UNIVERSIDAD DON BOSCO

TEMA:

" DISEÑO DE UN EQUIPO PARA TRANSMITIR SEÑALES FISIOLÓGICAS DE FORMA INALÁMBRICA "

TRABAJO DE GRADUACIÓN

PREPARADO PARA LA FACULTAD DE INGENIERÍA

PARA OPTAR AL GRADO DE

INGENIERO EN ELECTRÓNICA

PRESENTADO POR:

MANUEL DE JESUS HENRIQUEZ CORTEZ

WILFREDO GARCÍA REYES

FEBRERO DE 2005

SOYAPANGO EL SALVADOR AMERICA CENTRAL

UNIVERSIDAD DON BOSCO

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

SECRETARIO GENERAL

LIC. MARIO RAFAEL OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA

ING. ERNESTO GODOFREDO GIRÓN

ASESOR DEL TRABAJO DE GRADUACIÓN

ING. WENCESLAO RIVAS

JURADO EVALUADOR

ING. CALIXTO RODRIGUEZ

ING. JUAN CARLOS CASTRO

ING. JUAN RENE NUÑES

TUTOR DEL TRABAJO DE GRADUACIÓN

ING. EDUARDO RIVERA

ÍNDICE

INTRODUCCION

El siguiente documento presenta el desarrollo del tema de graduación: " Diseño de un Equipo para Transmitir Señales Fisiológicas de Forma Inalámbrica ".

En primer lugar se plantea la base teórica que determina el problema y soluciones que determinan o justifican la realización del proyecto. Se trata en el primer capítulo la base de los equipos que pueden utilizarse actualmente para monitorear el estado o condiciones de salud de una persona enferma del corazón.

El segundo capítulo trata sobre el problema planteado, al cual se le pretende dar solución. Continúa con la descripción de las soluciones posibles, calificándolas y haciendo una evaluación para determinar la más optima. Esto se realiza de forma escalonada, presentando para cada alternativa diferentes rutas o soluciones. En el apartado solución propuesta, se describe en forma de resumen, la alternativa que fue elegida. Al final de este capítulo se justifica el desarrollo del trabajo de graduación, planteando los beneficios que se consiguen con su implementación.

En el capítulo III, Arquitectura del sistema, se desarrollan las partes principales del hardware del sistema, planteando las herramientas de diseño que se utilizan con ellas.

En el Capítulo IV, llamado El Prototipo, se muestran y

analizan los programas empleados para comandar la operación del sistema. La explicación de los programas se facilita por medio del uso de flujogramas. Se explica y justifica por medio de cálculos cada etapa del sistema. El prototipo incluye la estación central (hardware y software) de la cual forma parte el computador y una estación remota física, la cual esta diseñada para manejar información de señales fisiológicas ya definidas.

Las referencias se encuentran detalladas al final del documento, con el fin de que el lector amplié sus conocimientos sobre los dispositivos y sistemas utilizados.

CAPÍTULO I

1.1 ANTECEDENTES HISTORICOS.

Un electrocardiograma o ECG, es un registro eléctrico del corazón y es usado en la investigación de enfermedades del corazón.

El fisiólogo británico Augustus D. Waller fue el pionero de la electrocardiografía y en 1887 publicó el primer electrocardiograma humano. Trece años más tarde, el premio Nóbel en medicina fue otorgado a un fisiólogo llamado Willem Einthoven, quien cambio este curioso fenómeno fisiológico en un indispensable aparato clínico de registro que se mantiene aún en nuestros días.[1]

1.2 SITUACIÓN ACTUAL.

El Electrocardiograma (su sigla en inglés es EKG) / El Examen de Estrés / El Monitor Holter

¿Qué es un electrocardiograma? [2]

Un electrocardiograma (llamado comúnmente EKG o ECG), es una medición de la actividad eléctrica del corazón. Colocando electrodos en lugares específicos del cuerpo (pecho, brazos y piernas), se puede obtener una representación gráfica, o trazado, de la actividad eléctrica. Los cambios en el trazado normal de un ECG pueden indicar una o más condiciones relacionadas con el corazón. Otras condiciones, que no son del corazón, también pueden causar cambios en el ECG.

Una descripción en el dominio de la frecuencia de una señal ECG es necesaria para determinar los parámetros del sistema de preprocesamiento lineal. Una información importante acerca del contenido de frecuencias de la señal del ECG es el ancho de banda. El rango básico de frecuencias del corazón es 0.5 a 3.0 Hz para un rango de latidos del corazón de 30 a 180 beats/ min. (bpm) [3]. La frecuencia más alta depende del estado de salud, edad y sexo del paciente, pero el valor típico es por 125 Hz[4].

En el caso de procesamiento de ECG en infantes es arriba de 150 HZ[4]. El rango normal de un corazón relajado es de 60 a 100 bpm[4].

Los siguientes anchos de banda son utilizados por diferentes aplicaciones del ECG.[5]

a- Ancho de banda clínico para un ECG estándar de 12 electrodos 0.05 a 100 Hz.[5]

b- Aplicaciones de monitoreo para Unidades de Cuidados Intensivos (ICU), 0.5 a 50 Hz. Es usado para detectar disturbios del ritmo (ejemplo, arritmias). El ancho de banda restringido atenúa sonidos con altas frecuencias causados por contracción de músculos y sonidos de baja frecuencia causados por movimientos de los electrodos.[5]

c- Medidor del rango del corazón / cardiotacómetro. Es un pasa banda centrado en 17 Hz y una selectividad (Q) entre 3 y 4.[5]

d- Medida del potencial de los latidos. Ancho de banda arriba de 500 Hz, es usado para medir el potencial de los latidos.[5]

¿Cómo se hace un ECG?[2]

Un ECG es uno de los procedimientos más rápidos y sencillos que se utilizan para evaluar el corazón. Se pondrán 12 electrodos diferentes (pequeños parches de plástico) en lugares específicos del pecho, los brazos y las piernas. Le colocarán ocho electrodos en el pecho y un electrodo en cada brazo y en cada pierna. Los electrodos pueden ser autoadhesivos, o se puede aplicar un gel para que los electrodos se peguen a la piel. El paciente tendrá que estar acostado o tendido en una camilla, y las derivaciones (cables) se conectarán a los electrodos colocados en la piel. Es necesario que esté muy quieto y que no hable durante el ECG, ya que cualquier movimiento puede crear interferencias en el trazado. Se empezará a obtener el trazado, que durará sólo unos minutos. No notará nada durante el registro. Una vez que se haya obtenido un trazado claro, le quitarán los electrodos y las derivaciones y podrá seguir con sus actividades normales, a menos que su médico le indique lo contrario. Un ECG puede indicar la presencia de arritmias (ritmo anormal del corazón), de daños en el corazón causados por isquemia (falta de oxígeno en el músculo cardiaco) o infarto de miocardio (MI o ataque al corazón), problemas en una o más de las válvulas cardiacas u otros tipos de condiciones cardiacas.

Existen procedimientos de ECG que son más complicados que el ECG básico. Entre estos procedimientos se incluyen los siguientes:[2] ECG de ejercicio o examen de esfuerzo: Se conecta al paciente a un aparato de ECG como describimos anteriormente. Sin embargo, en lugar de estar acostado, el paciente tiene que caminar en una cinta continúa o pedalear en una bicicleta estática mientras se registra el ECG. Este examen se hace para evaluar los cambios en el ECG durante una situación de estrés como el ejercicio.

- Electrocardiograma de Promediación de Señales: Este procedimiento se hace de la misma forma que un ECG en reposo, excepto que la actividad eléctrica del corazón se registra durante un período de tiempo más largo, generalmente de 15 a 20 minutos. El ECG de promediación de señales se hace cuando se sospecha una arritmia que no se ve en un ECG de reposo, ya que las arritmias pueden ser transitorias de naturaleza y no verse durante el corto período de tiempo que dura un ECG en reposo.

- Monitorización con Holter: Una monitorización con Holter es una grabación de ECG que se realiza durante 24 horas o más. Se pegan tres electrodos en el pecho del paciente y se conectan a un grabador de ECG portátil mediante cables de derivaciones. Durante este procedimiento, el paciente sigue con sus actividades cotidianas (excepto actividades como ducharse, nadar o cualquier otra cosa que pueda producir una sudoración excesiva que haga que los electrodos se aflojen o se caigan). Existen 2 tipos de monitorización con Holter:

☐ Registro continuo - el ECG se graba continúamente durante todo el período que dure el examen.

☐ Registro de eventos o grabador de captura - el ECG se graba sólo cuando el paciente siente los síntomas y aprieta el botón de grabación.

La monitorización con Holter se podría hacer cuando se sospecha una arritmia pero no aparece en el ECG de promediación de señales, ya que las arritmias pueden ser transitorias y no aparecer durante el corto período de grabación de un ECG en reposo o de un ECG de promediación de señales[2].

A pesar de que en la actualidad algunos sistemas utilizados para monitorear las señales fisiológicas poseen o se les puede convertir en elementos con acceso a red digital, no todos los

equipos poseen esta facilidad. Sin embargo la tendencia tecnológica apunta a la integración de los sistemas. No será extraño ver en el futuro que todos estos dispositivos tengan una forma de brindar información a grandes distancias.

Además dentro del desarrollo de equipos de transmisión inalámbrica, hay cierta tendencia, principalmente para el estudio de animales, de reducir el equipo de monitoreo, transmitiendo de forma inalámbrica la energía de operación del circuito, logrando con esto la disminución en el tamaño del equipo.

CAPÍTULO II. JUSTIFICACIÓN DEL PROYECTO

2.1 ENUNCIADO DEL PROBLEMA.

¿Que dificultades y que cuidados especiales presentan las personas en nuestro país que tienen diagnosticadas enfermedades del corazón para llevar una vida semi-normal?

2.2 ANALISIS DEL PROBLEMA.

La necesidad de cuidados intensivos y monitorización de pacientes ha sido reconocida durante siglos. La presencia de la enfermera durante las 24 horas del día en el caso de los pacientes en estado crítico se ha convertido con el tiempo en algo familiar dentro del hospital y en algunas casas. Pero solo en los últimos años se ha diseñado y fabricado equipo suficientemente seguro y preciso para ser empleado extensivamente en la monitorización de pacientes. La enfermera aún esta presente pero su papel ha cambiado un tanto, por cuanto ahora tiene a su disposición instrumentos potentes para adquirir y asimilar información de los pacientes que tiene a su cuidado. De este modo es capaz de prestar un servicio mejor a mayor número de pacientes

y es capaz de reaccionar mas rápidamente y de forma adecuada ante una situación de emergencia. El equipo de monitorización hace posible que acuda un médico o enfermera a tiempo para proporcionar una ayuda de emergencia, frecuentemente antes de que pueda producirse una lesión permanente. Con el aviso rápido y proporcionando una información tal como el registro electrocardiográfico antes, durante y después del comienzo de las dificultades cardiacas, el sistema de monitorización permite al médico dar rápidamente al paciente el medicamento adecuado. En algunos casos se puede automatizar incluso este proceso.

El concepto de cuidados coronarios intensivos ha tenido poca factibilidad hasta el desarrollo de equipo electrónico que era capaz de medir de forma fidedigna, permitiendo visualizar la actividad eléctrica del corazón de forma continúa. Con tales monitores cardiacos, ha sido posible detectar arritmias potencialmente fatales. Combinado con equipo de estimulación para reactivar el corazón al producirse una arritmia de este tipo, ahora se dispone de un sistema completo para evitar la muerte repentina en tales casos[6].

Dependiendo del tipo de problema cardiaco es posible que la persona pueda irse a su casa y deba llevar un régimen de vida especial, como un monitoreo constante del comportamiento del corazón y otras variables fisiológicas como la frecuencia

respiratoria y la concentración de oxígeno en la sangre del paciente y probablemente el consumo de medicamentos especiales, todo esto bajo la supervisión de una segunda y hasta tercera persona. Especialmente si el enfermo realiza movimientos libres y sin restricciones como ejercicios o deportes.

Estas enfermedades y cuidados especiales que deben seguir, limitan tanto a los enfermos como a los que los cuidan; sin embargo este monitoreo constante puede realizarlo una máquina, logrando con ello mejorar la calidad de vida de las personas y reducir los riesgos de empeorar las condiciones del paciente, por algún descuido.

2.3 Soluciones Posibles.

En toda etapa de solución de problemas es necesario plantearse diferentes opciones tecnológicas, las cuales van siendo mas específicas a medida se van depurando las alternativas con que se cuentan, y se profundizan las ideas.

Para tal efecto se plantearán las soluciones posibles en forma de organigramas, los cuales son mostrados en la figuras 2.1 a la 2.8 y desarrollado en las secciones siguientes.

En la evaluación de las alternativas posibles han sido considerados diferentes aspectos que determinen la solución mas acertada. Ellos son: El costo, disponibilidad, flexibilidad, facilidad de mantenimiento, seguridad, equipo requerido, forma de alimentación eléctrica y su ubicación.

2.3.1 Métodos.

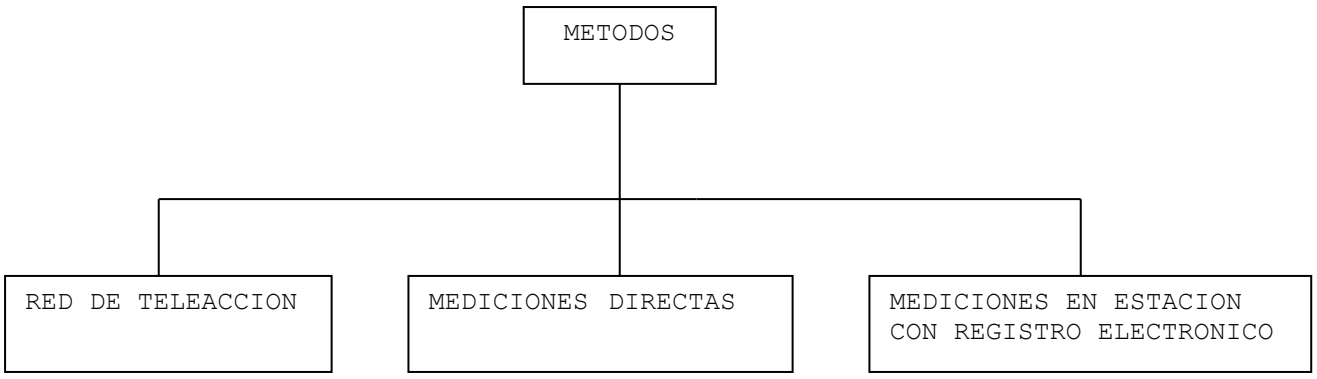


FIGURA 2.1 Métodos de Solución.

2.3.1.1 MEDICIONES DIRECTAS. [7]

Se define como la técnica que hace uso de personal, el cual deberá recopilar la información obtenida del paciente utilizando instrumentos de medición de señales fisiológicas. Estos instrumentos habrán de permanecer por un tiempo determinado en el sitio de permanencia del paciente durante todo el monitoreo. Cuando la información es recopilada por el personal, debe ser llevada o enviada a personal especializado para ser interpretadas y evaluadas.

Para la evaluación de esta alternativa, se consideran los siguientes aspectos:

- No es acorde con los niveles de estudio de ingeniería en electrónica.
- Es un método bastante bueno para la obtención de resultados,

al utilizar las muestras adecuadas.

- Supervisión humana directa.
- Implica el involucramiento de una cantidad considerable de personal para la obtención de las muestras y análisis de las mismas, lo que incrementa los costos.
- Es un método que requiere una buena cantidad de tiempo para observar los resultados y dar conclusiones.

2.3.1.2 MEDICION EN LA ESTACION CON REGISTRO ELECTRONICO.[7]

Se entiende como un sistema el cual posee sensores para la obtención de los parámetros fisiológicos y que dicha información este siendo periódicamente almacenada por un sistema de registro magnético o electrónico. Luego esta información almacenada es recopilada por personal asignado para ser llevada a un laboratorio en donde es leída y convenientemente procesada para la obtención de los resultados y generar reportes.

Para considerar esta alternativa se han evaluado los siguientes aspectos:

- Involucra mayor cantidad de personal para la recopilación de la información almacenada.
- Requiere un mínimo de mantenimiento.
- La información es confiable, pero la cantidad almacenada es limitada.

- Involucra mayor tiempo para la obtención de los resultados.
- Por lo antes expuesto se considera que es una alternativa que permite la obtención de datos confiables, pero involucra más costos operativos y además no va acorde con un sistema rápido para el análisis de la información de las señales fisiológicas de un paciente.

2.3.1.3 RED DE TELEMETRIA.[7]

Se entiende por utilizar un sistema para realizar el monitoreo de señales fisiológicas de un paciente, el cual sea capaz de obtener los datos de medición de los sensores en las estaciones remotas y que pueda enviar por algún medio de transmisión dicha información hacia una estación central, la cual este convenientemente ubicada y que pueda recibir dicha información, procesarla y luego presentarla.

Para la evaluación de esta alternativa se consideraron los siguientes aspectos:

- Facilita la obtención de información sin necesidad de estar presente en el sitio donde se encuentra el paciente.
- Reduce la cantidad de personal necesario para realizar las actividades de monitoreo, lo que reduce costos.
- Permite un mínimo de mantenimiento.

- Permite una obtención rápida de las señales fisiológicas, evitando retrasos en el análisis de la información.
- Facilita por medio de programación, alertar sobre niveles anormales en las señales.

Por los aspectos antes evaluados, se considera que esta es una alternativa bastante viable para la realización de la red de monitoreo y alerta en una persona enferma del corazón.

2.3.2 ELECCIÓN DEL MEDIO DE TRANSMISIÓN.

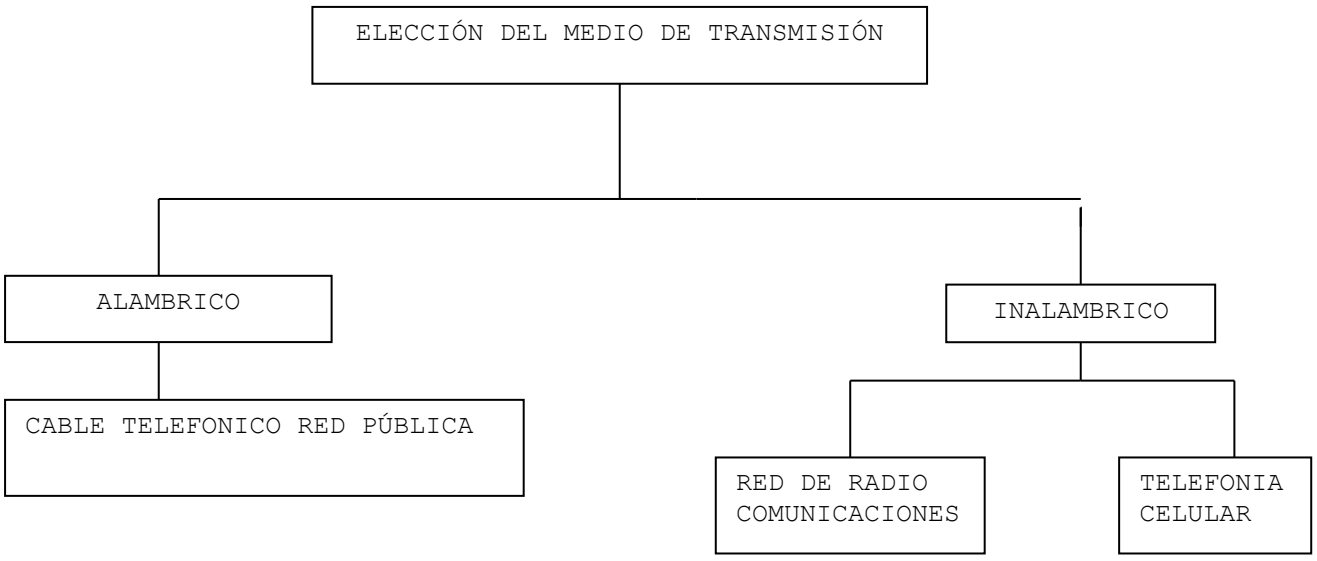


FIGURA 2.2 Elección del Método de Transmisión.

2.3.2.1 ALAMBRICO.

- CABLE TELEFONICO DE LA RED PÚBLICA CONMUTADA.[7]

Como se sabe la red pública se utiliza para conectar durante el tiempo necesario al abonado con su central telefónica en el desarrollo de una llamada. Esta red consta de los aparatos telefónicos, las líneas de los abonados, los circuitos, y todos los equipos de conmutación necesarios para su interconexión.

Por lo tanto para tomar una decisión sobre la viabilidad y factibilidad de uso de este medio de transmisión se consideraron los siguientes aspectos:

- El medio alámbrico limitaría demasiado el movimiento de la persona a monitorear, sería mas factible utilizarla para diagnósticos o períodos cortos de monitoreo.
- Limita la flexibilidad de reubicación de la estación remota, ya que en El Salvador todavía existen puntos específicos donde no hay red telefónica.
- Las líneas conmutadas son susceptibles a sufrir daños por las inclemencias del clima, ejemplo la caída de árboles sobre el tendido telefónico.
- El uso de una red telefónica implica gastos adicionales como la compra de modems de buena calidad, lo que incrementaría los costos del sistema.

Por las razones antes expuestas, para el propósito que se

persigue, esta opción fue descartada como medio de transmisión óptimo para el sistema de monitoreo de señales fisiológicas.

2.3.2.2 INALAMBRICO.

- SISTEMAS DE RADIO COMUNICACIONES MOVILES.

Es posible la transmisión de datos por un sistema de radiocomunicaciones móviles con un grado de fidelidad suficiente para aplicaciones en las cuales no sea de vital importancia que funcionen el 100% del tiempo. Es importante mencionar también que el tamaño de las unidades transmisoras móviles actuales es bien reducido, lo que beneficiaría al paciente, pues su ubicación sería más placentera.

- Los sistemas de Radiocomunicaciones móviles suelen tener un grado de susceptibilidad al ruido, dependiendo en gran medida del estado de los equipos relacionados y de la ubicación de los mismos, además de las fuentes de interferencia cercanas y de los obstáculos colocados entre los móviles.

Actualmente los sistemas de Radiocomunicaciones móviles están siendo sustituidos por nuevas tecnologías como la Telefonía Celular, la cual provee una comunicación del tipo FULL DUPLEX (transmite información en ambos sentidos de forma simultánea, debido a que las frecuencias de transmisión y de recepción están muy separadas entre si).

- Otro factor a tomar en cuenta es su alto costo y su gran cobertura.
- Reduce la cantidad de personal necesario para realizar las actividades de monitoreo a distancia.
- Permite un mínimo de mantenimiento.
- Permite una obtención rápida de las señales fisiológicas, evitando retrasos.
- Permite el libre movimiento del paciente dentro de la cobertura del equipo de Transmisión/ Recepción[A].

Por las razones antes expuestas es viable el uso de este equipo para el desarrollo del sistema propuesto.

- TRANSMISION POR MODULACION DE FRECUENCIA.

La modulación en frecuencia (Angulo) fue introducida por primera vez en 1931 como una alternativa para la Modulación en Amplitud (AM). Debido a que la Modulación en frecuencia (FM) era menos susceptible al ruido que la AM y, consecuentemente, mejoraría el funcionamiento de las comunicaciones por radio.[8]

Actualmente los sistemas de comunicación que utilizan FM se utilizan bastante en la rama de la Biomédica principalmente por lo sencillo de sus equipos y su inmunidad al ruido. Ejemplo de ello es que los sistemas de FM han sido un equipamiento estándar en el ámbito educativo durante varios años para los niños con

[A] En adelante Transmisor se abreviara como Tx y Receptor como Rx.

pérdidas auditivas.

Además son ampliamente utilizados en otros campos como:

- Seguridad.
- Estudio.
- Turismo.
- Deportes.
- Audífonos para comunicación alumno- maestro.
- Transmisor de muñeca con sintetizador para aplicaciones en escuelas y conferencias.
- Transmisor/ Micrófono Universal.

Por lo tanto para tomar una decisión sobre la viabilidad y factibilidad de uso de transmisor/ receptor FM se consideran los siguientes aspectos.

- Debido a la sencillez de los equipos es posible elaborarlos y ubicarlos fácilmente.
- Por el poco alcance requerido el equipo sería pequeño.
- El ambiente en que se utilizaría sería domiciliar, por lo cual no habrían cambios grandes y bruscos de temperatura, los cuales afectaran notablemente el funcionamiento del equipo.
- El costo de estos equipos sería mínimo.
- Facilidad de adquisición en el mercado local.

Por las razones antes expuestas, se considera que esta es la mejor alternativa para el desarrollo del Transmisor/ Receptor.

2.3.3 ELECCION DE LA FORMA DE DIÁLOGO ENTRE LA ESTACION CENTRAL Y ESTACION REMOTA.

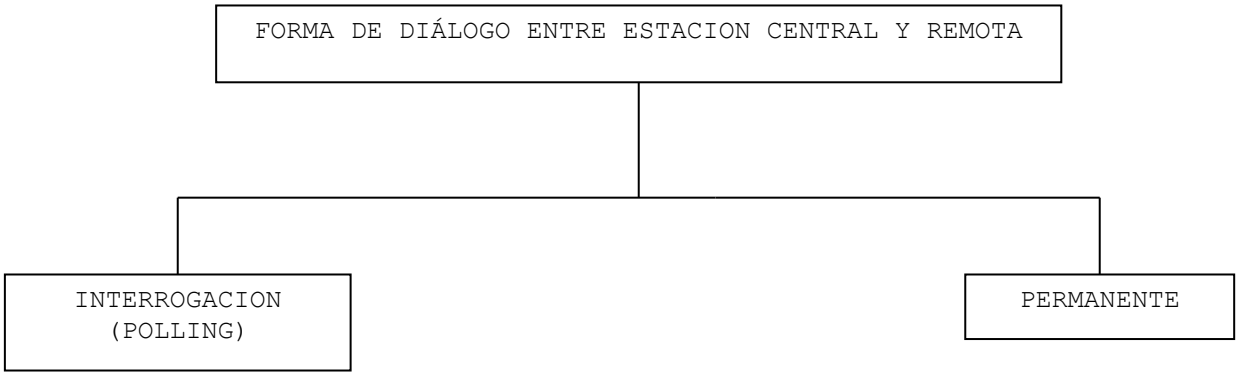


FIGURA 2.3 Forma de Dialogo entre Estaciones.

2.3.3.1 PERMANENTE.[7]

Es la forma de acceso para la obtención de la información, recopilada por la estación remota, con la cual existe un enlace permanente por parte de la estación central.

- Implica mayor consumo de energía por la operatividad de los equipos.
- La información es en tiempo real, lo que facilita tomar decisiones oportunas sobre posibles problemas.
- El diseño del equipo es sencillo, lo cual minimiza su tamaño.
- El costo de implementación y diseño es bajo.

Por las razones antes mencionadas, esta alternativa se tomará, para este proyecto.

2.3.3.2 INTERROGACION (POLLING).[7]

Forma de acceso para la obtención de la información monitoreada por las estaciones, la cual involucra la técnica de interrogación maestro- esclavo, mediante la cual la estación central esta interrogando y las estaciones remotas responden alternadamente. Cada estación remota o local tendrá asignado un código de identificación, así responderá solamente la estación o circuito a que se este llamando. La estación central estará cada cierto período de tiempo generando las señales convenientes, para comunicarse con las estaciones en una forma ordenada, es decir una a la vez.

Para la evaluación de esta alternativa se considera que:

- Cada estación o circuito tiene su propia identificación, todas escuchan y una sola responde.
- Permite el ahorro de tiempo de trabajo y energía, ya que no se envía información siempre.
- La información no es en tiempo real, hay pequeños vacíos de esta.
- Los circuitos son más complejos, por lo cual su tamaño aumenta con relación a la opción anterior.

En base a lo antes tratado se considera una alternativa viable para la etapa de adquisición y como una segunda alternativa para la etapa de transmisión/ recepción.

2.3.4 ELECCION EN BASE A EFECTIVIDAD DE COBERTURA.

En base a la cobertura deseada se menciona.

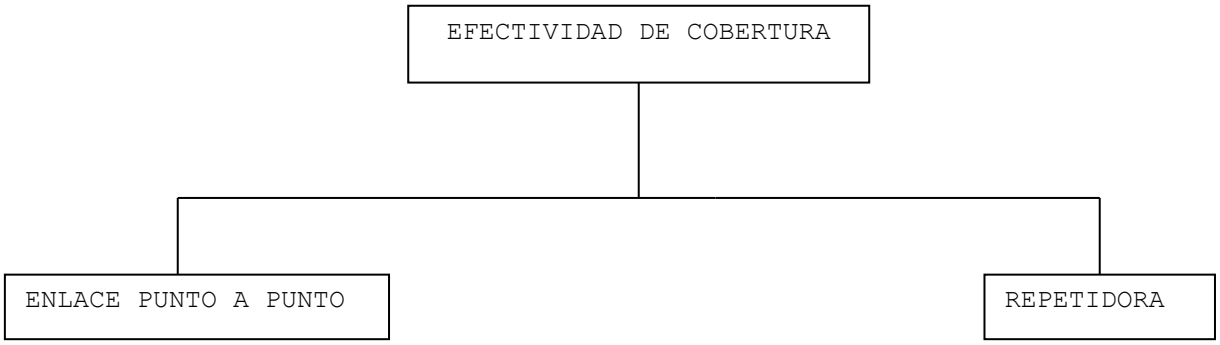


FIGURA 2.4 Soluciones de Acuerdo a la Cobertura Deseada.

2.3.4.1 ENLACE PUNTO A PUNTO.[7]

Los enlaces punto a punto son aplicables en sistemas de poca cobertura, ya que la transmisión de información se realiza directamente entre las dos estaciones que desean comunicarse entre si. La distancia máxima entre ambas estaciones depende de la capacidad de manejo de potencia de los transmisores y del tipo de propagación de la frecuencia seleccionada. Una comunicación punto a punto no quiere decir que los enlaces son permanentes. Ya que pueden utilizar un único canal de forma alternada.

Para la evaluación de esta alternativa se considera que:

- Poca cobertura con un solo enlace.
- Poca complejidad en un solo sistema.
- Bajo costo.

Debido a las alternativas antes mencionadas, esta es la opción elegida para el desarrollo del proyecto.

2.3.5 ELECCIÓN DE LA FORMA DE ALIMENTACIÓN DE LA TERMINAL REMOTA.

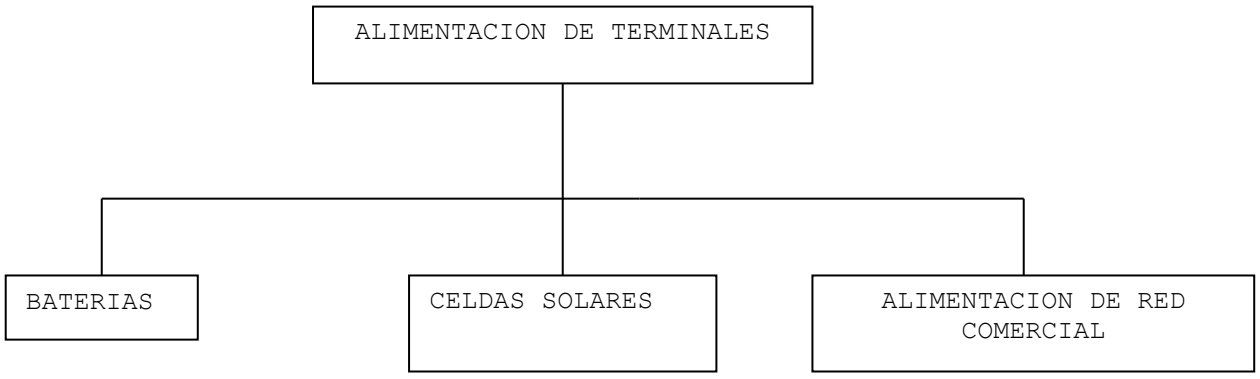


FIGURA 2.5 Formas de Alimentación del Equipo.

2.3.5.1 POR BATERIAS.

Para proporcionar la alimentación de la Terminal existen un par de alternativas.

En caso de ser por baterías, estas deben poseer la característica de ser recargables, para no incurrir en altos costos.

Además estas necesitan una fuente externa de carga asociada, para alimentar a la batería de reserva.

Otro factor a tomar en cuenta es su tamaño, pues alimentará a un circuito portátil.

Por las características antes mencionadas, esta es la opción que se adoptará para la Terminal remota.

2.3.5.2 POR ENERGIA SOLAR.[7]

Entre las características que posee este sistema tenemos:

- Hace uso de paneles solares de silicón de alta eficiencia, para traducir la luz solar a flujo eléctrico.
- No necesita mantenimiento.
- Es necesario un banco de baterías especiales, para acumular la energía.
- Se suministra energía durante las 24 horas del día, haciendo uso complementario de baterías.
- El número de paneles solares y número de baterías, varía dependiendo de la carga a suplir.
- Es necesario un control de carga entre el panel solar y las baterías.
- Esto elimina la necesidad del cambio constante de baterías.

Debido a lo antes expuesto esta alternativa fue dejada como segunda opción, si se desea mejorar el sistema.

2.3.5.3 A TRAVÉS DE LA RED COMERCIAL.

Entre las características que posee este sistema tenemos:

- Factibilidad de uso en un ambiente residencial.
- Susceptible a ruidos por otros equipos.
- Inyección de ruido debido a su frecuencia.

Por lo antes mencionado, se elige esta opción para la etapa de alimentación del equipo receptor.

2.3.6 ELECCIÓN DE LA TECNOLOGÍA DEL ELEMENTO TERMINAL Y DE LA ESTACIÓN CENTRAL.

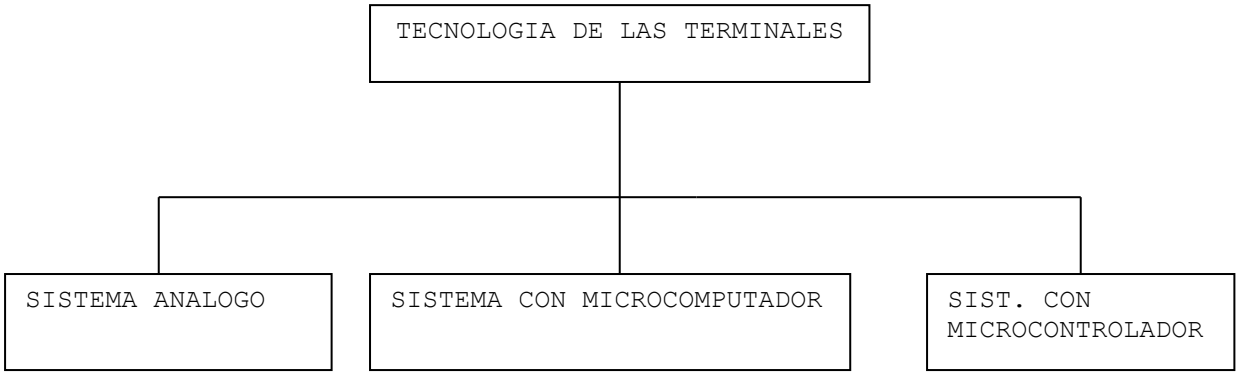


FIGURA 2.6 Tecnologías Posibles de las Estaciones.

2.3.6.1 SISTEMA ANALOGO.

Esta etapa comprende el uso de la tecnología analógica para el diseño del sistema de la Terminal remota haciendo uso de circuitería en base a transistores y amplificadores operacionales.

Esto le daría sencillez al circuito, así como bajo costo.

Fueron consideradas las siguientes alternativas, para evaluar la propuesta:

- Sencillez en el diseño y tamaño reducido.
- Poca flexibilidad para modificaciones futuras.

- En el ambiente residencial, no tenemos grandes variaciones de temperaturas.
- El costo sería reducido.
- Los materiales para la construcción son fácilmente accesibles.
- La información transmitida y recibida sería procesada de forma paralela.

Por las razones antes expuestas es viable el uso de esta tecnología para el proyecto.

2.3.6.2 SISTEMA MICROPROCESADO.[7]

Los sistemas microprocesados resultan bastante eficaces para el manejo de grandes volúmenes de información y tienen la habilidad de ser programados para operar los datos sin la intervención humana.

Fueron consideradas las siguientes características para evaluar esta alternativa:

- El espacio utilizado es grande en relación al sistema análogo.
- El costo del microcontrolador y elementos periféricos sería alto.
- El uso de microcontrolador requiere equipo externo para programarlo, esto incrementaría el costo.
- Son sistemas de alta velocidad para el manejo de datos.
- Permite realizar modificaciones futuras con el mismo hardware.

- La información transmitida/ recibida sería de forma serie.
- Se dificulta su adquisición en el mercado local.

Todos los aspectos antes evaluados, se consideran para hacer de esta alternativa algo no viable para EL TERMINAL REMOTO.

2.3.7.1 COMPUTADOR PERSONAL.

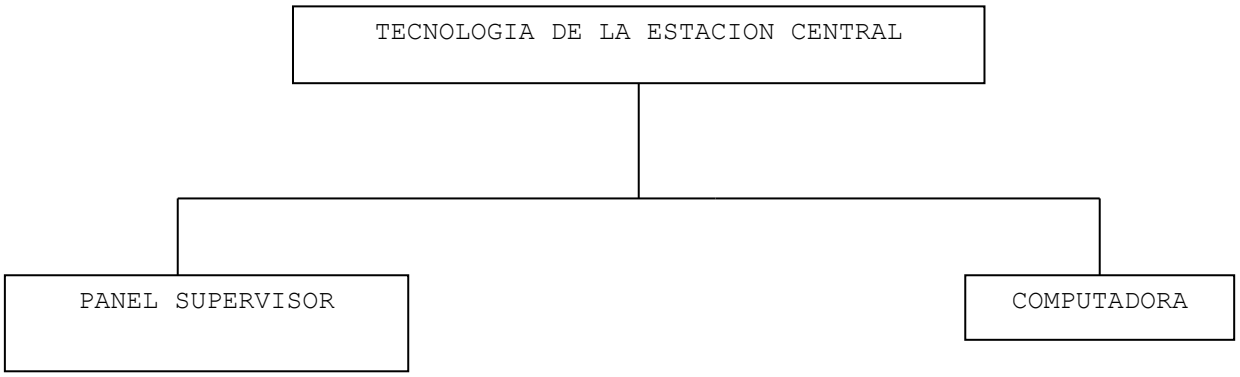


FIGURA 2.7 Alternativas para la Estación Central.

Los sistemas computarizados en la actualidad están tomando gran auge; debido a excelentes ventajas, como son: Gran capacidad de almacenamiento y manejo de información, altas velocidades de operación, facilidad de intercomunicación con dispositivos externos, soporte de softwares especializados que permiten de una forma práctica y fácil la interacción con usuarios o aplicaciones, etc.

Al aplicar la computadora como solución, obtenemos las siguientes ventajas: Tamaño compacto independiente en gran medida del

crecimiento de la red telemétrica, ya que cada estación nueva implica un campo de memoria nuevo y no una ampliación de hardware; posee puertos paralelos y seríales que facilitan la intercomunicación con los dispositivos periféricos para llegar a todos los elementos de la red.

Las computadoras son tan comunes que es fácil encontrar repuestos, evitando así tiempos muertos muy largos en la operación del sistema; la capacidad de almacenamiento de información es grande; no necesita un operador permanente, ya que los datos pueden ser almacenados automáticamente.

Todo lo antes mencionado, justifica la incorporación de la computadora como una buena solución al problema planteado.

2.3.8 ELECCIÓN DE LAS TÉCNICAS DE MEDICIÓN USADAS POR LA TERMINAL REMOTA.

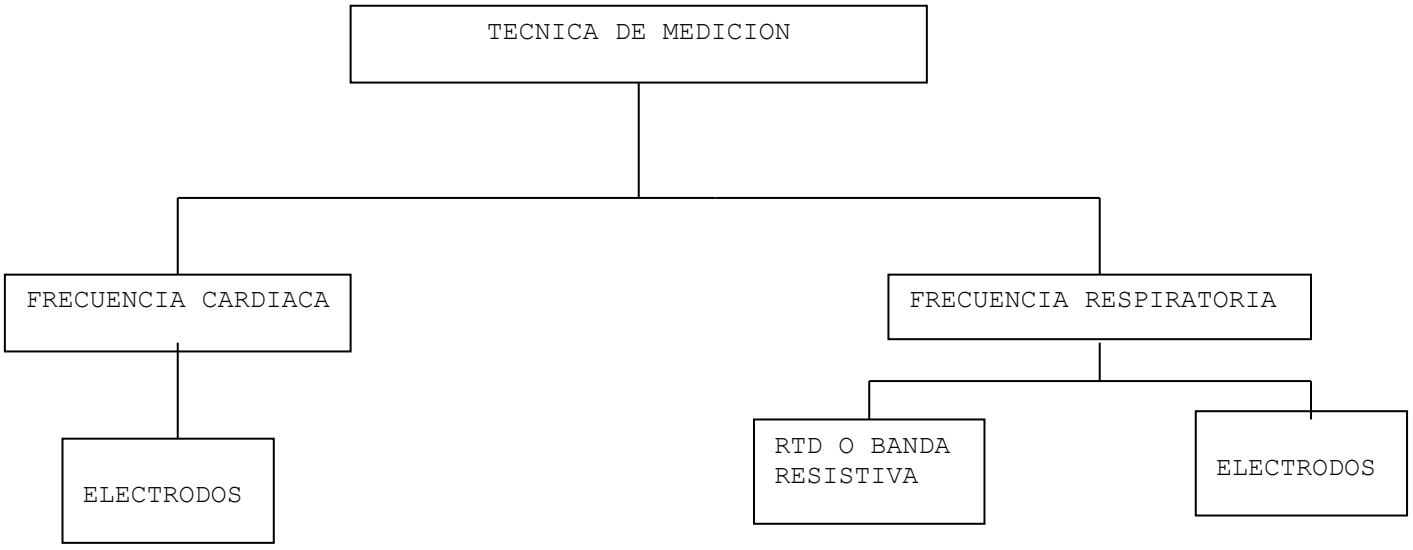


FIGURA 2.8 Técnicas de Medición Posibles.

2.3.8.1 MEDICION DE LA ACTIVIDAD ELECTRICA DEL CORAZON.

Para la detección de la actividad eléctrica del corazón se hace uso del método popular que es el de utilizar electrodos, estos dispositivos convierten los potenciales iónicos que produce el cuerpo en potenciales electrónicos.

Para medir fenómenos bioeléctricos se pueden utilizar una amplia variedad de electrodos, pero casi todos se pueden clasificar como pertenecientes a uno de tres tipos básicos: [6]

1) Micro electrodos. Electrodos utilizados para medir potenciales bioeléctricos cerca o dentro de una célula.[6]

2) Electrodos superficiales. Electrodos utilizados para medir potenciales ECG, EEG y EMG en la superficie de la piel.[6]
3) Electrodos de aguja. Electrodos utilizados para atravesar la piel para registrar potenciales EEG en una región local del cerebro o potenciales EMG en un grupo de músculos especifico. [6]

Para este proyecto se opto por el tipo de electrodo superficial, pues es el que mas se adapta a nuestras necesidades.

2.3.8.2 MEDICION DE LA FRECUENCIA RESPIRATORIA.

Existen dos opciones medirla a través del uso de electrodos y una fuente externa y medirla utilizando un resistor variable con la temperatura.

Utilizando los electrodos del ECG, se evitaría tirar mas cables con sensores al paciente, pero esto traería el inconveniente de tener que suministrar un pequeño voltaje para poner medir las variaciones de impedancia que origina el ritmo respiratorio. Este inconveniente no lo tiene el RTD, sin embargo el uso de este sensor implica molestias físicas al paciente, pues este debe estar montado sobre la nariz.

2.4 SOLUCION PROPUESTA.

Luego de haber realizado una depuración de algunas de las alternativas posibles, se presentará la solución adoptada para el problema antes planteado. En la figura 2.9 y 2.10 se presentan los diagramas en bloques de la estación remota y central del sistema.

Se propone un sistema biotelemétrico, el cual tendrá la finalidad de medir los parámetros fisiológicos de la frecuencia respiratoria y la actividad eléctrica del corazón a distancia. Se utilizará un sistema de radiocomunicaciones como medio de transmisión, utilizando un sistema punto a punto, pues el sistema tendrá poca área de cobertura y estos sistemas son muy sencillos y compactos. La forma de diálogo a emplear entre la estación central y la estación remota será el método permanente ya que se requiere información de forma continúa.

El suministro de energía de la estación remota se realizará a través de la alimentación de corriente continúa utilizando baterías recargables pues su precio es bajo a largo plazo y su tamaño compacto, permitiendo facilidad de movimiento al sistema. En cambio la estación central utilizará alimentación de corriente alterna del sistema público de energía eléctrica.

La estación remota hará uso de un sistema de tecnología analógica, el cual optimiza el uso del espacio permitiendo su

fácil ubicación y traslado.

Se ha decidido utilizar para la estación central una computadora personal, debido a su gran capacidad de manejo de información, alta velocidad de procesamiento, susceptibilidad a las expansiones y al ambiente amigable al usuario.

Respecto a elección de la técnica de medición para las dos señales fisiológicas a monitorear se utilizará electrodos para obtener la actividad eléctrica del corazón y NTC para la frecuencia respiratoria.

Las etapas de la estación remota y central del sistema, se describen a continuación.

- Amplificador ECG.

Es el encargado de amplificar el nivel de voltaje generado por la actividad eléctrica del corazón.

- Pneumografo.

Es aquel que traduce las variaciones provocadas por la respiración en cambios de frecuencia eléctricamente medibles.

- VCO.

Oscilador que varía la frecuencia de trabajo (sub.-portadora) en función a los cambios del voltaje que provienen del Pneumografo y del amplificador ECG.

- Transmisor FM.

En esta etapa se mezclan las dos señales fisiológicas y se

modulan en frecuencia, para luego ser transmitidas al espacio.

- Receptor FM.

Es el encargado de amplificar la señal proveniente del transmisor FM, para que tenga un nivel adecuado y luego ser procesada.

- Filtro pasabajos 1 KHZ.

Esta etapa toma la señal modulada en frecuencia de la salida del receptor FM y deja pasar solo la componente de baja frecuencia de esta señal compuesta, para luego obtener la señal de las variaciones eléctricas del corazón.

- Filtro pasa altos 10 KHZ.

Es la etapa encargada de obtener la sub. Portadora, para ser procesada.

- PLL.

El objetivo aquí es obtener variaciones de voltaje en función de variaciones en la frecuencia provenientes de las sub. Portadoras.

- Filtro pasa bajos 50 HZ.

Aquí se obtiene la señal eléctrica proveniente del pneumografo, para luego ser procesada; en lazo opuesto se obtiene la variación eléctrica del corazón.

- Etapa de Muestreo.

Aquí se elige una señal a la vez por medio de la conmutación de un interruptor, para ser digitalizada y procesada por el computador.

- Digitalizador.

Esta etapa convierte las señales analógicas provenientes de la etapa de muestreo, en señales en formato binario, para que sean procesadas por el computador.

- Computador personal.

Es el encargado con ayuda de software de procesar las señales digitalizadas, para luego ser presentadas en la pantalla y poder generar alarmas si el estado de las señales así lo indica.

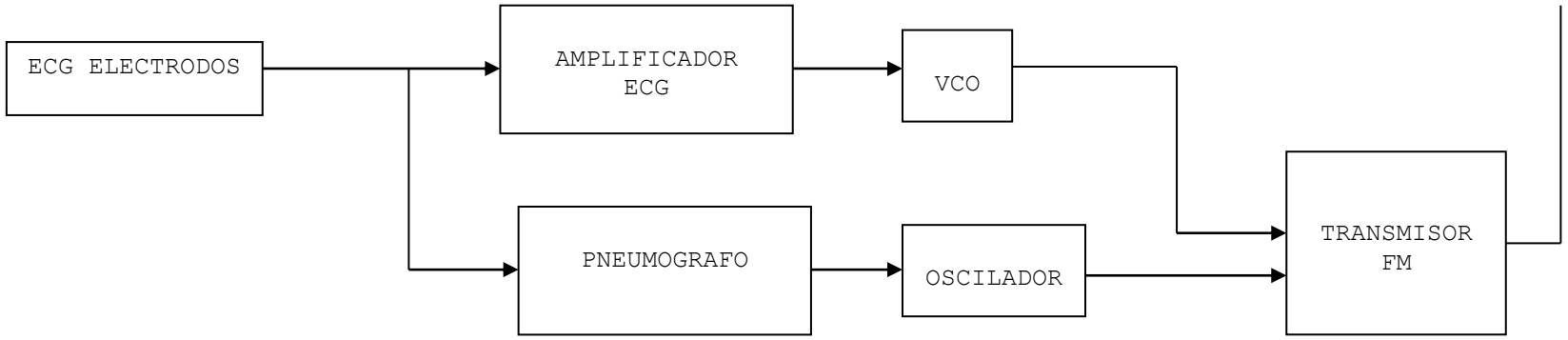


FIGURA 2.9 Diagrama en bloques de la estación remota.

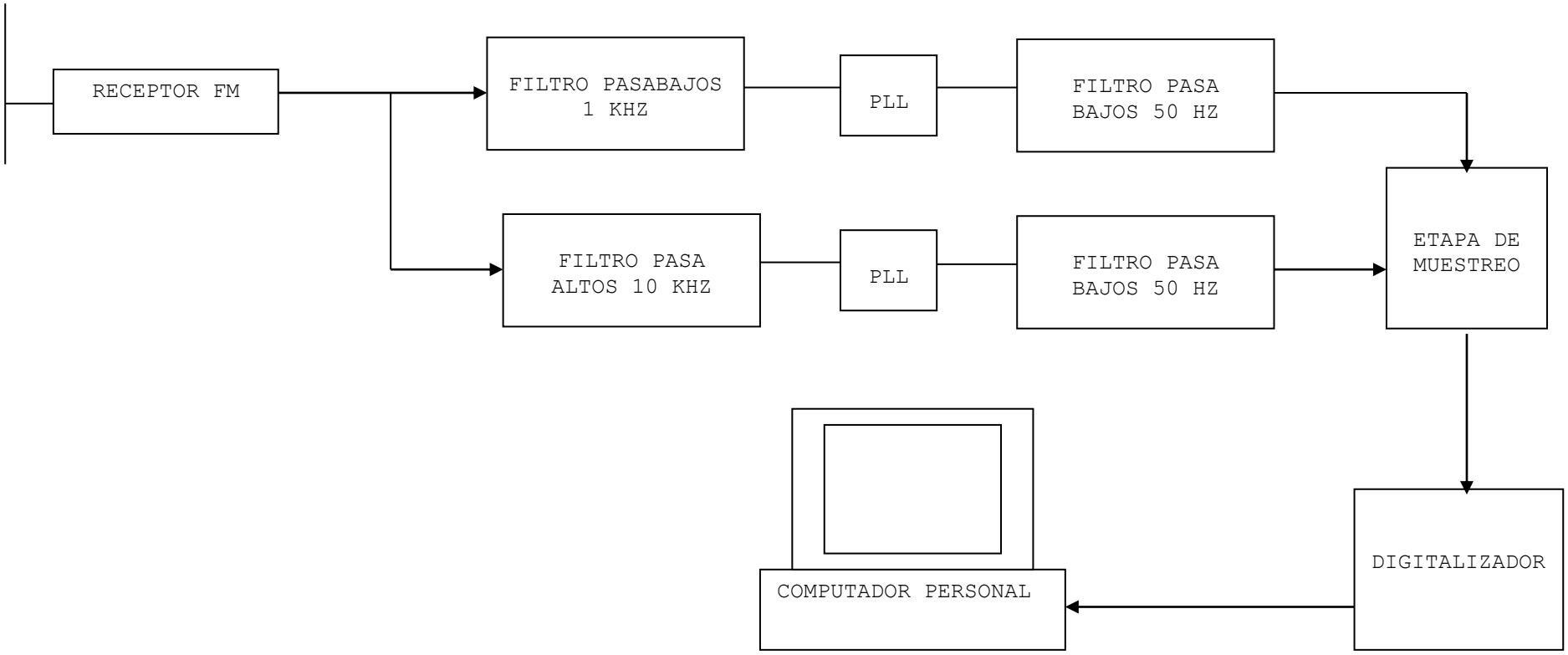


FIGURA 2.10 Diagrama en bloques de la estación central.

BENEFICIOS OBTENIDOS:

- Reducción del tiempo de respuesta para evitar problemas mayores, pues posee alarmas programables y la información del monitoreo es fácilmente accesible.
- Ahorro de costos de personal y equipos de monitoreo, pues puede monitorear ritmos anormales en los parámetros y activar alarmas de forma automática.
- Da la oportunidad de generar tecnologías propias para solventar necesidades locales.

CAPÍTULO III

ARQUITECTURA DEL SISTEMA

3.1 Hardware de la estación.

3.1.1 Multivibradores de Frecuencia Libre.

Llamados también multivibradores astables, son generadores de onda cuadrada que utilizan mecanismos biestables como Schmitt triggers y compuertas lógicas para producir señales de onda cuadrada, y un tiempo dependiente tal como una red RC o LC o un cristal de cuarzo para ordenar la frecuencia de oscilación.[11]

En la actualidad hay circuitos integrados especiales que se utilizan para construir multivibradores astables y monoestables entre ellos tenemos el IC 555 y el 74123.

3.1.1.1 El IC 555 como un multivibrador astable.

El IC es configurado para operación astable usando tres elementos externos, dos resistores y un capacitor.

La frecuencia de oscilación esta dada por la siguiente ecuación:

$$Fo = 1.44/((Ra+2Rb)*C).$$ *Ecuación 3.1*[11]

La duración del ciclo es expresada así,

*D(%)= 100 *(Ra+Rb)/(Ra+2Rb). Ecuación 3.2*[11]

Las ecuaciones muestran que las características de oscilación están en función de componentes externos y son independientes del voltaje de alimentación.

Si se desea, las características de tiempo de el 555 pueden ser moduladas por medio de la entrada de Control de Voltaje, pin 5 y es comúnmente referido como modulación de posición de pulso (PPM). Cambiando este voltaje desde su valor nominal de 2/3 Vcc puede resultar en tiempos de carga grandes y cortos, dependiendo de si su voltaje nominal (Vth) es incrementado o disminuido.[9]

En operación astable Vth varía el tiempo en alto (Th) de la siguiente manera.

*Th= (Ra+Rb)*C* ln((Vcc-Vth/2)/(Vcc-Vth)). Ecuación 3.3*[9]

3.1.1.2 El IC 74123 como multivibrador monoestable.

Este IC es un multivibrador DC re-disparable y la duración del pulso de salida es programado por la selección de una resistencia y capacitancia externa. Para un Cext>1000 pF, la duración del ancho del pulso es definido por:

Tw= 0.28 Rt* Cext*(1+0.7/Rt).* *Ecuación 3.4*[10]

Donde Rt es la resistencia externa y Cext el capacitor externo.

3.1.2 Demoduladores FM.

Los demoduladores de frecuencia modulada son circuitos dependientes de la frecuencia que producen una salida de voltaje la cual es directamente proporcional a la frecuencia instantánea de su entrada. Existen varios circuitos que son utilizados para demodular señales FM. En este proyecto se utiliza el PLL como demodulador, por su bajo costo y sencillez.

3.1.2.1 El PLL.

El diagrama de bloques muestra la constitución básica de un PLL.

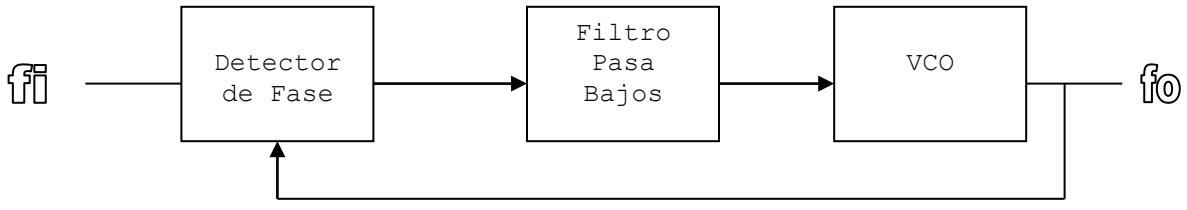


FIGURA 3.1 Diagrama de bloques de un PLL.

El VCO (oscilador controlado por voltaje) esta en oscilación libre, a una frecuencia que esta determinada por un resistor y capacitor externos. La frecuencia del VCO (fo) es retroalimentada al detector de fase donde es comparada con la frecuencia de la señal de entrada (fi). La salida del detector de fase es el error de voltaje. Este es un voltaje promedio de CD que es proporcional a la diferencia de frecuencia (fi- fo) y fase entre la señal de entrada y el VCO.

El error de voltaje es filtrado, eliminando ruido y componentes de alta frecuencia. Es a si como se completa el lazo. Una vez filtrado, el error de voltaje obliga a cambiar la frecuencia del VCO de tal manera que reduzca la diferencia entre la señal de entrada y la salida del VCO. Una vez el VCO comienza a cambiar la frecuencia, el lazo esta en estado de captura.

Este proceso continúa hasta que la frecuencia del VCO y la frecuencia de la señal de entrada, es exactamente la misma. En este punto, el lazo esta sincronizado o en fase cerrada. Esta acción se repite cada vez que la señal de entrada cambia en frecuencia cuando esta en cierre.

3.1.2.2 El NTE 989 como PLL.

Este IC es un circuito integrado de 14 terminales que puede ser conectado a componentes externos para formar un PLL.

Un resistor de temporización externo se conecta al Terminal 8 y un capacitor de temporizacion al Terminal 9; estos dos componentes determinan la frecuencia de operación libre del OCV, que viene dada por:

*Fo= 1/(3.7 Ro*Co). Ecuación 3.5*

El Terminal 7 es la salida de FM, en este Terminal se tiene una señal demodulada. Finalmente, esta señal pasa a otros amplificadores y a una bocina o altavoz en receptores FM comerciales.

La salida del Terminal 7 es filtrada con ayuda de un capacitor externo. La frecuencia de corte de este filtro esta dada por:

*Fc= 1/(2*3.1416*3500*Cf). Ecuación 3.6*

Donde Cf, es el valor del capacitor externo.

3.1.3 EL ADC DE APROXIMACIONES SUCESIVAS ADC0804[12].

El componente principal de la etapa de digitalización es el ADC0804, es el encargado de traducir un voltaje que este dentro de cierto rango a su equivalente digital.

Este es un IC CMOS de 20 pines que realiza conversión A/D usando el método de aproximaciones sucesivas.

Algunas de sus características más importantes son:

- Convierte el voltaje analógico de entrada a una salida digital de ocho bits. Las salidas digitales tienen búferes de triestado.
- Tiene un circuito interno generador de reloj que produce una frecuencia de f=1/(1.1RC)(ecuación 3.7), donde R y C son valores de componentes conectados externamente.
- Si se usa una frecuencia de reloj de 606 Khz. el tiempo de conversión es de aproximadamente 100*10E-06 s.
- selección de chip CS. Esta entrada debe estar en su estado activo en bajo para que las entradas RD y WR tengan algún efecto.
- LEER RD. Esta entrada se usa para habilitar los búferes de salidas digitales.
- ESCRIBIR WR. Se aplica un pulso bajo a esta entrada para señalar el inicio de una nueva conversión.

- INTERRUPCION INTR. Esta señal de salida pasará a alto al inicio de una conversión y retornará a bajo para señalar el fin de la conversión. En realidad es una señal de salida de fin de conversión, y le avisa al computador cuando los datos están listos para ser leídos.
- Vref/2. Esta es una entrada opcional que se puede usar para reducir el voltaje interno de referencia y por lo tanto cambiar el intervalo analógico de entrada que el convertidor puede manejar. Cuando esta entrada no esta conectada permanece en 2.5 V (Vcc/2) ya que Vcc se esta usando como el voltaje de referencia.
- CLK OUT. Para utilizar el reloj interno se conecta una resistencia a este pin y la señal de reloj aparece en este.
- CLK IN. Se usa para la entrada externa de reloj o para una conexión de un capacitor cuando se usa el reloj interno.

3.1.4 El Puerto Paralelo Mejorado (EPP) [13].

El puerto paralelo en el modo EPP, tiene dos estándares EPP 1.7 y EPP 1.9, las diferencias entre ambos afectan a los mecanismos de diferente manera. El EPP tiene un ciclo de transferencia en el orden de 500KB/S a 2MB/S. Esto se consigue permitiéndole al hardware del puerto, generar todas las señales de control de la comunicación.

Cuando se usa el puerto paralelo en modo EPP, algunas líneas toman diferentes nombres y tareas, a continuación se presenta

una tabla con la lista de esos pines(la función de los pines varia de acuerdo al modo de operación).

Pin	Compatibility mode	Nibble mode	Byte mode	EPP mode	ECP mode
1	nStrobe	HostClk	HostClk	nWrite	HostClk
2	Data 1	Data 1	Data 1	AD1	Data 1
3	Data 2	Data 2	Data 2	AD2	Data 2
4	Data 3	Data 3	Data 3	AD3	Data 3
5	Data 4	Data 4	Data 4	AD4	Data 4
6	Data 5	Data 5	Data 5	AD5	Data 5
7	Data 6	Data 6	Data 6	AD6	Data 6
8	Data 7	Data 7	Data 7	AD7	Data 7
9	Data 8	Data 8	Data 8	AD8	Data8
10	nAck	PtrClk	PtrClk	Intr	PeriphClk
11	Busy	PtrBusy	PtrBusy	nWait	PeriphAck
12	PError	AckDataReq	AckDataReq	User defined 1	NAckReverse
13	Select	Xflag	Xflag	User defined 3	Xflag
14	nAutoFd	HostBusy	HostBusy	nDStrb	HostAck
15	nFault	nDataAvail	NDataAvail	User defined 2	NPeriphRequest
16	nInit	nInit	NInt	nInt	NReverseRequest
17	nSelectIn	1284 Active	1284 Active	nAStrb	1284 Active
18	Pin 1 (nStrobe) ground return				
19	Pins 2 and 3 (Data 1 and 2) ground return				
20	Pins 4 and 5 (Data 3 and 4) ground return				
21	Pins 6 and 7 (Data 5 and 6) ground return				
22	Pins 8 and 9 (Data 7 and 8) ground return				
23	Pins 11 and 15 ground return				
24	Pins 10, 12, and13 ground return				
25	Pins 14, 16, and 17 ground return				

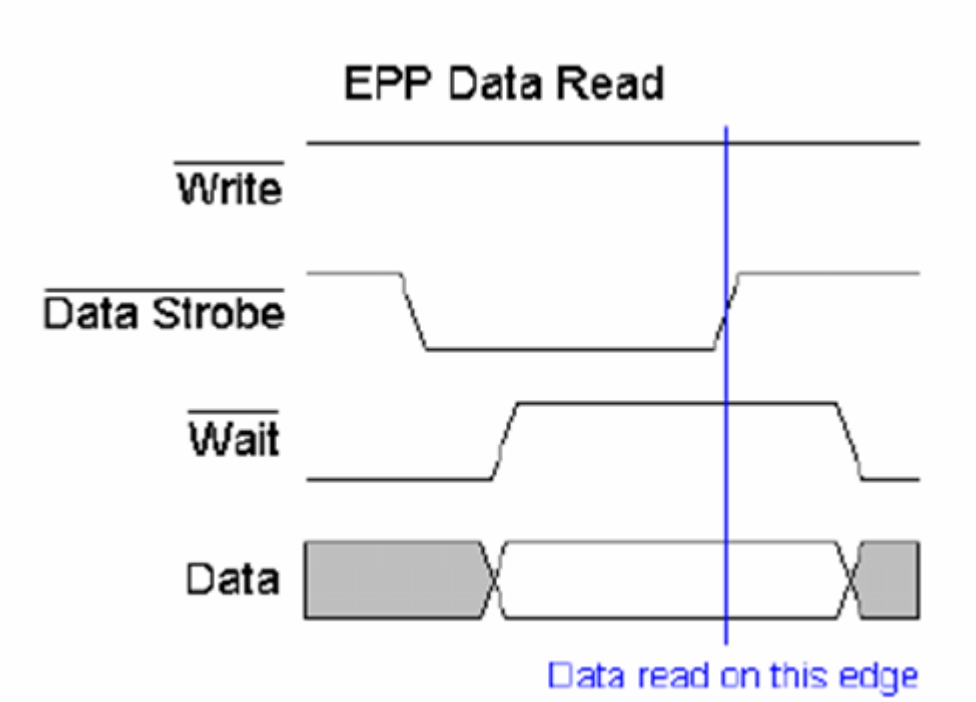
TABLA 3.1.4 Nombre de pin de acuerdo a modo de operación del puerto paralelo en un conector IEEE 1284-A (Ver anexo D).

Para hacer un intercambio de datos validos se debe cumplir el EPP Handshake ver figura 3.1.4. Como el hardware del puerto es el que realiza todo el trabajo, a si es el hardware externo el que debe responder a las señales y no algún software de intercambio de información.

Para iniciar un ciclo EPP, se deben configurar antes el registro de control del puerto estándar paralelo con xxxx0100, para poner las líneas nAddress Strobe, nData Strobe, nWrite y nReset inactivas.

Otra consideración importante es chequear y limpiar el Time Out BIT de registro de estado del EPP, de lo contrario los datos leídos o escritos son incorrectos.

EPP DATA READ CYCLE



1. **Program reads EPP Data Register. (Base + 4)**
2. nData Strobe is asserted if Wait is Low (O.K. to start cycle)
3. **Host waits for Acknowledgment by nWait going high**
4. Data is read from Parallel Port Pins.
5. **nData Strobe is de-asserted.**
6. EPP Data Read Cycle Ends.

FIGURA 3.1.4. Ciclo de lectura del puerto paralelo.

CAPÍTULO IV. EL PROTOTIPO

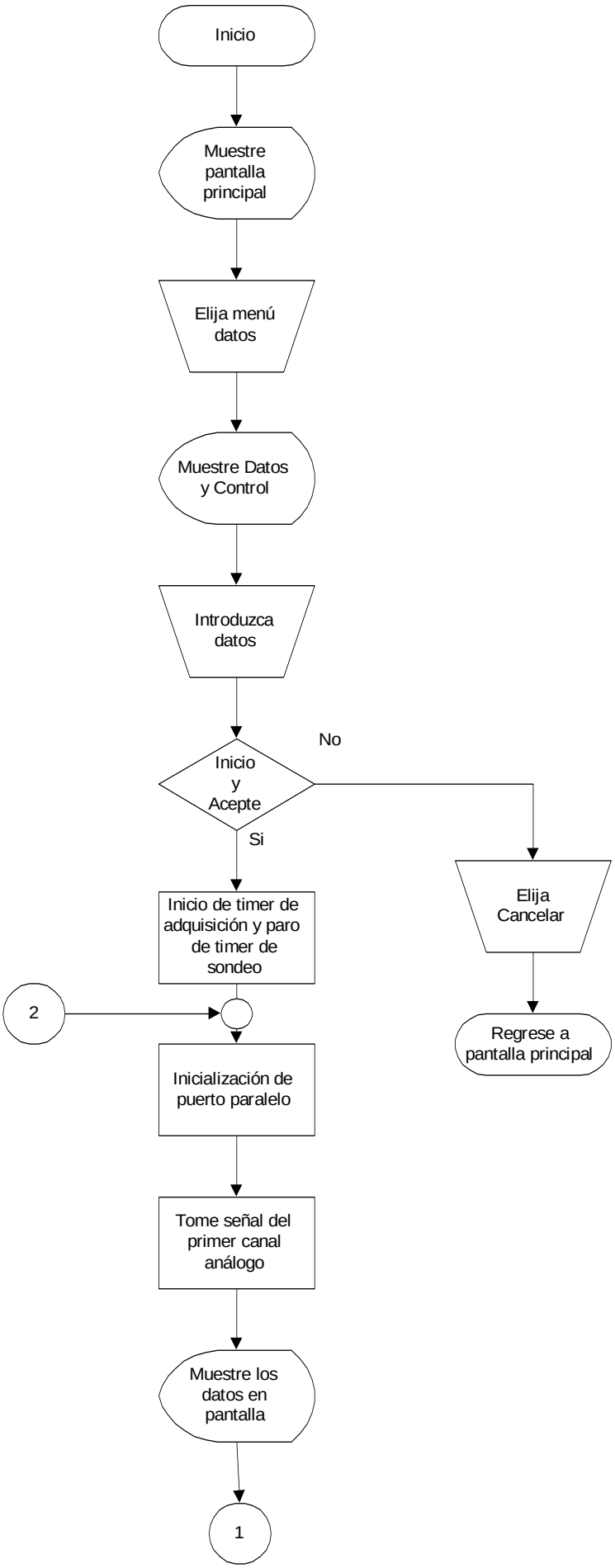
4.1 EL SOFTWARE DE LA ESTACION CENTRAL.

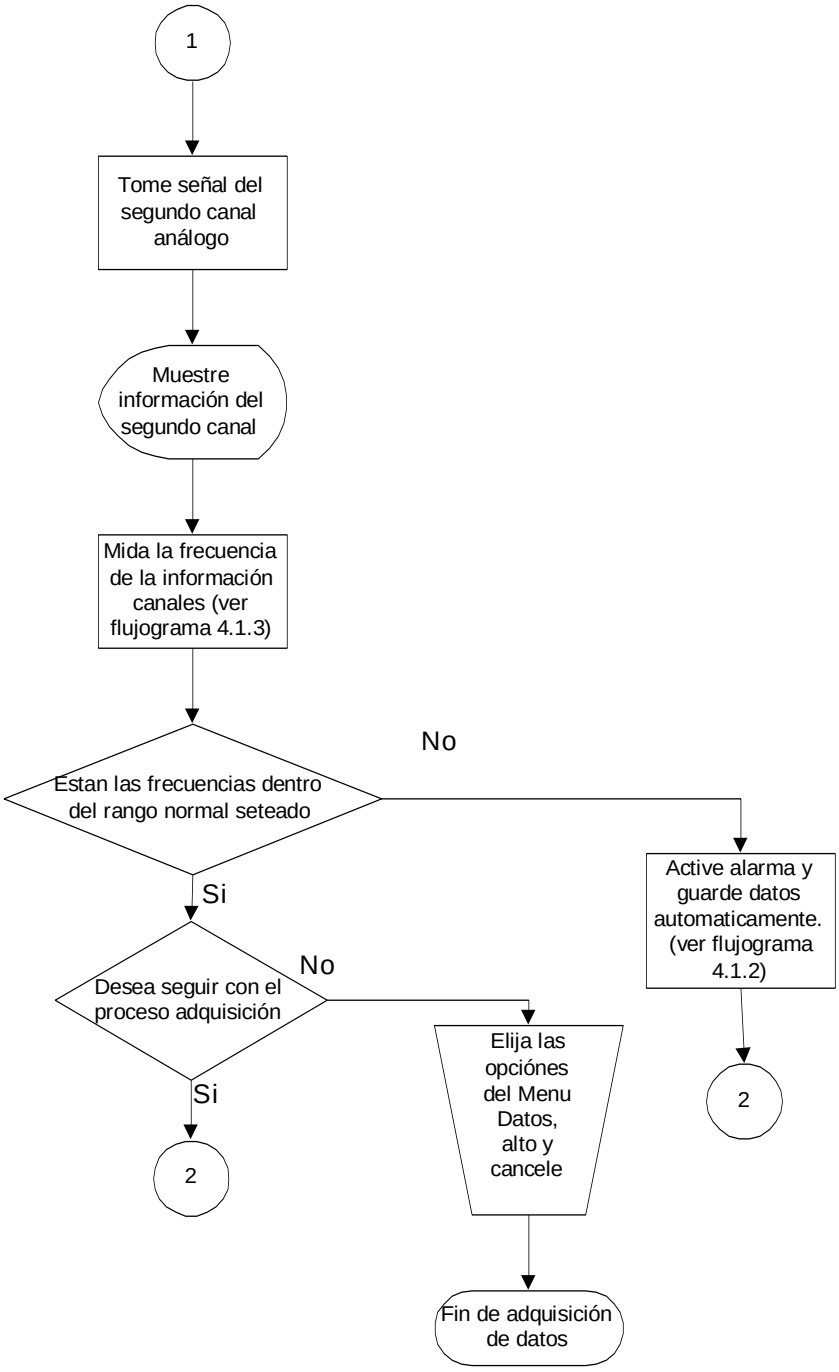
El software de la estación esta desarrollado en el lenguaje de programación Visual C++ 5.0. Las funciones principales de este programa son:

- Brindar un medio de adquisición, almacenamiento y presentación de información relativa al paciente.
- Gestionar la comunicación del puerto paralelo en modo EPP y circuitos periféricos.
- Monitorear la frecuencia de los parámetros adquiridos.

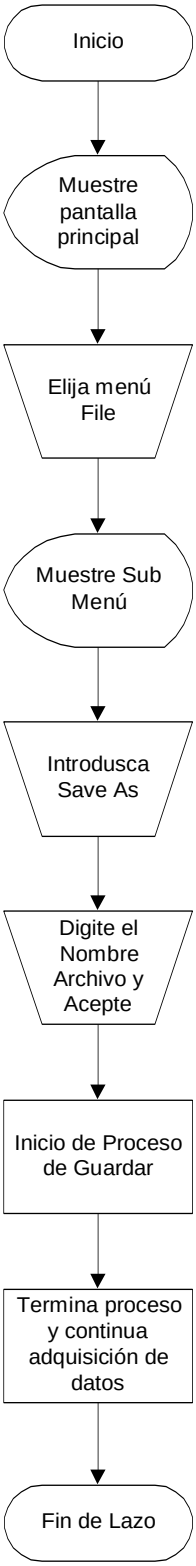
A continuación, se presentan los flujogramas del programa.

Flujograma 4.1.1 operación básica del programa y medición de frecuencia.

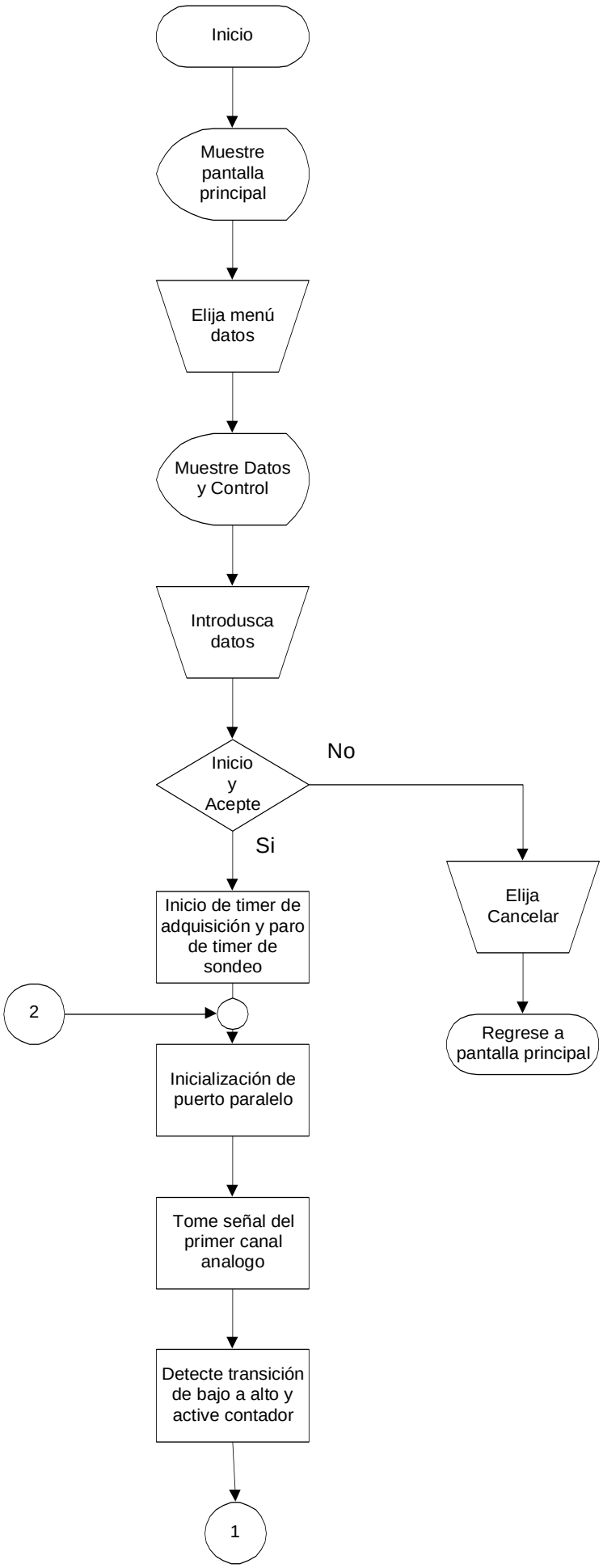


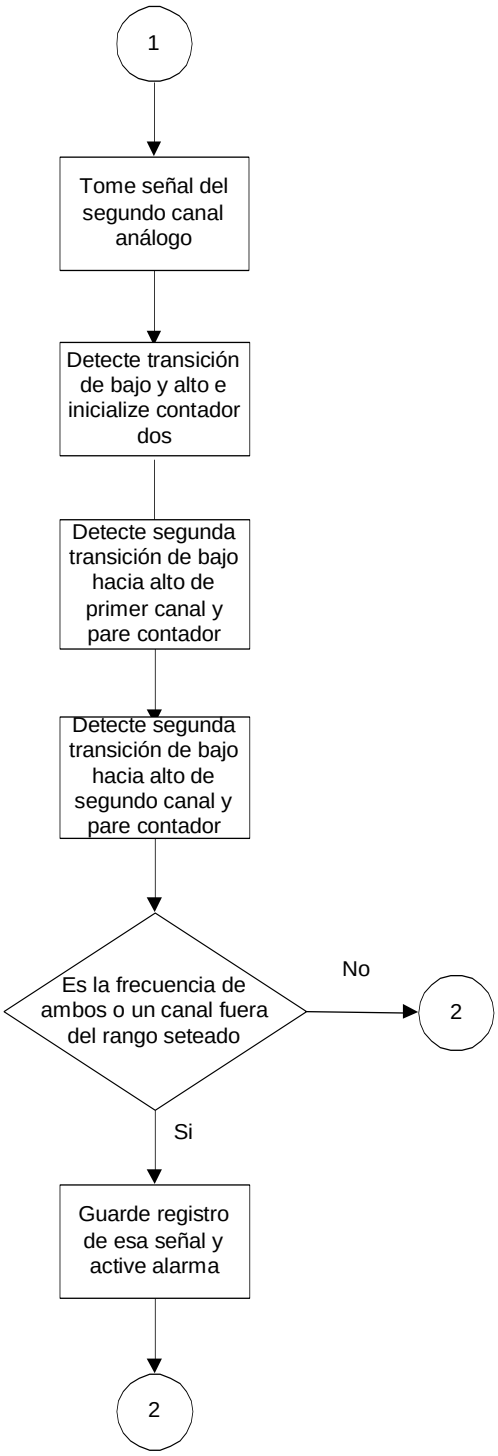

1
Tome señal del segundo canal análogo
Muestre información del segundo canal
Mida la frecuencia de la información canales (ver flujograma 4.1.3)
Estan las frecuencias dentro del rango normal seteado
No
Si
Active alarma y guarde datos automaticamente. (ver flujograma 4.1.2)
2
Desea seguir con el proceso adquisición
No
Si
Elija las opciónes del Menu Datos, alto y cancele
2
Fin de adquisición de datos

Flujograma 4.1.2 Almacenamiento normal de información.



Flujograma 4.1.3 medición de frecuencia.





El funcionamiento de la fracción del programa correspondiente al flujograma 4.1.1 es el siguiente: Al ejecutar el programa con extensión ¨exe¨ se crea sobre una ventana un fondo con cuadriculas y sobre él un menú para poder activar las funciones del programa. Seleccionando del menú ¨Datos¨, se

crea un cuadro de diálogo con campos de información del paciente y con botones de control de inicio o fin de adquisición. Si el usuario introduce la información del paciente y presiona el botón inicio, se activa el ciclo sin fin, de adquisición de datos; si por el contrario se elige terminar, el programa finaliza cualquier ciclo de adquisición y dando aceptar, desaparece el cuadro de diálogo.

Siguiendo la secuencia, luego de dar el comando de inicio, se inicia un timer, el cual es el que da el tiempo de muestreo en este programa, se inicializa el puerto paralelo en modo EPP y finalmente se toma la primera muestra del primer canal, en este punto se puede mostrar en pantalla.

Luego de esto el programa ordena al interruptor digital, conmutar, para tomar el dato del canal número dos y finalmente mostrarlo. Después del fin del primer ciclo, debo esperar un segundo ciclo, para poder obtener la frecuencia de las dos señales y poder a si procesarlas. Si las frecuencias están dentro del rango considerado como normal, el programa sigue un ciclo de muestreo, tras otro a menos que las frecuencias salgan de rango, activando un ciclo de almacenamiento y una señal de alarma; para finalmente seguir con la adquisición y despliegue de información.

Con respecto al funcionamiento de la fracción descrita en el flujograma 4.1.2 tenemos el comportamiento siguiente:

Luego de iniciar el programa ejecutable la función de ¨Save

As¨ puede accesarse en todo tiempo mientras no se este dentro del cuadro de diálogo de ¨Datos¨. Para ello se elige, del menú principal ¨Archivo¨, seguido el submenú Guarde Como y se da aceptar, con esto aparece un cuadro de diálogo donde se debe elegir un nombre y ubicación del archivo a guardar, para finalmente elegir ¨Aceptar¨ dentro de este cuadro, posteriormente se inicia el ciclo de almacenamiento, y termina asi el ciclo.

Finalmente una de las partes mas delicadas es la descrita por el flujograma 4.1.3, el cual funciona de la siguiente manera: Luego de iniciado un ciclo de lectura con el contador de frecuencia a cero, se detecta en cada canal una transición de bajo hacia alto, para poder iniciar el contador de la frecuencia respectiva, luego se finaliza en la siguiente transición respectiva de bajo hacia alto, deteniendo la cuenta y comparándola con un rango normal de frecuencias para cada parámetro fisiológico procesado; si los rangos de frecuencia están dentro de lo normal, el proceso continúa sus ciclos de adquisición y determinación de frecuencias, si no es así, se guarda una fracción de la señal anormal y se activa una alarma, para finalmente, seguir con los ciclos de adquisición y determinación de frecuencias.

Cabe mencionar, que el ritmo respiratorio se obtiene a través de una ecuación que relaciona al ritmo cardiaco. Los datos utilizados para elaborar la ecuación se limitan a una

condición normal de actividad y sin ninguna patología relacionada. El Ritmo Respiratorio esta entre 15- 20 BPM [18] y el Ritmo Cardiaco entre 60 y 100 BPM [18]. Graficando estos datos se obtuvo una relación logarítmica, por medio de la cual se logro la siguiente ecuación.

$$FR = K \ln FC \qquad \textit{Ecuación 4.1}$$

Se tomaron datos del punto medio de ambos rangos para la sustitución de valores y obtener K.

17.5 BPM= K ln 80 BPM

K= 3.99.

4.2 OPERACION DEL PROTOTIPO.

En este capítulo se detalla el funcionamiento de cada etapa del circuito y su aporte a las etapas adyacentes para el logro de funcionamiento óptimo del sistema.

4.2.1 ETAPA DEL RECEPTOR FM.

Esta cuenta con un receptor de un micrófono inalámbrico comercial FM y se encarga de proporcionar la señal compuesta por la mezcla de las señales fisiológicas en un rango máximo de 12 HZ y 10 KHZ.

4.2.2 FILTRO PASA BAJOS DE 1 KHZ DE LA ESTACION CENTRAL.

Este filtro se encarga de dejar pasar la señal fisiológica con la sub. Portadora de 1KHZ y esta compuesto por un filtro pasa bajos con circuito operacional.

Las ecuaciones para el diseño son las siguientes:

$$Fo = 1/(2*\pi*(mn)^{1/2}*R*C). \qquad \text{Ecuación 4.2}[9]$$

$$Q = (mn)^{1/2}/(n+1). \qquad \text{Ecuación 4.3}[9]$$

$$F\ actual = fc/f\ tabulada. \qquad \text{Ecuación 4.4}$$

Si R*= 10KΩ y C*=19.16ηF. Tengo fo2= 529 Hz.

n*= 2.4649.

Ahora, hago C≈ C* y n ≥ n*.

C= 0.015μF.

K= 2.868 y m= 2.4622.

Recalculando las resistencias, me queda que:

R= 7.5KΩ y mR= 18KΩ.

4.2.3 PLL LAZO 1:

Este circuito se encarga de traducir la información que viene en forma de variaciones de frecuencia del filtro anterior a su formato original (variaciones de voltaje) y ser digitalizadas.

Los componentes para el oscilador se calculan a si:

Si c= 0.1 μF

*RoCo=1/(fo*3.7).* *Ver ecuación 3.5*

Ro≈ 2.7 KΩ.

*Cf= 1/(2*π*3.5KΩ*fc); si fc=300 Hz.* *Ver ecuación 3.6*

Cf= 0.15 μF.

4.2.4 FILTRO PASA ALTOS DE 10 KHZ.

Este filtro se encarga de dejar pasar la señal fisiológica con la sub. Portadora de 8 KHZ y esta compuesto por un filtro pasa altos con circuito operacional.

Las ecuaciones para el diseño son las siguientes:

*Fo= 1/(2*π*(mn)^½*R*C).* *Ver Ecuación 4.2*

Q= (mn)^½/(n+1). *Ver Ecuación 4.3*

F actual= fc/f tabulada. Ver Ecuación 4.4

Los filtros se harán con capacitores iguales n=1.

De tablas se encuentra que f01=10.0704 y Q1= 3.559.

m = 50.665, supongo R= 10 KΩ.

*C= 1/(2*π*(mn)^½*R*fo). Ver Ecuación 4.2*

c= 222.0298 ρF≈ 220 ρF.

mR = 50.665*10 KΩ≈ 510 KΩ.

Para la Etapa II: fo2 =18.9 Khz. y Q2= 0.785.

m = 2.4649, supongo R= 10 KΩ.

C ≈ 510 ρF.

Re- calculando R's. R=10.51 KΩ≈ 10 KΩ.

mR ≈ 24 KΩ.

Para la amplificación de la señal se utiliza un filtro con fc de 10 Khz. y una ganancia de 50.

fo = 1/(2πR1C), la ganancia viene dada por: Ho= -R2/R1.

Si supongo C= 0.001 μF.

R1 ≈ 15 KΩ.

R2= 50*15 KΩ= 750 KΩ.

4.2.5 PLL LAZO 2 (NE 567):

Este circuito se encarga de traducir la información que viene en forma de variaciones de frecuencia del filtro anterior a su formato original y ser digitalizadas.

Por datos prácticos se encuentra f= 8 KHz.

Si C1= 0.02 μF.

De, R1= 1.1/ (fo*C1). E*cuación 4.5*

R1= 6.8 KΩ.

C2= 0.1 μF. Para una fc= 300 Hz.

C3= 2C2. *Ecuación 4.6*

C3= 0.2 μF.

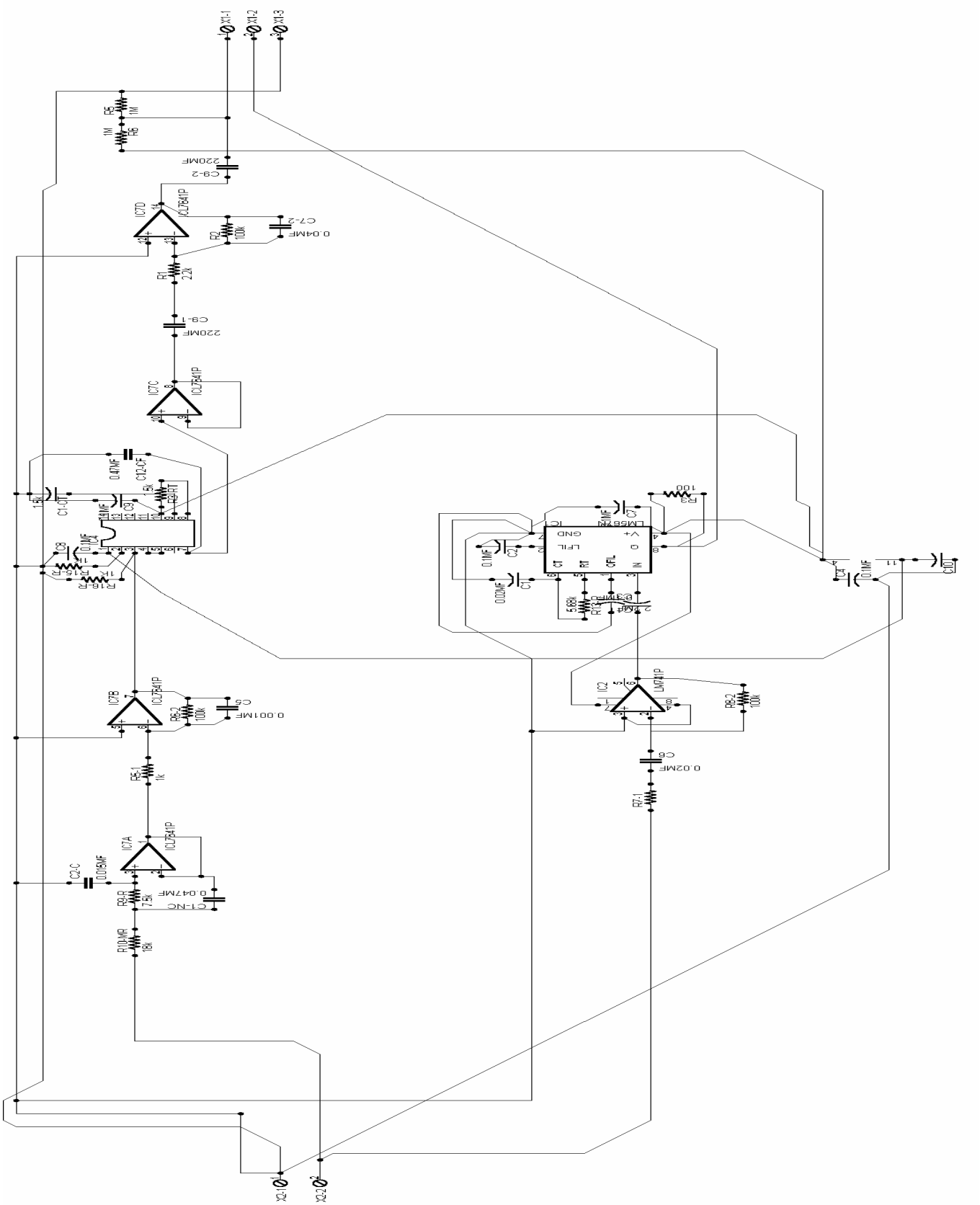


FIGURA 4.1 Demodulador FM

4.2.6 ETAPA DE MUESTREO.

Compuesto con un interruptor analógico y control digital por medio de la señal del pin 16 del puerto paralelo del computador, es el encargado de tomar una muestra a la vez y brindarla a la entrada del ADC.

4.2.7 DIGITALIZADOR.

El corazón de esta etapa es un ADC de aproximación sucesiva, el cual convierte las variaciones de voltaje de la señal muestreada, a un formato binario para ser enviadas al puerto paralelo del computador, por medio de la interacción de varias señales de control, proveniente del computador y activado por software.

Prácticamente el circuito responde o trabaja siguiendo un ciclo básico de lectura en el modo EPP del puerto paralelo.

La secuencia de funcionamiento es la siguiente:

1) En el ciclo de lectura la línea Write (1), cambia a uno lógico, por lo cual esto se invierte y habilita la salida de tres estados del ADC.
2) Luego la línea DATA STROBE (14), se va a cero, lo cual activa a un circuito monoestable de transición negativa, cuya salida esta conectada a la entrada Write del ADC (3). Con esto se inicia el ciclo de conversión del ADC.
3) La línea Wait del puerto paralelo (11) debe haber estado en cero lógico para que todo este ciclo comenzara.

Posteriormente al final de la conversión del ADC la salida INTR (5) del ADC, proporciona un uno lógico a la línea Wait, indicando que el ciclo de conversión finalizo.

4) Después de la transición de cero a uno de la línea Wait (11), la circuiteria del puerto lee el dato de entrada y en ese mismo instante vuelve la línea Data Strobe (14) a uno lógico.
5) Finalmente, la línea Wait debe volver a cero, para que el próximo ciclo se inicie. Esto se logra conectándola a la salida INTR invertida del ADC.

El control del canal leído, es manejado por el programa a través de la línea de Reset (16) del puerto paralelo. En cada ciclo se cambia de estado y de esta manera se alimenta al circuito de control del interruptor analógico con control digital.

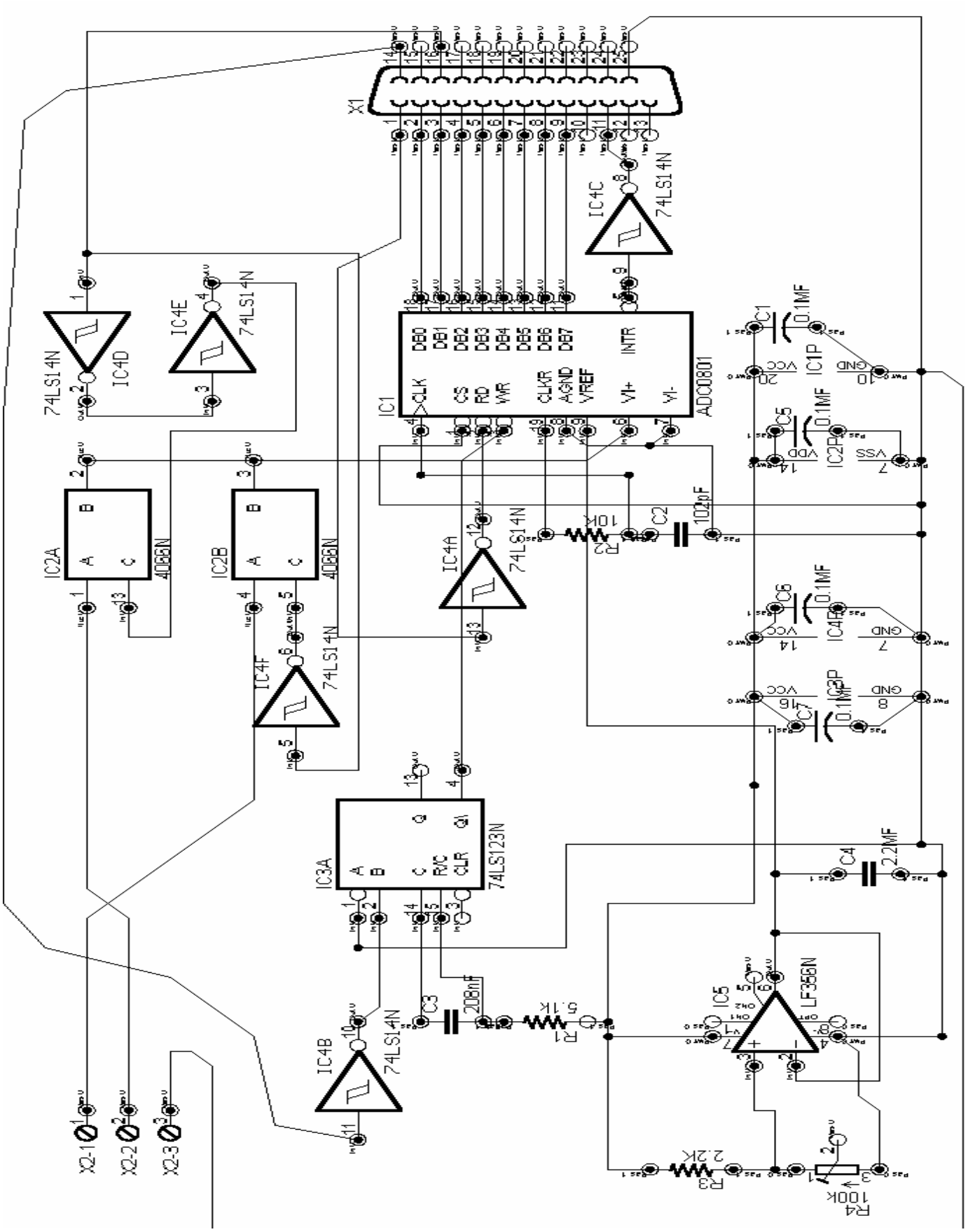


FIGURA 4.2 Adquisidor de datos

4.2.8 AMPLIFICADOR ECG.

Se encarga de aislar al paciente del equipo eléctrico de la estación remota, y amplificar el nivel de la actividad eléctrica del corazón con el fin de poder manipularla con efectividad.

Esto se realiza con circuitos operacionales con características especiales, resistencias limitadoras y diodos de protección, para cumplir con los parámetros de corriente de fuga a través del paciente del estándar HEI-95 (ver anexo A).

Las ecuaciones para el operacional de instrumentación son las siguientes:

Para etapa de los dos operacionales de entrada.

$$A1 = 2*R2/R1 + 1. \quad \textit{Ecuación 4.7}$$

Para la etapa del amplificador de salida.

$$A2 = R4/R3. \quad \textit{Ecuación 4.8}$$

Introduciendo valores:

Si R1= 390+/- 100 Ω.

R2= 5 KΩ.

R3= 500 Ω.

R4= 20 KΩ.

$A\tau = A1 * A2 \approx 27 * 40 = 1080.$

Con estos valores se logra una ganancia cercana a muchos prototipos de ECGs.

4.2.9 PNEUMOGRAFO.

A través de esta etapa se obtiene una señal que refleja la frecuencia respiratoria, esta etapa debe estar aislada del paciente cumpliendo con el parámetro de corriente de fuga según el estándar HEI-95 (ver anexo A).

El elemento sensor acá es un termistor y a través de un puente de weston, un amplificador operacional que trabaja en la región de saturación y un multivibrador astable. Se logra traducir las variaciones de temperatura del aire respirado en variaciones eléctricas, para que sean luego transmitidas.

4.2.10 VCO (IC 555).

Existe uno en la etapa remota y su fin es el de variar el ancho de banda de la señal de baja frecuencia, para poder de esta manera transmitirla, por medio de un equipo transmisor comercial.

Para el primer canal, se elige f1= 1 KHz.

(Ra+Rb)=1.44/(fC); si elegimos C= 0.1 μF y Ra= 750 KΩ. Ver Ecuación 3.1

Rb= 503 K$\Omega \approx 510$ KΩ.

4.2.11 MULTIVIBRADOR ASTABLE (IC 555).

Se compone de un IC 555, controlado por su pin de Reset. Para el segundo canal, se elige f2= 8 KHz.

Si, Rb=Ra= 10 $K\Omega \approx 10$ $K\Omega$. Utilizando Ecuación 3.1

Tengo que, C= 6 nF.

4.2.12 TRANSMISOR FM.

Este es un equipo de modulación FM comercial utilizado en los micrófonos inalámbricos, transistorizado y su objetivo es de transmitir la señal estéreo creada a partir de la combinación de las dos señales fisiológicas y lograr un alcance máximo de 10 mts, su frecuencia nominal de transmisión es de 125 MHz. Según su frecuencia la radiación emitida es no ionizante y según estudios realizados (ver anexo D) en el rango de 100 KHz a 10 GHz deben respetarse ciertos parámetros como limites en la tasa de absorción especifica de energía (SAR) y densidad de corriente, para prevenir tensiones corporales por calor y excesivo calor local tisular.

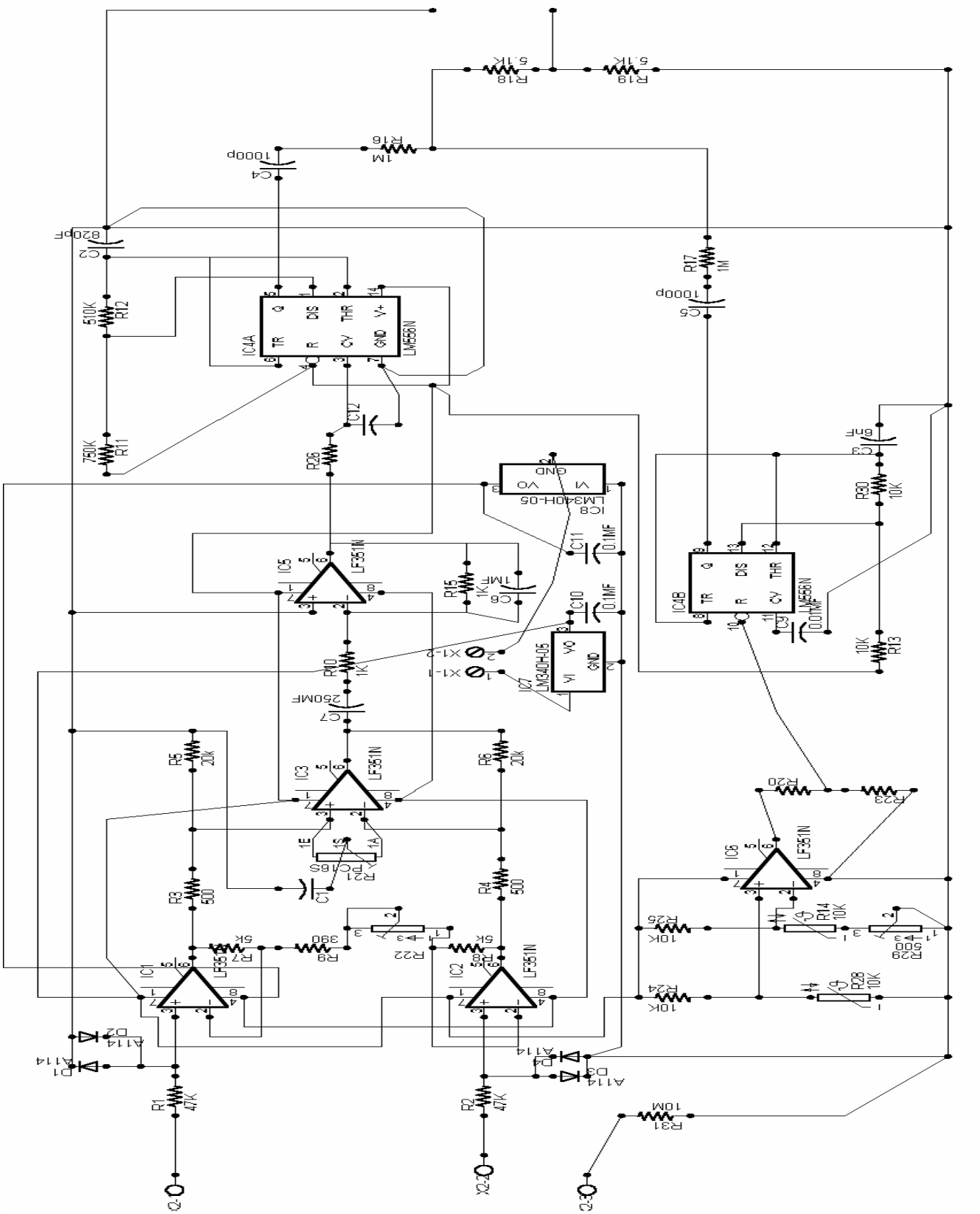


FIGURA 4.3 Procesador de señales

CAPÍTULO V. CONSTRUCCION DEL PROTOTIPO

5.1 CIRCUITO IMPRESO.

Para la elaboración de los circuitos en impreso se tomaron varias consideraciones como:

- Capacitores fijos para eliminación de ruidos e interferencias de la fuente de alimentación, según norma IEC 60384-14, 1993 [16]. Estos deben ser colocados lo más cerca al pin de alimentación. Se utilizaron capacitores de 0.1 MF.
- Anillos aterrizados en las entradas y salidas generales, para evitar el efecto de capacitancias parásitas (ver [14] y [15]).
- El grosor de la pista a tierra debe ser considerablemente ancha y abarcar lo máximo posible dentro del circuito, para evitar efectos antena. [17]

Se toma como base:

La resistividad del cobre = 1.723*10-08 Ω-m

L= 0.010 m

Ancho = 0.001 m

R= 1.723*10-08 Ω-m*0.01m/(0.01m*0.001m)=0.000017 Ω.

Ahora, tomo la potencia de un IC NE567= 300mW. Elijo una fuente de 5V. Obtengo la corriente total =300mW/5V=0.06Amp.

P cobre = 0.06A*0.06A*0.000017 Ω= 61 nW.

Lo cual nos da una pista para escoger un grosor mínimo de 1 mm de ancho en la pista y saber que no se me destruirá, ni afectará considerablemente al circuito. Otro factor a tomar en cuenta es que para las pistas de la cara superior del impreso se dificulta soldar, por lo cual se debe hacer lo mas ancha posible.

- Evitar en lo posible los ángulos agudos en las pistas. Pues aumentan la resistencia en las pistas y favorecen la interferencia por campo eléctrico.
- Tomar como base los 0.4 mm de distancia de separación entre partes de polaridad opuesta según la norma **IRAM 4220-1:1999**[1)] (ver tabla 4, apéndice A).

5.2 VISTAS DE LOS CIRCUITOS.

Básicamente se elaboraron tres circuitos impresos según los circuitos detallados en el capítulo anterior (ver las figuras 5.1 a 5.11), la escala es 1:1. Aquí se pueden apreciar fácilmente las curvas de las pistas, la distancia entre pistas es mayor a la considerada por la norma **IRAM 4220-1:1999**[1] y por último puede apreciarse fácilmente que la pista de tierra abarca siempre un área considerablemente mayor a las demás.

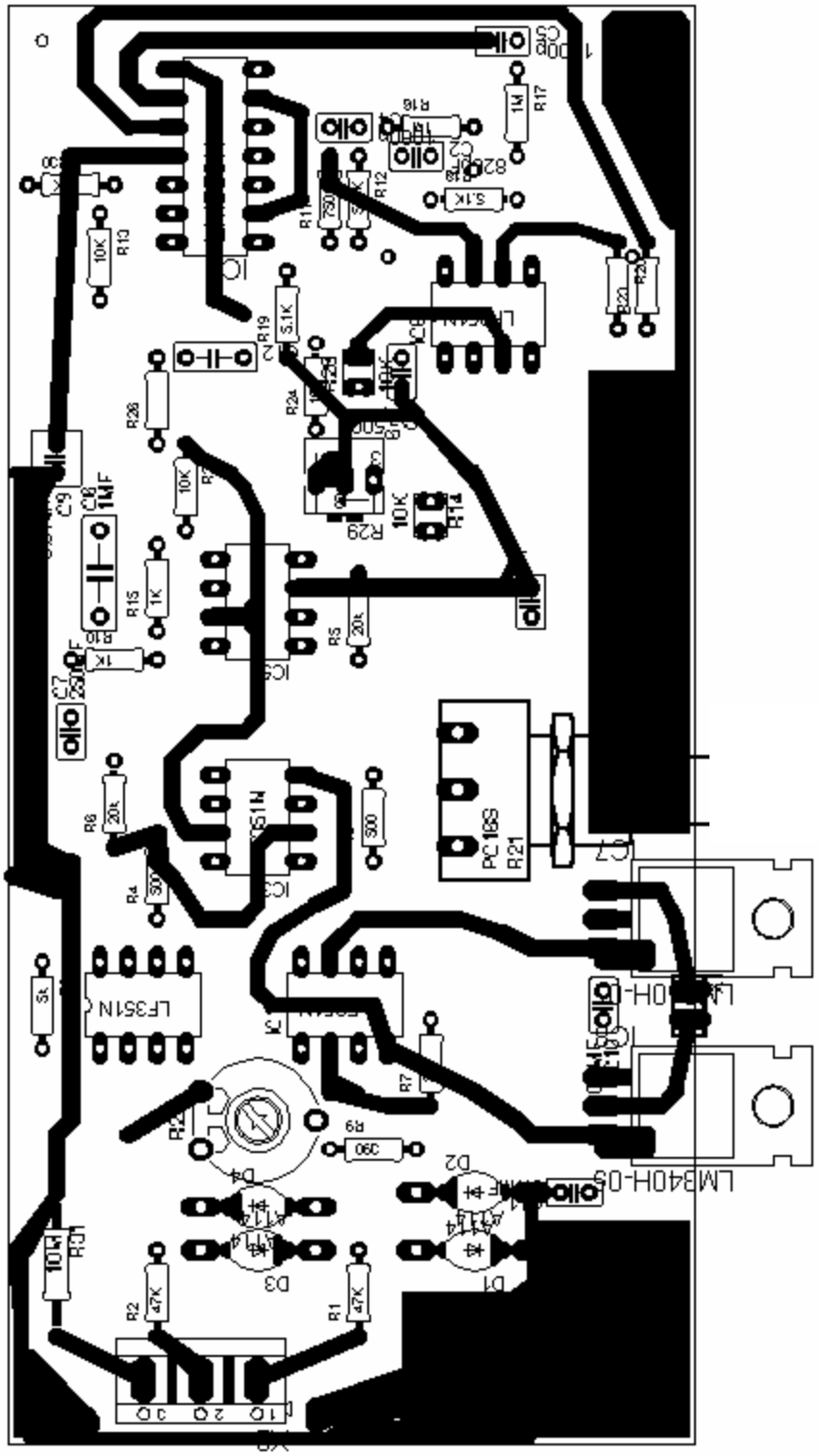


FIGURA 5.1 Circuito acondicionador de señales, vista de arriba.

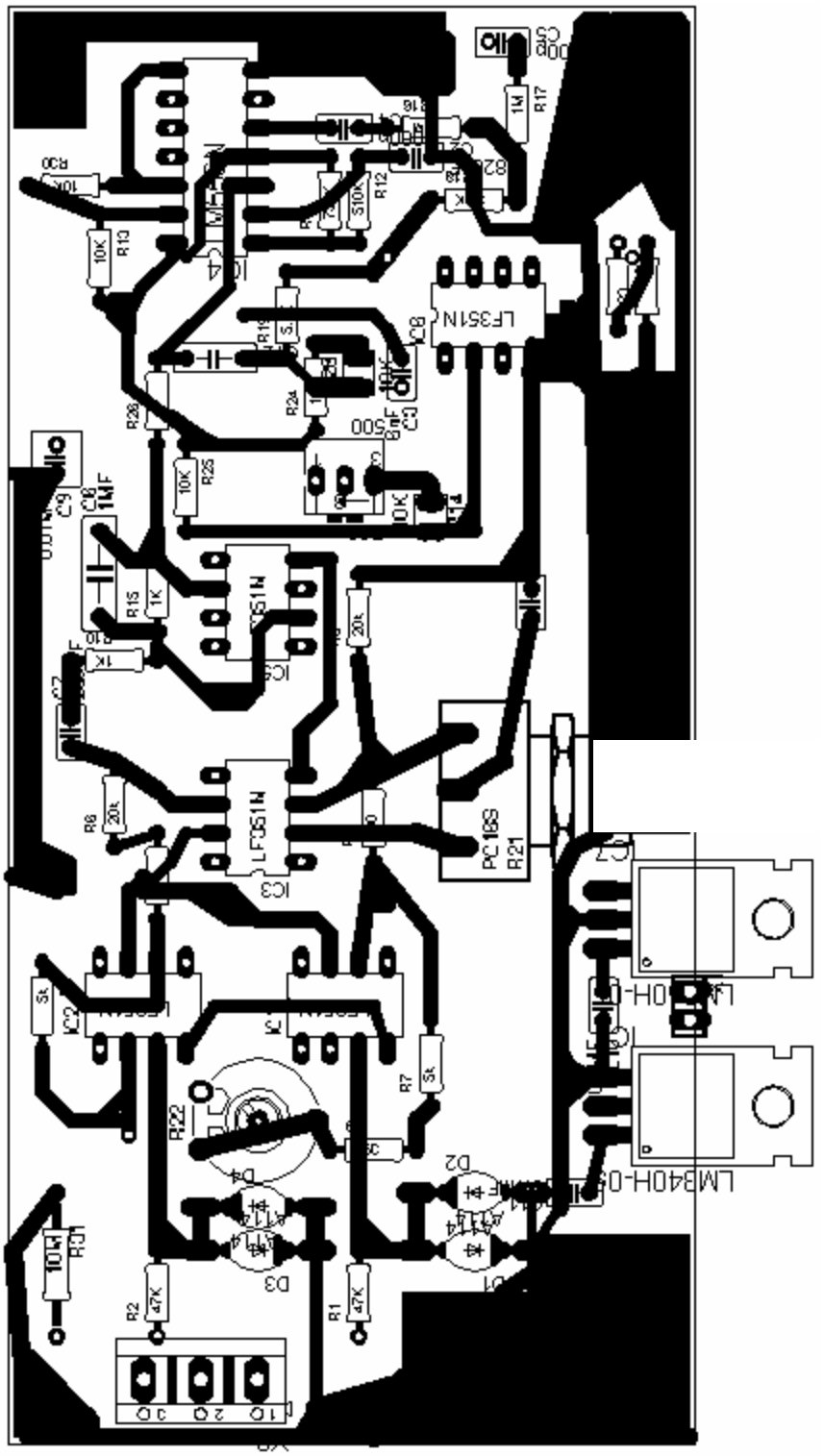


FIGURA 5.2 Circuito acondicionador de señales, vista de abajo.

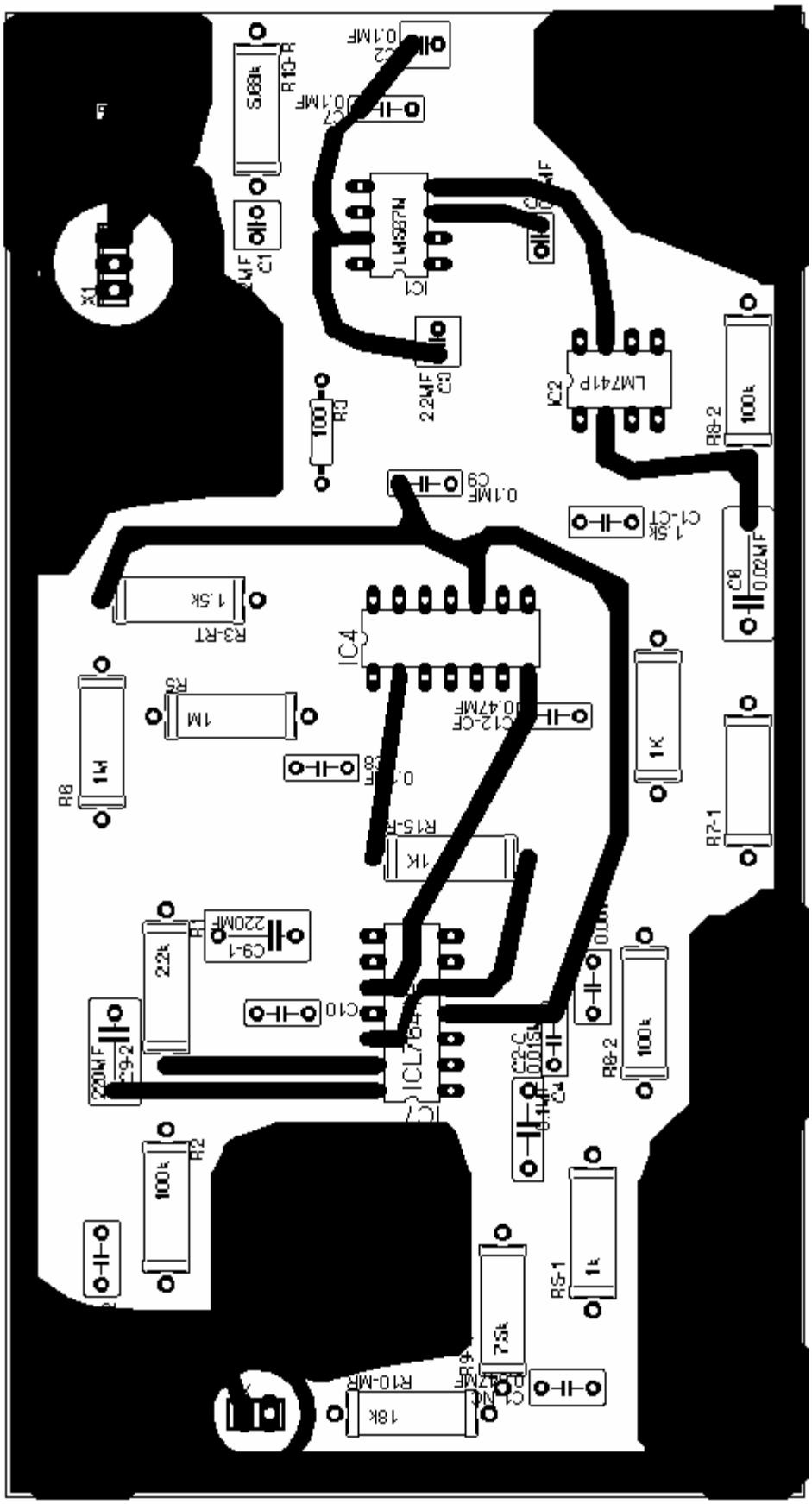


FIGURA 5.3 Circuito Demodulador, vista de arriba.

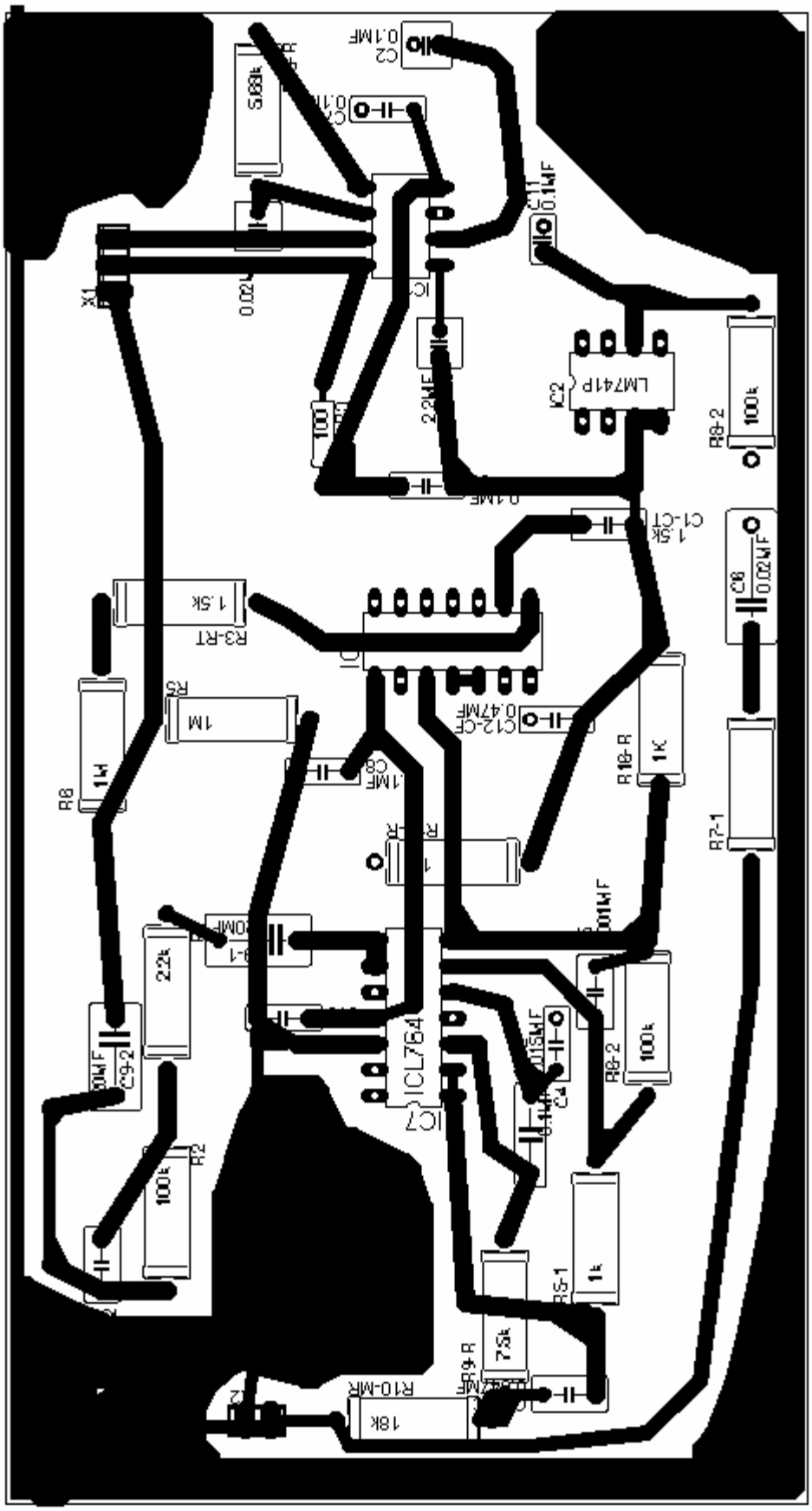


FIGURA 5.4 Circuito Demodulador, vista de abajo.

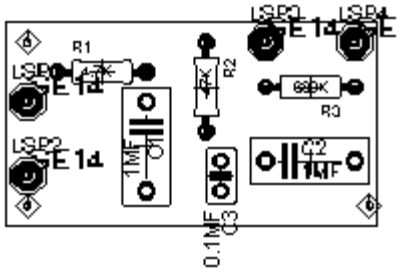


FIGURA 5.5 Filtro salida etapa ECG/ Adquisitor, lado de arriba.

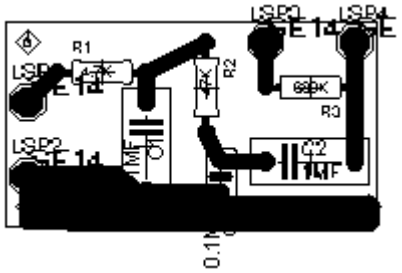


FIGURA 5.6 Filtro de salida ECG/ Adquisitor, lado de abajo.

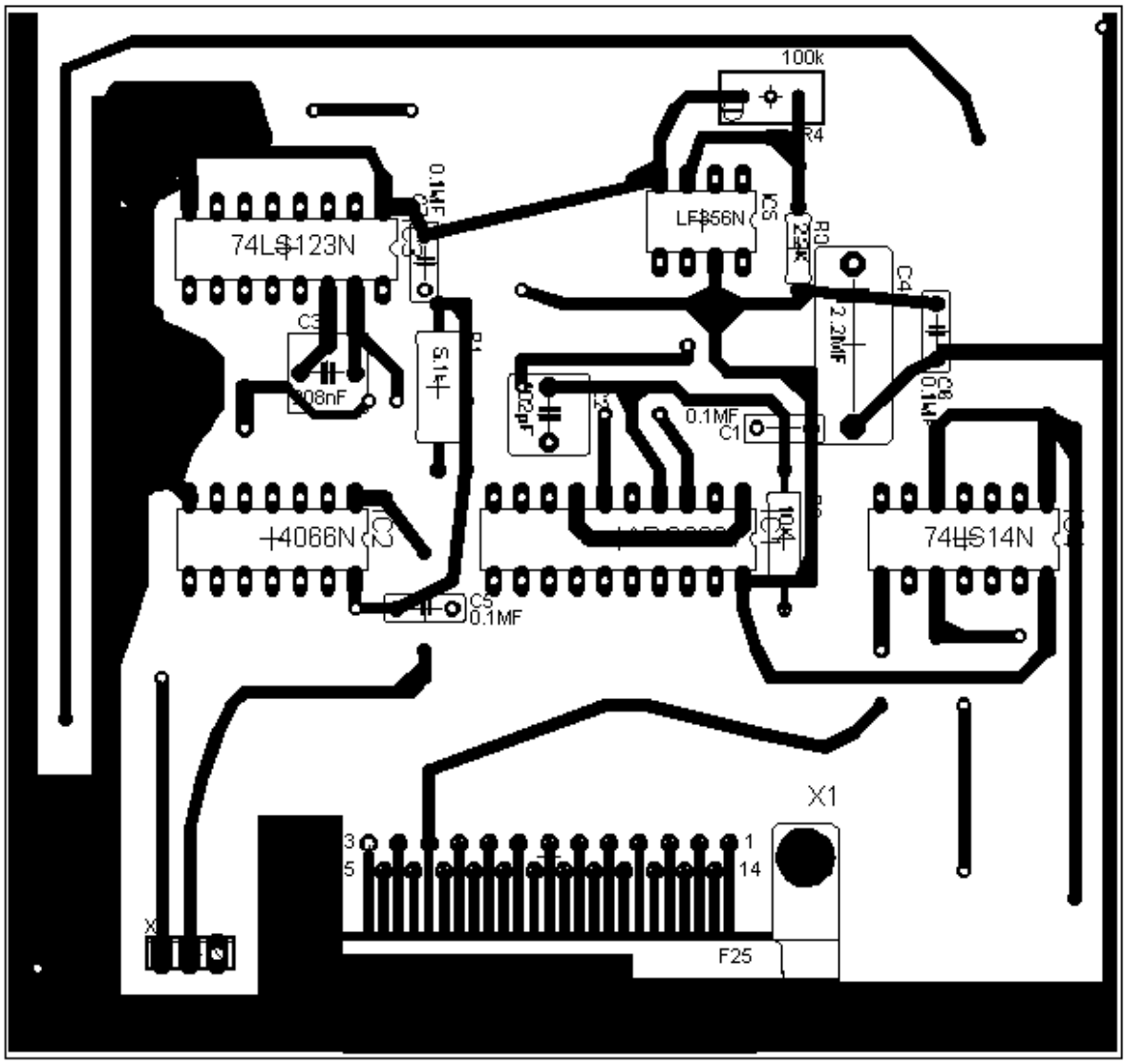


FIGURA 5.7 Circuito Adquisitor de Datos, vista de arriba.

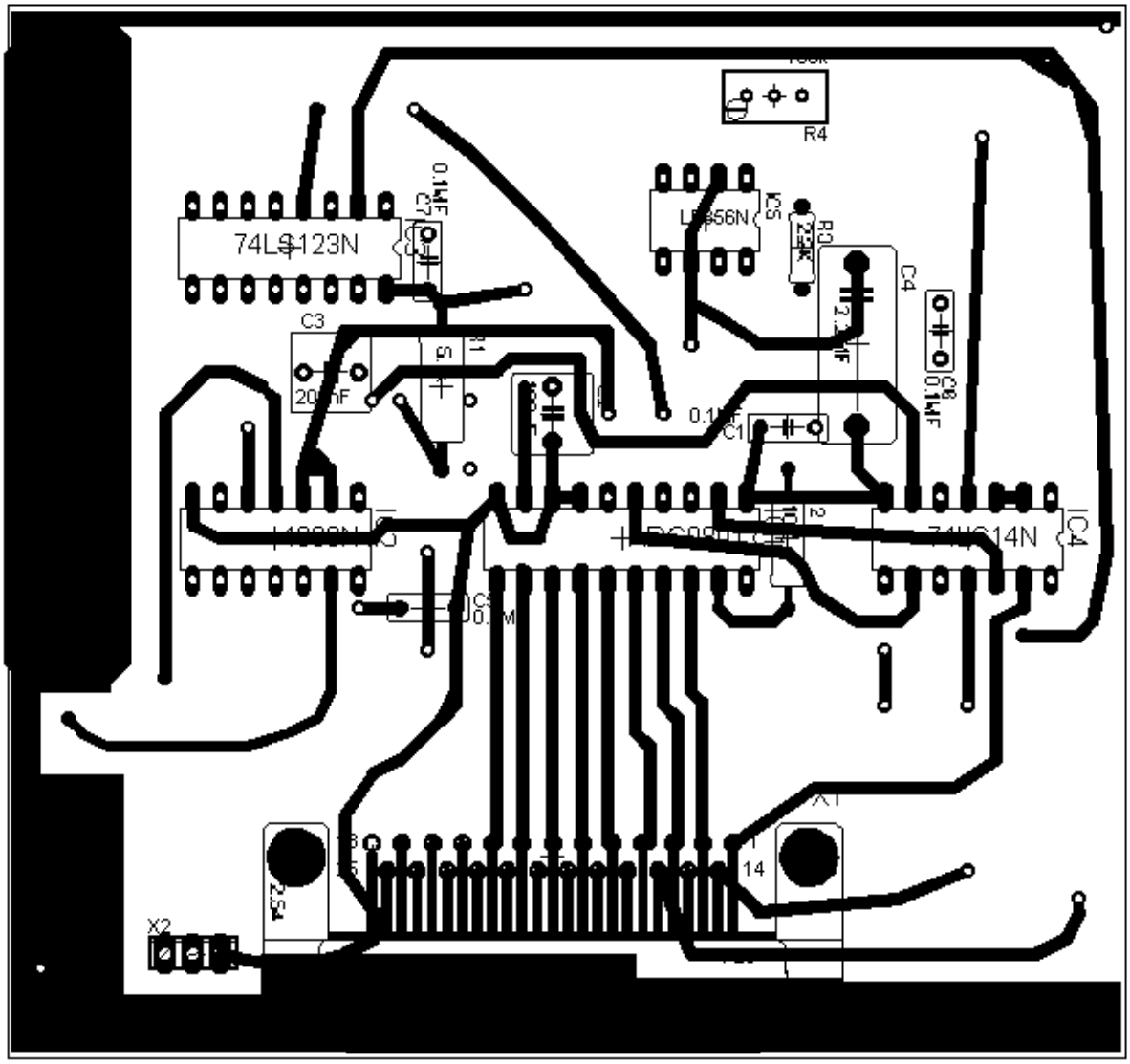


FIGURA 5.8 Circuito Adquisitor de Datos, vista de abajo.

5.3 ENVOLTURAS Y CUBIERTAS PROTECTORAS.

Para la implementación de la cubierta del sistema en contacto con el paciente, se tomará como referencia la norma IRAM 4220-1(ver apéndice A), lo cual expresa:

¨ Los APARATOS deben estar construidos y envueltos de tal manera que se dé una protección contra contactos con las partes BAJO TENSIÓN y con partes que se pueden volver BAJO TENSIÓN en las CONDICIONES DE PRIMER DEFECTO¨.

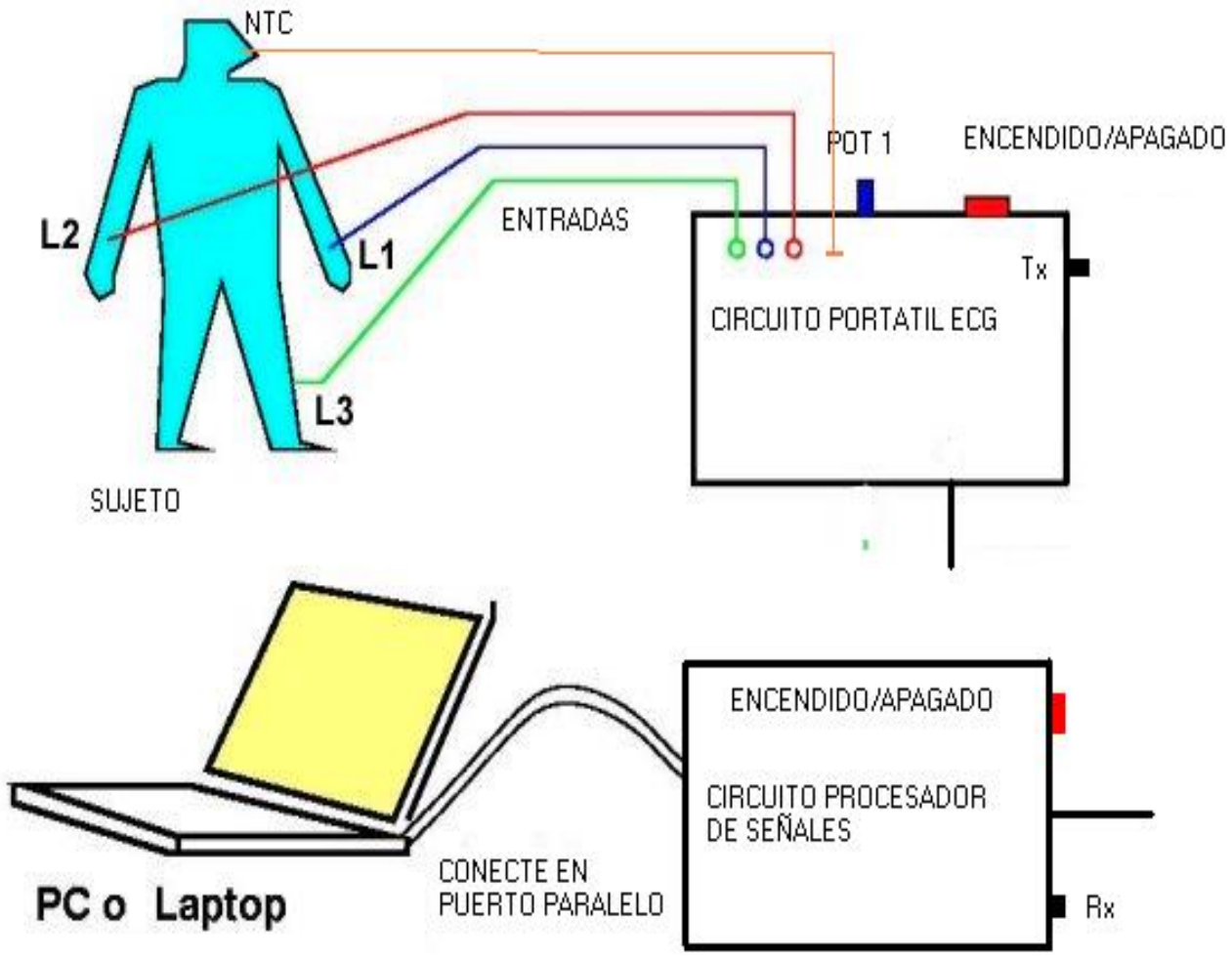


FIGURA 5.12 Esquema General del Circuito.

CAPÍTULO VI. PRUEBAS REALIZADAS CON EL PROTOTIPO

6.1 CLASIFICACION.

Según el tipo de protección contra los choques eléctricos el prototipo se clasifica como: APARATO CON UNA FUENTE DE ENERGÍA ELÉCTRICA INTERNA (ver norma IRAM 4220-1, *5.1) y según el grado de protección contra los choques eléctricos: es CF (ver norma IRAM 4220-1, *5.2, apéndice A).

De acuerdo al tipo de envolvente conductora prácticamente continua se clasifica como, equipo de Clase III.

Equipo de tipo CF [19]: Equipo que proporciona un mayor de protección contra descargas eléctricas, que el tipo BF, particularmente en relación con la corriente de fuga permisible, y dispone de una parte aplicable tipo F.

Los equipos y partes de los equipos previstos para aplicación cardiaca directa deberán ser de tipo CF, y podrán tener si las necesitan partes aplicables de tipo B y BF.

6.2 ANALISIS.[14]

El modo de medida de las señales electrofisiológicas es de manera diferencial. La señal de ECG es muy susceptible a ser afectada por diferentes tipos de señales eléctricas, algunas de carácter externo al circuito de medida y otras de carácter interno. Para eliminar el efecto de las fuentes de error se debe de tener un buen entendimiento de las mismas y la manera

en que estas afectan la señal a medir. Las principales fuentes de ruido son:

1. La fuente de alimentación: Ruido provocado por desequilibrios en el transformador de la fuente de alimentación, por acoples capacitivos entre el primario y el secundario y por el risado. [14]
2. Acoplamiento capacitivo interno entre diferentes componentes. [14]
3. Ruido de componentes: Ruido térmico introducido por las resistencias del circuito, ruido de componentes propios. [14]
4. Ruido de cuantificación: En caso de procesamiento digital de la señal. [14]

La elección de componentes de calidad en nuestro circuito y un buen diseño disminuye en gran parte los efectos antes mencionados. [14]

A continuación se presentan las soluciones para minimizar los distintos efectos de interferencia antes citados, en el mismo orden en que fueron expuestos:

1. Utilización de baterías en la etapa del amplificador del ECG.
2. Separación adecuada entre pistas y componentes (ver Capitulo V).
3. Uso de resistencias con suficiente capacidad de manejo de potencia y con bajo porcentaje de error.

4. Alta ganancia de voltaje en la etapa de amplificación de la señal a digitalizar.

6.2 PRUEBAS REALIZADAS.

Condiciones del equipo y del ambiente:

Fecha: 06 de Enero de 2005.

Lugar: Laboratorio de biomédica UDB.

Temperatura: 30 grados centígrados.

Equipo sin las cubiertas protectoras.

6.2.1 Determinación del ritmo cardíaco. Se inyectó una señal de ECG y se determinó su ritmo de forma aproximada, viendo y midiendo la señal en pantalla y por medio de un contador del programa. La señal era consistente en ambos casos. Primero se hizo con 30 BPM y luego con 60 BPM.

Señal de Prueba del simulador de laboratorio ECG	Señal de Salida del Demodulador
2mVp-p, 30 BPM	0.5 Vp-p, 30 BPM
2mVp-p, 60 BPM	0.5 Vp-p, 60 BPM

TABLA 6.1 Prueba realizada con el simulador de ECG

6.2.2 Determinación de fuentes de interferencia por equipos adyacentes.

1. Se realizó una llamada y se estableció comunicación con un abonado con un teléfono celular a una distancia menor de 1.5 metros y se obtuvo interferencia muy grande en el circuito receptor. A una distancia de 1.5 metros, la interferencia desapareció totalmente.

Señal de Prueba del simulador de laboratorio ECG	Señal de Salida del Demodulador
2mVp-p, 30 BPM	Ruido
2mVp-p, 60 BPM	Ruido

TABLA 6.2 Prueba realizada estableciendo una llamada con un télelefono celular

2. Se conectó y operó un ventilador de pedestal a 50 centímetros del circuito receptor y a 2 metros del transmisor y no se obtuvo interferencia alguna en la señal vista en el computador u osciloscopio (salida del demodulador).

Señal de Prueba del simulador de laboratorio ECG	Señal de Salida del Demodulador
2mVp-p, 30 BPM	0.5 Vp-p, 30 BPM
2mVp-p, 60 BPM	0.5 Vp-p, 60 BPM

TABLA 6.3 Prueba realizada con un ventilador de pedestal

3. Con el ventilador del literal anterior operando, se puso en funcionamiento a dos metros del receptor y transmisor un Rotador Cero lógico (utilizado en laboratorios para hacer mezclas de muestras) y la señal adquirida se mantuvo sin cambio.

Señal de Prueba del simulador de laboratorio ECG	Señal de Salida del Demodulador
2mVp-p, 30 BPM	0.5 Vp-p, 30 BPM
2mVp-p, 60 BPM	0.5 Vp-p, 60 BPM

TABLA 6.4 Prueba realizada con un ventilador de pedestal y rotador

6.2.3 Pruebas de alcance. Se alejo el transmisor del receptor a una distancia de diez metros y la señal se mantuvo sin cambio apreciable en el osciloscopio.

Señal de Prueba del simulador de laboratorio ECG	Señal de Salida del Demodulador
2mVp-p, 30 BPM	0.5 Vp-p, 30 BPM
2mVp-p, 60 BPM	0.5 Vp-p, 60 BPM

TABLA 6.5 Resultado de la prueba de alcance

6.3.4 Determinación de la Corriente de Fuga del Paciente Desde Parte Aplicable Hacia Envoltura, se utilizara la configuración del circuito de la figura 23, según su clasificación (ver apéndice A, norma IRAM 4220-1, apartado 19.4 *h).

La corriente obtenida con ayuda del circuito descrito en la figura 15, del apéndice A, norma IRAM 4220-1, fue cero amperios. Se efectuó la medida con el osciloscopio y fue menor a un microamperio.

CONCLUSIONES

La problemática en el área de salud en El Salvador respecto a las enfermedades del corazón y sus implicaciones es tal que requiere la utilización de nuevos métodos, donde la tecnología pueda ofrecer una alternativa para el monitoreo a distancia de personas con enfermedades del corazón, permitiendo a los conocedores tomar una acción orientada a la prevención de consecuencias serias.

La telemetría y telesupervisión implica el uso de muy poco personal técnico y más autonomía.

El proyecto desarrollado, provee una solución al cuidado medico y permite la aplicación de tecnologías propias para la solución de problemas locales.

Los principios básicos que demuestran que el sistema puede funcionar al ser implementado, son comprobados por medio del prototipo, el cual transmite los datos de los sensores de la actividad eléctrica del corazón y del ritmo respiratorio, por radio, entre el equipo portátil y la terminal remota, al final presenta los resultados de ambos sensores de forma gráfica en la pantalla de un computador.

A través de las pruebas realizadas pudo comprobarse que la única fuente de interferencia capas de interferir totalmente a cierta distancia del receptor es el teléfono celular, por lo cual debe guardarse una distancia mínima de un metro, entre estos aparatos en la fase de llamada.

POSIBLES MEJORAS

1) Incrementar el orden de los filtros, para disminuir la interferencia entre canales.
2) Incrementar el ancho de banda del NE567, para no tener una sintonización critica.
3) Utilizar otro transmisor receptor, con otros canales, por si existe algún canal con interferencia.
4) Mejorar la protección contra impactos y choques eléctricos de los chasis, con una protección más robusta.
5) Proveer de conectores a cables de sensores, para poder desmontarlos del equipo y darle más portabilidad al adaptador de señales.
6) Programar mas utilidades o servicios al programa, como acceso a internet con el fin de poder enviar o recibir información de sitios lejanos con fines de monitoreo..

Especificaciones Técnicas del Equipo

Adaptador de Señales:

Fuente de Alimentación: Dos baterías de 9 Voltios y una de 1.5 voltios para el transmisor FM.

Distancia Máxima de Trabajo entre Transmisor y Receptor: 10 mts.

Protecciones del Circuito para el Paciente:

Toques Eléctricos: Uso de diodos, en las entradas de los electrodos y resistencias limitadoras.

Tipo de Transmisor: FM de 150 Mhz, cobertura máxima 10 mts.

Demodulador:

Fuente de Alimentación: Fuente Bipolar de 5 voltios dc.

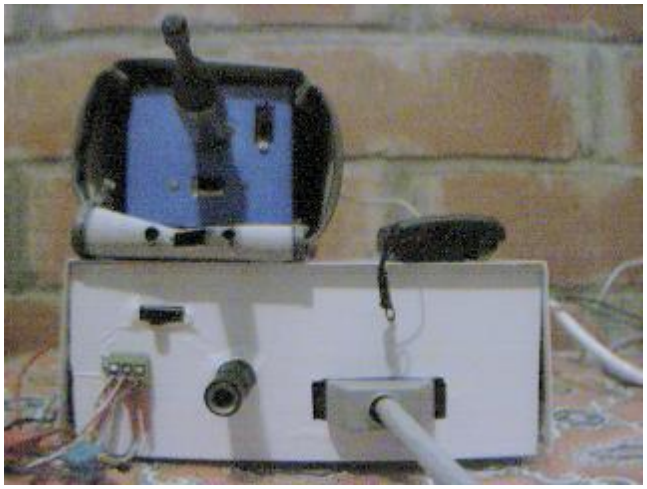
Tipo de Receptor: FM de 150 M hz.

Visualización:

Monitor de Computadora, con funciones propias del monitor.

Carcasa:

Metálica con leve protección a los impactos.



PROTOTIPO DEL SISTEMA. EL EQUIPO AZUL ES EL ADAPTADOR DE SEÑALES Y EL BLANCO ES EL DEMODULADOR

REFERENCIAS BIBLIOGRAFICAS

- [1] http:\www.Electrocardiagram (ECG - EKG) application SAFETY.htm(Electrocardiogram (ECG) project for DrDaq). Consultada el 10/02/2005. Página 1.

- [2] http:\www.ECG-MÉDICO\Heart Care - Electrocardiogram- Stress Test- Holter - Spanish Content - Methodist Health Care System, Houston, Texas.htm, Copyright © 2003 Methodist Health Care System, Houston, TX. All Rights Reserved. Visitado el 10/02/2005. Páginas 2, 4, 5, 6.

- [3] Provaznik, Ivo, Adaptative Systems in ECG Processing, Ph.D. Dissertation, Technical University of Brno, Checoslovaquia, 1995. Página 2, 3, 7.

- [4] Bronzino, J.D., The Biomedical Engineering Handbook, 1995. Página 2.

- [5] Tompkins, W.J., Biomedical Digital Signal Processing, Prentice Hall, Inc, 1993. Páginas 3, 7.

- [6] Leslie Cromwell, Instrumentación y medidas biomédicas, 1980. Página 9.

- [7] Mauricio Orlando Castillo, Tesis: Diseño e Implementación de una Red de Teleacción Aplicada a la Detección de Agentes Contaminantes del Medio Ambiente en el Area Metropolitana de San Salvador, 28 Febrero de 1998.

- [8] Wayne Tomasi, Electronic Communications Systems.

- [9] Sergio Franco, Design with Operational Amplifiers and Analogog Integrated Circuits, 1988. Páginas 40, 57, 58.

- [10] Texas Instruments, The TTL Logic Data Book. Página 41.

- [11] Albert Paúl Malvino, Principios de Electrónica, 1986.páginas 43,44.

- [12] Tocci Widmer, Sistemas Digitales Principios y Aplicaciones. Página 44.

- [13]Interfacing the enhanced Paralell Port, http://beyondlogic.org. Visitado el 10/02/2005. Página 45.

- [14] Br. Martin Oliveri, Elementos de Diseño de Circuitos de Amplificación del ECG, Universidad de la Republica, Uruguay, 2004.

- [15] http:\www.Scientific American Home Is Where the ECG Is -- Watch your heartbeat with do-it-yourself equipment, as described.htm. Consultada el 10/02/2005. Página 70.

- [16] http:\www.IEC - International Electrotechnical Commission INTERNATIONAL STANDARDS AND CONFORMITY ASSESSMENT.htm. Visitada el 01/02/2005. Página 70.

- [17] http:\www. Circuitos electrónicos compatibilidad electromagnetica.htm, consultada el 01/02/2005. Página 70.

- [18]http://www.healthsystem.virginia.edu/UVAHealth/adult_nontrauma_sp/vital.cfm, Visitada el 08/02/2005. Página 56.

- [19] Procedimiento de Inspección y Mantenimiento Preventivo/ Seguridad Eléctrica de los Equipos Electromedicos. Hospital Clínico Universitario, Lozano Bleza- Zaragoza, Servicio de Electromedicina, 11/02/2005.

APENDICE A. Seguridad hospitalaria.

A continuación se presentan un resumen de las normas de seguridad hospitalaria que se utilizan actualmente en el país, esto debido a que proyectos como el de la cooperación técnica alemana, han introducido al país, aparatos analizadores de seguridad eléctrica que cumplen ciertos estándares internacionales, nosotros hemos tomado como base los estándares IEC 601-1 y el HEI-95, a continuación se presenta un resumen de los limites permitidos por los dos estándares.

DESCRIPCIÓN DEL TEST		TIEMPO	VALORES LIMITE					
			CLASE I			CLASE II		
			B	BF	CF	B	BF	CF
VOLTAJE DE LINEA			-					
CONTINUIDAD DE CONDUCTOR DE TIERRA			0.2OHM			N/A		
AISLAMIENTO	L1-L2-CASE	5s.	2	2	20	7	7	70
RESITENCIA	AP-CASE	5s.	N/A	5	50	N/A	5	50
CONSUMO DE CORRIENTE								
TIERRA	N.P	2s.	500	500	500	N/A	N/A	N/A
FUGA	N.P. NO L2	2s.	1000	1000	1000	N/A	N/A	N/A
CORRIENTE	R.P.	2s.	500	500	500	N/A	N/A	N/A
UA	R.P NO L2	2s.	1000	1000	1000	N/A	N/A	N/A
CARCASA	N.P.	2s.	100	100	100	100	100	100
FUGA	N.P. NO L2	2s.	500	500	500	500	500	500
CORRIENTE	N.P NO TIERRA	2s.	500	500	500	N/A	N/A	N/A

TABLA 2a. Limites para el estándar IEC 601-1.

UA

UA	R.P	2s.	100	100	10	100	100	10
	R.P. NO L2	2s.	500	500	50	500	500	50
	R.P. NO	2s.	500	500	50	N/A	N/A	N/A
PACIENTE	N.P.	2s.	100	100	10	100	100	10
CORRIENTE	N.P. NO L2	2s.	500	500	50	500	500	50
UA	R.P. NO	2s.	500	500	50	N/A	N/A	N/A
	R.P.	2s	N/A	5000	50	N/A	5000	50
	R.P NO L2	2s.	N/A	5000	50	N/A	500	50
	R.P. NO	2s.	10	10	10	10	10	10
	MAINS ON	2s	N/A	5000	50	N/A	5000	50
PACIENTE	N.P.	2s.	10	10	10	10	10	10
AUXILIAR	N.P. NO L2	2s.	500	500	50	500	500	50
CORRIENTE		2s.	500	500	50	N/A	N/A	N/A
UA		2s.	10	10	10	10	10	10
		2s.	500	500	50	500	500	50
		2s.	500	500	50	N/A	N/A	N/A

TABLA 2b. Limites para el estándar IEC 601-1.

DESCRIPCIÓN DEL TEST		VALORES LIMITE					
		CLASE I			CLASE II		
		B	BF	CF	B	BF	CF
RESISTENCIA DE CONDUCTOR DE PROTECCIÓN A TIERRA		0.2OHM			N/A		
M ohm resistencia de aislamiento	principal L1-L2-carcasa	50	50	50	N/A	7	70
M ohm resistencia de aislamiento	todas a carcasa	N/A	50	50	50	5	50
corriente de fuga	pol normal	500	500	500	N/A	N/A	N/A
corriente de fuga	Rev pol	500	500	500	N/A	N/A	N/A
corriente de fuga de carcasa	norm pol	500	500	500	100	100	10
corriente de fuga de carcasa	norm pol, no tierra	500	500	500	**N/A**	N/A	N/A
corriente de fuga de carcasa	Rev pol,	500	500	500	100	100	10
corriente de fuga de carcasa	Rev pol, no tierra	500	500	500	N/A	N/A	N/A
corriente de fuga a paciente	Norm pol	500	500	50	100	100	10
corriente de fuga a paciente	Norm pol, no tierra	500	500	50	N/A	N/A	N/A
corriente de fuga a paciente	Rev pol	500	500	50	100	100	10
corriente de fuga a paciente	Rev pol, no tierra	500	500	50	N/A	N/A	N/A

TABLA 3. Limites para el estándar HEI-95.

El electrocardiógrafo se considera como un equipo de clase I, tipo CF, es decir que posee polarización a tierra, y un aislamiento para usos cardíacos, las tablas anteriores muestran los diferentes límites en dos estándares diferentes para un equipo que cumple las especificaciones mencionadas.

	Tensión continua	15	36	75	150	300	450	600	800	900	1200	
	Tensión alterna	12	30	60	125	250	400	500	660	750	1000	
Equivalente a la AISLACIÓN PRINCIPAL entre partes de polaridad opuesta	A-f	0,4	0,5	0,7	1	1,6	2,4	3	4	4,5	6	DISTANCIA EN AIRE
		0,8	1	1,3	2	3	4	5,5	7	8	11	LÍNEAS DE FUGA
AISLACIÓN PRINCIPAL o AISLACIÓN SUPLEMENTARIA	A – al, A-b A – c, A – j	0,8	1	1,2	1,6	2,5	3,5	4,5	6	6,5	9	DISTANCIA EN AIRE
	B – d, B-c	1,7	2	2,3	3	4	6	8	10,5	12	16	LÍNEAS DE FUGA
AISLACIÓN REFORZADA o AISLACIÓN DOBLE	A – a2 A – e, A-k	1,6	2	2,4	3,2	5	7	9	12	13	18	DISTANCIA EN AIRE
	B – a, B – e	3,4	4	4,6	6	8	12	16	21	24	32	LÍNEAS DE FUGA

TABLA 4. Líneas de fuga y distancias en aire en milímetros según norma IRAM 4220-1:1999[1)]

Según norma IRAM 4220-1:1999[1)]

***5.1** Según el tipo de protección contra los choques eléctricos:

a) APARATO excitado por una fuente de energía eléctrica externa:

- APARATO DE CLASE I;
- APARATO DE CLASE II.

b) APARATO CON UNA FUENTE DE ENERGÍA ELÉCTRICA INTERNA

5.2 Según el grado de protección contra los choques eléctricos:

- PARTE APLICABLE DEL TIPO B;
- PARTE APLICABLE DEL TIPO BF;
- PARTE APLICABLE DEL TIPO CF.

14.6 ***APARATOS DEL TIPO B, BF Y CF***

a) No utilizado.
b) No utilizado.

c) Las PARTES APLICABLES que están especificadas en los DOCUMENTOS

ACOMPAÑANTES como convenientes para una APLICACIÓN CARDIACA DIRECTA deben ser del TIPO CF.

14.5 APARATO CON UNA FUENTE INTERNA DE ENERGÍA ELÉCTRICA

**16. ENVOLTURAS y CUBIERTAS PROTECTORAS*

a) Los APARATOS deben estar construidos y envueltos de tal manera que se dé una protección contra contactos con las partes BAJO TENSIÓN y con partes que se pueden volver BAJO TENSIÓN en las CONDICIONES DE PRIMER DEFECTO.
Esta exigencia se aplica a todas las posiciones del APARATO cuando el mismo funciona como en USO NORMAL, aún después de la apertura de las tapas y puertas y de la remoción de partes sin la ayuda de una HERRAMIENTA o según las instrucciones de uso.

19.4 *Ensayos*

****h)*** *Medición de la CORRIENTE DE FUGA DEL PACIENTE*

Para las conexiones a la(s) PARTE(S) APLICABLE(S) ver el apartado 19.1e) y el Anexo K(de la norma).

6) *Los APARATOS CON FUENTE INTERNA DE ENERGÍA ELÉCTRICA se ensayan conforme a la FIGURA 23.*

Cuando la ENVOLTURA es de material aislante, se debe aplicar una lámina metálica como la descripta en el inciso 19.4g) 5).

Esquema equivalente a lo arriba indicado y que se utiliza en las FIGURAS subsiguientes

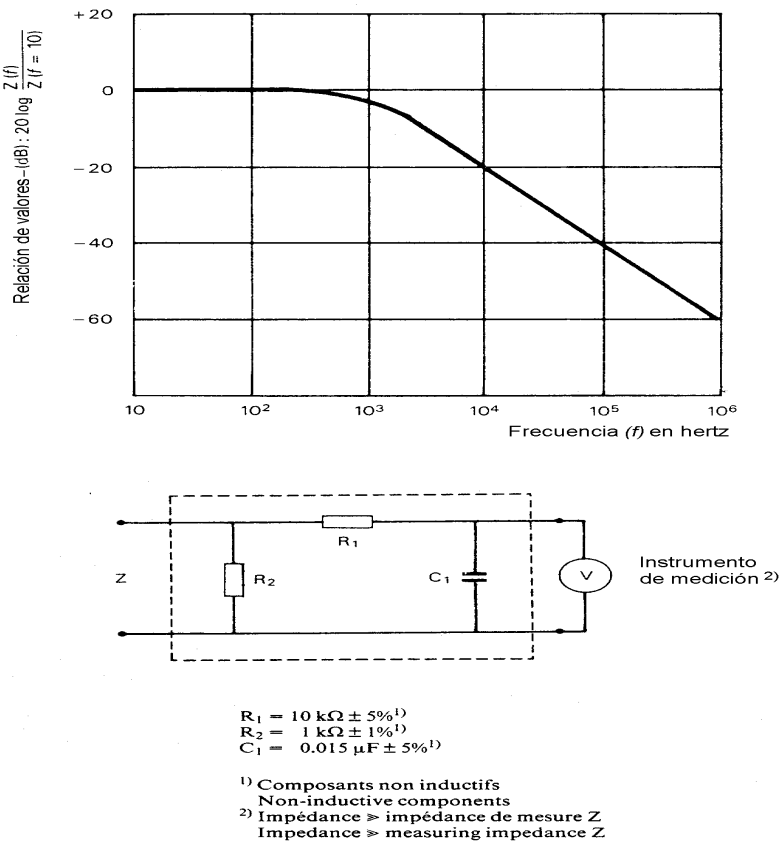
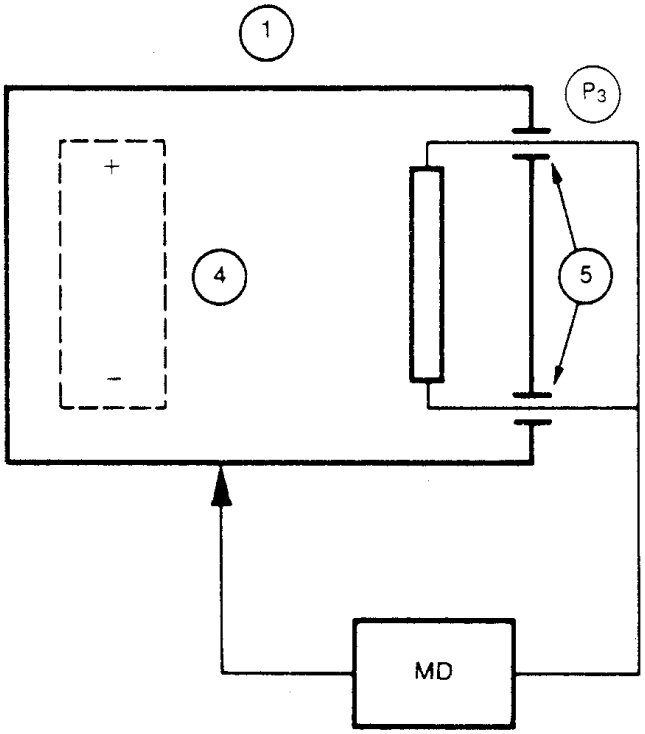


FIGURA 15 - Ejemplo de un dispositivo para medición y su respuesta en frecuencia

(ver apartado 19.4 e))

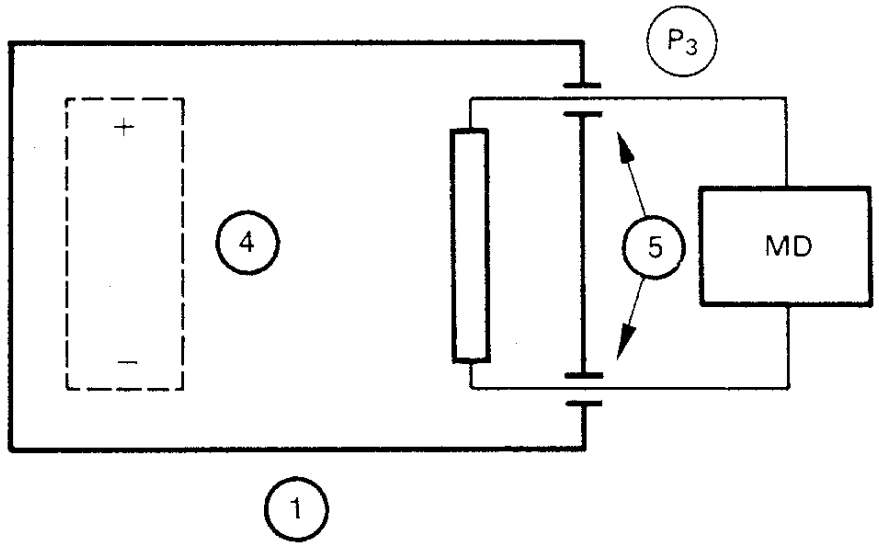
1) Componentes no inductivos
2) Impedancia » impedancia de medición Z



Ver leyendas luego de la FIGURA 27

Medir entre la PARTE APLICABLE y la ENVOLTURA (CONDICIÓN NORMAL). Efectuar, si fuera posible, el ensayo del apartado 17 a).

FIGURA 23 – Circuito para medición de la CORRIENTE DE FUGA DEL PACIENTE desde la PARTE APLICABLE hacia la ENVOLTURA del APARATO ALIMENTADO CON UNA FUENTE INTERNA DE ALIMENTACIÓN (ver apartado 19.4 h))



Ver leyendas luego de esta FIGURA

FIGURA 27 - Circuito para medición de la CORRIENTE AUXILIAR DEL PACIENTE de un APARATO ALIMENTADO CON UNA FUENTE INTERNA DE ENERGÍA ELÉCTRICA (ver apartado 19.4 j))

Leyendas relativas a los símbolos para las FIGURAS 10 a la 27

1 - ENVOLTURA DEL APARATO
2 - Alimentación especificada
3 - SECTOR DE ENTRADA O SALIDA DE LA SEÑAL en cortocircuito o cargada
4 - FUENTE INTERNA DE ENERGÍA ELÉCTRICA
5 - PARTE APLICABLE
6 - PARTE METÁLICA ACCESIBLE no siendo una PARTE APLICABLE y no PROTEGIDA POR PUESTA A TIERRA

T_1, T_2 transformadores de aislamiento monofásicos, bifásicos, polifásicos con potencia suficiente y tensión de salida ajustable

V (1, 2, 3) voltímetros que indican el valor eficaz, en caso pertinente y si fuera posible un solo aparato con un conmutador

S_1, S_2, S_3 interruptores monopolares que simulan la interrupción de un conductor de alimentación (CONDICIÓN DE PRIMER DEFECTO)

S_5, S_9 llaves de conmutación para la inversión de la polaridad de la TENSIÓN DE LA RED

S_7, S_8 interruptores monopolares que simulan la interrupción de

un único CONDUCTOR DE TIERRA DE PROTECCIÓN (CONDICIÓN DE PRIMER DEFECTO)

S_{10}, S_{11} llaves para la conexión de un BORNE FUNCIONAL DE TIERRA al punto de puesta a tierra del circuito de alimentación de medición

S_{12} llave para la conexión de una PARTE APLICABLE DEL TIPO F al punto de puesta a tierra del circuito de alimentación de medición

S_{13} llave para la conexión a tierra de una PARTE METÁLICA ACCESIBLE que no sea una PARTE APLICABLE y que no está PROTEGIDA POR PUESTA A TIERRA

P_1 tomacorrientes, fichas o bornes para la conexión del APARATO

P_2 tomacorrientes, fichas o bornes para la conexión a una fuente de alimentación especificada

P_3 tomacorrientes, fichas o bornes para las conexiones al PACIENTE

MD (1, 2, 3, 4) dispositivos de medición (ver FIGURA 15)

FE BORNE FUNCIONAL DE TIERRA

PE BORNE DE PROTECCIÓN A TIERRA

--- conexión opcional

R impedancia para la protección del usuario del aparato de ensayo

APENDICE B. EL PROGRAMA

```
// ChildFrm.cpp : implementation of the CChildFrame class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"
#include "ChildFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
```

```
////////////
// CChildFrame

IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
                //{{AFX_MSG_MAP(CChildFrame)
                                // NOTE - the ClassWizard will add
and remove mapping macros here.
                                //    DO NOT EDIT what you see in
these blocks of generated code !
                        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////
////////////
// CChildFrame construction/destruction

CChildFrame::CChildFrame()
{
        // TODO: add member initialization code here

}
```

```
CChildFrame::~CChildFrame()
{
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
// TODO: Modify the Window class or styles here by modifying
                    //  the CREATESTRUCT cs

          return CMDIChildWnd::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
////////////
// CChildFrame diagnostics

#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
                CMDIChildWnd::AssertValid();
}

void CChildFrame::Dump(CDumpContext& dc) const
{
```

```
                    CMDIChildWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
////////////
// CChildFrame message handlers

// DATOS.cpp : implementation file

//

#include "stdafx.h"
#include "Scribble.h"
#include "DATOS.h"
#include "ScribDoc.h"
#include "stdio.h"
#include "conio.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif


/////////////////////////////////////////////////////////////////////////
////////////
// DATOS dialog


DATOS::DATOS(CWnd* pParent /*=NULL*/)
                : CDialog(DATOS::IDD, pParent)
{

                    //{{AFX_DATA_INIT(DATOS)
                          m_fecha = 0;
                           m_edad = 0;
                        m_name = _T("");
                      Y_Multiplicador = 0;
                      X_Multiplicador = 0;
                     m_frecuencia2 = 0.0f;
                     m_frecuencia1 = 0.0f;
                       //}}AFX_DATA_INIT
}
```

```
void DATOS::DoDataExchange(CDataExchange* pDX)
{
                CDialog::DoDataExchange(pDX);
                    //{{AFX_DATA_MAP(DATOS)
            DDX_Control(pDX, IDC_BUTTON1, m_pipi);
            DDX_Control(pDX, IDC_BUTTON2, m_tutu);
              DDX_Text(pDX, IDC_EDIT3, m_fecha);
               DDX_Text(pDX, IDC_EDIT2, m_edad);
               DDX_Text(pDX, IDC_EDIT1, m_name);
          DDX_Text(pDX, IDC_EDIT5, Y_Multiplicador);
        DDV_MinMaxLong(pDX, Y_Multiplicador, 1, 1000);
          DDX_Text(pDX, IDC_EDIT4, X_Multiplicador);
        DDV_MinMaxLong(pDX, X_Multiplicador, 1, 1000);
           DDX_Text(pDX, IDC_EDIT7, m_frecuencia2);
           DDX_Text(pDX, IDC_EDIT6, m_frecuencia1);
                       //}}AFX_DATA_MAP

}


BEGIN_MESSAGE_MAP(DATOS, CDialog)
                    //{{AFX_MSG_MAP(DATOS)
            ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
            ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
```

```
            ON_BN_CLICKED(IDC_BUTTON4, OnButton4)
                        //}}AFX_MSG_MAP
END_MESSAGE_MAP()



#ifdef _DEBUG

CScribbleDoc* DATOS::GetDocument() // non-debug version is inline
{
 ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScribbleDoc)));
              return (CScribbleDoc*)m_pDocument;
}
#endif //_DEBUG



////////////////////////////////////////////////////////////////
////////////
// DATOS message handlers

void DATOS::OnButton2()
{

                      UpdateData(true);
  // TODO: Add your control notification handler code here
```

```
                    m_alfa=m_name;
                    m_beta=m_edad;
                    m_gama=m_fecha;
                m_mulX=X_Multiplicador;
                m_mulY=Y_Multiplicador;
                   c_fecha=m_fecha;

}

void DATOS::OnButton1()
{
  // TODO: Add your control notification handler code here
                    m_name=m_alfa;
                    m_edad=m_beta;
                    m_fecha=m_gama;
                X_Multiplicador=m_mulX;
                Y_Multiplicador=m_mulY;
                  UpdateData(false);
                      c_fecha=4;

}


void DATOS::OnButton4()
```

```
{
  // TODO: Add your control notification handler code here
                    m_frecuencia1=F1;
                    m_frecuencia2=F2;
                    UpdateData(false);
}

// MainFrm.cpp : implementation of the CMainFrame class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"

#include "MainFrm.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
                //{{AFX_MSG_MAP(CMainFrame)
                        // NOTE - the ClassWizard will add
and remove mapping macros here.
                        //    DO NOT EDIT what you see in
these blocks of generated code !
                    ON_WM_CREATE()
                    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
```

```
{
      ID_SEPARATOR,           // status line indicator
                        ID_INDICATOR_CAPS,
                         ID_INDICATOR_NUM,
                        ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////
////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
      if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
```

```
                                                    return -1;

            if (!m_wndToolBar.Create(this) ||

        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
                                {
                                      TRACE0("Failed to create
toolbar\n");
                                 return -1;      // fail to create
                                }

           if (!m_wndStatusBar.Create(this) ||

        !m_wndStatusBar.SetIndicators(indicators,
                                   sizeof(indicators)/sizeof(UINT)))
                                {
                                 TRACE0("Failed to create status
bar\n");
                                 return -1;      // fail to create
                                }

      // TODO: Remove this if you don't want tool tips
   m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
                                   CBRS_TOOLTIPS | CBRS_FLYBY |
```

```
CBRS_SIZE_DYNAMIC);

// TODO: Delete these three lines if you don't want the toolbar
to
                        //  be dockable
         m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
                EnableDocking(CBRS_ALIGN_ANY);
                DockControlBar(&m_wndToolBar);



                            return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
// TODO: Modify the Window class or styles here by modifying
                    //  the CREATESTRUCT cs

          return CMDIFrameWnd::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////
////////////
// CMainFrame diagnostics
```

```
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
                CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
                  CMDIFrameWnd::Dump(dc);
}

#endif //_DEBUG

////////////////////////////////////////////////////////////////////
////////////
// CMainFrame message handlers

// Scribble.cpp : Defines the class behaviors for the
application.
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
```

```
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"

#include "MainFrm.h"
#include "ChildFrm.h"
#include "ScribDoc.h"
#include "ScribVw.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
////////////
```

```
// CScribbleApp

BEGIN_MESSAGE_MAP(CScribbleApp, CWinApp)
                //{{AFX_MSG_MAP(CScribbleApp)

                                // NOTE - the ClassWizard will add
and remove mapping macros here.
                                //    DO NOT EDIT what you see in
these blocks of generated code!
                        //}}AFX_MSG_MAP
          // Standard file based document commands
         ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
               // Standard print setup command
 ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////
////////////
// CScribbleApp construction

CScribbleApp::CScribbleApp()
{
```

```
            // TODO: add construction code here,
   // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////
////////////
// The one and only CScribbleApp object

CScribbleApp theApp;

/////////////////////////////////////////////////////////////////////////
////////////
// CScribbleApp initialization

BOOL CScribbleApp::InitInstance()
{
                // Standard initialization
// If you are not using these features and wish to reduce the
size
  //  of your final executable, you should remove from the
following
  //  the specific initialization routines you do not need.

#ifdef _AFXDLL
```

```
Enable3dControls();                     // Call this when using
MFC in a shared DLL
#else
Enable3dControlsStatic();       // Call this when linking to MFC
statically
#endif

LoadStdProfileSettings();  // Load standard INI file options
(including MRU)


 // Register the application's document templates.  Document
templates
//  serve as the connection between documents, frame windows and
views.

              CMultiDocTemplate* pDocTemplate;
            pDocTemplate = new CMultiDocTemplate(
                                                IDR_SCRIBBTYPE,
                                  RUNTIME_CLASS(CScribbleDoc),
                                RUNTIME_CLASS(CChildFrame), //
custom MDI child frame
                                RUNTIME_CLASS(CScribbleView));
```

```
                AddDocTemplate(pDocTemplate);

                // create main MDI Frame window
            CMainFrame* pMainFrame = new CMainFrame;
           if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
                                                return FALSE;
                m_pMainWnd = pMainFrame;

// Enable drag/drop open.  We don't call this in Win32, since a
   //  document file extension wasn't chosen while running
AppWizard.
               m_pMainWnd->DragAcceptFiles();

                 // Enable DDE Execute open
                     EnableShellOpen();
                RegisterShellFileTypes(TRUE);

// Parse command line for standard shell commands, DDE, file open
                 CCommandLineInfo cmdInfo;
                 ParseCommandLine(cmdInfo);

     // Dispatch commands specified on the command line
             if (!ProcessShellCommand(cmdInfo))
                                                return FALSE;
```

```
// The main window has been initialized, so show and update it.
            pMainFrame->ShowWindow(m_nCmdShow);
                pMainFrame->UpdateWindow();

                        return TRUE;
}

////////////////////////////////////////////////////////////////
////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
                        CAboutDlg();

// Dialog Data
                  //{{AFX_DATA(CAboutDlg)
                enum { IDD = IDD_ABOUTBOX };
                        //}}AFX_DATA

    // ClassWizard generated virtual function overrides
```

```
                //{{AFX_VIRTUAL(CAboutDlg)
                        protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
                    //}}AFX_VIRTUAL

// Implementation
protected:
                  //{{AFX_MSG(CAboutDlg)
                                    // No message handlers
                       //}}AFX_MSG
                   DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
               //{{AFX_DATA_INIT(CAboutDlg)
                     //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
              CDialog::DoDataExchange(pDX);
               //{{AFX_DATA_MAP(CAboutDlg)
```

```
                        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
                  //{{AFX_MSG_MAP(CAboutDlg)
                                        // No message handlers
                        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CScribbleApp::OnAppAbout()
{
                        CAboutDlg aboutDlg;
                        aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
////////////
// CScribbleApp commands

// Scribble.cpp : Defines the class behaviors for the
application.
//
// This is a part of the Microsoft Foundation Classes C++
```

```
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"

#include "MainFrm.h"
#include "ChildFrm.h"
#include "ScribDoc.h"
#include "ScribVw.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////////////////////
// CScribbleApp

BEGIN_MESSAGE_MAP(CScribbleApp, CWinApp)
        //{{AFX_MSG_MAP(CScribbleApp)
            // NOTE - the ClassWizard will add and remove mapping macros here.
            //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Standard print setup command
    ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScribbleApp construction
```

```
CScribbleApp::CScribbleApp()
{
            // TODO: add construction code here,
   // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////
////////////
// The one and only CScribbleApp object

CScribbleApp theApp;

/////////////////////////////////////////////////////////////////////////
////////////
// CScribbleApp initialization

BOOL CScribbleApp::InitInstance()
{
                // Standard initialization
// If you are not using these features and wish to reduce the
size
  //  of your final executable, you should remove from the
following
  //  the specific initialization routines you do not need.
```

```
#ifdef _AFXDLL
Enable3dControls();                     // Call this when using
MFC in a shared DLL
#else
Enable3dControlsStatic();       // Call this when linking to MFC
statically
#endif

LoadStdProfileSettings();  // Load standard INI file options
(including MRU)


 // Register the application's document templates.  Document
templates
//  serve as the connection between documents, frame windows and
views.

              CMultiDocTemplate* pDocTemplate;
            pDocTemplate = new CMultiDocTemplate(
                                                IDR_SCRIBBTYPE,
                                  RUNTIME_CLASS(CScribbleDoc),
                                RUNTIME_CLASS(CChildFrame), //
custom MDI child frame
```

```
                                RUNTIME_CLASS(CScribbleView));

                AddDocTemplate(pDocTemplate);

              // create main MDI Frame window
            CMainFrame* pMainFrame = new CMainFrame;
          if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
                                                return FALSE;
                m_pMainWnd = pMainFrame;

// Enable drag/drop open.  We don't call this in Win32, since a
   //  document file extension wasn't chosen while running
AppWizard.
               m_pMainWnd->DragAcceptFiles();

                 // Enable DDE Execute open
                     EnableShellOpen();
                RegisterShellFileTypes(TRUE);

// Parse command line for standard shell commands, DDE, file open
                 CCommandLineInfo cmdInfo;
                ParseCommandLine(cmdInfo);

     // Dispatch commands specified on the command line
```

```
            if (!ProcessShellCommand(cmdInfo))
                                                    return FALSE;


// The main window has been initialized, so show and update it.
            pMainFrame->ShowWindow(m_nCmdShow);
                pMainFrame->UpdateWindow();

                        return TRUE;
}

////////////////////////////////////////////////////////////////////
////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
                        CAboutDlg();

// Dialog Data
                  //{{AFX_DATA(CAboutDlg)
                enum { IDD = IDD_ABOUTBOX };
                        //}}AFX_DATA
```

```
    // ClassWizard generated virtual function overrides
                //{{AFX_VIRTUAL(CAboutDlg)
                        protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
                    //}}AFX_VIRTUAL

// Implementation
protected:
                  //{{AFX_MSG(CAboutDlg)
                                    // No message handlers
                       //}}AFX_MSG
                   DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
               //{{AFX_DATA_INIT(CAboutDlg)
                     //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
```

```
                CDialog::DoDataExchange(pDX);
                 //{{AFX_DATA_MAP(CAboutDlg)
                      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
                 //{{AFX_MSG_MAP(CAboutDlg)
                                      // No message handlers
                       //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CScribbleApp::OnAppAbout()
{
                     CAboutDlg aboutDlg;
                     aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
////////////
// CScribbleApp commands

// ScribDoc.cpp : implementation of the CScribbleDoc class
//
```

```
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"
#include "ScribDoc.h"



#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////
////////////
```

```
// CScribbleDoc

IMPLEMENT_DYNCREATE(CScribbleDoc, CDocument)

BEGIN_MESSAGE_MAP(CScribbleDoc, CDocument)
                //{{AFX_MSG_MAP(CScribbleDoc)
         ON_COMMAND(ID_DATOS_AJUSTE, OnDatosAjuste)
                        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////
////////////
// CScribbleDoc construction/destruction

CScribbleDoc::CScribbleDoc()
{
                  m_pArches = new CBitmap;
                       if (m_pArches)
                              {


                              }
                 m_pBackground = m_pArches;

        // TODO: add one-time construction code here
```

```
}

CScribbleDoc::~CScribbleDoc()
{
                        if (m_pArches)
                                {
                                                delete m_pArches;
                                                m_pArches = NULL;
                                }
}

BOOL CScribbleDoc::OnNewDocument()
{
               if (!CDocument::OnNewDocument())
                                                        return FALSE;
                        InitDocument();
                         return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CScribbleDoc serialization
```

```
void CScribbleDoc::Serialize(CArchive& ar)
{
                    if (ar.IsStoring())
                              {
                              }
                             else
                              {
                              }
                m_strokeList.Serialize(ar);
}

/////////////////////////////////////////////////////////////////////
////////////
// CScribbleDoc diagnostics

#ifdef _DEBUG
void CScribbleDoc::AssertValid() const
{
                 CDocument::AssertValid();
}

void CScribbleDoc::Dump(CDumpContext& dc) const
{
                    CDocument::Dump(dc);
```

```
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
//
// CScribbleDoc commands

void CScribbleDoc::LoadBackground(CBitmap* pBackground)
{
                m_pBackground=pBackground;
}
void CScribbleDoc::UnloadBackground()
{
                    m_pBackground=NULL;
}



BOOL CScribbleDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
        if (!CDocument::OnOpenDocument(lpszPathName))
                                                return FALSE;
                        InitDocument();
                          return TRUE;
}
```

```
void CScribbleDoc::DeleteContents()
{
               while (!m_strokeList.IsEmpty())
                                {
                                 delete m_strokeList.RemoveHead();
                                }
                 CDocument::DeleteContents();
}

void CScribbleDoc::InitDocument()
{
        m_nPenWidth = 2; // default 2 pixel pen width
                        // solid, black pen
   m_penCur.CreatePen(PS_SOLID, m_nPenWidth, RGB(0,0,0));
}

CStroke* CScribbleDoc::NewStroke()
{
      CStroke* pStrokeItem = new CStroke(m_nPenWidth);
             m_strokeList.AddTail(pStrokeItem);
SetModifiedFlag();  // Mark the document as having been modified,
for
```

```
            // purposes of confirming File Close.
                    return pStrokeItem;
}




//////////////////////////////////////////////////////////////////
////////////
// CStroke

IMPLEMENT_SERIAL(CStroke, CObject, 1)
CStroke::CStroke()
{
// This empty constructor should be used by serialization only
}

CStroke::CStroke(UINT nPenWidth)
{
                m_nPenWidth = nPenWidth;
}

void CStroke::Serialize(CArchive& ar)
```

```
{
                    if (ar.IsStoring())
                            {


                                ar << (WORD)m_nPenWidth;
                                            ar << m_alfa;
                                            ar << m_beta;
                                            ar << m_gama;
                                    m_pointArray.Serialize(ar);
                            }
                           else
                            {
                                                    WORD w;
                                                   ar >> w;
                                            ar >> m_alfa;
                                            ar >> m_beta;
                                            ar >> m_gama;
                                         m_nPenWidth = w;
                                    m_pointArray.Serialize(ar);



                            }
}
```

```
BOOL CStroke::DrawStroke(CDC* pDC)
{

                        CPen penStroke;
if (!penStroke.CreatePen(PS_SOLID, m_nPenWidth, RGB(0,0,0)))
                                                return FALSE;
       CPen* pOldPen = pDC->SelectObject(&penStroke);

pDC->SetMapMode(MM_LOMETRIC);
pDC->SetViewportOrg(0,250);

                pDC->MoveTo(m_pointArray[0]);
       for (int i=1; i < m_pointArray.GetSize(); i++)
                                {
                                  pDC->LineTo(m_pointArray[i]);
                                }

                pDC->SelectObject(pOldPen);
                        return TRUE;
}

void CScribbleDoc::OnDatosAjuste()
{
```

```
        // TODO: Add your command handler code here

}
// ScribVw.cpp : implementation of the CScribbleView class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include "stdafx.h"
#include "Scribble.h"
#include "dos.h"
#include "ScribDoc.h"
#include "ScribVw.h"
#include "DATOS.h"
#include "math.h"
#include "conio.h"
```

```
#include "stdio.h"



#ifdef _DEBUG
#define new DEBUG_NEW

#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#define PORTADDRESS 0x378
#define DATA            PORTADDRESS+4
#define STATUS          PORTADDRESS+1
#define CONTROL         PORTADDRESS+2
#define ECR           PORTADDRESS+0x402


int                interflag,registroesp,
bitest,vall=0x01,i=0,j=0,k=0,T1=0,T2=0,T3=0,T4=0,q=0,dfiltro=19;
float T,TA;
long ENDA[700],F1=0,F2=0,F3=0;
CString                    m_alfa;
long    m_beta, m_gama, m_mulX, m_mulY, c_fecha=4;



/////////////////////////////////////////////////////////////////////////
```

```
////////////

// CScribbleView

IMPLEMENT_DYNCREATE(CScribbleView, CView)

BEGIN_MESSAGE_MAP(CScribbleView, CView)
                //{{AFX_MSG_MAP(CScribbleView)
          ON_COMMAND(ID_DATOS_AJUSTE, OnDatosAjuste)
                        ON_WM_TIMER()
                    ON_WM_RBUTTONDOWN()
                      //}}AFX_MSG_MAP
                // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////
////////////
// CScribbleView construction/destruction

CScribbleView::CScribbleView()
{
            // TODO: add construction code here
```

```
}

CScribbleView::~CScribbleView()
{
}

BOOL CScribbleView::PreCreateWindow(CREATESTRUCT& cs)
{
// TODO: Modify the Window class or styles here by modifying
                  //  the CREATESTRUCT cs

             return CView::PreCreateWindow(cs);
}

////////////////////////////////////////////////////////////////////
////////////
// CScribbleView drawing

void CScribbleView::OnDraw(CDC* pDC)
{
             CScribbleDoc* pDoc = GetDocument();
                      ASSERT_VALID(pDoc);
```

```
 // The view delegates the drawing of individual strokes to
                // CStroke::DrawStroke().
CTypedPtrList<CObList,CStroke*>& strokeList = pDoc->m_strokeList;
        POSITION pos = strokeList.GetHeadPosition();
                while (pos != NULL)
                        {
                                CStroke* pStroke =
strokeList.GetNext(pos);
                                pStroke->DrawStroke(pDC);
                        }

           SetTimer(ID_CLOCK_TIMER02,1000,NULL);

}

/////////////////////////////////////////////////////////////////////
////////////
// CScribbleView printing
```

```
BOOL CScribbleView::OnPreparePrinting(CPrintInfo* pInfo)
{
                    // default preparation
               return DoPreparePrinting(pInfo);
}

void CScribbleView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
      // TODO: add extra initialization before printing
}

void CScribbleView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
             // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////
////////////
// CScribbleView diagnostics

#ifdef _DEBUG
void CScribbleView::AssertValid() const
```

```
{
                    CView::AssertValid();
}

void CScribbleView::Dump(CDumpContext& dc) const
{
                       CView::Dump(dc);
}

CScribbleDoc* CScribbleView::GetDocument() // non-debug version
is inline
{
 ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScribbleDoc)));
              return (CScribbleDoc*)m_pDocument;
}
#endif //_DEBUG

////////////////////////////////////////////////////////////////
////////////
// CScribbleView message handlers


void CScribbleView::OnDatosAjuste()
{
```

```
        // TODO: Add your command handler code here
                        DATOS DIALOG1;
                    DIALOG1.DoModal();




}

void CScribbleView::Adquisicion()

{



int d, delay=14;
for (d=1; d<delay; d++)
{
    CClientDC dc(this);
    dc.BitBlt( 0,
0,dc.GetDeviceCaps(HORZRES),dc.GetDeviceCaps(VERTRES),
&m_dcRScreen, 0, 0, WHITENESS );
                dc.SetMapMode(MM_LOMETRIC);
                        int imax;
```

```
            int kmax;
 imax=dc.GetDeviceCaps(HORZRES);
 kmax=dc.GetDeviceCaps(VERTRES);
dc.SetViewportOrg(0,(kmax/2-77));

      _outp(CONTROL,0x24);

            do
                                                    {

     m_point[i].x=i*m_mulX;

   m_point[i].y=m_point[i+1].y;

            i++;



                                                    }
      while(i<=(imax-1));
            i=imax;
     interflag=_inp(STATUS);
```

```
                    if ((interflag)&&(vall))
                                                                    {

                       _outp(STATUS,01);

                     interflag=_inp(STATUS);
                                                                    }

            m_point[i].y=((_inp(DATA))-134)*m_mulY;
                     m_point[i].x=i*m_mulX;


                     dc.MoveTo(m_point[i]);
// Presentacion:Lazo de presentacion y limpieza.
// OFFSET DC 234
                                do
                                                                    {
                                                                  i--;

                                                dc.LineTo(m_point[i]);

                                                                    }
                             while(i>=0);
        if (((115*m_mulY)>(m_point[imax].y))&&(T2==0))
```

```
                                {
                                                        T1=T1+1;
                                }
      else if (((100*m_mulY)<(m_point[imax].y))&&(T1>0))
                                {
                                                        T2=T2+1;
                                }
                    else if ((T1>0)&&(T2>0))
                                {
                                                    T=(T1+T2)*0.0001;
                                                            F1=(1/T);
                                                             T1=T2=0;

                                                  if ((F1>110)||F1<40)
                                                                                {

_outp(061,0xff);

Aresguardo();
                                                                                }
```

```
}

dc.BitBlt( 0,
225,dc.GetDeviceCaps(HORZRES),dc.GetDeviceCaps(VERTRES),
&m_dcRScreen, 0, 0, WHITENESS );
dc.SetMapMode(MM_LOMETRIC);
int jmax;

jmax=dc.GetDeviceCaps(HORZRES);
kmax=dc.GetDeviceCaps(VERTRES);
dc.SetViewportOrg(0,(kmax-120));
_outp(CONTROL,0x20);

do
{

m_point2[j].x=j*m_mulX;

m_point2[j].y=m_point2[j+1].y;
```

```
                        j++;

                                                              }
                  while(j<=(jmax-1));
                        j=jmax;
                 interflag=_inp(STATUS);
                if ((interflag)&&(vall))
                                                              {

                   _outp(STATUS,01);

                 interflag=_inp(STATUS);
                                                              }

            m_point2[j].y=(_inp(DATA))*m_mulY;
                 m_point2[j].x=j*m_mulX;
// Filtro
          if ((m_point2[j].y)-10>(m_point2[j-1].y))
                                                                {
                                                              q++;
                                                    if (q<=dfiltro)
```

```
m_point2[j].y=m_point2[j-1].y;
                                                                }
                            else if          ((m_point2[j].y)-
10<(m_point2[j-1].y))
                                                                {
                                                              q=0;
                                                                }
//Fin del Filtro
                    dc.MoveTo(m_point2[j]);
// Presentacion:Lazo de presentacion y limpieza.

                                do
                                                                {
                                                              j--;

                                          dc.LineTo(m_point2[j]);

                                                                }
                            while(j>=0);
       if (((120*m_mulY)>(m_point2[jmax].y))&&(T4==0))
                                {
                                                        T3=T3+1;
                                }
```

```
    else if (((120*m_mulY)<(m_point2[jmax].y))&&(T3>0))
                                {
                                                        T4=T4+1;
                                }
                        else if ((T3>0)&&(T4>0))
                                {
                                                TA=(T3+T4)*0.0004;
                                                  F2=4*log(1/(T));
                                                        T3=T4=0;
                                                if ((F2>50)||F2<5)
                                                                        {

_outp(061,0x24);

Aresguardo();
                                                                        }


                                }

}
}
```

```
void CScribbleView::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
                        switch (nIDEvent)
                                {

                    case ID_CLOCK_TIMER02:
                                            if (c_fecha==m_gama)

                                                                {

                   KillTimer(ID_CLOCK_TIMER02);

                SetTimer(ID_CLOCK_TIMER01,1,NULL);
                                                                }

                                                            break;
                    case ID_CLOCK_TIMER01:
                                            if (c_fecha==m_gama)
                                                                {

                        Adquisicion();
```

```
                                                                    }
                                                                  else

                    KillTimer(ID_CLOCK_TIMER01);
                                                                 break;

                                        }
                     CView::OnTimer(nIDEvent);
}

void CScribbleView::Aresguardo()
{
CFile f;
f.Open("C:\\Try.TRY",CFile::modeCreate | CFile::modeWrite);
CArchive ar(&f,CArchive::store);
CClientDC dc(this);
int jmax;
jmax=dc.GetDeviceCaps(HORZRES);
                            if (k==0)
{
          m_pStrokeCur = GetDocument()->NewStroke();
}
do
```

```
                                                                {
                                                        m_pStrokeCur-
>m_pointArray.Add(m_point[k]);
        k++;
                                                                }
while(k<=jmax*3);

                                                        ar << m_alfa;
                                                        ar << m_beta;
                                                        ar << m_gama;
                                        m_pointArray.Serialize(ar);
ar.Close();
f.Close();

}




void CScribbleView::OnRButtonDown(UINT nFlags, CPoint point)
{
// TODO: Add your message handler code here and/or call default
CClientDC dc(this);
int jmax;
```

```
jmax=dc.GetDeviceCaps(HORZRES);
                          if (k==0)
{
         m_pStrokeCur = GetDocument()->NewStroke();
}

                                do
                                                                {

                                                        m_pStrokeCur-
>m_pointArray.Add(m_point[k]);
        k++;
                                                                }
while(k<=jmax*3);



            CView::OnRButtonDown(nFlags, point);
}
#if
!defined(AFX_DATOS_H__F9BCB822_9A9C_11D8_85AE_DC6FF9B3212D__INCLU
DED_)
#define
AFX_DATOS_H__F9BCB822_9A9C_11D8_85AE_DC6FF9B3212D__INCLUDED_
```

```
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// DATOS.h : header file
//

extern long         F1,F2;
/////////////////////////////////////////////////////////////////////
////////////
// DATOS dialog

class DATOS : public CDialog
{
// Construction
public:

   DATOS(CWnd* pParent = NULL);   // standard constructor
    friend class CStroke;
                friend class CScribbleView;



// Dialog Data
                    //{{AFX_DATA(DATOS)
```

```
                enum { IDD = IDD_DIALOG1 };
                        CButton                 m_pipi;
                        CButton                 m_tutu;
                         long                  m_fecha;
                         long                   m_edad;
                        CString                 m_name;
                         long           Y_Multiplicador;
                         long           X_Multiplicador;
                        float             m_frecuencia2;
                        float             m_frecuencia1;
                      //}}AFX_DATA


// Overrides
     // ClassWizard generated virtual function overrides
                    //{{AFX_VIRTUAL(DATOS)
                          protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
                       //}}AFX_VIRTUAL

// Implementation
protected:
```

```
            // Generated message map functions
                    //{{AFX_MSG(DATOS)
                afx_msg void OnButton2();
                afx_msg void OnButton1();
                afx_msg void OnButton4();
                        //}}AFX_MSG
                  DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_DATOS_H__F9BCB822_9A9C_11D8_85AE_DC6FF9B3212D__INCLU
DED_)
// MainFrm.h : interface of the CMainFrame class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
```

```
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CMDIFrameWnd
{
                DECLARE_DYNAMIC(CMainFrame)
public:
                        CMainFrame();

// Attributes
public:

// Operations
public:

// Overrides
     // ClassWizard generated virtual function overrides
                //{{AFX_VIRTUAL(CMainFrame)
```

```
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
                        //}}AFX_VIRTUAL

// Implementation
public:
                    virtual ~CMainFrame();
#ifdef _DEBUG
                virtual void AssertValid() const;
            virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
                    CStatusBar  m_wndStatusBar;
                     CToolBar    m_wndToolBar;

// Generated message map functions
protected:
                      //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
                                // NOTE - the ClassWizard will add
and remove member functions here.
                                //    DO NOT EDIT what you see in
these blocks of generated code!
                           //}}AFX_MSG
```

```
                    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
// Scribble.h : main header file for the SCRIBBLE application
//
// This is a part of the Microsoft Foundation Classes C++ library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
```

```
/////////////////////////////////////////////////////////////////////////
////////////
// CScribbleApp:
// See Scribble.cpp for the implementation of this class
//

class CScribbleApp : public CWinApp
{
public:
                        CScribbleApp();

// Overrides
     // ClassWizard generated virtual function overrides
                //{{AFX_VIRTUAL(CScribbleApp)
                            public:
                virtual BOOL InitInstance();
                            //}}AFX_VIRTUAL

// Implementation

                  //{{AFX_MSG(CScribbleApp)
                 afx_msg void OnAppAbout();
                                // NOTE - the ClassWizard will add
```

```
and remove member functions here.
                                //    DO NOT EDIT what you see in
these blocks of generated code !
                            //}}AFX_MSG
                        DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////
////////////
// ScribDoc.h : interface of the CScribbleDoc class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.
/////////////////////////////////////////////////////////////////////
////////////
```

```
extern CString                                          m_alfa;
extern long         m_beta, m_gama, m_mulX, m_mulY, c_fecha;


////////////////////////////////////////////////////////////////////
////////////
// class CStroke
//
// A stroke is a series of connected points in the scribble
drawing.
// A scribble document may have multiple strokes.

class CStroke : public CObject
{
public:
                  CStroke(UINT nPenWidth);
                     friend class DATA;
             UINT                    m_nPenWidth;

protected:
                          CStroke();
                  DECLARE_SERIAL(CStroke)
```

```
// Attributes
protected:
            // one pen width applies to entire stroke
public:
CArray<CPoint,CPoint>  m_pointArray;   // series of connected
points


// Operations
public:
                  BOOL DrawStroke(CDC* pDC);



public:
            virtual void Serialize(CArchive& ar);
};


class CScribbleDoc : public CDocument
{

// Propiedad ----------------------------------
```

```
private:
                    CBitmap* m_pArches;
                   CBitmap* m_pBackground;

public:
      CBitmap* GetBackground() {return m_pBackground;}
               void LoadBackground(CBitmap*);
                  void UnloadBackground();

protected: // create from serialization only
                        CScribbleDoc();
               DECLARE_DYNCREATE(CScribbleDoc)

// Attributes
protected:
   // The document keeps track of the current pen width on
   // behalf of all views. We'd like the user interface of
  // Scribble to be such that if the user chooses the Draw
 // Thick Line command, it will apply to all views, not just
          // the view that currently has the focus.

UINT            m_nPenWidth;        // current user-selected pen
width
CPen            m_penCur;           // pen created according to
```

```
            // user-selected pen style (width)
public:
      CTypedPtrList<CObList,CStroke*>     m_strokeList;
    CPen*           GetCurrentPen() { return &m_penCur; }

// Operations
public:
                    CStroke* NewStroke();
            virtual void Serialize(CArchive& ar);
// Overrides
     // ClassWizard generated virtual function overrides
                //{{AFX_VIRTUAL(CScribbleDoc)
                            public:
                virtual BOOL OnNewDocument();

     virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
                virtual void DeleteContents();
                        //}}AFX_VIRTUAL

// Implementation
public:
```

```
                virtual ~CScribbleDoc();
#ifdef _DEBUG
            virtual void AssertValid() const;
         virtual void Dump(CDumpContext& dc) const;
#endif

protected:
              void           InitDocument();

// Generated message map functions
protected:
                //{{AFX_MSG(CScribbleDoc)
              afx_msg void OnDatosAjuste();
                        //}}AFX_MSG
                  DECLARE_MESSAGE_MAP()
};
// ScribVw.h : interface of the CScribbleView class
//
// This is a part of the Microsoft Foundation Classes C++
library.
// Copyright (C) 1992-1997 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
```

```
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.
////////////////////////////////////////////////////////////////////
////////////

class CScribbleView : public CView
{

protected: // create from serialization only
                    CScribbleView();
             DECLARE_DYNCREATE(CScribbleView)

// Attributes
public:
               CScribbleDoc* GetDocument();
                  void Adquisicion(void);
                 void Adquisicion2(void);
                  void Aresguardo(void);
                    friend class DATOS;
           CArray<CPoint,CPoint>  m_pointArray;
         CPoint      m_point[2000],m_point2[2000];
                     CBrush brCross;
```

```
                        CBrush *pBrush;
                              CDC
                         m_dcRScreen;
                            CBitmap
                         m_bmRScreen;

protected:

    CStroke*    m_pStrokeCur;   // the stroke in progress

CPoint      m_ptPrev,m_toto; // the last mouse pt in the stroke
in progress
                            CStroke*                m_pDatos;

// Operations
public:

// Overrides
     // ClassWizard generated virtual function overrides
                //{{AFX_VIRTUAL(CScribbleView)
                            public:
                            virtual     void OnDraw(CDC* pDC);
       virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
```

```
                        protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
 virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
  virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
                        //}}AFX_VIRTUAL

// Implementation
public:
                virtual ~CScribbleView();
#ifdef _DEBUG
             virtual void AssertValid() const;
         virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
                //{{AFX_MSG(CScribbleView)
              afx_msg void OnDatosAjuste();
           afx_msg void OnTimer(UINT nIDEvent);
   afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
                        //}}AFX_MSG
                    DECLARE_MESSAGE_MAP()
```

```
};

#ifndef _DEBUG  // debug version in ScribVw.cpp
inline CScribbleDoc* CScribbleView::GetDocument()
   { return (CScribbleDoc*)m_pDocument; }
#endif
```

APENDICE C. COSTOS DEL PROYECTO.

ADQUISITOR:

IC ADC 0804	$20.00
IC 4066	$06.00
IC 74LS14	$02.00
IC 74LS123	$02.00

IC LF356N	$01.00
RECEPTOR:	
ICL 7641P	$03.00
IC LM565	$12.00
TRANSMISOR:	
ELECTRODOS DESECHABLES	$04.00
NTC 10 k	$10.00
IC LF351N	$09.00
IC LM 556	$02.00
RESISTORES	$07.00
CAPACITORES	$10.00
CIRCUITOS IMPRESOS	$20.00
CAJAS DE RESGUARDO	$17.00
TOTAL	**$125.00**

APENDICE D. Puerto paralelo

APENDICE E. datos NTE 2053