CENTRO DE INGENIERIA Y DESARROLLO INDUSTRIAL

CIDESI

Proyecto Industrial Terminal

GLCD y RS485 para control de Robot Explorador 2

PARA OBTENER LA ESPECIALIDAD EN

"TECNÓLOGO EN MECATRÓNICA"

PRESENTA

Alumno: Moisés Reyes Portillo

Tutor Académico: M.I. Julio Cesar Solano Vargas

QUERÉTARO, QRO. 2012

Índice

Resumen

Como parte del proyecto "Segunda generación de robot explorador", vigente en CIDESI (Centro de Ingeniería y Desarrollo Industrial), se desarrollaron nuevas implementaciones para el nuevo control remoto.
Se programó entonces un microcontolador PIC18F4550 encargado de la lectura e interpretación de 2 joysticks analógicos, despliegue datos y distintas opciones de menú en una pantalla LCD gráfica, y que se comunique con el DSP para el manejo y configuración del explorador mediante el protocolo serial RS-485.

Antecedentes

El Centro de Ingeniería y Desarrollo Industrial (CIDESI) proporciona a las industrias diversos servicios tecnológicos, entre ellos se encuentra la inspección no destructiva de tanques de almacenamiento de combustibles líquidos y gaseosos, ductos y válvulas recipientes sujetos a presión como calderas, intercambiadores de calor, reactores, autotanques, esferas, estructuras soldadas, partes críticas de aviones, entre otras, de acuerdo a la normatividad aplicable, aplicando distintas técnicas.
Una de las técnicas aplicadas es la inspección por ondas de ultrasonido con robot trepador de paredes en tanques.
Para éste propósito se desarrolló el proyecto "Robot Explorador", en el período 2000-2002.

Justificación

El desarrollo de tecnología en México es un punto clave en todos los proyectos en CIDESI. Busca entonces comenzar por los servicios que brinda a la industria, desarrollando la tecnología requerida.
Actualmente se encuentra en desarrollo la segunda generación del robot explorador, por lo que es necesario hacer mejoras tecnológicas principalmente, ya que han pasado poco más de 10 años.
El control remoto del proyecto previo, contaba con un microcontrolador de 8 bits de Microchip de la familia PIC16F, una pantalla LCD de 2 líneas por 16 caracteres, 1 joystick de 2 ejes digital, y comunicación RS-232 con el DSP encargado del control de motores y sensores.
El microcontrolador se cambió por un PIC18F4550, que aparte de mayor velocidad de procesamiento, está optimizado para programación en C y tiene módulo USB.
La pantalla se cambio por un GLCD (LCD gráfico) de 128x64 pixeles, lo que permite desplegar mayor información, dibujar menús e interfaces gráficas.
La comunicación RS-232 se adaptó, principalmente mediante hardware, a RS-485.
La comunicación RS-485 es más robusta ante el ruido y permite mayores distancias, lo que la hace adecuada para ambientes industriales y óptima para éste proyecto.
El joystick digital de 2 ejes, proporcionaba información muy pobre, 1 ó 0 correspondiente a sus posiciones. Asimismo es posible desviarse hacia un eje no deseado al realizar un movimiento mecánico de la palanca.
En éste proyecto se utilizaron 2 joysticks analógicos de 1 solo eje, lo que permitió una resolución correspondiente a la del convertidor analógico-digital del microcontrolador, en este caso de 10 bits y un aislamiento entre el control de desplazamiento y el control de giro.

Objetivos

Objetivo General

- Desarrollar mejoras en el control remoto de la segunda generación de robot explorador.

Objetivos Específicos

- Efectuar una lectura e interpretación óptima de los 2 canales analógicos.
- Desarrollar el código para dibujar en el GLCD que se integre a las diversas funciones del robot explorador y sea visualmente rápido y atractivo.
- Establecer una comunicación RS-485 óptima con el DSP encargado del movimiento de motores y lectura de sensores.

Especificaciones Técnicas

Al inicio del proyecto se contaba con una pantalla GLCD y un microcontrolador PIC18f4550.
Las características de éste modelo en particular, se describen en la tabla 1.

Modelo: CFAG12864A-TMI-V

CFA G 12864 A T M I V
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

1	Marca	Crystalfontz America, Inc.
2	Modo de Display	G→Tipo Gráfico
3	Dimensiones lógicas de Display	128x64 pixeles
4	Modelo Variante del PCB	A
5	Tipo de luz de fondo	T→LED, Blanco
6	Modo LCD	M→STN Negativo, Azul
7	Tipo de polarizador LCD/ Rango de Temperatura/dirección de vista	I→Transmisivo,W.T,6:00
8	Código Especial	V→Generador negativo integrado

Tabla 1: Características GLCD

La pantalla cuenta con 20 pines, los cuales se describe su función y enumeración en la tabla 2.

Pin No.	Symbol	Level	Description
1	GND	0V	Ground
2	V_{DD}	5.0V	Supply voltage for logic
3	V_o	(Variable)	Operating voltage for LCD
4	D/I	H/L	H: Data , L : Instruction
5	R/W	H/L	H: Read (MPU←Module) , L: Write (MPU→Module)
6	E	H	Enable signal
7	DB0	H/L	Data bus line
8	DB1	H/L	Data bus line
9	DB2	H/L	Data bus line
10	DB3	H/L	Data bus line
11	DB4	H/L	Data bus line
12	DB5	H/L	Data bus line
13	DB6	H/L	Data bus line
14	DB7	H/L	Data bus line
15	CS1	H	Select Column 1~ Column 64
16	CS2	H	Select Column 65~ Column 128
17	RST	L	Reset signal
18	Vout	□	Negative Voltage
19	A	□	Power Supply for LED backlight (+)
20	K	□	Power Supply for LED backlight (-)

Tabla 2: Patillaje GLCD

Se utilizan 14 pines (4→17) del microcontrolador para su manejo, la pantalla tiene sus propios controladores, seleccionados por los CS1 y CS2 (chip select). El diagrama 1 ilustra las conexiones con el microcontrolador así como su potenciómetro de intensidad de luz de fondo.

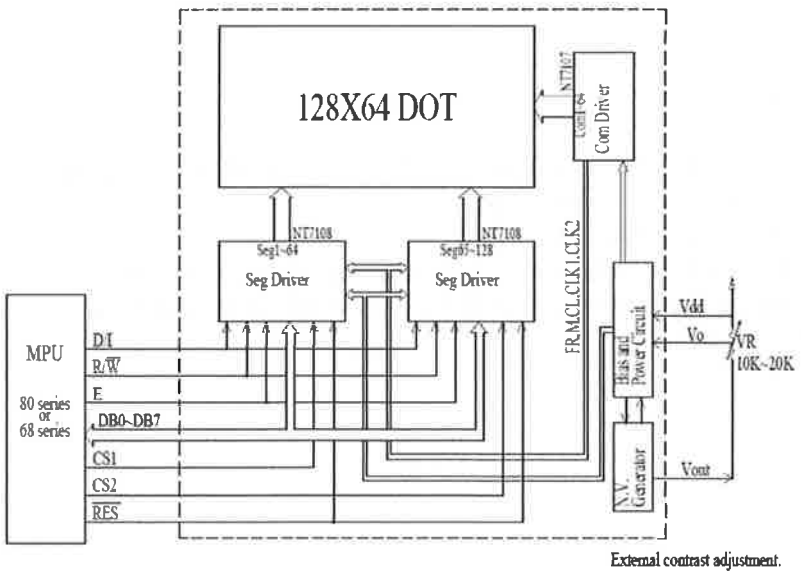


Figura 1: Conexión con un microcontrolador

Las instrucciones de control más sencillas son reset, activado por un nivel bajo, enable activado en alto, CS1 y CS2 activados en alto también.

Las demás instrucciones se dan por combinaciones de bits en los 8 bits de datos, y los controles RS (DI) y RW tal como se muestra en la tabla 3.

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display on/off	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is not affected. L: OFF, H: ON
Set address (Y address)	L	L	L	H	Y address (0-63)						Sets the Y address in the Y address counter.
Set page (X address)	L	L	H	L	H	H	H	Page (0-7)			Sets the X address at the X address register.
Display start line (Z address)	L	L	H	H	Display start line (0-63)						Indicates the display data RAM displayed at the top of the screen.
Status read	L	H	Busy	L	On/Off	Reset	L	L	L	L	Read status BUSY L: Ready H: In operation ON/OFF L: Display ON H: Display OFF RESET L: Normal H: Reset
Write display data	H	L	Write data								Writes data (DB0: 7) into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read display data	H	H	Read data								Reads data (DB0: 7) from display data RAM to the data bus.

Tabla 3: Combinación de bits para acciones de control

Se contó con un microcontrolador PIC18F4550 con las siguientes características principales

- USB V2.0
- ADC10
- Arquitectura optimizada para compiladores en C. Con paquete de instrucciones extendido.
- Modos de ahorro de energía:

Run, Idle, Sleep, Idle mode currents, Sleep mode currents

- PLL de alta precisión para velocidad USB, al utilizar cristal.
- 40 pin PDIP

El microcontrolador en su encapsulado PDIP de 40 pines, proporciona el número suficiente de pines para trabajar con pantalla (14 pines), con protocolo RS485 (3 pines), lectura de canales analógicos (2 pines), trabajar entradas como botones (2 pines), futura comunicación USB (3 pines), pines de alimentación (4 pines) y el cristal (2 pines) y MCLR (1 pin). Quedan disponibles quedan 9 pines libres. En la Figura 2 se muestra la distribución de los pines de éste microcontrolador.

40-Pin PDIP

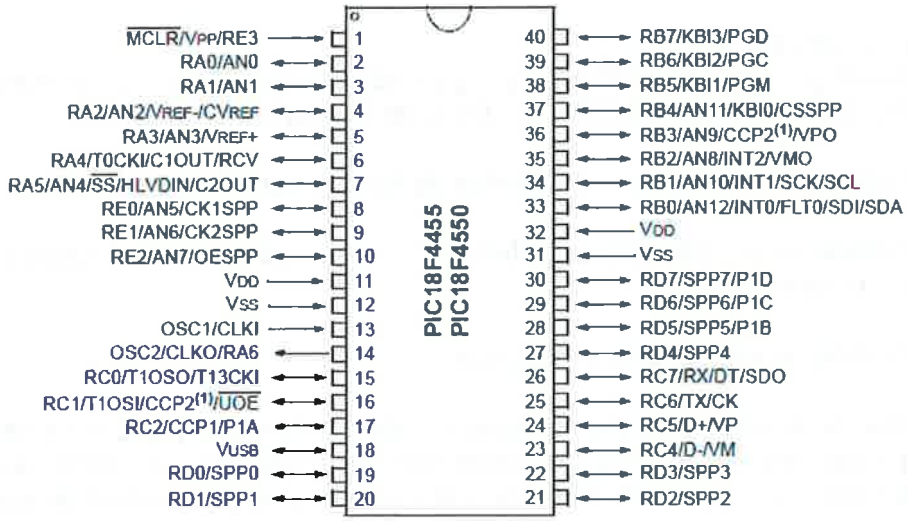


Figura 2: Patillaje PIC18F4550

Metodología

Diseño conceptual

1. Lluvia de ideas.

2. Escoger el microcontrolador de acuerdo a su compatibilidad con la pantalla en voltajes y corrientes. También tomando en cuenta la experiencia del programador con su lenguaje.

3. Escoger un circuito integrado para la conversión física de RS232 a RS485.

Diseño de Detalle

1. Investigar el funcionamiento y lógica de lectura/escritura de la pantalla GLCD modelo CFAG12864A-TMI-V de Crystalfontz.

2. Configurar el reloj, definir librerías de texto y pantalla.

3. Programar las interfaces gráficas, correspondientes al funcionamiento del programa.

4. Configurar los canales analógicos.

5. Unir las interfaces gráficas, según la lógica deseada y que se accedan a ellas mediante comparaciones con los controles, en este caso 2 botones, y 2 joysticks. Teniendo el usuario, siempre control de donde se encuentra en el programa.

6. Caracterizar el rango de bits que se tomará como estado de reposo mecánico para los 2 joysticks.

7. Configurar el módulo EUSART del microcontrolador, para envío RS232.

8. Realizar la interpretación de los canales analógicos, para el manejo de modo manual del robot. De manera que se pueda enviar información de los sentidos, y magnitud de la posición de ambos joysticks.

9. Localizar datos de envío y recepción, en las partes del programa que así lo requieran. Como son los datos correspondientes a los canales AD, y datos de control tanto enviados, como los que espera recibir para una correcta sincronización.

10. Programar un pin para que se active cuando un dato está en transmisión, para dejar lista una conversión física a RS485.

11. Conectar los transceptores en ambos lados a comunicar.

Pruebas y depuración.

1. De dibujo en pantalla GLCD.
2. De lectura e interpretación de los canales analógicos.
3. De recepción y transmisión RS-232. Depurar programa si es requerido.
4. De sincronización RS-232.
5. De recepción y transmisión RS-485.
6. De sincronización RS-485.

Desarrollo

Configuración de Reloj

Se tomó en cuenta que el proyecto de el robot explorador 2 tiene contemplada comunicación serial USB, y debido a los requerimientos del módulo USB, el tiempo del reloj interno debe ser 6 MHz para operación low-speed ó 48MHz para operación full-speed.

Se optó por la velocidad de 48MHz, para llegar a esta velocidad de procesamiento en el CPU a partir de un cristal del 20 MHz con el que se cuenta se debe entonces utilizar el PLL interno.
Un PLL es un circuito que puede generar una frecuencia, que sea múltiplo de una frecuencia de entrada.
El PLL de este microcontrolador recibe 4MHz de entrada y genera 96MHz de salida, por lo que se activó un pre-escalador, en nuestro caso dividimos los 20Mhz entre 5 para la entrada del PLL.
Posterior a esto se activa un pos-escalador (HSPLL) y se fija ÷2 (PLL5). También es necesario indicarle que dividirá ÷2 para la fuente del módulo USB (USBDIV).

Estas instrucciones están incluidas en la siguiente línea del código donde se indican los fusibles.
#fuses HSPLL, PLL5, NOWDT, NOPROTECT, NOLVP, CPUDIV2, USBDIV

Los fusibles restantes, indican que se deshabilita el watchdog, que se trabaja sin código protegido y que no se trabaja en modo de bajo voltaje.

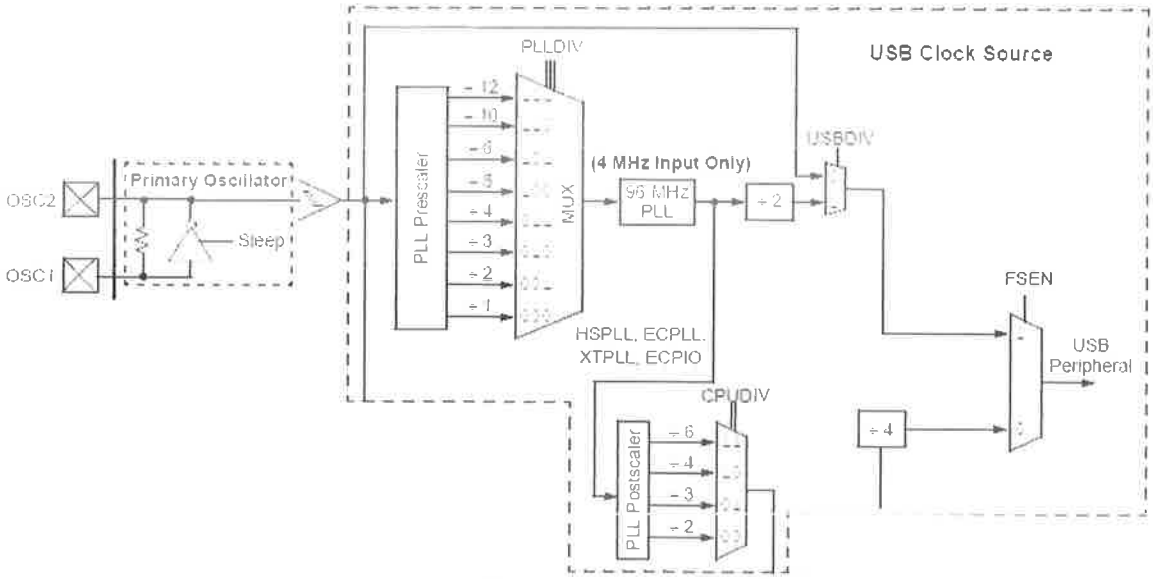


Figura 3: Opciones de Configuración de reloj USB y CPU

Programación GLCD

Para comenzar a dibujar las interfaces gráficas, se comienza a entender el funcionamiento de la pantalla, desglosando la Tabla 3 se tienen los siguientes controles básicos.
DISPLAY

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	1	1	1	D

Los datos aparecen cuando D es 1 y desaparecen cuando D es 0. Aunque los datos no aparezcan en pantalla cuando D=0, éstos permanecen en la memoria RAM de datos. Así pues los puedes aparecer de nuevo cambiando D de 0 a 1.

SET ADDRESS (Y ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

La dirección Y (AC0-AC5) de la memoria RAM de datos a desplegar, se activa en el contador de dirección Y. Una dirección se activa por instrucción y se incrementa automáticamente en 1 por una operación de lectura o escritura de datos.

SET PAGE (X ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	1	1	1	AC2	AC1	AC0

La dirección X (AC0-AC2) de la memoria RAM de datos a desplegar, se activa en el registro de dirección X. Escribiendo o leyendo desde o hacia la unidad de procesamiento, son acciones que se especifican en ésta página hasta que una nueva página sea activada.

DISPLAY START LINE (Z ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	1	AC5	AC4	AC3	AC2	AC1	AC0

La dirección Z (AC0-AC5) de la memoria RAM de datos a desplegar, se active en el registro de línea de comienzo y desplegada hasta arriba posición superior en pantalla. Cuando el ciclo de trabajo encargado de desplegar, sea 1/64, el número de datos total correspondiente a el número de líneas del GLCD. Se despliega entonces en la línea especificada en éste registro de instrucción

STATUS READ

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BUSY	0	ON/OFF	RESET	0	0	0	0

- **BUSY**

Cuando busy está en 1, el chip se encuentra ejecutando operaciones internas y no se aceptan instrucciones.
Cuando busy está en 0, el chip está listo para aceptar cualquier instrucción.
Cuando ON/OFF está en 1, el display físico se encuentra apagado.
Cuando ON/OFF está en 0, el display físico se encuentra encendido.

- **RESET**

Cuando RESET está en 1, el sistema esta siendo inicializado.
En ésta condición, no se aceptan instrucciones, excepto instrucciones de lectura de status.
Cuando RESET esta en 0, la inicialización ha finalizado y el sistema se encuentra en condiciones de operación normal.

WRITE DISPLAY DATA

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Escribe datos (D0-D7) en la memoria RAM de despliegue de datos. Después de la instrucción de escritura, la dirección Y se incrementa automáticamente.

READ DISPLAY DATA

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Lee datos (D0-D7) de la memoria RAM de despliegue de datos. Después de la instrucción de lectura, la dirección Y se incrementa automáticamente.

LIBRERIAS EN C para manejo de GLCD

Para el manejo de la pantalla se utilizaron las librerías, <HDM64GS12.c> y <graphics.c> del compilador PIC C, a continuación las funciones que ejecutan con sus parámetros:

<HDM64GS12.c>

glcd_init(mode)
* Debe ser llamada antes que cualquier otra función de pantalla.
- mode puede ser ON ó OFF (0,1) para apagar o encender el LCD.

glcd_pixel(x,y,color)
* Dibuja el pixel en las coordenadas y del color determinado.

glcd_fillScreen(color)
* LLena la pantalla del color seleccionado.

<graphics.c>

glcd_line(x1, y1, x2, y2, color)
Dibuja una línea de una coordenada a otra con el color seleccionado

glcd_rect(x1, y1, x2, y2, fill, color)

Dibuja un rectángulo de esquina a esquina en las 2 coordenadas con el color seleccionado, el rectángulo puede ser relleno o vacio (ON,OFF,YES,NO,0,1)

glcd_bar(x1, y1, x2, y2, width, color)

Dibuja una línea con el ancho y color seleccionado

glcd_circle(x, y, radius, fill, color)

Dibuja un círculo con centro en la coordenada y radio especificados.

glcd_text57(x, y, textptr, size, color)
Escribe texto con final nulo con la coordenada de la esquina superior izquierda de donde empieza el texto. Los caracteres son de 5 pixeles de ancho por 7 de alto. El parámetro size es un entero que escala el tamaño del texto.
Se especifica el color con ON, OFF, 0, 1

Se requería hacer un menú para distintos modos de funcionamiento del robot explorador, y se contó con los joysticks para desplazarse y dos botones.

Según el tamaño de letra y el espaciamiento diseñado, en la pantalla de menú caben 5 opciones. Se diseño un menú de 7 opciones para dejar establecido un código que permita desplazarse entre las opciones de menú, hacia arriba y hacia abajo. Actualmente solo se tienen 3 menús programados, pero se dejaron listos para futuras implementaciones.

Los cambios absolutos de pantalla fueron pensados de acuerdo al desplazamiento, borrando y escribiendo de abajo hacia arriba cada opción de menú si el efecto deseado fuese mover la pantalla hacia una opción de abajo y viceversa.

Se logró hacer un menú, que responde a desplazamientos del joystick de movimiento vertical, y el botón de selección nos lleva a la función de menú específica, y nos regresa de ésta hacia el menú en la posición inicial.

El código que hace las operaciones de dibujo y borrado más óptimas para la pantalla, se encuentra en la función Menu() y funciona mediante un switch, donde hay 18 cases.

Para ciertas opciones de menú hay mas de un case, ésto para optimizar el tiempo de respuesta en pantalla, ya que no se llega a la opción de menú siempre desde la anterior, si no que puede haber un corrimiento de menús, y un corrimiento de menús implica borrar toda la pantalla y rescribir todo.

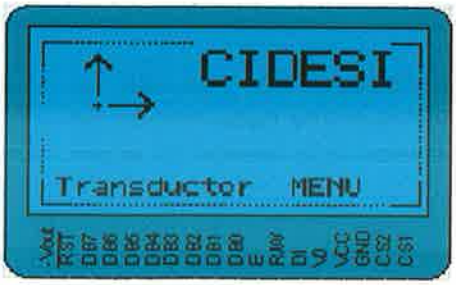
Con esta lógica, por cada elemento que se agregue al menú se deberán hacer 7 estructuras "case" más para la función seleccionadora "switch". Éstos 7 nuevos ´cases´ representan el nuevo cambio de pantalla hacia abajo apuntando al nuevo menú, los 4 desplazamientos del cuadro de selección hacia arriba, 1 desplazamiento de retorno a la nueva opción pero sin cambio de pantalla y el retorno a una pantalla anterior donde desaparece el nuevo menú.

a)

b)

c)

d)



e)

f)

g) h)

Figura 4: Simulación de interfaces gráficas, a) Presentación inicial CIDESI, b) Primeros 5 opciones de Menú, c) Desplazamiento de pantalla hacia 6ta opción de menú, d) Desplazamiento de pantalla hacia 7ma opción de menú, e) Modo de operación manual con joysticks hacia arriba y derecha, f) Modo de operación manual con joystick hacia arriba, g) Opción de menú de Calibrar Sensores para sensor 1 cercano, h) Opción de menú de Calibrar Sensores para sensor 1 lejano.

Se propone entonces una interfaz, que comience con una presentación del Centro de Ingeniería y Desarrollo Industrial como se muestra en la figura 4-a, después un tiempo que aparezca el menú donde el usuario podrá moverse mediante desplazamientos verticales y escoger alguna opción con el botón seleccionar (figuras 4-b,4-c y 4-d), y regresar hacia el menú con el mismo. A continuación algunos diagramas de flujo que ilustran el funcionamiento a grandes rasgos:

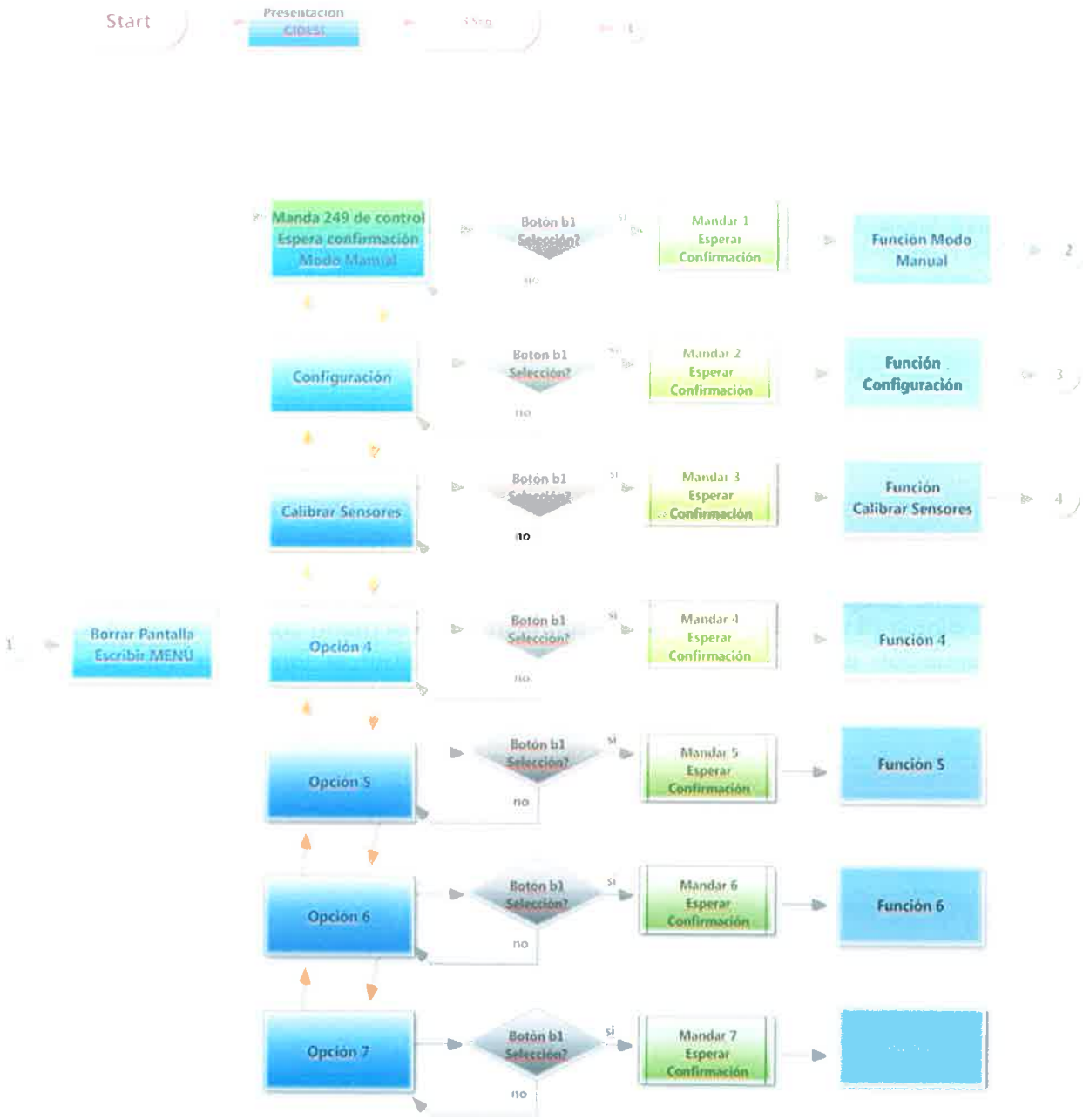


Figura 5: Menú principal

De las opciones que aparecen en el menú principal se tienen programadas 2, una de ellas es Operación manual, donde el usuario maneja el robot mediante los joysticks y tiene las opciones de volver a menú o bajar el transductor.

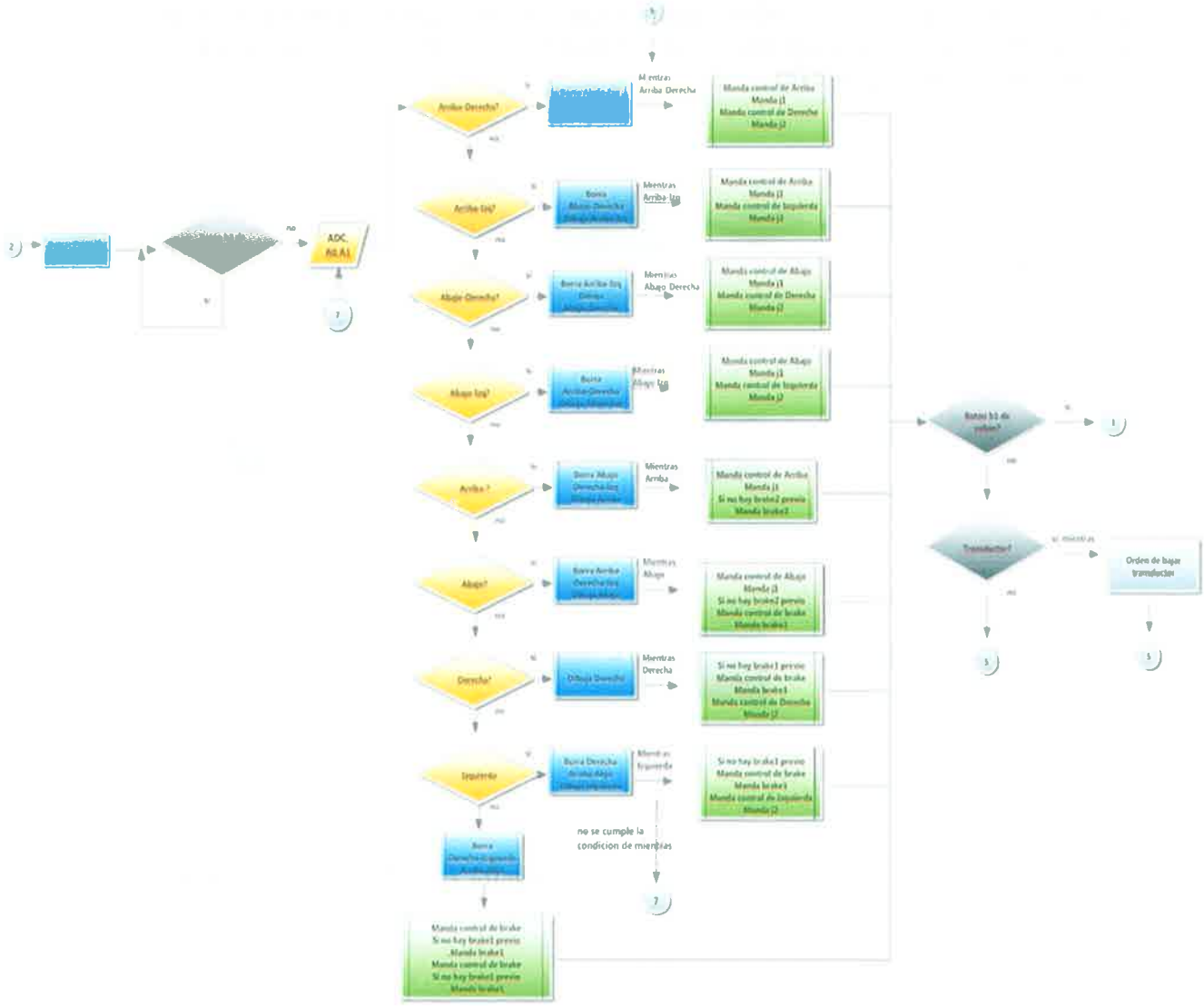


Figura 6: Operación Manual

La otra opción de menú programada es la de Calibrar Sensores donde se manda a ajustar cada sensor, primero por su rango cercano y luego el lejano. Los sensores tienen una configuración inicial, de esta forma si uno vuelve al menú y tan solo ha configurado dos sensores, tan solo éstos dos sensores tendrán nueva configuración.

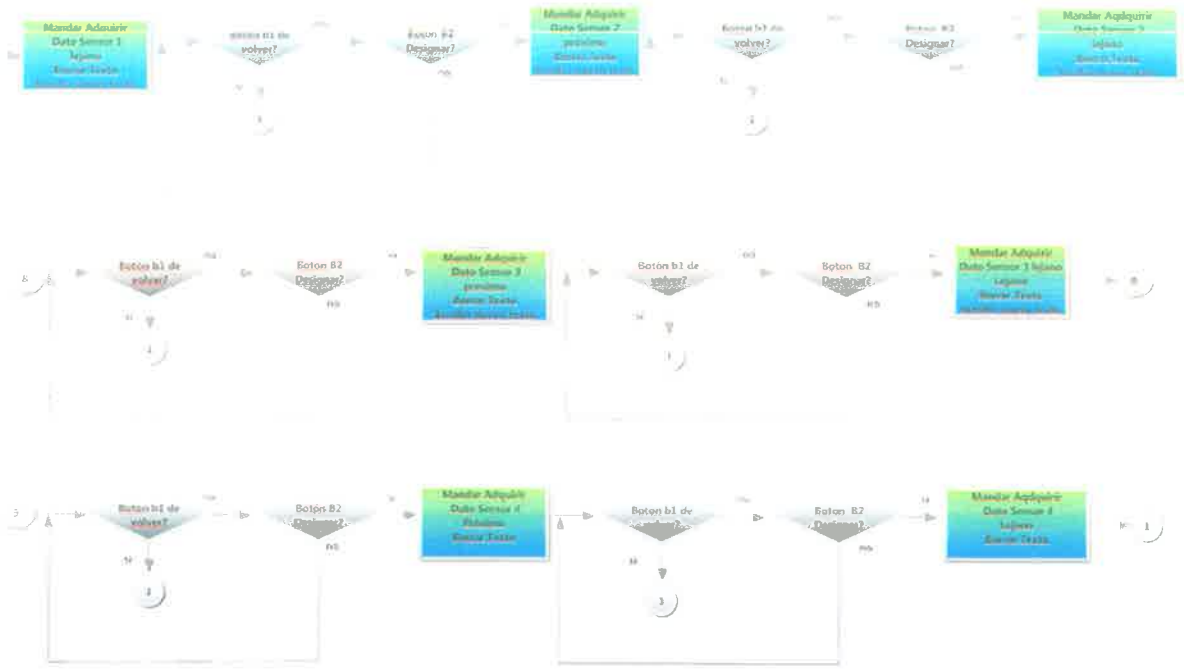


Figura 7: Calibración de Sensores

Para el desplazamiento entre opciones de menú, y para mandar datos para controlar el robot en modo de operación manual, se utilizan los joysticks, esto es, los canales analógicos. Por lo tanto, es necesario configurar el módulo ADC y realizar una lectura de los canales que nos interesan.

Para comenzar a entender una conversión analógica se muestra la lógica de programa paso a paso para configurar el módulo A/D:

Lógica Configurar el módulo A/D:

- Configurar los pines analógicos, voltajes de referencia y entradas-salidas digitales (registro ADCON0).
- Seleccionar el canal de entrada de la conversión analógica (registro ADCON2).
- Seleccionar el tiempo de adquisición (ADCON2).
- Seleccionar el reloj de conversión (ADCON2).
- Encender el módulo A/D.

1) Configurar las interrupciones A/D (si se requiere):
   - Limpiar bit ADIF (AD Interrupt Flag)
   - Activar bit ADIE(AD Interrupt Enable)
   - Activar bit GIE (Global Interrupts Enable)
2) Esperar el tiempo de adquisición (si se requiere).
3) Comenzar la conversión:
   - Activar el bit GO/DONE (registro ADCON0).
4) Esperar a que la conversión A/D se complete ya sea :
   - Consultando constantemente que el bit GO/DONE sea limpiado
     Ó
   - Esperando a la interrupción A/D.
5) Leer los registros de resultado (ADRESH:ADRESL); limpiar el bit ADIF, si se requiere.
6) Para la siguiente conversión volver al paso 1 ó 2 según se requiera.

Para el funcionamiento del ADC10 se utilizaron funciones del PIC C compiler donde se indican los canales a utilizar, y se designa el reloj interno RC.
Para frecuencias por encima de 1 MHz, el dispositivo debe estar en modo Sleep durante el tiempo de conversión, por lo que se habilita la interrupción AD, y el modo de interrupciones global para que pueda estar mandando a Sleep y despertando al final de cada conversión.

El pseudocódigo utilizado para la configuración de los canales A/D fue:

```
//*****************************************************************************************
Config ()
{
Habilitar Canales AN0 y AN1;
Habilitar reloj interno del ADC;
Habilitar interrupción AD;
Habilitar interrupciones globales;

}
//*****************************************************************************************
```

El Pseudocódigo utilizado para leer los datos del ADC10 durante todo el programa es la siguiente:

```
//*****************************************************************************************
void ADC_Lectura(void)
{
   Habilitar Canal 0;
Delay 20 us;
Conversión AD;
Modo de bajo consumo;     //Reduce potencia y ruido durante la conversión
Mientras no este lista la conversión {nop};
j1 = Lectura ADC;     //Lectura del canal 0, y asignación de valor.


Habilitar Canal 1;
Delay 20 us;
Conversión AD;
Modo de bajo consumo;      //Reduce potencia y ruido durante la conversión
Mientras no este lista la conversión {nop};
j2 = Lectura ADC;          //Lectura del canal 1, y asignación de valor
}
//*****************************************************************************************
```

Con éstos pseudocódigos de configuración y lectura, el programa estará haciendo conversiones A/D en los canales A0 y A1, y asignándoles el valor digital de 10 bits, a las variables j1 y j2 respectivamente. Entonces j1 tendrá el valor del joystick que corresponde a desplazamientos verticales y j2 tendrá el valor que corresponde al joystick que se mueve horizontalmente y controla el giro del robot.

El número de canales analógicos a utilizar son 2, correspondientes a los 2 joysticks que controlan los motores para la acción de avanzar y retroceder, y la acción de girar en ambos sentidos.

Para la habilitación de tan solo 2 canales analógicos se nos restringe a los canales AN0 y AN1 debido a que el registro de configuración para la configuración de canales ADCON1 en sus bits 3-0 se maneja de acuerdo a la Tabla 4.

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Tabla 4: Habilitación de canales A/D

Los Joysticks tienen una resistencia nominal de $10\,k\Omega$ $\pm 20\%$
Resistencia medida:

1. 9.24 $k\Omega$
2. 9.18 $k\Omega$

Estarán conectados al mismo voltaje de alimentación de 5V.
La corriente calculada entonces es de 0.5411 mA y 5446 mA lo que entra dentro de los parámetros.

Maximum output current sunk by any I/O pin.....25 mA

Maximum output current sourced by any I/O pin .....25 mA

El ligero cambio entre los potenciómetros corresponde a diferente sensibilidad en los joysticks.

Se conectan los joysticks de manera que en reposo el canal A0 y A1 leen la mitad de los bits del ADC10 (1023) y que la lectura del ADC en ADRES-H/L aumente al mover hacia arriba y hacia la derecha respectivamente, y viceversa.

Calibración de Joysticks por programa.

Es necesario caracterizar cada joystick en su estado de reposo. Están construidos para que en su estado de reposo presenten la mitad de su resistencia, lo que significa que entregan la mitad de su voltaje de alimentación al canal analógico en que estén conectados cada uno.
El voltaje de alimentación en este caso fue VD, sin embargo, este voltaje no permanece fijo, sino que presenta ligeras variaciones, asimismo, los potenciómetros tienen diferentes resistencias totales y mecánicamente el punto de reposo podría moverse un poco después de una acción mecánica en cualquiera de los 2 sentidos.
Es por eso que se caracterizan ambos joysticks haciendo mediciones del nivel en el que caen sus registros de memoria durante lapsos de reposo y después de movimientos en sus 2 sentidos.

El siguiente código se implementó para encontrar las mínimas y máximas mediciones de las lecturas en A0 y en A1 en estados de reposo:

```
int16 min1=1023, min2=1023, max1=0, max2=0, dif1=0, dif2=0;

 if (j1<min1)
   min1 = j1;
 if (j1>max1)
   max1 = j1;
  if (j2<min2)
   min2 = j2;
 if (j2>max2)
   max2 = j2;
 dif1=max1-min1; // nos da el máxima rango de bits para la posición de reposo en A0

 dif2=max2-min2; // nos da el máxima rango de bits para la posición de reposo en A1
```

Se sabe que el valor en reposo debe estar cerca de la mitad de los niveles del ADC10 por lo que se tiene $2^{\wedge 10}=1024$, $(1024-1)/2=511.5$ y el valor caería en 5011 o 512según el redondeo.

Durante una hora de prueba en reposo los resultados fueron los siguientes:

Estados/Valores	min1	max1	min2	max2	dif1	dif2
Reposo	473	537	479	533	64	54

Lectura, interpretación y envío de los datos de los canales ADC10 por RS-232.

La primera opción de menú en la pantalla será Modo Manual, donde se maneja el explorador mediante los 2 joysticks y los botones de transductor y volver a menú. Aquí es donde se tiene mayor variedad de datos a mandar por Tx, datos del ADC correspondientes a las 4 direcciones, datos de sincronización y las instrucciones volver y transductor.

Primero se considera que se tiene que hacer una conversión del dato de 10 bits del ADC10, a los 8 bits que maneja la comunicación RS232.
Para perder menos información se convertirán los datos de cada joystick en 8 bits hacia cada posición.

Por ejemplo el joystick uno manda un 1023 en su posición final hacia arriba, lo que debería corresponder a un 255 **'hacia arriba' cuando mandemos** nuestro dato y de manera contraria la posición final hacia abajo del mismo joystick da una lectura del ADC10 de 0 lo que deberá corresponder a un 255 **'hacia abajo' al mandar el dato vía** RS-485.
Como se deben mandar todos los datos por el mismo canal Tx de 8 bits, se limitaron los datos a 248 para así tener los números del 249 al 255 como datos de control.
Así cuando mande un 255, el programa lector lo interpreta como que el **siguiente dato de 8 bits, será del joystick 'hacia arriba', cuando lea un 254 será 'hacia abajo', un 253 será el dato asignado al joystick "hacia derecha" y un 252 'hacia izquierda'.**
Quedan 249, 250, 251 y 252 para otros posibles controles.

Los números de 8 bits reservados para datos de control se presentaron en el programa con sus instrucciones de declaración ´defines´ respectivos, de la siguiente forma:

```
#define Arriba 255
#define Abajo 254
#define Derecha 253
#define Izquierda 252
#define Brake1 251
#define Brake2 250
#define Volver 249
```

Ahora tan solo queda hacer el código que mande el control que indique hacia que dirección corresponde el siguiente dato, identifique los dos canales del ADC10, los convierta a un número de 8 bits del 0-248y lo envíe.

Si 'arriba' corresponde a una lectura en el registro ADRES para j1 (recordemos que j1 guarda el valor del canal analógico A0), del rango 537-1024, su conversión a 248 seria la siguiente:

$$ADC248 = \frac{(j1 - \max 1) * 248}{1023 - \max 1}$$

Si 'abajo' corresponde a una lectura en el registro ADRES para j1, del rango 0-473, su conversión a 248 seria la siguiente:

$$ADC248 = \frac{abs(j1 - \min 1) * 248}{\min 1}$$

Si 'derecha' corresponde a una lectura en el registro ADRES para j2(recordemos que j2 guarda el valor del canal analógico A1), del rango 533-1024, su conversión a 248 seria la siguiente:

$$ADC248 = \frac{(j2 - \max 2) * 248}{1023 - \max 2}$$

Si 'izquierda' corresponde a una lectura en el registro ADRES para j2, del rango 0-479, su conversión a 248 seria la siguiente:

$$ADC248 = \frac{abs(j2 - \min 2) * 248}{\min 2}$$

En el programa se separó la información de los canales del ADC ya convertidos a 248 datos de 8 bits, en 2 variables globales:

```
signed int32 ADC8_1,ADC8_2;
```

Se busca entonces hacer una lectura de los 2 canales del ADC10, después identificar las posiciones de los joysticks, mandar los datos del estado de los joysticks vía RS-485

La parte de las condiciones, considera los estados en reposo, y según cada estado te manda a su ciclo while en evaluación en corto circuito. En C se evalúan las condiciones de izquierda a derecha, entonces si una condición no se cumple, los operandos de la derecha no se evalúan, esto es una evaluación en corto circuito.

Durante el proceso se simuló en proteus para ir depurando las funciones de dibujo en pantalla.
Para estas simulaciones se utilizó la instrucción printf(), para visualizar en pantalla los números de control, datos del de la conversión AD y unos letreros de control que se agregaron como se muestra:

strcpy(28Posición,"Abajo-Izquierda"); //visualización en pantalla
printf("%s\r\n",28Posición); //visualización en pantalla

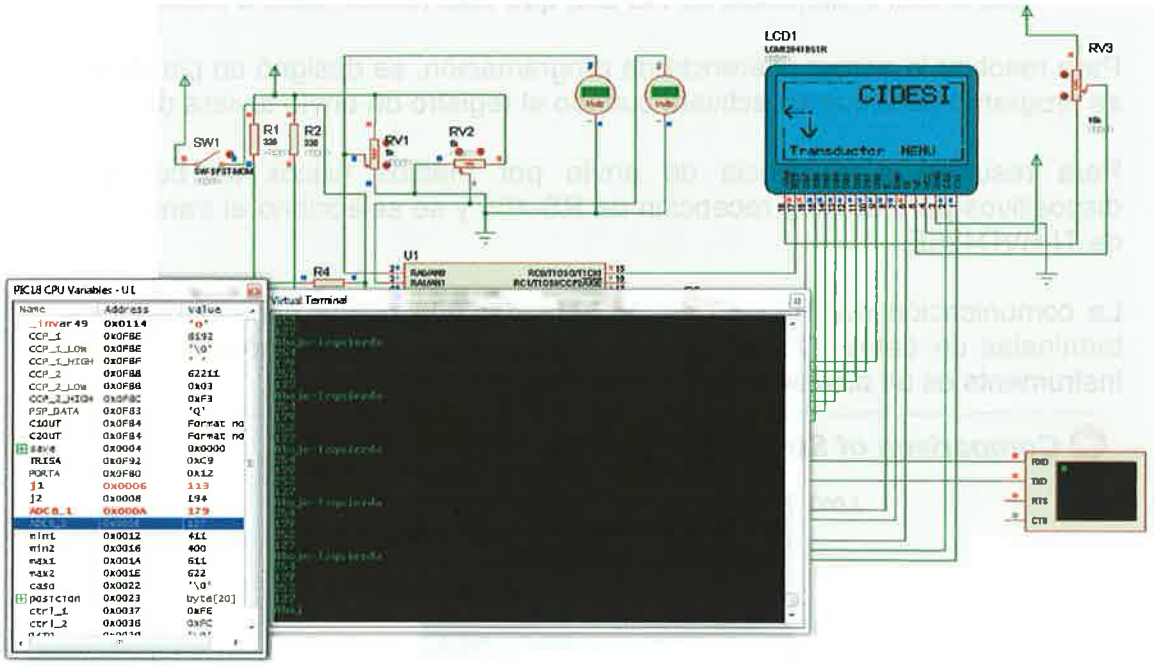


Figura 8: Simulación RS232

En la ventana de variables del PIC18 se pueden apreciar los valores de j1 y j2 del ADC10, y sus respectivas conversiones a datos de 0 a 248 bits. Estos últimos valores se despliegan en la terminal virtual, demostrando como manda los datos el control hacia el DSP para indicarle la dirección y el dato.
Para el programa final se utiliza ***putc();***

Adaptando RS232 a RS485

El microcontrolador PIC18F4550 nos permite generar comunicación RS-232 mediante la configuración de su módulo EUSART.

La diferencia principal entre los protocolos son las 2 siguientes:

- De programación, para RS-485 se requiere una señal en alto cuando se transmite por Tx.
- Física, ya que en protocolo RS-485 se envían los datos en voltaje diferencial a diferencia de RS-232 que está referenciado a masa.

Para resolver la primer diferencia de programación, se designó un pin libre, y se programó para que se activara cuando el registro de envío tuviera datos.

Para resolver la diferencia de envío por medios físicos se buscaron dispositivos para envío y recepción de RS-485 y se seleccionó el transceptor de TI HVD485E.

La comunicación por RS-485 es el estándar que puede conectar equipos terminales de datos (DTE) directamente sin necesidad de módems. Texas Instruments es un proveedor líder en transceptores RS-485.

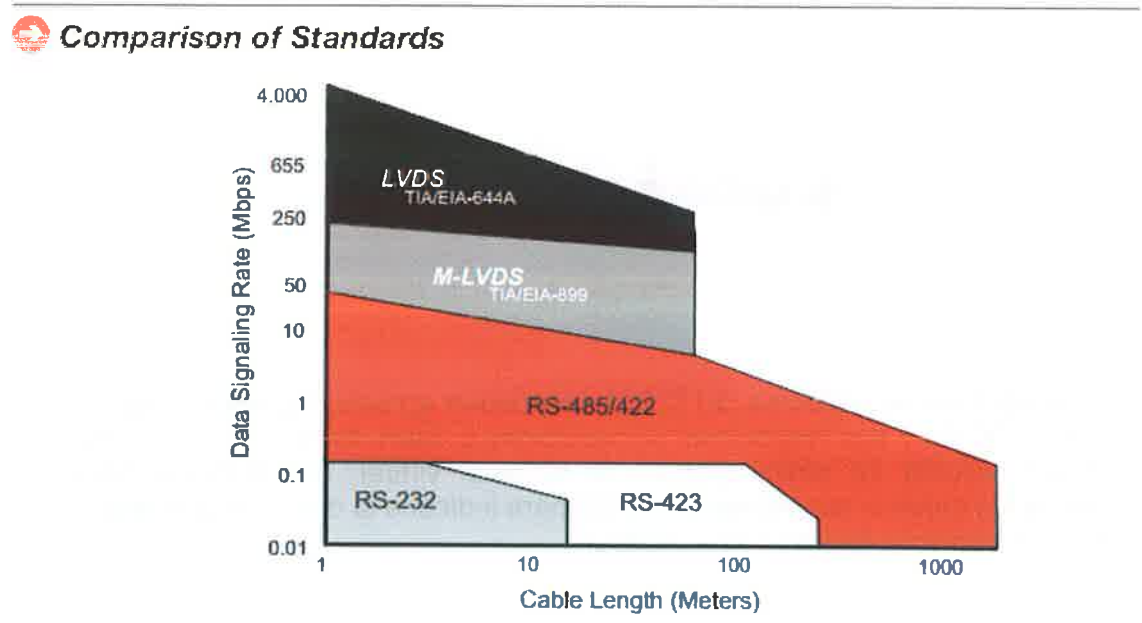


Figura 9: Comparación de estándares.

Como se puede apreciar en la imagen nos permite velocidades de hasta 50 Mbps, sin embargo como se convierte directamente de RS232 a RS485 se tiene como velocidad máxima la que permita el microcontrolador en su EUSART. La razón principal entonces para hacer éste cambio directo de protocolo, es la robustez ante el ruido, como muestra la gráfica permite mayores distancias a éstas velocidades.
Las siguientes son formas de establecer la línea de transmisión, siendo la primera directa, donde se requiere poca energía pero tiene muy alta reflexión en la línea por el desacoplo de impedancias (o falta de ellas).
La segunda opción minimiza las reflexiones a costo de la energía que disipan las R's. La tercer y más recomendable opción incluye a la segunda y agrega filtros de modo común, lo que la hace más robusta. Ésta última opción requiere de resistencias de precisión.

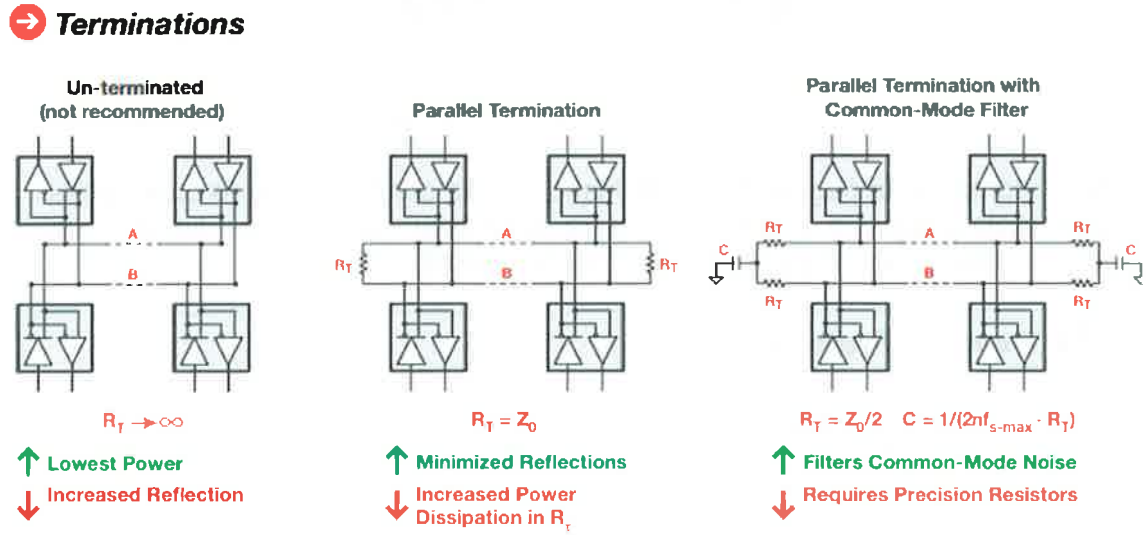


Figura 10: Terminaciones de línea

En las pruebas se utilizó la terminación en paralelo, que minimiza las reflexiones.

Device	DRV RCV	Duplex	Supply (V)	Features	Isolated	Signaling Rate (Mbps)	ESD (kV)	Receiver Fail-Safe	Nodes	Package
SN65HVD485E	1/1	Half	5	Half-Duplex Transceiver	No	10	15	Open	64	PDIP-8, SOIC-8, MSOP-8

Se utilizó la configuración Half-Duplex:

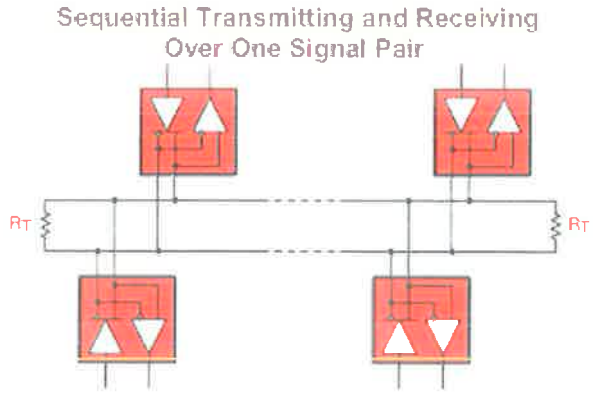


Figura 11: Configuración Half-Duplex

Se encontró en la página de Texas Instrument la configuración de conexiones con protecciones a transitorios, para un transceptor de su línea y un microcontrolador, tal como se muestra en la figura 12.

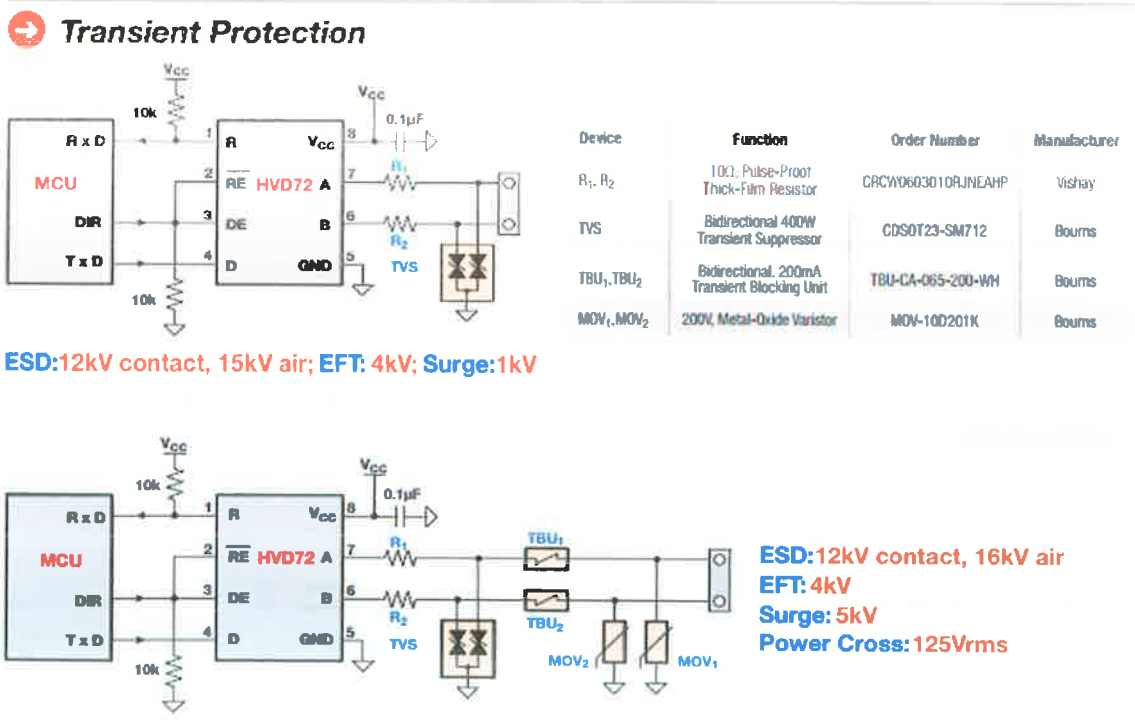


Figura 12: Protección ante transitorios

En caso de comunicarse más dispositivos se conectarían a la línea en sus canales A y B respectivamente de la siguiente forma

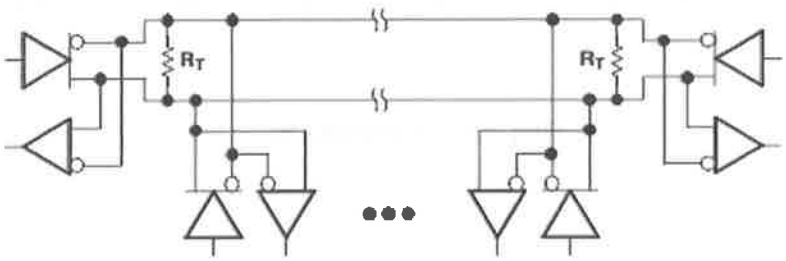


Figure 16. Typical Application Circuit

Figura 13: Conexión de más dispositivos a línea.

El consumo de energía es muy importante en varias aplicaciones. La corriente es entregada a la carga de la línea y al circuito transceptor. Para una configuración de línea típica de rs485, la carga sobre un driver activo consiste en todos los nodos receptores, más las resistencias terminales al final de cada línea.

La carga que presentan los nodos depende de la impedancia de entrada de los receptores. El estándar TIA/EIA-485-A define una unidad de carga como aquella que permite hasta 1mA. Con hasta 32 unidades de carga permitidas en la línea, la corriente total suministrada a todos los receptores, puede ser como máximo 32mA. El integrado HVD485E está catalogado como un dispositivo de ½ de carga, por lo que se pueden conectar hasta 64 en la línea.

Para una línea terminada con una resistencia de 120Ω en cada extremo, se tienen 25mA de corriente de salida diferencial siempre que el bus esté activo.

El circuito HDV485E típicamente puede soportar mas de 25mA para una carga de 60Ω, resultando así una salida de voltaje diferencial mas alta el que el mínimo requerido por el estándar.

El integrado utilizado necesita de 3 pines del microcontrolador, Tx,Rx y un pin que se active durante la transmisión que va conectado a las 2 entradas de control RE y DE, en una de ellas el integrado la niega para funcionar de acuerdo a la lógica mostrada en la Tabla 5.

La señal de salida viaja en voltaje diferencial entre los canales A y B. Los 8 pines del transceptor se muestran en la figura 14 y las compuertas internas en la figura 15.

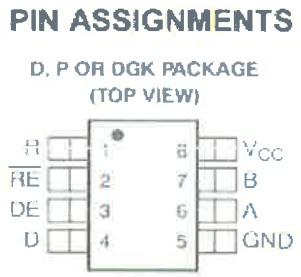


Figura 14: Patillaje HVD485E

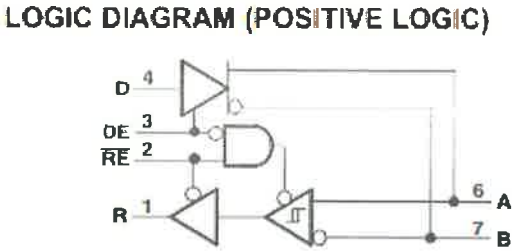


Figura 15: Diagrama Lógico

DRIVER				RECEIVER		
INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B	DIFFERENTIAL INPUTS $V_{ID} = V_A - V_B$	ENABLE $\overline{RE}$	OUTPUT R
H	H	H	L	$V_{ID} \leq -0.2$ V	L	L
L	H	L	H	-0.2 V $< V_{ID} < -0.01$ V	L	?
X	L	Z	Z	-0.01 V $\leq V_{ID}$	L	H
Open	H	H	L	X	H	Z
X	Open	Z	Z	Open circuit	L	H
				X	Open	Z

(1) H= high level; L = low level; Z = high impedance; X = irrelevant; ? = indeterminate

Tabla 5: Tabla de función

Resultados

Los cambios de interfaces gráficas fueron suaves y rápidos en todo momento.
Al ver que el movimiento entre opciones de menú era muy sensible, se designó un rango más amplio que el especificado como reposo mecánico de los joysticks (utilizado para operación manual), para quitarle sensibilidad, lo que resultó en un manejo mas práctico para el usuario.
A continuación se muestran unas imágenes de algunas de las interfaces gráficas:
La fotografía 1 es la primera interfaz gráfica en aparecer en el GLCD cuando se energizan la pantalla y el microcontrolador. Es la presentación de CIDESI.

Fotografía 1: Presentación Inicial

De las fotografías 2 a 4 se muestra el Menú completo con sus 7 opciones, donde se puede accesar a cada una, realizar las funciones internas, y volver al menú.

Fotografía 2: Primeras 5 opciones de menú

Fotografía 3: Desplazamiento de pantalla hacia 6ta opción de menú

Fotografía 4: Desplazamiento de pantalla hacia 7ma opción de menú

En la fotografía 5 se muestra la interfaz gráfica del modo de Operación Manual, donde se muestran las direcciones de los joysticks y es posible pedir que se baje el transductor ó volver al menú.

Fotografía 5: Opción de Operación manual con posiciones hacia arriba-derecha

En las fotografías 6 y 7 se muestra la gráfica dentro de la opción de menú de calibrar sensores. Se puede observar como al inicio te pide designar el valor del sensor 1 en su rango cercano, una vez designado, pide el rango lejano para el mismo sensor, y así funciona para los 4 sensores.

Fotografía 6: Opción de Calibrar Sensores para sensor 1 cercano

Fotografía 7: Opción de Calibrar Sensores para sensor 1 lejano

Quedó listo un programa con la interfaz gráfica, lectura de canales AD y envío/recepción de datos RS232 convertidos a RS485. A éste programa se le agregarán posteriormente funciones e implementaciones como parte del proyecto completo de segunda generación de robot explorador.

Futuro del Proyecto:

- Implementar el control del robot y sus variables de configuración, desde una computadora mediante comunicación USB y programación en LabView.
- Agregar funciones a las opciones de menú ya programadas del control.
- Aumentar la velocidad de comunicación RS-485, evaluando velocidad contra energía, ya que a mayor velocidad los dispositivos HDV485E consumen mayor energía. Si bien la velocidad máxima que permiten los transceptores HDV485E es de 10 Mbps, el microcontrolador tan solo es capaz de comunicarse a una velocidad máxima de 115,200 bps.
- Crear nuevos caracteres para la librería de texto. Por ejemplo los caracteres del logo de CIDESI.

- Mandar a modo de ahorro de energía el microcontrolador después de cierto tiempo en que lo joysticks estén en reposo y no haya ninguna acción de medición del transductor ó configuración.

Debido a que en posición de reposo hay cierta información en los canales AD no se puede despertar del sleep mediante interrupciones AD, así que se propone utilizar un operacional externo en configuración de comparador, para cada joystick.

Existen en el mercado circuitos integrados que han sido diseñados específicamente para realizar funciones de comparación. Entre ellos se pueden citar, por ejemplo, el LM311, el LM339 y el NE 529. Estos circuitos poseen una salida en modo colector abierto por lo que debe conectarse la misma a una resistencia de "pull-up" y a positivo.

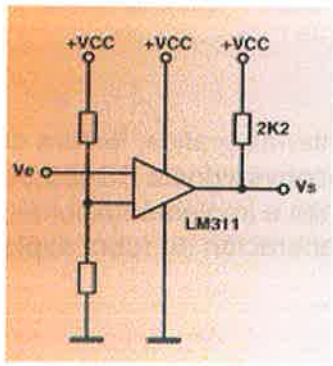


Figura 16: Comparador con resistencia pull-up.

Se propone utilizar el circuito integrado LM139 de Texas Instruments que consta de 4 comparadores de voltaje.

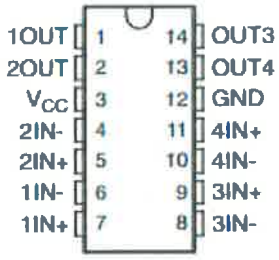


Figura 17: Patillaje CI Comparador LM139

Características principales:

- Amplio rango de voltajes de alimentación
  - Alimentación única de 2V-36V.
  - **Doble alimentación** $\pm 1V$ a $\pm 18V$.
- El rango diferencial de entradas aceptable es igual a la máxima alimentación de $\pm 36V$.
- El voltaje de entrada en modo común incluye masa.
- Voltaje bajo de salida de saturación.
- Salida compatible con TTL, MOS y CMOS.

- Si la tensión V+ es mayor que V- , la salida Vs será de nivel alto.

Se conectarían entonces como a V- los límites superiores del estado de reposo de ambos joysticks, esto es, los voltajes correspondientes a 537 y 533.

$$5V \rightarrow 1023$$
$$Vref-(1) \rightarrow 537$$

y

$$5V \rightarrow 1023$$
$$Vref-(2) \rightarrow 533$$

$$Vref-(1)=\frac{5V(537)}{1023}=2.6246V$$

$$Vref-(2)=\frac{5V(533)}{1023}=2.6051V$$

- Si la tensión V+ es menor que V-, la salida Vs será de nivel bajo

Se conectarían entonces como a V+ los límites inferiores del estado de reposo de ambos joysticks, esto es, los voltajes correspondientes a 473 y 479. La salida entonces la conectamos a una compuerta NOT y ya se tiene una señal digital que cambia a positivo cuando el voltaje es mejor a la referencia.

$$5V \rightarrow 1023$$
$$Vref+(1) \rightarrow 473$$

y

$$5V \rightarrow 1023$$
$$Vref+(2) \rightarrow 479$$

$$Vref+(1)=\frac{5V(473)}{1023}=2.312V$$

$$Vref+(2)=\frac{5V(479)}{1023}=2.341V$$

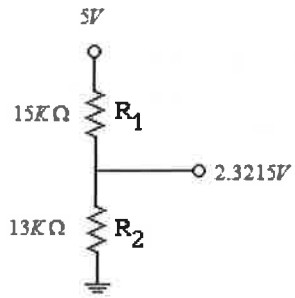
Se propone designar un pin del microcontrolador con capacidad de interrupción externa al que estarán conectadas las salidas de los comparadores, así se activa la interrupción y se despierta el microcontrolador cuando los joystick salgan de su estado mecánico de reposo.

Se propone obtener los voltajes mediante divisores de voltaje y utilizar tan solo 2, uno para los dos límites superiores y el otro para los dos límites inferiores.

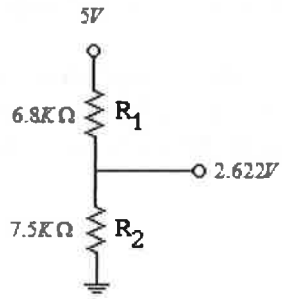
Se selecciona entonces el Vref+ que sea menor y el Vref- mayor.

$$Vref+ = 2.312V$$
$$Vref- = 2.624V$$

Para los divisores de voltaje se pueden utilizar valores relativamente altos, de esta forma, el divisor de voltaje no consume tanta corriente, y no afecta a los comparadores, ya que los OPAMP's son de alta impedancia de entrada. Considerando un ajuste a valores cerrados, o comerciales de resistencias se logra un voltaje Vref



y



Conclusiones

- Se logró la lectura e interpretación óptima de los 2 canales analógicos
- Los dibujos en pantalla se integran óptimamente al funcionamiento, al igual que el manejo de los joysticks y botones.
- Si bien durante ésta fase del proyecto, la velocidad de prueba de 9600 baudios tuvo buenos resultados, se recomienda implementar mejoras de sincronización en software, con el propósito de evitar fallas de recepción a mayores velocidades.
- Se contribuyó al desarrollo de tecnología en México, como parte de un proyecto activo. Me di cuenta que, como mexicanos, tenemos el potencial para desarrollar nuestra propia tecnología, sin embargo normalmente se utiliza equipo extranjero.
- Se lograron los objetivos del proyecto final de especialidad, que es el de llevar a cabo los conocimientos a una aplicación real, en este caso de programación de microcontroladores, y de electrónica.