Universidad Del Valle De Guatemala 24/02/2021

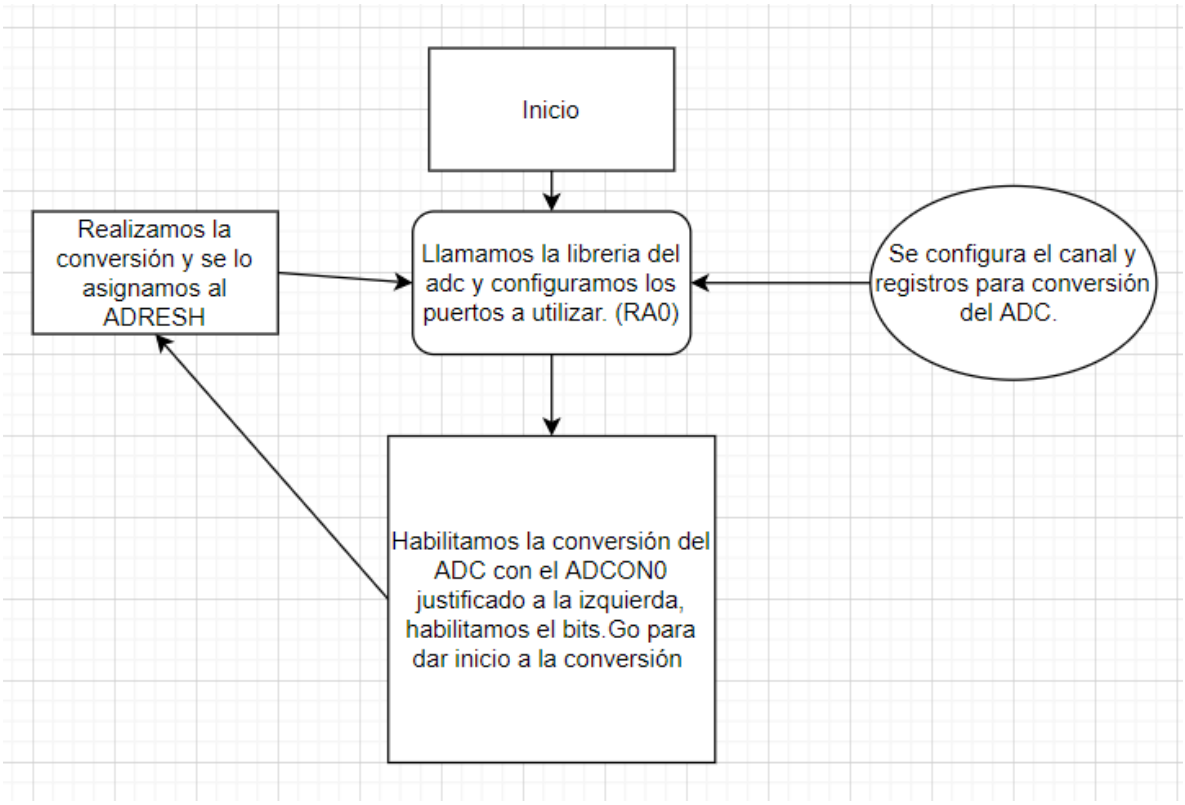
Digital 2 Helder Ovalle Barrios

Sección: 18349

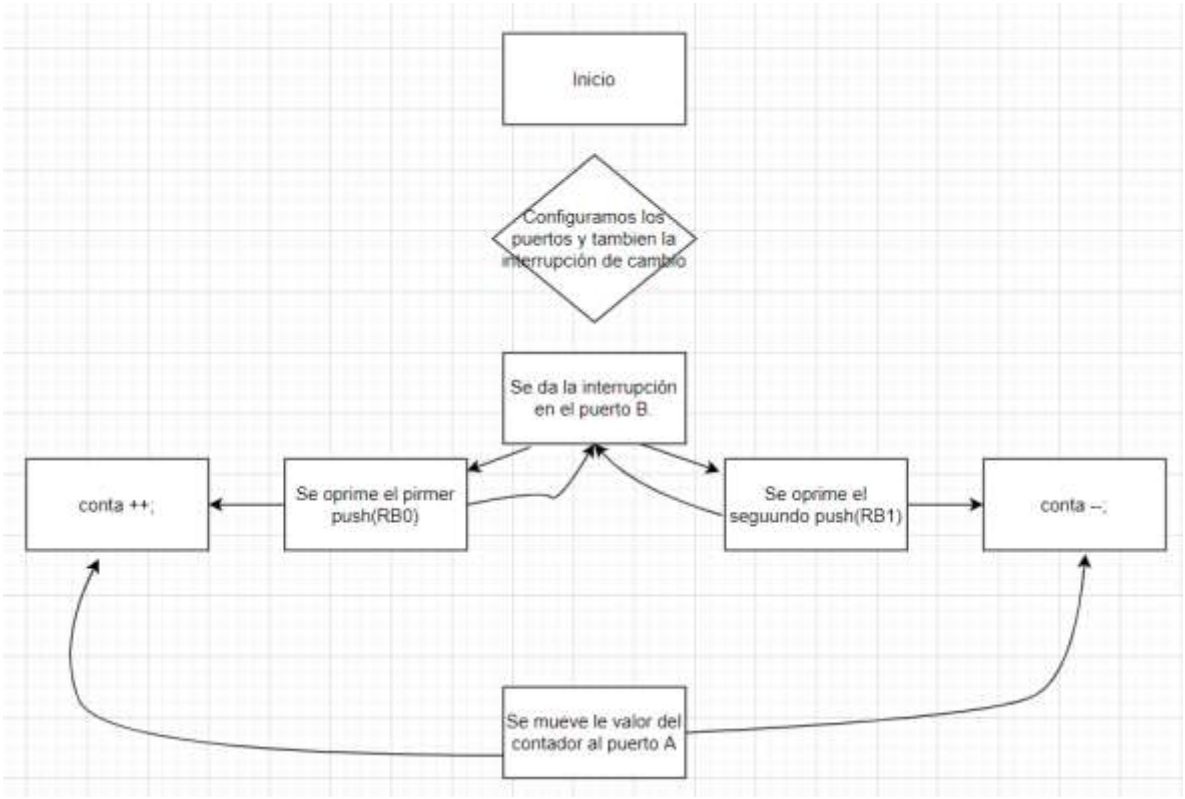
Laboratorio #2

Diagrama de flujo:

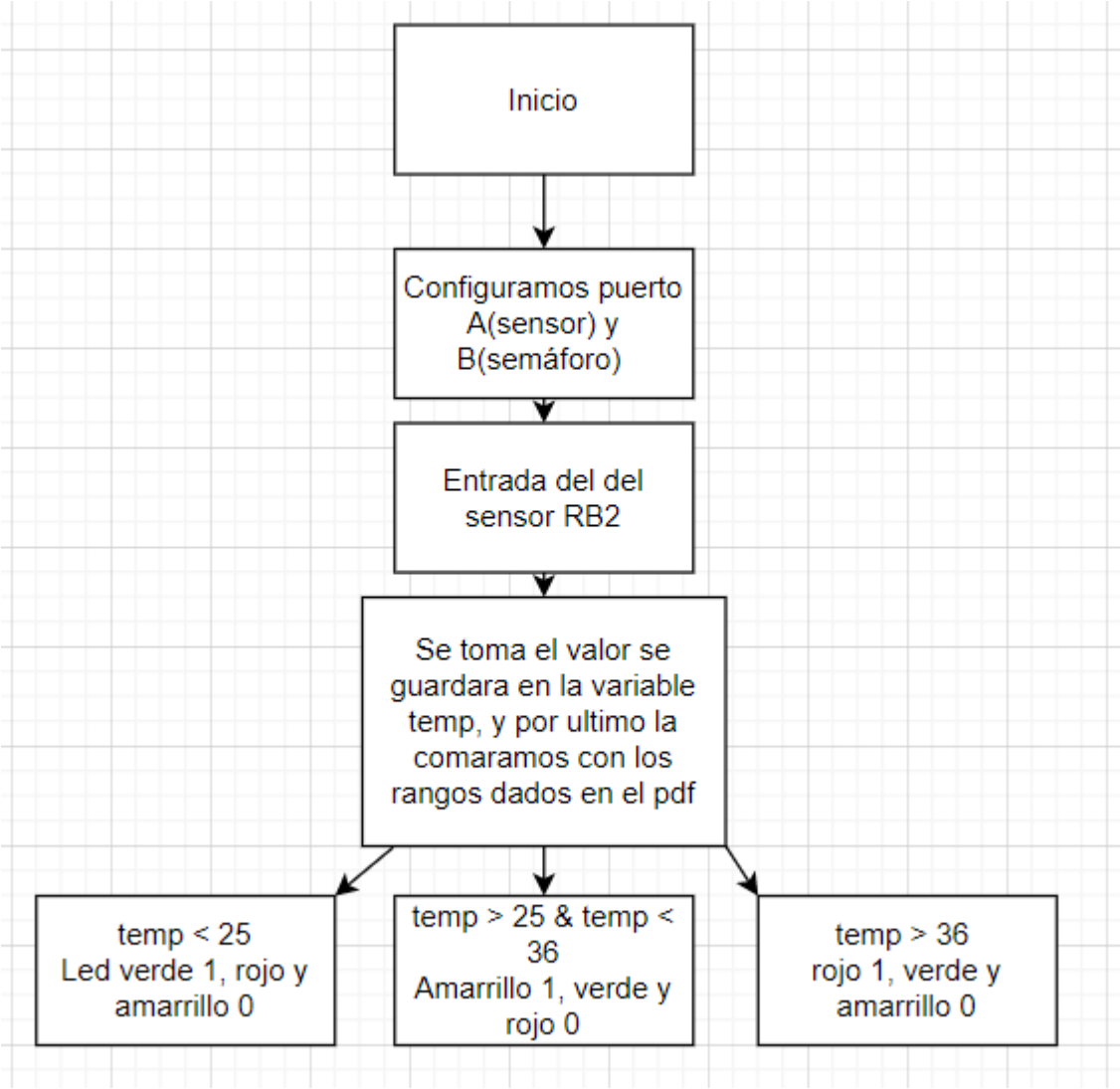
Esclavo adc:



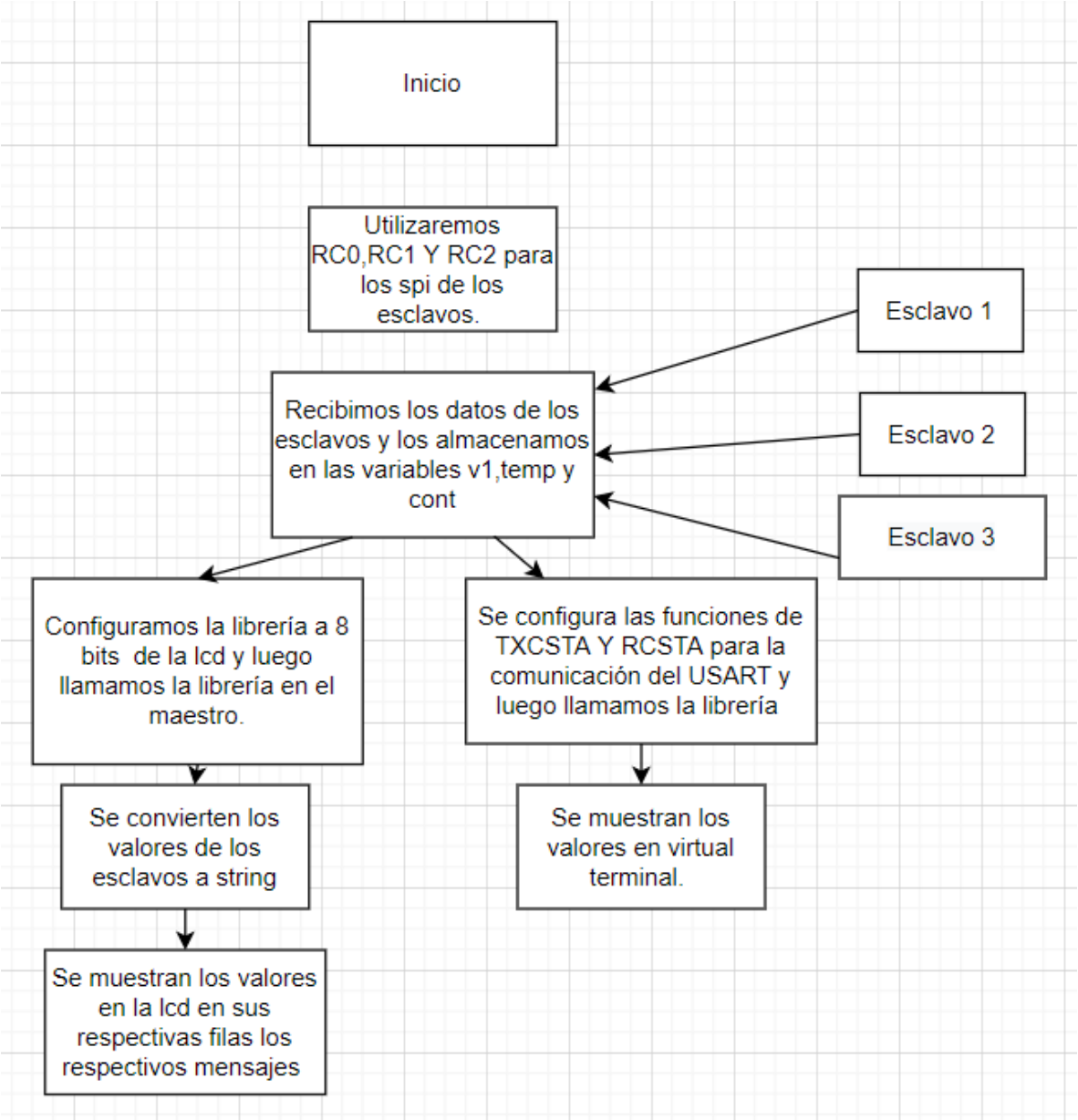
Esclavo contador:



Esclavo temperatura:


Inicio
Configuramos puerto A(sensor) y B(semáforo)
Entrada del del sensor RB2
Se toma el valor se guardara en la variable temp, y por ultimo la comaramos con los rangos dados en el pdf
temp < 25
Led verde 1, rojo y amarrillo 0
temp > 25 & temp < 36
Amarrillo 1, verde y rojo 0
temp > 36
rojo 1, verde y amarrillo 0

Todo unido:



Link de github:

https://github.com/Helder1121/Labsdigitaldos/tree/main/Proyecto

Progra comentada:

Maestro:

Librerias:

```
/*
 * File:   LCD.c
```

```
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 11:37 PM
 */

//Libreria de Pablo Mazariegos en clase de 4 bits modificada a unos de 8bits

#include <xc.h>
#include <stdint.h>
#include "LCD.h"
#define _XTAL_FREQ 8000000

//Funcion para indicar el caracter segun sea el tamaño del mismo.
void Puerto(uint8_t x){
        if(x & 1){D0 = 1;}else{D0 = 0;}
   if(x & 2){D1 = 1;}else{D1 = 0;}
   if(x & 4){D2 = 1;}else{D2 = 0;}
   if(x & 8){D3 = 1;}else{D3 = 0;}
   if(x & 16){D4 = 1;}else{D4 = 0;}
   if(x & 32){D5 = 1;}else{D5 = 0;}
   if(x & 64){D6 = 1;}else{D6 = 0;}
   if(x & 128){D7 = 1;}else{D7 = 0;}
}

//Funcion para imprimir el caracter
void LCD_CMD(char a){
   RS = 1;//Las direcciones a los caracteres
   Puerto(a);
   EN = 1;//Mandar el valor
   __delay_us(5);
   EN = 0;//Verificar si el valor de carac llego
```

```
    __delay_us(5);
    __delay_us(50);
}
//Funcion para mandar los datos a la LCD
void datosLCD(uint8_t x){
    RS = 0;//Modifica el contraste de la patalla
    Puerto(x);
    EN = 1;//Mandar el valor
    __delay_us(5);
    EN = 0;//Verificar si el valor de carac llego
    __delay_us(5);
    __delay_ms(2);
}
//Funcion para limpiar la LCD
void LCD_Limpia(void){
    datosLCD(0);
    datosLCD(1);
}
//Funcion para iniciar la LCD
//En base de la presentacion de clase.
void Lcd_Init(){
    __delay_ms(20);
    datosLCD (0x30);
    __delay_ms(5);
    datosLCD (0x30);
    __delay_us(100);
    datosLCD (0x30);
    __delay_us(100);
    datosLCD (0x38);
    __delay_us(60);
    datosLCD (0x08);
```

```
   __delay_us(60);
   datosLCD (0x01);
   __delay_ms(5);
   datosLCD (0x06);
   __delay_us(60);
   datosLCD (0x0C);
   __delay_us(60);
}
//Funcion para configurar el cursor
void Lcd_Set_Cursor(uint8_t x, uint8_t y){
            uint8_t a;
            if(x == 1){//Linea que se coloca arriba
              a = 0x80 + y;//direccion(hexadecimal) y posicion para colocarlo en la fila
                  //adecuada para ir leyendo adecuadamente
                        datosLCD(a);
   }
            else if(x == 2){//Linea que se coloca abajo
              a = 0xC0 + y;//direccion(hexadecimal) y posicion para colocarlo en la fila
                  //adecuada para ir leyendo adecuadamente
                        datosLCD(a);
   }
}
//Funcion para mandar un string
void Lcd_Write_String(char *a){
   //funcion para poder imprimir texto usando el puntero
   //para guardar la direccion del registro o valor de a
            int i;
            for(i=0;a[i]!='\0';i++)
              LCD_CMD(a[i]);
}
/*
```

```
 * File           : spi.c
 * Author         : Ligo George
 * Company        : electroSome
 * Project        : SPI Library for MPLAB XC8
 * Microcontroller : PIC 16F877A
 * Created on April 15, 2017, 5:59 PM
 */
//Extraido de https://electrosome.com/ indicado por Pablo Mazariegos
#include "SPI.h"

void spiInit(Spi_Type sType, Spi_Data_Sample sDataSample, Spi_Clock_Idle
sClockIdle, Spi_Transmit_Edge sTransmitEdge)
{
   TRISC5 = 0;
   if(sType & 0b00000100) //If Slave Mode
   {
      SSPSTAT = sTransmitEdge;
      TRISC3 = 1;
   }
   else            //If Master Mode
   {
      SSPSTAT = sDataSample | sTransmitEdge;
      TRISC3 = 0;
   }

   SSPCON = sType | sClockIdle;
}

static void spiReceiveWait()
{
   while ( !SSPSTATbits.BF ); // Wait for Data Receive complete
```

```
}

void spiWrite(char dat)  //Write data to SPI bus
{
   SSPBUF = dat;
}

unsigned spiDataReady() //Check whether the data is ready to read
{
   if(SSPSTATbits.BF)
      return 1;
   else
      return 0;
}

char spiRead() //REad the received data
{
   spiReceiveWait();       // wait until the all bits receive
   return(SSPBUF); // read the received data from the buffer
}
/*
 * File:   USART.c
 * Author: betov
 *
 * Created on 22 de febrero de 2021, 07:53 AM
 */


#include <xc.h>
#include <pic16f887.h>
#include "USART.h"
```

```
void _baudios(void){
   SPBRG = 12; //9600 baudios para 8MHZ
}
//Configuracion dada en el datasheet
void config_txsta(void){
   TXSTAbits.CSRC = 0;//Clock terminal
   TXSTAbits.TX9 = 0;//8 bits de transmicion
   TXSTAbits.TXEN = 1;//Transmicion habilitada
   TXSTAbits.SYNC = 0;//modo asincrono
   TXSTAbits.BRGH = 0;//low speed
   TXSTAbits.TRMT = 0;//Tsr full
   TXSTAbits.TX9D = 0;
}
//Configuracion dada en el datasheet
void config_rcsta(void){
   RCSTAbits.SPEN = 1;//Se habilita el puerto serial
   RCSTAbits.RX9 = 0;
   RCSTAbits.SREN = 0;
   RCSTAbits.CREN = 1;//Recibir habilitada
   RCREG = 0;
}
//Extraido de https://electrosome.com/uart-pic-microcontroller-mplab-xc8/
void Write_USART(uint8_t a){
   while(!TRMT);
   TXREG=a;
}
void Write_USART_String(char *a){
   uint8_t i;
   for(i=0;a[i]!='\0';i++){
      Write_USART(a[i]);
```

```
   }
}
uint8_t Read_USART(){
 while(!RCIF);
 return RCREG;
}
```

Main:

```
/*
 * File:   contador.c
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 01:36 PM
 */
//Basados en la implementación de comunicación SPI de Pablo
//*****************************************************************************
// 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 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)

//******************************************************************************
// Importación de librerías
//******************************************************************************
#include <xc.h>
#include <stdint.h>
#include <stdio.h>
#include "SPI.h"
#include "LCD.h"
#include "USART.h"

//******************************************************************************
// Variables
//******************************************************************************
#define _XTAL_FREQ 8000000
uint8_t cont = 0;
uint8_t ADC1 = 0;
uint8_t ADC2 = 0;
float v1,temp;
char data[20];//Variable mostrara los valos en la lcd
//******************************************************************************
//Portotipos de funciones
//******************************************************************************
void setup(void);
void contador(void);
void ADC_lectura(void);
float temperatura(void);
```

```
//*****************************************************************************
// Ciclo principal
//*****************************************************************************
void main(void){
   setup();

   _baudios();
   config_txsta();
   config_rcsta();
   Lcd_Init();
   LCD_Limpia();
   //*****************************************************************************
   // Loop principal
   //*****************************************************************************
   while(1){
      contador();
      ADC_lectura();
      //temperatura();
      LCD_Limpia();//Limpiamos la lcd
      Lcd_Set_Cursor(1,1);//Se mostrar en la primera fila de la lcd
      Lcd_Write_String("S1   CONT   S3");
      //Mensaje que se muestra en la terminal en la primera linea
//       v1 = ADC1*0.0196;
      temp = temperatura();
      sprintf(data, "%1.0f   %d   %3.0f" ,v1,cont,temp);

      Lcd_Set_Cursor(2,1);//Segunda fila
      Lcd_Write_String(data);//Mostrara el valor en la LCD

      Write_USART_String("S1   CONT   S3");
      //Mensaje que se muestra en la terminal en la segunda linea
```

```
        Write_USART(13);
        Write_USART(10);
        //Saltar lineas
        Write_USART_String(data);//Muestra en la terminal los valores
        Write_USART(13);
        Write_USART(10);
        //Saltar lineas
        __delay_ms(500);
    }
}

void ADC_lectura(void){
    PORTCbits.RC0 = 0;      //Slave Select
    __delay_ms(1);

    spiWrite(1);
    v1 = spiRead();

    __delay_ms(1);
    PORTCbits.RC0 = 1;      //Slave Deselect
    __delay_ms(1);
}

void contador(void){
    PORTCbits.RC1 = 0;     //Slave Select
    __delay_ms(1);

    spiWrite(1);
    cont = spiRead();

    __delay_ms(1);
```

```
    PORTCbits.RC1 = 1;      //Slave Deselect
    __delay_ms(1);
}

float temperatura(void){
    PORTCbits.RC2 = 0;      //Slave Select
    __delay_ms(1);

    spiWrite(1);
    temp = spiRead();

    __delay_ms(1);
    PORTCbits.RC2 = 1;       //Slave Deselect
    __delay_ms(1);
    return temp;
}

//*****************************************************************************
// Configuración
//*****************************************************************************
void setup(void){
    ANSEL = 0;
    ANSELH = 0;
    TRISB = 0;
    TRISE = 0;
    TRISD = 0;
    //Steo los puertos
    PORTE = 0;
    PORTD = 0;
    PORTB = 0;
```

```
    TRISC0 = 0;
    TRISC1 = 0;
    TRISC2 = 0;
    PORTCbits.RC0 = 1;
    PORTCbits.RC1 = 1;
    PORTCbits.RC2 = 1;
    PORTCbits.RC7 = 1;

    spiInit(SPI_MASTER_OSC_DIV4, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW,
        SPI_IDLE_2_ACTIVE);
}
```

Esclavo adc:

Librerías:

```
*
 * File:   ADC.c
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 02:32 PM
 */

#include <xc.h>
#include <stdint.h>
#include "ADC.h"
#define _XTAL_FREQ 8000000



void config_ADC(void){
    ADCON1 = 0b00000000;//Justificado a la izquierda
}
```

```
unsigned Canal_ADC(unsigned short x){ //Fosc/8,datasheet
   switch(x){
      //Canal analogico
      case 0:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 0;
         ADCON0bits.CHS0 = 0;//Canal00
         break;
      case 1:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 0;
         ADCON0bits.CHS0 = 1;//Canal1
         break;
      case 2:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 1;
         ADCON0bits.CHS0 = 0;//Canal2
         break;
      case 3:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 1;
         ADCON0bits.CHS0 = 1;//Canal3
         break;
      case 4:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 1;
```

```
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal4
    break;
case 5:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;//Canal5
    break;
case 6:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//Canal6
    break;
case 7:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal7
    break;
case 8:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal8
    break;
case 9:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
```

```
    ADCON0bits.CHS0 = 1;//Canal9
    break;
case 10:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//Canal10
    break;
case 11:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal11
    break;
case 12:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal12
    break;
case 13:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;//Canal13
    break;
case 14:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//CVref
```

```
            break;
        case 15:
            ADCON0bits.CHS3 = 1;
            ADCON0bits.CHS2 = 1;
            ADCON0bits.CHS1 = 1;
            ADCON0bits.CHS0 = 1;//Fixed Ref
            break;
        default:
            ADCON0bits.CHS3 = 0;
            ADCON0bits.CHS2 = 0;
            ADCON0bits.CHS1 = 0;
            ADCON0bits.CHS0 = 0;//Canal 0
            break;
    }
}
/*
 * File          : spi.c
 * Author        : Ligo George
 * Company       : electroSome
 * Project       : SPI Library for MPLAB XC8
 * Microcontroller : PIC 16F877A
 * Created on April 15, 2017, 5:59 PM
 */
//Extraido de https://electrosome.com/ indicado por Pablo Mazariegos
#include "SPI.h"

void spiInit(Spi_Type sType, Spi_Data_Sample sDataSample, Spi_Clock_Idle
sClockIdle, Spi_Transmit_Edge sTransmitEdge)
{
    TRISC5 = 0;
    if(sType & 0b00000100) //If Slave Mode
```

```
    {
        SSPSTAT = sTransmitEdge;
        TRISC3 = 1;
    }
    else            //If Master Mode
    {
        SSPSTAT = sDataSample | sTransmitEdge;
        TRISC3 = 0;
    }

    SSPCON = sType | sClockIdle;
}

static void spiReceiveWait()
{
    while ( !SSPSTATbits.BF ); // Wait for Data Receive complete
}

void spiWrite(char dat)  //Write data to SPI bus
{
    SSPBUF = dat;
}

unsigned spiDataReady() //Check whether the data is ready to read
{
    if(SSPSTATbits.BF)
        return 1;
    else
        return 0;
}
```

```
char spiRead() //REad the received data
{
   spiReceiveWait();       // wait until the all bits receive
   return(SSPBUF); // read the received data from the buffer
}
```

Main:

```
/*
 * File:   SPadc.c
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 04:48 PM
 */

//******************************************************************************
// 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 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)

//*****************************************************************************
// Importación de librerías
//*****************************************************************************
#include <xc.h>
#include <stdint.h>
#include "SPI.h"
#include "ADC.h"

//*****************************************************************************
// Variables
//*****************************************************************************
#define _XTAL_FREQ 8000000
uint8_t ADC = 0;
uint8_t volt, volt2;//variable para los voltajes en los pots
//*****************************************************************************
//Portotipos de funciones
//*****************************************************************************
void setup(void);
uint8_t adc_11(void);
uint8_t adc_21(void);
void Enviar_1(void);
void Enviar_2(void);
//*****************************************************************************
// COdigo de interrupcion
//*****************************************************************************
void __interrupt() isr(void){
   if(SSPIF == 1){
```

```
        spiWrite(ADC);
        SSPIF = 0;
    }
}
//*****************************************************************************
// Ciclo principal
//*****************************************************************************
void main(void){
    setup();
    config_ADC();
    //*************************************************************************
    // Loop principal
    //*************************************************************************
    while(1){
        //ADC_1();
        adc_21();
        ADC = adc_21();
        PORTD = ADC;
    }
}
//*****************************************************************************
// Configuración
//*****************************************************************************
void setup(void){
    ANSEL = 1;
    ANSELH = 0;
    TRISA = 1;
    TRISB = 0;
    TRISD = 0;

    //Seteo el puerto
```

```
    PORTA = 0;
    PORTB = 0;
    PORTD = 0;

    INTCONbits.GIE = 1;         // Habilitamos interrupciones
    INTCONbits.PEIE = 1;        // Habilitamos interrupciones PEIE
    PIR1bits.SSPIF = 0;         // Borramos bandera interrupción MSSP
    PIE1bits.SSPIE = 1;         // Habilitamos interrupción MSSP
    TRISAbits.TRISA5 = 1;       // Slave Select

    spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW,
            SPI_IDLE_2_ACTIVE);

}

//*****************************************************************************
// Funciones
//*****************************************************************************
uint8_t adc_11(void){
    Canal_ADC(0);//canal 0
    //Configuracion bits ADCON0
    ADCON0bits.ADCS0 = 1;//Clock ADC conversion
    ADCON0bits.ADCS1 = 0;
    ADCON0bits.ADON = 1;//Habilitamos el ADC
    __delay_ms(0.25);//Para la conversion
    ADCON0bits.GO = 1;//Inicia la conversion
    while (ADCON0bits.GO == 1){
        return  ADRESH;//Conversion de 0V-5V
    }
}
uint8_t adc_21(void){
```

```
    Canal_ADC(0);//Canal 0
    //Configuracion bits ADCON0
    ADCON0bits.ADCS0 = 1;//Clock ADC conversion
    ADCON0bits.ADCS1 = 0;
    ADCON0bits.ADON = 1;//Habilitamos el ADC
    __delay_ms(0.25);//Para la conversion
    ADCON0bits.GO = 1;//Inicia la conversion
    while (ADCON0bits.GO == 1){
       return  ADRESH; //Conversion
    }
}
void Enviar_1(void){//Envio de datos
    TXREG = volt;
    while (TXSTAbits.TRMT == 1){//Retorna y envia el voltaje a ADC1
       return;
    }
}
void Enviar_2(void){//Envio de datos
    TXREG = volt2;
    while (TXSTAbits.TRMT == 1){//Retorna y envia el voltaje a ADC2
       return;
    }
}
```

Esclavo contador:

Librerías:

```
/*
 * File          : spi.c
 * Author        : Ligo George
 * Company       : electroSome
 * Project       : SPI Library for MPLAB XC8
```

```
 * Microcontroller : PIC 16F877A
 * Created on April 15, 2017, 5:59 PM
 */
//Extraido de https://electrosome.com/ indicado por Pablo Mazariegos
#include "SPI.h"

void spiInit(Spi_Type sType, Spi_Data_Sample sDataSample, Spi_Clock_Idle
sClockIdle, Spi_Transmit_Edge sTransmitEdge)
{
   TRISC5 = 0;
   if(sType & 0b00000100) //If Slave Mode
   {
      SSPSTAT = sTransmitEdge;
      TRISC3 = 1;
   }
   else           //If Master Mode
   {
      SSPSTAT = sDataSample | sTransmitEdge;
      TRISC3 = 0;
   }

   SSPCON = sType | sClockIdle;
}

static void spiReceiveWait()
{
   while ( !SSPSTATbits.BF ); // Wait for Data Receive complete
}

void spiWrite(char dat)  //Write data to SPI bus
{
```

```
   SSPBUF = dat;
}

unsigned spiDataReady() //Check whether the data is ready to read
{
   if(SSPSTATbits.BF)
      return 1;
   else
      return 0;
}

char spiRead() //REad the received data
{
   spiReceiveWait();       // wait until the all bits receive
   return(SSPBUF); // read the received data from the buffer
}
```

Main:

```
/*
 * File:   contador.c
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 01:36 PM
 */
//Basados en la implementación de comunicación SPI de Pablo
//******************************************************************************
// 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 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)


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

// Importación de librerías

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

#include <xc.h>

#include <stdint.h>

#include "SPI.h"


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

// Variables

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

#define _XTAL_FREQ 8000000

uint8_t conta = 0;


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

//Portotipos de funciones
```

```
//*****************************************************************************
void setup();

//*****************************************************************************
// COdigo de interrupcion
//*****************************************************************************
void __interrupt() isr(void){
   if(SSPIF == 1){
      spiWrite(conta);
      SSPIF = 0;
      //Mandarlo al SPI
   }
}
//*****************************************************************************
// Ciclo principal
//*****************************************************************************
void main(void){
   setup();
   //*****************************************************************************
   // Loop principal
   //*****************************************************************************
   while(1){
      if (PORTBbits.RB0 == 0){
         __delay_ms(100);
         if (PORTBbits.RB0 == 1){
            conta ++;
            PORTD = conta;
         }
      }
      if (PORTBbits.RB1 == 0){
         __delay_ms(100);
```

```
        if (PORTBbits.RB1 == 1){
            conta --;
            PORTD = conta;
        }
      }
   }
}
//*****************************************************************************
// Configuración
//*****************************************************************************
void setup(void){
   ANSEL = 0;
   ANSELH = 0;


   TRISB = 3;
   TRISD = 0;

   //Steo los puertos
   PORTB = 0;
   PORTD = 0;

   INTCONbits.GIE = 1;        // Habilitamos interrupciones
   INTCONbits.PEIE = 1;       // Habilitamos interrupciones PEIE
   PIR1bits.SSPIF = 0;        // Borramos bandera interrupción MSSP
   PIE1bits.SSPIE = 1;        // Habilitamos interrupción MSSP
   TRISAbits.TRISA5 = 1;      // Slave Select

   spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW,
        SPI_IDLE_2_ACTIVE);
```

```
}
```

Esclavo temperatura:

Librerías:

```
/*
 * File          : spi.c
 * Author        : Ligo George
 * Company       : electroSome
 * Project       : SPI Library for MPLAB XC8
 * Microcontroller : PIC 16F877A
 * Created on April 15, 2017, 5:59 PM
 */
//Extraido de https://electrosome.com/ indicado por Pablo Mazariegos
#include "SPI.h"

void spiInit(Spi_Type sType, Spi_Data_Sample sDataSample, Spi_Clock_Idle
sClockIdle, Spi_Transmit_Edge sTransmitEdge)
{
   TRISC5 = 0;
   if(sType & 0b00000100) //If Slave Mode
   {
      SSPSTAT = sTransmitEdge;
      TRISC3 = 1;
   }
   else           //If Master Mode
   {
      SSPSTAT = sDataSample | sTransmitEdge;
      TRISC3 = 0;
   }

   SSPCON = sType | sClockIdle;
```

```
}

static void spiReceiveWait()
{
   while ( !SSPSTATbits.BF ); // Wait for Data Receive complete
}

void spiWrite(char dat)  //Write data to SPI bus
{
   SSPBUF = dat;
}

unsigned spiDataReady() //Check whether the data is ready to read
{
   if(SSPSTATbits.BF)
      return 1;
   else
      return 0;
}

char spiRead() //REad the received data
{
   spiReceiveWait();       // wait until the all bits receive
   return(SSPBUF); // read the received data from the buffer
}
/*
 * File:   ADC.c
 * Author: Helder Ovalle
 *
 * Created on 21 de febrero de 2021, 02:32 PM
 */
```

```
#include <xc.h>
#include <stdint.h>
#include "ADC.h"
#define _XTAL_FREQ 8000000

void config_ADC(void){
   ADCON1 = 0b00000000;//Justificado a la izquierda
   //ADCON1bits.VCFG0 = 1;
}

unsigned Canal_ADC(unsigned short x){ //Fosc/8,datasheet
   switch(x){
      //Canal analogico
      case 0:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 0;
         ADCON0bits.CHS0 = 0;//Canal00
         break;
      case 1:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 0;
         ADCON0bits.CHS0 = 1;//Canal1
         break;
      case 2:
         ADCON0bits.CHS3 = 0;
         ADCON0bits.CHS2 = 0;
         ADCON0bits.CHS1 = 1;
         ADCON0bits.CHS0 = 0;//Canal2
```

```
    break;
case 3:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal3
    break;
case 4:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal4
    break;
case 5:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;//Canal5
    break;
case 6:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//Canal6
    break;
case 7:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal7
    break;
```

```
case 8:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 0;//Canal8
   break;
case 9:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 1;//Canal9
   break;
case 10:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 1;
   ADCON0bits.CHS0 = 0;//Canal10
   break;
case 11:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 1;
   ADCON0bits.CHS0 = 1;//Canal11
   break;
case 12:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 1;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 0;//Canal12
   break;
case 13:
```

```
            ADCON0bits.CHS3 = 1;
            ADCON0bits.CHS2 = 1;
            ADCON0bits.CHS1 = 0;
            ADCON0bits.CHS0 = 1;//Canal13
            break;
        case 14:
            ADCON0bits.CHS3 = 1;
            ADCON0bits.CHS2 = 1;
            ADCON0bits.CHS1 = 1;
            ADCON0bits.CHS0 = 0;//CVref
            break;
        case 15:
            ADCON0bits.CHS3 = 1;
            ADCON0bits.CHS2 = 1;
            ADCON0bits.CHS1 = 1;
            ADCON0bits.CHS0 = 1;//Fixed Ref
            break;
        default:
            ADCON0bits.CHS3 = 0;
            ADCON0bits.CHS2 = 0;
            ADCON0bits.CHS1 = 0;
            ADCON0bits.CHS0 = 0;//Canal 0
            break;
    }
}
```

Main:

```
/*
 * File:   Temperatura.c
 * Author: Helder Ovalle
 *
```

```
 * Created on 21 de febrero de 2021, 06:20 PM
 */

//*****************************************************************************
// 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 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)

//*****************************************************************************
// Importación de librerías
//*****************************************************************************
#include <xc.h>
#include <stdint.h>
```

```
#include "ADC.h"
#include "SPI.h"

//*****************************************************************************
// Variables
//*****************************************************************************
#define _XTAL_FREQ 8000000
uint8_t ADC = 0;
float temp;
uint8_t volt, volt2;//variable para los voltajes en los pots
//*****************************************************************************
//Portotipos de funciones
//*****************************************************************************
void setup(void);
void semaf(uint8_t temp);
uint8_t adc_11(void);
//uint8_t adc_21(void);
//void Enviar_1(void);
//void Enviar_2(void);
//*****************************************************************************
// COdigo de interrupcion
//*****************************************************************************
void __interrupt() isr(void){
   if(SSPIF == 1){
      spiWrite(temp);
      SSPIF = 0;
      //Mandarlo al SPI
   }
}
//*****************************************************************************
// Ciclo principal
```

```
//*****************************************************************************
void main(void){
   setup();
   //*****************************************************************************
   // Loop principal
   //*****************************************************************************
   while(1){
      adc_11();
      //ADC_2();
      ADC = adc_11();
      temp = (1.95*ADC);//COnversion para los grados
      semaf(temp);
   }
}
void semaf(uint8_t temp){
   if (temp < 25){//Verde <25
      PORTD = 1;}
   else if (temp > 25 && temp < 36){//Amariillo para el rango de 25-36
      PORTD = 2;}
   else if (temp > 36){//Rojo para >36
      PORTD = 4;}
}
//*****************************************************************************
// Configuración
//*****************************************************************************
void setup(void){
   ANSEL = 0b00001000;
   ANSELH = 0;

   //TRISB = 0;
   TRISD = 0;
```

```
    //Steo el puerto
    PORTD = 0;
    PORTB = 0;

    INTCONbits.GIE = 1;         // Habilitamos interrupciones
    INTCONbits.PEIE = 1;        // Habilitamos interrupciones PEIE
    PIR1bits.SSPIF = 0;         // Borramos bandera interrupción MSSP
    PIE1bits.SSPIE = 1;         // Habilitamos interrupción MSSP
    TRISAbits.TRISA5 = 1;       // Slave Select

    spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW,
            SPI_IDLE_2_ACTIVE);

}

//*****************************************************************************
// Funciones
//*****************************************************************************
uint8_t adc_11(void){
    Canal_ADC(8);//canal 8
    //Configuracion bits ADCON0
    ADCON0bits.ADCS0 = 1;//Clock ADC conversion
    ADCON0bits.ADCS1 = 0;
    ADCON0bits.ADON = 1;//Habilitamos el ADC
    __delay_ms(0.25);//Para la conversion
    ADCON0bits.GO = 1;//Inicia la conversion
    while (ADCON0bits.GO == 1){
        //Conversion
    }
    return ADRESH;
}
```