ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Prototipo de Holter Digital"

TESIS DE GRADO

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN ELECTRICIDAD

ESPECIALIZACIÓN ELECTRÓNICA Y

TELECOMUNICACIONES

Presentado por:

JONATHAN LAUTARO BRIONES PEÑALOZA

GISSEL AINET GUARDADO PROAÑO

Guayaquil – Ecuador

2007

AGRADECIMIENTO

A Dios nuestro señor, a quien debemos agradecer todo en esta vida.

A nuestros padres, nuestros hermanos y familiares por su apoyo y palabras de aliento en los momentos difíciles.

A nuestro maestro y amigo, Ing. Miguel Yapur por sus consejos, enseñanzas y guía en la realización de esta tesis.

DEDICATORIA

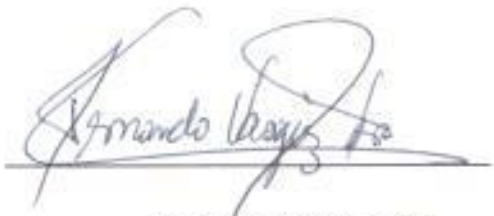
A Dios, a nuestros padres, hermanos, familiares y amigos por toda la fe y esperanza depositada en nosotros.

TRIBUNAL DE GRADUACION

Ing. Holger Cevallos
PRESIDENTE DEL TRIBUNAL

Ing. Miguel Yapur
DIRECTOR DEL TÓPICO

Ing. Hugo Villavicencio
VOCAL PRINCIPAL



Ing. Luis Vásquez
VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la **ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**".

Jonathan Briones Peñaloza

Gissel Guardado Proaño

RESUMEN

En este trabajo se ha elaborado un prototipo de Holter Digital, el cual es un instrumento que registra las ondas cardiacas y que en este caso almacenará la información en memoria para luego ser descargada a una computadora, en donde se observará la señal.

Muchos médicos han comprobado que, con frecuencia, el nerviosismo presente en la consulta afecta las variables medidas; esto dificulta el diagnóstico de ciertas dolencias cardiacas, razón por la cual se debe hacer un seguimiento de varias horas para determinar el problema, siendo ésta la razón por la que se necesita utilizar el Holter.

El producto final desarrollado en este trabajo cumple con las especificaciones anteriormente mencionadas y es económico en comparación con los escasos modelos existentes en el mercado.

Este instrumento ha sido desarrollado a nivel de prototipo e implementa varias etapas; en la primera se registran las ondas cardiacas, la cual es una señal analógica; luego es digitalizada para almacenarla en un banco de memorias; posteriormente, se descargará la información grabada a una computadora a través de un puerto serial, en donde se observará la onda cardiaca; así, el especialista podrá diagnosticar la enfermedad del paciente.

INDICE GENERAL

ABREVIATURAS

ADC	Convertidor analógico-digital
AV	Nodo atrio-ventricular
aVF	Derivación precordial. Vector aumentado a la pierna
aVL	Derivación precordial. Vector aumentado al brazo izquierdo
aVR	Derivación precordial. Vector aumentado al brazo derecho
DLL	Librería de enlace dinámico
E1	Voltaje de electrodo 1
E2	Voltaje de electrodo 2
EKG	Electrocardiograma
f_H	Frecuencia de corte superior
f_L	Frecuencia de corte inferior
f_O	Frecuencia del filtro notch
LA	Brazo izquierdo
LCD	Pantalla de cristal líquido
LL	Brazo derecho
LSB	Dígito binario menos significativo
MSB	Dígito binario más significativo
PC	Computadora personal
RA	Brazo derecho
RL	Pierna derecha
S-A	Nodo sino-atrial

SIMBOLOGÍA

Ag	Plata
Ag^+	Ión plata
Cl^-	Ión cloro
ClAg	Cloruro de plata
Hz	Hertz
KΩ	Kilo-ohmios
R	Resistencia
uF	Microfaradio
V	Voltio

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En 1949 el médico de Montana, Norman Jeff Holter desarrolla una mochila de 37 Kg que podia registrar el EKG de la persona que la portaba y transmitir la señal. Su sistema (el monitor Holter) fue posteriormente muy reducido en tamaño combinándose con la grabación en cinta y utilizado para el registro ambulatorio de EKGs.

Mediante este dispositivo se lograba registrar la actividad cardiaca de un paciente durante periodos largos de tiempo, según determinadas sintomatologías y después, reproducir toda la actividad a través de un monitor con el cual se realizaba el estudio de la actividad cardiaca del paciente.

El objetivo de este prototipo es obtener una referencia de como construir el circuito de este equipo a un bajo costo, en comparación con los productos existentes en el mercado, permitiendo de esta forma ser accesible a la población de bajos recursos.

En este proyecto se muestra la forma en que la señal cardiaca (señal analógica) de poca energía puede ser amplificada por dispositivos electrónicos, luego ser digitalizada, para después ser almacenada en un banco de memorias y estar lista para ser enviada a la computadora por uno de sus puertos.

Por medio de un software, la señal digital puede ser interpretada de tal manera que el médico la visualice en el monitor emitiendo su diagnóstico o llegando a otras conclusiones, dependiendo de la utilidad que se le quiera dar.

Este prototipo puede ser mejorado en algunos de sus componentes, principalmente en su tiempo de grabación añadiéndole más memorias. En cuanto al programa, éste podría proporcionar al usuario una herramienta que le permitiera almacenar la información en el computador como un archivo, así como transportarlo, ya sea a través del correo o cualquier medio convencional para almacenar la información

Capítulo 1

CONCEPTOS FUNDAMENTALES

1.1 El corazón y su anatomía

El corazón es el órgano principal del aparato circulatorio y se encarga de bombear la sangre por todo el cuerpo, haciéndola circular a través de los vasos sanguíneos (arterias, venas y vasos capilares) para suministrar oxígeno y nutrientes al cuerpo. Es el músculo que más trabaja en el cuerpo humano.

Se encuentra localizado casi en el centro del tórax, entre los dos pulmones, apoyándose sobre el músculo diafragma y precisamente sobre la parte central fibrosa de este músculo; está en una situación no totalmente medial, ya que en su parte inferior está ligeramente inclinado hacia el lado izquierdo (cerca de un cuarto a la derecha y tres cuartos a la izquierda de la línea medial).

El corazón posee tres capas, llamadas en su orden: pericardio, miocardio y endocardio. El miocardio es la capa gruesa constituida por el músculo liso. El pericardio es la capa externa que mira a la cavidad pericárdica. El endocardio es la capa interna, lisa y delgada, que reviste las superficies internas del corazón.

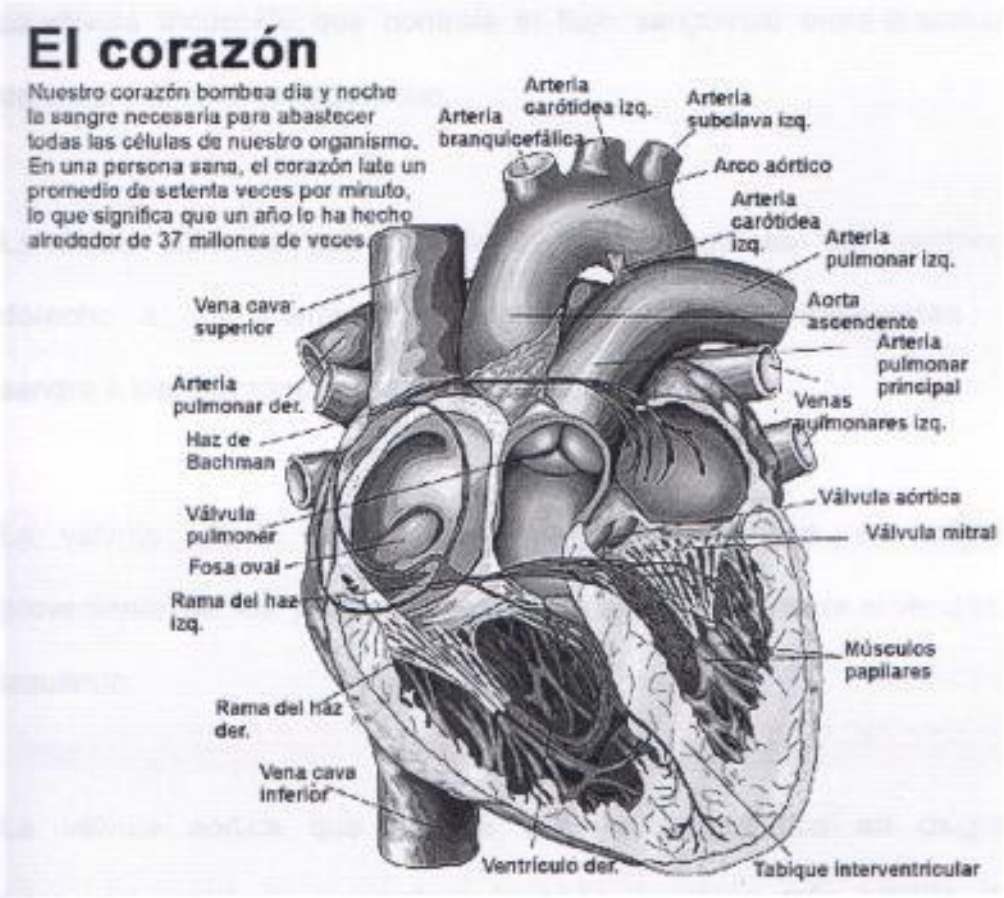


Figura 1.1.- El corazón

(Imagen tomada de www.elmojonformosa.com.ar/psiconeuroinmunoendocrinologia.htm)

El corazón es un órgano de paredes musculares que delimitan cuatro cavidades. Las cavidades superiores se denominan «aurícula izquierda» y «aurícula derecha» y las cavidades inferiores se denominan «ventrículo izquierdo» y «ventrículo derecho». Una pared muscular denominada «tabique» separa las aurículas izquierda y derecha y los ventrículos izquierdo y derecho. Ver figura 1.1.

Consta de 4 válvulas que controlan el flujo de la sangre desde su interior:

- La válvula tricúspide, que controla el flujo sanguíneo entre la aurícula derecha y el ventrículo derecho.

- La válvula pulmonar, que controla el flujo sanguíneo del ventrículo derecho a las arterias pulmonares, las cuales transportan la sangre a los pulmones para oxigenarla.

- La válvula mitral, que permite que la sangre rica en oxígeno proveniente de los pulmones pase de la aurícula izquierda al ventrículo izquierdo.

- La válvula aórtica, que permite que la sangre rica en oxígeno pase del ventrículo izquierdo a la aorta, la arteria más grande del cuerpo, la cual transporta la sangre al resto del organismo.

1.2 Generación de potenciales bioeléctricos

Las células humanas están formadas por la membrana celular, la cual rodea al citoplasma (sustancia rica en agua) que envuelve al núcleo. Es entre el interior y el exterior de la membrana celular que se forman los potenciales bioeléctricos,

Todas las células cardiacas tienen como característica común la propiedad de generar corrientes eléctricas de muy bajo valor como consecuencia de los desplazamientos iónicos a través de la membrana celular. Esto se debe fundamentalmente a las variaciones en las

concentraciones de potasio y sodio; y en menor medida al cloro y al calcio y que continuamente se están produciendo.

El potencial eléctrico que generan las células cardíacas recibe el nombre de **potencial de acción transmembrana** y de cuyas características va a depender la morfología del EKG normal y de todas sus variantes patológicas.

Se sabe que el potencial de acción transmembrana se genera por el movimiento de los iones de potasio desde el interior de la célula hacia el espacio intracelular, mientras que los iones de sodio se movilizan en sentido contrario, es decir desde el exterior de la célula hacia el interior, despolarizándola. Ver figura 1.2. Otros iones como el calcio y el cloro tienen también un papel destacado. Este flujo de iones a través de la membrana celular promueve un potencial eléctrico registrable en forma de curva, tal como se muestra en la misma figura:

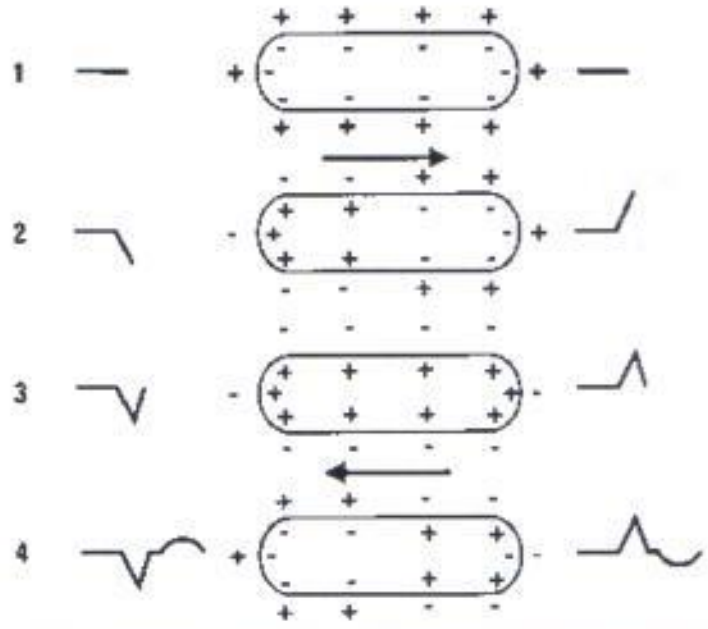


Figura 1.2.- Polarización de una célula cardiaca

Estas cargas intra y extracelulares le dan a la membrana una polaridad, positiva en su cara extracelular y negativa en su cara intracelular llamada potencial de membrana, es en promedio de alrededor de –90 milivoltios en estado de reposo.

1.3 Actividad eléctrica del corazón

Dentro del corazón existen ciertos grupos de células que tienen tareas específicas para su funcionamiento. Así, se encuentra un grupo que produce los estímulos, los cuales se reúnen en una zona que está sobre la aurícula derecha y, a la cual se ha denominado "Nódulo Sino-Atrial" **(nodo S-A)**; ver figura 1.1. Es aquí el lugar donde parten los impulsos eléctricos que salen hacia el resto del corazón. Éste es llamado también el marcapasos natural del corazón.

Los impulsos eléctricos se transmiten a través de las células de las aurículas, excitándolas y produciendo la contracción del músculo que conforma las aurículas (músculo atrial). Esta contracción empuja la sangre hasta los ventrículos.

Los impulsos eléctricos alcanzan posteriormente otro grupo especial de células, que está ubicado en el centro del corazón, en la línea que divide el lado izquierdo y derecho y las porciones superior e inferior. Este grupo de células es conocido como "Nódulo Atrio Ventricular" **(nodo A-V)**; ver figura 1.1. Dicho grupo tiene una tarea especial, consistente en dar un pequeño retraso a los impulsos eléctricos. Este retraso es necesario para que el estímulo a los ventrículos se dé cuando éstos ya estén completamente llenos de sangre que ha sido bombeada por las aurículas.

De esta manera se mantiene una sincronización entre las contracciones auriculares y las ventriculares y aquí juega un papel muy importante el nodo Atrio-Ventricular.

El impulso eléctrico se transmite desde el nodo Atrio-Ventricular hacia los ventrículos por medio de un conductor llamado "Haz de His", que es un grupo de células con características especiales de conducción; finalmente, este impulso es dispersado dentro de los ventrículos mediante otro grupo especial de células llamado "Fibras de Purkinje", en honor a su descubridor; ver figura 1.1. Así es bombeada la sangre hacia el cuerpo.

Este proceso se da a un ritmo que es variable para cada persona, pero que generalmente va entre 50 y 90 latidos por minuto.

1.4 ELECTROCARDIOGRAFÍA

Es el estudio de la actividad eléctrica del corazón. Este método diagnóstico no es invasivo, pero es seguro, accesible y de gran utilidad. Sirve para estudiar los trastornos de conducción eléctrica del corazón, importante en dos ámbitos:

1. Estudio de las alteraciones de la conducción cardiaca, como en los bloqueos cardiacos.
2. Partiendo del principio de que el músculo cardiaco sin aporte de suficiente de oxígeno presenta una señal eléctrica anormal, para el diagnóstico de las lesiones isquémicas del corazón; angina de pecho e infarto del miocardio.

1.4.1 Principio de funcionamiento de la electrocardiografía

La electrocardiografía consiste en recoger las variaciones de potencial eléctrico, amplificarlas y luego registrarlas. La actividad eléctrica del corazón se puede registrar, gracias a que las excitaciones son transmitidas a todas las células a su alrededor, por lo que llegan hasta la piel, donde forman potenciales eléctricos. Estos potenciales son de carácter iónico y no pueden ser medidos por los medidores de voltaje normales, por lo que es necesario adaptarlas a corrientes electrónicas (flujo de electrones), que la realizan los electrodos situados habitualmente en la superficie corporal y que conforman un sistema de derivaciones electrocardiográficas; ver sección siguiente.

Figura 1.3.- Derivación D1

(Imagen tomada de http://es.wikipedia.org/wiki/Electrocardiograma)

1.4.2 Derivaciones

La curva electrocardiográfica presenta distintas morfologías según donde esté colocado el electrodo.

Como el corazón es un órgano tridimensional, es necesario conocer la proyección de los vectores representativos de la actividad eléctrica sobre dos planos, frontal y transversal, para poder estar seguros del camino que sigue la misma; ver figuras

1.4 y 1.5. Para conocer bien la orientación de las fuerzas eléctricas generadas por el corazón, es conveniente registrarlas desde distintas derivaciones. Por ello, en el EKG convencional se utilizan 12 derivaciones, que son:

1. tres bipolares, del plano frontal, que recogen la actividad eléctrica entre dos puntos del cuerpo humano (D_1, D_2, D_3 o I, II y III).
2. Tres monopolares, también del plano frontal, que recogen la actividad eléctrica entre un punto del cuerpo humano y un punto llamado cero que está en el centro del corazón (VR, VL y VF).
3. Seis derivaciones del plano transversal, que son llamadas monopolares (V_1 a V_6).

Por lo tanto, hay doce derivaciones en total, cada una de las cuales registra información de partes concretas del corazón:

- Las derivaciones inferiores (I, II y aVF) detectan la actividad eléctrica desde el punto superior de la región inferior (pared) del corazón. Ésta es la cúspide del ventrículo izquierdo.
- Las derivaciones laterales (I, aVL, V_5 y V_6) detectan la actividad eléctrica desde el punto superior de la pared lateral del corazón, que es la pared lateral del ventrículo izquierdo.

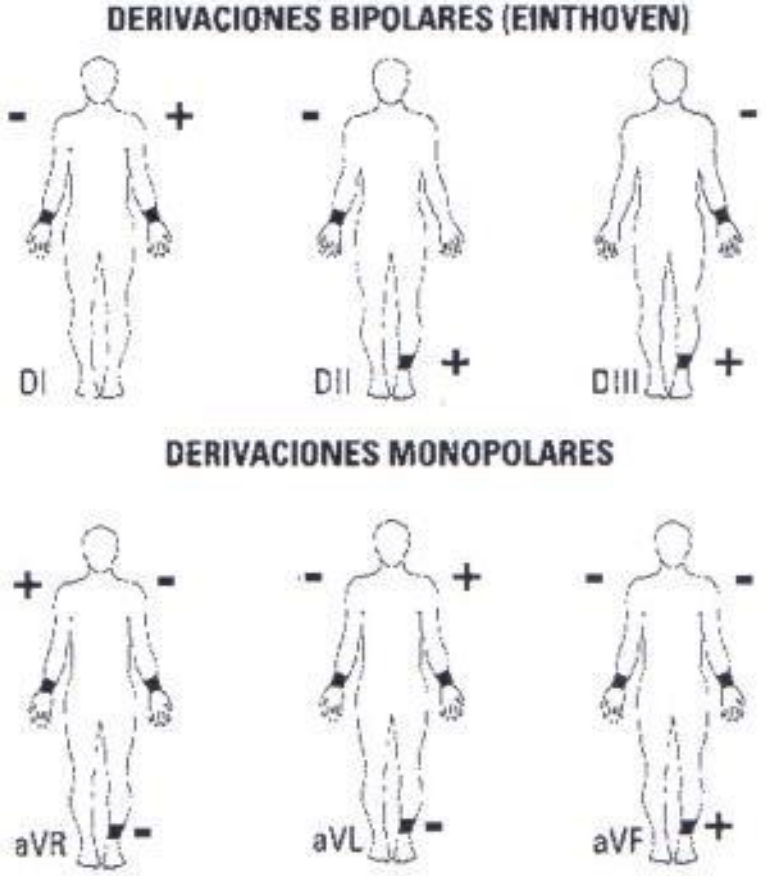


Figura 1.4.- Ubicación de los electrodos de acuerdo a la derivación

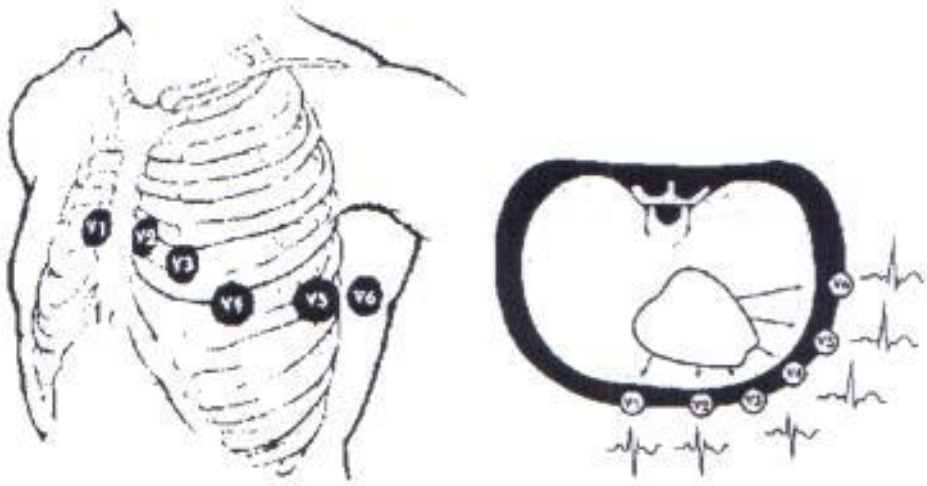


Figura 1.5.- Derivaciones de plano transversal

(Imágenes 1.4 y 1.5 tomadas de
http://dac.escet.urjc.es/PFC/cristina_rodriguez)

- Las derivaciones anteriores, V_1 a V_6 representan la pared anterior del corazón o la pared frontal del ventrículo izquierdo.
- aVR raramente se utiliza para la información diagnóstica, pero indica si los electrodos se han colocado correctamente en el paciente.

1.4.3 Electrodos de medida

Todo electrodo que intercambia iones con su medio y cuyo potencial depende de él, es considerado como uno de medida, pues brinda información sobre las características del medio. Para este caso se consideran los de tipo metálico, que son recubiertos por una capa de una sal o un hidróxido (generalmente muy insoluble) del mismo metal. Estos electrodos son hechos de materiales inertes; es decir, son metales *nobles* (oro, plata, platino) que forman una capa fina de óxido (relativamente insoluble) en su superficie y que sirven básicamente como substrato para una semi-reacción electroquímica.

El electrodo de plata es el más usado; se basa en la siguiente reacción química:

$$Ag \leftrightarrow Ag^{+} + e^{-}$$

$$Ag^{+} + Cl^{-} \leftrightarrow ClAg\downarrow \text{ (se precipita)}$$

El electrón queda libre y puede ser recogido para una posterior medición.

Existen varios tipos de electrodos de superficie, teniendo cada tipo de electrodos sus propias variedades. Los más usados son los de contacto directo, los mismos que existen en las variedades: planos, de succión, de brazalete, etc.

El electrodo se lo puede modelar eléctricamente por medio de un valor de resistencia en serie con la fuente de voltaje.

Las impedancias en electrodos son dependientes de la frecuencia y de la condición del acople piel-electrodo, la que se puede mejorar con el uso de pasta de acople (gel de cloruro de plata). Valores típicos son 6 KΩ a 100 Hz, 200Ω a 1000 Hz; y en el orden de 10KΩ para corriente continua.

El gel de acople es muy importante, ya que mejora considerablemente el desempeño de las interfaces piel-electrodo, por lo que se incrementa considerablemente la estabilidad del sistema.

Capítulo 2

EL HOLTER

La electrocardiografía es una técnica que permite detectar crecimientos de distintas cavidades cardiacas, trastornos del ritmo y de la conducción auriculoventricular e intraventricular, y alteraciones de la repolarización. Se emplea en el diagnóstico de valvulopatías, arritmias, miocardiopatías, enfermedades coronarias y muchas otras enfermedades cardiacas.

Pero la información que se obtiene en el consultorio presenta limitaciones, puesto que se debe tener ciertas consideraciones:

1. En las personas, ciertos parámetros pueden variar cuando acuden a la consulta de un médico debido al nerviosismo que experimentan.
2. Disturbios transitorios del ritmo cardiaco y de episodios intermitentes de isquemia cardiaca, puesto que la posibilidad de detectar estos eventos de carácter fugaz y espaciados es muy pequeña a través del EKG convencional.

2.1 ¿Qué es el Holter?

El Holter o electrocardiógrafo ambulatorio es un dispositivo portátil de pequeño tamaño que permite realizar un registro continuo de un electrocardiograma a lo largo de un período prolongado de tiempo, del cual posteriormente se extrae y se analiza la información grabada

Brinda la posibilidad de realizar un registro continuo del electrocardiograma durante las actividades diarias del individuo; de esta forma, se puede facilitar el seguimiento de las arritmias cardiacas, al permitir al médico estudiar los disturbios del ritmo cardiaco, cuantificar su frecuencia, analizar su complejidad y relacionarlo con los síntomas del paciente para facilitar el diagnóstico y su terapia.

Al evitarle a los pacientes exponerse a la tensión causada por tener que ir a un consultorio médico, proporciona datos más confiables , puesto que se registra el electrocardiograma en distintas situaciones a las que se expone el paciente, ya sea cuando realiza esfuerzos físicos o experimenta diferentes estados de ánimo.

2.2 Evolución del Holter

Hasta hace poco el Holter era un instrumento totalmente analógico, ver figura 2.1, que utilizaba cinta de audio para almacenar la información, algunos modelos aún se venden en el mercado. Hoy en día las cintas han sido substituidas por memorias sólidas que han permitido un almacenamiento totalmente digital de las señales captadas por los electrodos.

La gran ventaja de estos equipos con grabadores de memoria sólida es la reducción de tamaño, del peso y del consumo de energía, además de eliminar partes mecánicas como motor y engranajes que son más propensas al desgaste. Otra ventaja es la indiscutible mejora de la señal generada, eliminando el ruido que acompañaba a las cintas, sea durante la grabación o durante la reproducción.

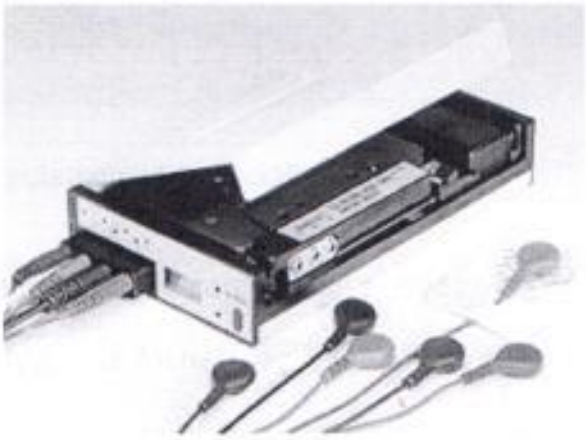


Figura 2.1 Holter de cinta de audio

(Imagen tomada de http://austincardio.com/spanish/ab400.htm)

2.2 Funcionamiento básico

En el consultorio médico se realiza la preparación y limpieza de la piel sobre la que se colocan los electrodos autoadhesivos que son conectados al Holter.

El paciente debe realizar todas sus actividades cotidianas con total normalidad mientras utiliza el aparato y anotar cualquier tipo de incidencia o síntoma que note, junto con la hora exacta a la que ocurrió. Luego de que ha utilizado el aparato por el tiempo prescrito por el médico, debe de acudir nuevamente donde el doctor para que le retire el dispositivo.

Posteriormente el médico se encargará de descargar la información a una computadora en donde podrán visualizar las señales cardiacas, analizarlas y luego dar su diagnóstico.

2.4 Características

El Holter tiene características especiales:

- Registro de la actividad cardiaca en periodos largos de tiempo.
- Muestra la hora y la fecha.
- Presenta bajo tamaño, peso y consumo de energía.
- Livianos.
- Bajo consumo de energía

Capítulo 3

EL PROTOTIPO DESARROLLADO

3.1 Descripción general del sistema

El prototipo desarrollado se basó en los sistemas Holter existentes en la actualidad, los cuales están compuestos de memorias de estado sólido donde almacenan la información, además de permitir la descarga directa a la memoria del computador mediante un puerto del mismo. Este dispositivo es casi por completo digital, excepto en la parte de adquisición de datos. La circuitería interna se describirá a lo largo de este capítulo.

3.2 Diagrama de bloques

El circuito ha sido implementado trabajando en 4 bloques principales:

Adquisición de datos

Corresponde a la etapa en la que se genera la onda cardiaca. Recibe la señal desde los electrodos y amplifica la diferencia de potenciales. En esta etapa la señal es amplificada y despojada de ruido en modo común.

Conversión analógica-digital

Debido a que el resto del circuito es digital, esta etapa adapta a la señal para que pueda ser manejada por las etapas siguientes.

Almacenamiento

En esta etapa toda la información que ha sido digitalizada es guardada en la memoria.

Transmisión de la información

La etapa en la que la información adquirida es enviada a la PC; incluye un programa que permite la visualización de las ondas en el monitor.

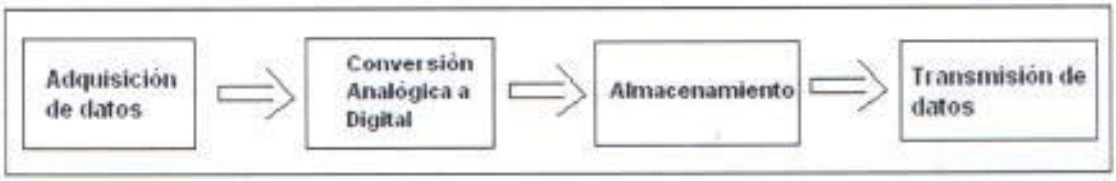


Figura 3.1.- Esquema general de Holter Digital

3.3 Descripción detallada del sistema

A continuación se procederá a describir las diferentes partes que constituyen las etapas del circuito del Holter Digital.

3.3.1 Adquisición de datos

El potencial de acción creado por la contracción de la pared del corazón transmite corrientes iónicas del corazón a lo largo del cuerpo. Las corrientes que se transmiten crean diversos potenciales en diversos puntos en el cuerpo, los cuales pueden ser detectados gracias a los electrodos ubicados en la superficie de la piel. Este potencial eléctrico medido es una señal AC con ancho de banda de 0.05 Hz a 100 Hz con valores de voltaje de un 1 mV pico a pico.

El ruido o artefacto que interfiere en la adquisición de datos proviene principalmente de dos fuentes:

1. Interferencia de 60 Hz
2. Voltaje offset del electrodo

Otros ruidos o altas frecuencias pueden ser producidos por:

1. Artefactos en el acople de piel-electrodo
2. Contracción del músculo .
3. La respiración (que puede ser rítmica o esporádica)
4. La interferencia electromagnética (EMI)
5. El ruido de otros dispositivos electrónicos que se aparezcan en la entrada.

Para eliminar este tipo de interferencias esta etapa consta de un amplificador de instrumentación, un filtro pasa banda, un filtro notch, y un amplificador sencillo.

3.3.1.1 Amplificador de instrumentación

Un amplificador de instrumentación es un dispositivo creado a partir de amplificadores operacionales. Está diseñado para tener una alta impedancia de entrada y un alto rechazo al modo común (CMRR). Se puede construir a base de componentes discretos o se puede encontrar encapsulado; en este trabajo se lo construyó con componentes discretos.

La operación que realiza es la sustracción de sus dos entradas, la cual es multiplicada después por un factor.

El amplificador operacional IC3, ver figura 3.2 y las resistencias R4, R5, R6 y R7 forman un "amplificador diferencial". Esta configuración es utilizada para eliminar el ruido en modo común,

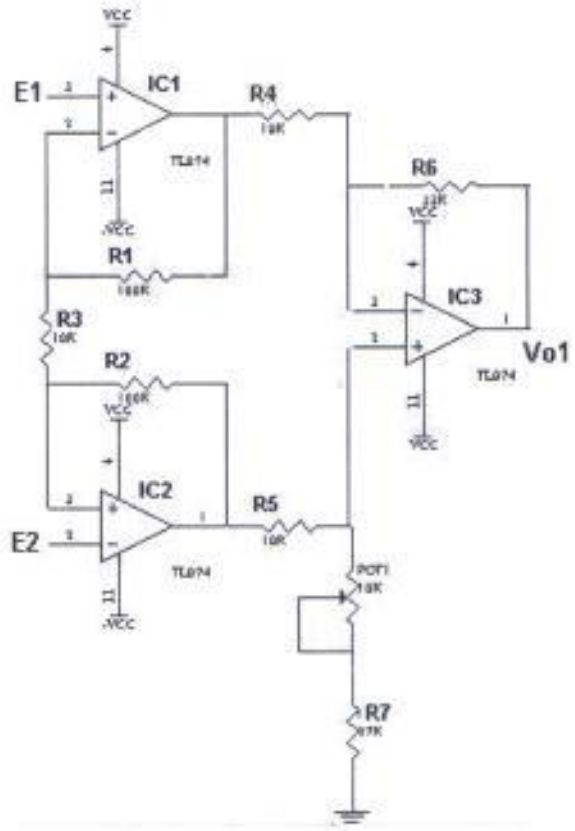


Figura 3.2.- Amplificador de instrumentación

el cual proviene de las líneas de distribución eléctrica de 60 Hz y de las otras clases de ruido expuestas anteriormente.

Se llama ruido en modo común porque aparece simultáneamente en ambos terminales de entrada del amplificador diferencial. Los amplificadores operacionales IC1 e IC2 se los usa para incrementar la impedancia de entrada al circuito. A un amplificador operacional en esta configuración se le llama "amplificador aislado". Dentro del amplificador aislado la resistencia R3 se usa para establecer la ganancia del circuito.

El amplificador de instrumentación se construye conectando los dos amplificadores aislados a un amplificador diferencial básico.

El voltaje en la salida del amplificador de instrumentación se describe con la siguiente expresión:

$$V_{01}= \left(1+\frac{2}{R_3}\right)\times(E_1-E_2)$$

3.3.1.2 Filtro pasa banda

El filtro pasa banda consiste de un filtro pasa alto y un filtro pasa bajo trabajando juntos para dejar pasar un rango de frecuencias. En nuestro electrocardiógrafo el rango de frecuencias permitidas va de 0.05 Hz a 100 Hz, de acuerdo a las normas internacionales. Se aplica el esquema de la figura 3.3.

La frecuencia de corte inferior se calcula de la siguiente manera:

$$f_L = \frac{1}{2\pi R_8 C_3}$$

La frecuencia de corte superior se calcula de la siguiente manera:

$$f_H = \frac{1}{2\pi R_{13} C_4}$$

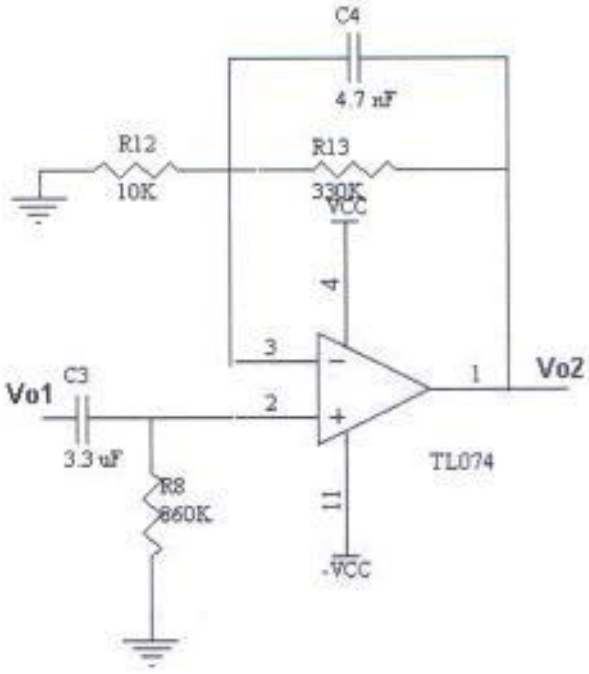


Figura 3.3.- Filtro pasa banda

3.3.1.3 Filtro notch

Dentro de las posibles causas de interferencia, la más común es la producida por las líneas de alimentación que inducen ruido con una frecuencia de 60 Hz. Para eliminar esta interferencia se añadió un filtro notch, el cual rechaza la "frecuencia seleccionada", y permite el paso del resto de frecuencias.

La frecuencia notch se calcula a partir de la siguiente fórmula:

$$f_o = \frac{1}{2\pi RC}$$

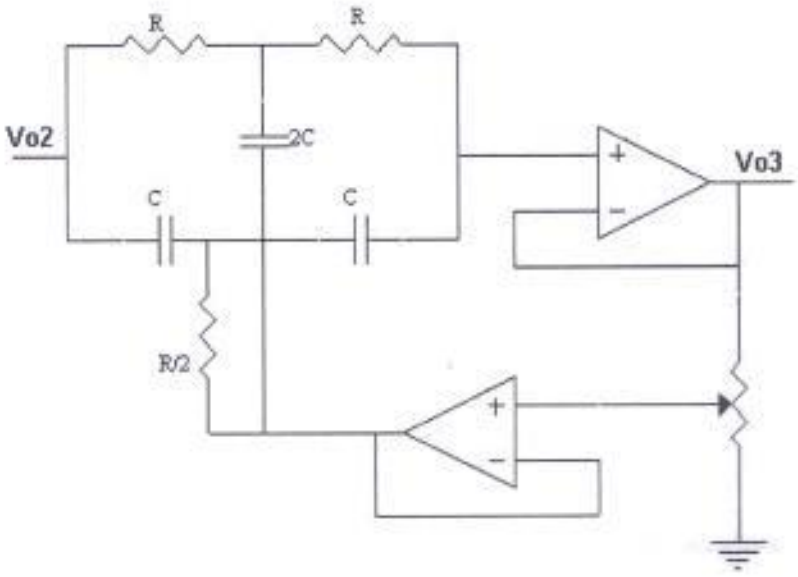


Figura 3.4.- Filtro Notch

3.3.1.4 Seguidor unitario

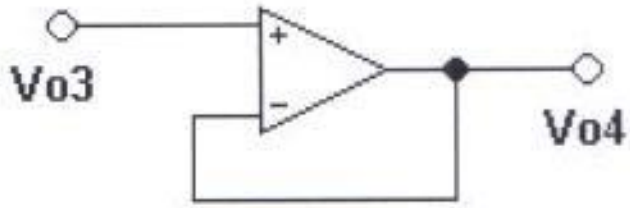


Figura 3.5.- Seguidor unitario

Se usa como un buffer para eliminar efectos de carga o para adaptar impedancias (conectar un dispositivo con gran impedancia a otro con baja impedancia y viceversa).

Como la tensión en los dos pines de entradas es igual:

$$Vo_4 = Vo_3$$

Su aplicación principal en este caso es aislar etapas, evitando que la siguiente etapa afecte a las etapas anteriores.

3.3.1.5 Amplificador inversor

Las dos funciones principales de este circuito consisten en:

1. Amplificar
2. Eliminar voltaje offset

Funciona como un amplificador inversor y suma un nivel DC, que puede ser variado con el potenciómetro hasta eliminar el nivel offset inducido en el circuito.

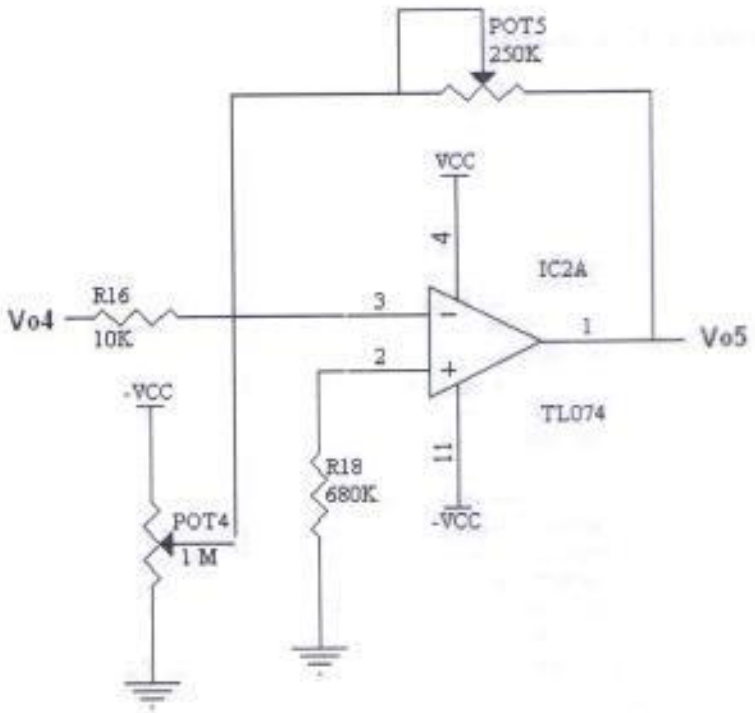


Figura 3.6.- Amplificador inversor

3.3.2 Conversión analógica-digital

La conversión analógica a digital (ADC) se realiza a través del microcontrolador 16F877, ver figura 3.7, el cual se encarga de tomar una muestra de la señal analógica (señal cardiaca) y convertirla en una palabra digital de 8 bits. Las especificaciones de este PIC se encuentran en el **Anexo L.**

Este microcontrolador tiene 8 entradas analógicas; se usa el pin 2 para el ingreso de la señal analógica, en donde cada entrada del microcontrolador internamente tiene un condensador el cual se carga a un valor igual al voltaje de la señal de entrada, el voltaje almacenado por el condensador es la entrada al módulo interno de conversión analógico a digital del integrado, la conversión es realizada a través de aproximaciones sucesivas.

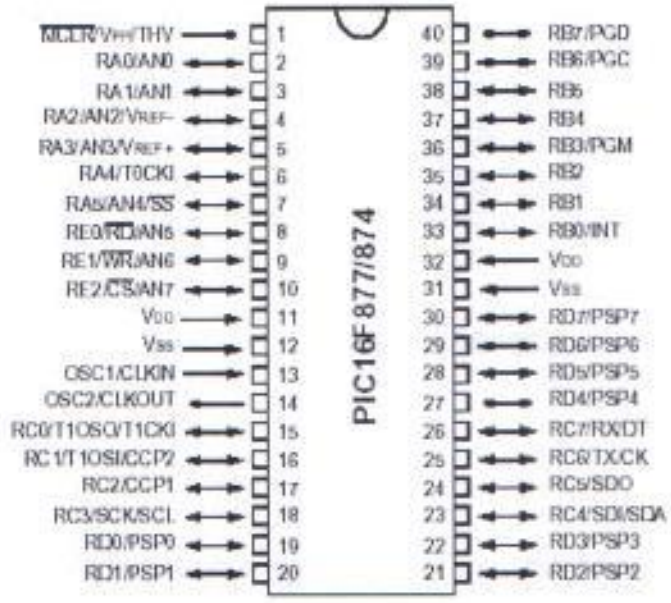


Figura 3.7.- Diagrama del PIC 16F877/874

La lógica de control modifica bit por bit el contenido del registro de control hasta que el contenido de éste se convierte en un equivalente digital de la entrada analógica. Esto se logra con una exactitud que depende de la resolución del ADC. La resolución del ADC se define como la menor variación en la entrada analógica que puede provocar un cambio en la salida digital. Mientras mayor sea la cantidad de bits en la salida del ADC, mayor es la resolución del mismo y por tanto, responde a cambios más pequeños en la señal de entrada.

3.3.3 Almacenamiento

El almacenamiento se realiza por medio de un banco de memorias, en total 10 memorias de 1 Megabyte cada una; éstas son controladas por el PIC 16F877 a través de multiplexores.

Las especificaciones de este PIC se encuentran en el **Anexo K.**

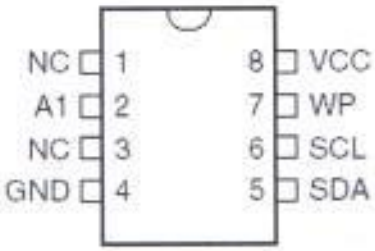


Figura 3.8.- Memoria 24C1024

Descripción de pines

Serial Clock (SCL).- La entrada SCL es usada para el reloj de la memoria, trabajando con flancos positivos para el ingreso de datos y flancos negativos para la salida de datos.

Serial Data (SDA).- El pin SDA es usado para la transferencia bidireccional de información. Este pin es colector abierto, lo cual permite que las memorias puedan ligarse para formar un bus común. En cualquier momento, todas las salidas de compuerta ligada al bus, excepto una, deben mantenerse en su estado alto o bajo, dependiendo de si se desea transmitir un 1 o un 0 en el bus.

Device / Page addresses (A1).- Se lo usa para habilitar a la memoria; es de gran ayuda para seleccionar una memoria de entre un grupo de esos integrados, usa lógica negativa.

Write protect (WP).- Este pin es útil para proteger el contenido completo de la memoria de operaciones inadvertidas de escritura. La escritura se habilita cuando recibe cero voltios (lógica negativa) y se inhabilita cuando recibe cinco voltios.

3.3.3.1 Proceso de escritura

De acuerdo a lo anteriormente expuesto se necesita que WP y A1 reciban cero voltios para que la memoria pueda guardar los datos que recibe en el pin SDA; de esta forma se puede habilitar una memoria y deshabilitar las otras usando los pines WP y A1. Debido a que se usan 10 memorias se requiere multiplexores para seleccionar a que memoria van las señales WP y A1.

Todas las memorias están conectadas a través de SDA, formando un bus de dato gracias a que son circuitos

integrados con colector abierto. Los datos son enviados por el microcontrolador, que también maneja los multiplexores que envían las señales WP y A1.

Podría pensarse que al trabajar WP y A1 en lógica negativa se pueda enviar la misma señal a ambos pines, pero realmente se necesita habilitar la memoria para luego activar la escritura, entonces WP debe tener un retardo con respecto a la señal en el pin A1, Por eso se necesitó usar 4 multiplexores, 2 para WP y 2 para A1.

3.3.3.2 Proceso de lectura

En el proceso de lectura A1 debe ser enviada a través de los multiplexores hacia la memoria seleccionada, WP debe estar en alto y se trabajará con los flancos negativos de SCL (esta señal también es proporcionada por el microcontrolador). Así, los datos serán enviados al integrado MAX232, el cual convertirá las señales de lógica TTL a interfaz RS232.

3.3.4 Transmisión de la información

La manera de conectar los dos dispositivos (PC y Holter) es mediante comunicación serie-asíncrona. En ellos los bits de datos se transmiten "en serie" (uno detrás de otro) y cada dispositivo tiene su propio reloj. Previamente se ha acordado que ambos dispositivos transmitirán datos a la misma velocidad.

3.3.4.1 Comunicación serie-asíncrona

Los datos serie se encuentran encapsulados en tramas de la forma:

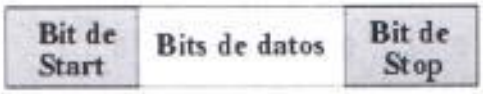


Figura 3.9.- Esquema de una trama asíncrona

Primero se envía un bit de inicio, a continuación los bits de datos (primero el bit de mayor peso) y finalmente los bits de parada.

El número de bits de datos y de bits de parada es uno de los parámetros configurables, así como el criterio de paridad par o impar para la detección de errores. Normalmente, las comunicaciones serie tienen los siguientes parámetros: 1 bit de Start, 8 bits de Datos, 1 bit de Stop y sin paridad, que es el esquema que hemos utilizado para el Holter.

En esta figura se puede ver un ejemplo de la transmisión del dato binario 10011010. La línea en reposo está a nivel alto:

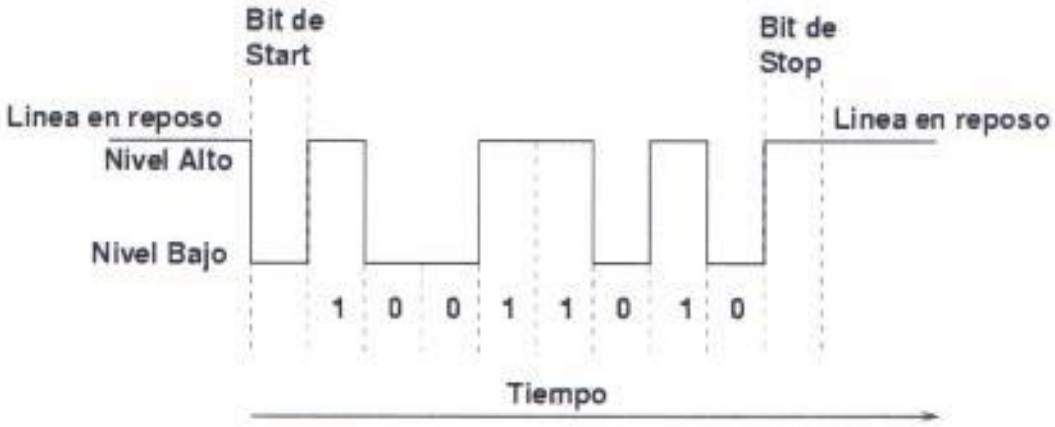


Figura 3.10.- Transmisión asíncrona

3.3.4.2 Norma RS232

La Norma RS-232 fue definida para conectar un ordenador a un modem. Además de transmitirse los datos de una forma serie-asíncrona son necesarias una serie de señales adicionales, que se definen en la norma. Las tensiones empleadas están comprendidas entre +15 y -15 voltios.

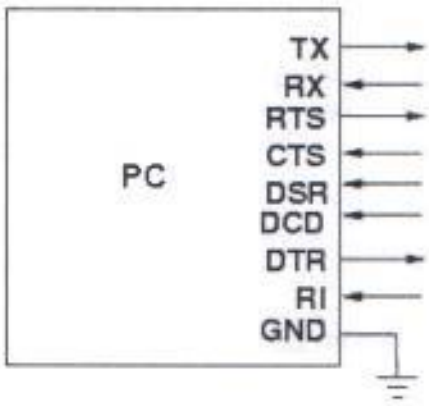


Figura 3.11.- Norma RS-232

3.3.4.3 Conexión de un microcontrolador al puerto serie de la PC

Para conectar la PC a un microcontrolador por el puerto serie se utilizan las señales Tx, Rx y GND. La PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles. Uno de estos circuitos, que se utiliza mucho, es el MAX232.

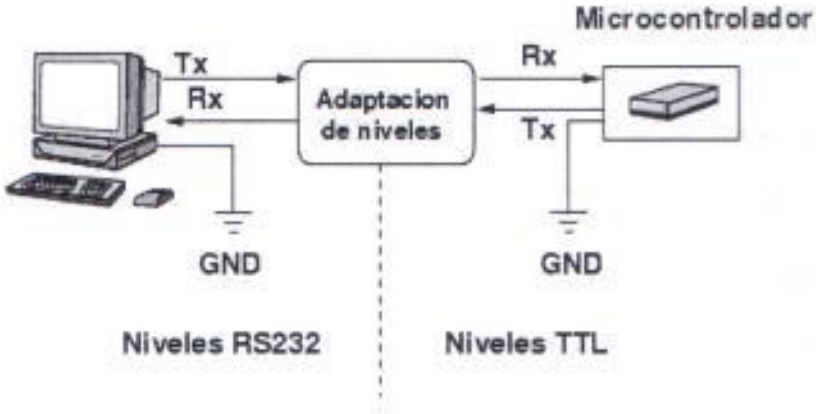


Figura 3.12.- Interconexión entre el computador y el microcontrolador

3.3.4.4 El circuito integrado MAX 232

Este circuito permite adaptar los niveles RS232 y TTL, permitiendo conectar un PC con un microcontrolador. Sólo es necesario este chip y 4 condensadores electrolíticos de 22 µF. El esquema es el siguiente:

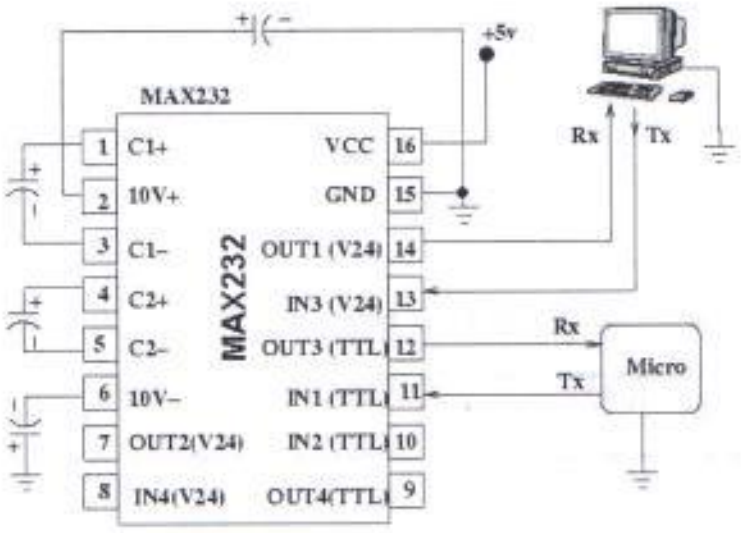


Figura 3.13.- Esquema de utilización del MAX232

3.3.5 Pantalla LCD

En la actualidad los módulos LCD existen en una gran variedad de versiones, clasificados en dos grupos. El primer grupo está referido a los módulos LCD de caracteres (solamente se podrán presentar caracteres y símbolos especiales en las líneas predefinidas en el módulo LCD) y el segundo grupo está referido a los módulos LCD matriciales (se podrán presentar caracteres, símbolos especiales y gráficos). Los módulos LCD varían su tamaño físico dependiendo de la marca; por lo tanto en la actualidad no existe un tamaño estándar para los módulos LCD.

3.3.5.1 Identificación de los pines de conexión de un módulo LCD no matricial

Los pines de conexión de un módulo LCD han sido estandarizados por lo cual en la mayoría de ellos son exactamente iguales siempre y cuando la línea de caracteres no sobrepase los ochenta caracteres por línea. Por otro lado, es de

suma importancia localizar exactamente cual es el pin número 1 ya que en algunos módulos se encuentra hacia la izquierda y en otros módulos se encuentra a la derecha.

TABLA 3.1.- Asignación de pines para un módulo LCD

Pin N-.	Sismología	Nivel	I/O	Función
1	VSS	-	-	0 Vlts. Tierra (GND).
2	VCC	-	-	+ 5 Vlts. DC.
3	Vee = Vc	-	-	Ajuste del contraste.
4	RS	0/1	I	0= Escribir en el módulo LCD. 1= Leer del módulo LCD
5	R/W	0/1	I	0= Entrada de una instrucción. 1= Entrada de un dato.
6	E	1	I	Habilitación del módulo LCD
7	DB0	0/1	I/O	BUS DE DATO LINEA 1 (LSB).
8	DB1	0/1	I/O	BUS DE DATO LINEA 2
9	DB2	0/1	I/O	BUS DE DATO LINEA 3
10	DB3	0/1	I/O	BUS DE DATO LINEA 4
11	DB4	0/1	I/O	BUS DE DATO LINEA 5
12	DB5	0/1	I/O	BUS DE DATO LINEA 6
13	DB6	0/1	I/O	BUS DE DATO LINEA 7
14	DB7	0/1	I/O	BUS DE DATO LINEA 8 (MSB).
15	A	-	-	LED (+) Back Light
16	K	-	-	LED (-) Back Light.

3.3.5.2 Interpretación del significado de los pines del módulo LCD

Los **pines número 1 y 2** están destinados para conectarles los 5 voltios que requiere el módulo para su funcionamiento y el pin **número 3** es utilizado para ajustar el contraste de la pantalla; es decir, colocar los caracteres mas oscuros o mas claros para poderse observar mejor.

La siguiente imagen muestra cómo deben estar conectados los tres primeros pines. La resistencia representada como Pot1 es un potenciómetro variable que puede oscilar entre 10 K y 20 K indiferentemente.

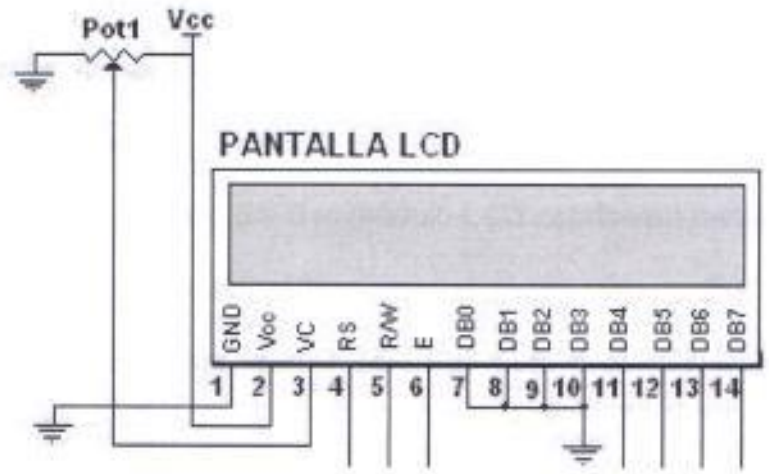


Figura 3.14.- Configuración de pines del módulo LCD

El **Pin número 4:** denominado "RS" trabaja paralelamente al bus de datos del módulo LCD (Bus de datos son los pines del 7 al 14). Este bus es utilizado de dos maneras, ya que se podrá colocar un dato que representa una instrucción o podrá colocar un dato que tan solo representa un símbolo o un carácter

alfanumérico; pero para que el módulo LCD pueda entender la diferencia entre un dato o una instrucción, se utiliza el pin número 4 para tal fin.

Si el pin número 4 = 0 le dirá al módulo LCD que está presente en el bus de datos una instrucción; por el contrario, si el pin número 4 = 1 le dirá al módulo LCD que esta presente un símbolo o un carácter alfa numérico.

El **pin número 5:** denominado "R/W" trabaja paralelamente al bus de datos del módulo LCD (Bus de datos son los pines del 7 al 14). También es utilizado de dos maneras, ya que se podrá decirle al módulo LCD que escriba en pantalla el dato que está presente en el bus; por otro lado también podrá leer que dato esta presente en el bus.

Si el pin número 5 = 0 el módulo LCD escribe en pantalla el dato que esta presente el bus; pero si el pin número 5 = 1 significa que se necesita leer el dato que esta presente en el bus del módulo LCD.

El **pin número 6:** denominado "E", significa habilitación del módulo LCD y tiene una finalidad básica: conectar y desconectar el módulo. Esta desconexión no estará referida al voltaje que le suministra la corriente al módulo; la desconexión significa tan solo que se hará caso omiso a todo lo que esté presente en el bus de datos de dicho módulo LCD.

En la mayoría de los circuitos electrónicos modernos que incluyan elementos electrónicos como microcontroladores, memorias y módulos LCD, utilizan el mismo bus de datos. Esto es para no tener un bus de datos independientemente por cada elemento electrónico, lo cual implicaría que los circuitos electrónicos sean mucho mas grandes por la cantidad de conexiones necesaria a cada uno de los elementos.

Ahora, como los microcontroladores, memorias y módulos LCD utilizan el mismo bus de datos, deberá existir en cada uno de ellos un pin de habilitación "E" que permita desconectar y conectar cuando sea necesario. Por ejemplo, si se necesita trabajar con la memoria RAM para obtener o escribir cierta información, será necesario que se deshabilite el módulo LCD para que no presente basura en la pantalla, o se ejecuten instrucciones no deseadas.

Los **pines desde el número 7 hasta el número 14** representan 8 líneas que se utilizan para colocar el dato que representa una instrucción para el módulo LCD o un carácter alfanumérico. El bus de datos es de 8 bits de longitud y el bit menos significativo esta representado en el Pin número 7, el pin más significativo está representado en el pin número 14.

3.3.6 Fuente de poder

Es la etapa encargada de proveer energía al Holter. Hay que distinguir que la parte analógica se polariza mediante dos baterias de 9 voltios, principalmente por los amplificadores operacionales TL074, la parte digital recibe el voltaje de +5 voltios proporcionados por el integrado 7805 (regulador de voltaje), el cual es polarizado con una de las baterías de +9 voltios.

Capítulo 4

DISEÑO DEL PROGRAMA CONTROLADOR

4.1 Descripción del programa de la interfaz gráfica

La presentación de las señales cardiacas a través del computador se realiza mediante un programa elaborado en visual Basic 5.0.

Visual Basic es un lenguaje de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Para el desarrollo del programa se utilizaron dos librerias dinámicas (dll) que nos facilitarían la tarea y son: iplotlibrary.dll y mscomm32.dll. La primera nos brinda facilidades para la graficación de las señales y la segunda para manejar el puerto serial.

4.1.1 Algoritmo del programa de la interfaz gráfica

El programa se basa en el diagrama de flujo mostrado en la figura 4.1. Inicialmente se activa el puerto serial, una vez que se recibe una señal llamad "ind" enviada por el PIC, el programa comienza a solicitar el dato al PIC, después que lo recibe empieza a graficar el punto, esta operación se vuelve un ciclo que continúa mientras la señal "ind" esté presente.

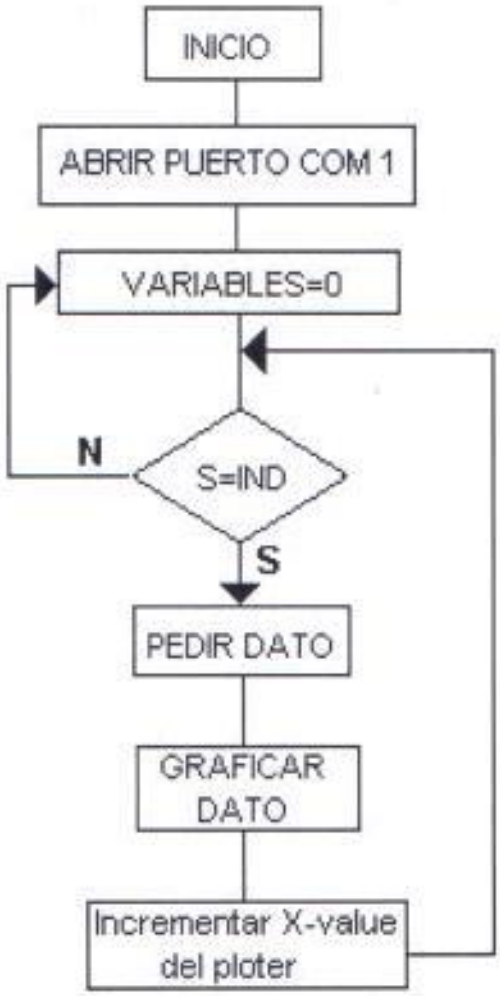


Figura 4.1.- Diagrama de flujo general

Las funciones presentes en la librería iplotlibrary.dll son las que se encargan del proceso de graficación. Cada punto se grafica en el eje y, conforme se solicitan datos se incrementa el valor X-value para que los puntos se continúen graficando a lo largo del eje x.

4.1.2 Código fuente del programa de la interfaz gráfica.

```
VERSION 5.00
Object = "{DA259054-D93B-498C-8C10-DEBD83EF1357}#1.0#0"; "iPlotLibrary.ocx"
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Begin VB.Form Form1
  Caption = "Form1"
  ClientHeight = 6060
  ClientLeft = 60
  ClientTop = 450
  ClientWidth = 15240
  LinkTopic = "Form1"
  ScaleHeight = 6060
  ScaleWidth = 15240
  StartUpPosition = 3 'Windows Default
  Begin VB.Timer Timer1
    Enabled = 0 'False
    Interval = 50
    Left = 1920
    Top = 6120
  End
  Begin MSCommLib.MSComm pserie
    Left = 600
    Top = 7200
    _ExtentX = 1005
    _ExtentY = 1005
    _Version = 393216
```

```
  DTREnable      = -1 'True
  RThreshold     = 1
End
Begin iPlotLibrary.iPlotX iPlotX
  Height         = 5595
  Left           = 120
  TabIndex       = 0
  Top            = 120
  Width          = 15000
  DataViewZHorz  = 1
  DataViewZVert  = 1
  XYAxesReverse  = 0  'False
  OuterMarginLeft = 5
  OuterMarginTop  = 5
  OuterMarginRight= 5
  OuterMarginBottom= 5
  PrintOrientation= 1
  PrintMarginLeft = 1
  PrintMarginTop  = 1
  PrintMarginRight= 1
  PrintMarginBottom= 1
  PrintShowDialog = -1 'True
  UpdateFrameRate = 60
  BackGroundColor = 0
  BorderStyle     = 2
  AutoFrameRate   = -1 'True
  HintsShow       = -1 'True
```

```
    HintsPause    = 500
    HintsHidePause = 2500
    TitleVisible   = -1 'True
    TitleText      = "Untitled"
    TitleMargin    = 0
    BeginProperty          TitleFont          {0BE35203-8F91-11CE-9DE3-
00AA004BB851}
      Name         = "Arial"
      Size         = 14.25
      Charset      = 0
      Weight       = 700
      Underline    = 0 'False
      Italic       = 0 'False
      Strikethrough = 0 'False
    EndProperty
    TitleFontColor = 16777215
    UserCanEditObjects= -1 'True
    LogFileName    = ""
    LogBufferSize  = 0
    OptionSaveAllProperties= 0 'False
    BeginProperty          HintsFont          {0BE35203-8F91-11CE-9DE3-
00AA004BB851}
      Name         = "MS Sans Serif"
      Size         = 8.25
      Charset      = 0
      Weight       = 400
      Underline    = 0 'False
      Italic       = 0 'False
```

```
    Strikethrough = 0 'False
  EndProperty
  HintsFontColor = -2147483640
  BeginProperty AnnotationDefaultFont {0BE35203-8F91-11CE-9DE3-
00AA004BB851}
    Name        = "MS Sans Serif"
    Size        = 8.25
    Charset     = 0
    Weight      = 400
    Underline   = 0 'False
    Italic      = 0 'False
    Strikethrough = 0 'False
  EndProperty
  AnnotationDefaultFontColor= 16777215
  AnnotationDefaultBrushStlye= 0
  AnnotationDefaultBrushColor= 16777215
  AnnotationDefaultPenStlye= 0
  AnnotationDefaultPenColor= 16777215
  AnnotationDefaultPenWidth= 1
  Object.Width        = 1000
  Object.Height       = 373
  UserCanAddRemoveChannels= 0 'False
  Object.Visible      = -1 'True
  Enabled     = -1 'True
  EditorFormStyle = 0
  CopyToClipBoardFormat= 0
  PrintDocumentName= "Untitled"
  PrinterName   = ""
```

```
ClipAnnotationsToAxes=  -1  'True
BackGroundGradientEnabled=  0  'False
BackGroundGradientDirection=  0
BackGroundGradientStartColor=  16711680
BackGroundGradientStopColor=  0
DataViewZHorz  =  1
DataViewZVert  =  1
ChannelCount  =  1
XAxisCount  =  1
YAxisCount  =  1
ToolBarCount  =  1
LegendCount  =  1
DataViewCount  =  1
DataCursorCount =  1
LimitCount  =  0
LabelCount  =  1
TranslationCount=  0
ToolBar0.Name  =  "Toolbar 1"
ToolBar0.Visible=  1
ToolBar0.Enabled=  1
ToolBar0.Layer  =  100
ToolBar0.PopupEnabled=  1
ToolBar0.Horizontal=  1
ToolBar0.ZOrder =  3
ToolBar0.StartPercent=  0
ToolBar0.StopPercent=  100
ToolBar0.ShowResumeButton=  1
```

```
ToolBar0.ShowPauseButton=  1
ToolBar0.ShowAxesModeButtons=  1
ToolBar0.ShowZoomInOutButtons=  1
ToolBar0.ShowSelectButton=  0
ToolBar0.ShowZoomBoxButton=  1
ToolBar0.ShowCursorButton=  1
ToolBar0.ShowEditButton=  1
ToolBar0.ShowCopyButton=  1
ToolBar0.ShowSaveButton=  1
ToolBar0.ShowPrintButton=  1
ToolBar0.ZoomInOutFactor=  2
ToolBar0.FlatBorder=  0
ToolBar0.FlatButtons=  0
ToolBar0.SmallButtons=  0
Legend0.Name  =  "Legend 1"
Legend0.Visible =  1
Legend0.Enabled =  1
Legend0.Layer  =  100
Legend0.PopupEnabled=  1
Legend0.Horizontal=  0
Legend0.ZOrder  =  2
Legend0.StartPercent=  0
Legend0.StopPercent=  100
Legend0.MarginLeft=  1
Legend0.MarginTop=  1
Legend0.MarginRight=  1
Legend0.MarginBottom=  1
```

```
Legend0.BackGroundColor= 8421504
Legend0.BackGroundTransparent= 1
Legend0.SelectedItemBackGroundColor= 65535
Legend0.SelectedItemFont.Charset= 1
Legend0.SelectedItemFont.Color= 0
Legend0.SelectedItemFont.Height= -11
Legend0.SelectedItemFont.Name= "MS Sans Serif"
Legend0.SelectedItemFont.Pitch= 0
Legend0.SelectedItemFont.Style= 0
Legend0.ShowColumnLine= 1
Legend0.ShowColumnMarker= 0
Legend0.ShowColumnXAxisTitle= 0
Legend0.ShowColumnYAxisTitle= 0
Legend0.ShowColumnXValue= 0
Legend0.ShowColumnYValue= 0
Legend0.ShowColumnYMax= 0
Legend0.ShowColumnYMin= 0
Legend0.ShowColumnYMean= 0
Legend0.CaptionColumnTitle= "Title"
Legend0.CaptionColumnXAxisTitle= "X-Axis"
Legend0.CaptionColumnYAxisTitle= "Y-Axis"
Legend0.CaptionColumnXValue= "X"
Legend0.CaptionColumnYValue= "Y"
Legend0.CaptionColumnYMax= "Y-Max"
Legend0.CaptionColumnYMin= "Y-Min"
Legend0.CaptionColumnYMean= "Y-Mean"
Legend0.Font.Charset= 1
```

```
Legend0.Font.Color= 16777215
Legend0.Font.Height= -11
Legend0.Font.Name= "MS Sans Serif"
Legend0.Font.Pitch= 0
Legend0.Font.Style= 0
Legend0.ColumnSpacing= 0.5
Legend0.RowSpacing= 0.25
Legend0.WrapColDesiredCount= 1
Legend0.WrapColAutoCountEnabled= 0
Legend0.WrapColAutoCountMax= 100
Legend0.WrapColSpacingMin= 2
Legend0.WrapColSpacingAuto= 1
Legend0.WrapRowDesiredCount= 5
Legend0.WrapRowAutoCountEnabled= 1
Legend0.WrapRowAutoCountMax= 100
Legend0.WrapRowSpacingMin= 0.25
Legend0.WrapRowSpacingAuto= 0
Legend0.ColumnTitlesVisible= 0
Legend0.ColumnTitlesFont.Charset= 1
Legend0.ColumnTitlesFont.Color= 16776960
Legend0.ColumnTitlesFont.Height= -11
Legend0.ColumnTitlesFont.Name= "MS Sans Serif"
Legend0.ColumnTitlesFont.Pitch= 0
Legend0.ColumnTitlesFont.Style= 1
Legend0.ChannelNameMaxWidth= 0
XAxis0.Name = "X-Axis 1"
XAxis0.Visible = 1
```

XAxis0.Enabled = 1

XAxis0.Layer = 100

XAxis0.PopupEnabled= 1

XAxis0.Horizontal= 1

XAxis0.ZOrder = 0

XAxis0.StartPercent= 0

XAxis0.StopPercent= 100

XAxis0.Min = 0

XAxis0.Span = 100

XAxis0.DesiredStart= 20

XAxis0.DesiredIncrement= 0

XAxis0.ReverseScale= 0

XAxis0.InnerMargin= 5

XAxis0.OuterMargin= 5

XAxis0.Title = "X-Axis 1"

XAxis0.TitleMargin= 0.25

XAxis0.TitleFont.Charset= 1

XAxis0.TitleFont.Color= 16777215

XAxis0.TitleFont.Height= -13

XAxis0.TitleFont.Name= "Arial"

XAxis0.TitleFont.Pitch= 0

XAxis0.TitleFont.Style= 1

XAxis0.TitleShow= 0

XAxis0.MajorLength= 7

XAxis0.MinorLength= 3

XAxis0.MinorCount= 1

XAxis0.LabelsVisible= 1

XAxis0.LabelsMargin= 0.25

XAxis0.LabelsFont.Charset= 1

XAxis0.LabelsFont.Color= 16777215

XAxis0.LabelsFont.Height= -11

XAxis0.LabelsFont.Name= "MS Sans Serif"

XAxis0.LabelsFont.Pitch= 0

XAxis0.LabelsFont.Style= 0

XAxis0.LabelSeparation= 2

XAxis0.LabelsPrecision= 3

XAxis0.LabelsPrecisionStyle= 0

XAxis0.LabelsFormatStyle= 0

XAxis0.DateTimeFormat= "hh:nn:ss"

XAxis0.LabelsMinLength= 5

XAxis0.LabelsMinLengthAutoAdjust= 0

XAxis0.ScaleLineShow= 1

XAxis0.ScaleLinesShow= 1

XAxis0.ScaleLinesColor= 16777215

XAxis0.StackingEndsMargin= 0.5

XAxis0.ScaleType= 0

XAxis0.TrackingEnabled= 1

XAxis0.TrackingStyle= 3

XAxis0.TrackingAlignFirstStyle= 2

XAxis0.TrackingScrollCompressMax= 0

XAxis0.CursorUseDefaultFormat= 1

XAxis0.CursorFormatStyle= 0

XAxis0.CursorDateTimeFormat= "hh:nn:ss"

XAxis0.CursorPrecisionStyle= 0

XAxis0.LabelsMargin= 0.25

XAxis0.LabelsFont.Charset= 1

XAxis0.LabelsFont.Color= 16777215

XAxis0.LabelsFont.Height= -11

XAxis0.LabelsFont.Name= "MS Sans Serif"

XAxis0.LabelsFont.Pitch= 0

XAxis0.LabelsFont.Style= 0

XAxis0.LabelSeparation= 2

XAxis0.LabelsPrecision= 3

XAxis0.LabelsPrecisionStyle= 0

XAxis0.LabelsFormatStyle= 0

XAxis0.DateTimeFormat= "hh:nn:ss"

XAxis0.LabelsMinLength= 5

XAxis0.LabelsMinLengthAutoAdjust= 0

XAxis0.ScaleLineShow= 1

XAxis0.ScaleLinesShow= 1

XAxis0.ScaleLinesColor= 16777215

XAxis0.StackingEndsMargin= 0.5

XAxis0.ScaleType= 0

XAxis0.TrackingEnabled= 1

XAxis0.TrackingStyle= 3

XAxis0.TrackingAlignFirstStyle= 2

XAxis0.TrackingScrollCompressMax= 0

XAxis0.CursorUseDefaultFormat= 1

XAxis0.CursorFormatStyle= 0

XAxis0.CursorDateTimeFormat= "hh:nn:ss"

XAxis0.CursorPrecisionStyle= 0

```
XAxis0.CursorPrecision= 3
XAxis0.CursorMinLength= 5
XAxis0.CursorMinLengthAutoAdjust= 0
XAxis0.LegendUseDefaultFormat= 1
XAxis0.LegendFormatStyle= 0
XAxis0.LegendDateTimeFormat= "hh:nn:ss"
XAxis0.LegendPrecisionStyle= 0
XAxis0.LegendPrecision= 3
XAxis0.LegendMinLength= 5
XAxis0.LegendMinLengthAutoAdjust= 0
XAxis0.CursorScaler= 1
XAxis0.ScrollMinMaxEnabled= 0
XAxis0.ScrollMax= 200
XAxis0.ScrollMin= 0
XAxis0.RestoreValuesOnResume= 1
XAxis0.MasterUIInput= 0
XAxis0.CartesianStyle= 0
XAxis0.CartesianChildRefAxisName= ""
XAxis0.CartesianChildRefValue= 0
XAxis0.GridLinesVisible= 1
YAxis0.Name = "Y-Axis 1"
YAxis0.Visible = 1
YAxis0.Enabled = 1
YAxis0.Layer = 100
YAxis0.PopupEnabled= 1
YAxis0.Horizontal= 0
YAxis0.ZOrder = 0
```

YAxis0.StartPercent= 0

YAxis0.StopPercent= 100

YAxis0.Min = 0

YAxis0.Span = 150

YAxis0.DesiredStart= 0

YAxis0.DesiredIncrement= 0

YAxis0.ReverseScale= 0

YAxis0.InnerMargin= 5

YAxis0.OuterMargin= 5

YAxis0.Title = "Y-Axis 1"

YAxis0.TitleMargin= 0.25

YAxis0.TitleFont.Charset= 1

YAxis0.TitleFont.Color= 16777215

YAxis0.TitleFont.Height= -13

YAxis0.TitleFont.Name= "Arial"

YAxis0.TitleFont.Pitch= 0

YAxis0.TitleFont.Style= 1

YAxis0.TitleShow= 0

YAxis0.MajorLength= 7

YAxis0.MinorLength= 3

YAxis0.MinorCount= 1

YAxis0.LabelsVisible= 1

YAxis0.LabelsMargin= 0.25

YAxis0.LabelsFont.Charset= 1

YAxis0.LabelsFont.Color= 16777215

YAxis0.LabelsFont.Height= -11

YAxis0.LabelsFont.Name= "MS Sans Serif"

YAxis0.LabelsFont.Pitch= 0

YAxis0.LabelsFont.Style= 0

YAxis0.LabelSeparation= 2

YAxis0.LabelsPrecision= 3

YAxis0.LabelsPrecisionStyle= 0

YAxis0.LabelsFormatStyle= 0

YAxis0.DateTimeFormat= "hh:nn:ss"

YAxis0.LabelsMinLength= 5

YAxis0.LabelsMinLengthAutoAdjust= 0

YAxis0.ScaleLineShow= 1

YAxis0.ScaleLinesShow= 1

YAxis0.ScaleLinesColor= 16777215

YAxis0.StackingEndsMargin= 0.5

YAxis0.ScaleType= 0

YAxis0.TrackingEnabled= 1

YAxis0.TrackingStyle= 0

YAxis0.TrackingAlignFirstStyle= 3

YAxis0.TrackingScrollCompressMax= 0

YAxis0.CursorUseDefaultFormat= 1

YAxis0.CursorFormatStyle= 0

YAxis0.CursorDateTimeFormat= "hh:nn:ss"

YAxis0.CursorPrecisionStyle= 0

YAxis0.CursorPrecision= 3

YAxis0.CursorMinLength= 5

YAxis0.CursorMinLengthAutoAdjust= 0

YAxis0.LegendUseDefaultFormat= 1

YAxis0.LegendFormatStyle= 0

YAxis0.LegendDateTimeFormat= "hh:nn:ss"

YAxis0.LegendPrecisionStyle= 0

YAxis0.LegendPrecision= 3

YAxis0.LegendMinLength= 5

YAxis0.LegendMinLengthAutoAdjust= 0

YAxis0.CursorScaler= 1

YAxis0.ScrollMinMaxEnabled= 0

YAxis0.ScrollMax= 100

YAxis0.ScrollMin= 0

YAxis0.RestoreValuesOnResume= 1

YAxis0.MasterUIInput= 0

YAxis0.CartesianStyle= 0

YAxis0.CartesianChildRefAxisName= ""

YAxis0.CartesianChildRefValue= 20

YAxis0.GridLinesVisible= 1

DataView0.Name = "Data View 1"

DataView0.Visible= 1

DataView0.Enabled= 1

DataView0.Layer = 100

DataView0.PopupEnabled= 1

DataView0.Horizontal= 0

DataView0.ZOrder= 0

DataView0.StartPercent= 0

DataView0.StopPercent= 100

DataView0.Title = ""

DataView0.BackgroundTransparent= 1

DataView0.BackgroundColor= 8421376

DataView0.GridXAxisName= "X-Axis 1"

DataView0.GridYAxisName= "Y-Axis 1"

DataView0.GridShow= 1

DataView0.GridLineColor= 32768

DataView0.GridLineShowLeft= 1

DataView0.GridLineShowRight= 1

DataView0.GridLineShowTop= 1

DataView0.GridLineShowBottom= 1

DataView0.GridLineShowXMajors= 1

DataView0.GridLineShowXMinors= 0

DataView0.GridLineShowYMajors= 1

DataView0.GridLineShowYMinors= 0

DataView0.GridLineMajorStyle= 0

DataView0.GridLineMinorStyle= 0

DataView0.GridLineXMajorCustom= 0

DataView0.GridLineXMajorColor= 32768

DataView0.GridLineXMajorWidth= 0

DataView0.GridLineXMajorStyle= 0

DataView0.GridLineXMinorCustom= 0

DataView0.GridLineXMinorColor= 32768

DataView0.GridLineXMinorWidth= 0

DataView0.GridLineXMinorStyle= 0

DataView0.GridLineYMajorCustom= 0

DataView0.GridLineYMajorColor= 32768

DataView0.GridLineYMajorWidth= 0

DataView0.GridLineYMajorStyle= 0

DataView0.GridLineYMinorCustom= 0

DataView0.GridLineYMinorColor= 32768

DataView0.GridLineYMinorWidth= 0

DataView0.GridLineYMinorStyle= 0

Channel0.Name = "Channel 1"

Channel0.Visible= 1

Channel0.Enabled= 1

Channel0.Layer = 100

Channel0.PopupEnabled= 1

Channel0.TitleText= "Channel 1"

Channel0.VisibleInLegend= 1

Channel0.RingBufferSize= 0

Channel0.TraceVisible= 1

Channel0.Color = 255

Channel0.TraceLineStyle= 0

Channel0.TraceLineWidth= 1

Channel0.MarkersAllowIndividual= 0

Channel0.MarkersPenUseChannelColor= 1

Channel0.MarkersBrushUseChannelColor= 1

Channel0.MarkersTurnOffLimit= 0

Channel0.MarkersVisible= 0

Channel0.MarkersSize= 3

Channel0.MarkersStyle= 0

Channel0.MarkersPenColor= 255

Channel0.MarkersPenStyle= 0

Channel0.MarkersPenWidth= 0

Channel0.MarkersBrushColor= 255

Channel0.MarkersBrushStyle= 0

```
Channel0.XAxisName= "X-Axis 1"
Channel0.YAxisName= "Y-Axis 1"
Channel0.XAxisTrackingEnabled= 1
Channel0.YAxisTrackingEnabled= 1
Channel0.LogFileName= ""
Channel0.LogBufferSize= 0
Channel0.DataStyle= 0
Channel0.Tag = 0
Channel0.OPCComputerName= "Local"
Channel0.OPCServerName= ""
Channel0.OPCItemName= ""
Channel0.OPCUpdateRate= 500
Channel0.OPCAutoConnect= 1
Channel0.FastDrawEnabled= 1
Channel0.InterpolationStyle= 0
Channel0.FillEnabled= 0
Channel0.FillReference= 0
Channel0.FillStyle= 0
Channel0.FillColor= 0
Channel0.FillUseChannelColor= 1
Channel0.DigitalEnabled= 0
Channel0.DigitalReferenceStyle= 0
Channel0.DigitalReferenceLow= 10
Channel0.DigitalReferenceHigh= 90
Channel0.BarEnabled= 0
Channel0.BarPenUseChannelColor= 1
Channel0.BarBrushUseChannelColor= 1
```

Channel0.BarReference= 0

Channel0.BarWidth= 5

Channel0.BarPenColor= 255

Channel0.BarPenWidth= 0

Channel0.BarPenStyle= 0

Channel0.BarBrushColor= 255

Channel0.BarBrushStyle= 0

Channel0.OPCXValueSource= 0

DataCursor0.Name= "Cursor 1"

DataCursor0.Visible= 0

DataCursor0.Enabled= 1

DataCursor0.Layer= 100

DataCursor0.PopupEnabled= 1

DataCursor0.ChannelName= "Channel 1"

DataCursor0.ChannelAllowAll= 1

DataCursor0.ChannelShowAllInLegend= 1

DataCursor0.Style= 0

DataCursor0.Font.Charset= 1

DataCursor0.Font.Color= -2147483640

DataCursor0.Font.Height= -11

DataCursor0.Font.Name= "MS Sans Serif"

DataCursor0.Font.Pitch= 0

DataCursor0.Font.Style= 0

DataCursor0.Color= 65535

DataCursor0.UseChannelColor= 1

DataCursor0.HintShow= 1

DataCursor0.HintHideOnRelease= 0

```
DataCursor0.HintOrientationSide= 0
DataCursor0.HintPosition= 50
DataCursor0.Pointer1Position= 50
DataCursor0.Pointer2Position= 60
DataCursor0.PointerPenWidth= 1
DataCursor0.MenuUserCanChangeOptions= 1
DataCursor0.MenuItemVisibleValueXY= 1
DataCursor0.MenuItemVisibleValueX= 1
DataCursor0.MenuItemVisibleValueY= 1
DataCursor0.MenuItemVisibleDeltaX= 1
DataCursor0.MenuItemVisibleDeltaY= 1
DataCursor0.MenuItemVisibleInverseDeltaX= 1
DataCursor0.MenuItemCaptionValueXY= "Value X-Y"
DataCursor0.MenuItemCaptionValueX= "Value X"
DataCursor0.MenuItemCaptionValueY= "Value Y"
DataCursor0.MenuItemCaptionDeltaX= "Period"
DataCursor0.MenuItemCaptionDeltaY= "Peak-Peak"
DataCursor0.MenuItemCaptionInverseDeltaX= "Frequency"
Label0.Name = "Title"
Label0.Visible = 1
Label0.Enabled = 1
Label0.Layer = 100
Label0.PopupEnabled= 1
Label0.Horizontal= 1
Label0.ZOrder = 2
Label0.StartPercent= 0
Label0.StopPercent= 100
```

```
    Label0.MarginLeft=  0
    Label0.MarginTop=  0
    Label0.MarginRight=  0
    Label0.MarginBottom=  0
    Label0.Caption  =  "Untitled"
    Label0.Alignment=  0
    Label0.Font.Charset=  1
    Label0.Font.Color=  16777215
    Label0.Font.Height=  -19
    Label0.Font.Name=  "Arial"
    Label0.Font.Pitch=  0
    Label0.Font.Style=  1
  End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim x As Long
Dim y As Long
Dim XValue As Double
Private Sub clearploter() 'inicializamos el ploter para mostrar nuevos datos
iPlotX.ClearAllData
iPlotX.XAxis(0).DesiredStart = 0
iPlotX.DeleteXAxis (0)
iPlotX.DeleteChannel (0)
```

```
XValue = 0
Call inicializarploter
End Sub
Private Sub Form_Load()
Call inicializarploter  'inicializamos el ploter
pserie.PortOpen = True  'habilitamos el puerto serial
End Sub
Private Sub inicializarploter()
iPlotX.RemoveAllChannels
iPlotX.RemoveAllXAxes
iPlotX.YAxis(0).Span = 255  'seleccionamos medida máxima del eje Y
iPlotX.AddChannel
iPlotX.Channel(0).TitleText = "Corazon"
iPlotX.Channel(0).Color = vbRed
iPlotX.AddXAxis
iPlotX.Channel(0).XAxisName = iPlotX.XAxis(0).Name
iPlotX.XAxis(0).LabelsFontColor = vbRed
End Sub
Private Sub pserie_OnComm()
Dim s As String
s = pserie.Input  'recibimos string del puerto serial
If s = "ind" Then 'trama "ind" nos indica comienzo de transmisión
 Call clearploter 'inicializamos el ploter para mostrar nuevos datos
 pserie.Output = Chr(9) 'envia señal al uC para pedir dato nuevo
Else
 y = Val(s)    'convierte cadena de caracteres a valor numérico
 dibuja        'dibuja en el ploter el dato adquirido
```

```
End If
End Sub
Private Sub dibuja()
iPlotX.Channel(0).AddXY XValue, y
XValue = XValue + 1     'incrementa el  X-Value del ploter
pserie.Output = Chr(9)  'envía señal al uC para pedir dato nuevo
End Sub
```

4.2 Programa del microcontrolador.

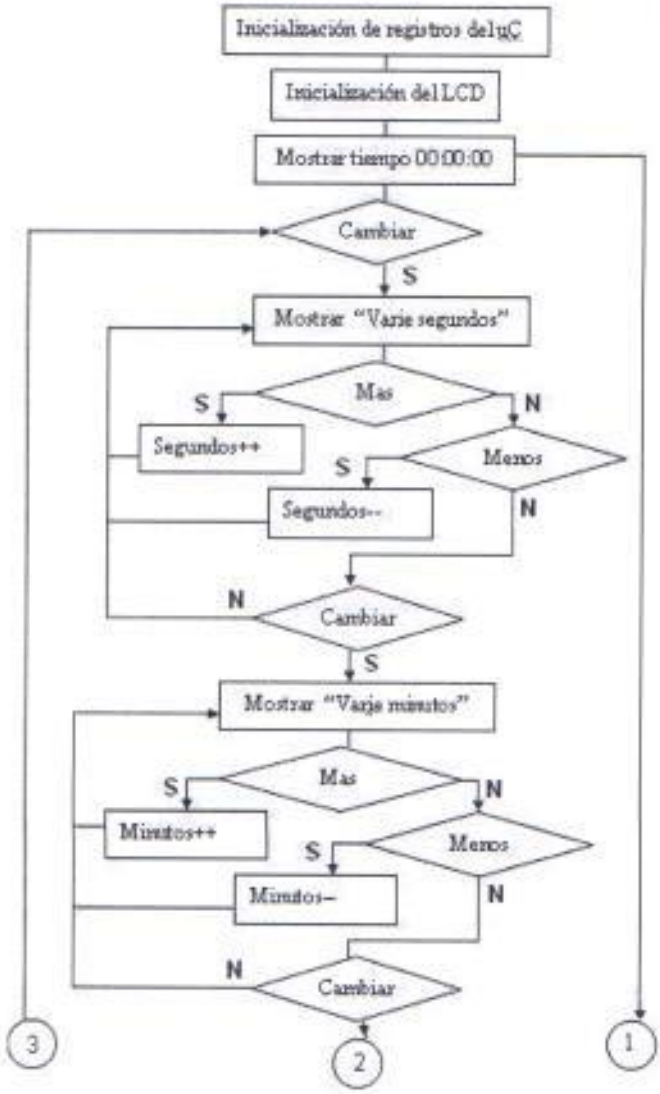


Fig. 4.2.- Organigrama del programa del microcontrolador

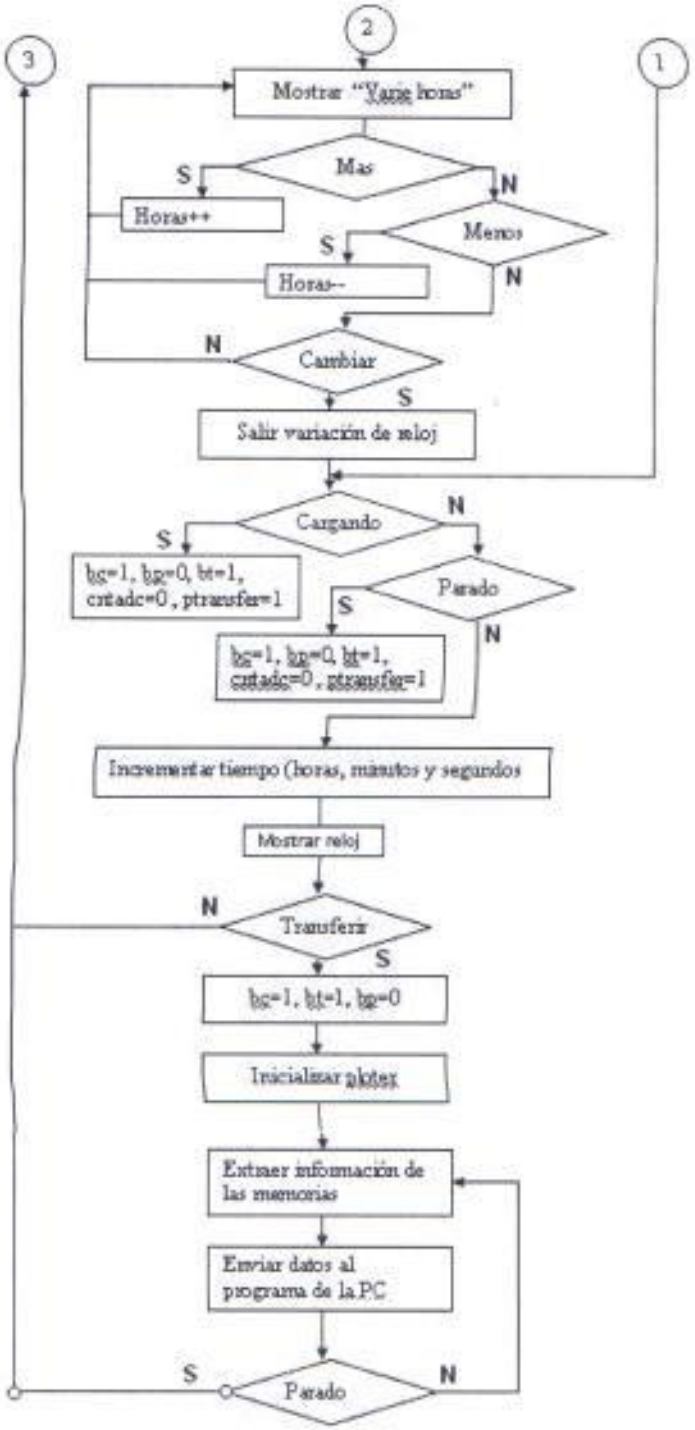


Fig. 4.3.- Continuación de organigrama del programa del microcontrolador

Como se puede apreciar en el diagrama, el programa principal del microcontrolador empieza inicializando los valores de los registros del microcontrolador, esto implica definir cuales son las entradas y las salidas,

configurar los registros, habilitar las interrupciones, configuración del convertidor analógico digital e inicializar el LCD.

Después muestra el tiempo cero, es decir, cero horas, cero minutos y cero segundos. Luego permite la configuración del tiempo, alterando las horas, minutos y segundos, esta parte del programa fue elaborada para que el paciente ingrese la hora exacta.

Con la utilización de las teclas "Cambiar", "Más" , "Menos" , "Cargando", "Parado" y "Transferir" se introduce el valor de estas entradas al programa , en Main.h se indica los pines del integrado que recibirán los niveles de voltajes de las entradas anteriormente mencionadas.

La tecla "Cambiar" permite la selección del parámetro que se va a modificar en el tiempo, presionando una vez se selecciona los segundos, presionando otra vez se selecciona minutos, si se presiona una tercera vez se selecciona horas y al presionar una cuarta vez se sale de la configuración del tiempo. Las teclas "Más" y "Menos" permiten aumentar o disminuir los valores de los parámetros que se han seleccionado.

Al seleccionar "Cargando" se realiza la grabación del dato obtenido del conversor analógico digital en las memorias, con la ayuda de un contador se define en cual memoria se realizará la grabación del dato, para más detalles del programa se puede revisar el Apéndice F.

Al seleccionar "Transferir" se utiliza un lazo for que empieza en cero y termina en el valor que haya alcanzado el acumulador que registra el almacenamiento de datos, este permite saber la cantidad máxima que se ha utilizado de las memorias, para obtener sólo los datos almacenados.

CONCLUSIONES Y RECOMENDACIONES

La medicina moderna requiere el uso de equipos de alta tecnología para el diagnóstico y tratamiento de ciertas enfermedades, debido a los elevados costos de esta clase de equipos, muy pocos profesionales cuentan con este tipo de instrumentos, por lo que existía la necesidad de desarrollar un equipo de bajo costo para que sea accesible.

Al finalizar este proyecto, se obtuvo un instrumento que cumple con los objetivos propuestos al ser simple en su diseño, confiable y de bajo costo en comparación con los miles de dólares que cuesta un equipo del mercado con similares características.

El ruido es una de las principales consideraciones que se deben tomar en cuenta cuando se va a diseñar un instrumento de precisión como el desarrollado. El Holter debe tener un alto rechazo al ruido, esto se obtiene utilizando un amplificador de instrumentación de buena calidad, que rechaza las señales comunes a ambas entradas, en este caso el ruido. Además se han diseñado un filtro para eliminar señales de frecuencia de 60 Hz, lo que al final ha permitido obtener una señal de muy buena calidad.

La capacidad de visualizar las imágenes desde cualquier computador que trabaje con el sistema operativo Windows, actualmente el más difundido, permite ver las imágenes en la mayoría de computadores conectando el Holter Digital al puerto serial. La interfaz gráfica es otra de las ventajas de este equipo debido a su sencillez, lo que facilita su manejo.

El modelo desarrollado puede servir de referencia para desarrollar otros equipos con características adicionales como podrían ser:

1. Comunicación inalámbrica con el computador, que constituye una tendencia de la tecnología, ya sea implementándola a través de bluetooth, puertos infrarrojos o mediante el estándar 802.11.
2. Añadir nuevos canales para poder estudiar nuevas derivaciones, lo que permitiría brindarle mayor información al médico para realizar su diagnóstico.
3. Almacenar las imágenes en un archivo en el computador, de esta manera el doctor podrá guardar la información de un paciente y realizar comparaciones con mediciones pasadas o las que se realicen a futuro, así como la posibilidad de enviar la información a un colega.
4. Mejorar el programa utilizado en el computador para que ayude a detectar las afecciones cardiacas, mediante el análisis de los patrones de las ondas, para que al detectar una alteración pueda relacionarla con una enfermedad y de esta manera ayudar en el diagnóstico.

APÉNDICES

APÉNDICE A

MANUAL DEL USUARIO

Se deberá colocar los electrodos en cada muñeca: en la derecha a la entrada V_{RA}, en la izquierda la entrada V_{LA} y en el pie derecho V_{RL}. Como se ilustra en la figura A.1.

Una vez instalados los electrodos se procederá a presionar el botón llamado "CAPTURAR", esto comenzará el proceso de grabación en las memorias.

Una vez que se haya capturado la información se debe presionar la tecla "PARADO" para detener el proceso de grabación.

Para mostrar las señales por el computador se debe abrir el programa de graficación y conectar el puerto serial del Holter al puerto serial COM1 del computador.. Para ejecutar el programa, en el menú Inicio, se debe pulsar "Proyecto1".

Posterior a eso se debe presionar el botón "TRANSFERIR" para que la señal sea mostrada en el computador.

Si se quiere detener el envío de la información debemos presionar el botón "PARADO" en el Holter.

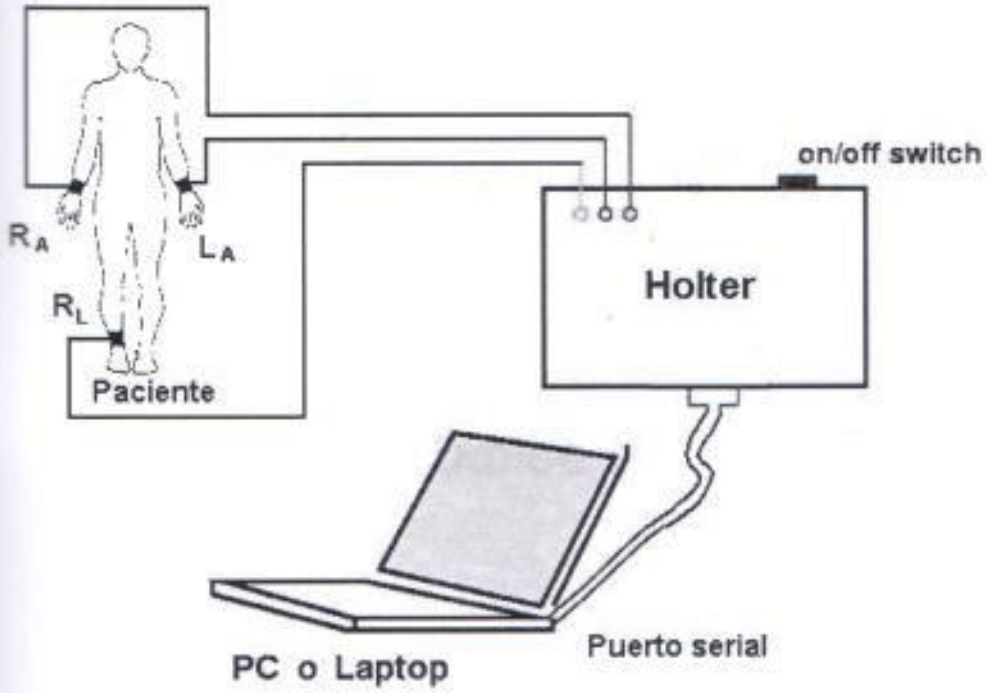
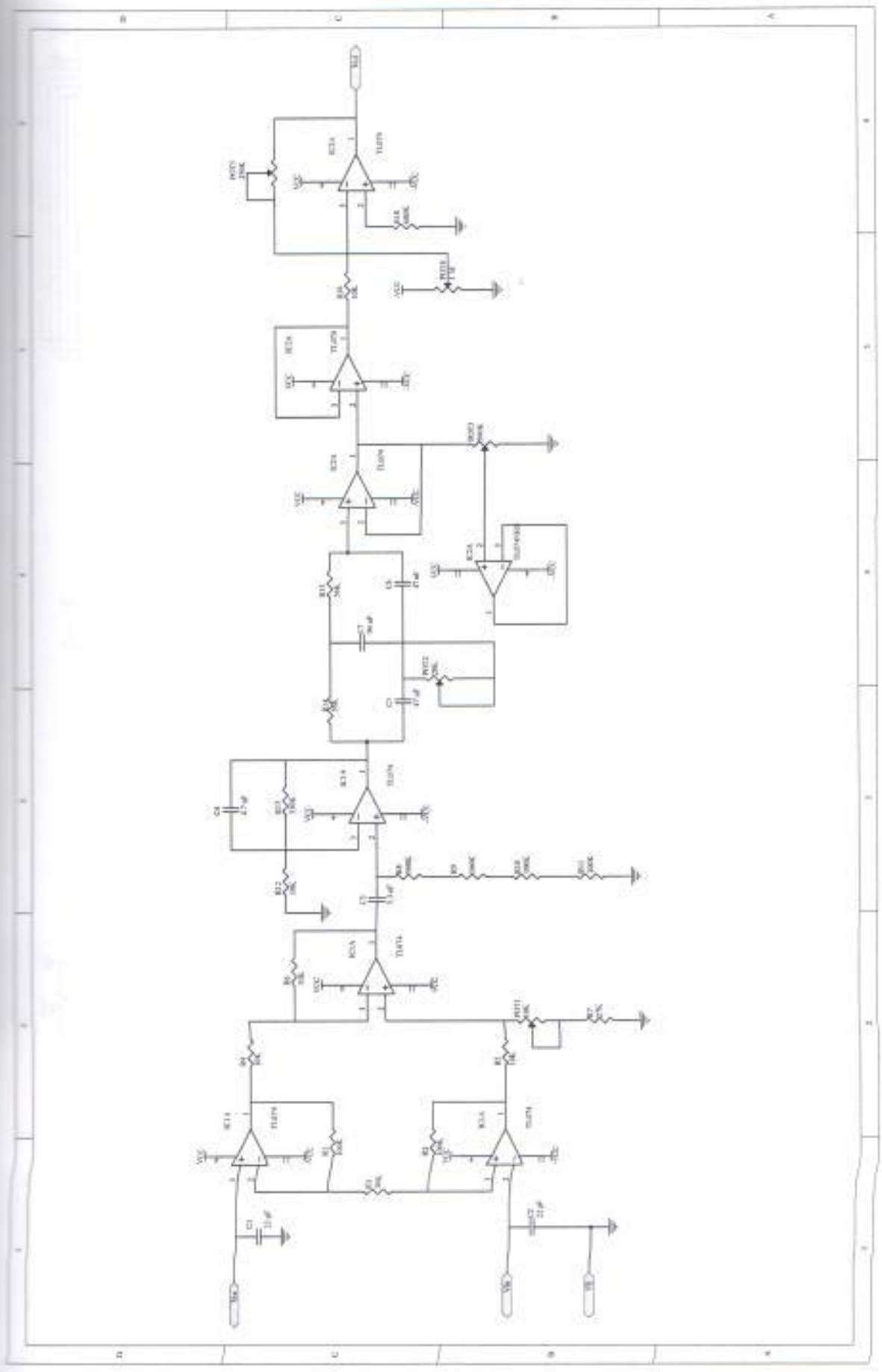
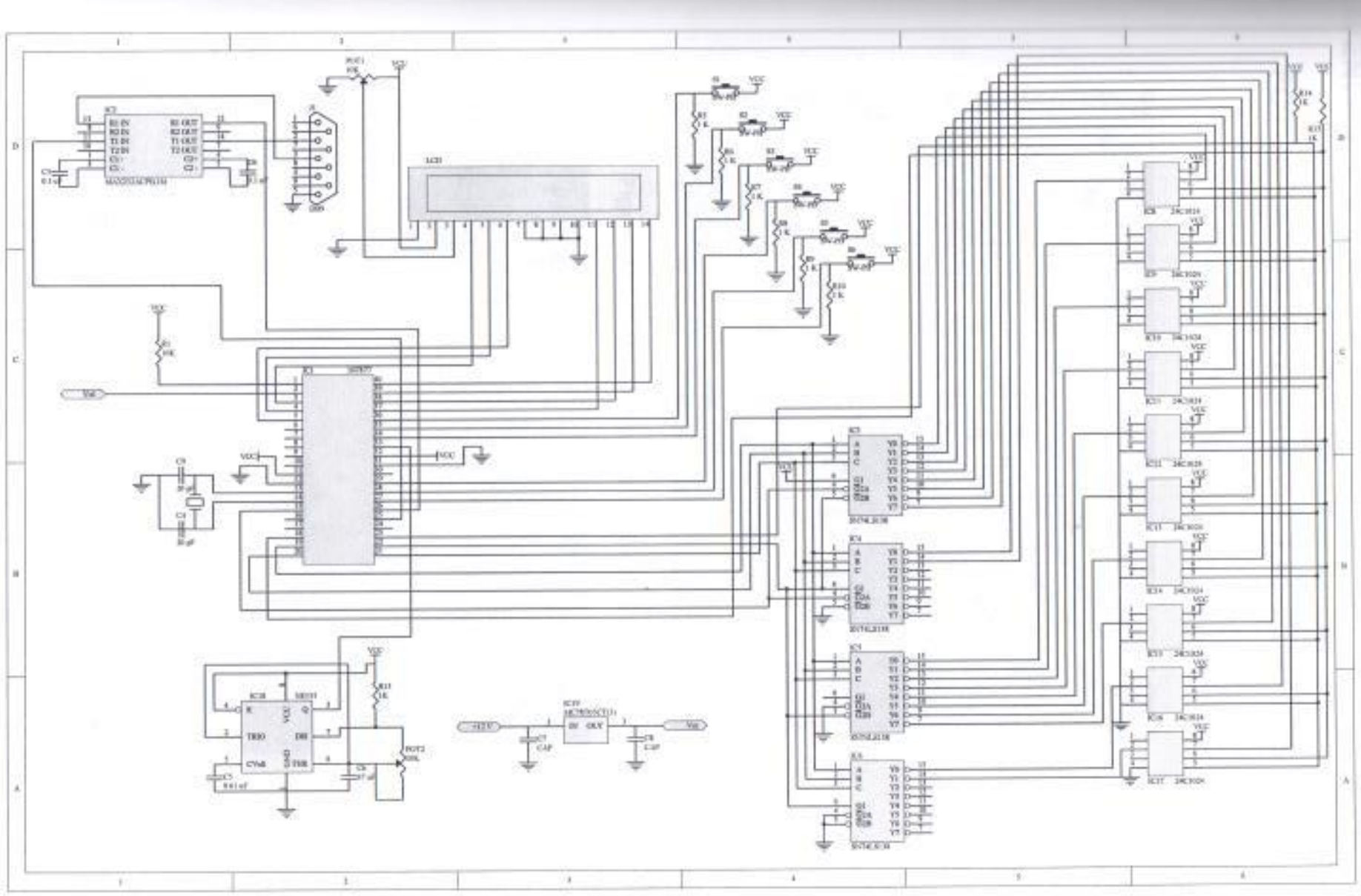


Figura A.1 Conexiones del Holter entre computador y paciente

APÉNDICE B

DIAGRAMAS DE LOS CIRCUITOS DISEÑADOS





APENDICE C

FOTOGRAFÍAS DEL EQUIPO Y LAS SEÑALES OBTENIDAS

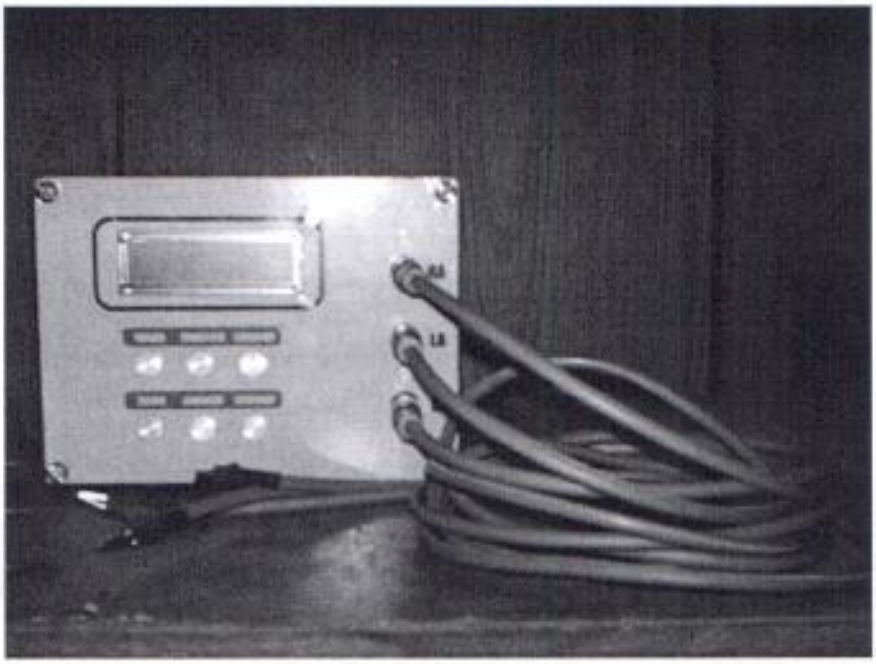


Figura C.1.- Imagen del prototipo desarrollado

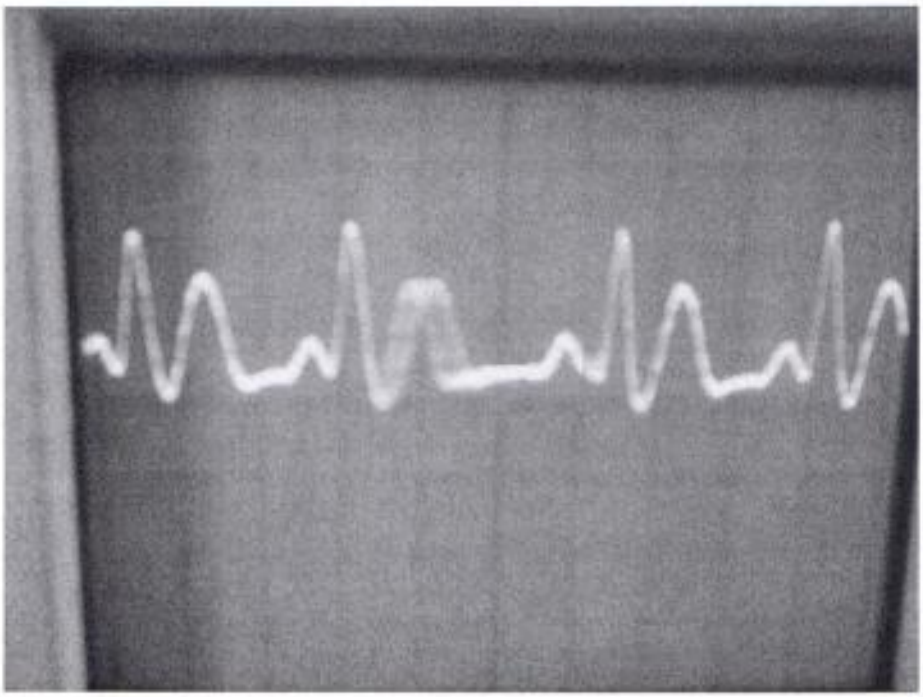


Figura C.2.- Señal cardiaca en el osciloscopio

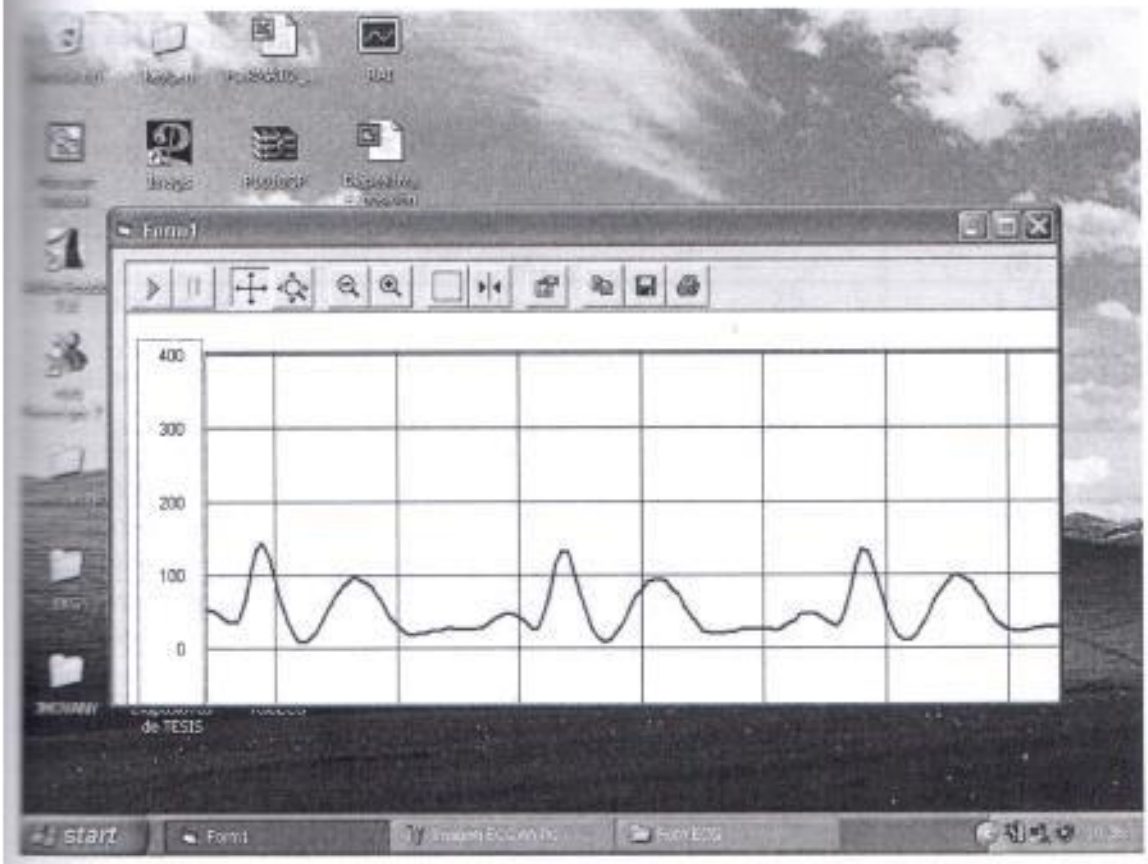


Figura C.3.- Pantalla del software del Holter Digital.

APÉNDICE D

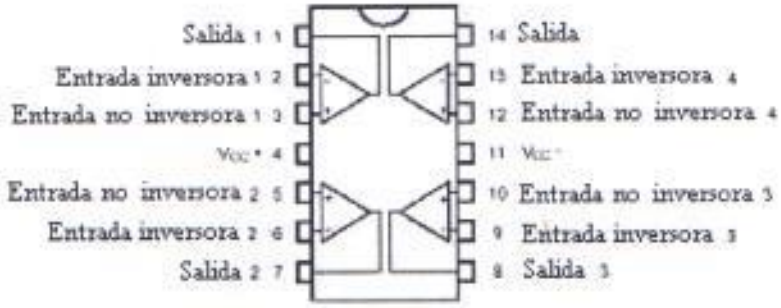
ANÁLISIS DE COSTOS

COMPONENTES	VALOR UNITARIO ($)	CANTIDAD	VALOR TOTAL ($)
Resistencias de 1/ 4 W ,10 %	0.05	25	1.25
Resistencias de 1/ 2 W ,10 %	0.05	8	0.40
Potenciometros de 0.5 W	0.35	7	2.45
Condensadores Electrolíticos	0.09	5	0.45
Condensadores Cerámicos	0.08	9	0.72
Baterias 9 V	1.00	2	2.00
Cajas metálicas	25.00	1	25.00
Placas	40		40.00
Regulador de Voltaje	0.36	1	0.36
Electrodos descartables	0.60	3	1.80
Circuito Integrado TL074	0.63	2	1.26
PIC 16F877	12.00	1	12.00
Circuito integrado LM555	0.50	1	0.50
Pantalla LCD	7.00	1	7.00
Oscilador de cuarzo	0.50	1	0.50
Memorias	5.00	10	50.00
Circuito Integrado MAX232	1.00	1	1.00
Conector DB9 hembra	0.35	1	0.35
Conector DB9 macho	0.35	1	0.35
Cables	1.00	3	3.00
Rollo de estaño	0.50	1	0.50
Cautin	3.00	1	3.00
TOTAL			150.39

APÉNDICE E

HOJA DE DATOS DE LOS CIRCUITOS INTEGRADOS

AMPLIFICADOR OPERACIONAL CUADRUPLE TL074



Descripción

Paquete integrado de cuatro amplificadores operacionales de bajo consumo y bajo ruido, de tecnología JFET

Características

Bajo ruido

Alta impedancia de entrada

Baja corriente de entrada offset y de polarización

Protección contra cortocircuitos

Parámetros

Símbolos	Parámetros	TL074	Unidades
Vcc	Voltaje de Alimentación	18	V
Vin	Voltaje de entrada	15	V
Ptot	Potencia de Salida	680	mW
tsc	Salida de corto	Infinito	
Toper	Temperatura de operación	-40 a 105	° C

TEMPORIZADOR LM555

Descripción General

Es un controlador muy estable capaz de producir retardos en el tiempo precisos u oscilaciones. Incluye terminales adicionales para disparar o restablecer si se desea. En el modo de operación de retraso, el tiempo se controla precisamente por medio de un resistor y un capacitor externo. Para una operación estable como oscilador, la frecuencia de oscilación libre y el ciclo de trabajo se controlan con precisión con dos resistores externos y un capacitor. El circuito puede dispararse y restablecerse en forma de ondas decrecientes, y la estructura de salida puede producir o absorver hasta 200 mA o manejar circuitos TTL.

Características

Temporización en microsegundos hasta horas

Opera en modo monoestable como astable

Ciclo de trabajo ajustable

La salida de alta corriente puede alimentar o tomar 200 mA

La salida puede impulsar TTL

Estabilidad de temperatura de 0.005% por °C

Aplicaciones

Temporización precisa

Generación de pulso

Temporización secuencial

Generación de retraso

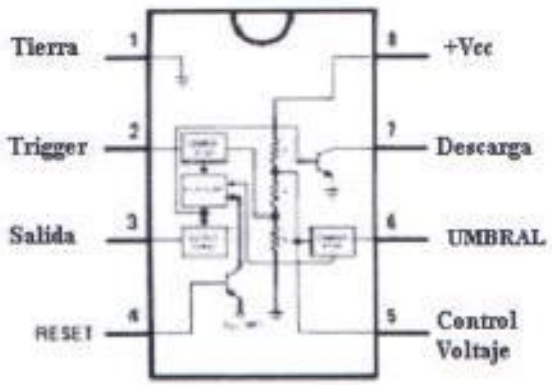
Modulación de ancho de pulso

Modulación de posición de pulso

Valores Máximos absolutos

Voltaje de alimentación	+18V, 600mW
Rango de temperatura de operación	0°C a 70°C
Rango de termperatura de almacenamiento	65°C a 150°C
Temperatura de la soldadura en la Terminal, 60 seg.	300°C

Diagrama



REGULADOR LM7805

El regulador de voltaje positivo de 3 terminales LM7805 emplea limitación interna de corriente, apagado térmico haciéndolo esencialmente indestructible. Esfuerzos considerables se han hecho para que este regulador sea fácil de usar, reduciendo el uso de componentes externos.

Características

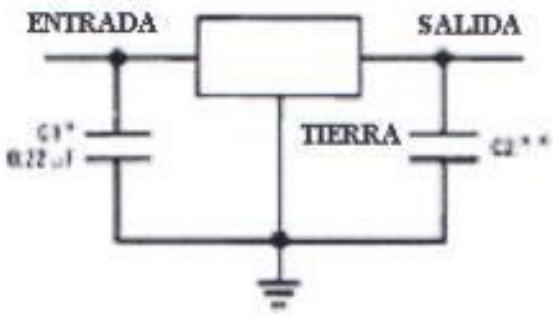
Protección interna para el exceso de temperatura.

Limitación interna de corriente de corto circuito.

Valores Máximos absolutos

Voltaje de alimentación	35V
Rango de temperatura de operación	150°C
Rango de termperatura de almacenamiento	-65°C a 150°C
Temperatura de la soldadura en la Terminal, 60 seg.	300°C

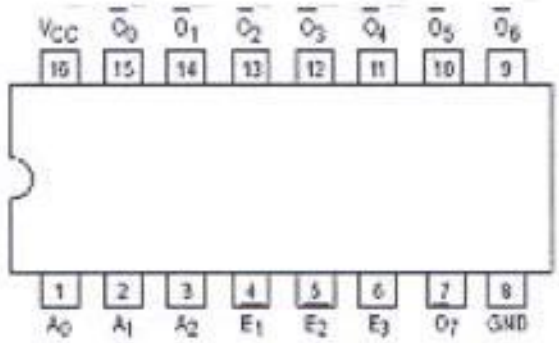
Aplicación típica



MULTIPLEXOR 74138

Es un multiplexor 1 a 8, ideal para memorias bipolares de alta velocidad.

Acepta tres entradas (Ao, A1, A2) y ocho salidas mutuamente exclusivas (A0-A7)



A0-A2	Selector de salida
E1. E2	Habilitador entrada
E3	Habilitador entrada
O0-O7	Salida

Rangos de operación garantizados

Parametro	Min	Typ	Max	Unit
Voltaje de entrada	4.5 4.75	5.0 5.0	5.5 5.25	V
Rango de temperatura de operación ambiente	-55 0	25 25	125 70	°C
Corriente de salida-Alta			-0.4	mA
Corriente de salida-Baja			4.0 8.0	mA

INTEGRADO MAX232

Es un receptor dual que incluye un generador de voltaje capacitivo que proveerá de niveles de voltaje acorde a la norma RS-232 . Estos receptores tienen un umbral tIPICo de 1.3 V y pueden aceptar hasta 30 V de entrada.

Características

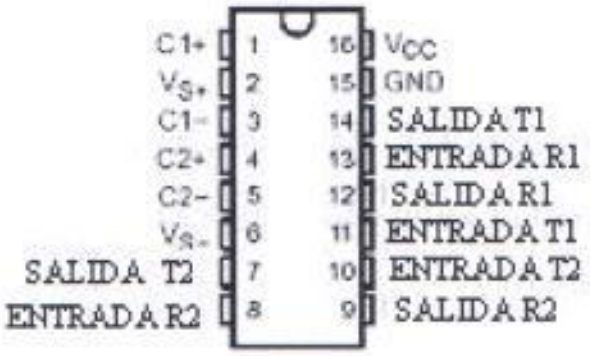
Opera hasta a 120 kbits/s

Aplicaciones en modems, terminales y computadoras

Cumple y excede recomendaciones TIA/EIA-232-F e ITU

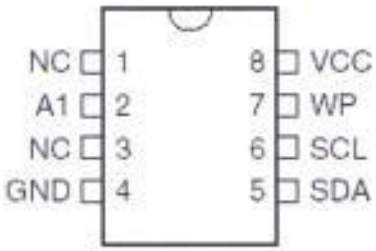
Condiciones de operación recomendadas

	Min	Nominal	Máx	Unit
Fuente de voltaje	4.5	5	5.5	V
Voltaje de entrada máxima	2			V
Voltaje de entrada mínimo			0.8	V
Temperatura de operación	0		70	°C



MEMORIA AT24C1024

Provee 1048576 bits de memoria serial de solo lectura que puede ser eléctricamente borrada y programada, organizada como 131072 palabras de 8 bits. Optimo para uso en aplicaciones industriales y comerciales donde se requiere bajo consumo de potencia.



Configuraciones de pines

CONFIGURACION	
NOMBRE	**FUNCIÓN**
A1	Entrada de dirección
SDA	Datos seriales
SCL	Entrada de reloj
WP	Protección contra escritura
NC	No conectado

Descripción de pines

SCL.- La entrada SCL es usada para el reloj del integrado, trabajando con flancos positivos para el ingreso de datos y flancos negativos para la salida de datos.

SDA.- El pin SDA es usado para la transferencia bidireccional de información. Este pin es colector abierto, esto permite que las memorias puedan ligarse para formar un bus común. En cualquier momento, todas las salidas de compuerta ligada al bus, excepto una, deben mantenerse

en su estado alto o bajo, dependiendo de si se desea transmitir un 1 o un 0 en el bus.

A1.- Usado para habilitar a la memoria, de gran ayuda para seleccionar una memoria entre un grupo de esos integrados, usa lógica negativa.

WP.- Este pin es útil para proteger el contenido completo de la memoria de operaciones inadvertidas de escritura. La escritura se habilita cuando recibe cero voltios (lógica negativa) y se inhabilita cuando recibe cinco voltios.

Características:

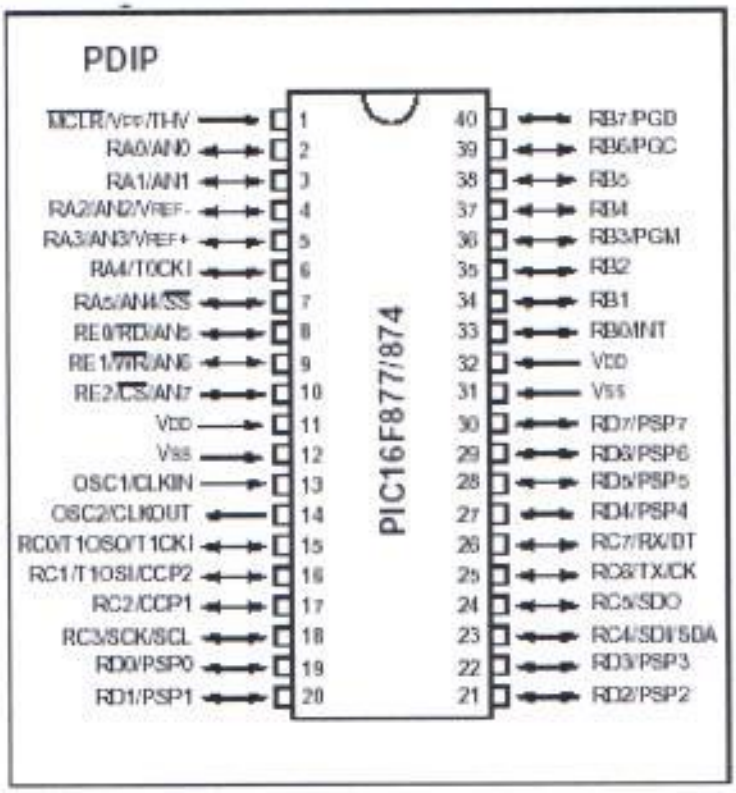
Bajo voltaje de operación (Vcc= 2.7 a 5.5V)

Protocolo de transferencia de datos bidireccional

Pin para protección de escritura para protección de datos.

Modo de lectura aleatorio y secuencial.

PIC 16F877/874



CPU RISC de alto desempeño

Solo 35 palabras de instrucción para aprender

Opción de oscilador seleccionable.

Tecnología Flash/EEPROM de alta velocidad y baja potencia.

Amplio rango de voltaje de opeación (2 a 5.5 V)

Convertidor analógico a digital.

Características

Frecuencia de operación	DC-20MHz
Memoria de datos (bytes)	192
Memoria de datos EEPROM	128K
Comunicación serial	USART

APÉNDICE F

CÓDIGO DEL PROGRAMA DEL MICROCONTROLADOR

```
#include <pic.h>
#include "display.h"
#include "main.h"
#include "delay.h"
#include "lcd.h"

#define nesclavow 0xa0
#define nesclavor 0xa1

const char *msgc =   "CARGANDO      ";
const char *msgt =   "TRANSFIRIENDO  ";
const char *msgmll = "MEMORIA LLENA  ";
const char *msgp =   "PARADO        ";
const char *msgtc =  "COMPLETADO     ";
const char *msgv =   "               ";
const char *msgs =   "VARIE SEGUNDOS ";
const char *msgm =   "VARIE MINUTOS  ";
const char *msgh =   "VARIE HORAS    ";

/////reloj
char hora, minuto, segundo;
bit  bsl;
int  cntms;
///i2c
 int cntadc;
///
bit bc, bp, bt, tin, ptranfer;
char dator;

//iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
void interruptisr(void)
{
   unsigned long var0;
  //int timer 2


    if((INTF)&&(INTE)){
      bsl = 1;
      INTF = 0;
                  }
```

```
if((ADIF)&&(ADIE)) {
 ///interrupción del convertidor análogo digital
 ///Carga en su respectiva memoria
  if(cntadc <= 0x0000ffff && cntadc >= 0x00000000 ){
  PORTD = 0;
  sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0001ffff && cntadc >= 0x00010000 ){
   PORTD = 1;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0002ffff && cntadc >= 0x00020000 ){
   PORTD = 2;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0003ffff && cntadc >= 0x00030000 ){
   PORTD = 3;
  sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0004ffff && cntadc >= 0x00040000 ){
   PORTD = 4;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0005ffff && cntadc >= 0x00050000 ){
   PORTD = 5;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0006ffff && cntadc >= 0x00060000 ){
   PORTD = 6;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0007ffff && cntadc >= 0x00070000 ){
   PORTD = 7;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0008ffff && cntadc >= 0x00080000 ){
   PORTD = 8;
   sendI2C(ADRESH, cntadc);
                                    }
 if(cntadc <= 0x0009ffff && cntadc >= 0x00090000 ){
   PORTD = 9;
   sendI2C(ADRESH, cntadc);
                                    }
///contador de registros de la memorias
 if(cntadc >= 0x000a0000)
    tin =1;
    cntadc++;

    ADIF = 0;
```

```
        }
}

//////////////////
i2csssssssi2csssssi2csssssssssssssssi2csssssss
void sendI2C(char dato, int direccion)
{
  char dmsb, dlsb;
  WP = 0;
  dmsb = direccion;
  direccion >>= 8;
  dlsb = direccion;

  ///start

  SSPIF = 0;
  SEN = 1;
  while(!SSPIF);
  ///envio nombre
  SSPIF = 0;
  SSPBUF = nesclavow;
  while(!SSPIF);
  ///direccionhigh
  SSPIF = 0;
  SSPBUF = dlsb;
  while(!SSPIF);
  ///direccionlow
  SSPIF = 0;
  SSPBUF = dmsb;
  while(!SSPIF);
  ///mandar dato;
  SSPIF = 0;
  SSPBUF = dato;
  while(!SSPIF);

   //stop
  SSPIF = 0;
  PEN = 1;
  while(!SSPIF);
  WP = 1;


}
////
char backI2C(int direccion)
{
   char dmsb, dlsb;
  dmsb = direccion;
```

```
direccion >>= 8;
dlsb = direccion;

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

//start
 SSPIF = 0;
 SEN = 1;
 while(!SSPIF);
 ///envio nombre
 SSPIF = 0;
 SSPBUF = nesclavow;
 while(!SSPIF);
 ///direccionhight
 SSPIF = 0;
 SSPBUF = dlsb;
 while(!SSPIF);
 ///direccionlow
 SSPIF = 0;
 SSPBUF = dmsb;
 while(!SSPIF);
 ///mandar dato;
 SSPIF = 0;
 PEN = 1;
 while(!SSPIF);

//-----------------------------------------
 ///start
 SSPIF = 0;
 SEN = 1;
 while(!SSPIF);

 ///envio para lectura
 SSPIF = 0;
 SSPBUF = nesclavor;
 while(!SSPIF);


  ///recibir dato;
  SSPIF = 0;
  RCEN = 1;
   while(!SSPIF);
   //stop
  SSPIF = 0;
  PEN = 1;
  while(!SSPIF);
  ///
 return SSPBUF;
```

```
}
//ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
void seteo(void)
{
  //OPTION = 0x02; //prt 8 -> 2uS
  //configuro timer0
  PS2 = 1;
  PS1 = 0;
  PS0 = 0;
  PSA = 0;
  T0CS = 0;
 //configuro disparo automatico
  CCP2CON = 0xfb;
  CCPR2H = 0xC3;
  CCPR2L = 0xff;
  T1CON = 0x00;
///configuracion de registros
  T2CON = 0x05;
  ADCON1 = 0x0e;
  ADCON0 = 0xc0;
  TRISA = 0xf1;
  TRISB = 0x0f;
  PORTB = 0x00;
  TRISC = 0xbe;
  PORTC = 0x00;
  TRISD = 0xf0;
  PORTD = 0x80;
  TRISE = 0x00;
  PORTE = 0x00;
 //I2C
  SSPSTAT = 0x00;  //400khz
  SSPADD = 0x0b;  //11
  //habilito transmision
  TXSTA = 0x24;
  RCSTA = 0x90;
  BRGH = 1;
  SPBRG = 129;
  SSPCON = 0x28;
 ///reseteo variables
 PORTD = 255;
 bc = 0;
 bp = 0;
 bt = 0;
 cntadc = 0;
 tin = 0;
 ptranfer = 0;
  //habilito interrupciones
  INTE = 1;

 // T0IE = 1;   //cada 2mS
```

```
//configuracion de convertidor analogico digital
ADIF = 0;
ADIE = 1;
RBIF = 0;
RBIE = 1;
GIE = 1;
PEIE = 1;


}
void reloj(void)
{
char var0, var1;
////contador de registros de reloj
segundo++;
if(segundo == 60){
 segundo = 0;
 minuto++;
 if(minuto == 60){
   minuto = 0;
   hora++;
  if(hora == 23){
    hora = 0;
          }

          }


          }
///mostrar en lcd el reloj
lcd_goto(3);
var0 = hora/10;              //obtiene el primer digito de la hora
var1 = var0+0x30;            //convierte el número a código ASCII(hex)
lcd_putch(var1);             //coloca el resultado en el LCD
var0 = hora - 10*var0;       //obtiene el segundo digito de la hora
var1 = var0+0x30;            //convierte el número a código ASCII(hex)
lcd_putch(var1);             //coloca el resultado en el LCD
lcd_putch(':');              //coloca : en el LCD
var0 = minuto/10;             //obtiene el primer digito de los  minutos
var1 = var0+0x30;             //convierte el número a código ASCII(hex)
lcd_putch(var1);              //coloca el resultado en el LCD
var0 = minuto - 10*var0;     //obtiene el segundo digito de los minutos
var1 = var0+0x30;            //convierte el número a código ASCII(hex)
lcd_putch(var1);             //coloca el resultado en el LCD

lcd_putch(':');                 //coloca : en el LCD
var0 = segundo/10;              //obtiene el primer digito de los segundos
var1 = var0+0x30;               //convierte el número a código ASCII(hex)
lcd_putch(var1);                //coloca el resultado en el LCD
```

```
  var0 = segundo - 10*var0;      //obtiene el segundo digito de los segundos
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD

}
drdrdrdrdrdrdrdrdrdrdrdrdrdrdrdrdrdrdr
id dreloj(void)
{
  char var0, var1;
  ///mostrar en lcd el reloj
  lcd_goto(3);
  var0 = hora/10;                //obtiene el primer digito de la hora
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD
  var0 = hora - 10*var0;         //obtiene el segundo digito de la hora
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD
  lcd_putch(':');                //coloca : en el LCD
  var0 = minuto/10;              //obtiene el primer digito de los minutos
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD
  var0 = minuto - 10*var0;       //obtiene el segundo digito de los minutos
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD

  lcd_putch(':');                //coloca : en el LCD
  var0 = segundo/10;             //obtiene el primer digito de los segundos
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD
  var0 = segundo - 10*var0;      //obtiene el segundo digito de los segundos
  var1 = var0+0x30;              //convierte el número a código ASCII(hex)
  lcd_putch(var1);               //coloca el resultado en el LCD

}
////mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
void main(void)
{
    int i;
    char j;
    seteo();                     ///cargo registros de configuración del uC
//inicio lcd
   lcd_init();
  // reloj
   ///mostrar en lcd el reloj tiempo 00:00:00

  lcd_goto(3);
  lcd_putch('0');
  lcd_putch('0');
  lcd_putch(':');
  lcd_putch('0');
```

```
lcd_putch('0');
lcd_putch(':');
lcd_putch('0');
lcd_putch('0');
 TMR1ON = 1;
 while(1){
////////////////////change
 if(cambiar){                    //variamos el reloj
  DelayMs(20);
  while(cambiar);
  //variamos los segundos
   lcd_goto(64);
   lcd_puts(msgs);               //muestro mensaje VARIE SEGUNDOS
  DelayMs(20);
  while(1){
   if(mas){
    segundo++;                   //se presionó tecla de aumento
    if(segundo == 60)
      segundo = 60;
    dreloj();                    //mostrar en LCD el tiempo
    while(mas);

        }
   if(menos){                    //se presionó tecla de disminución
    segundo--;
    if(segundo > 60)
      segundo = 0;
    dreloj();                    //mostrar en LCD el tiempo
    while(menos);

        }
    if(cambiar){                 //se presionó tecla de cambio
     DelayMs(20);
     while(cambiar);
     DelayMs(20);
     break;
           }
        }
  DelayMs(20);
  //variamos los minutos
   lcd_goto(64);
   lcd_puts(msgm);
  while(cambiar);                //aseguro que este pulsado la tecla
  while(1){
   if(mas){                      //se presionó tecla de aumento
    minuto++;
    if(minuto == 60)
      minuto = 60;
      dreloj();
    while(mas);
```

```
      }
 if(menos){                    //se presionó tecla de disminución
  minuto--;
 if(minuto > 60)
    minuto = 0;
    dreloj();
 while(menos);

      }
  if(cambiar){                 //se presionó tecla de cambio
   DelayMs(20);
   while(cambiar);
   DelayMs(20);
   break;
         }
      }
DelayMs(20);
//variamos las horas
 lcd_goto(64);
 lcd_puts(msgh);
while(cambiar);
while(1){
 if(mas){                      //se presionó tecla de aumento
  hora++;
 if(hora == 11)
    minuto = 11;
    dreloj();
 while(mas);

      }
 if(menos){                    //se presionó tecla de disminución
  hora--;
 if(hora > 11)
    hora = 0;
    dreloj();
 while(menos);

      }
 if(cambiar){                  //se presionó tecla de cambio
  DelayMs(20);
  while(cambiar);
  DelayMs(20);
  break;
         }
      }
  //salimos de variacion del reloj
  lcd_goto(64);
  lcd_puts(msgv);
```

```
        }



if(cargando == 1 && !bc){              //pregunta por cargar
  bc = 1;
  bp = 0;
  bt = 1;
  lcd_goto(64);
  lcd_puts(msgc);                      //muestro en lcd mensaje CARGANDO
  cntadc = 0;
  ADON = 1;
  ptranfer = 1;
          }
 if(parar == 1 && !bp){                //pregunta por parar
  ptranfer = 1;
  ADON = 0;
  tin = 0;
  bc = 0;
  bt = 0;
  bp = 1;
  lcd_goto(64);
  lcd_puts(msgp);                      //muestro en lcd mensaje PARADO
          }

  if(bsl){
    bsl = 0;
    reloj();
      }


// cntadc = 0x0009ffff;
  if(tin){
   ADON = 0;
   tin = 0;
   bc = 0;
   bt = 0;
   bp = 1;
   lcd_puts(msgmll);                   //muestro en lcd mensaje MEMORIA LLENA
       }

 if(transferir == 1 && !bt){           //pregunto por transferencia
   ADON = 0;
   bc = 1;
   bt = 1;
   bp = 0;
   lcd_goto(64);                       // lcd_goto(74);
   lcd_puts(msgt);                     //muestro en lcd mensaje TRANSFIRIENDO
```

```
//inicializamos el ploter
 if(ptranfer){
  ptranfer = 0;
  TXREG = 'i';                    //mando caracter a visual basic
  while(!TXIF);
  TXREG = 'n';
  while(!TXIF);
  TXREG = 'd';
  while(!TXIF);
  while(!RCIF){
   if(parar == 1)
     break;
        }
  if(RCREG > 0)
    RCIF = 0;
       }
 //Tomo datos de las memorias por I2C
 for(i=0; i<=cntadc;i++){
  if(i <= 0x0000ffff && i >= 0x00000000 ){
   PORTD = 0;
    dator = backI2C(i);
                                    }
  if(i <= 0x0001ffff && i >= 0x00010000 ){
   PORTD = 1;
   dator = backI2C(i);
                                    }
  if(i <= 0x0002ffff && i >= 0x00020000 ){
    PORTD = 2;
     dator = backI2C(i);
                                    }
  if(i <= 0x0003ffff && i >= 0x00030000 ){
    PORTD = 3;
    dator = backI2C(i);
                                    }
  if(i <= 0x0004ffff && i >= 0x00040000 ){
    PORTD = 4;
    dator = backI2C(i);
                                    }
  if(i <= 0x0005ffff && i >= 0x00050000 ){
   PORTD = 5;
   dator = backI2C(i);
                                    }
  if(i <= 0x0006ffff && i >= 0x00060000 ){
   PORTD = 6;
   dator = backI2C(i);
                                    }
  if(i <= 0x0007ffff && i >= 0x00070000 ){
   PORTD = 7;
   dator = backI2C(i);
```

```
                                    }
if(i <= 0x0008ffff && i >= 0x00080000 ){
 PORTD = 8;
 dator = backI2C(i);
                                    }
if(i <= 0x0009ffff && i >= 0x00090000 ){
 PORTD = 9;
 dator = backI2C(i);
                                }
///Transformamos el dato a bcd para ser enviado
 binbcd16d(dator);
 TXREG = mostrard3;          //datos enviados a visual basic
 while(!TXIF);
 TXREG = mostrard2;
 while(!TXIF);
 TXREG = mostrard1;
 while(!TXIF);
  //espero resepcion
  while(!RCIF){
    if(bsl){
     bsl = 0;
     reloj();
        }
   if(parar == 1)
     break;
        }
  if(RCREG > 0)
  RCIF = 0;
   if(parar == 1){
     lcd_goto(64);
     lcd_puts(msgp);          // muestro mensaje PARADO
      break;
          }

   if(bsl){
     bsl = 0;                 //muestro reloj
     reloj();
       }
            }
if(parar == 0){               //tecla de parar
 lcd_goto(64);
 lcd_puts(msgtc);             // muestro mensaje CARGANDO
        }
while(transferir);
      }
  }
}
```

DELAY.C

```
*
*       Delay functions
*       See delay.h for details
*
*       Make sure this code is compiled with full optimization!!!
*/

#include        "delay.h"

void
DelayMs(unsigned char cnt)
{
#if     XTAL_FREQ <= 2MHZ
        do {
                DelayUs(996);
        } while(--cnt);
#endif

#if   XTAL_FREQ > 2MHZ
        unsigned char i;
        do {
                i = 4;
                do {
                        DelayUs(250);
                } while(--i);
        } while(--cnt);
#endif
}

void
DelayMsi(unsigned char cnt)
{
#if     XTAL_FREQ <= 2MHZ
        do {
                DelayUs(996);
        } while(--cnt);
#endif

#if   XTAL_FREQ > 2MHZ
        unsigned char i;
        do {
                i = 4;
                do {
                        DelayUs(250);
                } while(--i);
        } while(--cnt);
#endif
}
```

DISPLAY.C

```
#include <pic.h>

#include "display.h"


//variables de asm
char r0, r1, r2, count, temp, L_byte, H_byte;


//variables de display
char mostrard1, mostrard2, mostrard3, mostrard4, mostrard5 ;
unsigned int vtemporal;

//Transformo binario a bcd
void  binbcd16d(unsigned int dato)
{
  char var0;
  vtemporal = dato;
  ///cargo archivos
  asm("movf _vtemporal,w ");
  asm("movwf _L_byte");
  asm("movf _vtemporal+1,w ");
  asm("movwf _H_byte");



#asm
B2_BCD

        bcf 0x03,0 ; clear the carry bit
        movlw 0x10
        movwf _count
        clrf _r0
        clrf _r1
        clrf _r2
loop16
        rlf _L_byte, f
        rlf _H_byte, f
        rlf _r2, f
        rlf _r1, f
        rlf _r0, f

        decfsz _count,f
        goto adjDEC
   goto fin
```

```
        adjDEC
        ;movlw _r2
       ; movwf 0x04
         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call adjBCD
   movlw 3
        addwf _r2,w
        movwf _temp
        btfsc _temp,3 ; test if result > 7
        movwf _r2
        movlw 0x30
        addwf _r2,w
        movwf _temp
        btfsc _temp,7 ; test if result > 7
        movwf _r2 ; save as MSD

       ; movlw _r1
       ; movwf 0x04
         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call adjBCD
   movlw 3
        addwf _r1,w
        movwf _temp
        btfsc _temp,3 ; test if result > 7
        movwf _r1
        movlw 0x30
        addwf _r1,w
        movwf _temp
        btfsc _temp,7 ; test if result > 7
        movwf _r1 ; save as MSD


        movlw _r0
        movwf 0x04
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;call adjBCD
   movlw 3
        addwf 0,w
        movwf _temp
        btfsc _temp,3 ; test if result > 7
        movwf 0
        movlw 0x30
        addwf 0,w
        movwf _temp
        btfsc _temp,7
        movwf 0 ; save as MSD
 goto loop16
fin

     movf   _r2,w
     movwf  _vtemporal
     movf   _r1,w
     movwf  _vtemporal+1
```

```
#endasm
    var0 = r2;
    var0 &= 0x0f;
    mostrard1 = display[var0];
    var0 = r2;
    var0 >>= 4;
    mostrard2 = display[var0];
    var0 = r1;
    var0 &= 0x0f;
    mostrard3 = display[var0];
    var0 = r1;
    var0 >>= 4;
    mostrard4 = display[var0];
    var0 = r0;
    var0 &= 0x0f;
    mostrard5 = display[var0];

}
```

LCD.C

```
/*
*       LCD interface example
*       Uses routines from delay.c
*       This code will interface to a standard LCD controller
*       like the Hitachi HD44780. It uses it in 4 bit mode, with
*       the hardware connected as follows (the standard 14 pin
*       LCD connector is used):
*
*       PORTB bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
*       PORTA bit 2 is connected to the LCD RS input (register select)
*       PORTA bit 3 is connected to the LCD EN bit (enable)
*
*       To use these routines, set up the port I/O (TRISA, TRISB) then
*       call lcd_init(), then other routines as required.
*
*/

#include <pic.h>
#include "lcd.h"
#include "delay.h"


/* write a byte to the LCD in 4 bit mode */

void
lcd_write(unsigned char c)
{
        PORTB = (PORTB & 0x07) | (c & 0xf0);
```

```
  LCD_EN = 1;
  DelayUs(255);
  LCD_EN = 0;
      PORTB = (PORTB & 0x07) | (c << 4);
      LCD_EN = 1;
  DelayUs(255);
  LCD_EN = 0;
}

/*
*       Clear and home the LCD
*/

void
lcd_clear(void)
{
      LCD_RS = 0;
      lcd_write(0x1);
      DelayMs(2);
}

/* write a string of chars to the LCD */

void
lcd_puts(const char * s)
{
      LCD_RS = 1; // write characters
      while(*s)
            lcd_write(*s++);
}

/* write one character to the LCD */

void
lcd_putch(char c)
{
      LCD_RS = 1; // write characters
  PORTB = (PORTB & 0x0f) | (c & 0xf0);
  LCD_EN = 1;
      DelayUs(255);
  LCD_EN = 0;
      PORTB = (PORTB & 0x0f) | (c << 4);
      LCD_EN = 1;
      DelayUs(255);
  LCD_EN = 0;
      DelayUs(40);
  LCD_RS = 0;
}
```

```
/*
 * Go to the specified position
 */


Void lcd_goto(unsigned char pos)
{
        LCD_RS = 0;
        lcd_write(0x80+pos);
}

/* initialise the LCD - put into 4 bit mode */

void
lcd_init(void)
{
   LCD_EN = 0;
        LCD_RS = 0; // write control bytes
        DelayMs(250);          // power on delay
        PORTB = 0x20;          // set 4 bit mode
   LCD_EN = 1;
        DelayMs(1);
   LCD_EN = 0;
   DelayMs(1);
        lcd_write(0x28);       // 4 bit mode, 1/16 duty, 5x8 font
        lcd_write(0x0F);       // display on, blink curson on
        lcd_write(0x06);       // entry mode
}
```

MAIN.H

```
#define         PORTBIT(adr, bit)    ((unsigned)(&adr)*8+(bit))
extern void sendI2C(char  , int );
static bit      cargando   @ PORTBIT(PORTD, 4);
static bit      transferir @ PORTBIT(PORTD, 5);
static bit      parar      @ PORTBIT(PORTD, 6);
static bit      WP         @ PORTBIT(PORTC, 0);


static bit      mas    @ PORTBIT(PORTB, 1);
static bit      menos  @ PORTBIT(PORTB, 2);
static bit      cambiar @ PORTBIT(PORTB, 3);
//
///
static bit      ledc   @ PORTBIT(PORTE, 0);
static bit      ledt   @ PORTBIT(PORTE, 1);
static bit      ledmll @ PORTBIT(PORTE, 2);
```

DELAY.H

```
/*
*       Delay functions for HI-TECH C on the PIC
*
*       Functions available:
*               DelayUs(x)      Delay specified number of microseconds
*               DelayMs(x)      Delay specified number of milliseconds
*
*       Note that there are range limits: x must not exceed 255 - for xtal
*       frequencies > 12MHz the range for DelayUs is even smaller.
*       To use DelayUs it is only necessary to include this file; to use
*       DelayMs you must include delay.c in your project.
*
*/

/*      Set the crystal frequency in the CPP predefined symbols list in
        HPDPIC, or on the PICC commmand line, e.g.
        picc -DXTAL_FREQ=4MHZ

        or
        picc -DXTAL_FREQ=100KHZ

        Note that this is the crystal frequency, the CPU clock is
        divided by 4.

*       MAKE SURE this code is compiled with full optimization!!!

*/

#ifndefXTAL_FREQ
#define     XTAL_FREQ 20MHZ                 /* Crystal frequency in MHz */
#endif

#define     MHZ   *1000L                        /* number of kHz in a MHz */
#define     KHZ   *1                    /* number of kHz in a kHz */

#if   XTAL_FREQ >= 12MHZ

#define     DelayUs(x)    { unsigned char _dcnt; \
                    _dcnt = (x)*((XTAL_FREQ)/(12MHZ)); \
                    while(--_dcnt != 0) \
                        continue; }
#else
```

```
#define         DelayUs(x)     { unsigned char _dcnt; \
                        _dcnt = (x)/((12MHZ)/(XTAL_FREQ))|1; \
                        while(--_dcnt != 0) \
                                continue; }
#endif

extern void DelayMs(unsigned char);


extern void DelayMsi(unsigned char);
```

DISPLAY.H

```
#define pdecimal   2
#define npdecimal  0xfd


extern char mostrard1;
extern char mostrard2;
extern char mostrard3;
extern char mostrard4;
extern char mostrard5;

extern void binbcd16d(unsigned int);
extern void mostrar_display(char);
extern void timer2(void);
```

LCD.H

```
/*
 *      LCD interface header file
 *      See lcd.c for more info
 */

static bit LCD_RS     @ ((unsigned)&PORTA*8+1);       // Register select
static bit LCD_RW     @ ((unsigned)&PORTA*8+2);       // Enable
static bit LCD_EN     @ ((unsigned)&PORTA*8+3);       // Enable

#define         LCD_STROBE        ((LCD_EN = 1),(LCD_EN=0))

/* write a byte to the LCD in 4 bit mode */

extern void lcd_write(unsigned char);
extern void lcd_func(unsigned char);
/* Clear and home the LCD */
```

```
extern void lcd_clear(void);

/* write a string of characters to the LCD */

extern void lcd_puts(const char * s);

/* Go to the specified position */

extern void lcd_goto(unsigned char pos);

/* intialize the LCD - call before anything else */

extern void lcd_init(void);

extern void lcd_putch(char);

/*      Set the cursor position */

#define         lcd_cursor(x)  lcd_write(((x)&0x7F)|0x80)
```

REFERENCIAS BIBLIOGRÁFICAS

1. Coughlin, R. and Driscoll F., Amplificadores Operacionales y Circuitos Integrados Lineales, México: Prentice Hall, 1999, pp.214-244.
2. Jacob, M., Applications and Design with Analog Integrated Circuits, New Jersey: Prentice Hall, 1982, pp. 359-425.
3. Maltzahn Von, W., and Yapur M., Medical Electronics, Folleto Técnico, Facultad de Ingeniería en Electricidad y Computación, ESPOL, 1987, pp. 1-20.
4. Iplotlibrary.dll for WIN 2000/XP, Disponible en http://www.iocomp.com
5. Comandos del puerto serial. Disponible en : http://support.microsoft.com/kb/139526/es
6. El corazón y su anatomía, Disponible en : http://www.virtual.unal.edu.co/cursos/medicina/2005050/docs_curso/contenido.html
7. EKG y derivaciones cardiacas , Disponible en: http://www.medspain.com/curso_ekg/cursoekg_indice.htm