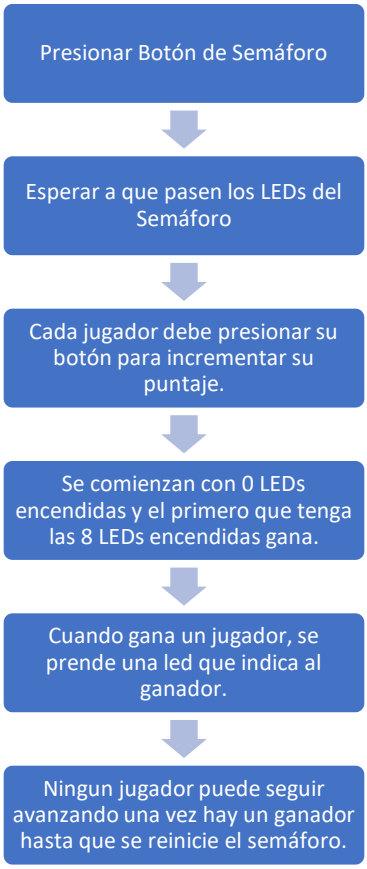
Universidad del Valle de Guatemala

Electrónica Digital II

Rodrigo Díaz, 18265

Lab #1

Pseudocódigo



Código Documentado

```
/*
 * File:   main.c
 * Author: Rodrigo Díaz
 *
 */
```

```
//*****************************************************************************
// Importación de librerías
//*****************************************************************************

#include <xc.h>

//*****************************************************************************
// Palabra de configuración
//*****************************************************************************

// CONFIG1
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)

// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)
```

```
//*****************************************************************************
// Variables
//*****************************************************************************
// la variable XTAL FREQ es necesaria para que funcionen los delays
#define _XTAL_FREQ 8000000
// Se asginan variables de Leds a los primeros 3 bits del puerto E
#define LED_Rojo PORTEbits.RE0
#define LED_Amarillo PORTEbits.RE1
#define LED_Verde PORTEbits.RE2
// se crean variables tipo char que se utilizaran en el programa
unsigned char check = 0;
unsigned char j1 = 0;
unsigned char j2 = 0;

//*****************************************************************************
// Prototipos de funciones
//*****************************************************************************
void setup(void);
void semaforo(void);
void avance(void);

//*****************************************************************************
// Ciclo principal
//*****************************************************************************

void main(void) {
```

```
  // mando a llamar a la funcion de setup
  setup();

  //*****************************************************************************
  // Loop principal
  //*****************************************************************************
  // como es un while(1) siempre se va a repetir este loop.
  while (1) {
   // se tiene un antirebote para el boton que activa el semáforo
    if (PORTBbits.RB0 == 0) {
      __delay_ms(50);
      if (PORTBbits.RB0 == 1) {
        // se manda a llamar primero al semaforo y después a la funcion
        // de avance para que participen los jugadores
        semaforo();
        avance();
      }
    }
  }


}

//*****************************************************************************
// Configuración
//*****************************************************************************

void setup(void) {
  // Todos los bits utilizados se configuran como salidas, menos los primeros
```

```
   // 3 bits del puerto B, debido a que allí estan los push. Ansel y Anselh
   // se ponen en 0 porque no se utilizarán señales analógicas.
   TRISE = 0;
   PORTE = 0;
   ANSEL = 0;
   ANSELH = 0;
   TRISB = 0b00000111;
   PORTB = 0;
   TRISC = 0;
   PORTC = 0;
   TRISD = 0;
   PORTD = 0;
   PORTA = 0;
   TRISA = 0;
}

//*****************************************************************************
// Funciones
//*****************************************************************************

void semaforo(void) {
   // Al inicio del juego, todos los puertos que involucren los LEDs de los
   // jugadores se apagan.
   PORTC = 0;
   PORTD = 0;
   PORTA = 0;
   // Se encienden los colores del semáforo con un delay especificado
   LED_Rojo = 1;
   __delay_ms(150);
```

```
    LED_Rojo = 0;
    LED_Amarillo = 1;
    __delay_ms(150);
    LED_Amarillo = 0;
    LED_Verde = 1;
    __delay_ms(150);
    LED_Verde = 0;
    // la variables check se enciende para poder comenzar a contar el punteo
    check = 1;
}

void avance(void) {
    // solamente si check es 1, se puede entrar al loop
    while (check == 1) {
        // se tiene un antirebote para el botón del jugador 1
        if (PORTBbits.RB1 == 0) {
            __delay_ms(50);
            if (PORTBbits.RB1 == 1) {
                // se revisa si es su primer botonazo para encender el primer
                // LED, sino solo hace un corrimiento de bits. Una vez se llega
                // al octavo LED, se apaga check para que no se pueda seguir
                //jugando y se enciente el LED que ganó el jugador 1.
                if (PORTC == 0) {
                    j1 = 0b00000001;
                    PORTC = j1;
                }
                else if (PORTC != 0) {
                    j1 = j1*2;
                    PORTC = j1;
```

```
            }
            if (PORTCbits.RC7 == 1) {
                check = 0;
                PORTAbits.RA0 = 1;
            }
        }
    }
    // antirebote para el jugador 2
    if (PORTBbits.RB2 == 0) {
        __delay_ms(50);
        if (PORTBbits.RB2 == 1) {
            // tiene la misma estructura que para el jugador 1, pero con
            //los puertos y variables del jugador 2
            if (PORTD == 0) {
                j2 = 0b00000001;
                PORTD = j2;
            }
            else if (PORTD != 0) {
                j2 = j2*2;
                PORTD = j2;
            }
            if (PORTDbits.RD7 == 1) {
                check = 0;
                PORTAbits.RA1 = 1;
            }
        }
    }
  }
}
```

Link Repositorio Github

https://github.com/RodDia2/Labs_Digital_2