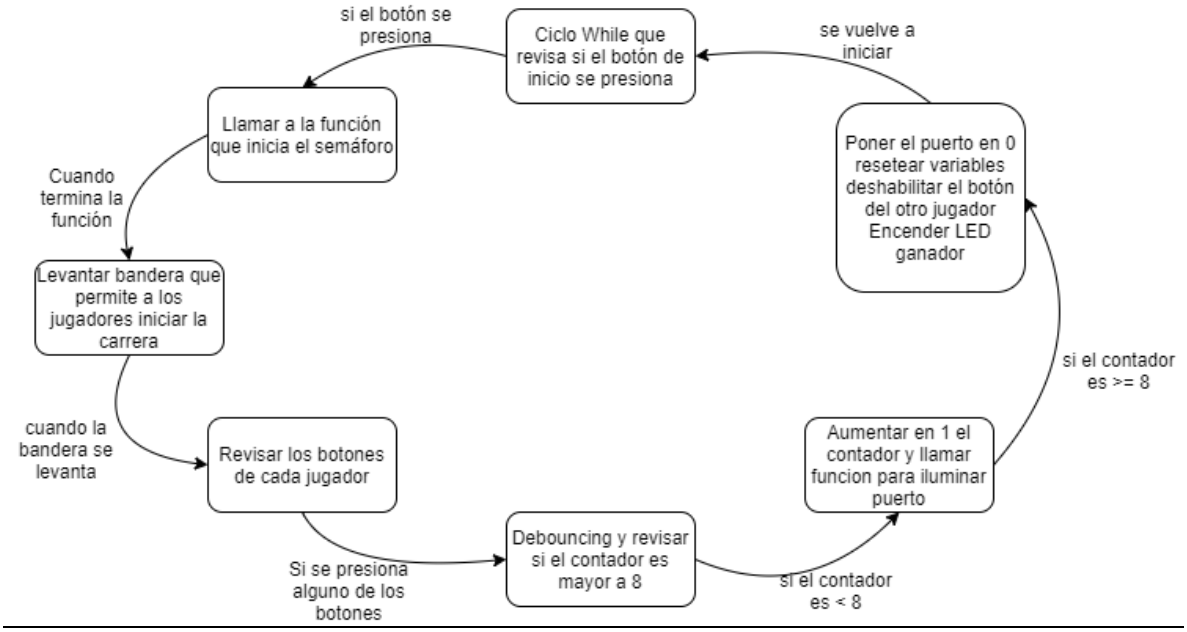
Universidad del Valle de Guatemala
Electrónica Digital 1
Sección 20

Katharine Senn Salazar
Carné 18012

Laboratorio No. 1

- Pseudocódigo



- Código MPLABX

```
//******************************************************************************

/*

 * File:   main.c

 * Author: katha

 *

 * Created on 25 de enero de 2021, 12:29 PM

 */


//******************************************************************************

//Librerias

//******************************************************************************

#include <xc.h>
```

```
// 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 MCLR)
#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)

#define _XTAL_FREQ 8000000

//*****************************************************************************
//Variables
//*****************************************************************************
int contador = 0;
int contador2 = 0;
int GO_var = 0; //Bandera que permite iniciar la carrera
int semaf_var = 0;
//*****************************************************************************
//Prototipos de funciones
//*****************************************************************************
void setup(void);
void semaf(void);
```

```
void leds_1(void);
void leds_2(void);
void ganador_1(void);
void ganador_2(void);

//*****************************************************************************
//Ciclo Principal
//*****************************************************************************

void main(void) {
  setup();

  while (1) {
    if (PORTAbits.RA0 == 0) { //al presionar el boton de inicio del semáforo
      PORTBbits.RB0 = 0;    //se hace un reset de variables y puertos
      PORTBbits.RB1 = 0;
      contador = 0;
      contador2 = 0;
      GO_var = 0;
      PORTC = 0b0000000;
      PORTD = 0b0000000;
      PORTBbits.RB0 = 0;
      PORTBbits.RB1 = 0;

      while (PORTAbits.RA0 == 0) { //el jugador no puede iniciar antes que se active el semádoto
          semaf_var = semaf_var;
          GO_var = 0;
        }
      semaf_var = 1;
      semaf();
    }

    if (GO_var == 1) {
```

```
        if (PORTAbits.RA1 == 0) {
           while (PORTAbits.RA1 == 0) { //ciclo para debouncing
              contador = contador;
           }
           contador = contador + 1;
           if (contador <= 8){
              leds_1(); //función para indicar que led debe encenderse
           }
           else{
            ganador_1();  //función que indica que ganó el jugador 1
           }
        }

        if (PORTAbits.RA2 == 0) {
           while (PORTAbits.RA2 == 0) { //ciclo para debouncing
              contador2 = contador2;
           }
           contador2 = contador2 + 1;
           if (contador2 <=8 ){
              leds_2();   //función para indicar que led debe encenderse
           }
           else{
            ganador_2();  //función que indica que ganó el jugador 2
           }

        }

     }
  }
}

//*******************************************************************************
//Configuracion
```

```
//*****************************************************************************

void setup(void) {
   ANSEL = 0;
   ANSELH = 0;
   TRISC = 0;
   PORTC = 0;
   TRISD = 0;
   PORTD = 0;
   TRISE = 0;
   PORTE = 0;
   TRISB = 0;
   PORTB = 0;
   TRISA = 0b00000111;
   PORTA = 0;




}
//*****************************************************************************
//Funciones
//*****************************************************************************

void semaf(void) {
   PORTEbits.RE0 = 1;
   __delay_ms(800);
   PORTEbits.RE0 = 0;
   PORTEbits.RE1 = 1;
   __delay_ms(800);
   PORTEbits.RE1 = 0;
   PORTEbits.RE2 = 1;
```

```
    __delay_ms(800);
    PORTEbits.RE2 = 0;
    GO_var = 1; //se levcanta la bandera que permite empezar a los usuarios
}

void leds_1(void) { //se asigna un valor al puerto C según el valor del contador
    if (contador == 1) {
        PORTC = 0b00000001;
    }
    else if (contador == 2) {
        PORTC = 0b00000010;
    }
    else if (contador == 3) {
        PORTC = 0b00000100;
    }
    else if (contador == 4) {
        PORTC = 0b00001000;
    }
    else if (contador == 5) {
        PORTC = 0b00010000;
    }
    else if (contador == 6) {
        PORTC = 0b00100000;
    }
    else if (contador == 7) {
        PORTC = 0b01000000;
    }
    else if (contador == 8) {
        PORTC = 0b10000000;
    }
}
void leds_2(void) { //se asigna un valor al puerto D según el valor del contador
    if (contador2 == 1) {
```

```
        PORTD = 0b00000001;
    }
    else if (contador2 == 2) {
        PORTD = 0b00000010;
    }
    else if (contador2 == 3) {
        PORTD = 0b00000100;
    }
    else if (contador2 == 4) {
        PORTD = 0b00001000;
    }
    else if (contador2 == 5) {
        PORTD = 0b00010000;
    }
    else if (contador2 == 6) {
        PORTD = 0b00100000;
    }
    else if (contador2 == 7) {
        PORTD = 0b01000000;
    }
    else if (contador2 == 8) {
        PORTD = 0b10000000;
    }
}
void ganador_1(void) { //se cambia el bit que muestra la victoria del jugador
    PORTC = 0b00000000; //se apaga todo el puerto de dicho jugador y se reinician los contadores
    GO_var = 0;
    PORTBbits.RB0 = 1;
    contador = 0;
    contador2 = 0;
}
void ganador_2(void) { //se cambia el bit que muestra la victoria del jugador
    PORTD = 0b00000000; //se apaga todo el puerto de dicho jugador y se reinician los contadores
```

```
    GO_var = 0;
    PORTBbits.RB1 = 1;
    contador = 0;
    contador2 = 0;
}
```

- Link Repositorio GitHub

https://github.com/sen18012/Labs_Digital_2