PROYECTO TD IIUTN-FRA 2018
REGISTRADOR DE TEMPERATURAS CON TERMOPARES

Garcia Leonel
e-mail: garcialeonel1990@gmail.com

RESUMEN:*El proyecto a realizar consiste en un registrador de temperaturas mediante 8 termopares tipo k con conexión a pc.*

PALABRAS CLAVE: Registrador – Termopar- temperaturas.

1 INTRODUCCIÓN

El principio de funcionamiento del registrador de temperaturas se basa en medir las temperaturas a través 8 termopares tipo k que nos dan una tensión en mV en relación a los ºC que miden. Dichos termopares se conectan a dos multiplexores, uno por cada polo, que nos permite ir seleccionando de a un termopar a la vez y así poder ingresar la tensión de salida de cada una a un circuito amplificador. El circuito cuenta con un circuito integrado que adecua el nivel de tensión para que dicho nivel pueda ser ingresado por una entrada al microcontrolador. La entrada del microcontrolador cuenta con un conversor analógico digital, con el que logramos transformar dicho valor de tensión a un valor de temperatura nuevamente. A dicho valor de temperatura hay que restarle la temperatura ambiente que se va a medir con un sensor LM 35 que se conecta a otra entrada del microcontrolador y este último se encarga de hacer la operación matemática. Las mediciones de temperatura se van a configurar cada cierto tiempo a través de un menú de opciones. También se va a poder ver cuál fue la temperatura máxima y mínima de cada termopar. El microcontrolador enviará todas las mediciones a través de su puerto serie al conversor serie usb, el cual nos brinda la conexión con la pc, y a su vez las mostrará en el display LCD. El envío de datos a la pc nos va a permitir obtener un análisis mucho más profundo de las temperaturas registradas, tener una base de datos de todas las mediciones realizadas, hacer comparaciones con mediciones anteriores y gráficos de temperatura vs tiempo.

2 DESCRIPCION DE LOS MODULOS DE HARDWARE PRINCIPALES

2.1 SENSOR DE TEMPERATURA

Los sensores que se ultilizaran van a ser 8 termopares tipo k.
Un termopar es un transductor formado por la unión de dos metales distintos que produce una diferencia de potencial muy pequeña (del orden de los milivoltios) que es función de la diferencia de temperatura entre uno de los extremos denominado punto caliente y el otro llamado punto frío.
Además de lidiar con la compensación de unión fría (punto frio), el instrumento de medición debe además enfrentar el hecho de que la energía generada por un termopar no es una función lineal de la temperatura. Esta dependencia se puede aproximar por un polinomio complejo (de grado 5 a 9, dependiendo del tipo de termopar).
Termopar tipo K (cromel/alumel): Tienen un rango de temperatura de –200 °C a +1372 °C y una sensibilidad 41 µV/°C aproximadamente.

Por lo cual en el proyecto la medición que se realiza con el termopar le debemos realizarle la corrección de la unión fría, linealizar la tensión generada por medición y por ultimo amplificar el nivel de tensión generado por °C para poder ingresar ese valor por una de las entrada del microcontrolador con ADC.

Figura 1.Termopar

2.2 MULTIPLEXOR

La El 74HC4067 es un multiplexor de 16 canales analógicos / digitales.
Cuenta con cuatro entradas de selección digital (S0, S1, S2 y S3), dieciséis entradas / salidas independientes (Yn), una entrada / salida común (Z) y una entrada de habilitación digital (E). Cuando E está en ALTO, deshabilita las entradas.

En el proyecto se van a utilizar dos multiplexores, los cuales me van a permitir ir seleccionando cada una de los termopares. Se utilizan dos porque es uno para cada polo de los termopares. Las entradas de selección de ambos multiplexores van a estar unidas con lo cual cambiando los valores de dichas entras voy a ir activando las diferentes entradas de los multiplexores en conjunto lo que me va a permitir tener a las salidas de ambos los polos de un termopar a la vez.

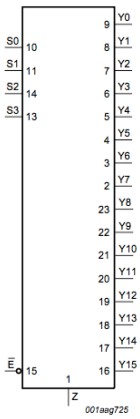


Figura 2.Multiplexor

2.3 AMPLIFICADOR DE TERMOPARES

Nuestro proyecto dispondrá de un circuito integrado AD595.

El AD595 es un amplificador de instrumentación y un compensador de la unión fría de los termopares en un chip monolítico. Este posee un amplificador precalibrado que genera 10mv/°C a la salida directamente proporcional a la señal de entrada del termopar. Dicho circuito integrado también corrige utilizando una función de transferencia la alinalidad de la tensión de salida con respecto a la temperatura.

En el proyecto se va a alimentar al circuito integrado con una tensión de -5 - 15V para poder medir todo el rango de temperaturas que pueden medir los termopares tipo k, permitiendo representar la información que genera cualquier equipo electrónico de una forma sencilla.

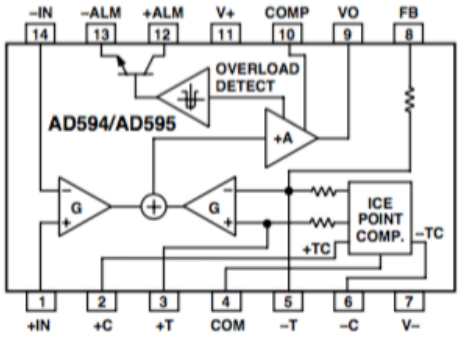


Figura 3. AD595

2.4 LM35

El LM35 es un sensor de temperatura con una precisión calibrada de 1 °C. Su rango de medición abarca desde -55 °C hasta 150 °C. La salida es lineal y cada grado Celcius equivale a 10 mV. Opera de 4v a 30v. El LM35 no requiere de circuitos adicionales para calibrarlo externamente. La baja impedancia de salida, su salida lineal y su precisa calibración hace posible que este integrado sea instalado fácilmente en un circuito de control. Debido a su baja corriente de alimentación se produce un efecto de auto calentamiento muy reducido. Se encuentra en diferentes tipos de encapsulado, el más común es el TO-92, utilizado por transistores de baja potencia.

En el proyecto se va a utilizar para medir la temperatura ambiente antes de empezar a tomar las mediciones con los termopares.

2.5 DISPLAY LCD

Nuestro proyecto dispondrá de un módulo LCD tipo 20x4, el CCM1602.

Las pantallas de cristal líquido LCD o display LCD para mensajes (Liquid Cristal Display) tienen la capacidad de mostrar cualquier carácter alfanumérico, permitiendo representar la información que genera cualquier equipo electrónico de una forma sencilla.

El proceso de visualización es gobernado por un microcontrolador incorporado a la pantalla, siendo el Hitachi 44780 el modelo de controlador más utilizado

Figura 4.Display LCD 2x20

2.6 CONVERSOR SERIE - USB

Nuestro proyecto dispondrá de un módulo Conversor USB a Serie TTL FT232 FTDI el cual nos brindará la comunicación del microcontrolador con la pc a través de un puerto USB.

En el proyecto el microcontrolador va a enviar los datos de las temperaturas registradas a través de su puerto serie interno a dicho modulo, el cual se encargará de generar un puerto serie virtual en la pc (COM) para generar la compatibilidad de la trasmisión de datos entre ambos dispositivos.

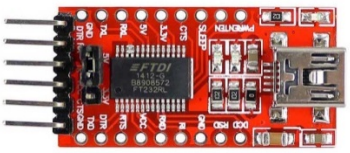


Figura 4. FTDI

2.7 MICROCONTROLADOR

El microcontroladorquevoy a utilizar en el proyectoes la version de 44 pins del MC9S08JM60. El cualvaser el encargado de llevar a cabotoda la logica de control del dispositivo registrador de temperaturas.

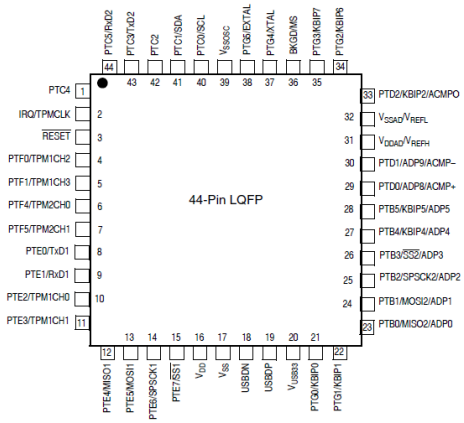


Figura 6. MicrocontroladorMC9S08JM60

3 DIAGRAMA BASICO DE LOS MODULOS PRINCIPALES

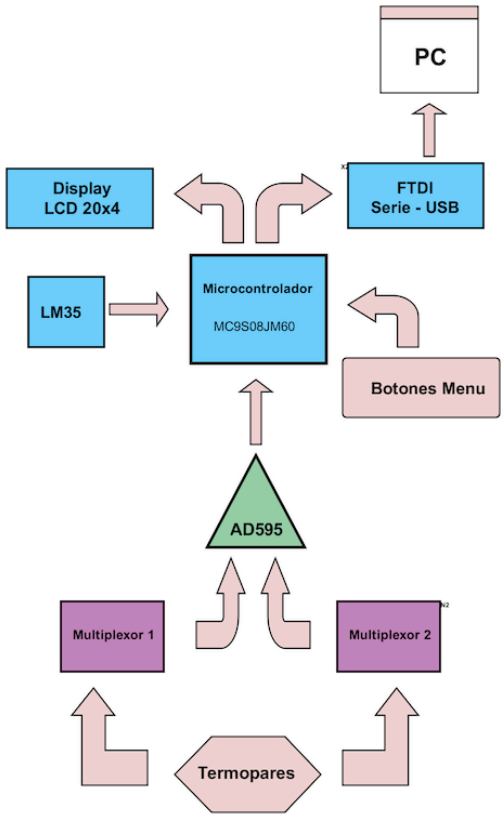


Figura 7. Diagrama básico del registrador de termopares.

3.1 Diagrama del Display LCD

Para manejar el LCD se utiliza el puerto E completo (Disp_0 a Disp_7), junto con el PTG4 (Disp_Tr) para el backlight.
El potenciómetro será utilizado para regular el contraste del LCD.

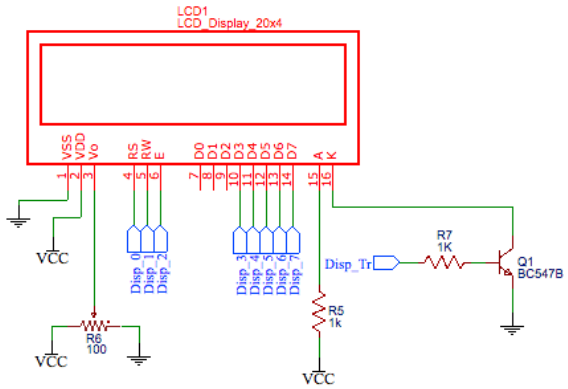


Figura 8. ConexiónDisplayLCD (Anexo III)

3.2 Diagrama del FTDI Serie – USB

Este módulo de comunicación Serie-USB está compuesto por el chip FT232RL. Esto permite que solamente necesitemos conectar los canales Rx y Tx para realizar la comunicación serial de datos con la pc vía USB.
Tx lo conectamos al puerto TXD2.
Rx lo conectamos al puerto RXD2.

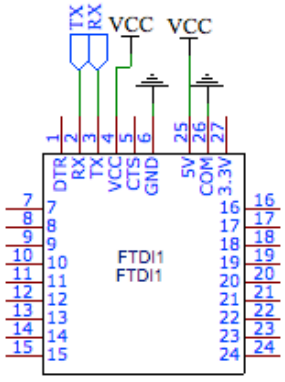


Figura 9.Conexión FTDI Serie – USB (Anexo III)

3.3 Diagrama del LM35

En este caso el sensor LM35 debe conectarse a 5V, GNDy su salida analógica (10mV por grado ºC) al canal ADP5 (PTB5).

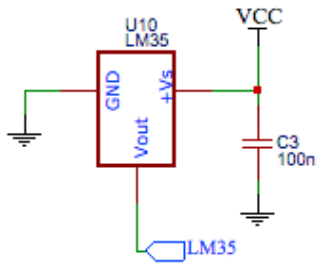


Figura 10.Conexión LM35 (Anexo III)

3.4 Diagrama de los Botones / Menu

Los pulsadores que utilizaremos son del tipo normal abierto.

Figura 11. Pulsador NA.

En cuanto a la conexión de los mismos irán conectados a distintos puertos de entrada con la característica de tener interrupción por teclado.

El pulsador 1 se utiliza para mostrar en el display la temperatura máxima registrada por los termopares de los 8 canales. El mismo se conecta al puerto KBIP0.

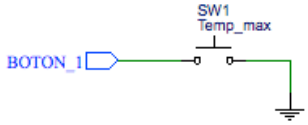


Figura 12. Conexión Pulsador 1. (Anexo III)

El pulsador 2 se utiliza para mostrar en el display la temperatura mínima registrada por los termopares de los 8 canales. El mismo se conecta al puerto KBIP1.

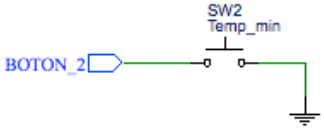


Figura 13. Conexión Pulsador 2. (Anexo III)

El pulsador 3 se utiliza dentro del menú de configuración de la comunicación serie para elegir las opciones"SI / +". El mismo se conecta al puerto KBIP2.

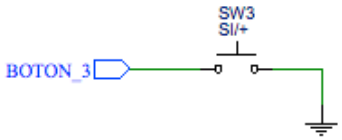


Figura 14. Conexión Pulsador 3. (Anexo III)

El pulsador 4 se utiliza dentro del menú de configuración de la comunicación serie para elegir las opciones "NO / ". El mismo se conecta al puerto KBIP6.

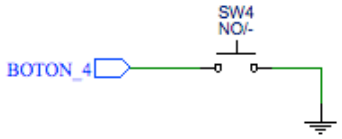


Figura 15. Conexión Pulsador 4. (Anexo III)

El pulsador 5 se utiliza para ingresar al menú de configuración serie y, una vez dentro del mismo, para elegir la opción "OK". El mismo se conecta al puerto KBIP7.

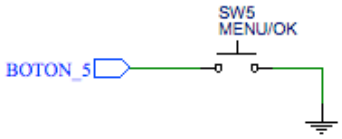


Figura 16. Conexión Pulsador 5. (Anexo III)

3.5 Diagrama delAD595

En cuanto al integrado AD595, el mismo cumple la función de amplificar la señal proveniente de los multiplexores que van seleccionando cada una de los termopares (M_out+ y M_out-). Además, podemos saber si hay o no un termopar conectado a la entrada mediante la señal de alarma que genera el integrado (Termo?). Dicha señal se conecta al microcontrolador por la entrada PTF0.
La salida analógica que genera el AD595 de 10mV por grado ºC se conecta a la entrada del microcontrolador ADP8 (PTD0).
La tensión de alimentación del integrado es de +15V para lograr una medición de temperatura positiva hasta llegar al rango máximo de los termopares tipo K (~1372ºC).

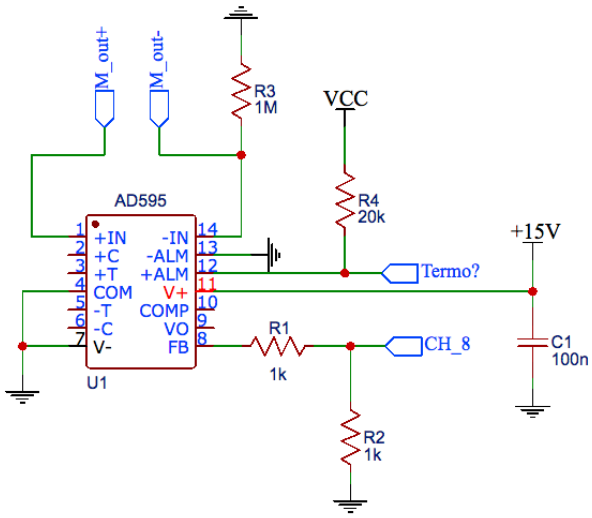


Figura 17. Conexión AD595. (Anexo III)

3.6 Diagrama de la Fuente de alimentación

Para la alimentación del registrador de temperaturas se utiliza un Jack DC 2.1mm. en cual se conecta una fuente tipo switching de 5V.

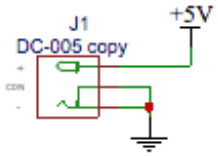


Figura 18. Jack DC. (Anexo III)

Para la alimentación general del circuito del registrador se utiliza un módulo boost DC MT3608 (1) el cual se regula en 5V de salida para tener una tensión de trabajo bien regulada.

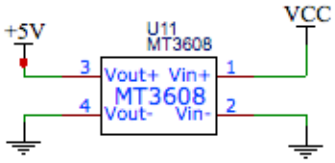


Figura 19. Módulo MT3608 (1). (Anexo III)

Para la alimentación del AD959 se utiliza otro módulo boost DC MT3608, pero esta vez la tensión de salida está regulada en +15V que es la tensión que ese necesita para este integrado.

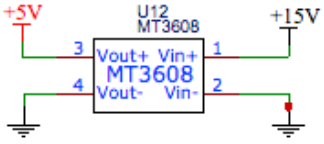


Figura 20. MóduloMT3608 (2). (Anexo III)

3.7 Diagrama de conexión de los multiplexores

Para obtener las temperaturas de 8 termopares se utilizan 2 multiplexores, uno para el terminal + y otro para el terminal – de cada termopar. Cada multiplexor cuenta con 16 entradas de las cuales solo se utilizan 8 (M1-M8). La salida del multiplexor que agrupa a los terminales + (M_out+) se conecta al terminal de entrada de termopar + del AD595; y la salida del multiplexor que agrupa a los terminales – (M_out-) al terminal – del AD595.
Las entradas de control de los multiplexores (MUX_S0 a MUX_S3) se conectan a las salidas del microcontroladorPTB0 a PTB3. Los multiplexores cuentan con una entrada de habilitación (MUX_EN) que se conecta a la salida del microcontrolador PTB4.

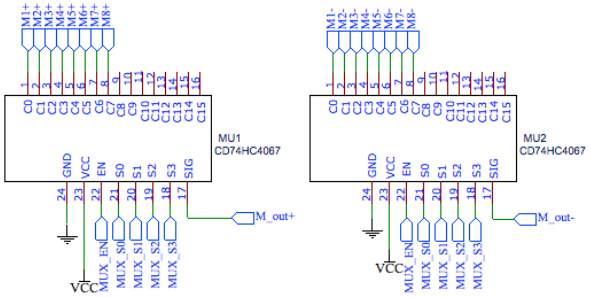


Figura 21. Diagrama de conexión multiplexores. (Anexo III)

3.8 Diagrama de conexión de los termopares.

La conexión de los termopares se realiza a través de fichas tipo hembra, las cuales cuentan con 2 terminales cada una.

Los terminales positivos de los termopares (M1+ a M8+) se conectan a las entradas del multiplexor que nuclea a todos los terminales positivos.

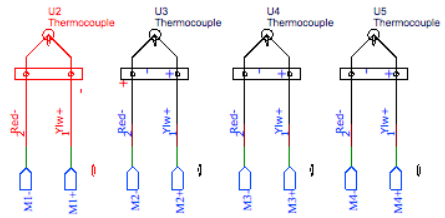


Figura 22. Conexionado Termopares (1) (Anexo III)

Los terminales negativos de los termopares (M1- a M8-) se conectan a las entradas del multiplexor que nuclea a todos los terminales negativos.

Los terminales negativos de los termopares (M1- a M8-) se conectan a las entradas del multiplexor que nuclea a todos los terminales negativos.

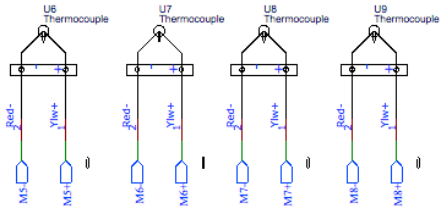


Figura 23. Conexionado Termopares (2) (Anexo III)

4 Funcionamiento

Al encender el registrador, podemos observar las temperaturas de los 8 termopares en el display.
Al oprimir el pulsador de "Temperatura Máxima", en el display se muestran las temperaturas máximas registradas hasta ese momento por cada termopar.
De forma análoga, al oprimir el pulsador "Temperatura Mínima" en el display se muestran las temperaturas mínimas registradas hasta ese momento por cada termopar.
En ambos casos, para volver a mostrar las temperaturas actuales de los termopares se debe volver a presionar el mismo pulsador.
Al presionar "Temperatura Máxima" y "Temperatura Mínima" al mismo tiempo, se muestra en el display la temperatura ambiente actual por unos y luego se vuelven a mostrar las temperaturas actuales de los 8 termopares de forma automática.
Para realizar la configuración de envío de datos vía comunicación serie se debe oprimir el pulsador "Menu/ OK", el display nos muestra las opciones "SI" y "NO".
Presionar el pulsador "NO/- " sirve para volver atrás sin realizar ninguna configuración o para cortar la transmisión de datos si está activa la comunicación serie.
Al presionar el pulsador que representa la opción "SI / +" ingresamos a la configuración del envío de datos vía comunicación serie.
Como primera opción vamos a configurar el intervalo entre envío y envío de temperaturas vía comunicación serial a la pc.
Con los pulsadores "NO/- " Y "SI / +", podemos aumentar o disminuir el tiempo del intervalo. Los intervalos posibles son 5,10,15,20,30 minutos. Para seleccionar la opción deseada se debe pulsar "Menú/OK".
Automáticamente se pasa al menú de "Tiempo total", que es el tiempo total que se van a enviar las temperaturas por comunicación serie a la pc. De la misma forma que antes con los pulsadores "NO/- " Y "SI / +", podemos aumentar o disminuir el tiempo total que se va a realizar la comunicación.
El tiempo total de registro posible va de 1 a 5Hs. Al seleccionar el tiempo total deseado se presiona el pulsador "Menú/OK" y la configuración está terminada.
Se enciende el led amarillo indicando que está activa la comunicación serie.
Al estar activo el envío de datos vía comunicación serial, no es posible consultar las temperaturas máximas ni mínimas.
Al finalizar el tiempo total del envío de temperaturas por comunicación vía serie, se realiza un envío extra con las temperaturas máximas y mínimas registradas durante el tiempo que se realizó la comunicación serie y también la temperatura ambiente actual al momento de finalización de la comunicación serie.

Figura 24. Diagrama de Flujo (Anexo I)

5 Bibliografia

[1] Manual del MC9S08JM60 [En línea]. Disponible en: https://www.nxp.com/docs/en/data-sheet/MC9S08JM60.pdf

[2] Manual del AD595 [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/AD594_595.pdf

[3] Manual del LM35 [En línea]. Disponible en: http://www.ti.com/lit/ds/symlink/lm35.pdf

[4] Manuales del Display LCD 20x4 [En línea]. Disponible en: https://drive.google.com/open?id=0B8i5PxQ5XywyLVdyM3dVdkQ1Mlk https://drive.google.com/open?id=0B8i5PxQ5XywyelAtUTNyZUZlWTA

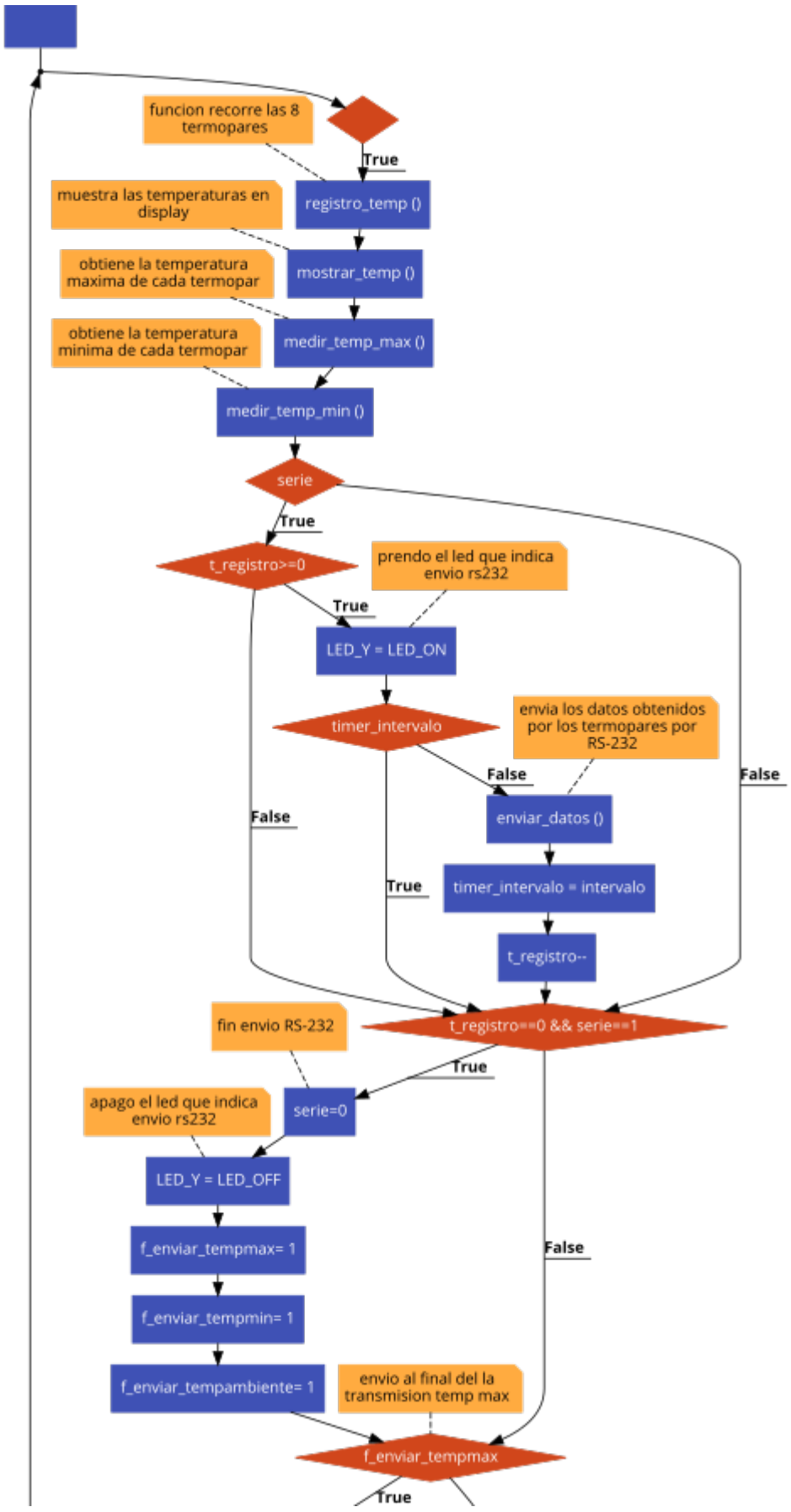
[5] Información sobre Termopares tipo K [En línea]. Disponible en: https://www.thermocoupleinfo.com/type-k-thermocouple.htm

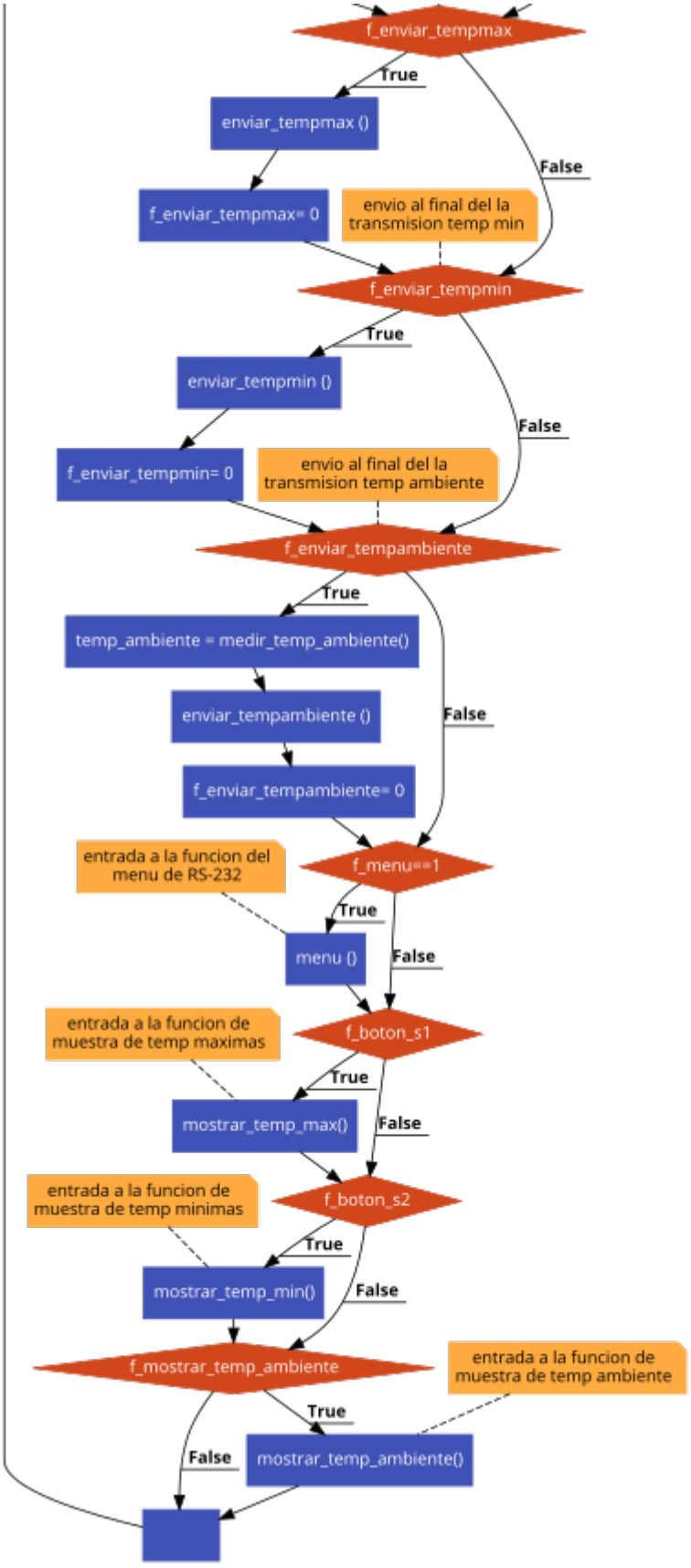
[6] Manuales del multiplexor 74HC4067 [En línea]. Disponible en: https://assets.nexperia.com/documents/data-sheet/74HC_HCT4067.pdf

[7] Manual del módulo FTDI232 [En línea]. Disponible en: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

ANEXO I:

Diagrama de flujo:


funcion recorre las 8 termopares
True
muestra las temperaturas en display
registro_temp ()
obtiene la temperatura maxima de cada termopar
mostrar_temp ()
obtiene la temperatura minima de cada termopar
medir_temp_max ()
medir_temp_min ()
serie
True
t_registro>=0
prendo el led que indica envio rs232
True
LED_Y = LED_ON
timer_intervalo
envia los datos obtenidos por los termopares por RS-232
False
False
False
enviar_datos ()
True
timer_intervalo = intervalo
t_registro--
fin envio RS-232
t_registro==0 && serie==1
True
apago el led que indica envio rs232
serie=0
LED_Y = LED_OFF
False
f_enviar_tempmax= 1
f_enviar_tempmin= 1
f_enviar_tempambiente= 1
envio al final del la transmision temp max
f_enviar_tempmax
True


f_enviar_tempmax
True
False
enviar_tempmax ()
f_enviar_tempmax= 0
envio al final del la transmision temp min
f_enviar_tempmin
True
False
enviar_tempmin ()
f_enviar_tempmin= 0
envio al final del la transmision temp ambiente
f_enviar_tempambiente
True
False
temp_ambiente = medir_temp_ambiente()
enviar_tempambiente ()
f_enviar_tempambiente= 0
entrada a la funcion del menu de RS-232
f_menu==1
True
False
menu ()
entrada a la funcion de muestra de temp maximas
f_boton_s1
True
False
mostrar_temp_max()
entrada a la funcion de muestra de temp minimas
f_boton_s2
True
False
mostrar_temp_min()
f_mostrar_temp_ambiente
entrada a la funcion de muestra de temp ambiente
True
False
mostrar_temp_ambiente()

ANEXO II:

Programa en C:

```
//-------------------------------------------------------------------------------------//
//----------- Declaracion de includes

#include<hidef.h> /* forEnableInterrupts macro */
#include "derivative.h" /* includeperipheraldeclarations */

#include "lcd_char.h"
#include "stdio.h"
#include "string.h"

//-------------------------------------------------------------------------------------//
//----------- Declaracion de definiciones

#define LED_Y               PTCD_PTCD2                      // pin= 42

#define LED_ON                          0
#define LED_OFF                                 1

#define BOTON_S1            PTGD_PTGD0              // pin= 21Min              KBI0
#define BOTON_S2            PTGD_PTGD1              // pin= 22  MAX            KBI1

#define BOTON_MENU          PTDD_PTDD2              // pin= 33 Si/Up           KBI2
#define BOTON_UP            PTGD_PTGD2              // pin= 34 Registro        KBI6
#define BOTON_DOWN          PTGD_PTGD3              // pin= 35 No/Down         KBI7

#define BOTON_T             PTFD_PTGF0              // pin= 4 para saber si hay termocupla

#define ADC_LM35            0x45                    // pin= 28

#define ADC_CANAL8          0x48                    // pin= 29


//-------------------------------------------------------------------------------------//
//---------------------------------- Declaracion de variable-----------------------------//

unsigned char TRIM_32KHz @0xFFAF;   // oscilador interno.

//----- Medicion termopares ------------

float temp_act[8]={0,0,0,0,0,0,0,0};  // vector temperatura actual

float temp_max[8]={0,0,0,0,0,0,0,0};            // vector temperatura maxima

float temp_min[8]={350,350,350,350,350,350,350,350};          // vector temperatura minima

float adc_ch8;                          // variable donde guardo el valor del adc en .flotante

float adc_lm35;                         // variable donde guardo el valor del adc en .flotante

float temp_ambiente;

signed int termo=0;                     // variable que identifica cada termopar

unsigned int mseg=0;        // variable delay

unsigned char f_enviar_tempmax= 0;

unsigned char f_enviar_tempmin= 0;
```

```
unsigned char f_enviar_tempambiente= 0;

//----------------------------------------

//------------ variables Menu -----------

unsigned int flag=0, f_menu=0, flag_set_point=0, f=0, serie=0; // variables banderas menu de opciones

int contador=0,t_registro=0, disp_intervalo=0, disp_t_registro=0;                    // variables seleccionmenu.

unsigned char f_boton_up, f_boton_menu, f_boton_down;       // botones seleccionmenu.

long intervalo=0,timer_intervalo=0;

//----------------------------------------

unsigned char f_boton_s1=0, f_boton_s2=0, buffer[30];           // botones temp Max y Min, buffer display

unsigned char f_mostrar_temp_ambiente =0;

//---------------- RS-232------------------

unsigned char timeout_rx2, f_rx_sci2, f_sci2,b_sci2[30],output[80];

int v=1;

//-------------------------------------------------------------------------------------//


//------------------------------ Prototipos funciones ----------------------------------//


void inicio_puertos(void);
void inicio_cop_lvi_irq(void);
void inicio_rtc(void);
void inicio_kbi(void);
void delay_mseg(unsignedint ms);
void inicio_adc(void);

void inicio_osc(void);
void inicio_sci2(unsignedintbr);
void sci2_byte(unsignedcharsci_tx);
void sci2_string(unsignedchar *dato);

void registro_temp (void);

float medir_temp (void);
void mostrar_temp(void);
void mostrar_temp_max(void);
void mostrar_temp_min(void);
void mostrar_temp_ambiente(void);

float medir_temp_max (void);
float medir_temp_min (void);
float medir_temp_ambiente(void);

unsignedcharenviar_datos (void);
unsigned char enviar_tempmax (void);
unsigned char enviar_tempmin (void);
unsigned char enviar_tempambiente (void);
```

```
void menu (void);

void confi_rs232(void);

//-------------------------------------------------------------------------------------//


//---------------------------------------- Main -----------------------------------------//

void main(void)
{
        inicio_cop_lvi_irq();
        inicio_puertos();
        inicio_rtc();
        inicio_kbi();
        inicio_adc();
        inicio_osc();
        inicio_sci2(9600);
        lcd_char_init4();                       //inicializacion del display

        EnableInterrupts;// habilita todas las interrupciones


        PTBD = 0x00;                            // cargo toddo el puerto B con cero.
        LED_Y = LED_OFF;                                        // apago el led amarillo porque

        for(;;)
        {
                registro_temp ();  // funcion recorre las 8 termopares

                mostrar_temp ();  // muestra las temperaturas en display

                medir_temp_max ();          // obtiene la temperatura maxima de cada termopar

                medir_temp_min ();          // obtiene la temperatura minima de cada termopar


                if (serie)
                {
                        if(t_registro>=0)
                        {
                                LED_Y = LED_ON;          // prendo el led que indica envio rs232
                                if(!timer_intervalo)
                                {
                                        enviar_datos ();   // envia los datos obtenidos por los termopares por RS-232
                                        timer_intervalo = intervalo;
                                        t_registro--;
                                }
                        }
                }

                if (t_registro==0 && serie==1)
                        {
                        serie=0;// fin envio RS-232
                        LED_Y = LED_OFF;// apago el led que indica envio rs232
                        f_enviar_tempmax= 1;
                        f_enviar_tempmin= 1;
                        f_enviar_tempambiente= 1;
                        }
```

```
                if (f_enviar_tempmax)       // envio al final del la transmisiontempmax
                        {
                                enviar_tempmax ();
                                f_enviar_tempmax= 0;
                        }

                if (f_enviar_tempmin)       // envio al final del la transmisiontemp min
                        {
                                enviar_tempmin ();
                                f_enviar_tempmin= 0;
                        }

                if (f_enviar_tempambiente)          // envio al final del la transmisiontemp ambiente
                        {
                                temp_ambiente = medir_temp_ambiente();
                                enviar_tempambiente ();
                                f_enviar_tempambiente= 0;
                        }

                if (f_menu==1) menu ();     // entrada a la funcion del menu de RS-232

                if (f_boton_s1) mostrar_temp_max();         // entra funcion de muestra tempmaximas

                if (f_boton_s2) mostrar_temp_min();         // entra funcion de muestra tempminimas

                if(f_mostrar_temp_ambiente) mostrar_temp_ambiente(); // entra funcion muestra temp ambiente

        } //------ FOR(ever)

}//----- mainend

//-------------------------------------------------------------------------------------//

//------------------------------ Declaracion de interrupciones------------------------------//
interrupt 29 void RTC_ISR(void)
{
        RTCSC_RTIF = 1;         // limpio la bandea

        if(mseg != 0) mseg--;
        if (timer_intervalo !=0) timer_intervalo--;
}
```

```
interrupt 26 void ADC_ISR(void)
{
	if(ADCSC1 == (ADC_CANAL8 | 0x80))
		{
			adc_ch8 = ADCR;                                                    // 0x48
			ADCSC1 = ADC_LM35; // se pasa al canal del lm35   //COCO 0 =  conversion no terminada
		}
			//AEIN    1 = Conversion complete interruptenabled

			//ADCO  0 =  conversion no continua

			//ADCH  0100 = seleccionado canal 8
		if(ADCSC1 == (ADC_LM35 | 0x80))
		{
			adc_lm35 = ADCR;
			ADCSC1 = ADC_CANAL8;
		}
}

interrupt 25 void KBI_ISR(void)
{
	int a;

	if(BOTON_MENU==0)
	{
		for(a=0;a<5000;a++);
		if (BOTON_MENU==0) f_menu = 1;
	}

	if(BOTON_S1==0)
	{
		for(a=0;a<3000;a++);
		if(BOTON_S1==0 && BOTON_S2==1 && serie==0)f_boton_s1 =~f_boton_s1;
		if(BOTON_S1==0 && BOTON_S2==0 && serie==0) f_mostrar_temp_ambiente=~f_mostrar_temp_ambiente;
		while(BOTON_S1==0);
		for(a=0;a<3000;a++);
	}

	if(BOTON_S2==0)
	{
		for(a=0;a<3000;a++);
		if(BOTON_S1==1 && BOTON_S2==0  && serie==0)f_boton_s2 =~f_boton_s2;
		if(BOTON_S1==0 && BOTON_S2==0 && serie==0)
f_mostrar_temp_ambiente=~f_mostrar_temp_ambiente;
		while(BOTON_S2==0);
		for(a=0;a<3000;a++);
	}

	KBISC_KBACK = 1;                                                           // limpio la bandera
}


interrupt 23 void int_sci2(void)                    // interrumpe rx sci2
{
	if(timeout_rx2 != 0) f_sci2++;
	SCI2S1=SCI2S1;
	b_sci2[f_sci2]=SCI2D;
	f_rx_sci2 = 1;                                                  // limpia el flag
	timeout_rx2=50;
```

```
//---------------------------------- Declaracion de funciones-----------------------------//

void inicio_puertos(void)
{
                        //PUERTO A
        PTAD = 0;                               // configuro las salidas=0
        PTADD = 0xFF;                   // configuro todos el puerto como salidas
        PTAPE = 0xFF;                   // habilito pullup
        PTASE = 0xFF;

                        //PUERTO B
        PTBD = 0;                               // salidas=0
        PTBDD = 0x1F;                   // configuro salidas
        PTBPE = 0xFF;                   // habilito pullup
        PTBSE = 0xFF;

                        //PUERTO C
        PTCD = 0;                               // salidas=0
        PTCDD = 0xFF;                   // configuro salidas
        PTCPE = 0xFF;                   // habilito pullup
        PTCSE = 0xFF;

                        //PUERTO D
        PTDD = 0;                               // salidas=0
        PTDDD = 0xFB;                   // configuro salidas, PTD2 como entrada
        PTDPE = 0xFF;                   // habilito pullup
        PTDSE = 0xFF;

                        //PUERTO E
        PTED = 0;                               // salidas=0
        PTEDD = 0xFF;                   // configuro salidas
        PTEPE = 0xFF;                   // habilito pullup
        PTESE = 0xFF;

                        //PUERTO F
        PTFD = 0;                               // salidas=0
        PTFDD = 0xFC;                   // configuro salidas/ f0 como entrada
        PTFPE = 0xFF;                   // habilito pullup
        PTFSE = 0xFF;

                        //PUERTO G
        PTGD = 0;                               // salidas=0
        PTGDD = 0xF0;                   // configuro salidas, PTG0/PTG1/PTG2/PTG3 entrada
        PTGPE = 0xFF;                   // habilito pullup
        PTGSE = 0xFF;
}

void inicio_cop_lvi_irq(void)
{
        IRQSC = 0;                                      // IRQ disable
        SOPT1 = 0;                                              // copdisable
        SOPT2 = 0;
        SPMSC1 = 0;                                     // LVD disable
        SPMSC2 = 0;
}

void inicio_rtc(void)
{
        RTCMOD = 0;
        RTCSC = 0x18;                           // RTCLKS=00->1Khz,RTIE=1,RTCPS=0x8->1mseg
}
```

```
void inicio_kbi(void)
{
	KBISC = 0x04;                                                                   // KBACK=1
	KBIPE = 0xC7;                                                   // KBP0=1,KBP1=1,KBP2=1,KBP6=1,KBP7=1
	KBIES = 0x00;                                                   // KBP0 Y KBP1  Fallingedge/lowlevel
	KBISC = 0x06;                                                   // KBACK=1,KBIE=1,KMOD=1
}

void delay_mseg(unsignedint ms)
{
	mseg = ms;
	while(mseg!=0);
}

void inicio_osc(void)
{
	MCGTRM = TRIM_32KHz;
	MCGC2 = 0x40;                                                   // Fref = 31.25KHz; BDIV=1; VDIV=4; R=1;
	MCGC1 = 0x04;                                                   // M = VDIV * 256 ;
	MCGC3 = 0x01;                                                   // FLL = M * fref / R ;Fbus = (FLL/BDIV) / 2
	while(MCGSC_LOCK == 0) ;                                        // espera enganche FLL
}

void inicio_adc(void)
{
	APCTL1 = 0x20;
	APCTL2 = 0x01;
	ADCCFG = 0xF5;                          // 12bit,clock/8,bus_clock/2,low speed,longsample
	ADCSC2 = 0;
	ADCSC1 = 0x48;                          // habilito Int, conversion simple
}


//---------------------------------- Registro termopares ----------------------------------//
void registro_temp (void)
{
//----------------------- Termopar 1 -----------------------------------
	if (PTBD_PTBD0 == 0 || PTBD_PTBD1 == 0 )
		{
			termo=0;
			delay_mseg (20);
			temp_act[0]= medir_temp();
			PTBD = 0x01;
		}

//----------------------- Termopar 2 -----------------------------------

		if (PTBD_PTBD0 == 1 || PTBD_PTBD1 == 0 )
			{
				termo = 1;
				delay_mseg (20);
				temp_act[1]= medir_temp();
				PTBD = 0x02;
			}

//----------------------- Termopar 3 -----------------------------------

		if (PTBD_PTBD0 == 0 || PTBD_PTBD1 == 1 )
			{
				termo = 2;
				delay_mseg (20);
				temp_act[2]= medir_temp();
				PTBD = 0x03;
			}
```

```
//----------------------- Termopar 4 ----------------------------------

                        if (PTBD_PTBD0 == 1 || PTBD_PTBD1 == 1 )
                                {
                                        termo = 3;
                                        delay_mseg (20);
                                        temp_act[3]= medir_temp();
                                        PTBD = 0x04;
                                }

//----------------------- Termopar 5 ----------------------------------

                        if (PTBD_PTBD0 == 0 || PTBD_PTBD1 == 0 || PTBD_PTBD2 == 1 )
                                {
                                        termo = 4;
                                        delay_mseg (20);
                                        temp_act[4]= medir_temp();
                                        PTBD = 0x05;
                                }

//----------------------- Termopar 6 ----------------------------------

                        if (PTBD_PTBD0 == 1 || PTBD_PTBD1 == 0 || PTBD_PTBD2 == 1 )
                                {
                                        termo = 5;
                                        delay_mseg (20);
                                        temp_act[5]= medir_temp();
                                        PTBD = 0x06;
                                }

//----------------------- Termopar 7 ----------------------------------


                        if (PTBD_PTBD0 == 0 || PTBD_PTBD1 == 1 || PTBD_PTBD2 == 1 )
                                {
                                        termo = 6;
                                        delay_mseg (20);
                                        temp_act[6]= medir_temp();
                                        PTBD = 0x07;
                                }

//----------------------- Termopar 8 ----------------------------------

                        if (PTBD_PTBD0 == 1 || PTBD_PTBD1 == 1 || PTBD_PTBD2 == 1)
                                {
                                        termo = 7;
                                        delay_mseg (20);
                                        temp_act[7]= medir_temp();
                                        PTBD = 0x00;
                                }

}
```

```
//------------------------- Medicion de Temperatura (Act, Max, Min )-----------------------//

float medir_temp (void)
{
	float b=0;
	int a;

	for (a=0;a<100;a++) // tomo n cantidad de muestras.
		{
			if (PTFD_PTFD0 != 0)

				b+= (adc_ch8*1.221*0.1);

			else // si no hay termocupla conectada en esa salida hago lo siguente

			b=0; // mando al valor actual cero.

		}
	b=b/100; // saco el promedio de las muestras
	return b;
	}

float medir_temp_max (void)
{
int r=0;

for (r=0;r<8;r++)
	{
		if (temp_act[r]!=0)
		{
			if (temp_act[r]>temp_max[r])
				temp_max[r]=temp_act[r];
		}
		else
			temp_max[r]=0;
	}
}

float medir_temp_min (void)
{
int r=0;

for (r=0;r<8;r++)
	{
		if (temp_min[r]!=0)
		{
			if (temp_act[r]<temp_min[r])
				temp_min[r]=temp_act[r];
		}
		else
			temp_min[r]=350;
	}
}
```

```
float medir_temp_ambiente(void)
{
	float b=0;
	int a;
	for (a=0;a<50;a++) // tomo n cantidad de muestras.
	{
		b+= (adc_lm35*1.221*0.1);
	}
	b=b/50; // saco el promedio de las muestras
	return b;
}

//---------------------------- Mostrar en Display -------------------------------------------//

void mostrar_temp (void)
{
	sprintf(&buffer[0] ,"T1:%.1f	",temp_act[0]); // minimo escribir 10 caracteres
	sprintf(&buffer[10],"T5:%.1f	",temp_act[4]);
	line_lcd4(buffer,1);

	sprintf(&buffer[0] ,"T2:%.1f	",temp_act[1]);
	sprintf(&buffer[10],"T6:%.1f	",temp_act[5]);
	line_lcd4(buffer,2);

	sprintf(&buffer[0] ,"T3:%.1f	",temp_act[2]);
	sprintf(&buffer[10],"T7:%.1f	",temp_act[6]);
	line_lcd4(buffer,3);

	sprintf(&buffer[0] ,"T4:%.1f	",temp_act[3]);
	sprintf(&buffer[10],"T8:%.1f	",temp_act[7]);
   line_lcd4(buffer,4);
}

void mostrar_temp_max (void)
{
	KBIPE = 0x01; // desabilito los otros botones
	delay_mseg (100);
	while (f_boton_s1 )			// si se presiono el boton S2
	{
		sprintf(&buffer[0] ,"T1M:%.1f	",temp_max[0]); // minimo escribir 10 caracteres
		sprintf(&buffer[10],"T5M:%.1f	",temp_max[4]);
		line_lcd4(buffer,1);

		sprintf(&buffer[0] ,"T2M:%.1f	",temp_max[1]);
		sprintf(&buffer[10],"T6M:%.1f	",temp_max[5]);
		line_lcd4(buffer,2);

		sprintf(&buffer[0] ,"T3M:%.1f	",temp_max[2]);
		sprintf(&buffer[10],"T7M:%.1f	",temp_max[6]);
		line_lcd4(buffer,3);

		sprintf(&buffer[0] ,"T4M:%.1f	",temp_max[3]);
		sprintf(&buffer[10],"T8M:%.1f	",temp_max[7]);
		line_lcd4(buffer,4);
	}
	KBIPE = 0xC7;  // habilito los botones otra vez
}
```

```
void mostrar_temp_min (void)
{
	KBIPE = 0x02;		// desabilito los otros botones
	delay_mseg (100);
	while (f_boton_s2)
	{
		int r=0;

		for (r=0;r<8;r++)
		{
			if (temp_min[r]== 350)temp_min[r]=0;
		}
		sprintf(&buffer[0] ,"T1m:%.1f	",temp_min[0]); // minimo escribir 10 caracteres
		sprintf(&buffer[10],"T5m:%.1f	",temp_min[4]);
		line_lcd4(buffer,1);

		sprintf(&buffer[0] ,"T2m:%.1f	",temp_min[1]);
		sprintf(&buffer[10],"T6m:%.1f	",temp_min[5]);
		line_lcd4(buffer,2);

		sprintf(&buffer[0] ,"T3m:%.1f	",temp_min[2]);
		sprintf(&buffer[10],"T7m:%.1f	",temp_min[6]);
		line_lcd4(buffer,3);

		sprintf(&buffer[0] ,"T4m:%.1f	",temp_min[3]);
		sprintf(&buffer[10],"T8m:%.1f	",temp_min[7]);
		line_lcd4(buffer,4);
	}
	KBIPE = 0xC7;		// habilito los botones otra vez
}

void mostrar_temp_ambiente(void)
{
	int a=5;
	KBIPE = 0x00;
	while (a!=0)
	{
		temp_ambiente = medir_temp_ambiente();
		delay_mseg (100);
		line_lcd4("		",1);
		line_lcd4("  Temp. Ambiente:   ",2);
		line_lcd4("		",4);
		sprintf(&buffer[0] ,"	%.1f	",temp_ambiente);
		line_lcd4(buffer,3);
		a--;
	}
	delay_mseg (100);
	f_mostrar_temp_ambiente=~f_mostrar_temp_ambiente;
	KBIPE = 0xC7;
}


//------------------------------ Configcomunicacion Serie ------------------------------------//

void inicio_sci2(unsignedintbr)
{
	unsigned char a;
	SCI2S1 = SCI2S1;
	a = SCI2D;
	SCI2BD = (500000/br);	// BaudRate = BUSCLK / ([SBR12:SBR0] x 16)=(8M/16)/br
	SCI2C1 = 0x00;			// Loopmodedisabled, disable SCI, Tx output notinverted,
	SCI2C2 = 0x2C;			// Enable SCI receiveinterrupts, Enabletransmitter and
	SCI2C3 = 0x00;			// Disableall error interrupts
}
```

```
void sci2_byte(unsigned char sci_tx)
{
  SCI2D = sci_tx;                                                    // cargo el buffer del transmisor
while(SCI2S1_TC==0);                                                 // espero completar la transmision
}
void sci2_string(unsignedchar *dato)
{
while(*dato != '\0') sci2_byte(*(dato++));                  // envia datos por SCI
}


//-------------------------------------------------------------------------------------------//

//---------------------------------- Envio datos RS-232 ---------------------------------------//

unsigned char enviar_datos (void)
{
        while (v)
        {
                sci2_byte(0x0D); //          fin y enter por si hay basura
                sci2_byte(0x0A);
                sprintf(&output[0],"T1,T2,T3,T4,T5,T6,T7,T8 ");         // envia encabezado
                sci2_string(output);
                sci2_byte(0x0D);
                sci2_byte(0x0A);

                v=0;
        }

        sprintf(&output[0],"%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f",temp_act[0],temp_act[1],temp_act[2],temp_act[3],te
mp_act[4],temp_act[5],temp_act[6],temp_act[7]);

        sci2_string(output);                                         // envia cadena
        sci2_byte(0x0D);
        sci2_byte(0x0A);
        delay_mseg(10);

}

unsigned char enviar_tempmax (void)
{
        delay_mseg(2000);
        sprintf(&output[0],"T1M,T2M,T3M,T4M,T5M,T6M,T7M,T8M ");// envia encabezado
        sci2_string(output);
        sci2_byte(0x0D);
        sci2_byte(0x0A);
        sprintf(&output[0],"%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f",temp_max[0],temp_max[1],temp_max[2],temp_max
[3],temp_max[4],temp_max[5],temp_max[6],temp_max[7]);
        sci2_string(output);                                         // envia cadena
        sci2_byte(0x0D);
        sci2_byte(0x0A);
}
```

```
Unsigned char enviar_tempmin (void)
{
        int r=0;
        for (r=0;r<8;r++)
                {
                        if (temp_min[r]== 350)temp_min[r]=0;
                }
        delay_mseg(2000);
        sprintf(&output[0],"T1m,T2m,T3m,T4m,T5m,T6m,T7m,T8m ");// envia encabezado
        sci2_string(output);
        sci2_byte(0x0D);
        sci2_byte(0x0A);
        sprintf(&output[0],"%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f",temp_min[0],temp_min[1],temp_min[2],temp_min[3]
,temp_min[4],temp_min[5],temp_min[6],temp_min[7]);
        sci2_string(output);                                    // envia cadena
        sci2_byte(0x0D);
        sci2_byte(0x0A);
        delay_mseg(10);
}

unsigned char enviar_tempambiente (void)
{
        delay_mseg(2000);
        sprintf(&output[0],"Temperatura ambiente: ");           // envia encabezado
        sci2_string(output);
        sci2_byte(0x0D);
        sci2_byte(0x0A);
        sprintf(&output[0],"%.1f",temp_ambiente);
   sci2_string(output);                                         // envia cadena
        sci2_byte(0x0D);
        sci2_byte(0x0A);
}

//-------------------------------------------------------------------------------------------//
```

```
void menu (void)
{
	KBIPE = 0xC4;		// desabilito los botones de temperaturas max y min

	while (f_menu==1 &&flag_set_point==0)	// entra al presionar
	{
		line_lcd4("                ",1);	//muestra este menu en display
		line_lcd4("      RS-232      ",2);	// hasta presionar SI o NO
		line_lcd4("                ",3);
		line_lcd4("  SI          NO  ",4);

		if(BOTON_UP==0)			// opcion de SI
		{
			delay_mseg(50);
			if(BOTON_UP==0)
			{
				flag=1;
				confi_rs232();
			}
		}

		if(BOTON_DOWN==0)		// opcion de NO
		{
			delay_mseg(50);
			if(BOTON_DOWN==0)
			{
				flag_set_point=0;
				f_menu=0;
				serie=0;
				LED_Y = LED_OFF;
				KBIPE = 0xC7;
			}
		}
	}
}

//-------------------------------------------------------------------------------------------//

void confi_rs232(void)
{
	int a;
		while (flag==1 &&flag_set_point==0)
		{
			f=1;
			contador=5;
			line_lcd4("                ",1);
			line_lcd4("   Congifuracion   ",2);
			line_lcd4("      RS-232      ",3);
			line_lcd4("                ",4);
			delay_mseg(1500);
```

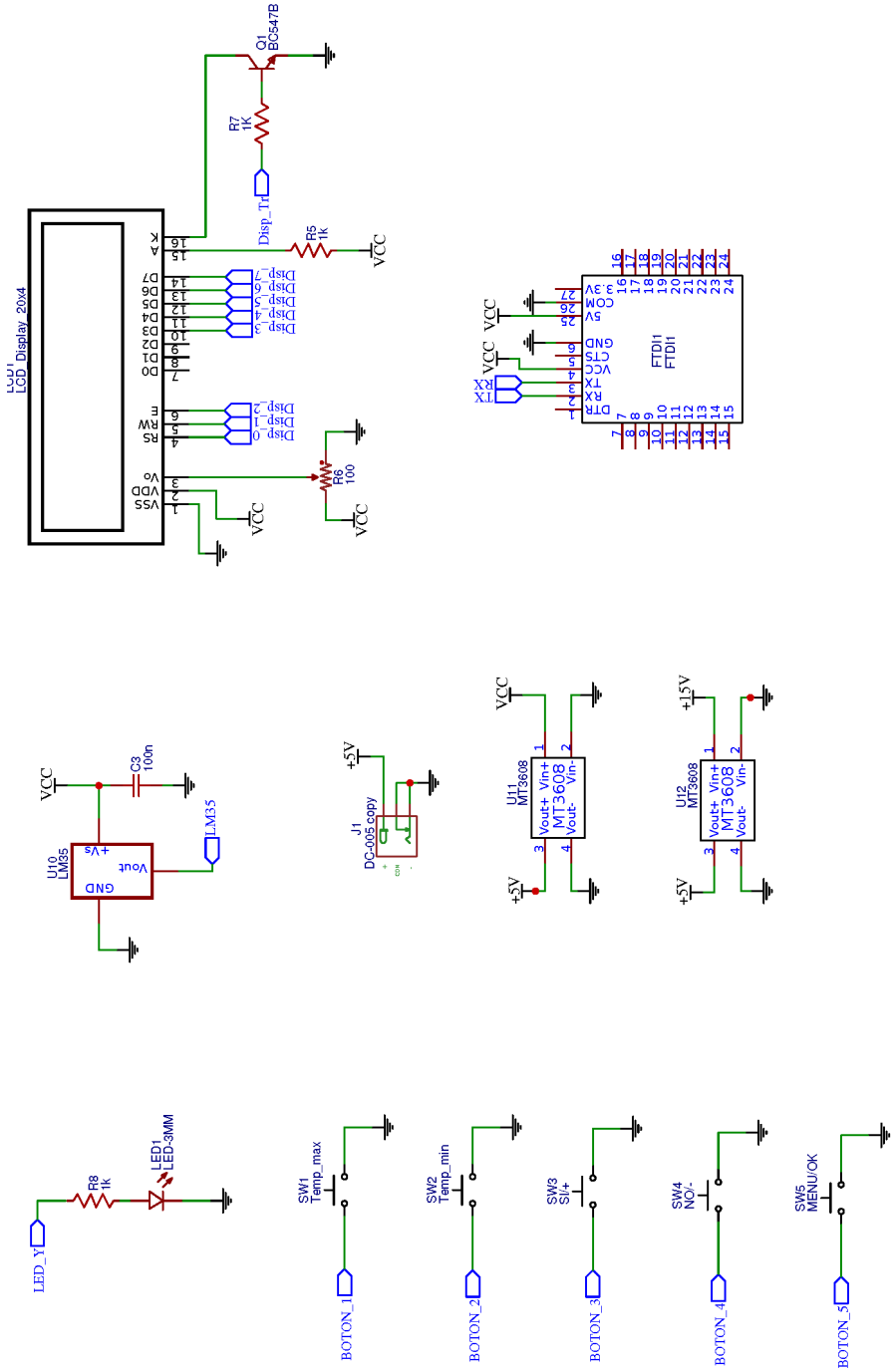
```
//---------------------------- INTERVALO ----------------------------------------------

                        while(BOTON_MENU!=0 && f==1)
                        {
                                line_lcd4("Intervalo medicion: ",1);
                                line_lcd4("                    ",3);
                                line_lcd4("                    ",4);

                                if(BOTON_UP==0)
                                {
                                        delay_mseg(20);
                                        contador= contador +5;
                                        if(contador == 25)contador = contador+5;
                                }


                                if(BOTON_DOWN==0)
                                {
                                        delay_mseg(20);
                                        contador=contador - 5;
                                        if(contador == 25)contador =contador-5;
                                }

                                if(contador<=0) contador=30;
                                if(contador>30) contador=5;

                                sprintf(&buffer[0],"       %i min     ",contador);
                                line_lcd4(buffer,2);

                        }


                        if(BOTON_MENU==0 && f==1)
                        {
                                delay_mseg(100);
                                line_lcd4("                    ",3);
                                line_lcd4("                    ",4);
                                line_lcd4("   ud. selecciono:  ",1);
                                sprintf(&buffer[0],"       %i min      ",contador);
                                line_lcd4(buffer,2);
                                disp_intervalo=contador;
                                intervalo=contador*60000;
                                timer_intervalo = intervalo;
                                delay_mseg(1500);
                                flag_set_point=1;
                        }
                }

//-------------------------------------------------------------------------------
```

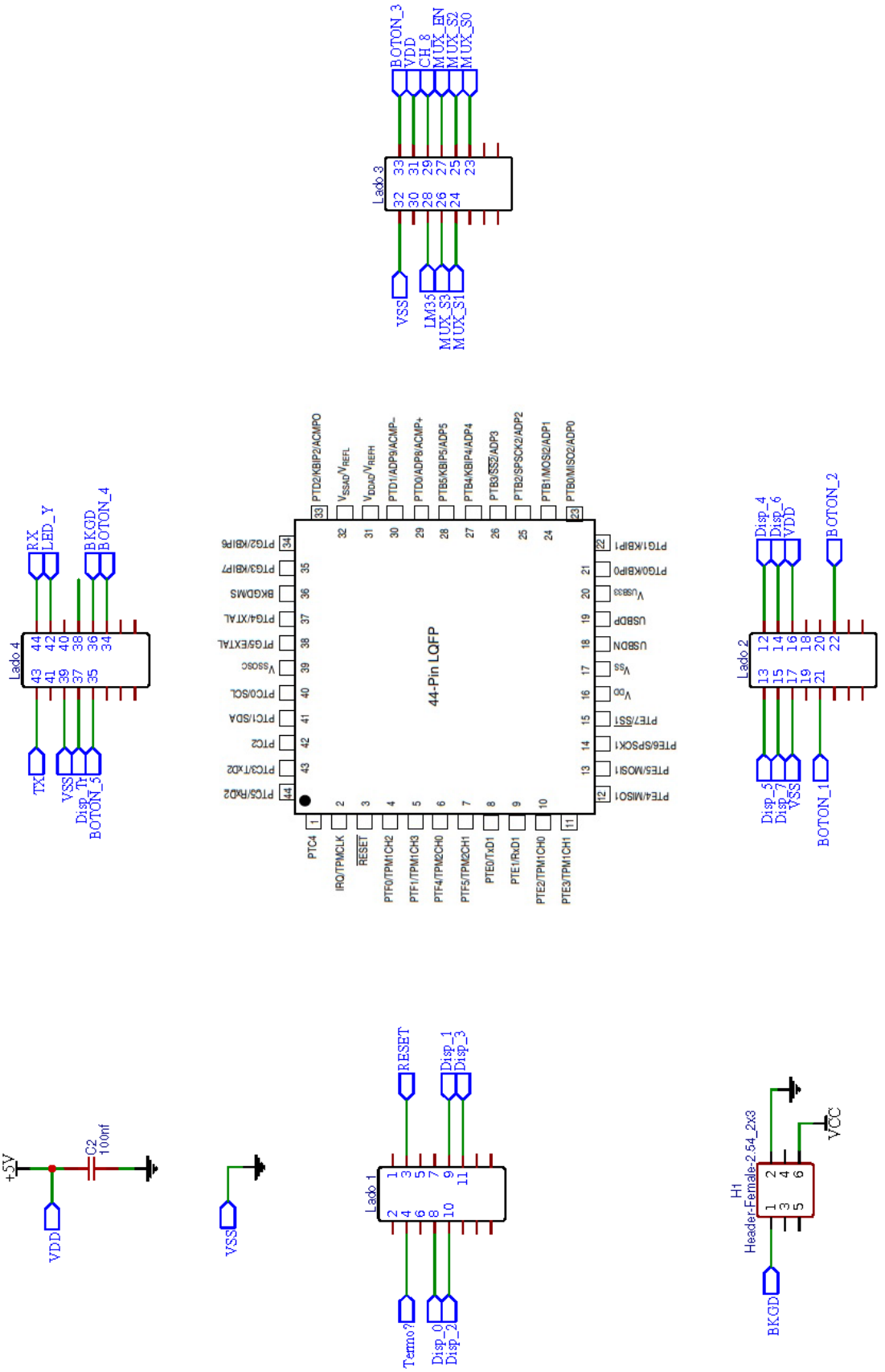
```
//---------------------------------- TIEMPO TOTAL ---------------------------------------
        while(flag==1 &&flag_set_point==1)
        {
                f=1;
                contador=1;

                while(BOTON_MENU!=0 && f==1)
                {
                        line_lcd4("    Tiempo total    ",1);
                        line_lcd4("                    ",3);
                        line_lcd4("                    ",4);

                        if(BOTON_UP==0)
                        {
                                delay_mseg(20);
                                contador++;
                        }

                        if(BOTON_DOWN==0)
                        {
                                delay_mseg(20);
                                contador--;
                        }

                        if(contador>5) contador=1;
                        if(contador<=0) contador=5;
                        sprintf(&buffer[0],"       %i HS        ",contador);
                        line_lcd4(buffer,2);
                }

                if(BOTON_MENU==0 && f==1)
                {
                        delay_mseg(100);
                        line_lcd4("  ud. selecciono:   ",1);
                        line_lcd4("                    ",3);
                        line_lcd4("                    ",4);
                        sprintf(&buffer[0],"       %i HS        ",contador);
                        line_lcd4(buffer,2);
                        disp_t_registro=contador;
                        t_registro=contador;
                        delay_mseg(1500);
                }

//-----------------------------------------------------------------------------------------
```

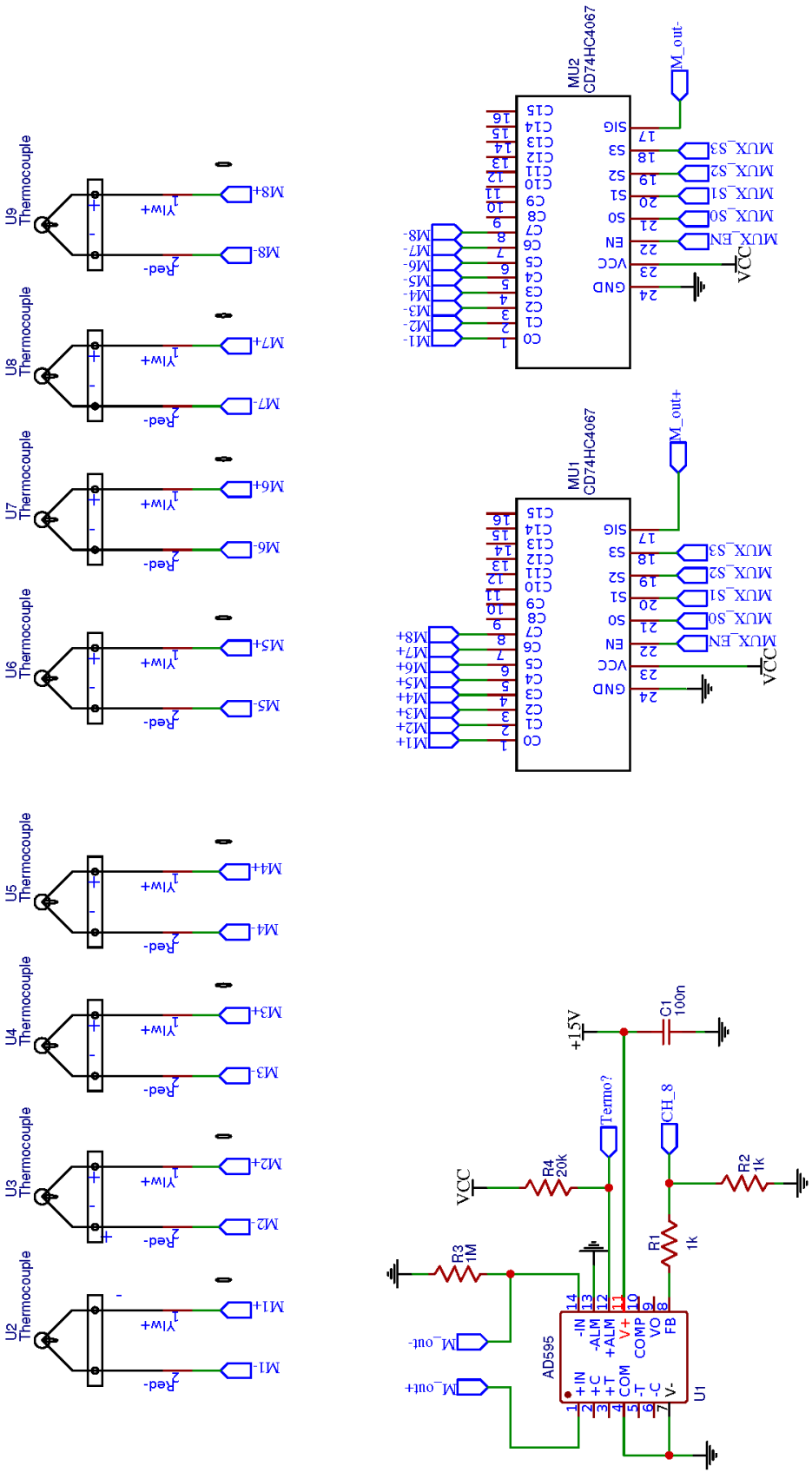
```
//------------------------------ Displayconfig terminada---------------------------------

                        line_lcd4("                    ",1);
                        line_lcd4("    Configuracion   ",2);
                        line_lcd4("         OK         ",3);

                        sprintf(&buffer[0],"%i min      ",disp_intervalo);
                        line_lcd4(buffer,4);
                        sprintf(&buffer[10],"      %i HS",disp_t_registro);
                        line_lcd4(buffer,4);
                        delay_mseg(2000);
                        t_registro=(disp_t_registro*60/disp_intervalo);
                        flag_set_point=0;
                        f_menu=0;
                        flag=0;
                        serie=1;
                        v=1;
            }

  KBIPE = 0xC7;                 // habilito los botones otra vez
}

//-------------------------------------------------------------------------------------------//
```

ANEXO III:

Esquemático:




BOTON_3
VDD
CH_8
MUX_EN
MUX_S2
MUX_S0
Lado 3
VSS
LM35
MUX_S3
MUX_S1
RX
LED_Y
BKGD
BOTON_4
Lado 4
TX
VSS
Disp_7
BOTON_5
PTD2/KBIP2/ACMPO
PTD1/ADP9/ACMP-
PTD0/ADP8/ACMP+
PTB5/KBIP5/ADP5
PTB4/KBIP4/ADP4
PTB3/SS2/ADP3
PTB2/SPSCK2/ADP2
PTB1/MOSI2/ADP1
PTB0/MISO2/ADP0
PTG1/KBIP1
PTG0/KBIP0
USBDP
USBDN
PTE7/SS1
PTE6/SPSCK1
PTE5/MOSI1
PTE4/MISO1
PTG2/KBIP6
PTG3/KBIP7
BKGDMS
PTG4/XTAL
PTG5/EXTAL
PTC0/SCL
PTC1/SDA
PTC2
PTC3/TxD2
PTC5/RxD2
44-Pin LQFP
PTC4
IRQ/TPMCLK
RESET
PTF0/TPM1CH2
PTF1/TPM1CH3
PTF4/TPM2CH0
PTF5/TPM2CH1
PTE0/TxD1
PTE1/RxD1
PTE2/TPM1CH0
PTE3/TPM1CH1
Lado 2
Disp_5
Disp_7
VSS
BOTON_1
Disp_4
Disp_6
VDD
BOTON_2
+5V
C2
100nf
VDD
VSS
Lado 1
Termo?
Disp_0
Disp_2
RESET
Disp_1
Disp_3
H1
Header-Female-2.54_2x3
BKGD
VCC


U9 Thermocouple
U8 Thermocouple
U7 Thermocouple
U6 Thermocouple
U5 Thermocouple
U4 Thermocouple
U3 Thermocouple
U2 Thermocouple
Ylw+
Red-
M8+
M8-
M7+
M7-
M6+
M6-
M5+
M5-
M4+
M4
M3+
M3
M2+
M2-
M1+
M1-
MU2 CD74HC4067
MU1 CD74HC4067
M_out-
M_out+
SIG
S3
S2
S1
S0
EN
VCC
GND
MUX_S3
MUX_S2
MUX_S1
MUX_S0
MUX_EN
+15V
C1 100n
R4 20k
R2 1k
R1 1k
R3 1M
Termo?
CH_8
AD595
U1