UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INGENIERÍA

TESIS

Desarrollo de un sistema para la adquisición de señales biométricas en roedores

Que como requisito para obtener el título de
Ingeniero Electromecánico
con
Línea Terminal en Mecatrónica

PRESENTA:
Victor Hugo Hernández Morales

DIRIGIDO POR:
Director: Dr. Roque Alfredo Osornio Ríos
Co-Director: Dr. Jesús Rooney Rivera Guillén

Centro Universitario
San Juan del Rio, Querétaro. Agosto 2015

RESUMEN

La adquisición de señales biométricas ha sido indispensable para monitorear el cambio fisiológico o problemas presentados por algún medicamento o daños adquiridos a lo largo de la vida. Se desarrolló un sistema de adquisición de señales biométricas, empleado para mostrar el Electrocardiograma de un roedor y lograr observar sus cambios fisiológicos mediante el comportamiento cardíaco. El proyecto consiste en monitorear el ECG en una computadora o laptop, concentrándose en obtener las derivaciones bipolares del roedor de forma no invasiva para evitar dañarlo. La técnica para adquirir el ECG de roedores es a partir de electrodos o agujas de plata que se colocan en sus patas, formando el diferencial eléctrico entre dos puntos teniendo un tercero como referencia. La adquisición del ECG se procesa mediante las siguientes etapas: etapa de preamplificacion, etapa de filtrado empleando filtros analógicos butterworth; filtro pasabanda con frecuencias de corte 0.05 Hz a 150 Hz y filtro notch a 60 Hz, etapa de amplificación y digitalización donde se amplifica la señal ECG para hacer una conversión analógico-digital, usando un convertidor de alta resolución de 24 bits a una capacidad de 8ksps (muestras por segundo). Todas estas etapas son alimentadas con baterías para reducir el ruido de la red eléctrica y del medio ambiente. Los datos adquiridos se representan en una interfaz en la computadora para visualizar el Electrocardiograma. Empleando estas etapas la señal ECG se adquiere con una resolución alta y con niveles de ruido bajos, ya que es necesario reducir el ruido de señales del ambiente, del rozamiento del electrodo con la piel o patas y ruido por el movimiento. Los resultados fueron satisfactorios, al representar la señal ECG en la computadora de varios individuos de pruebas, como el roedor experimental así como implementar el sistema en humanos. Debido a que el sistema se desarrolla con una tarjeta FPGA, se deja la plataforma para una amplia gama de aplicaciones. Así como el sistema de digitalización, puede emplearse para adquirir otras bioseñales ya sea para humanos, roedores u otros animales.

(**Palabras clave**: señales biométricas, Electrocardiograma (ECG), derivaciones bipolares, preamplificacion, etapa de filtrado, digitalización)

DEDICATORIAS

"Dedicado a mi madre Rosa María, hermanos Oliver, Diego, Miguel y Leonardo

A toda mi familia Morales Hernández

Y a mi novia Elizabeth"

Agradecimientos

A todas las personas que me brindaron su amistad, compañerismo y apoyo, ya que sin esas personas no sería posible la persona que soy ahora.

A mi familia que siempre me ha apoyado en todo momento, la cual ha sido la principal motivación, soporte e inspiración para lograr concluir este proyecto de vida.

A mis amigos que forman parte de una segunda familia, una que me ofrece su ayuda y consejos.

A los profesores Dr. Roque Alfredo, al Dr. Jesús Rooney, al Dr. Luis Morales, al M. en C. Benigno Muñoz por su confianza en este proyecto y apoyo en la realización de esta tesis.

A todos los profesores de la carrera por brindarme su conocimiento y paciencia, por hacer mi formación profesional de alta calidad tanto académicamente como personalmente.

A la Universidad Autónoma de Querétaro por permitirme concluir mis estudios y brindarme las herramientas necesarias para formar mi carrera profesional.

ÍNDICE GENERAL

CONTENIDO **PÁGINA**

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Introducción

A lo largo del tiempo ha sido indispensable conocer el funcionamiento de herramientas, procesos o fenómenos naturales de los cuales se interpreta su comportamiento con el fin de obtener un beneficio y así manipular o mejorar los sistemas, pero tenemos que incluir mecanismos que se deben entender para el bienestar propio, como lo son los seres vivos ya que se puede aprender y mejorar el estilo de vida a partir de analizar el comportamiento de los organismos, ya que también somos un sistema se logra apreciar el funcionamiento y la interacción con el medio a partir de representar los organismos con métodos apropiados de visualización, esto se logra con sensores, dispositivos electrónicos o eléctricos, etc., de esta forma se sabe cuándo se presentan alteraciones o reacciones en el cuerpo ante distintos cambios físicos. Mencionando algunos puntos de la importancia del monitoreo de señales de los organismos, uno de ellos es el corazón ya que influye en procesos bioquímicos y metabólicos, es decir, mostrando las señales del corazón (ECG; Electrocardiograma) conocemos varias características que ayudan a identificar problemas o cambios del sistema.

En este trabajo se presenta un dispositivo el cual muestra el Electrocardiograma de un roedor, el Electrocardiograma es el ritmo cardíaco del corazón, el cual representa la actividad fisiológica del ser vivo, es decir, se puede predecir o informar sobre algún padecimiento físico a partir de observar las señales biométricas del corazón. La prioridad sobre este sistema es no invadir el cuerpo del roedor evitando cirugías, midiendo las señales biométricas en un sistema vivo, así que se concentrará en detectar las señales del corazón a partir de un circuito electrónico diseñado por el tesista, por lo que se requiere realizar varias etapas para preparar el dispositivo, las cuales serán descritas por temas. Es un dispositivo donde la transmisión de señales debe tener un trato adecuado para evitar pérdidas de datos

se procura cuidar las etapas de acondicionamiento. Primero se adecuara una etapa de voltaje de referencia ya que es la fuente de alimentación de los circuitos debemos evitar caídas de voltaje y adecuar los voltajes que vayamos a requerir. Como siguiente paso se describirá una etapa de pre-amplificación y filtrado, la cual se deberá adecuar al registro de señales, voltajes que se adquieren del roedor y frecuencias presentes en el Electrocardiograma, en esta etapa se trataran las señales que se adecuen al sistema tomando las limitaciones con las que se cuenta, las cuales se retomaran en su tema. La etapa de conversión de datos, es fundamental ya que se requiere mostrar los datos en una pantalla por tanto la pérdida de valores causa deficiencias en el sistema, al igual que en todas las etapas se necesitan dispositivos especiales, es adecuado la búsqueda de elementos que recapitularemos en su sección. Para ultimar detalles del proyecto aquí presentado se mencionara brevemente el cometido de esta tesis, se realiza un dispositivo de adquisición de la bioseñal Electrocardiograma de un roedor, donde describiremos las etapas necesarias así como elementos auxiliares que beneficiaran la adquisición de datos adecuadamente.

Ahora bien este tema tiene algunos predecesores, debido a que es un método no invasivo (se desea evitar el daño al animal) y hacer eficiente la adquisición de señales, algunas empresas han optado por esta opción debido a su practicidad, aunque también se tienen restricciones por la movilidad de los ratones pero se ve la forma de afrontar los inconvenientes.

1.1 Antecedentes

La investigación sobre nuevos suplementos alimenticios, medicinas y reacciones fisiológicas hacen que las pruebas sean cada vez más cuidadosas para determinar los efectos primarios y secundarios, por lo tanto se han llevado a cabo proyectos, tesis, investigaciones, etc., con el fin de encontrar la manera adecuada de

registrar los cambios. Enfocando estudios en el tema del Electrocardiograma se rescatan algunos relacionados con el tema, ya que pueden guiar este trabajo así como resaltar puntos importantes que se deben tomar en cuenta para conseguir mejorar el medio de adquisición de bioseñales (ECG).

A propósito de los trabajos desarrollados, primero se enuncia una tesis realizada, Montes (2000), en la Universidad Autónoma de Querétaro, donde se "desarrolló un sistema de detección de Electrocardiograma para seres humanos", el cual consistía en realizar un electrocardiógrafo el cual fuera más accesible, con menor costo, fácil manejo y adaptable a una PC. Se pueden encontrar puntos importantes a resaltar ya que se diseñó un circuito electrónico, el cual tiene etapas de amplificación, filtro y conversión, se cuidaron detalles como protección al humano por la alimentación en corriente alterna, es decir, aislamiento y seguridad eléctrica.

En dicha tesis, se sabe que se maneja el Electrocardiograma para humanos lo que indica que los niveles de voltajes son distintos que en roedores, por lo que es consistente que a nivel regional con la UAQ, no se tiene un sistema de este tipo, aunque se trata de hacer uso de procedimientos o experiencias obtenidas en el trabajo de Montes. Se debe resaltar que a nivel nacional se tienen algunos antecedentes en el área de bioseñales tanto como algunos artículos y tesis que guían este trabajo hacia una mejor presentación.

A nivel nacional se encuentra que en el Instituto Politécnico Nacional, se presenta un "diseñó de un sistema multicanal portátil de registro de bioseñales", se llevó a cabo la investigación del sistema portátil multicanal de registro y amplificación de bioseñales. En donde se contempla la flexibilización del sistema hacia aplicaciones cardíacas para hacerlo compatible con estudios EEG

(electroencefalograma) y ECG (Electrocardiograma). El equipo portátil muestra etapas de acondicionamiento de señales similares a cualquier dispositivo empleado en el tratado de bioseñales, contempla amplificación, digitalización, transmisión y recepción. (Tejeda, 2009).

Una tesis presentada en la Universidad Autónoma de Yucatán, Gonzalez (2010) muestra cómo se "diseñó y construyó un sistema para la detección de señales electromiográficas", se adquirieren las señales electromiográficas provenientes de músculos. La finalidad del trabajo consiste en plantear la primera etapa para la construcción de un sistema de prótesis mioelectrica que puede reconocer los potenciales de acción generados por los músculos. Se analiza por medio de electrodos en los cuales se le aplican las etapas de acondicionamiento general; amplificación, filtrado y conversión analógica digital, se especifican los procedimientos que emplean para manejar voltajes en rango de milivolts.

Los trabajos anteriores se presentan como análisis de bioseñales, se debe mencionar que no se aplican a señales en ratones, por lo tanto los niveles de voltaje difieren pero al ser adquiridos de órganos vivos proporcionan interés y se aprenden algunas técnicas o procesos requeridos, se presentan etapas de acondicionamiento similares a emplear por esta razón son etapas las cuales no se pueden omitir, cada proyecto maneja diferentes dispositivos, tanto electrónicamente como para procesar los datos pero plantean en común amplificación, filtros y digitalización de las señales. Por eso es que representan antecedentes en esta tesis, se muestran acondicionamientos de bioseñales, así como una representación gráfica en una pantalla LCD o mediante impresión de datos en tiempo real. Ya que se encuentran diseños y estudios alrededor de México, esto indica que se tiene interés en el estudio de señales biométricas para la creación o manipulación de diferentes dispositivos, así como el monitoreo de órganos físicos vivos.

Con el fin de justificar más la viabilidad de este trabajo, se realizó una búsqueda para encontrar países donde se estén creando dispositivos graficadores de bioseñales, ya que es importante dar a conocer la relevancia del trabajo y aplicaciones que requieren sistemas de adquisición de señales biométricas.

En la Universidad Industrial de Santander, Figueroa & Medina (2009) crearon el "diseño e implementación de un electrocardiógrafo inalámbrico digital para ratas de laboratorio utilizando tecnología bluetooth", este trabajo presenta un diseño apropiado para que el Electrocardiograma pueda ser llevado por una rata Wistar adulta sin generarles ningún inconveniente. Las especificaciones de este trabajo muestran características interesantes ya que el sistema se incorpora en la rata y pueden registrarse las señales en movimiento; en diferentes medios o bajo el efecto determinado de fármacos. Se acoplaron 3 etapas de filtrado, ya que se trata de eliminar el ruido del movimiento, propias del sistema, entre otras. Debido a que emplean un sistema en microprocesadores el procesar las señales es más costoso en recursos electrónicos y deficiente por lo que implementarlo en una tarjeta basada en FPGA puede expandirse el procesamiento a distintos niveles.

El presente trabajo realizado por Figueroa y Medina, se muestra un prototipo sobre la obtención de señales biométricas en ratas, donde la velocidad de transferencia de datos es restringida debido al uso de bluetooth y procesador empleado, este sistema se adecuo para un cierto tamaño y peso de ratas Wistar, es decir, solo para unos grupos específicos de ratas. Con el fin de mejorar el trabajo se realiza una velocidad de transferencia mayor y aplicando el Electrocardiograma a un grupo mayor de ratas, permitiendo un mayor peso y tamaño.

Se encuentra en la Universidad de AZUAY el "diseño y construcción de un monitor de tres parámetros fisiológicos", Feijo & Reyes (2010) se basan en el diseño

y construcción de un monitor de parámetros fisiológicos basado en microcontroladores. El equipo es capaz de monitorear tres bioseñales diferentes: Electrocardiograma, señal fonocardiográfica y ritmo respiratorio. Este trabajo se basó en etapas analógicas y digitales, se incluye la visualización de las principales señales electrocardiográficas y la frecuencia cardíaca. La técnica fonocardiográfica pretende monitorear los sonidos del corazón con el fin de ser estudiados.

Se da a conocer estos proyectos internacionales, con el motivo de entender nuevas técnicas así como rectificar las etapas básicas de obtención de las señales biométricas. En el trabajo de Feijo y Reyes se plantea una opción más para estudiar el Electrocardiograma en humanos por lo tanto solo sirve de ayuda auxiliar en el acondicionamiento de señales biométricas. Ya que se ha dado a conocer algunos trabajos de graduación fuera del país, se resalta que se han involucrado varios sistemas en apoyo a diferentes áreas bioquímicas con propósitos para ratas y humanos.

Como se ha mencionado se cuenta con tesis a diferentes niveles, ya que es de importancia citar fuentes de trabajo en nuestra región, nación e internacional para dar una gama previa de sistemas de monitoreo, no obstante se encuentran distintos artículos donde se estudia el Electrocardiograma en roedores de laboratorio, por tal motivo es necesario hacer mención de ellos.

El artículo “Detección de signos vitales en ratas mediante métodos no invasivos”, Flores Chávez, et al (2002) describen un equipo que permite obtener parámetros fisiológicos en ratas, por medio de métodos incruentos y minimizando las molestias al espécimen. Los parámetros fisiológicos son: Electrocardiograma (ECG), pulso pletismográfico (PP), presión sistólica (PS), presión diastólica (PD), frecuencia cardíaca (FC) y frecuencia respiratoria (FR). Este sistema se adapta a una caja especial de acrílico a la que se le añadió una base plana de acrílico que contiene

electrodos para obtener el Electrocardiograma diferencial, de donde se extrae la frecuencia cardíaca. Cada señal, luego de amplificarse y filtrarse, pasa a un convertidor analógico-digital, y de aquí a una computadora personal, cuya programación permite presentar, procesar y almacenar los datos. Este equipo se probó en 22 ratas.

Un trabajo importante de Farmer & Levy (1967), relacionado con bioseñales es "Un método simple para la grabación del Electrocardiograma y el ritmo cardíaco de animales conscientes", se detectan mediante una placa con 4 electrodos, las señales eléctricas detectadas por los electrodos de placa fueron amplificadas por una ACI y preamplificadores. Este método se ha utilizado para obtener grabaciones de ratones conscientes, ratas, gatos y perros, el contacto de las patas sobre la placa se logra mediante el uso de compresas de gasa humedecido con solución salina isotónica. Se probaron distintas drogas en los animales para analizar las reacciones.

En la instrumentación de un laboratorio se desarrolló un artículo por parte de Heredia & Góngora (1994) nombrado "Evaluación de una tarjeta de computadora para la adquisición y desplegado de señales electrocardiográficas de ratas", se describe un sistema para la adquisición, desplegado y almacenaje de señales ECG el cual contiene; circuitos de filtrado, una fuente de alimentación aislada, resolución de 12 bits y un programa de control desarrollado en Turbo Pascal.

Se describe en el trabajo de Soldan, et al (1997) la "Instrumentación para ECG y respiración de ratas despiertas sin aprisionar" la grabación del Electrocardiograma y respiración de una rata despierta y con libertad de movimiento. La técnica presentada requiere de una simple cirugía incrustando dos electrodos y un cable de tierra donde se colocan sobre el pecho vía subcutánea, a partir de estos

cables es posible recoger el ECG. Utilizando filtrado de la señal se pudo obtener ECG de ratas anestesiadas y despiertas sin la necesidad de los artefactos de movimiento.

Un artículo nombrado “Método no invasivo para la grabación del Electrocardiograma en ratas conscientes: viabilidad para el análisis de la variabilidad de la frecuencia cardíaca”, consiste en una herramienta bien establecida para la evaluación autónoma cardíaca en animales. El estudio prueba un método no invasivo para la grabación de ECG en ratas consientes, se diseñó una chaqueta de algodón elástica para adaptarse a la rata con dos piezas de electrodos de platino adjuntos en su superficie interior, permitiendo registrar el ECG en conciencia pero rasurando y tratando al animal bajo anestesia. Trabajo realizado por las personas Pereira Junior, et. al, (2010).

En el trabajo de Chu (2001), el “Método no invasivo de grabación de Electrocardiograma en ratones conscientes”, se desarrolla un sistema para obtener ECG en ratones conscientes sin anestesia o implantes y usando un software accesible por Internet creado para el análisis de señales ECG. El sistema incluye electrodos individuales incrustados en una plataforma configurada con 3 electrodos en contacto con 3 patas.

Un artículo sobre “Un sistema de imágenes de ultra alta resolución ECG de miocardio para pequeños animales” muestra las imágenes cardíacas de una cámara de centelleo. Se realizó un estudio de perfusión miocárdica en un ratón normal de 25 gramos a más de 400 latidos por minuto, donde obtuvieron la resolución submilimétrica con resolución espacial. (Wu, et. al, 1999).

Los párrafos anteriores mostrados reflejan una cantidad significativa de proyectos en la adquisición de señales en animales y humanos, donde se mencionan los procesos elaborados, herramientas empleadas, condiciones de trabajo, etc., todos

distinguiendo características y propósitos en toxicología, medicina o química. Pero el software o herramientas que se emplean en los trabajos mencionados son de tipo comercial, al ser de esta categoría el acceso o capacidad de procesamiento es limitada, por lo que se busca desarrollar un sistema de libre acceso con altas capacidades de expandir las tarjetas electrónicas o el procesamiento de las señales.

De esta forma se trata de mejorar la resolución, velocidad de transferencia, sin invadir o causar daño sobre la rata, mostrar las señales en un equipo de cómputo estándar haciendo fácil maniobrar el sistema de adquisición de señales biométricas, crear el sistema con menor costo y estandarizado para implementarlo en distintas ratas de peso y tamaño. Así como proporcionar el estudio y herramientas necesarias para abrir paso en este tema dentro de esta casa de estudios.

1.2 Objetivos

1.2.1 General

Desarrollar un sistema para la adquisición y monitoreo de la señal Electrocardiograma mediante un método no invasivo aplicado a roedores consientes bajo efectos toxicológicos empleando procesamiento en hardware.

1.2.2 Particulares

I. Adquirir conocimientos sobre señales biométricas, revisando los artículos y tesis previas a este proyecto para el acondicionamiento y procesamiento de la señal Electrocardiograma.

II. Definir los parámetros de amplificación, filtrado y digitalización así como los dispositivos electrónicos competentes para el acondicionamiento de la señal ECG.

III. Diseñar las tarjetas electrónicas PCB's (Printed Circuit Board) para las etapas de preamplificacion, filtrado y conversión analógica-digital para hacer una adquisición y transmisión de datos eficiente.

IV. Fabricar las tarjetas electrónicas PCB's para cada etapa de acondicionamiento de la señal Electrocardiograma; preamplificacion, filtrado y digitalización, para reducir el ruido se implementara en una máquina de diseño de PCB's.

V. Desarrollar pruebas de funcionamiento para las tarjetas electrónicas PCB's y verificar que la funcionalidad del sistema sea el correcto adquiriendo señales ECG de humanos y ratas.

VI. Conjuntar las etapas de acondicionamiento y digitalización que permita la conexión y funcionamiento general del sistema de tal forma que se observe la señal ECG en un osciloscopio.

VII. Crear una interfaz de comunicación y una interfaz de usuario que permita visualizar las señales del Electrocardiograma en una computadora para tener una visualización digital.

1.3 Descripción del problema

Actualmente se han desarrollado varios sistemas que miden el Electrocardiograma, en la región se cuenta pero para humanos, nacionalmente se cuentan con sistemas en roedores que realizan sus estudios bajo condiciones de tamaño, peso y en algunos se realizan cirugías al roedor para poder analizar su ritmo cardíaco, es decir, se daña la integridad del roedor. También se cuenta con investigaciones alrededor del mundo ya que se conocen sistemas que se han desarrollado y logrado resultados sin la necesidad de invadir el roedor pero restringiendo el uso o tratando al roedor, como rasurar el cuerpo para mejorar el registro de ECG, para obtener dichos sistemas es difícil tenerlos a la mano o se desconoce el sistema y es difícil la manipulación, en algunos casos son comerciales por lo que el costo y mantenimiento de los sistemas es muy grande.

No dejando de lado que los experimentos en animales han incrementado debido a aspectos científicos y éticos, ya que la facilidad de experimentación es mucho mejor y es mejor vista para mejorar la calidad de vida, por esta razón es importante conocer patrones de los animales sujetos a pruebas. Actualmente es difícil depender de trabajos donde se desconocen diferentes factores o bajo qué condiciones se elaboraron dichos sistemas, aunque pueden ayudar a identificar procesos o etapas pero no con la profundidad que se requiere. Un punto importante, es mencionar que incluso en nuestra casa de estudios se requiere analizar señales biométricas para analizar los propios productos o productos toxicológicos que se generan dentro de la institución, por tal motivo es un problema conseguir aparatos a costos elevados y que funcionan bajo ciertas condiciones, así como comprar todos los insumos necesarios para el funcionamiento adecuado del sistema.

Así que el desarrollo de este proyecto permitirá dar solución e iniciar con una plataforma para adquirir señales biométricas, donde se requiere conocer los parámetros y condiciones del sujeto de prueba.

1.4 Justificación

Se sabe que los avances científicos ayudan a mejorar las condiciones de vida cuidando que los resultados sean los esperados, es por eso que en esta ocasión aplicando en el tema de adquisición de señales biométricas, se conoce que los roedores son usados en laboratorios es por eso que se brinda un sistema no invasivo de adquisición del Electrocardiograma, no es necesario hacer una exposición al cuerpo del ratón para registrar sus bioseñales de esta forma se hace resaltar el trabajo ya que no se afecta la vida de los roedores y consiguiendo estudios sin dañar la integridad de los seres vivos.

En este caso se trabaja en unión con la Facultad de Química debido a que se requiere un sistema de adquisición de señales biométricas en sus especímenes experimentales, ya que actualmente se producen suplementos alimenticios en dicha facultad por lo que se necesita registrar los cambios fisiológicos y registrar las causas-beneficios de los suplementos, por lo tanto se requiere analizar las señales del Electrocardiograma y denotar las características de ella. Mediante este proyecto se puede beneficiar a ambas facultades, en nuestra facultad se aprende y aplican los conocimientos de ingeniería y en la facultad vecina se proporciona la visualización del Electrocardiograma, con la facilidad que se requiere para los experimentos de toxicología, esto es una causa rescatable ya que se aplica en la misma institución. Se reducen costos ya que comercialmente son caros los sistemas de adquisición de Electrocardiograma. Al conseguir registrar cambios fisiológicos por medio del corazón, permite analizar al roedor ante los suplementos alimenticios de prueba que se estén produciendo en la Facultad de Química. De esta forma pueden avanzar con

los experimentos y concluir sobre las cualidades o aportaciones que contienen los suplementos toxicológicos.

Los factores que influyen en la fabricación de un electrocardiógrafo para ratones se enuncian a continuación: al ser un método de adquisición de señales biométricas no invasivo se procura proteger el roedor ya que son más seguros, es decir, no se invade al ser vivo se preserva el individuo de experimentación así podemos reducir costos al no extinguir la vida del roedor y continuar con el proceso de investigación, los costos de producción del sistema son menores en comparación con sistemas actualmente en el mercado, esto representa un punto importante al calcular los insumos y aparatos electrónicos necesarios para la obtención de señales.

Debido a que la globalización en el ámbito científico ha combinado varias ramas científicas, genera ambición el avanzar en todos los campos de estudio ya que se complementan para finalizar un proyecto y realizarlo con mayor eficiencia, es grato resaltar y considerar la aplicación de ingeniería en ramas donde se pueda brindar una mejor calidad de vida, ofreciendo el conocimiento adquirido a lo largo de los estudios ingenieriles hacia nuevos caminos que generen intereses académicamente y científicamente.

1.5 Planteamiento general

Se enmarca en la Figura 1-1, las etapas principales que se requieren para desarrollar el sistema de adquisición de la señal biológica Electrocardiograma. Se describe cada etapa para aclarar dicho procedimiento.

Los sensores se encargan de registrar los pulsos eléctricos generados por el corazón, en donde se plantean distintos electrodos para conducir los pequeños niveles de voltaje generados por el corazón. Se deben colocar en ciertas zonas para adquirir

con mejor resolución la bioseñal así mismo nos permite reducir el ruido de los movimientos musculares o ruidos ambientales incluso de los propios elementos de adquisición.

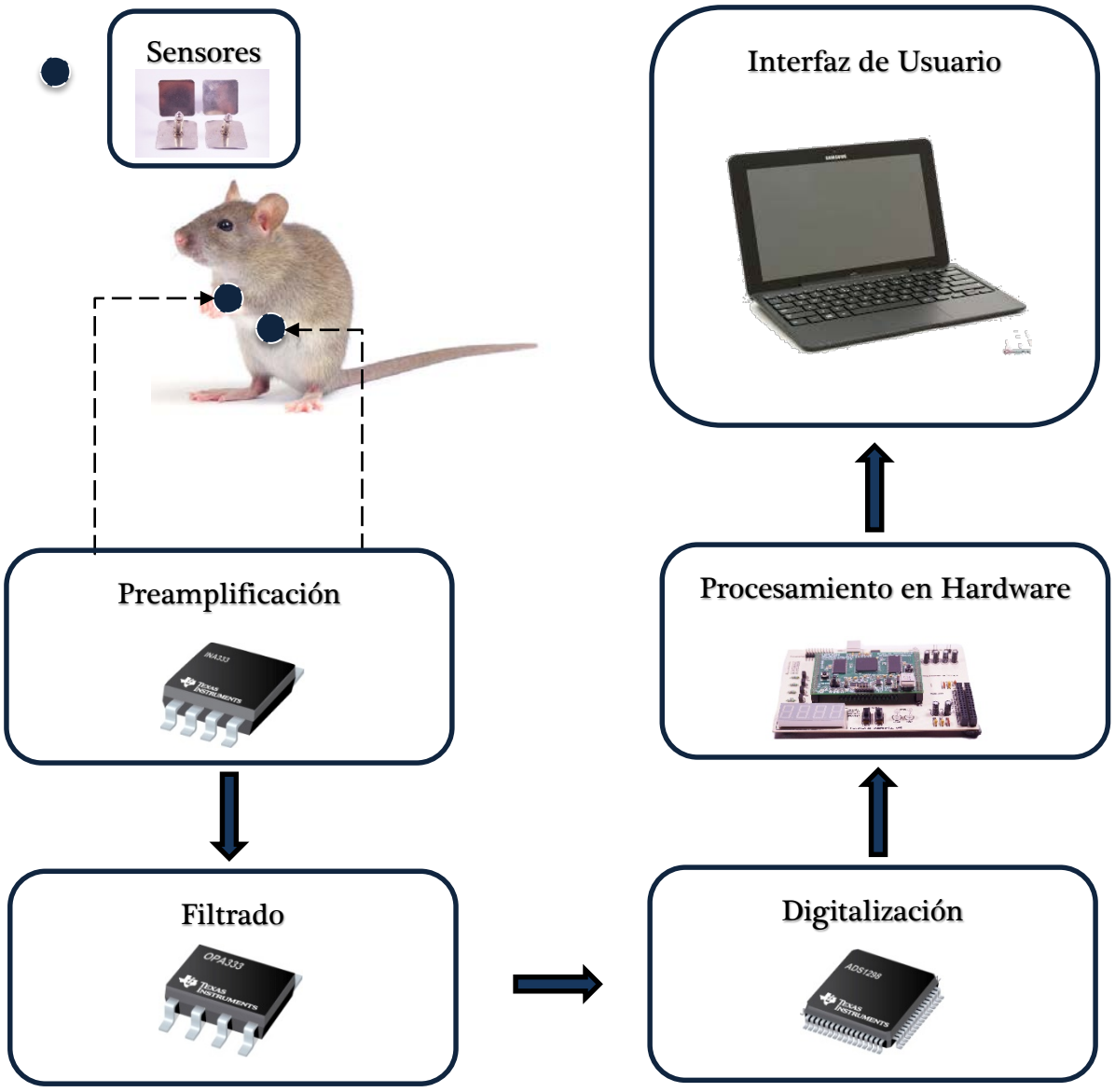


Figura 1-1 Diagrama General de Adquisición del Electrocardiograma

La etapa de pre-amplificación da un aumento en el nivel de voltaje adquirido de los electrodos, se emplea para que al filtrar las señales sea con un adecuado rango de voltaje y que sean detectables las señales por las etapas siguientes.

Mediante el filtrado se reduce el ruido del ambiente, el movimiento muscular, ruido ambiental provocado por las redes eléctricas y ruido propio de la transferencia de los datos. Se conjuntan las etapas de preamplificacion, filtrado y digitalización, debido a que el filtro debe proporcionar una mejor resolución de los datos. El almacenamiento y transmisión de datos registra los datos y permite una interfaz de transferencia para que sea posible la visualización del Electrocardiograma en una computadora.

Como elemento final, la interfaz de usuario permite observar la señal ECG en la pantalla de una computadora para que se puedan analizar por el usuario o experto en el comportamiento del miocardio.

Con las etapas descritas se puede interpretar que se consigue adquirir las señales eléctricas del corazón mediante un método no invasivo y procurando tener una resolución alta así como una digitalización eficaz que permita observar y manipular las señales.

2. Fundamentación Teórica

2.1 Estado del arte

Fundamentalmente los avances científicos en la obtención de señales biométricas se ha incrementado para obtener las bioseñales cada vez con mejores técnicas y desarrollando mejores sistemas, enfocándose en señales biométricas de un roedor, se puede describir que en un principio se generaban las señales a partir de inserción de electrodos dentro del individuo a estudiar, es decir, para capturar las señales del Electrocardiograma se tiene que insertar en los órganos vivos a cuerpo abierto, una placa semiconductora que pueda transmitir los cambios eléctricos del corazón, donde las incisiones se realizan en ciertas partes del cuerpo. Con el paso del tiempo la tecnología ha podido brindar cambios en los sistemas de adquisición de señales biométricas, los sensores de adquisición del Electrocardiograma se han actualizado a tal grado de que sea innecesario medir los pulsos eléctricos del corazón realizando cirugías, se logra con sensores capacitivos o electrodos de aguja de los cuales serán implementados en este trabajo. (Chu, 2001)

La velocidad de transmisión de datos es una característica importante en sistemas analógicos, de tal forma que se puede identificar para saber la capacidad del sistema, cuando se muestrea y transmiten datos a una alta velocidad se puede reproducir fielmente la señal analógica en cuestión por lo que se tienen características distintas entre los sistemas de adquisición anteriores al presentado en este trabajo, por tal motivo es importante rescatar que los elementos de transferencia y adquisición de datos son más rápidos cuando se diseñan en hardware, aquí la importancia de mencionarlo, este sistema se implementa en hardware por lo tanto se procura que sea un sistema en tiempo real.

Actualmente en México se cuenta con equipos electrocardiográficos que monitorean el comportamiento del corazón, la capacidad de estos equipos en roedores son bastante caros, debido a las marcas y protección de información. Se propone un sistema que pueda seguir los parámetros y estándares adecuados, los cuales puedan generar una base de adquisición de señales biométricas, que pueda mostrar el acondicionamiento de la señal Electrocardiograma, donde se oriente a futuros proyectos que trabajen con señales biométricas. Ofreciendo como conocimiento la experiencia generada por este apartado así como bases sólidas sobre bioseñales.

2.2 Electrocardiograma y métodos de medición en roedores.

El Electrocardiograma (ECG) proporciona una representación visual de los cambios espacio-temporales en la distribución del campo de voltaje extracelular presente en la superficie del cuerpo y es directamente relacionada con el campo de voltaje local generada por su fuente, el corazón. La utilidad potencial de la ECG en toxicología se deriva de su capacidad para reflejar instantánea cambios en los procesos bioquímicos y metabólicos que sea directa o indirectamente afectan a las propiedades biofísicas de la membrana celular cardíaca.

El ECG es un registro instantáneo de toda la actividad eléctrica que ocurre en un momento dado en el corazón y no de cualquier célula individual. Factores que influyen en la forma del ECG, tal como se representa en Fig. 2-1, son la colocación del electrodo sobre la ubicación anatómica del corazón.

La representación de la actividad eléctrica detectada por el ECG, es descrita en la Fig. 2-1, la ilustración muestra los corazones de humanos y ratas. La forma de los corazones difieren ligeramente así como la gráfica de cada órgano representado.

Si se analiza a detalle se puede describir las diferencias entre las respuestas de cada ejemplar, teniendo en cuenta la duración de la señal, la ausencia de una onda Q y el segmento ST en la rata/ratón de ECG en relación con el ECG humano.

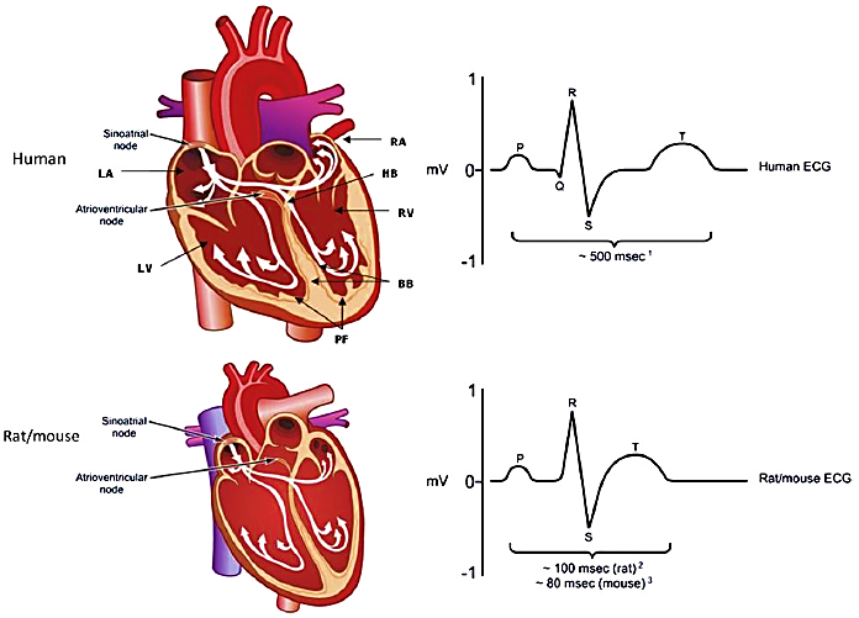


Figura 2-1.- Comportamiento ECG humano y rata. (Aimen K. Farraj, 2011)

Una caracteristica importante del comportamiento del ECG es la frecuencia de polarizacion del corazón, en los roedores es alrededor de 10 Hz en condiciones normales, la cual puede variar al encontrarse en situaciones que aceleren su metabolismo.

Con el fin de comprender la monitorizacion del ECG en superficies de ratas, se compara el comportamiento con el monitoreo de humano. El ECG registra la suma de las fuerzas electricas generadas por todas las celulas activas dentro del corazon (se despolarizan y se polarizan) sobre la superficie del cuerpo. La forma del ECG (direccion y magnitud) se determina por como estas fuerzas electricas estan alineadas respecto a los ejes de referencia establecidos por las posiciones de los

electrodos sobre el cuerpo. La amplitud o fuerza de la deflexión es determinado por la cantidad de tejido o de despolarizacion en un momento dado.

La dirección en la que una onda de despolarización viaja en relacion con los electrodos de registro también determina el tamaño de la señal grabada. En el Electrocardiograma humano, se emplean múltiples derivaciones para obtener una mayor precisión.

En la Fig. 2-2, muestra las derivaciones, es decir, la colocación de los electrodos donde puede obtenerse una mejor resolución del ECG. En A se ilustra las derivaciones de los seres humanos y en B la de los roedores, donde se coloca un electrodo negativo cerca del hombro derecho y el electrodo positivo a la izquierda del espacio xifoides.

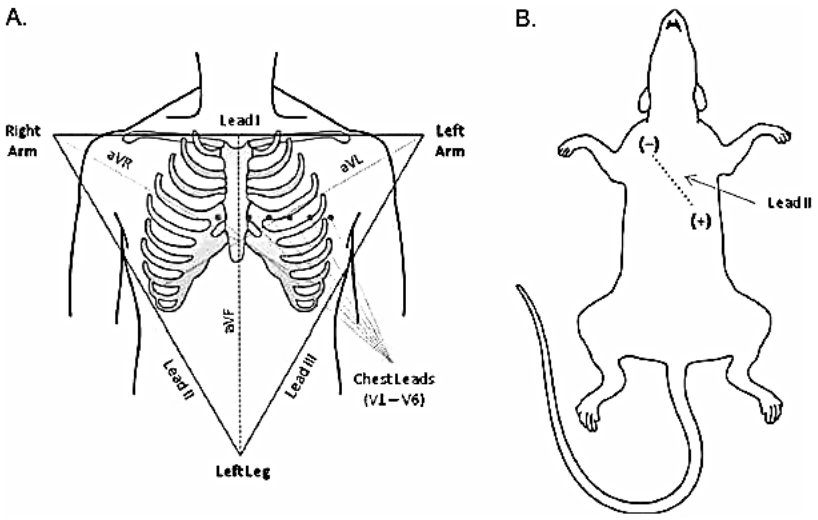


Figura 2-2.- Derivaciones A. Humano B. Roedor. (Aimen K. Farraj, 2011).

Estudios sobre el Electrocardiograma en ratas indican que puede obtenerse el ECG, si se colocan los electrodos sobre las extremidades del ratón. Se toma en cuenta que para la reproducibilidad de las señales es mejor cuando se tienen más

electrodos en la posición correcta, la colocación de sensores en patas puede ser usado para una mejor interpretación de las señales ante la poca movilidad de las ratas.

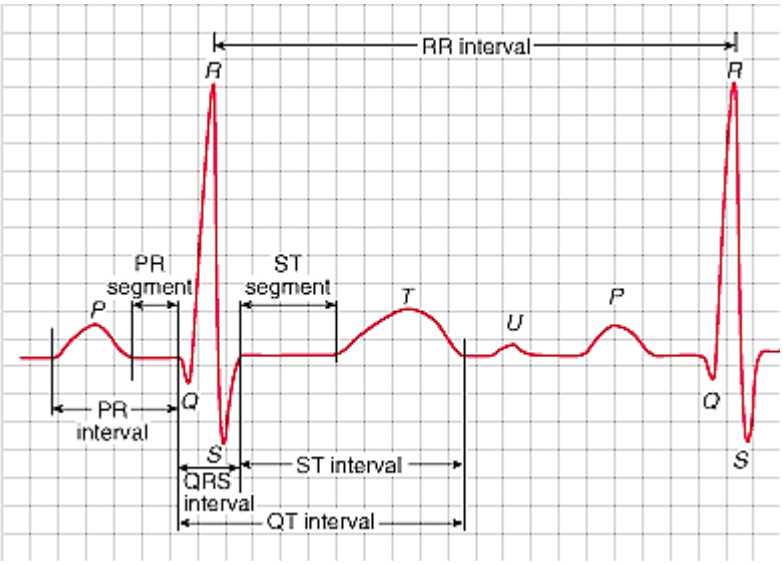


Figura 2-3.- Parámetros del Electrocardiograma (Electrocardiografía.es, 2010)

En la figura 2-3 se pueden observar los parámetros de la señal ECG, donde se encuentra la duración en tiempo de los segmentos para ratas como los siguientes:

Segmento RR de aproximadamente 155 ms, segmento PT de 80 ms a 100 ms, segmento PR alrededor de 50 ms, segmento QT de 50 ms a 70 ms, segmento QRS de 15 ms a 20 ms. Estos datos son examinados de artículos donde se genera el Electrocardiograma para ratas/ratones.

Continuando con las características del ECG en roedores se debe mencionar los rangos de voltaje que se aprecian en las señales, por lo cual se presentan gráficas de experimentos sobre ratas, donde se puede entender claramente el voltaje que se espera conseguir con sensores o electrodos ECG.

A continuación se ilustra en la Figura 2-4, los diferentes niveles de voltaje que emplean distintos artículos.

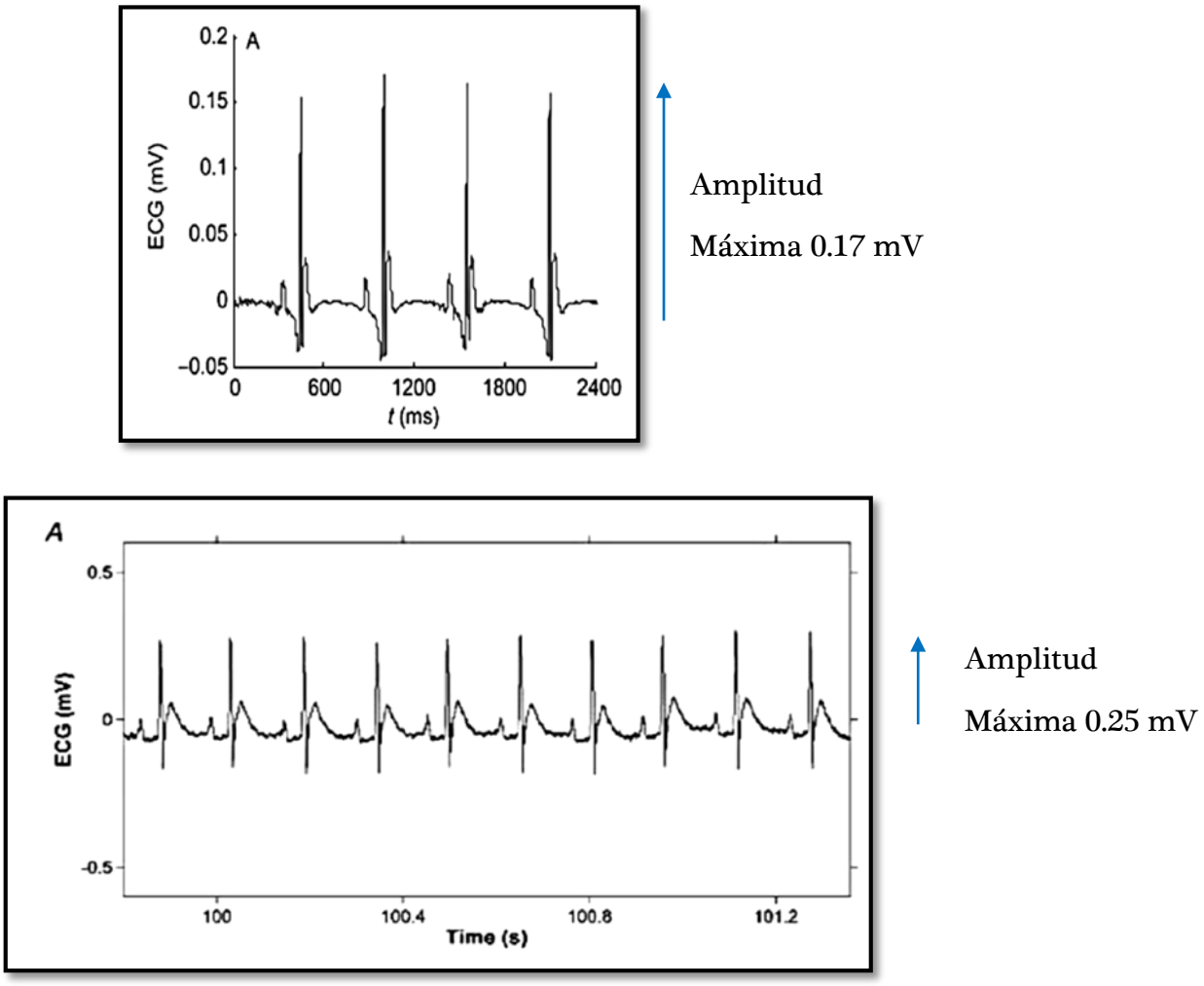


Figura 2-4.- Niveles de voltaje ECG en roedores. (Farmer & Levy, 1967)

Aquí se muestra la configuración de voltaje obtenidas en artículos, en la Figura 2-4 la gráfica con amplitud máxima de 0.17 mV es obtenida de "Complexity and characteristic frequency studies in ECG signals of mice based on multiple scale factors", la ilustración con amplitud máxima de 0.25 mV de la fuente "Non invasive method for electrocardiogram recording in conscious rats: Feasibility for heart rate variability analysis".

2.3 Sensores para adquisición de señales

Un sensor es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida transducible que es función de la variable medida.

Los sensores también son conocidos como elementos primarios de medición, y bajo este término el concepto es que son los elementos que están en contacto con la variable y utilizan o absorben energía del medio controlado para dar al sistema de medición una indicación en respuesta a la variación de la variable controlada.

El estudio de los sensores se puede realizar con más facilidad si se clasifican de acuerdo a algún criterio, ya que los sensores existentes para las distintas magnitudes físicas son demasiados.

2.3.1 Tipos de sensores

Para el estudio de un gran número de sensores se suele acudir a su clasificación de acuerdo con la magnitud medida: temperatura, presión, caudal, humedad, posición, velocidad, aceleración, fuerza, par, etc.

Desde el punto de vista de la ingeniería electrónica, es más atractiva la clasificación de los sensores de acuerdo con el parámetro variable:

- Resistivos
- Capacitivos
- Inductivos
- Magnéticos
- Ópticos

2.3.2 Electrodos para ECG

Los electrodos son instrumentos con base de metal que perciben las ondas eléctricas generadas durante el ciclo cardíaco y que conectados a un galvanómetro (el electrocardiógrafo), permiten observar en forma gráfica la actividad eléctrica cardíaca desde distintos ángulos. Cada electrodo monitorea la actividad eléctrica que se acerca o se aleja del sitio donde se encuentra colocado.

Se puede encontrar con una gama extensa de electrodos, ya que actualmente se cuenta con diferentes materiales y marcas por lo cual se enunciaran solo algunos de los que se pueden emplear para el registro del Electrocardiograma (Figura 2-5).

Figura 2-5.- Tipos de Electrodos. (Leonhard Lang GmbH, 2011).

2.4 Amplificadores Operacionales

Se puede definir un amplificador operacional, como un componente con una gran ganancia, cuyo circuito básico o de partida es un par diferencial. En cuanto a su modo de operación, está determinado por el lazo de realimentación (positiva, negativa), el tipo de elementos contenidos en el mismo, así como su disposición en dicho lazo. Consiguiéndose de esta forma, que el mismo OpAmp sea capaz de realizar distintas operaciones. (Fernández, 2008). (Feijo& Reyes, 2010).

2.4.1 Amplificador inversor

Se denomina amplificador inversor ya que la señal de salida es igual a la señal de entrada pero con la fase invertida 180 grados. A continuación se desarrolla el análisis del circuito. Figura 2-7 amplificador inversor.

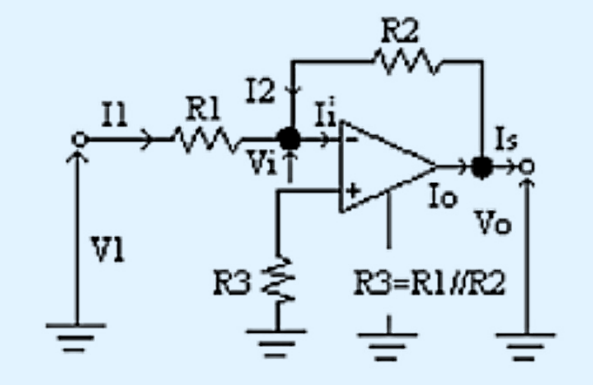


Figura 2-6.- Amplificador Inversor (Feijo & Reyes, 2010)

Se obtiene el voltaje de salida Vo:

$$Vo = -V1 * \frac{R2}{R1} \qquad (2\text{-}1)$$

2.4.2 Amplificador no inversor

Este circuito se presenta como una aplicación interesante por su capacidad para mantener la fase de una señal, el análisis se realiza de forma análoga al anterior. La figura 2-8 muestra un esquema del amplificador no inversor.

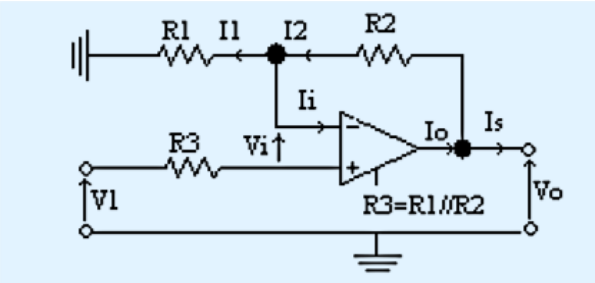


Figura 2-7.- Amplificador No inversor. (Feijo & Reyes, 2010)

Se obtiene que el voltaje de salida es igual a:

$$Vo = V1 * (1 + \frac{R2}{R1}) \qquad (2\text{-}2)$$

2.4.3 Sumador Inversor

En el circuito de la figura 2-9, se hace resaltar que en el amplificador inversor, la tensión V (+) está conectada a masa por lo que la tensión V (-) estará a una masa virtual.

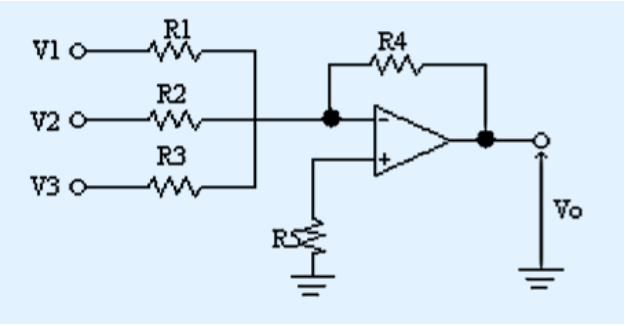


Figura 2-8.- Sumador Inversor. (Feijo & Reyes, 2010)

El voltaje de salida es:

$$Vo = -(V1 * \frac{R4}{R1} + V2 * \frac{R4}{R2} + V3 * \frac{R4}{R3}) \qquad (2\text{-}3)$$

Una característica importante de esta configuración es el hecho de que la mezcla de señales lineales, en el nodo suma, no produce interacción entre las entradas, puesto que todas las fuentes de señal alimentan el punto de tierra virtual. El circuito puede acomodar cualquier número de entradas añadiendo resistencias de entrada adicionales en el nodo suma.

Aunque los circuitos precedentes se han descrito en términos de entrada y resistencias de realimentación, las resistencias se pueden reemplazar por elementos complejos, y los axiomas de los amplificadores operacionales se mantendrán como verdaderos. Dos circuitos que demuestran esto, son dos configuraciones del amplificador inversor.

2.4.4 Sumador No Inversor

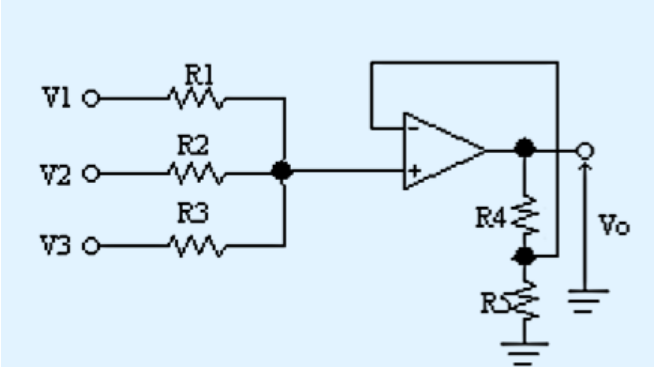


Figura 2-9.- Sumador No Inversor. (Feijo & Reyes, 2010)

En la figura 2-10 se observa el esquema de un sumador no inversor. El análisis del sumador no inversor se realiza de manera semejante que el sumador inversor de ahí tenemos que el voltaje de salida es:

$$Vo = (V1 * \frac{R4}{R1} + V2 * \frac{R4}{R2} + V3 * \frac{R4}{R3}) \qquad (2\text{-}4)$$

2.4.5 Amplificador Integrador

Un circuito integrador realiza un proceso de suma llamado "integración". La tensión de salida del circuito integrador es proporcional al área bajo la curva de entrada (onda de entrada), para cualquier instante. La figura 2-11 muestra el circuito de un amplificador integrador.

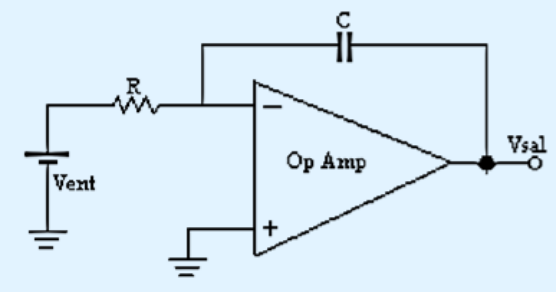


Figura 2-10.- Amplificador Integrador. (Feijo & Reyes, 2010)

2.4.6 Amplificador derivador

Se trata de un circuito constituido por un capacitor C y una resistencia R, el cual actúa como un filtro pasivo para altas frecuencias, debido a que no intervienen elementos amplificadores, como transistores o circuitos integrados, este tipo de filtro atenúa bajas frecuencias según la formula empírica.

Este circuito se utiliza para detectar flancos de subida y bajada de una señal provocando una mayor diferencia en los flancos de entrada y salida de la señal, de donde la variación con el tiempo se hace más notoria. Estas zonas de la señal son además las que corresponden a las altas frecuencias, mientras que las zonas planas están compuestas por frecuencias más bajas.

Este tipo de circuitos son más conocidos como filtro RC pasivo pasa alto, se utiliza para las frecuencias superiores al valor especificado. Este circuito, separa la corriente continua entre circuitos ya que el condensador interrumpe el paso de la

corriente continua, dejando pasar solo el pulso correspondiente al flanco de entrada y salida. La señal derivada puede utilizarse para disparar algún componente de la cadena electrónica como puede ser un trigger. En la figura 2-13 se muestra el esquema de un amplificador derivador.

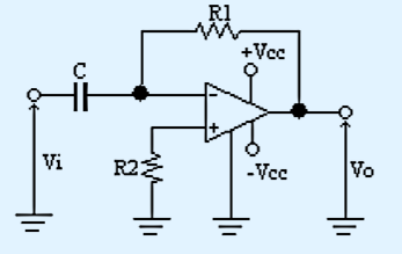


Figura 2-11.- Amplificador derivador. (Feijo & Reyes, 2010).

2.4.7 Amplificadores Instrumentales

Un amplificador instrumental es un dispositivo creado a partir de amplificadores operacionales. Se puede construir a base de componentes discretos o se puede encontrar encapsulado como se puede observar en la figura 2-14. La operación que realiza es la resta de sus dos entradas multiplicadas por un factor o ganancia.

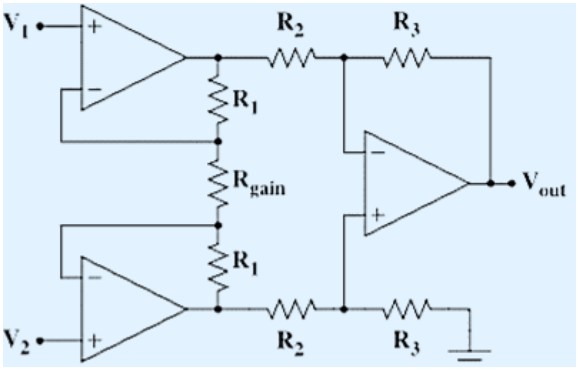


Figura 2-12.- Amplificador Instrumental. (Feijo & Reyes, 2010).

Existe en equipos de industria, en equipos de electromedicina, y en muchas aplicaciones, la necesidad de medir señales muy pequeñas del orden de microvoltios o milivoltios en la presencia de grande señales de ruido provenientes de distintas fuentes, como pueden ser motores, tubos de iluminación de descarga gaseosa, etc. Para realizar las mencionadas mediciones estos deberán utilizar en su entrada Amplificadores de Instrumentación con una alta impedancia de entrada y un alto Rechazo de Modo Común (CMRR).

El circuito de la figura 2-14 se divide en 2 partes para aclarar su funcionamiento.

Primer Etapa

En la primera etapa se obtiene la ganancia de los dos primeros operacionales con sus respectivas señales de ingreso:

Entonces se tiene que la ganancia diferencial es

$$Gd1 = (\frac{Vo1 - Vo2}{S1 - S2}) \quad (2\text{-}5)$$

Donde Vo1 y Vo2 son salidas de voltaje de los dos primeros amplificadores operacionales. S1 y S2 son los voltajes de entrada presentes en la configuración.

Podemos describir las siguientes formulas.

$$Vo1 - Vo2 = (2R1 + Rg) * I \quad (2\text{-}6)$$

$$S1 - S2 = RgI \quad (2\text{-}7)$$

Las cuales reemplazamos en Gd1 y reescribimos la ecuación.

$$Gd1 = 1 + (\frac{2R1}{Rg}) \quad (2\text{-}8)$$

Segunda Etapa

En esta etapa debido al valor de las resistencias se obtiene una ganancia unitaria. Esta parte del circuito es especialmente diseñada para garantizar simetría en la amplificación del circuito. Entonces:

$$Gd2 = 1 \quad (2\text{-}9)$$

Por lo tanto la ganancia total del circuito es modo diferencial es la ganancia de la primer etapa.

$$Gd = 1 + (\frac{2R1}{Rg}) \quad (2\text{-}10)$$

Es necesario mencionar que las entradas de voltaje pueden ser voltajes flotantes, lo cual hace la relevancia de esta configuración para futuras aplicaciones dentro de este proyecto.

2.5 Filtros de Señales

Un filtro deja pasar una banda de frecuencia mientras rechaza otras. Los filtros pueden ser pasivos o activos. Los filtros pasivos se constituyen con resistencias, condensadores e inductores. Se usan generalmente por encima de 1MHz, no tienen ganancia en potencia y son relativamente difíciles de sintonizar. Los filtros activos se construyen con resistencias, condensadores y amplificadores operacionales. Se usan por debajo de 1MHz, tienen ganancia en potencia y son relativamente fáciles de sintonizar.

Los filtros pueden separar las señales deseadas de las no deseadas. Se puede dividir los filtros en analógicos y digitales, para hacer su compresión clara revisaremos características de ambos diseños. (Gaspar, 2009).

2.5.1 Filtros Analógicos Activos

Los filtros son circuitos capaces de controlar las frecuencias permitiendo el paso de estas dependiendo de su valor. Se llaman activos ya que constan de elementos pasivos (R-C) y elementos activos como el OP-AMP. Los elementos R-C están compuestos por una resistencia y un condensador, dependiendo del número de elementos usados se determinará el orden del filtro así como su respuesta y su calidad.

Para cualquier tipo de filtro se emplean las siguientes definiciones:

Frecuencia de corte: es aquella en que la ganancia del circuito cae a -3 db por debajo de la máxima ganancia alcanzada. En los filtros pasa y elimina banda, existen dos: una superior y otra inferior.

Banda pasante: conjunto de frecuencias de ganancia superior a la de corte en un margen menor o igual a 3 db.

Calidad: especifica la eficacia del filtro, es decir, la identidad de su respuesta. Se mide en dB.

Los filtros activos se diferencian de los filtros comunes, en que estos últimos son solamente una combinación de resistencias, capacitores e inductores. En un filtro común, la salida es de menor magnitud que de la entrada. En cambios los filtros activos se componen de resistores, capacitores y dispositivos activos como Amplificadores Operacionales o transistores. En un filtro activo la salida puede ser igual o de mayor magnitud que la de entrada.

2.5.1.1 Filtro Pasa bajo

Es aquel que permite el paso de frecuencias bajas, desde frecuencia cero o continua hasta una determinada frecuencia. Presentan ceros a alta frecuencia y polos a bajas frecuencias. El esquema de la figura 2-15 muestra un filtro pasa bajo activo.

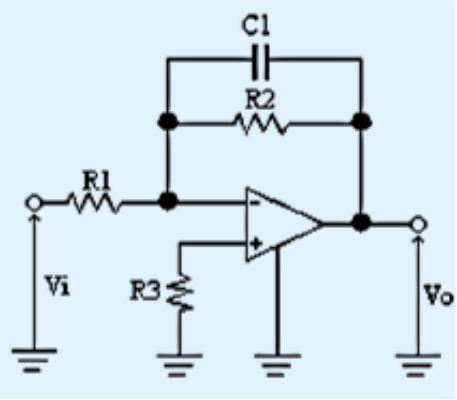


Figura 2-13.- Filtro pasa bajo activo. (Feijo & Reyes, 2010).

La frecuencia de corte del filtro pasa bajo viene dada por la siguiente ecuación:

$$fc = \frac{1}{2 * \pi * R2 * C1} \qquad (2\text{-}11)$$

2.5.1.2 Filtro Pasa alto

Es el que permite el paso de frecuencia desde una frecuencia de corte determinada hacia arriba, sin que exista un límite superior especificado. Presentan ceros a bajas frecuencias y polos a altas frecuencias. La figura 2-16 muestra un filtro pasa alto activo.

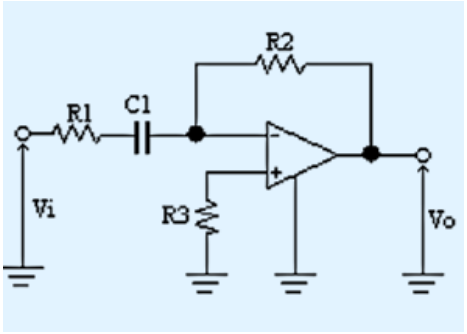


Figura 2-14.- Filtro Pasa alto. (Feijo & Reyes, 2010).

La frecuencia de corte del filtro pasa alto viene dada por la siguiente ecuación:

$$fc = \frac{1}{2 * \pi * R1 * C1} \qquad (2\text{-}12)$$

2.5.1.3 Filtro Pasa banda

Son aquellos que permiten el paso de componentes frecuenciales contenidos en un determinado rango de frecuencias, comprendido entre una frecuencia de corte superior y otra inferior. En la figura 2-17 se observa el esquema de un filtro pasa banda activo.

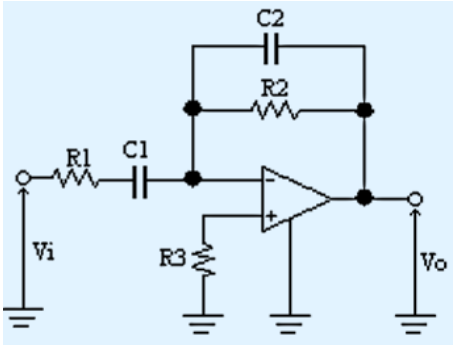


Figura 2-15.- Filtro Pasa banda. (Feijo & Reyes, 2010).

El filtro pasa banda tiene dos frecuencias de corte una baja y otra alta que se observan en las siguientes ecuaciones:

El filtro pasa banda tiene dos frecuencias de corte una baja y otra alta que se observan en las siguientes ecuaciones:

Frecuencia de corte alta

$$fc = \frac{1}{2 * \pi * R1 * C1} \quad (2\text{-}13)$$

Frecuencia de corte baja

$$fc = \frac{1}{2 * \pi * R2 * C2} \quad (2\text{-}14)$$

Un filtro ideal sería el que contienen unas bandas pasante y corte totalmente planas y unas zonas de transición entre ambas nulas, pero en la práctica esto nunca se consigue, siendo normalmente más parecido al ideal cuando mayor sea el orden del filtro, para medir cuan bueno es un filtro se puede emplear el denominado factor Q. En los filtros de órdenes altos suele aparecer un rizado en las zonas de transición conocido como efecto Gibbs.

2.5.1.4 Filtro Elimina banda

Este filtro dificulta el paso de componente frecuenciales contenidos en un determinado rango de frecuencias, comprendido entre una frecuencia de corte superior y otra inferior. En la figura 2-18 se muestra un filtro elimina banda activo. Las frecuencias de corte son las siguientes.

$$fc = \frac{1}{2 * \pi * R2 * C2} \quad (2\text{-}15)$$

$$fc = \frac{1}{2 * \pi * R1 * C1} \quad (2\text{-}16)$$

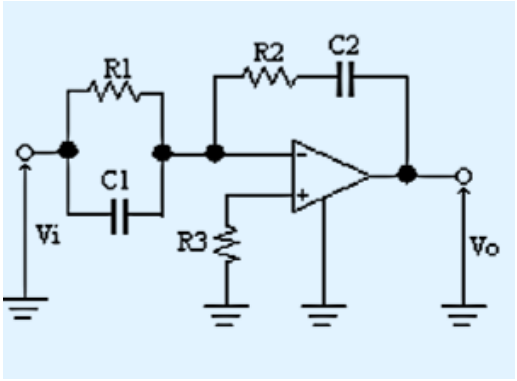


Figura 2-16.- Filtro Elimina banda. (Feijo & Reyes, 2010).

2.5.2 Filtros Digitales

Un filtro digital, es un filtro que opera sobre señales digitales. Es una operación matemática que toma una secuencia de números (señal de entrada) y la modifica produciendo otra secuencia de números (señal de salida) con el objetivo de resaltar o atenuar ciertas características.

Puede existir como una formula en un papel, como un loop en un programa de computadora o como un circuito integrado en un chip.

Las aplicaciones de los filtros digitales son: separación de señales que fueron combinadas desafortunadamente (ruido, interferencias proveniente de otros sistemas), recuperación de señales distorsionadas de alguna forma, síntesis de sonido: creación o modificación de señales para moldear espectros o formas de onda y lograr el efecto auditivo buscado.

El desempeño de los filtros digitales es ampliamente superior a los filtros analógicos. En muchas ocasiones, la motivación para muestrear una señal es emplear un filtro digital. (Curso Tratamiento Digital de Señal, 1999)

2.5.2.1 Filtros IIR (Respuesta al impulso infinita)

Mediante la ecuación en recurrencia. En este caso, el filtro se define por los coeficientes de recursión. La salida en cada instante involucra además de muestras de la entrada, muestras previas de la salida.

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + b_0 x[n] + b_1 x[n-1] + b_2 y[n-2] \qquad (2\text{-}17)$$

El filtro es recursivo si tiene algún coeficiente de realimentación no nulo. En ese caso, es un filtro IIR. En caso contrario, no hay realimentación y el filtro es FIR, o equivalentemente, no recursivo.

2.5.2.2 Filtros FIR (Respuesta al impulso finita)

Convolución de la señal de entrada con la respuesta al impulso del filtro. En este caso, la salida del filtro en cada instante es un promedio ponderado de la muestra actual y muestras pasadas de la entrada. La función de Transferencia tiene un denominador constante y sólo tiene ceros.

$$y[n] = (x * h)[n] = \sum_k x[k]h[n-k] \qquad (2\text{-}18)$$

No hay recursión, es decir, la salida depende sólo de la entrada y no de valores pasados de la salida.

La respuesta es de duración finita ya que si la entrada se mantiene en cero durante M periodos consecutivos, la salida será también cero.

2.6 Conversores Analógicos-Digitales

El objetivo básico de un ADC es transformar una señal eléctrica análoga en un número digital equivalente. Esta función exige que los pasos intermedios se realicen de forma óptima para no perder información. Según el tipo de componente y su aplicación existen distintos parámetros que lo caracterizan, éstos pueden ser: la velocidad de conversión, la resolución, los rangos de entrada, etc.

Por ejemplo, una mayor cantidad de bit, implica mayor precisión, pero también mayor complejidad. Un incremento en un solo bit permite disponer del doble de precisión (mayor resolución), pero hace más difícil el diseño del circuito, además, la conversión podría volverse más lenta.

El diagrama de bloques de la Fig.2-18 muestra la secuencia desde que la variable física entra al sistema hasta que es transformada a señal digital (código binario). Para dicha señal ingrese al convertidor análogo - digital, ésta debe ser muestreada, es decir, se toman valores discretos en instantes de tiempo de la señal análoga, lo que recibe el nombre de sampling. Matemáticamente es el equivalente a multiplicar la señal análoga por una secuencia de impulsos de periodo constante. Como resultado se obtiene un tren de impulsos con amplitudes limitadas por la envolvente de la señal analógica.

Para garantizar la toma de muestra y la conversión de forma correcta se debe considerar la velocidad de muestreo, para lo cual el Teorema de Nyquist, establece que la frecuencia de muestreo fs, debe ser como mínimo el doble que el ancho de banda de la señal muestreada. Si no ocurre esta situación, se tiene lugar el fenómeno denominado aliasing. (Huircán, 2008).

$$fs > 2 * fm \qquad (2\text{-}19)$$

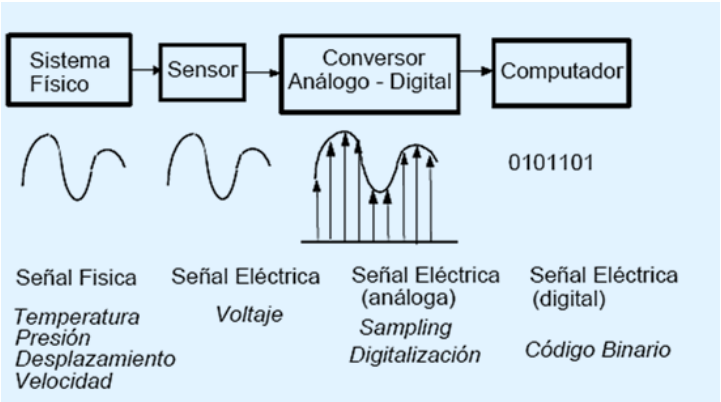


Figura 2-17.- Conversión análogo-digital. (Huircán, 2008).

Los dispositivos ADC convierten un nivel de tensión analógico en una palabra digital correspondiente. Si n es el número de bit obtenidos de la palabra, esto significa que habrá 2^n niveles de tensión diferentes. Todo convertidor ADC debe procurar que el conjunto de bit obtenidos a la salida sea un reflejo lo más exacto posible del valor analógico correspondiente. Se usan un gran número de métodos para convertir señales analógicas a la forma digital, los que más usados son: Rampa de escalera, aproximaciones sucesivas, paralelo (flash), doble rampa, voltaje a frecuencia, tipo serie.

2.6.1 Tipos de convertidores ADC

A. Convertidor Análogo-Digital de Rampa de Escalera.

Se basa en la comparación de la señal analógica de entrada con una señal de rampa definida con precisión. El esquema se muestra en la Fig. 2-20.

Se comienza activando un pulso de inicio en la lógica de control, con esta acción el contador se inicializará en cero, entregando en sus salidas el código binario del cero digital. La secuencia pasa directamente como entrada paralelo al DAC que responde con 0 [V] a la salida. Esta señal es usada como entrada de referencia a un

comparador, el cual compara la magnitud de la señal analógica de entrada con el valor entregado por el conversor. Del valor que proporcione el comparador dependerá que el contador continúe contando o bien, se detenga, pues si el comparador entrega un "1", entonces el reloj continuará alimentando al comparador. De lo contrario si entrega un"0", el contador se detendrá. La lógica del comparador es si la señal de entrada es mayor que la referencia, entonces el comparador responderá con un "1"y se incrementa la cuenta en 1 digital, y así sucesivamente, sólo la cuenta se detendrá cuando la respuesta del DAC sea mayor que la entrada de la señal analógica. En este caso, el reloj se detendrá y se tendrá la salida digital del valor de cuenta anterior.

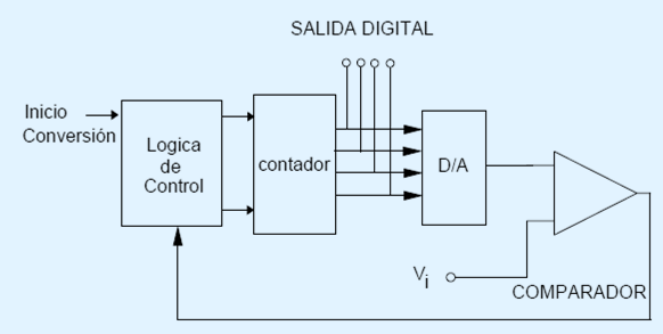


Figura 2-18.- Conversor de rampa escalera. (Huircán, 2008).

B. Convertidor Análogo-Digital por Aproximaciones Sucesivas

Es una técnica de conversión más efectiva que la anterior. Se utiliza ampliamente debido a su combinación de alta resolución y velocidad. El esquema es prácticamente el mismo, difieren en que el contador dentro del registro no es un contador secuencial de uno en uno, sino un contador programable que se incrementa o decrementa de acuerdo a la influencia del bit de mayor peso (SAR). De esta manera no es necesario contar con 2^n veces como lo hacía el contador tipo rampa, ahora la cuenta máxima solo es de n veces. El esquema de la Fig. 2-20, muestra este convertidor.

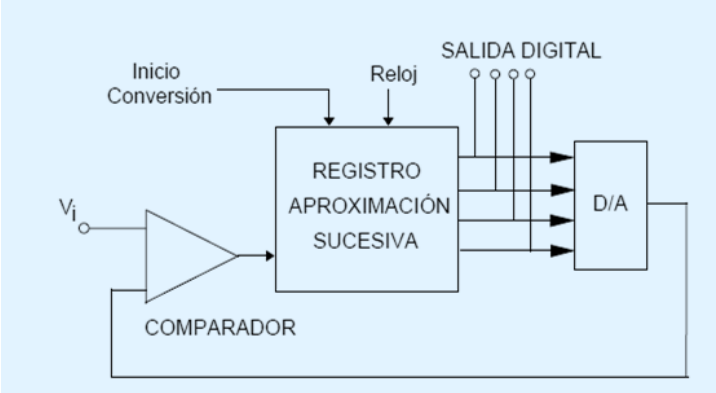


Figura 2-19.- Conversor de Aproximación sucesiva. (Huircán, 2008).

El SAR pone el bit MSB en "1" y todos los restantes en "0". La cantidad es tomada por el DAC de tal manera que su equivalente analógico se compara con la señal de entrada. Si la salida del DAC es mayor que la entrada, se elimina el "1" del bit MSB y se pone a "1" el bit inmediatamente anterior, con todos los demás bit en "0", y así sucesivamente hasta que se logre encontrar una secuencia análoga pero que resulta ser menor que la entrada de la señal, cuando ocurra esto, el bit mantendrá su valor y se pone a "1" el bit inmediatamente anterior. El procedimiento anterior se repite hasta terminarse de probar "1" en cada bit del contador. Lo anterior equivale a un tanteo digital, a medida que se avanza, el procedimiento se va estabilizando hasta llegar un valor estable y que corresponderá con el valor de la medición. La figura 2-21, se muestra la salida característica de este tipo de conversor.

El ADC de aproximaciones sucesivas es de los más utilizados, es posible encontrar modelos capaces de suministrar 16 bits en la salida y realizar la conversión en un tiempo de unas decenas de microsegundos. Los modelos de 12 y 8 bits, son los más comunes y ofrecen una elevada velocidad a un precio ajustado.

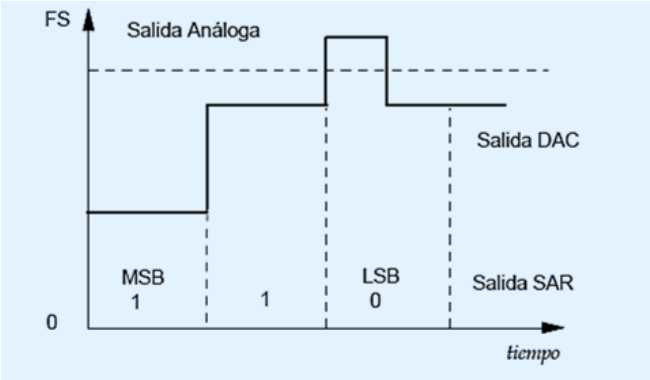


Figura 2-20.- Curva de salida del DAC. (Huircán, 2008).

C. Convertidor Análogo-Digital Paralelo (FLASH)

Los conversores de tipo flash o conversión directa, parten de una concepción radicalmente opuesta: la velocidad es el objetivo básico de esta arquitectura y el costo que se debe pagar por ello es un circuito muy complejo aunque sencillo a nivel de concepto. Dos señales participan en la etapa de entrada, la propia señal analógica que se debe convertir y una señal de referencia. En la configuración básica, la señal analógica se aplica a las puertas no inversoras de un cierto número de amplificadores operacionales que, utilizados como comparadores, están dispuestos en paralelo, a la entrada de un decodificador. A la entrada inversora de cada comparador se aplica la tensión de referencia, que a su vez ataca una red de resistencia de valores idénticos y dispuestos en serie. El resultado es la diferencia de tensión entre dos comparadores sucesivos es de 1 LSB. La complejidad de la arquitectura flash se deriva precisamente del elevado número de comparadores necesarios a medida que aumenta el número de bits que se desea obtener a la salida. El número de éstos es 2^{n-1}, donde n es el número de bits de salida, no es de extrañar que los conversores de tipo flash ven limitada su resolución por su elevada integración. El resultado es que no existe ningún convertidor flash que ofrezca una resolución de 16 bit, y que más allá no son prácticos teniendo en cuenta el tamaño del chip, el correcto funcionamiento de los

comparadores e incluso el precio. Este tipo de conversor por razón de velocidad es ampliamente usado en el campo de las telecomunicaciones, los instrumentos de medida y, en general, el tratamiento de señales rápidas como la de vídeo.

D. Convertidor de Doble Rampa

Los de tipo rampa tienen como punto fuerte la precisión (ver Fig. 2-22), y al mismo tiempo, sólo pueden aplicarse a señales cuyo nivel oscile de forma muy lenta (un valor típico de velocidad de muestreo es de 10 muestras por segundo). Este dispositivo consiste en un integrador basado en un amplificador operacional.

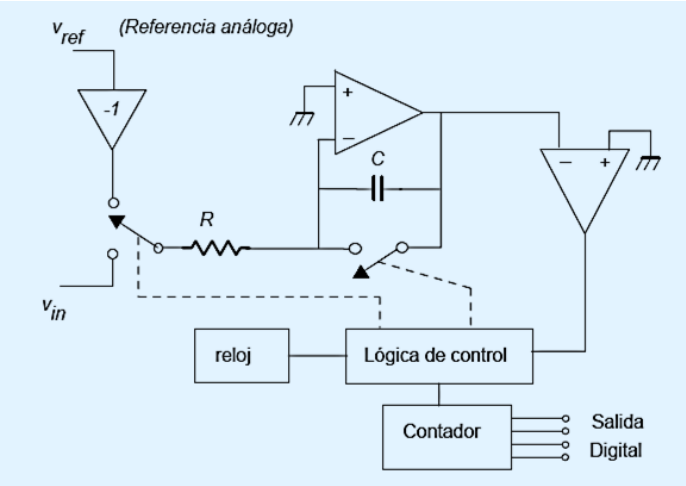


Figura 2-21 Conversor Doble rampa. (Huircán, 2008).

Para dos entradas, la señal analógica que se va a digitalizar y una señal de referencia de valor constante. Un interruptor se encarga de que una de las dos esté conectada en todo momento al amplificador integrador. Otro interruptor se halla en paralelo con el condensador, el que permite la intervención que éste o no. El resultado de la actuación coordinada de ambos interruptores es que en la salida se obtenga una señal de doble rampa. Una de subida (la carga del condensador con la tensión analógica en la entrada) y la de bajada (con la tensión de referencia a la

entrada). El cálculo de la señal digitalizada se fundamenta en la relación entre los tiempos de subida y bajada, de acuerdo con la ecuación:

$$\frac{ts}{tm} = \frac{Vref}{Va} \qquad (2\text{-}20)$$

Donde ts, es el tiempo de subida o de muestreo y tm el de bajada o de medida, Vref es la tensión de referencia y Va es la tensión analógica.

Los tiempos de muestreo y de medida son detectados por un contador que se encuentra a la salida del integrador y dependen de la resistencia, el condensador y la tensión de entrada. Dada sus especiales características, los ADC de doble rampa se utilizan, por ejemplo, en los voltímetros digitales, por su exactitud e inmunidad al ruido. Pueden alcanzar una resolución de hasta 18 o 20 bits.

E. Convertidor Sigma-Delta

Convertidores sigma-dela han recibido recientemente una mayor atención debido a su buena actuación y facilidad de implementación VLSI, como se describe.

Un diagrama de bloques simplificado de un convertidor A / D se muestra en la figura. 2-23, en donde la señal de entrada es una forma de onda de banda limitada con espectro de potencia en el ancho de banda. Este esquema presenta las principales operaciones realizadas por un convertidor, sin considerar su implementación hardware. El primer bloque (sobremuestreo y cuantificador) puede ser visto como un sistema no lineal, que digitaliza la señal de entrada, mediante la realización de un sobremuestreo usando el teorema de Nyquist. Su salida se puede considerar como una señal discreta en el tiempo, que representa la versión cuantificada de la secuencia de muestras. La señal discreta se diezmó a la tasa de Nyquist, después de un filtrado de paso bajo de forma pronunciada selectiva. El sobremuestreo proporción y el número de niveles de cuantificación están estrictamente

relacionadas. En la mayoría de los casos se reduce a una señal binaria, con tasa de bits igual a fs.

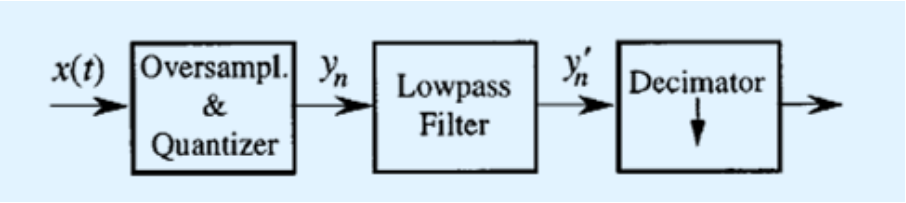


Figura 2-22.- Diagrama a bloques de un Conversor Sigma-Delta. (Letizia Lo Presti, 2000)

2.7 Interfaz y Comunicación Periférica

2.7.1 Interfaz USB

El USB es un bus punto a punto: dado que el lugar de partida es el host (PC o hub), el destino es un periférico u otro hub. No hay más que un único host (PC) en una arquitectura USB.

Los PC estándar tienen dos tomas USB, lo que implica que, para permitir más de dos periférico simultáneamente, es necesario un hub. Algunos periféricos incluyen un hub integrado, por ejemplo, el teclado USB, al que se le puede conectar un Mouse USB.

Los periféricos comparten la banda de paso del USB. El protocolo se basa en el llamado paso de testigo (token). El ordenador proporciona el testigo al periférico seleccionado y seguidamente, éste le devuelve el testigo en su respuesta.

Este bus permite la conexión y la des-conexión en cualquier momento sin necesidad de apagar el equipo.

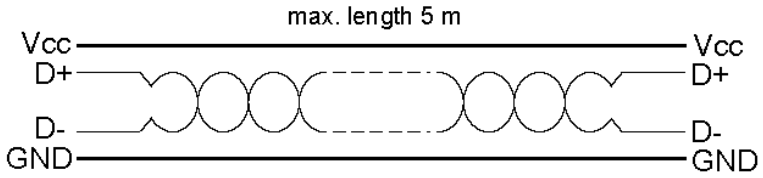


Figura 2-23.- Interfaz física USB. (Pérez, 2000)

Aspecto eléctrico: A nivel eléctrico, el cable USB transfiere la señal y la alimentación sobre 4 hilos, como se muestra en la figura 2-24.

A nivel de alimentación, el cable proporciona la tensión nominal de 5 V. Es necesario definir correctamente el diámetro del hilo con el fin de que no se produzca una caída de tensión demasiado importante en el cable. Una resistencia de terminación instalada en la línea de datos permite detectar el puerto y conocer su configuración (1,5 o 12 Mbits/s).

A nivel de señal, se trata de un par trenzado con una impedancia característica de 90 Ω La velocidad puede ser tanto de 12 Mbits/s como de 1,5 Mbits/s. La sensibilidad del receptor puede ser de, al menos, 200mV y debe poder admitir un buen factor de rechazo de tensión en modo común. El reloj se transmite en el flow de datos, la codificación es de tipo NRZI, existiendo un dispositivo que genera un bit de relleno (bit stuffing) que garantiza que la frecuencia de reloj permanezca constante. Cada paquete va precedido por un campo de sincronismo.

2.7.2 Protocolo SPI (Interfaz Serial Periférica)

SPI es un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas porta la información entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es full duplex. Dos de estas líneas trasfieren los datos (una en

cada dirección) y la tercer línea es la del reloj. Algunos dispositivos solo pueden ser transmisores y otros solo receptores, generalmente un dispositivo que tramite datos también puede recibir. (Pérez, Protocolo SPI, 2000)

Un ejemplo podría ser una memoria EEPROM, el cual es un dispositivo que puede transmitir y recibir información.

Los dispositivos conectados al bus son definidos como maestros y esclavos. Un maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control.

Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de una línea selectora llamada Chip Select o Select Slave, por lo tanto es esclavo es activado solo cuando esta línea es seleccionada. Generalmente una línea de selección es dedicada para cada esclavo. En un tiempo determinado T1, solo podrá existir un maestro sobre el bus. Cualquier dispositivo esclavo que no esté seleccionado, debe deshabilitarse (ponerlo en alta impedancia) a través de la línea selectora (chip select).

El bus SPI emplea un simple registro de desplazamiento para transmitir la información.

Modos de transferencia del SPI

Interfaz SPI permite transmitir y recibir datos simultáneamente en dos líneas (MOSI y MISO). Polaridad del reloj (CPOL) y la fase del reloj (CPHA) son los principales parámetros que definen el formato de reloj para ser utilizada por el bus SPI. En función del parámetro CPOL, el reloj del SPI puede ser invertido o no invertido. CPHA este parámetro se utiliza para desplazar la fase de muestreo. Si CPHA = 0 los datos se muestrean en el principal (primer) flanco de reloj. Si CPHA =

1 los datos se muestrean al final (segundo) flanco de reloj, sin importar si ese flanco de reloj está aumentando o disminuyendo.

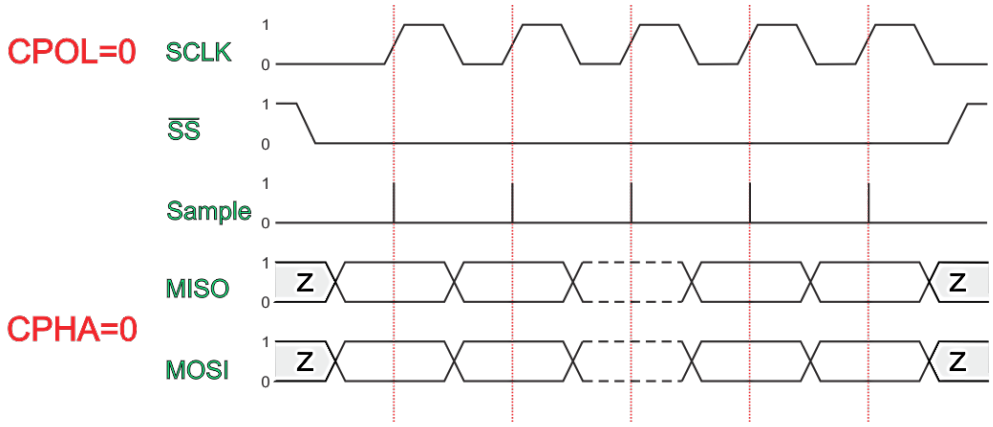


Figura 2-24.- Modo de Transferencia I, CPOL=0 y CPHA=0

En la figura 2-25 se muestra que los datos deben estar disponibles antes de levantarse la primera señal de reloj. El estado de reposo reloj es cero. Los datos sobre MISO y MOSI líneas deben ser estables mientras que el reloj es alta y se puede cambiar cuando el reloj es baja. Los datos son capturados en transición de bajo a alto del reloj y se propaga en alto a bajo transición de reloj.

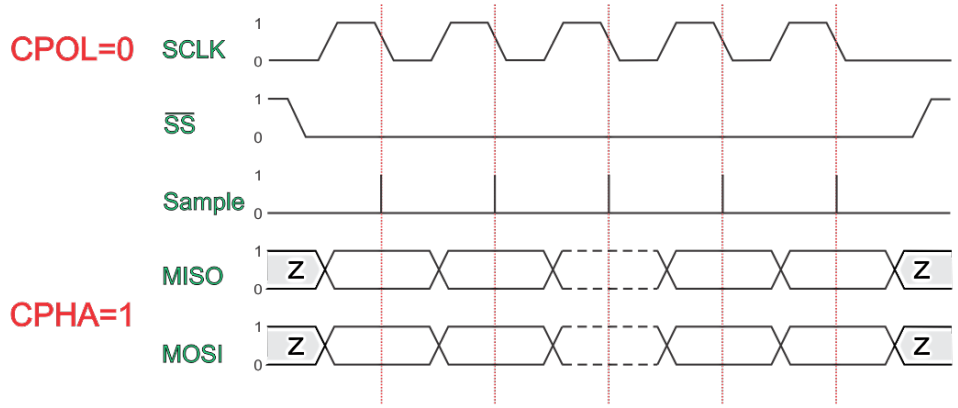


Figura 2-25.- Modo de trasferencia II, CPOL=0 y CPHA=1

En figura 2-26. La primera señal de reloj de levantamiento se puede utilizar para preparar los datos. El estado de reposo del reloj es cero. Los datos sobre MISO y MOSI líneas que deben ser estables mientras que el reloj es nivel bajo y se puede cambiar cuando el reloj está en alto. Los datos son capturados en la transición de alto a bajo del reloj y se propaga en bajo a alto transición de reloj.

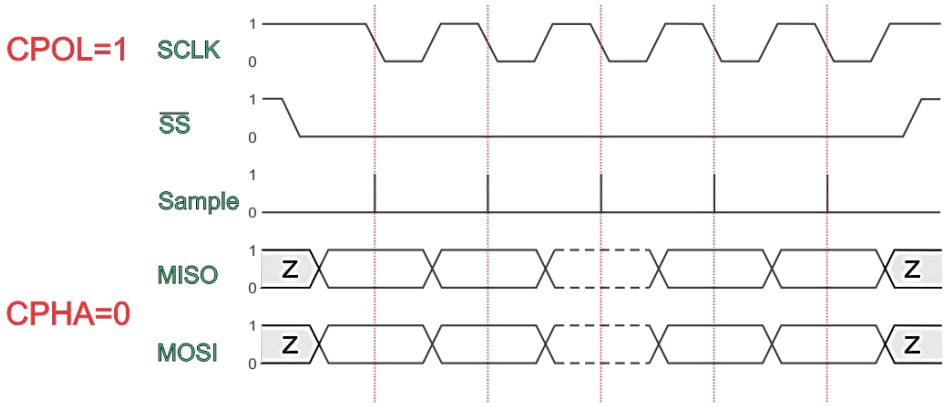


Figura 2-26.- Modo de transferencia III, CPOL=1 y CPHA=0

Para la figura 2-27, los datos deben estar disponibles antes de caer la primera señal de reloj. El estado de reposo reloj es uno. Los datos sobre MISO y MOSI líneas deben ser estables mientras que el reloj es baja y se puede cambiar cuando el reloj es alta. Los datos son capturados en la transición de alto a bajo del reloj y se propaga en bajo a alto transición de reloj.

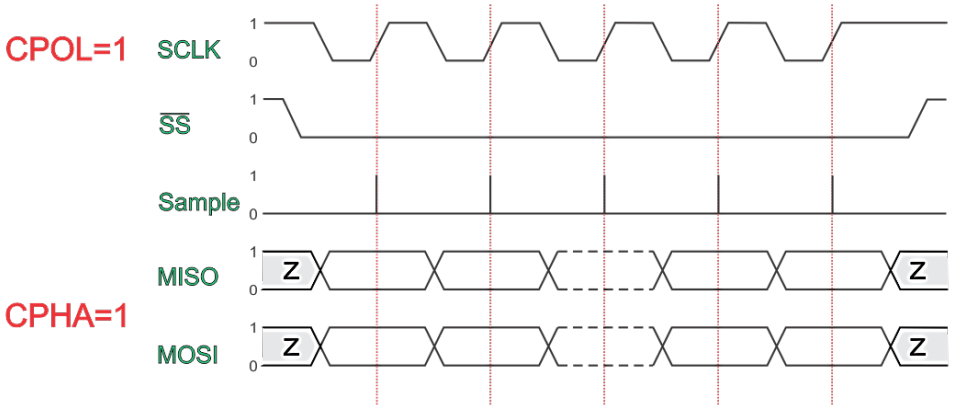


Figura 2-27.- Modo de transferencia IV, CPOL=1 y CPHA=1

En la figura 2-28 la primera señal de caída de reloj puede ser utilizada para preparar los datos. El estado de reposo reloj es uno. Los datos sobre MISO y MOSI líneas deben ser estables mientras que el reloj es alta y se puede cambiar cuando el reloj es baja. Los datos son capturados en transición de bajo a alto del reloj y se propaga en alto a bajo transición de reloj.

2.7.3 RAM (Memoria de acceso aleatorio)

Una de las características distintivas de las RAM (Random Access Memory) es la posibilidad de leer datos, como escribirlos rápidamente. La otra característica distintiva es que una RAM es volátil. Una RAM debe estar siempre alimentada por corriente eléctrica, si se interrumpe la alimentación se pierden los datos. Las dos formas básicas de memoria de acceso aleatorio son la RAM dinámica (DRAM – Dynamic RAM) y la RAM estática (SRAM – Static RAM). Una DRAM está hecha con celdas que almacenan los datos como cargas eléctricas en condensadores. (CCM, 2015)

La presencia o ausencia de carga en un condensador se interpreta como el 1 o 0 binarios. Ya que los condensadores tienen una tendencia natural a descargarse, las DRAM requieren refrescos periódicos para mantener memorizados los datos. Una

SRAM es un dispositivo digital, basado en los mismos elementos que se usan en el procesador. En una SRAM los valores binarios se almacena utilizando flip-flops, los cuales mantienen sus datos en tanto se mantengan alimentados. Otras formas de RAM son la RAM pseudo estática (PSRAM – Pseudo Static RAM) y la RAM no volátil (NVRAM – NonVolatile RAM). La PSRAM es una DRAM con un controlador refrescador de memoria embebido. La NVRAM es una variación especial de la RAM que es capaz de mantener datos, incluso luego de que se remueve la alimentación de poder.

3. Metodología

En esta sección se manejan los procedimientos que se requieren para generar una adquisición de la señal ECG, es decir, las etapas de acondicionamiento, así como como la selección de los materiales, diseño de diagramas e implementación de las interfaces que se usan para el sistema de adquisición de señales biométricas.

Se explica en cada sección los elementos necesarios a los que se recurrió para acondicionar la señal Electrocardiograma. Se busca optimizar los materiales y recursos para reducir gastos, para que la implementación del sistema sea fácil de usar y la aplicación del producto sencillo de ejecutar.

3.1 Planteamiento de la Metodología

La metodología describe como principal sujeto experimental un ser vivo, en este caso el roedor. Por lo que en el ser vivo se buscan los puntos de apoyo de los sensores, donde se podrá adquirir la señal ECG con mejor resolución. Los animales empleados en este proyecto son cuidados bajo los reglamentos de la Universidad de Minnesota el documento "Research Animal Resources", el cual se puede disponer en internet, este documento trata de cubrir todos los tratados que deben cuidarse cuando se expone un animal a experimentación. El documento tiene como objetivo proporcionar el cuidado, la salud y el bienestar de los animales utilizados para la investigación y la educación en la Universidad de Minnesota, estos mismos reglamentos se usan para la Universidad Autónoma de Querétaro.

Para administrar a las necesidades relacionadas con los animales de investigadores de la Universidad y educadores a través de la difusión de conocimientos y recursos. Y para servir al público, garantizando el cumplimiento de

todas las normas legales y éticas relacionadas con el uso de animales para la investigación y la educación en la Universidad Autónoma de Querétaro.

Con ayuda de los sensores o electrodos se podrá obtener el ECG (Sección 2-3), el primer paso es pre-amplificar la señal ECG, el siguiente paso es filtrar la señal mediante los filtros pasa-altas y pasa-bajas generando un filtro pasa-banda, los filtros se basan en la función butterworth. Y además se agregó un filtro notch para eliminar la frecuencia de 60 Hz, presente en el ambiente que nos rodea o de alguna emisión de algún dispositivo eléctrico cercano a nuestro sistema de adquisición.

Con nuestra señal filtrada, se envía al convertidor ADS para digitalizar nuestra bioseñal, donde se cuida que la conversión tenga una buena resolución y evitar que ruido externo afecte nuestro sistema. La alimentación del convertidor es configurada mediante reguladores de voltaje de alta precisión, para establecer un rango de valores lo más exacto posible. (Sección 2-6).

Se usa el protocolo SPI, para la comunicación con el ADS, donde se maneja el modo de transferencia SPI, con CPOL = 0 y CPHA = 1, que indica la funcionalidad del reloj (CLOCK), son los principales parámetros que definen el formato del reloj para ser usado por el bus del SPI. (Pérez, Protocolo SPI, 2000), (Figura 2-26).

La comunicación se diseña en hardware mediante la programación VHDL, por lo que se hace un driver para el ADS1298, se emplea el puerto USB como medio de interfaz entre la tarjeta FPGA y una computadora. Cuando la comunicación entre los sistemas FPGA-PC se enlaza correctamente, se diseña una interfaz de usuario la cual permite la visualización de la señal ECG en pantalla del computador. El siguiente diagrama (figura 3-1) puede describir el planteamiento de la metodología visualmente.

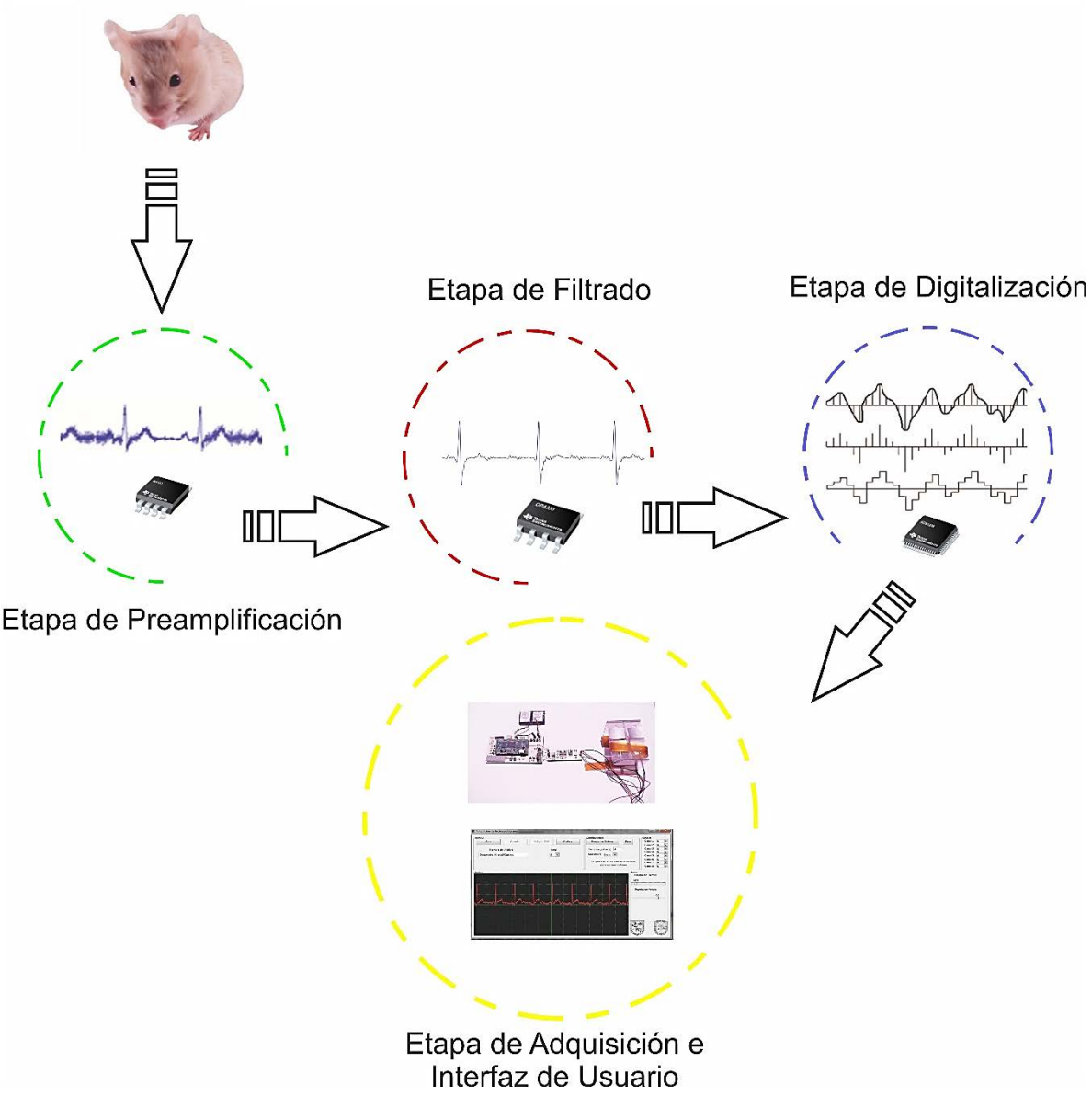


Figura 3-1.- Planteamiento de la Metodología

La adquisición de la señal ECG tiene varias etapas para obtener una visualización digital en una pantalla de computadora, por lo que se describen las etapas que más adelante se conocerán a fondo. A continuación cada etapa con las características de conexión.

Etapa de preamplificación.- Primer paso al sistema de adquisición.

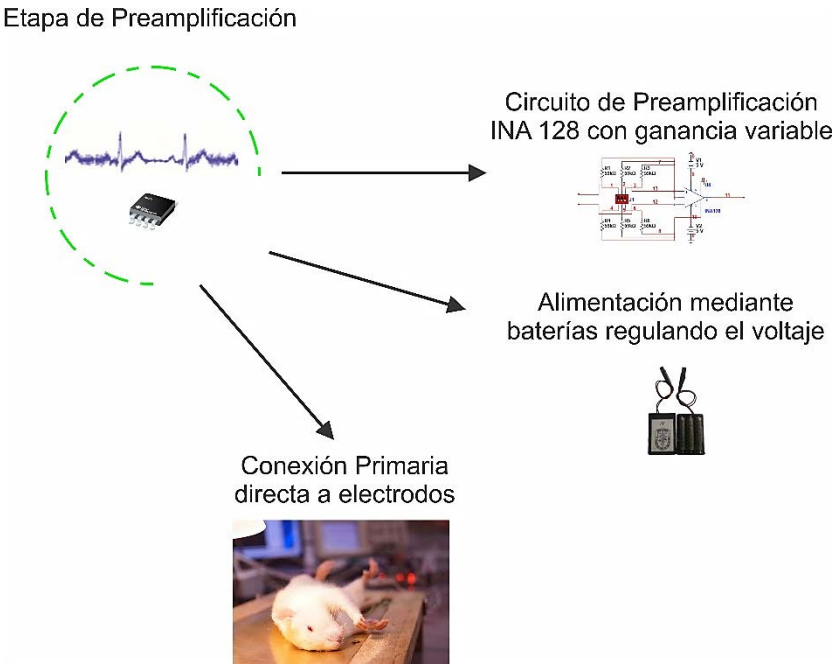


Figura 3-2.- Etapa de Preamplificación

Etapa de filtrado.- Reducir el ruido ambiental, muscular y red eléctrica.

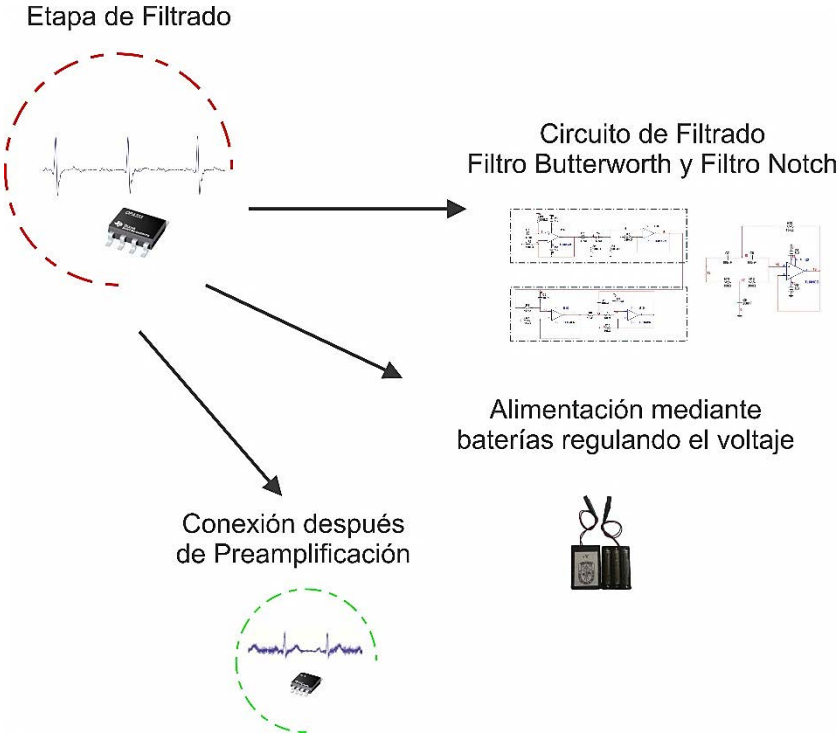


Figura 3-3.- Etapa de Filtrado

Etapa de digitalización.- Convertir datos analógicos a digitales con buena resolución.

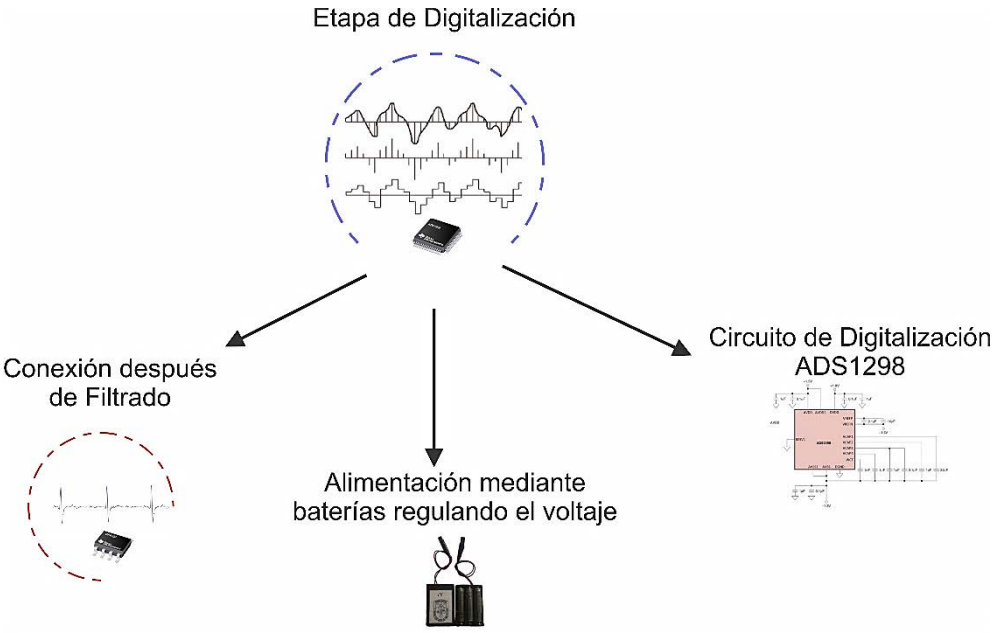


Figura 3-4.- Etapa de digitalización

Etapa de Adquisición e Interfaz de Usuario.-

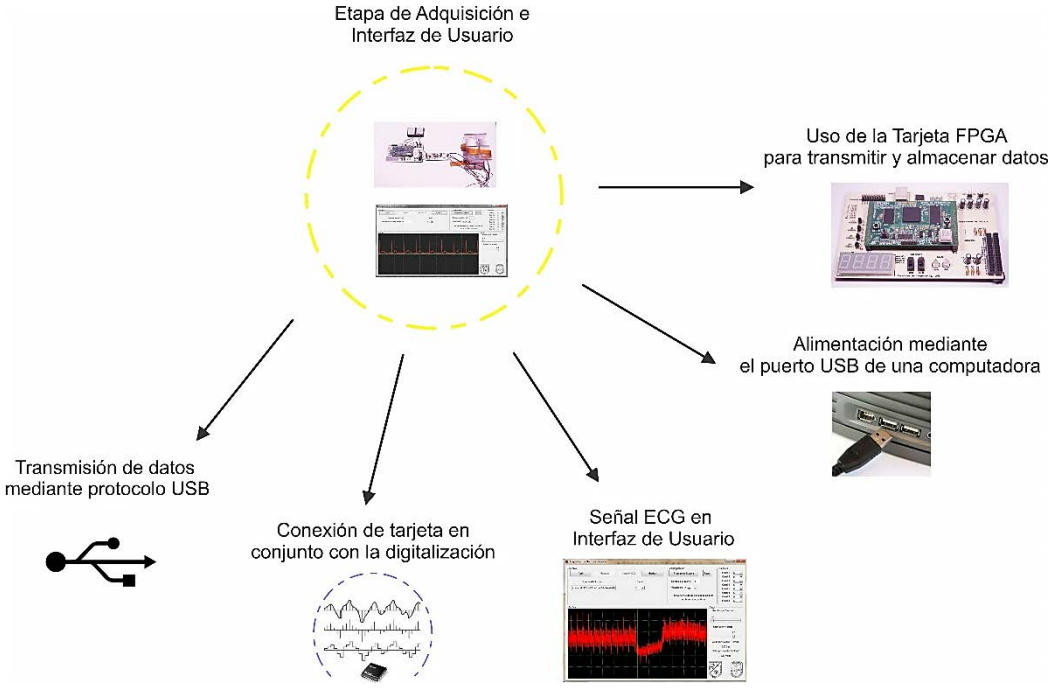


Figura 3-5.- Etapa de adquisición de datos e Interfaz de Usuario

3.2 Derivaciones del Electrocardiograma en Roedores.

Las derivaciones cardíacas son el registro de la diferencia de potenciales eléctricos entre dos puntos, ya sea entre dos electrodos (derivación bipolar) o entre un punto virtual y un electrodo (derivaciones monopolares). Cada derivación es un punto de vista distinto del mismo estimulo eléctrico. (Sección 2-2).

Existen 12 derivaciones electrocardiográficas, de las cuales se dividen en derivaciones bipolares I, II y III, de las cuales solo se requieren dos puntos de apoyo para obtener el diferencial eléctrico del corazón, también se cuenta con derivaciones monopolares aVR, aVL y aVF de las cuales se toma un punto positivo y dos puntos suman la referencia y por ultimo las derivaciones precordiales V1 a V6.

Para objeto de estudio en este proyecto se toman las derivaciones bipolares I, II y III como las necesarias para obtener el ECG, recordando la figura de las derivaciones (Figura 3.6), se muestra la derivación II en la rata y humano.

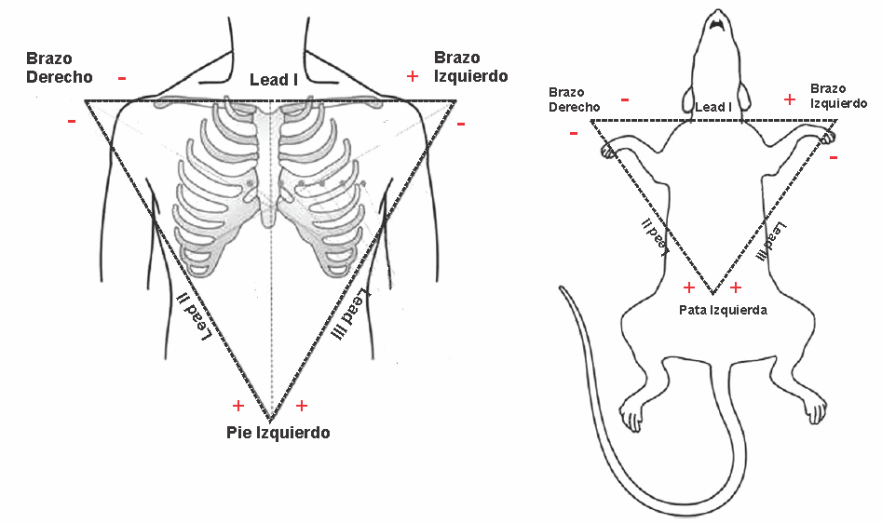


Figura 3-6.- Derivaciones A. Humano B. Roedor

Empleando las derivaciones bipolares se toma en cuenta que los puntos donde se colocan los electrodos son en las patas de la rata, para hacer una medición

de Electrocardiograma no invasivo, evitando dañar la integridad del animal. Una forma alternativa de colocar los electrodos es ponerlos en el pecho de la rata, por lo que se tiene que rasurar el sujeto de experimentación para que la zona de contacto sea adecuada.

El Electrocardiograma para humanos generalmente se encuentra por debajo de los 30-50 Hz pero cuando existe algún problema o se requiere hacer un diagnóstico se extiende el rango de frecuencias ya que el ECG se ve alterado con una reacción toxicológica o problemas del corazón, en los roedores sucede algo similar solo que el rango de frecuencias puede ser mayor hasta unos 55 Hz. Estos datos se toman para el diseño de los filtros pasa-banda.

3.3 Selección y Funcionalidad de Dispositivos Electrónicos

Para lograr una adquisición de Electrocardiograma eficaz se debe hacer una selección de dispositivos que cumplan con los estándares requeridos. Se busca que requieran poca alimentación, ya que el sistema se alimentara con baterías así que se busca maximizar la carga de las baterías, los dispositivos deben ser de alta precisión y alto nivel de CMRR (rechazo de modo común) (Sección 2-4-7).

Para comenzar se usa la tarjeta UPDSH del Grupo de Mecatrónica HSP Digital, como medio de comunicación y procesamiento de señales del convertidor, ya que es el microprocesador, usando un lenguaje en VHDL, se puede tener expectativas de que las aplicaciones sobre el Electrocardiograma pueden ser muy amplias. Se decidió escoger esta tarjeta debido a su capacidad de almacenamiento en memoria RAM, contiene dos memorias RAM, una estática y una dinámica, debido a la cantidad de datos obtenidos por la digitalización nos obliga a usar la memoria dinámica ya que puede almacenar hasta 256 Mb, es decir, podemos almacenar una

cantidad relativamente grande de datos mientras la memoria estática almacena 4 Mb lo que limita nuestro tiempo de grabación, un factor que interviene en este aspecto es que la velocidad de transmisión de datos por el USB no es tan rápida para poder enviar buses de datos a una velocidad de 8kHz, con esto se quiere decir que la interfaz de comunicación USB no nos satisface para visualizar el ECG en tiempo real, por esta situación se dio la alternativa de almacenar los datos en la memoria dinámica y así evitar perder datos de la bioseñal. (Sección 2-7-3).

El manejo en una plataforma en VHDL nos permite configurar y crear nuestro sistema, es otra característica por la cual se implementa en esta tarjeta.

La tarjeta electrónica se muestra en la figura 3-7.

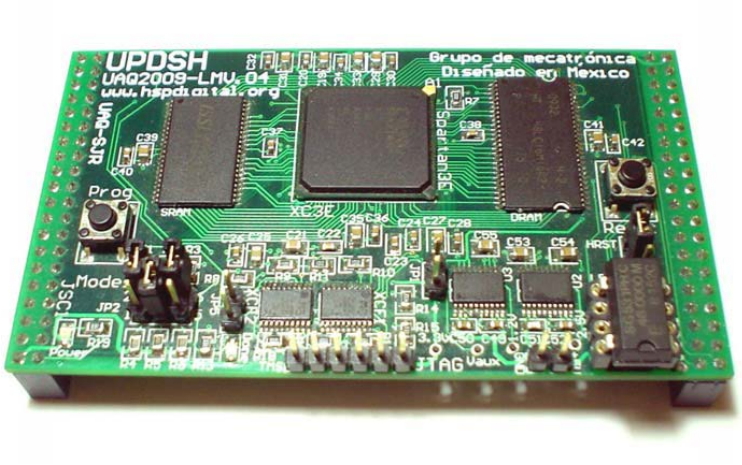


Figura 3-7.- Tarjeta electrónica sistema UPDSH. (Grupo de Mecatrónica-UAQ, 2009)

Como primer dispositivo a seleccionar es el preamplificador ya que al obtener el ECG con los electrodos o sensores, se debe amplificar para eliminar todo el ruido que provenga del ambiente o del mismo sensor.

Los amplificadores que se encontraron fueron los amplificadores de instrumentación de Texas Instrument, la serie "INAxxx" de los cuales son diseñados para trabajar en niveles de voltaje bajos, instrumentación médica y bajo nivel en

consumo de corriente que es una característica importante al alimentar con baterías. Así que se seleccionó el INA128, el cual posee las siguientes especificaciones.

Características	Capacidad
Baja corriente de entrada	5nA
Alto CMRR (rechazo de modo común)	120 db
Amplio rango de voltaje	+/-2.25 V a +/-18 V
Baja corriente de reposo	700 uA
Entradas protegidas	40 V

Tabla 3-1.- Características del Amplificador Operacional INA128

Las aplicaciones por las cuales fue seleccionado son: Amplificador de Termopar, Instrumentación Médica y en Adquisición de datos.

En la figura 3-8 se muestra el amplificador operacional INA128, una ilustración de sus características físicas y empaquetado.

Figura 3-8.- Amplificador de Instrumentación INA128. (Texas Instrument, INA128, 1995)

El dispositivo INA128 contiene información de su configuración para adquirir el Electrocardiograma, en la figura 3-9 se observa el esquema recomendado.

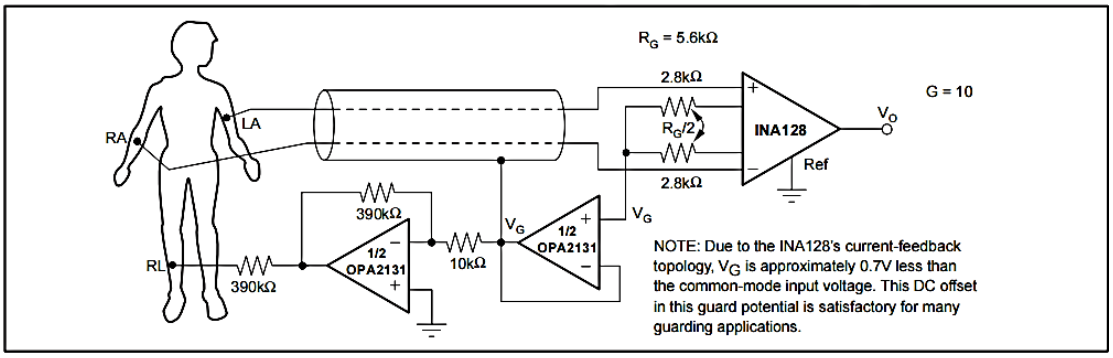


Figura 3-9.- Esquema amplificador ECG con INA128

En este esquema se puede ver que los electrodos se pueden conectar directamente al amplificador operacional, ya que el mismo INA128 contiene dentro un aislamiento, el cual evita cualquier corriente de retorno que pueda ser peligroso para el individuo, de la misma forma se emplea para roedores, ya que se emplea la misma técnica de aislamiento.

El diagrama que se muestra en la figura 3-9 nos sirve de base para crear el diseño que se implementa en este proyecto.

La ganancia selecciona las resistencias Rg las cuales se rigen por la siguiente tabla mostrada en la figura 3-10.

INA128:

$$G = 1 + \frac{50k\Omega}{R_G}$$

DESIRED GAIN (V/V)	INA128	
	R_G (Ω)	NEAREST 1% R_G (Ω)
1	NC	NC
2	50.00k	49.9k
5	12.50k	12.4k
10	5.556k	5.62k
20	2.632k	2.61k
50	1.02k	1.02k
100	505.1	511
200	251.3	249
500	100.2	100
1000	50.05	49.9
2000	25.01	24.9
5000	10.00	10
10000	5.001	4.99

Figura 3-10.- Tabla de ganancias de INA128

Para la etapa de filtrado se usan amplificadores operacionales TL084 ya que tienen un funcionamiento apropiado para diseñar filtros activos, debido a que las frecuencias presentes en el ECG son bajas para la funcionalidad del op-amp, en la tabla 3-2 se fijan las características de funcionamiento del TL08.

Características	Capacidad
Bajo consumo de energía	1.4 mA/Ch
Baja distorsión armónica	0.003 %
Corriente de polarización baja	30 pA
Voltaje de polarización	+/- 15 V

Tabla 3-2.- Características del OpAmp TL084

El op-amp TL084 se emplea en muchas aplicaciones electrónicas por sus características mencionadas, así que en este proyecto se logra usarlo para crear un filtro butterworth pasa-banda, tomando como base un filtro pasa-altas y un filtro pasa-bajas.

El filtro notch implementado en este proyecto requiere de un TL081, el cual es de la misma familia del TL08xx, donde la única diferencia entre ellos es el número de op-amps contenidos en el empaquetado. Esto se hace para evitar desperdiciar op-amps y malgastar la energía de las baterías.

A continuación se muestran los empaquetados usados en el proyecto (figura 3-11) de los amplificadores operacionales TL08xx.

Figura 3-11.- OpAmps Familia TL08xx. (Texas Instrument, TL08xx JFET-Input Operational Amplifiers, 2015)

La digitalización es una parte muy importante porque se requiere una precisión alta y evitar minimizar el ruido de fuentes externas o provocadas por el medio ambiente, así que el convertidor ADS1298 es el seleccionado por que entre sus diversas aplicaciones algunas son instrumentación médica. Este dispositivo cuenta con una amplia gama de configuraciones por lo que se concentran en la tabla 3-3.

Características	Capacidad
Alta resolución	24 bits
Velocidad de datos	500sps a 32Ksps
Ganancia programable	1, 2, 3, 4, 6, 8, 12
Canales	8
Bajo nivel de potencia	0.75mW/canal
Alimentación bipolar	1.65V a 3.6V
Corriente de polarización de entrada	200 pA
Alto CMRR (rechazo al modo común)	-115db

Tabla 3-3.- Características del convertidor ADS1298

Las aplicaciones recomendadas son: Instrumentación Médica para ECG, EMG, EEG y Adquisición de señales multicanal simultánea. Estas aplicaciones son principalmente para bioseñales por lo que su alta resolución en voltaje y tiempo, es el convertidor seleccionado para este proyecto.

El empaquetado usado para el convertidor se muestra en la figura 3.12.

Figura 3-12.- Convertidor ADS1298. (Texas Instrument, ADS129x Low-Power, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements, 2015)

El convertidor ADS1298 tiene una resolución de 24 bits distribuidos con los voltajes de alimentación, es decir, el rango de voltaje que se quiere trabajar. En este proyecto se trabajara con valores de +/- 2.5V por lo que si hacemos un cálculo de la resolución mínima que puede captar dicho convertidor es:

$$Resolución\ mínima\ = \frac{5\ V}{2^{24} - 1} = 29.8\ uV$$

Para que la resolución se mantenga y no pierda datos es necesario que su fuente de alimentación sea estable y precisa, por lo que se seleccionaron reguladores de voltaje de alta precisión, los cuales se alimentan de baterías para evitar ruido de la red eléctrica.

Los reguladores LT1963 y LT3015 (Linear Technology), regulación a +2.5V y -2.5V respectivamente, estos reguladores tienen una alta precisión regulación y tienen muy bajo ruido en la salida de voltaje. Estas características hacen que sean ideales para alimentar nuestro convertidor analógico-digital. Las características de dichos reguladores de voltaje están contenidos en las siguientes tablas 3-4 y 3-5.

Características	Capacidad
Voltaje de entrada	+/- 20V
Corriente de salida	1.5 A
Bajo nivel de ruido	40 uVrms
Voltajes de salida	1.5V, 1.8V, 2.5V y 3.3V

Tabla 3-4.- Características del Regulador de voltaje LT1963

El regulador positivo a +2.5V contiene un protector de corriente de retorno que evita cualquier retorno de corrientes no deseadas, las aplicaciones para cual fue diseñado es suministrar potencia lógica. Y su empaquetado se muestra en la figura 3-13.

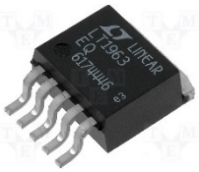


Figura 3-13.- Regulador de voltaje LT1963. (Linear Technology, LT1963 series, 2005).

Características	Capacidad
Voltaje de entrada	-1.8V hasta -30V
Corriente de salida	1.5 A
Bajo nivel de ruido	60 uVrms
Voltajes de salida	-2.5V, -3V, -3.3V, -5V, -12V y -15V

Tabla 3-5.- Características del Regulador de voltaje LT3015

Este tipo de regulador LT3015 es el complemento negativo del LT1963, es por esta razón que se decidió trabajar en conjunto de estos dispositivos. Entre las aplicaciones principales se encuentra que se usa en instrumentación de bajo ruido, alimentación industrial y como alimentación negativa lógica.

En la figura 3-14 se puede ver el dispositivo empleado así como el empaquetado que se usa (Q package).

Figura 3-14.- Regulador de voltaje LT3015. (Linear Technology, LT3015, 2011).

Los elementos hasta ahora descritos mencionan los elementos electrónicos que componen el sistema de adquisición de Electrocardiograma, pero también se requieren elementos secundarios para comprobar el funcionamiento o auxilian para identificar si el sistema se encuentra en proceso de adquisición, envío, etc. Por lo que se deben mencionar los otros dispositivos secundarios, los cuales se enuncian en la siguiente tabla 3-6.

Dispositivo electrónico	Características
Display de 7 segmentos a 4 dígitos	El display es de ánodo común y es conectado a la tarjeta de procesamiento UPDSH
Led's	Led's de alta luminosidad para enviar registros o señales al termino de los procesos y verificar funcionalidad del código.
Switches	Los switches sirven como elemento secundario y verificar procesos en hardware.
Conector USB tipo B	El conector es esencial ya que por interfaz usb es como se hace la comunicación con PC y tarjeta UPDSH.
Circuito Integrado LM311	Este elemento sirve para identificar si existe carga en las baterías ya que si se descargan a un nivel menor de 2.5 V el convertidor y el sistema de acondicionamiento no funcionan correctamente.

Tabla 3-6.- Dispositivos electrónicos secundarios.

Por ultimo cabe mencionar que se seleccionaron baterías para alimentar el sistema, las baterías alimentaran el convertidor ADS1298 y el sistema de acondicionamiento con el fin de disminuir el ruido inducido por la red eléctrica. Los datos ya digitalizados se almacenan en el sistema UPDSH, donde este sistema ya es alimentado por medio del USB. Así que las baterías solo se encargan de alimentar la parte de acondicionamiento de la señal y la conversión de datos.

Las baterías son AAA a un nivel de voltaje de 1.2V c/u y almacenan hasta 1800 mAh además de que son recargables para su uso continuo del sistema y evitando gastos extras.

Figura 3-15.- Paquete de Baterías AAA

Un objetivo a cumplir es que el sistema sea más económico que los productos comerciales con los que se disponen. Por lo que en la tabla 3-7 señala la lista de materiales empleados, la cantidad y precios encontrados de manera regional (Querétaro). Los precios pueden variar pero se respeta un margen adecuado para canalizar correctamente los costos.

Producto	Cantidad	Precio Unitario	Precio Total
Tarjeta Sistema UPDSH	1	$4000	$4000
Convertidor ADS1298	1	$680	$680
Amplificador Operacional INA128	1	$134	$134
Circuitos Integrados (TLOXX,LM311)	4	$6	$24
Reguladores de Voltaje	2	$90	$180
Placas Fenólicas de cobre	2	$20	$40
Electrodos (placa y parche)	8	$122.50	$980
Display 7 Segmentos 4 dígitos	1	$45	$45
Resistencias	42	$0.50	$21
Capacitores	22	$1.50	$33
Led's y switchs	5	$4	$20
Accesorios varios (conectores, cable, pines)	6	$10	$60
Baterías	6	$23.75	$142.50

Tabla 3-7.- Materiales del sistema

El costo total es de $6,359.50 en material empleado para el sistema de señales biométricas.

3.4 Diseño de Tarjetas Electrónicas de Acondicionamiento y Adquisición.

El diseño de un sistema electrónico debe ser probado con anterioridad para entender el funcionamiento, conexión, posibles fallas así como sus limitaciones, procurando que se elabore cuidando los aspectos mencionados ya que pueden ocasionar el mal funcionamiento del sistema.

En una primera búsqueda se encuentran las conexiones recomendadas por el fabricante y después se elaboraron los esquemas ya que se estuviera seguro de su conexión. Y en otros casos se buscó el mejor diseño de acondicionamiento, como ayuda para elaborar el mejor esquema.

En primera instancia se muestra el circuito de preamplificación (Figura 3-16) donde se considera un selector de ganancias mediante un dip-switch para cubrir algunos rangos de amplificación de ECG de humanos y roedores, las ganancias son 500, 1000 y 5000. Quiere decir que puede seleccionarse otra ganancia mediante hardware para inducir una amplificación mayor o menor.

El esquema muestra la señal de salida ECG 1, será transmitida a la siguiente etapa de filtrado donde se colocara igualmente la entrada y salida de la señal procesada.

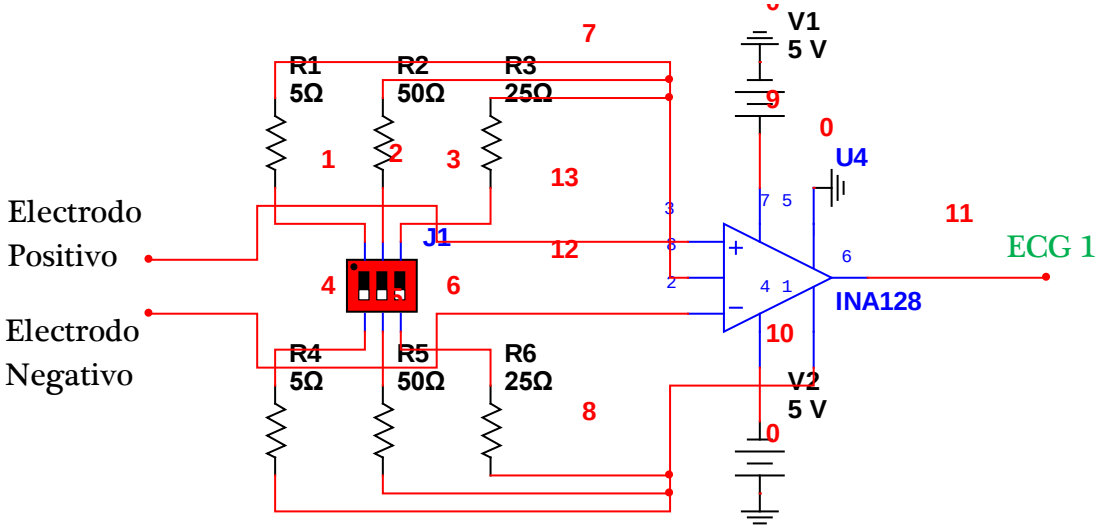


Figura 3-16.- Esquema Circuito de Preamplificación

Para proseguir se ven los esquemas para el filtrado de la señal ECG, comparando filtros se encontró que en la función chebyshev I en su banda de paso se observa un rizado que no es conveniente para la señal ECG, con la función chebyshev II su rizado se hace presente en la banda de supresión pero su ganancia no es tan plana por último la función butterworth su ganancia es más plana que los otros dos filtros y contiene un rizado en la banda de paso y en la banda de supresión, por lo tanto se decidió usar la función butterworth, así que para el diseño de los filtros butterworth se utiliza el esquema y parámetros de la célula Sallen-Key de sexto orden. Para los cálculos se tomó en cuenta que la frecuencia de corte superior es de 150 Hz y frecuencia de corte inferior de 0.05 Hz. (Sección 2-5)

Para este diseño se utilizara la conexión en cascada de un filtro pasa-baja con un filtro pasa-alta para facilitar el diseño. El filtro pasa-alto requiere la frecuencia de corte a 0.05 Hz y seleccionamos un valor de capacitor comercial y disponible C = 4.7uF así como una ganancia de K = 1.

Con la ecuación siguiente:

$$R = \frac{1}{2\pi * f * C} = \frac{1}{2\pi * 0.05Hz * 4.7uF} = 680k\Omega$$

Los cálculos arrojan un valor cercano a 680 kΩ pero se toma este valor por ser comercial. El filtro pasa-bajas requiere la frecuencia de corte a 150 Hz, seleccionando un capacitor de valor comercial C = 0.1 uF y ganancia de K = 1.

Con la ecuación:

$$R = \frac{1}{2\pi * f * C} = \frac{1}{2\pi * 150Hz * 0.1uF} = 10k\Omega$$

Se toma el valor más cercano al comercial para poder encontrar un diseño real. A continuación se observa el diseño de los filtros pasa-altas y pasa-bajas, esquema Sallen-Key tipo BP (figura 3-17).

El mismo esquema indica donde se coloca la señal ECG 1 de la etapa anterior, después de pasar los filtro butterworth señala la señal ECG 2, la cual se guiara a la siguiente etapa de filtro notch.

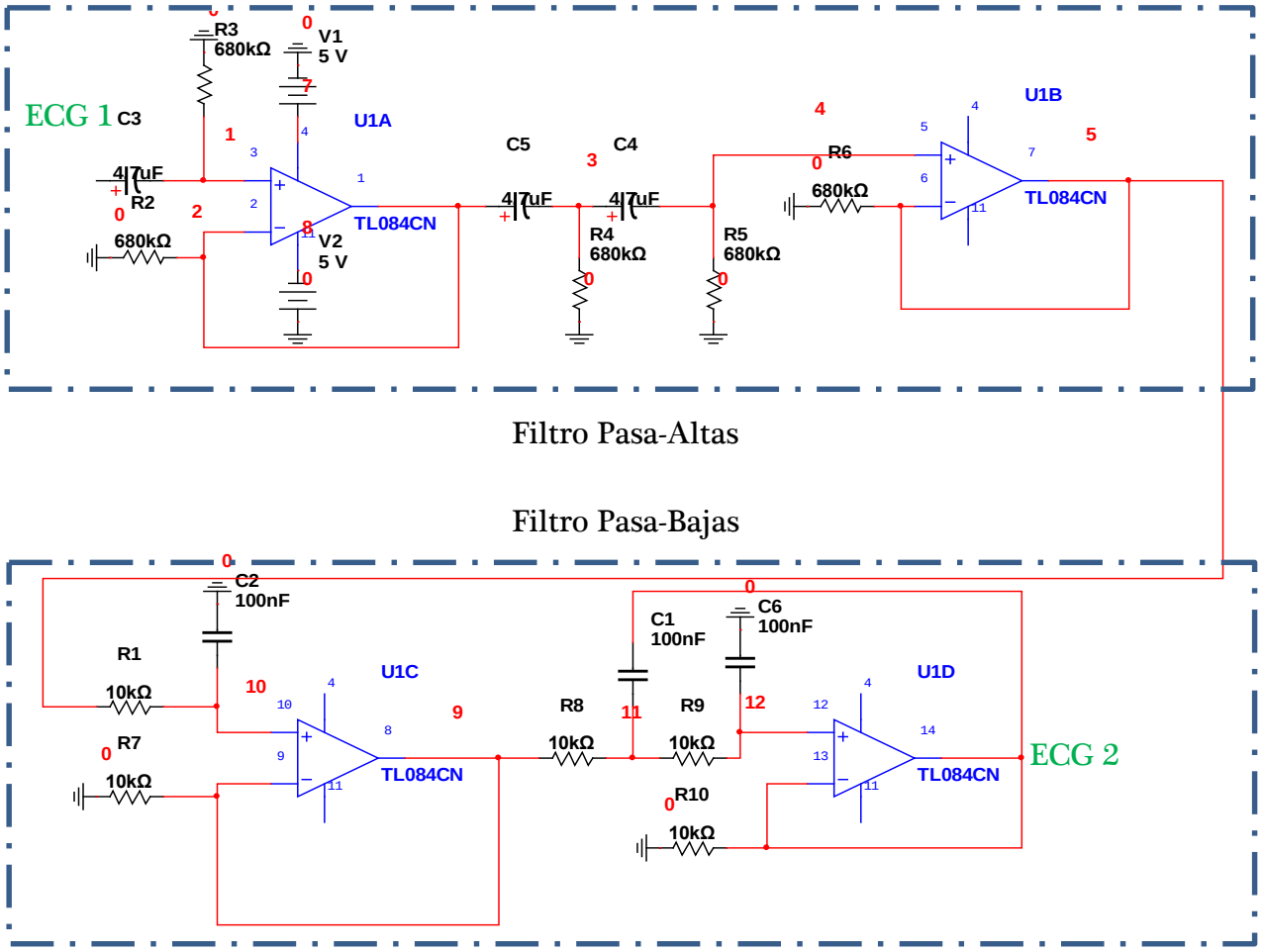


Figura 3-17.- Esquema de Filtro pasa-altas y pasa-bajas con célula Sallen-Key de sexto orden

Los filtros en cascada dejan un ancho de banda de 0.05 Hz y 150 Hz para reducir el ruido proporcionado por el rozamiento del electrodo con la piel o ruido de la transmisión de los datos por el cable y ruido por el movimiento muscular del ser vivo.

Un filtro que se agrega a este esquema de la figura 3-17, es el filtro notch a 60 Hz, para eliminar solo la frecuencia de 60 Hz ya que en cualquier sistema puede estar presente esta frecuencia, debido a que la red eléctrica proporciona esta frecuencia, por lo tanto se considera el circuito de la figura 3-18 para el diseño del filtro notch. (Sección 2-5-1-4)

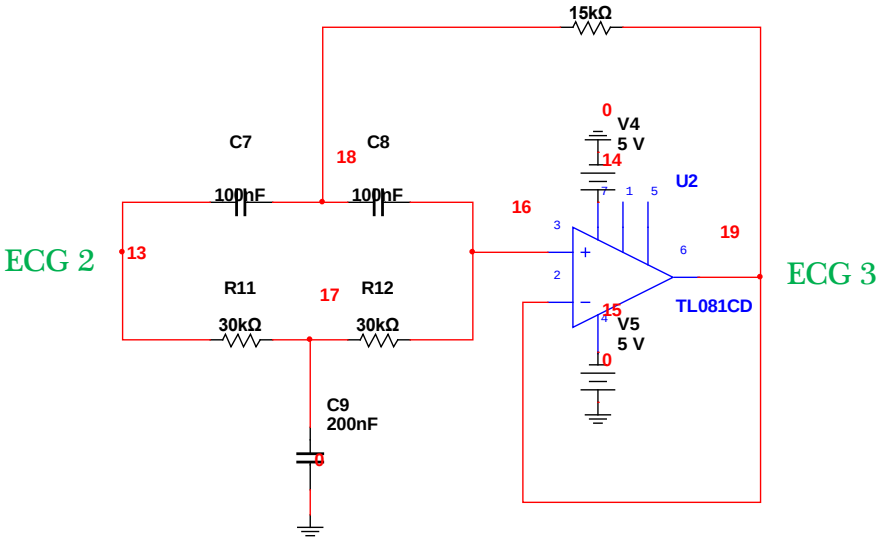


Figura 3-18.- Esquema de Filtro Notch.

Con los diagramas presentados se pueden unir ambas partes para lograr la etapa de filtrado, así que se hace una simulación del circuito para observar su comportamiento en la gráfica de bode y examinar si funciona como fue diseñado, es decir, si las frecuencias de corte son las esperadas.

La señal de salida ECG 3 se encuentra después del acondicionamiento, por lo que está lista para ser enviada al sistema de digitalización.

En la siguiente figura 3-19 se muestra la gráfica de bode de la unión de los circuitos de las figuras 3-17 y 3-18, para lograr el filtro pasa-banda con un filtro notch a 60 Hz.

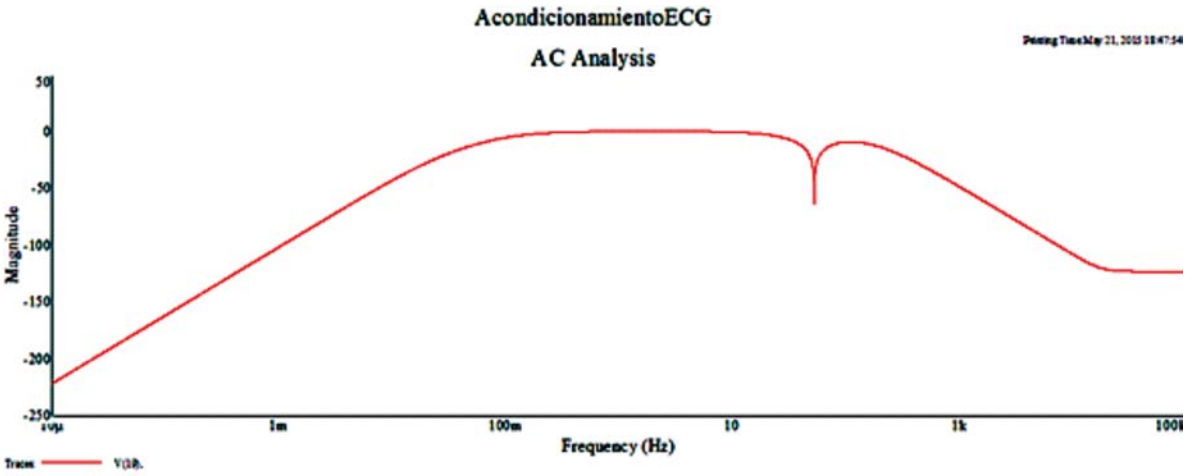


Figura 3-19.- Grafica de Bode de los Filtros Pasabanda y Notch

Se puede observar claramente la funcionalidad de los filtros en la gráfica de bode, ya que se muestran las frecuencias que son atenuadas, como era de esperarse el filtro no funciona idealmente, pero se logra eliminar las frecuencias que no se requieren. Un punto importante es denotar que el ancho de banda del filtro contiene una ganancia de uno de esta forma nuestra señal Electrocardiograma no tendrá una alteración de amplitud mayor o menor a la que nosotros calculemos.

Para la siguiente etapa; la digitalización de la señal se requiere del convertidor analógico-digital ADS 1298, el cual está diseñado para obtener señales biométricas, esta etapa se desarrolló en una tarjeta diferente al sistema de acondicionamiento de la señal ECG, para poder manipular diferentes entradas, es decir, que pueda funcionar para adquirir señales no solo precedentes del corazón, también de músculos (electromiografía), actividad eléctrica en el cerebro (electroencefalograma), etc. Las características de funcionamiento nos indican que puede manipular alto nivel de muestreo y una alta resolución. Por lo que en el diseño

se cuida el esquema y ruteo para preservar estos niveles de trabajo, así como se cuida la fuente de alimentación para no perder datos.

El convertidor ADS1298 (figura 3-20), tiene una configuración de 64 patitas por lo que contiene 16 entradas bipolares, lo cual genera 8 canales de entrada (pines del 1 al 16), estos canales se colocan a pines hembra para su fácil conexión.

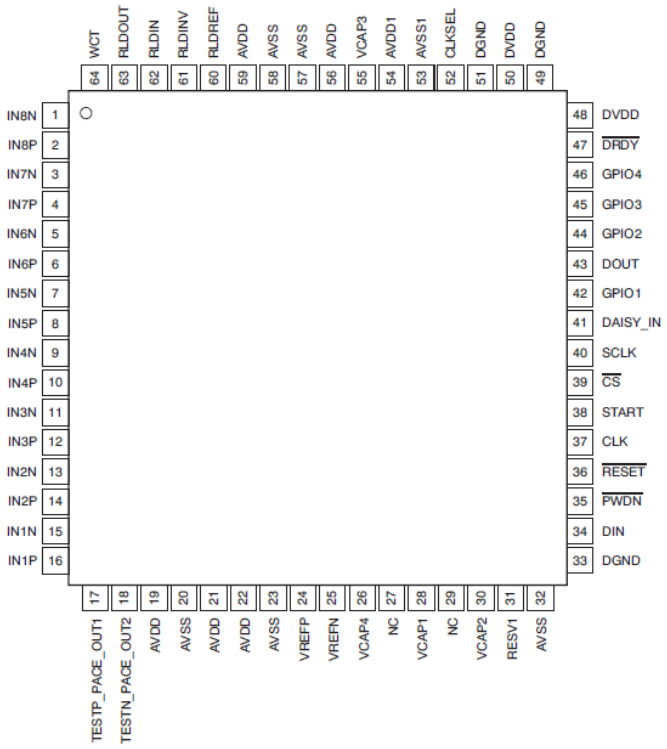


Figura 3-20.- Convertidor ADS1298 configuración de pines.

Los pines de configuración son del 34 al 47 y el pin 52, mediante estos pines se hace la transmisión de datos, generan la comunicación SPI (SCLK, CS, DIN, DOUT), el reseteo del sistema o el comienzo. Unas variables son para comprobar la comunicación periférica (GPIO 1-4). Estos pines se direccionan al sistema UPDSH, ya que el procesamiento se realiza en esta tarjeta.

La conexión de pines RLD (pin 60 a 63), se emplean para generar el registro de la pierna derecha, la cual se hizo en el circuito de acondicionamiento por lo que se dejaron libres estas entradas. Y la terminal WCT, se usa para el registro de las doce derivaciones. Lo cual en este proyecto solo se consideran 3 derivaciones.

El esquema de conexión de alimentación se obtuvo de la data general del convertidor ADS1298, el cual se muestra a continuación en la figura 3-21.

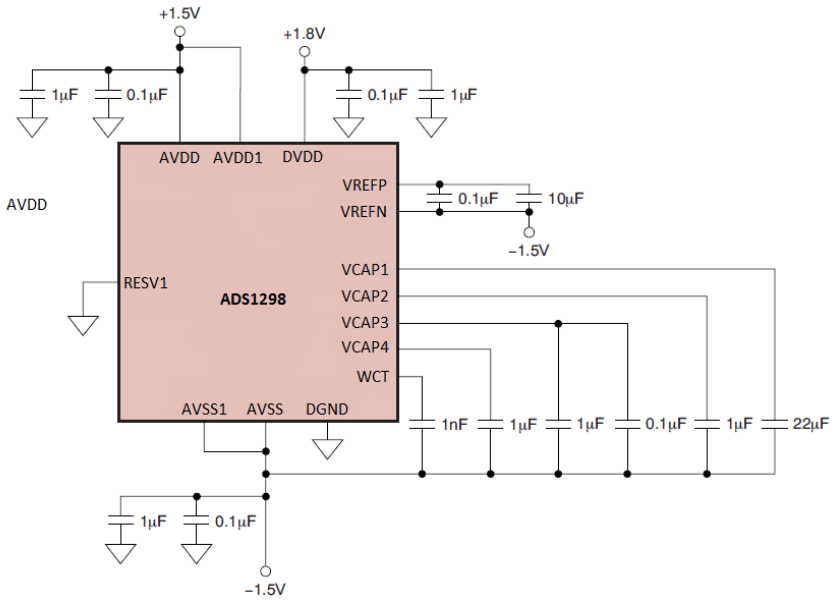


Figura 3-21.- Operación Bipolar del Convertidor ADS1298

Este esquema muestra la conexión de la alimentación bipolar, la cual sufre unos cambios ya que en nuestro caso alimentaremos con una fuente dual de +/- 2.5V (AVDD y AVSS) y la fuente digital de 3.3V (DVDD proveniente de la tarjeta FPGA – sistema UPDSH).

Se siguieron algunas recomendaciones de la data del convertidor, donde indica las conexiones o colocación de capacitores ya que deben colocarse lo más cerca posible del dispositivo.

La fuente dual de +/- 2.5V es generada por baterías, donde se regula el voltaje con los dispositivos LT1963 y LT3015, mediante los siguientes diagramas se diseña la fuente dual.

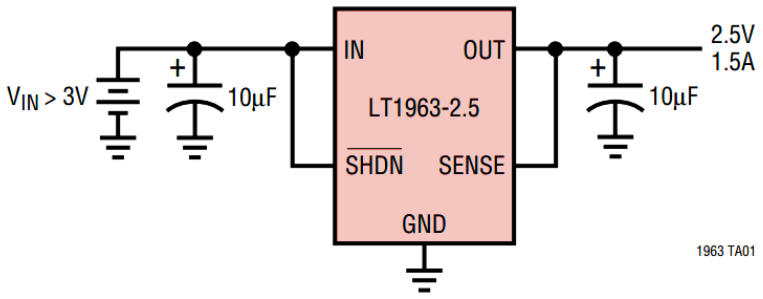


Figura 3-22.- Diagrama de conexión de LT1963

La figura 3-22 bosqueja la conexión básica para regular el voltaje positivo, donde se requiere de capacitores para eliminar el rizo que pueda contener la fuente. Y puede soportar hasta 1.5A para proporcionar la potencia necesaria para cada dispositivo conectado. Y en la figura 3-23 se puede ver el regulador de voltaje negativo, donde se permite en la entrada del dispositivo un voltaje máximo de -30V.

Para la comodidad del diseño se buscan los empaquetados similares y que puedan ser soldados correctamente, ya que una fuga de voltaje genera ruido en el sistema o pueden generarse problemas de corto circuito en la PCB. Recordando que los voltajes de alimentación no deben estar en puentes de cable porque podrían generar una antena y producir ruido, lo que es malo para el diseño.

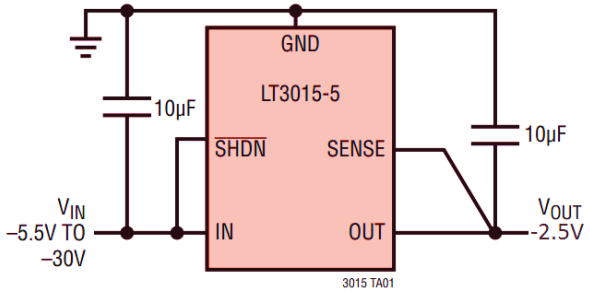


Figura 3-23.- Diagrama de conexión de LT3015

Para la tarjeta de digitalización se añadieron elementos auxiliares para comprobar el funcionamiento de la tarjeta. Los elementos de visualización: display de 7 segmentos, led's y switchs.

Se prueban los elementos individualmente para comprobar el correcto y apropiado funcionamiento, para evitar modificar una PCB debido a que es difícil cambiar las conexiones y el rendimiento no sería óptimo.

En la siguientes etapas se muestra la conexión de los elementos de interacción con el usuario como display de 7 segmentos, led's, switch y el comparador, ya que este último funciona para verificar si el nivel de voltaje de las baterías es el indicado, ya que si las baterías contienen un valor por debajo de +/- 2.5V el sistema no funcionaría correctamente, por lo que obtendríamos datos erróneos.

En la figura 3-24 se muestra la conexión correcta del display de 7 segmentos de 4 dígitos el cual se configura para hacer un rastreo de dígitos en programación VHDL, representando con flechas los conectores que van a la tarjeta FPGA. Este circuito señala el tipo de transistor empleado, ya que proporciona mejor potencia para transmitir el voltaje (Transistor BC548).

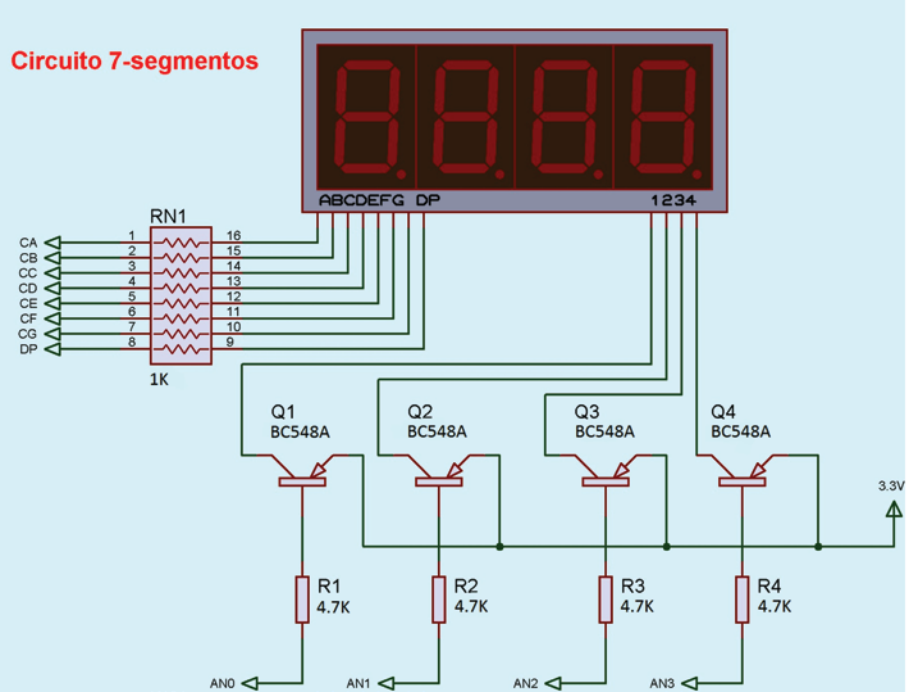


Figura 3-24.- Circuito del Display de 7 Segmentos ánodo común.

Para el circuito de los switchs, se ve en la figura 3-25 la conexión básica pull-down. En si el arreglo de resistencia funciona para drenar corriente que pueda generar una falla en la tarjeta FPGA.

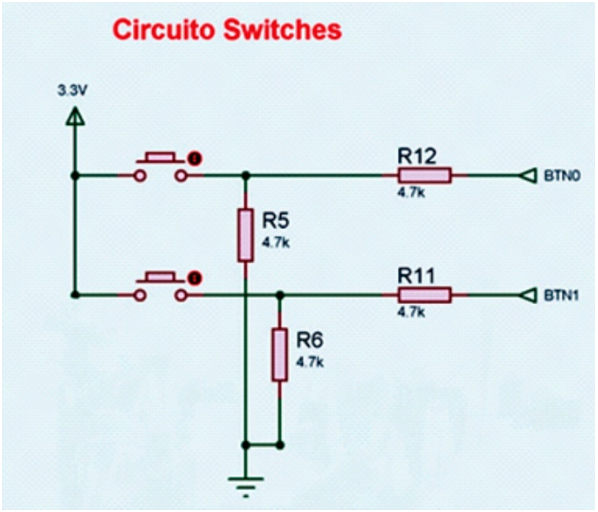


Figura 3-25.- Circuito para switchs.

En la figura 3-26 se observa el arreglo para los led's, donde se coloca la resistencia de 220Ω, la cual genera una corriente de 15mA, suficiente para encender apropiadamente los led's.

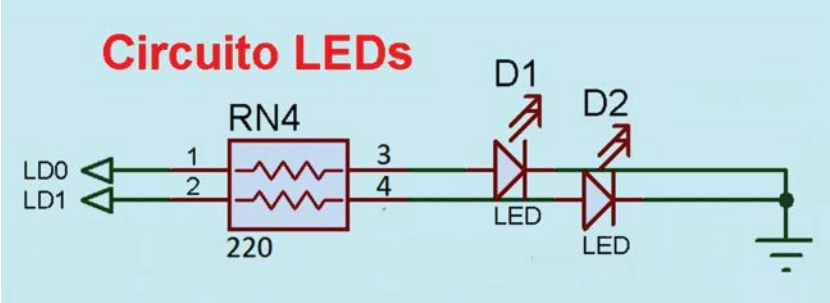


Figura 3-26.- Circuito para led's.

Para la figura 3-27, se puede observar el circuito de comparación de voltaje, el cual enciende un led cuando el voltaje mínimo requerido (+/- 2.5V) es aceptable y se apaga cuando es necesario recargar las baterías para seguir haciendo adquisición de señales. El op-amp empleado es el LM311, donde ya observamos las características eléctricas en la sección anterior. El arreglo de resistencias R5, funciona como divisor de tensión y fijar el voltaje a 2.7V como valor nominal de funcionamiento.

La ecuación siguiente formula el voltaje de comparación empleado para detectar cuando las baterías aun proporcionan la energía apropiada.

$$V_{nominal} = \frac{1k\Omega}{1k\Omega + 220\Omega} * 3.3V = 2.7V$$

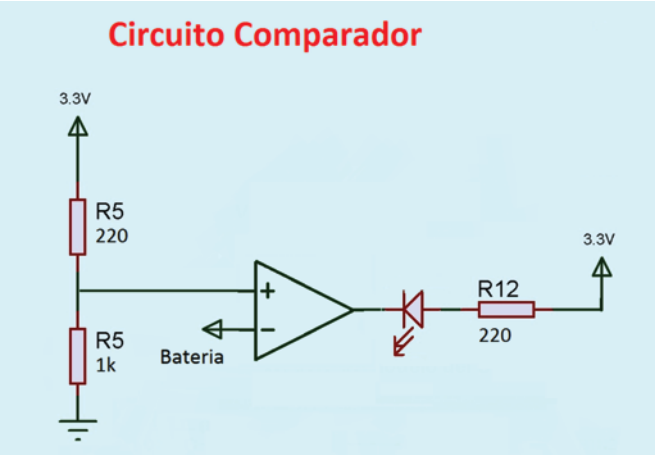


Figura 3-27.- Circuito de comparación de voltaje.

La interfaz de comunicación USB entre la tarjeta FPGA y la computadora requiere de un conector y el esquema básico de USB el cual es visto en la figura 3-28.

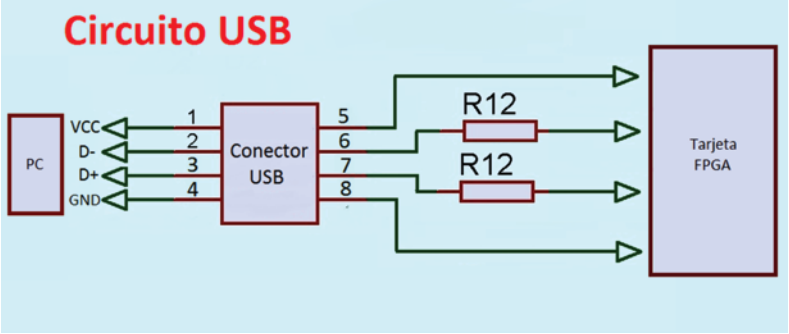


Figura 3-28.- Circuito de Interfaz USB

La conexión entre pc-tarjeta requiere un par de resistencias de 33Ω (R12) así se evita cualquier corriente que pueda ocasionar corrientes que dañen el sistema, el circuito de interfaz usb es sencillo pero es un protocolo de comunicación altamente usado por su versatilidad y transmisión de datos en oficinas, debido a los niveles de voltaje de trabajo del USB, no es recomendable usar esta interfaz en industrias.

Los circuitos se implementan en la plataforma de software Altium Designer 08, se hacen dos tarjetas de desarrollo, la primera con el circuito de acondicionamiento con etapa de preamplificacion y los filtros pasa-banda y notch. La segunda tarjeta electrónica se diseña implementando la digitalización y demás dispositivos electrónicos que ayudan a visualizar las señales o registros, como display de 7 segmentos, led's, switchs y el indicador de carga en baterías. Todo el sistema fue conectado empleando la misma referencia tierra, por lo que se considera una desventaja en el sistema pero se completa de esta forma ya que ayuda a referenciar los niveles de voltaje en animales.

Los esquemas, conexiones de todo el sistema y ruteo se encuentran en la sección de Apéndice.

En la figura 3-29 y 3-30 se muestran las tarjetas de adquisición de señales biométricas, en la primera tarjeta electrónica se observa la tarjeta de acondicionamiento de la señal ECG y en la segunda tarjeta se señala la tarjeta de adquisición, transmisión y digitalización de señales biométricas.

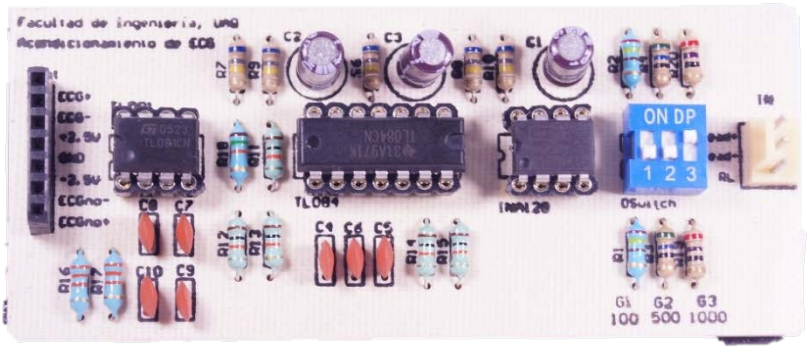


Figura 3-29.- Tarjeta de acondicionamiento de la señal ECG

En la figura 3-29 se pueden observar las etapas de acondicionamiento así como la ganancia selectora que nos proporciona rangos de operación más flexibles al obtener señales ECG.

El sistema de adquisición de señales biométricas se muestra en la figura 3-30, donde se compone de la tarjeta con el convertidor analógico-digital y de los elementos auxiliares de visualización. Y con el sistema UPDSH montado sobre la tarjeta de adquisición, la cual sirve para almacenar datos, crear la comunicación SPI y USB.

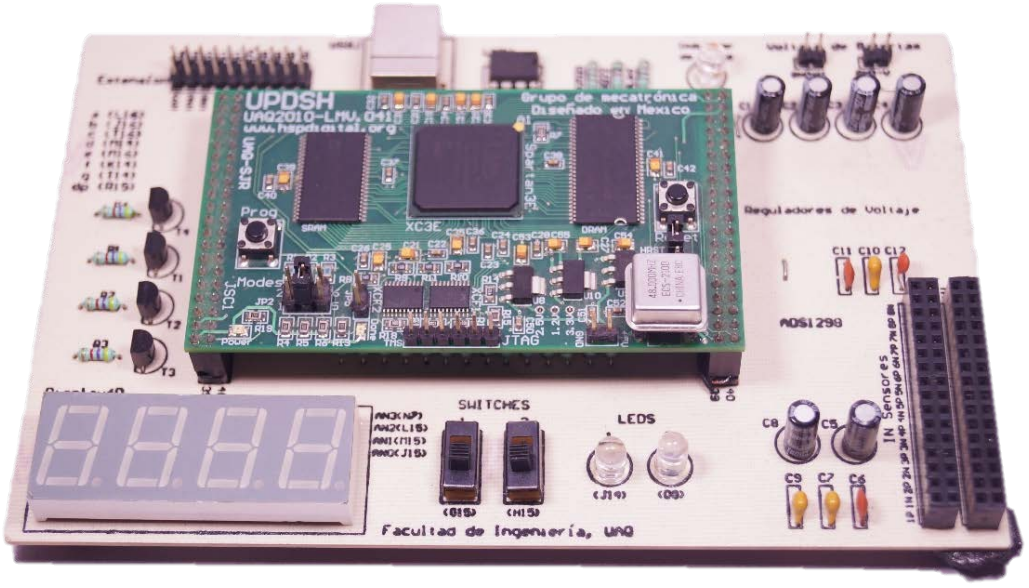


Figura 3-30.- Tarjeta de adquisición de señales biométricas

3.5 Implementación de Interfaz SPI

Los protocolos de comunicación son empleados para la transmisión de datos entre dispositivos electrónicos como convertidores, magnetómetros, acelerómetros, microprocesadores, etc. La interfaz de comunicación del convertidor ADS1298, se diseñó en hardware, mediante la programación en VHDL empleando el sistema UPDSH.

El protocolo que usa el convertidor analógico-digital es la interfaz SPI, comúnmente empleado para la transmisión de datos, configuraciones de dispositivos

y control de señales. La implementación del protocolo SPI se diseña a partir de los requerimientos del convertidor ADS1298 (figura 3-31), el cual menciona en su datasheet que debe tener una configuración de CPOL = 0 y CPHA = 1; esto quiere decir que la señal de reloj SCLK comenzara en bajo cuando la señal CS habilite la recepción de datos (CPOL=0 el reloj en estado estacionario es 0) y la adquisición de datos se registra en el primer cambio de estado de reloj, es decir, en el flanco descendente de SCLK (CPHA=1). Las señales deben cumplir un tiempo mínimo para el correcto funcionamiento del componente electrónico.

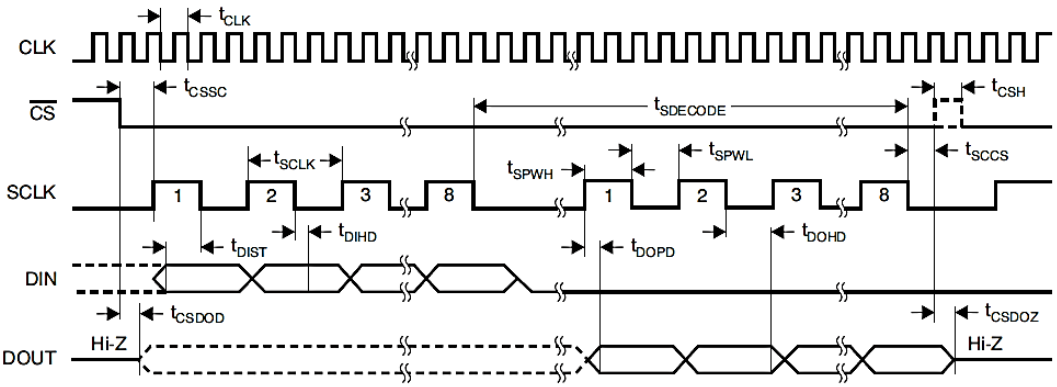


Figura 3-31.- Interfaz de comunicación serial del convertidor ADS1298.

Los requerimientos de tiempo se obtienen de la información proporcionada en línea por Texas Instruments, tabla 3-8 señala los tiempos mínimos recomendados. La señal "Master clock" se genera dentro del convertidor por lo que no requiere crearla, debido a la alimentación de 3.3V en DVDD funciona de 414 ns a 514 ns.

Las señales CS, SCLK y DIN se configuran por lo que el tiempo mínimo debe ser calculado, checando la latencia de la configuración de la máquina de estados (FSM), aunque el dispositivo FPGA trabaja a una frecuencia de operación a 48 MHz se debe comprobar.

Cada ciclo de trabajo del reloj maestro en la tarjeta FPGA es de 20.83 ns, en la secuencia de las máquinas de estado implica que el cambio de estado mínimo es de 20.83 ns, por lo que el tiempo t_{CSSC}=20.83 ns, donde es mayor al que indica la tabla 3-8, de tal forma se asegura que el tiempo requerido de latencia es apropiado. En el uso de t_{SCLK} se colocó la variable de tiempo de 8 ciclos de reloj por lo que es de 166.64 ns. El retardo de las demás señales es muy pequeño debido al campo de trabajo usado en t_{SCLK} y el tiempo en los cambios de estados.

		2.7V ≤ DVDD ≤ 3.6V			1.65V ≤ DVDD ≤ 2V			
PARAMETER	**DESCRIPTION**	**MIN**	**TYP**	**MAX**	**MIN**	**TYP**	**MAX**	**UNIT**
t_{CLK}	Master clock period	414		514	414		514	ns
t_{CSSC}	CS low to first SCLK, setup time	6			17			ns
t_{SCLK}	SCLK period	50			66.6			ns
$t_{SPWH, L}$	SCLK pulse width, high and low	15			25			ns
t_{DIST}	DIN valid to SCLK falling edge: setup time	10			10			ns
t_{DIHD}	Valid DIN after SCLK falling edge: hold time	10			11			ns
t_{DOHD}	SCLK falling edge to invalid DOUT: hold time	10			10			ns
t_{DOPD}	SCLK rising edge to DOUT valid: setup time			17			32	ns
t_{CSH}	CS high pulse	2			2			t_{CLKs}
t_{CSDOD}	CS low to DOUT driven	10			20			ns
t_{SCCS}	Eighth SCLK falling edge to CS high	4			4			t_{CLKs}
$t_{SDECODE}$	Command decode time	4			4			t_{CLKs}
t_{CSDOZ}	CS high to DOUT Hi-Z			10			20	ns
$t_{DISCK2ST}$	DAISY_IN valid to SCLK rising edge: setup time	10			10			ns
$t_{DISCK2HT}$	DAISY_IN valid after SCLK rising edge: hold time	10			10			ns

Tabla 3-8 Requerimientos de tiempo para interfaz serial

Inicializar el sistema requiere de un diagrama de flujo, mostrado en la figura 3-32, donde se observa los pasos a seguir para hacer conversiones analógicas digitales empleando el dispositivo ADS1298.

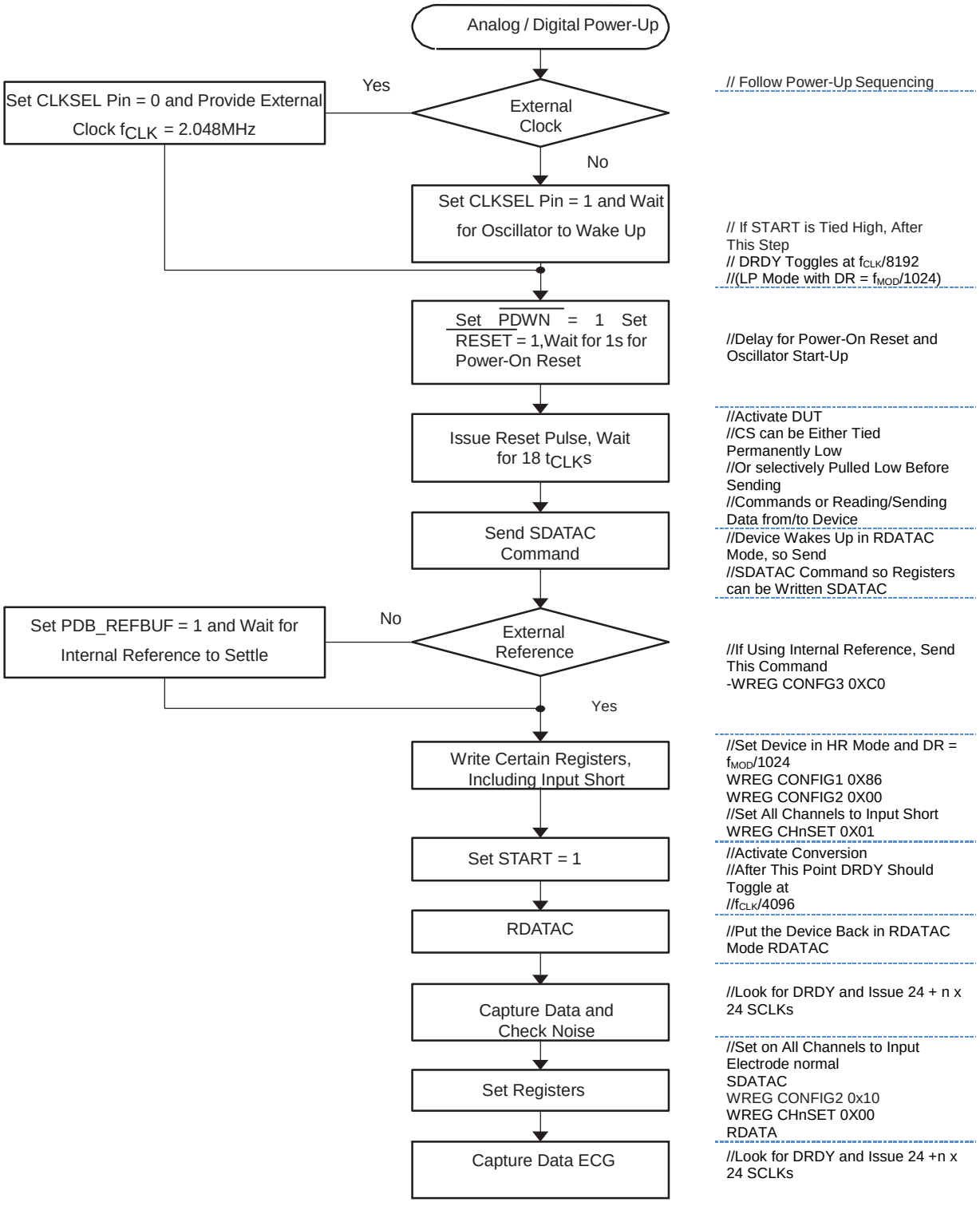


Figura 3-32. Diagrama de flujo de inicialización

El diagrama de la figura 3-32 señala los pasos de iniciación que se requiere para configurar el dispositivo de acuerdo a la función de ecg, como primer paso es configurar el reloj maestro, el cual se selecciona el reloj interno por lo que se debe colocar el pin CLKSEL en alto. Después se coloca PDWN y RESET en nivel alto para encender el convertidor y se debe esperar 1 segundo a la respuesta del mismo dispositivo. Al tener las habilitaciones compuestas está listo para empezar a codificar información y configurar el dispositivo, así que se puede continuar generando la señal CS que permita el inicio del protocolo SPI.

El dispositivo por default se encuentra en estado de RDATAC (lectura de datos continua) por lo que se debe enviar la trama SDATAC para detener la lectura. Los comandos son enlistados en la tabla 3-9 donde se pueden apreciar las tramas que se requieren para colocar en varios modos de operación el convertidor ADS1298. En este caso primero enviamos el op-code 0001 0001, que nos permite colocar el dispositivo en modo SDATAC.

COMMAND	DESCRIPTION	FIRST BYTE	SECOND
System Commands			
WAKEUP	Wake-up from standby mode	0000 0010 (02h)	
STANDBY	Enter standby mode	0000 0100 (04h)	
RESET	Reset the device	0000 0110 (06h)	
START	Start/restart (synchronize) conversions	0000 1000 (08h)	
STOP	Stop conversion	0000 1010 (0Ah)	
Data Read Commands			
RDATAC	Enable Read Data Continuous mode. This mode is the default mode at power-up.(1)	0001 0000 (10h)	
SDATAC	Stop Read Data Continuously mode	0001 0001 (11h)	
RDATA	Read data by command; supports multiple read	0001 0010 (12h)	
Register Read Commands			
RREG	Read *n nnnn* registers starting at address *r rrrr*	001*r rrrr* (2xh)	000*n nnnn*
WREG	Write *n nnnn* registers starting at address *r rrrr*	010*r rrrr* (4xh)	000*n nnnn*

Tabla 3-9.- Definición de Comandos SPI

El siguiente paso en el diagrama de flujo es colocar el voltaje de referencia, sea el interno o colocar el externo, en esta aplicación se coloca la referencia interna por lo que se debe enviar el registro de escritura "WREG CONFG3 0XC0", como se observa en la tabla 3-9 se requiere enviar dos tramas una indicando que dirección se va a escribir y hasta que registro se configurará. Se interpreta mejor con la figura 3-28. Se observa que la señal se envía OPCODE1 en este caso es 0100 0011 y después OPCODE2 = 0000 0000, ya que el primer OPCODE indica que registro se inicia se coloca la dirección 0x03 que pertenece a CONFG3, el segundo OPCODE señala hasta que dirección se va a configurar por lo tanto se envía la trama 0x00 ya que se considera que solo se escribirá en un registro, si es 0x01 indica que son 2 direcciones consecutivas, para 0x02 se configuran 3 direcciones y así sucesivamente.

Los siguientes datos que se envían por Din son los bits de configuración de la dirección 0x03, se especifica en el diagrama de la figura 3-33 (WREG CONFG3 0xC0), por lo que los siguientes bits correspondientes son 1100 0000.

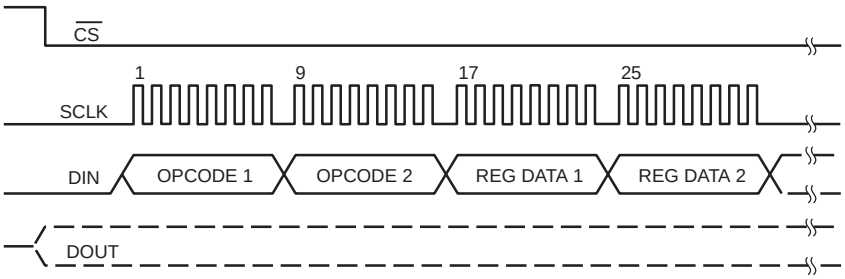


Figura 3-33. Ejemplo escritura de comandos

De esta forma se escriben registros para el uso del convertidor, en este ejemplo se muestra como se activa la referencia interna. Para continuar con el diagrama de flujo se configura una prueba de ruido, donde indica los registros a modificar los cuales se generan de la misma manera que el ejemplo anterior. Para conocer todas las direcciones de configuración del convertidor se puede consultar la

datasheet del dispositivo. Mientras que en la tabla 3-10 se señalan las direcciones empleadas en el diagrama de flujo.

ADDR	Register	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
01h	CONFIG1	HR	DAISY_EN	CLK_EN	0	0	DR2	DR1	DR0
02h	CONFIG2	0	0	WCT_CH	INT_TEST	0	TEST_A	TEST_FRE	TEST_FR
03h	CONFIG3	$\overline{\text{PD\_REFBUF}}$	1	VREF_4V	RLD_MEAS	RLDREF_INT	$\overline{\text{PD\_RLD}}$	RLD_Loff SENS	RLD_STAT
05h	CH1SET	PD1	GAIN12	GAIN11	GAIN10	0	MUXn2	MUXn1	MUXn0
06h	CH2SET	PD2	GAIN22	GAIN21	GAIN20	0	MUX22	MUX21	MUX20
07h	CH3SET	PD3	GAIN32	GAIN31	GAIN30	0	MUX32	MUX31	MUX30
08h	CH4SET	PD4	GAIN42	GAIN41	GAIN40	0	MUX42	MUX41	MUX40
09h	CH5SET	PD5	GAIN52	GAIN51	GAIN50	0	MUX52	MUX51	MUX50
0Ah	CH6SET	PD6	GAIN62	GAIN61	GAIN60	0	MUX62	MUX61	MUX60
0Bh	CH7SET	PD7	GAIN72	GAIN71	GAIN70	0	MUX72	MUX71	MUX70
0Ch	CH8SET	PD8	GAIN82	GAIN81	GAIN80	0	MUX82	MUX81	MUX80

Tabla 3-10.- Asignación de Registros

Al configurar el dispositivo se debe ingresar la ganancia de los canales, la velocidad de muestreo y muestras a obtener, la ganancias de los canales se coloca en las direcciones 0x05 a 0x0C, y la velocidad de muestreo en CONFIG1 (0x01). Se establece el pin START en alto, aunque puede estar desde el principio en lógica positiva y funciona correctamente. Al leer la prueba de ruido (seleccionada en CONFIG2) se debe mandar el comando RDATAC de la tabla 3-9. Por lo que se reciben los datos de todos los canales, paquete de 216 bits (24 bits de registro y 24 bits por cada canal, recordando que el dispositivo maneja 8 canales, es decir, 24 bits x 8 canales = 192 bits (canales) + 24 bits (registro)).

Para configurar los canales y realizar lecturas de Electrocardiograma se debe enviar el comando SDATAC para detener la prueba de ruido y realizar las configuraciones correspondientes que se describen en el diagrama de figura 3-32 (Set Registers). Las configuraciones se pueden manipular mediante software, se configura la velocidad de muestreo, el tiempo a grabar y las ganancias de cada canal (registros antes mencionados), estos valores se configuran en el registro 1 (CONFG1).

Se detiene la prueba de ruido mediante “CONFIG2 0x10”. Y se selecciona en los canales que se obtendrán señales provenientes de electrodos normales “CHnSET 0x00”. La interfaz en software se observara con más detalle en la siguiente sección.

Para recibir datos de Electrocardiograma es preferible manejar el comando de lectura RDATA y enviar cada que se reciben paquetes de 216 bits. La diferencia entre los modos de recepción RDATAC y RDATA es que al recibir el paquete de 216 bits, RDATA tiene que ser enviado antes de cada adquisición de datos para evitar pérdidas. Ya que es la mejor forma de recibir la señal biométrica.

La interfaz del protocolo SPI, es regido por una máquina de estados finitos (FSM), diseñada en el lenguaje de programación en hardware “VHDL”, registra señales de control, envío y recepción de datos. Se observa en la figura 3-34 el diagrama a bloques de la máquina de estados.

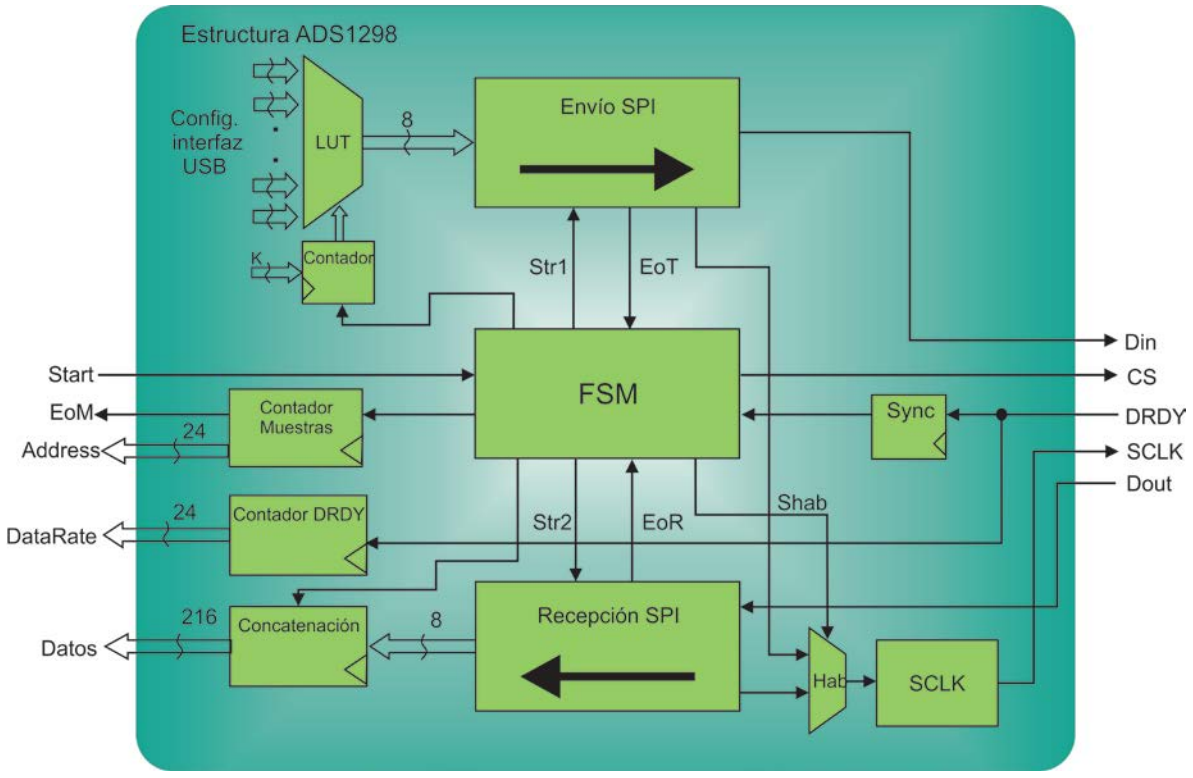


Figura 3-34.- Diagrama a bloques de estructura de ADS1298

Este diagrama representa las señales de control y módulos que se usan para el envío y recepción de datos, FSM controla el registro de cada una de ellas. La modulación de SCLK se realiza con un registro combinacional el cual genera un reloj de aproximadamente 6 MHz, usando un oscilador de 48 MHz, el cual se habilita cuando se hace uso del convertidor tanto para enviar como para recibir.

Como se observa en figura 3-31, se conoce que CS activa el protocolo SPI, por lo tanto es una señal enviada por FSM para iniciar el protocolo de comunicación. En el bloque LUT se almacena la configuración y direcciones a escribir las cuales son configurables mediante la interfaz periférica USB, el envío de datos se realiza serialmente por el puerto Din, lo cual es controlado por el modulo "Envío SPI", por aquí se envían todas las configuraciones de las direcciones vistas anteriormente o entrar en modos de lectura/escritura.

Posteriormente se reciben los datos, cuando el convertidor tiene un señal digitalizada, en el puerto DRDY cambia de estado lógico de alto a bajo, es decir, el registro "Sync" modula la señal DRDY para detectar el cambio de estado lógico y generar un pulso que pueda inicializar la recepción de datos (pin Dout, paquetes de 216 bits). El módulo "Recepción SPI" adquiere los datos en paquetes de 8 bits por lo que tiene que recibir 27 paquetes de 8 bits, donde contiene la información de todos los canales. Al acumular los paquetes en el bloque concatenación son enviados a otra máquina de estados, que se encarga de almacenar los datos en la memoria dinámica de la tarjeta UPDSH, dicha máquina de estados se observara con detalle más adelante.

La señal DRDY se monitorea para evaluar la velocidad de muestreo, ya que indica que el convertidor tiene una nueva conversión preparada, puede ofrecer el tiempo que hace la conversión de datos, este dato se genera con un bloque contador

DRDY con una resolución de 24 bits. El valor DataRate corrobora que nuestro sistema esté trabajando a la velocidad de muestreo configurada.

El bloque de "Contador Muestras" genera el conteo de la muestra que está siendo adquirida, lo que puede ser nombrado como dirección de datos, con ello se comprueba que los datos están siendo adquiridos continuamente y sin perder alguna conversión digital.

La interfaz SPI se genera por medio de bloques controlados por FSM, los cuales a su vez contienen módulos internos. Por ejemplo "Envío SPI" contiene una máquina de estados que registra el flujo de datos, un comparador para evaluar la cantidad de bits que se envían por el puerto Din y un bloque de transmisión paralelo-serial, esto porque las configuraciones se almacenan en registros de 8 bits y tienen que ser enviados bit a bit. El bloque "Recepción SPI" contiene módulos similares a "Envío SPI", con la diferencia de que recibe los datos bit a bit y se tienen que almacenar en variables de 8 bits, es decir, una transmisión serial-paralela.

Cuando se adquieren las muestras solicitadas se van almacenando en la memoria dinámica del sistema UPDSH, por lo que se controla con otra máquina de estados finitos, la cual se dedica a almacenar datos en memoria y al obtener todas las muestras solicitadas, emplea la interfaz USB para enviar los datos a una computadora receptora del registro ECG.

La figura 3-35 muestra la máquina de estados finitos para controlar el almacenamiento y envío de datos a computadora. Se encarga principalmente de usar bloques creados por el Grupo de Mecatrónica de la UAQ. Los bloques desarrollados por el grupo son dos: control-almacenamiento de la memoria dinámica y el protocolo de comunicación USB versión 5.0.

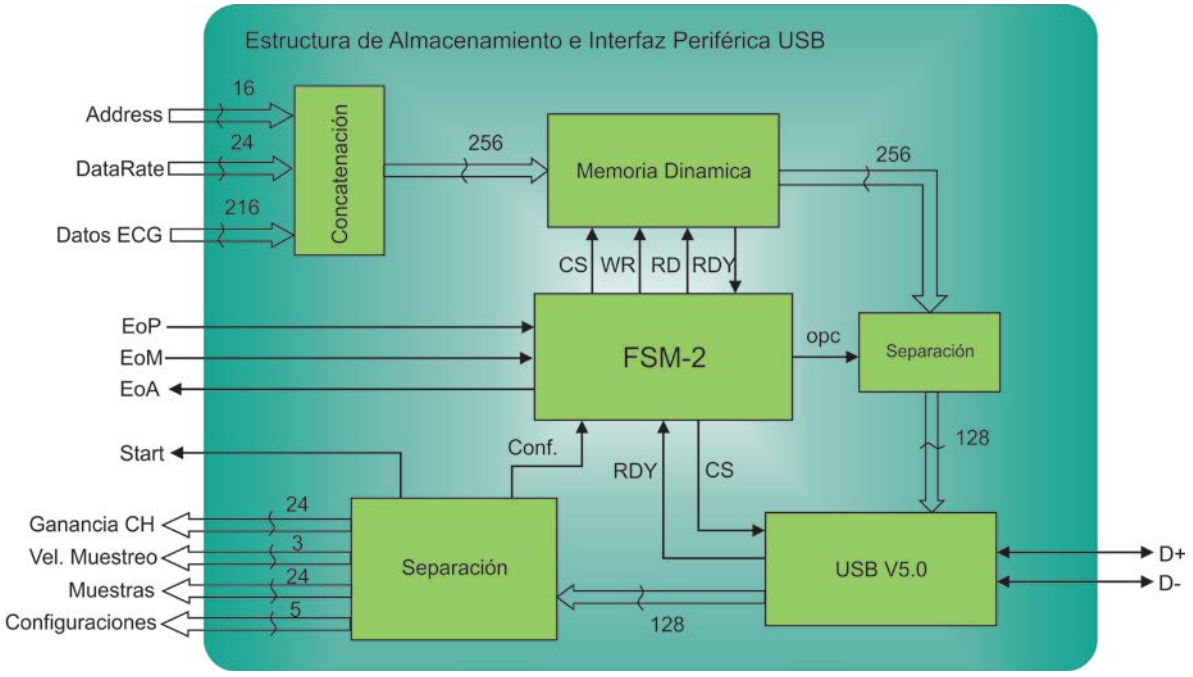


Figura 3-35.- Diagrama a bloques de estructura de almacenamiento e interfaz periférica USB

La máquina de estados finitos FSM-2 muestra el control de almacenamiento de datos, permite obtener configuraciones desde el puerto periférico USB y enviar paquetes de datos mediante este protocolo. El bloque de memoria dinámica permite guardar datos del ancho de bus que se requiera, ya que es configurable la cantidad de bits almacenados en cada registro. Se escogió este tipo de memoria debido a que la capacidad de almacenamiento es mayor en comparación con una memoria estática.

El estado inicial de la máquina de estados finitos es recibir datos o configuraciones a través del puerto USB, donde la tarjeta espera a que reciba los datos o inicio (start) del sistema, ya que implementando esta estructura se puede manipular el sistema de adquisición de señales biométricas por medio de software en el computador.

Cuando se ha iniciado el sistema se desarrolla la estructura de adquisición de señales biométricas de FSM (figura 3-34), mientras que FSM-2 captura y guarda todas las muestras solicitadas, empleando el módulo de Memoria Dinámica.

Al generar el tiempo de grabación requerido; FSM-2 hace que los datos guardados sean enviados por el puerto USB con transmisión a la computadora personal donde se adquieren los datos mediante una interfaz gráfica; que puede interpretar los datos y mostrar una gráfica digital de la señal adquirida. El protocolo de USB se representa con el bloque "USB V5.0", el cual enlaza la tarjeta UPDSH con la computadora.

El bloque de concatenación es usado para colocar el ancho de bus que se requiere al recibir los datos en la memoria dinámica, 256 bits los cuales se configuran en hardware. Cuando se envían los paquetes de información por el USB es necesario dividir los datos (bloque de separación), para que concuerde el ancho de bus requerido por el protocolo USB V5.0, él envió de datos es de 128 bits, es decir, se envía el paquete de información de la memoria dinámica de 256 bits en dos de 128 bits dirigidos a la PC.

Por último el modulo que recibe los datos de USB "separación 2" distribuye la información de entrada como: velocidad de muestreo, ganancia de canales, muestras a grabar y configuraciones. Por medio del esquema de interfaz periférica se logra el almacenamiento y comunicación del dispositivo con el computador.

Aplicando los esquemas principales del driver ads1298 e interfaz periférica se complementa la programación en hardware para implementar el sistema de adquisición de señales biométricas, dicha tarjeta fue explicada en el sub-tema anterior.

3.6 Implementación de Interfaz de Usuario

Los sistemas de implementación en señales biométricas requieren de visualización gráfica para complementar y observar el comportamiento de las señales procesadas, debido a que es de importancia obtener datos y graficar para observar la actividad eléctrica del corazón, gráficas que brinden apoyo visual al sistema desarrollado en este proyecto.

La interfaz de usuario gráfica se diseña en una plataforma de programación C++, donde se crean botones, cajas de texto, espacio gráfico, labels, etc., para crear la visualización de los datos obtenidos por el sistema de adquisición de señales biométricas. En la figura 3-36 hace referencia a la interfaz gráfica creada para la visualización del Electrocardiograma.

Se emplean comandos de creación de mallas para distribuir las secciones que se observan en la figura 3-36; Archivo, Configuración, Canales, Gráfica y Datos.

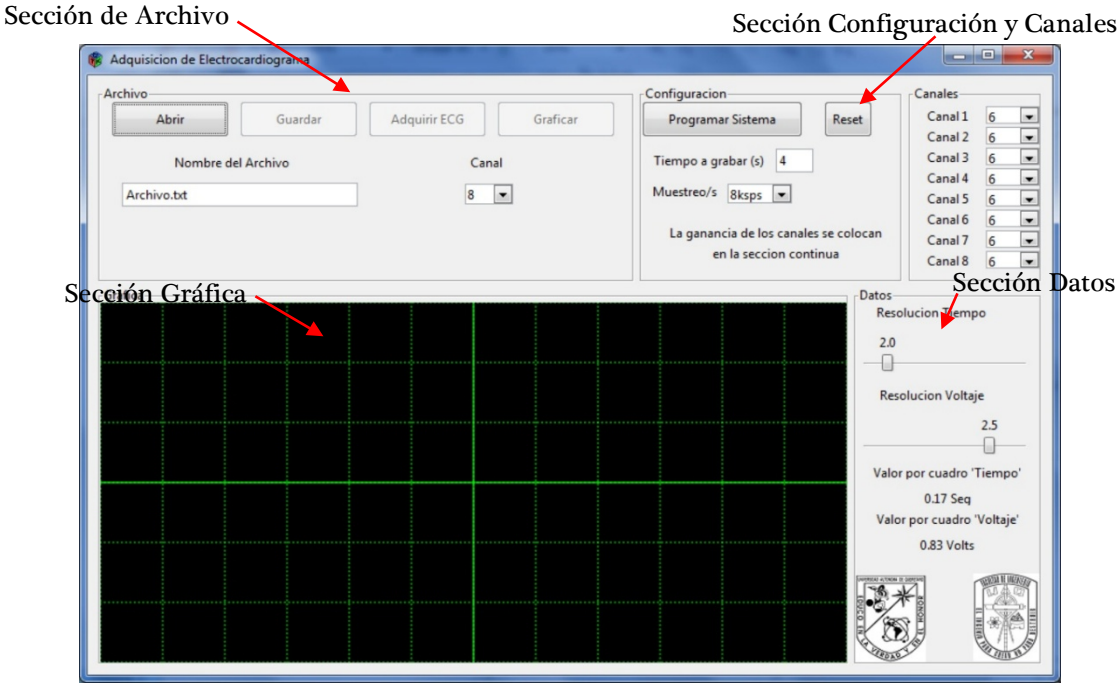


Figura 3-36.- Interfaz gráfica de adquisición de Electrocardiograma

Al iniciar el programa verifica que el sistema está conectado a la computadora, ya que el sistema de adquisición de señales biométricas se configura con el driver USB para identificar los valores de VID y PID. Dichos valores de VID y PID deben corresponder en el software del programa como en el sistema en hardware, los valores son USB_VID 0x2329 y USB_PID 0x1EF9.

Si el programa identifica el sistema envía una ventana emergente como en la figura 3-37a, por el contrario si no está conectado el sistema mediante el USB emerge una ventana con el mensaje de la figura 3-37b.

Figura 3-37.- Mensajes emergentes. Izquierda a, Derecha b

Cuando se accede a la interfaz gráfica, las secciones vistas en la figura 3-31 contienen diferentes operaciones para abrir o guardar archivos, enviar configuraciones al sistema de adquisición de señales, entre otras por lo que se explicara cada sección para dar a conocer la funcionalidad de la interfaz de usuario.

Sección Archivo (Figura 3-38).- contiene 4 botones los cuales tienen las funciones: el botón Abrir tiene la característica de abrir el archivo que este escrito en la caja de texto, al hacer click sobre el botón envía un mensaje que indica si es válido y existe el nombre del archivo. Por lo que está listo para ser graficado.

El botón Guardar, guarda el archivo con el nombre escrito en la caja de texto, que se genera al adquirir datos mediante el sistema de adquisición de señales biométricas. El cual contiene la información de los 8 canales y velocidad de muestreo

a la cual está trabajando, así como guarda la ganancia configurada de cada canal. Se habilita después de que el sistema haya hecho una adquisición de datos.

Botón Adquirir ECG, se encuentra inicialmente deshabilitado debido a que se debe configurar el sistema para poder comenzar la adquisición de datos. Al configurar el sistema se puede dar click sobre él, donde se inicia la adquisición de datos del sistema, es decir, envía el bit de "start" al sistema para iniciar la adquisición y digitalización de los datos obtenidos del acondicionamiento de la señal ECG. Cuando comienza a adquirir los datos, la interfaz interpreta, cuantifica la escala de los datos para graficarlos en la sección Gráfica, la cual se describirá en su sección, se gráfica el canal que este seleccionado en la caja de texto combinado. En el instante que gráfica la señal obtenida, los datos se almacenan en un archivo secundario que sirve de respaldo hasta el momento en que se guarda el archivo completo con otro nombre.

Para accionar el botón Graficar, se debe haber abierto un archivo para que se habilite el botón. Cuando se inicia al dar click se lee el archivo seleccionado y gráfica conforme va leyendo la información. Gráfica el canal almacenado indicado en la caja de texto combinado.

Esta sección está relacionada con el manejo de archivos, información almacenada y por almacenar. Así como la visualización de datos gráficamente.

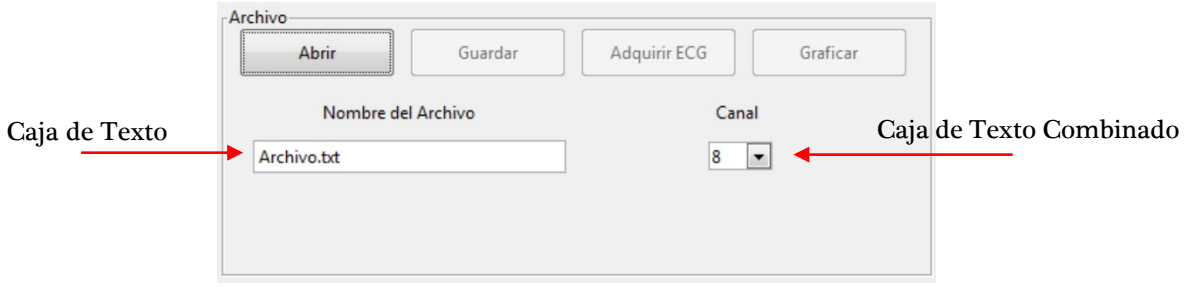


Figura 3-38.- Sección de Archivo

Sección Configuración y Canales (Figura 3-39).- para adquirir datos se requiere configurar el dispositivo, por lo que esta sección está destinada a seleccionar las configuraciones que el usuario manipula para adquirir la señal ECG, ya que depende de la ganancia correcta para evitar la saturación del dispositivo, así como periodos de muestreo y tiempo de grabación que se adapte al individuo de prueba. Las pruebas pueden ser en humanos o ratas por lo que la ganancia se considera un factor relevante que ayuda a obtener una mejor resolución de las señales obtenidas.

Principalmente existe una configuración por default, donde las ganancias de los canales son de 6, pero puede cambiarse a 1, 2, 3, 4, 6, 8 y 12, debido a que son las ganancias del convertidor ADS1298; solo que se configuran mediante software. El tiempo de grabación seleccionado es de 4 segundos y la máxima grabación es de x segundos a una velocidad de 8 mil muestras por segundo (ksps).

El botón Programar Sistema envía las configuraciones seleccionadas mediante el puerto de acceso universal USB, se envían en paquetes de 256 bits, donde se codifican en la tarjeta basada en FPGA mediante el módulo de recepción y transmisión de datos (Figura 3-35). El botón Reset se ejecuta para el convertidor ADS1298 por lo que reestablece los valores por default y en la interfaz de usuario formatea la plantilla de gráfica.

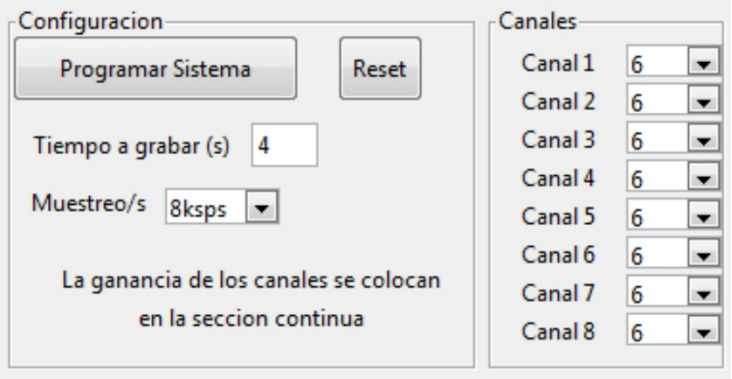


Figura 3-39.- Sección de configuraciones

Sección de Gráfica (Figura 3-40).- la visualización de la señal adquirida nos muestra el comportamiento eléctrico del corazón, donde se emplea una zona de dibujo la cual es de color negro, tiene dimensiones en x = 700 pixeles y en y = 285 pixeles.

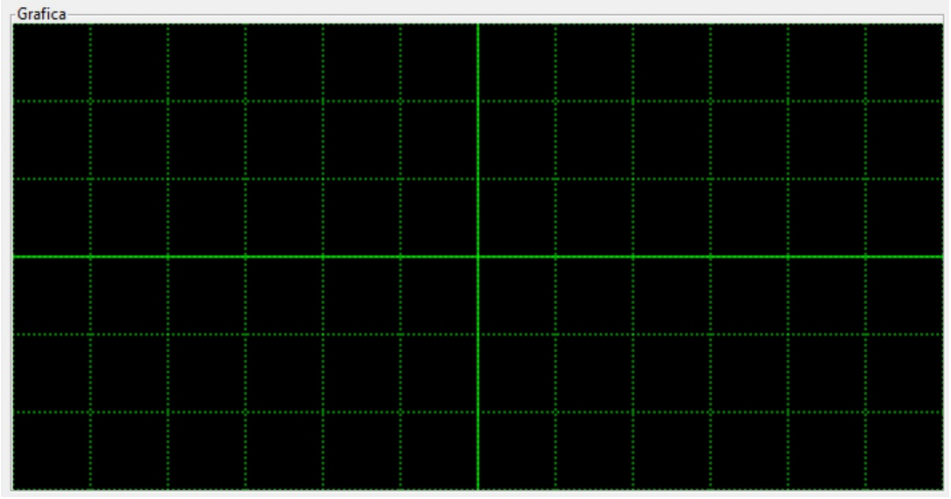


Figura 3-40.- Sección de Gráfica

Sección Datos (Figura 3-41).- se muestra la resolución que puede ser seleccionada para el tiempo y voltaje, el cual genera la resolución por cuadro de la sección gráfica.

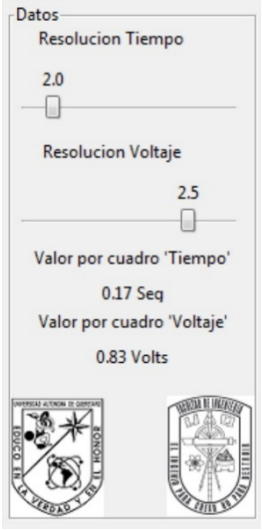


Figura 3-41.- Sección Datos

3.7 Implementación del Sistema de Adquisición de Señal Electrocardiograma.

Al conjuntar las etapas de acondicionamiento de la señal ECG (las cuales son preamplificacion y filtrado) con la tarjeta de digitalización de datos, así como conectar el sistema a una computadora mediante el bus de serie universal (USB), se puede adquirir la señal ECG mediante los electrodos de plata, al colocarlos en los puntos específicos vistos a lo largo de la metodología. En la figura 3-42 se puede ver todo el sistema conectado, donde se puede observar las tarjetas, baterías y electrodos.

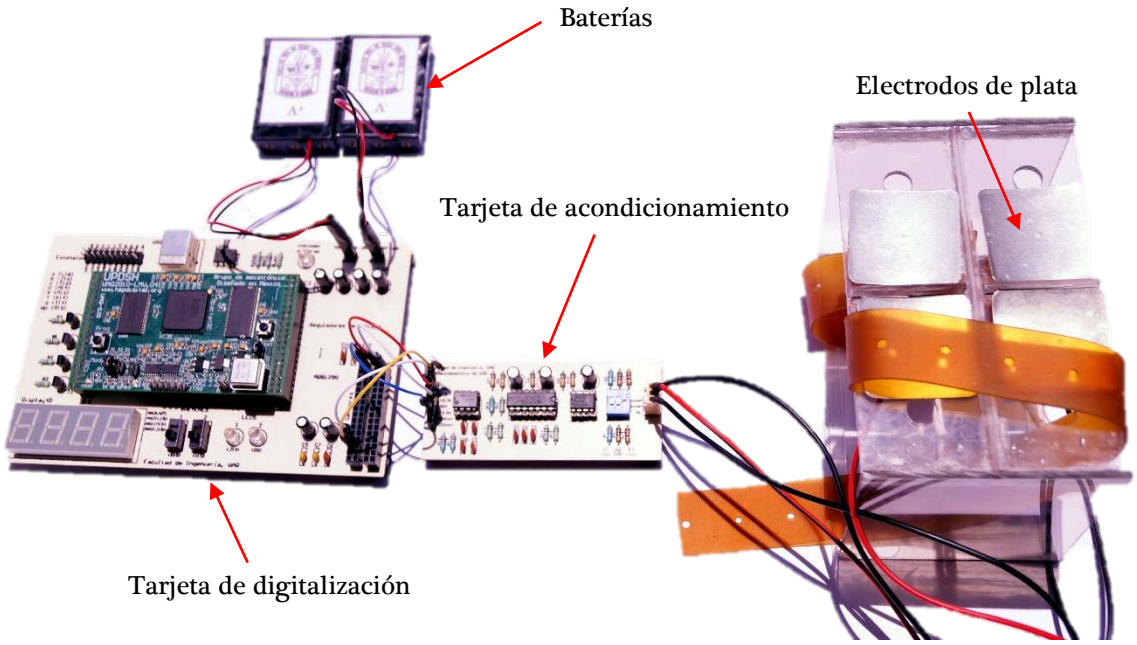


Figura 3-42.- Sistema de adquisición de señal ECG

Los electrodos de plata adquieren la señal Electrocardiograma, se emplean para pruebas de rendimiento en humanos, aunque se manejan electrodos de parche para comparar las adquisiciones de los datos y en ratas se emplean electrodos de aguja, se conecta con cables a la tarjeta de acondicionamiento la cual contiene los filtros descritos anteriormente, la señal ECG es transmitida por las tarjetas de acondicionamiento y digitalización mediante cables, los cuales alimentan la tarjeta de acondicionamiento. La alimentación de +/- 2.5 V es generada por las baterías de

litio, dichas baterías están conectadas a la tarjeta de digitalización para que esta zona contenga el menor ruido posible.

La tarjeta de digitalización es conectada a una computadora personal donde en el subtema anterior queda descrito el software, el cual reconoce el sistema cuando es conectado. Para checar las conexiones correspondientes de los pines así como las alimentaciones, programación de la tarjeta basada en FPGA y programación del software es descrito en el Apéndice.

Al obtener y realizar pruebas sobre el sistema fue necesario emplear filtros notch, las gráficas de la señal ECG obtenidas se mostraran en la sección de resultados, por lo que en el presente subtema se muestra el comportamiento del filtro notch a 60 Hz, esto debido que se desea reducir al máximo el ruido en la señal ECG. Es una implementación que se agrega al sistema debido a que reducir la frecuencia de 60 Hz es la más complicada ya que está presente en el ambiente.

El filtro fue diseñado en MatLab debido a su facilidad de implementar sistemas digitales, el filtro notch fue aplicado a la frecuencia de 60 Hz y armónicos generados por esta frecuencia fundamental. En la figura 3-43 se observa la gráfica de bode del filtro, donde en las abscisas son los decibeles permitidos por el comportamiento del filtro y las ordenadas son las frecuencias normalizadas debido a la velocidad de muestreo de la señal grabada.

El filtro tiene una curva muy inclinada por lo que esto genera una disminución de magnitud, esto hace que se deba calcular un porcentaje para obtener valores reales de amplitud en las señales ECG. Este filtro digital se aplica especialmente a señales obtenidas por ratas ya que las señales de los animales contienen mayor ruido y controlar al individuo presenta una mayor dificultad.

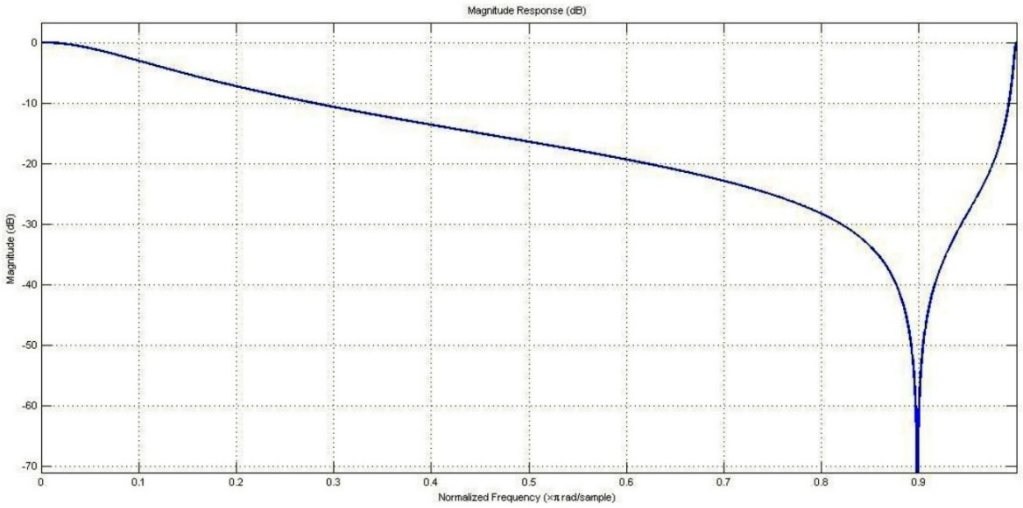


Figura 3-43.- Grafica de Bode de Filtro Notch

El filtro se comportará distinto ya que dependerá de la velocidad de muestreo para conocer las zonas de la gráfica, esto quiere decir que para cada señal obtenida con el sistema de adquisición de señales biométricas, se debe conocer la frecuencia de operación para saber aplicar el filtro.

La implementación del sistema se entiende como la unión de las tarjetas que se diseñan en este proyecto, la correcta colocación de los sensores en este caso de electrodos ECG para la mejor obtención de resultados y una visualización practica capaz de graficar los resultados. El funcionamiento de los filtros es indispensable ya que forman parte esencial del acondicionamiento de la bioseñal ECG, se elaboran y diagnostican tanto en software como en hardware debido a su importancia tan relevante, la digitalización también se diseña digitalmente y analógicamente en PCB, esto hace que sea un trabajo diseñado electrónicamente desde las raíces.

Es por esto que es una plataforma de adquisición de señales biométricas diseñada al 100% por esta institución.

4. Pruebas y Resultados

Con el fin de validar el correcto funcionamiento del sistema propuesto, se propone una serie de experimentos, las pruebas se desarrollaron tanto en humanos como roedores ya que se requiere comprobar el alcance del sistema y la capacidad a la que puede expandirse la aplicación del sistema desarrollado en esta tesis.

En esta sección, se realizan distintas adquisiciones de señales, algunas adquisiciones se obtienen de un generador de señales para comprobar la adquisición y digitalización de señales, es decir, se comprueba cada dispositivo empleado en el sistema de adquisición de señales biométricas. En las pruebas sobre sistemas biológicos, se obtuvo el Electrocardiograma en 3 distintos humanos, donde algunas pruebas se hacen mediante electrodos de parche y en otras empleando electrodos de placa de plata, a los cuales se les realizaron 5 pruebas a cada individuo para hacer una comprobación del sistema de acondicionamiento de la señal ECG, la adquisición en humanos se realizó mientras estuvieran en reposo. Para concluir que el sistema puede ser implementado en roedores, la cual es la finalidad de este proyecto, se hacen pruebas sobre una rata Wistar, en el roedor se empleó usar electrodos de aguja para su mejor adquisición, bajo la influencia de cloroformo ya que es recomendable evitar el movimiento muscular del roedor, se obtuvieron alrededor de 10 pruebas, los resultados se comparan con un sistema de adquisición comercial ya que con esto se obtiene una validación de los resultados generados. Algunas pruebas tanto en humanos como ratas se diseñaron a distintas velocidades de muestreo para saber la calidad del sistema.

Las pruebas y resultados presentados en esta sección se generan a partir de equipo del laboratorio de la institución UAQ, esto quiere decir que el proyecto es en su totalidad producido por este Centro Universitario, el cual los resultados muestra que puede ser competitivo con dispositivos de adquisición comerciales.

4.1 Análisis sobre Dispositivos Electrónicos.

Las primeras pruebas realizadas se crearon para los elementos electrónicos utilizados en este proyecto para verificar el rendimiento de los mismos. Como primer dispositivo se presenta los reguladores de voltaje, es de importancia corroborar la precisión de voltaje que suministra para evitar caídas de voltaje o mal funcionamiento de los dispositivos que se alimentan de baterías.

Baterías y Reguladores de Voltaje.- Se obtuvieron adquisiciones de voltaje de DC con el convertidor ADS1298, pero este último fue alimentado por voltaje positivo con una fuente de voltaje DC a +2.5V y para voltaje negativo con las baterías que pasan por el regulador de voltaje negativo (-2.5V).

Con las condiciones de alimentación mencionadas se realizaron adquisiciones de voltaje en DC de una misma fuente de poder, esto para comprobar que se requiere de una precisión de voltaje alta y limpieza al ser energizado el convertidor ADS1298, con el fin de adquirir mejores señales y con el menor ruido ambiental posible.

En la figura 4-1 se puede observar como del lado izquierdo se adquirió una señal DC positiva la cual tiene varios picos e invariaciones de voltaje y por otro lado de lado derecho se cuenta con una adquisición de voltaje DC negativo pero se logra percibir que el ruido es menor. Esto concluye que la alimentación del convertidor influye a obtener mejores digitalizaciones del sistema, por lo que queda comprobado

que el alimentar con baterías y emplear los reguladores de voltaje LT1963 y LT3015 ayudan a reducir el ruido ambiental.

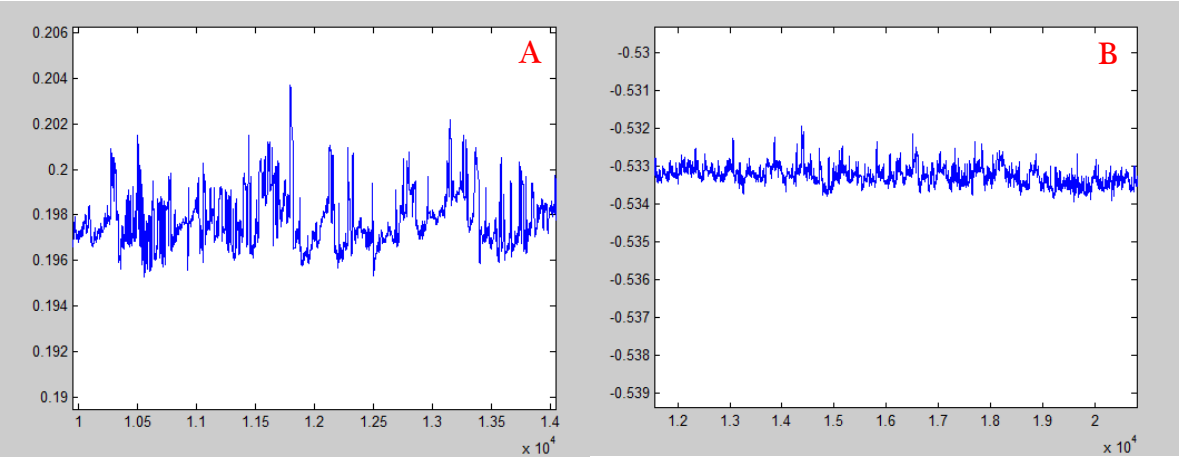


Figura 4-1.- Adquisición de voltaje DC energizado (a) sin baterías y (b) con baterías

Por otra parte se considera que a lo largo del acondicionamiento de señales biométricas el ruido es un mal que estará presente en todas las adquisiciones de señales pero con el tratado correcto y considerando detalles como la alimentación de los dispositivos, se puede lograr una mejor resolución de las señales evitando que pueda ser propagado el ruido.

Preamplificación y Filtros analógicos.- Estas etapas son esenciales por lo que se requiere una alta precisión de los elementos y diseño, como se conoce la preamplificación se logra con un amplificador de instrumentación y los filtros empleados son filtro butterworth con frecuencia de corte superior de 150 Hz y frecuencia de corte inferior de 0.05 Hz, colocando en cascada un filtro notch a 60 Hz. Para comprobar el rendimiento del acondicionamiento analógico de la señal ECG se tienen análisis mostrados en osciloscopios para lograr obtener evidencia de la capacidad del sistema analógico.

En la figura 4-2 se muestra una señal sinusoidal que pasa por los filtros donde se puede apreciar que en la zona de ganancia 0 (figura 3-14), es decir, en frecuencias por debajo de los 60 Hz la señal de salida debería ser la misma que la de entrada, por lo que se observa la señal tiene una frecuencia de 37 Hz. La señal 1 es la entrada y la señal 2 es la salida, se muestra que tiene una reducción de amplitud y un desfasamiento lo cual se verá reflejado cuando se obtengan las señales ECG. Pero se respeta el diagrama de bode (figura 3-14), al saber que es analógico es difícil lograr un comportamiento ideal por lo que se considera como buen desempeño del filtro al permitir las señales dentro del rango de frecuencias de corte.

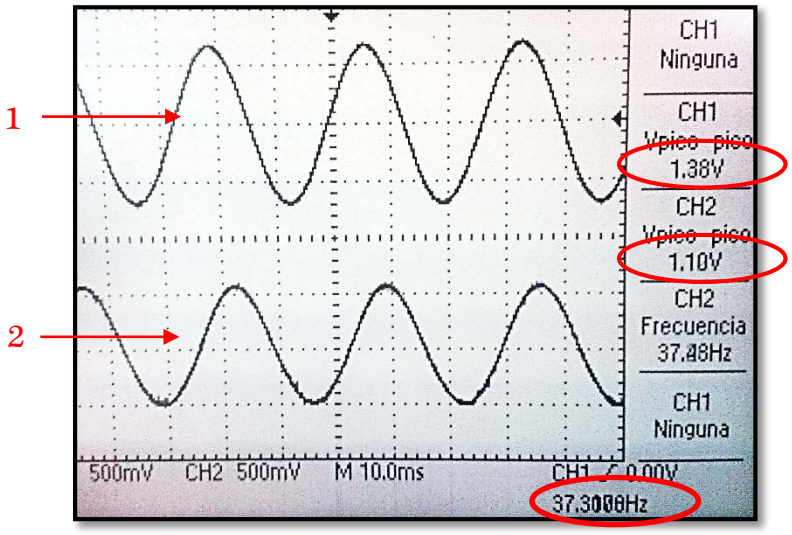


Figura 4-2.- Señal sinusoidal 37 Hz antes y después del filtro butterworth

Para comprobar que el filtro elimina las frecuencias por arriba de 150 Hz se observa la figura 4-3, donde se señala la frecuencia de la señal sinusoidal así como la entrada y salida, 1 y 2 respectivamente. La frecuencia de la señal de entrada es de 374 Hz. La imagen refleja que al generar una señal a una frecuencia mayor a la permitida por el filtro, este la rechaza por lo que disminuye su amplitud tratando de eliminar esta señal.

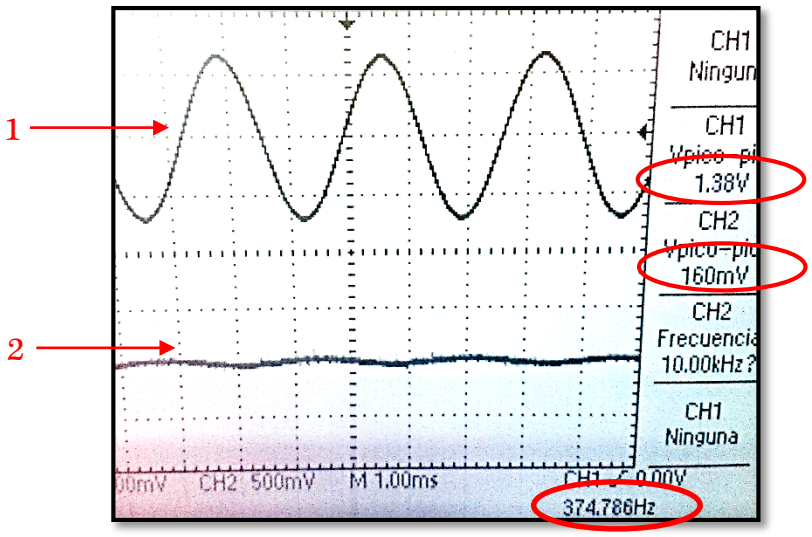


Figura 4-3.- Señal sinusoidal 374 Hz antes y después del filtro butterworth

Al conocer el funcionamiento del filtro se realizaron pruebas de visualización del Electrocardiograma en humanos para comprobar que la señal se acondiciona correctamente empleando la etapa analógica (preamplificación y filtro).

La figura 4-4 muestra la señal ECG obtenida de un humano, en donde se puede observar que la señal 1 es la entrada y solo ha sido preamplificada, por lo que se recuerda que las señales ECG de un humano son alrededor de 1 mV es por eso es que antes de ingresar la señal al filtro es amplificada, la señal 2 es después de haber pasado los filtros, tanto el butterworth y el notch.

En la figura de la señal Electrocardiograma se denotan las amplitudes y frecuencia de la señal para un mejor entendimiento.

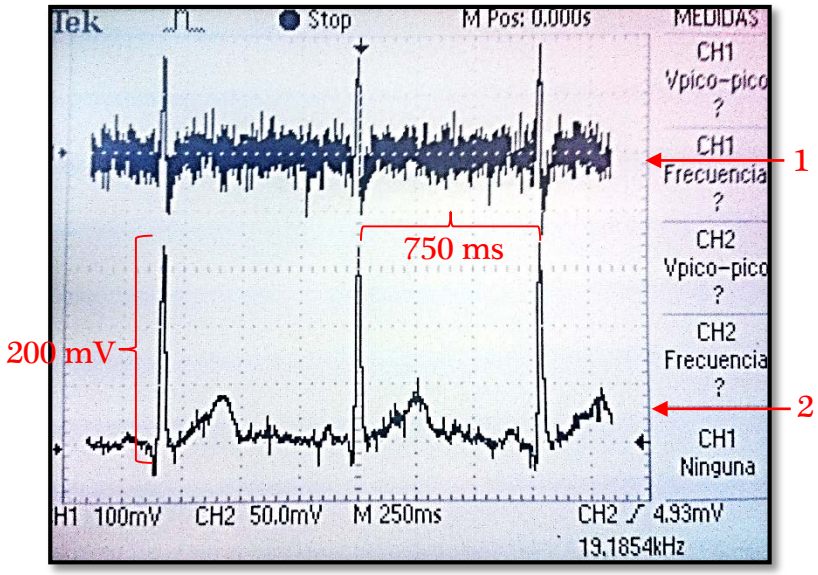


Figura 4-4.- Señal ECG de Humano antes y después de filtros.

Como se observa en la figura 4-4 la señal ECG es filtrada al obtenerla con los electrodos, esta derivación es la I por lo que se colocó un electrodo en el pecho lado derecho el cual es un polo negativo, el otro electrodo se colocó en el pecho lado izquierda el cual es un polo positivo, la existencia de un tercer electrodo evita que se tenga más ruido del que se puede apreciar en la ilustración, es por tal motivo que el tercer electrodo se coloca en el pie derecho.

Se pueden calcular los datos de amplitud y frecuencia, por una parte la amplitud mostrada en la figura es de 200 mV pero está sometida a una preamplificación de 100 es por esto que la amplitud real del ECG es de 2 mV. La frecuencia cardíaca de la señal es el inverso de 750 ms es igual a 1.3 Hz, lo que indica que en un minuto se tienen 80 lpm (latidos por minuto). Los datos están dentro de rangos de trabajo normal de un corazón.

Con estas imágenes queda confirmado el rendimiento del acondicionamiento de la señal ECG, para pruebas en ratas se realizaran al final de la sección para obtener pruebas finales ya que ese es el objetivo del proyecto.

Digitalización.- Al generar la señal ECG se debe digitalizar para poder realizar mayor procesamiento, es por esto que las señales ECG obtenidas anteriormente se almacenaran en una computadora. Los resultados almacenados se pueden observar en las siguientes figuras. Se recuerda que el análisis sobre el dispositivo es el convertidor ADS1298, el encargado de convertir los datos analógicos a digitales.

La figura 4-5 muestra la obtención de una señal sinusoidal con frecuencia de 400 Hz, se eligió esta frecuencia ya que la señal ECG está muy por debajo de esta frecuencia, es por esto que se decide comprobar el rendimiento del convertidor a una frecuencia mayor. La adquisición de datos se hizo a una velocidad de 8 ksps (mil muestras por segundo). La amplitud de la señal fue tomada dentro del rango de operación de +/- 2.5V, amplitud de 0.3 V.

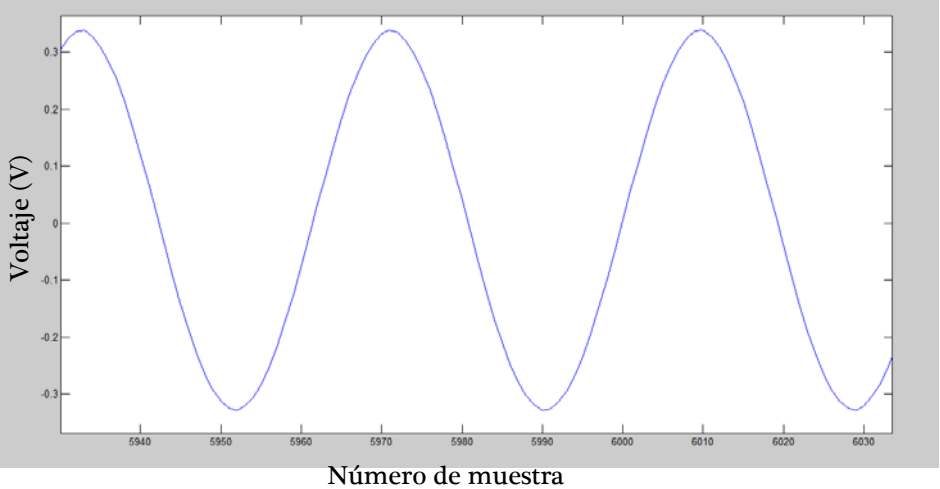


Figura 4-5.- Señal sinusoidal de 400 Hz digitalizada con el convertidor ADS1298

Una prueba más fue tomando la señal ECG tomada de un electrodo de parche de un humano y digitalizarla mediante el convertidor ADS1298. En la figura 4-6 se visualiza la señal digitalizada ECG a una velocidad de 8ksps.

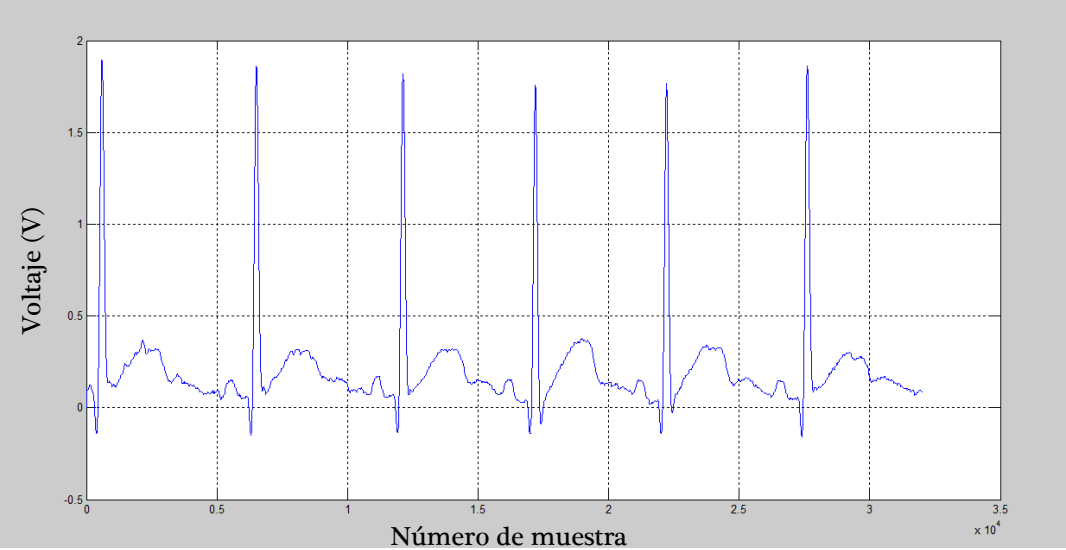


Figura 4-6.- Señal ECG de Humano digitalizada a 8ksps.

La derivación I es la señal ECG que se observa en la figura 4-6, como se puede observar es graficada en el software MatLab, al contener los datos en un archivo. La señal fue amplificada por 8 una vez más por el convertidor, recordando que tiene un selector de ganancias. Ya que la señal viene amplificada por 100 se debe calcular que la ganancia total es de 800, haciendo los cálculos se tiene una amplitud real de 2.2 mV. Para calcular la frecuencia cardíaca se toman los picos de la señal denominada R-R, donde se tienen 6000 muestras aproximadamente, esto quiere decir que si la muestra se grabó a 8 ksps, la frecuencia cardíaca es de 0.75 Hz (45 lpm).

Se enuncia que la frecuencia cardíaca normal oscila entre 40 lpm a 100 lpm, esto quiere decir que nuestras tomas están en un rango correctamente calculado.

Al concluir las pruebas de digitalización se obtuvieron resultados satisfactorios ya que el convertidor genera muy buena resolución en cuanto a voltaje y tiempo, debido a que las dos gráficas dan a conocer una resolución alta.

4.2 Comprobación de Comunicación Periférica

La transmisión de datos es igual de importante que el acondicionamiento y digitalización porque podemos tomar evidencia de las señales adquiridas y enviarlas a distintos módulos. La comunicación periférica se comprende entre el almacenamiento en la memoria dinámica de la tarjeta UPDSH del Grupo de Mecatrónica de la UAQ y el envío de datos por medio del puerto universal serial bus (USB), sin olvidar el protocolo SPI para controlar el convertidor ADS1298.

Al obtener los datos y guardar el archivo se puede ver su contenido, el cual en la figura 4-7 puede observarse. La primera columna indica la velocidad de muestreo a la que se configuro el sistema (comprobación del Protocolo SPI), en este caso se usa la velocidad de 8 ksps, la segunda columna es un registro de direcciones en donde fue almacenada la información en la memoria dinámica (comprobación del almacenamiento en memoria dinámica), la tercera columna indica un registro estable el cual debe ser siempre el mismo para indicar que la transmisión de datos vía USB ha sido satisfactoria (comprobación de la transmisión de datos por USB).

La comprobación de buen funcionamiento es sencillo pero esencial ya que si se pierden datos en la transmisión no se podrán tomar los datos válidos.

	1	2	3	4	5	6	7	8	9
1	33869	0	12582917	16775356	16387946	16775126	16775586	16775836	16775
2	5983	1	12582917	16776126	16390086	16774936	16775616	16775766	16775
3	5982	2	12582917	389640	16336766	16669406	980	16731686	16773
4	5983	3	12582917	1377310	16151516	16296566	4850	16598016	16768
5	5983	4	12582917	1467240	16045056	16183396	7020	16573186	16767
6	5983	5	12582917	1398920	15990756	16158596	7350	16573506	16766
7	5983	6	12582917	1339670	15955536	16143626	7480	16573676	16766
8	5983	7	12582917	1286200	15932396	16133886	7450	16573676	16765
9	5984	8	12582917	1238290	15915736	16127406	7240	16573686	16765
10	5983	9	12582917	1194420	15904166	16123096	7340	16573596	16764

Figura 4-7.- Contenido de Archivo al adquirir datos

Comprobación del Protocolo SPI.- La configuración y preparación del dispositivo ADS1298 es esencial para convertir los datos adecuadamente, la forma en que corroboramos la funcionalidad del sistema es que el dispositivo entregue la velocidad de muestreo y sea registrado para su visualización en software.

El número entregado en estas pruebas fue alrededor de 5980 (primera columna del archivo), este conteo indica el tiempo en el que tarda en convertir un dato analógico a digital, ya que se usó un oscilador de 48 MHz se realiza la conversión para saber exactamente el tiempo entregado.

$$Velocidad\ de\ Muestreo = (5980 * \frac{1}{48000000})^{-1} = 8026\ Hz$$

Al realizar el cálculo se denota que la frecuencia de operación es la seleccionada por lo que el protocolo se ejecuta correctamente. Para evaluar los canales son de la columna 4 en adelante, ya que si se requiere observar la conversión de datos como se realizó en el subtema anterior para graficar la señal obtenida. En la siguiente figura 4-8 se grafican varias velocidades de muestreo para evaluar que mantengan la misma velocidad a lo largo de varias pruebas.

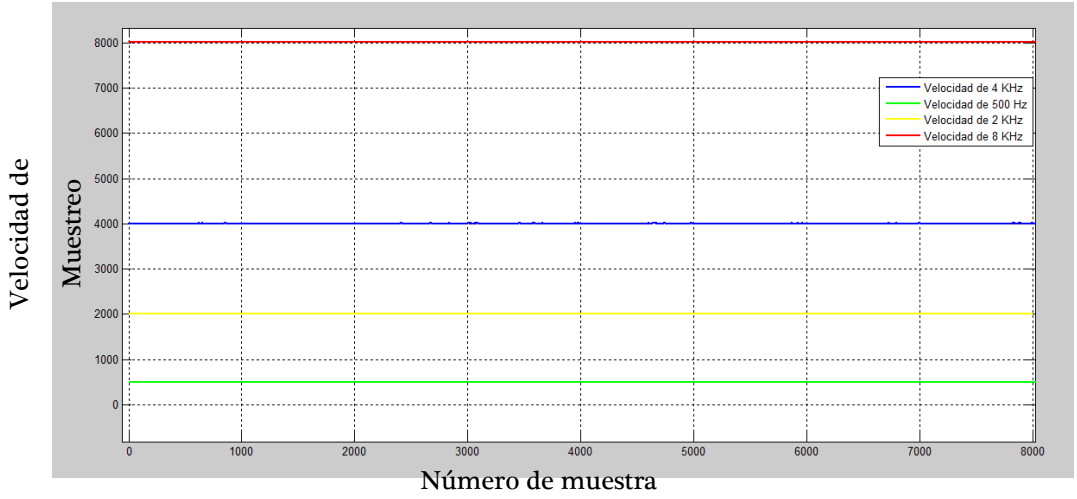


Figura 4-8.- Pruebas de rendimiento de velocidad de muestreo.

Comprobación del almacenamiento en memoria dinámica.- Se ha mencionado que la forma de comprobar que el almacenamiento en memoria dinámica sea el correcto, es por medio de observar la figura 4-7, donde se observa que para no perder datos se debe respetar una relación de la columna 2 ya que es un contador de registros al graficar la columna debe ser una recta en forma de rampa. Esto es debido a que en programación en lenguaje VHDL se crea un conteo por cada muestra adquirida, a lo cual se le llama dirección, misma que es almacenada en la memoria dinámica si al adquirir los datos en una computadora se encuentra el comportamiento de un contador ascendente, se almacenaron y transmitieron correctamente los datos.

La figura 4-9 muestra la recta rampa de datos adquiridos de la figura 4-6 (señal ECG), se puede apreciar que la recta no tiene ningún sobre salto ni alteración por lo que el almacenamiento de datos en la memoria dinámica es el ideal.

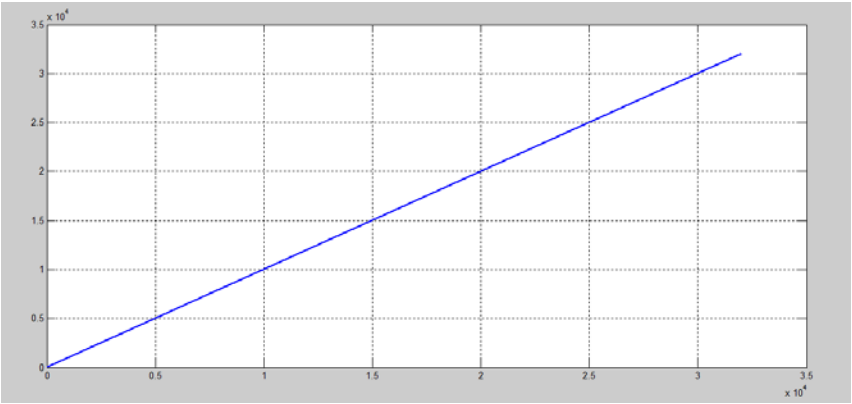


Figura 4-9.- Gráfica de direcciones al almacenar en memoria dinámica

Comprobación de transmisión de datos por USB.- La manera en que se comprobó la correcta transmisión de datos fue enviando bits de registro cada recepción de datos, es decir, cada que se reciben datos lleva un registro de 24 bits el cual los primeros 4 bits más significativos contienen la trama “1100” y los últimos 4

bits menos significativos pueden ser configurables a lo que se registró como “0101”. Como se observa en la figura 4-7 la tercera columna está el registro en decimal es “12582917” al pasarlo al sistema binario corresponde “11000000000000000000101”. Donde se puede corroborar que los bits más significativos y los menos significativos son iguales, al graficar la columna 3 se debe mantener el mismo valor para indicar que la transmisión de USB se está realizando correctamente. La figura 4-10 muestra que el valor se mantiene en la adquisición de la señal ECG mostrada anteriormente.

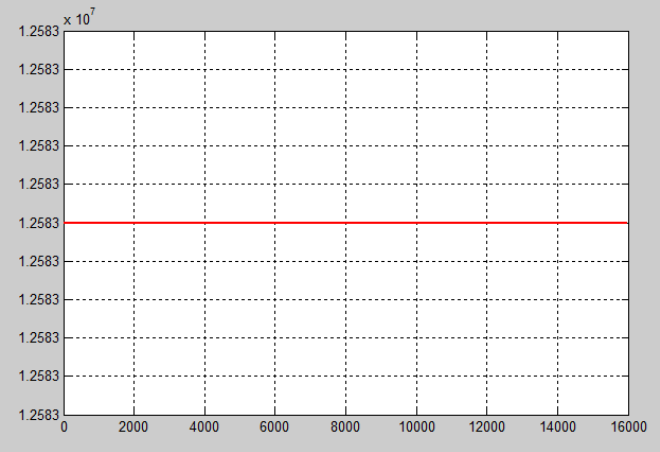


Figura 4-10.- Registro de indicación de recepción correcta.

Con las pruebas indicadas en este subtema se comprueba que la comunicación periférica funciona correctamente y los datos no se pierden en ningún momento.

4.3 Funcionalidad de Interfaz de Usuario

La creación de una interfaz es con la finalidad de graficar las señales grabadas o adquiridas en “tiempo real”. Así que se presentaran algunos resultados obtenidos de la interfaz de usuario al graficar las señales obtenidas anteriormente.

Se presenta en la figura 4-11 la interfaz de usuario creada en Gtk bajo el lenguaje de programación C++. Las funciones que se configuran para la mejora de adquisición de señales biométricas son de utilidad para calibrar mejor la resolución de las bioseñales. En la figura se gráfica una señal ya grabada la cual ya se ha mostrado para análisis en la sección pasada, en la imagen se observa que se coloca el nombre del archivo a graficar después se selecciona el canal que es de interés y se pulsa el botón graficar para iniciar la visualización. Se observa claramente la señal ECG así como las leyendas de resolución de voltaje y tiempo.

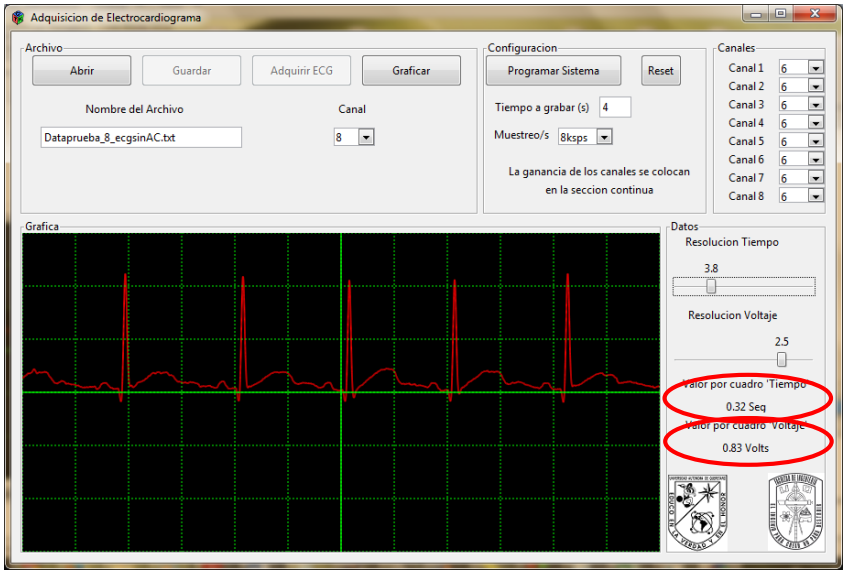


Figura 4-11.- Interfaz de usuario, graficando una señal ECG derivación I.

Para adquirir las señales lo más próximo a tiempo real se debe conectar todas las etapas de preamplifación, filtrado y digitalización, de esta forma solo se "programa el sistema" con las características que desee el usuario, continuando con "adquirir ECG", este procedimiento sirve para visualizar las señales y guardar un respaldo de lo que se está proyectando en la sección de gráfica. Los valores de la señal ECG son 90 lpm y 2 mV, calculados por la ganancia seleccionada y muestreo.

4.4 Rendimiento del Sistema de Adquisición de Señal Electrocardiograma

La ejecución de todo el sistema comprende desde la selección de la ganancia hasta la digitalización e implementación de filtros digitales, por lo que en primera instancia se mostraran algunos resultados generados en humanos ya que se validara completamente el sistema y después para cumplir con el objetivo principal se darán a conocer las pruebas al implementar el sistema en ratas.

En las siguientes figuras (figura 4-12 a 4-15) se colocan las adquisiciones en humanos de distintos individuos de prueba.

Para la figura 4-12 se realizó en un hombre de 22 años con un peso promedio de 62 kg, en estado de reposo.

Individuo 1.- La señal ECG derivación I se adquirió con bajos niveles de ruido y con una estabilidad bastante buena, donde los niveles de amplitud oscilan entre 2 mV. Fue muestreada a una velocidad de 8 ksps, bajo niveles normales sin anestesia. La adquisición fue realizada con electrodos de parche.

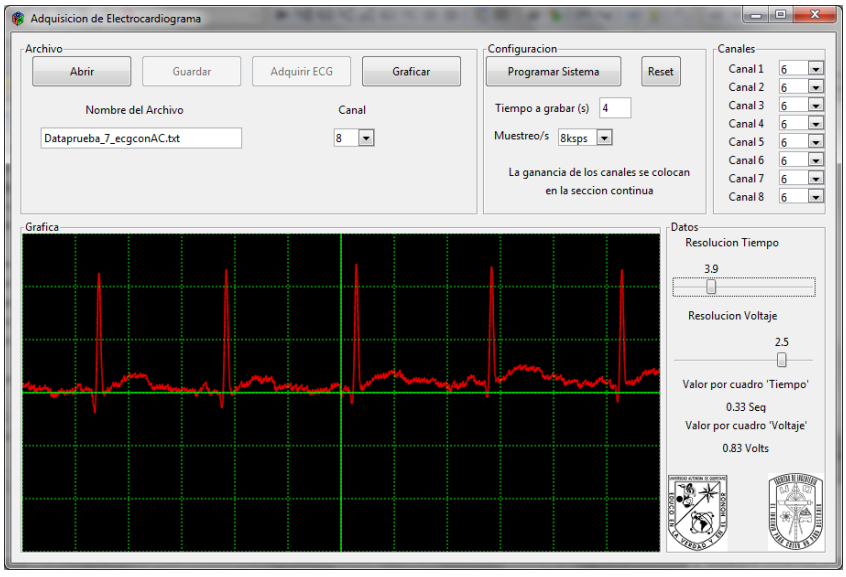


Figura 4-12.- Adquisición de señal ECG en individuo 1

Individuo 2.- La señal adquirida en la figura 4-13 derivación II, obtenida de un hombre de 24 años con un peso alrededor de 70 kg, muestra que existe una perturbación en el individuo de prueba, ya que las señales no se encuentran estables, los niveles de voltaje son de 1 mV y el individuo no fue anestesiado, sin ninguna perturbación ni problemas del corazón por lo que el sistema presento algunas variaciones. La adquisición fue realizada con electrodos de placa de plata, por lo que se puede deber la desestabilidad del sistema por el mal contacto de los electrodos sobre la piel.

En esta etapa se considera que es mejor el uso de los electrodos de parche ya que se obtiene una mejor adquisición empleando el mismo sistema y los mismos cables de transmisión de datos. Aunque se evaluó la posibilidad de que los cables implementados no realizan un buen contacto, así que se considera que tanto los cables como los electrodos deben tener un buen acoplamiento y aislados para evitar cualquier intromisión de señales o movimientos de los músculos del cuerpo humano.

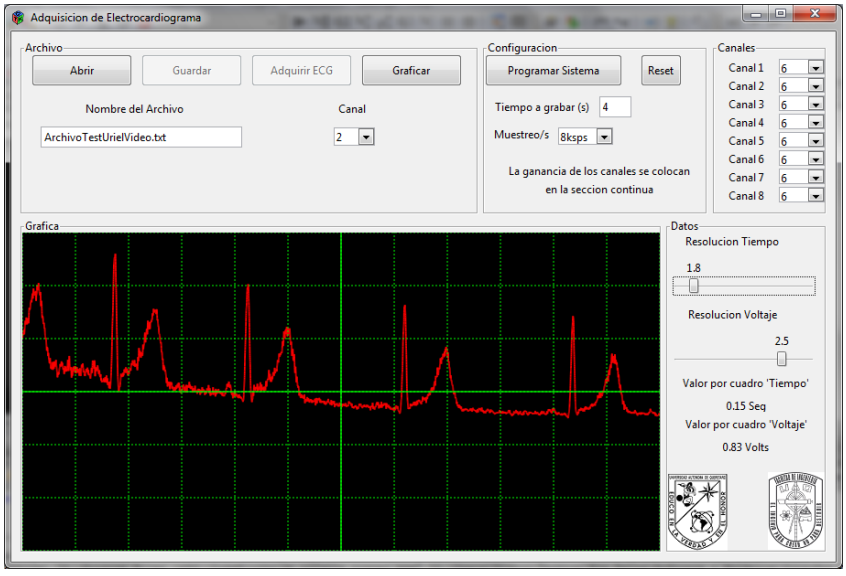


Figura 4-13.- Adquisición de señal ECG en individuo 2.

Individuo 3.- La figura 4-14 muestra como la señal ECG derivación I, se obtiene con demasiado de la frecuencia de 60 Hz, esto debido a que se adquirió la señal con la computadora conectada, es decir, debido a que el sistema no tiene un aislamiento de tierra en la zona de digitalización, se filtró la red eléctrica en el sistema por lo que produjo que la frecuencia de 60 Hz fuera introducida en la adquisición de los datos.

Cabe mencionar que las pruebas fueron realizadas en una mujer de 23 años con un peso de 67 kg y sin ningún reporte de problemas en el corazón sin ninguna anestesia aplicada y bajo ningún medicamento. Esto da a entender que los problemas de adquisición fueron por la intervención de la red eléctrica en el sistema.

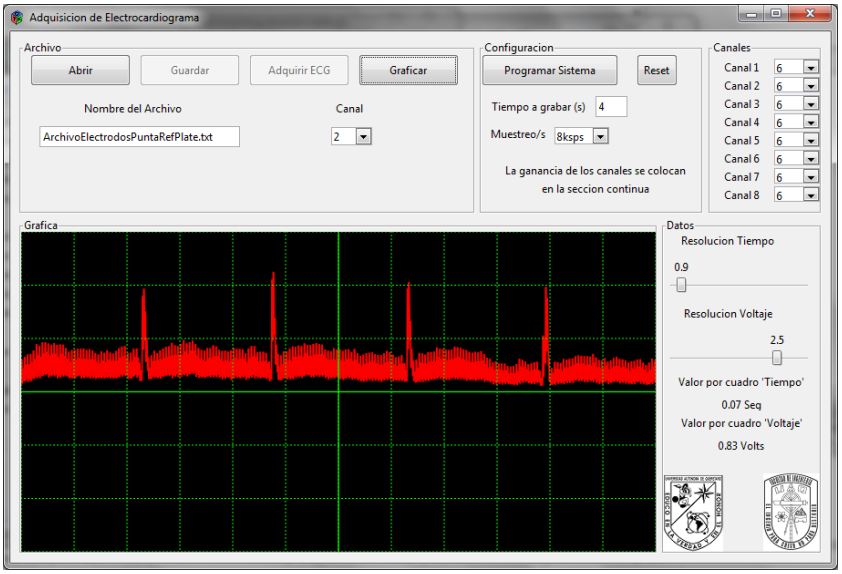


Figura 4-14.- Adquisición de señal ECG en individuo 3.

Debido a esta distorsión en la señal se realizó un análisis de Fourier para comprobar que la frecuencia presente es de 60 Hz, por lo que se muestra en la Figura 4-15.

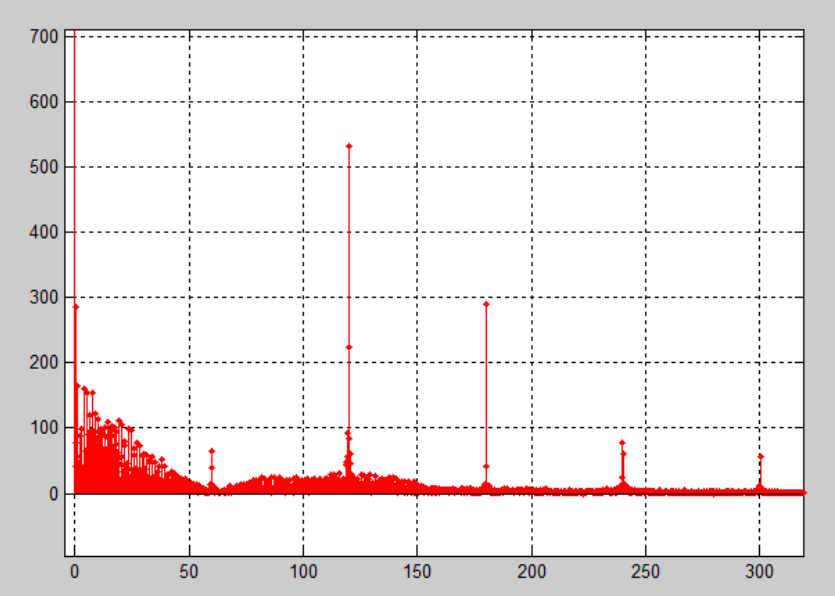


Figura 4-15.- Análisis de Fourier de la señal ECG en individuo 3

Se observa en la figura 4-15 que las frecuencias que están interviniendo y generan ruido son 60 Hz, 120 Hz, 180 Hz, 240 Hz y 300 Hz, es decir, interviene la frecuencia fundamental de 60 Hz y sus armónicos obtenidos de la red eléctrica. Se implementó el filtro digital de la figura 3-43, para comprobar que aun teniendo problemas con el sistema se puede manipular la señal ECG para limpiar el ruido introducido en la adquisición de la señal ECG. La figura 4-16 muestra la señal de la figura 4-14 con y sin ruido de la frecuencia de 60 Hz.

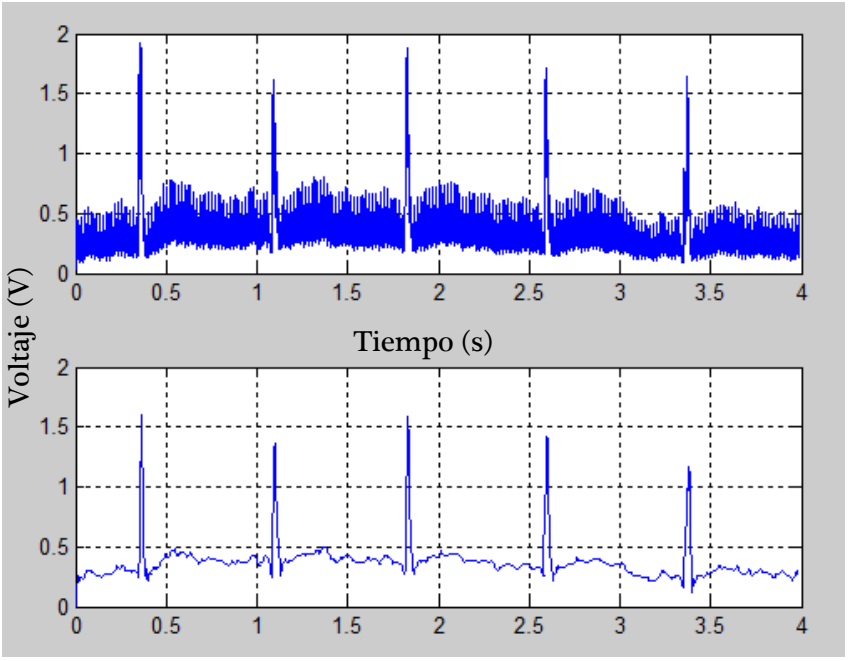


Figura 4-16.- Adquisición de señal ECG con/sin ruido de individuo 3.

Mediante estas pruebas se conoce el nivel de voltaje es alrededor de 1.5 mV que el equipo de adquisición de la señal ECG, tiene un buen rendimiento por lo que se prosigue a experimentar con las ratas.

Al adquirir las señales en roedores se tiene que implementar otro electrodo, de aguja, debido a que los electrodos de placa no tiene la suficiente conductancia para que las pequeñas señales eléctricas presentes en las patas puedan ser captadas.

Los cuidados y tratados de las ratas (ratas Wistar u otros animales) se basan en reglamentos de la Universidad de Minnesota “Research Animal Resources”. La colocación de los electrodos y anestesia del roedor fueron implementados por un especialista en experimentación con animales de la institución UAQ de la facultad de Ciencias Naturales.

Se colocan los electrodos de aguja sobre el roedor como se muestra en la figura 4-17, el electrodo 1 es la referencia negativa del diferencial eléctrico, electrodo 2 es la referencia positiva y por último el electrodo 3 es la referencia de tierra.

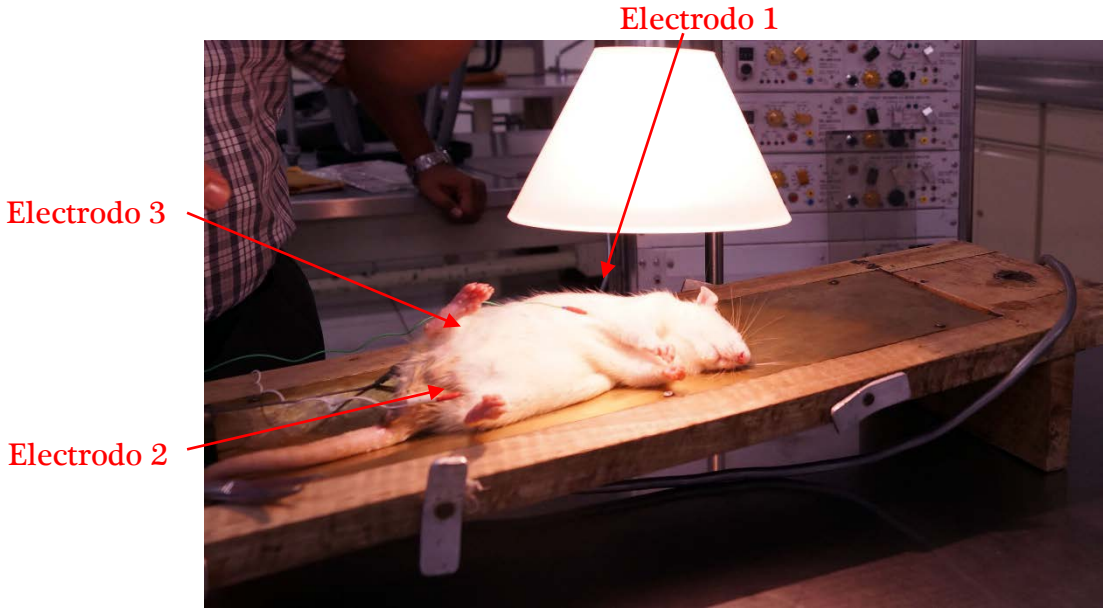


Figura 4-17.- Colocación de electrodos en rata Wistar.

La tarjeta de acondicionamiento de la señal ECG se selecciona una ganancia de 5000 (en roedores), prosigue con alimentar la tarjeta y las entradas de las señales filtradas. Luego la etapa de digitalización se configura mediante software y la inicialización de la adquisición de la señal ECG. Estas etapas ya fueron comprobadas por lo que solo es necesario conectar adecuadamente, en la sección de apéndice se muestran las conexiones para saber los pines de alimentación y las señales.

En la figura 4-18 se puede visualizar la señal ECG de la rata obtenida mediante el software de adquisición de señal Electrocardiograma. Se presentaran más pruebas generadas por el roedor examinado para determinar la coherencia entre los datos.

El individuo de experimentación es una rata de aproximadamente 500 gramos, no presenta ninguna anomalía ni enfermedad la cual pueda causar un mal funcionamiento del corazón. Por lo que debería presentar características bioeléctricas normales.

La figura 4-18 muestra la señal adquirida mediante la conexión de la computadora a la red eléctrica, esto debido a que la tierra de la red eléctrica ayudo a referenciar la señal ECG, es decir, el no colocar un aislamiento en la tarjeta de digitalización hizo que la propia referencia de la red eléctrica ayudara a visualizar la señal aunque esta conexión hace que intervenga la frecuencia de 60 Hz. Si la computadora no era conectada a la referencia la señal no podía ser visualizada, ya que es difícil colocar los electrodos correctamente en la rata, esto quiere decir que el diseño de la tarjeta permitió adquirir aunque con ruido la señal ECG.

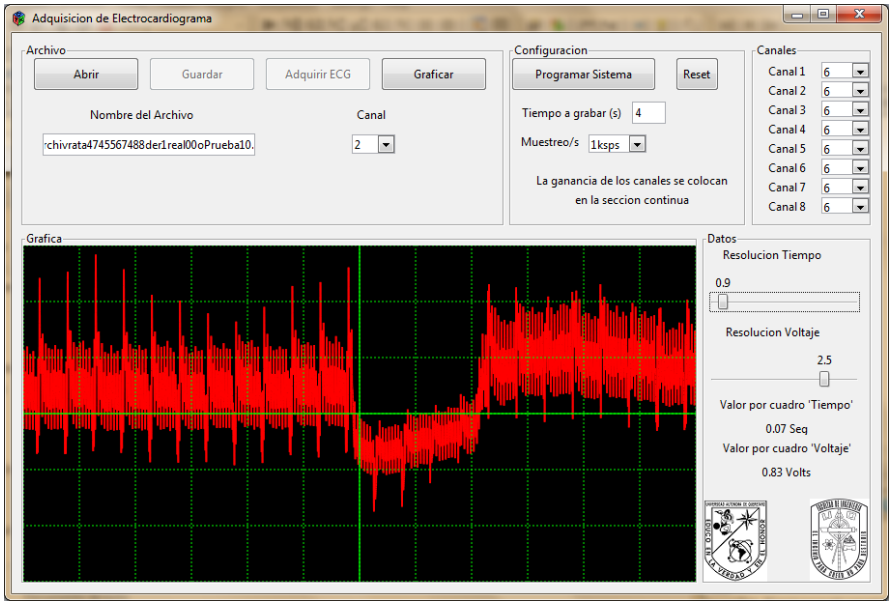


Figura 4-18.- Adquisición de señal ECG en ratas derivación I.

Como se observa en la figura 4-17 el ruido es excesivo en la señal Electrocardiograma por lo que se tiene que aplicar el filtro digital notch a 60 Hz y

sus armónicos, esto es debido a que la señal en ratas contiene demasiado ambiental. Este comportamiento se manifestó por que la rata fue iluminada con un foco y la señal de la red eléctrica se filtró entre la adquisición de las señales.

A continuación se aplicó el filtro notch en la señal ECG de la rata, la figura 4-19 muestra la señal 1 sin filtro y la señal 2 con filtro. La señal filtrada queda con buena resolución, donde se puede observar los complejos de la señal (P, Q, R, S, T).

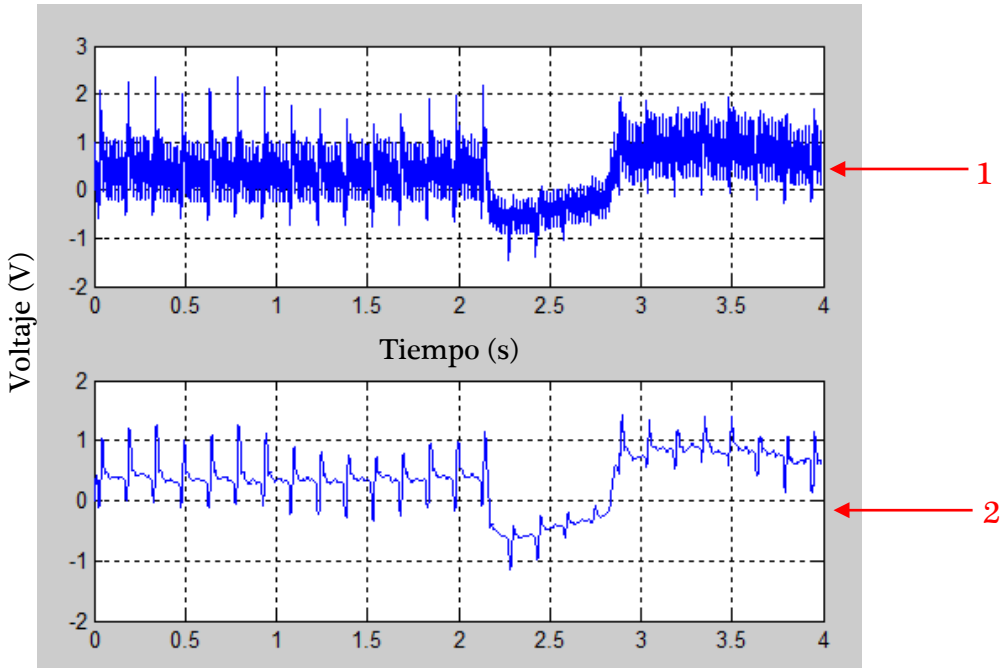


Figura 4-19.- Señal ECG de rata Wistar derivación I.

La figura 4-20 marca los complejos P, Q, R, S y T presentes en las ratas.

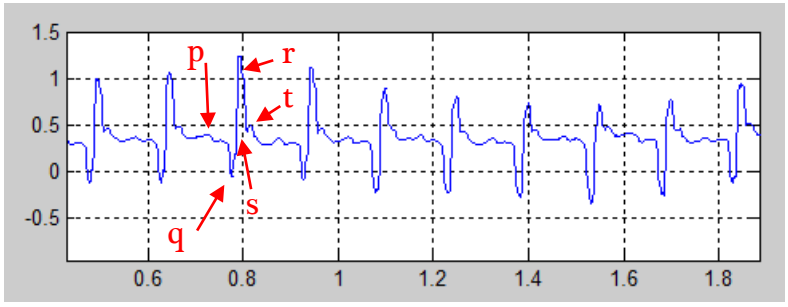


Figura 4-20.- Complejos de la señal ECG de rata Wistar

Los análisis espectrales de las señales ECG con y sin filtro se observan en la siguientes figuras, la figura 4-21 muestra la transformada rápida de Fourier de la señal 1 de la figura 4-19, este análisis es creada para verificar que las frecuencias de ruido son a 60 Hz, donde se puede observar que son de 60 Hz y armónicos.

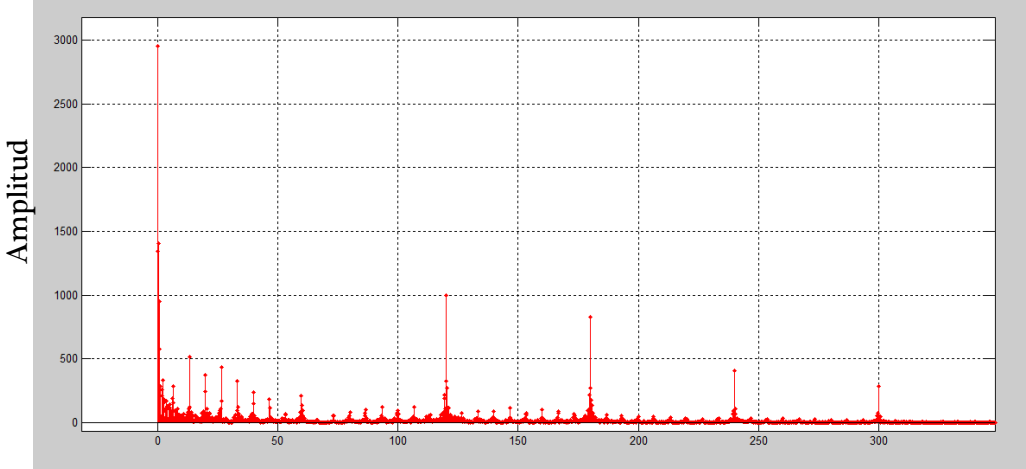


Figura 4-21.- Análisis espectral de la señal ECG sin filtro.

A continuación en la figura 4-22 se observa como el filtro digital reduce el excedente de ruido y comparando las gráficas se visualizan las frecuencias que se procuran reducir.

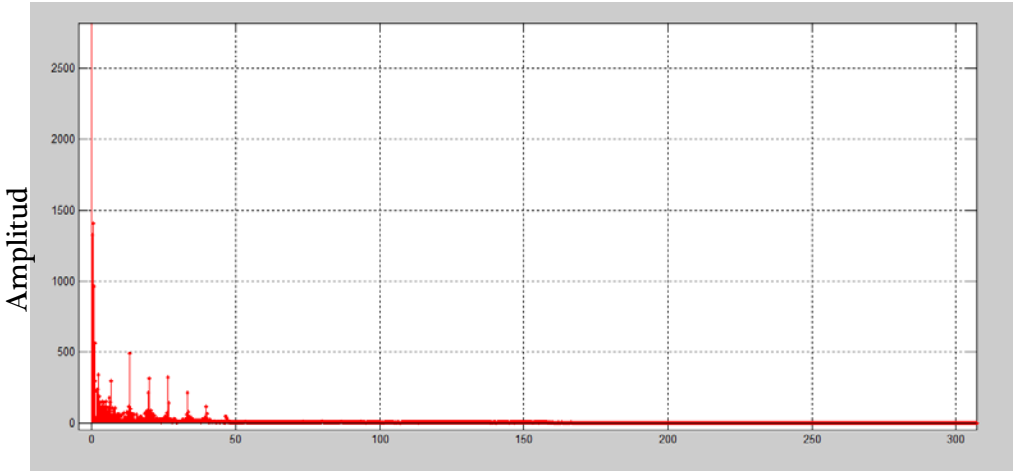


Figura 4-22.- Análisis espectral de la señal ECG con filtro

Realizando este análisis espectral se pueden notar las frecuencias presentes en la señal Electrocardiograma de ratas. En esta primer señal ECG de rata se tiene una amplitud de voltaje de 0.40 mV a una frecuencia de 400 latidos por minuto. Comparando estos resultados con trabajos o artículos se encuentra que están dentro de rangos típicos.

En la tabla 4-1 se reflejan los resultados de las pruebas obtenidas ya que se calcula la amplitud de voltaje así como la frecuencia cardíaca en ratas Wistar, se tomó un espécimen para realizar el experimento.

Los resultados mostrados generan variaciones tanto en voltaje como en la frecuencia cardíaca pero siguen manteniendo un rango que está dentro de la función normal del corazón de una rata.

Número de prueba	Amplitud de voltaje	Frecuencia Cardíaca
1	0.40 mV	400 ltp
2	0.17 mV	400 ltp
3	0.25 mV	428 ltp
4	0.15 mV	444 ltp
5	0.27 mV	420 ltp

Tabla 4-1.- Mediciones de la señal ECG en una rata Wistar ante varias pruebas.

Se anexan algunas gráficas de visualización de Electrocardiograma en ratas para verificar que se han obtenido mayor cantidad de muestras, en la figura 4-23. Como se ha visto la señal ECG de ratas es dificil obtener la claridad como en humanos pero se busca tener la mejor resolución e interpretacion de los componentes de la señal.

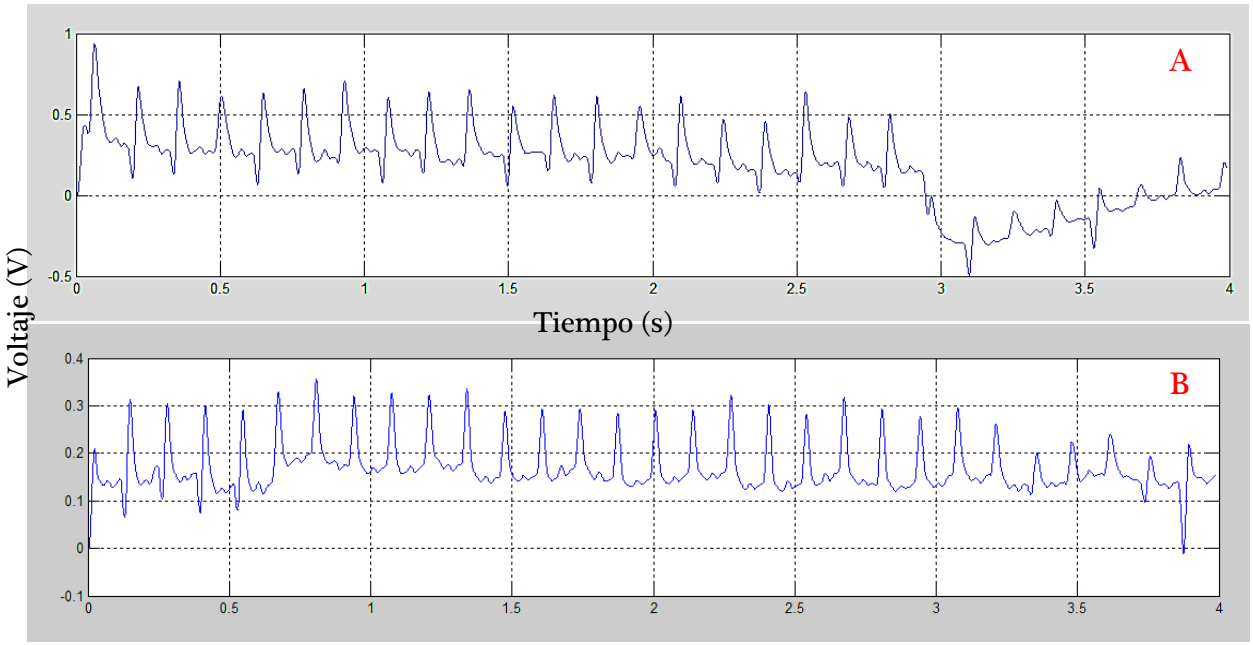


Figura 4-23.- Señales ECG filtradas de rata Wistar (a) derivación I (b) derivación II

La comparación con que se cuenta en la Universidad es un polígrafo analogico el cual traza las señales con un plumin en una hoja de papel, en la figura 4-24 se tiene una imagen del sistema.

Figura 4-24.- Polígrafo propiedad de la UAQ

Esta maquina adquiere señales biométricas analogicamente, por lo que es el sistema de comparación. Se puede apreciar la obtencion de la señal ECG (figura 4-25) mediante el polígrafo a una velocidad de 25 mm/s.

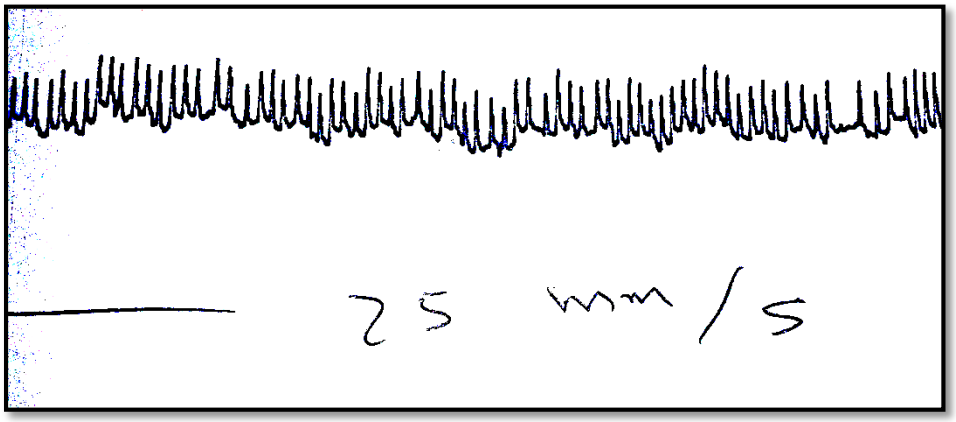


Figura 4-25.- Adquisición señal ECG en rata Wistar mediante un polígrafo, velocidad de 25 mm/s.

Claramente se puede observar un patron similar con los resultados adquiridos, aunque el sistema propuesto en este proyecto tiene mejor resolución y se pueden adquirir mejores lecturas y/o para observar los complejos que contiene la señal ECG de la rata asi como calcular el ritmo cardíaco.

En la figura 4-25 se tiene una velocidad de muestreo distinta aunque la resolución no mejora y el ruido es excesivo en estas señales adquiridas por el polígrafo. Como se observa la señal graficada no ayuda a ver con claridad la señal ECG.

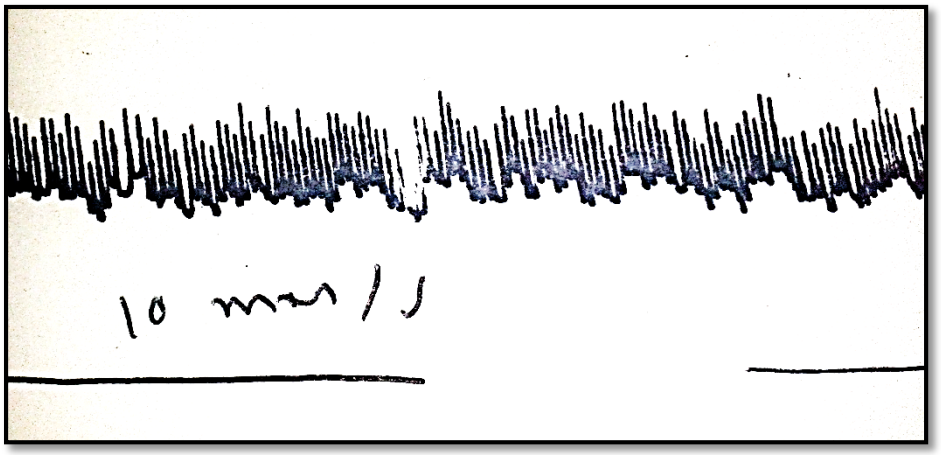


Figura 4-26.- Adquisición señal ECG en rata Wistar mediante un polígrafo, velocidad 10 mm/s.

Se realiza una comparacion entre las figuras 4-23, 4-25 y 4-26 por lo que se puede deducir que el sistema digital mejora la calidad de las señales biométricass pero las pruebas en el polígrafo ayudan a validar los resultados. Con esto se concluye que el sistema propuesto tiene un mejor rendimiento y con un espacio menor al sistema analogico que se cuenta dentro de la institución UAQ.

5. Conclusiones y Prospectivas

Conclusiones

El desarrollo de un sistema de adquisición de señales biométricas reconfigurable, permite enlazar el campo electrónico con el fisiológico, debido que al implementar un circuito que está conectado a un ser vivo, se cuida primordialmente la seguridad del mismo. La preparación para afrontar el proyecto fue básica, debido a que no se tenía conocimiento sobre señales biométricas y fue muy enriquecedor comenzar un proyecto así que se aprendió de las características de la señal Electrocardiograma.

Una de las principales tareas realizadas, fue diseñar tarjetas electrónicas (PCB) para tener una plataforma de adquisición de señales biométricas, se aplican filtros analógicos y digitales en la señal ECG, tratando de observar solamente las señales de interés, como los complejos, amplitudes y frecuencia cardíaca de la señal. Las etapas del sistema se diseñaron de tal forma que el ruido en la transmisión de la señal ECG fuera mínimo, aunque siempre estará presente el ruido ambiental, por movimiento muscular, por fricción del electrodo, etc. Esto permitió alcanzar los objetivos de diseño de tarjetas electrónicas, tanto diseño de parámetros de amplificación, filtrado y digitalización.

El diseño digital se crea de forma que pueda ser usado como modulo en futuras aplicaciones, se emplea un lenguaje de programación en hardware como lo es VHDL, la descripción de un Protocolo SPI e interconexión con módulos de USB y memoria dinámica fueron algunas de las tareas diseñadas en hardware. La implementación en FPGA permite tener un campo de acción grande para aplicaciones futuras donde se puedan desarrollar módulos de caracterización de las

señales y cuantificar algunos problemas fisiológicos sin necesidad de una computadora.

La creación de un software permitió la visualización de las señales, esto es fundamental ya que pudiéramos adquirir las señales ECG pero sin pruebas visuales no se tiene idea de las características de lo que se ha obtenido. Actualmente mostrar las señales es una etapa rigurosa en cualquier sistema para observar el comportamiento del sistema.

Las conclusiones mencionadas hacen que todos los objetivos han sido realizados, teniendo resultados y conclusiones del trabajo que pueden ayudar a que en un futuro pueda ser útil para continuar con la investigación en este campo de biomédica. Se encontraron tanto resultados buenos como malos por lo que la mejora del sistema es algo indispensable.

El sistema de adquisición de señales biométricas está adaptado para futuras aplicaciones en bioseñales, donde al conectar algún sistema de acondicionamiento de señal biométrica pueda digitalizar las señales y visualizarlas.

Prospectivas

La aplicación biomédica actualmente tiene muchas aplicaciones ya que se requiere mejorar la calidad de vida, es por ello que trabajar en esta área tiene muchos avances futuros.

Mejorar el diseño electrónico de las tarjetas o incluso llegar a incluir más herramientas de procesamiento digital que permita crear un sistema de adquisición de señales biométricas que pueda ser de clase comercial.

Este sistema puede llegar a adquirir otras señales biométricas siempre y cuando se cuente con un acondicionamiento de la señal en cuestión, en este proyecto

se concentró con la señal Electrocardiograma pero puede ser aplicable para otras señales como Electroencefalograma o Electromiograma o señales que trabajan a niveles de voltaje bajos, ya que por las características de diseño puede obtener una buena resolución para visualizar señales de voltaje en milivolts o rangos menores.

Por el diseño en una plataforma FPGA se puede llegar a tener conciencia de desarrollar más el sistema, implementando una interfaz autoritaria sin necesidad de una pantalla de computadora o que la interfaz de usuario sea mejor empleando un protocolo de comunicación más rápido como el Ethernet para lograr una transmisión de datos eficiente y poder monitorear la señal ECG en tiempo real.

Se puede proponer realizar mayor procesamiento en hardware para cuantificar las características de la señal ECG y dar un pronóstico de las enfermedades que pueda poseer el individuo de prueba.

El trabajo futuro sobre esta área apenas ha comenzado por lo que se puede llegar a alcanzar varios avances tecnológicos implementando nuevas técnicas o llegar a monitorear varias variables al mismo tiempo de un mismo individuo para proporcionar un diagnostico general.

6. Referencias

Aimen K. Farraj, M. S. (2011). The Utility of the Small Rodent Electrocardiogram in Toxicology. *TOXICOLOGICAL SCIENCE*, 11 - 30.

CCM. (2015). *CCM*. Obtenido de http://es.ccm.net/contents/398-memoria-de-acceso-aleatorio-memoria-ram-o-pc

Chu, V. (2001). Method for non-invasively recording electrocardiograms in conscious mice. *BMC Physiology*, 1-6.

Curso Tratamiento Digital de Señal. (1999). Diseño de Filtros Digitales.

Electrocardiografía.es. (2010). *Electrocardiografia.ES*. Obtenido de http://www.electrocardiografia.es/

Farmer, J., & Levy, G. (1967). A simple method for recording the electrocardiogram and heart rate from conscious animals.

Feijo Romero, M. A., & Reyes Sánchez, B. A. (Junio de 2010). Diseño y Construcción de un Monitor de Tres Parámetros Fisiologicos. *TESIS*. Cuenca, Provincia de Azuay, Ecuador.

Fernández, H. (2008). Amplificador Operacional.

Figueroa Suarez, L., & Medina Pulido, D. R. (Diciembre de 2009). Diseño e Implementación de un electrocardiografo inalambrico digital para ratas de laboratorio utilizando tecnologia bluetooth. *TESIS*. Bucaramanga, Santander, Colombia.

Flores Chávez, P. L., Infante Vázquez, O., Sánchez Torres, G., Martinez Memíje, R., & Rodríguez Rossini, G. (2002). Detección de signos vitales en ratas mediante metodos no invasivos. *Publicación en Linea*.

Gaspar. (2009). Tipos de Filtros.

Gonzalez, I. A. (Septiembre de 2010). Diseño y Construcción de un Sistema para la Detección se Señales Electromiográficas. *TESIS*. Merida, Yucatan, México.

Grupo de Mecatrónica-UAQ. (2009). *Unidad de Procesamiento Digital de Señales en Hardware.* Querétaro.

Heredia López, F. J., & Góngora Alfaro, J. L. (1994). Evaluación de una tarjeta de computadora para adquisición y desplegado de señales electrocardiográficas de ratas. *Rev Biomed, Vol 5, No 1*, 3-11.

Huircán, J. I. (2008). Conversores Análogico-Digital y Digital-Análogo: Conceptos Básicos.

Leonhard Lang GmbH. (2011). Electrodos para ECG SKINTACT, económicos y multiuso.

Letizia Lo Presti, M. I. (2000). Efficient Modified-Sinc Filters for Sigma–Delta A/D Converters. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*.

Linear Technology. (2005). *1.5A, Low Noise, Fst Transient Response LDO Regulators.*

Linear Technology. (2011). *1.5A, Low Noise, Negative Linear Regulator with Precision Current Limit.*

Montes, L. E. (18 de Enero de 2000). Monitor Electrocardiografo. *TESIS*. Querétaro, Querétaro, México.

Pereira Junior, P. P., Marocolo, M., Rodrigues, F. P., Medei, E., & Nascimiento, J. (2010). Noninvasive method for electrocardiogram recording in conscious rats:. *Anais da Academia Brasileira de Ciências (*, 431-437.

Pérez, E. L. (2000). Protocolo SPI.

Pérez, E. L. (2000). Protocolo USB (universal serial bus). *Paraninfo*.

PlesseySemiconductors. (2010). EPIC Ultra High Impedance ECG Sensor. Data Sheet 291766 issue 1.

Soldan, S., Reisman, S., Bergen, M., & Servatius, R. (1997). Instrumentation of the Awake Unrestrained Rat for ECG and Respiration. *IEEE* .

Tejeda, J. J. (Junio de 2009). Diseño e Investigación de un Sistema Multicanal Portatil de Registro de Bioseñales. *TESIS*. Mexico, D.F., Mexico.

Texas Instrument. (1995). *Precision, Low Power Instrumentation Amplifiers.*

Texas Instrument. (2015). *ADS129x Low-Power, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements.*

Texas Instrument. (2015). *TL08xx JFET-Input Operational Amplifiers.*

Wu, M., Tang, H., O'Connell, J., Gao, D., Ido, A., Da Silva, A., . . . Dae, M. (1999). An Ultra High Resolution ECG-Gated Myocardial Imaging System for Small Animals. *IEEE*.

7. Apéndice

Esta sección está destinada a conocer los lenguajes de programación empleados y diseño de tarjetas electrónicas así como la conexión para posible uso.

7.1 Diseño de Tarjetas Electrónicas

En esta sección se muestran las tarjetas electrónicas diseñadas para el acondicionamiento de la señal Electrocardiograma, así como los diseños implementados en Altium Designer.

La Figura 7-1 muestra la tarjeta de adquisición de bioseñales, donde se emplea la tarjeta UPDSH como medio de procesador, el sistema contiene un convertidor ADS1298, 2 reguladores de voltaje a +/- 2.5V, un display de 7 segmentos de 4 dígitos, un puerto USB tipo B, 2 led's, 2 switchs y un puerto de expansión I/O.

La Figura 7-1 muestra la parte superior de la tarjeta.

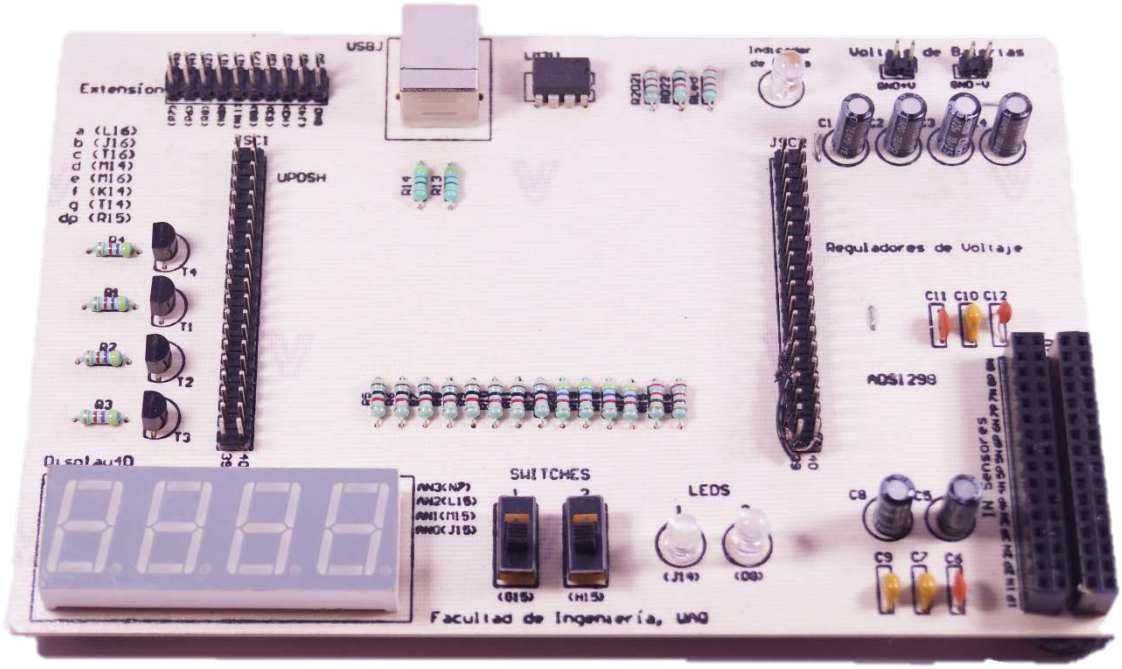


Figura 7-1.- Tarjeta de Desarrollo de adquisición de señales biométricas

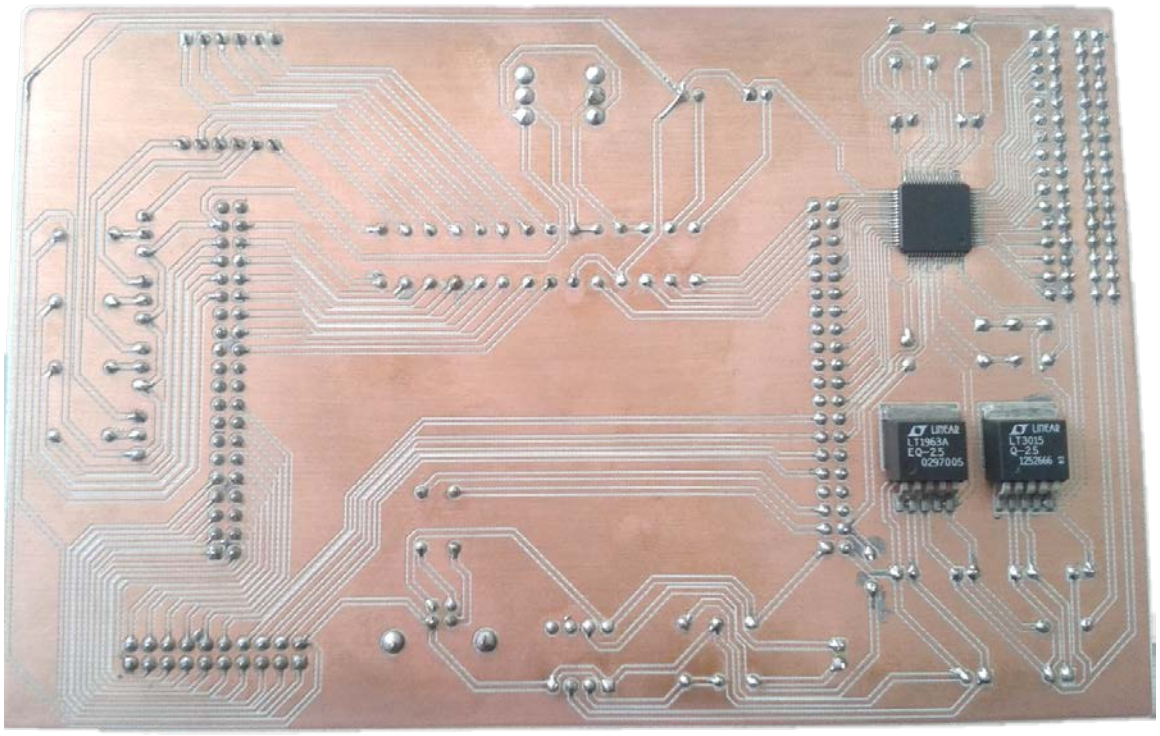


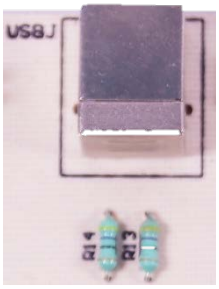
Figura 7-2.- Tarjeta de Desarrollo de adquisición de señales biométricas parte trasera

La parte inferior muestra el convertidor ADS1298 (Figura 7-2), al cual se alimenta mediante los reguladores de voltaje a +/- 2.5V, igualmente colocados en esta cara, para lograr una mejor precisión y calidad en la conversión de los datos.

La tarjeta emplea los siguientes puertos del sistema UPDSH, los cuales son categorizados por la función a la que se configuro en hardware.

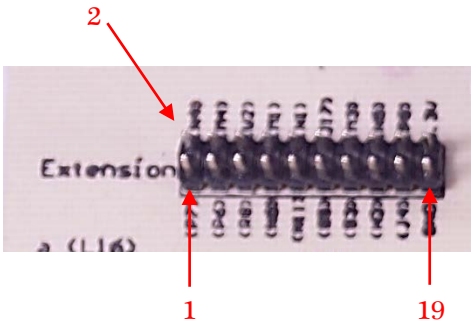
USB

Conexión	Puerto
USB_DM	L3
USB_DP	L1



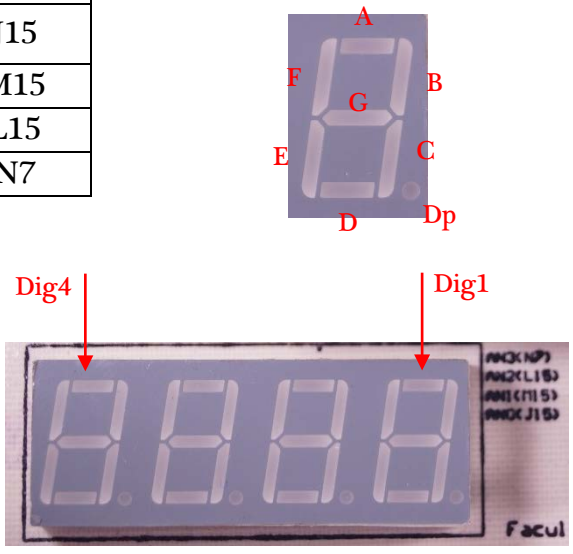
Extensión.- Los conectores 6, 8 y 10 se pueden usar solamente como entradas debido a su configuración en hardware, pero el resto de ellos se pueden usar como entrada o salida.

Conector	Puerto	Conector	Puerto
1	P7	2	N4
3	P6	4	M4
5	R8	6	V2
7	N8	8	M1
9	N11	10	N1
11	G5	12	J17
13	G3	14	M3
15	H3	16	N5
17	J4	18	P8
19	GND	20	3.3V



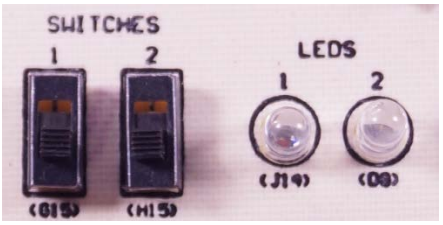
Display de 7 Segmentos.- El display enciende enviando un 0 al segmento que deseamos usar y un 1 al digito del display.

Display7seg	Puerto	Digito	Puerto
A	L16	Dig1	J15
B	J16	Dig2	M15
C	T16	Dig3	L15
D	M14	Dig4	N7
E	M16		
F	K14		
G	T14		
Dp	R15		



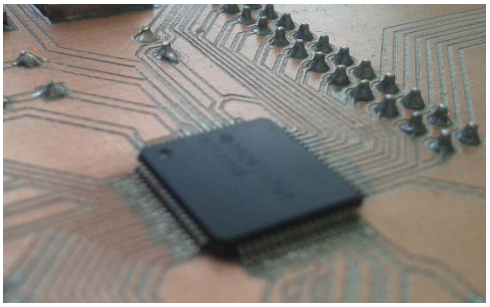
Switches y Led's

Switch	Puerto
1	G15
2	H15
Led's	Puerto
1	J14
2	D11



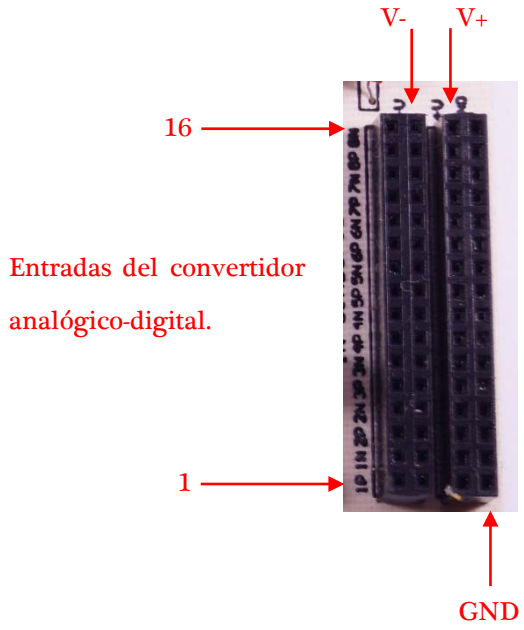
Convertidor ADS1298.- El convertidor se encuentra en la parte de debajo de la tarjeta debido a que es de montaje superficial. Los puertos de entrada del convertidor están colocados a pines hembra.

Pin ADS	Puerto
CLKSel	H4
DRDY	G4
GPIO4	J5
GPIO3	D5
GPIO2	D7
GPIO1	D10
Start	F14
CLK	H14
Reset	G16
PWDN	H16
SCLK	E10
CS	D14
Din	G14
Dout	D9



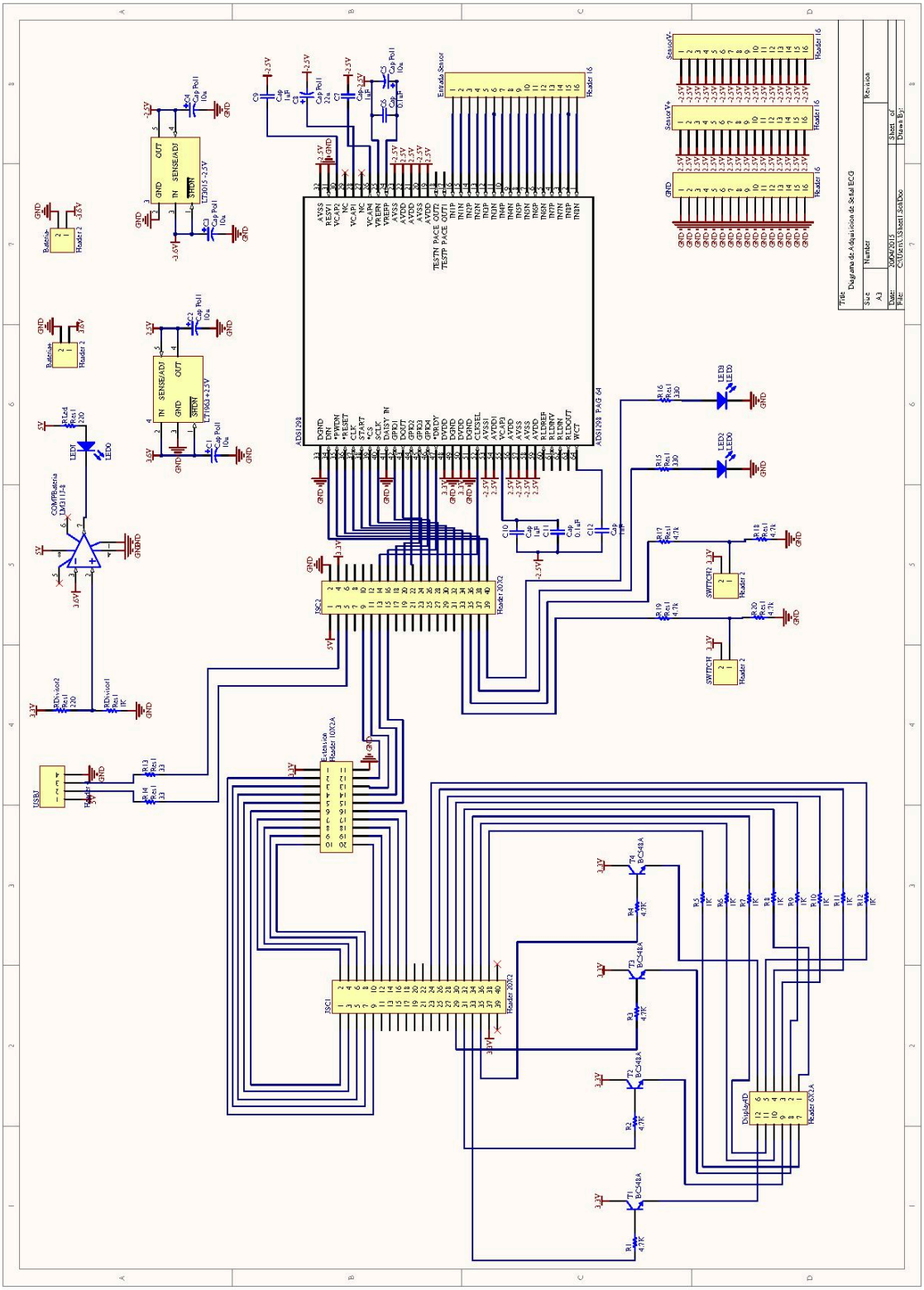
Conector de Entradas analógicas del convertidor.- El convertidor tiene una capacidad de 8 entradas analógicas, las cuales pueden ser configuradas desde software. Las entradas contienen pines positivo y negativo de cada canal.

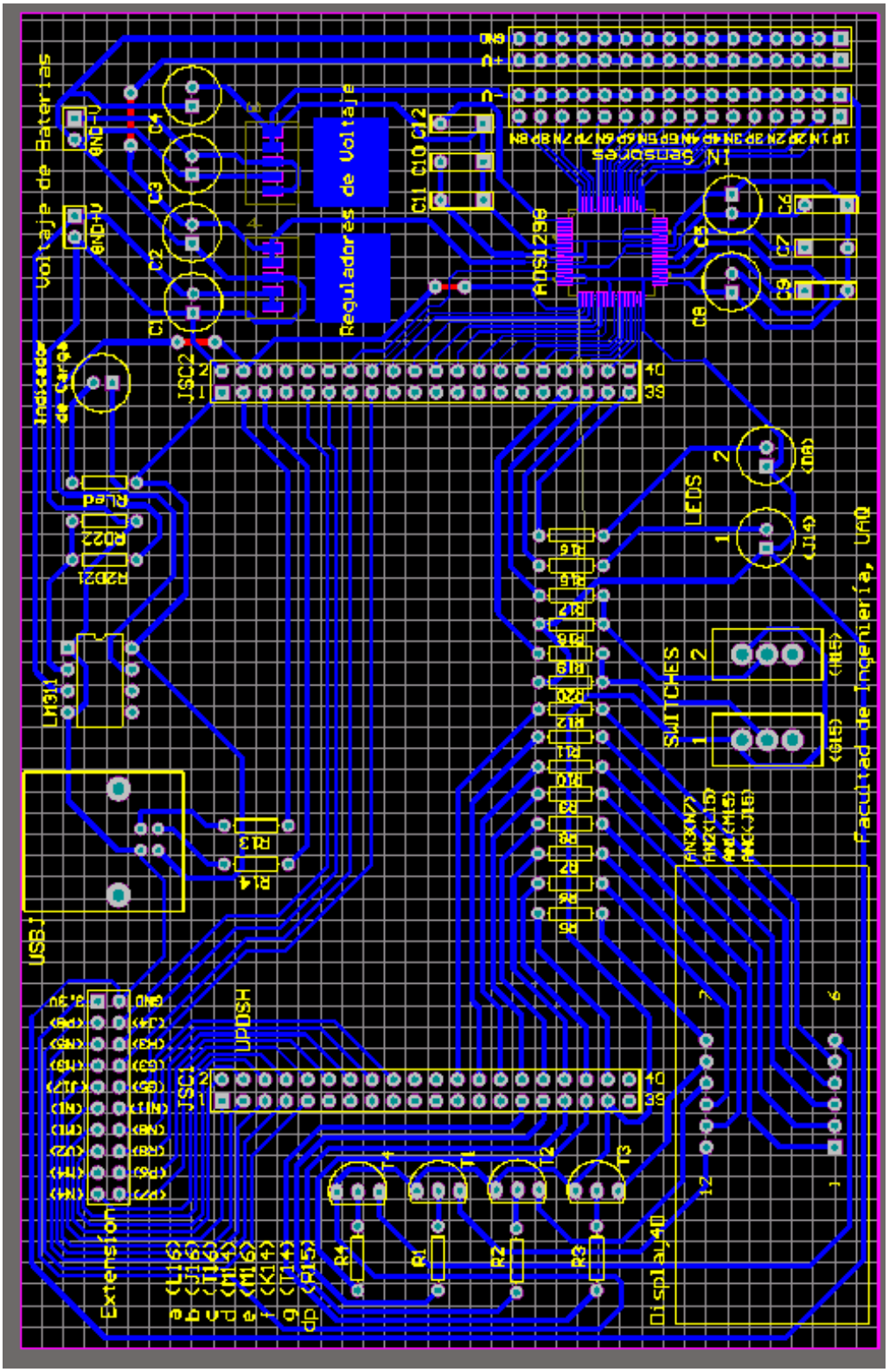
Pin	Entrada ADS
1	IN1P
2	IN1N
3	IN2P
4	IN2N
5	IN3P
6	IN3N
7	IN4P
8	IN4N
9	IN5P
10	IN5N
11	IN6P
12	IN6N
13	IN7P
14	IN7N
15	IN8P
16	IN8N



Los indicadores de voltaje positivo y negativo, muestran que la columna que señala V- (columna 2), es porque toda la columna proporciona el voltaje de -2.5V. Al igual que la columna mostrada con V+ (columna 3), nos da un voltaje de +2.5V. Y la columna 4 proporciona en cada pin tierra (GND).

A continuación se muestra el diseño implementado en el software Altium, señala el esquema y muestra el ruteo del circuito de la tarjeta de adquisición de señales biométricas.




Voltaje de Baterias
Indicador de Carga
Reguladores de Voltaje
Sensores
LEDS
SWITCHES
USB
Extension
Display4Bit
Facultad de Ingeniería, UAQ

En la figura 7-3 se puede ver la tarjeta de acondicionamiento del Electrocardiograma, donde cuenta con un preamplificador, un filtro butterworth pasa-altas, pasa-bajas y un filtro notch a 60 Hz. Con alimentación preferencial a +/- 2.5V.

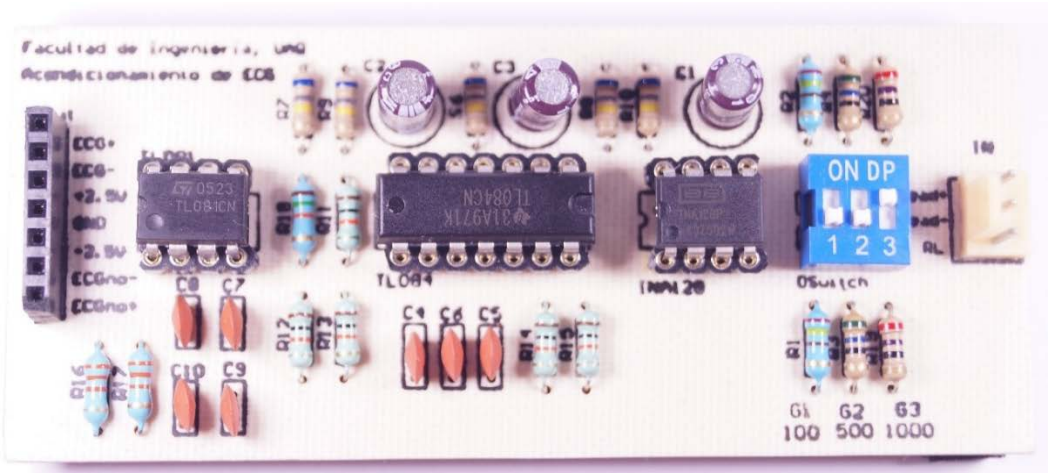
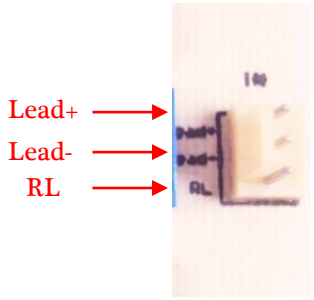


Figura 7-3.- Sistema de acondicionamiento ECG

El sistema de acondicionamiento cuenta con un selector de ganancias el cual puede auxiliar en la toma de la señal ECG, puede ser usado para roedores y humanos simplemente cambiando la ganancia.

Entradas de Electrocardiograma para registro de derivaciones bipolares.

Pin	Característica
Lead+	Se conecta la derivación positiva del ser vivo (Punto de referencia positivo).
Lead-	Se conecta la derivación negativa del ser vivo (Punto de referencia negativo)
RL	Se conecta la referencia tierra del ser vivo (Punto de referencia tierra).

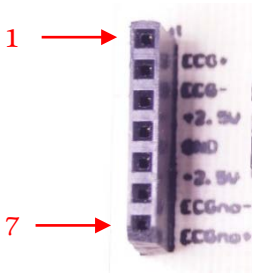


La configuración de la tarjeta para la ganancia es mediante el dip-switch. En la tarjeta se ve en 1 una ganancia distinta y fue porque se amplió el rango, es decir, el valor de 100 realmente es de 5000.

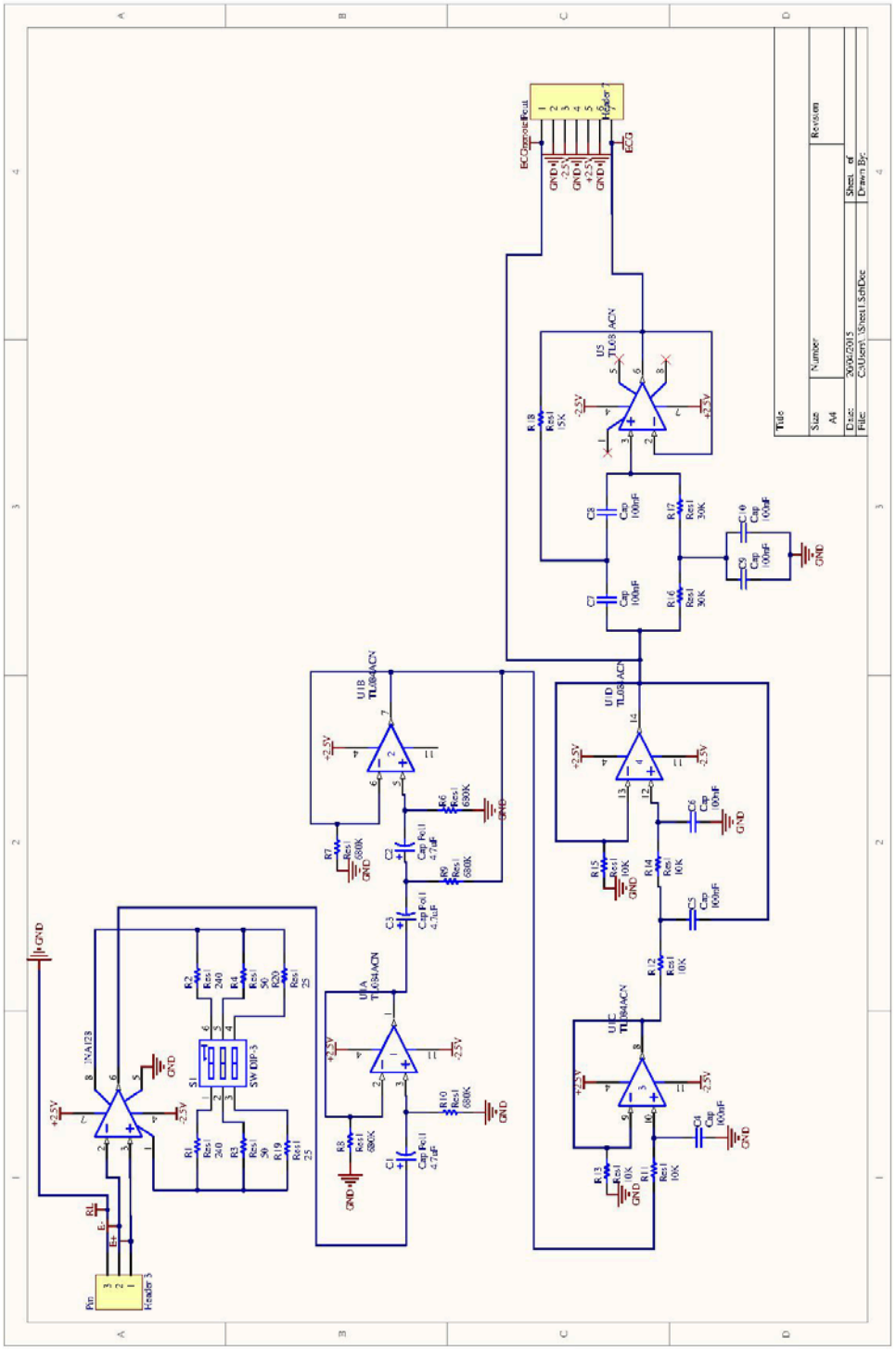
Dip Switch	Ganancia
1	5000
2	500
3	1000

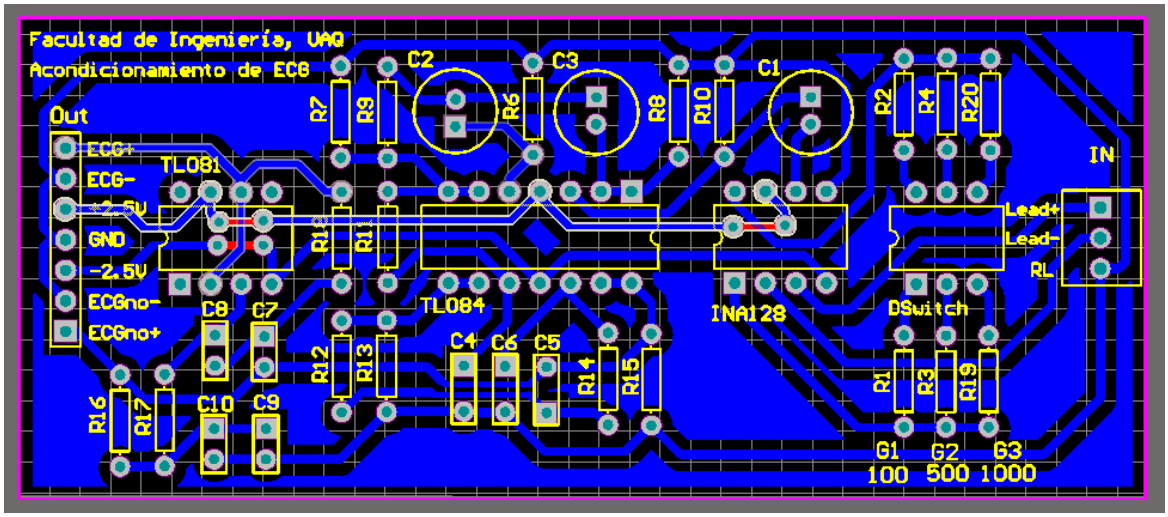
Alimentación y salidas de la tarjeta.

Pin	Característica
1	Salida ECG positivo
2	Salida ECG negativo
3	Entrada Voltaje +2.5V
4	Entrada Tierra GND
5	Entrada Voltaje -2.5V
6	Salida ECG negativo antes de filtro notch
7	Salida ECG positivo antes de filtro notch



En las figuras siguientes se puede observar el esquema y ruteo de la tarjeta de acondicionamiento de ECG.




Facultad de Ingeniería, UAQ
Acondicionamiento de ECG
Out
ECG+
ECG-
-2.5V
GND
-2.5V
ECGno-
ECGno+
TL081
TL084
INA128
DSwitch
IN
Lead+
Lead-
RL
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R19
R20
G1
G2
G3
100
500
1000

7.2 Interfaz en Hardware

La interfaz en hardware se describe mediante el protocolo de comunicación SPI y el uso de módulos USB y Memoria Dinámica. A continuación se muestran los bloques principales e interfaz ISE donde fue programada bajo un lenguaje de programación VHDL, Figura 7-1.

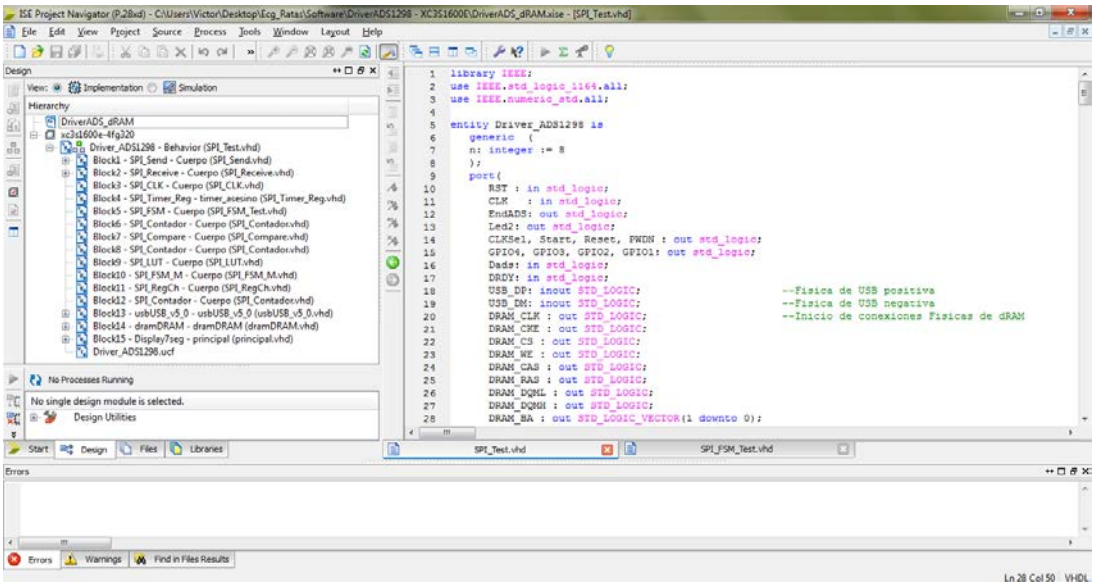


Figura 7-4.- Software de Programación ISE

Driver_ADS1298 (modulo principal)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity Driver_ADS1298 is
    generic (
    n: integer := 8
    );
    port(
        RST : in std_logic;
        CLK : in std_logic;
        EndADS: out std_logic;
        Led2: out std_logic;
        CLKSel, Start, Reset, PWDN : out std_logic;
        GPIO4, GPIO3, GPIO2, GPIO1: out std_logic;
        Dads: in std_logic;
        DRDY: in std_logic;
        USB_DP: inout STD_LOGIC;                    --Fisica de USB positiva
        USB_DM: inout STD_LOGIC;                    --Fisica de USB negativa
        DRAM_CLK : out STD_LOGIC;                   --Inicio de conexiones Fisicas
                                                      de dRAM
```

```
21          DRAM_CKE : out STD_LOGIC;
22          DRAM_CS : out STD_LOGIC;
23          DRAM_WE : out STD_LOGIC;
24          DRAM_CAS : out STD_LOGIC;
25          DRAM_RAS : out STD_LOGIC;
26          DRAM_DQML : out STD_LOGIC;
27          DRAM_DQMH : out STD_LOGIC;
28          DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
29          DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
30          DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0);    --Fin de conexiones Fisicas de
                                                                dRAM
31          Din : out std_logic;
32          CS  : out std_logic;
33          SCLK: out std_logic;
34          AN  :   out std_logic_vector(3 downto 0);       --Asignacion de los display
35          D7S :   out std_logic_vector(7 downto 0)        --Asignacion de los 7 segmentos
36          );
37  end Driver_ADS1298;
38
39  architecture Behavior of Driver_ADS1298 is
40  component SPI_FSM
41      port(
42          RST : in std_logic;
43          CLK : in std_logic;
44          STR0: in std_logic;
45          STRA: in std_logic;
46          EoT : in std_logic;
47          EoR : in std_logic;
48          E1  : in std_logic;
49          E2  : in std_logic;
50          E3  : in std_logic;
51          STR1: out std_logic;
52          STR2: out std_logic;
53          SHab: out std_logic;
54          CS  : out std_logic;
55          Ksel: out std_logic;
56          OPC : out std_logic_vector(1 downto 0);
57          OPR : out std_logic_vector(1 downto 0);
58          OPC2: out std_logic_vector(1 downto 0);
59          SEND: out std_logic
60          );
61  end component;
62
63  component SPI_CLK
64      generic (
65      n: integer := 8
66      );
67      port(
68          RST : in std_logic;
69          CLK : in std_logic;
70          Hab : in std_logic;
71          K   : in std_logic_vector(n-1 downto 0);
72          Z   : out std_logic;
73          SCLK : out std_logic
74          );
75  end component;
76
77  component SPI_Send
78      generic (
79      n: integer := 8
80      );
81      port(
82          RST : in std_logic;
83          CLK : in std_logic;
84          STR1: in std_logic;
85          Z   : in std_logic;
86          Dreg: in std_logic_vector(n-1 downto 0);
87          Hab1: out std_logic;
88          Din : out std_logic;
```

```
89          EoT : out std_logic
90          );
91  end component;
92
93  component SPI_Receive
94      generic (
95      n: integer := 8
96      );
97      port(
98          RST : in std_logic;
99          CLK : in std_logic;
100         STR2: in std_logic;
101         Dads: in std_logic;
102         Z   : in std_logic;
103         Hab2: out std_logic;
104         Dout: out std_logic_vector(n-1 downto 0);
105         EoR : out std_logic
106         );
107 end component;
108
109 component SPI_Compare
110     generic (
111     n: integer := 8
112     );
113     port(
114         RST : in std_logic;
115         CLK : in std_logic;
116         OPC : in std_logic_vector(1 downto 0);
117         K   : in std_logic_vector(n-1 downto 0);
118         E   : out std_logic
119         );
120 end component;
121
122 component SPI_LUT
123     port(
124         D   : in std_logic_vector(7 downto 0);
125         VelMuestreo : in std_logic_vector(2 downto 0);
126         Ch1g,Ch2g,Ch3g,Ch4g,Ch5g,Ch6g,Ch7g,Ch8g : in std_logic_vector(2 downto 0);
127         Dl  : out std_logic_vector(7 downto 0)
128         );
129 end component;
130
131 component SPI_Contador
132     generic (
133     n: integer := 8
134     );
135     port(
136         RST : in std_logic;
137         CLK : in std_logic;
138         OPC : in std_logic_vector(1 downto 0);
139         K   : in std_logic_vector(n-1 downto 0);
140         D   : out std_logic_vector(n-1 downto 0);
141         E   : out std_logic
142         );
143 end component;
144
145 component SPI_FSM_M
146     port(
147         RST : in std_logic;
148         CLK : in std_logic;
149         SEND: in std_logic;
150         EndADS: out std_logic;
151         EoR : in std_logic;
152         EoP : in std_logic;
153         EoM : in std_logic;
154         EoM2: in std_logic;
155         RDYR: in std_logic;
156         RDY : in std_logic;
157         CSR : out std_logic;
158         OPC3: out std_logic_vector(1 downto 0);
```

```
159         OPC4: out std_logic_vector(1 downto 0);
160         CSU : out std_logic;
161         WR  : out std_logic;
162         RD  : out std_logic;
163         MUX : out std_logic;
164         FSM : out std_logic;
165         Conf: in std_logic;
166         STR0: out std_logic;
167         STR0aux: in std_logic;
168         RDY_R: in std_logic;
169         CSU_R: out std_logic
170         );
171 end component;
172
173 component  SPI_RegCh
174     generic (
175     n: integer := 8
176     );
177     port(
178         RST : in std_logic;
179         CLK : in std_logic;
180         OPC : in std_logic_vector(1 downto 0);
181         D   : in std_logic_vector(n-1 downto 0);
182         Dl  : out std_logic_vector(27*n-1 downto 0)
183         );
184 end component;
185
186 component SPI_Timer_Reg
187     generic(n: integer:= 16);
188     port(
189     RST: in std_logic;
190     CLK: in std_logic;
191     PLS: in std_logic;
192     Q:   out std_logic_vector(n-1 downto 0)
193     );
194 end component;
195
196 component usbUSB_v5_0
197    generic(VID : integer := 0;
198            PID : integer := 0);
199     port (
200          RST: in STD_LOGIC;
201          CLK48: in STD_LOGIC;
202          SLOT: in STD_LOGIC_VECTOR (7 downto 0);
203          USB_DP: inout STD_LOGIC;
204          USB_DM: inout STD_LOGIC;
205          CS: in STD_LOGIC_VECTOR (2 downto 1);
206          STALL: in STD_LOGIC_VECTOR (2 downto 1);
207          RDY: out STD_LOGIC_VECTOR (2 downto 1);
208          Di : in STD_LOGIC_VECTOR(127 downto 0);
209          Do : out STD_LOGIC_VECTOR(255 downto 0)
210      );
211 end component;
212
213 component dramDRAM
214     generic(n : integer := 256;
215                 m : integer := 4);
216      port(
217          RST : in STD_LOGIC;
218          CLK : in STD_LOGIC;
219          CS : in STD_LOGIC;
220          WR : in STD_LOGIC;
221          RD : in STD_LOGIC;
222          RDY : out STD_LOGIC;
223          K : in STD_LOGIC_VECTOR(m-1 downto 0);
224          Din : in STD_LOGIC_VECTOR(n-1 downto 0);
225          Dout : out STD_LOGIC_VECTOR(n-1 downto 0);
226          DRAM_CLK : out STD_LOGIC;
227          DRAM_CKE : out STD_LOGIC;
```

```
228           DRAM_CS : out STD_LOGIC;
229           DRAM_WE : out STD_LOGIC;
230           DRAM_CAS : out STD_LOGIC;
231           DRAM_RAS : out STD_LOGIC;
232           DRAM_DQML : out STD_LOGIC;
233           DRAM_DQMH : out STD_LOGIC;
234           DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
235           DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
236           DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0)
237           );
238   end component;
239
240   component Display7seg
241       port(
242       CLK :   in  std_logic;
243       RST :   in  std_logic;
244       b0,b1,b2,b3 :   in  std_logic_vector(3 downto 0);
245       led :   out std_logic_vector(3 downto 0);     --Asignacion de los display
246       dis :   out std_logic_vector(7 downto 0)      --Asignacion de los 7 segmentos
247       );
248   end component;
249
250   --Variables de control en Maquinas de estados
251   signal EoT, EoR, STR1, STr2, Z, SHab, Hab, Hab1, Hab2, HabUSBR, STRA,
252          E1, EoP, EoM, EoM2, Ksel, RDY, MUX, CSU, CSU_R, CSU_R1, RDY_R, D1, Q1,
          FSM,Conf: std_logic;
253   signal K, K1, K2, D, Dl, Dout: std_logic_vector(n-1 downto 0);
254   signal Dreg  : std_logic_vector(27*n-1 downto 0);
255   signal DRDYa, OPC, OPR ,OPC2, OPC3, OPC4: std_logic_vector(1 downto 0);
256   signal K3 , Address, Address2 : std_logic_vector(23 downto 0);
257   signal Qtim : std_logic_vector(23 downto 0);
258   --Variables de dRAM
259   signal CSR, WR, RD, RDYR: std_logic;
260   signal DinR, DoutR : std_logic_vector(255 downto 0);
261   --Variables de USB
262   signal CSUSB,RDYUSB: std_logic_vector(1 downto 0);
263   signal DiUSB: std_logic_vector(127 downto 0);
264   signal DoUSB: std_logic_vector(255 downto 0);
265   --variables de display
266   signal b0,b1,b2,b3  :   std_logic_vector(3 downto 0);
267   signal VelMuestreo : std_logic_vector(2 downto 0);
268   signal Ch1g,Ch2g,Ch3g,Ch4g,Ch5g,Ch6g,Ch7g,Ch8g: std_logic_vector(2 downto 0);
269   signal STR0,STR0aux,SEND: std_logic;
270   begin
271       --Datos de configuracion de envio de paquetes y pulso de reloj
272       K <= "00000111";                                    --Ancho de pulso del reloj SCLK
273       K1 <= "00010011" when Ksel='0' else "00100001" when Ksel='1';  --Escritura de
                                                                          opcodes
274       K2 <= "00011011";--Muestra de 8 canales y 1 registro 27 paquetes de 8 bits 216 bits
275       --Habilitaciones y registro de DRDY para flanco descendente
276       Hab <= Hab1 when SHab='0' else Hab2 when SHab='1';     --Habilitar reloj
                                                                  envio o recepcion
277
278       DRDYa <= (others =>'0') when RST='1' else (DRDY & DRDYa(1)) when rising_edge(CLK);
279       STRA  <= (not(DRDYa(1)))and(DRDYa(0));   --Deteccion de flanco descendente de DRDY
280       -- Variables de corroboracion si llegan los paquetes completos
281       GPIO4 <= '0';
282       GPIO3 <= '1';
283       GPIO2 <= '0';
284       GPIO1 <= '1';
285       --Visualizacion de direcciones en display
286       b0 <= Address(3 downto 0) when FSM='0' else Address2(3 downto 0) when FSM='1';
287       b1 <= Address(7 downto 4) when FSM='0' else Address2(7 downto 4) when FSM='1';
288       b2 <= Address(11 downto 8) when FSM='0' else Address2(11 downto 8) when FSM='1';
289       b3 <= Address(15 downto 12) when FSM='0' else Address2(15 downto 12) when FSM='1';
290
291       ------   Envio a DRAM
292       DinR <= Qtim & Address(15 downto 0) & Dreg;   -- Q 24 bits, Address 16 bits, Dreg
                                                          216 bits
293
```

```
294      ------   Envio a USB
295      DiUSB <= DoutR(127 downto 0) when MUX='0' else DoutR(255 downto 128) when MUX='1';
296      CSUSB <= CSU & CSU_R1;
297      RDY <= RDYUSB(1);
298
299      ------   Recibir de USB
300      --DoUSB es distribuido para configurar dispositivo
301      RDY_R <= RDYUSB(0);
302
303      Conf <= DoUSB(0);--Bit de Configuracion
304
305      -- Variables de iniciacion de ADS
306      CLKSel <= DoUSB(104);
307      Start <= '1';-- Start puede estar en alto siempre.
308      Reset <= DoUSB(96);
309      PWDN <= DoUSB(112);
310      Led2 <= DoUSB(120);
311
312      STR0aux <= DoUSB(128);
313
314     --  Configuraciones de la tarjeta
315      process(Conf)
316      begin
317          if(Conf='1')then
318              Ch1g <= DoUSB(3 downto 1);
319              Ch2g <= DoUSB(10 downto 8);
320              Ch3g <= DoUSB(18 downto 16);
321              Ch4g <= DoUSB(26 downto 24);
322              Ch5g <= DoUSB(34 downto 32);
323              Ch6g <= DoUSB(42 downto 40);
324              Ch7g <= DoUSB(50 downto 48);
325              Ch8g <= DoUSB(58 downto 56);
326              VelMuestreo <= DoUSB(66 downto 64);
327              K3 <= DoUSB(95 downto 72); --Muestras generales
328          end if;
329      end process;
330
331      ------   Protocolo SPI
332      Block1: SPI_Send generic map(n)  port map(RST, CLK, STR1, Z, Dl, Hab1, Din, EoT);
333      Block2: SPI_Receive generic map(n) port map(RST,CLK,STR2,Dads,Z,Hab2,Dout,EoR);
334
335      Block3: SPI_CLK generic map(n) port map(RST, CLK, Hab, K, Z, SCLK);
336      Block4: SPI_Timer_Reg generic map(24) port map(RST, CLK, STRA, Qtim);
337
338      Block5: SPI_FSM port
         map(RST,CLK,STR0,STRA,EoT,EoR,E1,EoP,EoM,STR1,STR2,SHab,CS,Ksel,OPC,OPR,OPC2,SEND);
339      --Contador Registros LUT
340      Block6: SPI_Contador generic map(n)  port map(RST, CLK, OPC, K1, D, E1);
         --Contador Paquetes de 27
341      Block7: SPI_Compare  generic map(n)  port map(RST, CLK, OPR, K2, EoP);
         --Contador Muestras
342      Block8: SPI_Contador generic map(24) port map(RST, CLK, OPC2, K3, Address, EoM);
343
344      Block9: SPI_LUT port map(D,VelMuestreo,Ch1g,Ch2g,Ch3g,Ch4g,Ch5g,Ch6g,Ch7g,Ch8g,Dl);
345
346      ------  Envio dram y usb
347      Block10: SPI_FSM_M  port map(RST,CLK,SEND,EndADS, EoR, EoP, EoM, EoM2, RDYR, RDY,
348                                           CSR, OPC3, OPC4, CSU, WR, RD, MUX, FSM,
349                                           Conf,STR0,STR0aux,RDY_R,CSU_R1);
350      Block11: SPI_RegCh generic map(n) port map(RST, CLK, OPC3, Dout, Dreg);
         --Genera registro de 216 bits = 1 Paquete de 27
351
352      Block12: SPI_Contador generic map(24) port map(RST, CLK, OPC4, K3, Address2, EoM2);
         --Contador Muestras
353
354      ------   Envio USB
355      Block13: usbUSB_v5_0 generic map(9001,7929)--- Driver USB
356               port map (RST,CLK,"00000001",USB_DP,USB_DM,CSUSB,"00",RDYUSB,DiUSB,DoUSB);
357
```

```
358      ------   Uso de dram
359      Block14: dramDRAM generic map(256,4)          --- Driver dRAM
360                        port map (RST,CLK,CSR,WR,RD,RDYR,"1111",DinR,DoutR,
361
         DRAM_CLK,DRAM_CKE,DRAM_CS,DRAM_WE,DRAM_CAS,DRAM_RAS,DRAM_DQML,
         DRAM_DQMH,DRAM_BA,DRAM_A,DRAM_DQ);
362      ------   Encender display 7 segmentos
363      Block15: Display7seg port map (CLK,RST,b0,b1,b2,b3,AN,D7S);
364  end Behavior;
```

Bloque 1 SPI_Send

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_Send is
 5      generic (
 6      n: integer := 8
 7      );
 8      port(
 9          RST : in std_logic;
10          CLK : in std_logic;
11          STR1: in std_logic;
12          Z   : in std_logic;
13          Dreg: in std_logic_vector(n-1 downto 0);
14          Hab1: out std_logic;
15          Din : out std_logic;
16          EoT : out std_logic
17          );
18  end SPI_Send;
19
20  architecture Cuerpo of SPI_Send is
21  component SPI_C81
22      generic (
23      n: integer := 8
24      );
25      port(
26          RST : in std_logic;
27          CLK : in std_logic;
28          En: in std_logic_vector(1 downto 0);
29          D: in std_logic_vector(n-1 downto 0);
30          Din: out std_logic
31          );
32  end component;
33
34  component SPI_Compare
35      generic (
36      n: integer := 8
37      );
38      port(
39          RST : in std_logic;
40          CLK : in std_logic;
41          OPC: in std_logic_vector(1 downto 0);
42          K   : in std_logic_vector(n-1 downto 0);
43          E: out std_logic
44          );
45  end component;
46
47  component SPI_FSM_S
48      port(
49          RST : in std_logic;
50          CLK : in std_logic;
51          STR1    : in std_logic;
52          E   : in std_logic;
53          Z   : in std_logic;
54          OPR: out std_logic_vector(1 downto 0);
55          OPC: out std_logic_vector(1 downto 0);
56          Hab: out std_logic;
```

```
57          EoT: out std_logic
58          );
59  end component;
60
61  signal E: std_logic;
62  signal OPR, OPC: std_logic_vector(1 downto 0);
63  signal K: std_logic_vector(n-1 downto 0);
64  begin
65      K <= "00001000";
66      Block1: SPI_FSM_S port map(RST, CLK, STR1, E, Z, OPR, OPC, Hab1, EoT);
67      Block2: SPI_C81 generic map(n) port map(RST, CLK, OPR, Dreg, Din);
68      Block3: SPI_Compare generic map(n) port map(RST, CLK, OPC, K, E);
69  end Cuerpo;
```

El bloque 1 se sub-divide en 3 bloques más por lo que se describen a continuación.

Sub-Bloque 1 de Bloque 1 SPI_FSM_S

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_FSM_S is
 5      port(
 6          RST : in std_logic;
 7          CLK : in std_logic;
 8          STR1    : in std_logic;
 9          E   : in std_logic;
10          Z   : in std_logic;
11          OPR: out std_logic_vector(1 downto 0);
12          OPC: out std_logic_vector(1 downto 0);
13          Hab: out std_logic;
14          EoT: out std_logic
15          );
16  end SPI_FSM_S;
17
18  architecture Cuerpo of SPI_FSM_S is
19  signal Qn, Qp: std_logic_vector(3 downto 0);
20  signal Y: std_logic_vector(5 downto 0);
21  begin
22      process(Qp, STR1, E, Z)
23      begin
24          case Qp is
25              when "0000" =>
26                          if(STR1='1') then
27                              Qn <= "0001";
28                          else
29                              Qn <= "0000";
30                          end if;
31                          Y <= "001100";
32              when "0001" => Qn <= "0010";
33                          Y <= "000010";
34              when "0010" =>
35                          if(Z='1') then
36                              Qn <= "0011";
37                          else
38                              Qn <= "0010";
39                          end if;
40                          Y <= "000010";
41              when "0011" => Qn <= "0100";
42                          Y <= "010010";
43              when "0100" =>
44                          if(Z='1') then
45                              Qn <= "0101";
46                          else
```

```
                            Qn <= "0100";
                        end if;
                        Y <= "000010";
            when "0101" => Qn <= "0110";
                        Y <= "000110";
            when "0110" =>
                        if(E='1') then
                            Qn <= "1001";
                        else
                            Qn <= "0111";
                        end if;
                        Y <= "000010";
            when  "0111" =>
                        if(Z='1') then
                            Qn <= "1000";
                        else
                            Qn <= "0111";
                        end if;
                        Y <= "000010";
            when  "1000" => Qn <= "0100";
                        Y <= "100010";
            when "1001" => Qn <= "0000";
                        Y <= "000001";
            when others => Qn <= (others =>'0');
                        Y <= "111100";
        end case;
    end process;

    Qp <= (others =>'0') when RST='1' else Qn when rising_edge(CLK);

    OPR <= Y(5 downto 4);
    OPC <= Y(3 downto 2);
    Hab <= Y(1);
    EoT <= Y(0);
end Cuerpo;
```

Sub-Bloque 2 de Bloque 1 SPI_C81

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity SPI_C81 is
    generic (
    n: integer := 8
    );
    port(
        RST : in std_logic;
        CLK : in std_logic;
        En: in std_logic_vector(1 downto 0);
        D: in std_logic_vector(n-1 downto 0);
        Din: out std_logic
        );
end SPI_C81;

architecture Cuerpo of SPI_C81 is
signal Qp, Qn: std_logic_vector(n-1 downto 0);
begin
    process(En, Qp,D)
    begin
        case En is
            when "00" => Qn <= Qp;
            when "01" => Qn <= D;
            when "10" => Qn <= Qp(n-2 downto 0)&'0';
            when others => Qn <= (others =>'0');
        end case;
        Din <= Qp(n-1);
    end process;
```

```
31
32      process(CLK, RST)
33      begin
34          if(RST='1') then
35              Qp <= (others =>'0');
36          elsif(CLK'event and CLK='1') then
37              Qp <= Qn;
38          end if;
39      end process;
40  end Cuerpo;
```

Sub-Bloque 3 de Bloque 1 SPI_Comparador

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3  use IEEE.numeric_std.all;
 4
 5  entity SPI_Compare is
 6      generic (
 7      n: integer := 8
 8      );
 9      port(
10          RST : in std_logic;
11          CLK : in std_logic;
12          OPC : in std_logic_vector(1 downto 0);
13          K   : in std_logic_vector(n-1 downto 0);
14          E   : out std_logic
15          );
16  end SPI_Compare;
17
18  architecture Cuerpo of SPI_Compare is
19  signal Qp, Qn: std_logic_vector(n-1 downto 0);
20  begin
21      process(OPC, Qn, Qp,K)
22      begin
23          case OPC is
24              when "00" => Qn <= Qp;
25              when "01" => Qn <= std_logic_vector(unsigned(Qp)+1);
26              when others => Qn <= (others =>'0');
27          end case;
28      end process;
29
30      E <= '1' when Qp=K else '0';
31
32      process(CLK, RST)
33      begin
34          if(RST='1') then
35              Qp <= (others =>'0');
36          elsif(CLK'event and CLK='1') then
37              Qp <= Qn;
38          end if;
39      end process;
40
41  end Cuerpo;
```

Bloque 2 SPI_Receive

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity SPI_Receive is
5      generic (
6      n: integer := 8
7      );
```

```
 8      port(
 9          RST : in std_logic;
10          CLK : in std_logic;
11          STR2: in std_logic;
12          Dads: in std_logic;
13          Z   : in std_logic;
14          Hab2: out std_logic;
15          Dout: out std_logic_vector(n-1 downto 0);
16          EoR : out std_logic
17          );
18  end SPI_Receive;
19
20  architecture Cuerpo of SPI_Receive is
21  component SPI_C18
22      generic (
23      n: integer := 8
24      );
25      port(
26          RST : in std_logic;
27          CLK : in std_logic;
28          En: in std_logic_vector(1 downto 0);
29          D: in std_logic;
30          Dout: out std_logic_vector(n-1 downto 0)
31          );
32  end component;
33
34  component SPI_Compare
35      generic (
36      n: integer := 8
37      );
38      port(
39          RST : in std_logic;
40          CLK : in std_logic;
41          OPC: in std_logic_vector(1 downto 0);
42          K   : in std_logic_vector(n-1 downto 0);
43          E: out std_logic
44          );
45  end component;
46
47  component SPI_FSM_R
48      port(
49          RST : in std_logic;
50          CLK : in std_logic;
51          STR2: in std_logic;
52          E   : in std_logic;
53          Z   : in std_logic;
54          OPR: out std_logic_vector(1 downto 0);
55          OPC: out std_logic_vector(1 downto 0);
56          Hab: out std_logic;
57          EoR: out std_logic
58          );
59  end component;
60
61  signal E: std_logic;
62  signal OPR, OPC: std_logic_vector(1 downto 0);
63  signal K: std_logic_vector(n-1 downto 0);
64  begin
65      K <= "00001000";
66      Block1: SPI_FSM_R port map(RST, CLK, STR2, E, Z, OPR, OPC, Hab2, EoR);
67      Block2: SPI_C18 generic map(n) port map(RST, CLK, OPR, Dads, Dout);
68      Block3: SPI_Compare generic map(n) port map(RST, CLK, OPC, K, E);
69  end Cuerpo;
```

El bloque 2 se sub-divide en 3 bloques más por lo que se describen a continuación.

Sub-Bloque 1 de Bloque 2 SPI_FSM_R

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_FSM_R is
 5      port(
 6          RST : in std_logic;
 7          CLK : in std_logic;
 8          STR2: in std_logic;
 9          E   : in std_logic;
10          Z   : in std_logic;
11          OPR: out std_logic_vector(1 downto 0);
12          OPC: out std_logic_vector(1 downto 0);
13          Hab: out std_logic;
14          EoR: out std_logic
15          );
16  end SPI_FSM_R;
17
18  architecture Cuerpo of SPI_FSM_R is
19  signal Qn, Qp: std_logic_vector(3 downto 0);
20  signal Y: std_logic_vector(5 downto 0);
21  begin
22      process(Qp, STR2, E, Z)
23      begin
24          case Qp is
25              when "0000" =>
26                          if(STR2='1') then
27                              Qn <= "0001";
28                          else
29                              Qn <= "0000";
30                          end if;
31                          Y <= "001100";
32              when "0001" => Qn <= "0010";
33                          Y <= "110010";
34              when "0010" =>
35                          if(Z='1') then
36                              Qn <= "0011";
37                          else
38                              Qn <= "0010";
39                          end if;
40                          Y <= "000010";
41              when "0011" =>
42                          if(Z='1') then
43                              Qn <= "0100";
44                          else
45                              Qn <= "0011";
46                          end if;
47                          Y <= "000010";
48              when "0100" => Qn <= "0101";
49                          Y <= "010110";
50              when "0101" =>
51                          if(E='1') then
52                              Qn <= "0111";
53                          else
54                              Qn <= "0110";
55                          end if;
56                          Y <= "000010";
57              when  "0110" =>
58                          if(Z='1') then
59                              Qn <= "0011";
60                          else
61                              Qn <= "0110";
```

```
                        end if;
                        Y <= "000010";
            when "0111" => Qn <= "0000";
                        Y <= "000001";
            when others => Qn <= (others =>'0');
                        Y <= "001100";
        end case;
    end process;

    Qp <= (others =>'0') when RST='1' else Qn when rising_edge(CLK);

    OPR <= Y(5 downto 4);
    OPC <= Y(3 downto 2);
    Hab <= Y(1);
    EoR <= Y(0);
end Cuerpo;
```

Sub-Bloque 2 de Bloque 2 SPI_C18

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity SPI_C18 is
    generic (
    n: integer := 8
    );
    port(
        RST : in std_logic;
        CLK : in std_logic;
        En: in std_logic_vector(1 downto 0);
        D: in std_logic;
        Dout: out std_logic_vector(n-1 downto 0)
        );
end SPI_C18;

architecture Cuerpo of SPI_C18 is
signal Qp, Qn: std_logic_vector(n-1 downto 0);
begin
    process(En, Qp,D)
    begin
        case En is
            when "00" => Qn <= Qp;
            when "01" => Qn <= Qp(n-2 downto 0)&D;
            when others => Qn <= (others =>'0');
        end case;
        Dout <= Qp;
    end process;

    process(CLK, RST)
    begin
        if(RST='1') then
            Qp <= (others =>'0');
        elsif(CLK'event and CLK='1') then
            Qp <= Qn;
        end if;
    end process;
end Cuerpo;
```

Sub-Bloque 3 de Bloque 2 SPI_Comparador

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

```

```
 5  entity SPI_Compare is
 6      generic (
 7      n: integer := 8
 8      );
 9      port(
10          RST : in std_logic;
11          CLK : in std_logic;
12          OPC : in std_logic_vector(1 downto 0);
13          K   : in std_logic_vector(n-1 downto 0);
14          E   : out std_logic
15          );
16  end SPI_Compare;
17
18  architecture Cuerpo of SPI_Compare is
19  signal Qp, Qn: std_logic_vector(n-1 downto 0);
20  begin
21      process(OPC, Qn, Qp,K)
22      begin
23          case OPC is
24              when "00" => Qn <= Qp;
25              when "01" => Qn <= std_logic_vector(unsigned(Qp)+1);
26              when others => Qn <= (others =>'0');
27          end case;
28      end process;
29
30      E <= '1' when Qp=K else '0';
31
32      process(CLK, RST)
33      begin
34          if(RST='1') then
35              Qp <= (others =>'0');
36          elsif(CLK'event and CLK='1') then
37              Qp <= Qn;
38          end if;
39      end process;
40
41  end Cuerpo;
```

Bloque 3 SPI_CLK

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3  use IEEE.numeric_std.all;
 4
 5  entity SPI_CLK is
 6      generic (
 7      n: integer := 8
 8      );
 9      port(
10          RST : in std_logic;
11          CLK : in std_logic;
12          Hab: in std_logic;
13          K: in std_logic_vector(n-1 downto 0);
14          Z: out std_logic;
15          SCLK : out std_logic
16          );
17  end SPI_CLK;
18
19  architecture Cuerpo of SPI_CLK is
20  signal Qp, Qn: std_logic_vector(n-1 downto 0);
21  signal Q, En, Za, D: std_logic;
22  begin
23      En <= Hab and not(Za);
24      Qn <= (others => '0') when En='0' else std_logic_vector(unsigned(Qp)+1);
25      Qp <= (others => '0') when RST='1' else Qn when rising_edge(CLK);
26      Za <= '1' when Qp=K else '0';
```

```
27      Z  <= Za;
28
29      D <= Za xor Q;
30      Q <= '0' when RST='1' else D when rising_edge(CLK);
31      SCLK <= Q;
32  end Cuerpo;
```

Bloque 4 SPI_Timer_Reg

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_unsigned.all;
 4
 5  entity SPI_Timer_Reg is
 6      generic(n: integer:= 16);
 7      port(
 8      RST: in std_logic;
 9      CLK: in std_logic;
10      PLS: in std_logic;
11      Q:   out std_logic_vector(n-1 downto 0)
12      );
13  end SPI_Timer_Reg;
14
15  architecture timer of SPI_Timer_Reg is
16  signal qp,qn: std_logic_vector(n-1 downto 0);
17  begin
18
19      qp <= (others=> '0') when rst = '1' else qn when rising_edge(clk);
20      qn <= (qp + 1) when pls = '0' else (others=> '0');
21
22      q <= (others => '0') when rst = '1' else qp when (rising_edge(clk) and pls = '1');
23
24  end timer;
```

Bloque 5 SPI_FSM

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_FSM is
 5      port(
 6          RST : in std_logic;
 7          CLK : in std_logic;
 8          STR0: in std_logic;
 9          STRA: in std_logic;
10          EoT : in std_logic;
11          EoR : in std_logic;
12          E1  : in std_logic;
13          E2  : in std_logic;
14          E3  : in std_logic;
15          STR1: out std_logic;
16          STR2: out std_logic;
17          SHab: out std_logic;
18          CS  : out std_logic;
19          Ksel: out std_logic;
20          OPC : out std_logic_vector(1 downto 0);
21          OPR : out std_logic_vector(1 downto 0);
22          OPC2: out std_logic_vector(1 downto 0);
23          SEND: out std_logic
24          );
25  end SPI_FSM;
26
27  architecture Cuerpo of SPI_FSM is
28  signal Qn, Qp: std_logic_vector(4 downto 0);
29  begin
30      process(Qp, STR0, STRA, EoT, EoR, E1, E2, E3)
```

```
31      begin
32          OPC2 <= "00";
33          SEND <= '0';
34          case Qp is
35              when "00000" =>
36                          if(STR0='1') then
37                              Qn <= "00001";
38                          else
39                              Qn <= "00000";
40                          end if;
41                          CS <= '1'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "10"; OPR <= "10"; OPC2 <= "10";
42              when "00001" => Qn <= "00010";
43                          CS <= '0'; STR1 <= '1'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "00"; OPR <= "10";
44              when "00010" =>
45                          if(EoT='1') then
46                              Qn <= "00011";
47                          else
48                              Qn <= "00010";
49                          end if;
50                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "00"; OPR <= "10";
51              when "00011" =>     Qn <= "00100";
52                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "01"; OPR <= "10";
53              when "00100" =>
54                          if(E1='1') then
55                              Qn <= "00101";
56                          else
57                              Qn <= "00001";
58                          end if;
59                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "00"; OPR <= "10";
60              when "00101" =>
61                          if(STRA='1') then
62                              Qn <= "00110";
63                          else
64                              Qn <= "00101";
65                          end if;
66                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                            OPC <= "00"; OPR <= "00";
67              when "00110" => Qn <= "00111";
68                          CS <= '0'; STR1 <= '0'; STR2 <= '1'; SHab <= '1'; Ksel <= '0';
                            OPC <= "00"; OPR <= "00";
69              when "00111" =>
70                          if(EoR='1') then
71                              Qn <= "01000";
72                          else
73                              Qn <= "00111";
74                          end if;
75                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '0';
                            OPC <= "00"; OPR <= "00";
76              when "01000" => Qn <= "01001";
77                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '0';
                            OPC <= "00"; OPR <= "01";
78              when "01001" =>
79                          if(E2='1') then
80                              Qn <= "01010";
81                          else
82                              Qn <= "00110";
83                          end if;
84                          CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '0';
                            OPC <= "00"; OPR <= "00";
85              when "01010" => Qn <= "01011";
86                          CS <= '0'; STR1 <= '1'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                            OPC <= "00"; OPR <= "10";
87              when "01011" =>
88                          if(EoT='1') then
```

```
89                          Qn <= "01100";
90                      else
91                          Qn <= "01011";
92                      end if;
93                      CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                        OPC <= "00"; OPR <= "10";
94              when "01100" => Qn <= "01101";
95                      CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                        OPC <= "01"; OPR <= "10";
96              when "01101" =>
97                      if(E1='1') then
98                          Qn <= "01110";
99                      else
100                         Qn <= "01010";
101                     end if;
102                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                        OPC <= "00"; OPR <= "10";
103             when "01110" =>
104                     if(STRA='1') then
105                         Qn <= "01111";
106                     else
107                         Qn <= "01110";
108                     end if;
109                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "10";
110             --rdata
111             when "01111" => Qn <= "10000";
112                     CS <= '0'; STR1 <= '1'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00";
113             when "10000" =>
114                     if(EoT='1') then
115                         Qn <= "10001";
116                     else
117                         Qn <= "10000";
118                     end if;
119                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00";
120             --
121             when "10001" => Qn <= "10010";
122                     CS <= '0'; STR1 <= '0'; STR2 <= '1'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00";
123             when "10010" =>
124                     if(EoR='1') then
125                         Qn <= "10011";
126                     else
127                         Qn <= "10010";
128                     end if;
129                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00";
130             when "10011" => Qn <= "10100";
131                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "01";
132             when "10100" =>
133                     if(E2='1') then
134                         Qn <= "10101";
135                     else
136                         Qn <= "10001";
137                     end if;
138                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00";
139             when "10101" => Qn <= "10110";
140                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
                        OPC <= "00"; OPR <= "00"; OPC2 <= "01";
141             when "10110" =>
142                     if(E3='1') then
143                         Qn <= "00000";
144                     else
145                         Qn <= "01110";
146                     end if;
147                     CS <= '0'; STR1 <= '0'; STR2 <= '0'; SHab <= '1'; Ksel <= '1';
```

```
                            OPC <= "00"; OPR <= "00";
148             when others => Qn <= (others =>'0');
149                            CS <= '1'; STR1 <= '0'; STR2 <= '0'; SHab <= '0'; Ksel <= '0';
                               OPC <= "10"; OPR <= "10"; OPC2 <= "10";
150           end case;
151       end process;
152       Qp <= (others =>'0') when RST='1' else Qn when rising_edge(CLK);
153   end Cuerpo;
```

Bloque 6 SPI_Contador

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3  use IEEE.numeric_std.all;
 4
 5  entity SPI_Contador is
 6      generic (
 7      n: integer := 8
 8      );
 9      port(
10          RST : in std_logic;
11          CLK : in std_logic;
12          OPC : in std_logic_vector(1 downto 0);
13          K   : in std_logic_vector(n-1 downto 0);
14          D   : out std_logic_vector(n-1 downto 0);
15          E   : out std_logic
16          );
17  end SPI_Contador;
18
19  architecture Cuerpo of SPI_Contador is
20  signal Qp, Qn: std_logic_vector(n-1 downto 0);
21  begin
22      process(OPC, Qn, Qp,K)
23      begin
24          case OPC is
25              when "00" => Qn <= Qp;
26              when "01" => Qn <= std_logic_vector(unsigned(Qp)+1);
27              when others => Qn <= (others =>'0');
28          end case;
29          D <= Qp;
30      end process;
31
32      E <= '1' when Qp=K else '0';
33
34      process(CLK, RST)
35      begin
36          if(RST='1') then
37              Qp <= (others =>'0');
38          elsif(CLK'event and CLK='1') then
39              Qp <= Qn;
40          end if;
41      end process;
42
43  end Cuerpo;
```

Bloque 7 Comparador

```
1   library IEEE;
 2  use IEEE.std_logic_1164.all;
 3  use IEEE.numeric_std.all;
 4
 5  entity SPI_Compare is
 6      generic (
 7      n: integer := 8
```

```
 8      );
 9      port(
10          RST : in std_logic;
11          CLK : in std_logic;
12          OPC : in std_logic_vector(1 downto 0);
13          K   : in std_logic_vector(n-1 downto 0);
14          E   : out std_logic
15          );
16  end SPI_Compare;
17
18  architecture Cuerpo of SPI_Compare is
19  signal Qp, Qn: std_logic_vector(n-1 downto 0);
20  begin
21      process(OPC, Qn, Qp,K)
22      begin
23          case OPC is
24              when "00" => Qn <= Qp;
25              when "01" => Qn <= std_logic_vector(unsigned(Qp)+1);
26              when others => Qn <= (others =>'0');
27          end case;
28      end process;
29
30      E <= '1' when Qp=K else '0';
31
32      process(CLK, RST)
33      begin
34          if(RST='1') then
35              Qp <= (others =>'0');
36          elsif(CLK'event and CLK='1') then
37              Qp <= Qn;
38          end if;
39      end process;
40
41  end Cuerpo;
```

Bloque 8 y Bloque 12 emplean el mismo módulo “contador” que el bloque 6, por lo que no se escribirán los bloques.

Bloque 9 SPI_Lut

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_LUT is
 5      port(
 6          D   : in std_logic_vector(7 downto 0);
 7          VelMuestreo : in std_logic_vector(2 downto 0);
 8          Ch1g,Ch2g,Ch3g,Ch4g,Ch5g,Ch6g,Ch7g,Ch8g : in std_logic_vector(2 downto 0);
 9          Dl  : out std_logic_vector(7 downto 0)
10          );
11  end SPI_LUT;
12
13  architecture Cuerpo of SPI_LUT is
14  signal Qp, Qn: std_logic_vector(7 downto 0);
15  begin
16      Registro: process(D)
17      begin
18          case D is
19              when "00000000" => Dl <= "00010001"; --s_datac         --inicio primera fase
20              when "00000001" => Dl <= "01000011"; --opcode wreg reg3
21              when "00000010" => Dl <= "00000000"; --opcode wreg reg3
22              when "00000011" => Dl <= "11000000"; --reg3
23              when "00000100" => Dl <= "01000001"; --opcode wreg reg1
24              when "00000101" => Dl <= "00000001"; --opcode wreg reg1 a reg2
```

```
25              when "00000110" => Dl <= "10000"&VelMuestreo; --reg1
26              when "00000111" => Dl <= "00000000"; --reg2
27              when "00001000" => Dl <= "01000101"; --opcode wreg ChnSet 1
28              when "00001001" => Dl <= "00000111"; --opcode wreg ChnSet 1 a ChnSet 8
29              when "00001010" => Dl <= '0'&Ch1g&"0001"; --Chn1
30              when "00001011" => Dl <= '0'&Ch2g&"0001"; --Chn2
31              when "00001100" => Dl <= '0'&Ch3g&"0001"; --Chn3
32              when "00001101" => Dl <= '0'&Ch4g&"0001"; --Chn4
33              when "00001110" => Dl <= '0'&Ch5g&"0001"; --Chn5
34              when "00001111" => Dl <= '0'&Ch6g&"0001"; --Chn6
35              when "00010000" => Dl <= '0'&Ch7g&"0001"; --Chn7
36              when "00010001" => Dl <= '0'&Ch8g&"0001"; --Chn8
37              when "00010010" => Dl <= "00010000"; --r_datac        --fin primera fase
38              when "00010011" => Dl <= "00010001"; --s_datac        --inicio segunda fase
39              when "00010100" => Dl <= "01000010"; --opcode wreg reg2
40              when "00010101" => Dl <= "00000000"; --opcode wreg reg2
41              when "00010110" => Dl <= "00010000"; --reg2
42              when "00010111" => Dl <= "01000101"; --opcode wreg ChnSet 1
43              when "00011000" => Dl <= "00000111"; --opcode wreg ChnSet 1 a ChnSet 8
44              when "00011001" => Dl <= '0'&Ch1g&"0000"; --Chn1
45              when "00011010" => Dl <= '0'&Ch2g&"0000"; --Chn2
46              when "00011011" => Dl <= '0'&Ch3g&"0000"; --Chn3
47              when "00011100" => Dl <= '0'&Ch4g&"0000"; --Chn4
48              when "00011101" => Dl <= '0'&Ch5g&"0000"; --Chn5
49              when "00011110" => Dl <= '0'&Ch6g&"0000"; --Chn6
50              when "00011111" => Dl <= '0'&Ch7g&"0000"; --Chn7
51              when "00100000" => Dl <= '0'&Ch8g&"0000"; --Chn8
52              when "00100001" => Dl <= "00010010"; --r_data         --fin segunda fase
53              when others     => Dl <= "00010010";
54          end case;
55      end process;
56  end Cuerpo;
```

Bloque 10 SPI_FSM_M

```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity SPI_FSM_M is
 5      port(
 6          RST : in std_logic;
 7          CLK : in std_logic;
 8          SEND: in std_logic;
 9          EndADS: out std_logic;
10          EoR : in std_logic;
11          EoP : in std_logic;
12          EoM : in std_logic;
13          EoM2: in std_logic;
14          RDYR: in std_logic;
15          RDY : in std_logic;
16          CSR : out std_logic;
17          OPC3: out std_logic_vector(1 downto 0);
18          OPC4: out std_logic_vector(1 downto 0);
19          CSU : out std_logic;
20          WR  : out std_logic;
21          RD  : out std_logic;
22          MUX : out std_logic;
23          FSM : out std_logic;
24
25          Conf: in std_logic;
26          STR0: out std_logic;
27          STR0aux: in std_logic;
28          RDY_R: in std_logic;
29          CSU_R: out std_logic
30          );
31  end SPI_FSM_M;
```

```
32
33  architecture Cuerpo of SPI_FSM_M is
34  signal Qn, Qp: std_logic_vector(4 downto 0);
35  begin
36      process(Qp, EoR, EoP, EoM, EoM2, RDY, RDYR, RDY_R, Conf)
37      begin
38          ENDADS <= '0';
39          FSM <= '0';
40          CSU_R <= '0';
41          STR0 <= '0';
42          case Qp is
43          --    when "00000" => Qn <= "00001";  Estados de prueba para verificar USB
44          --    CSU_R <= '1'; CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <=
                  '0'; RD <= '0'; MUX <= '0';
45  --          when "00001" =>
46  --                      if(RDY_R='1') then
47  --                          Qn <= "00010";
48  --                      else
49  --                          Qn <= "00001";
50  --                      end if;
51  --                     CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                           RD <= '0'; MUX <= '0';
52              when "00010" => Qn <= "00011";
53                          CSU_R <= '1';   CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <=
                            '0'; WR <= '0'; RD <= '0'; MUX <= '0';
54              when "00011" =>
55                          if(RDY_R='1') then
56                              Qn <= "00100";
57                          else
58                              Qn <= "00011";
59                          end if;
60                          CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                            RD <= '0'; MUX <= '0';
61              when "00100" =>
62                          if(Conf='1') then
63                              Qn <= "00010";
64                          else
65                              STR0 <= STR0aux;
66                              Qn <= "00101";
67                          end if;
68                          CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                            RD <= '0'; MUX <= '0';
69              --Estados de guardar en memoria DRAM y envio por USB
70              --Guardar en DRAM
71              when "00101" =>
72                          if(EoR='1') then
73                              Qn <= "00110";
74                          else
75                              Qn <= "00101";
76                          end if;
77                          CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                            RD <= '0'; MUX <= '0';
78              when "00110" =>     Qn <= "00111";
79                          CSR <= '1'; OPC3 <= "01"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                            RD <= '0'; MUX <= '0';
80              when "00111" =>
81                          if(EoP='1') then
82                              Qn <= "01000";
83                          else
84                              Qn <= "00101";
85                          end if;
86                          CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                            RD <= '0'; MUX <= '0';
87              when "01000" => Qn <= "01001";
88                          CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '1';
                            RD <= '0'; MUX <= '0';
89              when "01001" =>
90                          if(EoM='1') then
91                              Qn <= "01010";
92                          else
```

```
 93                          Qn <= "00101";
 94                      end if;
 95                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0';
 96             --Reiniciar contador de DRAM
 97             when "01010" => --Qn <= "01011";
 98                      if(RDYR='1') then
 99                          Qn <= "01011";
100                      else
101                          Qn <= "01010";
102                      end if;
103                      CSR <= '0'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
104             --Leer dato de DRAM
105             when "01011" =>     Qn <= "01100";
106                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '1'; MUX <= '0'; FSM <= '1';
107             when "01100" =>
108                      if(RDYR='1') then
109                          Qn <= "01101";
110                      else
111                          Qn <= "01100";
112                      end if;
113                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0';   FSM <= '1';
114             --Enviar datos por USB
115             when  "01101" => Qn <= "01110";
116                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '1'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
117             when  "01110" =>
118                      if(RDY='1') then
119                          Qn <= "01111";
120                      else
121                          Qn <= "01110";
122                      end if;
123                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0';   FSM <= '1';
124             when  "01111" => Qn <= "10000";
125                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '1'; WR <= '0';
                         RD <= '0'; MUX <= '1';   FSM <= '1';
126             when  "10000" =>
127                      if(RDY='1') then
128                          Qn <= "10001";
129                      else
130                          Qn <= "10000";
131                      end if;
132                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '1'; FSM <= '1';
133             when  "10001" => Qn <= "10010";
134                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "01"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
135             when  "10010" =>
136                      if(EoM2='1') then
137                          Qn <= "10011";
138                      else
139                          Qn <= "01011";
140                      end if;
141                      CSR <= '1'; OPC3 <= "00"; OPC4 <= "00"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
142             when  "10011" => Qn <= "00010";
143                      CSR <= '1'; OPC3 <= "10"; OPC4 <= "10"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
144             when others => Qn <= "00010";
145                      CSR <= '0'; OPC3 <= "10"; OPC4 <= "10"; CSU <= '0'; WR <= '0';
                         RD <= '0'; MUX <= '0'; FSM <= '1';
146         end case;
147     end process;
148     Qp <= "00010" when RST='1' else Qn when rising_edge(CLK);
149 end Cuerpo;
```

Bloque 11 SPI_RegCh

```
library IEEE;
use IEEE.std_logic_1164.all;

entity SPI_RegCh is
    generic (
    n: integer := 8
    );
    port(
        RST : in std_logic;
        CLK : in std_logic;
        OPC : in std_logic_vector(1 downto 0);
        D   : in std_logic_vector(n-1 downto 0);
        Dl  : out std_logic_vector(27*n-1 downto 0)
        );
end SPI_RegCh;

architecture Cuerpo of SPI_RegCh is
signal Qp, Qn: std_logic_vector(27*n-1 downto 0);
begin
    process(OPC, Qp,D)
    begin
        case OPC is
            when "00" => Qn <= Qp;
            when "01" => Qn <= Qp(((27*n-1)-8) downto 0) & D;
            when others => Qn <= (others =>'0');
        end case;
        Dl <= Qp;
    end process;

    process(CLK, RST)
    begin
        if(RST='1') then
            Qp <= (others =>'0');
        elsif(CLK'event and CLK='1') then
            Qp <= Qn;
        end if;
    end process;
end Cuerpo;
```

Bloque 13 USB_v5_0 (propiedad del Grupo de Mecatrónica de la UAQ, contiene sub-bloques que pertenecen a este grupo por lo que no son propiedad intelectual del tesista, esta razón hace que no se enlisten los módulos dentro del bloque 13)

```
--------------------------------------------------------------------------------
--          Universidad Autónoma de Querétaro
--             Laboratorio de Mecatronica
--  Grupo de Sistemas Digitales en Logica Programable
--                       GSDLP
--------------------------------------------------------------------------------
-- Descripción
--
--  Unidad USB de propósito general, con 1 endpoint de 16 bytes de entrada (0x82) y
-- 1 endpoint de 32 bytes de salida (0x01).
--------------------------------------------------------------------------------
-- Módulo
--
--   USB_5_0 (RST,usbCLK48,usbVP,usbVM,usbRCV,usbVPO,usbVMO,usbOE,
--              CS,STALL,RDY,Di,Do);
--
--------------------------------------------------------------------------------
-- Terminales
```

```
19  --
20  -- Nombre     Tipo   Ancho    Descripcion
21  --  RST         E       1      Reset asíncrono
22  --  CLK48       E       1      Reloj de 48MHz
23  --  usbVP       E       1      Señal VP del transceptor
24  --  usbVM       E       1      Señal VM del transceptor
25  --  usbRCV      E       1      Señal RCV del transceptor
26  --  usbVPO      S       1      Señal VPO del transceptor
27  --  usbVMO      S       1      Señal VPM del transceptor
28  --  usbOE       S       1      Señal OE del transceptor
29  --  CS          E       2      Habilitación de endpoint
30  --  STALL       E       2      Condición de STALL para el endpoint
31  --  RDY         S       2      Endpoint listo
32  --  Di          E       16     Dato a enviar en el endpoint 0x82
33  --  Do          S       32     Dato recibido en el endpoint 0x01
34  ----------------------------------------------------------------------------
35  -- Historial
36  --
37  -- 19 Feb 2007    Luis Morales Velázquez (Autor)
38  ----------------------------------------------------------------------------
39
40  library IEEE;
41  use IEEE.STD_LOGIC_1164.all;
42
43  -- VID 9731 0x2603
44  -- PID 6530 0x1982
45
46  entity usbUSB_v5_0 is
47     generic(VID : integer := 0;
48             PID : integer := 0);
49      port (
50           RST: in STD_LOGIC;
51           CLK48: in STD_LOGIC;
52           SLOT: in STD_LOGIC_VECTOR (7 downto 0);
53           USB_DP: inout STD_LOGIC;
54           USB_DM: inout STD_LOGIC;
55           CS: in STD_LOGIC_VECTOR (2 downto 1);
56           STALL: in STD_LOGIC_VECTOR (2 downto 1);
57           RDY: out STD_LOGIC_VECTOR (2 downto 1);
58           Di : in STD_LOGIC_VECTOR(127 downto 0);
59           Do : out STD_LOGIC_VECTOR(255 downto 0)
60       );
61  end usbUSB_v5_0;
62
63  architecture usbUSB_v5_0 of usbUSB_v5_0 is
64
65      component UTMI
66      port (
67          RST: in STD_LOGIC;
68          CLK: in STD_LOGIC;
69          VP: in STD_LOGIC;
70          VM: in STD_LOGIC;
71          RCV: in STD_LOGIC;
72          VPO: out STD_LOGIC;
73          VMO: out STD_LOGIC;
74          OE: out STD_LOGIC;
75          RxActive: out STD_LOGIC;
76          RxValid: out STD_LOGIC;
77          RxError: out STD_LOGIC;
78          DataOut: out STD_LOGIC_VECTOR (7 DOWNTO 0);
79          TxValid: in STD_LOGIC;
80          TxReady: out STD_LOGIC;
81          DataIn: in STD_LOGIC_VECTOR (7 DOWNTO 0);
82          LineState: out STD_LOGIC_VECTOR (1 DOWNTO 0);
83          USBReset : in STD_LOGIC);
84      end component;
85
86      component usbSIE
87      generic(dVID : integer := 0;
```

```
88              dPID : integer := 0);
89       port(
90           RST : in STD_LOGIC;
91           CLK : in STD_LOGIC;
92           SLOT: in STD_LOGIC_VECTOR (7 downto 0);
93           RxActive : in STD_LOGIC;
94           RxValid : in STD_LOGIC;
95           RxError : in STD_LOGIC;
96           DataIn : in STD_LOGIC_VECTOR(7 downto 0);
97           TxValid : out STD_LOGIC;
98           TxReady : in STD_LOGIC;
99           DataOut : out STD_LOGIC_VECTOR(7 downto 0);
100          LineState : in STD_LOGIC_VECTOR(1 downto 0);
101          CS : in STD_LOGIC_VECTOR(2 downto 1);
102          STALL : in STD_LOGIC_VECTOR(2 downto 1);
103          RDY : out STD_LOGIC_VECTOR(2 downto 1);
104          EPD2 : in STD_LOGIC_VECTOR(127 downto 0);
105          EPD1 : out STD_LOGIC_VECTOR(255 downto 0));
106      end component;
107
108     component usbTransceiver
109       port(
110          DP : inout STD_LOGIC;
111          DM : inout STD_LOGIC;
112          usbVPO : in STD_LOGIC;
113          usbVMO : in STD_LOGIC;
114          usbOE : in STD_LOGIC;
115          usbVP : out STD_LOGIC;
116          usbVM : out STD_LOGIC;
117          usbRCV : out STD_LOGIC);
118     end component;
119
120      signal RxActive,RxValid,RxError,TxValid,TxReady : std_logic;
121      signal DataOut,DataIn : std_logic_vector(7 downto 0);
122      signal LineState : std_logic_vector(1 downto 0);
123     signal VP,VM,RCV,VPO,VMO,OE : std_logic;
124
125  begin
126
127      FUTMI:  UTMI port map (RST,CLK48,VP,VM,RCV,VPO,VMO,OE,
128                             RxActive,RxValid,RxError,DataOut,
129                                     TxValid,TxReady,DataIn,LineState,'0');
130
131      SIE:        usbSIE generic map (VID,PID) port map
(RST,CLK48,SLOT,RxActive,RxValid,RxError,DataOut,
132                                TxValid,TxReady,DataIn,LineState,CS,STALL,RDY,
133                                        Di,Do);
134
135     TRC:     usbTransceiver port map (USB_DP,USB_DM,VPO,VMO,OE,VP,VM,RCV);
136
137  end usbUSB_v5_0;
```

Bloque 14 dram_DRAM (propiedad del Grupo de Mecatrónica de la UAQ, contiene sub-bloques que pertenecen a este grupo por lo que no son propiedad intelectual del tesista, esta razón hace que no se enlisten los módulos dentro del bloque 14)

```
1  ----------------------------------------------------------------------------------
2  --
3  -- Universidad Autonoma de Queretaro
4  -- UAQ-SJR mechatronics group
5  --
6  -- Design      :
7  -- Author      : Luis Morales Velazquez
8  --
```

```
 9  ----------------------------------------------------------------------------------
10  --
11  -- Description : Dynamic memory handler in a fifo manner usign a 48 MHz clock.
12  --
13  ----------------------------------------------------------------------------------
14
15  library IEEE;
16  use IEEE.STD_LOGIC_1164.all;
17  use IEEE.NUMERIC_STD.all;
18
19  entity dramDRAM is
20       generic(n : integer := 256;
21                 m : integer := 4);
22       port(
23           RST : in STD_LOGIC;
24           CLK : in STD_LOGIC;
25           CS : in STD_LOGIC;
26           WR : in STD_LOGIC;
27           RD : in STD_LOGIC;
28           RDY : out STD_LOGIC;
29           K : in STD_LOGIC_VECTOR(m-1 downto 0);
30           Din : in STD_LOGIC_VECTOR(n-1 downto 0);
31           Dout : out STD_LOGIC_VECTOR(n-1 downto 0);
32           DRAM_CLK : out STD_LOGIC;
33           DRAM_CKE : out STD_LOGIC;
34           DRAM_CS : out STD_LOGIC;
35           DRAM_WE : out STD_LOGIC;
36           DRAM_CAS : out STD_LOGIC;
37           DRAM_RAS : out STD_LOGIC;
38           DRAM_DQML : out STD_LOGIC;
39           DRAM_DQMH : out STD_LOGIC;
40           DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
41           DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
42           DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0)
43           );
44  end dramDRAM;
45
46  architecture dramDRAM of dramDRAM is
47
48     component dramMT48LC16
49       port(
50           RST : in STD_LOGIC;
51           CLK : in STD_LOGIC;
52           CS : in STD_LOGIC;
53           WR : in STD_LOGIC;
54           RD : in STD_LOGIC;
55           RDY : out STD_LOGIC;
56           K : in STD_LOGIC_VECTOR(8 downto 0);
57           ADDR : in STD_LOGIC_VECTOR(23 downto 0);
58           Din : in STD_LOGIC_VECTOR(15 downto 0);
59           Dout : out STD_LOGIC_VECTOR(15 downto 0);
60           DRAM_CLK : out STD_LOGIC;
61           DRAM_CKE : out STD_LOGIC;
62           DRAM_CS : out STD_LOGIC;
63           DRAM_WE : out STD_LOGIC;
64           DRAM_CAS : out STD_LOGIC;
65           DRAM_RAS : out STD_LOGIC;
66           DRAM_DQML : out STD_LOGIC;
67           DRAM_DQMH : out STD_LOGIC;
68           DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
69           DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
70           DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0));
71     end component;
72
73     component dramUReg
74       generic(n : integer := 8;
75                 m : integer := 16);
76       port(
77           RST : in STD_LOGIC;
```

```
 78          CLK : in STD_LOGIC;
 79          OPR : in STD_LOGIC_VECTOR(1 downto 0);
 80          DPI : in STD_LOGIC_VECTOR(m-1 downto 0);
 81          DSI : in STD_LOGIC_VECTOR(n-1 downto 0);
 82          DPO : out STD_LOGIC_VECTOR(m-1 downto 0);
 83          DSO : out STD_LOGIC_VECTOR(n-1 downto 0));
 84    end component;
 85
 86    component dramGCount
 87      generic(n : integer := 19);
 88      port(
 89          RST : in STD_LOGIC;
 90          CLK : in STD_LOGIC;
 91          OPC : in STD_LOGIC_VECTOR(1 downto 0);
 92          K : in STD_LOGIC_VECTOR(n-1 downto 0);
 93          E : out STD_LOGIC);
 94    end component;
 95
 96    component dramACount
 97      generic(n : integer := 4;
 98              m : integer := 8);
 99      port(
100          RST : in STD_LOGIC;
101          CLK : in STD_LOGIC;
102          OPC : in STD_LOGIC_VECTOR(1 downto 0);
103          K : in STD_LOGIC_VECTOR(n-1 downto 0);
104          E : out STD_LOGIC;
105          Q : out STD_LOGIC_VECTOR(m-1 downto 0));
106    end component;
107
108    component dramFSM
109      port(
110          RST : in STD_LOGIC;
111          CLK : in STD_LOGIC;
112          CS : in STD_LOGIC;
113          WR : in STD_LOGIC;
114          RD : in STD_LOGIC;
115          RDY : out STD_LOGIC;
116          Z : in STD_LOGIC;
117          E : in STD_LOGIC;
118          CSM : out STD_LOGIC;
119          WRM : out STD_LOGIC;
120          RDM : out STD_LOGIC;
121          RYM : in STD_LOGIC;
122          LDA : out STD_LOGIC;
123          SEL : out STD_LOGIC;
124          OPA : out STD_LOGIC_VECTOR(1 downto 0);
125          OPC : out STD_LOGIC_VECTOR(1 downto 0));
126    end component;
127
128    component dramGReg
129     generic(n : integer := 8);
130      port(
131          RST : in STD_LOGIC;
132          CLK : in STD_LOGIC;
133          LDR : in STD_LOGIC;
134          D : in STD_LOGIC_VECTOR(n-1 downto 0);
135          Q : out STD_LOGIC_VECTOR(n-1 downto 0));
136    end component;
137
138    signal CSM,WRM,RDM,RYM,E,Z,LDA,SEL : std_logic;
139    signal OPC,OPA                     : std_logic_vector(1 downto 0);
140    signal KP                          : std_logic_vector(8 downto 0);
141    signal Di,Do                       : std_logic_vector(15 downto 0);
142    signal ADD,ADR,ADDR                : std_logic_vector(23 downto 0);
143
144 begin
145
146    KP <= std_logic_vector(to_unsigned(n/16-1,9));
147
```

```
148     Mux: process(SEL,ADD,ADR)
149     begin
150        if SEL = '1' then
151           ADDR <= ADD;
152        else
153           ADDR <= ADR;
154        end if;
155     end process Mux;
156
157     Reg:     dramUReg generic map (16,n) port map(RST,CLK,OPC,Din,Do,Dout,Di);
158
159     AGN:     dramACount generic map (9,24) port map (RST,CLK,OPA,"111111111",E,ADD);
160
161     AdReg:   dramGReg generic map (24) port map (RST,CLK,LDA,ADD,ADR);
162
163     Count:   dramGCount generic map (m) port map (RST,CLK,OPC,K,Z);
164
165     SDRAM:   dramMT48LC16 port map (RST,CLK,CSM,WRM,RDM,RYM,KP,ADDR,Di,Do,
166              DRAM_CLK,DRAM_CKE,DRAM_CS,DRAM_WE,DRAM_CAS,DRAM_RAS,DRAM_DQML,DRAM_DQMH,
167              DRAM_BA,DRAM_A,DRAM_DQ);
168
169     Control: dramFSM port map(RST,CLK,CS,WR,RD,RDY,Z,E,CSM,WRM,RDM,RYM,LDA,SEL,OPA,OPC);
170
171  end dramDRAM;
```

Bloque 15 Display7seg

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity Display7seg is
 5      port(
 6      clk :   in  std_logic;
 7      rst :   in  std_logic;
 8      b0,b1,b2,b3 :   in  std_logic_vector(3 downto 0);
 9      led :   out std_logic_vector(3 downto 0);--asignacion de los display
10      dis :   out std_logic_vector(7 downto 0)--asignacion de los 7 segmentos
11      );
12  end Display7seg;
13
14  architecture principal of Display7seg is
15  component siete
16      port(
17      a   :   in  std_logic_vector(3 downto 0);
18      l   :   out std_logic_vector(7 downto 0)
19      );
20  end component;
21  component contador
22  generic(
23  n: integer:= 16
24  );
25  port(
26  clk :   in  std_logic;
27  rst :   in  std_logic;
28  f   :   in  std_logic_vector(n-1 downto 0);
29  z   :   out std_logic
30  );
31  end component;
32  component maq_dis
33      port(
34      clk :   in  std_logic;
35      rst :   in  std_logic;
36      z   :   in  std_logic;
37      ss  :   out std_logic_vector(1 downto 0)
38      );
39  end component;
```

```
component demux
    port(
    sel :   in  std_logic_vector(1 downto 0);
    x   :   out std_logic_Vector(3 downto 0)
    );
end component;

component mux is
    generic(
    m: integer:= 4
    );
    port(
    a1,a2,a3,a4 :   in   std_logic_vector(m-1 downto 0);
    sel         :   in   std_logic_vector(1 downto 0);
    x           :   out  std_logic_vector(m-1 downto 0)
    );
end component;

signal ff : std_logic_vector(15 downto 0);
signal zz : std_logic;
signal sss: std_logic_vector(1 downto 0);
signal dat: std_logic_vector(3 downto 0);
begin
    ff <= "1100001101010000";

    blk1:  siete  port map(dat,dis);
    blk2:  contador  port map (clk,rst,ff,zz);
    blk3:  maq_dis port map(clk,rst,zz,sss);
    blk4:  demux port map (sss,led);
    blk5:  mux port map (b0,b1,b2,b3,sss,dat);
end principal;
```

El bloque 15 se sub-divide en 5 bloques más por lo que se describen a continuación.

Sub-Bloque 1 de Bloque 15 Siete

```
library ieee;
use ieee.std_logic_1164.all;

entity siete is
    port(
    a   :   in  std_logic_vector(3 downto 0);
    l   :   out std_logic_vector(7 downto 0)
    );
end siete;

architecture siete of siete is
begin
    process(a)
    begin
        case a is
            when "0000"=> l<="10000001";--0
            when "0001"=> l<="11001111";--1
            when "0010"=> l<="10010010";--2
            when "0011"=> l<="10000110";--3
            when "0100"=> l<="11001100";--4
            when "0101"=> l<="10100100";--5
            when "0110"=> l<="10100000";--6
            when "0111"=> l<="10001111";--7
            when "1000"=> l<="10000000";--8
            when "1001"=> l<="10001100";--9
            when "1010"=> l<="10001000";--A
            when "1011"=> l<="11100000";--B
            when "1100"=> l<="10110001";--C
            when "1101"=> l<="11000010";--D
            when "1110"=> l<="10110000";--E
```

```
31              when others=> l<="10111000";--F
32          end case;
33      end process;
34  end siete;
```

Sub-Bloque 2 de Bloque 15 Contador

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_arith.all;
 4  use ieee.std_logic_unsigned.all;
 5
 6  entity contador  is
 7  generic(
 8  n: integer:= 16
 9  );
10  port(
11  clk :   in  std_logic;
12  rst :   in  std_logic;
13  f   :   in  std_logic_vector(n-1 downto 0);
14  z   :   out std_logic
15  );
16  end contador;
17
18  architecture contador of contador is
19  signal qp,qn,a: std_logic_vector(n-1 downto 0);
20  signal zz: std_logic;
21  begin
22      a <= (others => '0');
23      combinacional: process(qp,zz,a,f)
24      begin
25      if(qp = a) then
26          zz <= '1';
27      else
28          zz <= '0';
29      end if;
30      if(zz = '0' ) then
31          qn <= qp - 1;
32      else
33          qn <= f;
34      end if;
35      z <= zz;
36      end process combinacional;
37
38      secuencial: process(clk,rst)
39      begin
40          if (rst = '1') then
41              qp <=(others => '0');
42          elsif(clk'event and clk = '1') then
43              qp <= qn;
44          end if;
45          end process secuencial;
46      end contador;
47
```

Sub-Bloque 3 de Bloque 15 Maq_Dis

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity maq_dis is
5      port(
6      clk :   in  std_logic;
```

```
 7      rst :   in  std_logic;
 8      z   :   in  std_logic;
 9      ss  :   out std_logic_vector(1 downto 0)
10      );
11  end maq_dis;
12
13  architecture maq_dis of maq_dis is
14  signal qp, qn   : std_logic_vector(1 downto 0);
15  begin
16      combinacional: process(qp,z)
17      begin
18          case qp is
19              when "00" =>
20              if(z = '0') then
21                  qn <= qp;
22              else
23                  qn <= "01";
24              end if;
25                  ss <= "00";
26              when "01" =>
27              if(z = '0') then
28                  qn <= qp;
29              else
30                  qn <= "10";
31              end if;
32              ss <= "01";
33              when "10" =>
34              if(z = '0') then
35                  qn <= qp;
36              else
37                  qn <= "11";
38              end if;
39              ss <= "10";
40              when others =>
41              if(z = '0') then
42                  qn <= qp;
43              else
44                  qn <= "00";
45              end if;
46              ss <= "11";
47              end case;
48              end process combinacional;
49
50              secuencial: process(clk,rst)
51              begin
52                  if(rst = '1') then
53                      qp <= (others => '0');
54                  elsif(clk'event and clk = '1') then
55                      qp <= qn;
56                  end if;
57                  end process secuencial;
58              end maq_dis;
```

Sub-Bloque 4 de Bloque 15 Demux

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity demux is
 5      port(
 6      sel :   in  std_logic_vector(1 downto 0);
 7      x   :   out std_logic_vector(3 downto 0)
 8      );
 9      end demux;
10
11  architecture demux of demux is
12  begin
13      process(sel)
```

```
14      begin
15          x<=(others => '1');
16          case sel is
17          when "00" =>
18          x <= "0001";
19          when "01" =>
20          x <= "0010";
21          when "10" =>
22          x <= "0100";
23          when others =>
24          x <= "1000";
25      end case;
26  end process;
27  end demux;
28
```

Sub-Bloque 4 de Bloque 15 Mux

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity mux is
 5      generic(
 6      m: integer:= 4
 7      );
 8      port(
 9      a1,a2,a3,a4 :   in   std_logic_vector(m-1 downto 0);
10      sel         :   in   std_logic_vector(1 downto 0);
11      x           :   out  std_logic_vector(m-1 downto 0)
12      );
13  end mux;
14
15  architecture mux of mux is
16  begin
17      process(sel,a1,a2,a3,a4)
18      begin
19          case sel is
20              when "00"=>
21              x <= a1;
22              when "01"=>
23              x <= a2;
24              when "10"=>
25              x <= a3;
26              when others =>
27              x <= a4;
28          end case;
29      end process;
30  end mux;
31
```

En el siguiente listado se encuentran las direcciones asignadas mediante el modulo principal Driver_ADS1298 y la tarjeta basada en FPGA “UPDSH” (Archivo UCF).

```
1
2  # PlanAhead Generated physical constraints
3
4  # Tarjeta Adquisicion ECG
5
6  NET "CLK" LOC = B8;
```

```
7  NET "RST" LOC = A15;

 8  NET "RST" PULLDOWN;
 9
10  # I/O Extension
11
12  #NET "" LOC = N4;
13  #NET "" LOC = M4;
14  #NET "" LOC = V2;
15  #NET "" LOC = M1;
16  #NET "" LOC = N1;
17  #NET "" LOC = J17;
18  #NET "" LOC = M3;
19  #NET "" LOC = N5;
20  #NET "" LOC = P8;
21
22  #NET "" LOC = P7;
23  #NET "" LOC = P6;
24  #NET "" LOC = R8;
25  #NET "" LOC = N8;
26  #NET "" LOC = N11;
27  #NET "" LOC = G5;
28  #NET "" LOC = G3;
29  #NET "" LOC = H3;
30  #NET "" LOC = J4;
31
32  # Señales de Convertidor ADS1298
33
34  NET "GPIO4" LOC = J5;
35  NET "GPIO3" LOC = D5;
36  NET "GPIO2" LOC = D7;
37  NET "GPIO1" LOC = D10;
38      ##Señales de Iniciacion de ADS
39  #NET "CLKads" LOC = H14;
40  NET "CLKSel" LOC = H4;
41  NET "Start" LOC = F14;
42  NET "Reset" LOC = G16;
43  NET "PWDN" LOC = H16;
44      ##Señales de Protocolo SPI
45  NET "SCLK" LOC = E10;
46  NET "CS" LOC = D14;
47  NET "Dads" LOC = D9;
48  NET "Din" LOC = G14;
49  NET "DRDY" LOC = G4;
50
51  # Display de 7 Segmentos 4D
52
53  NET "D7S[0]" LOC = T14; #G
54  NET "D7S[1]" LOC = K14;
55  NET "D7S[2]" LOC = M16;
56  NET "D7S[3]" LOC = M14;
57  NET "D7S[4]" LOC = T16;
58  NET "D7S[5]" LOC = J16;
59  NET "D7S[6]" LOC = L16; #A
60  NET "D7S[7]" LOC = R15; #DP
61
62  NET "AN[0]" LOC = J15;
63  NET "AN[1]" LOC = M15;
64  NET "AN[2]" LOC = L15;
65  NET "AN[3]" LOC = N7;
66
67  # Memoria Dinamica
68
69  NET "DRAM_CLK" LOC =D1;
70  NET "DRAM_CKE" LOC =C2;
71  NET "DRAM_CS" LOC = E13;
72  NET "DRAM_WE" LOC = B10;
73  NET "DRAM_CAS" LOC =A11;
74  NET "DRAM_RAS" LOC = B11;
75  NET "DRAM_DQML" LOC = A10;
76  NET "DRAM_DQMH" LOC = D2;
77  NET "DRAM_BA[0]" LOC = A13;
78  NET "DRAM_BA[1]" LOC = B13;
79  NET "DRAM_A[0]" LOC =B14;
80  NET "DRAM_A[1]" LOC =B16;
81  NET "DRAM_A[2]" LOC =A16;
82  NET "DRAM_A[3]" LOC =C17;
83  NET "DRAM_A[4]" LOC =H17;
84  NET "DRAM_A[5]" LOC =F15;
85  NET "DRAM_A[6]" LOC =F18;
86  NET "DRAM_A[7]" LOC =F17;
87  NET "DRAM_A[8]" LOC =D16;
88  NET "DRAM_A[9]" LOC =D17;
89  NET "DRAM_A[10]" LOC =A14;
90  NET "DRAM_A[11]" LOC =C18;
91  NET "DRAM_A[12]" LOC =C1;
92  NET "DRAM_DQ[0]" LOC =C3;
93  NET "DRAM_DQ[1]" LOC =A4;
94  NET "DRAM_DQ[2]" LOC =B4;
95  NET "DRAM_DQ[3]" LOC =A6;
96  NET "DRAM_DQ[4]" LOC =B6;
97  NET "DRAM_DQ[5]" LOC =C7;
98  NET "DRAM_DQ[6]" LOC =A8;
99  NET "DRAM_DQ[7]" LOC =C9;
100 NET "DRAM_DQ[8]" LOC =E1;
101 NET "DRAM_DQ[9]" LOC =E2;
102 NET "DRAM_DQ[10]" LOC =F1;
103 NET "DRAM_DQ[11]" LOC =F2;
104 NET "DRAM_DQ[12]" LOC =H1;
105 NET "DRAM_DQ[13]" LOC =H2;
106 NET "DRAM_DQ[14]" LOC =J1;
107 NET "DRAM_DQ[15]" LOC =J2;
108
109 # Puerto USB
110
111 NET "USB_DM" LOC = L3;
112 NET "USB_DP" LOC = L1;
113 NET "USB_DP" PULLUP;
114
115 # Switches
116
117 #NET "STR0" LOC = G15;
118 #NET "Send" LOC = H15;
119
120 # Leds
121
122 NET "EndADS" LOC = J14;
123 NET "Led2" LOC = D11;
```

7.3 Interfaz de Usuario

La interfaz de usuario se creó en el software de programación Code Blocks con un lenguaje de programación C++ e interfaz gráfica Gtk. En la Figura 7-2 se enmarca el software de programación y módulos empleados.

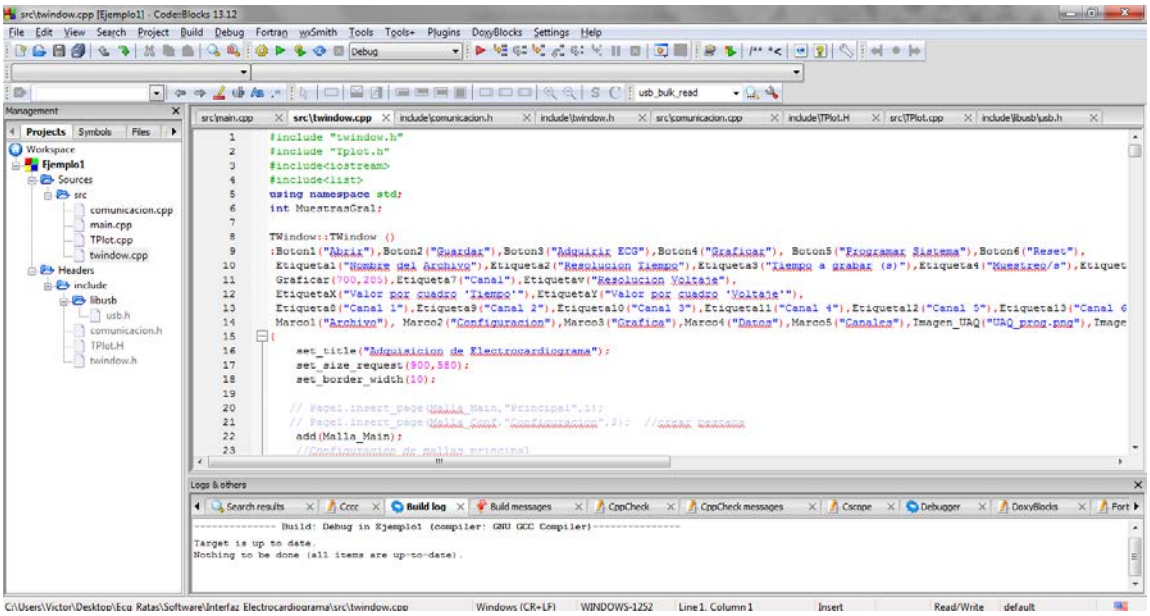


Figura 7-5.- Interfaz de programación Code Blocks

El primer código fuente es el "twindow.cpp" el cual es el modulo principal.

```
#include "twindow.h"
#include "Tplot.h"
#include<iostream>
#include<list>
using namespace std;
int MuestrasGral;

TWindow::TWindow ()
:Boton1("Abrir"),Bot/**<  */on2("Guardar"),Boton3("Adquirir ECG"),Boton4("Graficar"),
 Boton5("Programar Sistema"),Boton6("Reset"),
 Etiqueta1("Nombre del Archivo"),Etiqueta2("Resolucion Tiempo"),Etiqueta3("Tiempo a
 grabar (s)"),Etiqueta4("Muestreo/s"),Etiqueta5("La ganancia de los canales se
 colocan"),Etiqueta6("en la seccion continua"),
 Graficar(700,285),Etiqueta7("Canal"),Etiquetav("Resolucion Voltaje"),
 EtiquetaX("Valor por cuadro 'Tiempo'"),EtiquetaY("Valor por cuadro 'Voltaje'"),
 Etiqueta8("Canal 1"),Etiqueta9("Canal 2"),Etiqueta10("Canal 3"),Etiqueta11("Canal
 4"),Etiqueta12("Canal 5"),Etiqueta13("Canal 6"),Etiqueta14("Canal
 7"),Etiqueta15("Canal 8"),
 Marco1("Archivo"),Marco2("Configuracion"),Marco3("Grafica"),Marco4("Datos"),
 Marco5("Canales"),Imagen_UAQ("UAQ_prog.png"),Imagen_UAQ2("UAQ2.png")
{
    set_title("Adquisicion de Electrocardiograma");
    set_size_request(900,580);
    set_border_width(10);
```

```
19
20      // Page1.insert_page(Malla_Main,"Principal",1);
21      // Page1.insert_page(Malla_Conf,"Configuracion",2);  //crear pestana
22       add(Malla_Main);
23       //Configuracion de mallas principal
24       //Malla Main
25       Malla_Main.set_homogeneous(TRUE);
26       Malla_Main.resize(480,800);              // y luego en x
27       // Marco1
28       Malla_Main.attach(Marco1,0,500,0,190);
29       Marco1.add(Malla1);
30
31       Malla1.set_homogeneous(TRUE);
32       Malla1.resize(190,500);              // y luego en x
33
34       Malla1.attach(Boton1,10,120,0,35);      // xi xf yi yf
35       Malla1.attach(Boton2,130,240,0,35);
36       Malla1.attach(Boton3,250,360,0,35);
37       Malla1.attach(Boton4,370,480,0,35);
38       Malla1.attach(Entrada1,20,240,80,100);
39       Malla1.attach(Etiqueta1,20,225,50,70);
40       Malla1.attach(Etiqueta7,340,380,50,70);
41       Malla1.attach(CBT4,340,385,80,100);
42       // Marco2
43       Malla_Main.attach(Marco2,505,735,0,190);  //
44       Marco2.add(Malla2);
45
46       Malla2.set_homogeneous(TRUE);
47       Malla2.resize(190,230);              // y luego en x
48
49       Malla2.attach(Boton5,0,150,0,35);
50       Malla2.attach(Boton6,170,215,0,35);
51       Malla2.attach(Etiqueta3,10,115,50,65);
52       Malla2.attach(Etiqueta4,10,70,80,95);
53       Malla2.attach(CBT1,80,140,80,99);
54       Malla2.attach(Etiqueta5,10,230,120,135);
55      // Malla2.attach(CBT2,50,90,108,127);
56       Malla2.attach(Etiqueta6,10,230,140,155);
57      // Malla2.attach(CBT3,170,210,108,127);
58       Malla2.attach(Entrada2,125,160,50,65);
59       // Marco5
60       Malla_Main.attach(Marco5,738,800,0,190);  //
61       Marco5.add(Malla5);
62
63       Malla5.set_homogeneous(TRUE);
64       Malla5.resize(190,60);              // y luego en x
65
66       Malla5.attach(Etiqueta8,10,35,10,20);
67       Malla5.attach(Etiqueta9,10,35,30,40);
68       Malla5.attach(Etiqueta10,10,35,50,60);
69       Malla5.attach(Etiqueta11,10,35,70,80);
70       Malla5.attach(Etiqueta12,10,35,90,100);
71       Malla5.attach(Etiqueta13,10,35,110,120);
72       Malla5.attach(Etiqueta14,10,35,130,140);
73       Malla5.attach(Etiqueta15,10,35,150,160);
74
75       Malla5.attach(Chg1,40,65,5,22);
76       Malla5.attach(Chg2,40,65,25,42);
77       Malla5.attach(Chg3,40,65,45,62);
78       Malla5.attach(Chg4,40,65,65,82);
79       Malla5.attach(Chg5,40,65,85,102);
80       Malla5.attach(Chg6,40,65,105,122);
81       Malla5.attach(Chg7,40,65,125,142);
82       Malla5.attach(Chg8,40,65,145,162);
83       // Marco3
84       Malla_Main.attach(Marco3,0,700,195,480);  // xi xf yi yf  Grafica en x = 700,y= 285
 85       Marco3.add(Graficar);                                  152       CBT4.append_text("8");
 86       // Marco4                                              153       CBT4.set_active(7);
 87       Malla_Main.attach(Marco4,705,800,195,480);  //        154
 88       Marco4.add(Malla4);                                    155       Chg1.append_text("6");
```

```
 89
 90      Malla4.set_homogeneous(TRUE);
 91      Malla4.resize(285,95);    // y luego en x
 92
 93      Malla4.attach(Etiqueta2,0,80,0,20);
 94      Malla4.attach(hscale,5,90,30,70);
 95      Malla4.attach(Etiquetav,0,80,80,100);
 96      Malla4.attach(hscalev,5,90,110,150);
 97      Malla4.attach(EtiquetaX,0,95,160,170);
 98      Malla4.attach(ValorX,0,95,185,195);
 99      Malla4.attach(EtiquetaY,0,95,200,220);
100      Malla4.attach(ValorY,0,95,230,240);
101      Malla4.attach(Imagen_UAQ,0,42,258,300);
102      Malla4.attach(Imagen_UAQ2,64,95,257,300);
103      // Crear cajas horizontales en caja vertical
104  /*    m_VBox.pack_start(m_HBox1);
105      m_VBox.pack_start(m_HBox2);
106      m_VBox.pack_start(m_HBox3);
107      // Crear tres botones en caja 1 horizontal
108      m_HBox1.pack_start(Boton1);
109      m_HBox1.pack_start(Boton2);
110      m_HBox1.pack_start(Boton3);
111
112      m_HBox2.pack_start(Entrada1);
113      Entrada1.set_text("Prueba de Entrada");
114      m_HBox3.pack_start(Etiqueta1);
115      Etiqueta1.set_text("Prueba etiqueta");
116     */
117      Entrada1.set_text("Archivo.txt");
118      Entrada2.set_text("4");
119      CBT1.append_text("8ksps");
120      CBT1.append_text("4ksps");
121      CBT1.append_text("2ksps");
122      CBT1.append_text("1ksps");
123      CBT1.append_text("500sps");
124      CBT1.set_active(0);
125
126   /*   CBT2.append_text("1");
127      CBT2.append_text("2");
128      CBT2.append_text("3");
129      CBT2.append_text("4");
130      CBT2.append_text("5");
131      CBT2.append_text("6");
132      CBT2.append_text("7");
133      CBT2.append_text("8");
134      CBT2.set_active(7);*/
135
136     /* CBT3.append_text("6");
137      CBT3.append_text("1");
138      CBT3.append_text("2");
139      CBT3.append_text("3");
140      CBT3.append_text("4");
141      CBT3.append_text("8");
142      CBT3.append_text("12");
143      CBT3.set_active(0);*/
144
145      CBT4.append_text("1");
146      CBT4.append_text("2");
147      CBT4.append_text("3");
148      CBT4.append_text("4");
149      CBT4.append_text("5");
150      CBT4.append_text("6");
151      CBT4.append_text("7");

219     Chg8.append_text("1");
220     Chg8.append_text("2");
221     Chg8.append_text("3");
222     Chg8.append_text("4");
223     Chg8.append_text("8");
```

```
156      Chg1.append_text("1");
157      Chg1.append_text("2");
158      Chg1.append_text("3");
159      Chg1.append_text("4");
160      Chg1.append_text("8");
161      Chg1.append_text("12");
162      Chg1.set_active(0);
163
164      Chg2.append_text("6");
165      Chg2.append_text("1");
166      Chg2.append_text("2");
167      Chg2.append_text("3");
168      Chg2.append_text("4");
169      Chg2.append_text("8");
170      Chg2.append_text("12");
171      Chg2.set_active(0);
172
173      Chg3.append_text("6");
174      Chg3.append_text("1");
175      Chg3.append_text("2");
176      Chg3.append_text("3");
177      Chg3.append_text("4");
178      Chg3.append_text("8");
179      Chg3.append_text("12");
180      Chg3.set_active(0);
181
182      Chg4.append_text("6");
183      Chg4.append_text("1");
184      Chg4.append_text("2");
185      Chg4.append_text("3");
186      Chg4.append_text("4");
187      Chg4.append_text("8");
188      Chg4.append_text("12");
189      Chg4.set_active(0);
190
191      Chg5.append_text("6");
192      Chg5.append_text("1");
193      Chg5.append_text("2");
194      Chg5.append_text("3");
195      Chg5.append_text("4");
196      Chg5.append_text("8");
197      Chg5.append_text("12");
198      Chg5.set_active(0);
199
200      Chg6.append_text("6");
201      Chg6.append_text("1");
202      Chg6.append_text("2");
203      Chg6.append_text("3");
204      Chg6.append_text("4");
205      Chg6.append_text("8");
206      Chg6.append_text("12");
207      Chg6.set_active(0);
208
209      Chg7.append_text("6");
210      Chg7.append_text("1");
211      Chg7.append_text("2");
212      Chg7.append_text("3");
213      Chg7.append_text("4");
214      Chg7.append_text("8");
215      Chg7.append_text("12");
216      Chg7.set_active(0);
217
218      Chg8.append_text("6");
```

```
224      Chg8.append_text("12");
225      Chg8.set_active(0);
226      //Y=CBT1.get_active_row_number();
227      CBT1.signal_changed().connect(sigc::mem_fun(*this,&TWindow::CBT1muestreo));
228  //    CBT3.signal_changed().connect(sigc::mem_fun(*this,&TWindow::CBT3g));
229
230      Boton1.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button1_clicked));
231      Boton2.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button2_clicked));
232      Boton3.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button3_clicked));
233      Boton4.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button4_clicked));
234      Boton5.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button5_clicked));
235      Boton6.signal_clicked().connect(sigc::mem_fun(*this,&TWindow::on_button6_clicked));
236
237      hscale.signal_value_changed().connect(sigc::mem_fun
         (*this,&TWindow::on_hscale_change));
238      hscalev.signal_value_changed().connect(sigc::mem_fun
         (*this,&TWindow::on_hscalev_change));
239
240      //CBT7.signal_changed().connect(sigc::mem_fun(*this,&TWindow::on_CBT_clicked7));
241      //HScrollbar = Gtk.HScrollbar(None);
242
243      hscale.set_range(0.1,15);
244      hscale.set_digits(1);
245      hscale.set_value(2);
246      hscale.set_increments(0.1,10);
247      /** En muestras**/
248      //hscale.set_range(500,30000);
249  //    hscale.set_digits(0);
250  //    hscale.set_value(5000);
251  //    hscale.set_increments(500,1000000);
252
253      hscalev.set_range(0.5,3.0);
254      hscalev.set_digits(1);
255      hscalev.set_value(2.5);
256      hscalev.set_increments(0.1,10);
257
258      Boton2.set_sensitive(FALSE);
259      Boton3.set_sensitive(FALSE);
260      Boton4.set_sensitive(FALSE);
261
262      int Muestreoaux,VelM;
263      Muestreoaux=CBT1.get_active_row_number();
264      //std::cout<<Tiempoaux<<std::endl;
265      //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
266      if(Muestreoaux==0) VelM=8000;
267      if(Muestreoaux==1) VelM=4000;
268      if(Muestreoaux==2) VelM=2000;
269      if(Muestreoaux==3) VelM=1000;
270      if(Muestreoaux==4) VelM=500;
271      VelocidadMuestreo=VelM;
272
273      Graficar.inicio(0);
274
275      if(!dev1.usbInit())
276      {
277          Gtk::MessageDialog dialog(*this, "Error de configuracion");
278          dialog.set_secondary_text("No se encuentra dispositivo conectado");
279          dialog.run();
280      }
281      else
282      {
283          Gtk::MessageDialog dialog(*this, "Dispositivo conectado");
284          dialog.set_secondary_text("Sistema de adquisicion identificado.");
285          dialog.run();
286      }
287
288      show_all_children();
289      Page1.show();
290      Page1.show_all_children(TRUE);
291  }
```

```
292
293 TWindow::~TWindow()
294 {
295     dev1.usbTerminate();
296 }
297 // Abrir Archivo
298 void TWindow::on_button1_clicked()
299 {
300     FILE *DocTXT;
301     Glib::ustring archivo_text;
302     const char *archivo;
303
304     archivo_text = Entrada1.get_text();
305     archivo = archivo_text.c_str();
306
307     DocTXT = fopen(archivo,"r+");
308
309     if(DocTXT)
310     {
311         Gtk::MessageDialog dialog(*this, "Archivo abierto");
312         dialog.set_secondary_text("Archivo leido correctamente");
313         dialog.run();
314         Boton4.set_sensitive(TRUE);
315     }
316     else
317     {
318         Gtk::MessageDialog dialog(*this, "Archivo no encontrado");
319         dialog.set_secondary_text("Escribe correctamente el nombre del archivo");
320         dialog.run();
321         Boton4.set_sensitive(FALSE);
322     }
323     fclose(DocTXT);
324 }
325 // Guardar archivo
326 void TWindow::on_button2_clicked()
327 {
328     /*
329     float y = 32.5, w, numero2;
330     char cadena[20];
331     unsigned char Bits1,Bits2;
332     Bits1 = 130;
333     Bits2 = 17;
334
335     Glib::ustring entry_text;
336     std::cout<<"Boton2 presionado"<<std::endl;
337     printf("Se presiono boton 2");
338
339     entry_text = Entrada1.get_text();
340
341     w = atof(entry_text.c_str());
342     numero2 = (Bits1<<8) + Bits2;
343
344     sprintf(cadena,"%f",numero2);
345     Etiqueta1.set_text(cadena);
346     Graficar.set_config(w);
347     */
348     float Timer, Addres, Reg, CH0, CH1, CH2, CH3, CH4, CH5, CH6, CH7;
349     FILE *DocTXT, *DocTXTsave;
350     int long_text=0;
351     float valor;
352
353     Glib::ustring archivo_text;
354     const char *archivo;
355
356     archivo_text = Entrada1.get_text();
357     archivo = archivo_text.c_str();
358
359     DocTXT = fopen("DataGeneral.txt","r");
360     DocTXTsave = fopen(archivo,"w");
```

```
361
362     while(fscanf(DocTXT,"%f",&valor)!=EOF) long_text++;
363     cout<<long_text/11<<std::endl;
364     rewind(DocTXT);
365     //std::cout<<22%11<<std::endl;
366     for (int i=1; i<=long_text; i++)
367     {
368         if(i%11==0)
369             fscanf(DocTXT,"%f",&valor),fprintf(DocTXTsave,"%f \n",valor);
370         else
371             fscanf(DocTXT,"%f",&valor),fprintf(DocTXTsave,"%f ",valor);
372     }
373     fclose(DocTXT);
374     fclose(DocTXTsave);
375 }
376 // Adquirir Seniales ECG
377 void TWindow::on_button3_clicked()
378 {
379     Boton3.set_sensitive(FALSE);
380     Graficar.inicio(0);
381     Boton5.set_sensitive(FALSE);
382     Boton6.set_sensitive(FALSE);
383     adquisicion = Glib::Thread::create(sigc::mem_fun(*this, &TWindow::Hilo), true);
384 }
385 //Boton para Graficar
386 void TWindow::on_button4_clicked()
387 {
388     Graficar.inicio(0);
389     plot = Glib::Thread::create(sigc::mem_fun(*this, &TWindow::HiloGraficar), true);
390
391 }
392 //PROGRAMAR sistema
393 void TWindow::on_button5_clicked()
394 {
395     Graficar.inicio(0);
396     char cad[32];
397     int Muestreo,Muestreoaux,VelM,Tiempo,Ch1g;
398     int Reset=1,CLKsel=1,PDWN=1,Config=1;
399     Glib::ustring entry_text;
400
401     //Adquirir seniales de comboboxtext y tiempo
402     // Ganancia de canales  3 bits c/u
403     //CBT1 es muestras por segundo
404     // Entrada de tiempo (s)
405     entry_text = Entrada2.get_text();
406     Tiempo = atoi(entry_text.c_str());
407     //Velocidad de muestreo  3 bits
408     Muestreoaux=CBT1.get_active_row_number();
409     //std::cout<<Tiempoaux<<std::endl;
410     //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
411     if(Muestreoaux==0) Muestreo=2, VelM=8000;
412     if(Muestreoaux==1) Muestreo=3, VelM=4000;
413     if(Muestreoaux==2) Muestreo=4, VelM=2000;
414     if(Muestreoaux==3) Muestreo=5, VelM=1000;
415     if(Muestreoaux==4) Muestreo=6, VelM=500;
416
417     MuestrasGral = Tiempo*VelM;
418
419     //std::cout<<Tiempo<<std::endl;
420     //Envio de ganancias
421     Ch1g=Chg1.get_active_row_number();
422     cad[0]=((Ch1g<<1)&7)+(Config&1); cad[1]=Chg2.get_active_row_number();
        cad[2]=Chg3.get_active_row_number();
423     cad[3]=Chg4.get_active_row_number(); cad[4]=Chg5.get_active_row_number();
        cad[5]=Chg6.get_active_row_number();
424     cad[6]=Chg7.get_active_row_number(); cad[7]=Chg8.get_active_row_number();
        cad[8]=Muestreo;
425     cad[9]=(MuestrasGral);  cad[10]=(MuestrasGral>>8); cad[11]=(MuestrasGral>>16);
426     cad[12]=Reset; cad[13]=CLKsel; cad[14]=PDWN; cad[15]=0x01;
427     cad[16]=0x00; cad[17]=0x00; cad[18]=0x00; cad[19]=0x00; cad[20]=0x00; cad[21]=0x00;
```

```
    cad[22]=0x00; cad[23]=0x00;
    cad[24]=0x00; cad[25]=0x00; cad[26]=0x00; cad[27]=0x00; cad[28]=0x00; cad[29]=0x00;
    cad[30]=0x00; cad[31]=0x00;

  //  Etiqueta1.set_text(CH1G);
   // for(int i=0;i<12;i++)
  //  std::cout<<(int)cad[i]<<std::endl;

    dev1.usbWr(cad);
    Boton3.set_sensitive(TRUE);
}

//RESET
void TWindow::on_button6_clicked()
{
//    fclose(DocTXT);
    Graficar.inicio(0);
    char cad[32];
    int Reset=0,CLKsel=0,PDWN=0,Config=1;

    //Envio de USB
    cad[0]=Config&1; cad[1]=0x00; cad[2]=0x00;
    cad[3]=0x00; cad[4]=0x00; cad[5]=0x00;
    cad[6]=0x00; cad[7]=0x00;

    cad[8]=0x00;  cad[9]=0x00;  cad[10]=0x00; cad[11]=0x00; cad[12]=Reset;
    cad[13]=CLKsel; cad[14]=PDWN; cad[15]=0x00;
    cad[16]=0x00; cad[17]=0x00; cad[18]=0x00; cad[19]=0x00; cad[20]=0x00; cad[21]=0x00;
    cad[22]=0x00; cad[23]=0x00;
    cad[24]=0x00; cad[25]=0x00; cad[26]=0x00; cad[27]=0x00; cad[28]=0x00; cad[29]=0x00;
    cad[30]=0x00; cad[31]=0x00;

   // std::cout<<(int)cad[0]<<std::endl;
    dev1.usbWr(cad);
    Boton3.set_sensitive(FALSE);
}
//CBT-s para registro de canales no se usan
void TWindow::CBT1muestreo()
{
    /**
    int Muestreoaux,VelM;
    Muestreoaux=CBT1.get_active_row_number();
//    //std::cout<<Tiempoaux<<std::endl;
//    //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
    if(Muestreoaux==0) VelM=8000;
    if(Muestreoaux==1) VelM=4000;
    if(Muestreoaux==2) VelM=2000;
    if(Muestreoaux==3) VelM=1000;
    if(Muestreoaux==4) VelM=500;
    VelocidadMuestreo=VelM;

    Graficar.set_resolx(hscale.get_value()*VelocidadMuestreo); */
}
void TWindow::CBT3g()
{

}
void TWindow::on_hscale_change()
{
//    int muestras;
//    int Muestreoaux,VelM;
//    Muestreoaux=CBT1.get_active_row_number();
////    //std::cout<<Tiempoaux<<std::endl;
////    //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
//    if(Muestreoaux==0) VelM=8000;
//    if(Muestreoaux==1) VelM=4000;
//    if(Muestreoaux==2) VelM=2000;
//    if(Muestreoaux==3) VelM=1000;
//    if(Muestreoaux==4) VelM=500;
```

```
492 //    VelocidadMuestreo=VelM;
493
494     Graficar.set_resolx(hscale.get_value()*VelocidadMuestreo);
495     datoscuadrosx = Glib::Thread::create(sigc::mem_fun(*this, &TWindow::HiloDatosx),
        true);
496 }
497 void TWindow::on_hscalev_change()
498 {
499     Graficar.set_resoly(hscalev.get_value());
500     datoscuadros = Glib::Thread::create(sigc::mem_fun(*this, &TWindow::HiloDatos),
        true);
501 }
502 void TWindow::HiloDatosx()
503 {
504     char cadena2[5];
505    // float auxseg;
506 /**
507     int Muestreoaux,VelM;
508     Muestreoaux=CBT1.get_active_row_number();
509     //std::cout<<Tiempoaux<<std::endl;
510     //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
511     if(Muestreoaux==0) VelM=8000;
512     if(Muestreoaux==1) VelM=4000;
513     if(Muestreoaux==2) VelM=2000;
514     if(Muestreoaux==3) VelM=1000;
515     if(Muestreoaux==4) VelM=500;
516     VelocidadMuestreo=VelM;
517 **/
518 //    auxseg = hscale.get_value()/VelocidadMuestreo;
519     sprintf(cadena2,"%.2f Seg",hscale.get_value()/12);
520     ValorX.set_text(cadena2);
521 }
522 void TWindow::HiloDatos()
523 {
524     /**Obtener dato de frecuencia del TXT y al Adquirir datos para sacar el valor por
        cuadro en tiempo
525     y voltaje Tiempo = 12(rayas)
526     Modificar TXT para que imprima el valor de la ganancia del canal
527     Para mostrar segundos en barra hscalex  segundos = Muestras del hscale(barra) /
        Muestras de TXT(ksps)
528     Para mostrar valor por cada cuadro en x cada cuadro = segundos / 12
529     Para mostrar valor por cada cuadro en y cadacuadro = voltaje de hscale /3       */
530     //ValorX.set_value();
531     char cadena[5];
532
533     sprintf(cadena,"%.2f Volts",hscalev.get_value()/3);
534     ValorY.set_text(cadena);
535 }
536 void TWindow::HiloGraficar()
537 {
538     //Graficar.inicio(0);
539     FILE *DocTXT;
540     Glib::ustring archivo_text;
541     const char *archivo;
542
543     double ChSel,j=0;
544     float valor=0;
545     double Tiempo=0,long_text=0;
546     float Voltajeextra;
547 //   list <double> Voltaje;
548     //Voltaje.clear();
549
550     archivo_text = Entrada1.get_text();
551     archivo = archivo_text.c_str();
552
553     DocTXT = fopen(archivo,"r");
554     /** Calcula el numero de datos del TXT*/
555     while(fscanf(DocTXT,"%f",&valor)!=EOF) long_text++;
556     rewind(DocTXT);
557     /** Coloca el canal que se va a graficar*/
```

```
558      ChSel=CBT4.get_active_row_number()+4;
559      //Graficar.inicio(0);
560  //    Graficar.set_resolx(hscale.get_value());
561  //    Graficar.set_resoly(hscalev.get_value());
562   //   Graficar.inicio(0);
563
564      Graficar.inicio(1);
565      for (int i=1; i<=long_text; i++)
566      {
567          if(i==(ChSel+11*j))
568          {
569              fscanf(DocTXT,"%f",&Voltajeextra),j++;//cout<<Voltaje<<endl;
570              Tiempo=j;
571              //Voltajeextra = ((Voltajeextra-(Voltajeextra > (8388608.0)-
                   1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
572              Graficar.set_config(Tiempo,long_text,Voltajeextra);
573              //fscanf(DocTXT,"%f",&valor),j++;//cout<<Voltaje<<endl;
574              //valor = ((valor-(valor > (8388608.0)-
                   1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
575              //   Voltaje.push_back(valor);
576              //Graficar.set_config(Tiempo,Voltaje,long_text);
577          }
578          else
579              fscanf(DocTXT,"%f",&Voltajeextra);
580      }
581      //cout<<"INICIO"<<endl;
582      //Graficar.inicio(1);
583      fclose(DocTXT);
584      /** procesos de prueba */
585      //   Voltaje.push_front(5);
586    //  Voltaje.push_front(6);
587     // Voltaje.push_front(7);
588
589  //    list<double>::iterator it = Voltaje.begin();
590
591  //    while(it!=Voltaje.end())
592  //    {
593  //      cout<<*it++<< endl;
594  //    }
595
596      //ResTime=hscale.get_value();
597  //    std::cout<<ResTime<<std::endl;
598    //  Graficar.set_config(ResTime);
599    /***/
600    //Graficar.set_config(Tiempo,Voltaje,long_text);
601  //    list<double>::iterator it = Voltaje.begin();
602  //
603  //    while(it!=Voltaje.end())
604  //    {
605  //      cout<<*it++<< endl;
606  //    }
607  }
608  void TWindow::Hilo()
609  {
610       // int x;
611     // char cadena[20];
612     // x=25;
613
614     // sprintf(cadena,"%d",x);
615
616     // std::cout<<"Boton1 presionado"<<std::endl;
617     // printf("Adquisicion de datos");
618     // Etiqueta1.set_text(cadena);
619   //   list <long double> Voltaje;
620   //   Voltaje.clear();
621      int ChSel;
622
623     // unsigned char cadena1[16],cadena2[16];
624      char cadena1[16],cadena2[16];
```

```
625      float Timer=0, Addres=0,Reg=0,CH0=0,CH1=0,CH2=0,CH3=0,CH4=0,CH5=0,CH6=0,CH7=0;
626      FILE *DocTXT;
627      DocTXT = fopen("DataGeneral.txt","wt");
628  //antes de adquirir datos enviar STRA de inicio
629      char cad[32];
630      int Reset=1,CLKsel=1,PDWN=1;
631      int STRA=1,Config=0;
632      //Envio de USB
633      cad[0]=Config; cad[1]=0x00; cad[2]=0x00;
634      cad[3]=0x00; cad[4]=0x00; cad[5]=0x00;
635      cad[6]=0x00; cad[7]=0x00;
636
637      cad[8]=0x00;  cad[9]=0x00;  cad[10]=0x00; cad[11]=0x00; cad[12]=Reset;
         cad[13]=CLKsel; cad[14]=PDWN; cad[15]=0x00;
638      cad[16]=STRA; cad[17]=0x00; cad[18]=0x00; cad[19]=0x00; cad[20]=0x00; cad[21]=0x00;
         cad[22]=0x00; cad[23]=0x00;
639      cad[24]=0x00; cad[25]=0x00; cad[26]=0x00; cad[27]=0x00; cad[28]=0x00; cad[29]=0x00;
         cad[30]=0x00; cad[31]=0x00;
640     // std::cout<<(int)cad[0]<<std::endl;
641      dev1.usbWr(cad); /* Agregado */
642      Graficar.inicio(1);
643      //Coloca datos en primer fila del txt
644      int Muestreoaux,VelM;
645      Muestreoaux=CBT1.get_active_row_number();
646      //std::cout<<Tiempoaux<<std::endl;
647      //Muestras generales 24 bits Muestreo codificacion de valores en ADS1298
648      if(Muestreoaux==0) VelM=8000;
649      if(Muestreoaux==1) VelM=4000;
650      if(Muestreoaux==2) VelM=2000;
651      if(Muestreoaux==3) VelM=1000;
652      if(Muestreoaux==4) VelM=500;
653      VelocidadMuestreo=VelM;
654      Graficar.set_resolx(hscale.get_value()*VelocidadMuestreo);
655
656      fprintf(DocTXT,"%d %lf %lf %d %d %d %d %d %d %d %d\n",VelM,Addres,Reg,
657              Chg1.get_active_row_number(),Chg2.get_active_row_number(),
                Chg3.get_active_row_number(),
658             Chg4.get_active_row_number(),Chg5.get_active_row_number(),
                Chg6.get_active_row_number(),
659             Chg7.get_active_row_number(),Chg8.get_active_row_number());
660      //Inicia Adquisicion
661      for (int i=1; i<=MuestrasGral; i++)
662      {
663          dev1.usbRd(cadena1);
664          dev1.usbRd(cadena2);
665
666          CH7=((unsigned char)cadena1[2]<<16)+((unsigned char)cadena1[1]<<8)+(unsigned
                 char)cadena1[0];
667          CH6=((unsigned char)cadena1[5]<<16)+((unsigned char)cadena1[4]<<8)+(unsigned
                 char)cadena1[3];
668          CH5=((unsigned char)cadena1[8]<<16)+((unsigned char)cadena1[7]<<8)+(unsigned
                 char)cadena1[6];
669          CH4=((unsigned char)cadena1[11]<<16)+((unsigned char)cadena1[10]<<8)+(unsigned
                 char)cadena1[9];
670          CH3=((unsigned char)cadena1[14]<<16)+((unsigned char)cadena1[13]<<8)+(unsigned
                 char)cadena1[12];
671
672          CH2=((unsigned char)cadena2[1]<<16)+((unsigned char)cadena2[0]<<8)+(unsigned
                 char)cadena1[15];
673          CH1=((unsigned char)cadena2[4]<<16)+((unsigned char)cadena2[3]<<8)+(unsigned
                 char)cadena2[2];
674          CH0=((unsigned char)cadena2[7]<<16)+((unsigned char)cadena2[6]<<8)+(unsigned
                 char)cadena2[5];
675          Reg=((unsigned char)cadena2[10]<<16)+((unsigned char)cadena2[9]<<8)+(unsigned
                 char)cadena2[8];
676          Addres=((unsigned char)cadena2[12]<<8)+((unsigned char)cadena2[11]);
677          Timer=(((unsigned char)cadena2[15])<<16)+(((unsigned
                     char)cadena2[14])<<8)+((unsigned char)cadena2[13]);
678
679          //Conversion a voltaje
```

```
680          CH7 = ((CH7-(CH7 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
681          CH6 = ((CH6-(CH6 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
682          CH5 = ((CH5-(CH5 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
683          CH4 = ((CH4-(CH4 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
684          CH3 = ((CH3-(CH3 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
685          CH2 = ((CH2-(CH2 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
686          CH1 = ((CH1-(CH1 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
687          CH0 = ((CH0-(CH0 > (8388608.0)-1.0)*(16777216.0))*(5.0/(16777216.0))/1.0);
688          //Valor de velocidad de adquisicion de datos
689          Timer = 1.0/(Timer/48000000.0);
690
691          //printf("%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf\n",Timer,Addres,Reg,CH0,
                       CH1, CH2, CH3, CH4, CH5, CH6, CH7);
692          fprintf(DocTXT,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf
                     %lf\n",Timer,Addres,Reg, CH0, CH1, CH2, CH3, CH4, CH5, CH6, CH7);
693  /** adquisicion*/
694          ChSel=CBT4.get_active_row_number();
695
696          if(ChSel==0) Graficar.set_config(i,MuestrasGral,CH0);//Voltaje.push_back(CH0);
697          if(ChSel==1) Graficar.set_config(i,MuestrasGral,CH1);//Voltaje.push_back(CH1);
698          if(ChSel==2) Graficar.set_config(i,MuestrasGral,CH2);//Voltaje.push_back(CH2);
699          if(ChSel==3) Graficar.set_config(i,MuestrasGral,CH3);//Voltaje.push_back(CH3);
700          if(ChSel==4) Graficar.set_config(i,MuestrasGral,CH4);//Voltaje.push_back(CH4);
701          if(ChSel==5) Graficar.set_config(i,MuestrasGral,CH5);//Voltaje.push_back(CH5);
702          if(ChSel==6) Graficar.set_config(i,MuestrasGral,CH6);//Voltaje.push_back(CH6);
703          if(ChSel==7) Graficar.set_config(i,MuestrasGral,CH7);//Voltaje.push_back(CH7);
704          //Graficar.set_config(i,Voltaje,MuestrasGral);
705      }
706      //Boton4.set_sensitive(TRUE);
707      fclose(DocTXT);
708      Boton2.set_sensitive(TRUE);
709      Boton3.set_sensitive(TRUE);
710      Boton5.set_sensitive(TRUE);
711      Boton6.set_sensitive(TRUE);
712  }
```

Bloque fuente de gráficos “Tplot.cpp”

```
 1  #include "Tplot.h"
 2  #include <iostream>
 3
 4  Tplot::Tplot(int w, int h)
 5  : r_circ(0.0)
 6  {
 7     set_size_request(w,h);
 8     Voltaje.clear();
 9
10  }
11
12  Tplot::~Tplot()
13  {
14  //    delete[] voltajey;
15  //    delete[] tiempox;
16      Voltaje.clear();
17  }
18
19  bool Tplot::on_expose_event(GdkEventExpose* event){
20  Glib::RefPtr<Gdk::Window> window = get_window();
21
22  if(!window) return true;
23
24  Cairo::RefPtr<Cairo::Context> cr= window-> create_cairo_context();
25
26  if(event){
27      cr->rectangle(event->area.x, event -> area.y, event->area.width, event-
>area.height);
28      cr-> clip();
```

```
 29  }
 30
 31  update_screen(cr);
 32  return true;
 33  }
 34
 35  void Tplot::update_screen(Cairo::RefPtr<Cairo::Context> cr){
 36  Gtk::Allocation allocation= get_allocation();
 37  width =allocation.get_width();
 38  height = allocation.get_height();
 39  xc= (double) (width / 2);
 40  yc= (double) (height/ 2);
 41  clear_background(cr);
 42  }
 43
 44  void Tplot::clear_background(Cairo::RefPtr<Cairo::Context> cr){
 45
 46  static const std::valarray<double> dashes(2, 1.0);
 47  int ndash= sizeof(dashes)/sizeof(dashes[0]);
 48  int i;
 49  Rx= width/resolx;
 50  Ry= yc/resoly;
 51
 52  /** Resolucion de pantalla es width y height
 53  */
 54  ////Rellena el fondo cr=widget dibujo de color negro
 55  cr-> save();
 56  cr->set_source_rgb(0.0, 0.0, 0.0);
 57  cr->paint();
 58  cr->restore();// y esta :D
 59
 60  cr->set_source_rgb(255.0, 0.0, 0.0); // cambia de color el puntero que dibuja en
pantalla
 61
 62  if(StartGraf)
 63  {
 64
 65  list<double>::iterator it = Voltaje.begin();
 66  list<double>::iterator it2 = Voltaje.begin();
 67
 68  *it2 = *it2++;
 69
 70  double scalax=1,scalaaux=0,scalaaux2=0,scalanom=0;
 71  double scalay=0,scalay2=0;
 87  //for(i=1;i<=n_datos;i++)
 88  if(Tiempo>resolx)
 89  {
 90          scalanom = Tiempo-resolx;
 91          for(int j=1;j<=scalanom;j++) *it=*it++,*it2=*it2++;
 92  }
 93
 94  while(it2!=Voltaje.end())
 95  {
 96      scalay = yc-(*it++)*Ry;
 97      scalay2 = yc-(*it2++)*Ry;
 98      scalaaux = scalax*Rx;
 99      scalaaux2 = (scalax+1)*Rx;
100
101      cr->save();
102      cr->set_line_width(1.8);
103      cr->move_to(scalaaux,scalay);
104      cr->line_to(scalaaux2,scalay2);
105  //    cr->move_to(tiempox[i],voltajey[i]);
106  //    cr->line_to(tiempox[++i],voltajey[++i]);
107      cr->stroke();
108      cr->restore();
109  //    cout<<*itxi++<<"\t";
110  //    cout<<*ityi++<<"\t";
111  //    cout<<*itxf++<<"\t";
112  //  cout<<scalax<<endl;
```

```
113     scalax++;
114 }
115 }
116 else
117 {
118     Voltaje.clear();
119     Tiempo=0;
120 }
121
122
123 // Crear grid en pantalla grafica
124 cr->set_source_rgb(0.0, 255.0, 0.0);
125 // en x
126 for(i=0;i<=12;i++)
127 {
128     cr->save();
129     cr->set_line_width(1.0);
130     cr->move_to(i*width/12,0);
131     cr->line_to(i*width/12,height);
132     cr->set_dash(dashes,ndash);
133     cr->stroke();
134     cr->restore();
135 }
136 // en y
137 for(i=0;i<=6;i++)
138 {
139     cr->save();
140     cr->set_line_width(1.0);
141     cr->move_to(0,i*height/6);
142     cr->line_to(width,i*height/6);
143     cr->set_dash(dashes,ndash);
144     cr->stroke();
145     cr->restore();
146 }
147     // Crear Ejes x, y de color verde para identificar origenes
148     cr->set_source_rgb(0.0, 255.0, 0.0);
149     cr->save();
150     cr->set_line_width(1.3);
151     cr->move_to(0,height/2);
152     cr->line_to(width,height/2);
153     cr->stroke();
154     cr->restore();
155
156     cr->save();
157     cr->set_line_width(1.3);
158     cr->move_to(width/2,0);
159     cr->line_to(width/2,height);
160     cr->stroke();
161     cr->restore();
162
163     Glib::RefPtr<Gdk::Window> window= get_window();
164         if(window){
165             Gdk::Rectangle
r(0,0,get_allocation().get_width(),get_allocation().get_height());
166             window->invalidate_rect(r,false);
167
168         }
169 }
170 //void Tplot::set_config(double Datox,list<double> Datoy,double Datoextra);
171 void Tplot::set_config(double Datox,double long_text,double Datoextra)
172 {
173     Tiempo = Datox;
174     Voltaje.push_back(Datoextra);
175     n_datos = long_text/11;
176
177 }
178 void Tplot::set_rcirc(double r)
179 {
180     r_circ = r;
```

```
}
//mandar configuraciones del plot
void Tplot::set_resolx(int resolucionx)
{
    resolx = resolucionx;
}
void Tplot::set_resoly(float resoluciony)
{
    resoly = resoluciony;
}
void Tplot::inicio(int start)
{
    StartGraf = start;
    Voltaje.clear();
}
```

Bloque de comunicación USB “comunicación.cpp” creado en la institución UAQ para uso de nuevos proyectos.

```

#ifdef __WIN32__
   #include "./libusb/usb.h"
#else
   #include <usb.h>
#endif

#include <stdio.h>
#include "comunicacion.h"

//9001,7929

#define USB_VID 0x2329
#define USB_PID 0x1EF9
#define TimeOut 100000
#define USB_OUT 0x01
#define USB_IN  0x82

Comunicacion::Comunicacion()
{
dev = NULL;
}

Comunicacion::~Comunicacion()
{
dev = NULL;
}

usb_dev_handle *Comunicacion::open_dev(int VID, int PID)
{
  struct usb_bus *bus;
  struct usb_device *dev;

  for(bus = usb_get_busses(); bus; bus = bus->next)
    {
      for(dev = bus->devices; dev; dev = dev->next)
        {
          printf("vid=%X pid=%X \n",dev->descriptor.idVendor, dev-
>descriptor.idProduct);
          if(dev->descriptor.idVendor == VID
             && dev->descriptor.idProduct == PID)
            {
              return usb_open(dev);
            }
        }
    }
  return NULL;
```

```
47  }
48
49  int Comunicacion::usbInit(void)
50  {
51
52    usb_init(); // initialize the library
53    usb_find_busses(); // find all busses
54    usb_find_devices(); // find all connected devices
55
56    dev = open_dev(USB_VID,USB_PID);
57    if(!dev)
58      {
59        printf("1 %s \n",usb_strerror());
60        return 0;
61      }
62    if(usb_set_configuration(dev, 1) < 0)
63      {
64        usb_close(dev);
65        printf("2 %s \n",usb_strerror());
66        return 0;
67      }
68
69    if(usb_claim_interface(dev, 0) < 0)
70      {
71        usb_close(dev);
72        printf("3 %s \n",usb_strerror());
73        return 0;
74      }
75      return 1;
76  }
77  //---------------------------------------------------------------------------
78  int Comunicacion::usbTerminate(void)
79  {
80     if(dev)
81     {
82         usb_release_interface(dev, 0);
83         usb_close(dev);
84     }
85     return 1;
86  }
87
88  void Comunicacion::usbWr(char *cad)
89  {
90  usb_bulk_write(dev, USB_OUT, (char *)cad, 32, TimeOut);
91  }
92
93  int Comunicacion::usbRd(char *cad)
94  {
95   char Data[16];
96   int m;
97
98   m=usb_bulk_read(dev, USB_IN, Data, 16, TimeOut);
99
100  if(m!=16)
101     printf("Error! ");
102
103  for (int i = 0; i <= 15; i++)
104  cad[i]=Data[i];
105  }
```

A continuación se enlistan las cabeceras de los registros.

Cabecera de el modulo principal “twindow.h”

```
1  #ifndef __TWINDOW_H__
```

```
#define __TWINDOW_H__

#include<gtkmm.h>
#include<comunicacion.h>
#include"Tplot.h"
#include<iostream>
#include<list>
using namespace std;
//#include<scale.h>

class TWindow : public Gtk::Window {
protected:

    Gtk::VBox m_VBox;
    Gtk::HBox m_HBox1,m_HBox2,m_HBox3;
    Gtk::Button Boton1,Boton2,Boton3,Boton4,Boton5,Boton6;

    Gtk::Entry Entrada1,Entrada2;
    Gtk::Label Etiqueta1,Etiqueta2,Etiqueta3,Etiqueta4,Etiqueta5,Etiqueta6,
               Etiqueta7,Etiquetav,EtiquetaX,EtiquetaY,ValorX,ValorY;
    Gtk::Label Etiqueta8,Etiqueta9,Etiqueta10,Etiqueta11,Etiqueta12,Etiqueta13,
               Etiqueta14,Etiqueta15;
    Gtk::Notebook Page1;
    Gtk::Table Malla_Main,Malla1,Malla2,Malla4,Malla5;
    Gtk::Frame Marco1,Marco2,Marco3,Marco4,Marco5;
    Gtk::Image Imagen_UAQ,Imagen_UAQ2;
    Gtk::ComboBoxText CBT1,CBT4;
    Gtk::ComboBoxText Chg1,Chg2,Chg3,Chg4,Chg5,Chg6,Chg7,Chg8;
    Glib::Thread *adquisicion, *plot, *datoscuadros, *datoscuadrosx;
    Gtk::HScale hscale,hscalev;
    Tplot Graficar;
    Comunicacion dev1;
    float numero;
    int VelocidadMuestreo;

   // FILE *DocTXT;

public:
    TWindow();
    ~TWindow();

    void on_button1_clicked();
    void on_button2_clicked();
    void on_button3_clicked();
    void on_button4_clicked();
    void on_button5_clicked();
    void on_button6_clicked();
    void on_hscale_change();
    void on_hscalev_change();
    void CBT3g();
    void CBT1muestreo();
    void Hilo();
    void HiloGraficar();
    void HiloDatos();
    void HiloDatosx();
};

#endif // __TWINDOW_H__
```

Cabecera de el modulo fuente de graficos “Tplot.h”

```
#ifndef TPLOT_H
#define TPLOT_H
#include <gtkmm.h>
#include<iostream>
#include<list>
using namespace std;

```

```
 8  class Tplot : public Gtk::DrawingArea {
 9      public:
10          Tplot(int w, int h);
11          virtual ~Tplot();
12          void set_rcirc(double r);
13          void set_config(double Datox,double long_text,double Datoextra);
14          //void set_config(double Datox,list<double> Datoy,double Datoextra);
15          void set_resolx(int resolucionx);
16          void set_resoly(float resoluciony);
17          void inicio(int start);
18          double Tiempo;
19          double resolx,resoly;
20          //list<long double> Voltaje;
21          list<double> Voltaje;
22          double n_datos;
23  //          double *voltajey;
24  //          double *tiempox;
25          int StartGraf;
26          double Rx,Ry;
27
28      protected:
29          bool on_expose_event(GdkEventExpose* event);
30          void update_screen(Cairo::RefPtr<Cairo::Context> cr);
31          void clear_background(Cairo::RefPtr<Cairo::Context> cr);
32
33      private: //no queremos que nadie acceda a ellas
34          double width, height;
35          double xc, yc;
36          double r_circ;
37  };
38
39  #endif // TPLOT_H
40
```

Cabecera de el modulo de comunicación “comunicación.h”

```
 1  #ifdef __WIN32__
 2     #include "./libusb/usb.h"
 3  #else
 4     #include <usb.h>
 5  #endif
 6
 7  //#define USB_VID 0x2329
 8  //#define USB_PID 0x1EF9
 9  //#define TimeOut 50000
10  #define USB_OUT 0x01
11  #define USB_IN  0x82
12
13  #define WORD unsigned short
14
15  class Comunicacion{
16
17   private:
18        usb_dev_handle *dev; // the device handle
19        usb_dev_handle *open_dev(int VID, int PID);
20
21
22   public:
23        int usbRd(char *cad);
24        void usbWr(char *cad);
25        int usbInit(void);
26        int usbTerminate(void);
27
28        Comunicacion();
29        virtual ~Comunicacion();
30
```

```
31 };
```

Librería de comunicación USB creada en la institución UAQ para la creación de nuevos proyectos.

```
 1 #ifndef __USB_H__
 2 #define __USB_H__
 3
 4 #include <stdlib.h>
 5 #include <windows.h>
 6
 7 /*
 8  * 'interface' is defined somewhere in the Windows header files. This macro
 9  * is deleted here to avoid conflicts and compile errors.
10  */
11
12 #ifdef interface
13 #undef interface
14 #endif
15
16 /*
17  * PATH_MAX from limits.h can't be used on Windows if the dll and
18  * import libraries are build/used by different compilers
19  */
20
21 #define LIBUSB_PATH_MAX 512
22
23
24 /*
25  * USB spec information
26  *
27  * This is all stuff grabbed from various USB specs and is pretty much
28  * not subject to change
29  */
30
31 /*
32  * Device and/or Interface Class codes
33  */
34 #define USB_CLASS_PER_INTERFACE     0   /* for DeviceClass */
35 #define USB_CLASS_AUDIO               1
36 #define USB_CLASS_COMM                2
37 #define USB_CLASS_HID                   3
38 #define USB_CLASS_PRINTER             7
39 #define USB_CLASS_MASS_STORAGE      8
40 #define USB_CLASS_HUB                    9
41 #define USB_CLASS_DATA                10
42 #define USB_CLASS_VENDOR_SPEC         0xff
43
44 /*
45  * Descriptor types
46  */
47 #define USB_DT_DEVICE           0x01
48 #define USB_DT_CONFIG           0x02
49 #define USB_DT_STRING           0x03
50 #define USB_DT_INTERFACE    0x04
51 #define USB_DT_ENDPOINT     0x05
52
53 #define USB_DT_HID          0x21
54 #define USB_DT_REPORT       0x22
55 #define USB_DT_PHYSICAL 0x23
56 #define USB_DT_HUB          0x29
57
58 /*
59  * Descriptor sizes per descriptor type
60  */
61 #define USB_DT_DEVICE_SIZE      18
62 #define USB_DT_CONFIG_SIZE      9
```

```
63  #define USB_DT_INTERFACE_SIZE       9
64  #define USB_DT_ENDPOINT_SIZE        7
65  #define USB_DT_ENDPOINT_AUDIO_SIZE  9   /* Audio extension */
66  #define USB_DT_HUB_NONVAR_SIZE      7
67
68
69  /* ensure byte-packed structures */
70  #include <pshpack1.h>
71
72
73  /* All standard descriptors have these 2 fields in common */
74  struct usb_descriptor_header {
75    unsigned char  bLength;
76    unsigned char  bDescriptorType;
77  };
78
79  /* String descriptor */
80  struct usb_string_descriptor {
81    unsigned char  bLength;
82    unsigned char  bDescriptorType;
83    unsigned short wData[1];
84  };
85
86  /* HID descriptor */
87  struct usb_hid_descriptor {
88    unsigned char  bLength;
89    unsigned char  bDescriptorType;
90    unsigned short bcdHID;
91    unsigned char  bCountryCode;
92    unsigned char  bNumDescriptors;
93  };
94
95  /* Endpoint descriptor */
96  #define USB_MAXENDPOINTS    32
97  struct usb_endpoint_descriptor {
98    unsigned char  bLength;
99    unsigned char  bDescriptorType;
100   unsigned char  bEndpointAddress;
101   unsigned char  bmAttributes;
102   unsigned short wMaxPacketSize;
103   unsigned char  bInterval;
104   unsigned char  bRefresh;
105   unsigned char  bSynchAddress;
106
107   unsigned char *extra;   /* Extra descriptors */
108   int extralen;
109 };
110
111 #define USB_ENDPOINT_ADDRESS_MASK   0x0f    /* in bEndpointAddress */
112 #define USB_ENDPOINT_DIR_MASK         0x80
113
114 #define USB_ENDPOINT_TYPE_MASK      0x03    /* in bmAttributes */
115 #define USB_ENDPOINT_TYPE_CONTROL       0
116 #define USB_ENDPOINT_TYPE_ISOCHRONOUS   1
117 #define USB_ENDPOINT_TYPE_BULK          2
118 #define USB_ENDPOINT_TYPE_INTERRUPT   3
119
120 /* Interface descriptor */
121 #define USB_MAXINTERFACES   32
122 struct usb_interface_descriptor {
123   unsigned char  bLength;
124   unsigned char  bDescriptorType;
125   unsigned char  bInterfaceNumber;
126   unsigned char  bAlternateSetting;
127   unsigned char  bNumEndpoints;
128   unsigned char  bInterfaceClass;
129   unsigned char  bInterfaceSubClass;
130   unsigned char  bInterfaceProtocol;
131   unsigned char  iInterface;
```

```
132
133   struct usb_endpoint_descriptor *endpoint;
134
135   unsigned char *extra;   /* Extra descriptors */
136   int extralen;
137 };
138
139 #define USB_MAXALTSETTING   128 /* Hard limit */
140
141 struct usb_interface {
142   struct usb_interface_descriptor *altsetting;
143
144   int num_altsetting;
145 };
146
147 /* Configuration descriptor information.. */
148 #define USB_MAXCONFIG       8
149 struct usb_config_descriptor {
150   unsigned char  bLength;
151   unsigned char  bDescriptorType;
152   unsigned short wTotalLength;
153   unsigned char  bNumInterfaces;
154   unsigned char  bConfigurationValue;
155   unsigned char  iConfiguration;
156   unsigned char  bmAttributes;
157   unsigned char  MaxPower;
158
159   struct usb_interface *interface;
160
161   unsigned char *extra;   /* Extra descriptors */
162   int extralen;
163 };
164
165 /* Device descriptor */
166 struct usb_device_descriptor {
167   unsigned char  bLength;
168   unsigned char  bDescriptorType;
169   unsigned short bcdUSB;
170   unsigned char  bDeviceClass;
171   unsigned char  bDeviceSubClass;
172   unsigned char  bDeviceProtocol;
173   unsigned char  bMaxPacketSize0;
174   unsigned short idVendor;
175   unsigned short idProduct;
176   unsigned short bcdDevice;
177   unsigned char  iManufacturer;
178   unsigned char  iProduct;
179   unsigned char  iSerialNumber;
180   unsigned char  bNumConfigurations;
181 };
182
183 struct usb_ctrl_setup {
184   unsigned char  bRequestType;
185   unsigned char  bRequest;
186   unsigned short wValue;
187   unsigned short wIndex;
188   unsigned short wLength;
189 };
190
191 /*
192  * Standard requests
193  */
194 #define USB_REQ_GET_STATUS          0x00
195 #define USB_REQ_CLEAR_FEATURE       0x01
196 /* 0x02 is reserved */
197 #define USB_REQ_SET_FEATURE         0x03
198 /* 0x04 is reserved */
199 #define USB_REQ_SET_ADDRESS         0x05
200 #define USB_REQ_GET_DESCRIPTOR      0x06
201 #define USB_REQ_SET_DESCRIPTOR      0x07
```

```
202 #define USB_REQ_GET_CONFIGURATION   0x08
203 #define USB_REQ_SET_CONFIGURATION   0x09
204 #define USB_REQ_GET_INTERFACE         0x0A
205 #define USB_REQ_SET_INTERFACE         0x0B
206 #define USB_REQ_SYNCH_FRAME         0x0C
207
208 #define USB_TYPE_STANDARD       (0x00 << 5)
209 #define USB_TYPE_CLASS          (0x01 << 5)
210 #define USB_TYPE_VENDOR         (0x02 << 5)
211 #define USB_TYPE_RESERVED       (0x03 << 5)
212
213 #define USB_RECIP_DEVICE        0x00
214 #define USB_RECIP_INTERFACE 0x01
215 #define USB_RECIP_ENDPOINT  0x02
216 #define USB_RECIP_OTHER         0x03
217
218 /*
219  * Various libusb API related stuff
220  */
221
222 #define USB_ENDPOINT_IN         0x80
223 #define USB_ENDPOINT_OUT        0x00
224
225 /* Error codes */
226 #define USB_ERROR_BEGIN         500000
227
228 /*
229  * This is supposed to look weird. This file is generated from autoconf
230  * and I didn't want to make this too complicated.
231  */
232 #define USB_LE16_TO_CPU(x)
233
234 /* Data types */
235 /* struct usb_device; */
236 /* struct usb_bus; */
237
238 struct usb_device {
239   struct usb_device *next, *prev;
240
241   char filename[LIBUSB_PATH_MAX];
242
243   struct usb_bus *bus;
244
245   struct usb_device_descriptor descriptor;
246   struct usb_config_descriptor *config;
247
248   void *dev;      /* Darwin support */
249
250   unsigned char devnum;
251
252   unsigned char num_children;
253   struct usb_device **children;
254 };
255
256 struct usb_bus {
257   struct usb_bus *next, *prev;
258
259   char dirname[LIBUSB_PATH_MAX];
260
261   struct usb_device *devices;
262   unsigned long location;
263
264   struct usb_device *root_dev;
265 };
266
267 /* Version information, Windows specific */
268 struct usb_version {
269   struct {
270     int major;
```

```
    int minor;
    int micro;
    int nano;
  } dll;
  struct {
    int major;
    int minor;
    int micro;
    int nano;
  } driver;
};


struct usb_dev_handle;
typedef struct usb_dev_handle usb_dev_handle;

/* Variables */
#ifndef __USB_C__
#define usb_busses usb_get_busses()
#endif



#include <poppack.h>


#ifdef __cplusplus
extern "C" {
#endif

  /* Function prototypes */

  /* usb.c */
  usb_dev_handle *usb_open(struct usb_device *dev);
  int usb_close(usb_dev_handle *dev);
  int usb_get_string(usb_dev_handle *dev, int index, int langid, char *buf,
                     size_t buflen);
  int usb_get_string_simple(usb_dev_handle *dev, int index, char *buf,
                            size_t buflen);

  /* descriptors.c */
  int usb_get_descriptor_by_endpoint(usb_dev_handle *udev, int ep,
                                     unsigned char type, unsigned char index,
                                     void *buf, int size);
  int usb_get_descriptor(usb_dev_handle *udev, unsigned char type,
                         unsigned char index, void *buf, int size);

  /* <arch>.c */
  int usb_bulk_write(usb_dev_handle *dev, int ep, char *bytes, int size,
                     int timeout);
  int usb_bulk_read(usb_dev_handle *dev, int ep, char *bytes, int size,
                    int timeout);
  int usb_interrupt_write(usb_dev_handle *dev, int ep, char *bytes, int size,
                          int timeout);
  int usb_interrupt_read(usb_dev_handle *dev, int ep, char *bytes, int size,
                         int timeout);
  int usb_control_msg(usb_dev_handle *dev, int requesttype, int request,
                      int value, int index, char *bytes, int size,
                      int timeout);
  int usb_set_configuration(usb_dev_handle *dev, int configuration);
  int usb_claim_interface(usb_dev_handle *dev, int interface);
  int usb_release_interface(usb_dev_handle *dev, int interface);
  int usb_set_altinterface(usb_dev_handle *dev, int alternate);
  int usb_resetep(usb_dev_handle *dev, unsigned int ep);
  int usb_clear_halt(usb_dev_handle *dev, unsigned int ep);
  int usb_reset(usb_dev_handle *dev);

  char *usb_strerror(void);

  void usb_init(void);
```

```
341    void usb_set_debug(int level);
342    int usb_find_busses(void);
343    int usb_find_devices(void);
344    struct usb_device *usb_device(usb_dev_handle *dev);
345    struct usb_bus *usb_get_busses(void);
346
347
348    /* Windows specific functions */
349
350    #define LIBUSB_HAS_INSTALL_SERVICE_NP 1
351    int usb_install_service_np(void);
352    void CALLBACK usb_install_service_np_rundll(HWND wnd, HINSTANCE instance,
353                                                LPSTR cmd_line, int cmd_show);
354
355    #define LIBUSB_HAS_UNINSTALL_SERVICE_NP 1
356    int usb_uninstall_service_np(void);
357    void CALLBACK usb_uninstall_service_np_rundll(HWND wnd, HINSTANCE instance,
358                                                  LPSTR cmd_line, int cmd_show);
359
360    #define LIBUSB_HAS_INSTALL_DRIVER_NP 1
361    int usb_install_driver_np(const char *inf_file);
362    void CALLBACK usb_install_driver_np_rundll(HWND wnd, HINSTANCE instance,
363                                               LPSTR cmd_line, int cmd_show);
364
365    #define LIBUSB_HAS_TOUCH_INF_FILE_NP 1
366    int usb_touch_inf_file_np(const char *inf_file);
367    void CALLBACK usb_touch_inf_file_np_rundll(HWND wnd, HINSTANCE instance,
368                                               LPSTR cmd_line, int cmd_show);
369
370    #define LIBUSB_HAS_INSTALL_NEEDS_RESTART_NP 1
371    int usb_install_needs_restart_np(void);
372
373    const struct usb_version *usb_get_version(void);
374
375    int usb_isochronous_setup_async(usb_dev_handle *dev, void **context,
376                                    unsigned char ep, int pktsize);
377    int usb_bulk_setup_async(usb_dev_handle *dev, void **context,
378                             unsigned char ep);
379    int usb_interrupt_setup_async(usb_dev_handle *dev, void **context,
380                                  unsigned char ep);
381
382    int usb_submit_async(void *context, char *bytes, int size);
383    int usb_reap_async(void *context, int timeout);
384    int usb_reap_async_nocancel(void *context, int timeout);
385    int usb_cancel_async(void *context);
386    int usb_free_async(void **context);
387
388
389  #ifdef __cplusplus
390  }
391  #endif
392
393  #endif /* __USB_H__ */
394
```