PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Yvo Efraím Henríquez Guerra

Desarrollo de proyectos de ingeniería biomédica basados en ASIC

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 17 de agosto de 2018

Desarrollo de proyectos de ingeniería biomédica basados en ASIC

Yvo Efraím Henríquez Guerra

Informe Final para optar al título de Ingeniero Civil Electrónico,
aprobada por la comisión de la
Escuela de Ingeniería Eléctrica de la
Facultad de Ingeniería de la
Pontificia Universidad Católica de Valparaíso
conformada por

Sr. Juan Vignolo Barchiesi
Profesor Guía

Sr. Sebastián Fingerhuth Massmann
Segundo Revisor

Sr. Jorge Mendoza Baeza
Director de Escuela

Valparaíso, 17 de agosto de 2018

A Dios, por todo lo recibito a lo largo de mi vida.

A mis padres, que siempre me han apoyado.

A Camila, pilar fundamental en este logro y en mi vida.

Agradecimientos

Gracias a Dios por todo.

Gracias a mis padres, que siempre me apoyaron, y siempre han estado junto a mí, los amo mucho.

Camila, gracias por estar a mi lado, por hacerme mejor persona, por ser quien eres. Gracias por apoyarme y nunca dejarme caer.

A mis amigos, agradezco a ustedes por la buena onda, el compañerismo y el desinterés que siempre han mostrado hacia mí.

Valparaíso, 17 de agosto de 2018

Yvo Henríquez

Resumen

Se estudiaron las características de las señales de los biopotenciales tales como EEG, EMG y ECG, poniendo especial énfasis en éste último, para elegir un ASIC que cumpliera con las características para adquirir y enviar dichas señales desde el cuerpo humano a un PC.

En base a sus características, precio y disponibilidad en el mercado, se seleccionó un ASIC para emplearlo en el proyecto.

El ASIC seleccionado fue programado mediante un microcontrolador. Por medio del cual se recibieron las señales del ASIC y se enviaron empleando el puerto USB a un PC para ser procesadas digitalmente y analizadas.

Se desarrolló un programa en Python para procesar digitalmente las señales enviadas desde el ASIC y desplegarlas en una interfaz gráfica desarrollada en la misma plataforma.

Con lo anterior se creó una aplicación biomédica, construyendo un electrocardiógrafo basado en un ADS1298. Para lograrlo, primero se evaluó la respuesta del ADS1298 a señales conocidas generadas por un generador de funciones, para corroborar que el ASIC no modificara la señal de entrada.

Las evaluaciones y resultados comprobaron que el empleo de un ASIC, para reemplazar la etapa análoga de los circuitos de adquisición de biopotenciales, no modifica las formas de onda de las señales probadas, además, facilita el diseño circuital y permite programar distintas características del ADS1298 mediante software, sin modificar el hardware del proyecto, o de proyectos futuros.

Palabras claves: ADS1298, ASIC, Microcontrolador, Comunicación SPI, Electrocardiógrafo.

Abstract

The characteristics of biopotential signals such as EEG, EMG and ECG were studied, placing special emphasis on the latter, to choose an ASIC that would comply with the characteristics to acquire and send these signals from the human body to a PC.

Based on its characteristics, price and availability in the market, an ASIC was selected to be used in the project

The selected ASIC was programmed through a microcontroller, through which the ASIC signals were received and sent using the USB port to a PC to be processed digitally and analyzed.

A program was developed in Python, to digitally process the signals sent from the ASIC and display them in a graphical interface developed on the same platform.

With the above, a biomedical application was created, building an electrocardiograph based on an ADS1298. To achieve this, the response of the ADS1298 to known signals generated by a function generator was first evaluated to verify that the ASIC did not modify the input signal.

The evaluations and results verified that the use of an ASIC, to replace the analogous stage of the biopotential acquisition circuits, is advisable, since it facilitates the circuit design and allows to program different characteristics through software, without modifying the hardware of the project, or Future projects.

Key words: ADS1298, ASIC, Microcontroller, SPI Communication, Electrocardiograph

Índice general

Introducción

Durante mucho tiempo, los potenciales de origen biológico o biopotenciales han demostrado ser una herramienta muy útil de diagnóstico médico. Los métodos de diagnóstico basados en la interpretación de estos biopotenciales, por ejemplo, el electrocardiograma y el electroencefalograma, son utilizados en forma habitual en la práctica médica. Se trata de técnicas no invasivas, de bajo costo y de rápida interpretación por un profesional entrenado [1]

En el siglo pasado, para conseguir señales de buena calidad, era usual recurrir a técnicas invasivas como abrasión de la piel o electrodos subcutáneos. Estas soluciones apuntaban a mejorar la conexión electrodo-piel. Hoy en día la tendencia es desplazar la complejidad de la adquisición de biopotenciales, al instrumento médico, intentando ser lo menos invasivo posible con el paciente. Gracias al avance tecnológico esto es cada día más fácil [2].

Diversos dispositivos diseñados para estudio de los biopotenciales, hoy en día están basado en circuitos integrados de propósito general, dentro de los que se encuentran algunos proyectos desarrollados en el Laboratorio de Sistemas Electrónicos e Instrumentación (Labsei) [3], [4] [5], relacionados con electrocardiogramas, electroencefalogramas o electromiogramas. Los proyectos antes mencionados se emplean como base para comenzar el estudio de las señales de interés, y se usan como guía para definir las características del ASIC a elegir.

Montar aplicaciones biomédicas basadas en circuito integrados de propósito general requiere de un tiempo de dedicación para su diseño, y ocupan una superficie física de un tamaño considerable, pensando en la implementación de un dispositivo de medición de biopotenciales portátil.

La tendencia en la ingeniería moderna, es hacer que los dispositivos electrónicos tengan más capacidad de procesamiento de datos, y ocupen el menor espacio posible. Justamente esta es una de las principales característica la cumplen los ASIC, circuitos diseñados específicamente para una aplicación, sacando el máximo provecho a los componentes electrónicos y minimizando el espacio que estos utilizan. [6]

Un Aplication Specific Integrated Circuit o circuito integrado de aplicación específica, mejor conocido como ASIC por sus siglas en inglés, es un circuito integrado configurable que ha sido diseñado para un propósito u aplicación específica para un producto electrónico específico.

Con los últimos avances en las tecnologías de miniaturización y las herramientas de diseño, la complejidad máxima, y por ende la funcionalidad, en un ASIC ha crecido desde 5.000 puertas lógicas a más de 100 millones.

Los niveles de configuración de un ASIC pueden estar en el campo de lo físico (construcción del hardware) o a nivel lógico (configuración por software) [7].

En el Laboratorio de Sistemas Electrónicos e Instrumentación se han desarrollado proyectos relacionados a biopotenciales pero nunca se ha empleado, para dicho propósito, un ASIC.

Tomando en cuenta las características de las señales de interés para el proyecto, se definen una serie de requisitos que un ASIC debiese cumplir. Se evalúan distintos ASIC que cumplan con lo definido anteriormente y se escoge uno para programarlo y desarrollar una aplicación biomédica con la señal de mayor importancia para el laboratorio.

Dado que la principal motivación es generar conocimiento académico respecto a los ASIC es que se elige la señal de electrocardiograma para probar el funcionamiento del ASIC seleccionado. La programación del chip se hará mediante un microcontrolador, que a su vez estará conectado al puerto USB de un PC, donde se recibirán las señales captadas y se procesarán digitalmente.

Python es uno de los lenguajes de programación más empleados actualmente [8]. El lenguaje se centra en la legibilidad y debido a su sintaxis, el programador escribe menos líneas de código que en otros lenguajes como C++ o Java, además la versatilidad y las capacidades de Python son inmensas [8], y debido a esta característica es que la segunda etapa del proyecto, que comprende el procesamiento digital de la señal y en el despliegue de ésta en una interfaz gráfica, se desarrolla en Python.

Para el diseño de los circuitos y conexiones del ASIC seleccionado, se emplean recomendaciones y sugerencias dadas por el fabricante la hoja de datos del chip, en hojas técnicas relacionadas con electrocardiógrafos y en los foros técnicos de la marca disponibles vía web [9] [10] [11].

Se escoge la señal de ECG para desarrollar el proyecto, por ser la de mayor interés académico para los docentes del LABSEI. Además, el éxito del proyecto, hará posible, con los conocimientos adquiridos sobre el ASIC, emplearlo en futuros proyectos de cursos relacionados a la instrumentación electrónica, o al procesamiento digital de señales, inclusive en un curso donde se mezclen ambas disciplinas. Del mismo modo, es posible seguir desarrollando las ideas que este proyecto dejará, para mejorarlas, probarlas de distintas maneras o también para continuar y abarcar otras disciplinas, como el control automático, o la transmisión de los datos por radiofrecuencia.

Además de comprobar el funcionamiento del proyecto, realizando pruebas con el electrocardiógrafo construido en pacientes reales, se comprobará la respuesta en frecuencia del ASIC seleccionado, para revisar que éste último no modifique ni la forma de la señal, ni tampoco las componentes en frecuencia de la señal original. Las pruebas para comprobar que el ASIC no

modifique la señal, se hacen conectando un generador de funciones a la entrada del dispositivo, y se compara la señal de entrada con la señal recibida en el PC, tanto en el dominio del tiempo como en el dominio de la frecuencia.

El prototipo final del proyecto se desarrolla en un tablero de experimentación, el que quedará disponible en el LABSEI para poder ser modificado o para realizar nuevos proyectos basados en los resultados del proyecto que se describe en el presente informe.

Objetivos generales

Evaluar soluciones basadas en circuitos integrados de aplicaciones específicas (ASIC) para implementar ECG, EEG y EMG.

Objetivos específicos

- Estudiar proyectos de ECG, EEG y EMG, realizados anteriormente, los cuales están basados en circuitos integrados de uso general.
- Estudiar las características de ASIC adecuados.
- Seleccionar un ASIC y montarlo en un tablero de experimentación.
- Conectar el ASIC a una CPU para programarlo y recibir datos.
- Implementar y evaluar una aplicación biomédica.

1 Problemática y solución propuesta

En el presente proyecto se evalúa el uso de un circuito integrado de aplicación específica (ASIC, de sus siglas en inglés) para obtener señales de ECG, EMG y EEG. El ASIC es montado en un tablero de experimentación y conectado a un PC, desde donde se programa y a la vez se reciben los datos capturados. De este modo ya no se desarrollará la etapa de conversión de la señal por partes, empleando distintos circuitos integrados, sino que estará integrada completamente dentro de un mismo chip.

1.1 Motivación y beneficios del proyecto

La principal utilidad que presta el desarrollo del proyecto es la familiarización con el uso de ASICs, en reemplazo de los circuitos integrados (IC) de propósito general usados hasta ahora en el Laboratorio de Sistemas Electrónicos e Instrumentación (LABSEI) de la Pontificia Universidad Católica de Valparaíso. El uso de ASICs disminuye notablemente el tiempo de diseño de los circuitos para adquirir y procesar los biopotenciales.

El principal beneficio del desarrollo del proyecto tiene que ver con adquirir conocimientos y técnicas, a nivel académico, sobre el empleo de un ASIC, pues es la primera vez que se emplea este tipo de circuitos integrados en el LABSEI. Este conocimiento no es solo para el alumno proyectista, sino también para los docentes, Profesor Guía y Correferentes, quienes en un futuro próximo podrían emplear esta tecnología para dictar cursos nuevos o agregarla a cursos ya existentes. Por ejemplo, se puede usar esta tecnología en el curso Laboratorio de Instrumentación Electrónica, un proyecto que compare la creación de un electrocardiograma con IC de propósito general con otro creado con ASIC, de este modo el estudiante podría implementar ambas técnicas, aprendería a programar el ASIC, sin perder la didáctica que conlleva diseñar en base circuitos integrados de propósito general. También se podría implementar un nuevo curso, que mezcle el procesamiento digital de señales, junto al diseño de circuitos, así el empleo de ASICs haría que el tiempo dedicado al diseño circuital sea menor, permitiendo una mayor dedicación a la etapa de procesamiento digital.

1.2 Trabajos similares desarrollados en Labsei

En el estudio previo, se sigue como referencia un proyecto por cada una de las señales de biopotenciales a estudiar.

En el caso de ECG se considera un proyecto [4] que consiste en visualizar un electrocardiograma en un PC. Para esto se construye un circuito compuesto de una etapa análoga, que permita adquirir y amplificar las señales eléctricas del corazón, y luego una etapa digital, que convierta los datos a digital empleando un microcontrolador. En último lugar se envían los datos convertidos al PC y se realiza el filtrado digital empleando el software Matlab para su visualización.

Con respecto a EMG la referencia es un proyecto [3] en el que se presenta el desarrollo de un antebrazo artificial y un sistema de adquisición de datos EMG. En este proyecto se detallan los elementos y metodologías utilizados en la adquisición de los potenciales de acción generados por los músculos y como traducirlos a movimiento mecánico en una prótesis. Para lo anterior se utiliza el software LabVIEW.

Por último, para EEG se estudia un proyecto [5] en el que se describe cómo implementar un dispositivo que sea capaz de detectar y clasificar las ondas cerebrales de un individuo, mediante electrodos activos, electrónica análoga a fin de adquirir y amplificar las señales, y electrónica digital para procesar y analizarlas espectralmente.

1.3 Señales bioeléctricas y sus características

Los potenciales bioeléctricas del cuerpo humano, en general no son determinísticos. Sus magnitudes varían con el tiempo, incluso cuando todos los factores que las originan están controlados. Por ejemplo, entre dos individuos, estando ambos sanos y controlando las variables externas, las medias de las mediciones de ellos pueden tener variaciones no despreciables.

La principal problemática para la adquisición de las señales de los biopotenciales es el ruido generado por los equipos en el entorno, el movimiento de las conexiones o el producido por el mismo cuerpo.

1.3.1 Electrocardiograma

Con propósito de obtener las señales de la actividad eléctrica del corazón, se emplean electrodos en contacto con la piel, estos pueden ser ubicados en 12 formas diferentes, lo que se conoce como derivaciones, estas a la vez se pueden clasificar en derivación bipolar, unipolar y precordial.

Derivación bipolar: Registra la diferencia de potencial entre las extremidades del cuerpo.

Derivación unipolar: Miden el potencial entre un punto teórico central (generalmente en el pecho) y una extremidad del cuerpo.

Derivación precordial: Se localizan los electrodos en el torso (vertebras), y así miden el potencial absoluto en la zona donde se encuentren ubicados.

1.3.2 Señal electrocardiográfica

Normalmente, el rango de frecuencia de una señal ECG va entre aproximadamente 0,05 [Hz] a 150 [Hz]. La señal de ECG se caracteriza por cinco picos y valles marcados por las letras P, Q, R, S, T (Ver Figura 1-1). En ocasiones hay otro pico llamado U, que representa la línea isoeléctrica. Las ondas P, QRS y T reflejan la despolarización eléctrica rítmica y la repolarización del miocardio asociada con las contracciones de las aurículas y ventrículos. El ECG se utiliza clínicamente en el diagnóstico de diversas anomalías y afecciones asociadas con el corazón. La amplitud típica de la onda QRS es de 1 [mV], pero la ANSI recomienda que un equipo de ECG pueda recibir señales de hasta 10 [mV] peak to peak [12]. El tiempo que duran los diversos picos en el ECG se menciona en la Tabla 1-1[13].

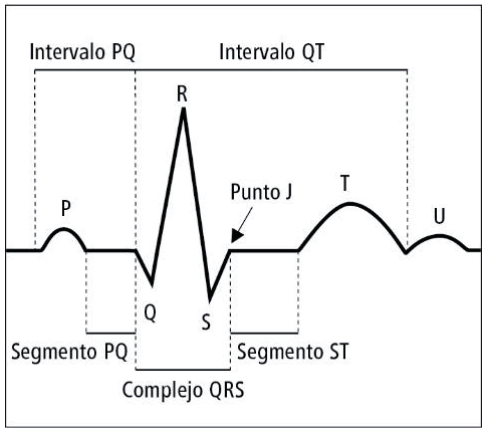

Figura 1-1: Señal Electrocardiográfica (fuente: http://empendium.com)

Tabla 1-1: Duración de cada intervalo de la señal ECG.

Intervalo	Duración [ms]
P-R	120 a 200
Q-T	350 a 440
S-T	50a 150
Onda P	110
QRS	90

Se cree que la onda U es producida por la repolarización ventricular de las células de Purkinje. [14]

1.3.3 Electromiograma

Para adquirir la señal electromiográfica, hay dos métodos, la electromiografía invasiva y la superficial, la primera se obtiene introduciendo al interior del musculo agujas que funcionan como electrodos y captan el potencial de cada neurona motriz. Considerando que cada musculo está compuesto por varias neuronas motrices, el método invasivo es más preciso que la

electromiografía superficial (SEMG), la cual capta el potencial del musculo utilizando electrodos tipo parche puestos sobre la piel [3].

Los registros obtenidos mediante la SEMG muestran actividad poblacional de las unidades motoras, es decir de todas las neuronas motoras juntas, tal como muestra la Figura 1-2: Registro de actividad poblacional mediante SEMG (fuente: http://jn.physiology.org) El uso de electrodos superficiales es mucho más adecuado para el estudio del comportamiento promedio de la actividad eléctrica de un músculo o grupo de músculos, siendo muy útil en la detección de fatiga muscular y en el monitoreo de rendimiento de deportistas. Además, para realizar la electromiografía invasiva se necesitan conocimientos médicos, por este hecho, esta se descarta para el proyecto.

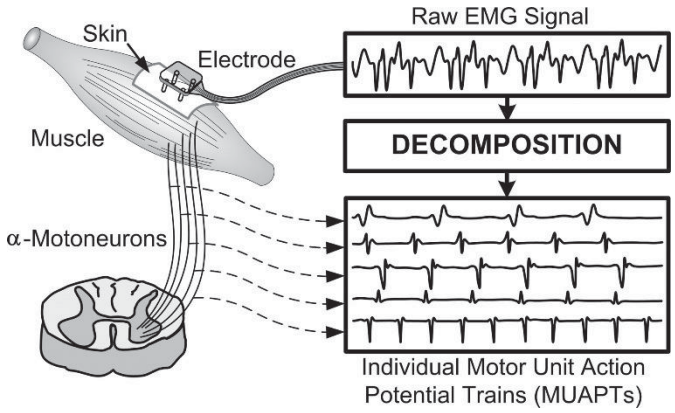

Figura 1-2: Registro de actividad poblacional mediante SEMG (fuente: http://jn.physiology.org)

1.3.4 Señal Miográfica

La amplitud de la señal EMG es de naturaleza aleatoria y puede ser representada mediante una distribución gaussiana [3]. Los valores en que oscila la tensión van entre 50 [µV] y 5 [mV] peak to peak y el espectro de la señal, para los registros superficiales, está entre 0 hasta 500 [Hz], siendo la banda de mayor energía la comprendida entre 50 y 150 [Hz], como se muestra en la Figura 1-3.

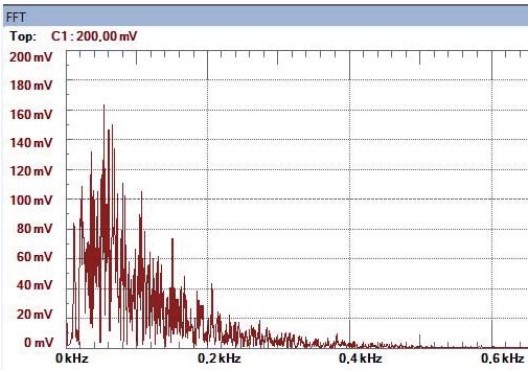

Figura 1-3: Espectro de registro EMG, mostrando frecuencia de mayor actividad [3].

1.3.5 Electroencefalograma

El sistema 10-20 o sistema internacional 10-20 es un método internacionalmente reconocido que indica cómo ubicar los electrodos en el cuero cabelludo, todo esto en el contexto de una prueba EEG o un experimento relacionado al área [15]. Este método fue desarrollado para asegurar un estándar, a modo de que los estudios de un sujeto pudieran compararse a lo largo del tiempo, y la vez pudieran ser comparados con los de otra persona.

Las coordenadas utilizan letras para identificar el lóbulo y un número para identificar la ubicación del hemisferio. Las letras F, T, C, P y O representan los lóbulos frontal, temporal, central, parietal y occipital, respectivamente. Los números pares se refieren a las posiciones de los electrodos en el hemisferio derecho, mientras que los números impares se refieren a los del hemisferio izquierdo como se muestra en la Figura 1-4.

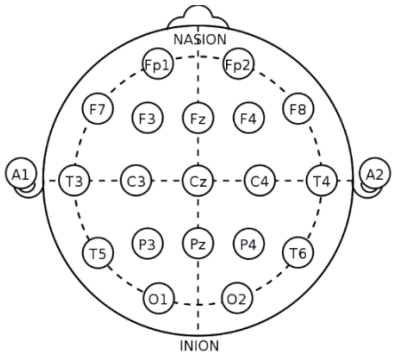

Figura 1-4: Ubicación de los electrodos para realizar un EEG.

1.3.6 Señal electroencefalográfica

Son caracterizadas por sus amplitudes extremadamente pequeñas del orden de los micro volts. Son difíciles de interpretar ya que representan la actividad comprendida de billones de neuronas transmitidas por las membranas del cerebro, fluidos y el cuero cabelludo.

Las principales ondas que se presentan en los EEG son [16]:

Delta: Tiene una frecuencia de 3 Hz o menos. Tiende a ser de mayor amplitud y más lentas.

Theta: Tiene una frecuencia de 3,5 a 7,5 Hz y se clasifica como actividad "lenta".

Alfa: Su frecuencia varía entre 7,5 y 13 Hz. Por lo general se ve mejor en las regiones posteriores de la cabeza.

Beta: La actividad beta es una actividad "rápida". Tiene una frecuencia de 14 Hz o superior. Por lo general se ve en ambos lados en distribución simétrica y es más evidente frontalmente. Es el ritmo dominante en los pacientes que están en estado alerta, ansiosos o tienen los ojos abiertos.

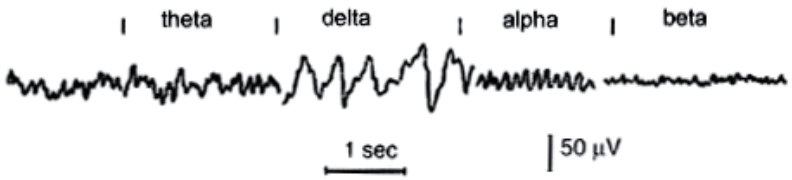

Figura 1-5: Ondas principales del EEG [16].

1.4 Selección del ASIC adecuado

Un circuito integrado de aplicación específica, ASIC por su sigla en inglés, es un circuito integrado configurable que ha sido diseñado para un propósito o aplicación específica en un dispositivo electrónico.

Tabla 1-2 Tensión y frecuencia de señales de biopotenciales [17].

Señal	Magnitud [mV]	Frecuencia [Hz]
ECG	0.4 – 10	0.125 – 150
EMG	0.05 – 10	0 – 500
EEG	0.002 – 0.1	2 – 1000

Para el caso del tema estudiado, y para poder adquirir las señales con las características que se muestran en la Tabla 1-2, se necesita un ASIC que cumpla con los siguientes requisitos [18]:

- El amplificador debe tener una ganancia alta a fin de amplificar la débil señal de entrada y así esta no se veas afectada por el ruido interno del amplificador. Además, debe tener alta impedancia de entrada de modo que proporcionen una carga mínima de la señal de biopotencial. Una carga eléctrica alta puede dar lugar a distorsiones.
- Los amplificadores operacionales tienen una cantidad de corriente que fluye a través de sus conexiones de entrada para que los transistores de entrada estén correctamente sesgados. Este flujo de corriente debe ser limitado para la seguridad del individuo en que se colocan los electrodos y debe ser inferior al orden de [nA].
- La relación de rechazo de modo común (CMRR) es la capacidad de un amplificador para rechazar la ganancia común y aumentar la señal diferencial. Cuanto mayor sea el CMRR, menor es el ruido de modo común.

Teniendo en cuenta los requerimientos mencionados, se procede a seleccionar el ASIC adecuado. Para esto se analizan ASICs de la marca Texas Instruments, la cual ha desarrollado una familia de circuitos integrados para la captura de Biopotenciales. También se consideran dos ASICs de la marca Analog Devices.

Cada dispositivo tiene distintas características, tales como resolución de bits para la digitalización de las señales, número de canales o precio.

Además se hizo un conteo de resultados obtenidos al buscar en google cada uno de los ASIC, con el fin de identificar la popularidad de cada uno, esperando que los más populares tengan más referencias respecto a problemas que puedan surgir cuando se esté en la etapa de desarrollo del proyecto. En la Tabla 1-3 se ingresa además el biopotencial para el cual el fabricante recomienda el uso de cada chip.

Tabla 1-3: Comparación ASIC [19], [20], [21], [9], [22].

ASIC	Resolución (Bits)	Biopotencial	Canales	CMRR (dB)	Potencia (mW)	Popularidad Miles de menciones	Precio USD
ADAS1000	24	ECG.	5	110	21	70	33
AD7768	24	EEG-EMG-ECG	8	95	75	100	35
ADS1299	24	EEG-ECG.	8	110	10	27	56
ADS1298	24	EEG-EMG-ECG	8	115	6	330	42
ADS1198	16	ECG	8	105	4,4	33	29
ADS1296	24	EEG-EMG-ECG	6	115	5,3	17	35
ADS1294	24	EEG-EMG-ECG	4	115	4,1	24	24

Considerando la información entregada por la Tabla 1-3, y teniendo en cuenta que todos los ASIC considerados utilizan comunicación SPI, se selecciona el chip de Texas Instruments ADS1298.

1.4.1 ADS 1298

El circuito integrado ADS1298 es una interfaz analógica (Analog Eront-End) de baja potencia, diseñado para mediciones de biopotenciales. Tiene 8 canales de entrada con un ADC $\Delta\Sigma$ (delta-sigma) de hasta 24 bits para cada canal, soportando muestreo simultáneo de todos los canales a una tasa máxima de 32 [ksps] (ver Figura 1-6).

El chip es especialmente adecuado para sistemas de monitoreo de ECG portátiles con muchas características propias implementadas para este propósito (Terminal de Goldberg y Wilson Center, detección de PACE y Right Leg Drive (RLD) de apoyo) [9].

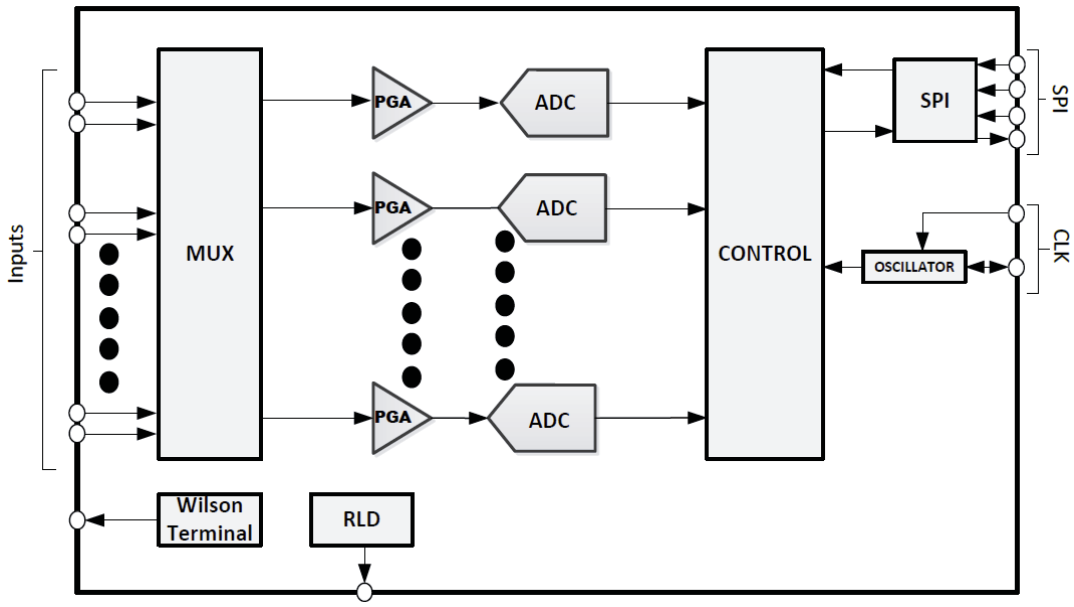

Figura 1-6: Diagrama de bloques del ADS1298 basado en [9].

1.5 Características del ADS1298

De la hoja de datos se detallan las siguientes características [9].

1.5.1 Características eléctricas

La fuente de alimentación para el ADS1298 puede ser unipolar o bipolar; la fuente análoga se extiende de 2.7 [V] a 5.25 [V] y la digital de 1.65 [V] a 3.6 [V]. Tiene un rango de temperatura de operación que es de -40 [°C] a +85 [°C]

1.5.2 Entrada analógica

La entrada analógica del ADS1298 es totalmente diferencial. El voltaje positivo de entrada es INP y la entrada negativa es INN, la señal INP-INN de red puede extenderse entre ±VREF / Ganancia, donde VREF es tensión de referencia.

1.5.3 Amplificador de ganancia programable (PGA)

El ADS1298 tiene ocho PGA de bajo ruido. Proporciona siete ajustes de ganancia de 1, 2, 3, 4, 6, 8 y 12. De forma predeterminada, esta ganancia se establece en 6, para la cual el ancho de banda es 64 [kHz] a temperatura ambiente.

1.5.4 Filtro EMI

ADS1298 proporciona un filtro EMI para proteger contra perturbaciones creadas por inducción electromagnética. El filtro EMI utilizado es un filtro RC con un ancho de banda a -3dB de aproximadamente 3 [MHz] en todos los canales.

1.5.5 Multiplexor de entrada

El ADS1298 proporciona multiplexores de entrada con propósito de configurar las opciones de conmutación de la señal fácilmente. INP y INN son separados para cada uno de los ocho bloques. El ADS1298 ofrece las opciones de entrada de electrodo normal, cortocircuito de entrada, mediciones de suministro, sensor de temperatura, señales de prueba, entradas de accionamiento de brazo derecho, entrada diferencial auxiliar, señales de excitación de desconexión y entradas auxiliares de un solo extremo. Los pines TEST PACE OUT1, TEST PACE OUT2 y RLD IN son comunes a los ocho bloques de entrada y se usan para pruebas y funciones específicas de ECG.

1.5.6 Modulador Tiempo Continuo Delta Sigma (CT Δ Σ)

Lo que diferencia al convertidor análogo digital (ADC) CT Δ Σ de otros ADC es que idealmente evita la necesidad de cualquier otro filtro anti-alias [23]. El convertidor CT Δ Σ usa sobre muestreo a una velocidad mucho mayor que la velocidad de datos de salida deseada.

En la Figura 1-7 se muestra el diagrama de bloques detallado del modulador delta sigma de primer orden. En primer lugar, la entrada analógica entra en un amplificador de diferencia, que resta (delta) el valor actual de la salida para dejar en error; este error se filtra utilizando un filtro de bucle que es básicamente un integrador (sigma). La salida del integrador se convierte de análogo a digital utilizando un ADC. Esta señal digital es devuelta y convertida a analógica usando un DAC de manera que puede ser restada de la entrada.

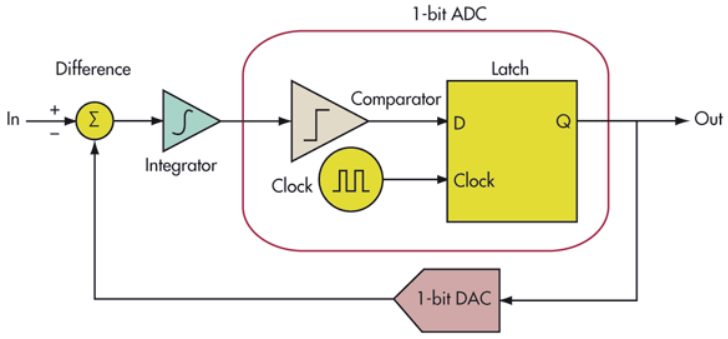

Figura 1-7: Diagrama de bloques de un oscilador $\Delta\Sigma$ de primer orden (fuente: http://www.electronicdesign.com).

El ADS1298 consta de un modulador CT Δ Σ de 16 bits de segundo orden. Un modulador CT Δ Σ de segundo orden ayuda a producir flujos de bits que producen menos ruido en la salida del filtro paso bajos. Se produce simplemente conectando en cascada otra etapa de la entrada al modulador de primer orden. El modulador muestrea la señal de entrada a la velocidad de FMOD = FCLK / 4 para el modo de alta resolución y FMOD = FCLK / 8 para el modo de baja potencia, donde FCLK denota la frecuencia de la señal en el pin de CLK (2048 [MHz]).

1.5.7 Filtro de paso bajos y filtro de decimación

En el ADS1298 el filtro digital en cada canal es un filtro sinc de tercer orden. El filtro sinc es básicamente un filtro pasa bajos de tercer orden, con una tasa de decimación variable. Los datos se suministran a esta sección del filtro desde el modulador a la velocidad de FMOD. El filtro sinc atenúa el ruido de alta frecuencia del modulador, luego diezma el flujo de datos para obtener la tasa de muestreo deseada. La función de transferencia en el dominio de la frecuencia del filtro sinc se muestra en la ecuación siguiente, donde N es la razón de decimación.

$$H(f) = \left| \frac{\sin\left(\frac{N\pi f}{f_{MOD}}\right)}{N \cdot \sin\left(\frac{\pi f}{f_{MOD}}\right)} \right|^3 \tag{1-1}$$

1.5.8 Referencia

El ADS1298 ofrece la opción de utilizar referencias internas y externas. Cuando se utiliza una alimentación analógica de 3 [V] se debe utilizar una tensión de referencia de 2,4 [V] y cuando se utiliza una alimentación de 5 [V] puede ser de 2,4 [V] o 4 [V].

1.5.9 Reloj

El ADS1298 tiene la opción de usar relojes internos y externos. El reloj interno funciona a una frecuencia de 2.048 [MHz]. El pin de selección de reloj CLKSEL se utiliza para seleccionar el reloj externo o interno. La frecuencia de reloj externo debe oscilar entre 0.5 [MHz] y 2.25 [MHz].

1.5.10 Formato de datos

El ADS1298 emite un valor de 24 bits de datos por canal en formato binario de dos complementos, con el bit más significativo primero. A una velocidad de datos de 16 [ksps] y 32 [ksps] se tienen códigos faltantes y sólo tienen 19 bits y 17 bits de resolución respectivamente.

1.5.11 Comunicación

El ADS1298 necesita ser programado para operar a una velocidad de datos específica. La única manera de comunicarse con el ADS1298, para programar y leer datos de conversión desde el chip, es a través de la comunicación SPI (del inglés Serial Peripheral Interface).

SPI es un estándar de comunicación que sirve para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj [24].

En este estándar, los dispositivos se comunican en modo maestro / esclavo en el que el dispositivo maestro inicia el marco de datos (ver Figura 1-8). La comunicación SPI tiene cuatro señales lógicas.

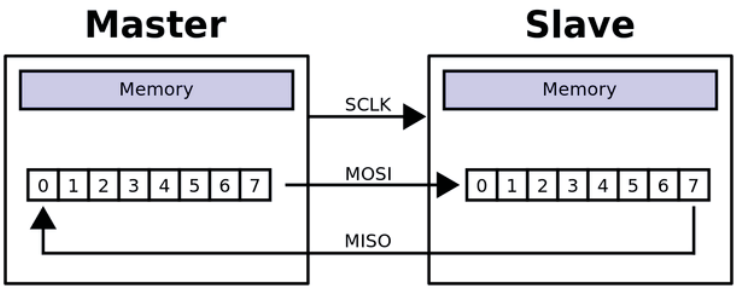

Figura 1-8: Comunicación SPI (fuente: https://vidaembebida.wordpress.com)

1.5.12 Encapsulado

El ADS1298, está disponible en dos encapsulados, los que se muestran en la Figura 1-9.

Los encapsulados tipos QFP (Quad Flat Pack) son cuadrados, con terminales en ala de gaviota en sus cuatro lados. Cuando estos últimos son de paso fino hay que tener mucho cuidado en su manipulación para no dañarlos.

Por su parte el encapsulado BGA (Ball Grid Array) posee sus pines en forma de bolas de soldadura de estaño, ubicadas en la superficie inferior del componente. Se elimina la complicación de pitch demasiado finos, pero la soldadura deja de estar visible, esto genera un problema considerando que si el chip queda mal soldado queda inutilizable.

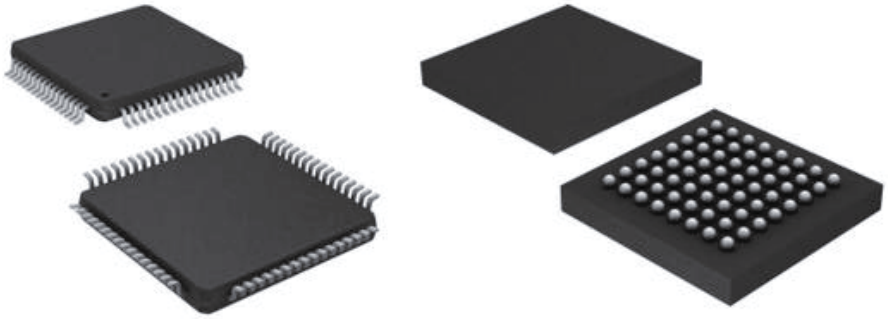

Figura 1-9: Encapsulado del ADS1298, 64-TQFP (izquierda) y 64-NFBGA (derecha) (fuente: www.digikey.com)

Finalmente, para no correr el riesgo de perder el chip debido a una soldadura displicente, se opta por el ADS1298 con encapsulado 64-TQFP.

En la Figura 1-10 se muestran los terminales del ADS1298 IPAG (con encapsulado 64-TQFP).

Como no es posible montar el dispositivo directamente en una protoboard, se necesita de un adaptador para poder realizar las pruebas en el laboratorio, tal como el de la Figura 1-11.

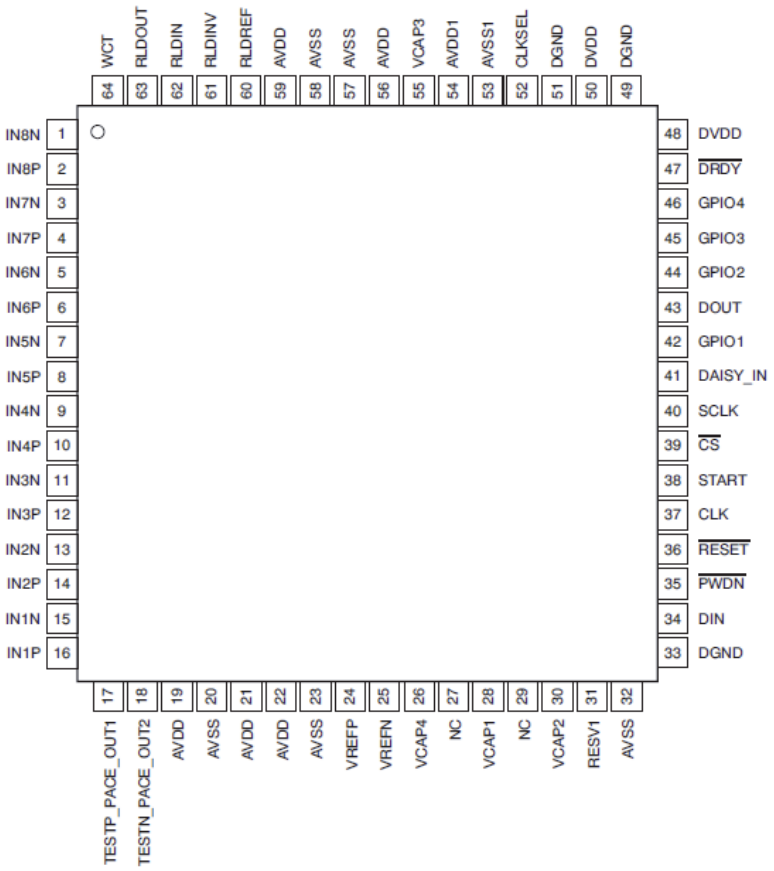

Figura 1-10: Terminales del ADS1298 IPAG [9].

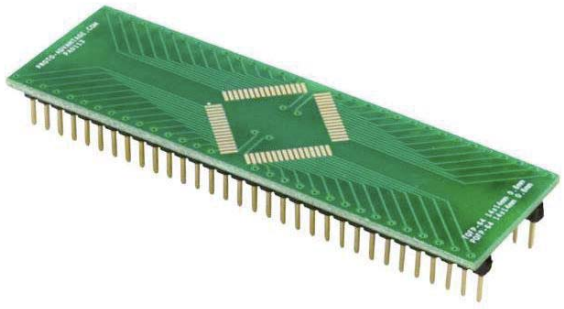

Figura 1-11: Adaptador 64-TQFP a DIP (fuente: www.digikey.com)

1.5.13 Mercado

Se cotizaron valores en distintas tiendas dedicadas al mercado de la electrónica, con presencia nacional e internacional. El desglose de precios, tanto para el chip como para el adaptador, se muestran en la Tabla 1-4.

Tabla 1-4: Valores en el mercado.

Proveedores	ASIC	Adaptador
Mouser	$41.8 USD	$270 USD
RS Chile	$36.616 CLP	$26.379 CLP
Digikey	$41.81 USD	$15.79 USD
Newark	$41.81 USD	-
TI Store	$43.83 USD	-

1.6 Propuesta de evaluación del desempeño del ADS1298

Dado que la señal de más interés para el LABSEI es la señal de ECG, se decide implementar un electrocardiógrafo basado en el ADS1298 en su parte análoga, para posteriormente realizar el procesamiento digital de la señal, y su visualización, en una interfaz gráfica diseñada en Python.

Para diseñar el proyecto, primero se debe comprender completamente la señal que se procesará en el sistema. La Figura 1-12 muestra la señal electrocardiográfica, tal como aparece en la entrada del sistema de medición de ECG. Consta de tres componentes: la señal de ECG real (diferencial), la compensación del electrodo diferencial y la señal de modo común [10].

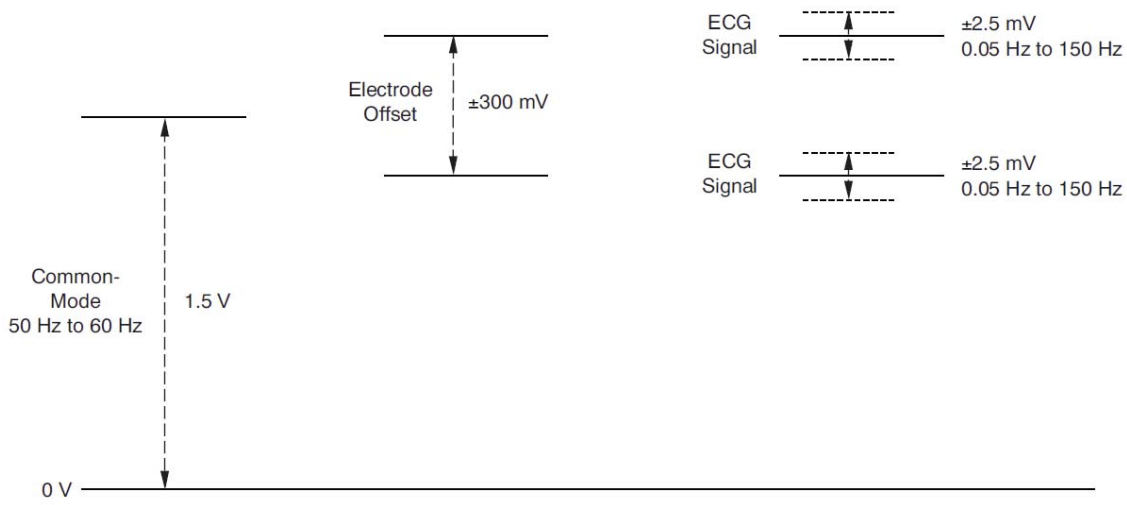

Figura 1-12: Características de la señal de ECG [10].

La señal de ECG diferencial real que aparece entre los electrodos en cualquier configuración de derivación, está limitada a ± 5 mV de magnitud y de 0,05 Hz a 150 Hz de frecuencia. Estos valores determinan las características que debe tener la interfaz analógica, en este caso, el ADS1298.

La interfaz piel-electrodo proporciona una compensación DC adicional de aproximadamente 300 [mV]. Este desplazamiento debe manipularse de manera que la cadena de señal no esté saturada.

Además de estas dos señales, el cuerpo humano puede captar grandes señales de interferencia de líneas eléctricas, luces fluorescentes, etc. Esta interferencia puede manifestarse como una señal de modo normal o una señal de modo común. La interferencia de modo normal puede mitigarse con un filtro rechazo de banda de 50 Hz / 60 Hz, dependiendo de la frecuencia de la red, implementado por software. La interferencia en modo común, por otro lado, generalmente se contrarresta de una de las siguientes formas:

- Aumentando, tanto como sea posible, el aislamiento de tierra de la interfaz analógica;
- Aumentar el rechazo en modo común de los circuitos de procesamiento de señal (del orden de 100 dB);
- Conducir el cuerpo del paciente con una señal de modo común fuera de fase (también llamada como el impulso de pierna derecha).

El ADS1298, tiene un alto rechazo de modo común, del orden de los 115 [dB], y además incluye la generación del impulso de pierna derecha.

1.6.1 Alimentación del ADS1298

El ADS1298 tiene tres fuentes de alimentación: AVDD, AVDD1 y DVDD. AVDD1 proporciona el suministro al bloque de bomba de carga y tiene transitorios en fCLK. Se debe conectar AVDD1 y AVSS1 a AVDD y AVSS respectivamente. Además, hay que conectar cada suministro del ADS1298 con condensadores de desacoplo, estos deben ser cerámicos sólidos de 1 [µF] y 0.1 [µF]. La alimentación del ADS1298 debe ser unipolar o bipolar [9], para facilitar el diseño del circuito, se opta por la alimentación unipolar.

La alimentación del circuito, proviene directamente del puerto USB del PC, de este modo la fuente análoga AVDD es de 5 [V], y se emplea un regulador de tensión para que la fuente digital DVDD sea 3.3 [V]

1.7 Conclusión del capítulo

Las señales bioeléctricas son en general de baja magnitud, alcanzando el orden de [mV] para la señal de ECG y el orden de [µV] para la señal de EEG.

El ancho de banda requerido para cada señal es distinto, un EEG por ejemplo tiene un ancho de banda del orden de [kHz], pero la mayor parte de la información se concentra en una frecuencia menor a 30[Hz] [5]. Algo similar sucede con el caso de EMG, en el cual la mayor información está contenida entre 50 [Hz] y 150 [Hz] [3].

Todos los proyectos realizados anteriormente en Labsei comparten una característica común; tienen una parte analógica y otra digital, ambas necesarias para adquirir, amplificar y convertir la señal bioeléctrica a digital, y de este modo, poder emplearla según el propósito de cada

proyecto, y en todos la parte análoga está compuesta por varios circuitos integrados de propósito general, por lo que el diseño circuital abarca un espacio considerable.

El empleo de un ASIC reemplaza en parte la etapa de diseño, disminuyendo el tiempo dedicado al desarrollo de éste, ocupando un espacio mucho menor. Además, agrega exactitud en las características definidas por el fabricante, por ejemplo para la ganancia o frecuencia de muestreo.

Emplear el ASIC como base para el proyecto permite ampliar conocimientos con fines académicos, de docentes y estudiantes, sobre el empleo de esta tecnología.

Se encontraron varios ASIC diseñados para la adquisición y digitalización de biopotenciales, entre ellos, a partir de características tales como desempeño, recomendaciones del fabricante, popularidad y precio, se seleccionó el ADS1298, el que es un circuito integrado diseñado para adquirir ECG, EME y EEG. El ASIC seleccionado posee 8 canales, cada uno con un convertidor ADC delta sigma de 24 bit de resolución. Su tasa de muestreo alcanza los 32 [ksps].

Para no correr el riesgo de dejar inutilizable el chip, debido a una soldadura deficiente, se opta por usar el ADS1298 IPAG, que tiene un encapsulado 64-TQFP. Además, por ser de montaje superficial, se debe adquirir un adaptador para montar el circuito en un tablero de experimentación.

2 Evaluación del ASIC seleccionado

El ADS1298 necesita ser programado para operar a una velocidad de datos específica. La única manera de comunicarse con el ADS1298, para programar y leer datos de conversión desde el chip, es a través de la comunicación SPI (del inglés Serial Peripheral Interface). Es por ese motivo que se busca un microcontrolador, capaz de comunicarse vía SPI con el ADS1298 y además vía USB con un computador, para poder programarlo.

En la Figura 2-1 se muestra la configuración de pines del PIC18F4550, este microcontrolador posee un puerto serial SPI de hasta 10 [Mb/s] y uno USB de hasta 12 [Mb/s], tiene una memoria de programa flash de 32 [kbytes], una RAM de 2048 [Bytes] y una EPROM de 256 [Bytes] [25]. Estas características, sumado a la gran disponibilidad en el mercado, el precio del microcontrolador - $4250 [26]- y las recomendaciones averiguadas [11], justifican la elección del dispositivo para el proyecto.

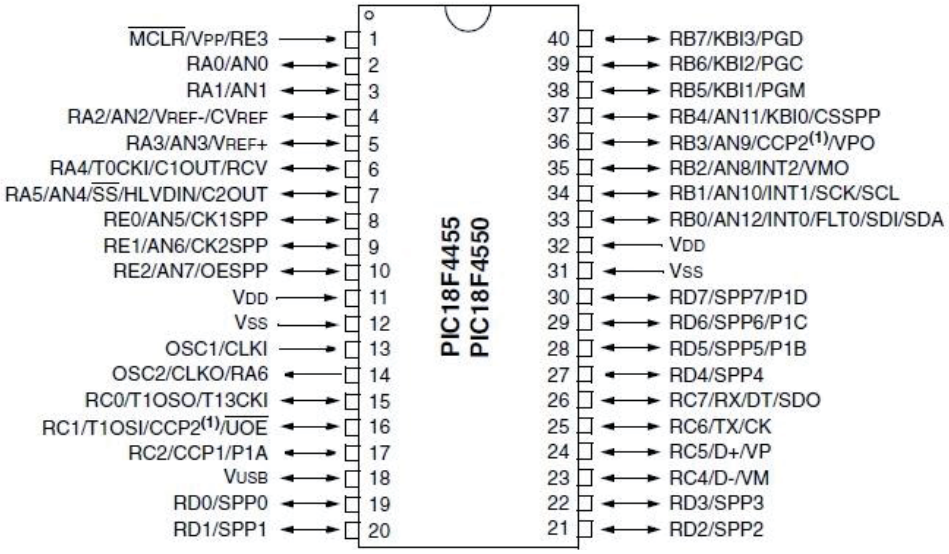

Figura 2-1: Terminales del PIC18F4550 [22].

2.1 Conexiones en el ADS1298

Las conexiones se proponen de acuerdo a las configuraciones mostradas en las hojas de datos del ADS1298 y del PIC18F4550 [9] [25]. Además, se usa como apoyo las guías de Texas Instruments [27], [28] y las consultas realizadas permanentemente en el foro de la marca [11].

El diseño que será montado en el tablero de experimentación se muestra en la Figura 2-2.

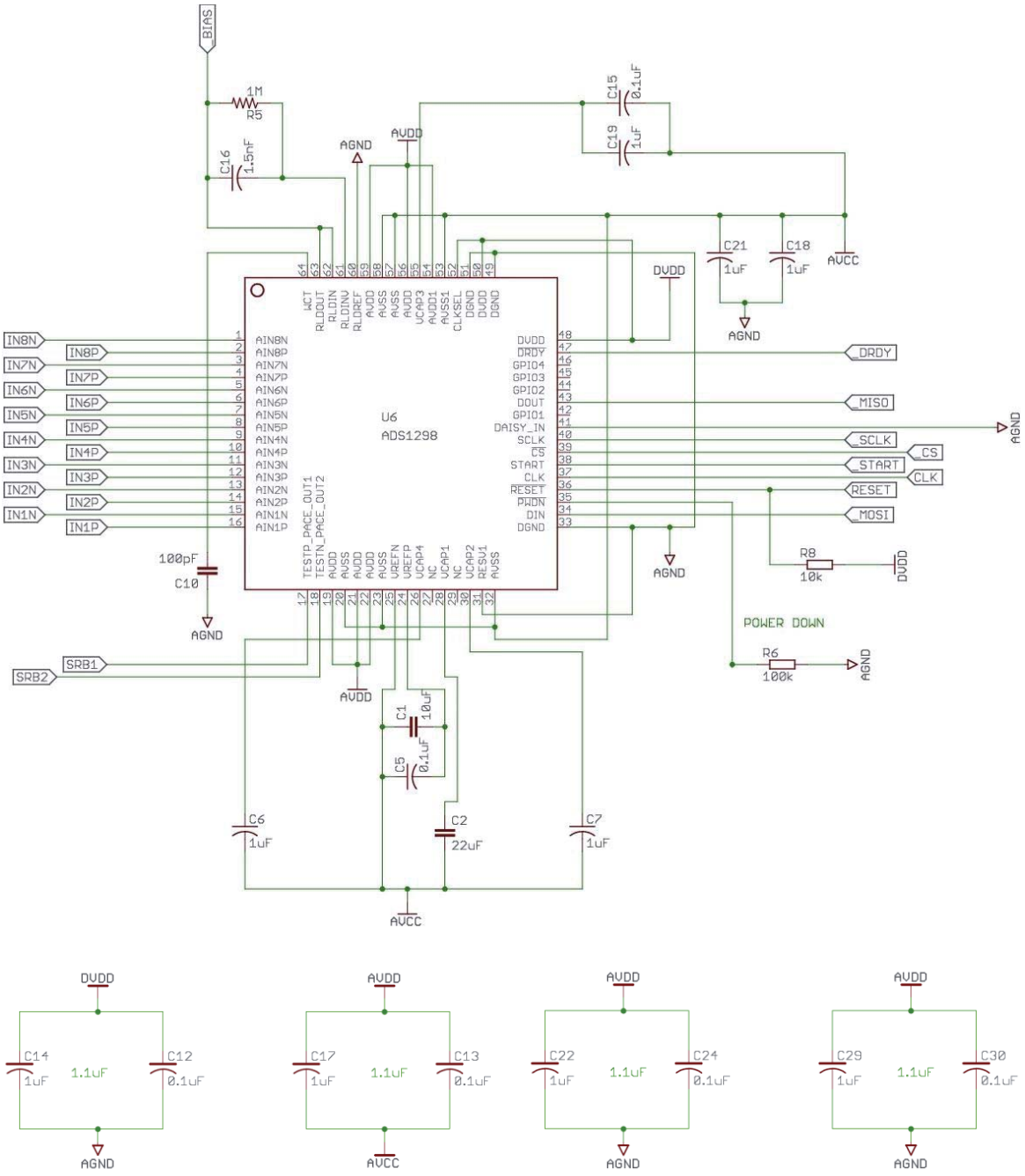

Figura 2-2: Conexiones en el ADS1298 [11].

Para mitigar el ruido de modo común, el ADS1298 permite conectar a las entradas negativas de cada canal, la señal de pierna derecha, en este caso el terminal “BIAS” en la Figura 2-2, por medio de una resistencia. Pero también permite conectar el terminal WCT como señal de pierna derecha, en este caso la conexión reemplaza a la señal negativa de cada canal [9].

La comunicación entre el ADS1298 y el microcontrolador PIC18F4550 se realiza conectando los pines como se muestra en la Figura 2-3. El microcontrolador actúa como maestro y el ADS1298 lo hace como esclavo.

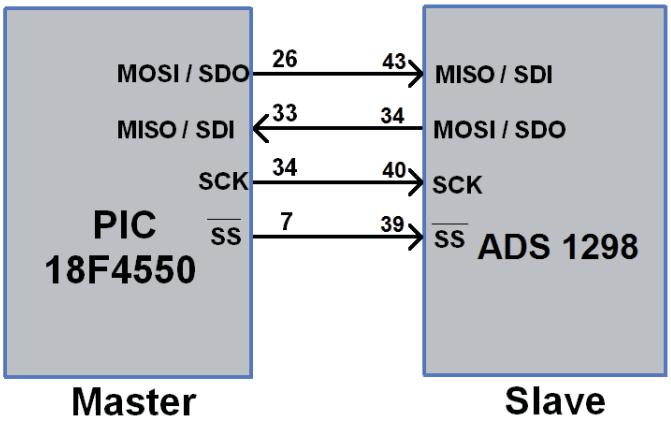

Figura 2-3: Comunicación entre ADS1298 y PIC18F4550 (modificación propia).

Para la comunicación entre el microcontrolador y el computador se utiliza el puerto USB como alimentación, como se muestra en la Figura 2-4.

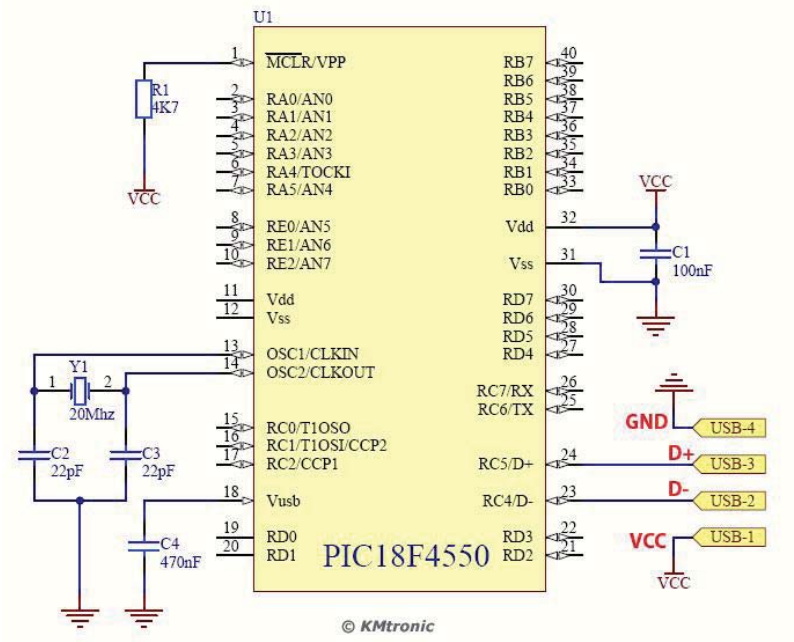

Figura 2-4: Comunicación entre el microcontrolador y el computador (fuente: http://www.edaboard.com).

2.1.1 Programación del ASIC

El programa cargado en el microcontrolador se desarrolla modificando los registros que permitan usar el ADS1298 del modo deseado. Estos registros son los siguientes:

ID: Registro que permite identificar el dispositivo y seleccionar el número de canales.

CONFIG1: Permite elegir el modo de alta resolución o de bajo consumo, y la frecuencia de muestreo para cada uno de los canales del ADS1298. Para el proyecto se fija en el modo de bajo consumo muestreando a 4 Ksps.

CONFIG2: Configura las señales de prueba del chip, estas señales se dejan en sus valores por defecto, de este modo se generan internamente.

CONFIG3: Configura las referencias y el funcionamiento de la RLD. Se fija el voltaje de referencia por defecto (2.4 [v]) y se activa la conexión externa del circuito de pierna derecha RLD.

LOFF: Configuración de inicio para el ADS1298, se configura para que el dispositivo inicie al conectar la fuente de alimentación.

CHnSET: Configuración para cada uno de los canales. Para el proyecto se dejan desactivados los canales del 1 al 7 y se deja encendido el canal 8, fijando la ganancia de los amplificadores de ganancia programable PGA en 6.

RLD_SENSP y RLD_SENSN: Permiten elegir la parte positiva o negativa de cada uno de los canales como señal RLD, para el proyecto se usará un electrodo que capture dicha señal, de modo que se dejan deshabilitados ambos registros.

LOFF_SENSP y LOFF_SENSN: Permite elegir la señal negativa o positiva de cada canal para determinar el encendido del dispositivo, como se fija el encendido de acuerdo a la alimentación, estos registros quedan deshabilitados.

LOFF_FLIP: Permite invertir la polaridad de la alimentación del circuito. Se fija el registro para no hacer cambios en la polaridad.

LOFF_STATP y LOFF_STATN: Permite saber si el electrodo positivo o negativo de cada canal está o no encendido. Dado que el registro LOFF_SENSP y LOFF_SENSN están deshabilitados, este registro se ignora.

Los registros **GPIO, PACE, RESP, CONFIG4, WCT1, WCT2,** no son utilizados para este proyecto, y por esto es que se fijan todos en 0x00.

Los códigos de operación del ADS1298 se muestran en la **¡Error! No se encuentra el origen de la eferencia.**. Cada uno de estos códigos requiere una cierta cantidad de ciclos del reloj para iniciar o para terminar, los que se consideran en la etapa de comunicación SPI, pues realizar, por

ejemplo, una lectura de los valores del ADS1298, en un transición entre el estado standby al estado de encendido, arrojará datos erróneos para el análisis.

COMMAND	DESCRIPTION	FIRST BYTE	SECOND BYTE
SYSTEM COMMANDS			
WAKEUP	Wakeup 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 back.	0001 0010 (12h)	—
REGISTER READ COMMANDS			
RREG	Read *n nnnn* registers starting at address *r rrrr*	001*r rrrr* (2xh)(2)	000*n nnnn* (2)
WREG	Write *n nnnn* registers starting at address *r rrrr*	010*r rrrr* (4xh)(2)	000*n nnnn* (2)

Figura 2-5: Códigos de operación del ADS1298 [9].

2.1.2 Microcontrolador

La programación del ADS1298 se realiza por medio del microcontrolador PIC18F4550, elegido para el desarrollo del proyecto por permitir una comunicación mediante SPI y USB HID, la primera característica empleada para comunicarse con el ADS1298 y la segunda para hacerlo con un PC que recibirá los datos adquiridos. De este modo el PIC, funcionará como un dispositivo de enlace en el proyecto.

El código para programar el microcontrolador se desarrolla en MPLAB X Integrated Development Environment (IDE) [29] empleando el compilador C COMPILER de CCS [30]. Las librerías y comandos del compilador se detallan en el manual que se usa para desarrollar el código [31].

En el programa del microcontrolador primero se modifican los registros del PIC18F4550 para generar la señal de reloj del ADS1298 y luego se modifican los registros del ADS1298 para que funcione con las características deseadas.

2.1.3 Comunicación entre el ADS1298 Y EL PIC18F4550

La comunicación entre el chip y el microcontrolador se realiza mediante el protocolo SPI, donde el PIC actúa como maestro en la comunicación.

El PIC funciona con una frecuencia de 48 [MHz], esta frecuencia permite la comunicación USB FULL SPEED entre el PIC y el PC [25], necesaria para enviar la tasa de datos que el proyecto requiere. Para la comunicación SPI el ADS1298 requiere una frecuencia entre 1.95 [MHz] y 2.4 [MHz]. La frecuencia del reloj para la comunicación SPI se fija en 2[MHz].

El PIC18F4550 permite fijar el reloj de la comunicación SPI dividiendo el reloj principal del sistema o empleando la mitad de la frecuencia del Timer 2. Para lograr la frecuencia de 2 [MHz], se opta por emplear el Timer 2 oscilando a 4[MHz].

Las funciones empleadas para escribir y leer datos en chip por SPI son **SPI_WRITE()** y **SPI_READ()** respectivamente, después de las cuales se debe esperar 8 ciclos de reloj SPI para no perder datos.

2.1.4 Programación del ADS1298

La programación del ADS1298 se realiza, como se indica en su hoja de datos [9], enviando tres comandos seguidos como se muestra en la Figura 2-6.

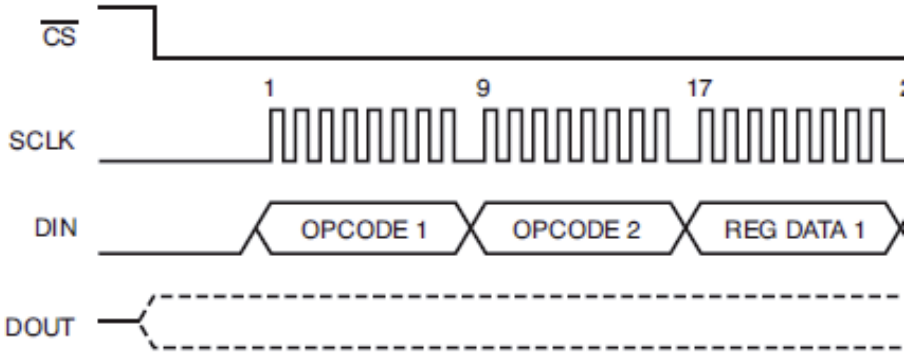

Figura 2-6: Comandos para modificar registros del ADS1298 [9].

El primer comando tiene el formato **010R RRRR**, donde **R**, es la dirección donde se escribirá el registro. El segundo comando es **000N NNNN**, donde **N** es el número de registros que se escribirán. Por último se envía el valor del registro para la configuración requerida.

A grandes rasgos, la programación del chip fija el Canal 1 para la conexión de los electrodos, con ganancia 6 y muestreo a 4 [ksps], fija el voltaje de referencia en 4 [v], y selecciona la señal RLD a partir del electrodo conectado a la pierna derecha y a la parte positiva y negativa del Canal 1.

2.1.5 Envío de datos al PC

Cuando el ADS1298 ya está programado, desde el PIC se pone el pin START del chip en alto, de este modo comienza la captura y conversión de datos. Y cuando hay un paquete de datos listos para leer, el ADS1298 pone encero su pin DRDY, señal que se emplea para adquirir los datos en el PIC18F4550.

Cada vez que el pin DRDY está en cero se ejecuta la secuencia de lectura de datos mediante el comando SPI_READ(0). En la comunicación SPI se intercambia un byte entre el maestro y el esclavo, como en este caso no importa la información a enviar desde el maestro, sino la recibida, se envía un cero.

La salida de datos del ADS1298 se muestra en la Figura 2-7, los primeros 24 bit (3 bytes), corresponden al estado del dispositivo, y es por esto que son leídos, pero desechados. Los siguientes 3 bytes corresponden a la lectura del Canal 1, en formato MSB.

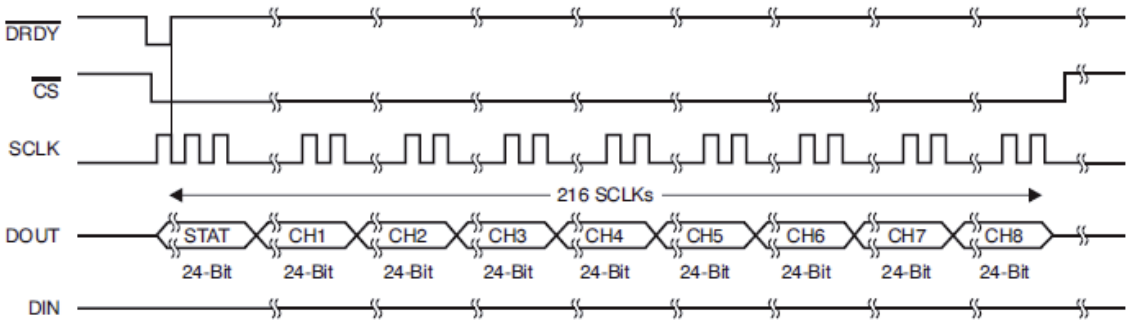

Figura 2-7: Protocolo de salida de datos del ADS1298 [9].

El PIC18F4550 es configurado como dispositivo USB HID, siendo interrogado por el PC cada 1 [ms]. Cuando el microcontrolador haya recibito un total de 20 datos desde el ADS1298, 60 bytes en total, estos serán enviados al PC con una velocidad efectiva de 12 [kB/s], o 96 [kb/s].

2.1.6 Comunicación entre el PIC y el PC

El módulo de adquisición de datos usado es que se emplea en el curso de Instrumentación Electrónica [32], al que se le agregaron modificaciones para recibir datos de 24 bits.

El sistema está desarrollado en Python, y su función es recibir las muestras provenientes del microcontrolador, desplegarlas en pantalla en tiempo real y almacenarlas en un archivo de texto para su posterior procesamiento digital.

La interfaz gráfica del programa se muestra en la Figura 2-8.

Cada vez que se presiona el botón **Conectar**, el programa comienza a recibir datos desde el puerto USB, siempre que estos existan, si no hay datos, permanece a la espera de que hayan. En el momento en que se presiona el botón se crea un archivo de texto donde se guardan los datos recibidos. Dado que la salida del ADS1298 es de 24 bit, se agrega una etapa al programa para unir 3 datos, cada uno de 8 bits, en uno, siguiendo la siguiente ecuación:

$$VALOR = 256^4 \cdot \boldsymbol{BYTE1} + 256^2 \cdot \boldsymbol{BYTE2} + 256 \cdot \boldsymbol{BYTE3} \tag{2-1}$$

El ancho del eje x la pantalla se fija para una cierta cantidad de muestras, cuando estas muestras se cumplen, la pantalla, que ya está llena, se borra y se refresca con nuevos datos.

Para terminar el proceso se presiona el botón nuevamente, ahora lleva el nombre **Desconectar**, y de este modo se puede tener acceso a los datos adquiridos, que están en el archivo de texto creado.

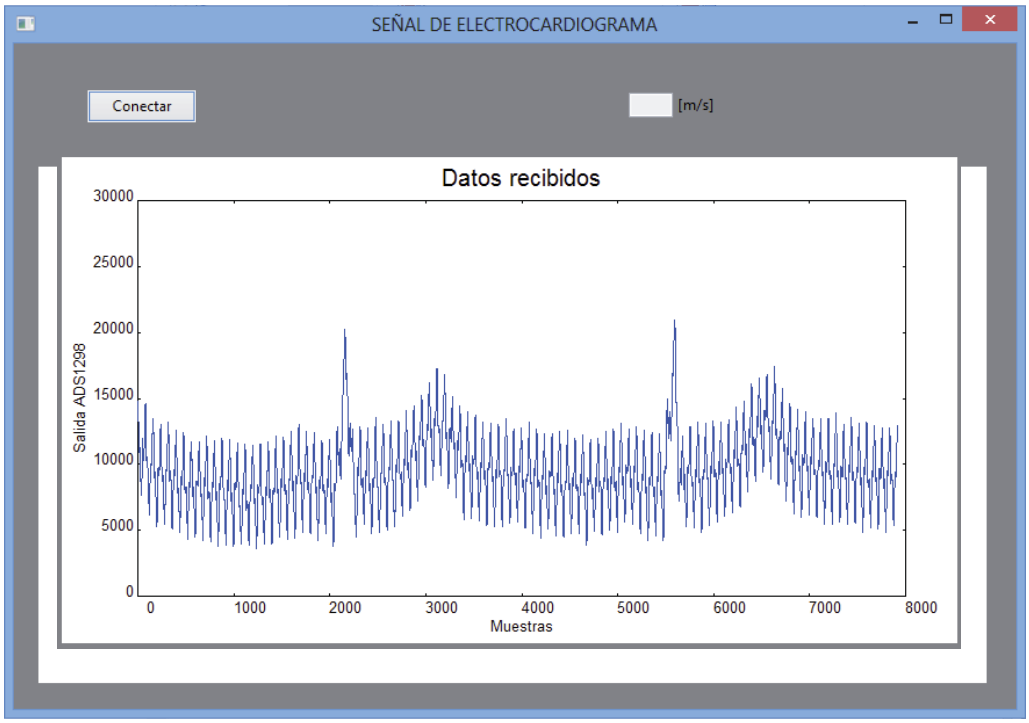

Figura 2-8: Interfaz gráfica sistema de adquisición de datos.

2.2 Diseño de la interfaz gráfica de usuario

Basándose en la interfaz mostrada en la Figura 2-8, se diseña la interfaz gráfica final del proyecto, como se muestra en la Figura 2-9. En la figura se puede apreciar que se agregaron dos botones, el botón “Filtro”, que permite filtrar en tiempo real la señal que se está recibiendo desde el puerto USB. El filtro diseñado es un filtro FIR de orden 1024, pasa bajos que corta a 35 [Hz], misma frecuencia de corte empleada en un ECG de monitoreo.

Para realizar el filtrado, ingresan 4000 muestras al filtro, equivalentes a un segundo de muestras, y se obtiene una muestra filtrada, luego, cuando hay veinte nuevas muestras, ingresan nuevamente 4000 muestras al filtro y se obtiene una segunda muestra filtrada. El programa desarrollado en Python permite desplegar la señal de ECG sin filtrar, es decir, tal cual como llega desde el ADS1298, o filtrándola, en tiempo real. Presionado el botón “Filtro”, es posible in cambiando de forma inmediata la manera en que se visualiza la señal.

Para desplegar las muestras sin filtrar, el programa muestra una muestra cada veinte que recibe. Disminuyendo la tasa de muestreo de 4000 [m/s] a 200 [m/s].

Además del botón para filtrar, se agrega el “Grabar”, si dicho botón es presionado, las muestras que llegan desde el ADS1298 son guardadas en un archivo de texto, para posteriormente poder realizar algún tipo de procesamiento digital a la señal, este archivo se guarda automáticamente cuando se termina la conexión por medio del puerto USB y se actualiza automáticamente si se vuelve a establecer la conexión y se presiona nuevamente grabar.

En la parte inferior de la interfaz gráfica se muestran dos recuadros, en uno de ellos se despliega la frecuencia de muestreo de datos, calculada dentro del mismo programa desarrollado en Python, y en el otro se muestran los latidos por minuto del paciente. Para calcular dicho valor, se crea una etapa de código que busca los peak de la señal de ECG para calcular la cantidad de muestras que separa a cada uno, luego, teniendo en cuenta la tasa de muestreo de 4000 [m/s], se procede a calcular la frecuencia cardiaca en latidos por minuto.

Figura 2-9: Interfaz gráfica final.

Los colores de la interfaz gráfica se definieron a partir de un estudio de varios monitores de electrocardiograma, donde se pudo observar que predomina el fondo negro con la señal en color verde.

2.3 Conclusión del capítulo

La manera en que se montará el ASIC en el tablero de experimentación se diseña a partir de las recomendaciones de la Hoja de Datos del ADS1298, las recomendaciones de las hojas técnicas de la marca Texas Instruments y el foro de consultas de la misma marca. Además se siguen las recomendaciones de los profesores Guía y Correferentes, a fin de obtener el mejor resultado.

Debido al especial cuidado que se debe tener con el ADS1298, para no dañarlo producto de descargas electrostáticas, todo el circuito es montado en el tablero de experimentación antes de montar el chip. De este modo, el ADS1298 y su adaptador se agregan al diseño circuital final, luego de haber realizado todas las pruebas previas que aseguren la no existencia de corto circuitos en el tablero de experimentación.

El proceso de adquisición de los datos provenientes del ADS1298 consta de dos etapas, primero los datos llegan mediante SPI al microcontrolador, y luego estos son enviados desde éste al PC mediante el puerto USB.

Se debe tener especial precaución en la comunicación SPI, respetando los tiempos de establecimiento del ADS1298 luego de realizar un intercambio de datos, pues, si no se respeta este tiempo, la lectura de los datos será errónea. Para no realizar lecturas, en momentos

equivocados, se utiliza el pin DRDY del ADS1298, el cual toma un valor bajo, cada vez que hay datos listos para leer.

Los datos del ADS1298 tienen una resolución de 24 bits, es decir 3 bytes, pero en ambas comunicaciones, SPI y USB, se mueven datos de un byte solamente, y es solo en el software desarrollado en el PC que estos datos se unen para tener los datos finales.

3 Resultados

Como se ha mencionado anteriormente, la evaluación del proyecto se basa en el desarrollo de un electrocardiógrafo basado en un ADS1298, que a su vez se comunica y es programado por un PIC18F4550. Este último tiene la tarea de enviar los datos desde el chip al PC donde la señal es procesada digitalmente. El proceso descrito anteriormente se muestra en la Figura 3-1

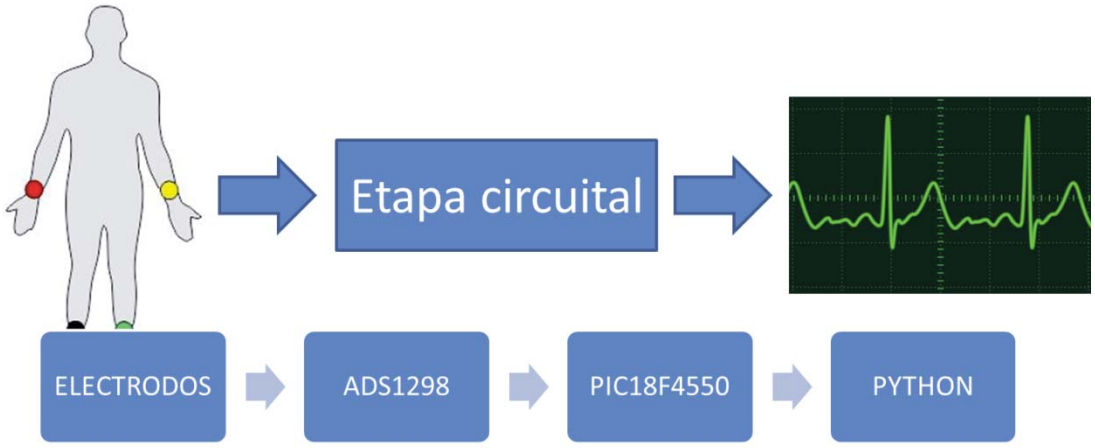

Figura 3-1: Diagrama de Bloques del proyecto.

3.1 Funcionamiento del proyecto

Como se indicó, el proyecto se puede dividir en dos bloques, la etapa análoga y la digital.

En la etapa análoga el microcontrolador envía instrucciones al ADS1298, por medio del puerto SPI, para programar su funcionamiento y recibe, por medio del mismo puerto, los datos capturados por el chip. La tasa de muestreo se fijó en 4000 [m/s] con el fin de tener suficientes muestras para hacer un buen filtrado de la señal.

En la etapa análoga se agregó un pulsador que funciona como interrupción en el microcontrolador, cada vez que la interrupción se ejecuta, el microcontrolador modifica los registros del ADS1298 para cambiar la ganancia de sus amplificadores.

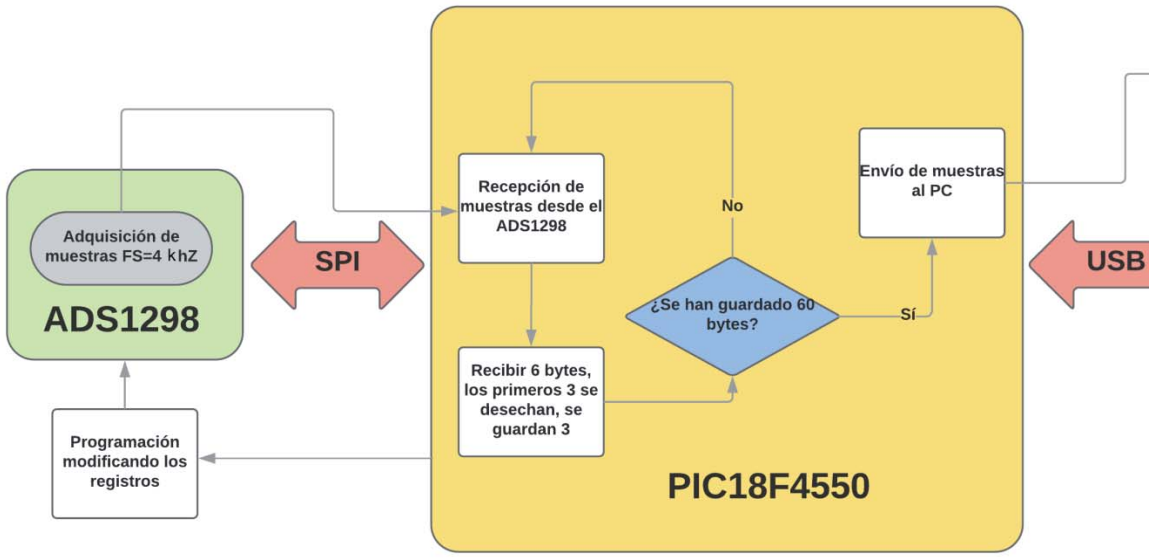

Figura 3-2: Etapa análoga.

En la Figura 3-2 se muestra un diagrama de bloques que explica el funcionamiento de la etapa análoga del proyecto, mientras que en la Figura 3-3 se muestra las etapas del procesamiento para el despliegue de la señal en la interfaz gráfica desarrollada en Python.

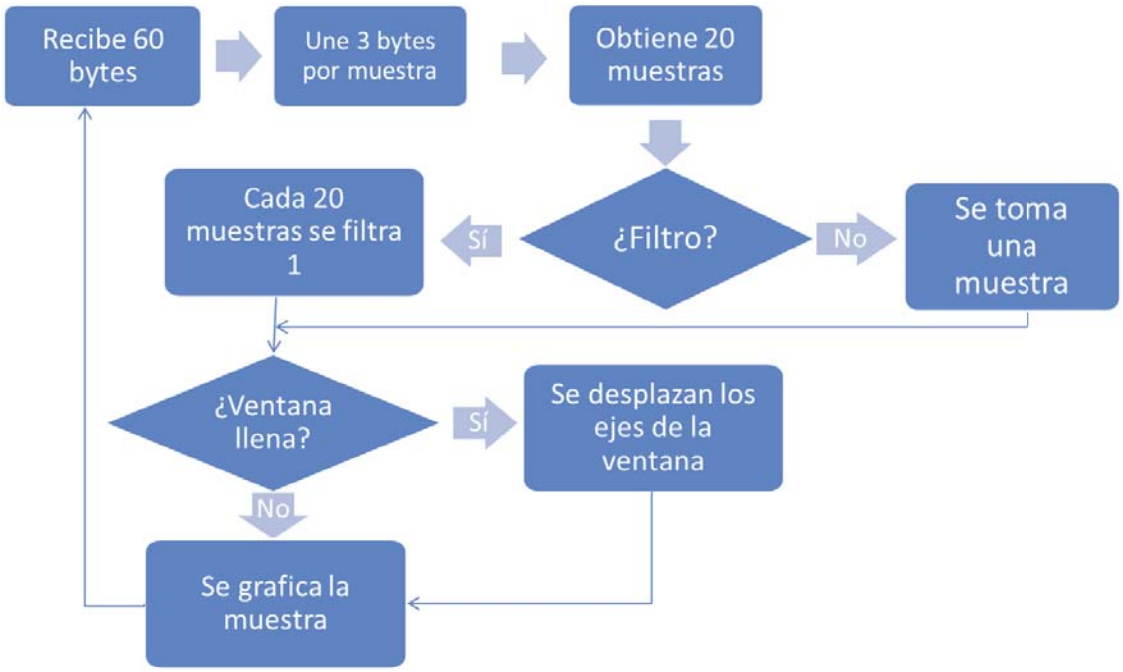

Figura 3-3: Etapa de procesamiento en Python.

3.2 Respuesta del ADS1298 a frecuencia típicas de un ECG

La evaluación de un especialista sobre un electrocardiograma es visual, y no toma en cuenta el espectro de la señal. De todos modos, y a manera de garantizar que el ADS1298 responda de manera adecuada a las frecuencias propias de un electrocardiograma, se hicieron pruebas en el laboratorio, conectando a la entrada del ADS1298 un generador de funciones. Se ingresaron sinusoides de distintas frecuencias entre 1 [Hz] y 100 [Hz], las señales se enviaron al PC desde el chip y posteriormente a dichas señales almacenadas en el PC se les realizó un análisis espectral, esperando encontrar la misma componente de frecuencia en el espectro, que la de la sinusoide generada. En la Figura 3-4, Figura 3-5 y Figura 3-6, se puede apreciar que el análisis espectral de la señal en el PC, tiene la misma frecuencia que la señal generada por el generador de funciones, esto demuestra que al ADS1298 no altera las frecuencias de las señales para los rangos probados, y por esto, no hay problemas para emplearlo en el desarrollo de un electrocardiógrafo. Cabe destacar que además en las figuras se precia la componente del ruido de la línea de 50 [Hz]

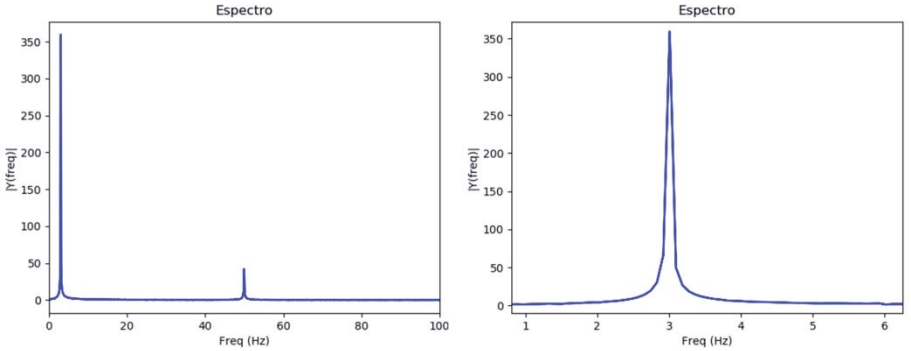

Figura 3-4: Prueba con sinusoide de 3 [Hz].

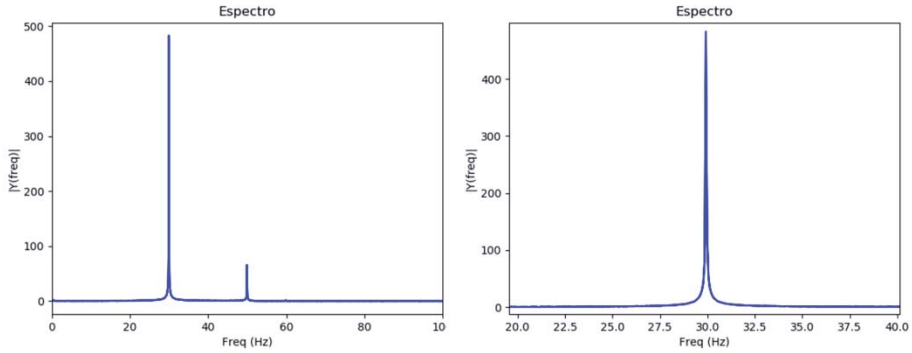

Figura 3-5: Prueba con sinusoide a 30 [Hz].

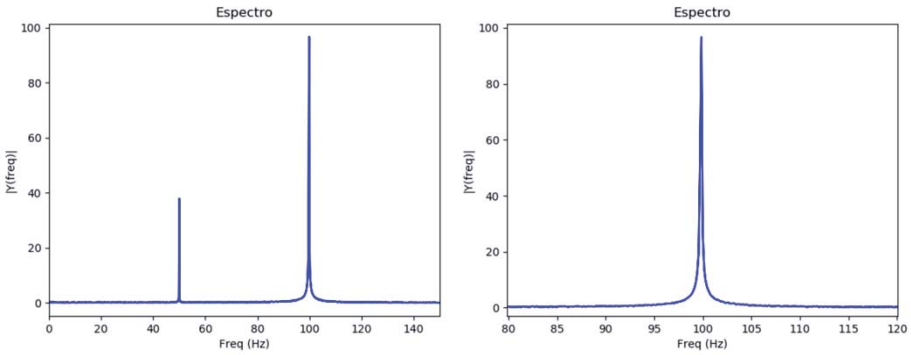

Figura 3-6: Prueba con sinusoide a 100 [[Hz].

3.3 Pruebas realizadas con pacientes reales

Como se mencionó, en análisis de la señal de electrocardiograma es visual, y por esto es que la amplitud máxima de la señal del ECG debe ocupar todo el espacio disponible en pantalla, esto permite que se pueda realizar un diagnóstico de forma correcta.

En la Figura 3-7, se aprecia el resultado final de las pruebas realizadas en pacientes reales. En la imagen se puede apreciar, que la señal esta filtrada en la primera etapa, y luego, al presionar nuevamente el botón Filtrar, la señal mostrada en pantalla es la misma que el ADS1298 envía al PC por medio del microcontrolador.

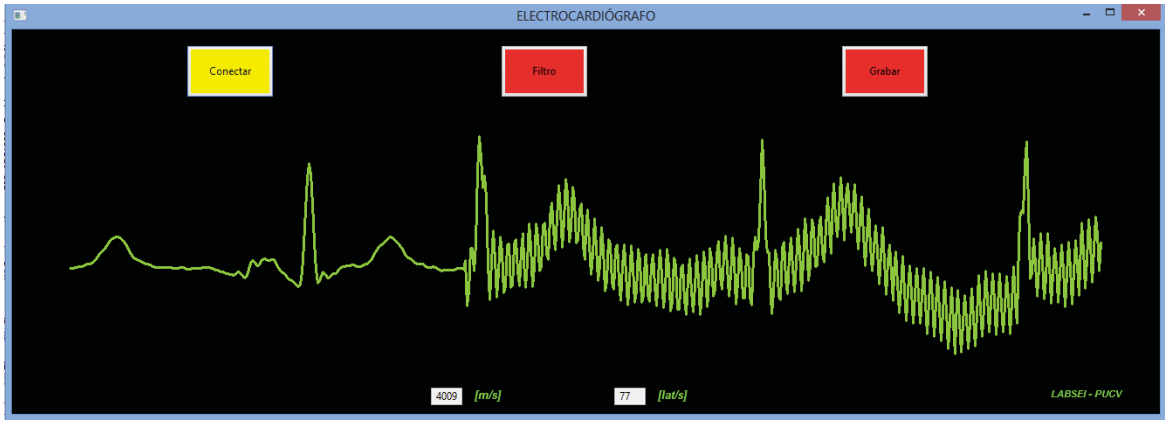

Figura 3-7: Resultado de la prueba del electrocardiógrafo.

Con la señal filtrada, es posible observar cualquier cambio anómalo en la señal de ECG, lo que permitiría a un especialista diagnosticar o descartar patologías asociadas al sistema cardiaco.

Durante la presentación del proyecto a los profesores Guía, y Coreferentes, se demostró además que el ADS1298 puede variar la ganancia del sistema en tiempo real, que para el proyecto, se definió como interrupción al presionar el botón, de este modo, es posible seguir incursionando con el proyecto, por ejemplo continuarlo con control automático de la ganancia, o de la tasa de muestreo.

3.4 Conclusiones del capítulo

Se realizan pruebas para verificar el funcionamiento de ADS1298 a las frecuencias de interés para electrocardiograma, conectando un generador de funciones a la entrada del chip. Al analizar los resultados, se puede concluir que el chip responde correctamente a las frecuencias probadas.

La interfaz gráfica del proyecto se desarrolla en Python, en esta plataforma se procesan las muestras recibidas por el PC y se despliegan en una ventana que permite filtrar en tiempo real la señal.

La señal de ECG mostrada en pantalla es como se espera, se pueden apreciar el complejo PQRS de la señal de electrocardiograma, lo que permitiría que un especialista realice un diagnóstico correcto funcionamiento del sistema cardiaco de un paciente.

Discusión y conclusiones

El diseño de circuitos empleando un ASIC, en desmedro de un diseño con circuitos integrados de propósito general, tiene como principal beneficio disminuir el tiempo de desarrollo y el espacio utilizado. Como el fin de este proyecto no es generar un equipo con pretensión de venta, la disminución de espacio no es una utilidad directa.

Un beneficio importante del proyecto es la familiarización con este tipo de circuitos integrados. Emplear el ASIC en el laboratorio permite adquirir conocimientos sobre este tipo de tecnología pensado en un uso académico.

A lo largo del proyecto se estudiaron otros proyectos que emplean circuitos de propósito general para adquirir señales de ECG, EMG y EEG y convertirlas a señales digitales, realizados anteriormente en LABSEI. Luego del estudio de estos proyectos, se concluye que todos tienen en común un importante tiempo de dedicación para el diseño de la etapa de adquisición, amplificación y digitalización de las señales.

Hoy en día hay en el mercado una gran variedad de chips diseñados específicamente para la adquisición se señales bioeléctricas.

Dentro de los ASIC elegidos para comparar, se seleccionó el ADS1298 IPAG. Este ASIC es un circuito integrado que cumple con las características necesarias para adquirir los 3 biopotenciales en estudio. La decisión fue tomada a partir de comparaciones entre características técnicas, popularidad, recomendaciones y precios de cada uno de los ASICs comparados.

El ADS1298 se montó en un adaptador para poder emplearlo en un tablero de experimentación en el laboratorio.

Dado que para el LABSEI, la señal de mayor interés es la señal de electrocardiograma, y considerando que el proyecto tiene como una de sus finalidades, obtener aprendizaje en el ámbito académico sobre el empleo de ASIC para la captura de biopotenciales, se decide evaluar el proyecto construyendo un electrocardiógrafo basado en un ADS1298.

El estudio de las características de la señal de electrocardiograma, en conjunto con las recomendaciones del fabricante, permiten diseñar un diagrama circuital para emplear el

ADS1298 como electrocardiógrafo. La alimentación del dispositivo, que en un principio se hacía con una fuente digital, con la que se controlaba la corriente para no dañar el ASIC, que por cierto es muy delicado, en definitiva se hizo directamente desde el puerto USB del PC. La tensión análoga se fijó en 5 [v] y la digital en 3.3 [v].

Para poder programar el ASCI se emplea el microcontrolador PIC18F4550, y se desarrolla un código para programarlo empleando el software MPLAB.

Lo más complejo de la programación, fue lograr interpretar de manera correcta los datos. Debido a que el ADS1298 tiene una resolución de 24 bits, es que se necesitan tres lecturas sucesivas mediante el puerto SPI para obtener un byte a la vez, y de este modo tener un dato de lectura. Por medio del puerto USB también se enviaron las muestras divididas en tres partes, y en el pc, en Python, se creó una porción de código dedicada a unir los tres bytes para crear una sola muestra. Cada paquete de datos enviado desde el microcontrolador al PC contiene 60 bytes, es decir 20 muestras.

Debido a que se escogió la señal de electrocardiograma como señal principal para el proyecto, se evaluó el desempeño del ADS1298 a las frecuencias propias de un ECG, de este modo, se conectó un generador de funciones a la entrada del canal del ADS1298 habilitado para recibir la señal, se generaron sinusoides a frecuencias conocidas, y se hizo un análisis espectral de las sinusoides recibidas en el PC. Se hicieron pruebas entre 1 y 100 [Hz], en todas las pruebas las componentes en frecuencia de la señal recibida en el PC coincidieron con las de la sinusoide generada por el generador de funciones. El experimento descrito permite concluir que no es posible detectar alteraciones en las componentes en frecuencias de las señales que entran y salen del ASIC.

Para diseñar la interfaz gráfica, donde se despliega la señal del electrocardiograma, se usó como base una aplicación empleada en LABSEI en el curso de Instrumentación Electrónica, que se usa para recibir datos de 8 bit del puerto USB y desplegarlos en pantalla. La nueva versión diseñada recibe datos de 24 bit, los grafica en tiempo real y permite filtrar, también en tiempo real, solo presionando un botón. El filtro diseñado es un filtro FIR pasa bajos, con frecuencia de corte en 35 [Hz], emulando a un electrocardiograma ambulatorio, como los que se realizan en cualquier centro asistencial del país.

Debido a que se trabaja con 4000 [m/s], para asegurar que los buffer del sistema no se desbordaran, se implementó una etapa en el código en Python para graficar en pantalla una muestra de cada veinte, o en su defecto, si el botón de filtrado se presiona, la señal se filtra para obtener una muestra nueva, cada 20 que ingresan al filtro. De este modo se garantiza que el sistema se mantenga funcionando a la misma tasa de datos que ingresan.

Al realizar pruebas con pacientes, se apreció que la señal filtrada, y la señal sin filtrar, tenían una diferencia, como puede verse en la Figura 3-7, pero el aporte del ruido que afecta al ADS1298, cuando se realizan las pruebas en condiciones normales, con un paciente que cuida no moverse durante las pruebas, no es tan notorio, aunque de todos modos la señal filtrada es la que permitiría, a un experto, hacer un diagnóstico sobre alguna patología cardiaca.

Por último, en el proyecto, se agregó una etapa en el código en Python, para calcular la frecuencia cardiaca. Esta etapa se desarrolló creando un algoritmo que busca los peak de la señal de ECG, para contar la cantidad de muestras que hay entre uno y otro, y a partir de este dato, se calcula el tiempo que hay entre ambos peak, empleando la frecuencia de muestreo. Con este dato obtenido se proyecta la tasa de latidos por minuto. Debido al movimiento de los cables que conectan los electrodos que están en el paciente con el circuito, algún movimiento del paciente o a alguna perturbación externa, este cálculo podría no ser correcto.

Se concluye, que emplear la correlación de señales, sería más preciso, a la hora de calcular la frecuencia cardiaca, y de este modo, no se tendría la incertidumbre a la hora de mover los cables de los electrodos, haciendo más confiable aún el cálculo mostrado. Esto queda propuesto para trabajos futuros dónde se emplee el ADS1298.

Glosario

*.txt		Archivo informático con estructura series de líneas de texto.
Algoritmo		Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.
Aliasing		Efecto que causa que señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente.
Código		Líneas de texto con pasos que debe seguir la computadora para ejecutar dicho programa.
Espectro		Conjunto de ondas electromagnéticas, ordenadas según su frecuencia
FFT		Transformada rápida de Fourier o Fast Fourier Transform.
Filtro Pasa Bajo	FPB	Filtro que deja pasar aquellas frecuencias que están bajo de una determinada frecuencia.
Frecuencia	f	Número de vibraciones, ondas o ciclos de un fenómeno periódico realizados en una unidad de tiempo determinada.
Frecuencia de muestreo	fs	Número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, durante el proceso necesario para convertirla de analógica en digital.
Ganancia	G	Amplificar una señal.
Hertz	Hz	Unidad de frecuencia del Sistema Internacional, de símbolo Hz, que equivale a la frecuencia de un fenómeno periódico cuyo período es 1 segundo.

Bibliografía

[1] E. M. Spinelli y M. A. Mayosky, «Amplificadores de Instrumentación en Aplicaciones Biomédicas.,» La Plata, 2007.

[2] A. Martínez García y J. Roca González, «ESTUDIO TÉCNICO DE CONFIGURACIONES OPERACIONALES PARA EL DISEÑO DE ETAPAS DE AMPLIFICACIÓN PARA EL REGISTRO DE BIOPOTENCIALES,» Cartagena, 2013.

[3] S. I. Bruna Stevenson y J. Vignolo Barchiesi, «Desarrollo de una prótesis mioeléctrica controlada mediante un FPGA,» Valparaíso, 2017.

[4] S. Huerta y D. Zumaeta, «Visualización de un electrocardiograma en un PC.,» Valparaíso, 2017.

[5] D. Yunge Sepúlveda y J. Vignolo Barchiesi, «Dispositivo para detectar y clasificar ondas cerebrales,» Valparaiso, 2010.

[6] L. A. Certés, «Presente y futuro de la microelectrónica,» *Revista Ingeniería e Investigación,* nº 41, pp. 64-67, 1998.

[7] J. GUEVARA, L. LÓPEZ y M. EMMANUEL, «Circuitos Integrados configurables ASIC,» [En línea]. Available: http://icprgm-asic.blogspot.com/. [Último acceso: 15 agosto 2018].

[8] Aula Formativa, [En línea]. Available: https://blog.aulaformativa.com/razones-aprender-python/. [Último acceso: 15 Agosto 2018].

[9] Texas Instruments, «Data Sheet ADS1294/6/8,» [En línea]. Available: http://www.ti.com/lit/ds/symlink/ads1296r.pdf.

[10] Texas Instruments, «Analog Front-End Design for ECG Systems Using Delta-Sigma ADCs,» Application Report, 2010.

[11] TI E2E™ Community, «Texas Instruments,» [En línea]. Available: https://e2e.ti.com/support/data_converters/precision_data_converters/f/73/tags/ads1298.

[12] American National Standards Institute, *Diagnostic electrocardiographic devices,* Arlington, 2001.

[13] A. Malhotra y M. Ryschka, «A System for Multi-Modal Assessment of Cardiaovascular Parameters Design, Test and Measurements,» Lübeck, 2013.

[14] D. U. Silverthorn, Fisiología Humana un enfoque integrado, Madrid: Panamericana, 2008.

[15] Wikimedia Foundation, Inc., «Wikipedia The fFee Encyclopedia,» [En línea]. Available: https://en.wikipedia.org/wiki/10%E2%80%9320_system_(EEG). [Último acceso: 21 Septiembre 2017].

[16] Physiology Laboratory of McGill University, «The McGill Physiology Virtual Laboratory,» McGill University, 2005. [En línea]. Available: http://www.medicine.mcgill.ca/physio/vlab/default.htm. [Último acceso: 20 Septiembre 2017].

[17] F. N. Guerrero, M. Haberman y E. Spinelli, «Sistema multicanal para adquisición de biopotenciales,» *Revista Ingeniería Biomédica,* vol. 8, nº 15, pp. 18-26, 2014.

[18] J. G. Webster, Medical Instrumentation Application and Design, John Wiley & Sons, INC, 2010.

[19] Analog Devices, «Data Sheet AD7768,» [En línea]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD7768-7768-4.pdf.

[20] Analog Devices, «Data Sheet ADS1000,» [En línea]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADAS1000_1000-1_1000-2.pdf.

[21] Texas Instruments, «Data Sheet ADS1198,» [En línea]. Available: http://www.ti.com/lit/ds/symlink/ads1198.pdf.

[22] Texas Instruments, «Data Sheet ADS1299,» [En línea]. Available: http://www.ti.com/lit/ds/symlink/ads1299.pdf.

[23] J. Ankit , «Low cost instrumentation and interface for neuronal recordings,» Pennsylvania, 2012.

[24] Fundación Wikimedia, Inc., «Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface. [Último acceso: 20 Septirmbre 2017].

[25] Microchip, «PIC18F2455/2550/4455/4550 Data Sheet,» [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf.

[26] RS Componentes Electrónicos Limitada, «RS Componentes,» [En línea]. Available: http://cl.rsdelivers.com/product/microchip/pic18f4550-i-p/microchip-pic18f4550-i-p-8bit-pic-microcontroller-48mhz-32-kb-256-b-flash-40-pin-pdip/6230803.aspx. [Último acceso: 22 Octubre 2017].

[27] M. Hann y J. Duenas, «Analog Fundamentals of the ECG Signal Chain,» Texas Instruments.

[28] Texas Instruments, «ADS1298ECG-FE/ADS1198ECG-FE User's Guide,» 2010.

[29] Microchip, «(IDE), MPLAB X Integrated Development Environment,» [En línea]. Available: http://www.microchip.com/mplab/mplab-x-ide.

[30] CCS , «C COMPILER,» [En línea]. Available: http://www.ccsinfo.com/content.php?page=compilers.

[31] CCS, «MANUAL CCS C COMPILER,» [En línea]. Available: http://www.ccsinfo.com/downloads/ccs_c_manual.pdf.

[32] J. Vignolo Barchiesi, «SISTEMA DE ADQUISICION DE DATOS,» Instrumentación Electrónica - EIE.

A Listado de Códigos

A continuación se presentan los listados de códigos empleados en el proyecto

A.1 Código Microcontrolador

El código del microcontrolador está desarrollado en el software MPLAB.

Listado 0-1: Código PIC18F4550.

```
/*PROGRAMA Y RECIBE DATOS DESDE EL ADS1298 Y LOS ENVÍA AL PC VIA USB*/
#include <18F4550.h>
#fuses NOWDT,NOPROTECT,NOLVP,NODEBUG,HSPLL,PLL5,CPUDIV1
#fuses NOMCLR,VREGEN,USBDIV
#use delay(clock=48000000)


#include <pic18_usb.h>            // Driver USB del PIC18F4550
#include <Descriptor_easyHID.h>   // Descriptor HID de USB
#include <USB.c>                  // Funciones USB de C

// Declaración de constantes y variables


int8 datos1[60];          // Bytes a transmitir por paquetes
int8 contador = 0;        // contador de datos en el arreglo
int8 datos_no_usados[6];  // Datos de estado, no utilizados
int8 i;
int8 a = 1;               // Variable para escoger ganancias
int8 ganancia = 0x10;


#INT_EXT2

  void Int_RB2(){
      output_low(pin_A5);
      if (a<5){
          output_low(pin_D0);
          Spi_write(0x45); //Chn 1  gain 1(0x10), gain 2 (0x20) gain 3
         (ox30) gain 4 (ox40) gain 6 (0x00) gain 8 (0x50) gain 12 (0x60)
          delay_us(3);
          Spi_Write(0x01);
          delay_us(3);
          Spi_Write(ganancia);
          delay_us(3);
          ganancia = ganancia + 0x10;
          a = a+1;
         }
      else{
          ganancia = 0x00;
          Spi_write(0x45);
```

```
42            delay_us(3);
43            Spi_Write(0x01);
44            delay_us(3);
45            Spi_Write(ganancia);
46            delay_us(3);
47            ganancia = 0x10;
48            a = 1;
49         }
50   }
51
52
53
54
55   void Recoger_SPI() {
56      output_low(pin_A5); // CS bajo, comienza comunicación SPI
57      delay_us(1);
58      for (i = 0;i <= 5;i++ )
59      {datos_no_usados[i] = SPI_READ(0);
60
61      delay_us(3);}
62
63         datos1[contador] = datos_no_usados[3];
64         datos1[contador+1] = datos_no_usados[4];
65         datos1[contador+2] = datos_no_usados[5];
66
67
68           if (contador == 57)    // Si se completó el arreglo
69          {      // Transmite el arreglo via USB HID//
70               usb_put_packet(1, datos1, 60, USB_DTS_TOGGLE);
71               contador = 0;     // reinicia contador de datos en el arreglo
72
73          }
74           else
75            contador+=3; // Incrementa contador de bytes recibidos, 3 bytes
76                         equivalen a una muestra
77   }
78
79
80   void main() {
81
82       setup_timer_2(t2_div_by_1,3,1); // Timer 2 como clock del SPI a 2
83                                          MHz(4MHz/2)//
84
85       setup_adc_ports(NO_ANALOGS);; // Todos los pines como digitales
86
87       set_tris_D(0b11010100); // Configura puerta D
88
89       output_D(0); // Todas las salidas de la puerta Dpuestas en 0
90
91
92       output_high(pin_D6); // Enciende ADS1298
93
94       output_high(pin_A5); // CS alto, comunicación SPI en espera
95
96       setup_spi(spi_master | spi_l_to_h | spi_clk_t2); // SPI maestro,
97                                                           frec. 2MHz
98
99
100      delay_us(50);
101
102      output_low(pin_A5); // CS bajo, comienza comunicación SPI
103      delay_us(1);
104
105      //Configuración del ADS1298//
106
107      Spi_write(0x41);// Config1 - 4ksps (0x42), 2ksps (0x43) 1ksps (0x44)
108      delay_us(3);
109      Spi_Write(0x01);
110      delay_us(3);
111      Spi_Write(0x42);
112              delay_us(3);
```

```
113
114     Spi_write(0x42); // Config2 - default
115     delay_us(3);
116     Spi_Write(0x01);
117     delay_us(3);
118     Spi_Write(0x12);
119             delay_us(3);
120
121     Spi_write(0x43); // Config3 - VREFF 4V (f4)  vreff 2.5v(d4),
122                         RLD CONECTADO
123     delay_us(3);
124     Spi_Write(0x01);
125     delay_us(3);
126     Spi_Write(0xf4);
127             delay_us(3);
128
129     Spi_write(0x44); // LOFF - default
130     delay_us(3);
131     Spi_Write(0x01);
132     delay_us(3);
133     Spi_Write(0x02);
134             delay_us(3);
135
136     Spi_write(0x45); // Chn 1  Encendido
137     delay_us(3);
138     Spi_Write(0x01);
139     delay_us(3);
140     Spi_Write(0x40);
141             delay_us(3);
142
143     Spi_write(0x46); // Chn2 APAGADO
144     delay_us(3);
145     Spi_Write(0x01);
146     delay_us(3);
147     Spi_Write(0x81);
148             delay_us(3);
149
150     Spi_write(0x47); // Chn 3
151     delay_us(3);
152     Spi_Write(0x01);
153     delay_us(3);
154     Spi_Write(0x81);
155             delay_us(3);
156
157     Spi_write(0x48); // Chn 4
158     delay_us(3);
159     Spi_Write(0x01);
160     delay_us(3);
161     Spi_Write(0x81);
162             delay_us(3);
163
164     Spi_write(0x49); // Chn 5
165     delay_us(3);
166     Spi_Write(0x01);
167     delay_us(3);
168     Spi_Write(0x81);
169             delay_us(3);
170
171     Spi_write(0x4A); // Chn 6
172     delay_us(3);
173     Spi_Write(0x01);
174     delay_us(3);
175     Spi_Write(0x81);
176             delay_us(3);
177
178     Spi_write(0x4B); // Chn 7
179     delay_us(3);
180     Spi_Write(0x01);
181     delay_us(3);
182     Spi_Write(0x81);
183             delay_us(3);
```

```
184
185     Spi_write(0x4C); // Chn 8
186     delay_us(3);
187     Spi_Write(0x01);
188     delay_us(3);
189     Spi_Write(0x81);
190             delay_us(3);
191
192     Spi_write(0x4D); // RLD SENSE P - Referencia de electrodos
193     delay_us(3);
194     Spi_Write(0x01);
195     delay_us(3);
196     Spi_Write(0x01);
197             delay_us(3);
198
199     Spi_write(0x4D); // RLD SENSE N - Referencia de electrodos
200     delay_us(3);
201     Spi_Write(0x01);
202     delay_us(3);
203     Spi_Write(0x01);
204             delay_us(3);
205
206     Spi_write(0x51); // LOFF FLIP - Desactivado
207     delay_us(3);
208     Spi_Write(0x01);
209     delay_us(3);
210     Spi_Write(0x00);
211     delay_us(3);
212
213
214    output_high(pin_D7); // Pin STAR del ADS en alto, inicia conversión
215                            de datos
216
217    delay_us(600); // 1160 ciclos del clock (580 ms) - Data Sheet
218
219    usb_init_cs();               // Inicializa hardware USB
220
221    enable_interrupts(int_ext2); // Activar interrupcion externa
222    ext_int_edge(2,L_TO_H);      // Config. de int. por flanco de subida
223    enable_interrupts(GLOBAL);   // Todas las interrupciones desactivadas
224
225
226     while (TRUE)
227         {
228         if(input(pin_D3)==0)        // Cuando el DRDY este en low
229         {usb_task();                // Verifica conexión con el host
230           if (usb_enumerated()) {   // Si a ha sido enumerado por el host
231              output_high(PIN_D4);   // Enciende Led verde
232              Recoger_SPI();
233              delay_us(2);           // Espera estabilidad del pin DRDY
234            }
235         }
236     }
237 }
238
239
240
241
```

A.2 Código programa Python

El siguiente código corresponde al procesamiento digital y la interfaz gráfica desarrollada en Python.

Listado 0-2: Código ECG Python.

```
# -*- coding: cp1252 -*-
"""
La primera línea del programa corresponde a la codificación,
ésta puede cambiar dependiendo del ordenador
"""
# 1.- Recibe códigos de 24 bit del ADS1298 los filtra y los grafica
# 2.- Mide la frecuencia de muestreo del ADS1298

# Primera versión: Felipe Serey Mendoza (2013)
# Segunda versión: Adrián Gallardo Aros (2014)
# Tercera versión: Juan Vignolo Barchiesi (2015)
# Cuarta versión : Yvo Henríquez Guerra (2018)

#------------------------------------------------------------

import pywinusb.hid as hid                    # Módulo USB HID
import wx                                          # Módulo de interfaz
gráfica
import wx.lib.plot as plot                        # Módulo para dibujar
gráficos
import time                                   # Módulo para medir tiempo
import numpy as np                            # Modulo para calculos
import statistics                             # Modulo de estadística
from scipy.signal import firwin, fftconvolve # Modulo para el filtrado


#------------------------------------------------------------

class Ventana(wx.Frame):
   """
   Definición de la clase 'Ventana'
   """

#------------------------------------------------------------

   def __init__(self,parent,title):
      """
      Inicialización: se ejecuta automáticamente al instanciar un objeto
      """
      wx.Frame.__init__( self, parent, title = title,      # Ventana del
programa
                                    size = (1331, 479))
      self.NDT = 20                      # Número de muestras en la trama
USB
      self.graph_cont = 0              # Contador de eje X del gráfico
      self.cont = 0                    # Contador de datos recibidos
      self.contador = 1
      self.alive = False               # Estado de conexión USB
      self.Interfaz()                # Rutina de generación de interfaz
gráfica
      self.Centre()                    # Centra ventana en pantalla
      self.SetSizeHints(1331, 479, 1331, 479)       # Tamaño mínimo y
máximo de la ventana
      self.Show()                                    # Despliega la ventana
      self.Bind(wx.EVT_CLOSE, self.OnClosePanel)  # Definición de evento
al cerrar el programa
      self.guardar_datos = 0                    # Variable para guardar
datos
      self.media =                           # Inicio de Variables
      self.arreglo = []
      self.lista2=[]
      self.plotear = 0
      self.pos_plot = 1
      self.veinte_muestras = 0
      self.lista_3=[]
      self.conteo = 0
      self.filtrar = 0
      self.peak = 0
```

```
69        self.peak_delay = 0
70        self.latidos = 0
71        self.mean = 0
72
73
74        self.fs =4000                          # Frecuencia de muestreo del
75  ADS1298
76
77        self.filtro_FIR = firwin(1024, [0.0015,0.0175],
78                                 window='hamming',
79                                 pass_zero=False)      # Diseño del filtro FIR
80  pasabanda entre 2 y 35 Hz
81
82  #-------------------------------------------------------
83
84     def sample_handler(self,data):
85        """
86        Se ejecuta al recibir un paquete desde el dispositivo USB/HID
87        """
88
89        if self.alive:                                          # Si llegó
90  paquete USB
91           for indice in range(1, self.NDT+1):                 # Lee bytes
92  desde trama USB entrante
93              pos_x = indice + self.graph_cont                 # Posición x
94  del dato
95
96              """
97              Esta etapa agrupa los datos recibidos de 8 bit en muestras
98  de 24 bit con signo
99              """
100
101             if np.int8(data[self.contador])>=0:
102                datos = 65536*np.int8((data[self.contador])+
103 256*np.uint8((data[self.contador+1]))+
104 np.uint8((data[self.contador+2])))
105             else:
106                datos       =       65536*(np.int8(data[self.contador]))-
107 256*np.uint8((data[self.contador+1]))-np.uint8((data[self.contador+2]))
108
109             """
110             Esta etapa agrupa los datos recibidos de 8 bit en muestras
111 de 24 bit con signo
112             """
113
114             dato = -datos-self.media  # Resta el valor medio a cada dato
115             newlist = (dato)      # Coordenadas (x,y) de dato en gráfico
116             self.arreglo.append(-datos) # Agrega el dato al arreglo para
117 calcular valor medio
118             self.lista.append(newlist)  #  Agrega  punto  a  lista  que
119 entrará al filtro
120             if self.guardar_datos ==1 # Si se presiona el boton "Grabar"
121                text_file.write(str(dato) + '\n')            # Graba las
122 muestras a 4000 m/s
123             self.contador +=3                               # Incrementa
124 variable para leer siguientes 3 bytes
125
126             if self.plotear >=1: # Espera a tener un segundo de muestras
127                if self.peak_delay >= 2000:   # Define maximo de 120
128                                              latidos por segundo
129                   if dato >= 0.9*self.peak: # Criterio para buscar
130                                             maximos
131                      self.latidos = ((4000/float(self.peak_delay))*60)
132                                           # Calcula latidos por minuto
133                      self.latidos = int(round(self.latidos))# Redondea
134                                           latidos al entero más cercano
135                      self.lat.SetValue(str(self.latidos)  # Muestra
136                                                    latidos por segundo
137                      self.peak_delay = 0 # Reinicia conteo para calcular
138                                            latidos por minuto
139             self.peak_delay+=1
```

```
140                 if self.peak_delay >=4000: # Define minimo de 60 latidos
141                                              por segundo
142                     self.peak_delay = 0    # Reinicia conteo para calcular
143                                              latidos por minuto
144
145
146             if self.mean == 50 # Cada 50 muestras actualiza el valor
147                                  mediode la señal
148                 self.media = (np.mean(self.arreglo))
149                 self.mean = 0
150             self.mean+=1
151
152             if self.filtrar == 1: # Si se preciona boton filtrar
153                 if self.plotear >=1: # Si hay más de un segundo de datos
154                   if self.veinte_muestras == 20: # Cada 20 muestras
155                                                    entra al filtro
156                     self.lista = np.asarray(self.lista) # Convierte
157                                                           lista a arreglo
158                     self.lista = self.lista [20:4020] # Actualiza la
159                                                         lista que entra al filtro
160                     self.lista_3 = self.lista # Variable que se filtra
161                     self.lista = np.array(self.lista).tolist()
162             # Convierte arreglo en lista para agregar el siguiente dato
163                     self.filtrado = fftconvolve(self.filtro_FIR,
164                                                 self.lista_3) # Filtra la señal
165                     grafica=self.pos_plot,self.filtrado[2500]  # Guarda
166                                                               la muestra filtrada
167                     self.lista2.append(grafica)# Agrega la muestra a la
168                                                  lista que se graficará
169
170                     if self.pos_plot<= 600: # Verifica que la ventana
171                                               no esté llena
172                        if self.conteo ==7: # Cada 7 muestras se
173                                              actualiza la gráfica
174                           line = plot.PolyLine(self.lista2, legend='',
175                                     colour='green',width=3)
176                                    # Características del gráfico
177                           gc = plot.PlotGraphics([line])
178                           self.graph.Draw(gc, xAxis=(1,600),)
179                           self.conteo = 0 # Reinicia contador
180                                             de 7 muestras
181                        self.conteo+=1 # Incrementa contador de 7
182                                         muestras
183                     else:
184                        self.lista2 = self.lista2[1:601] # Si se llenó
185                       la ventana comienza el desplazamiento del gráfico
186                        if self.conteo ==7:
187                           line = plot.PolyLine(self.lista2, legend='',
188                                                colour='green', width=3)
189                                    # Características del gráfico
190                           gc = plot.PlotGraphics([line])
191                           self.graph.Draw(gc,xAxis=(self.pos_plot-600,
192                                           self.pos_plot),) # Los ejes
193                                            se ajustan al desplazamiento
194
195
196
197                           self.conteo =0 # Reinicia contador de 7
198                                            muestras
199                        self.conteo+=1 # Incrementa contador de 7
200                                         muestras
201                     self.pos_plot +=1 # Actualiza posición de la
202                                         siguiente muestra
203                     self.veinte_muestras =0 # Reinicia contador de
204                                               ingreso al filtro (20 muestras)
205
206                 self.veinte_muestras +=1 # Incrementa contador de
207                                            muestras para entrar al filtro
208
209             else:
210                 if self.plotear >=1:
```

```
211                 if self.veinte_muestras == 20: # Cada 20 muestras
212                                                  grafica una
213                     self.lista = np.asarray(self.lista) # Convierte
214                                                        lista a arreglo
215                     self.lista = self.lista [19:4019] # Actualiza la
216                                                  lista que entra al gráfico
217                     self.lista_3 = self.lista # Variable que se
218                                                 graficará
219                     self.lista = np.array(self.lista).tolist()
220             # Convierte arreglo en lista para agregar el siguiente dato
221                     grafica=self.pos_plot,self.lista[1] # Guarda la
222                                                       muestra a graficar
223                     self.lista2.append(grafica)   # Agrega la muestra a
224                                                  la lista que se graficará
225
226                     if self.pos_plot<= 600: # Verifica que la ventana
227                                               no esté llena
228                        if self.conteo ==7: # Cada 7 muestras se
229                                               actualiza la gráfica
230                           line = plot.PolyLine(self.lista2, legend='',
231                                               colour='green', width=3)
232                                            # Características del gráfico
233                           gc = plot.PlotGraphics([line])
234                           self.graph.Draw(gc, xAxis=(1,600))
235                           self.conteo = 0 # Reinicia contador de 7
236                                               muestras
237                        self.conteo+=1 # Incrementa contador de 7
238                                            muestras
239                     else:
240                        self.lista2 = self.lista2[1:601] # Si se llenó
241                       la ventana comienza el desplazamiento del gráfico
242                        if self.conteo ==7:
243                           line = plot.PolyLine(self.lista2, legend='',
244                                               colour='green', width=3)
245                                            # Características del gráfico
246                           gc = plot.PlotGraphics([line])
247                           self.graph.Draw(gc,xAxis=(self.pos_plot-600,
248                                            self.pos_plot)) # Los ejes se
249                                               ajustan al desplazamiento
250                           self.conteo =0 # Reinicia contador de 7
251                                            muestras
252                        self.conteo+=1 # Incrementa contador de 7
253                                            muestras
254                     self.pos_plot +=1 # Actualiza posición de la
255                                            siguiente muestra
256                     self.veinte_muestras =0 # Reinicia contador de
257                                            ingreso al gráfico (20 muestras)
258
259                 self.veinte_muestras +=1 # Incrementa contador de
260                                            muestras para entrar al gráfico
261         self.cont += self.NDT # Incrementa contador de datos recibidos
262         self.graph_cont += self.NDT # Incrementa contador de 1 segundo
263                                         de muestras
264         self.contador = 1
265         self.cntBtn.SetBackgroundColour('Green') # Indica que la
266                                                  conección fue correcta
267         if self.graph_cont >3999: # Verifica si se recibió un segundo
268                                         (4000 muestras) de la señal
269            self.plotear+= 1 # Comienza el filtrado
270            self.peak = max(self.lista) # Busca el maximo de la lista
271
272            self.arreglo = [] # Reinicia el arreglo
273            self.graph_cont = 0 # Reinicia el contador de 1 segundo de
274                                      muestras
275            self.contador = 1 # Reinicia variable para leer 3 bytes
276            self.vel = self.cont / (time.clock() - self.t0) # Calcula la
277                                                  frecuencia de muestreo
278            self.valor = int(round(self.vel)) # Redondea frec. De
279                                            muestreo al entero más cercano
280            self.frecMstr.SetValue(str(self.valor))# Despliega
281                                                  frecuencia de muestreo
```

```
282
283 #-------------------------------------------------------
284
285     def OnConectar(self,evt):
286         """
287         Se ejecuta al presionar botón "Conectar" / "Desconectar"
288         """
289         filtro = hid.HidDeviceFilter(vendor_id=0x0480,
290                                      product_id=0x0300) # Filtro de
291                                                         dispositivo HID deseado
292
293         hid_devices = filtro.get_devices() # Aplica filtro a dispositivos
294                                            USB conectados
295         global text_file    # Archivo donde se grabarán las muestras
296         if not self.alive:    # Si no está conectado...
297             if hid_devices: # Si el dispositivo HID deseado es
298                             encontrado...
299                 self.device = hid_devices[0] # Define el dispositivo
300                                              conectado
301                 self.device.open()  # Abre dispositivo conectado
302                 self.device.set_raw_data_handler(self.sample_handler)# Envía
303                                                 datos recibidos a sample_handler
304                 self.alive = True  # Indicador de conexión establecida
305                 self.cntBtn.SetLabel('Desconectar') # Cambia etiqueta de
306                                                       botón de conexión
307                 self.Save.SetBackgroundColour('Red') # Cambia color del
308                                               boton Grabar (no se guardan muestras)
309                 self.t0 = time.clock() # Tiempo en seg. desde la llamada
310                                        anterior
311                 self.cont = 0   # Inicializa contador de datos recibidos
312                 self.contador = 1  # Reinicia variable para leer 3 bytes
313                 self.graph_cont = 0 # Inicializa contador de eje X del
314                                     gráfico
315                 self.lista = []  # Reinicia listas y variables del gráfico
316                 self.lista2 = []
317                 self.lista_3 =[]
318                 self.plotear = 0
319                 self.veinte_muestras =0
320                 self.pos_plot = 1
321                 text_file = open("dat.txt", "w") # Abre el archivo para
322                                                    grabar datos
323
324
325         else:                           # Si está conectado...
326             self.device.close()         # Cierra conexión con dispositivo USB
327             self.alive = False          # Indicador de conexión terminada
328             text_file.close()           # Cierra el archivo de datos
329             self.cntBtn.SetLabel('Conectar') # Cambia rótulo de botón de
330                                                conexión
331             self.cntBtn.SetBackgroundColour('yellow') # Si no esta
332                                                 conectado el boton es amarillo
333
334 #-------------------------------------------------------
335
336     def Grabar(self,evt):
337         """
338         Se ejecuta al presionar el botón "Grabar"
339         """
340         self.guardar_datos=1 # Permite guardar muestras datos a 4000 m/s
341         self.Save.SetBackgroundColour('Green') # Boton de color verde
342
343 #-------------------------------------------------------
344
345     def Filtrar(self,evt):
346         """
347         Se ejecuta al presionar el botón "FILTRAR"
348         """
349         if self.filtrar == 0:
350             self.filtrar = 1        # Activa el filtrado y cambia el color
351             self.Filtro.SetBackgroundColour('Green')
352
```

```
353         else:
354            self.filtrar = 0  # Si el filtro está activado, lo desactiva
355            self.Filtro.SetBackgroundColour('red')
356 
357 #-----------------------------------------------------
358 
359    def OnClosePanel(self,evt):
360       """
361       Se ejecuta al cerrar la ventana del programa
362       """
363       if self.alive:      # Si está recibiendo paquetes USB
364          self.device.close() # Desconecta dispositivo antes de cerrar el
365                              programa
366          text_file.close()     # Cierra el archivo de datos
367       self.Destroy()           # Cierra la ventana
368       evt.Skip()               # Termina el progama
369 
370 #-----------------------------------------------------
371 
372    def Interfaz(self):
373       """
374       Carga elementos de la interfaz gráfica
375       """
376       self.panel = wx.Panel(self, wx.ID_ANY)  # Muestra mismo panel en
377                                                 todas las plataformas
378       self.panel.SetBackgroundColour('Black')  # Color de la ventana
379 
380       self.cntBtn = wx.Button(self.panel, wx.ID_ANY,  # Botón "Conectar"
381                               'Conectar', pos = (200,18),
382                               size = (100,60))
383       self.cntBtn.SetBackgroundColour('yellow') # Color del boton
384                                                   "Conectar"
385 
386       self.Save = wx.Button(self.panel, wx.ID_ANY,    # Botón "Grabar"
387                               'Grabar', pos = (950,18),
388                               size = (100,60))
389       self.Save.SetBackgroundColour('Red')   # Color del botón "Grabar"
390 
391       self.Filtro = wx.Button(self.panel, wx.ID_ANY,  # Botón "Conectar"
392                               'Filtro', pos = (560,18),
393                               size = (100,60))
394       self.Filtro.SetBackgroundColour('red') # Color del botón "Filtro"
395 
396       self.frecMstr = wx.TextCtrl(self.panel, wx.ID_ANY, '',
397                                size = (36,20), pos = (480,410),
398                              style = wx.TE_READONLY) # Recuadro para
399                                                        desplegar muestras/seg
400 
401 
402       text1 = wx.StaticText(self.panel, -1, '[m/s]', (530,410))
403                                           # Recuadro muestras por segundo
404       font = wx.Font(10, wx.DECORATIVE, wx.ITALIC, wx.BOLD)
405       text1.SetFont(font)
406       text1.SetForegroundColour((0,255,0))
407 
408       self.lat = wx.TextCtrl(self.panel, wx.ID_ANY, '',  # Recuadro para
409                                                            desplegar latidos/seg
410                              size = (36,20), pos = (690,410),
411                              style = wx.TE_READONLY)
412 
413       text2 = wx.StaticText(self.panel, -1, '[lat/s]', (740,410))
414                                             # Recuadro latidos por segundo
415       font = wx.Font(10, wx.DECORATIVE, wx.ITALIC, wx.BOLD)
416       text2.SetFont(font)
417       text2.SetForegroundColour((0,255,0))
418 
419       text = wx.StaticText(self.panel, -1,    # Etiqueta LABSEI
420                            'LABSEI - PUCV', (1190, 410))
421       font = wx.Font(8, wx.DECORATIVE, wx.ITALIC, wx.BOLD)
422       text.SetFont(font)
423       text.SetForegroundColour((0,255,0))
```

```
424
425         self.graph = plot.PlotCanvas(self.panel, pos = (10,80)) # Recuadro
426                                                                    gráfico
427         self.graph.SetBackgroundColour('black')       # Color de fondo
428         self.graph.SetInitialSize(size = (1290,319))  # Tamaño inicial del
429                                                         gráfico
430
431         self.Bind (wx.EVT_BUTTON, self.OnConectar, self.cntBtn)  # Define
432                                  interrupción al presionar botón 'Conectar'
433         self.Bind (wx.EVT_BUTTON, self.Grabar, self.Save)        # Define
434                                   interrupción al presionar botón 'Grabar'
435         self.Bind (wx.EVT_BUTTON, self.Filtrar, self.Filtro)     # Define
436                                   interrupción al presionar botón 'Filtro'
437
438
439 #-------------------------------------------------------
440
441 app = wx.App()                    # Arranca el frame del módulo wxpython
442 Ventana(None, title = 'ELECTROCARDIÓGRAFO')  # Título de la ventana del
443                                                programa
444 app.MainLoop()                                # Loop principal
445
```

B Diagrama circuital

A continuación se presentan los diagramas circuitales definitivos.