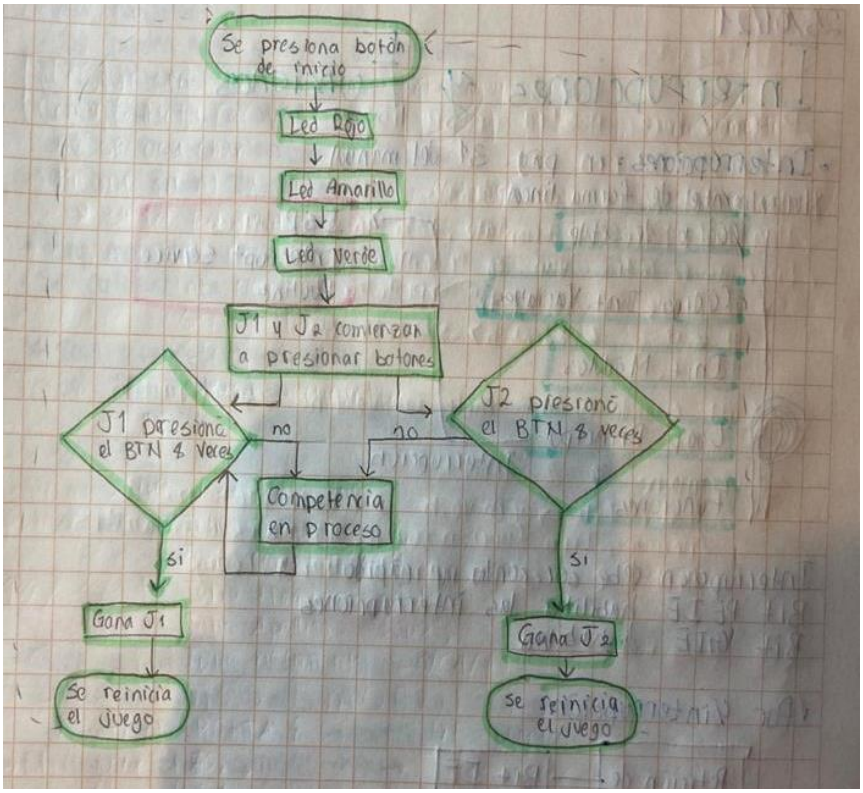
Universidad del Valle de Guatemala 28/01/2021

Electrónica Digital Sección 10 Nicole Prem 18337

Laboratorio # 1

Juego Carrera

Pseudocódigo



Código documentado

```c
/*
 * File:   main.c
 * Author: Nicole Prem 18337
 *
 * Created on 21 de enero de 2021, 06:39 PM
 */

//******************************************************************************
//Configuración de la palabra
//******************************************************************************
#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)
//******************************************************************************
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
#include <xc.h>
//******************************************************************************
//Prototipos de funciones
//******************************************************************************
void setup(void);
void semaforo(void); //función para desplegar la secuencia de semáforo
void ledsJ1(unsigned char C1); //función para controlar los les del jugagor 1
void ledsJ2(unsigned char C2); //función para controlar los leds del jugador 2
//******************************************************************************
```

```c
void ledsJ1(unsigned char C1); //función para controlar los les del jugagor 1
void ledsJ2(unsigned char C2); //función para controlar los leds del jugador 2
//******************************************************************************
//Variables
//******************************************************************************
#define LEDR PORTEbits.RE0 //Led Rojo
#define LEDA PORTEbits.RE1 //Led Amarillo
#define LEDV PORTEbits.RE2 //Led verde
#define _XTAL_FREQ 8000000 //Frecuencia de eintrada
unsigned char C1; //parámetro de entrada para función de leds J1
unsigned char C2; //parámetro de entrada para función de leds J2
unsigned char contadorJ1 = 0; //Contador del jugador 1, si llega a 8 gana
unsigned char contadorJ2 = 0; //Contador del jugador 2, si llega a 8 gana
unsigned int bandera = 1; //bandera para identificar al ganador

//******************************************************************************
//Declaración de entradas, salidas y limpieza de puertos
//******************************************************************************
void setup(void) {
    ANSEL = 0;
    ANSELH = 0;
    TRISE = 0; //Puerto de los LEDS
    TRISC = 0; //Puerto LEDS jugador 1
    TRISD = 0; //Puerto LEDS jugador 2
    TRISB = 0b00000001; //Puerto de botón semáforo e indicador del ganador
    TRISA = 0b00000011; //Puerto para botones de los jugadores
    //Limpieza de puertos:
    PORTE = 0;
    PORTC = 0;
    PORTD = 0;
    PORTB = 0;
    PORTA = 0;
}
void main(void) {

    while (1) {
        setup();
```

```
void main(void) {

    while (1) {
        setup();
        //condiciones iniciales de los contadores de los jugadores
        contadorJ1 = 0;
        contadorJ2 = 0;
        bandera = 1;
        if (PORTBbits.RB0 == 0) { //Se presiona botón de semáfoto
            semaforo();

            while (bandera == 1) {
                if (PORTAbits.RA0 == 0) { //Se presiona botón J1
                    __delay_ms(50); //Antirebote
                    if (PORTAbits.RA0 == 1) {//antirebote
                        contadorJ1++;
                        ledsJ1(contadorJ1);

                        if (contadorJ1 == 8) { //verificación si ya ganó
                            bandera = 2;
                        }
                    }
                } else if (PORTAbits.RA1 == 0) { //Se presiona botón J2
                    __delay_ms(50); //antirebote
                    if (PORTAbits.RA1 == 1) {
                        contadorJ2++;
                        ledsJ2(contadorJ2);

                        if (contadorJ2 == 8) { //verificación si ya ganó
                            bandera = 2;
                        }
                    }

                }

            }
```

207:31

```
    }

}
//Rutina para que se haga el inicio de la carrera con semaforo

void semaforo(void) {
    LEDR = 1;
    LEDA = 0;
    LEDV = 0;
    __delay_ms(500);

    LEDR = 0;
    LEDA = 1;
    LEDV = 0;
    __delay_ms(500);

    LEDR = 0;
    LEDA = 0;
    LEDV = 1;
    __delay_ms(500);
}

// Rutina para desplegar los leds del J1

void ledsJ1(unsigned char C1) {
    if (C1 == 8) { //se verifica si ya presionó el botón 8 veces y ganó
        PORTBbits.RB2 = 1;
        PORTC = 0b10000000;
        __delay_ms(500);
    }
/*Case que indica el led que se enciende dependiendo de la cantidad de veces
 que ha presionado el botón */
        switch (C1) {
            case 1:
                PORTCbits.RC0 = 1;
                break;
            case 2:
```

116:1

```
/*Case que indica el led que se enciende dependiendo de la cantidad de veces
 que ha presionado el botón */
        switch (C1) {
            case 1:
                PORTCbits.RC0 = 1;
                break;
            case 2:
                PORTCbits.RC0 = 0;
                PORTCbits.RC1 = 1;
                break;
            case 3:
                PORTCbits.RC1 = 0;
                PORTCbits.RC2 = 1;
                break;
            case 4:
                PORTCbits.RC2 = 0;
                PORTCbits.RC3 = 1;
                break;
            case 5:
                PORTCbits.RC3 = 0;
                PORTCbits.RC4 = 1;
                break;
            case 6:
                PORTCbits.RC4 = 0;
                PORTCbits.RC5 = 1;
                break;
            case 7:
                PORTCbits.RC5 = 0;
                PORTCbits.RC6 = 1;
                break;

        }
}

// Rutina para desplegar los leds del J2
void ledsJ2(unsigned char C2) {
    if (C2 == 8) { //se verifica si ya presionó 8 veces el botón y ganó
```

116:1

Source History

```
    if (C2 == 8) { //se verifica si ya presionó 8 veces el botón y ganó
        PORTBbits.RB3 = 1;
        PORTD = 0b10000000;
        __delay_ms(500);
    }
/*Case que indica el led que se enciende dependiendo de la cantidad de veces
 que ha presionado el botón */
    switch (C2) {
        case 1:
            PORTDbits.RD0 = 1;
            break;
        case 2:
            PORTDbits.RD0 = 0;
            PORTDbits.RD1 = 1;
            break;
        case 3:
            PORTDbits.RD1 = 0;
            PORTDbits.RD2 = 1;
            break;
        case 4:
            PORTDbits.RD2 = 0;
            PORTDbits.RD3 = 1;
            break;
        case 5:
            PORTDbits.RD3 = 0;
            PORTDbits.RD4 = 1;
            break;
        case 6:
            PORTDbits.RD4 = 0;
            PORTDbits.RD5 = 1;
            break;
        case 7:
            PORTDbits.RD5 = 0;
            PORTDbits.RD6 = 1;
            break;
    }
}
```

207:6

Link del respositorio de GitHub

https://github.com/nicoleprem/Digital-2.git