UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

TECNICAS DIGITALES 2 2020

MEDIDOR Y CONTROLADOR DE PRESION DE AIRE

INFORME

INDICE

Descripción del proyecto:

Se desea realizar un dispositivo que sirva de interface para un sensor analógico de presión. El sensor en cuestión es un sensor de presión absoluta que trabaja sobre una línea de aire comprimido, la salida de dicho sensor es del tipo 4-20mA y el rango te trabajo es de 0 a 10bar.
Este dispositivo tiene como objetivo mantener la línea de aire comprimido en un rango de presión predefinido por el usuario, y en caso de no encontrarse en dicho límite actúa como dispositivo de seguridad liberando la presión en la línea y haciendo sonar una alarma.
El núcleo de este proyecto es un Cortex M3 de la familia de STM el cual lleva implementado un sistema de control del estilo TDS el cual nos asegura el tiempo de respuesta en caso de presentarse una falla.
Por ser requerimiento de otra materia, se conectaron todos los periféricos a una línea de datos I2C. Estos periféricos son:

- Teclado Capacitivo (IQS316)
- Display LCD de 16 Caracteres 2 líneas con conversor I2C (PCF8754)
- Conversor Analógico Digital (ADS1115)
- Memoria EEPROM para parámetros de configuración (24LC16)
- Dos salidas a relé con optoacoplador de protección.

Desarrollo en Kicad

Se optó por realizar dos PCB, uno para el teclado con el display y otro para el microcontrolador y los demás periféricos. Se desarrolló de esta manera ya que el teclado capacitivo necesitar ser implementado en el PCB y aplicado directamente sobre el panel frontal. Entonces, se hizo un PCB denominado PanelFrontal el cual lleva los pads del teclado capacitivo en la cara BOTTOM del PCB, se ubicaron los componentes en la cara TOP y se limitó el uso a solo componentes de montaje superficial, es así que al no tener componentes “through hole” no hay nada que sobresalga del lado BOTTOM y es así que el PCB puede ser apoyado directamente sobre el frente del gabinete. En el PCB denominado PlacaBase se implementó el micro-controlador y todos los periféricos ya que estos requieren varios componentes “through hole” y además de eso no hay una uniformidad de alturas lo cual complicaría el desarrollo en una sola placa junto al teclado.

Calculo de los componentes:

Fuente de Alimentación:

Nos vamos a encontrar con dos tipos de alimentación, el microcontrolador, la memoria EEPROM y el ADC se alimentan con 3,3V mientras que los demás dispositivos se alimentan con 5V. Es por esto que se decidió alimentar el dispositivo con una fuente externa de 5V DC 1A y se implementó en la placa base una fuente de alimentación de 3.3v basada en un LD1117 con sus respectivos capacitores.

Capacitores de los cristales:

Para seleccionar estos capacitores vamos a referirnos a la hoja de datos de los cristales. Para el cristal de 8 MHz (ABM7-8.000MHZ-D2Y-T) vemos que la hoja de datos nos dice que capacitancia de carga es de 18 pF por ende se conectan dos capacitores de 18 pF en los pines del cristal a masa. Para el cristal de 32.768 KHz (ABS25-32.768KHZ-T) la hoja de datos nos dice que la capacitancia de carga es de 12.5 pF, en este caso se selecciona un capacitor de 12pF para conectar en los pines del

cristal. Se diseña el circuito impreso de tal manera que ambas pistas que conectan al microcontrolador con el cristal tengan el mismo largo y la misma geometría.

Leds internos de indicación:

Los leds se utilizaron los LTST-C171KRKT y LTST-C170KGKT ambos son excitados por una tensión de 3.3v ya que uno está conectado a una salida del microcontrolador y el otro directamente a los 3.3v de alimentación. Según la hoja de datos observamos que típicamente tienen 2v de tensión en directa. Se decidió usar una resistencia limitadora de 100 Ohms lo cual nos limitaría la corriente del diodo en 13 mA. Lo cual está por debajo de la corriente máxima de 30ma que acusa el fabricante.

$$I_F = \frac{3.3v - 2v}{100\ \Omega} = 13\ mA$$

Resistencias de los Optoacopladores:

Si nos referimos al datasheet de los optoacopladores PC817 vemos que la tensión en directa es de 1.2v mientras que la corriente es de 20 mA. En base a estos valores se calculó la resistencia limitadora.

$$R = \frac{3.3v\ -\ 1.2v}{20\ mA} = 105\ \Omega$$

Se adoptó una resistencia de 100 ohm siendo el valor comercial más cercano.

Corriente de base en los transistores de los relés:

Se eligió los transistores BC846B para activar los relés. Estos transistores tienen una resistencia de base de 680 ohms. Si suponemos que los optoacopladores tiene la resistencia adecuada para su funcionamiento podemos calcular la corriente de base según la siguiente formula:

$$I_B = \frac{5V - V_{BE}}{680\Omega} = \frac{5V - 0.7v}{680\Omega} = 6.3\ mA$$

Teniendo en cuenta que el relé consume 80 mA, esta corriente nos asegura que el transistor se encuentra saturado con una tensión VCE menor a 0,2v

Resistencia de entrada del ADC:

El sensor de presión con el que vamos a trabajar posee una salida de 4-20mA donde 4ma representa 0bar y 20mA 10bar. Es por esto que vamos a ajustar la resistencia para que esta corriente nos produzca una caída de tensión que podamos medir con el ADC. Se utilizó una resistencia de 100 ohms la cual se la eligió con una tolerancia de 0,1% ya que es fundamental que este valor de resistencia sea preciso ya que de ella depende la precisión de nuestra medición. Con este valor, la tensión en la entrada del ADC en el rango mínimo será de 0.4v y en el rango máximo de 2v. La tensión máxima admisible en la entrada del ADC es de 3.3v. Con esta configuración el ADC además de permitirnos medir el valor arrojado por el sensor nos da un margen como para medir si la entrada de corriente esta fuera de rango. El ADC tiene una resolución de 16bits, por ende tenemos una resolución de 3,3v / 2^16 = 50 µv/div, siendo para el rango mínimo de 0,4v / (50 µv/div) = 7943 y para el rango máximo 2v / (50 µv/div) = 39718

Direcciones del Bus I2C:

Cada periférico tiene su propia dirección I2C para que todos puedan funcionar en el mismo bus. Estas especificaciones las obtenemos de las hojas de datos de los componentes. En general la mayoría tiene al menos un pin que podemos conectar a GND o Vcc para terminar de definir la dirección.

Para el caso de ADC el pin de configuración de dirección se conectó a GND asignándole una dirección “1001000x” (0x50h). En la memoria EEPROM los pines de dirección no se encuentran conectados ya que este tamaño de memoria no los implementa. La dirección I2C se usa en parte para direccionar internamente los bloques de memoria, es por esto que para la memoria se reservan las direcciones 0x60h hasta la 0x6Fh. El conversor I2C para el display LCD se le cablearon sus 3 pines de dirección a masa fijándole una dirección I2C de 0x70h. Para el integrado que controla el teclado capacitivo su pin de dirección se encuentra conectado a masa asignándole una dirección I2C de 0xE1h.

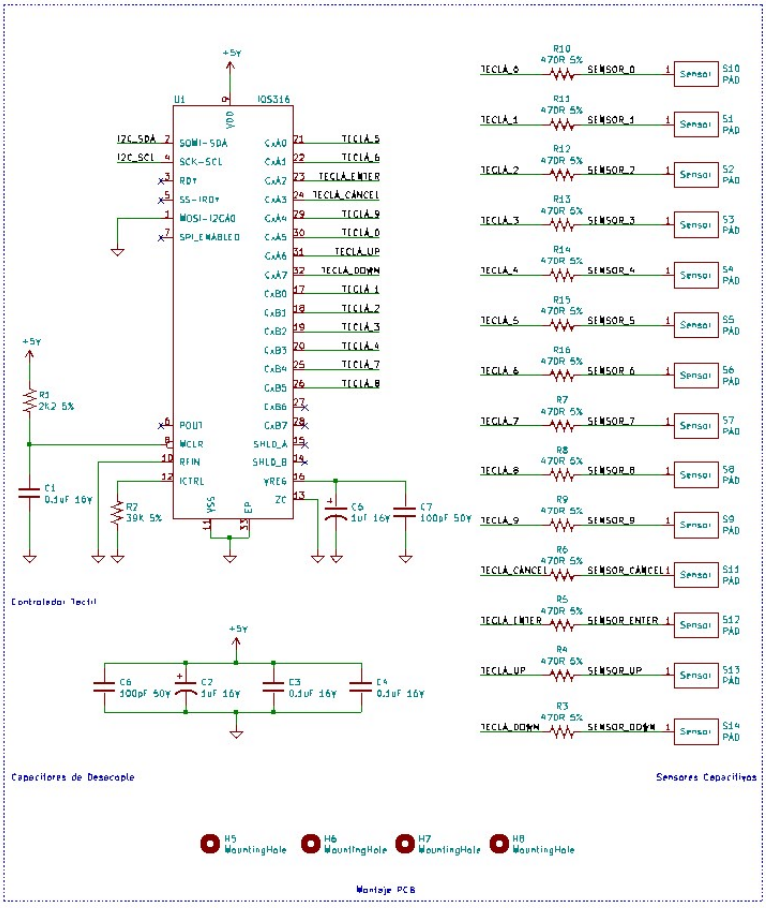
De esta manera, todos los periféricos nos quedan con direcciones distintas posibilitando la conexión al mismo bus I2C.

Consumo total del Dispositivo:

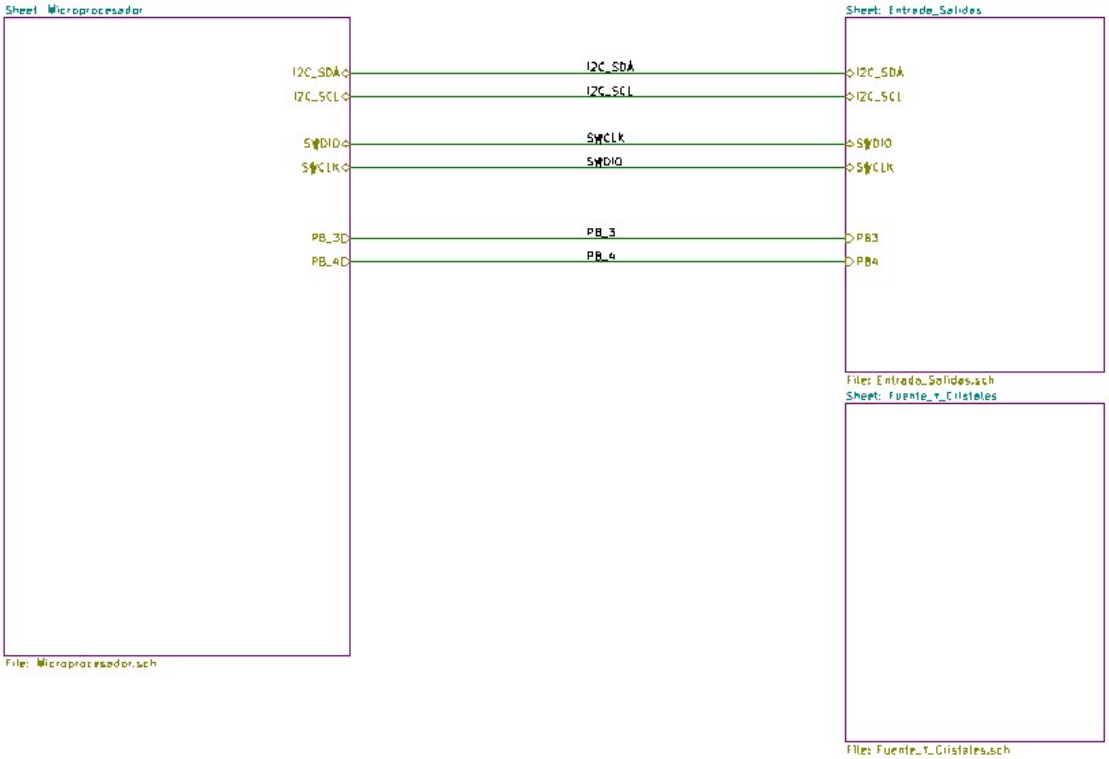
Se analizaron los consumos máximos que podría llegar a tener el dispositivo así poder calcular la fuente de alimentación y el ancho de las pistas. Se analizaron solo los componentes más relevantes. Cada relé consume 80mA estando activados, el LCD tiene un consumo relevante el backligth que según la hoja de datos alcanza los 156 mA como máximo y podríamos sumarle algunos miliamperes mas por los demás componentes lo cual nos daría aproximadamente 330 mA sobre la línea de 5v.

Diagrama Esquemático:

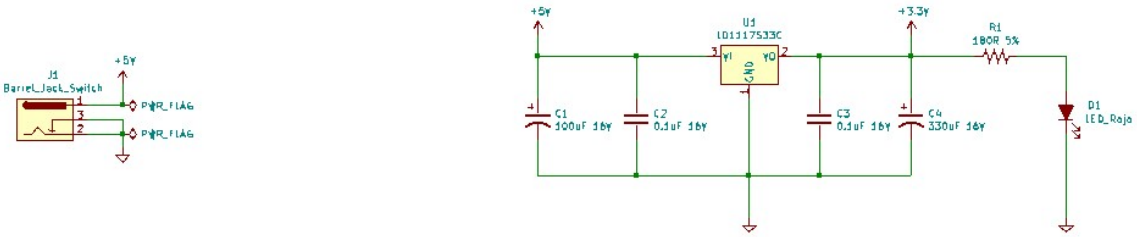
Panel Frontal:



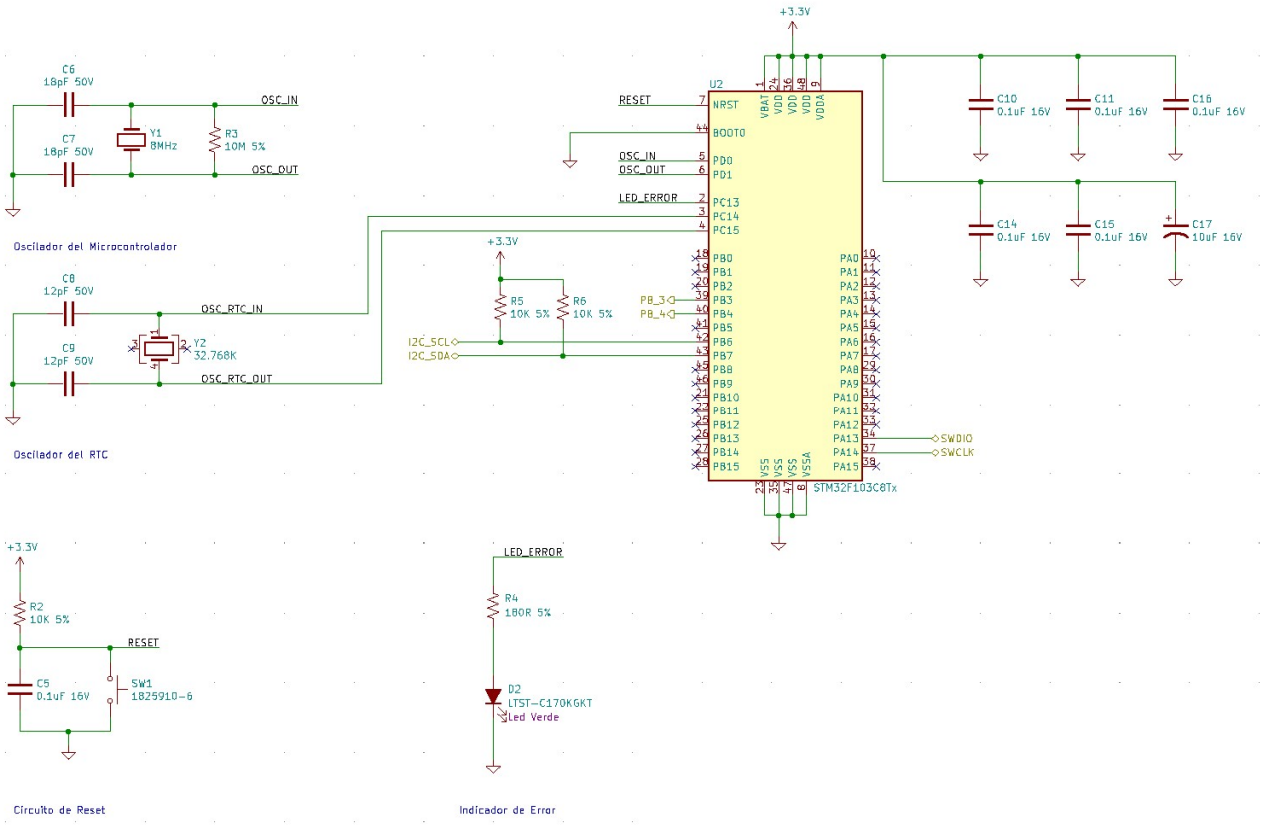
Placa Base:



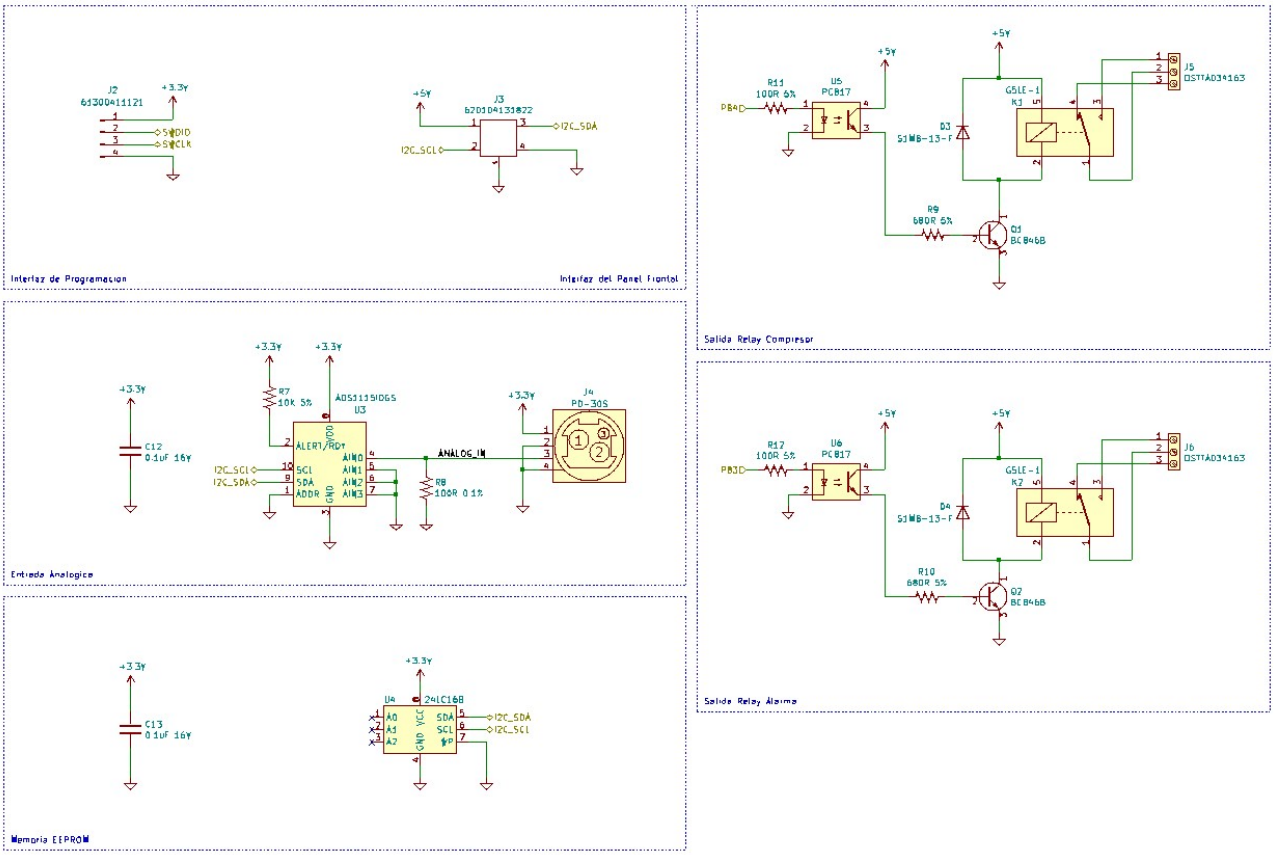
Hoja jerárquica principal.



Fuente de alimentación.



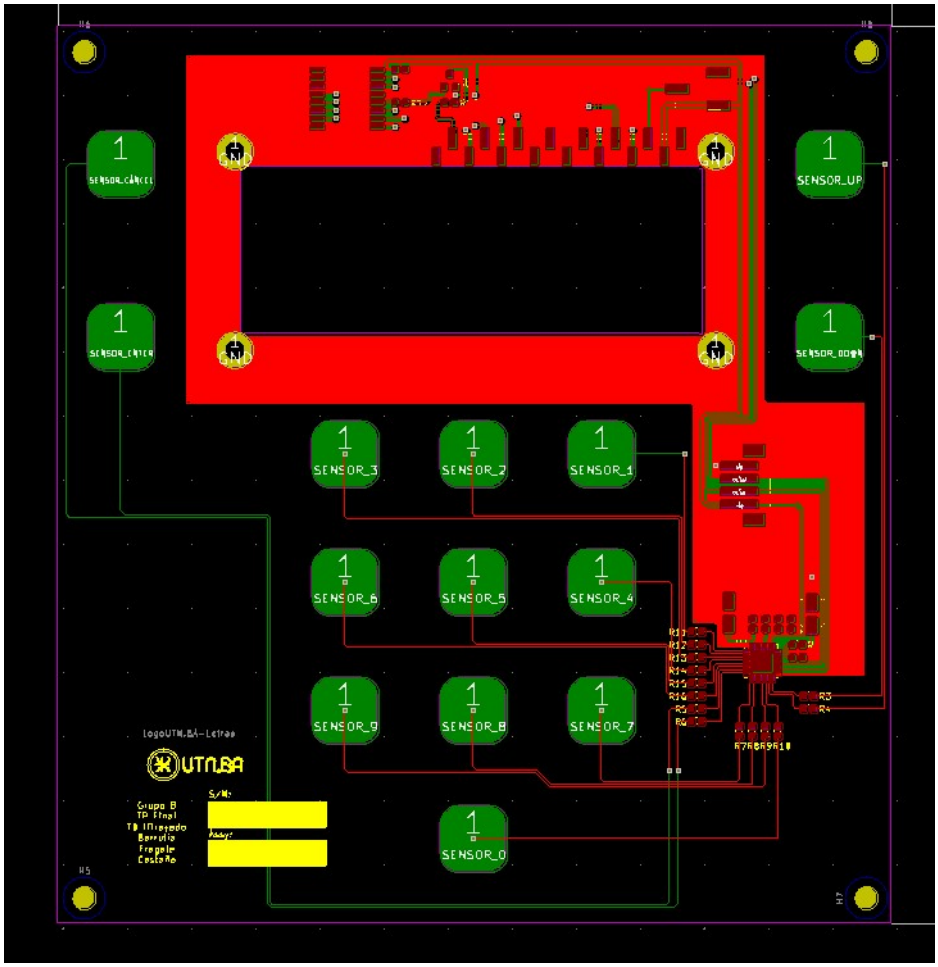
Esquemático del microcontrolador.



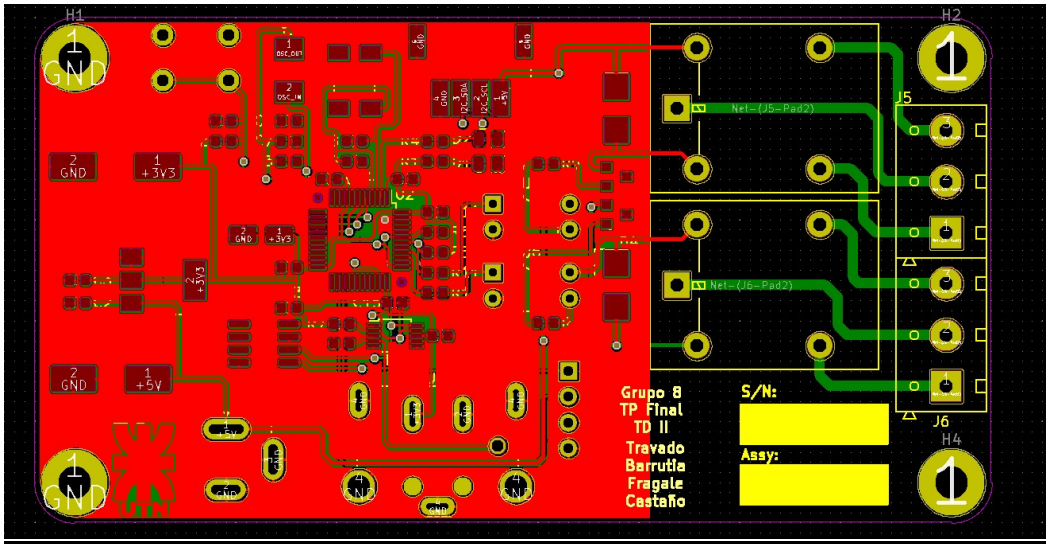
Esquemático de las entradas y salidas.

Circuito Impreso:

Panel Frontal:

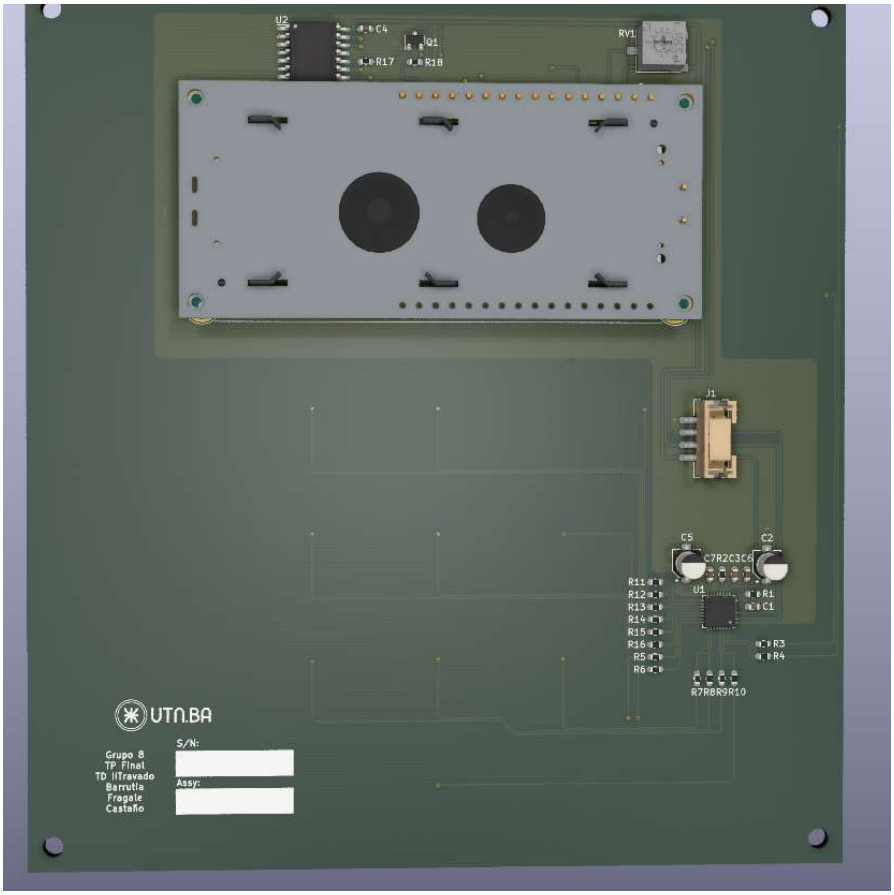


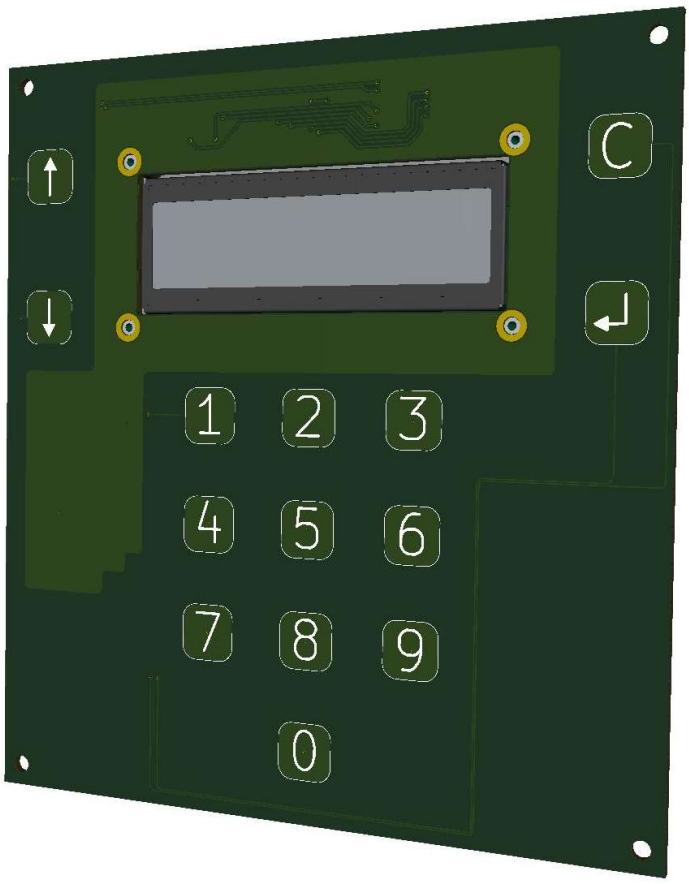
Placa base:



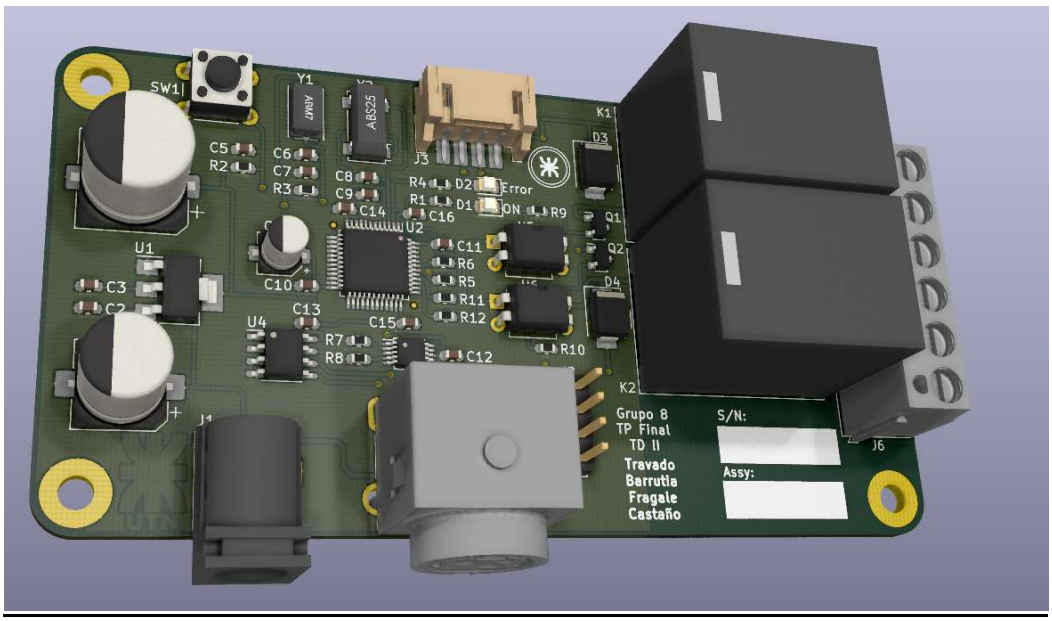
Vista 3D de la placa terminada:

Panel Frontal:

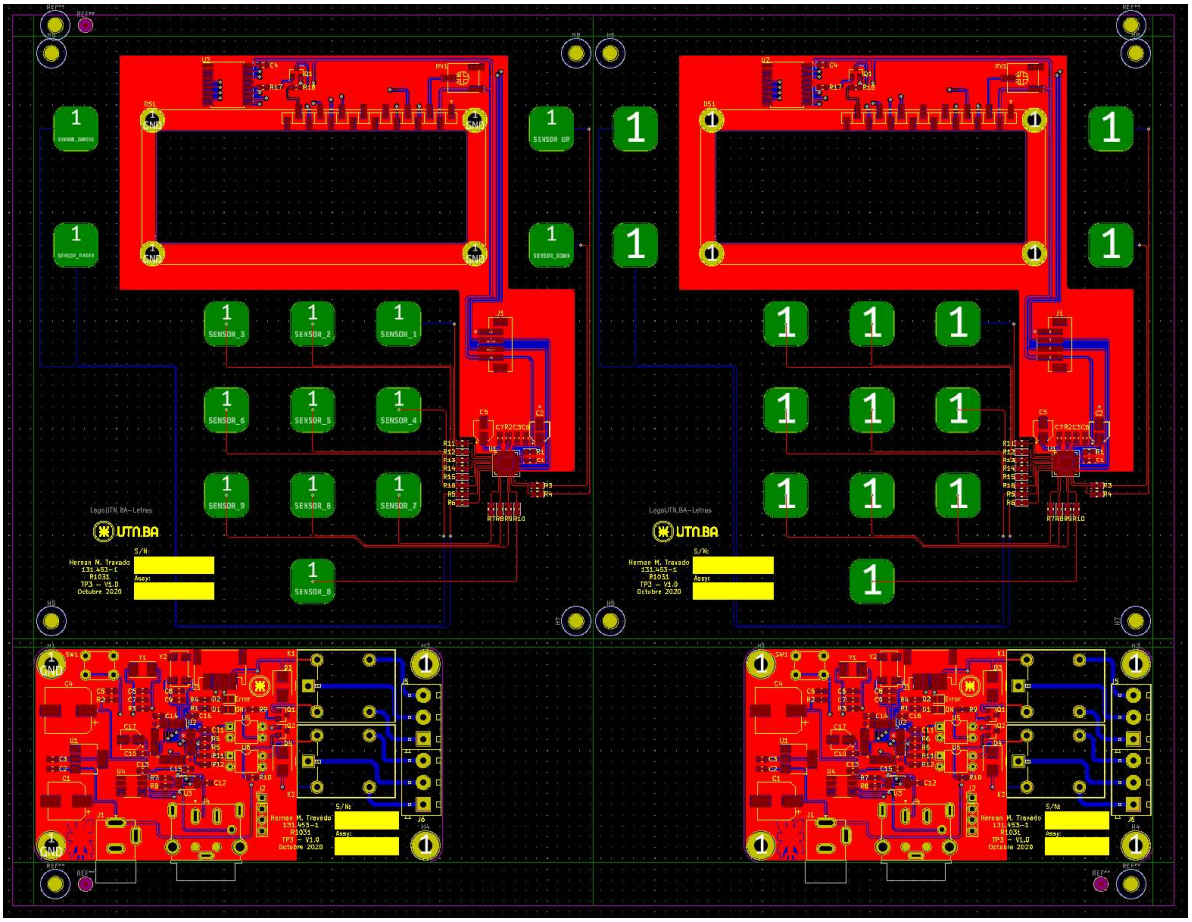



C
1
2
3
4
5
6
7
8
9
0

Placa Base:


SW1
Y1
J3
K1
D3
Error
ON
Q1
Q2
D4
U1
U2
U4
K2
J4
J6
Grupo 8
TP Final
TD II
Travado
Barrutia
Fragale
Castaño
S/N:
Assy:

Panelizado del PCB:



Lista de Materiales (BOM):

Cant.	Valor	Fabricante	Numero de Parte
	Panel Frontal		
3	0.1uF 16V	Samsung Electro-Mechanics	CL10B104KO8NNNC
2	1uF 16V	Panasonic Electronic Components	EEE-FC1V1R0R
2	100pF 50V	Samsung Electro-Mechanics	CL10C101JB8NNNC
1	WC1602A	Newhaven Display Intl	NHD-0216K1Z-FL-YBW
4	MountingHole_Pad	Wurth Elektronik	9774070243R
1	620304124022	Wurth Elektronik	620304124022
1	BC857B	ON Semiconductor	BC857BLT1G
1	2K2 5%	Yageo	RC0603JR-072K2L
1	39K 5%	Yageo	RC0603JR-0739KL
14	470R 5%	Yageo	RC0603JR-07100RL
1	680R 5%	Yageo	RC0603JR-07680RL
1	10R 5%	Yageo	RC0603JR-0710RL
1	10K 10%	Bourns Inc.	3361P-1-103GLF
14	PAD		
1	IQS316	Azoteq (Pty) Ltd.	IQS316-0-QFR

1	PCF8574A	NXP USA Inc.	PCF8574AT/3,518
	Placa Base		
1	100uF 16V	Panasonic Electronic Components	EEE-HB1C101AP
10	0.1uF 16V	Samsung Electro-Mechanics	CL10B104KO8NNNC
1	330uF 16V	Panasonic Electronic Components	EEE-HB1C331AP
2	18pF 50V	Samsung Electro-Mechanics	CL10C180JB8NNNC
2	12pF 50V	Samsung Electro-Mechanics	CL10C120JB8NNNC
1	10uF 16V	Panasonic Electronic Components	EEE-HB1C100AR
1	LTST-C171KRKT	Lite-On Inc.	LTST-C171KRKT
1	LTST-C170KGKT	Lite-On Inc.	LTST-C170KGKT
2	S1MB-13-F	Diodes Incorporated	S1MB-13-F
1	PJ-002A	CUI Devices	PJ-002A
1	61300411121	Wurth Elektronik	61300411121
1	620104131822	Wurth Elektronik	620104131822
1	PD-30S	CUI Devices	PD-30S
2	OSTTA034163	On Shore Technology Inc.	OSTTA034163
2	G5LE-1	Omron Electronics Inc-EMC Div	G5LE-14 DC5
2	BC846B	ON Semiconductor	BC846BLT3G
2	180R 5%	Yageo	RC0603JR-07180RL
4	10K 5%	Yageo	AC0603JR-0710KL
1	10M 5%	Yageo	RC0603JR-0710ML
1	100R 0.1%	Yageo	RT0603BRD07100RL
2	680R 5%	Yageo	RC0603JR-07680RL
2	100R 5%	Yageo	RC0603JR-07100RL
1	1825910-6	TE Connectivity ALCOSWITCH Switches	1825910-6
1	LD1117S33C	STMicroelectronics	LD1117S33CTR
1	STM32F103C8Tx	STMicroelectronics	STM32F103C8T6TR
1	ADS1115IDGS	Texas Instruments	ADS1115IDGSR
1	24LC16B	Microchip Technology	24LC16BT-I/SN
2	PC817	SHARP/Socle Technology	PC81710NIP1B
1	8MHz	Abracon LLC	ABM7-8.000MHZ-D2Y-T
1	32.768K	Abracon LLC	ABS25-32.768KHZ-T

Link a Octopart:

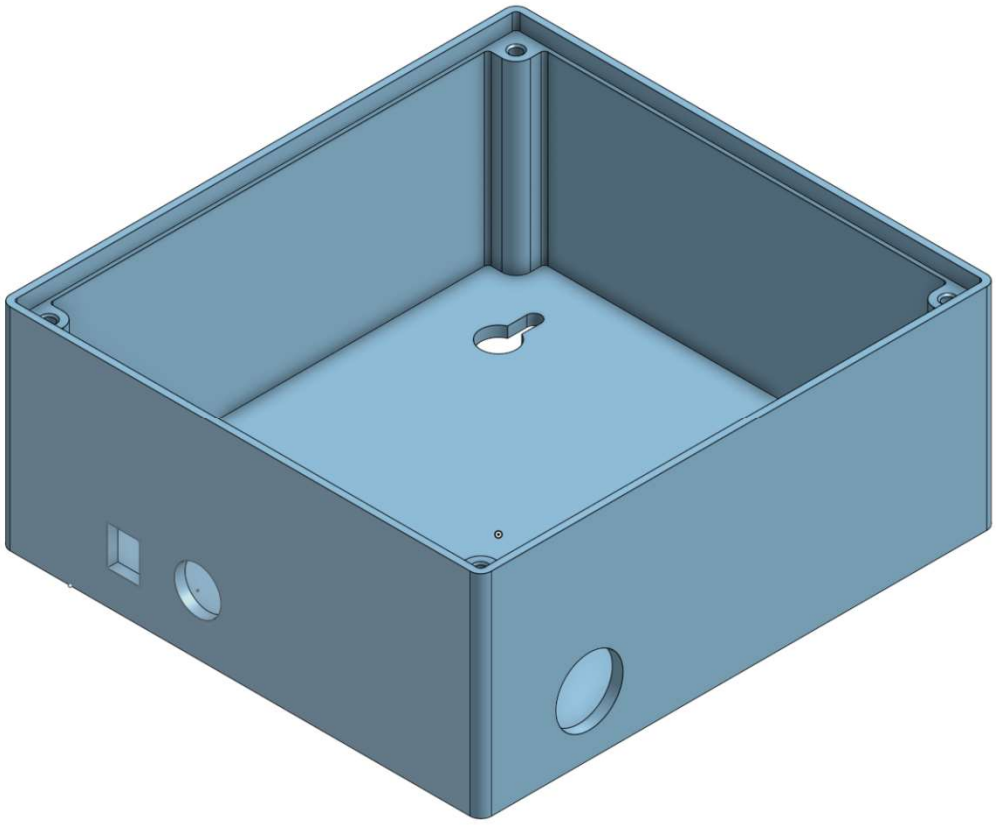
Panel Frontal:

https://octopart.com/bom-tool/DTLM5kml

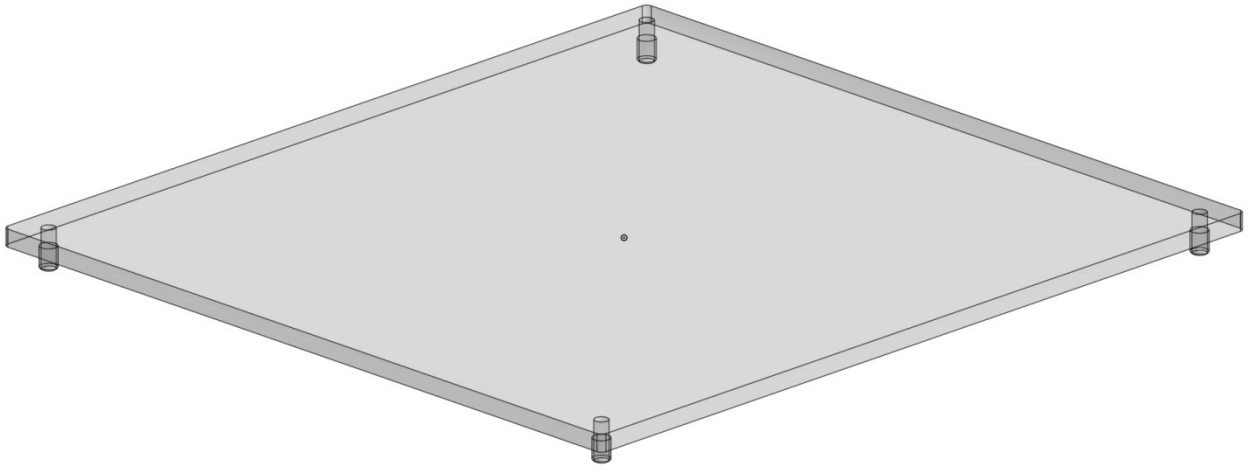
Placa Base:

https://octopart.com/bom-tool/Puk0rRIS

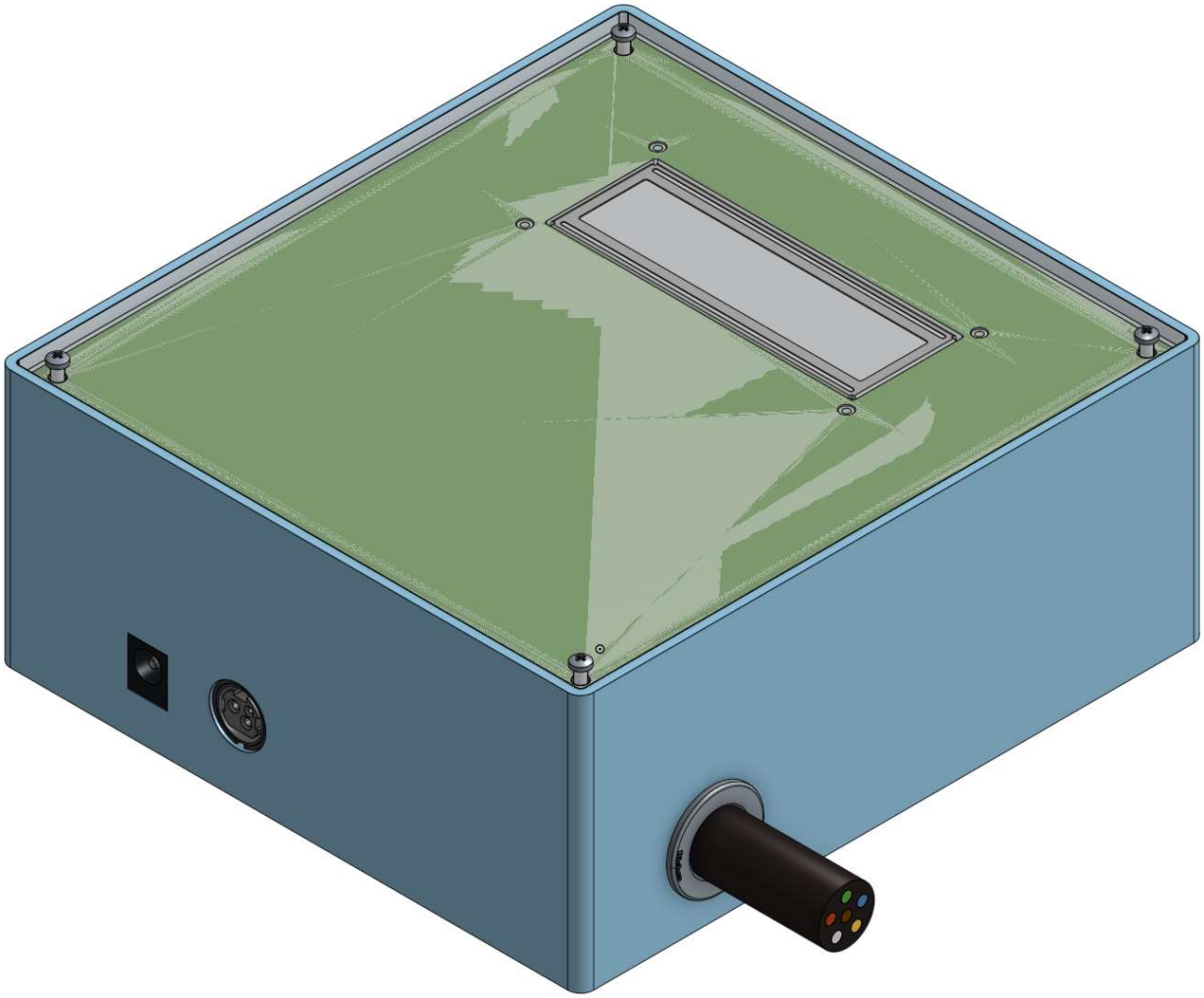
Gabinete en 3D:



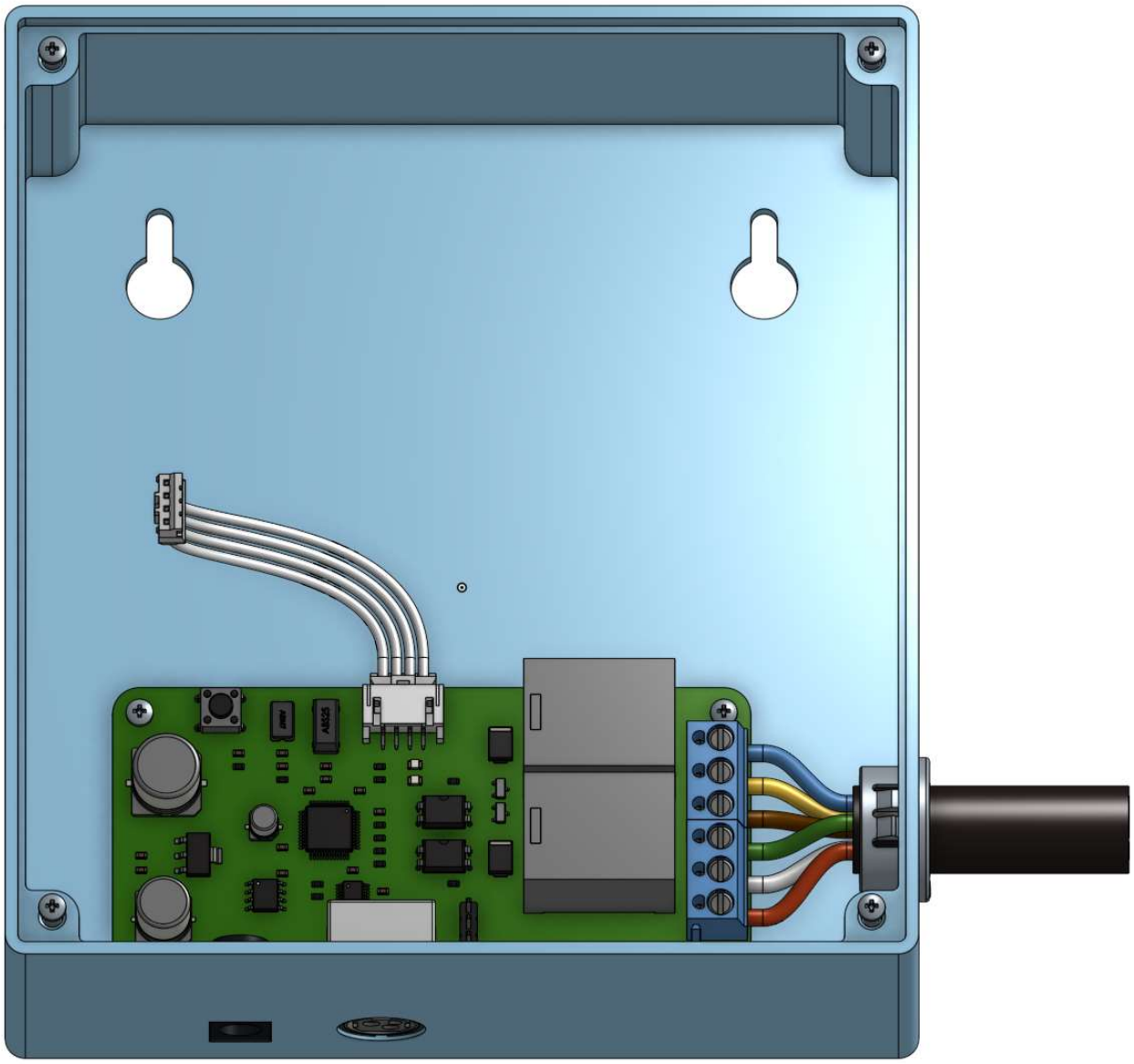
Gabinete - Base



Gabinete – Frente Acrílico



Gabinete – Ensamblaje



Gabinete – Vista Interior

Link a ON-Shape:

https://cad.onshape.com/documents/e1ddf4852ee30d10dbbbdde3/w/8d886781dbbcdb7a099e42b0/e/7b9d0c14e845cfce50a3e853

Programa

Para realizar este proyecto optamos con un sistema TDS con un tick de 1ms.

Cuenta con las siguientes tareas:

- Tarea ADC
- Tarea Promediar
- Tarea Actualizar Salidas
- Tarea Leer Teclado
- Tarea Actualizar LCD
- Tarea Sistema
- Tarea Control Presion

Main:

```
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_TIM2_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

  //LLamo a la funcion que inicializa
  setup();

  while (1)
  {
      loop();
  }
}

```

Funcion setup, Inicializa el TDS

```
40 void setup(){
41
42     dwt_init();
43
44     //*************** CARGAR CONFIGURACIONES DE FABRICA ***************///
45     // Descomentar la siguiente linea para restarurar las configuraciones de fabrica
46
47     //Inicializar_Configuracion();
48     //*************** CARGAR CONFIGURACIONES DE FABRICA ***************///
49
50     CargarConfiguracionFromMemoria();
51
52     inicializar_teclado();
53
54     LCD_init();
55     CambiarEstadoAlarma(inactivo);
56
57
58     ADC_init();
59     ticks = 0;
60     PantallaIndicacion(0);
61
62     //************** INICIALIZACION DE PLANIFICADOR *************************///
63
64     inicializar_despachador(lista_principal,
65         MAX_LEN_TASK_LIST, MONITOR_I_Start, MONITOR_I_Stop, falla_sistema);
66
67     agregar_tarea(lista_principal, tarea_adc, NULL, 0, 1, 0, 100);
68     agregar_tarea(lista_principal, tarea_promediar, NULL, 0, 1, 0, 50);
69     agregar_tarea(lista_principal, tarea_actualizar_salidas, NULL, 0, 1, 0, 10);
70     agregar_tarea(lista_principal, tarea_leer_teclado, NULL, 0, 1, 0, 60);
71     agregar_tarea(lista_principal, Tarea_actualizar_lcd, NULL, 0, 1, 0, 100);
72     agregar_tarea(lista_principal, tarea_sistema, NULL, 0, 1, 0, 150);
73     agregar_tarea(lista_principal, tarea_control_presion, NULL, 0, 1, 0, 20);
74 }
```

Funcion loop:

```
78 void loop(){
79
80     dwt_reset()
81     if (ticks>=SYSTEM_TICK_MS)
82     {
83         ticks=0;
84         despachar_tareas();
85     }
86     tics_despachador = dwt_read();
87     if (tics_despachador > wcet_todo)
88         wcet_todo = tics_despachador;
89 }
```

Tarea ADC:

```
35 void tarea_adc(void *p)
36 {
37     valor_adc = LeerWordFromI2C(DIR_ADC);
38 }
```

Tarea Promediar:

```
40 void tarea_promediar(void *p)
41 {
42     static int ix = 0;
43 
44     uint32_t acumulador = 0;
45     mediciones_ADC[ix++] = valor_adc;
46     if (ix == LEN_PROMEDIO)
47     {
48         flag_datos_completos = 1;
49         ix = 0;
50     }
51 
52     if(flag_datos_completos)
53     {
54         for(uint8_t i = 0; i< LEN_PROMEDIO; i++)
55             acumulador += mediciones_ADC[i];
56 
57         valor_promediado = acumulador / LEN_PROMEDIO;
58         corrienteMedida=(valor_promediado*4096/32768)*10;
59         presionMedida= (corrienteMedida-4000);
60         if(presionMedida<0)presionMedida=0;
61         presionMedida=presionMedida*625/1000;
62     }
63 }
```

Tarea Actualizar Salidas:

```
void tarea_actualizar_salidas(){

    if(estadoAlarmaNuevo != estadoAlarma){
        estadoAlarma = estadoAlarmaNuevo;
        EscribirPin(PuertoSalidas, PIN_ALARMA, estadoAlarmaNuevo);
        EscribirPin(PuertoSalidas, PIN_BUZZER, estadoAlarmaNuevo);
    }

    if(estadoCompresorNuevo != estadoCompresor){
        estadoCompresor = estadoCompresorNuevo;
        EscribirPin(PuertoSalidas, PIN_COMPRESOR, estadoCompresorNuevo);
    }

    if(estadoDescargaNuevo != estadoDescarga){
        estadoDescarga = estadoDescargaNuevo;
        EscribirPin(PuertoSalidas, PIN_DESCARGA, estadoDescargaNuevo);
    }

    //Para hacer sonar el buzzer

    if (estadoAlarma == activo){
        CambiarPin(PuertoSalidas, PIN_BUZZER);
    }
}
```

Tarea Leer Teclado:

```
void tarea_leer_teclado(){

    switch(estado){

        //bcet = 192 uS
        //wcer = 192 uS
        case escribir_fila:
            EscribirByteToI2c(TECLADO_ADDR, array_filas[row]);
            estado = leer_columnas;
            break;

        //bcet = 192 uS
        //wcer = 192 uS
        case leer_columnas:
            LecturaTeclado = LeerByteFromI2C(TECLADO_ADDR);
            estado = procesar_columnas;
            break;
        //bcet = 1.3 uS
        //wcet = 2.3 uS
        case procesar_columnas:
            switch(row){
                case 0:
                    if(!(LecturaTeclado & COL1)){
                        DatosTeclas |= 0x01;
                    }else if(!(LecturaTeclado & COL2)){
                        DatosTeclas |= 0x02;
                    }else if(!(LecturaTeclado & COL3)){
                        DatosTeclas |= 0x04;
                    }else if(!(LecturaTeclado & COL4)){
                        DatosTeclas |= 0x08;
                    }
```

```
                    break;
                case 1:
                    if(!(LecturaTeclado & COL1)){
                        DatosTeclas |= 0x10;
                    }else if(!(LecturaTeclado & COL2)){
                        DatosTeclas |= 0x20;
                    }else if(!(LecturaTeclado & COL3)){
                        DatosTeclas |= 0x40;
                    }else if(!(LecturaTeclado & COL4)){
                        DatosTeclas |= 0x80;
                    }
                    break;
                case 2:
                    if(!(LecturaTeclado & COL1)){
                        DatosTeclas |= 0x100;
                    }else if(!(LecturaTeclado & COL2)){
                        DatosTeclas |= 0x200;
                    }else if(!(LecturaTeclado & COL3)){
                        DatosTeclas |= 0x400;
                    }else if(!(LecturaTeclado & COL4)){
                        DatosTeclas |= 0x800;
                    }
                    break;
                case 3:
                    if(!(LecturaTeclado & COL1)){
                        DatosTeclas |= 0x1000;
                    }else if(!(LecturaTeclado & COL2)){
                        DatosTeclas |= 0x2000;
                    }else if(!(LecturaTeclado & COL3)){
                        DatosTeclas |= 0x4000;
                    }else if(!(LecturaTeclado & COL4)){
                        DatosTeclas |= 0x8000;
                    }
                    break;
            }
            if(row == ROWS-1){
                estado = procesar_teclas;
            }else{
                row++;
                estado = escribir_fila;
            }
            break;
        //bcet = 26.2 uS
        //wcet = 30.4 uS
        case procesar_teclas:
            ProcesarTeclas();
            estado = escribir_fila;
            DatosTeclas=0;
            row=0;
            break;
    }
}
```

Tarea Actualizar LCD:

```
void Tarea_actualizar_lcd(void* p)
{
    switch(Estado_pantalla)
    {
        case LCD_IDLE:
            break;
        case LCD_CLEAR_SCREEN:
            Dato_LCD=0x01;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //INDICACION DE PRESION
        case MAIN_INDICACION:
            LCD_Escribir_String("PRESION ACTUAL: ");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=MAIN_INDICACION_2;}
            break;
        case MAIN_INDICACION_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=MAIN_INDICACION_3;}
            break;
        case MAIN_INDICACION_3:
            LCD_Escribir_String(LCD_VARIABLE);
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=MAIN_INDICACION_4;}
            break;
        case MAIN_INDICACION_4:
            LCD_Escribir_String("BAR");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //CONFIGURACION
        case MAIN_CONFIG:
            LCD_Escribir_String("CONFIGURACION");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //ALARMAS
        case CONFIG_ALARMAS:
            LCD_Escribir_String("ALARMAS");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //ALARMA BAJA PRESION
        case CONFIG_ALARMAS_BAJA:
            LCD_Escribir_String("BAJA PRESION");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_ALARMAS_BAJA_2;}
            break;
        case CONFIG_ALARMAS_BAJA_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=CONFIG_ALARMAS_BAJA_3;}
            break;
        case CONFIG_ALARMAS_BAJA_3:
```

```
            LCD_Escribir_String(LCD_VARIABLE);
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_ALARMAS_BAJA_4;}
            break;
        case CONFIG_ALARMAS_BAJA_4:
            LCD_Escribir_String("BAR");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //ALARMA ALTA PRESION
        case CONFIG_ALARMAS_ALTA:
            LCD_Escribir_String("ALTA PRESION");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_ALARMAS_ALTA_2;}
            break;
        case CONFIG_ALARMAS_ALTA_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=CONFIG_ALARMAS_ALTA_3;}
            break;
        case CONFIG_ALARMAS_ALTA_3:
            LCD_Escribir_String(LCD_VARIABLE);
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_ALARMAS_ALTA_4;}
            break;
        case CONFIG_ALARMAS_ALTA_4:
            LCD_Escribir_String("BAR");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        //VARIABLES DE OPERACION
        case CONFIG_VARIABLES:
            LCD_Escribir_String("VARIABLES DE ");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_VARIABLES_2;}
            break;
        case CONFIG_VARIABLES_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=CONFIG_VARIABLES_3;}
            break;
        case CONFIG_VARIABLES_3:
            LCD_Escribir_String("OPERACION");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // COMPRESOR RANGO MINIMO
        case CONFIG_VARIABLES_RANGOMIN:
            LCD_Escribir_String("RANGO MINIMO");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMIN_2;}
            break;
        case CONFIG_VARIABLES_RANGOMIN_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMIN_3;}
            break;
        case CONFIG_VARIABLES_RANGOMIN_3:
            LCD_Escribir_String(LCD_VARIABLE);
```

```
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMIN_4;}
            break;
        case CONFIG_VARIABLES_RANGOMIN_4:
            LCD_Escribir_String("BAR");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // COMPRESOR RANGO MAXIMO
        case CONFIG_VARIABLES_RANGOMAX:
            LCD_Escribir_String("RANGO MAXIMO");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMAX_2;}
            break;
        case CONFIG_VARIABLES_RANGOMAX_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMAX_3;}
            break;
        case CONFIG_VARIABLES_RANGOMAX_3:
            LCD_Escribir_String(LCD_VARIABLE);
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=CONFIG_VARIABLES_RANGOMAX_4;}
            break;
        case CONFIG_VARIABLES_RANGOMAX_4:
            LCD_Escribir_String("BAR");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // DEBUG
        case MAIN_DEBUG:
            LCD_Escribir_String("DEBUG");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // SALIDAS
        case DEBUG_SALIDAS:
            LCD_Escribir_String("SALIDAS");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // TEST ALARMAS
        case DEBUG_SALIDAS_TEST_ALARMAS:
            LCD_Escribir_String("PROBAR ALARMA");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=DEBUG_SALIDAS_TEST_ALARMAS_2;}
            break;
        case DEBUG_SALIDAS_TEST_ALARMAS_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=DEBUG_SALIDAS_TEST_ALARMAS_3;}
            break;
        case DEBUG_SALIDAS_TEST_ALARMAS_3:
            LCD_Escribir_String("1:ENC  2:APAG");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        // TEST COMPRESOR
        case DEBUG_SALIDAS_TEST_COMPR:
            LCD_Escribir_String("PROBAR COMPRESOR");
```

```
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=DEBUG_SALIDAS_TEST_COMPR_2;}
        break;
    case DEBUG_SALIDAS_TEST_COMPR_2:
        Dato_LCD=0xC0;
        LCD_Escribir_Comando();
        if(lcd_ocupado == 0)
        {Estado_pantalla=DEBUG_SALIDAS_TEST_COMPR_3;}
        break;
    case DEBUG_SALIDAS_TEST_COMPR_3:
        LCD_Escribir_String("1:ENC  2:APAG");
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=LCD_IDLE;}
        break;
    // VALVULA DE DESCARGA
    case DEBUG_SALIDAS_TEST_DESC:
        LCD_Escribir_String("PROBAR VALV DESC");
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=DEBUG_SALIDAS_TEST_DESC_2;}
        break;
    case DEBUG_SALIDAS_TEST_DESC_2:
        Dato_LCD=0xC0;
        LCD_Escribir_Comando();
        if(lcd_ocupado == 0)
        {Estado_pantalla=DEBUG_SALIDAS_TEST_DESC_3;}
        break;
    case DEBUG_SALIDAS_TEST_DESC_3:
        LCD_Escribir_String("1:ENC  2:APAG");
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=LCD_IDLE;}
        break;


    // ENTRADAS
    case DEBUG_ENTRADAS:
        LCD_Escribir_String("ENTRADAS");

        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=LCD_IDLE;}
        break;
    // TEST TECLADO
    case DEBUG_ENTRADAS_TEST_TECLA:
        LCD_Escribir_String("PROBAR TECLADO");
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=DEBUG_ENTRADAS_TEST_TECLA_2;}
        break;
    case DEBUG_ENTRADAS_TEST_TECLA_2:
        Dato_LCD=0xC0;
        LCD_Escribir_Comando();
        if(lcd_ocupado == 0)
        {Estado_pantalla=DEBUG_ENTRADAS_TEST_TECLA_3;}
        break;
    case DEBUG_ENTRADAS_TEST_TECLA_3:
        LCD_Escribir_String("TECLA: ");
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=DEBUG_ENTRADAS_TEST_TECLA_4;}
        break;
    case DEBUG_ENTRADAS_TEST_TECLA_4:
        LCD_Escribir_String(LCD_VARIABLE);
        if(LCD_STATUS == LCD_DONE)
        {Estado_pantalla=LCD_IDLE;}
        break;
```

```
        // TEST LAZO I
        case DEBUG_ENTRADAS_TEST_ADC:
            LCD_Escribir_String("PROBAR LAZO 20mA");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=DEBUG_ENTRADAS_TEST_ADC_2;}
            break;
        case DEBUG_ENTRADAS_TEST_ADC_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=DEBUG_ENTRADAS_TEST_ADC_3;}
            break;
        case DEBUG_ENTRADAS_TEST_ADC_3:
            LCD_Escribir_String("LAZO: ");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=DEBUG_ENTRADAS_TEST_ADC_4;}
            break;
        case DEBUG_ENTRADAS_TEST_ADC_4:
            LCD_Escribir_String(LCD_VARIABLE);
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=DEBUG_ENTRADAS_TEST_ADC_5;}
            break;
        case DEBUG_ENTRADAS_TEST_ADC_5:
            LCD_Escribir_String("mA");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        case ALERTA_ALTA_PRESION:
            LCD_Escribir_String("ALERTA:");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=ALERTA_ALTA_PRESION_2;}
            break;
        case ALERTA_ALTA_PRESION_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=ALERTA_ALTA_PRESION_3;}
            break;
        case ALERTA_ALTA_PRESION_3:
            LCD_Escribir_String("ALTA PRESION!!!");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        case ALERTA_BAJA_PRESION:
            LCD_Escribir_String("ALERTA:");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=ALERTA_BAJA_PRESION_2;}
            break;
        case ALERTA_BAJA_PRESION_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=ALERTA_BAJA_PRESION_3;}
            break;
        case ALERTA_BAJA_PRESION_3:
            LCD_Escribir_String("BAJA PRESION!!!");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
        case ALERTA_FALLA_LAZO:
            LCD_Escribir_String("ALERTA:");
            if(LCD_STATUS == LCD_DONE)
```

```
            {Estado_pantalla=ALERTA_FALLA_LAZO_2;}
            break;
        case ALERTA_FALLA_LAZO_2:
            Dato_LCD=0xC0;
            LCD_Escribir_Comando();
            if(lcd_ocupado == 0)
            {Estado_pantalla=ALERTA_FALLA_LAZO_3;}
            break;
        case ALERTA_FALLA_LAZO_3:
            LCD_Escribir_String("FALLA DE LAZO!!!");
            if(LCD_STATUS == LCD_DONE)
            {Estado_pantalla=LCD_IDLE;}
            break;
    }

}
```

Tarea Sistema:

```
void tarea_sistema(void *p)
{
    ProcesarTeclado();

    switch(accionCore)
    {

        ///MENU INDICACION///
        case IND_BORR_LCD:
            flag_debug = 0;
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=IND_ACT_LCD;
            break;
        case IND_ACT_LCD:
            control=PantallaIndicacion(presionMedida);
            if(control == WRITING_OK) accionCore=IND_LEE_TECL;
            break;
        case IND_ACT_VAL_LCD:
            control=PantallaActualizarIndicacion(presionMedida);
            if(control == WRITING_OK) accionCore=IND_LEE_TECL;
            break;
        case IND_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=DBG_BORR_LCD;
                    break;
                case TECLA_B:
                    accionCore=CFG_BORR_LCD;
                    break;
                case TECLA_C:
                    accionCore=IND_BORR_LCD;
                    break;
                case TECLA_D:
                    break;
                case NO_KEY:
                    break;
            }
            refrescarPantalla++;
            if(refrescarPantalla==500)
            {
                accionCore=IND_ACT_VAL_LCD;
```

```
            refrescarPantalla=0;
        }
        break;


    ////MENU CONFIG///

        case CFG_BORR_LCD:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=CFG_ACT_LCD;
            break;
        case CFG_ACT_LCD:
            control=PantallaConfig();
            if(control == WRITING_OK) accionCore=CFG_LEE_TECL;
            break;
        case CFG_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=IND_BORR_LCD;
                    break;
                case TECLA_B:
                    accionCore=DBG_BORR_LCD;
                    break;
                case TECLA_C:
                    accionCore=IND_BORR_LCD;
                    break;
                case TECLA_D:
                    accionCore=SUBM_CFG_ALARM_BORR;
                    break;
                case NO_KEY:
                    break;
            }
            break;

        /// COMIENZO SUBMENU ALARMAS ///

        case SUBM_CFG_ALARM_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM;
            break;
        case SUBM_CFG_ALARM:
            control=PantallaAlarm();
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_LEE_TECL;
            break;
        case SUBM_CFG_ALARM_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_CFG_VAR_BORR;
                    break;
                case TECLA_B:
                    accionCore=SUBM_CFG_VAR_BORR;
                    break;
                case TECLA_C:
                    accionCore=CFG_BORR_LCD;
                    break;
                case TECLA_D:
                    accionCore=SUBM_CFG_ALARM_MAX_BORR;
                    break;
                case NO_KEY:
                    break;
```

```
            }
            break;

    /// COMIENZO SUBMENU ALARMAS MAXIMO///

        case SUBM_CFG_ALARM_MAX_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MAX;
            break;
        case SUBM_CFG_ALARM_MAX:
            control=PantallaAltaPresion(configuracion.alarma_alta_presion);
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MAX_LEE_TECL;
            break;
        case SUBM_CFG_ALARM_MAX_ACT:
            control=PantallaActualizarAltaPresion(variable_temp);
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MAX_LEE_TECL;
            break;
        case SUBM_CFG_ALARM_MAX_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_CFG_ALARM_MIN_BORR;
                    variable_temp=0;
                    break;
                case TECLA_B:
                    accionCore=SUBM_CFG_ALARM_MIN_BORR;
                    variable_temp=0;
                    break;
                case TECLA_C:
                    accionCore=SUBM_CFG_ALARM_BORR;
                    variable_temp=0;
                    break;
                case TECLA_D:

if((variable_temp>configuracion.compresor_alta_presion)&&(variable_temp<=10000))
                    {
                        configuracion.alarma_alta_presion=variable_temp;
                        Actualizar_EEPROM(OFFSET_ALARMA_ALTA);
                        variable_temp=0;
                    }
                    else
                    {
                        accionCore=SUBM_CFG_ALARM_MAX_BORR;
                        variable_temp=0;
                    }
                    break;
                case NO_KEY:
                    break;
                default:
                    ModificarVariable(teclaPresionada);
                    accionCore=SUBM_CFG_ALARM_MAX_ACT;
                    break;
            }
            break;

    /// COMIENZO SUBMENU ALARMAS MINIMO///

        case SUBM_CFG_ALARM_MIN_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MIN;
            break;
        case SUBM_CFG_ALARM_MIN:
```

```
            control=PantallaBajaPresion(configuracion.alarma_baja_presion);
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MIN_LEE_TECL;
            break;
        case SUBM_CFG_ALARM_MIN_ACT:
            control=PantallaActualizarBajaPresion(variable_temp);
            if(control == WRITING_OK) accionCore=SUBM_CFG_ALARM_MIN_LEE_TECL;
            break;
        case SUBM_CFG_ALARM_MIN_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_CFG_ALARM_MAX_BORR;
                    variable_temp=0;
                    break;
                case TECLA_B:
                    accionCore=SUBM_CFG_ALARM_MAX_BORR;
                    variable_temp=0;
                    break;
                case TECLA_C:
                    accionCore=SUBM_CFG_ALARM_BORR;
                    variable_temp=0;
                    break;
                case TECLA_D:

if((variable_temp<configuracion.compresor_baja_presion)&&(variable_temp>=0))
                    {
                        configuracion.alarma_baja_presion=variable_temp;
                        Actualizar_EEPROM(OFFSET_ALARMA_BAJA);
                        variable_temp=0;
                    }
                    else
                    {
                        accionCore=SUBM_CFG_ALARM_MIN_BORR;
                        variable_temp=0;
                    }
                    break;
                case NO_KEY:
                    break;
                default:
                    ModificarVariable(teclaPresionada);
                    accionCore=SUBM_CFG_ALARM_MIN_ACT;
                    break;
            }
            break;

    /// COMIENZO MENU VARIABLES DE OPERACION ///

        case SUBM_CFG_VAR_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_CFG_VAR;
            break;
        case SUBM_CFG_VAR:
            control=PantallaVarOp();
            if(control == WRITING_OK) accionCore=SUBM_CFG_VAR_LEE_TECL;
            break;
        case SUBM_CFG_VAR_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_CFG_ALARM_BORR;
                    break;
                case TECLA_B:
```

```
                    accionCore=SUBM_CFG_ALARM_BORR;
                    break;
                case TECLA_C:
                    accionCore=CFG_BORR_LCD;
                    break;
                case TECLA_D:
                    accionCore=SUBM_CFG_COMPR_MAX_BORR;
                    break;
                case NO_KEY:
                    break;
            }
            break;

        /// COMIENZO SUBMENU COMPRESOR MAXIMO///

        case SUBM_CFG_COMPR_MAX_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MAX;
            break;
        case SUBM_CFG_COMPR_MAX:
            control=PantallaRangoMax(configuracion.compresor_alta_presion);
            if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MAX_LEE_TECL;
            break;
        case SUBM_CFG_COMPR_MAX_ACT:
            control=PantallaActualizarRangoMax(variable_temp);
            if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MAX_LEE_TECL;
            break;
        case SUBM_CFG_COMPR_MAX_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_CFG_COMPR_MIN_BORR;
                    variable_temp=0;
                    break;
                case TECLA_B:
                    accionCore=SUBM_CFG_COMPR_MIN_BORR;
                    variable_temp=0;
                    break;
                case TECLA_C:
                    accionCore=SUBM_CFG_VAR_BORR;
                    variable_temp=0;
                    break;
                case TECLA_D:

    if((variable_temp>configuracion.compresor_baja_presion)&&(variable_temp<configuraci
on.alarma_alta_presion))
                    {
                        configuracion.compresor_alta_presion=variable_temp;
                        Actualizar_EEPROM(OFFSET_COMPRESOR_ALTA);
                        variable_temp=0;
                    }
                    else
                    {
                        accionCore=SUBM_CFG_COMPR_MAX_BORR;
                        variable_temp=0;
                    }
                    break;
                case NO_KEY:
                    break;
                default:
                    ModificarVariable(teclaPresionada);
                    accionCore=SUBM_CFG_COMPR_MAX_ACT;
```

```
                    break;
                }
                break;

            /// COMIENZO SUBMENU COMPRESOR MINIMO///

            case SUBM_CFG_COMPR_MIN_BORR:
                control=BorrarPantalla();
                if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MIN;
                break;
            case SUBM_CFG_COMPR_MIN:
                control=PantallaRangoMin(configuracion.compresor_baja_presion);
                if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MIN_LEE_TECL;
                break;
            case SUBM_CFG_COMPR_MIN_ACT:
                control=PantallaActualizarRangoMin(variable_temp);
                if(control == WRITING_OK) accionCore=SUBM_CFG_COMPR_MIN_LEE_TECL;
                break;
            case SUBM_CFG_COMPR_MIN_LEE_TECL:
                switch(teclaPresionada)
                {
                    case TECLA_A:
                        accionCore=SUBM_CFG_COMPR_MAX_BORR;
                        break;
                    case TECLA_B:
                        accionCore=SUBM_CFG_COMPR_MAX_BORR;
                        break;
                    case TECLA_C:
                        accionCore=SUBM_CFG_VAR_BORR;
                        break;
                    case TECLA_D:

    if((variable_temp<configuracion.compresor_alta_presion)&&(variable_temp>configuraci
on.alarma_baja_presion))
                        {
                            configuracion.compresor_baja_presion=variable_temp;
                            Actualizar_EEPROM(OFFSET_COMPRESOR_BAJA);
                            variable_temp=0;
                        }
                        else
                        {
                            accionCore=SUBM_CFG_COMPR_MIN_BORR;
                            variable_temp=0;
                        }
                        break;
                    case NO_KEY:
                        break;
                    default:
                        ModificarVariable(teclaPresionada);
                        accionCore=SUBM_CFG_COMPR_MIN_ACT;
                        break;
                }
                break;



        ///MENU DEBUG///
        case DBG_BORR_LCD:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=DBG_ACT_LCD;
            break;
        case DBG_ACT_LCD:
```

```
            control=PantallaDebug();
            if(control == WRITING_OK) accionCore=DBG_LEE_TECL;
            break;
        case DBG_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=CFG_BORR_LCD;
                    break;
                case TECLA_B:
                    accionCore=IND_BORR_LCD;
                    break;
                case TECLA_C:
                    accionCore=IND_BORR_LCD;
                    break;
                case TECLA_D:
                    accionCore=SUBM_DBG_SAL_BORR;
                    flag_debug = 1;
                    break;
                case NO_KEY:
                    break;
            }
            break;

        /*************SUBMENU DEBUG **********************/
        case SUBM_DBG_SAL_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_DBG_SAL;
            break;
        case SUBM_DBG_SAL:
            control=PantallaSalidas();
            if(control == WRITING_OK) accionCore=SUBM_DBG_SAL_LEE_TECL;
            break;
        case SUBM_DBG_SAL_LEE_TECL:
            switch(teclaPresionada)
                {
                    case TECLA_A:
                        accionCore=SUBM_DBG_TEC_BORR;
                        break;
                    case TECLA_B:
                        accionCore=SUBM_DBG_ADC_BORR;
                        break;
                    case TECLA_C:
                        accionCore=IND_BORR_LCD;
                        break;
                    case TECLA_D:
                        accionCore=SUBM_SAL_TCOMP_BORR;
                        break;
                    case NO_KEY:
                        break;
                }
            break;
        case SUBM_DBG_ADC_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_DBG_ADC;
            break;
        case SUBM_DBG_ADC:
            control=PantallaTestLazo(0);
            if(control == WRITING_OK) accionCore=SUBM_DBG_ADC_LEE_TECL;
            break;
        case SUBM_DBG_ADC_ACT_LCD:
            control=PantallaActualizarTestLazo(corrienteMedida);
```

```
            if(control == WRITING_OK) accionCore=SUBM_DBG_ADC_LEE_TECL;
            break;
        case SUBM_DBG_ADC_LEE_TECL:
            switch(teclaPresionada)
                {
                    case TECLA_A:
                        accionCore=SUBM_DBG_SAL_BORR;
                        break;
                    case TECLA_B:
                        accionCore=SUBM_DBG_TEC_BORR;
                        break;
                    case TECLA_C:
                        accionCore=IND_BORR_LCD;
                        break;
                    case TECLA_D:
                        break;
                    case NO_KEY:
                        break;
                }
            refrescarPantalla++;
            if(refrescarPantalla==500)
            {
                accionCore=SUBM_DBG_ADC_ACT_LCD;
                refrescarPantalla=0;
            }
            break;
        case SUBM_DBG_TEC_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_DBG_TEC;
            break;
        case SUBM_DBG_TEC:
            control=PantallaTestTeclado(0);
            if(control == WRITING_OK) accionCore=SUBM_DBG_TEC_LEE_TECL;
            break;
        case SUBM_DBG_TEC_LEE_TECL:
            switch(teclaPresionada)
                {
                    case TECLA_A:
                        accionCore=SUBM_DBG_ADC_BORR;
                        break;
                    case TECLA_B:
                        accionCore=SUBM_DBG_SAL_BORR;
                        break;
                    case TECLA_C:
                        accionCore=IND_BORR_LCD;
                        break;
                    case TECLA_D:
                        break;
                    case NO_KEY:
                        break;
                    default:
                        PantallaActualizarTecla_TEST_TECLADO(teclaPresionada);
                        break;
                }
            break;

        /************* FIN SUBMENU DEBUG ********************/

        /************** SUBMENU TEST SALIDAS ***************/
            case SUBM_SAL_TCOMP_BORR:
                control=BorrarPantalla();
                if(control == WRITING_OK) accionCore=SUBM_SAL_TCOMP;
```

```
            break;
        case SUBM_SAL_TCOMP:
            control=PantallaTestCompr();
            if(control == WRITING_OK) accionCore=SUBM_SAL_TCOMP_LEE_TECL;
            break;
        case SUBM_SAL_TCOMP_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_SAL_TDESC_BORR;
                    break;
                case TECLA_B:
                    accionCore=SUBM_SAL_TALARM_BORR;
                    break;
                case TECLA_C:
                    accionCore=SUBM_DBG_SAL_BORR;
                    break;
                case TECLA_D:
                    break;
                case NO_KEY:
                    break;
                case TECLA_1:
                    //Prender COMPRESOR
                    CambiarEstadoCompresor(activo);
                    break;
                case TECLA_2:
                    //Apagar COMPRESOR
                    CambiarEstadoCompresor(inactivo);
                    break;
            }
            break;

        case SUBM_SAL_TALARM_BORR:
            control=BorrarPantalla();
            if(control == WRITING_OK) accionCore=SUBM_SAL_TALARM;
            break;
        case SUBM_SAL_TALARM:
            control=PantallaTestAlarm();
            if(control == WRITING_OK) accionCore=SUBM_SAL_TALARM_LEE_TECL;
            break;
        case SUBM_SAL_TALARM_LEE_TECL:
            switch(teclaPresionada)
            {
                case TECLA_A:
                    accionCore=SUBM_SAL_TCOMP_BORR;
                    break;
                case TECLA_B:
                    accionCore=SUBM_SAL_TDESC_BORR;
                    break;
                case TECLA_C:
                    accionCore=SUBM_DBG_SAL_BORR;
                    break;
                case TECLA_D:
                    break;
                case NO_KEY:
                    break;
                case TECLA_1:
                    //Prender ALARMA
                    CambiarEstadoAlarma(activo);
                    break;
                case TECLA_2:
                    //Apagar ALARMA
```

```
                CambiarEstadoAlarma(inactivo);
                break;
        }
        break;

    case SUBM_SAL_TDESC_BORR:
        control=BorrarPantalla();
        if(control == WRITING_OK) accionCore=SUBM_SAL_TDESC;
        break;
    case SUBM_SAL_TDESC:
        control=PantallaTestDesc();
        if(control == WRITING_OK) accionCore=SUBM_SAL_TDESC_LEE_TECL;
        break;
    case SUBM_SAL_TDESC_LEE_TECL:
        switch(teclaPresionada)
        {
            case TECLA_A:
                accionCore=SUBM_SAL_TALARM_BORR;
                break;
            case TECLA_B:
                accionCore=SUBM_SAL_TCOMP_BORR;
                break;
            case TECLA_C:
                accionCore=SUBM_DBG_SAL_BORR;
                break;
            case TECLA_D:
                break;
            case NO_KEY:
                break;
            case TECLA_1:
                //Prender Valvula de Descarga
                CambiarEstadoDescarga(activo);
                break;
            case TECLA_2:
                //Apagar Valvula de Descarga
                CambiarEstadoDescarga(inactivo);
                break;
        }
        break;

        //Pantallas de visualizacion de error
    case PANTALLA_ALERTA_ALTA_PRESION_BORR:
        control=BorrarPantalla();
        if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_ALTA_PRESION;
        break;
    case PANTALLA_ALERTA_ALTA_PRESION:
        control=PantallaAlertaAltaPresion();
        if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_LEE_TECL;
        break;
    case PANTALLA_ALERTA_BAJA_PRESION_BORR:
        control=BorrarPantalla();
        if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_BAJA_PRESION;
        break;
    case PANTALLA_ALERTA_BAJA_PRESION:
        control=PantallaAlertaBajaPresion();
        if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_LEE_TECL;
        break;
    case PANTALLA_ALERTA_ERROR_LAZO_BORR:
        control=BorrarPantalla();
        if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_ERROR_LAZO;
        break;
    case PANTALLA_ALERTA_ERROR_LAZO:
```

```
                control=PantallaAlertaLazo();
                if(control == WRITING_OK) accionCore=PANTALLA_ALERTA_LEE_TECL;
                break;
            case PANTALLA_ALERTA_LEE_TECL:
                switch(teclaPresionada)
                {
                    case TECLA_A:
                        break;
                    case TECLA_B:
                        break;
                    case TECLA_C:
                        //Si apreto la tecla C salgo del error y vuelvo a controlar la
presion
                        accionCore=IND_BORR_LCD;
                        volverAlControlDePresion();
                        break;
                    case TECLA_D:
                        break;
                    case NO_KEY:
                        break;
                }
                break;
    }
}
```

Tarea Control Presión:

```
void tarea_control_presion(void *p){

    static uint32_t tiempo_en_alta_presion = 0;
    static uint32_t tiempo_en_baja_presion = 0;

    //Si estoy en debug no controlo nada..
    if(flag_debug) return;

    //Incremento el contador de ciclos de ejecucion hasta que llegue a START_UP_TIME
    if(tiempo_de_comienzo < START_UP_TIME)tiempo_de_comienzo++;

    switch(estado_ejecucion){
    case START_UP:
        if(configuracion.flags & estado_lazo_sensor){
            /* Agregar aca control del lazo de 4-20mA */
            if(flag_datos_completos && ((corrienteMedida >
configuracion.sensor_maximo_adc )||(corrienteMedida <
configuracion.sensor_minimo_adc))){
                //Espero que el flag se ponga en 1 para tener una corriente medida
valida.
                estado_ejecucion = ERROR_LAZO;
                break;
            }
        }
        CambiarEstadoAlarma(inactivo);
        CambiarEstadoDescarga(inactivo);

        if(presionMedida >= configuracion.compresor_alta_presion){
            CambiarEstadoCompresor(inactivo);
        }
        if(presionMedida <= configuracion.compresor_baja_presion){
            CambiarEstadoCompresor(activo);
        }
```

```
        if(presionMedida >= configuracion.alarma_alta_presion){
            estado_ejecucion = SOBRE_PRESION;
            CambiarEstadoCompresor(inactivo);
            CambiarEstadoAlarma(activo);
            CambiarEstadoDescarga(activo);
            tiempo_en_alta_presion = 0;
        }

        //Si cumpli el tiempo de START_UP me paso a full mode
        if(tiempo_de_comienzo >= START_UP_TIME) estado_ejecucion=FULL_MODE;
        break;
    case FULL_MODE:
        if(configuracion.flags & estado_lazo_sensor){
            /* Agregar aca control del lazo de 4-20mA */
            if(flag_datos_completos && ((corrienteMedida >
configuracion.sensor_maximo_adc )||(corrienteMedida <
configuracion.sensor_minimo_adc))){
                //Espero que el flag se ponga en 1 para tener una corriente medida
valida.
                estado_ejecucion = ERROR_LAZO;
                break;
            }
        }

        CambiarEstadoAlarma(inactivo);
        CambiarEstadoDescarga(inactivo);

        if(presionMedida >= configuracion.compresor_alta_presion){
            CambiarEstadoCompresor(inactivo);
        }
        if(presionMedida <= configuracion.compresor_baja_presion){
            CambiarEstadoCompresor(activo);
        }

        if(presionMedida >= configuracion.alarma_alta_presion){
            estado_ejecucion = SOBRE_PRESION;
            CambiarEstadoCompresor(inactivo);
            CambiarEstadoAlarma(activo);
            CambiarEstadoDescarga(activo);
            tiempo_en_alta_presion = 0;
        }

        if(presionMedida <= configuracion.alarma_baja_presion){
            estado_ejecucion = BAJA_PRESION;
            CambiarEstadoCompresor(activo);
            CambiarEstadoAlarma(activo);
            CambiarEstadoDescarga(inactivo);
            tiempo_en_baja_presion = 0;
        }

        break;
    case SOBRE_PRESION:
        if(configuracion.flags & estado_lazo_sensor){
            /* Agregar aca control del lazo de 4-20mA */
            if(flag_datos_completos && ((corrienteMedida >
configuracion.sensor_maximo_adc )||(corrienteMedida <
configuracion.sensor_minimo_adc))){
                //Espero que el flag se ponga en 1 para tener una corriente medida
valida.
                estado_ejecucion = ERROR_LAZO;
                break;
            }
```

```
        }

        if(tiempo_en_alta_presion++ > TIEMPO_ESPERA_ALTA_PRESION){
            CambiarEstadoAlarma(activo);
            CambiarEstadoDescarga(activo);
            CambiarEstadoCompresor(inactivo);
            estado_ejecucion=SOBRE_PRESION_CRITICA;
        }
        if(presionMedida <= configuracion.compresor_alta_presion){
            CambiarEstadoAlarma(inactivo);
            CambiarEstadoDescarga(inactivo);
            if(tiempo_de_comienzo < START_UP_TIME)
                estado_ejecucion=START_UP;
            else
                estado_ejecucion=FULL_MODE;
        }
        break;
    case SOBRE_PRESION_CRITICA:
        //De aca no me muevo.. exploto todo.. msddd
        accionCore=PANTALLA_ALERTA_ALTA_PRESION_BORR;
        estado_ejecucion=ERROR_PERMANENTE;
        break;
    case BAJA_PRESION:
        if(configuracion.flags & estado_lazo_sensor){
            /* Agregar aca control del lazo de 4-20mA */
            if(flag_datos_completos && ((corrienteMedida >
configuracion.sensor_maximo_adc )||(corrienteMedida <
configuracion.sensor_minimo_adc))){
                //Espero que el flag se ponga en 1 para tener una corriente medida
valida.
                estado_ejecucion = ERROR_LAZO;
                break;
            }
        }

        if(tiempo_en_baja_presion++ > TIEMPO_ESPERA_BAJA_PRESION){
            CambiarEstadoAlarma(activo);
            CambiarEstadoDescarga(inactivo);
            CambiarEstadoCompresor(inactivo);
            estado_ejecucion=BAJA_PRESION_CRITICA;
        }
        if(presionMedida > configuracion.compresor_baja_presion){
            CambiarEstadoAlarma(inactivo);
            CambiarEstadoDescarga(inactivo);
            CambiarEstadoCompresor(activo);
            if(tiempo_de_comienzo < START_UP_TIME)
                estado_ejecucion=START_UP;
            else
                estado_ejecucion=FULL_MODE;
        }
        break;
    case BAJA_PRESION_CRITICA:
        //De aca no me muevo.. exploto todo.. msddd
        accionCore=PANTALLA_ALERTA_BAJA_PRESION_BORR;
        estado_ejecucion=ERROR_PERMANENTE;
        break;
    case ERROR_LAZO:
        //Murio el sensor :(
        CambiarEstadoAlarma(activo);
        CambiarEstadoDescarga(inactivo);
        CambiarEstadoCompresor(inactivo);
        accionCore=PANTALLA_ALERTA_ERROR_LAZO_BORR;
```

```
        estado_ejecucion=ERROR_PERMANENTE;
        break;
    case ERROR_PERMANENTE:
        //Si llego aca la unica manera de salir es reseteando el dispositivo
        break;
    }

}
```

Tiempos medidos en Blue Pill

Peor tiempo de ejecucion total: 21074 -> 292.7us
Total Carga del CPU = 0.2927ms / 1ms = 26.2%

Tarea ADC 81us
Tarea Promediar 22us
Tarea Actualizar Salidas 5us
Tarea Leer Teclado 58us
Tarea Actualizar LCD 66us
Tarea Sistema 133us
Tarea Control Presion 7us

Total peores tiempos de ejecucion = 372us

Links del proyecto

Todo el proyecto

https://drive.google.com/drive/folders/1ttFjJWzeJG8FkLTUAjmdmmnhQa715c8h?usp=sharing

Archivos De KiCad

https://drive.google.com/drive/folders/1RYt2ux4yVCW2OJ3BZ_4A3EixvheEzl0A?usp=sharing

Código Fuente

https://drive.google.com/drive/folders/1Yz9x5iTaVb_Nc2Ezr2y9P9aMWv6eJKZf?usp=sharing

Gabinete

https://drive.google.com/drive/folders/1QXD6Tq9vlW564AlzLLeEOgONPD0ZU_WA?usp=sharing

Hojas De Datos

https://drive.google.com/drive/folders/1TfDo1f43l2kIA2IVsM4seWLPGiB_O4Od?usp=sharing

Informe

https://drive.google.com/drive/folders/1x6xh0yBe1f3DkmDoJzRTCuk4TncekRsM?usp=sharing

Manuales

https://drive.google.com/drive/folders/1AHNvz0aHsR_AXplOHaR-9UXpwtBo-xeU?usp=sharing

PreProyecto

https://drive.google.com/drive/folders/1-eYMSBOoJydX0WHzY6-WSbfjlyc9BBEJ?usp=sharing

Videos

https://drive.google.com/drive/folders/1Y4wJPmkJJgLEg0r3f41NbgZQ3tz8tUtV?usp=sharing

Fotos

https://drive.google.com/drive/folders/1qTJLUy0-bXxDE65scOKkm8jtmn9GBNU4?usp=sharing

Gitlab

https://gitlab.frba.utn.edu.ar/TD2-htravado/ProyectoFinalTD2.git