Universidad del Valle de Guatemala

Digital 2 Sección 10

Rodrigo García 19085

MiniProyecto 1

Link a Repositorio: https://github.com/gar19085/Digital2-Gar19085.git

Link de video: https://youtu.be/UPqZuZn1LSE

Esquemático:

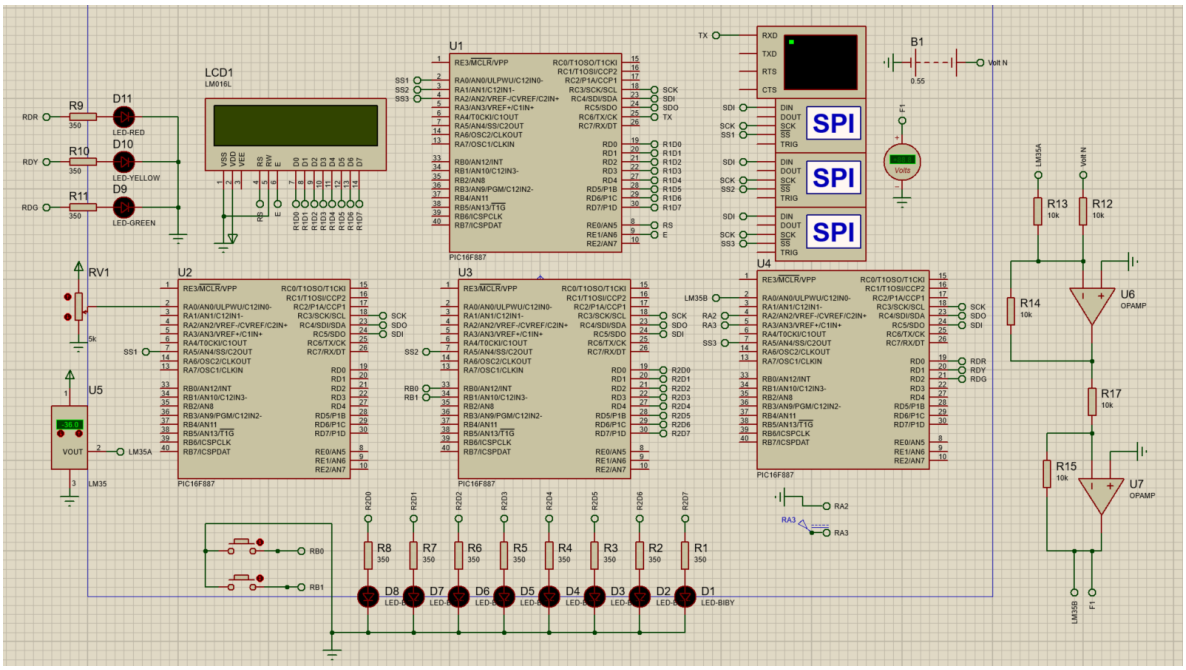
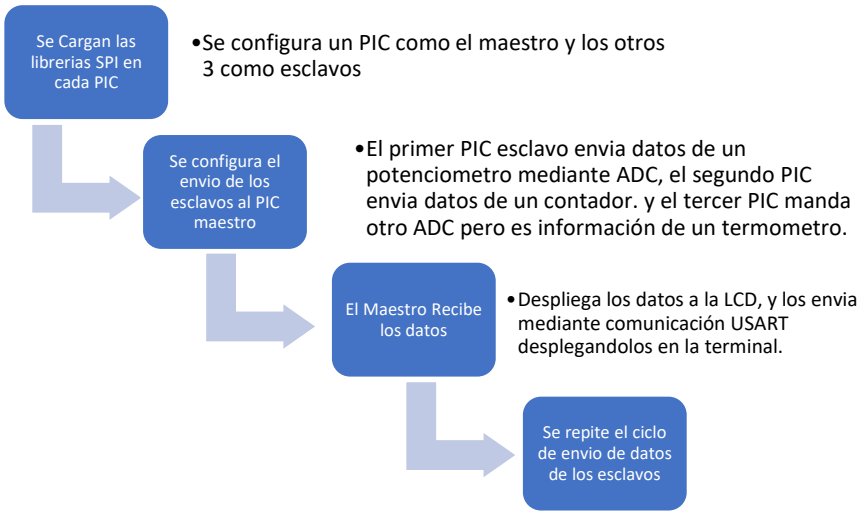
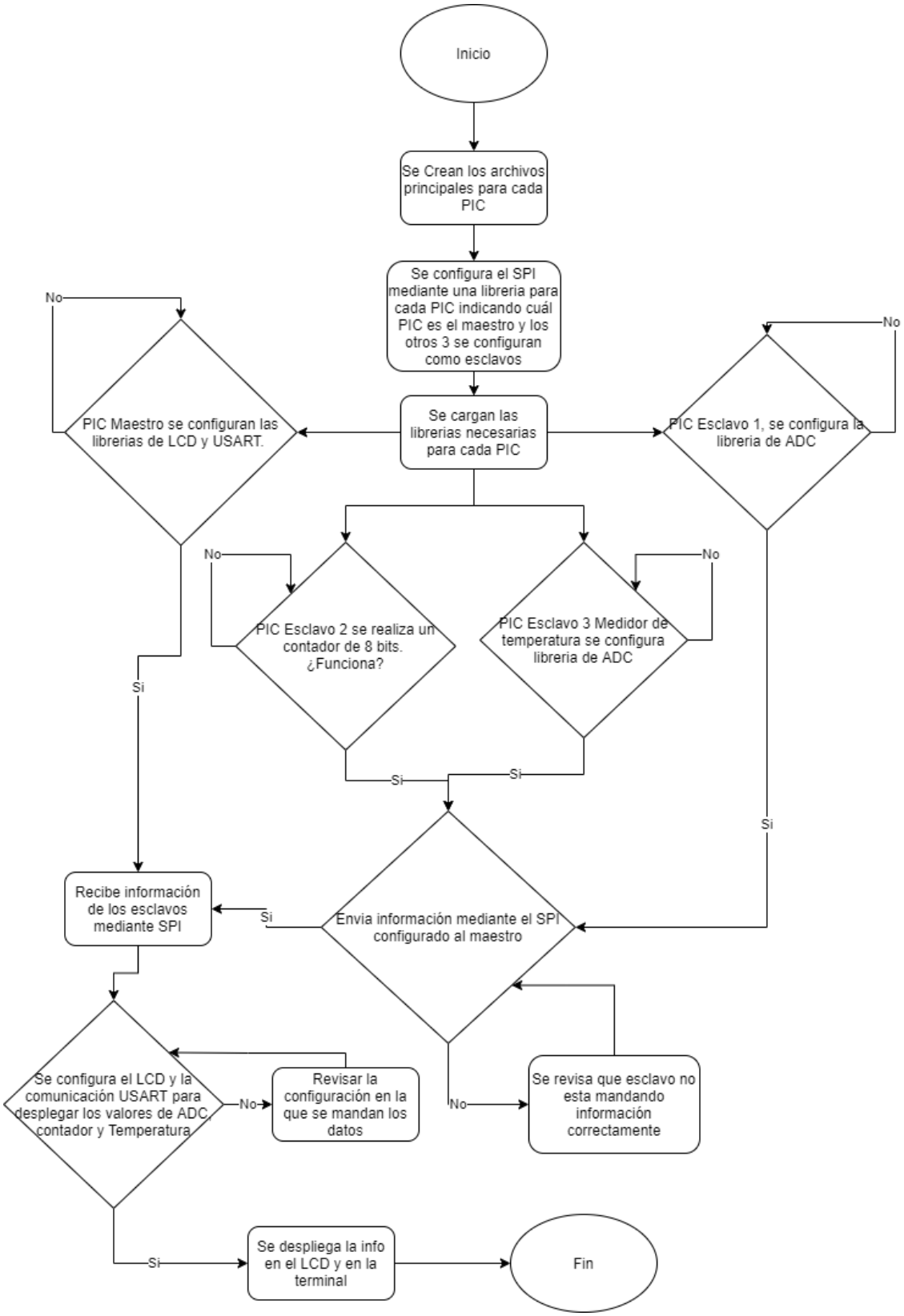


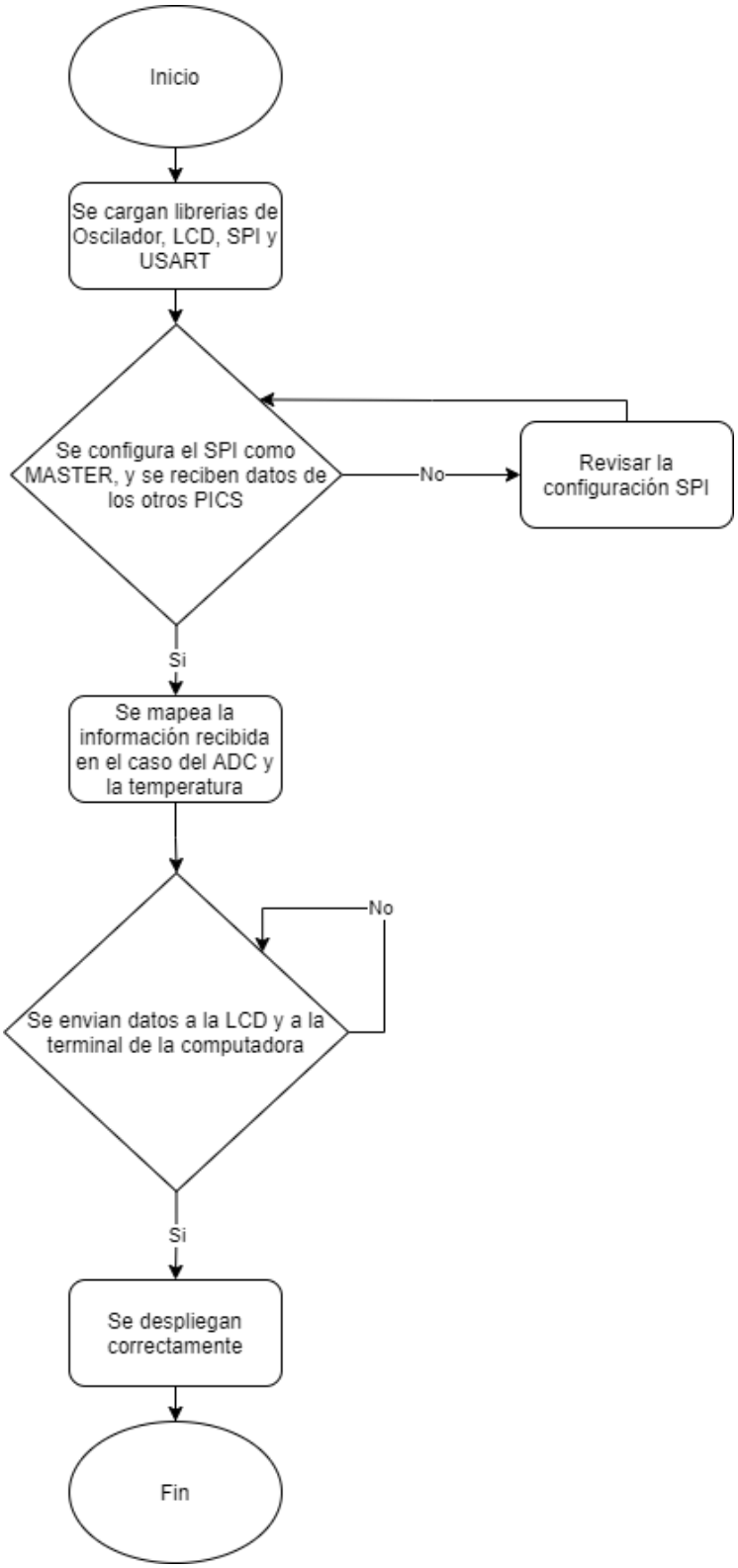
Diagrama de Flujo:



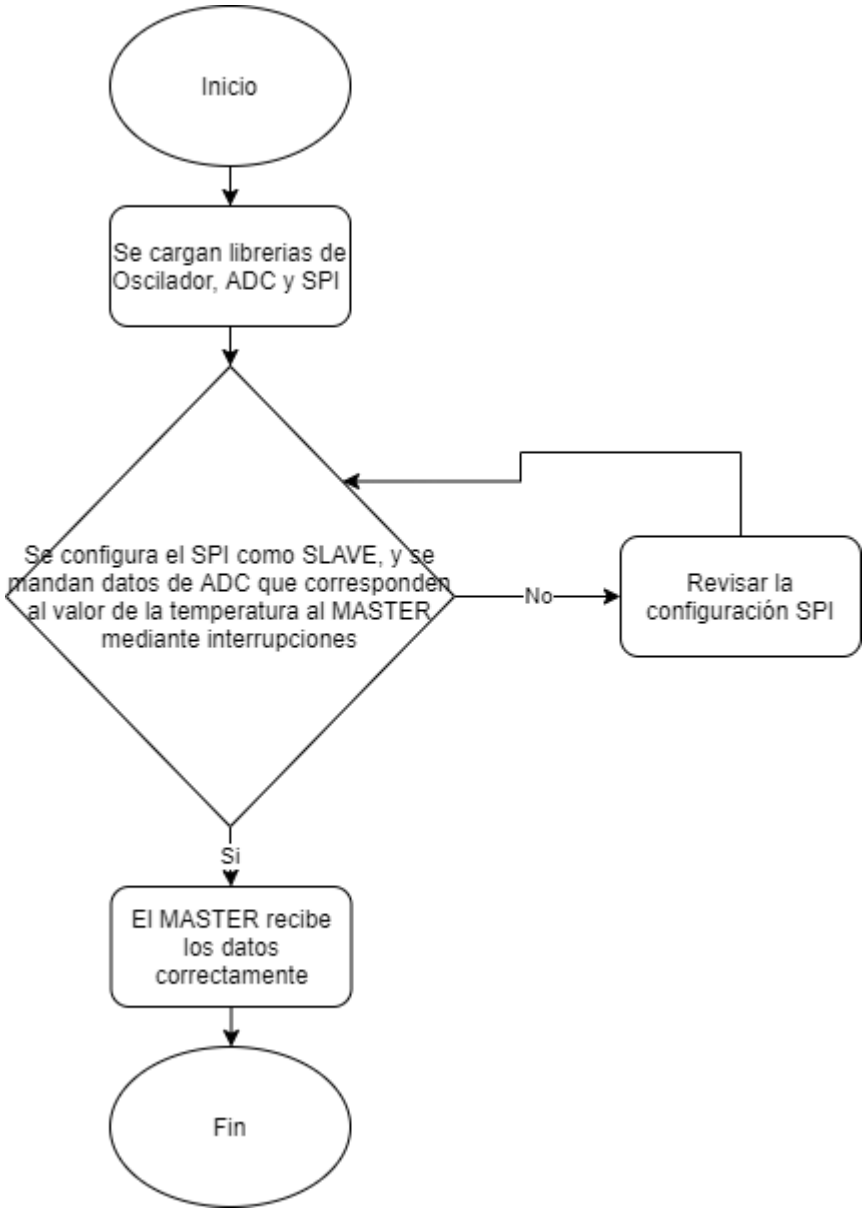
Seudocódigo General:



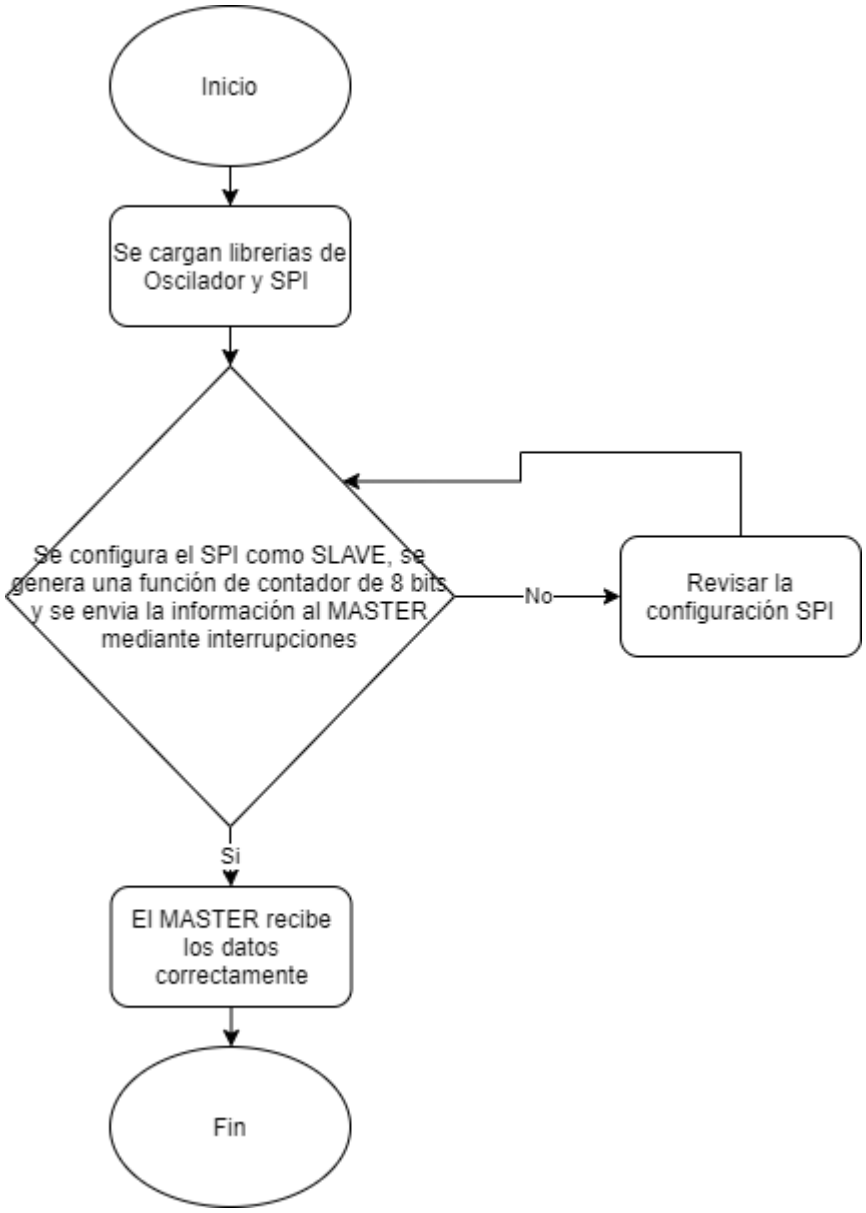
Seudocódigo PIC Maestro:



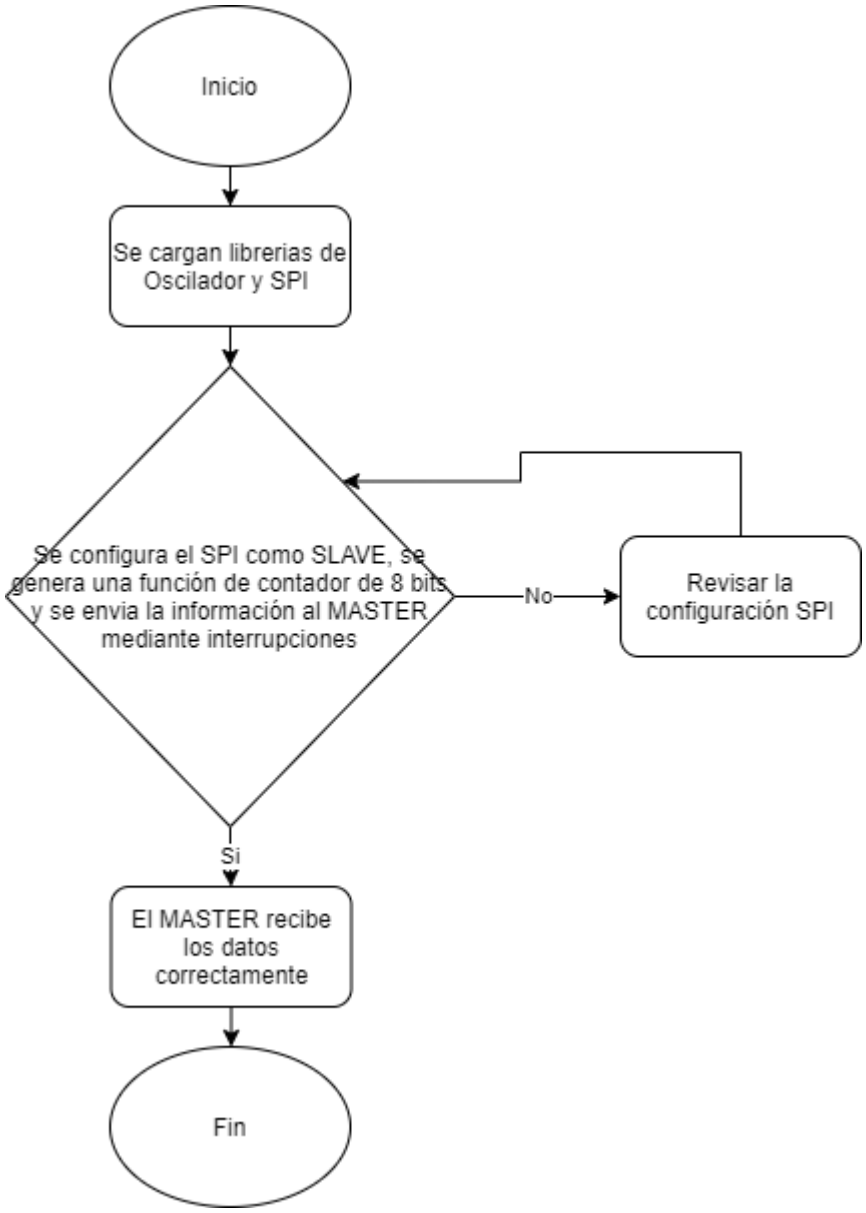
Seudocódigo PIC Esclavo ADC:



Seudocódigo PIC Esclavo Contador:



Seudocódigo PIC Esclavo Temperatura:



Código:

MAESTRO

```
/*
 * File:   MASTER.c
 * Author: RODRIGO GARCIA
 *
 * Created on 11 de febrero de 2021, 06:08 PM
 */

// PIC16F887 Configuration Bit Settings
```

```
// 'C' source line config statements

// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled a
nd 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/MCL
R 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 cod
e protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disab
led)
#pragma config IESO = OFF       // Internal External Switchover bit (Interna
l/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 p
in 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 bi
ts (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.


#include <xc.h>
#include <stdint.h>
#include "LCD.h"
#include "Osciladador.h"
#include "USART.h"
#include "SPI.h"

#define _XTAL_FREQ 4000000
```

```
uint8_t CONT1;
uint8_t CONT2;
uint8_t CONT3;
uint8_t ADC1;
uint8_t ADC12;
uint8_t ADC13;
uint8_t TEM1;
uint8_t TEM2;
uint8_t TEM3;
uint8_t CLN;
uint8_t valorADC;
uint8_t CONTADOR;
uint8_t Temp;
uint8_t Tem1;
uint8_t Tem2;
uint8_t SND;
uint8_t TGLTX;

void Setup(void);
void SlaveADC(void);
void SlaveCont(void);
void SlaveTemp(void);
void SEND(void);
void INFOCONT(void);
const char* STRINGCONT(char C1, char C2, char C3);
void INFOADC(void);
const char* STRINGADC(char C1, char C2, char C3);
void INFOTEMPP(void);
void INFOTEMPN(void);
void CONTROLTEMP(void);
const char* STRINGTEMPP(char C1, char C2, char C3);
const char* STRINGTEMPN(char C1, char C2, char C3);

void __interrupt() isr(void){
    if(PIR1bits.TXIF == 1){ //HABILITAO INTERRUPECIONES DEL TX
        SEND();
        SND++;
        PIE1bits.TXIE = 0;
        PIR1bits.TXIF = 0;
    }
}

void main(void) {
    initOsc(8);    //LLAMO CONFIGURACIÓN PARA EL OSCILADOR INTERNO
    Conf_TXR();    //CONFIG DE COMUNICACIÓN USART
```

```
    Conf_RXT();
    Setup();
    spiInit(SPI_MASTER_OSC_DIV4, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW,
 SPI_IDLE_2_ACTIVE); //CONFIG DE SPI COMO MASTER
    LCD_init();
    LCD_Cmd(0x8A);
    LCD_Goto(1,1);
    LCD_Print("VOLT");
    LCD_Goto(7,1);
    LCD_Print("CONT");
    LCD_Goto(13,1);
    LCD_Print("TEMP");
    while(1){
        TGLTX++; //CONTADOR PARA DELAY DEL TX
        if (TGLTX > 2){ //DELAY DEL TX PARA QUE MANDE INFORMACIÓN CORRECTAME
NTE
            PIE1bits.TXIE = 1;
            TGLTX = 0;  }
        SlaveADC(); //LAMO RUTINA DE QUE SIRVE PARA RECIBIR DATOS DEL SLAVEA
DC
        INFOADC();//LLAMO FUNCION DE MAPEO
        SlaveCont();//LAMO RUTINA DE QUE SIRVE PARA RECIBIR DATOS DEL SLAVEC
ONT
        INFOCONT();//FUNCION DE MAPEO
        SlaveTemp();//LAMO RUTINA DE QUE SIRVE PARA RECIBIR DATOS DEL SLAVET
EMP
        LCD_Goto(2,2);
        LCD_Print(STRINGADC(ADC1, ADC12, ADC13));
        LCD_Goto(8,2);
        LCD_Print(STRINGCONT(CONT1, CONT2, CONT3));
        LCD_Goto(13,2);
        CONTROLTEMP();     //FUNCION PARA DETERMINAR SIGNO POSITIVO Y NEGATI
VO
    }
}

void Setup(){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;
```

```
    TRISA = 0;
    TRISB = 0;
    TRISC = 0b00010000;
    TRISD = 0;
    TRISE = 0;
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS
    PIR1bits.TXIF = 0;
    PIE1bits.TXIE = 1;
}

void SlaveADC(void){//FUNCIÓN QUE SIRVE PARA RECIBIR LOS DATOS DEL SPI ESCLA
VO
    PORTAbits.RA0 = 0;
    __delay_ms(1);

    spiWrite(CLN);
    valorADC = spiRead();

    __delay_ms(1);
    PORTAbits.RA0 = 1;
}

void SlaveCont(void){//MISMA RUTINA QUE EN SlaveADC
    PORTAbits.RA1 = 0;
    __delay_ms(1);

    spiWrite(CLN);
    CONTADOR = spiRead();

    __delay_ms(1);
    PORTAbits.RA1 = 1;
}

void SlaveTemp(void){//MISMA RUTINA QUE EN SlaveADC
    PORTAbits.RA2 = 0;
    __delay_ms(1);

    spiWrite(CLN);
    Temp = spiRead();

    __delay_ms(1);
    PORTAbits.RA2 = 1;
```

```
}


void INFOCONT(void){ //MAPEO DEL CONTADOR
    CONT1 = CONTADOR/100;
    CONT2 = ((CONTADOR-(CONT1*100))/10);
    CONT3 = (CONTADOR-(CONT1*100))-(CONT2*10);
    CONT1 = CONT1+0x30;
    CONT2 = CONT2+0x30;
    CONT3 = CONT3+0x30;
}

const char* STRINGCONT(char C1, char C2, char C3){ //SE GENERA UNA MATRIZ PA
RA MANDAR LA INFORMACIÓN A LA LCD
    char TEMP[3];
    TEMP[0] = C1;
    TEMP[1] = C2;
    TEMP[2] = C3;
    return TEMP;
}

void INFOADC(void){//MAPEO DEL ADC
    ADC1 = valorADC/51;
    ADC12 = (valorADC-(ADC1*51))/10;
    ADC13 = (valorADC-(ADC1*51))-(ADC12*10);
    ADC1 = ADC1+0x30;
    ADC12 = ADC12+0x30;
    ADC13 = ADC13+0x30;
}

const char* STRINGADC(char C1, char C2, char C3){
    char TEMP[4];
    TEMP[0] = C1;
    TEMP[1] = 0x2E;
    TEMP[2] = C2;
    TEMP[3] = C3;
    return TEMP;
}

void INFOTEMPP(void){//MAPEO VALORES POSITIVOS TEMPERATURA
    Tem1 = ((Temp-68)*150)/187;
    TEM1 = Tem1/100;
    TEM2 = (Tem1-(TEM1*100))/10;
    TEM3 = (Tem1-(TEM1*100)-(TEM2*10));
    TEM1 = TEM1+0x30;
```

```
    TEM2 = TEM2+0x30;
    TEM3 = TEM3+0x30;
}
void INFOTEMPN(void){//MAPEO VALORES NEGATIVOS TEMPERATURA
    Tem2 = ((Temp*(-55))/68)+55;
    TEM1 = Tem2/100;
    TEM2 = (Tem2-(TEM1*100))/10;
    TEM3 = (Tem2-(TEM1*100)-(TEM2*10));
    TEM1 = TEM1+0x30;
    TEM2 = TEM2+0x30;
    TEM3 = TEM3+0x30;
}
void CONTROLTEMP(void){//FUNCION QUE SIRVE PARA AGREGAR EL SIGNO + O - DEPEN
DIENDO DEL VALOR DE LA TEMPERATURA
    if(Temp >= 68){
        INFOTEMPP();
        LCD_Print(STRINGTEMPP(TEM1, TEM2, TEM3));
    }
    else if(Temp < 68){
        INFOTEMPN();
        LCD_Print(STRINGTEMPN(TEM1, TEM2, TEM3));
    }
}

const char* STRINGTEMPP(char C1, char C2, char C3){
    char TEMP[4];
    TEMP[0] = 0x2B;
    TEMP[1] = C1;
    TEMP[2] = C2;
    TEMP[3] = C3;
    return TEMP;
}

const char* STRINGTEMPN(char C1, char C2, char C3){
    char TEMP[5];
    TEMP[0] = 0x2D;
    TEMP[1] = C1;
    TEMP[2] = C2;
    TEMP[3] = C3;
    return TEMP;
}

void SEND(void){//RUTINA QUE FUNCIONA PARA MANDAR LA INFORMACIÓN A LA TERMIN
AL DE LA COMPUTADORA
    switch(SND){
```

```
        case 0:
            TXREG = 0x28;
            break;
        case 1:
            TXREG = ADC1;
            break;
        case 2:
            TXREG = 0x2E;
            break;
        case 3:
            TXREG = ADC12;
            break;
        case 4:
            TXREG = ADC13;
            break;
        case 5:
            TXREG = 0x29;
            break;
        case 6:
            TXREG = 0x2C;
            break;
        case 7:
            TXREG = 0x28;
            break;
        case 8:
            TXREG = CONT1;
            break;
        case 9:
            TXREG = CONT2;
            break;
        case 10:
            TXREG = CONT3;
            break;
        case 11:
            TXREG = 0x29;
            break;
        case 12:
            TXREG = 0x2C;
            break;
        case 13:
            TXREG = 0x28;
            break;
        case 14:
            if(Temp >= 68){
             TXREG = 0x2B;
```

```
            }else if (Temp < 68){
             TXREG = 0x2D;
            }
            break;
        case 15:
            TXREG = TEM1;
            break;
        case 16:
            TXREG = TEM2;
            break;
        case 17:
            TXREG = TEM3;
            break;
        case 18:
            TXREG = 0x29;
            break;
        case 19:
            TXREG = 0x0D;
            SND = 0;
            break;
    }
}
/*
 * File:
 * Author: RODRIGO GARCÍA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _LCD_H_
#define _LCD_H_

#include <xc.h> // include processor files - each processor file is guarded.

#include <stdint.h>

void LCD_init(void);
void LCD_Goto(uint8_t col, uint8_t row);
void LCDPutC(char LCD_Char);
void LCD_Print(char *LCD_Str);
void LCD_Cmd(uint8_t Command);
void LCD_PORT(char a);
//void LCD_WRITENIBBLE(uint8_t n);
```

```
#endif  /* _LCD_H_ */

/*
 * File:   LCD.c
 * Author: rodri
 * REFERENCIA: https://simple-circuit.com/pic-microcontroller-mplab-xc8-lcd/
 * https://electrosome.com/lcd-pic-mplab-xc8/
 * Created on 4 de febrero de 2021, 10:48 AM
 */

#include <xc.h>
#include <stdint.h>
#include "LCD.h"

#define LCD_FIRST_ROW           0x80
#define LCD_SECOND_ROW          0xC0
#define LCD_THIRD_ROW           0x94
#define LCD_FOURTH_ROW          0xD4
#define LCD_CLEAR               0x01
#define LCD_RETURN_HOME         0x02
#define LCD_ENTRY_MODE_SET      0x04
#define LCD_CURSOR_OFF          0x0C
#define LCD_UNDERLINE_ON        0x0E
#define LCD_BLINK_CURSOR_ON     0x0F
#define LCD_MOVE_CURSOR_LEFT    0x10
#define LCD_MOVE_CURSOR_RIGHT   0x14
#define LCD_TURN_ON             0x0C
#define LCD_TURN_OFF            0x08
#define LCD_SHIFT_LEFT          0x18
#define LCD_SHIFT_RIGHT         0x1E


char a;

//LCD module connections
#define LCD_RS   RE0
#define LCD_EN   RE1
#define LCD_RW   RE2
#define LCD_D0   RD0
#define LCD_D1   RD1
#define LCD_D2   RD2
#define LCD_D3   RD3
```

```
#define LCD_D4   RD4
#define LCD_D5   RD5
#define LCD_D6   RD6
#define LCD_D7   RD7
#define LCD_RS_DIR  TRISE0
#define LCD_EN_DIR  TRISE1
#define LCD_RW_DIR  TRISE2
#define LCD_D0_DIR  TRISD0
#define LCD_D1_DIR  TRISD1
#define LCD_D2_DIR  TRISD2
#define LCD_D3_DIR  TRISD3
#define LCD_D4_DIR  TRISD4
#define LCD_D5_DIR  TRISD5
#define LCD_D6_DIR  TRISD6
#define LCD_D7_DIR  TRISD7
//End LCD module connections
#define _XTAL_FREQ 8000000


void LCD_init(void);
void LCD_Goto(uint8_t col, uint8_t row);
void LCDPutC(char LCD_Char);
void LCD_Print(char *LCD_Str);
void LCD_Cmd(uint8_t Command);
void LCD_PORT(char a);

void LCD_PORT(char a){
    PORTD = a;
}

void LCD_init(void){
  LCD_Cmd(0x38);
  LCD_Cmd(0x0c);
  LCD_Cmd(0x06);
  LCD_Cmd(0x80);
}

void LCD_Goto(uint8_t col, uint8_t row){
switch(row)
  {
    case 2:
      LCD_Cmd(LCD_SECOND_ROW + col - 1);
      break;
    case 3:
      LCD_Cmd(LCD_THIRD_ROW  + col - 1);
```

```
      break;
    case 4:
      LCD_Cmd(LCD_FOURTH_ROW + col - 1);
    break;
    default:      // case 1:
      LCD_Cmd(LCD_FIRST_ROW  + col - 1);
  }
}

void LCDPutC(char LCD_Char){
  LCD_RS = 1;
  LCD_PORT(LCD_Char);
  LCD_EN = 1;
  __delay_us(40);
  LCD_EN = 0;
  LCD_RS = 0;
}

void LCD_Print(char*LCD_Str){
    int i;
  for(i=0;LCD_Str[i]!='\0';i++)
     LCDPutC(LCD_Str[i]);
}

void LCD_Cmd(uint8_t Command){
  LCD_PORT(Command);
  LCD_RS = 0;
  LCD_EN = 1;
  __delay_ms(5);
  LCD_EN = 0;
}


/*
 * File:
 * Author: RODRIGO GARCÍA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef __OSCILADOR_H_
#define __OSCILADOR_H_
```

```
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)

#include <xc.h> // include processor files - each processor file is guarded.

#include<stdint.h>
//*****************************************************************
// Prototipo de la función para inicializar Oscilador Interno
// Parametros: Opción de frecuencia a utilizar ver pág. 62 Manual
//*****************************************************************
void initOsc(uint8_t frec);

#endif  /* XC_HEADER_TEMPLATE_H */
/*
 * File:   Oscilador.c
 * Author: RODRIGO GARCIA
 * Author Original: Pedro Mazariegos
 * Repositorio de GitHUB de el: https://github.com/pdmazariegos-
uvg/ie3027/tree/master/Ejemplos
 *
 * Created on 1 de febrero de 2021, 10:52 PM
 */

#include <stdint.h>
#include <pic16f887.h>
#include "Oscilador.h"
//*****************************************************************
// Función para inicializar Oscilador Interno
// Parametros: Opción de frecuencia a utilizar ver pág. 62 Manual
//*****************************************************************
void initOsc(uint8_t frec){
    switch(frec){
        case 0:                             // 31 KHz
            OSCCONbits.IRCF0 = 0;
            OSCCONbits.IRCF1 = 0;
            OSCCONbits.IRCF2 = 0;
            break;
        case 1:                             // 125 KHz
            OSCCONbits.IRCF0 = 1;
            OSCCONbits.IRCF1 = 0;
            OSCCONbits.IRCF2 = 0;
            break;

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

```
             * Acá se debería de programar para todas las demás
             * frecuencias, colocando un caso por cada una de
             * las opciones que tiene el microcontrolador
             ***********************************************/

        case 7:                             // 8 MHz
            OSCCONbits.IRCF0 = 1;
            OSCCONbits.IRCF1 = 1;
            OSCCONbits.IRCF2 = 1;
            break;
        case 8:                            // 4 MHz
            OSCCONbits.IRCF0 = 0;
            OSCCONbits.IRCF1 = 1;
            OSCCONbits.IRCF2 = 1;
            break;
    }

    OSCCONbits.SCS = 1;      // Se utilizará el reloj interno para el sistem
a
}LA LIBRERIA DEL OSCILADOR ES LA MISMA PARA TODOS LOS PICS
/*
 * File:
 * Author: RODRIGO GARCIA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _USART_
#define _USART_

#include <xc.h> // include processor files - each processor file is guarded.

#include <stdint.h>

void Conf_TXR(void);
void Conf_RXT(void);
void TRANSMITIR(char *VAL);

#endif  /* XC_HEADER_TEMPLATE_H */



#include <xc.h>
```

```c
#include <stdint.h>
#include "USART.h"

void Conf_TXR(void);
void Conf_RXT(void);

void Conf_TXR(void){
    TXSTAbits.SYNC = 0;
    TXSTAbits.TXEN = 1;
    TXSTAbits.BRGH = 1;
    TXSTAbits.TX9 = 0;
    BAUDCTLbits.BRG16 = 0;
    SPBRG = 25;
}

void Conf_RXT(void){
    RCSTAbits.SPEN = 1;
    RCSTAbits.CREN = 1;
    RCSTAbits.FERR = 0;
    RCSTAbits.OERR = 0;
    RCSTAbits.RX9 = 0;
    PIE1bits.RCIE = 1;
}

void TRANSMITIR(char *VAL){
    TXREG = VAL;
    TXREG = 0x2E;
}

LA LIBRERIA SPI ES LA MISMA EN TODOS LOS PICS
/*
 * File            : spi.h
 * Author          : Ligo George
 * Company         : electroSome
 * Project         : SPI Library for MPLAB XC8
 * Microcontroller : PIC 16F877A
 * Created on April 15, 2017, 5:59 PM
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef __SPI_H
#define __SPI_H
```

```
#include <xc.h> // include processor files - each processor file is guarded.

#include <pic16f887.h>
typedef enum
{
    SPI_MASTER_OSC_DIV4  = 0b00100000,
    SPI_MASTER_OSC_DIV16 = 0b00100001,
    SPI_MASTER_OSC_DIV64 = 0b00100010,
    SPI_MASTER_TMR2      = 0b00100011,
    SPI_SLAVE_SS_EN      = 0b00100100,
    SPI_SLAVE_SS_DIS     = 0b00100101
}Spi_Type;

typedef enum
{
    SPI_DATA_SAMPLE_MIDDLE   = 0b00000000,
    SPI_DATA_SAMPLE_END      = 0b10000000
}Spi_Data_Sample;

typedef enum
{
    SPI_CLOCK_IDLE_HIGH  = 0b00010000,
    SPI_CLOCK_IDLE_LOW   = 0b00000000
}Spi_Clock_Idle;

typedef enum
{
    SPI_IDLE_2_ACTIVE    = 0b00000000,
    SPI_ACTIVE_2_IDLE    = 0b01000000
}Spi_Transmit_Edge;


void spiInit(Spi_Type, Spi_Data_Sample, Spi_Clock_Idle, Spi_Transmit_Edge);
void spiWrite(char);
unsigned spiDataReady();
char spiRead();

#endif  /* SPI_H */
/*
 * File:   SPI.c
 * Author: rodri
 *
 * Created on 14 de febrero de 2021, 11:22 PM
 */
```

```
/*
 * 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
 */

#include "SPI.h"

void spiInit(Spi_Type sType, Spi_Data_Sample sDataSample, Spi_Clock_Idle sCl
ockIdle, 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
}
```

ESCLAVO ADC

```
/*
 * File:   SlaveADC.c
 * Author: rodri
 *
 * Created on 12 de febrero de 2021, 11:40 AM
 */

// PIC16F887 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled a
nd 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/MCL
R 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 cod
e protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disab
led)
#pragma config IESO = OFF       // Internal External Switchover bit (Interna
l/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 p
in 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 bi
ts (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.



#include <xc.h>
#include <stdint.h>
#include "Oscilador.h"
#include "ADC.h"
#include "SPI.h"

uint8_t valorADC = 0;
uint8_t ADCGO = 0;


void Setup(void);
void ADCG(void);

void __interrupt() isr(void){
    if(PIR1bits.SSPIF == 1){
        spiWrite(valorADC);
        PIR1bits.SSPIF = 0;
    }
    if(PIR1bits.ADIF==1){ //CONFIGURACIÓN PARA LAS INTERRUPCIONES DEL ADC
        PIR1bits.ADIF = 0;
        valorADC = ADRESH; //LE INDICO A MI VARIABLE valorADC EL VALRO DE AD
RESH QUE CORRESPONDE
                           //AL VALOR DEL ADC
    }
    if (INTCONbits.TMR0IF == 1){   //CONFIGURACIÓN PARA UTILIZAR LA NTERRUPC
IÓN DE TMR0
        TMR0=236;
        INTCONbits.TMR0IF = 0;
        ADCGO++; //SE VA SUMANDO CADA VEZ ESTA VARIABLE YA QUE ES UN DELAY D
E ADQUISICIÓN EXTRA
    }
}


void main(void) {
```

```
    initOsc(8);
    Setup();
    initADC(1,0);
    while(1){
        ADCG();
        PORTD = valorADC;
    }
}

void Setup(void){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0b00000001;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;

    TRISA = 0b00100001;
    TRISB = 0;
    TRISC = 0b00001000;
    TRISD = 0;
    TRISE = 0;
    OPTION_REG = 0b10000011;
    spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW, SPI
_IDLE_2_ACTIVE);
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
    INTCONbits.T0IE = 1; //HABILITO LAS INTERRUPCIONES DEL TMR0
    INTCONbits.T0IF = 0;
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS
    PIE1bits.ADIE = 1; //HABILILTO LAS INTERRUPCIONES DEL ADC
    PIR1bits.ADIF = 0;
    PIR1bits.SSPIF = 0;
    PIE1bits.SSPIE = 1;

}

void ADCG(void){//GENERO UN DELAY DE ADQUISICIÓN EL CUAL FUNCIONA DE LA SIGU
IENTE MANERA
    if(ADCGO > 20){ //CUANDO ADCGO SEA MÁS GRANDE QUE 20 YA QUE ESTE VA A ES
TAR SUMANDOSE CONSTANTEMENTE EN LA INTERRUPCIÓN
        ADCGO = 0; //SE SETEA EN 0 NUEVAMENTE
```

```
        ADCON0bits.GO_nDONE = 1; //SE HABILITA EL GO DEL ADC PARA QUE LA CON
FIGURACIÓN ADC FUNCIONE CORRECTAMENTE
    }                           //DE ESTA MANERA PUEDE VOLVER A COMENZAR NU
EVAMENTE SIN PROBLEMAS
}
/*
 * File:
 * Author: RODRIGO GARCIA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef ADC_H_
#define ADC_H_

#include <xc.h> // include processor files - each processor file is guarded.

#include<stdint.h>

void initADC(uint8_t frec, uint8_t can);

#endif  /* ADC_H_*/
/*
 * File:   Contador.c
 * Author: RODRIGO GARCIA
 *
 * Created on 1 de febrero de 2021, 09:52 PM
 */


/*
 INCLUIR LIBRERIAS CREADAS
 */

#include <xc.h>
#include <stdint.h>
#include <pic16f887.h>
#include "ADC.h"


void initADC(uint8_t frec, uint8_t can){
    switch(frec){
        case 0:
```

```
        ADCON0bits.ADCS = 0b00;  //FOSC/2
        break;
        case 1:
        ADCON0bits.ADCS = 0b01; //FOSC/8
        break;
        case 2:
        ADCON0bits.ADCS = 0b10; //FOSC/32
        break;
        case 3:
        ADCON0bits.ADCS = 0b11; //FRc (500kHz)
        break;
        default:
        ADCON0bits.ADCS = 0b00;
        break;
    }
    switch(can){
        case 0:
        ADCON0bits.CHS = 0b0000; //CANAL AN0
        break;
        case 1:
        ADCON0bits.CHS = 0b0001; //CANAL AN1
        break;
        case 2:
        ADCON0bits.CHS = 0b0010; //CANAL AN2
        break;
        case 3:
        ADCON0bits.CHS = 0b0011; //CANAL AN3
        break;
        case 4:
        ADCON0bits.CHS = 0b0100; //CANAL AN4
        break;
        case 5:
        ADCON0bits.CHS = 0b0101; //CANAL AN5
        break;
        case 6:
        ADCON0bits.CHS = 0b0110; //CANAL AN6
        break;
        case 7:
        ADCON0bits.CHS = 0b0111; //CANAL AN7
        break;
        case 8:
        ADCON0bits.CHS = 0b1000; //CANAL AN8
        break;
        case 9:
        ADCON0bits.CHS = 0b1001; //CANAL AN9
```

```
        break;
        case 10:
         ADCON0bits.CHS = 0b1010; //CANAL AN10
        break;
        case 11:
         ADCON0bits.CHS = 0b1011; //CANAL AN11
        break;
        case 12:
         ADCON0bits.CHS = 0b1100; //CANAL AN12
        break;
        case 13:
         ADCON0bits.CHS = 0b1101; //CANAL AN13
        break;
        case 14:
         ADCON0bits.CHS = 0b1110; //CVref
        break;
        case 15:
         ADCON0bits.CHS = 0b1111; //Fixed Ref
        break;
        default:
         ADCON0bits.CHS = 0b0000;
        break;
    }
    ADCON0bits.GO = 0; //CONVERSIÓN STATUS BIT EN 0
    ADCON0bits.ADON = 1; //ENABLE BIT DEL ADC EN 1
    ADCON1=1;
}
```

ESCLAVO CONTADOR

```
/*
 * File:   SlaveCont.c
 * Author: rodri
 *
 * Created on 12 de febrero de 2021, 11:55 AM
 */


// PIC16F887 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
```

```
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled a
nd 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/MCL
R 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 cod
e protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disab
led)
#pragma config IESO = OFF       // Internal External Switchover bit (Interna
l/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 p
in 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 bi
ts (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.



#include <xc.h>
#include <stdint.h>
#include "Oscilador.h"
#include "SPI.h"

#define SUM RB0
#define RES RB1
char FLAG1;
char FLAG2;

void Setup(void);
```

```
void __interrupt() isr(void){
    if(PIR1bits.SSPIF == 1){
        spiWrite(PORTD);
        PIR1bits.SSPIF = 0;
    }
    if(INTCONbits.RBIF == 1){ //CONFIGURACIÓN DE INTERRUPCIÓN EN EL PUERTOB
                if(SUM == 1){ //SI MI PUSH BUTTON DE SUMA ESTA EN 1
                    FLAG1=1; //SE ACTIVA MI BANDERA
                }
                if(FLAG1 == 1 && SUM == 0){ //SI MI FLAG1 = 1 Y MI PUSH BUTT
ON EN 0
                    FLAG1=0; //SE APAGA EL FLAG1
                    PORTD++; //SE LE SUMA 1 AL PUERTO D EN DONDE SE ENCENTRA
 EL CONTADOR
                }
                if(RES==1){ //MISMO PROCEDIMIENTO SOLO QUE ESTE RESTA VALORE
S AL CONTADOR
                    FLAG2=1;
                }
                if(FLAG2==1 && RES==0){
                    FLAG2=0;
                    PORTD--;
                }
        INTCONbits.RBIF = 0;
    }
}




void main(void) {
    initOsc(8);
    Setup();
}

void Setup(void){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;
```

```
    TRISA = 0b00100000;
    TRISB = 0b00000011;
    TRISC = 0b00001000;
    TRISD = 0;
    TRISE = 0;
    OPTION_REG = 0b00000011;
    spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW, SPI
_IDLE_2_ACTIVE);
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
    INTCONbits.T0IE = 1; //HABILITO LAS INTERRUPCIONES DEL TMR0
    INTCONbits.T0IF = 0;
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS

    IOCBbits.IOCB0 = 1;//HABILITO LOS INTERRUPTS ON CHANGE PARA LOS PUSH
    IOCBbits.IOCB1 = 1;
    INTCONbits.RBIE = 1; //HABILILTO LAS INTERRUPCIONES DEL PUERTO B
    INTCONbits.RBIF = 0;
    PIR1bits.SSPIF = 0;
    PIE1bits.SSPIE = 1;
}
```

ESCLAVO TEMPERATURA

```
/*
 * File:   SlaveTemp.c
 * Author: rodri
 *
 * Created on 12 de febrero de 2021, 11:55 AM
 */

// PIC16F887 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled a
nd 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/MCL
R 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 cod
e protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disab
led)
#pragma config IESO = OFF       // Internal External Switchover bit (Interna
l/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 p
in 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 bi
ts (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.



#include <xc.h>
#include <stdint.h>
#include "Oscilador.h"
#include "ADC.h"
#include "SPI.h"

uint8_t TEMP = 0;
uint8_t ADCGO;

void Setup(void);
void ADCG(void);

void __interrupt() isr(void){
    if(PIR1bits.SSPIF == 1){
        spiWrite(TEMP);
        PIR1bits.SSPIF = 0;
    }
    if(PIR1bits.ADIF==1){ //CONFIGURACIÓN PARA LAS INTERRUPCIONES DEL ADC
        PIR1bits.ADIF = 0;
        TEMP = ADRESH; //LE INDICO A MI VARIABLE valorADC EL VALRO DE ADRESH
 QUE CORRESPONDE
```

```
                            //AL VALOR DEL ADC

    }
    if (INTCONbits.TMR0IF == 1){   //CONFIGURACIÓN PARA UTILIZAR LA NTERRUPC
IÓN DE TMR0
        TMR0=236;
        INTCONbits.TMR0IF = 0;
        ADCGO++; //SE VA SUMANDO CADA VEZ ESTA VARIABLE YA QUE ES UN DELAY D
E ADQUISICIÓN EXTRA
    }
}

void main(void) {
    initOsc(8);
    Setup();
    initADC(1,0);
    while(1){
        ADCG();
        PORTB = TEMP;
    }
}

void Setup(void){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0b00000001;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;

    TRISA = 0b00100001;
    TRISB = 0;
    TRISC = 0b00001000;
    TRISD = 0;
    TRISE = 0;
    OPTION_REG = 0b10000011;
    spiInit(SPI_SLAVE_SS_EN, SPI_DATA_SAMPLE_MIDDLE, SPI_CLOCK_IDLE_LOW, SPI
_IDLE_2_ACTIVE);
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
    INTCONbits.T0IE = 1; //HABILITO LAS INTERRUPCIONES DEL TMR0
    INTCONbits.T0IF = 0;
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS
```

```
    PIE1bits.ADIE = 1; //HABILILTO LAS INTERRUPCIONES DEL ADC
    PIR1bits.ADIF = 0;
    PIR1bits.SSPIF = 0;
    PIE1bits.SSPIE = 1;
}


void ADCG(void){//GENERO UN DELAY DE ADQUISICIÓN EL CUAL FUNCIONA DE LA SIGU
IENTE MANERA
    if(ADCGO > 20){ //CUANDO ADCGO SEA MÁS GRANDE QUE 20 YA QUE ESTE VA A ES
TAR SUMANDOSE CONSTANTEMENTE EN LA INTERRUPCIÓN
        ADCGO = 0; //SE SETEA EN 0 NUEVAMENTE
        ADCON0bits.GO_nDONE = 1; //SE HABILITA EL GO DEL ADC PARA QUE LA CON
FIGURACIÓN ADC FUNCIONE CORRECTAMENTE
    }                           //DE ESTA MANERA PUEDE VOLVER A COMENZAR NU
EVAMENTE SIN PROBLEMAS
}
/*
 * File:
 * Author: RODRIGO GARCIA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef ADC_H_
#define ADC_H_

#include <xc.h> // include processor files - each processor file is guarded.

#include<stdint.h>

void initADC(uint8_t frec, uint8_t can);

#endif  /* ADC_H_*/
/*
 * File:   ADC.c
 * Author: RODRIGO GARCIA
 *
 * Created on 11 de febrero de 2021, 05:13 PM
 */

/*
 INCLUIR LIBRERIAS CREADAS
```

```
 */

#include <xc.h>
#include <stdint.h>
#include "ADC.h"


void initADC(uint8_t frec, uint8_t can){
    switch(frec){
        case 0:
         ADCON0bits.ADCS = 0b00;  //FOSC/2
        break;
        case 1:
         ADCON0bits.ADCS = 0b01; //FOSC/8
        break;
        case 2:
         ADCON0bits.ADCS = 0b10; //FOSC/32
        break;
        case 3:
         ADCON0bits.ADCS = 0b11; //FRc (500kHz)
        break;
        default:
         ADCON0bits.ADCS = 0b00;
        break;
    }
    switch(can){
        case 0:
         ADCON0bits.CHS = 0b0000; //CANAL AN0
        break;
        case 1:
         ADCON0bits.CHS = 0b0001; //CANAL AN1
        break;
        case 2:
         ADCON0bits.CHS = 0b0010; //CANAL AN2
        break;
        case 3:
         ADCON0bits.CHS = 0b0011; //CANAL AN3
        break;
        case 4:
         ADCON0bits.CHS = 0b0100; //CANAL AN4
        break;
        case 5:
         ADCON0bits.CHS = 0b0101; //CANAL AN5
        break;
        case 6:
```

```
         ADCON0bits.CHS = 0b0110; //CANAL AN6
        break;
        case 7:
         ADCON0bits.CHS = 0b0111; //CANAL AN7
        break;
        case 8:
         ADCON0bits.CHS = 0b1000; //CANAL AN8
        break;
        case 9:
         ADCON0bits.CHS = 0b1001; //CANAL AN9
        break;
        case 10:
         ADCON0bits.CHS = 0b1010; //CANAL AN10
        break;
        case 11:
         ADCON0bits.CHS = 0b1011; //CANAL AN11
        break;
        case 12:
         ADCON0bits.CHS = 0b1100; //CANAL AN12
        break;
        case 13:
         ADCON0bits.CHS = 0b1101; //CANAL AN13
        break;
        case 14:
         ADCON0bits.CHS = 0b1110; //CVref
        break;
        case 15:
         ADCON0bits.CHS = 0b1111; //Fixed Ref
        break;
        default:
         ADCON0bits.CHS = 0b0000;
        break;
    }
    ADCON0bits.GO = 0; //CONVERSIÓN STATUS BIT EN 0
    ADCON0bits.ADON = 1; //ENABLE BIT DEL ADC EN 1
    ADCON1=1;
    ADCON1bits.VCFG0 = 1;
    ADCON1bits.VCFG1 = 1;
}
```