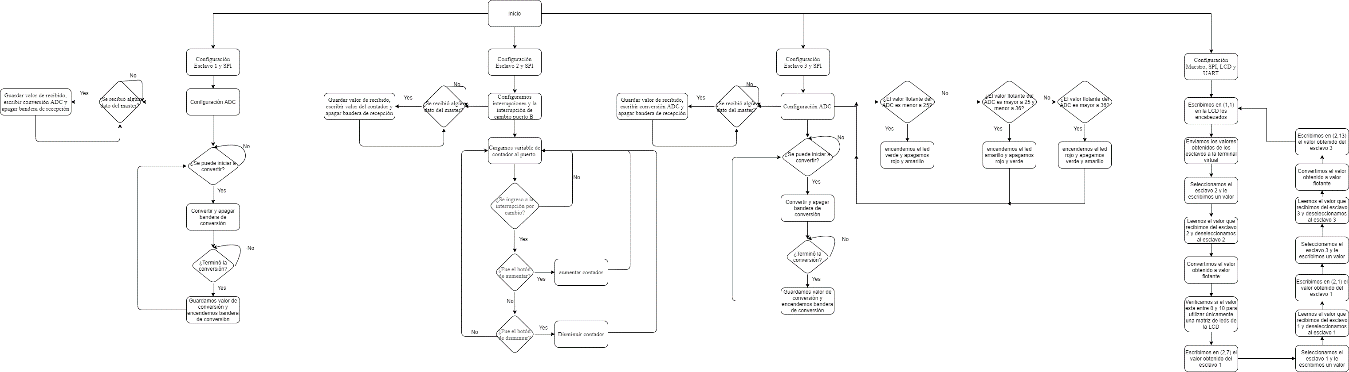
Universidad del Valle de Guatemala Cristopher René Sagastume 18640  
Electrónica Digital 2  
Sección 30 4/02/2021

**Diagrama General**



\*\* se adjunta también un PDF con el diagrama para mejor visualización

**Pseudocódigo Slave1**

1. Configuración PIC esclavo 1
2. Configuración comunicación SPI
3. Configuración ADC
4. ¿Se puede iniciar la conversión?
   1. No: retornar al paso 4)
   2. Si: convertir y apagar bandera de conversión
   3. ¿Terminó la conversión?
      * 1. No: retornar al paso c)
        2. Si: Guardamos el valor de la conversión, encendemos la bandera de conversión y regresamos al paso 4)
5. ¿Se recibió algún dato del máster?
   1. No: regresamos al paso 5)
   2. Si: Guardamos el valor recibido, escribimos el valor del ADC, apagamos la bandera de recepción y retornamos al paso 5)

**\*\* pasos 4) y 5) se ejecutan en paralelo por las interrupciones**

**Pseudocódigo Slave2**

1. Configuración PIC esclavo 2
2. Configuración comunicación SPI
3. Configuramos interrupciones generales y del puerto B
4. Cargamos la variable del contador al puerto.
5. ¿Se ingreso a la interrupción por cambio?
   1. No: Retornamos al paso 4)
   2. Si: ¿Fue el botón de aumentar?
      1. Si: aumentamos contador y regresamos al paso 4)
      2. No: ¿Fue el botón de disminuir?
         1. Si: disminuimos contador y regresamos al paso 4)
         2. No: regresamos al paso 4).
6. ¿Se ingreso a la interrupción por recepción SPI?
   1. No: regresamos al paso 6)
   2. Si: Guardamos el valor recibido, escribimos el valor del contador, apagamos la bandera de recepción y regresamos al paso 6).

**\*\*Pasos 4) y 6) se ejecutan en paralelo**

**Pseudocódigo Slave3**

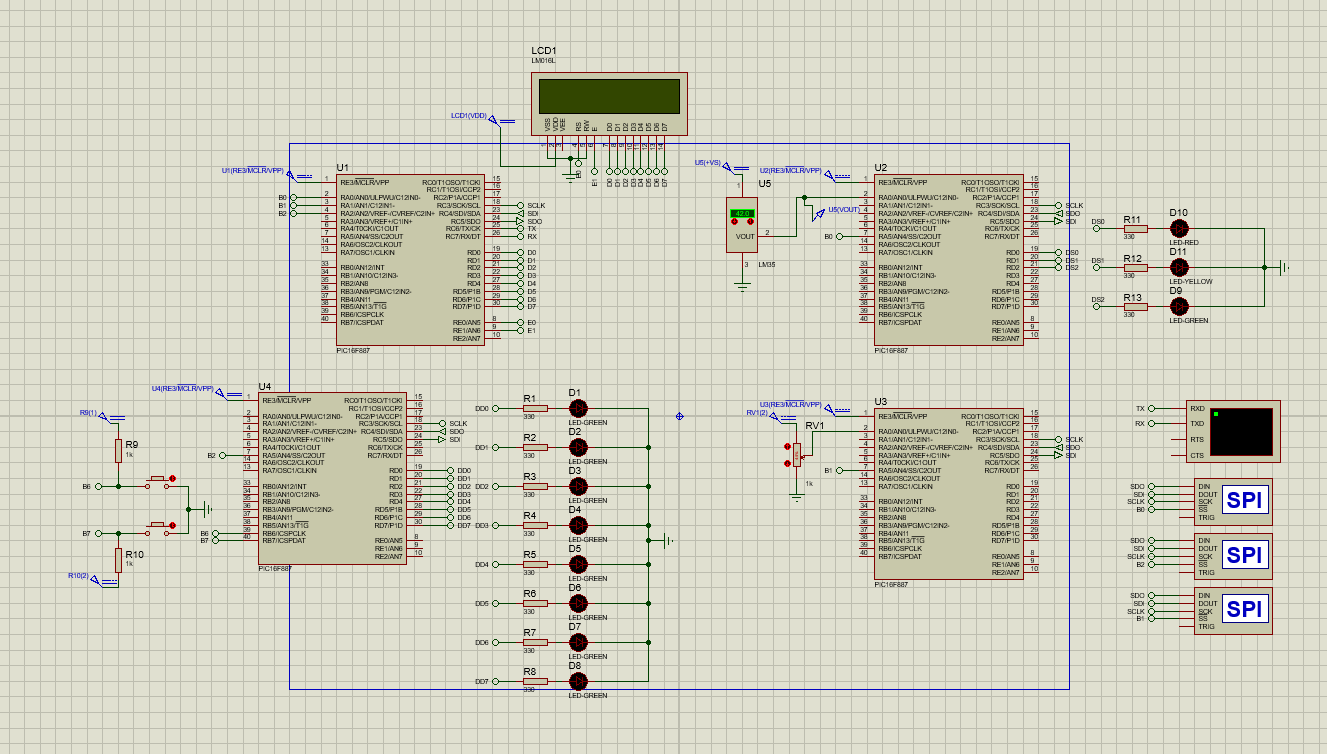
1. Configuración PIC esclavo 3
2. Configuración comunicación SPI
3. Configuración ADC
4. Convertimos el valor obtenido a flotante
5. ¿Se puede iniciar la conversión?
   1. No: retornar al paso 5)
   2. Si: convertir y apagar bandera de conversión
   3. ¿Terminó la conversión?
      * 1. No: retornar al paso c)
        2. Si: Guardamos el valor de la conversión, encendemos la bandera de conversión y regresamos al paso 4)
6. ¿Se recibió algún dato del máster?
   1. No: regresamos al paso 6)
   2. Si: Guardamos el valor recibido, escribimos el valor del ADC, apagamos la bandera de recepción y retornamos al paso 6)
7. ¿El valor flotante del ADC es menor a 25?
   1. Si: encendemos el led verde, apagamos el rojo, amarillo y regresamos al paso 7)
   2. No: ¿El valor flotante del ADC es mayor a 25 y menor a 36?
      1. Si: encendemos el led amarillo, apagamos rojo, verde y regresamos al paso 7)
      2. No: ¿El valor flotante del ADC es mayor a 36?
         1. Si: encendemos el led rojo, apagamos amarillo, verde y regresamos al paso 7)
         2. No: retornamos al paso 7)

**\*\*pasos 5), 6) y 7) se ejecutan en paralelo**

**Pseudocódigo Master**

1. Configuramos Master, SPI, LCD y UART
2. Escribimos en (1,1) en la LCD los encabezados
3. Enviamos los valores obtenidos de los esclavos a la terminal virtual
4. Seleccionamos el esclavo 2 y le escribimos un dato
5. Leemos el valor que recibimos del esclavo 2 y deseleccionamos al esclavo 2
6. Convertimos el valor obtenido a valor flotante
7. Verificamos si el valor esta entre 0 y 10 para utilizar únicamente una matriz de leds de la LCD
8. Escribimos en (2,7) el valor obtenido del esclavo 2
9. Seleccionamos el esclavo 1 y le escribimos un dato
10. Leemos el valor que recibimos del esclavo 1 y deseleccionamos al esclavo 1
11. Convertimos el valor obtenido a valor decimal
12. Escribimos en (2,1) el valor obtenido del esclavo 1
13. Seleccionamos el esclavo 3 y le escribimos un dato
14. Leemos el valor que recibimos del esclavo 3 y deseleccionamos al esclavo 3
15. Convertimos el valor obtenido a valor flotante
16. Escribimos en (2,13) el valor obtenido del esclavo 3 y retornamos al paso2)

**Esquemático Proteus**

****

**Código comentado**

**Main**

/\*

\* File: Carrera.c

\* Author: Cristopher Sagastume 18640

\*

\* Created on 22 de enero de 2021, 08:46 AM

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// PALABRA DE CONFIGURACIÓN

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CONFIG1

#pragma config FOSC = INTRC\_CLKOUT// Oscillator Selection bits (INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)

#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 enabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is enabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is enabled)

#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3/PGM pin has PGM function, low voltage programming enabled)

// 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 <stdio.h>

#include <xc.h>

#include<stdint.h>

#include "ADC.h"

#include "LCD.h"

#include "USART.h"

#define \_XTAL\_FREQ 8000000

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Variables

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

uint8\_t flag = 1;

uint8\_t turno = 1;

uint8\_t retorno;

uint8\_t valor\_AN0;

uint8\_t valor\_AN1;

double valor1;

double valor2;

float x;

char s[20];

uint8\_t count = 0;

uint8\_t recibido;

uint8\_t enviado;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Prototipo de funciones

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup(void);

void \_\_interrupt() ISR(void);

double conversor(uint8\_t x);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Ciclo Principal

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void main(void) {

setup();

Config\_USARTT();

recibir();

Lcd\_Init();

Lcd\_Clear();

while (1) {

Lcd\_Set\_Cursor(1, 1); //colocamos el cursor en posicón 1,1

Lcd\_Write\_String("S1: S2: S3:"); //escribimos los encabezados

retorno = ADC\_con(flag, turno); //convertimos los valores de ADC

valor1 = conversor(valor\_AN0); //convertimos la resolución del ADC a

//valores de voltaje en decimal

valor2 = conversor(valor\_AN1);

Lcd\_Set\_Cursor(2, 1);//colocamos el cursor en posición 2,1

sprintf(s, "%3.2fV", valor1); //guardamos los valores de la conversión

//en el array s con un formato de 2 decimales en caracter flotante

Lcd\_Write\_String(s); //escribimos los valores en la LCD

// enviar(s);

Lcd\_Set\_Cursor(2, 7);

sprintf(s, "%3.2fV", valor2);

Lcd\_Write\_String(s);

enviar(s);

\_\_delay\_ms(300);

if (recibido == 0x2B) { //0x2B == signo "+" revisamos si lo recibimos

count++; //aumentamos el contador

recibido = 0;//vaciamos la variable de recibido

}

else if (recibido == 0x2D){//0x2D == signo "-" revisamos si lo recibimos

count--;//disminuimos el contador

recibido = 0; //vaciamos la variable de recibido

}

sprintf(s, "%d", count);//se guarda el valor de contador en s con

//formado de decimal

Lcd\_Set\_Cursor(2, 14);

Lcd\_Write\_String(s);

// enviar(s);

if (count >= 0 && count < 10) {//limpiamos los espacios de decenas y

//centenas si en caso no se necesitaran

Lcd\_Set\_Cursor(2, 15);

Lcd\_Write\_String(" ");

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Configuración

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup(void) {

TRISD = 0b00000000;// puerto D como salida

TRISC = 0b10000000; //activamos el RX como entrada

TRISE = 0b00000000;

ANSEL = 0b00000011;

PORTC = 0;//limpiamos puertos

PORTD = 0;

PORTE = 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funciones

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double conversor(uint8\_t val) {

x = 0.0195 \* val; //5V/256bits=0.0195 convertir de bits a voltaje

return (x);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Interrupciones

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void \_\_interrupt() ISR(void) {

if (PIR1bits.ADIF == 1) {//verificamos si fue interrupt ADC

if (retorno == 0) { //verificamos si fue del AN0

valor\_AN0 = ADRESH; //guardamos el valor

turno = 0;//cambiamos a conversión de AN1

flag = 1;

} else if (retorno == 1) {//verificamos si fue del AN1

valor\_AN1 = ADRESH;

turno = 1;//cambiamos a conversión de AN0

flag = 1;//se setea flag para indicar que puede volver a convertir

}

// flag = 1;

PIR1bits.ADIF = 0; //apagamos la bandera de ADC

}

if (PIR1bits.RCIF == 1){//verificamos si fue interrupcion de la recepción USART

if (RCSTAbits.OERR == 1) {//verificamos si hubo algún error de overrun

RCSTAbits.CREN = 0;

\_\_delay\_us(300);

} else {

recibido = RCREG; //guardamos el valor recibido en una variable

}

}

}

**ADC**

/\*

\* File: ADC.c

\* Author: SAGASTUME

\*

\* Created on 2 de febrero de 2021, 11:46 PM

\*/

#include<stdint.h>

#include <xc.h>

#define \_XTAL\_FREQ 8000000

uint8\_t ADC\_con(uint8\_t flag, uint8\_t turno) {

uint8\_t var;

ADCON1bits.ADFM = 0; //se justifica la resolución del ADC a la izquierda en ADRESH

ANSEL = 0b00000011; //se configura el RA0 como entrada analógica

INTCON = 0b11101000; //se configuran las interrupciones GIE, PIE, T0IE y RBIE

//ADCON0 = 0b01000001;//frecuencia de oscilacion 1/8 canal analógico AN0 y

//encender ADC

PIE1bits.ADIE = 1; //se configura la interrupcion del ADC

PIR1bits.ADIF = 0; //se apaga la bandera de interrupcion ADC

if (flag == 1 && turno == 1) {

ADCON0 = 0b01000001;//frecuencia de oscilacion 1/8 canal analógico AN0 y

//encender ADC

\_\_delay\_us(20);

ADCON0bits.GO = 1; //se indica que empiece a convertir al ADC

flag = 0;

var=0;

} else if (flag == 1 && turno == 0) {

ADCON0 = 0b01000101;//frecuencia de oscilacion 1/8 canal analógico AN1 y

//encender ADC

\_\_delay\_us(20);

ADCON0bits.GO = 1; //se indica que empiece a convertir al ADC

flag = 0;

var=1;

}

return (var);

}

**LCD**

/\*

\* File: LCD.c

\* Author: electroSome.com

\* Editor: Cristopher Sagastume

\*

\* modified on 6 de febrero de 2021, 01:03 AM

\*/

#include<stdint.h>

#include <xc.h>

#define \_XTAL\_FREQ 8000000

#define RS PORTEbits.RE0

#define EN PORTEbits.RE1

//LCD Functions Developed by electroSome

void Lcd\_Port(char a) {

PORTD = a; //se carga a al puerto D completo

}

void Lcd\_Cmd(char a) {

RS = 0; // => RS = 0 se mandan comandos

Lcd\_Port(a);

EN = 1; // => E = 1

\_\_delay\_ms(4);

EN = 0; // => E = 0

}

void Lcd\_Clear(void) {

Lcd\_Cmd(0);

Lcd\_Cmd(1);

}

void Lcd\_Set\_Cursor(char a, char b) {

char temp;

if (a == 1) {

temp = 0x80 + b - 1; //dirección de posicion de DDRAM al puerto

Lcd\_Cmd(temp);

} else if (a == 2) {

temp = 0xC0 + b - 1;

Lcd\_Cmd(temp);

}

}

void Lcd\_Init(void) {

Lcd\_Port(0x00);

\_\_delay\_ms(20);

Lcd\_Cmd(0x3F); //0b00111111 valores de iniciación de LCD

\_\_delay\_ms(10);

Lcd\_Cmd(0x3F);//0b00111111

\_\_delay\_us(200);

Lcd\_Cmd(0x3F);//0b00111111

/////////////////////////////////////////////////////

Lcd\_Cmd(0x38);//0b00111000 modo de funcionamiento

Lcd\_Cmd(0x0C);//0b00001100 encender visualizador

Lcd\_Cmd(0x06);//0b00000110 modo de entrada

}

void Lcd\_Write\_Char(char a) {

RS = 1; // => RS = 1 envio de datos

Lcd\_Port(a); //cargando los datos al puerto

EN = 1;

\_\_delay\_us(40); //transición EN:1->0 para indicar que se mandan datos/comandos

EN = 0;

\_\_delay\_us(40);

}

void Lcd\_Write\_String(char \*a) {

int i;

for (i = 0; a[i] != '\0'; i++) //ciclo cargar direcciones/valores a LCD

Lcd\_Write\_Char(a[i]);

}

**USART**

/\*

\* File: USART.c

\* Author: SAGASTUME

\*

\* Created on 6 de febrero de 2021, 12:41 AM

\*/

#include <xc.h>

#define \_XTAL\_FREQ 8000000

void Config\_USARTT(void) {

TXSTAbits.SYNC = 0; //modo asincrono

TXSTAbits.TXEN = 1; //activamos la transmisión

TXSTAbits.BRGH = 0; //velocidad baja de baud rate (velocidad de trabajo de la comuniación)

BAUDCTLbits.BRG16 = 1; //geneardor de 16 bits de baud rate

SPBRG = 25; // tener un baudrate a 19230

RCSTAbits.SPEN = 1; //se activa la comunación del RX/TX

INTCONbits.GIE = 1; //se activan la interrupción global

return;

}

void enviar(char \*valor) {

TXREG = valor[0];

while (TRMT == 0) {//bucle si TRMT sigue lleno

}

TXREG = valor[1];//si TRMT esta vacio le cargamos otro valor al TXREG->TRMT

while (TRMT == 0) {

}

TXREG = valor[2];

while (TRMT == 0) {

}

TXREG = valor[3];

while (TRMT == 0) {

}

TXREG = 0x20;

while (TRMT == 0) {

}

}

void recibir(void) {

RCSTAbits.CREN = 1; //activamos el recibidor contunio asincrono

RCSTAbits.FERR = 0; //apagamos el error de frame bit

PIE1bits.RCIE = 1; //ecendemos la interrupción de recepción de la comunicación USART

RCSTAbits.OERR = 0; //apagamos el overrun error

}

**Link del repositorio**

<https://github.com/sag18640/Electronica_Digital_2.git>