Universidad del Valle de Guatemala
Natalia de León Bercián
Electrónica Digital 2
carne: 18193
febrero 2021
sección: 20

Mini proyecto # 1

Comunicación SPI

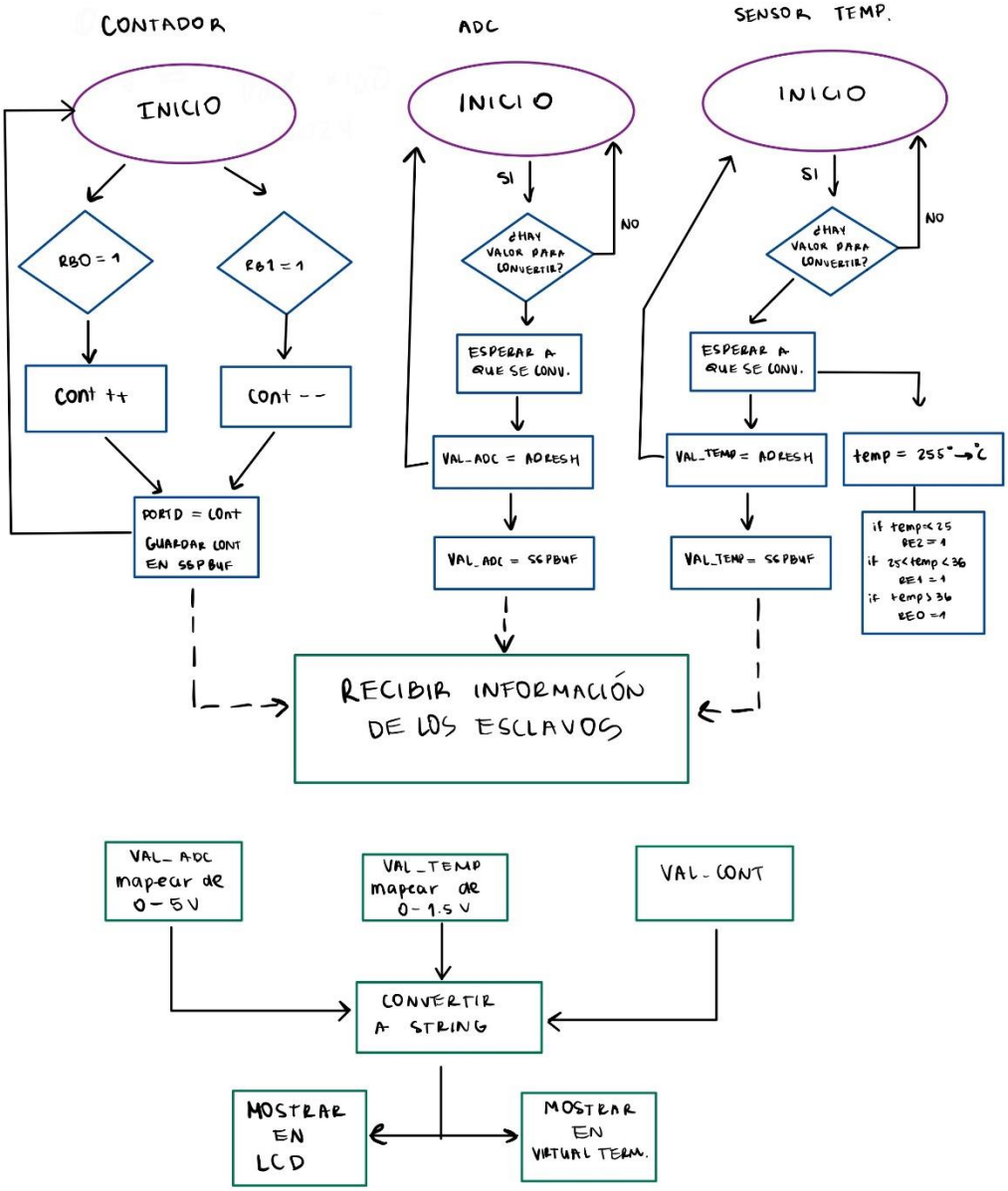
Link video

https://youtu.be/gHJDICKVJik

Link Github

https://github.com/nataliadlb/LABS_REPOSITORIO.git

Diagrama de flujo



Pseudocodigo

CONTADOR

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

// Definición e importación de librerías

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

#include <xc.h>

#include <stdint.h>

#include "SPI.h"

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

// Definición de variables

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

#define _XTAL_FREQ 8000000


uint8_t contador; //Variable de incremento para
contador

uint8_t debouncing1 = 0; //Variable que controla
debouncing de un push

uint8_t debouncing2 = 0;

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

// Definición de funciones para que se puedan colocar
después del main de lo

// contrario hay que colocarlos todas las funciones antes
del main

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

void setup(void);

void Config_INTERRUPT(void);

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

// Código de Interrupción

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

void __interrupt() isr(void){

   if (INTCONbits.RBIF == 1){  // Interrupcion on change

        if (PORTBbits.RB0 == 1){ //debouncing

           debouncing1 = 1;

           contador = contador;

        }

        if (PORTBbits.RB1 == 1){ //debouncing

           debouncing2 = 1;

           contador = contador;

        }

        if(PORTBbits.RB0 == 0 && debouncing1 ==
1){//hasta revisar bandera...

           contador++;                // de deboucing y que el
boton no...

           PORTD = contador;          //este presionado, se
aumenta o...

           debouncing1 = 0;          //decrementa.

        }

        if(PORTBbits.RB1 == 0 && debouncing2 == 1){

           contador--;

           PORTD = contador;

           debouncing2 = 0;

        }

        INTCONbits.RBIF = 0; //limpiar bandera

      }



}

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

```
// Código Principal
//*****************************************
*********************************
void main(void) {
   contador = 0;
   setup();

//*****************************************
*****************************
   // Loop infinito

//*****************************************
*****************************
   while(1){
   }
   return;
}
//*****************************************
*********************************
// Función de Inicialización
//*****************************************
*********************************
void setup(void){
   ANSEL = 0;
   ANSELH = 0;


   TRISA = 0;
   TRISB = 0b00000011; // push, como entradas
   TRISC = 0;
   TRISD = 0;
   TRISE = 0;


   PORTB = 0;
   PORTD = 0;
   PORTA = 0;
   PORTB = 0;
   PORTC = 0;
   PORTD = 0;
   PORTE = 0;
   Config_INTERRUPT();
   //CONFIG spi //
}


void Config_INTERRUPT(void) {
   INTCONbits.GIE = 1;
   INTCONbits.RBIE = 1;
   INTCONbits.RBIF = 0;
   IOCB = 0b00000011;
   INTCONbits.PEIE = 1;      // Habilitamos interrupciones
PEIE
   PIR1bits.SSPIF = 0;              // Borramos bandera
interrupción MSSP
   PIE1bits.SSPIE = 1;          // Habilitamos interrupción
MSSP
}
```

ADC

```
//*************************************
*********************************//
//DEFINE                                    //
//*************************************
*********************************//
#define _XTAL_FREQ 8000000
```

```
//*************************************
**********************************//
//VARIABLES                            //
//*************************************
**********************************//
uint8_t ADC_val;


//*************************************
**********************************//
//PROTOTIPOS DE FUNCIONES
//
//*************************************
**********************************//
void setup(void);
void Config_INTERRUPT(void);


//*************************************
**********************************//
//INTERRUPCIONES                       //
//*************************************
**********************************//


void __interrupt() ISR(void) {

  //---- interrupcion SPI ----//
  if(SSPIF == 1){
    spiWrite(ADC_val);
    SSPIF = 0;
  }
}
//*************************************
**********************************//
//PROGRAMACION PRINCIPAL
//
//*************************************
**********************************//


void main(void) {
  setup();
//*************************************
**********************************//
  //LOOP PRINCIPAL                     //

//*************************************
**********************************//
  while (1) {
    __delay_ms(2);
    ADCON0bits.GO = 1; //Inicio de conversion ADC
    while (ADCON0bits.GO != 0) { //Mientras no se
haya termindo una convers.
    ADC_val = ADRESH;
    PORTD = ADC_val;
    }
  }
}


//*************************************
**********************************//
//CONFIGURACION (puertos, bits...)
//
//*************************************
**********************************//


//----- puertos -----//
void setup(void) {
```

```
  initOsc(0b00000111);
  nRBPU = 0;

  ANSEL = 0b00000001; //RA0 como analogico
  ANSELH = 0;

  TRISA = 0b00000001; //potenciometro, como entrada
  TRISB = 0;
  TRISC = 0;
  TRISD = 0;
  TRISE = 0;
  PORTA = 0;
  PORTB = 0;
  PORTC = 0;
  PORTD = 0;
  PORTE = 0;
  ADCON0 = 0b01000001;
  Config_INTERRUPT();
```

```
//Config SPI
}

//----- interrupciones -----//

void Config_INTERRUPT(void) {
  INTCONbits.GIE = 1;          // Habilitamos interrupciones
  INTCONbits.PEIE = 1;         // Habilitamos interrupciones PEIE
 // PIR1bits.SSPIF = 0;        // Borramos bandera interrupción MSSP
 // PIE1bits.SSPIE = 1;        // Habilitamos interrupción MSSP
}
//*************************************************************//
//FUNCIONES                                                    //
//*************************************************************//
```

SENSOR DE TEMPERATURA

```
//*************************************************************//
//DEFINE                                                       //
//*************************************************************//
#define _XTAL_FREQ 8000000

//*************************************************************//
//VARIABLES                                                    //
//*************************************************************//
int mv_temp_val;
//int temp;
uint8_t temp_val;

//*************************************************************//
//PROTOTIPOS DE FUNCIONES                                      //
//*************************************************************//
```

```
void setup(void);
void Config_INTERRUPT(void);
void semaforo(void);
//*****************************************************************************//
//INTERRUPCIONES                                                               //
//*****************************************************************************//

void __interrupt() ISR(void) {

  //---- interrupcion SPI ----//
}

//*****************************************************************************//
//PROGRAMACION PRINCIPAL                                                       //
//*****************************************************************************//

void main(void) {
  setup();

//*****************************************************************************//
  //LOOP PRINCIPAL                                                             //
//*****************************************************************************//
  while (1) {
    //Tomado de mi laboratorio # 3
    __delay_ms(2);
    ADCON0bits.GO = 1; //Inicio de conversion ADC
    while (ADCON0bits.GO != 0) { //Mientras no se haya termindo una convers.
      temp_val = ADRESH;
      mv_temp_val = ((ADRESH * 150) / 255); //mapearlo de 0-1.5V
    }
    semaforo();
  }

}

//*****************************************************************************//
//FUNCIONES                                                                    //
//*****************************************************************************//

void semaforo(void){//Encender led segun rango de temperatura

  if (mv_temp_val <= 25){
    RE0 = 0;
    RE1 = 0;
    RE2 = 1;
  }
  else if (mv_temp_val > 25 && mv_temp_val <= 36){
    RE0 = 0;
    RE1 = 1;
    RE2 = 0;
  }
  else if (mv_temp_val > 36){
    RE0 = 1;
    RE1 = 0;
    RE2 = 0;
```

```
   }
}

//*****************************************
*******************************//
//CONFIGURACION (puertos, bits...)                    //
//*****************************************
*******************************//

//----- puertos -----//
void setup(void) {
   initOsc(0b00000111);
   nRBPU = 0;

   ANSEL = 0b00000001; //RA0 como analogico
   ANSELH = 0;

   TRISA = 0b00000001; //sensor, como entrada
   TRISB = 0;
   TRISC = 0;
   TRISD = 0;
   TRISE = 0; // semaforo como output

   PORTA = 0;
   PORTB = 0;
   PORTC = 0;
   PORTD = 0;
   PORTE = 0;
   //  ADCON1 = 0;
   ADCON0 = 0b01000001;
   ADCON1bits.VCFG0 = 1;
   //ADCON1 = 0x07;

   Config_INTERRUPT() ;
   //CONFIG SPI
}

//----- interrupciones -----//
```

MAESTRO

```
//*****************************************
*************************//
//VARIABLES                                   //
//*****************************************
*************************//
char data_cont[8];
char data_ADC[8];
char data_TEMP[8];
uint8_t hola_esclavo; //para que el maestro hable con esclavos
uint8_t cont = 0;
uint8_t val_ADC = 0;
uint8_t val_TEMP = 0;
int mv_temp_val_M;
float ADC_val_M;
//*****************************************
*************************//
//PROTOTIPOS DE FUNCIONES                         //
//*****************************************
*************************//
void setup(void);
void ADC_to_string(void);
void Show_val_LCD(void);
void Show_val_VT(void);
void Mapeo_M(void);
void SPI_CONT(void);
void SPI_ADC(void);
```

```
void SPI_TEMP(void);

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

//INTERRUPCIONES                                 //

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


void __interrupt() ISR(void) {

}


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

//PROGRAMACION PRINCIPAL                                 //

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


void main(void) {

  setup();

  cont = 0;

  Lcd_Init();

  Lcd_Clear();

  Lcd_Set_Cursor(1,1); //nombres S1, S2 y S3

  Lcd_Write_String("CONT   ADC  TEMP");



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

  //LOOP PRINCIPAL                                 //

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

  while (1) {

    SPI_CONT(); //Activar, guardar valor y desactivar esclavo
cont

    SPI_ADC();  //Activar, guardar valor y desactivar esclavo
ADC

    SPI_TEMP(); //Activar, guardar valor y desactivar esclavo
temp

    Mapeo_M();  //Mapear valores del ADC  de temp

    ADC_to_string(); //convertir a strings los valores

    Show_val_VT();    // mandar y mostrar los valores en la
termianl virtual

    Show_val_LCD();   // mandar y mostrar los valores en la
LCD

  }

}


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

//FUNCIONES                                 //

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


//------ FUNCIONES MAESTRO ------//


void ADC_to_string(void){ //Volver texto los valores para LCD
y Terminal virtual

  sprintf(data_cont, "%.3i", cont);

  sprintf(data_ADC, "%1.2fV", ADC_val_M);

  sprintf(data_TEMP, "%.2i", mv_temp_val_M);

}


void Mapeo_M(void){ //mapear valores del ADC y temp

  ADC_val_M = ((val_ADC * 5.0) / 255); //mapear de 0-5 en
floats

  mv_temp_val_M = ((val_TEMP * 150) / 255); //mapear de 0-
1.5 en int

}


void Show_val_LCD(void){ //mostrar valores en la LCD, luego
de SPI

  Lcd_Set_Cursor(2,1);

  Lcd_Write_String(data_cont); //valor cont
```

```
   Lcd_Set_Cursor(2,7);
   Lcd_Write_String(data_ADC);  //valor ADC
   Lcd_Set_Cursor(2,16);
   Lcd_Write_String("C");
   Lcd_Set_Cursor(2,14);
   Lcd_Write_String(data_TEMP); //valor temp

}
void Show_val_VT(void){
   Write_USART_String("CONT:  \n");
   Write_USART_String(data_cont); //enviar el string del valor de cont a VT
   Write_USART_String("  \n");
   Write_USART_String("ADC:  \n");
   Write_USART_String(data_ADC); //enviar el string del valor de ADC a VT
   Write_USART_String("  \n");
   Write_USART_String("TEMP:  \n");
   Write_USART_String(data_TEMP); //enviar el string del valor de temp a VT
   Write_USART_String("°C  \n");
   Write_USART(13);//13 y 10 la secuencia es para dar un salto de linea
   Write_USART(10);
}

//------ FUNCIONES ACTIVACION ESCLAVOS ------//

void SPI_CONT(void){ //CONTADOR, seleccionar y guardar valor
}

void SPI_ADC(void){ // ADC, seleccionar y guardar valor
}
void SPI_TEMP(void){//TEMP, seleccionar y guardar valo
}

//************************************************************************//
//CONFIGURACION                                   //
//************************************************************************//

//----- puertos -----//
void setup(void) {
   initOsc(0b00000111); //8MHz
   ANSEL = 0;
   ANSELH = 0;

   TRISA = 0;
   TRISB = 0;
   //TRISCbits.TRISC6 = 0;
   //TRISCbits.TRISC7 = 1; // RX

   TRISD = 0;
   TRISE = 0;

   PORTA = 0;
   PORTB = 0;
   PORTC = 0;
   PORTD = 0;
   PORTE = 0;

   //----- SPI -----//
   TRISC0 = 0;
   PORTCbits.RC0 = 1;
   TRISC1 = 0;
   PORTCbits.RC1 = 1;
   TRISC2 = 0;
   PORTCbits.RC2 = 1;
```

```
    //----- USART -----//
    USART_Init_BaudRate();
    USART_Init();
    //SPI CONFIGUACION
}
```