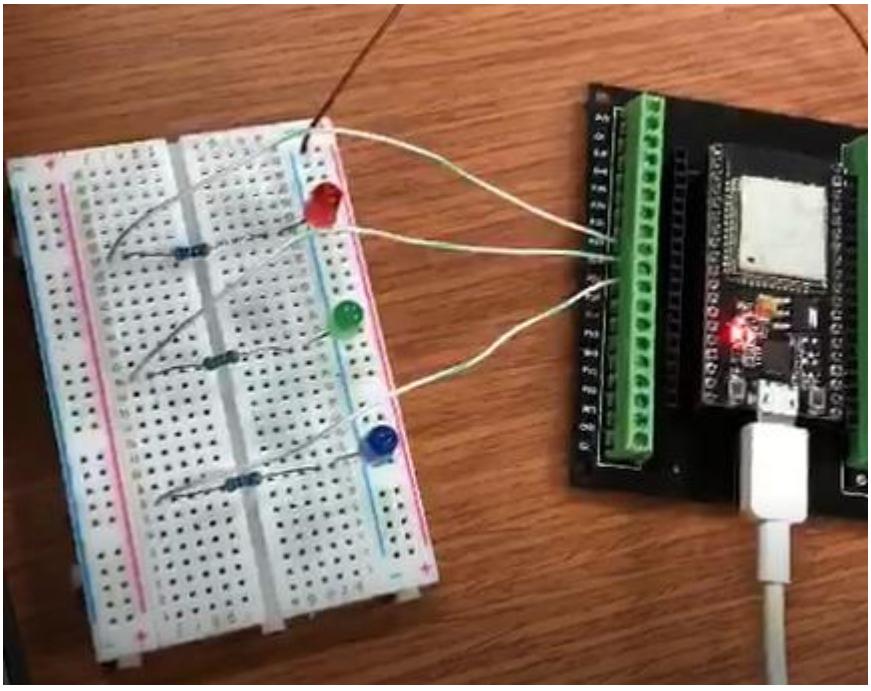
Por último, se estableció el cuerpo principal del algoritmo que consiste la inicialización de los puertos configurados por medio de la librería GPIO. Después, se estructuro un algoritmo repetitivo infinito donde se enciende el led rojo por 500 ms, después el led azul por 1000 ms, sin apagar el led rojo, y por ultimo se enciende el led verde por 1500 ms. Al repetirse la secuencia ocurre lo inverso, la variable set_level se invierte produciendo que el led rojo se apague por 500 ms, después el led azul por 1000 ms y por último el led verde por 1500 ms.

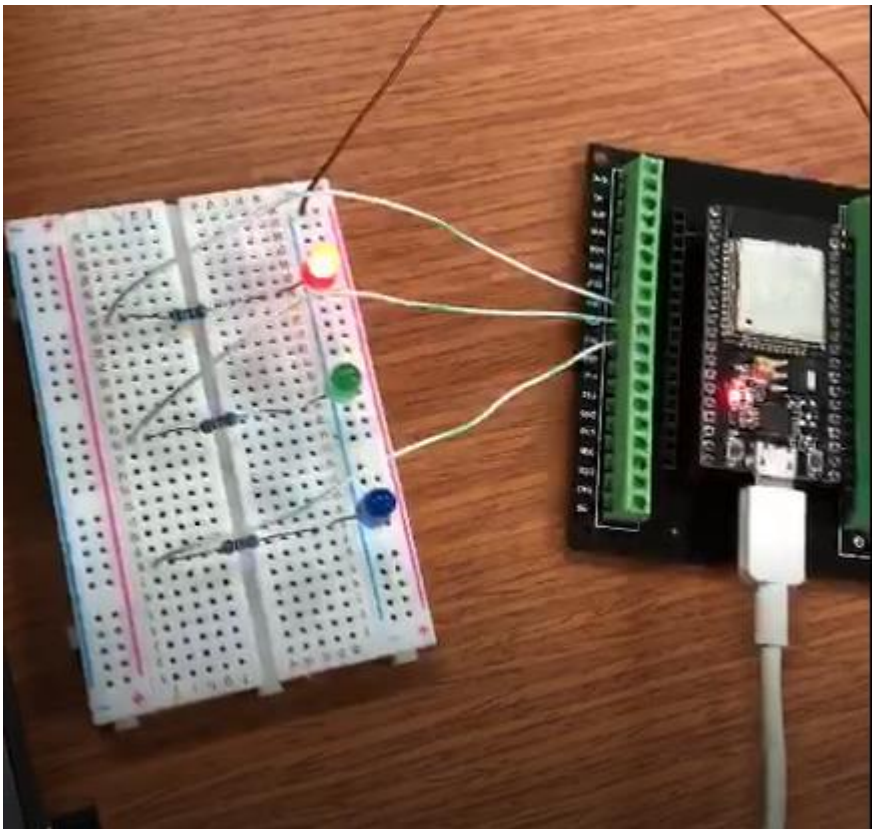
```c
void app_main()
{

    init_hw();

    while(1)
    {
        set_level = !set_level;
        gpio_set_level(RLED, set_level);
        vTaskDelay(pdMS_TO_TICKS(500));
        gpio_set_level(BLED, set_level);
        vTaskDelay(pdMS_TO_TICKS(1000));
        gpio_set_level(GLED, set_level);
        vTaskDelay(pdMS_TO_TICKS(1500));
            }

}
```

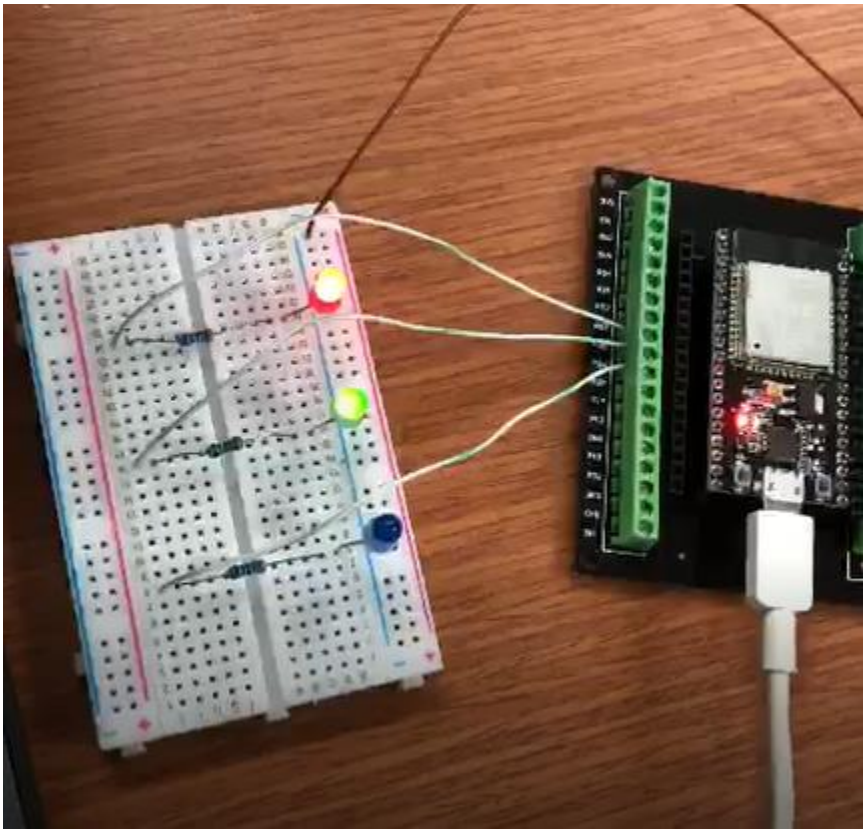
El codigo completo se ajunta en la sección de anexos.

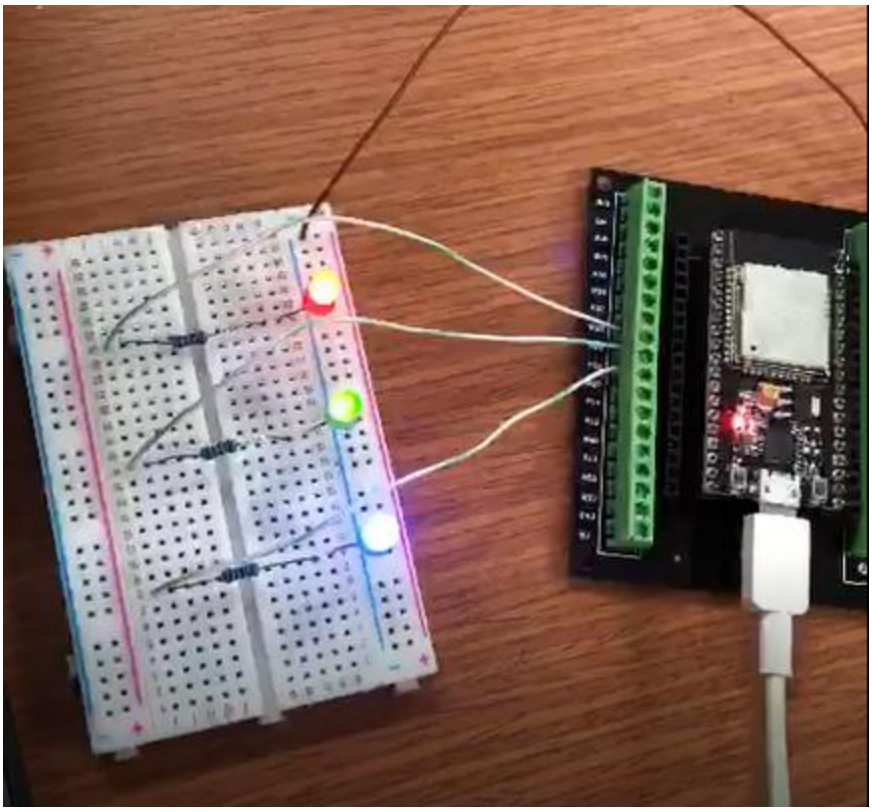
4. Resultados

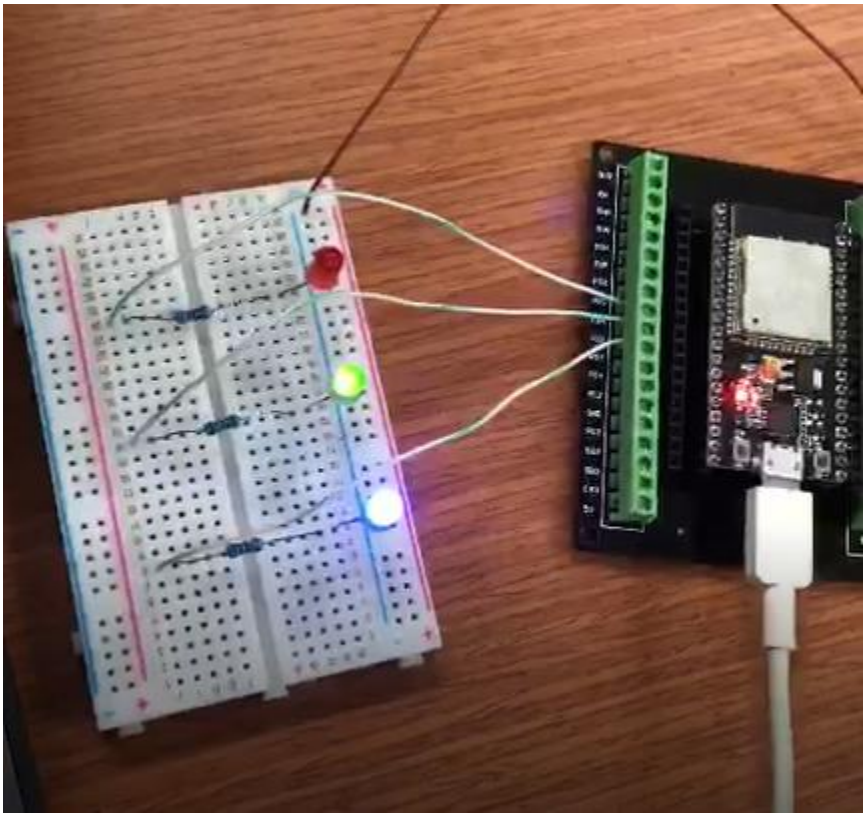
En las siguientes imágenes se mostrará la secuencia de encendido y apagado de los leds. Adicionalmente se adjuntará un video del funcionamiento en la entrega del informe.

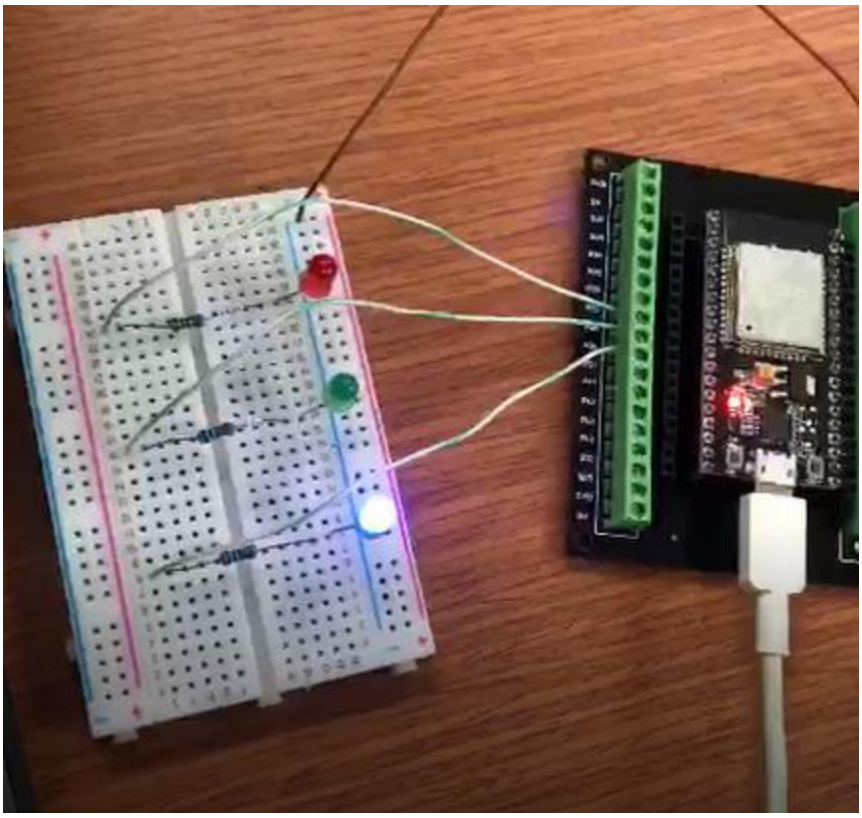












5. Conclusiones

En esta experiencia de laboratorio se logró familiarizarse con el entorno de Visual Studio Code y la interconexión de la computadora con el computador. También la estructura básica de un programa utilizando la extensión platformIO. Este procedimiento nos permitirá posteriormente crear algoritmos más complejos en futuras experiencias de laboratorio.