

# III LED Oscillator

Contreras R Orlando<sup>1,2\*</sup> and Garcia B Iker<sup>1,2\*†</sup>

<sup>3</sup>Departamento de Sistemas Electrónicos, UAA, Av. Universidad 940,  
Aguascalientes, 20131, Aguascalientes, México.

\*Corresponding author(s). E-mail(s): [{al348390,al307630}@edu.uaa.mx](mailto:{al348390,al307630}@edu.uaa.mx);

†Ambos autores contribuyeron de forma equitativa a este trabajo.

## Abstract

El objetivo de la siguiente práctica es generar un péndulo LED por medio de recorrimientos, su valor correspondiente es mandado a los GPIO de salida y poder modificar la velocidad a la que se mueve el pendulo por medio de botones.

**Keywords:** ASM, ARM, Shift, Registros

## 1 Introducción

En esta práctica se busco hacer recorrer un péndulo sin perder los valores, además con dos botones podemos modificar la velocidad de éste mismo. Cuando se realiza códigos en ensamblador se debe considerar la configuración de los registros, es decir debe configurarse todo de la forma correcta, para esto se debe consultar el Reference Manual para revisar cada configuración de forma correcta

## 2 Objetivos

- Diseñar un código funcional para generar un pendulo led con velocidad variable.
- Uso de los GPIO de entrada y Salida.
- Configuracion de puertos de manera efectiva.

### 3 Pseudocódigo

A continuación, se presenta el pseudocódigo que representa la lógica del programa en ensamblador:

---

**Algorithm 1** Control de LEDs con desplazamiento y ajuste de velocidad

---

**Input:** Ninguno

**Output:** Control de LED con movimiento y velocidad ajustable por botones

```
1 confRCC() confGPIOC()
2 r5 ← led_delay r4 ← led_delay r2 ← 0x0070          // Estado inicial del LED
3 r3 ← 0 r11 ← 1000000                                // Máximo delay
4 r10 ← 54000                                           // Mínimo delay
5 while true do
    // --- Movimiento a la derecha ---
6    while r2 ≠ 0x0007 do
7        r3 ← r2 ≫ 2 r1 ← r3 Escribir r1 en GPIOA_ODR r2 ← r2 ≫ 1 Delay()
8        Leer botones de GPIOB_IDR en r7 r7 ← r7 & 0x03
9        if r7 = 0x02 then
10           suma()
11        else if r7 = 0x01 then
12           resta()
    // --- Movimiento a la izquierda ---
13    while r2 ≠ 0x0E00 do
14        r3 ← r2 ≫ 2 r1 ← r3 Escribir r1 en GPIOA_ODR r2 ← r2 ≪ 1 Delay()
15        Leer botones de GPIOB_IDR en r7 r7 ← r7 & 0x03
16        if r7 = 0x02 then
17           suma()
18        else if r7 = 0x01 then
19           resta()
```

---

---

**Algorithm 2** Subrutinas del sistema

---

**Función confRCC():**

└ Activar relojes para GPIOA y GPIOB mediante RCC\_AHB1ENR

**Función confGPIOC():**

└ Configurar pines 0-7 de GPIOA como salida Configurar velocidad máxima para  
└ GPIOA Configurar pines 0 y 1 de GPIOB como entrada con pull-down

**Función Delay():**

└ Decrementar r5 hasta que sea 0 Restaurar r5 ← r4

**Función suma():**

└ if r5 < r11 then  
└ Incrementar r5 en 1 r4 ← r5

**Función resta():**

└ if r5 > r10 then  
└ Decrementar r5 en 1 r4 ← r5

---

 [GitHub Repository](#)

## 4 Explicación del Código Ensamblador

Este es el código original en ensamblador que corresponde al pseudocódigo anterior:

```
;Direcciones relojes
RCC_BASE EQU 0x40023800
RCC_AHB1ENR EQU (RCC_BASE + 0x30)

; ----- GPIO A -----
GPIOA_BASE EQU 0x40020000
GPIOA_MODER EQU (GPIOA_BASE + 0x00)
GPIOA_OTYPER EQU (GPIOA_BASE + 0x04)
GPIOA_OSPEED EQU (GPIOA_BASE + 0x08)
GPIOA_PUPDR EQU (GPIOA_BASE + 0x0C)
GPIOA_IDR EQU (GPIOA_BASE + 0x10)
GPIOA_ODR EQU (GPIOA_BASE + 0x14)
GPIOA_BSSR EQU (GPIOA_BASE + 0x18)

; ----- GPIO B -----
GPIOB_BASE EQU 0x40020400
GPIOB_MODER EQU (GPIOB_BASE + 0x00)
GPIOB_OTYPER EQU (GPIOB_BASE + 0x04)
GPIOB_OSPEED EQU (GPIOB_BASE + 0x08)
GPIOB_PUPDR EQU (GPIOB_BASE + 0x0C)
GPIOB_IDR EQU (GPIOB_BASE + 0x10)
GPIOB_ODR EQU (GPIOB_BASE + 0x14)
GPIOB_BSSR EQU (GPIOB_BASE + 0x18)

led_delay EQU 400000

AREA my_data, DATA, READWRITE
AREA myCode, CODE, READONLY
ENTRY
EXPORT __main

__main
BL confRCC
BL confGPIOC
LDR R5,=led_delay
LDR R4,=led_delay
MOVW R2, #0x0070
EOR R3,R3
LDR R11, =1000000
MOV R10, #54000
LDR R0,=GPIOA_ODR

MOVDER
LSR R3, R2, #2
MOV R1, R3
STR R1,[R0]
CMP R2, #7
BEQ MOVIZQ
LSR R2,R2, #1
BL Delay

LDR R6, =GPIOB_IDR
LDR R7,[R6]
AND R7,#0x0003
CMP R7,#0x0002
BEQ suma
CMP R7,#0x0001
BEQ resta
B MOVDER

MOVIZQ
LSR R3, R2, #2
MOV R1, R3
STR R1,[R0]

CMP R2, #0xE00
BEQ MOVDER
LSL R2,R2, #1
BL Delay

suma
CMP R5, R11
BXHI LR
ADD R5,R5,#1
MOV R4,R5
BX LR

resta
CMP R5,R10
BXLO LR
SUB R5,R5,#1
MOV R4,R5
BX LR

;===== Subrutinas =====

confRCC
LDR R0,=RCC_AHB1ENR
LDR R1,[R0]
ORR R1,R1,#0x03
STR R1,[R0]
BX LR

confGPIOC
LDR R0,=GPIOA_MODER
LDR R1,[R0]
LDR R2,=0x00005555
ORR R1,R2
STR R1,[R0]

LDR R0,=GPIOA_OSPEED
LDR R1,[R0]
LDR R2,=0x0000FFFF
ORR R1,R2
STR R1,[R0]

LDR R0,=GPIOB_PUPDR
LDR R1,[R0]
LDR R2,=0x00000005
ORR R1,R2
STR R1,[R0]
BX LR

Delay
SUBS R5,R5,#1
BNE Delay
MOV R5,R4
BX LR

ALIGN
END
```

## 4.1 Direcciones de Registro

- **RCC\_BASE**: Dirección base del Registro de Control del Reloj.
- **GPIOA\_BASE** y **GPIOB\_BASE**: Direcciones base de los puertos GPIO A y B.
- **Registros MODER, OTYPER, OSPEED, PUPDR**: Configuración de los GPIOs.
- **Registros IDR y ODR**: Entrada y salida de datos.

## 4.2 Inicialización del Sistema

El código comienza habilitando los relojes de los periféricos GPIO mediante el registro **RCC\_AHB1ENR**.

## 4.3 Configuración de GPIO

- Se configuran los pines del puerto A como salida.
- Se ajusta la velocidad y se establece un **pull-down** en los pines de entrada del puerto B.

## 4.4 Lógica de Control

El código maneja un desplazamiento de bits en los registros **GPIOA\_ODR**, alternando entre izquierda y derecha.

## 4.5 Subrutinas

- **confRCC**: Habilita los relojes de los GPIO.
- **confGPIOC**: Configura los pines de entrada y salida.
- **Delay**: Introduce retardos en el sistema.
- **suma/resta**: Ajusta el temporizador dinámicamente.

# 5 Conclusiones

Esta práctica nos presentó varios problemas a la hora de implementar el código, pero esto nos sirvió para aprender a resolver problemas comunes, nos sirvió de práctica para poder depurar de forma efectiva, consideramos que el depurar un código es una herramienta muy esencial tanto como la lógica que se trabaja en equipo, son cosas muy necesarias para trabajar de manera ordenada y eficiente.