INFORME PRÁCTICA 1: BLINK

Para esta práctica hemos utilizado el microcontrolador **ESP32**

1ª parte: LED ENCENDIDO DURANTE UN TIEMPO INFINITO

Código 1

```
#include <Arduino.h>
#define DELAY 500

void setup() {
    pinMode(21, OUTPUT);
}

void loop() {
    digitalWrite(21, HIGH);
    delay(DELAY);
    digitalWrite(21, LOW);
    delay(DELAY);
}
```

El *setup()* se configuran las variables y, como en este caso, se establecen los pines de salida. El *pinMode* define el pin del microcontrolador que he utilizado para este programa: el numero **21**.
Agregamos la función *delay()*, la cual determina el tiempo que hay entre que se enciende y se apaga el led. Definimos la duración al principo del código, como una constante.

2ª parte: LEDS ALTERNADOS "ON" Y "OFF"

Código 2

```
#include <Arduino.h>
#define DELAY 1000

void setup() {
pinMode(23, OUTPUT);
pinMode(21, OUTPUT);
Serial.begin(115200);
}

void loop(){
    digitalWrite(21, HIGH);
    Serial.println("ON");
    delay(DELAY);
    digitalWrite(21, LOW);
    Serial.println("OFF");
    delay(DELAY)
}
```

Para este 2ndo código, simplemente añadimos la función *Serial.println()* para imprimir por puerto si el led está encendido "*ON*", o apagado "*OFF*".

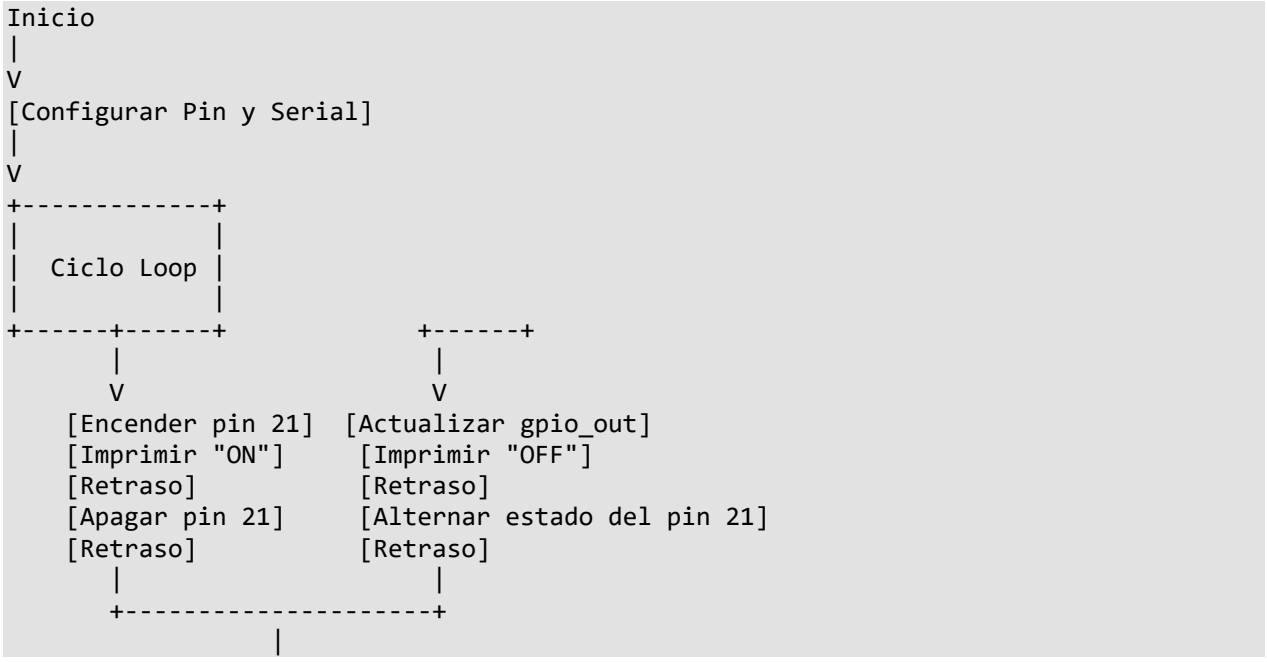
3ª parte: LEDS ALTERNADOS "ON" Y "OFF"

Código 3

```
#include <Arduino.h>
#define DELAY 500

uint32_t *gpio_out = (uint32_t *)GPIO_OUT_REG;

void setup() {
pinMode(21, OUTPUT);
Serial.begin(115200);
}

void loop(){
    digitalWrite(21, HIGH);
    *gpio_out |= (1 << 21);
    Serial.print("ON");
    delay(DELAY);
    digitalWrite(21, LOW);
    *gpio_out ^= (1 << 21);
    Serial.print("OFF");
    delay(DELAY)
}
```

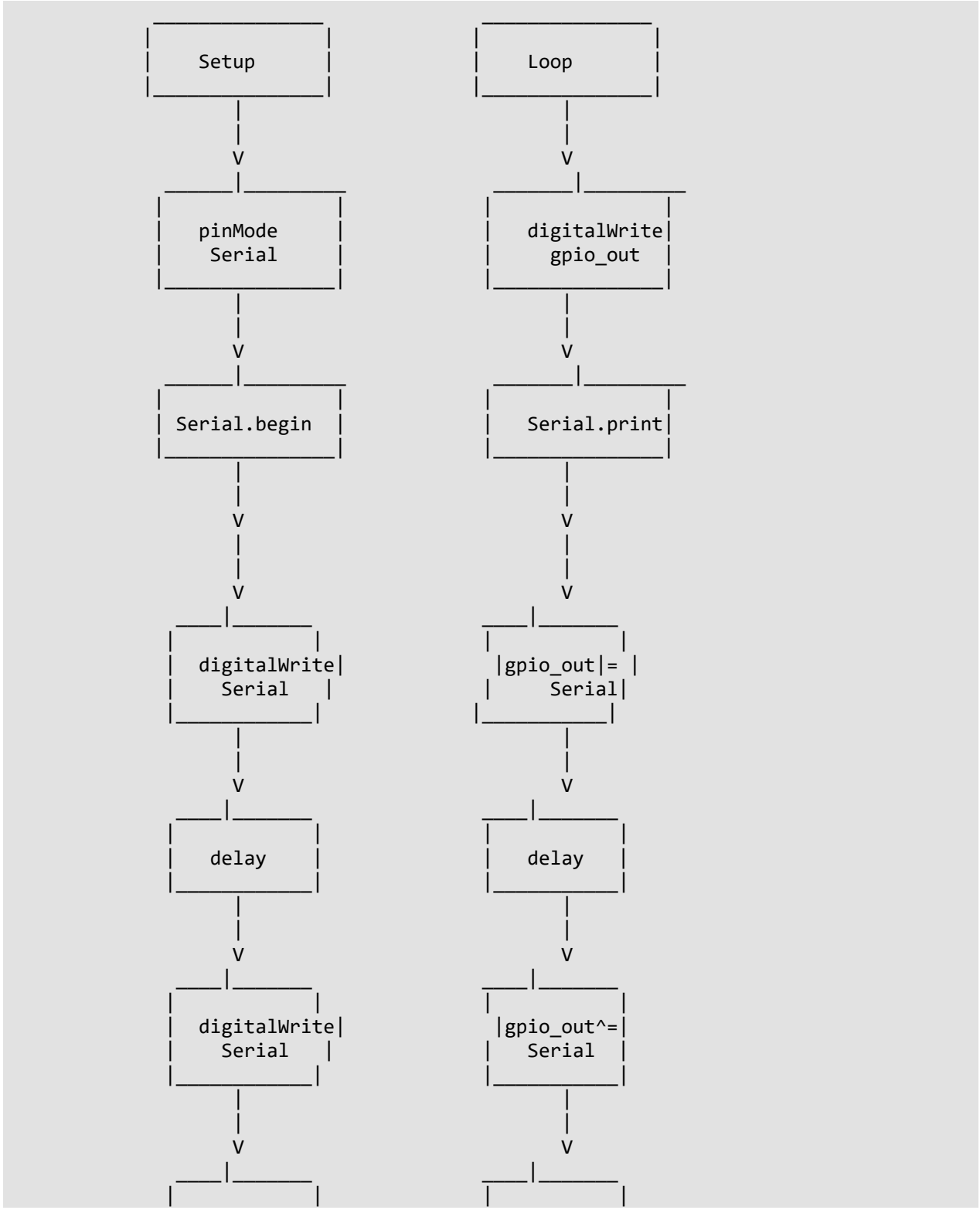
Para este 3er código hemos declarado al principio un puntero a un valor sin signo de 32 bits, llamado *gpio_out* y lo inicializamos con la dirección del registro de salida GPIO. Ya en el *loop()*, hemos añadido **gpio_out |= (1 << 21)**, lo que establece el bit correspondiente al pin 21 del registro de salida GPIO en 1, sin afectar el estado de los demás bits en el registro. Y también añadimos, ***gpio_out ^= (1 << 21)**, que invierte el estado del bit correspondiente al pin 21 del registro de salida GPIO. Es decir, si el bit estaba en 1, lo cambia a 0; y si estaba en 0, lo cambia a 1.

Diagrama de flujo del código



```
                V
              Fin
```

Diagrama de tiempos del código



Pregunta: En el programa que se ha realizado, cuál es el tiempo libre que tiene el procesador?

Para calcular el tiempo libre del procesador en el programa proporcionado, necesitamos conocer la duración total del ciclo *loop()* y la suma de todas las demoras (*delay()*) dentro de ese ciclo.
En el ciclo *loop()*, hay dos llamadas a *delay(DELAY)* y cada una de ellas pausa la ejecución durante un período de tiempo definido por la constante DELAY, que se establece en 500 ms.

Por lo tanto, el tiempo total que el procesador pasa ejecutando las instrucciones del *loop()* es la suma de los tiempos de todas las instrucciones dentro de este ciclo, que en este caso es la duración de los dos retrasos *delay()*.
do que cada retraso *delay(DELAY)* es de 500 ms, el tiempo total que el procesador pasa ejecutando las instrucciones es de 1000 ms (500+500).

El **tiempo libre** del procesador será el tiempo restante en un segundo después de haber ejecutado todas las instrucciones. Dado que el tiempo total de ejecución es de 1000 milisegundos, el tiempo libre del procesador será de 1000 milisegundos (1 segundo) menos el tiempo de ejecución, es decir, 1000 milisegundos - 1000 milisegundos = **0 milisegundos**.