INSTITUTO TECNOLOGICO DE TUXTLA GUTIERREZ

RESIDENCIA PROFESIONAL

REPORTE DE RESIDENCIA

ASESOR:

ING. ALVARO HERNANDEZ SOL

NOMBRE DEL PROYECTO:

ELECTROCARDIOGRAFO

(Software para la visualización cardiaca en un LCD grafico)

RESIDENTE:

GARCIA OCEGUERA JUAN CARLOS

ESPECIALIDAD:

ING. ELECTRONICA

Diciembre de 2008

INDICE

1.- INTRODUCCIÓN

El desarrollo del proyecto de residencia profesional se baso básicamente en la elaboración de un programa en el lenguaje de programación C, utilizando el programa mikroC para un microcontrolador PIC18f258. Este programa debería de ser capaz de manejar un LCD grafico de 240X128 pixeles con el objetivo de elaborar una etapa de visualización de una señal, en este caso dicha señal es la cardiaca. Esta etapa será complementada con el desarrollo de filtros digitales y analógicos para obtener la señal cardiaca. El programa fue elaborado de tal forma que fuese entendible realizándolo así con técnicas de programación tales como la programación estructurada que se basa en funciones para realizar tareas especificas, además se añaden librerías para no hacer tan extenso el programa principal. También se desarrollo el hardware necesario para la realización de pruebas físicas, cabe mencionar también que por los parámetros eléctricos que maneja el LCD grafico se desarrollo una fuente de voltaje para el funcionamiento de este.

2.- JUSTIFICACIÓN DEL PROYECTO

La verificación de enfermedades cardiacas esta muy extendido en nuestro país, y de sobremanera en nuestro estado. Añadiendo a esto último, tanto la geografía, y la marginación que se tiene en nuestro estado, genera una gran dificultad para que el grueso de la población cuente con un servicio medico, con el cual se pueda realizar chequeos de este tipo. Y considerando que los dispositivos que realizan esta función son de origen extranjero, muy caros y en algunas ocasiones de uso muy complicado. Por ello la necesidad de desarrollar un dispositivo medico de bajo costo y de fácil uso, que permita tener una mejor atención de este tipo de males. De igual manera, se estará determinando que tanto afecta, en el funcionamiento de un visualizado de este tipo, el uso de filtros digitales en lugar de filtros analógicos. Comprobando de igual manera, que tan complicado será implementarlos. La visualización se llevará a cabo en una pantalla LCD grafica, la cual es de bajo costo comparado con monitores y además que es muy portable y ahorradora de energía.

Con los resultados de esta investigación se pretende impactar desde varias áreas:

Científica: Formación de recursos humanos de alto nivel y publicación de los resultados en revistas nacionales arbitradas.

Tecnológico: Desarrollo de prototipos que realicen el filtrado de la señal cardiaca usando para esto filtros digitales. Logrando con esto una comparativa de este tipo de filtros vs. Filtros analógicos, dando como conclusión cual de los dos es mejor en esta aplicación.

Social: La posibilidad de proporcionar un sistema que ayude a los médicos, aun los no especialistas en el estado de Chiapas, para llevar a cabo este tipo de análisis. Además de contar con un dispositivo de bajo costo, en comparación de los sistemas de tipo comercial que muchas veces tienen muchas opciones que no son utilizadas, trayendo consigo un alto costo de dichos aparatos.

3.- OBJETIVO DEL PROYECTO

Objetivos generales

Desarrollar un programa para un microcontrolador PIC18f258 para la visualización de la señal cardiaca en una pantalla LCD grafica de 240x128 pixeles.

Objetivos específicos:

- Desarrollar una fuente de voltaje que cumpla con las especificaciones eléctricas del LCD gráfico.
- Diseñar el hardware para las pruebas del programa.
- Programar funciones para la inicialización del LCD gráfico.
- Realizar funciones para la visualización de la señal.
- Programar funciones para guardar y leer datos para una memoria RAM.
- Desarrollar funciones para realizar un zoom a la señal.

4.- CARACTERIZACIÓN DEL ÁREA DE TRABAJO

El desarrollo del trabajo de residencia profesional se realizo en el área de desarrollo tecnológico del departamento de ingeniería electrónica. En esta área es en donde se desarrollan los proyectos tecnológicos que alberga el departamento de ingeniería electrónica, el área cuenta con los materiales y equipo necesarios a utilizar para el desarrollo de los proyectos.

5.- PROBLEMAS A RESOLVER

1.- Desarrollar una fuente de voltaje para la alimentación del circuito principal y del LCD gráfico.
2.- Desarrollo del Hardware para el circuito principal.
3.- Desarrollar funciones de programación que nos permita inicializar al LCD gráfico.
4.- Desarrollar funciones que nos permita visualizar una señal en el LCD grafico.
5.- Desarrollar funciones que nos permita guardar y leer datos para una memoria RAM.
6.- Desarrollar funciones para el manejo de la señal visualizada de tal forma que podamos seleccionar un rango de dicha señal y poderla ampliar (aplicarle un zoom).

6.- ALCANCES Y LIMITACIONES

El desarrollo del proyecto se enfocara al desarrollo de un programa para la visualización de la señal cardiaca en un LCD gráfico; guardarla y leerla de una memoria RAM y poderla ampliarla aplicándole un zoom. Se limitará a la visualización de una sola señal en nuestro LCD grafico debido a que este cuenta con poca resolución y visualizar mas señales causaría una distorsión de estas.

7.- MARCO TEÓRICO

Enfermedad cardíaca.

La enfermedad cardíaca es cualquier trastorno que afecta la capacidad del corazón para funcionar normalmente y cuya causa más común es el estrechamiento u obstrucción de las arterias coronarias que suministran sangre al corazón mismo.

Las enfermedades cardiacas pueden provocar:

- Arritmia (pulso irregular)
- Un embolismo (un coágulo de sangre que se ha desplazado desde su lugar de origen y ha taponado un vaso) que puede conducir a un bloqueo de las arterias
- Dilatación (extensión de las cámaras, causando que su interior se vuelva más grande debido al aumento de la presión)
- Hipertrofia (aumento del grosor de las paredes del corazón, lo que provoca una disminución en el tamaño de las cámaras y también una pérdida de flexibilidad del corazón)
- Incapacidad de mantener las elevadas demandas de oxígeno para eliminar los productos de deshecho durante la realización de ejercicio físico
- Contracción insuficiente (llenado o vaciado incompleto)

- Dolor, debido normalmente a la isquemia (falta de oxígeno debida a la reducción del flujo sanguíneo)
- Regurgitación (reflujo sanguíneo que provoca un aumento de la presión en los vasos sanguíneos de los pulmones y del hígado)
- Estenosis (apertura insuficiente de las válvulas del corazón)
- Muerte tisular (pérdida permanente de tejido cardiaco debido a la falta de oxígeno)

Las enfermedades cardíacas pueden deberse a:

- Abuso de alcohol
- Uso de esteroides anabolizantes
- Aterosclerosis
- Trastornos autoinmunes
- infecciones bacterianas
- Consumo de cocaína
- Defectos congénitos (presentes desde el nacimiento)
- Diabetes
- Dieta desequilibrada, con alto contenido en grasas y en colesterol
- Hipertensión
- Lesión o trauma
- Sedentarismo
- Hábito tabáquico
- Trastornos tiroideos (hipo o hiper función)
- Toxinas, como el mercurio y algunas veces fármacos quimioterápicos y retrovirales (tratamiento para el HIV/SIDA)
- infecciones víricas

Electrocardiograma.

Si se colocan unos electrodos sobre la piel a uno y otro lado del corazón, pueden registrarse las diferencias de voltaje, que son un reflejo de la actividad eléctrica del corazón en su funcionamiento habitual, registrarse fácilmente y analizarse después. El trazado de tales registros se conoce como **electrocardiograma (ECG)**.

Las **derivaciones** son las combinaciones de puntos corporales desde los cuales se registra rutinariamente el ECG. En las derivaciones **estándar o bipolares**, se coloca un electrodo en cada uno de los vértices del hipotético **triángulo de Einthoven**, siendo estos vértices los extremos de los brazos (muñecas) y la pierna izquierda (tobillo) (Fig.1). Se coloca también un electrodo de toma de tierra en el tobillo derecho. Estas derivaciones son de varios tipos, y registran las siguientes diferencias de potencial:

Derivación I: El polo negativo del electrocardiógrafo se conecta al brazo derecho y el polo positivo, al izquierdo. Por tanto cuando el lugar donde el brazo derecho se une al tórax es electronegativo con respecto al punto de unión del brazo izquierdo al tórax, se registrarán potenciales positivos, es decir por encima de la línea de voltaje cero del ECG. Cuando se den las circunstancias opuestas, se registrarán potenciales negativos.

Derivación II: El polo negativo se conecta al brazo derecho, y el positivo a la pierna izquierda. Como el brazo derecho es electronegativo con respecto a la pierna izquierda, se registrarán potenciales (u ondas) positivas.

Derivación III: el polo negativo está conectado al brazo izquierdo, y el positivo a la pierna izquierda. Esto significa que el electrocardiógrafo registra ondas positivas cuando el brazo izquierdo es negativo con respecto a la pierna izquierda.

El término bipolar significa que el ECG es registrado por dos electrodos aplicados al cuerpo.

Otras derivaciones que se utilizan en electrocardiografía son las **derivaciones monopolares** y las **derivaciones precordiales**. En estas derivaciones, uno de los electrodos, denominado eléctrodo de referencia, es construido por el propio aparato, combinando los polos de tres extremidades.

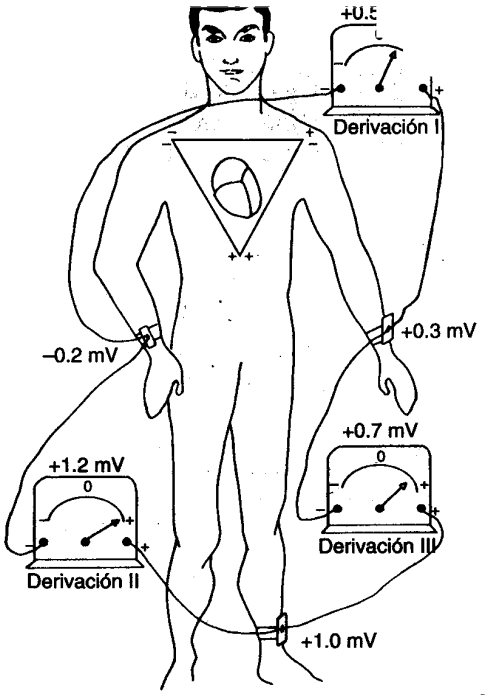


Figura 1.- Derivaciones

El otro registra las diferencias de potencial entre el electrodo de referencia y la extremidad correspondiente en el caso de las monopolares; o, como en el caso de las precordiales, entre el electrodo de referencia y un punto determinado del tórax.

El registro normal (Fig.2) del ECG de mamíferos consta de una **onda P,** el **complejo QRS** y una **onda T**. La onda P y el complejo QRS son ondas de despolarización... La onda P se debe a los potenciales eléctricos generados cuando las aurículas se despolarizan antes de cada contracción, y la onda QRS se produce al comenzar la contracción de los ventrículos. La onda T es consecuencia de la repolarización de los ventrículos. Existe una onda T auricular que queda enmascarada por el complejo QRS. El ECG es útil para obtener información sobre la actividad cardiaca.

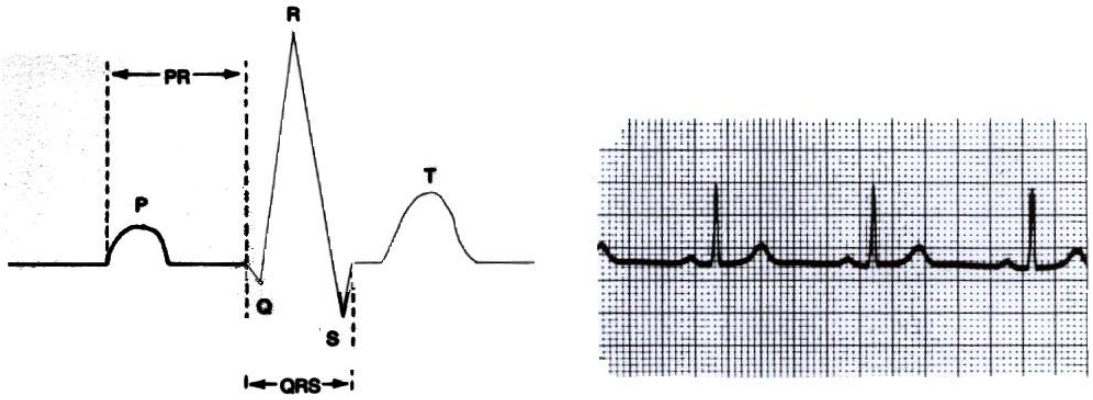


Figura 2.- Electrocardiograma normal

Electrocardiógrafo

Se llama **electrocardiógrafo** al aparato diseñado específicamente para registrar los ECG. Está constituido por un amplificador de señales eléctricas al que se conectan por un lado, los electrodos que se colocan en la superficie corporal, y por otro, un dispositivo de registro, bien gráfico en papel o pantalla, o bien informático. El equipo amplifica esos voltajes y produce la desviación de una plumilla, que los registra sobre un papel que se mueve a una velocidad de 25 mm/segundo. Una diferencia de potencial de 1 mV entre dos puntos del cuerpo produce una desviación de la plumilla de 1 cm. El espectro en frecuencias de la señal electrocardiográfica normalmente no tiene componentes arriba de los 60Hz en pacientes normales, por lo que se considera adecuado un ancho de banda de trabajo entre 0.05 y 150Hz para electrocardiógrafos.

Microcontroladores.

Un microcontrolador es un circuito integrado o un chip que incluye en su interior las tres unidades funcionales de un ordenador.: CPU, memoria y unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado. Aunque sus presentaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de especialización.

Un microcontrolador típico tendrá un generador de reloj integrado y uno pequeña cantidad de memoria RAM Y ROM/ EPROM/EEPROM, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores de analógico a digital, temporizadores, UART´s, USART´s y buses de interfaz serie especializados, como I2C y CAN.

El microcontrolador Pic18f258.

Este microcontrolador (fig. 4) es fabricado por Microchip familia a la cual se le denomina PIC. El modelo 18f258 posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y practico.

Algunas de estas características se muestran a continuación:

- Soporta modo de comunicación serial, posee dos pines para ello.

- Amplia memoria para datos y programa.
- Memoria reprogramable: La memoria en este PIC es la que se denomina FLASH; este tipo de memoria se puede borrar electrónicamente (esto corresponde a la "F" en el modelo).
- Set de instrucciones reducido, pero con las instrucciones necesarias para facilitar su manejo.

CARACTERÍSTICAS	16F877
Frecuencia máxima	DX-40MHz
Memoria de programa flash palabra de 16 bits	32KB
Posiciones RAM de datos	368
Posiciones EEPROM de datos	256
Puertos E/S	A,B,C,D
Número de pines	28
Interrupciones	16
Timers	4
Módulos CCP	2
Comunicaciones Serie	I2C, USART
Comunicaciones paralelo	PSP
Líneas de entrada de CAD de 10 bits	6
Juego de instrucciones	35 Instrucciones
Longitud de la instrucción	14 bits
Arquitectura	Harvard
CPU	Risc
Canales Pwm	4

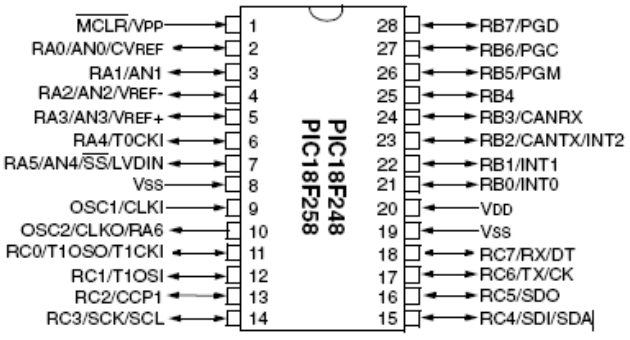


Figura 3.- PIC18f258

Puerto serial.

El puerto serial se constituye como una de las más básicas conexiones externas a un computador, y aunque hoy en día la más utilizada es su forma USB, el puerto serial ha estado junto a nuestros computadores por más de veinte años. Su principal función es enviar y recibir datos, bit por bit.

A groso modo, un puerto serial posee un conector estándar y trabaja con protocolo que permiten la conexión de dispositivos al computador. Estos puertos son denominados seriales debido a que este tipo de puertos serializa la información, en otras palabras, toma un byte de datos y transmite cada uno de los 8 bits uno a uno.

Los puertos seriales se conocen también con el nombre de puertos de comunicación o COM, y tienen la característica de ser bidireccionales. Ésta característica permite a cada uno de estos dispositivos tanto recibir como enviar datos. Su normal funcionamiento depende de un chip especial denominado UART debido a las siglas en inglés para "Universal Asynchronous Reciever/Transmitter". Este chip controlador toma la salida paralela del bus del computador y lo convierte en forma serial, lo que permite la transmisión de los datos a través del puerto.
Dentro de sus principales ventajas se encuentra la necesidad de sólo un cable para poder transmitir los 8 bits, sin embargo, se demora 8 veces más en realizar esta transmisión que si contáramos con 8 cables, como sucede con un puerto paralelo.

I2C.

I²C es un bus de comunicaciones serie. Su nombre viene de *Inter-Integrated Circuit* (Circuitos Inter-Integrados). La versión 1.0 data del año 1992 y la versión 2.1 del año 2000, su diseñador es Philips. La velocidad es de 100Kbits por segundo en el modo estándar, aunque también permite velocidades de 3.4 Mbit/s. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas empotrados (*Embedded Systems*) y generalizando más para comunicar circuitos integrados entre si que normalmente residen en un mismo circuito impreso.

La principal característica de I²C es que utiliza dos líneas para transmitir la información: una para los datos y por otra la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Como suelen comunicarse circuitos en una misma placa que comparten una misma masa esta tercera línea no suele ser necesaria.

Las líneas se llaman:

- SDA: datos
- SCL: reloj
- GND: masa

Conversión analógica-digital

Una conversión analógica-digital (CAD)(ó ADC) consiste en la transcripción de señales analógicas en señales digitales, con el propósito de facilitar su procesamiento

(codificación, compresión, etc.) y hacer la señal resultante (la digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

Digitalización

La digitalización o conversión analógica-digital (conversión A/D) consiste básicamente en realizar de forma periódica medidas de la amplitud de la señal y traducirlas a un lenguaje numérico. La conversión A/D también es conocida por el acrónimo inglés ADC (*analogic to digital converter*).

En esta definición están presentes cuatro procesos que intervienen en la conversión analógica-digital:

1. **Muestreo**: El muestreo (en inglés, *sampling*) consiste en tomar muestras periódicas de la amplitud de onda. La velocidad con que se toman esta muestra, es decir, el número de muestras por segundo, es lo que se conoce como frecuencia de muestreo.
2. **Retención** (En inglés, *Hold*): Las muestras tomadas han de ser retenidas (retención) por un circuito de retención (Hold), el tiempo suficiente para permitir evaluar su nivel (cuantificación). Desde el punto de vista matemático este proceso no se contempla ya que se trata de un recurso técnico debido a limitaciones prácticas y carece, por tanto, de modelo matemático.
3. **Cuantificación**: En el proceso de cuantificación se mide el nivel de voltaje de cada una de las muestras. Consiste en asignar un margen de valor de una señal analizada a un único nivel de salida. Incluso en su versión ideal, añade, como resultado, una señal indeseada a la señal de entrada: el ruido de cuantificación.
4. **Codificación**: La codificación consiste en traducir los valores obtenidos durante la cuantificación al código binario. Hay que tener presente que el código binario es el más utilizado, pero también existen otros tipos de códigos que también son utilizados.

LCD gráfico (GLCD).

Una pantalla LCD (fig. 5) o pantalla de cristal liquido de las siglas en inglés Liquid Crystal Display, es un tipo de pantalla generalmente a dos colores monocromáticos que nos permite, con la conexión y programas adecuados, visualizar caracteres e imágenes.

Funcionamiento:

Cuando la corriente circula entre los electrodos transparentes que circulan por el cristal liquido con la forma a representar (por ejemplo, un segmento de un número) el material cristalino se reorienta alterando su transparencia. Así, muestran cualquier forma que les pongas. Ya que puede dar corriente a todos los electrodos que dispones en la pantalla.

En un LCD grafico, todo el LCD funciona como un solo caracter dándonos mas libertad para mostrar información que separado en pequeños displays para cada letra (Alfanumérico).

Figura 4.- LCD gráfico

Memoria RAM

RAM son las siglas de random access memory, un tipo de memoria de ordenador a la que se puede acceder aleatoriamente; es decir, se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes. La memoria RAM es el tipo de memoria más común en ordenadores y otros dispositivos como impresoras.

Hay dos tipos básicos de memoria RAM

- RAM dinámica (DRAM)
- RAM estática (SRAM)

Los dos tipos de memoria RAM se diferencian en la tecnología que utilizan para guardar los datos, la memoria RAM dinámica es la más común.

La memoria RAM dinámica necesita actualizarse miles de veces por segundo, mientras que la memoria RAM estática no necesita actualizarse, por lo que es más rápida, aunque también más cara. Ambos tipos de memoria RAM son volátiles, es decir, que pierden su contenido cuando se apaga el equipo.

Memoria RAM KM62256ALP-10

Es fabricada por SAMSUNG (fig. 5), es tipo de memoria puede operar a varios rangos de temperatura y puede mantener los datos a un bajo voltaje.

- La organización de la memoria es de 32Kbx8
- La alimentación de voltaje es de 5 V ±10%
- Retención de dato a bajo voltaje de 2 V
- Salida de dato triestado y compatible con TTL
- Numero de pines 28
- Velocidad 55/70 ns.

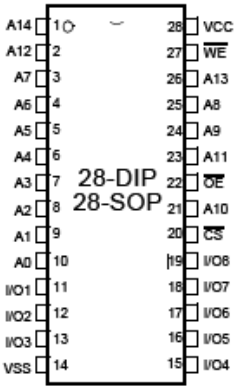


Figura 5.- RAM KM62256ALP-10

Contador binario

En electrónica digital, un contador es un circuito secuencial construido a partir de biestables y puertas lógicas capaz de realizar el cómputo de los impulsos que recibe en la entrada destinada a tal efecto, almacenar datos o actuar como divisor de frecuencia. Habitualmente, el cómputo se realiza en un código binario, que con frecuencia será el binario natural o el BCD natural (contador de décadas).

Contador binario 74LS393 (fig. 6)

Es un cantador binario de décadas y cada encapsulado contiene dos contadores trabaja a una velocidad máxima de 50MHz y trabaja típicamente a 5 V.

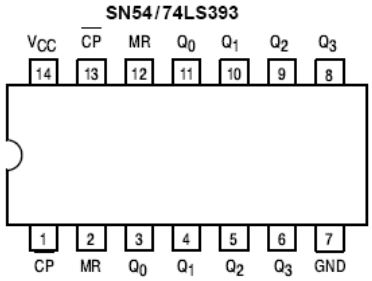


Figura 6.- Contador binario 74LS393

8.- DESARROLLO DE PROYECTO

8.1.- Desarrollo De Hardware

A continuación se hará mención del diseño de hardware necesario para llevar a cabo las pruebas del programa a desarrollar.

8.1.1.- Fuente De Voltaje Regulada A +5 Y -15 Volts.

Se desarrollo una fuente (fig. 7) de voltaje con las características mencionadas anteriormente con el objetivo de alimentar al LCD gráfico el cual utiliza un voltaje de +5 y -15 Volts. Además la fuente también debe alimentar el circuito de la tarjeta principal.

La fuente consta de un transformador de 36 Volts a 1 Amper con derivación central, un puente de diodos a 1Amper, 2 capacitores de 470 uF a 36 Volts, y 3 reguladores de voltaje: 7805, 7812, 7915.

El funcionamiento es simple, primero se reduce la el voltaje de 120 V ac a 36 V ac por medio del transformador. Posteriormente se rectifica la señal con el puente de diodos y se filtra con los capacitores. Para obtener el voltaje de +5 Volts se reduce el voltaje con un regulador de 12 Volts (C.I 7812) y luego con uno de 5 Volts (C.I 7805). Para el voltaje de -15 Volts se realiza con u regulador de -15 Volts (C.I 7915). La tierra del circuito se obtiene de la derivación central del transformador.

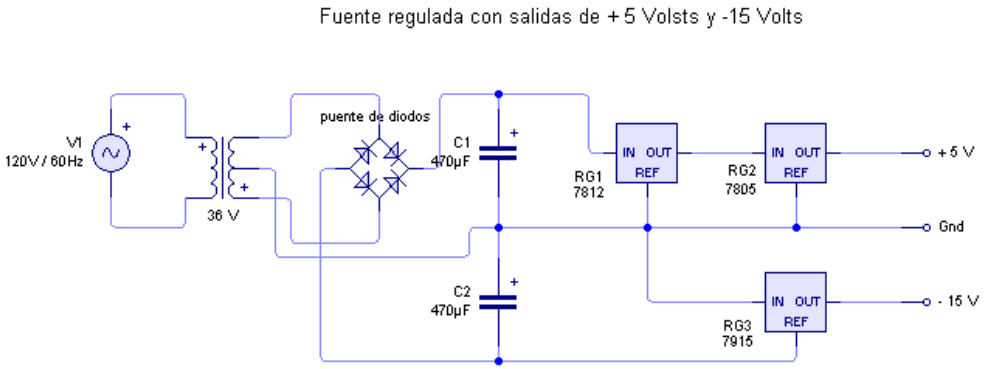


Figura 7.- fuente de voltaje regulada a +5 v -15 Volts

El diseño del circuito en PCB se muestra en la siguiente figura.

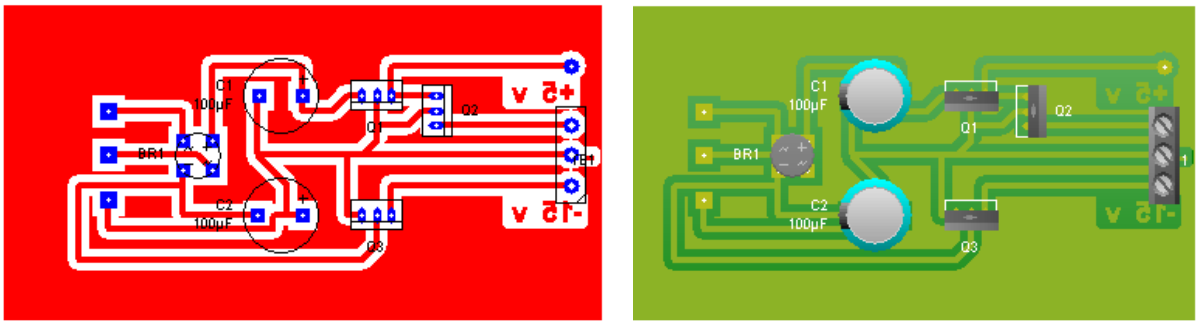


Figura 8.- Diseño de la fuente de voltaje en PCB

Diseño del circuito en una tarjeta para circuito impreso (fig. 9)

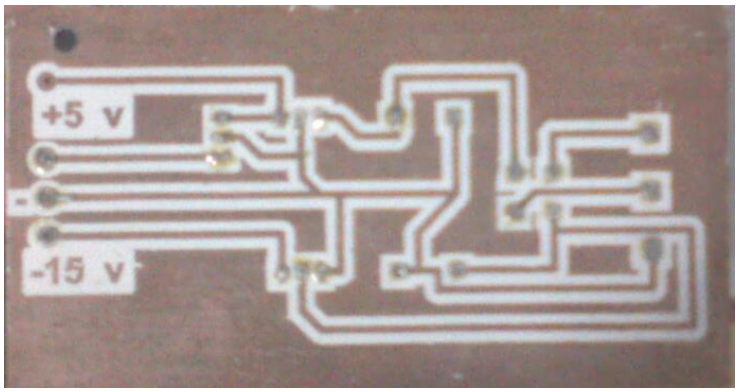


Figura 9 circuito impreso

Diseño final de la Fuente de Voltaje (fig. 10)

Figura 10.- Diseño final de la fuente de voltaje

8.1.2.- Diseño Del La Tarjeta Principal

A continuación se presenta el diseño del circuito principal (fig. 11). El circuito fue diseñado de tal forma que integre el PIC18f258, la memoria RAM y el LCD.

El circuito consta principalmente del PIC18f258, la memoria RAM 62256, 2 buffers 74ls541, 2 contadores binarios 74ls393 y un teclado de 4 botones (arriba, abajo, izquierda y derecha).

La implementación del circuito se desarrollo de la siguiente forma: se dispuso del puerto B del microcontrolador PIC18f258 para el manejo del LCD grafico, la memoria RAM y el teclado. Para ello se utilizaron buffers para poder multiplexar al LCD o al teclado, mientras que la memoria RAM se conecto directamente al puerto del microcontrolador. Se implementaron contadores binarios de décadas para direccionar a la memoria, para ello se utilizo el puerto RA4 para el incremento de los contadores y el puerto RA5 para el reset de estos. Se utilizaron los puertos RC3, RC4, RC5 para el control de lectura y escritura de la memoria. Para el control del LCD grafico se utilizaron los puertos RC0, RC1, RC2 y RA1; se utilizaron el puerto RA3 para el control de datos que se dirigen al LCD grafico y RA2 para el la habilitación del teclado. Se implemento el puerto RA0 como canal analógico de entrada de la señal a capturar y se dejaron libres los puertos RC6 yRC7 para una comunicación I2C futura.

Tabla 1.- Asignación de puertos.

PUERTO	ASIGNACION
Puerto B	Datos LCD gráfico Memoria RAM Teclado
RA4 RA5	Incremento Contadores Limpiar Contadores
RC3, RC4, RC5	Lectura y Escritura de la Memoria
RC0, RC1, RC2 y RA1	Control del LCD
RA3	Buffer del LCD Grafico
RA2	Buffer del Teclado
RA0	Canal Analógico
RC6 y RC7	Comunicación I2C

La forma de multiplexar la memoria, el LCD grafico y el teclado es de la siguiente forma: para hacer una lectura o escritura de la memoria se deshabilitan el buffer tanto del LCD como del teclado, para mandar a visualizar datos en el LCD se habilita su buffer correspondiente, se deshabilita la memoria por medio del los puertos de control de esta y también se deshabilita el buffer del teclado. Y para utilizar el teclado se habilita su buffer correspondiente, se deshabilita la memoria y el buffer del LCD. Cabe mencionar que el puerto B que es el que corresponde al bus, se debe de configurar ya sea como entrada de datos o como salida, según la tarea que se este realizando.

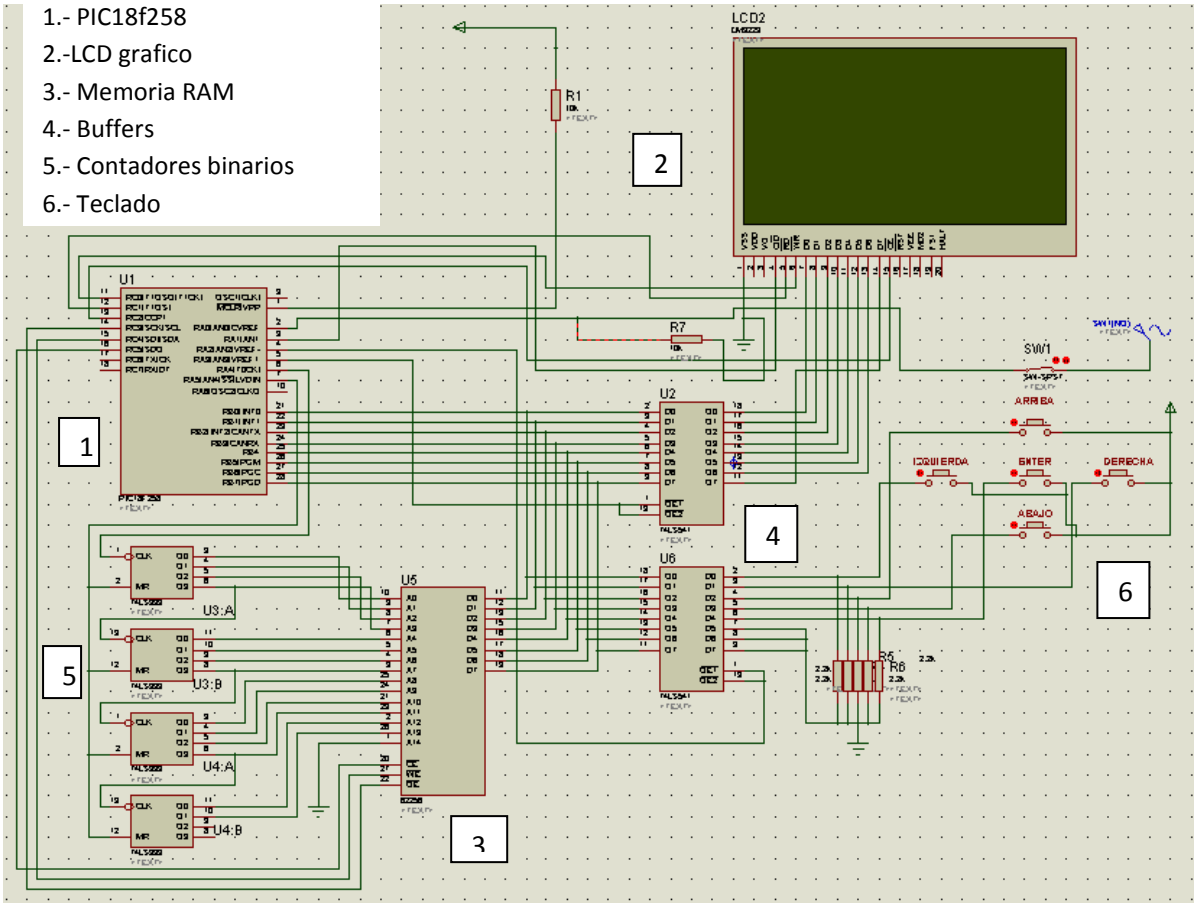
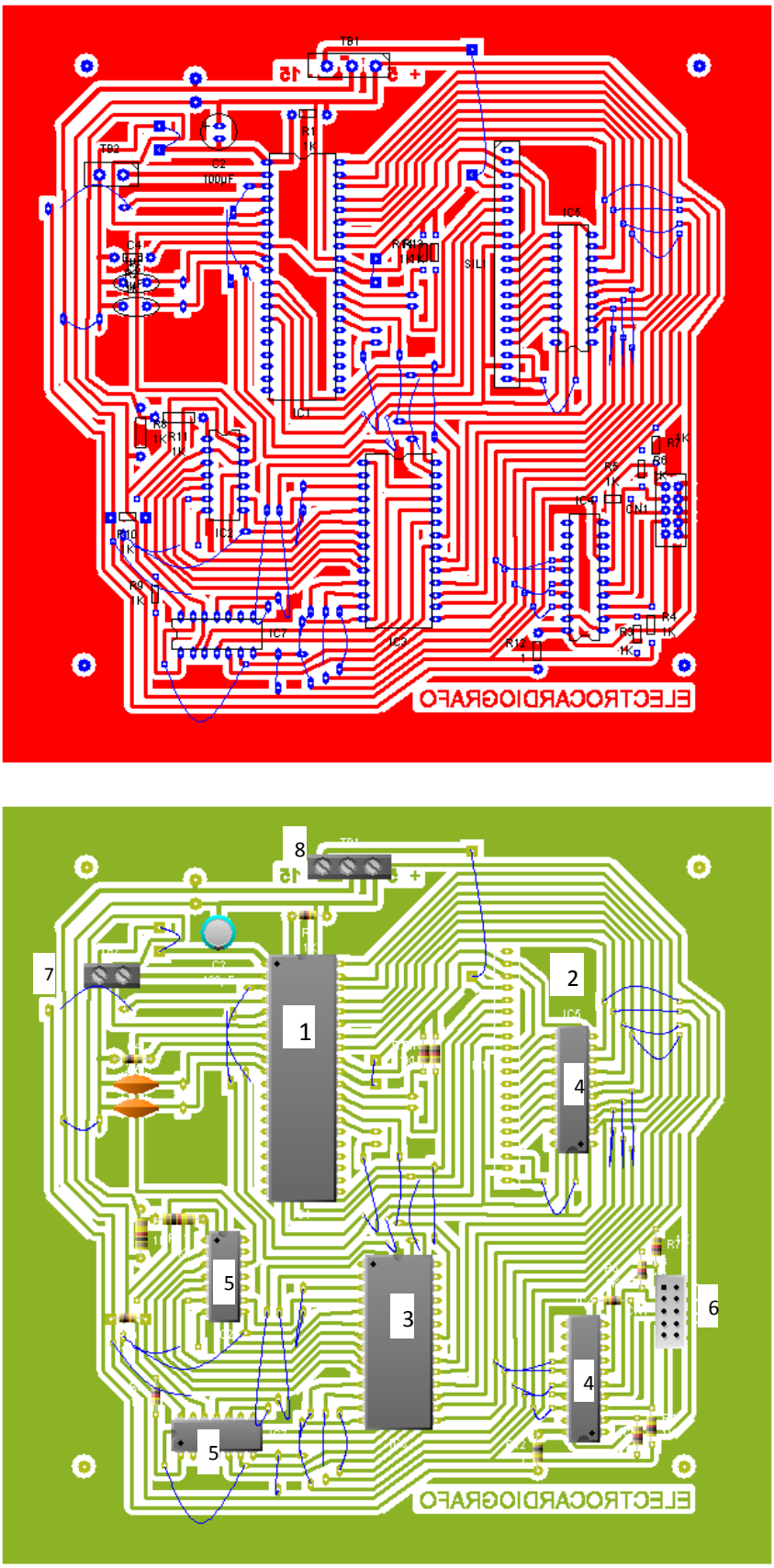


Fig. 11 Diseño del circuito principal

El diseño del circuito en PCB se muestra en la figura 12 (imagen a escala).



1.- PIC18f258.
2.-Pines para conexión del LCD grafico.
3.- Memoria RAM.
4.- Buffers.
5.- Contadores binarios.
6.- Pines para conexión del teclado.
7.- conector para la señal cardiaca.
8.- Alimentación del circuito.

Fig. 12 Diseño de la tarjeta principal en PCB

* Diseño del circuito principal en una tarjeta para circuito impreso. Vista inferior (fig. 13)
* Diseño del circuito principal en una tarjeta para circuito impreso. Vista superior (fig. 14)
* Diseño del circuito impreso del teclado en PCB (fig. 15)
* Tarjeta final del circuito impreso del teclado (fig. 16). Vista superior.
* Sistema completo en la figura 17.
* Vista interior (fig. 18) y superior (fig. 19).
* En la figura 20 se presenta el montaje del sistema con su fuente de alimentación.

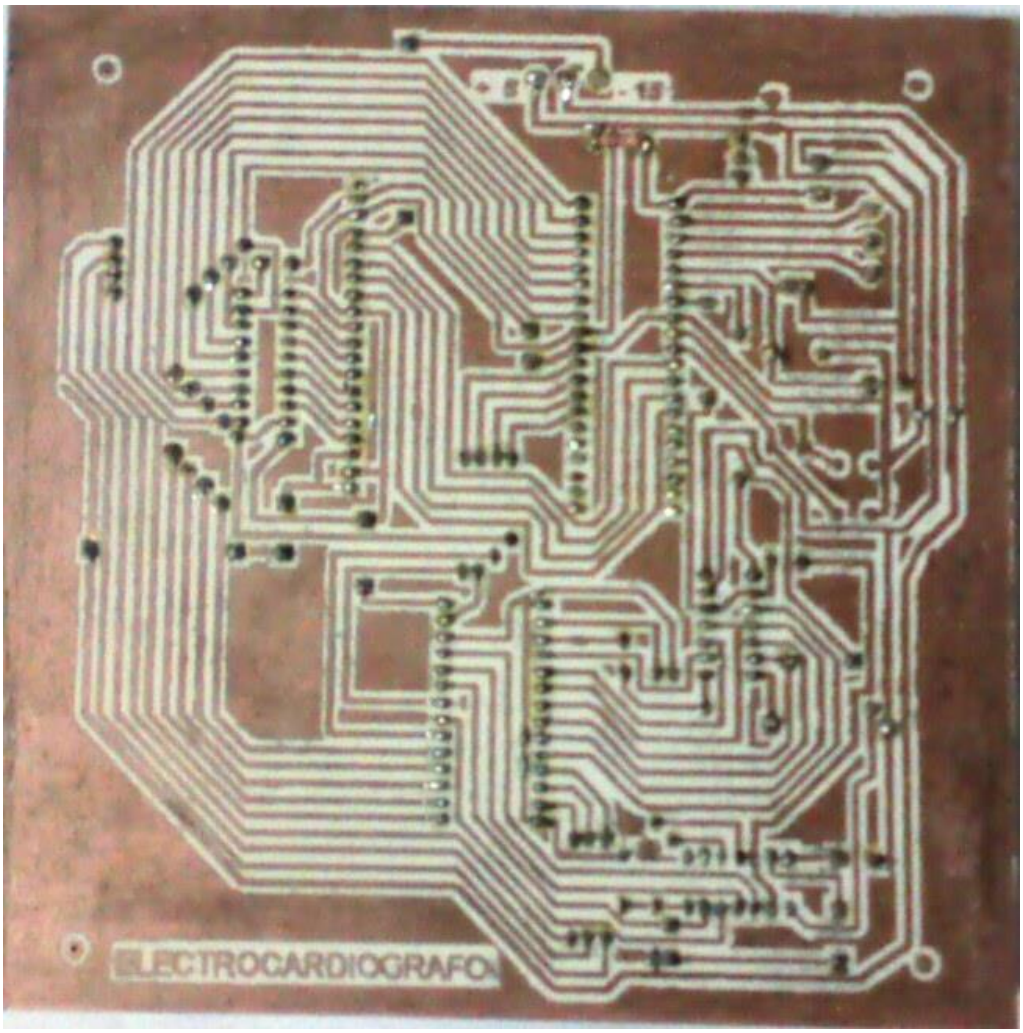


Fig. 13 Circuito impreso de la tarjeta principal

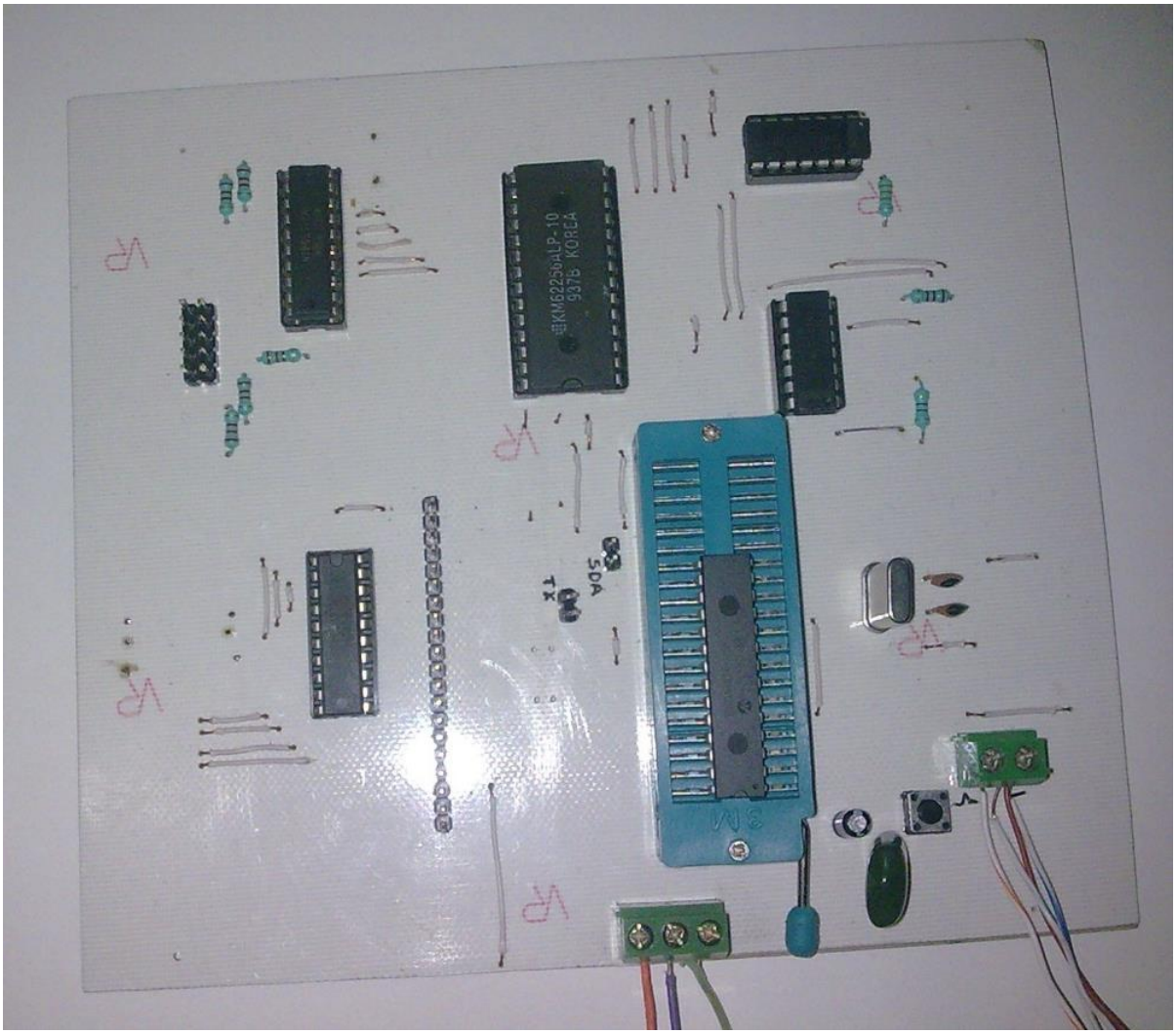


Fig. 14 Circuito impreso de la tarjeta principal vista superior

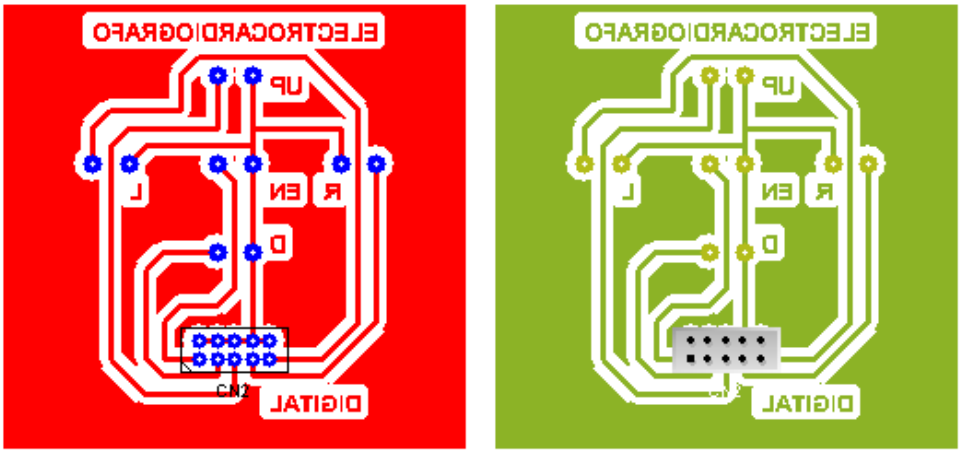


Fig. 15 Diseño del teclado en PCB

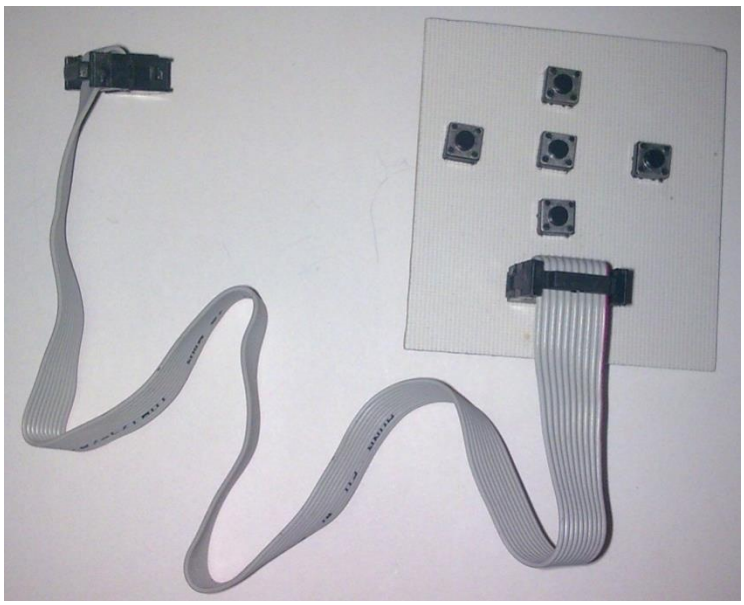


Fig. 16 Teclado del circuito

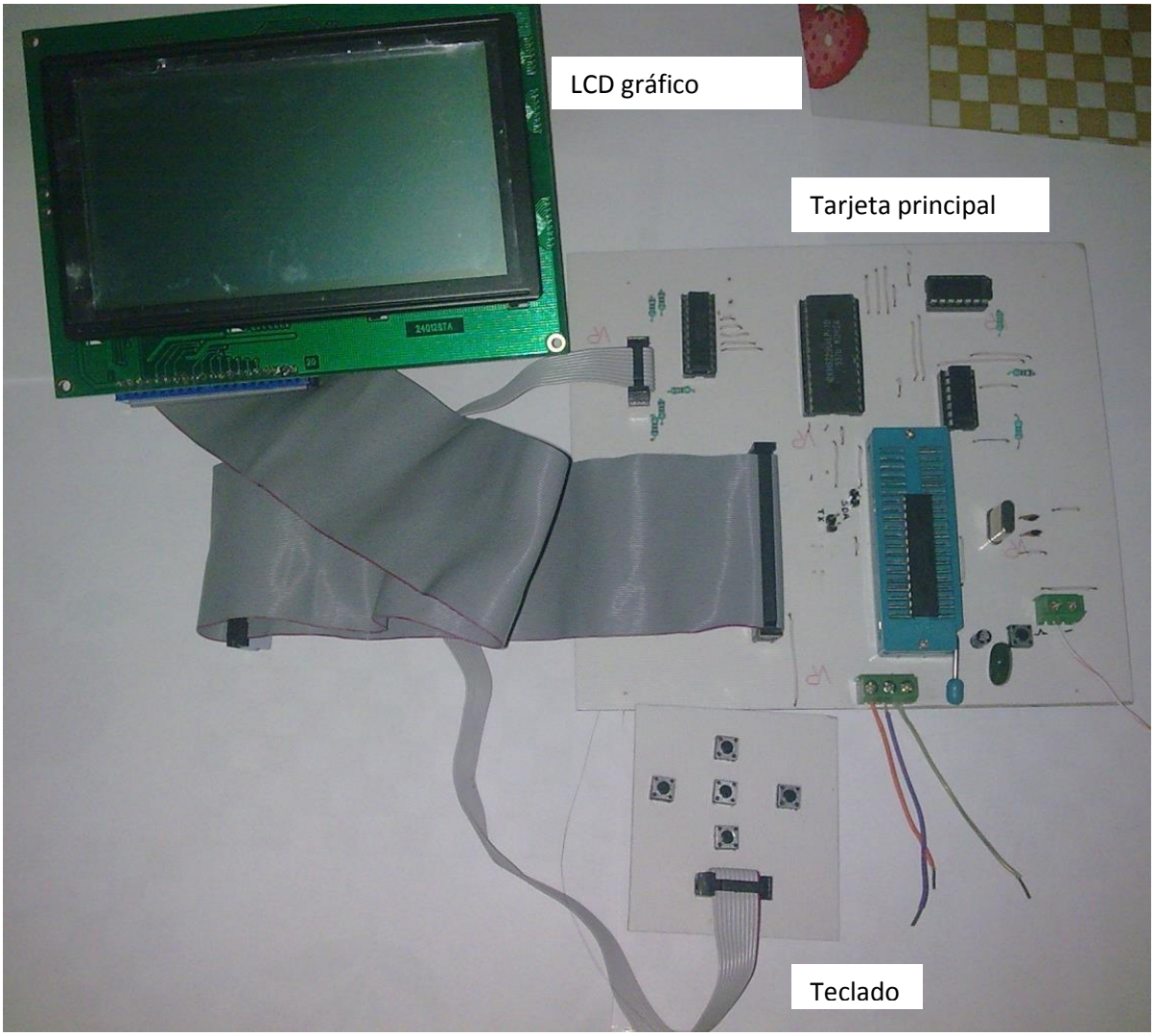


Fig. 17 Sistema implementado

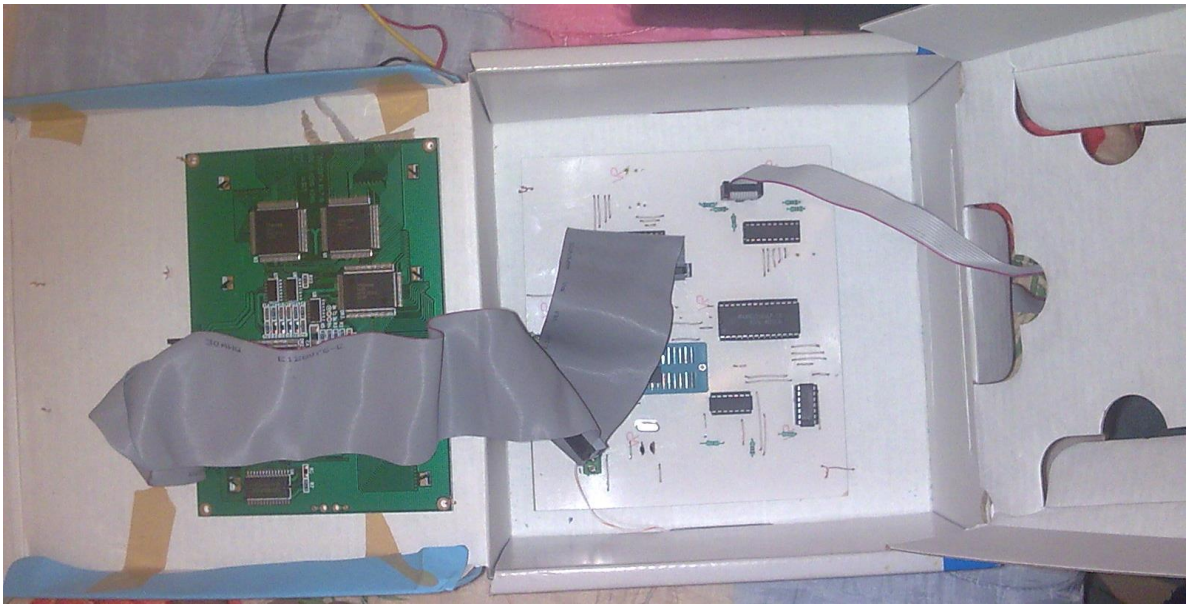


Fig. 18 Vista interior del sistema

Fig. 19 Vista superior del sistema

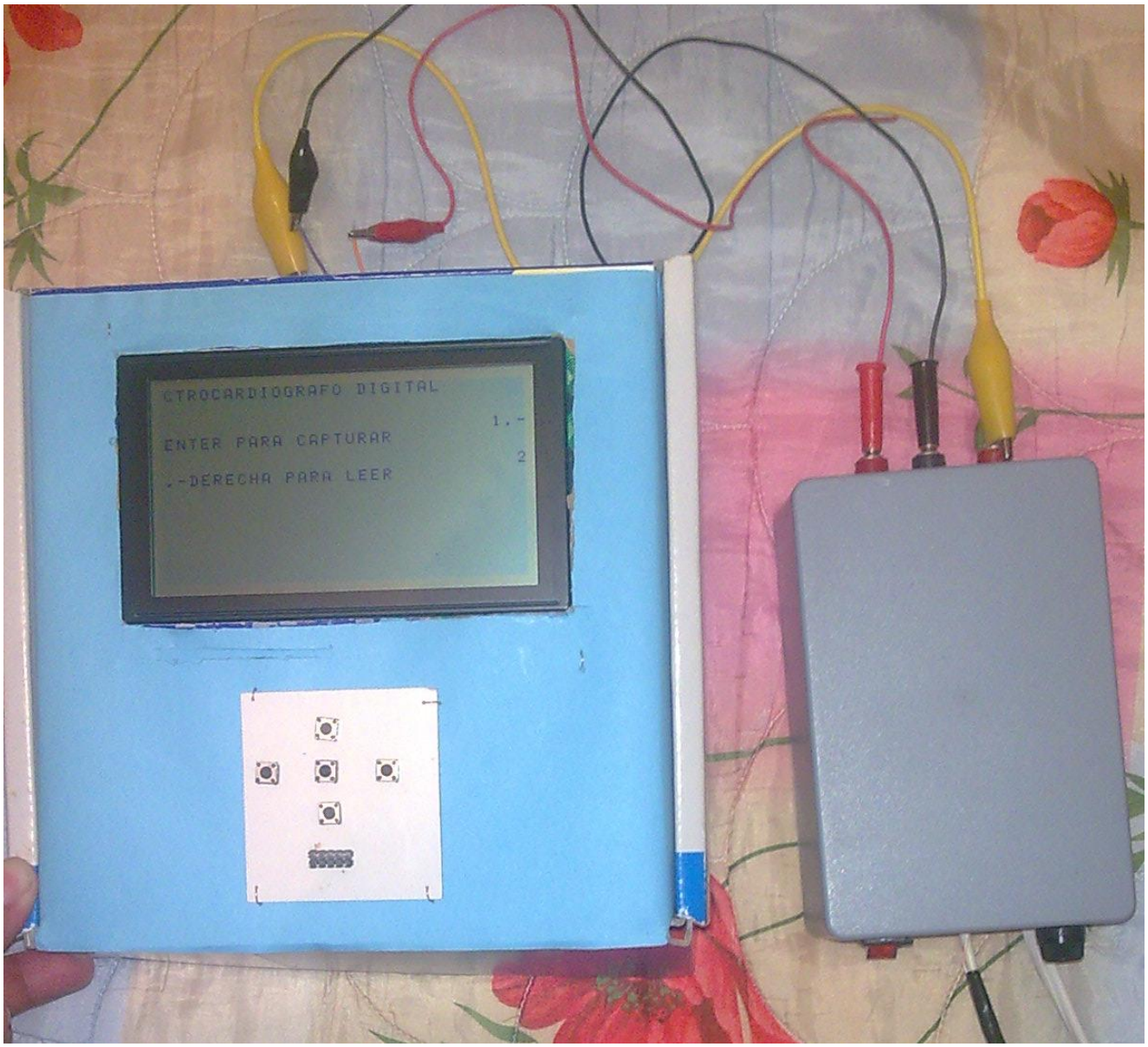


Fig. 20 Sistema con su fuente de alimentacion

8.2 Desarrollo Del Software

A continuación presentamos las etapas de programación y el diagrama de flujo (fig. 21).

Las etapas de programación están basadas en la lógica siguiente:

1.- Presentar un menú con las opciones de capturar muestra o leer muestra.
1.- Capturar la señal por medio de un canal analógico del microcontrolador y digitalizarla.
2.- Guardar los datos obtenidos en una memoria RAM.
3.- Graficar 2 de cada dato obtenido en la forma original de la señal.
4.- Leer la memoria para obtener los datos capturados y graficar cada 4 datos leídos.
5.- Poder analizar la señal detalladamente con la opción de seleccionar un rango de la grafica y aplicarle un zoom a dicho rango seleccionado.

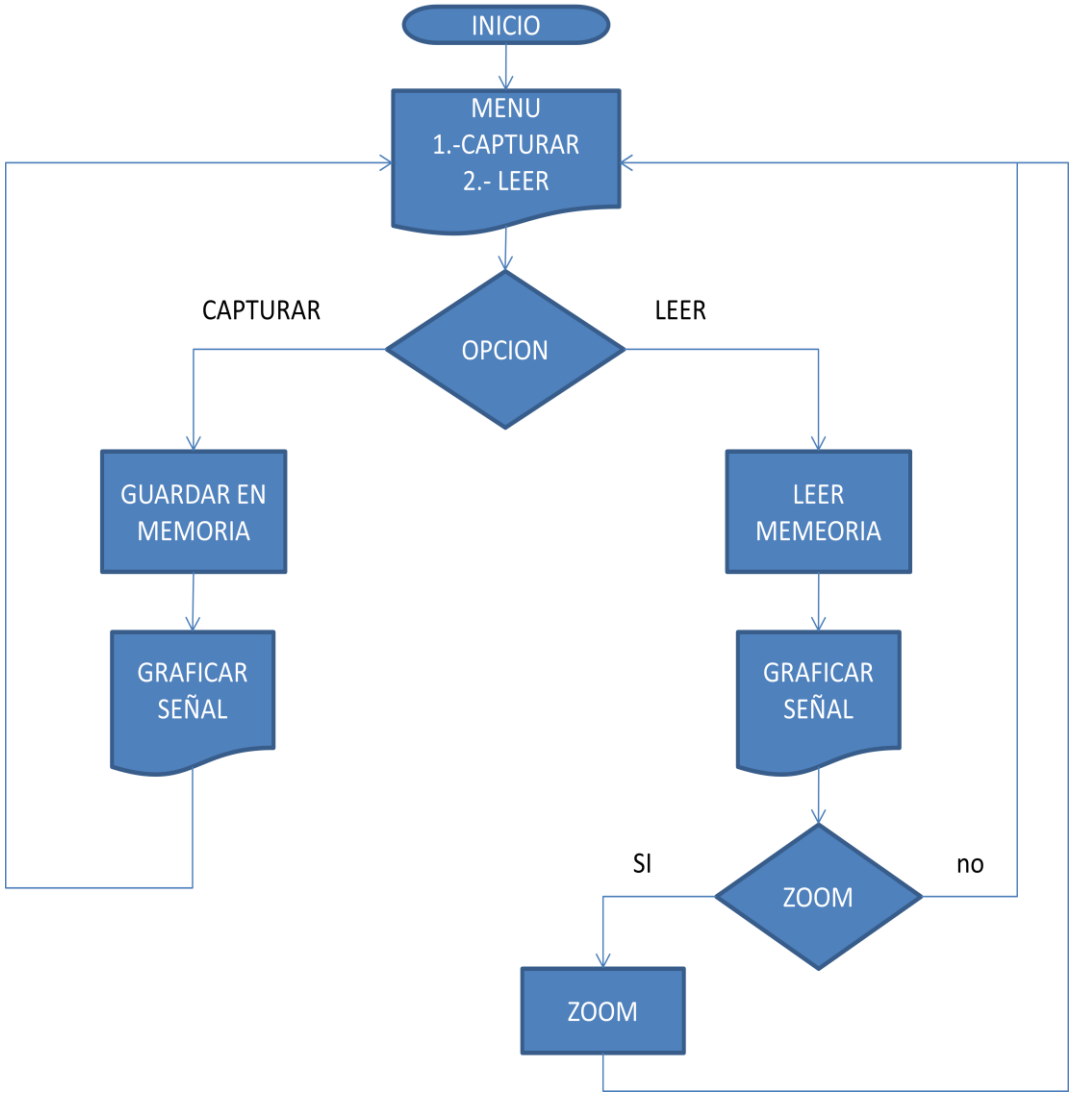


Fig. 21 diagrama de flujo.

8.2.1.- Librerías De Programación

Para la realización del programa se procedió a realizarlo por partes, es decir se segmentó en funciones y en librerías, las cuales se explicaran a continuación. El código de programación se presenta en el anexo A.

Las librerías que contiene el programa son las siguientes:

- funciones_graficas_basicas.c
- pantallas.c
- teclado.c
- memoria.c

8.2.2.- Librería funciones_graficas_basicas.c

Esta librería aloja las funciones necesarias para poder realizar la comunicación del LCD grafico con el pico y sobre todo para poder realizar las operaciones necesarias a los registros del LCD grafico.

Definiendo variables para los puertos

Función escribe_dato( )

Función para escribir datos en la memoria del LCD grafico. Sus parámetros son el dato a escribir y no regresa datos.

void escribe_dato(short d)

Ejemplo:

```
escribe_dato(dato);
```

Función escribe_comando( )

Función para escribir un comando al LCD grafico. Sus parámetros son el dato del comando a escribir y no regresa datos.

void escribe_comando(short d)

Ejemplo:

```
escribe_comando(0xC4);
```

Función envia_comando( )

Función para enviar un comando al LCD, sus parámetros son DATO a escribir, COMANDO a enviar y un numero 0 o 2 para ver que parte del dato se escribe y no regresa datos.

void envia_comando(int dat, short comand, short num)

Ejemplo:

envia_comando(direc,0x24,2);
envia_comando(0,0x80,0)

Función envia_cadena( )

Manda un comando para direccionar a la memoria de área de texto y escribe un carácter, sus parámetros son la dirección y el carácter a escribir y no regresa datos.

void envia_cadena(int direc,char das)

Ejemplo:

envia_cadena(adress,” a”);

Función envia_dato( )

Manda a encender o a borrar un pixel, sus parámetros son la dirección y 1 si se desea encender el pixel o 0 si se desea borrar y no regresa datos.

void envia_dato(int dir, char da)

Ejemplo:

Borrar un pixel:

envia_dato(adress,0);

Encender un pixel

envia_dato(adress,0);

8.2.3.- Librería pantallas.c

Esta librería aloja los códigos de las imágenes utilizadas para visualizarlas en el LCD grafico, los códigos fueron creados con la herramienta de mikroC.

8.2.4.- Librería teclado.c

La librería teclado.c aloja las funciones que se necesitan para entrar a modo teclado activando y desactivando el bus de datos que se esta utilizando.

Función teclado( )

Esta función nos permite activar el teclado con el que consta el hardware y activa el puerto del bus como entrada esperando a que sea oprimido un botón del teclado la función es de tipo vacio.

void teclado(void)

Ejemplo:

```
teclado();
```

Función restaurar( )

Esta función activa el puerto b como salida y activa el bus para el LCD gráfico. La función es de tipo vacio.

void restaurar(void)

Ejemplo:

```
restaurar();
```

8.2.5.- Librería memoria.c

Esta librería contiene 3 funciones para poder escribir datos en la memoria RAM o leer los datos que contiene esta.

Función retardo( )

Esta función se dedica a generar un pequeño retardo el cual se necesita para poder hacer una asignación de bit a bit de un puerto. La función es de tipo vacio.

void retardo(void)

Ejemplo:

```
retardo();
```

Función escribe_memoria( )

Esta función permite guardar un dato en la memoria de forma consecutiva direccionándola a través de un contador, el dato se envía a través del puerto B y el control de escritura se realiza por medio de el puerto C, primero se guardan los primeros 8 bit menos significativos y luego los 8 mas significativos. Sus parámetros son el dato a escribir y no retorna valores.

void escribe_memoria(unsigned dato)

Ejemplo:
Escribir el valor de una variable:

```
escribe_memoria(dato);
```

Escribir una constante:

```
escribe_memoria(10);
```

Función lee_memoria( )

Esta función permite leer un dato en la memoria de forma consecutiva direccionándola a través de un contador, el dato recuperado a través del puerto B; y el control de escritura se realiza por medio de el puerto C, primero se leen los primeros 8 bit menos significativos y luego los 8 mas significativos. No tiene parámetros y retorna en dato leído.

unsigned lee_memoria(void)

Ejemplo:

guardar = lee_memoria();

8.2.6.- Funciones Del Programa Principal

A continuación se presenta la descripción de las funciones utilizadas en el programa principal. La utilización de estas funciones y código del programa principal se muestra en el anexo B.

Función inicia_grafico( )

Esta función se encarga de inicializar el LCD grafico, para poder utilizar el área de texto y gráfico de este. La función es de tipo vacio.

void inicia_grafico(void)

Ejemplo:

inicia_grafico();

Función mensaje( )

Esta función permite mandar a imprimir en el LCD grafico una cadena de texto en mayúsculas o minúsculas con la opción de poder colocarlo a través de coordenadas (*x,y*). Sus parámetros son: coordenada *y*, coordenada *x* y la cadena de texto entre comillas dobles. Y no retorna ningún valor.

void mensaje(short y1, short x1, char *ptr)

Ejemplo:

mensaje(1,2,"Hola mundo");

Función pixel( )

La función pixel nos permite mandar a encender un pixel del LCD grafico a través coordenadas (x,y) y borra posiciones delante de donde se coloca el pixel a encender. Sus parámetros son la coordenada *y* y la coordenada *x*. La función no regresa ningún valor.

void pixel(unsigned piy, unsigned pixi)

Ejemplo:

pixel(30,45);
pixel(y,x);

Función pixel2( )

Al igual que la función pixel esta función manda a encender un pixel colocando la coordenada donde este se encenderá pero esta función no borra direcciones. Sus parámetros son coordenada *y*, coordenada *x*. Y no retorna valores.

void pixel2(unsigned piy, unsigned pixi)

Ejemplo:

pixel2(y,x);
pixel2(2,60);

Función borra_pixel( )

Al contrario de la función pixel2( ) esta borra un pixel colocando la coordenada (x,y). Sus parámetros son coordenada *y*, coordenada *x*. Y no retorna valores.

void borra_pixel(unsigned piy, unsigned pixi)

Ejemplo:

borra_pixel(y,x); borra_pixel(2,60);

Función cargar_imagen( )

Esta función nos permite mandar a visualizar una imagen de 240x128 pixeles al LCD grafico esta imagen debe estar guardada en la librería *pantallas.c*. Sus parámetros son el nombre de la imagen a visualizar. La función no regresa valores.

void cargar_imagen(unsigned char const panta[3840])

Ejemplo:

cargar_imagen(dibujo);

Función carga_corazon( )

Esta función manda a imprimir en la parte superior derecha del LCD un pequeño corazón con relleno o sin relleno las imágenes están guardadas en la librería *pantallas.c*. sus parámetros son el nombre de la imagen a mostrar *coraz* si es corazón con relleno y *cora* sin relleno. La función no retorna valores.

void carga_corazon(const unsigned char *ap)

Ejemplo:

Corazón con relleno

carga_corazon(coraz);

Corazón sin relleno

carga_corazon(cora);

Función borra_memoria( )

La función borrar_memoria( ) permite mandar a borrar la memoria grafica o de texto. Sus parámetros son 1 si se desea borrar la memoria de texto o cero para borrar la memoria grafica. No retorna valores.

void borra_memoria(unsigned short m1) // 0 = grafico; 1 = texto

Ejemplo:
Borrar memoria gráfica

```
borra_memoria(0);
```

Borrar memoria de texto

```
borra_memoria(1);
```

Función borra_linea( )

Esta función borra una línea de texto visualizada en el LCD grafico, sus parámetro son el numero de línea que hay que borrar. No regresa valores.

void borra_linea(unsigned short lin)

Ejemplo:

```
borra_linea(8);
```

Función aumenta_contador( )

Esta pequeña función nos permite aumentar el conteo de los contadores para direccionar la memoria. La función es de tipo vacio.

void aumenta_contador(void)

Ejemplo: aumenta_contador( );

Función reset_contadores(void)

Esta función manda a limpiar los contadores y es de tipo vacía.

void reset_contadores(void)

Ejemplo:

```
Reset_contadores( );
```

Función timer( )

En esta función es en donde configuramos una interrupción la cual se lleva a cabo cada 2.5 ms.

void timer(void)

Ejemplo:

```
timer( );
```

Función menu( )

Esta función manda a visualizar un menú con las opciones de capturar o leer una muestra. La función es de tipo vacía.

void menu(void)

Ejemplo:

menu( );

Función mover_cursor( )

Esta función manda a visualizar instrucciones para desplazar en la pantalla un cursor. La función es de tipo vacía.

void mover_cursor(void)

Ejemplo:

mover_cursor( );

Función mover_cursor( )

Esta función manda a visualizar instrucciones para realizar una segunda lectura de la señal o aplicarle un zoom. La función es de tipo vacía.

void mensajezoom(void)

Ejemplo:

mensajezoom( );

Función inicio( )

La función inicio inicializa el programa y nos dala bienvenida al programa. La función no retorna valores.

void inicio(void)

Ejemplo:

inicio( );

Función interrup( )

Esta función hace una interrupción para leer el puerto analógico cada 2.5 ms almacena el dato obtenido en la memoria RAM, haciéndole un offset es decir le suma un valor en este caso de 50. Esta función es propia del compilador.

Función capturar_muestra( )

Esta función manda activar la interrupción para capturar muestras por el puerto analógico y al mismo tiempo grafica las muestras obtenidas, además aquí es en donde se

controla el número de muestras a capturar. Una vez terminado esta acción es desactivada la interrupción

void capturar_muestra(void)

Ejemplo:

capturar_muestra( );

Función leer_muestra( )

La función leer_muestra( ) permite leer de la memoria las muestras capturadas y graficarlas, en este caso se grafican 1 de cada 4 muestras leídas pero es modificable. Al momento de visualizar 240 muestras se detiene la lectura y nos pregunta si deseamos seguir leyendo o si deseamos realizar una ampliación de la imagen. La función no retorna valores.

void leer_muestra(void)

Ejemplo:

leer_muestra( );

Función iniciozoom( )

Esta función nos ingresa a la opción de realizar una ampliación de la grafica si así se desea o también se puede salir de esta opción. Si se desea realizar el zoom se despliegan en pantalla unos cursores para seleccionar la parte a realizarle la ampliación y posteriormente se calculan los rangos a graficar con la llamada de la función *zoomx( ).* Sus parámetros son la variable *con* de la función leer muestra que permite saber en que dirección de memoria termino la lectura.

void iniciozoom(unsigned con)

Ejemplo:

iniciozoom(con);

Función zoomx( )

Esta función calcula el inicio de la posición de la memoria en donde hay que empezar a leer y aplicarle el zoom a la parte de la grafica seleccionada. Sus parámetros son la variable *con* de la función *iniciozoom( ),* la variable *xi* que es valor del inicio de la parte seleccionada y la variable *xf* que es valor del final de la parte seleccionada de la grafica. Posteriormente estos valores recalculados para valores de dirección de memoria que se mandan a la función *graficarzoom( ).* La función no retorna valores.

void zoomx(unsigned con, unsigned xi, unsigned xf)

Ejemplo: zoomx(con,xi,xf);

Función graficarzoom( )

Esta función grafica los datos leídos de la memoria con un escalamiento dando la sensación de un zoom.

void graficarzoom(unsigned xi, unsigned xf)

Ejemplo:

graficarzoom(xi,xf);

Función cursor( )

Despliega en pantalla un pequeño cursor que podemos moverlo en forma horizontal a la derecho o izquierda y ajustarlo en la posición que deseemos y nos retorna el valor de la posición en donde el cursor allá quedado. Esta este valor será la posición de inicio de la selección de la grafica.

unsigned cursor(void)

Ejemplo:

posición = curcor( );

Función cursor1( )

Despliega en pantalla un pequeño cursor que podemos moverlo en forma horizontal a la derecho o izquierda y ajustarlo en la posición que deseemos y no retorna el valor de la posición en donde el cursor allá quedado. Esta posición será el valor de la posición final de la parte seleccionada de la grafica.

unsigned cursor1(void)

Ejemplo:

posicion2 = curcor1( );

Función grafica( )

La función *grafica( )* grafica los valores que se están leyendo del puerto analógico o de la memoria. Sus parámetros son la posición en y en donde empezara a graficar un escalamiento para unir punto apunto graficado y un incremento.

void grafica(short a,short b, short c,short d)

Ejemplo:

grafica(36,17,29,10);

8.2.7.- Estructura Del Programa

La estructura del programa se detalla a continuación y se presenta como diagrama de flujo en la figura 22.

1.- Visualización de una imagen de bienvenida.
2.- Visualización de un mensaje de bienvenida.
3.- Presentación de un menú con las opciones de capturar o leer una señal.
4.- Si se selecciona la opción capturar. El programa leerá el canal analógico y tomará muestras de la señal, las cuales las guardará en la memoria RAM y se graficara una de cada cuatro muestras. Este proceso continuara hasta llegar a mil muestras capturadas.
5.- Si se selecciona la opción leer. Se procederá a leer la memoria y se graficaran 1 de cada 4 datos leídos. Al momento de tener una pantalla llena del LCD el programa nos preguntara si deseamos realizar un zoom a la señal, de ser a si se procederá seleccionar un rango de la señal y aplicarle el zoom. De lo contrario continuara el proceso de lectura hasta llegar a mil muestras.
6.- Al terminar los procesos de cualquiera de las opciones seleccionadas, el programa nos regresa al menú principal.

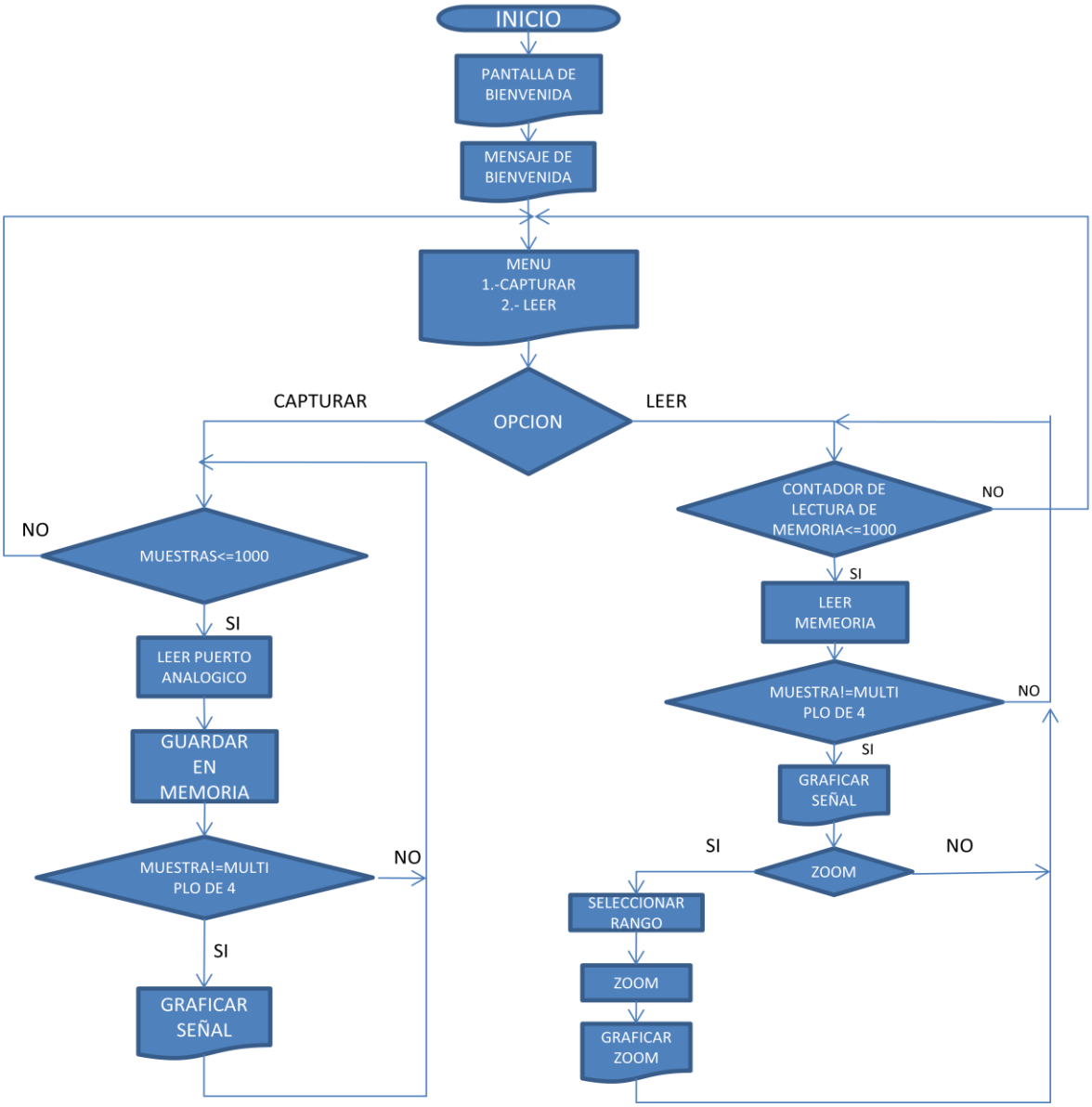


Figura 22.- Diagrama de flujo.

9.- SIMULACIÓN DEL PROGRAMA EN ISIS PROFESIONAL (SOFTWARE DE SIMULACIÓN).

Pantalla de bienvenida (fig.23). En esta figura se aprecia una imagen en el LCD gráfico la cual se visualiza en cuanto comienza a funcionar el circuito.

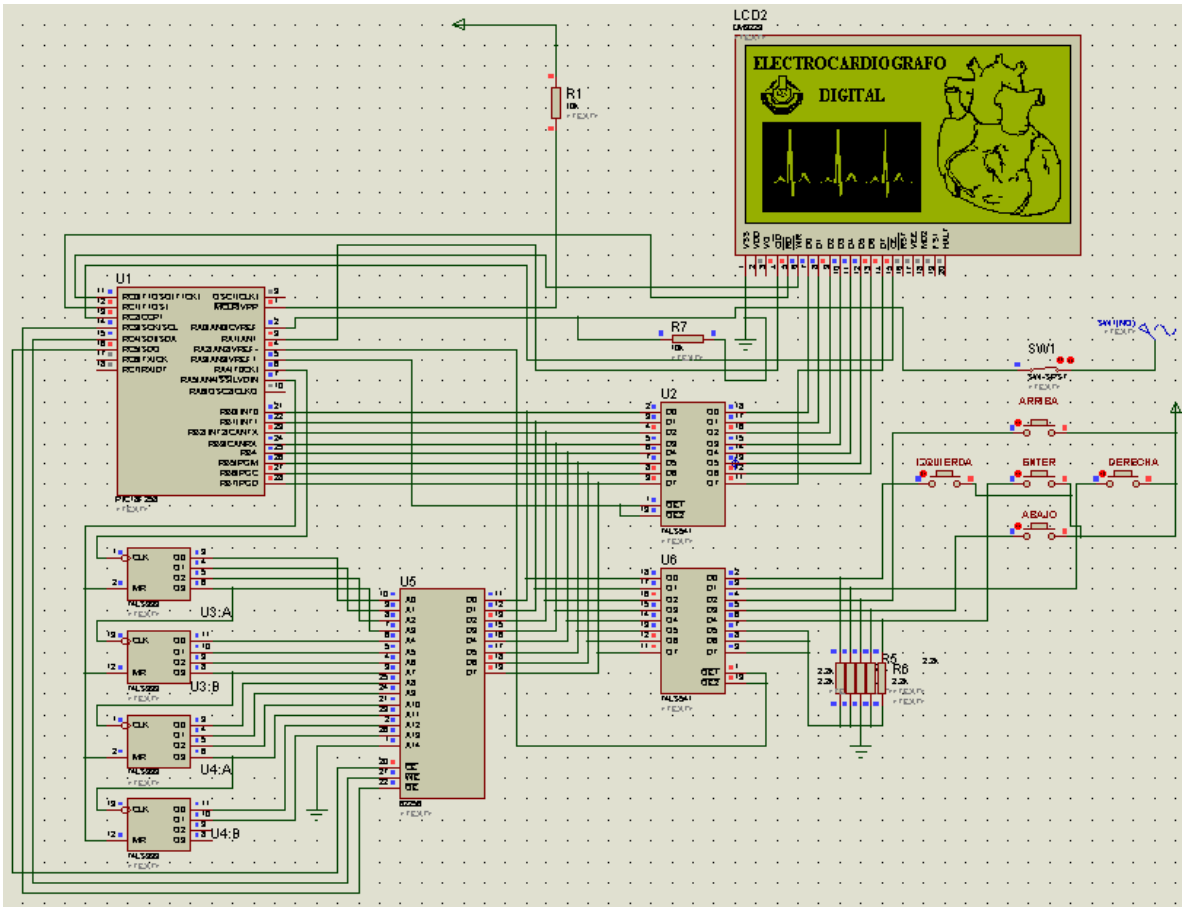


Figura 23.- Pantalla de bienvenida

Mensaje de bienvenida (fig. 24) . Este mensaje es visualizado posteriormente a la pantalla de bienvenida.

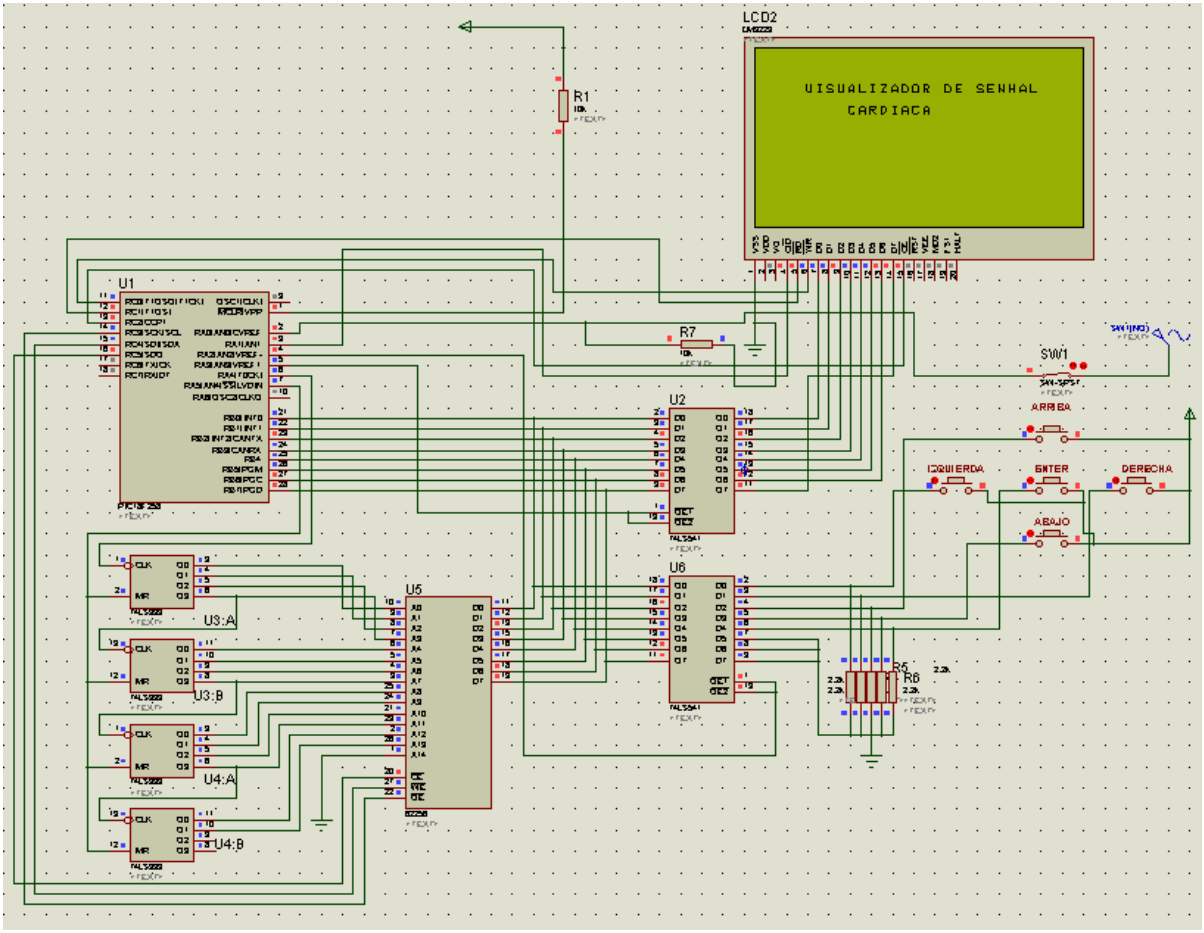


Figura 24.- Mensaje de bienvenida

Visualización del menú (fig. 25). Esta visualización nos presenta un menú con las opciones de capturar una señal o leer una señal guardada en la memoria RAM, para elegir dichas opciones se realiza a través del teclado.

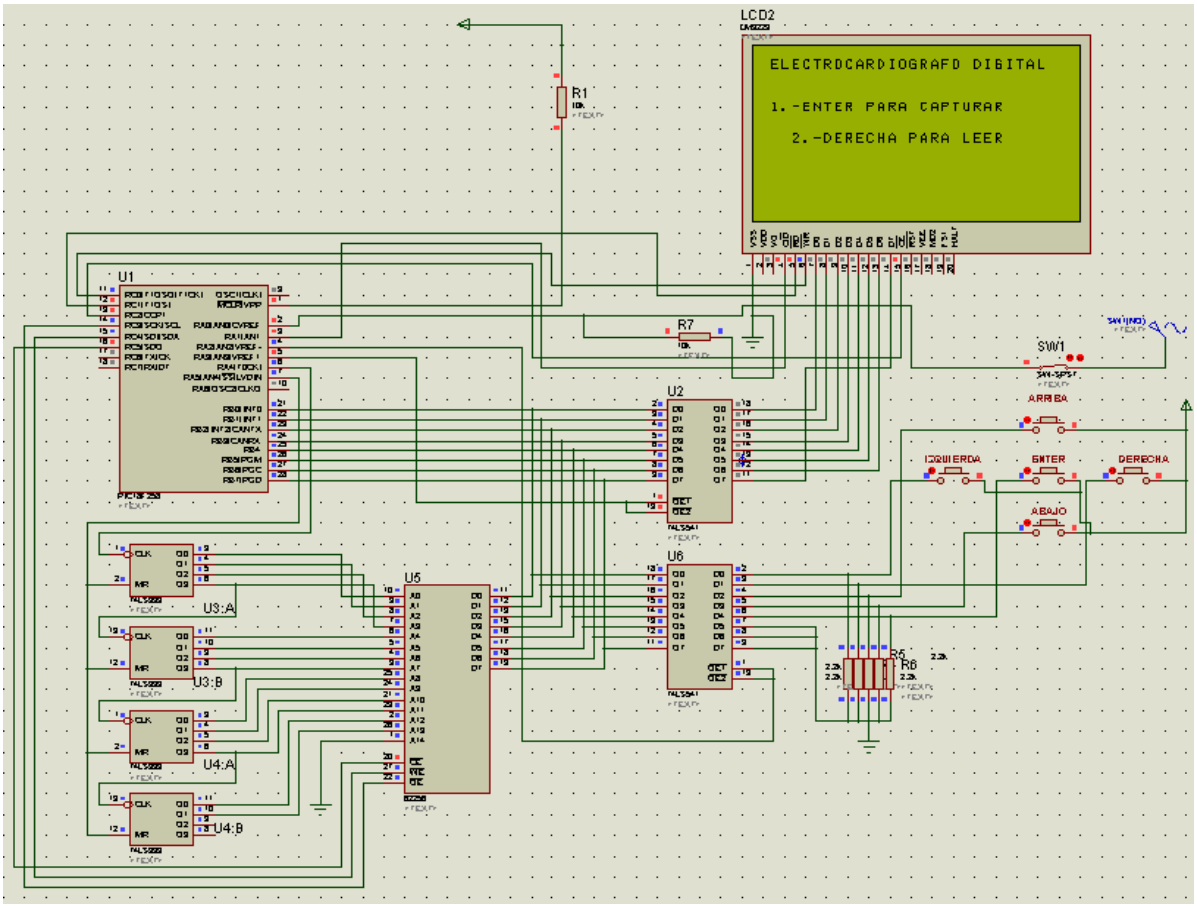


Figura 25.- Visualización del menú

Visualización de una señal simulada (fig. 26). A continuación se visualiza una señal senoidal la cual comienza a visualizarse y a guardarse cuando elegimos del menú la opción de capturar señal.

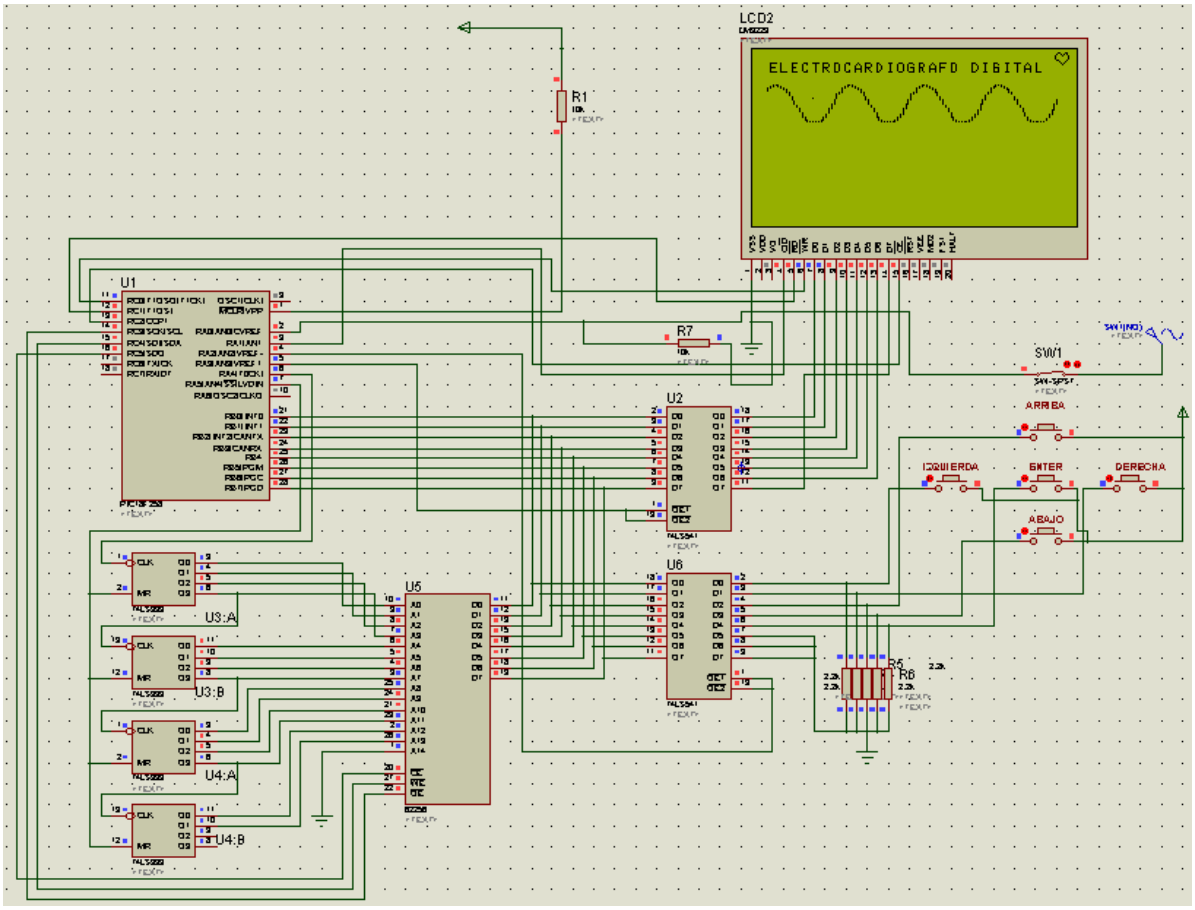


Figura 26.- Visualización de la captura de una señal

Simulación de la lectura de una señal guardada en la memoria RAM (fig. 27). Cuando elegimos la opción del menú, leer señal, se comienza la lectura de la memoria RAM y se presenta la grafica de la señal guardada.

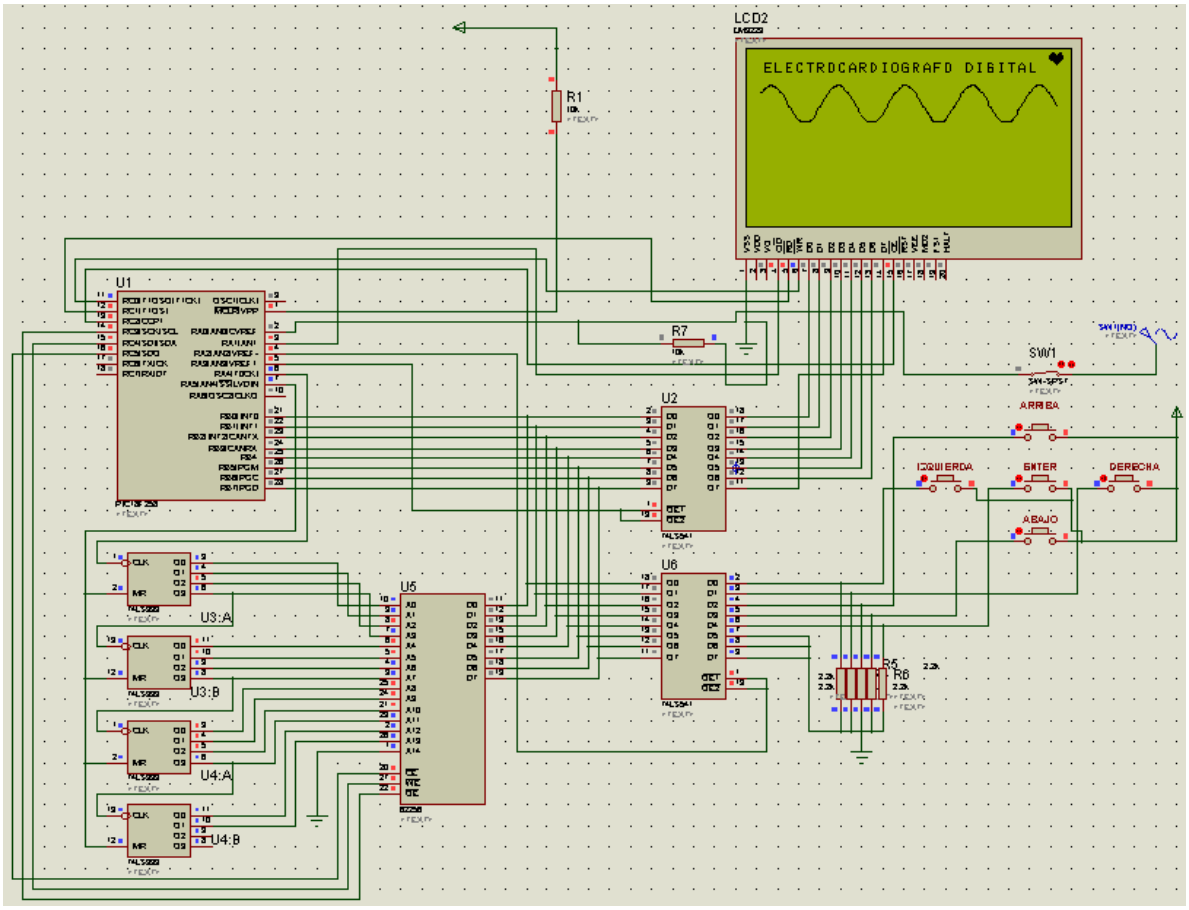


Figura 27.- Señal leída de la memoria RAM

Simulación de un zoom. Para realizar un zoom a la grafica primeramente debemos de seleccionar un rango de esta por medio de unos cursores como se muestra en la figura 25 posteriormente elegimos la opción zoom que nos presenta la pantalla y obtenemos el zoom del rango seleccionado (fig. 28).

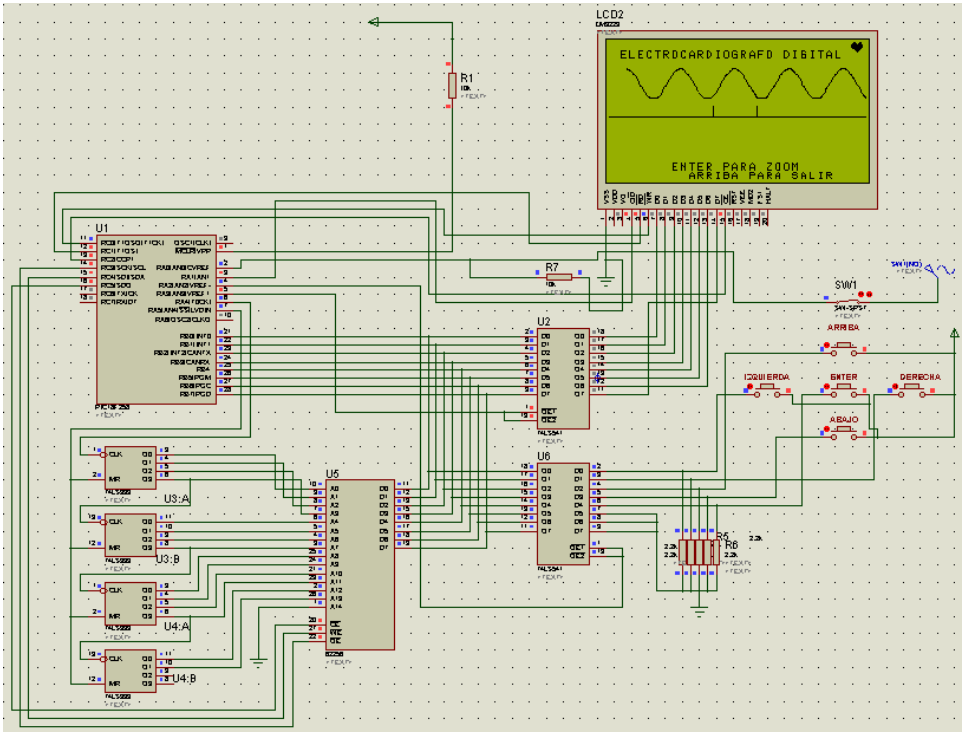


Figura 28.- Selección de un rango de la señal

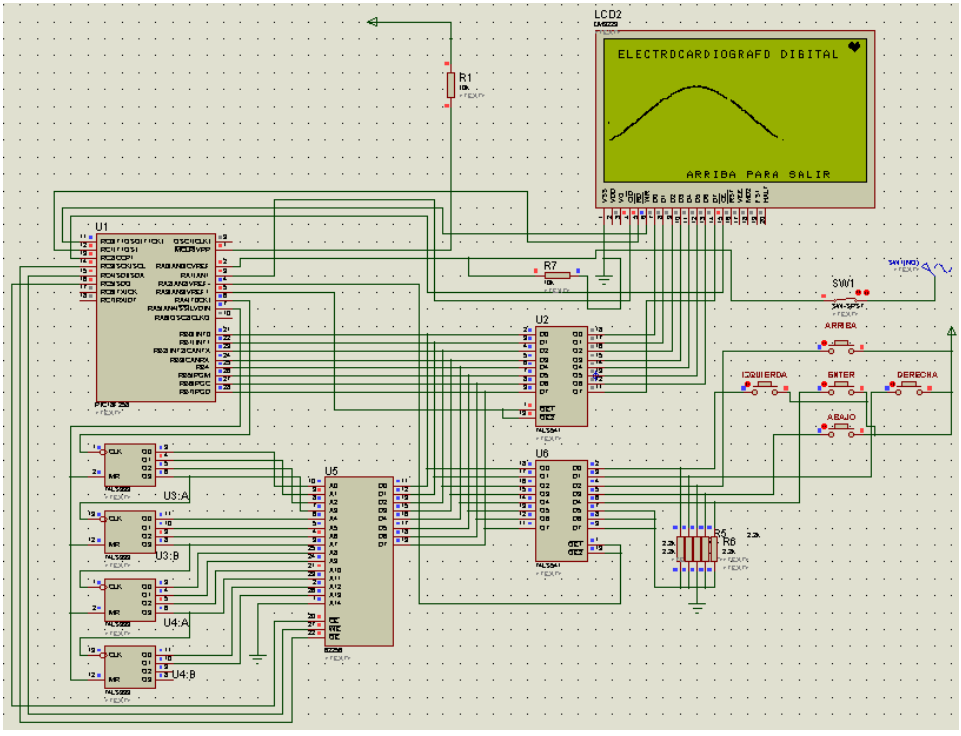


Figura 29.- Zoom del rango seleccionado de la gráfica

10.- RESULTADOS OBTENIDOS

En la figura 30 se aprecia la imagen de bienvenida al iniciar el funcionamiento del sistema.

Figura 30.- Imagen de bienvenida.

A continuación presentamos en la figura 31 la visualización del menú, en donde podemos elegir la opción de capturar o leer una señal.

Figura 31.- Visualización del menú

A continuación presentamos en la figura 32 la captura de una señal que corresponde a la carga y descarga de un capacitor. Este proceso comienza cuando elegimos del menú la opción de capturar señal.

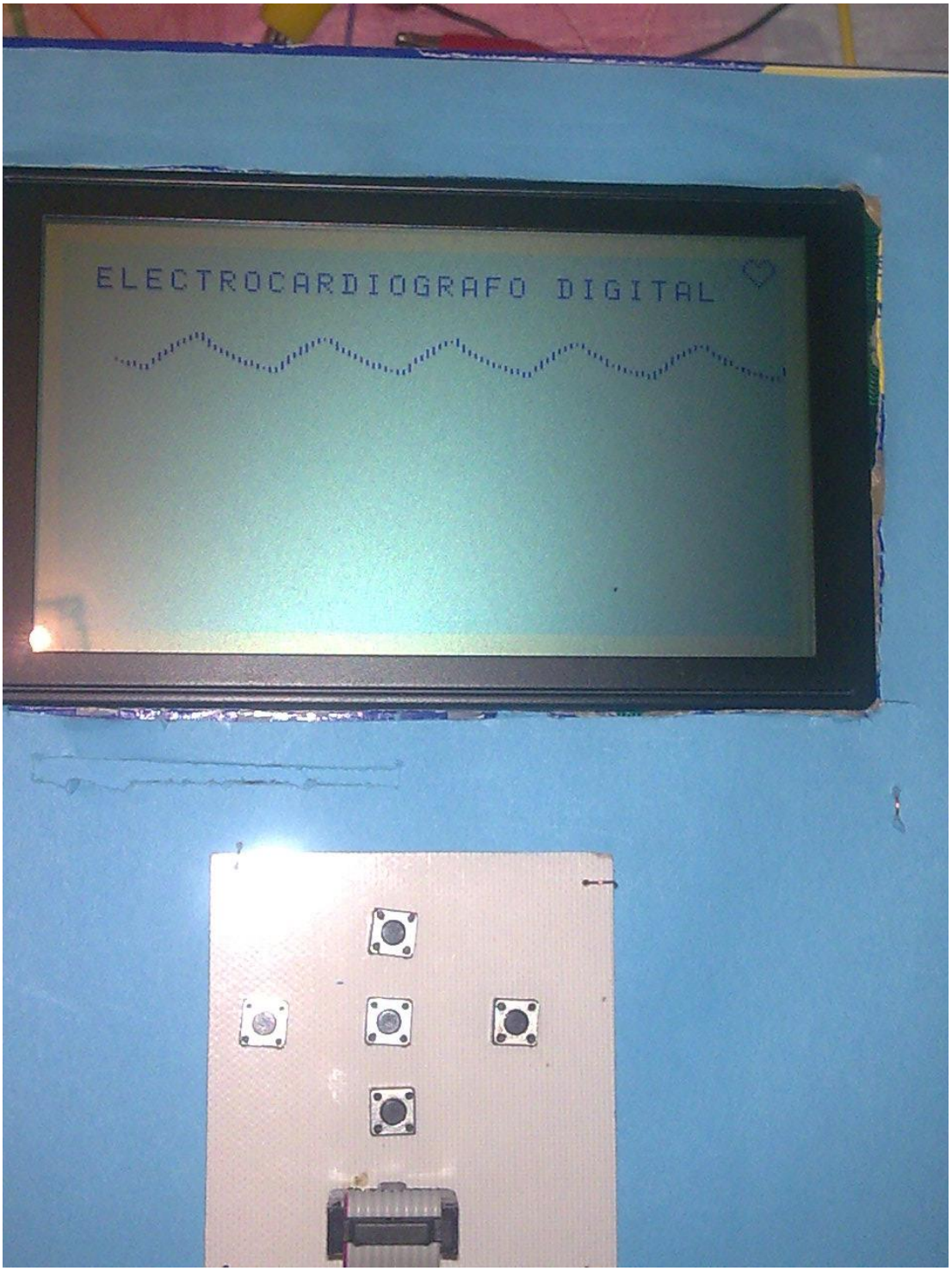


Figura 32.- Señal capturada

En la figura 33 presentamos la visualización de la lectura de una señal guardada en memoria. Este proceso se realiza cuando de la elegimos la opción leer memoria del menú principal.

Figura 33.- Lectura de una señal.

En la figura 34 se muestra la selección de una parte de la grafica visualizada para aplicarle un zoom. Esta selección se realiza por medio de dos cursores como podemos apreciar en la imagen. Los cursores los manejamos por medio del teclado.

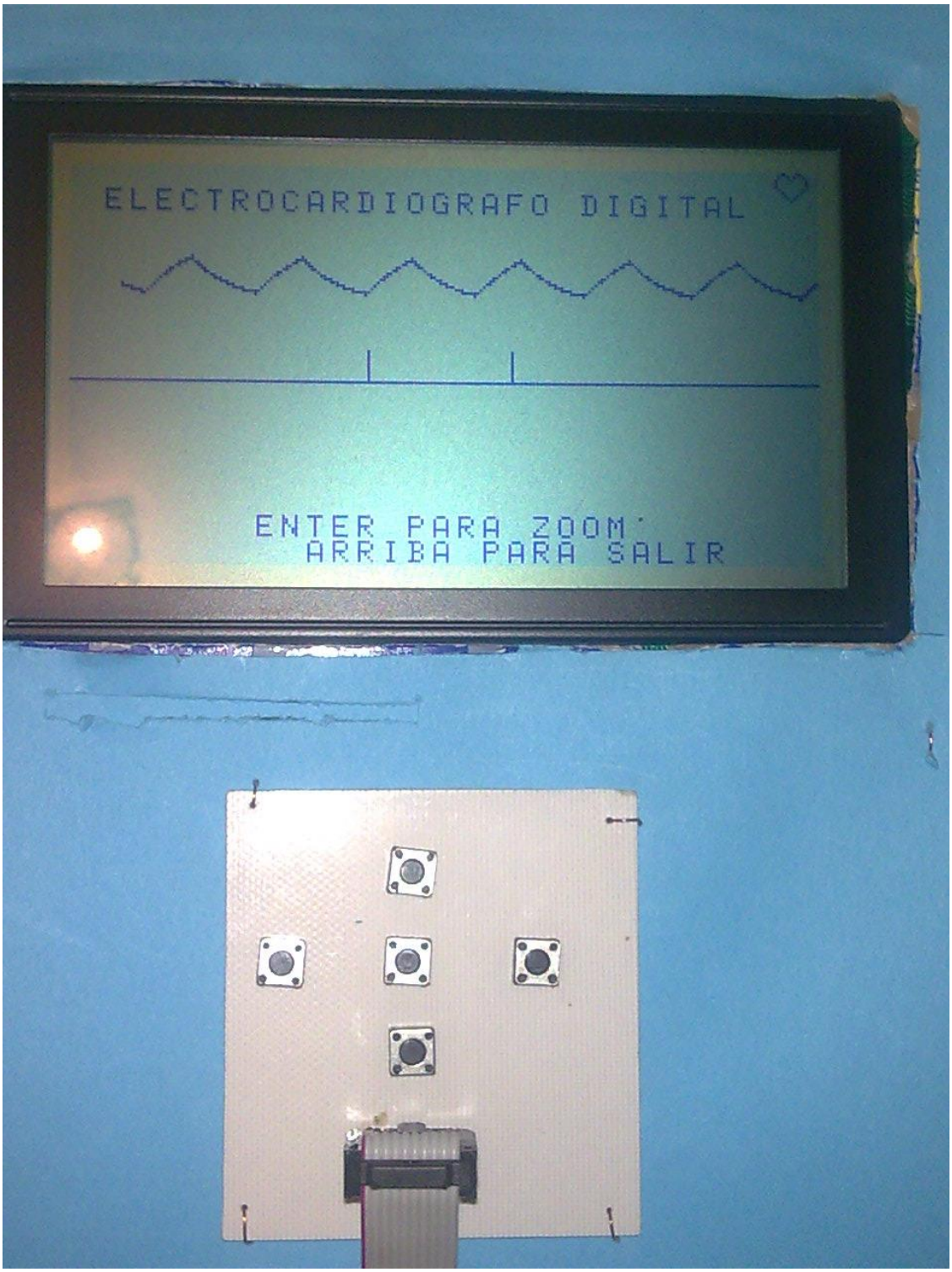


Figura 34.- Selección de un rango de la grafica

Como se aprecia en la figura anterior se selecciono un rango de la gráfica y a continuación en la figura 35 se muestra el zoom del rango seleccionado.

Figura 37.- Zoom de un rango seleccionado de la grafica

11.- CONCLUSIÓN

Los resultados que se obtuvieron físicamente fueron satisfactorios cumpliendo las expectativas buscadas. La fuente de voltaje desarrollada cumplió con las características eléctricas para la alimentación del circuito. La prueba del sistema nos arrojo resultados satisfactorios ya que se logro obtener resultados similares a los de la simulación por software, aun que se tuvieron algunos problemas con respecto a la reacción del circuito, causados por la frecuencia con la que trabajan los circuitos digitales. Estos problemas se corrigieron de tal forma que el circuito funciona de forma correcta. Cabe mencionar que el programa desarrollado fue para visualización de una sola señal esto debido al inconveniente que nuestra pantalla LCD gráfica es de poca resolución.

Con estos resultados obtenidos concluyó que se realizo lo que se buscaba cumpliendo así los objetivos planteados.

12.- RECOMENDACIONES

Las recomendaciones de uso del sistema son las siguientes:

1.- Es importante verificar las la los voltajes de alimentación del sistema. Los cuales son +5 Volts, -15 Volts y tierra.

2.- Verificar las conexiones del LCD gráfico y el teclado a la tarjeta principal.

3.- Si el sistema no inicializa correctamente, hay que proceder a reinicializarlo.

4.- Si se hace una nueva versión del programa, verificar la posición del microcontrolador.

13.- TRABAJOS FUTUROS

La fuente de voltaje la podemos mejorar, construyendo otra con un sistema de regulación más complejo o podemos adquirir una del mercado.

En cuanto a al circuito principal, este se puede diseñar como circuito de doble cara o también, se podría a mandar a construir la tarjeta de forma industrial.

Las pruebas de visualización nos arrojaron resultados buenos, aun que un poco limitado por la resolución de la pantalla LCD gráfica. A si que se podría adquirir otra de mayor tamaño y a si poder visualizar mas de una señal.

Con respecto a la programación se pueden mejorar las funciones de Zoom para una mejor presentación de visualización.

14.- BIBLIOGRAFÍA

- Sistemas digitales
  Autor. Morris Mano
  Ed. Prentice Hall.

- Sistemas Digitales
  Autor. Enrique Mandado
  Ed. Alfa-Omega

- Señales y sistemas continuos discretos
  Autor: Soliman, Samir S.
  Ed. Prentice-Hall International

- Microcontroladores PIC. Diseño práctico de aplicaciones
  Autor: Angulo Usategui, José María y Angulo Martínez, Ignacio
  Ed. McGraw-Hill

- Apuntes de Microcontroladores
  6º semestre.
  Ing. Electrónica.

Citas de internet

- http://www.alldatasheet.com/datasheet-pdf/pdf/93916/MICROCHIP/PIC18F258.html
- http://fisiopuj.tripod.com/Guias/1_Electrocardiograma.pdf
- http://senalesysistemas.googlepages.com/
- http://www.mikroe.com/en/download/
- http://www.ferato.com/wiki/index.php/Electrocardiograma

15.- ANEXO A

Librerías de programación:

Librería funciones_graficas_basicas.c

```
#define WR PORTC.F0
#define RD PORTC.F1
#define CE PORTC.F2
#define CD PORTA.F1


void escribe_dato(short d)
{
 RD = 1;
 WR = 0;
 CD = 0;
 PORTB = d;
 CE = 0;
 CE = 1;
}

void escribe_comando(short d)
{
 RD = 1;
 WR = 0;

 CD = 1;
 PORTB = d;
 CE = 0;
 CE = 1;
}


void envia_comando(int dat, short comand, short num)
{
 Delay_us(50);
 if(num != 0)
   { escribe_dato(dat);
     Delay_us(50);
     if(num == 2)
       { escribe_dato(dat>>8);
         Delay_us(50);
       }
   }
 escribe_comando(comand);
}

void envia_cadena(int direc,char das)
{
  Delay_us(50);
```

```
  envia_comando(direc,0x24,2);
  Delay_us(50);
  escribe_dato(das-32);
  Delay_us(50);
  escribe_comando(0xC4);
}

void envia_dato(int dir, char da)
{
  Delay_us(50);
  envia_comando(dir,0x24,2);
  Delay_us(50);
  escribe_dato(da);
  Delay_us(50);
  escribe_comando(0xC4);

}

unsigned short lee_dato(int dir1)
{ unsigned short r;
  Delay_us(50);
  envia_comando(dir1,0x24,2);
  Delay_us(50);
  escribe_comando(0xC5);
  Delay_us(50);
  TRISB = 255;
  RD = 0;
  WR = 1;
  CD = 0;
  CE = 0;
  r = PORTC;
  CE = 1;
  TRISB = 0;
  PORTA = r;
  Delay_us(5000);
  return r;

}
```

Librería memoria.c

```
void retardo(void)
{
}

void escribe_memoria(unsigned dato)
{
   retardo();
  PORTA.F4=1;   //Pulso de incremento de direcion en memoria
  retardo();
  PORTA.F4=0;

  PORTB = dato;
  retardo();
  PORTC.F5 = 0; // DESHABILITAR
  retardo();
  PORTC.F4 = 1; //
  retardo();
  PORTC.F3 = 1; //
  retardo();
  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 0; // ESCRIBIR
  retardo();
  PORTC.F3 = 1; //
  retardo();
  Delay_us(1);
  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 1; // DESHABILITAR
  retardo();
  PORTC.F3 = 1; //
  retardo();

  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 1; // DESHABILITAR
  retardo();
  PORTC.F3 = 1; //
  retardo();

  PORTA.F4=1;   //Pulso de incremento de direcion en memoria
  retardo();
  delay_us(20);
  PORTA.F4=0;

  retardo();
  PORTB = dato>>8;
  retardo();
  PORTC.F5 = 0; // DESHABILITAR
```

```
  retardo();
  PORTC.F4 = 1; //
  retardo();
  PORTC.F3 = 1; //
  retardo();
  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 0; // ESCRIBIR
  retardo();
  PORTC.F3 = 1; //
  Delay_us(1);
  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 1; // DESHABILITAR
  retardo();
  PORTC.F3 = 1; //
  retardo();

  PORTC.F5 = 1; //
  retardo();
  PORTC.F4 = 1; // DESHABILITAR
  retardo();
  PORTC.F3 = 1; //
  retardo();
  //retardo();
  //retardo();
}

unsigned lee_memoria(void)
{  unsigned dato, dato1;

  PORTA.F4=1;   //Pulso de incremento de direcion en memoria
  retardo();
  delay_us(20);
  PORTA.F4=0;
  retardo();

  PORTC.F5 = 0; // DESHABILITAR
  retardo();
  PORTC.F4 = 1; //
  retardo();
  PORTC.F3 = 1; //
  retardo();

  PORTC.F5 = 0; //
  retardo();
  PORTC.F4 = 1; // LEER
  retardo();
  PORTC.F3 = 0; //
  Delay_ms(5);
```

```
dato = PORTB;
retardo();
PORTC.F5 = 0; //
retardo();
PORTC.F4 = 1; // DESHABILITAR
retardo();
PORTC.F3 = 1; //
retardo();

PORTC.F5 = 1; //
retardo();
PORTC.F4 = 1; // DESHABILITAR
retardo();
PORTC.F3 = 1; //
retardo();

PORTA.F4=1;   //Pulso de incremento de direcion en memoria
retardo();
delay_us(20);
PORTA.F4=0;
retardo();
PORTC.F5 = 0; // DESHABILITAR
retardo();
PORTC.F4 = 1; //
retardo();
PORTC.F3 = 1; //
retardo();

PORTC.F5 = 0; //
retardo();
PORTC.F4 = 1; // LEER
retardo();
PORTC.F3 = 0; //
retardo();
Delay_ms(5);
dato1 = PORTB;
retardo();
PORTC.F5 = 0; //
retardo();
PORTC.F4 = 1; // DESHABILITAR
retardo();
PORTC.F3 = 1; //
retardo();

PORTC.F5 = 1; //
retardo();
PORTC.F4 = 1; // DESHABILITAR
retardo();
PORTC.F3 = 1; //
retardo();
```

```
  dato1 = dato1<<8;
  retardo();
  dato = dato1 | dato;
  PORTC.F5 = 1; //
  retardo();

  PORTC.F4 = 1; // DESHABILITAR
  retardo();
  PORTC.F3 = 1; //
  return dato;
}
```

Librería teclado.c

```
void teclado(void)
{

 trisb=0b11111111;
 PORTA.F3=1;//deshabilitar buffer de glcd
 portc.f5=1;//
 portc.f6=1;//deshabilitar memoria
 portc.f7=1;//
 porta.f2=0; //habilitar bufer teclado

}



void restaurar(void)
{
trisb=0b00000000;
PORTA.F3=0;//habilitar buffer de glcd
porta.f2=1;//habilitar bufer teclado
}
```

16.- ANEXO B

Código fuente del programa principal.

```
#include "funciones_graficas_basicas.c"
#include "pantallas.c"
#include "teclado.c"
#include "memoria.c"
unsigned short   x2;
int x1;
unsigned x, xp,x11,xp0,xp00,xp1,xp2;
unsigned adress, direccion,cont,multiplo=1,comp=1;
unsigned short t1,t;
short salir=0,zoom=0;

void inicia_grafico(void)
{
 delay_ms(5000);
 envia_comando(0x0000,0x40,2); // coloca direccion inicial de texto
 envia_comando(0x0200,0x42,2); // coloca direccion inicial de grafico
 envia_comando(0x001e,0x41,2); // coloca numero de direcciones de texto por linea
 envia_comando(0x001E,0x43,2); // coloca direccion de datos graficos
 envia_comando(0,0x80,0);      // Coloca modo encendido
 envia_comando(0x0002,0x22,2); // Coloca el registro de inicio (offset)
 envia_comando(0,0xa1,0);
 envia_comando(0,0x9c,0);      // Coloca modo del display (text on, grafic off, Cursor off)
 envia_comando(0x0000,0x24,2); // coloca direccion de texto
}

void mensaje(short y1, short x1, char *ptr)
{ short lon;
  adress = 0x0000 + (30*(y1-1)) + x1-1;
  lon = strlen(ptr);

  for(x=0; x<lon; x++)
     { envia_cadena(adress,*ptr);
       ptr++;
       adress++;
     }

}

void pixel(unsigned piy, unsigned pixi)
{  unsigned short xb, pixeles[]={0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8};
  unsigned adelante, ad, xc;
  t = pixi;
  t1 = t>>3;
  piy--;
  adress = 0x0200 + t1 + piy*30;
  t = pixi - (t1*8);
```

```
  // colocamos la direccion y enviamos la posicion del pixel a encender
  Delay_ms(1);
  envia_comando(adress,0x24,2);
  Delay_ms(1);
  escribe_comando(pixeles[t]);

   // borramos 1 direcciones adelante
  adelante = ( (adress-0x0200)%30)+2;
  if(piy<54)
     xc=1052;
  else
     if(piy<91)
        xc=1620;
     else
        xc=2760;
  adress = xc+adelante;
  if(adress != ad)
    {  ad = adress;
       for(xb=0;xb<36; xb++)
          { adress = adress + 30;
            envia_dato(adress,0);
          }
    }
}
void pixel2(unsigned piy, unsigned pixi)
{  unsigned short xb, pixeles[]={0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8};
  unsigned adelante, ad, xc;
  t = pixi;
  t1 = t>>3;
  piy--;
  adress = 0x0200 + t1 + piy*30;
    t = pixi - (t1*8);
 // colocamos la direccion y enviamos la posicion del pixel a encender
  Delay_ms(1);
  envia_comando(adress,0x24,2);
  Delay_ms(1);
  escribe_comando(pixeles[t]);
}
void borra_pixel(unsigned piy, unsigned pixi)
{  unsigned short xb, pixeles[]={0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8};
  unsigned adelante, ad, xc;
  t = pixi;
  t1 = t>>3;
  piy--;
  adress = 0x0200 + t1 + piy*30;
  // colocamos  la direccion y enviamos la posicion del pixel a encender
  Delay_ms(1);
  envia_dato(adress,0);
}
```

```
void cargar_imagen(unsigned char const panta[3840])
{ unsigned char dato;
 adress = 0x0200;
  for(xp=0; xp<128; xp++)
    { for(x=0;x<30;x++)
       { //if(panta==1)
           dato =  panta[x+xp*30];
         //else
          // dato = presenta[x+xp*30];
         envia_dato(adress,dato);
         adress++;
       }
//       adress = adress + 14;
    }
}

void carga_corazon(const unsigned char *ap)
{ unsigned char dato;
  unsigned short xz;
  adress = 0x021c;
  for(xp=0; xp<10; xp++)
    { for(xz=0;xz<2;xz++)
       { dato = *ap++;
         envia_dato(adress,dato);
         adress++;
       }
      adress = adress + 28;
    }
}


void borra_memoria(unsigned short m1)   // 0 = grafico;  1 = texto
{ int fin;
  if(m1==0)
    { adress = 0x0200;
      fin = 0x3fff;
    }
  else
    { adress = 0x0000;
      fin = 0x0200;
    }
  for(xp=0; xp<fin; xp++)
    {
         envia_dato(adress,0);
         adress++;
    }
}
```

```
void borra_linea(unsigned short lin)
{
  adress = 0x0000;
  adress = adress + (lin-1) * 30;
  for(xp=0; xp<30; xp++)
     {
          envia_dato(adress,0);
          adress++;
     }
}


void aumenta_contador(void)
{
   PORTA.F4=1;   //Pulso de incremento de direcion en memoria
   asm nop;
   PORTA.F4=0;
}

void reset_contadores(void)
{
 PORTA.F5 = 1;
 Delay_us(50);
 PORTA.F5 = 0;
}
void timer(void)
 {
            T0CON=0b11000110;
            INTCON=0b10100100;
            TMR0L =63;
 }

void menu(void)
  {
  mensaje(2,2,"ELECTROCARDIOGRAFO DIGITAL");
  mensaje(6,2,"1.-ENTER PARA CAPTURAR");
  mensaje(9,4,"2.-DERECHA PARA LEER");
  }



void mover_cursor(void)
  {
  mensaje(13,1,"*DERECHA O IZQUIERDA");
  mensaje(14,1," PARA MOVER CURSOR");
  mensaje(16,8,"**ARRIBA PARA AJUSTAR");
  }
void mensajezoom(void)
{
     mensaje(13,2,"DERECHA PARA CONTINUAR");
```

```
     mensaje(15,2,"ENTER PARA SELECCION");
}

  void inicio(void);
  void leer_muestra(void);
  void capturar_muestra(void);
  void grafica(short a,short b, short c,short d);
  void iniciozoom(unsigned con);
  void zoomx(unsigned con,unsigned xi,unsigned xf);
  void graficarzoom(unsigned xi, unsigned xf);
  unsigned cursor(void);
  unsigned cursor1(void);
unsigned short corazon;
/*********************************************************//////////////////////
void main(void)
{

 unsigned i=0;
 char cad[10];
 TRISC=0B11000000;
 PORTC=0b00100000;
 TRISB = 0;
 PORTB = 0;
 TRISA= 0b0000001;
 ADCON1 = 14;
 porta=0b00000100;
 porta.f6=1;
 corazon= 0;

 inicio();
  while(1)
{
  menu();
  do{teclado();}while(!(portb==16||portb==2));
  switch(portb)
    {
      case 16:    reset_contadores();
                  capturar_muestra();
                  break;

       case 2:    leer_muestra();
                  break;
      default:    break;

    }

    TRISB=0B00000000;
    PORTA.F3=0;
    borra_memoria(0);
    borra_memoria(1);
```

```
  }// fin del while
} // fin del main

void inicio(void)
{
reset_contadores();
delay_ms(300);
inicia_grafico();
borra_memoria(0);
cargar_imagen(corazon3);
delay_ms(10000);
for(x1=0; x1<5; x1++)
{ carga_corazon(coraz);
  Delay_ms(850);
  carga_corazon(cora);
  Delay_ms(850);
}
borra_memoria(0);
borra_memoria(1);

  { mensaje(4,5,"VISUALIZADOR DE SENHAL");
    mensaje(6,9,"CARDIACA");
    Delay_ms(2000);
  }


borra_memoria(0);
borra_memoria(0);
borra_memoria(1);


}

//funcion para leer del puerto analogico cada 10 ms
void interrupt ( void )
{
TMR0L=63;
INTCON.F2 = 0;
cont++;
     xp0 = ADC_Read(0);
     if(xp0>1023)
        xp0=1023;
     escribe_memoria(xp0);

}

void capturar_muestra(void)
{
               multiplo=1;
               timer();
```

```
            x = 1;
            xp0 = 512;
            restaurar();
            borra_linea(6);
            borra_linea(9);
            reset_contadores();
            for(x1=0;x1<320;x1++)
              {
                 if(x1==multiplo)
                {
                 delay_ms(50);
            grafica(36,17,29,10);


             if(multiplo==1) multiplo=0;
             multiplo=multiplo+2;   //linea para cambiar nº de muestras a graficar
             if(multiplo>=999) multiplo=0;

                }
            xp00 = xp0;
            x++;
            if(x==240) x = 1;


            }
            T0CON=0b00000000;
            INTCON=0b00000000;
            reset_contadores();
            x=1;
            xp0=512;

              borra_memoria(0);
              borra_memoria(1);

}

void leer_muestra(void)
{
unsigned leer;
unsigned i=0;
unsigned con=0,num;
 char text[8] ;
   multiplo=1;
   x = 1;
   xp0 = 512;
   restaurar();
   borra_linea(5);
   borra_linea(6);
   borra_linea(9);
   reset_contadores();
```

```
   TRISB = 0x0ff;       //modo entrada
   PORTA.F3=1;          //desactivar buffer
   for(x1=0;x1<960;x1++)
    {
      con++;
      con++;
      xp0=lee_memoria();
     if(x1==multiplo)
       {
         grafica(36,17,29,10);

         if(multiplo==1) multiplo=0;
         multiplo=multiplo+4;
         if(multiplo>=959) multiplo=0;

         x++;
        }
     xp00 = xp0;
      delay_ms(10);
        if(x==240)
           {  x = 1;
              iniciozoom(con);
              delay_ms(10);
           }
     delay_ms(10);
     TRISB = 0x0ff;    //modo entrada
     PORTA.F3=1;       //deshabilito buffer
   }
         asm nop;
         portc.f5=1;////
         portc.f6=1;//deshabilitar memoria
         portc.f7=1;////
         delay_ms(5000);
         reset_contadores();
         delay_ms(10);
}

void grafica(short a,short b, short c,short d)
{
     PORTA.F3=0;       //activar buffer
     asm{nop;nop;nop;nop}
     TRISB = 0x000;    //modo salida

      if(xp0<xp00)
      {
       for(x11=xp0;x11<xp00; x11+=d)
          pixel(a+(b-x11/c),x);
      }
     else
      { for(x11=xp00;x11<xp0; x11+=d)
```

```
            pixel(a+(b-x11/c),x);
        }
       pixel(a+(b-x11/c),x);

       if(xp00<xp0 && corazon == 1)
        {
         corazon = 0;
         carga_corazon(coraz);
        }
       else
        if(xp00>xp0 && corazon == 0)
         { corazon = 1;
           carga_corazon(cora);
         }

}
void iniciozoom(unsigned con)
{
unsigned xi=0,xf=0;

                mensajezoom();
                while(salir==0)
                {
                 do{
                 teclado();
                 if(portb.f4==1)   //ZOOM
                 {
                  zoom=1;
                  salir=1;
                  }
                 if(portb.f1==1)  //continuar lecctura
                 salir=1;
                 }while(portb==0);
                 restaurar();
                }

                if(zoom==1)
                {
                borra_linea(13);
                borra_linea(15);
                xi=cursor(); ////////////////////////////////////////////////////
                asm nop;
                xf=cursor1();
                reset_contadores();
                asm nop;
                zoomx(con,xi,xf);
                asm nop;
                borra_memoria(0);
                asm nop;
                mensaje(2,2,"ELECTROCARDIOGRAFO DIGITAL");
```

```
            asm nop;
            }
            salir=0;
            zoom=0;


}



void zoomx(unsigned con, unsigned xi, unsigned xf)
 {  short sal=1, a=0;
    unsigned start=0, stop=0,i=0,j=0,contador=0;
    //start=con-480+xi*2;
    start=con-1920+(xi)*8;
    xi=xi*4;
    xf=xf*4;
    //stop=con-960-xf*2;

   while(sal==1)
 {

    if(a==1)
    {
      borra_memoria(0);
      mensaje(2,2,"ELECTROCARDIOGRAFO DIGITAL");
      reset_contadores();
      for(i=0;i<start;i++)
      {asm nop;
         aumenta_contador();
         }
      ////////////////////
       delay_ms(1000);
       graficarzoom(xi,xf);/////////////////

       ////////////////////
      a=0;

    }
    restaurar();
    mensaje(16,10,"ARRIBA PARA SALIR");


   do
   {

    teclado();
    if(portb.f2==1)  //SALIR
    {sal=0;a=0;}
```

```
   if(portb.f4==1) ///ZOOM
    {a=1;}
    delay_ms(1);
  }while(portb==0);
  restaurar();

  //borra_linea(16);
 }
 reset_contadores();
// contador=(240-xf)*2;
      for(i=0;i<con;i++)
        { asm nop;
        aumenta_contador();}


 }

 void graficarzoom(unsigned xi, unsigned xf)
 {
  unsigned j=0;
  long graf=0;

            x = 1;
            xp0 = 512;
           TRISB = 0x0ff;       //modo entrada
           asm {nop nop nop}
           PORTA.F3=1;          //desactivar buffer glcd
           asm {nop nop nop}
           porta.f2=1;
           asm {nop nop nop}
            portc.f5=1;//
            asm {nop nop nop}
            portc.f6=1;//deshabilitar memoria
            asm {nop nop nop}
            portc.f7=1;//
            asm {nop nop nop}
           for(j=xi;j<xf;j++)
           {
           graf=lee_memoria();
           xp0=graf;
           if(xp0>=1023)
           xp0=1023;
           if(xp0>=0&&xp0<=1023)
           grafica(80,17,15,10);
           xp00 = xp0;
           x++;
           if(x==240)
           {
             x = 1;
           }
```

```
        TRISB = 0x0ff;    //modo entrada
        asm {nop nop nop}
        PORTA.F3==1;      //deshabilito buffer
        }


 }



unsigned cursor(void)
{
unsigned y=0,xx=2,salir=1,z=0,cont=2,cont1=2,j=0;
  for(j=0;j<240;j++)
       pixel2(70,j);
 mover_cursor();
 while(salir==1)
 {

 if(cont1<cont)
 {
    z=cont1+1;
 }
 if(xx==cont)
    z=cont-1;
 for(y=60;y<=69;y++)
  borra_pixel(y,z);

 for(y=60;y<=69;y++)
  pixel2(y,xx);
  do
  {
   delay_ms(10);
   teclado();
   if(portb.f1==1)  //derecha
      {xx++;
      cont=xx;}
   if(portb.f0==1)  //izquierda
      {
      xx--;
      cont1=xx;
      }
  if(portb.f2==1)
    salir=0;
  }
  while(portb==0);
  restaurar();
```

```
 }
  borra_linea(16);
return xx;
}

unsigned cursor1(void)
{
 unsigned y=0,xx=238,sal=1,z=0,cont3=238,cont4=238;
 mensaje(16,8,"**ABAJO PARA AJUSTAR");
 while(sal==1)
 {
    if(cont4<cont3)
    {
    z=cont4+1;
    }
    if(xx==cont3)
    z=cont3-1;
 /*borrar cursor*/
 for(y=60;y<=69;y++)
  borra_pixel(y,z);
 /*dibujar cursor*/
 for(y=60;y<=69;y++)
  pixel2(y,xx);

   do
   {
     delay_ms(10);
     teclado();
     if(portb.f1==1) //derecha
      {xx++;
      cont3=xx;}
     if(portb.f0==1) //izquierda
      {
      xx--;
      cont4=xx;
      }
    if(portb.f3==1)
    sal=0;
   }
   while(portb==0);
  restaurar();

 }
  borra_linea(13);
  borra_linea(14);
  borra_linea(16);
  mensaje(15,8,"ENTER PARA ZOOM");
 return xx;
}
```