Neiva, 2 de noviembre de 2019.

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Juan Camilo Caicedo España, con C.C. No.1´075.292.562, Autor(es) de la tesis y/o trabajo de grado o titulado DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO PARA LA ADQUISICIÓN, ACONDICIONAMIENTO, TRANSMISIÓN INALÁMBRICA Y ALMACENAMIENTO DE SEÑALES BIOELÉCTRICAS ECG OPTIMIZADO PARA SU USO PORTABLE presentado y aprobado en el año 2019 como requisito para optar al título de Ingeniero Electronico.

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

Juan Camilo Caicedo España

Firma: Juan Camilo Caicedo E.

TÍTULO COMPLETO DEL TRABAJO: DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO PARA LA ADQUISICIÓN, ACONDICIONAMIENTO, TRANSMISIÓN INALÁMBRICA Y ALMACENAMIENTO DE SEÑALES BIOELÉCTRICAS ECG OPTIMIZADO PARA SU USO PORTABLE

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
CAICEDO ESPAÑA	JUAN CAMILO

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
SALGADO PATRON	JOSE DE JESUS

PARA OPTAR AL TÍTULO DE: INGENIERO ELECTRONICO

FACULTAD DE: **INGENIERIA**

PROGRAMA O POSGRADO: INGENIERIA ELECTRONICA

CIUDAD: NEIVA **AÑO DE PRESENTACIÓN: 2019** **NÚMERO DE PÁGINAS: 94**

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas_X_ Fotografías___ Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos_X_ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento: NINGUNO

MATERIAL ANEXO: NINGUNO

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*: NINGUNO

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	**Inglés**
1. ADS1298	**ADS1298**
2. Derivaciones unipolares	**Unipolar Leads**
3. Derivaciones bipolares	**Bipolar Leads**
4. Protocolo TCP/IP	**TCP/IP Protocol**
5. Redis	**Redis**

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El dispositivo para la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioeléctricas ECG diseñado, dispone del circuito integrado (IC) de montaje superficial (SMD), ADS1298, el cual permite el correcto sensado de la señales producidas por el corazón, las cuales son ingresadas a través de sus entradas diferenciales, logrando una resolución por muestra de 24 bits, gracias a la funciones provistas por los muestreadores Sigma-Delta integrados en el IC mencionado. Este dispositivo tiene la capacidad de sensar las señales deseadas por el usuario a una frecuencia de muestreo que varía desde 500 muestras por segundo (SPS), hasta 32.000 muestras por segundo (KSPS), siendo estas manipuladas por una interfaz de comunicación SPI. Además, cuenta con la placa de desarrollo, Raspberry Pi Zero W, permitiendo una permanente comunicación entre el usuario y el dispositivo diseñado, lo cual es logrando por medio del envió de órdenes, por protocolo SSH, y recepción de muestras por medio de protocolo TCP/IP con ayuda de la interfaz WIFI integrada en la placa de desarrollo. Para el almacenamiento de las muestras, se cuenta con el gestor de bases de datos, Redis, que permite acceso a la información por medio de su característica de guardado en memoria. Además, el proyecto presentado, cuenta con una interfaz gráfica, diseñada sobre Python 3.7, que permite al usuario hacer una correcta visualización de un examen ECG estándar de 12 derivaciones en tiempo real, además de contar con la función de guardado de datos del paciente.

ABSTRACT: (Máximo 250 palabras)

Device for the acquisition, conditioning, wireless transmission, and storage of bioelectrical signals designed ECG, has integrated circuit (IC) surface mount (SMD), ADS1298, which allows the correct sensing of signals produced by the heart, which are entered through its differential inputs, achieving a resolution by sample of 24-bit, thanks to the functions provided by Sigma-Delta converters mentioned IC integrated. This device is capable of sensing signals desired by the user at a sampling rate that varies from 500 samples per second (SPS), up to 32000 samples per second (KSPS), being manipulated by an SPI communication interface. In addition, development, Raspberry Pi Zero W, plate features allowing a permanent communication between the user and the designed device, which is achieving by means of the orders, sent by the SSH protocol, and reception of samples through TCP/IP protocol with the help of the WIFI interface integrated in the Development Board. For the storage of samples, there are the non-relational database manager, Redis, allowing quick and easy access to information through its characteristic of saved in memory. Also, the project has a graphical interface, designed on the language Python 3.7, which allows the user to make a correct display of ECG examination standard 12 leads in real-time, as well as having the function of saving data of the patient. The interface has a panel of notifications and the function of digital filtering, for a thorough analysis of the collected signals.

APROBACION DE LA TESIS

Nombre Presidente Jurado: José De Jesús Salgado Patrón

Firma: 

Nombre Jurado: Jesús David Quintero Polanco

Firma: 

Nombre Jurado: Diego Fernando Sendoya

Firma: 

DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO PARA LA ADQUISICIÓN, ACONDICIONAMIENTO, TRANSMISIÓN INALÁMBRICA Y ALMACENAMIENTO DE SEÑALES BIOELÉCTRICAS ECG OPTIMIZADO PARA SU USO PORTABLE

JUAN CAMILO CAICEDO ESPAÑA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2019

DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO PARA LA ADQUISICIÓN, ACONDICIONAMIENTO, TRANSMISIÓN INALÁMBRICA Y ALMACENAMIENTO DE SEÑALES BIOELÉCTRICAS ECG OPTIMIZADO PARA SU USO PORTABLE

JUAN CAMILO CAICEDO ESPAÑA

Trabajo De Grado Para Optar Al Título De Ingeniero Electrónico

Director
Ing. José De Jesús Salgado Patrón, MgC

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2019

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, Mayo 27 de 2019

DEDICATORIA

Este trabajo está dedicado a todas las personas que me han permitido con su grano de arena, y sabias palabras, ser el hombre que soy ahora, a mis padres, hermanos, mis amigos, a la mujer que tengo a mi lado hoy, y a todos los que me faltan, esto es para ustedes.

Juan Camilo Caicedo España

AGRADECIMIENTOS

Se merecen agradecimientos especiales, el ingeniero José De Jesús Salgado Patrón, y la Ingeniera Ángela Patricia Guerrero Castillo, sin ustedes, este trabajo no podría ser lo que es, a ustedes muchas gracias.

Juan Camilo Caicedo España

TABLA DE CONTENIDOS

Pág

INTRODUCCIÓN **15**

1. PLANTEAMIENTO DEL PROBLEMA **16**

2. JUSTIFICACION **17**

3. OBJETIVOS **18**

3.1. OBJETIVO GENERAL 18

3.2. OBJETIVOS ESPECÍFICOS 18

4. MARCO TEÓRICO **19**

4.1. SEÑALES BIOELÉCTRICAS 21

4.1.1. Electrocardiograma (ECG) 22

4.1.2. Derivaciones de un ECG 22

4.2. MODELO DE ELECTROCARDIÓGRAFO 24

4.2.1. Etapas de un ECG convencional 24

4.3. ESTUDIO DE CIRCUITOS INTEGRADOS DE TIPO SMD 25

4.3.1. Definición 25

4.3.2. Circuitos Candidatos 26

4.3.3. Integrado ADS1298 27

4.4. ELECCIÓN DISPOSITIVO DE DESARROLLO 31

4.4.1. Rsapberry Pi Zero W 32

5. DESARROLLO PRÁCTICO **33**

5.1. DISEÑO DE SISTEMA DE ADQUISICIÓN 33

5.1.1. Etapa 1: Alimentación ADS1298 33

5.1.2. Etapa 2: Filtrado 35
5.1.3. Etapa 3: Funcionamiento 36
5.2. VERIFICACIÓN DE FUNCIONAMIENTO 40
5.2.1. Prueba de lectura de registro de identificación 40
5.2.2. Prueba de recolección y transmisión de señal cuadrada de prueba generada por el mismo ADS1298 44
5.3. DISEÑO DE INTERFAZ GRÁFICA PARA PRESENTACIÓN DE DATOS 59
5.3.1. Panel 1: Datos del paciente 59
5.3.2. Panel 2: Caja de notificaciones 60
5.3.3. Panel 3: Botones 60
5.3.4. Panel 4: Pestañas de graficación 64
5.3.5. Panel 5: Carga de datos y revision de conexiones 65

6. RESULTADOS **67**

6.1. RECOLECCION DE SEÑAL ECG GENERADA POR SIMULADOR . . . 67
6.1.1. Montaje de prueba con simulador 67
6.1.2. Señal Recolectada 69
6.2. RECOLECCIÓN DE SEÑALES DE UN SUJETO DE PRUEBAS REAL . 70

7. CONCLUSIONES **74**

8. RECOMENDACIONES **76**

9. TRABAJOS FUTUROS **77**

BIBLIOGRAFÍA **78**

ANEXOS **80**

LISTA DE FIGURAS

Pág

Figura 1. Valores de Señales bioeléctricas 21

Figura 2. Ondas ECG 22

Figura 3. Triángulo de Einthoven 23

Figura 4. Derivaciones Unipolares y Precordiales 23

Figura 5. Diagrama de bloques para sensado de señal ECG 24

Figura 6. Tabla comparativa de integrados candidatos 27

Figura 7. Esquematico ADS1298 28

Figura 8. Multiplexor de Entrada ADS1298 29

Figura 9. Amplificador de Ganancia Programable (PGA) 29

Figura 10. Terminal Central De Wilson (WCT) 30

Figura 11. Circuito de Pierna Derecha (RLD) 31

Figura 12. Características Raspberry Pi Zero W 32

Figura 13. Condiciones Recomendadas de Operacion ADS1298 33

Figura 14. Conexion Bipolar de Alimentación 34

Figura 15. Fuentes de Alimentación AVDD Y AVSS 35

Figura 16. Filtros de Entrada 36

Figura 17. Calculos Derivaciones Precordiales 36

Figura 18. Derivaciones Aumentadas en Funcion de las derivaciones unipolares D1, D2, D3 37

Figura 19. Diseño Final 38

Figura 20. Diseño Final: Vista Superior 38

Figura 21. Diseño Final: Vista Superior Raspberry Pi Zero W 39

Figura 22. Diseño Final: Vista Lateral Derecha 39

Figura 23. Diseño Final: Vista Lateral Izquierda 39

Figura 24. Protocolo SPI: Líneas de comunicación 41
Figura 25. ADS1298: Comandos Interfaz SPI 42
Figura 26. ADS1298: Mapa de Registros 43
Figura 27. ADS1298: Lectura del Registro de Identificación 44
Figura 28. ADS1298: Registro de Identificación 44
Figura 29. ADS1298: Registro de Configuración 1 (CONFIG1) 45
Figura 30. ADS1298: Registro de Configuración 2 (CONFIG2) 46
Figura 31. ADS1298: Registro de Configuración 3 (CONFIG3) 46
Figura 32. ADS1298: Registro de Configuración de Canales 1 a 8 (CHnSET) . . 47
Figura 33. ADS1298: Escritura de registros para prueba de señal cuadrada . . . 48
Figura 34. ADS1298: Modo de lectura Single-Shot 49
Figura 35. Distribución de Pines Raspberry Pi 49
Figura 36. Contenido de la Palabra estado 50
Figura 37. Tamaño Total de Trama Generada 50
Figura 38. Tramas obtenidas con el algoritmo de captura 51
Figura 39. Diagrama de flujo Algoritmo de envío 53
Figura 40. Forma de trama serializada 53
Figura 41. Llegada de tramas enviadas 54
Figura 42. Error de socket roto 54
Figura 43. Tramas Almacenadas en DB Redis 55
Figura 44. Algoritmo de prevención y organización de tramas 56
Figura 45. Pasos graficación de tramas 56
Figura 46. ADS1298: Formato de palabra 57
Figura 47. Algoritmo de graficación de tramas 58
Figura 48. Gráficas de señal cuadrada canal 1 58
Figura 49. Interfaz Gráfica Diseñada 59

Figura 50. ADS1298: Registro de LOFF 60
Figura 51. ADS1298: Tabla de Ruido intrínseco 61
Figura 52. ADS1298: Registro RLD SENSEP 61
Figura 53. ADS1298: Registro RLD SENSEN 62
Figura 54. ADS1298: Registro LOFF SENSEP 62
Figura 55. ADS1298: Registro LOFF SENSEN 62
Figura 56. ADS1298: Registro de configuración 4 (CONF4) 63
Figura 57. ADS1298: Registro WCT 1 63
Figura 58. ADS1298: Registro WCT 2 63
Figura 59. Interfaz Grafica: Ventana de busqueda Archivo .dat 66
Figura 60. Caracteristicas: Dynatech Nevada INC Patient Simulator 67
Figura 61. Dynatech Nevada INC Patient Simulator 68
Figura 62. Características Cable Schiller 2.400 071 68
Figura 63. Conexiones entre cable y simulador 69
Figura 64. Señal Recolectada, simulador 69
Figura 65. :Señal Recolectada Filtrada, simulador 70
Figura 66. Electrocardiografo ECG100G 71
Figura 67. Montaje Sujeto real 71
Figura 68. Resultados obtenidos de ECG100G 72
Figura 69. Resultados obtenidos dispositivo diseñado sin filtrar 72
Figura 70. Resultados obtenidos dispositivo diseñado filtrados 73

LISTA DE ANEXOS

Pág

Anexo A. Códigos de Control ADS1298 79
Anexo B. Código de Interfaz Gráfica 85

GLOSARIO

ADS1298: Integrado de tipo SMD especializado en la captura de señales Bioelécricas.
Bluetooth: Es una tecnología que permite la interconexión inalámbrica de dispositivos electrónicos.
ECG: Es la representación de gráfica de las señales bioeléctricas generadas por el corazón.
GHz: (Gigahertz)Es una unidad de frecuencia.
IP: (Internet Protocol)Es un número que identifica, de manera lógica y jerárquica, a una Interfaz en red.
IPv4: (Internet Protocol version 4)El Protocolo de Internet versión 4.
MHz: (Megahertz)Es una unidad de medida de la frecuencia.
Python: Lenguaje de programación.
Raspberry Pi Zero W: placa de desarrollo de tamaño y costo reducidos que permite la integración de múltiples funciones como lo es la transmisión de datos vía WIFI o Bluetooth, entre otras, controlado por un sistema operativo propio.
Redis: Gestor de base de datos no relacional.
RLD: (Driven right leg) Circuito manejador de pierna derecha.
SPS: (Samples per second) Unidad de medida para toma de muestras.
TCP: (Transmission Control Protocol)Protocolo de control de transmisión.
WCT: (Wilson Central Terminal) Señal de la terminal central de Wilson.
Wi-Fi: Es una tecnología que permite la interconexión inalámbrica de dispositivos electrónicos.

RESUMEN

El dispositivo para la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioeléctricas ECG diseñado, dispone del circuito integrado (IC) de montaje superficial (SMD), ADS1298, el cual permite el correcto sensado de la señales producidas por el corazón, las cuales son ingresadas a través de sus entradas diferenciales, logrando una resolución por muestra de 24 bits, gracias a la funciones provistas por los muestreadores Sigma-Delta integrados en el IC mencionado. Este dispositivo tiene la capacidad de sensar las señales deseadas por el usuario a una frecuencia de muestreo que varía desde 500 muestras por segundo (SPS), hasta 32.000 muestras por segundo (KSPS), siendo estas manipuladas por una interfaz de comunicación SPI. Además, cuenta con la placa de desarrollo, Raspberry Pi Zero W, permitiendo una permanente comunicación entre el usuario y el dispositivo diseñado, lo cual es logrando por medio del envió de órdenes, por protocolo SSH, y recepción de muestras por medio de protocolo TCP/IP con ayuda de la interfaz WIFI integrada en la placa de desarrollo.

Para el almacenamiento de las muestras, se cuenta con el gestor de bases de datos no relacional, Redis, que permite un fácil y rápido acceso a la información por medio de su característica de guardado en memoria. Además, el proyecto presentado, cuenta con una interfaz gráfica, diseñada sobre el lenguaje Python 3.7, que permite al usuario hacer una correcta visualización de un examen ECG estándar de 12 derivaciones en tiempo real, además de contar con la función de guardado de datos del paciente. La interfaz tiene un panel de notificaciones y la función de filtrado digital, para un análisis minucioso de las señales recolectadas.

PALABRAS CLAVES

ADS1298; derivaciones unipolares; derivaciones bipolares; protocolo TCP/IP; Redis.

Abstract

Device for the acquisition, conditioning, wireless transmission, and storage of bioelectrical signals designed ECG, has integrated circuit (IC) surface mount (SMD), ADS1298, which allows the correct sensing of signals produced by the heart, which are entered through its differential inputs, achieving a resolution by sample of 24-bit, thanks to the functions provided by Sigma-Delta converters mentioned IC integrated. This device is capable of sensing signals desired by the user at a sampling rate that varies from 500 samples per second (SPS), up to 32000 samples per second (KSPS), being manipulated by an SPI communication interface. In addition, development, Raspberry Pi Zero W, plate features allowing a permanent communication between the user and the designed device, which is achieving by means of the orders, sent by the SSH protocol, and reception of samples through TCP/IP protocol with the help of the WIFI interface integrated in the Development Board.

For the storage of samples, there are the non-relational database manager, Redis, allowing quick and easy access to information through its characteristic of saved in memory. Also, the project has a graphical interface, designed on the language Python 3.7, which allows the user to make a correct display of ECG examination standard 12 leads in real-time, as well as having the function of saving data of the patient. The interface has a panel of notifications and the function of digital filtering, for a thorough analysis of the collected signals.

Keywords

ADS1298; unipolar leads, bipolar leads, TCP/IP protocol; Redis.

INTRODUCCIÓN

Las señales bioeléctricas han sido desde los años 50 un tema de absoluta importancia en el campo de la medicina, esto respondiendo a la necesidad de los especialistas de obtener una rápida y fiable respuesta a un problema presentado en un paciente, reduciendo así un amplio espectro de posibles tratamientos para su pronta mejoría. "Estas señales son el resultado de la conductividad eléctrica que se produce en el cuerpo debido al movimiento de iones; La adquisición de estas señales implica transformar estas corrientes de iones en corrientes eléctricas susceptibles de ser manejadas por la instrumentación electrónica que se tiene hoy en día." Presedo, *Adquisición de señales biológicas*

Entre las múltiples señales que se encuentran en el cuerpo humano sobresalen las señales electromiograma (EMG), electroencefalograma (EEG), electrooculograma (EOG) y electrocardiograma (ECG). Esta última cobra mayor importancia debido a que el resultado que se obtiene de la misma tiene hoy un gran potencial para la detección de enfermedades y afecciones que afectan uno de los órganos más importantes del ser humano, el corazón.

Cotidianamente en el campo de la medicina un electrocardiograma es realizado por un médico especialista, el cual imprime y lee para apoyarse junto con otros exámenes para dar un diagnóstico preciso y contundente sobre el estado actual del paciente. Sin embargo, este proceso se torna difícil y tedioso en zonas donde los centros de salud son de bajo nivel, debido a factores cruciales como lo son el alto costo del equipo junto con su excesivo tamaño, y alto consumo de energía hacen que la realización de un examen simple, se convierta en una dura tarea a cumplir. Un electrocardiograma que pueda hacerse de manera rápida, segura, confiable y de bajo costo sería un avance enorme para estas comunidades que requieren una atención segura y confiable.

1. PLANTEAMIENTO DEL PROBLEMA

Con el descubrimiento de las señales bioelectricas que genera el cuerpo humano, siendo una de ellas la señal ECG, se ha tratado de extraer la mayor información de ésta para el correcto diagnóstico de las enfermedades de los pacientes. Esto se ha logrado hoy en día por medio de instrumentación electrónica que permite adquirir estas señales para su posterior análisis. Esto fue susceptible de mejorarse con la llegada del transistor, pues permitió la disminución de tamaño de los dispositivos electrónicos diseñados para tal fin. Estos equipos son los que están presentes en los puestos de salud como hospitales, clínicas, e instituciones prestadoras de salud. Estos equipos son costosos de producir, pues deben tener una fiabilidad técnica de alto desempeño, presentan un considerable gasto de energía para su funcionamiento, además, su voluminoso tamaño junto con los instrumentos que requiere para funcionar lo vuelve incomodo, difícil de manejar, y un estorbo para el paciente, sumando a todo esto la poca escalabilidad de los dispositivos, pues como se observa, a estos equipos no se les puede colocar un aditamento posible para su mejora.

Esta problemática de equipos con estas características se ve aumentada en lugares donde los puestos de salud son de bajo rango y no poseen capital suficiente para la compra de los mismos, por eso se hace necesaria la pregunta: ¿Se puede realizar el diseño y la implementación de un dispositivo que permita la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioelectricas ECG, que sea una verdadera alternativa confiable, rápida y cómoda debido a su bajo costo de producción, su considerable ahorro de energía, su alta escalabilidad que permita la adición de material para mejoras del mismo, y sea completamente portable, de un tamaño considerablemente reducido, cómodo para el paciente y rápido para el médico especialista?

Este proyecto es presentado como una opción confiable, de bajo costo, rápida, con una alta escalabilidad, de un tamaño considerablemente menor a las que ofrece el mercado, pues tiene por objetivo desarrollar el diseño e implementación de un dispositivo para la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioeléctricas ECG optimizado para su uso portable.

2. JUSTIFICACION

El presente proyecto de grado se enfocará en la producción de un dispositivo escalable, que permita la rápida adquisición, el correcto acondicionamiento, la hábil transmisión inalámbrica y el inteligente almacenamiento de la información como mínimo 4 de las 12 derivaciones de un examen ECG convencional simultáneamente en tiempo real con una resolución mínima de 12 bits y una tasa de muestreo mínima de 2000 Hertz. Esto con el fin de desarrollar una herramienta de diagnóstico precisa, confiable, de bajo costo, con un alto grado de escalabilidad, de un tamaño considerablemente menor que las expuestas en el mercado que ofrezca al médico especialista elementos de juicio objetivos y verídicos que permita disminuir el espectro de posibles afecciones del corazón a ser tratadas.

Con lo anteriormente dicho, este trabajo cobra una gran importancia porque al desarrollarse una herramienta de diagnóstico con un integrado de tipo SMD, permitirá que el producto final tenga un grado de integración considerablemente más grande reduciendo así su tamaño en comparación con los dispositivos en el mercado actual teniendo por meta un tamaño aproximado de 7 cm x 7 cm, conservando la confiabilidad y certeza del examen, reduciendo su costo de producción, con un ahorro de energía sustancial, con la ventaja de poder ser utilizado en lugares distantes de centros médicos donde el diagnóstico de estas enfermedades se hace difícil, tedioso, y en la mayoría de los casos costosos para las personas que requieren de ellos.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Diseñar e implementar un dispositivo para la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioeléctricas ECG optimizado para su uso portable

3.2. OBJETIVOS ESPECÍFICOS

• Discriminar los circuitos integrados de tipos SMD del mercado idóneos para el correcto tratamiento de las señales bioeléctricas ECG.

• Diseñar el dispositivo de sensado de la señal ECG.

• Implementar el dispositivo de sensado de la señal ECG.

• Realizar la adquisición, acondicionamiento, y transmisión inalámbrica de la señal ECG.

• Generar el software de visualización, registro y almacenamiento de los datos adquiridos para ser presentados al usuario.

• Ejecutar una validación de los resultados obtenidos con los resultados que se adquieran de un examen electrocardiográfico realizado con un sistema del mercado, avalado por un especialista.

4. MARCO TEÓRICO

Desde que los métodos no invasivos de análisis se volvieron más confiables y rápidos se han realizado una cantidad extensa de trabajos e investigaciones para realizar mejoras sustanciales en los dispositivos, aquí se mencionaran varios de ellos.

Iniciando a nivel regional se encuentran dos casos significativos, el primero de ellos fue presentado en el año 2003 al programa de Ingeniería Electrónica de la Universidad Surcolombiana, el trabajo de grado titulado "Electrocardiógrafo digital portátil" por Carolina Pascuas y Mauro Fernando Vargas como requisito para optar al título de Ingeniero electrónico."En este trabajo se plantea un diseño etapa por etapa. Básicamente cuenta con una interfaz de conversión analógica/digital para la adquisición de señales dieléctricas, una interfaz digitalizadora que establecen la comunicación con el PC y un software de fácil manejo y entorno gráfico agradable que permite la captura y visualización de la señal cardíaca." Suárez, *Electrocardiógrafo digital portátil*

Como segundo caso en el ámbito regional, se tiene el trabajo presentado al programa de Ingeniería Electrónica de la Universidad Surcolombiana, el proyecto de grado titulado "Prototipo electrocardiógrafo inalámbrico para la detección de enfermedades que desencadenen la muerte súbita, con software de diagnóstico médico aproximado" por Sergio Augusto Tabares Chavarro y Jefferson Perdomo Trujillo. "En este proyecto, fue diseñado un dispositivo electrocardiógrafo que cuenta con diagnóstico para la derivación DII, permitiendo la visualización de las demás derivaciones, una a su vez. De igual modo, el dispositivo electrocardiógrafo, permite realizar la prueba de electrocardiograma (ECG) de forma inalámbrica para facilitar la aplicación de dicha prueba en cualquier lugar o situación. El prototipo está diseñado para facilitar la aplicación de la prueba de ECG en circunstancias cotidianas y desde la comodidad de su hogar, sin perder la eficacia en la prueba. No se puede olvidar que los resultados de la misma, deben ser verificados por personal calificado."Trujillo., *Prototipo electrocardiógrafo inalámbrico para la detección de enfermedades que desencadenen la muerte súbita, con software de diagnóstico médico aproximado*

A nivel nacional se presenta el trabajo en la Universidad San Buenaventura de Cali, titulado "Sistema Portable para la Adquisición y Procesamiento de Señales ECG, con aplicabilidad en dispositivos Móviles" por Santiago VillaFuerte Echeverri y Harby Torres Avila. En este proyecto se "pretende el desarrollo de un sistema de adquisición, procesamiento y visualización de señales ECG, usando a la ingeniería electrónica como una herramienta tecnológica que permita el desarrollo investigativo en el área de las aplicaciones biomédicas."Ávila, *Sistema Portable para la Adquisición y Procesamiento de Señales ECG, con aplicabilidad en dispositivos Móviles*

Como segundo caso en el ámbito regional, se tiene el proyecto presentado al programa de ingeniería electrónica de la Escuela Colombiana de Ingeniería Julio Garavito titulado "Diseño de un dispositivo para la adquisición de una señal EEG que incluya la reducción de artefactos oculares." realizado por Ángela Patricia Guerrero Castillo, que tenia por "objetivo diseñar y probar un dispositivo para la adquisición de señales EEG de miembro inferior que incluya un procesamiento para reducir los artefactos causados por actividad

ocular."Patricia Guerrero Castillo, *Diseño de un dispositivo para la adquisición de una señal ECG que incluya la reducción de artefactos oculares.*

A nivel internacional, en primer lugar se encuentra el trabajo presentado en la Universidad de San Carlos de Guatemala, a la facultad de ingeniera, titulado "Diseño e Implementación de un electrocardiógrafo portátil y del sistema de procesamiento digital de señales eléctricas del corazón, para monitoreo y análisis médico", por Marlon Arturo Pérez Rodas en la que se presenta "una solución en un sistema unificado de electrocardiografía y monitoreo de un paciente, con un dispositivo portátil, de bajo consumo de energía, poco peso y totalmente móvil, que se implementará en La Unidad de Cirugía Cardiovascular de Guatemala (UNICAR)."Rodas, *Diseño e Implementación de un electrocardiógrafo portátil y del sistema de procesamiento digital de señales eléctricas del corazón, para monitoreo y análisis médico*

En segundo lugar se tiene el trabajo presentado a la Universidad Politécnica Salesiana, Sede Cuenca, en Ecuador, titulado "Diseño y construcción de un electrocardiógrafo de 12 derivaciones para el análisis de señales cardíacas" por Guillermo Eduardo Vega Picón, este trabajo de tesis expone "el diseño y construcción de un electrocardiógrafo para análisis de señales cardíacas, este equipo será un prototipo el cual constara con algunas características que poseen los electrocardiógrafos para pruebas de esfuerzo, es decir que este equipo permitirá a la persona analizada realizar ejercicio físico mientras se realiza el examen."Picón, *Diseño y construcción de un electrocardiógrafo de 12 derivaciones para el análisis de señales cardiacas*

Seguidamente se ilustra el trabajo de grado presentado en el Instituto Politécnico Nacional, a la Escuela Superior de Ingeniería Mecánica y Eléctrica, el trabajo de grado titulado "Sistema de Adquisición y Transmisión inalámbrica de señales electrocardiográficas" presentado por Luis Enrique Islas Contreras, Oscar Ortiz, e Iván Alfredo Hernández Pérez. "En él se presenta como objetivo implementar un sistema de comunicación alámbrica, el cual permitirá monitorear las señales electrocardiográficas de un paciente a una distancia mayor a la que ofrecen actualmente los sistemas ECG alámbricos, se pretende que dicho monitoreo pueda ser a través de un ordenador persona común, lo cual ofrece ventaja ya que no se necesitará monitor o equipo especializado o dedicado."Pérez, *Sistema de Adquisición y Transmisión inalámbrica de señales electrocardiográficas*

Se postula el Trabajo "Design of ECG Homecare:12-lead ECG acquisition using single channel ECG device developed on AD8232 analog front end" presentado en Electrical Engineering and Informatics (ICEEI), 2015 International Conference on, en el año 2015, por los investigadores Muhammad Wildan Gifari, Hasballah Zakaria, y Richard Mengko. En él se explica los dispositivos de electrocardiograma (ECG) miden la actividad eléctrica del músculo cardíaco para determinar las condiciones del corazón. La calidad de la señal ECG es el factor clave para determinar las enfermedades del corazón. Sin embargo, la disponibilidad de dispositivos de ECG está causando el diagnóstico de ECG lento y difícil. Este artículo presenta el diseño del dispositivo ECG portátil de un solo canal desarrollado en la plataforma de chips AD8232. Para ampliar la capacidad de un dispositivo portátil ECG, también se prueba una técnica de adquisición de ECG de 12 derivaciones. Los resultados mostraron que el módulo ECG de canal único con una técnica adicional de adquisición de

ECG de 12 derivaciones puede servir a la función del dispositivo de cuidados a domicilio de ECG, lo que hace que el diagnóstico de ECG sea mucho más accesible.[1]

Como última muestra, se observa el trabajo "Design of ECG signal acquisition system based on ADS1291" presentado en Communication Problem-Solving (ICCP), 2016 International Conference On, por los investigadores Jian-Zhi Chen, Liang-Hung Wang, Fa-Xiang Wang, Ming-Hui Fan, en el cual se enuncia la descripción en detalle un sistema de adquisición de electrocardiograma (ECG) compuesto por ADS1291, MSP430 y CC2541. ADS1291 amplifica la señal ECG analógica débil y la transforma en señales digitales y luego la envía al sistema de fondo y la muestra. El paciente puede utilizarlo para adquirir señales de ECG de manera eficaz y sencilla.[2]

4.1. SEÑALES BIOELÉCTRICAS

"Existen una gran variedad de fuentes de señales electrofisiológicas que son de gran interés para el estudio médico, tales como: el corazón, los ojos, el cerebro, los músculos, etc. El más conocido de los estudios es el ECG, en el cual se estudia la actividad cardíaca a través del registro de las señales eléctricas que emite el corazón. Otros son el EEG (electroencefalograma) donde se registra la actividad eléctrica del cerebro; el ERG (electroretinograma) y EOG (electrooculograma) que registran la actividad eléctrica de la retina y el movimiento ocular respectivamente; el EMG (electromiograma) utilizado para la valoración en actividad muscular; EGG asociado a los movimientos peristálticos gastro-intestinales."Oliveri, *Elementos de diseño de circuitos de Amplificación del ECG*

Los valores típicos de amplitud y frecuencia de las señales mas estudiadas en la academia (ECG,EEG,EMG) son presentado en la siguiente imagen.

Figura 1. Valores de Señales bioeléctricas

Señal	Amplitud (mV)	Rango Frecuencial (Hz)
ECG	0.02 - 5.0	0.05 - 100
EEG	0.0002 - 0.3	DC - 150
EMG	0.1 - 5.0	DC - 10000

Fuente: `www.rgi.tut.fi/KURSSIT/7102500/biovahvistin/biovahvistin.pdf`

[1] Mengko, *Design of ECG Homecare:12-lead ECG acquisition using single channel ECG device developed on AD8232 analog front end.*

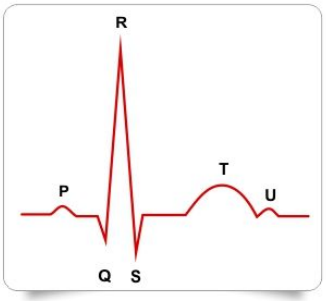
[2] Fan, *Design of ECG signal acquisition system based on ADS1291.*

4.1.1. Electrocardiograma (ECG)

"Un electrocardiograma (ECG) es una medida indirecta de la actividad eléctrica cardíaca. De hecho, es la única medida no invasiva de la que se dispone para este fin. Permite identificar alteraciones anatómicas (por ejemplo, el crecimiento de cavidades) del ritmo e incluso hemodinámicas (como la sobrecarga de presión a nivel cardíaco); pero también procesos sistemáticos como alteraciones iónicas."Autores, *Manual ECG, Electrocardiografía,*

El registro de la actividad eléctrica se presenta en forma de ondas, como dicta a continuación.

Figura 2. Ondas ECG



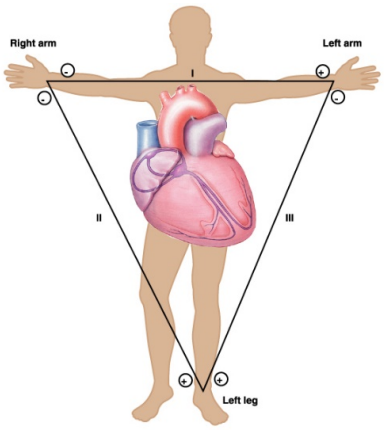
Fuente: `http://tes.juanjosemillan.es/ondas-u-en-electrocardiogramas`

En un ECG es de suma importancia la ubicación de los electrodos en el paciente puesto que la correcta ubicación de los mismos da una lectura verídica de los signos que se desean captar. Para propósitos de este proyecto el ECG se hará con una consideración de 12 derivaciones, de las cuales seis analizan la actividad cardíaca en el plano frontal, estas derivaciones son conocidas como bipolares y aumentadas. Las otras seis los hacen en el plano horizontal, conocidas también como derivaciones precordiales.

4.1.2. Derivaciones de un ECG

Las derivaciones bipolares, también conocidas como D1, D2, D3, registran la diferencia de potencial entre dos puntos. D1 lo hace entre brazo izquierdo y el brazo derecho, D2 registra la actividad entre la pierna izquierda y el brazo derecho, y finalmente D3 hace lo mismo entre la pierna izquierda y el brazo izquierdo, a este posicionamiento de electrodos se conoce como el triángulo de Einthoven, como se muestra en la siguiente figura.

Figura 3. Triángulo de Einthoven



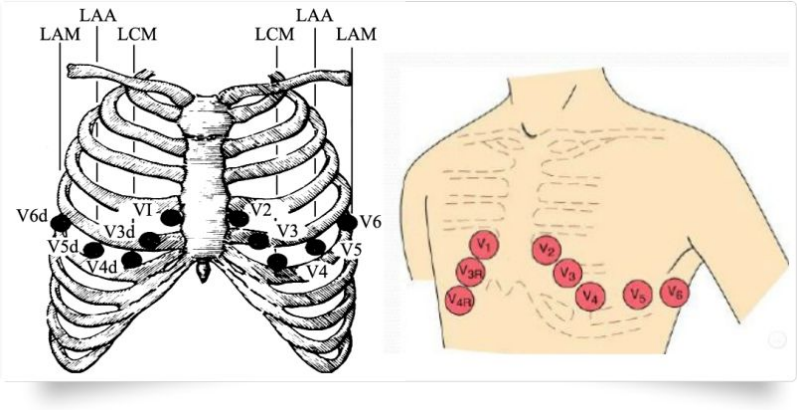
Fuente: `http://163.178.103.176/Fisiologia/cardiovascular/pracb_1/FG14_20.jpg`

Las derivaciones unipolares de los miembros forman el plano frontal y se denominan aumentadas porque miden los potenciales absolutos del brazo derecho, brazo izquierdo, y pie izquierdo.[3]

Las derivaciones precordiales son las derivaciones empleadas para precisar con exactitud las perturbaciones miocárdicas del lado izquierdo y del lado derecho y distinguir las lesiones de la pared anterior y de la pared posterior. Estas 6 derivaciones permiten el registro de potenciales que escapaban a las 6 derivaciones anteriormente citadas; abarcan el tórax, partiendo de su lado derecho y llegan hasta la línea axilar media, es decir, rodean el corazón a manera de un semicírculo[4].

A continuación se muestran las derivaciones unipolares y precordiales

Figura 4. Derivaciones Unipolares y Precordiales



Fuente: `https://www.cuidandote.net/articulos/ECG/ecg03.jpg`

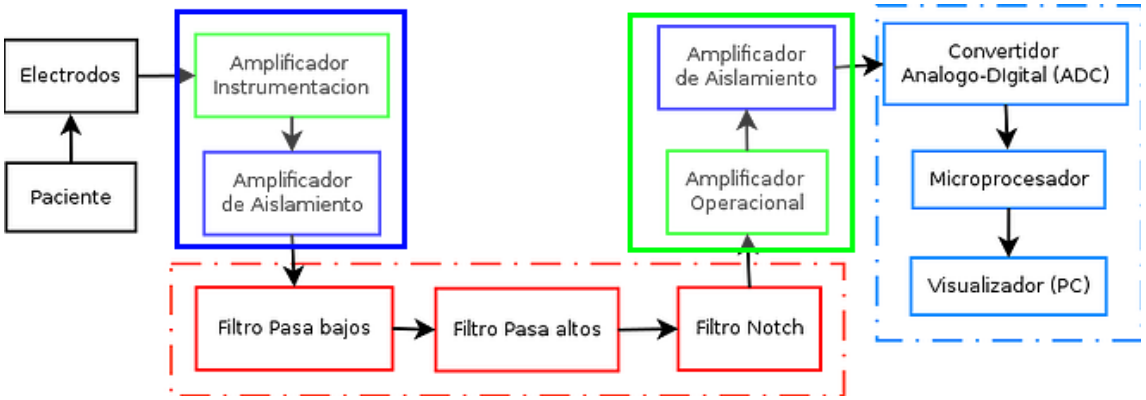
[3] Sandri, *Electrocardiografía*.
[4] Cuba, *Las derivaciones del electrocardiograma*.

Una vez explicadas de manera clara, algunas de las señales bioeléctricas que se pueden encontrar como fuente, y haciendo un especial análisis en la señal de estudio para este proyecto como lo es la señal electrocardiógrafica o ECG, de sus características principales y su origen en el cuerpo, se procede a explicar la composición y funcionamiento de un sistema para el sensado de la señal en cuestión.

4.2. MODELO DE ELECTROCARDIÓGRAFO

El modelo de un electrocardiógrafo es descrito en un diagrama de bloques como muestra la siguiente figura.

Figura 5. Diagrama de bloques para sensado de señal ECG



Fuente: Blog Grupo de Investigacion SIMBIO

4.2.1. Etapas de un ECG convencional

Según el diagrama de bloques presentado, se tienen las siguientes etapas:

• Etapa de Adquisición de la señal ECG: Esta etapa se compone de dos elementos, el paciente fuente principal de la señal de estudio, y los electrodos conectados a el paciente de la manera mencionada en el apartado anterior.

• Etapa de Pre-Amplificación (Bloque Azul): Etapa compuesta por un amplificador de instrumentación encargado de recibir la señal recolectada del paciente. Es utilizado por su alta impedancia de entrada y elevada relación de rechazo en modo común lo cual permite al diseñador amplificar la señal de interés sin distorsionarla ni perder información importante de la misma. Usualmente el factor de amplificación de la señal en este punto está entre 500 y 1000. Luego de este viene un amplificador de aislamiento que es el encargado de aislar al paciente de cualquier corriente de fuga que pueda presentarse desde el aparato electrónico hacia este.

• Etapa de Filtrado (Bloque Rojo): Etapa compuesta de 3 clases de filtros que permiten retirar de la señal de interés cualquier componente que represente un error o ruido; filtro

pasa-bajos que permite el paso de componentes frecuenciales por debajo de cierto rango elegido con un nivel de atenuación escogido, generalmente estos valores son 100 hz a -3dB. Filtro pasa-altos que permite el paso de componentes frecuenciales por arriba de cierto rango y nivel de atenuación elegidos, generalmente estos valores son 0.05 hz a -3dB. Filtro Notch encargado de rechazar una componente frecuencial de interés, generalmente esta componente rechazada corresponde a 60 hz debido a que es la frecuencia que se encuentra en la señal de red eléctrica.

• Etapa de Amplificación (Bloque Verde): Etapa encargada de amplificar de manera final la señal de interés. El factor de amplificación utilizado acá es variante pues depende del sistema que reciba la señal.

• Etapa de Procesamiento (Bloque Azul Cían): Etapa encargada de realizar las conversiones de la señal análogica a señal digital por medio de un circuito ADC. Cabe aclarar que en este punto, en la señal de entrada se ve agregado un ruido de cuantización inherente al proceso de conversión descrito anteriormente, por lo cual para lidiar con este inconveniente algunos diseños utilizan filtros digitales para solucionar este apartado. La señal en este punto ya pasa a estar digitalizada, y puesta en pequeñas partes que de ahora en mas serán llamadas muestras. Las muestras obtenidas son enviadas un microcontrolador o circuito electrónico que se encarga de redirigirlas a un servidor por medio de un transmisor inalámbrico o alámbrico, o por el contrario presentarlo en un software visualizador.

Una vez estudiado el modelo básico de cualquier electrocardiógrafo se procede a explicar el desarrollo del primer objetivo específico, que no es más que buscar en las opciones el mercado posibles candidatos realizando un imperioso análisis de las opciones tomando en cuenta los criterios para el desarrollo del proyecto.

4.3. ESTUDIO DE CIRCUITOS INTEGRADOS DE TIPO SMD

4.3.1. Definición

"Los circuitos integrados SMD o Surface Mount Devices, por sus siglas en inglés son componentes micro miniaturizados con o sin terminales que se sueldan directamente en unas zonas conductoras situadas en la superficie de la PCB llamadas huellas ("lands") sin la necesidad de ser insertados y atravesar la tarjeta."EcuRed, *Tecnología de Montaje Superficial*

Al utilizar integrados de este tipo en el presente proyecto, se inicia a cubrir uno de los objetivos planteados que es la reducción del tamaño del producto pues estos circuitos son considerablemente más pequeños llegando a medir milímetros, logrando un grado de integración importante junto a una reducción sustancial del consumo de energía sumado a que el uso de este tipo de circuito permite unir las etapas de pre-amplificación, filtrado y amplificación en una sola, minimizando el proceso de diseño. Estas razones fueron las que llevaron a la escogencia de este tipo de integrados para la realización del presente proyecto.

4.3.2. Circuitos Candidatos

Después de una exhaustiva y ardua búsqueda de dispositivos SMD disponibles en el mercado se eligieron 3 candidatos que cumplían con los requerimientos del proyecto tales como diseño especializado para la toma de señales bioeléctricas, conversores ADC de alta resolución, bajo consumo de potencia, tasa de muestreo de señal variable, alta relación de rechazo en modo común (CMRR), entre otras. Estos candidatos escogidos fueron ADS1199, ADS1298, ADS1299.

Los circuitos aquí mencionados, cumplen con la condición primordial de ser diseñados específicamente para la obtención de señales bioeléctricas, pero a su vez éstos están diseñados para facilitar la obtención de una de estas señales en específico, por ejemplo los candidatos ADS1198 y ADS1298 cuentan en su diseño con 3 características útiles para la detección correcta de un ECG, como son el circuito integrado de pierna derecha o RLD, terminal especial central de wilson o WCT y la terminal central de Goldberger o GCT estas dos ultimas necesarias para la obtención de un ECG completo, es decir, contemplando sus 12 derivaciones. Mientras que el candidato ADS1299 no posee estas características necesarias para el desarrollo del presente proyecto, por lo cual es descartado desde ahora. Ahora haciendo un análisis minucioso de las hojas de datos de los candidatos restantes se encuentran diferencias importantes tales como:

• Resolución de conversores: ADS1198 posee una resolución de conversión de 16 bits por muestra mientras que ADS1298 posee una resolución de 24 bits, por muestra, lo cual se traduce en un mejor escalamiento de las muestras tomas.

• Frecuencia de muestreo: ADS1198 tiene frecuencias de muestreo que varían desde 250 a 8000 muestras por segundo y su contra parte tiene un rango más amplio de frecuencias que va de 250 a 32000 muestras por segundo.

• CMRR: mientras el primero pose un relación de rechazo en modo común de -105dB y el segundo -115dB.

Teniendo en cuenta las características mostradas en la figura 6, y sopesando las diferencias mostradas en el apartado anterior junto al precio mostrado por ambos candidatos, se escoge el circuito ADS1298 por sus altas prestaciones y características que son consideradas idóneas para el desarrollo del presente proyecto y del cual se analizará a fondo en el siguiente apartado.

Las principales características de los candidatos mencionados son detalladas en la siguiente imagen.

Figura 6. Tabla comparativa de integrados candidatos

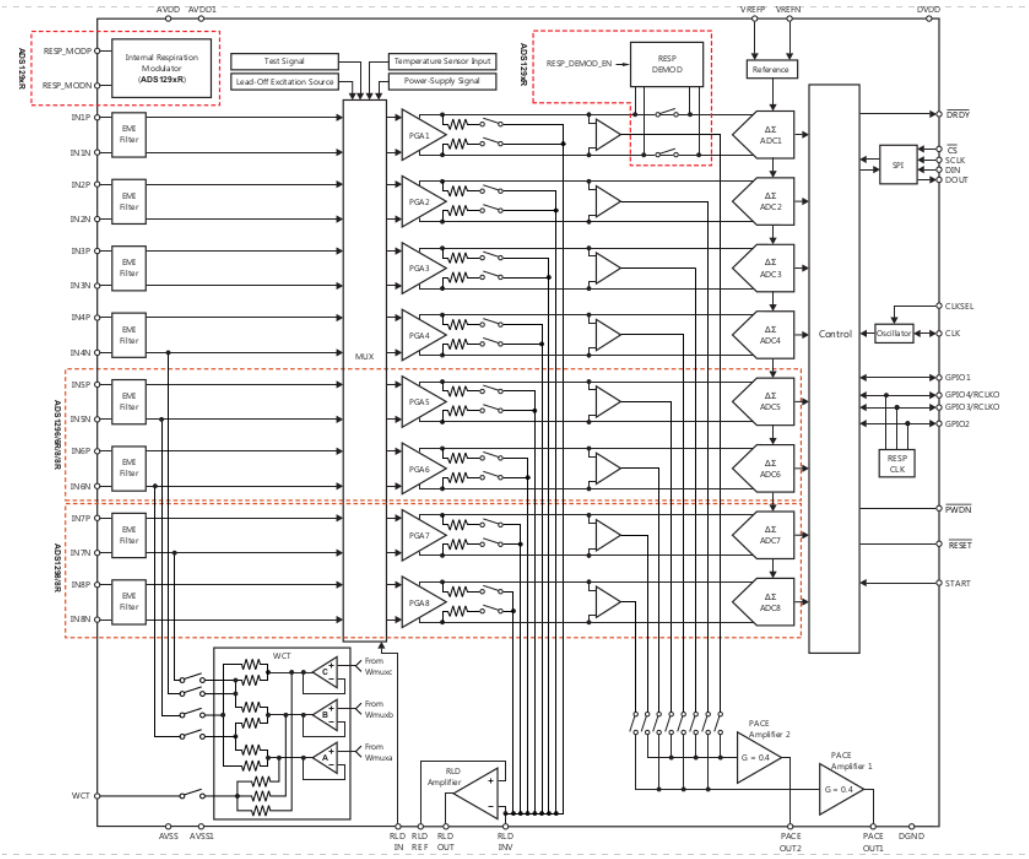
	ADS1299 Order now Online datasheet Datasheet Tools & Software	ADS1298 Order now Online datasheet Datasheet Tools & Software	ADS1198 Order now Online datasheet Datasheet Tools & Software
Resolution (Bits)	24	24	16
Number of input channels	8	8	8
Sample rate (Max) (kSPS)	16	32	8
Interface	SPI	SPI	SPI
Architecture	Delta-Sigma	Delta-Sigma	Delta-Sigma
Input type	Differential	Differential Single-Ended	Differential Single-Ended
Multi-channel configuration	Simultaneous Sampling	Simultaneous Sampling	Simultaneous Sampling
Package Group	TQFP | 64	NFBGA | 64 TQFP | 64	NFBGA | 64 TQFP | 64
Power consumption (Typ) (mW)	41	6	4.3
SNR (dB)	121	112	97
Approx. price (US$)	36.00 | 1ku	22.75 | 1ku	14.35 | 1ku

Fuente:
`http://www.ti.com/assets/js/compareParts/compare.html?familyId=2019&parts=ADS1299,ADS1298,ADS1198&cols=o1,p84,p1028,p593max,p158,p89,p776,p3091,p2954,p989typ,p88,p1130&lang=en`

4.3.3. Integrado ADS1298

El circuito integrado ADS1298 se presenta como un dispositivo diseñado específicamente para el sensado de la señal ECG, con 8 canales dedicados para realizar la inserción de la señal de interés, cuya configuración puede ser cambiada de entrada diferencial a entrada directa según el diseñador requiera.

Figura 7. Esquematico ADS1298



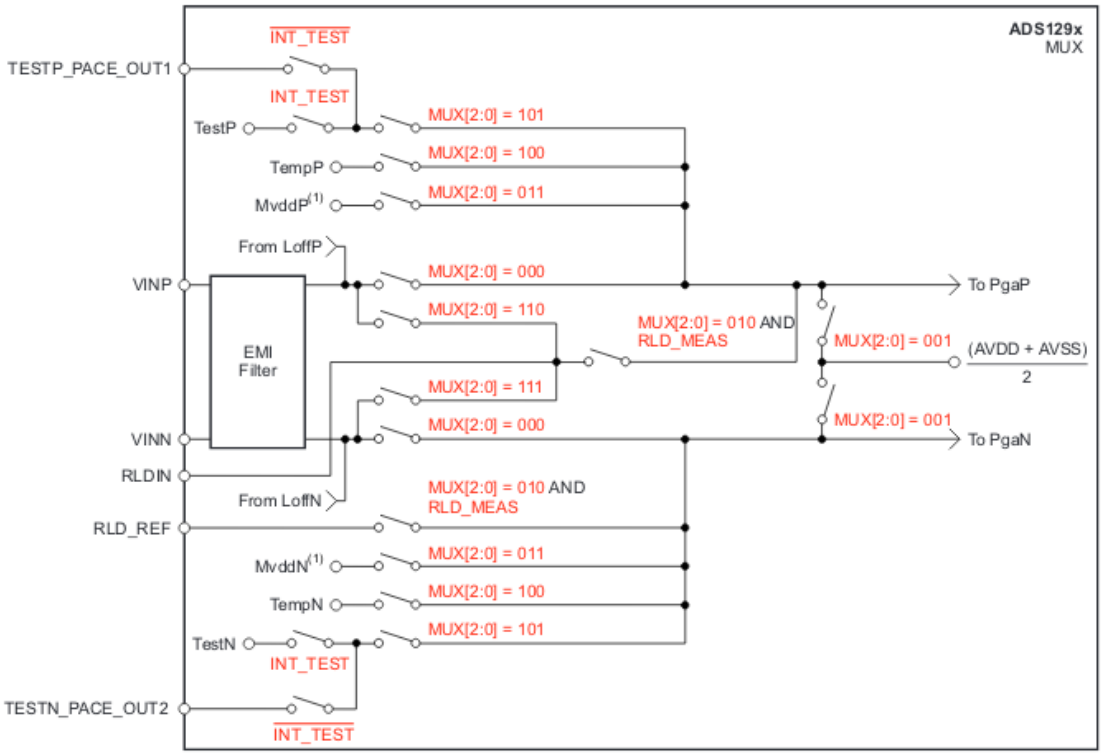
Fuente: Datasheet ADS1298, pag 25

Entre las características principales de este dispositivo se encuentran:

• Conversores ADC Delta-Sigma: es un tipo especial de conversor de señales analógicas puesto que utiliza un sobre-muestreo de la señal y la técnica de modulación Delta-Sigma, permitiendo así desplazar el ruido producido por la conversión fuera del ancho de banda de interés de la señal de entrada. El conversor aquí implementado tiene una resolución de 24 bits, una tasa de muestreo que puede variar según lo defina el diseñador que va desde 250 muestras, pasando por 500, 1000, 2000, 4000, 8000, 16000, hasta 32000 muestras por segundo.

• Multiplexor de entrada: en cada uno de los 8 canales posee un multiplexor que permite al diseñador realizar una amplia lista de funciones entre las cuales se encuentran la generación de una señal de prueba de forma cuadrada con parámetros específicos que sirve como diagnóstico del dispositivo, medición de los voltajes de alimentación, medición de la temperatura entre otras.

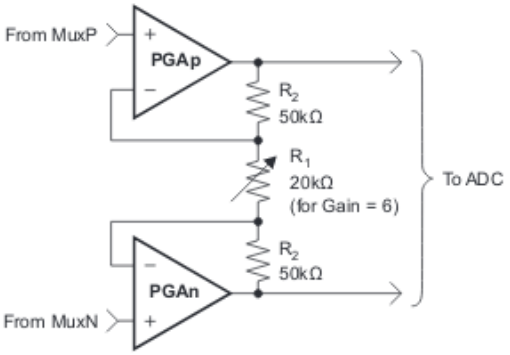
Figura 8. Multiplexor de Entrada ADS1298



Fuente: Datasheet ADS1298, pag 27

• Amplificadores de ganancia programable (PGA): este permite al diseñador amplificar la señal de entrada en un factor que considere correcto para su diseño. Las ganancias permitidas son 1,2,3,4,6,8 y 12.

Figura 9. Amplificador de Ganancia Programable (PGA)

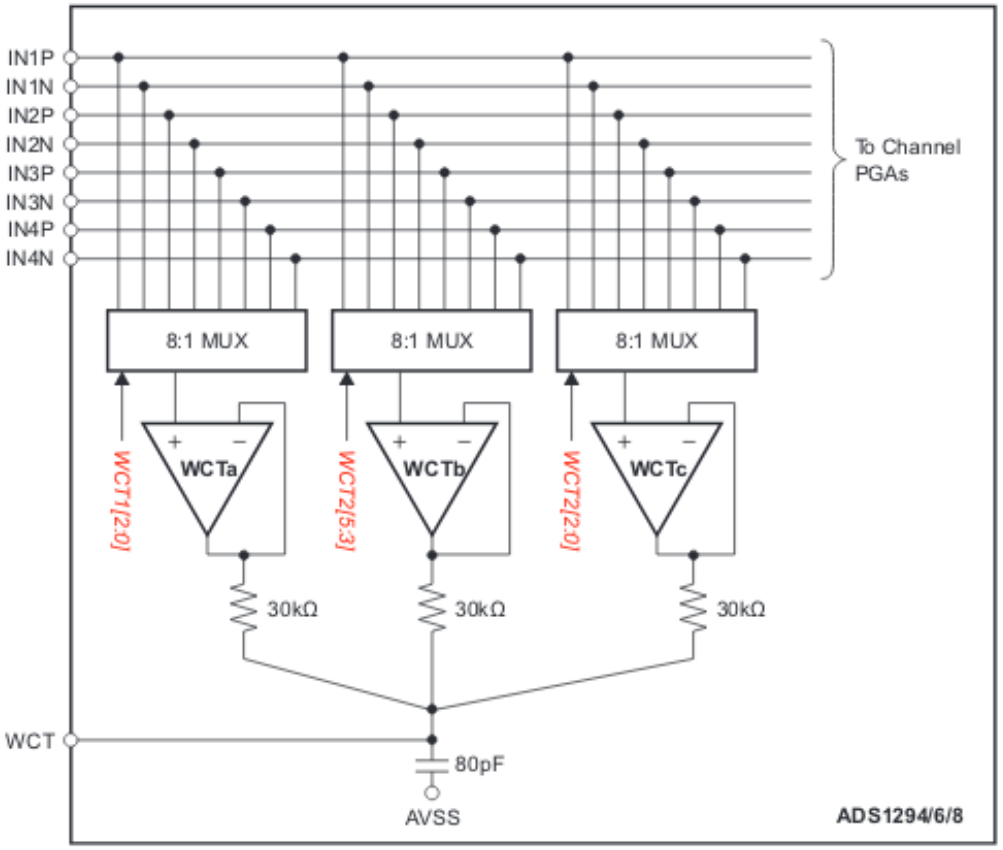


Fuente: Datasheet ADS1298, pag 31

• Terminal Central de Wilson (WCT): Terminal especialmente diseñada para obtener la

señal igual al promedio de las señales de entrada del brazo derecho, brazo izquierdo, y pierna izquierda, necesaria para determinar correctamente las derivaciones precordiales indicadas en apartados anteriores.

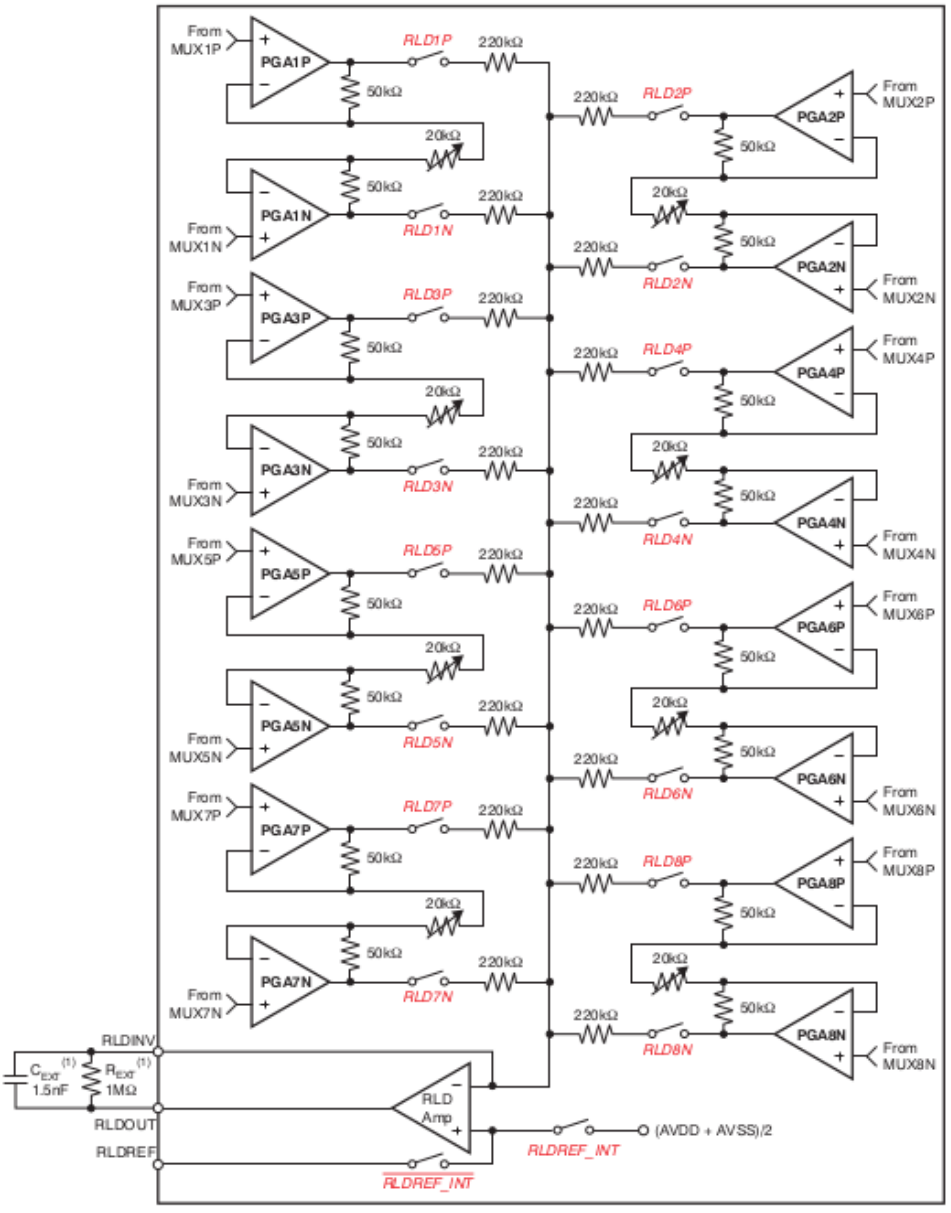
Figura 10. Terminal Central De Wilson (WCT)



Fuente: Datasheet ADS1298, pag 36

• Circuito de pierna derecha (RLD): Es el circuito necesario para la protección del paciente, pues crea una referencia virtual para la pierna derecha del paciente, haciendo uso de las señales de brazo derecho, brazo izquierdo y pierna derecha.

Figura 11. Circuito de Pierna Derecha (RLD)



Fuente: Datasheet ADS1298, pag 42

4.4. ELECCIÓN DISPOSITIVO DE DESARROLLO

Una vez elegido el circuito encargado de captar, filtrar y convertir en muestras digitales la señal ECG de interés, se hace necesario hacer el estudio y elección de un dispositivo de desarrollo. Esta etapa del proceso se corresponde con la etapa de procesamiento presentada en la figura 5, teniendo en cuenta que el dispositivo escogido en esta etapa, será encargado del proceso de manipulación y envió de tramas, mientras que el componente más exigente del procesamiento será realizado por el servidor diseñado que se explicará en apartados

posteriores. Esta plataforma tiene que poseer el puerto de comunicación SPI que es el protocolo que utiliza el ADS1298 para comunicarse y ser programado, por lo cual se realizó una búsqueda en el mercado de las opciones posibles y se escogió la siguiente:

4.4.1. Rsapberry Pi Zero W

El Raspberry Pi es un micro ordenador o una placa de computadora SBC de bajo costo desarrollada en el Reino Unido por la Fundación Raspberry Pi, de manera que pueda fomentar la enseñanza de la computación y programación en las escuelas y colocarla al alcance de todos, como fue en un pasado cercano, por ejemplo en los años 80 cuando los niños creaban programas y juegos en sus computadoras personales.[5]. Sus características son presentadas en la siguiente imagen:

Figura 12. Características Raspberry Pi Zero W

- **CPU:** Broadcom BCM2835 **1Ghz**: Un procesador **mononúcleo** que incorpora una potencia de un 40% superior a la Raspberry Pi 1.
- Memoria **Ram** de **512MB** tipo LPDDR2.
- Lector para **tarjetas Micro-SD**.
- Dos **conexiones micro-USB** para alimentación e Intercambio de Datos.
- **40 Pines** de conexión **GPIO** externos iguales que los modelos A+,B+ Y 2B.
- **Salida Mini-HDMI** para audio y **video** hasta **1080P.**
- **GPU** Dual Core VideoCore IV con 512MB de RAM.
- Salida de video RCA (Pines para soldar puerto)
- **Dimensiones**: 65MM x 30mm x 5mm

Fuente:`https://tublogen3d.com/raspberry-pi/zero-w/`

Siendo Raspberry Pi Zero W un ordenador de tamaño reducido, se hace necesario instalar en él un sistema operativo que permita el control adecuado de todas las funciones que ésta proporciona, para tal fin se hace uso del sistema operativo Raspbian, sistema basado en la distribución de Linux conocido como Debian. Luego de realizar la instalación de dicho sistema, se activan las interfaces SPI, SSH, puertos GPIO, y ademas se determina una IP fija para facilitar el manejo de la placa de forma remota.

[5]Culturacion, *http://culturacion.com/raspberry-pi-que-es-caracteristicas-y-precios/*.

5. DESARROLLO PRÁCTICO

5.1. DISEÑO DE SISTEMA DE ADQUISICIÓN

Una vez escogido la placa de desarrollo a utilizar y el circuito encargado de la adquisición, se procede a realizar el diseño del sistema completo, siempre teniendo en cuenta las especificaciones técnicas del ADS1298. Las etapas de diseño son: Alimentación, Filtrado, y Funcionamiento.

5.1.1. Etapa 1: Alimentación ADS1298

Esta etapa consiste en proveer de manera correcta la alimentación eléctrica necesaria para el buen funcionamiento del sistema de adquisición. Las condiciones necesarias para el funcionamiento correcto se encuentran reunidas en dos apartados: Condiciones Recomendadas de Operación, en este apartado se especifican los rangos recomendados de voltaje de alimentación, rango de entrada de señales analógicas, voltaje de referencia para los conversores, entradas digitales, entrada de reloj y temperatura, estos rangos son presentados en la siguiente imagen:

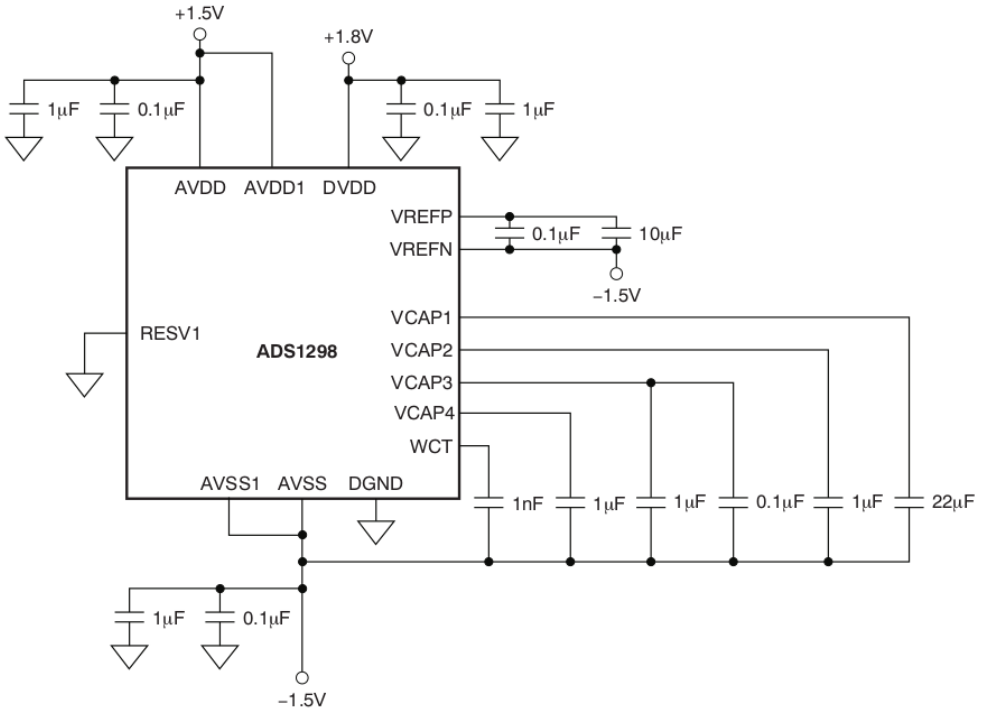
Figura 13. Condiciones Recomendadas de Operacion ADS1298

		MIN	NOM	MAX	UNIT
POWER SUPPLY					
Analog power supply (AVDD – AVSS)		2.7	3	5.25	V
Digital power supply (DVDD)		1.65	1.8	3.6	V
AVDD – DVDD		–2.1		3.6	V
ANALOG INPUTS					
Full-scale differential input voltage range (AINP – AINN)		$\pm V_{REF}$ / Gain			V
Common-mode input voltage		See the *Input Common-Mode Range* subsection of the *PGA Settings and Input Range* section			
VOLTAGE REFERENCE INPUTS					
Differential reference voltage	3-V supply V_{REF} = (VREFP – VREFN)		2.5		V
	5-V supply V_{REF} = (VREFP – VREFN)		4		V
Negative input (VREFN)			AVSS		V
Positive input (VREFP)			AVSS + 2.5		V
CLOCK INPUT					
External clock input frequency	CLKSEL pin = 0	1.94	2.048	2.25	MHz
DIGITAL INPUTS					
Input Voltage		DGND		DVDD	V
TEMPERATURE RANGE					
Operating temperature range	Commercial grade	0		70	°C
	Industrial grade	–40		85	°C

Fuente: DataSheet ADS1298, pag 12

El segundo apartado son las Recomendaciones de alimentación, en este punto se ilustra de manera sencilla las dos maneras recomendadas para la alimentación del ADS1298, la conexión unipolar y la conexión bipolar. La diferencia entre ambas radica en la referencia de la fuente positiva, en la primera, la fuente positiva esta referenciada a un valor de 0 voltios, mientras que en la segunda configuración ésta se encuentra referenciada con un valor de voltaje simétrico al cual se alimenta la fuente positiva. Este último detalle es crucial pues permite al ADS1298 disfrutar de una polarización más estable en comparación a la configuración unipolar, por lo cual se escoge esta última.

Figura 14. Conexion Bipolar de Alimentación



Fuente:DataSheet ADS1298, pag 97

Los valores de AVDD, AVSS, DVDD Y DGND, son escogidos por conveniencia a 2.5 voltios, -2.5 voltios, 3.3 voltios, y 0 voltios respectivamente, debido a que la mayoría de sistemas electrónicos en el mundo son polarizados con esta convención, eso si siempre teniendo en cuenta las condiciones de operación enunciadas anteriormente (1) AVDD-AVSS <= 5.25 voltios, (2) DVDD <= 3.6 voltios.

Para lograr los valores indicados anteriormente se hace uso en primera medida, de los puertos GPIO 1,2,6 de la placa de desarrollo Raspberry Pi Zero W, que proveen 3.3 voltios, 5 voltios, y 0 voltios (Tierra) respectivamente. Esto permite obtener dos valores necesarios (DVDD y DGND), para obtener los valores escogidos de AVDD y AVSS se plante el siguiente diseño haciendo uso de los siguientes circuitos, todos bajo la configuración recomendada por cada una de las hojas de datos respectivas. Como guía para el diseño de fuentes presentado, se tomó como base el modulo de evaluación (SDU) del ADS1298 provisto por Texas Instruments.

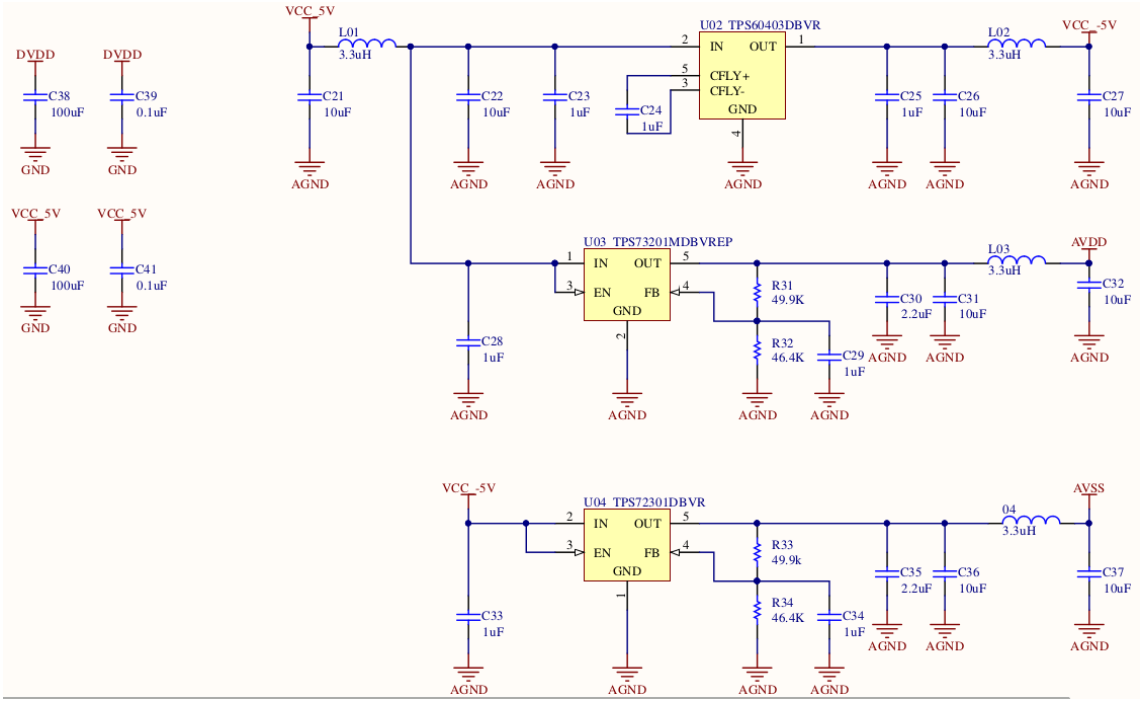
• Inversor de Voltaje TPS60403DBVR: pone en su salida, el voltaje invertido puesto en su entrada, con una corriente máxima de 60 mA. La utilización de este circuito se hace necesaria para invertir el voltaje de 5 voltios que provee el pin GPIO 2 de la placa de desarrollo a -5 voltios.

• Regulador de Voltaje TPS73201MDBVREP: Es el encargado de regular el voltaje positivo de 5 voltios, reduciéndolo hasta el valor deseado de 2.5 voltios.

• Regulador de Voltaje TPS72301DBVR: Es el encargado de regular el voltaje positivo de -5 voltios, reduciéndolo hasta el valor deseado de -2.5 voltios.

La siguiente imagen muestra el diseño terminado:

Figura 15. Fuentes de Alimentación AVDD Y AVSS

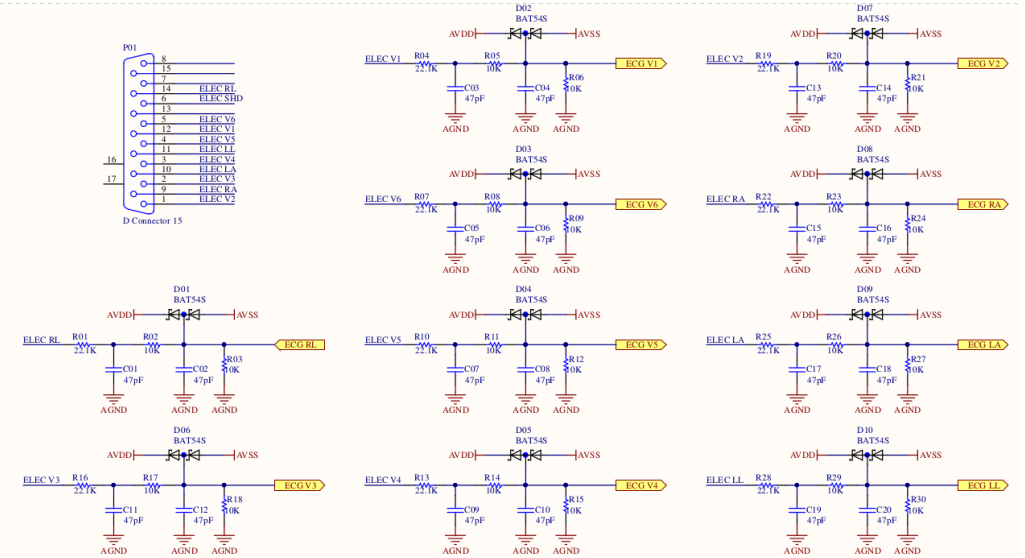


Fuente: Autor (2019)

5.1.2. Etapa 2: Filtrado

Entre de las ventajas que presenta la utilización del integrado ADS1298 existen dos que merecen mención en este apartado, la primera es la implementanción de un filtro de interferencia electromagnética (EMI FILTER) de 3 Mhz a -3 dB, y la segunda es conversión Sigma-Delta, que permite entre otras cosas, la disminución de requerimientos de filtrado para un sistema de sensado de señales bioeléctricas por lo cual, basta con la utilización de filtros pasivos de un sólo polo ubicados en cada una de las entradas que vayan conectadas al paciente, ademas acompañados de diodos de alta velocidad BAT54S con el fin de asegurar la protección del integrado. Los filtros pasivos tiene frecuencias de corte 153 khz y 338 khz y son presentados en la siguiente imagen:

Figura 16. Filtros de Entrada



Fuente: Autor (2019)

5.1.3. Etapa 3: Funcionamiento

Como ya se mencionó la configuración a utilizar será la bipolar, y se hacen las siguientes configuraciones para lograr un optimo desempeño:

• Conexiones a extremidades: estas se hacen para obtener las señales del plano frontal, correspondiente a las derivaciones bipolares a saber D2 que es la diferencia entre la señal de la pierna izquierda (LL) y la señal del brazo derecho (RA), esta conexión se hará a la entrada diferencial #1, IN1P y IN1N respectivamente. D1 que es la diferencia entre la señal del brazo izquierdo (LA) y la señal del brazo derecho (RA), esta conexión se hará a la entrada diferencial #2, IN2P y IN2N respectivamente.

• Conexiones WCT: Como ya se mencionó anteriormente, el integrado ADS1298 posee en su composición un bloque dedicado a producir la señal WCT que es el promedio de las señales de las extremidades del paciente RA, LA, LL, para así poder restar ésta última a las señales del plano sagital, pára así obtener las derivaciones precordiales V1, V2, V3, V4, V5, V6. En la siguiente imagen se ilustra de mejor manera este apartado.

Figura 17. Calculos Derivaciones Precordiales

V1	Precordial	V1-(RA+LA+LL)/3
V2	Precordial	V2-(RA+LA+LL)/3
V3	Precordial	V3-(RA+LA+LL)/3
V4	Precordial	V4-(RA+LA+LL)/3
V5	Precordial	V5-(RA+LA+LL)/3
V6	Precordial	V6-(RA+LA+LL)/3

Fuente:`http://www.dalcame.com/ecg.html`

El pin generador de la señal WCT se conectará a las entradas diferenciales negativas restantes (IN3N A IN8N) y las señales del plano sagital a las entradas positivas como se muestra a continuación: V2 a IN3P, V3 a IN4P, V4 a IN5P, V5 a IN6P, V6 a IN7P, y V1 a IN8P.

• Conexión RLD: esta conexión es necesaria para poder referenciar al paciente a una tierra virtual, permitiendo la correcta recolección de las señales de las extremidades.

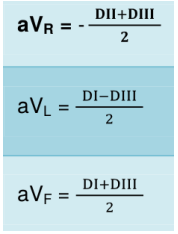
Los pines CLKSEL, CS, RESET, y PWDN, son todos puestos a fuente digital (DVDD) pues sólo se activan en bajo y su utilización a travéz de medios físicos es descartada, permitiendo así desactivarlos de manera física. Además se implementá un cristal de 2.048 Mhz, para la generación de la señal maestra de reloj CLK, necesaria para el correcto funcionamiento del integrado.

Para la obtención de las 4 señales restantes se hará el procesamiento por software necesario a saber:

• D3: Es importante conocer que las derivaciones estándares están íntimamente relacionadas, guardando una proporción entre sí, de modo que el voltaje de los fenómenos que se recogen en D1, D2 y D3 tienen una relación matemática enunciada en la ley del propio Einthoven, que postula: D2 es igual a la suma de D1 más D3.[6] Por lo tanto se puede obtener la señal D3 haciendo una simple resta entre D2 y D1 (D3 = D2-D1).

• Derivaciones Aumentadas: Las derivaciones aumentadas aVR, aVL, aVF se obtienen por medio de procesos matemáticos partiendo de las derivaciones unipolares D1, D2, y D3, como muestra la siguiente figura.

Figura 18. Derivaciones Aumentadas en Funcion de las derivaciones unipolares D1, D2, D3

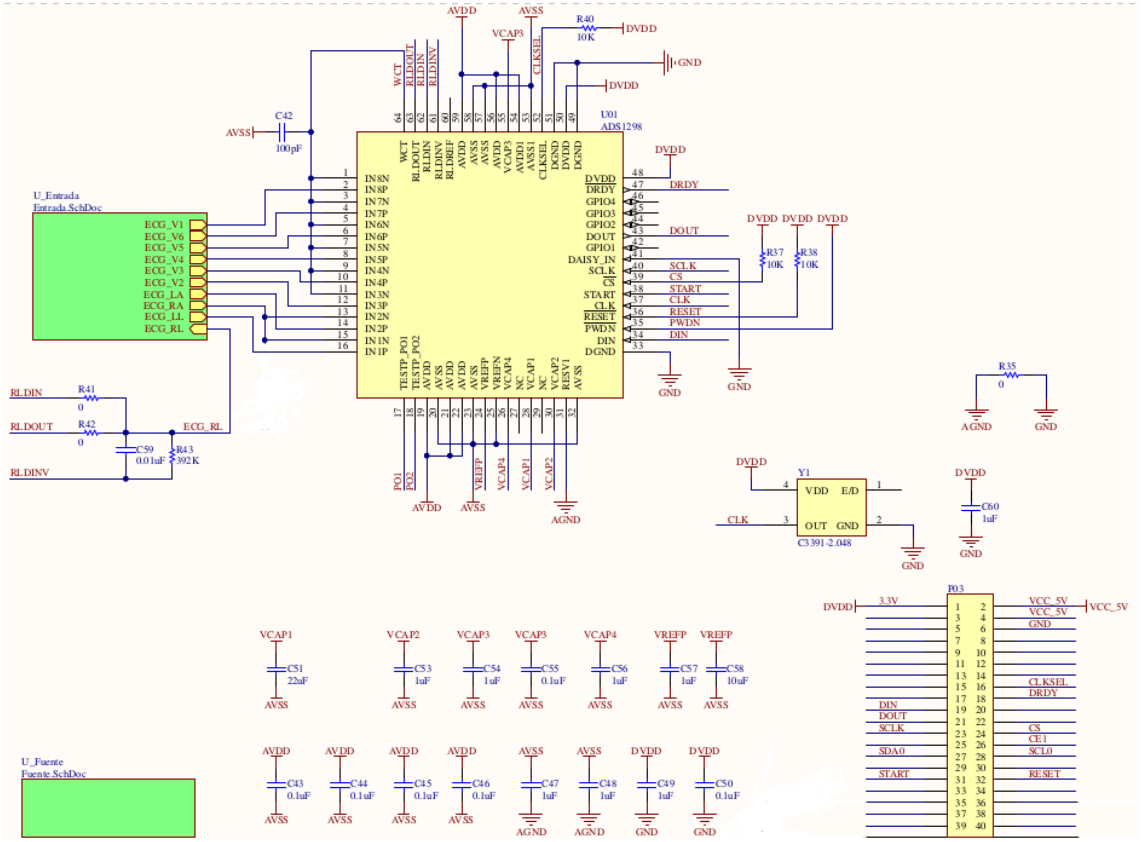


Fuente: `http://www.einthoven.nl/Einthoven-algemeen/einthoven_historical_pictures.html`

El diseño final es presentado en la siguiente imagen:

[6] Cuba, *Las derivaciones del electrocardiograma.*

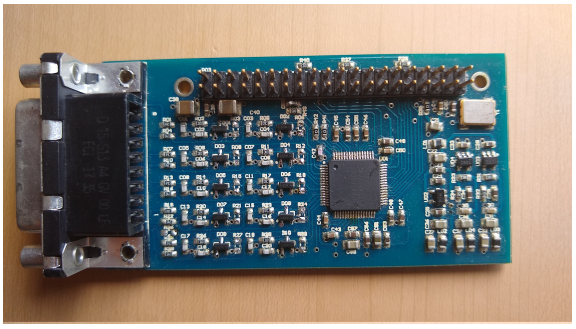
Figura 19. Diseño Final



Fuente: Autor (2019)

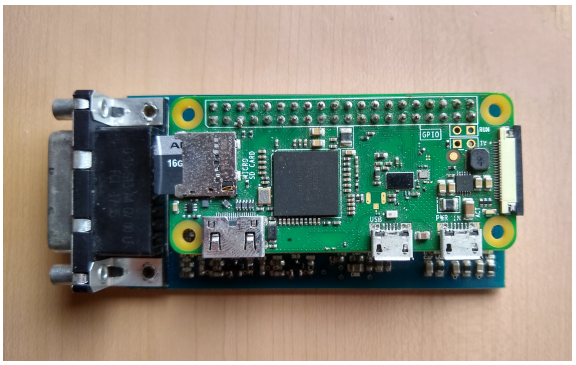
A continuación se muéstra las vistas del diseño implementado:

Figura 20. Diseño Final: Vista Superior



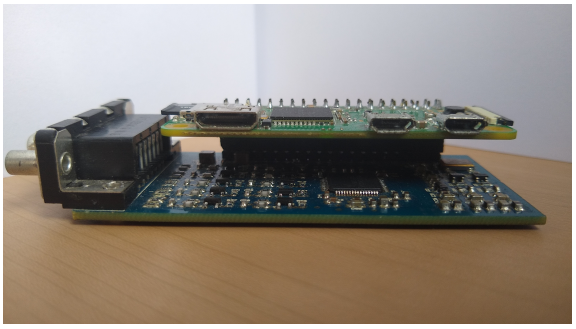
Fuente: Autor (2019)

Figura 21. Diseño Final: Vista Superior Raspberry Pi Zero W



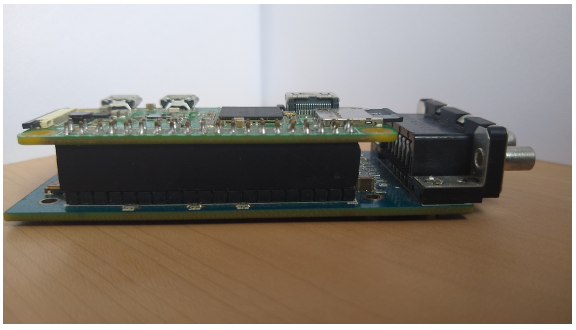
Fuente: Autor (2019)

Figura 22. Diseño Final: Vista Lateral Derecha



Fuente: Autor (2019)

Figura 23. Diseño Final: Vista Lateral Izquierda



Fuente: Autor (2019)

5.2. VERIFICACIÓN DE FUNCIONAMIENTO

Una vez diseñado e implementado el sistema de adquisición de señales bioeléctricas, se procede a realizar la verificación de funcionamiento, por medio de las siguientes pruebas: (1) Lectura de registro de identificación, (2) Recolección y transmisión de señal cuadrada de prueba generada por el mismo ADS1298. Todo el diseño de software para llevar a cabo estas pruebas, y el diseño final serán realizados bajo el lenguaje de programación Python, pues este integra una facilidad considerable de sintaxis y rápido funcionamiento, que lo hace indicado para las prestaciones requeridas por el proyecto. La instalación de librerías e interpretes necesarios para el funcionamiento del proyecto se irán mencionando a medida de que se va explicando cada apartado. Al instalar el sistema operativo Raspbian en instancias anteriores, este sistema trae consigo la implementación del lenguaje Python 3.7 por defecto.

5.2.1. Prueba de lectura de registro de identificación

Esta prueba consiste en realizar la lectura del registro de identificación del integrado ADS1298. Para ello primero se debe crear el script con las características idóneas para la correcta comunicación entre la placa de desarrollo Raspberry Pi Zero W, y la placa diseñada con el integrado ADS1298. Para comunicar las dos placas se hace uso del sistema de comunicación basado en el protocolo SPI que tiene el integrado ADS1298. Este protocolo de comunicaciones es utilizado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Consiste en 4 lineas de información de las cuales una de estas está dedicada al envío de la señal de reloj por lo cual también es conocido como un protocolo síncrono.

Las 4 líneas son:

• Línea SCLK: Una línea dedicada al envío de la señal de reloj necesaria para la transmisión y recepción de tramas entre el maestro y el esclavo. Generalmente esta señal contiene 8 bits de unos y ceros simétricos (10101010).

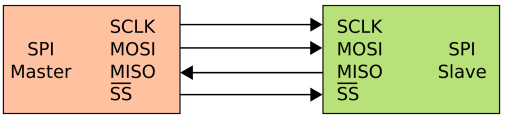
• Línea CS: También conocida como línea de autorización, generalmente activa en bajo. Dependiendo de si ésta activa o no, permite la decodificación de las tramas que entren al circuito.

• Línea DOUT: Permite el envío de información desde el maestro hacia el esclavo.

• Línea DIN: Permite el envió de información desde el esclavo hacia el maestro.

En la siguiente figura se muestran las líneas anteriormente mencionadas:

Figura 24. Protocolo SPI: Líneas de comunicación



Fuente:
`https://es.wikipedia.org/wiki/Serial_Peripheral_Interface#/media/File:SPI_single_slave.svg`

Como se menciono en el desarrollo del Marco teórico, se dejó activo la interfaz SPI de la placa de desarrollo Raspberry Pi Zero W, para poder comunicar las dos placas. Siendo Python un lenguaje ampliamente difundido se tienen librerías ya probadas para todo tipo de funciones, para la presente comunicación SPI se hace uso de la librería SPIdev. Esta es una librería especializada en utilizar el puerto SPI que tiene integrado la placa de desarrollo escogida.

Para realizar esta prueba se tienen en cuenta el periodo de muestreo (T_{DR}), que será puesto al mínimo, en el modo High Resolution Mode, es decir 0.002 segundos, correspondiente a una frecuencia de muestreo de 500 muestras por segundo, cuatro veces el periodo de la señal de reloj maestra ($4T_{CLK}$) es decir, 0.4882x10-^{6} segundos, el numero de bits (N_{BITS}), que es 24 pues es la resolución en la que trabaja los convertidores Sigma-Delta, y el numero de canales ($N_{CHANNELS}$),igual a 8 pues se utilizarán los 8 canales disponibles, el resultado de periodo mínimo que obtenemos es de 9.0909x10-^{6}, es decir 110 Khz. Para evitar dificultades de decodificación por estar muy cerca del limite de la frecuencia necesaria, se toma un valor aproximadamente 10 veces mayor al encontrado, dejando como parámetro 1 Mhz. Estas consideraciones se hacen teniendo en cuenta la formula mostrada en la página 59 de la hoja de datos del integrado ADS1298.

Una vez se ha configurado en su totalidad el protocolo de comunicación SPI, se procede a estudiar los códigos de comunicación que tiene el integrado ADS1298, estos pueden ser encontrados en la hoja de datos del mismo y se presentan en la siguiente figura:

Figura 25. ADS1298: Comandos Interfaz SPI

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)

Fuente: DataSheet ADS1298, pag 61

En este apartado hay que hacer dos menciones: la primera de ellas corresponde a las restricciones que poseen los códigos de RREG y WREG, pues estos comandos serán totalmente ignorados por parte del ADS1298, si este se encuentra operando en el modo de lectura continua RDATAC, que es el modo por defecto cuando se inicializa el integrado, por eso se hace necesario, antes que nada enviar el código de parada del modo de lectura continua SDATAC. Como segunda mención, es necesario decir que los códigos RREG y WREG tienen una estructura especial pues se componen de dos partes distintas, la primera de ellas es el código propio (001 para RREG y 010 para WREG) concatenado con la dirección del registro (r rrrr) del cual se quiere iniciar la lectura o escritura. Para eso se observa al mapa de registros que se encuentran en la hoja de datos del ADS1298, mostrado en la siguiente figura:

Figura 26. ADS1298: Mapa de Registros

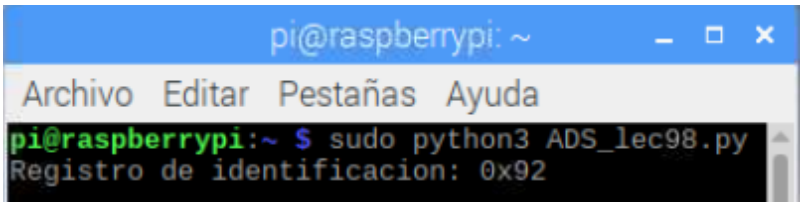
ADDRESS	REGISTER	RESET VALUE (Hex)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DEVICE SETTINGS (READ-ONLY REGISTERS)										
00h	ID	xx	DEV_ID7	DEV_ID6	DEV_ID5	1	0	DEV_ID2	DEV_ID1	DEV_ID0
GLOBAL SETTINGS ACROSS CHANNELS										
01h	CONFIG1	06	HR	DAISY_EN	CLK_EN	0	0	DR2	DR1	DR0
02h	CONFIG2	40	0	0	WCT_CHOP	INT_TEST	0	TEST_AMP	TEST_FREQ1	TEST_FREQ0
03h	CONFIG3	40	PD_REFBUF	1	VREF_4V	RLD_MEAS	RLDREF_INT	PD_RLD	RLD_LOFF_SENS	RLD_STAT
04h	LOFF	00	COMP_TH2	COMP_TH1	COMP_TH0	VLEAD_OFF_EN	ILEAD_OFF1	ILEAD_OFF0	FLEAD_OFF1	FLEAD_OFF0
CHANNEL-SPECIFIC SETTINGS										
05h	CH1SET	00	PD1	GAIN12	GAIN11	GAIN10	0	MUX12	MUX11	MUX10
06h	CH2SET	00	PD2	GAIN22	GAIN21	GAIN20	0	MUX22	MUX21	MUX20
07h	CH3SET	00	PD3	GAIN32	GAIN31	GAIN30	0	MUX32	MUX31	MUX30
08h	CH4SET	00	PD4	GAIN42	GAIN41	GAIN40	0	MUX42	MUX41	MUX40
09h	CH5SET (1)	00	PD5	GAIN52	GAIN51	GAIN50	0	MUX52	MUX51	MUX50
0Ah	CH6SET (1)	00	PD6	GAIN62	GAIN61	GAIN60	0	MUX62	MUX61	MUX60
0Bh	CH7SET (1)	00	PD7	GAIN72	GAIN71	GAIN70	0	MUX72	MUX71	MUX70
0Ch	CH8SET (1)	00	PD8	GAIN82	GAIN81	GAIN80	0	MUX82	MUX81	MUX80
0Dh	RLD_SENSP (2)	00	RLD8P (1)	RLD7P (1)	RLD6P (1)	RLD5P (1)	RLD4P	RLD3P	RLD2P	RLD1P
0Eh	RLD_SENSN (2)	00	RLD8N (1)	RLD7N (1)	RLD6N (1)	RLD5N (1)	RLD4N	RLD3N	RLD2N	RLD1N
0Fh	LOFF_SENSP (2)	00	LOFF8P	LOFF7P	LOFF6P	LOFF5P	LOFF4P	LOFF3P	LOFF2P	LOFF1P
10h	LOFF_SENSN (2)	00	LOFF8N	LOFF7N	LOFF6N	LOFF5N	LOFF4N	LOFF3N	LOFF2N	LOFF1N
11h	LOFF_FLIP	00	LOFF_FLIP8	LOFF_FLIP7	LOFF_FLIP6	LOFF_FLIP5	LOFF_FLIP4	LOFF_FLIP3	LOFF_FLIP2	LOFF_FLIP1
LEAD-OFF STATUS REGISTERS (READ-ONLY REGISTERS)										
12h	LOFF_STATP	00	IN8P_OFF	IN7P_OFF	IN6P_OFF	IN5P_OFF	IN4P_OFF	IN3P_OFF	IN2P_OFF	IN1P_OFF
13h	LOFF_STATN	00	IN8N_OFF	IN7N_OFF	IN6N_OFF	IN5N_OFF	IN4N_OFF	IN3N_OFF	IN2N_OFF	IN1N_OFF
GPIO AND OTHER REGISTERS										
14h	GPIO	0F	GPIOD4	GPIOD3	GPIOD2	GPIOD1	GPIOC4	GPIOC3	GPIOC2	GPIOC1
15h	PACE	00	0	0	0	PACEE1	PACEE0	PACEO1	PACEO0	PD_PACE
16h	RESP	00	RESP_DEMOD_EN1	RESP_MOD_EN1	1	RESP_PH2	RESP_PH1	RESP_PH0	RESP_CTRL1	RESP_CTRL0
17h	CONFIG4	00	RESP_FREQ2	RESP_FREQ1	RESP_FREQ0	0	SINGLE_SHOT	WCT_TO_RLD	PD_LOFF_COMP	0
18h	WCT1	00	aVF_CH6	aVL_CH5	aVR_CH7	aVR_CH4	PD_WCTA	WCTA2	WCTA1	WCTA0
19h	WCT2	00	PD_WCTC	PD_WCTB	WCTB2	WCTB1	WCTB0	WCTC2	WCTC1	WCTC0

Fuente: DataSheet ADS1298, pag 65

La segunda parte de estos códigos (000n nnnn), corresponde a la cantidad de registros que se desean leer. Para efectos de simplicidad, se leerá un registro a la vez, es decir que si se desean leer 5 registros se tendrá que enviar 5 veces el comando de lectura para cada uno de los registros que se deseen leer. Como esta prueba consiste en leer el registro e identificación, el código para dicho fin corresponde a: 001(0 0000), RREG + Dirección de registro de identificación (0 0000)m formando el byte 0x20, luego debe enviarse la segunda parte del código que como se explicó es la cantidad de registros a leer y se hará uno por uno por lo cual el siguiente byte es 0x00. Por ultimo se debe enviar un tercer byte nulo, es decir, 0x00, todo esto con el fin de que la respuesta que sea enviada por el ADS1298 pueda ser correctamente guardada en memoria por la placa de desarrollo.Todo esto se lleva acabo con la instrucción id_register = spi.xfer2([0x20,0,0]). Esta misma explicación toma validez para la instrucción de escritura de registros, siempre teniendo en cuenta que para esta última, el inicio del primer byte es 010. El código diseñado para la lectura de registros posee por nombre ADS_lec98.py y se encuentra en los anexos del presente documento.

En la siguiente figura se muestra el resultado de la lectura del registro de identificación:

Figura 27. ADS1298: Lectura del Registro de Identificación



Fuente: Autor (2019)

Para corroborar que el resultado obtenido por el algoritmo diseñado es correcto, se revisa la tabla del registro de identificación, en la hoja de datos del ADS1298, como muestra la siguiente figura:

Figura 28. ADS1298: Registro de Identificación

7	6	5	4	3	2	1	0
DEV_ID[7:5]			1	0	DEV_ID[2:0]		
R-x			R-2h		R-x		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 17. ID Control Register Field Descriptions

Bit	Field	Type	Reset	Description
7:5	DEV_ID[7:5]	R	xh	**Device ID** These bits indicate the device family. 000 = Reserved 011 = Reserved 100 = ADS129x device family 101 = Reserved 110 = ADS129xR device family 111 = Reserved
4:3	RESERVED	R	2h	**Reserved** Always read back 2h
2:0	DEV_ID[2:0]	R	xh	**Channel ID** These bits indicates number of channels. 000 = 4-channel ADS1294 or ADS1294R 001 = 6-channel ADS1296 or ADS1296R 010 = 8-channel ADS1298 or ADS1298R 011 = Reserved 111 = Reserved

Fuente: DataSheet ADS1298, pag 66

Haciendo la conversión correspondiente del resultado obtenido por el algoritmo (0x92), se encuentra que su equivalente en el código binario es (1001 0010). Encontrándose así correspondencia con la tabla del código de identificación pues ésta indica que los primeros 3 bits (100) corresponden a la familia de integrados ADS129x, los dos siguientes bits (10) son códigos reservados que se tienen guardado en memoria, y los últimos 3 bits (010) indican puntualmente que el dispositivo utilizado es el ADS1298, por lo cual se considera que esta prueba ha sido exitosa y se puede pasar al siguiente apartado.

5.2.2. Prueba de recolección y transmisión de señal cuadrada de prueba generada por el mismo ADS1298

Una vez logrado leer el registro de identificación, y de entender el manejo de las instrucciones de lectura y escritura de registros, se procede a dar uso a una de las características mas importantes y útiles del ADS1298 que es la generación de una onda cuadrada de frecuencia y amplitud modificables por el usuario, para comprobar que el dispositivo funciona correctamente. Para ello se hace uso de los registros de configuración,

CONFIG1, CONFIG2, CONFIG3, y CH1SET a CH8SET, con direcciones 0x01,0x02,0x03 y 0x05 a 0x0C respectivamente. A continuación se explicará que función cumple cada uno de estos registros, y como son configurados para obtener el resultado deseado.

• Registro de Configuración 1 (CONFIG1): Es el registro encargado de definir las configuraciones básicas de funcionamiento del ADS1298 como muestra la figura:

Figura 29. ADS1298: Registro de Configuración 1 (CONFIG1)

7	6	5	4	3	2	1	0
HR	$\overline{\text{DAISY\_EN}}$	CLK_EN	0	0	DR[2:0]		
R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-6h		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 67

El bit 7 (HR) indica que modo de conversión se utilizará, si el low-power mode o el high-resolution mode, la única diferencia entre estos dos modos, es que con el primero la máxima frecuencia muestreo va hasta 16 kSPS y en el segundo va hasta 32 kSPS, para efectos de simplicidad se utilizará el high-resolution mode, esto se logra haciendo 1 este bit. El bit 6 (DAISY_IN)indica si el integrado ADS1298 se conectará y retro-alimentará con un integrado de la misma familia, como en el desarrollo del presente proyecto sólo se utilizará el integrado ADS1298, se hace este bit igual a 0. El bit 5 (CLK_EN) determina si se usa la salida de la señal maestra de reloj para sincronizar el funcionamiento con otros integrados de la misma familia, como se mencionó anteriormente no se hace uso de otro integrado por lo cual este bit también se pone en 0. Los bits 4 y 3 son reservados de memoria, por lo cual a la hora de escribirlos siempre enviará el bit que se encuentra guardado, en este caso es 0 para el bit 4 y 0 para el bit 3. Los bits 2, 1 y 0 (DR) corresponden a la frecuencia de muestreo de la señal que se quiere recolectar, como se explicó anteriormente para estas pruebas la frecuencia de muestreo se escogerá en 500 SPS, por lo cual estos bits toman el valor 110 respectivamente. Haciendo un recuento se tiene que el registro de configuración 1 (CONFIG1) quedará configurado con los bits 1000 0110, correspondiente al código 0x86 en el sistema hexadecimal.

• Registro de Configuración 2 (CONFIG2): Es el registro que determina las características de la señal de prueba que generara el ADS1298 para corroborar su funcionamiento. En la siguiente figura se muestra los bits pertinentes:

Figura 30. ADS1298: Registro de Configuración 2 (CONFIG2)

7	6	5	4	3	2	1	0
0	0	WCT_CHOP	INT_TEST	0	TEST_AMP	TEST_FREQ[1:0]	
R/W-1h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 68

En este registro los bits 7 y 6 son reservados así que se escribirán los valores original 0 y 0 respectivamente. El bit 5 indica las frecuencias de disparo del bloque que tiene la función de generar la señal WCT (Terminal central de wilson) este bit es innecesario configurarlo para la prueba que se realizará, sin embargo, se dejará configurado en 0 para evitar problemas de disparo en la generación de la señal antes mencionada cuando se vayan a tomar los datos de un paciente real. Este 0 indica que las frecuencias serán variables como indica la tabla 7 de la hoja de datos del integrado ADS1298. El bit 4 indica como se manejará la señal de prueba, si ésta es ingresada externamente se coloca 0, si por el contrario la señal de prueba quiere generarse internamente desde el ADS1298, se coloca 1, por lo cual se tomará esta ultima opción. El bit 3 es reservado, así que se escribe con el valor original 0. El bit 2 determina la amplitud de la señal de prueba generada, si se coloca 0 la señal tendrá una amplitud igual al voltaje de referencia dividido 2400 veces, si se coloca 1, será dos veces el voltaje de referencia dividido entre el mismo escalar mencionado en al opción anterior, para este bit se fija el valor 0. Por último, los bits 1 y 0 indican la frecuencia que tendrá la señal de prueba, por lo cual se escoge los valores 0 y 0 para lo bits debido a que esta frecuencia corresponde a la frecuencia de reloj maestra divida en un escalar correspondiente a 2 elevado a la potencia 21. Haciendo un recuento se tiene que el registro de configuración 2 (CONFIG2) quedará configurado con los bits 0001 0000, correspondiente al código 0x10 en el sistema hexadecimal.

• Registro de configuración 3 (CONFIG3): Es el encargado de manejar los buffers de referencia, y la operación del bloque manejador de pierna derecha (RLD). Los bits pertenecientes a este registro son mostrados en la siguiente figura:

Figura 31. ADS1298: Registro de Configuración 3 (CONFIG3)

7	6	5	4	3	2	1	0
$\overline{\text{PD\_REFBUF}}$	1	VREF_4V	RLD_MEAS	RLDREF_INT	$\overline{\text{PD\_RLD}}$	RLD_LOFF_SENS	RLD_STAT
R/W-0h	R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 69

En este registro, el bit 7 (PD_REFBUF) indica el funcionamiento del buffer de referencia, si este es manejado externamente se coloca 0, si es generado internamente se coloca 1, que será el caso. El bit 6 es reservado en memoria, por lo cual se escribe su estado original de 1. El bit 5 (VREF_4V) indica el voltaje de referencia positivo que será usado por los ADC Sigma-Delta para realizar las conversiones, este es puesto en 0, que indica un VREFP igual a 2.4 voltios. Los demás bits del registro no son necesarios para realizar la prueba que se describe por lo cual, se escribe en cada uno de ellos 0. Haciendo un recuento se tiene que el registro de configuración 3 (CONFIG3) quedará configurado con los bits 1100 0000, correspondiente al código 0xC0 en el sistema hexadecimal.

• Registros de Canales 1 a 8 (CHnSET): Estos son los encargados de manejar las entradas diferenciales del ADS1298, y las características que poseen. A continuación se muestran los bits pertenecientes a estos registros:

Figura 32. ADS1298: Registro de Configuración de Canales 1 a 8 (CHnSET)

7	6	5	4	3	2	1	0
PD*n*	GAINN*n*[2:0]			0	MUX*n*[2:0]		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

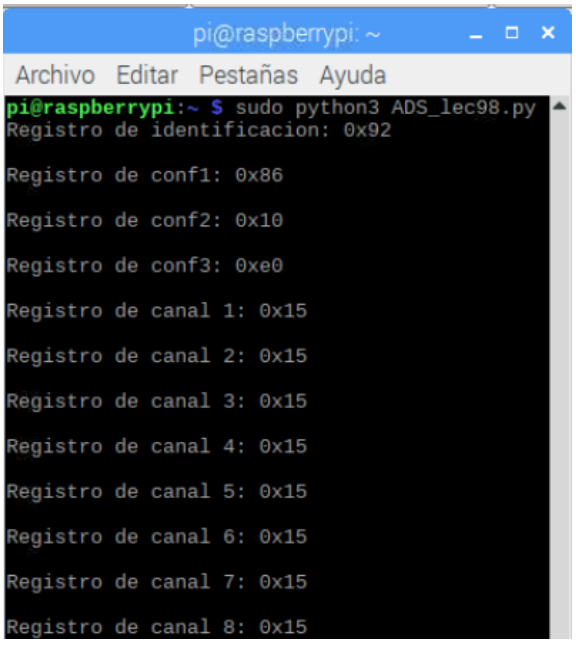
Fuente: DataSheet ADS1298, pag 69

En estos registros, el bit 7 (PDn) indica el estado del canal, si este está encendido (0) o apagado (1), por lo cual este bit estará escrito con 0. Los bits 6,5,4 (GAINn) indican el escalar de los amplificadores de ganancia programable, estos serán escritos con los bits 0,0,1 respectivamente, para obtener una ganancia de 1, es decir que la señal cuadrada generada tenga la amplitud definida en el registro de configuración 2. El bit 3 es un bit reservado en memoria por lo cual se escribe su estado original 0. Los bits 2,1,0 son los bits que indican qué función esta cumpliendo cada canal, por lo cual es necesario escribir en todos los canales 1,0,1 respectivamente en cada bit para que estén preparados para conducir la señal de prueba deseada. Haciendo un recuento se tiene que los registros de configuración de los canales 1 a 8 (CHnSET) quedarán configurados con los bits 0001 0101, correspondiente al código 0x15 en el sistema hexadecimal.

Una vez explicados los registros a utilizar y la configuración que llevará cada uno, se realiza el algoritmo contenido en el script de nombre ADS_esc98pru.py, anexo al presente documento, teniendo por modelo el script de lectura, con la modificación de código de lectura por el de escritura en la siguiente linea del código: wri_conf1_register = spi.writebytes([0x41,0,0x86]), teniendo en cuenta que este caso no es necesario enviar ningún bit nulo, puesto que el comando de escritura necesita de el código WREG concatenado con la dirección de registro de la cual se inicia la escritura (0x41) para el registro de configuración 1, la cantidad de registros a escribir (0x00) es decir, sólo 1 porque se hará la escritura registro por registro, y el valor que se desea escribir en cada registro (0x86) para el registro 1.

En la siguiente figura se muestra el resultado de la escritura de los registros mencionados anteriormente:

Figura 33. ADS1298: Escritura de registros para prueba de señal cuadrada

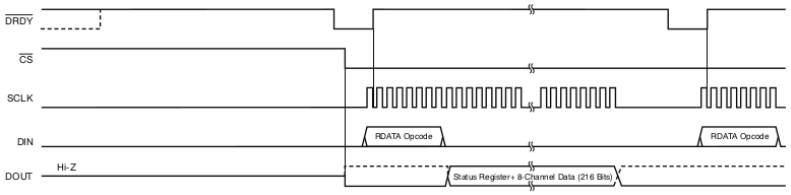


Fuente: Autor (2019)

Una vez configurados los registros para realizar la prueba, se hace necesario diseñar un algoritmo que pueda capturar las tramas que se generarán de la prueba. Para esto, se hace uso del modo de lectura de tramas por el método Single-shot el cual se ajusta mejor a las exigencias del proyecto esto debido a que en opinión del diseñador es mucho mejor enviar los códigos de lectura cada vez que lo indique la señal DRDY.

• Modo de disparo (Single-Shot): Las señales que intervienen y su funcionamiento son iguales que en el modo anterior, solo que en este el que ordena el sincronismo es la placa de desarrollo que controlé el ADS1298, pues en primera instancia es necesario enviar el código de parada de modo continuo (SDATAC) pues este es el modo que está activo por defecto, luego debe enviarse el comando START que ordena al integrado iniciar las conversiones de las señales de entrada, pero para poder leerlas debe enviarse el código RDATA, que permite en este momento enviar las tramas que se tengan en memoria a la placa de desarrollo, cada vez que se desee leer una trama nueva, es decir cuando la señal DRDY, pase de estado alto a bajo, se debe enviar el código RDATA, para poder nuevamente enviar a la placa de desarrollo la trama convertida. La siguiente figura ilustra lo anteriormente dicho:

Figura 34. ADS1298: Modo de lectura Single-Shot



Fuente: DataSheet ADS1298, pag 63

Ya discutidos las opciones de conversión se genera el algoritmo para la recolección de tramas que esta anéxo a este documento bajo el nombre de ADS_cap98.py. Este algoritmo tiene la particularidad de manejar los puertos GPIO de la placa de desarrollo, por lo cual se hace uso de la librería RPi.GPIO importada con el nombre GPIO para simplicidad de escritura del código. Una vez importada, se plantean las constantes que manejará el algoritmo, estas corresponden a los códigos que se enviarán al ADS1298 para ser controlado, y además se plantean como constantes los puertos GPIO de los cuales hará uso el programa. Para saber qué puertos se utilizarán, es necesario remitirse a la documentación de raspberry pi, pues existen dos nomenclaturas para los puertos, una que toma en cuenta el orden en el que están distribuidos físicamente sobre la placa de desarrollo, conocido como numeración normal, y nomenclatura BCM que toma en cuenta el orden de los pines en el microprocesador que usa la placa de desarrollo que es el Bradcom BCM2835. Ésta ultima nomenclatura será la que se use en el programa diseñado y es invocada con la siguiente línea GPIO.setmode(GPIO.BCM). En la siguiente figura se ilustra la distribución de pines teniendo en cuenta las dos nomenclaturas:

Figura 35. Distribución de Pines Raspberry Pi

Fuente: `https://es.pinout.xyz/#`

Como se puede apreciar los pines que se necesitan colocar como constantes son los pines de START, RESET,DRDY,CLKSEL, que corresponden en el diseño y la nomenclatura BCM como los pines 6, 12, 24, y 23 respectivamente. Puestas estas constantes se debe definir que tipo de función cumplirán dichos pines, como se sabe, los pines de START, RESET, y CLKSEL, se manejarán como salidas, pues permitirán controlar adecuadamente el integrado ADS1298, el pin DRDY será puesto como entrada, debido a que este pin recibirá la señal que indica que las conversiones están listas para ser recolectadas por la placa de desarrollo, además de esto se debe definir el estado de este pin en alto pues la detección de tramas se basa en el análisis de flanco de bajada, pues cuando este se presenta quiere decir que las tramas están listas. Realizado este proceso, se configura la interfaz SPI con los mismo criterios utilizados en los códigos de lectura, y escritura de registros. Seguidamente se colocan los pines CLKSEL y RESET en alto para desactivarlos y que no interrumpan el funcionamiento deseado. Se hace una espera de 0.1 segundos y se envía al ADS1298 el código de parada del modo de lectura continua, pues es el modo que está activo por defecto en el integrado. Se define la función llamada capturá, que se encargará de capturar los datos, cuando esta sea llamada, esta se compone del envío del código de lectura de datos (RDATA), una vez enviado el código, se lee el resultado obtenido con el comando de la librería SpiDev (readbytes), esta función tiene por parametro de entrada, la cantidad de bytes a ser leídos por lo tanto, se debe determinar la longitud de la trama que se va a leer, para ello se revisa la documentación del ADS1298, en el apartado Data Retrieval. En él se enumeran dos aspectos que interesan para este caso, la palabra estado, y la longitud total de la trama generada. La siguiente figura ilustra el contenido de la palabra estado:

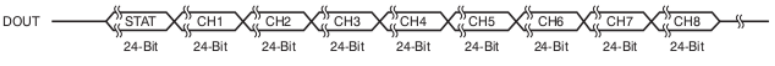
Figura 36. Contenido de la Palabra estado

Fuente: DataSheet ADS1298, pag 53

La palabra estado se compone de un conjunto de 4 bits determinados (1100), un byte que permite saber el estado de las conexiones de las entradas diferenciales positivas, un byte que permite saber el estado de las conexiones de las entradas diferenciales negativas, y por último un conjunto de 4 bits que muestra el estado de los puertos GPIO que posee el ADS1298 para un total de 24 bits.

Con respecto al tamaño total de la trama a enviar, este apartado muestra que la trama es encabezada por la palabra estado ya explicada, y sucedida por conjuntos de 24 bits que tienen la muestra detectada por cada una de las entradas diferenciales, como muestra la siguiente figura:

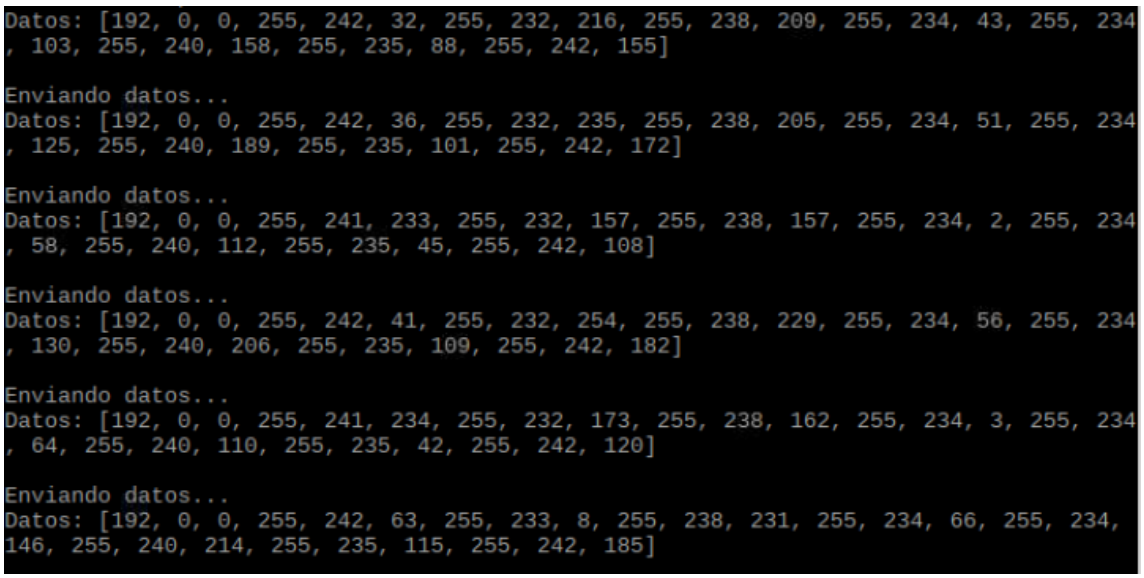
Figura 37. Tamaño Total de Trama Generada



Fuente: DataSheet ADS1298, pag 60

Este tamaño siempre será el mismo, así algunos canales estén inactivos, pues en este caso, el ADS1298 envía una muestra en ceros del mismo tamaño. En ese orden de ideas, se tiene un conjunto de 24 bits correspondientes a la palabra estado, y un conjunto de 192 bits correspondientes a las muestras de los 8 canales a sensar, para un tamaño total de la trama a recibir de 216 bits. Este valor se divide entre 8 debido a que la función de lectura de la librería SpiDev, recibe como parámetro de entrada la cantidad de Bytes a leer, esto da un resultado de 27 bytes a leer, que se ven reflejados en la siguiente línea resp = spi.readbytes(27). Por ultimo también se escribe el comando print, para poder ver en pantalla la trama capturada. Terminada la función captura, lo único que falta es definir un evento en el pin DRDY que detecte el cambio en el estado de la señal DRDY generada por el ADS1298, y al detectar este cambio, ejecute la función anteriormente diseñada. Se coloca una sentencia Try, con el fin de ejecutar una espera de 600 segundos para poder capturar correctamente las tramas. En la siguiente figura se presentan las tramas capturadas con el algoritmo diseñado:

Figura 38. Tramas obtenidas con el algoritmo de captura



Fuente: Autor (2019)

Como se puede observar las tramas han sido correctamente capturadas, pero aún no pueden ser graficadas pues aún se encuentran en la Raspberry Pi Zero W, para dar cumplimiento a uno de los objetivos principales de este proyecto, que es la transmisión inalámbrica de estos datos, se inicia la búsqueda de protocolos que comunicación inalámbrica que puedan ser usados con los recursos que posee la placa de desarrollo. El recurso de la placa de desarrollo Raspberry Pi Zero W elegido para transmitir datos inalámbricamente es Wifi que cuenta con los protocolos de comunicación distintos que serán explicados a continuación:

Entre los protocolos que utiliza Wifi están el protocolo SSH, que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de

conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.[7]. La encriptación que hace este protocolo es robusta, de 128 bits, haciendo que la información enviada sea completamente segura y pueda ser descifrada sólo por quienes mantienen la comunicación. Otro protocolo que utiliza Wifi es el protocolo UDP, pertenece al nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.[8]. Por último se tiene el protocolo TCP el cual es uno de los principales protocolos de la capa de transporte del modelo TCP/IP. En el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él, (es decir, el protocolo IP).[9]. Este garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.[10]

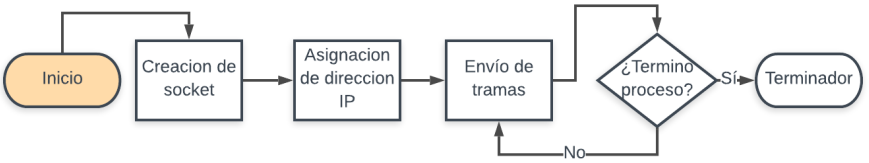
Después de analizar los protocolos de transmisión inalámbrica disponibles para los dispositivos que posee la placa de desarrollo Raspberry Pi Zero W, se escoge, el protocolo TCP, debido a que es el protocolo que asegura la entrega segura y sin errores de los datos a enviar, función vital en el desarrollo del proyecto, debido a que la señal bioeléctrica ECG se presentará para el estudio de enfermedades del paciente, a un especialista, y al tener un error en su envió puede desencadenar en un mal diagnostico por parte de este, llevando a consecuencias indeseables. Ademas de esto es un protocolo rápido, pues no posee una encriptación robusta como el protocolo SSH.

Para implementar este protocolo de envío de información en el código de captura es necesario invocar la librería socket de python que permite crear un objeto que se encargue de la transmisión de las tramas a medida de que estas son capturadas. Una vez invocada la librería socket se procede a declarar como constante, la dirección IP del servidor al cual se enviarán las tramas que en este caso será 192.168.1.5, y el puerto por el cual estas entrarán que para efectos de simplicidad se escogerá el puerto 10000. Posteriormente se crea el objeto sock, con el constructor socket.socket(), se otorgan los atributos de ser un socket de flujo, haciendo que sea un socket del tipo TCP/IPV4 todo con la siguiente línea sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Fijados los parámetros del socket, se le ordena conectarse a la dirección y puerto definidos como constantes anteriormente con la siguiente línea , una vez hecho esto cuando las tramas son capturadas, se ordena al socket que envíe las tramas al socket servidor que estará al pendiente de recibir dichas tramas.

[7]SSH, *https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html*.
[8]UDP, *www.ionos.es/digitalguide/servidores/know-how/udp-user-datagram-protocol/*.
[9]TCP, *https://es.ccm.net/contents/281-protocolo-tcp*.
[10]TCP, *http://www.masadelante.com/faqs/tcp-ip*.

Figura 39. Diagrama de flujo Algoritmo de envío



Fuente: Autor (2019)

Para recibir las tramas se realiza un algoritmo que implemente un socket que actué como servidor para la recepción de las tramas capturadas, así como un algoritmo que permita la detección y organización de las mismas. La diferencia entre estos dos algoritmos radica en que este último, tendrá la tarea de "escuchar" cualquier otro socket, que se encuentre conectado a la red, esta escucha debe ser permanente, y debe estar diseñado para que cuando la transmisión termine, este proceso siga activo a la espera de más tramas. Para realizar el envio de tramas de manera correcta es nesesario convertir el objeto tipo lista en un objeto de tipo bytes, para lo cual se busca en la documentación del lenguaje encontrándose la librería Pickle. Es un paquete que permite serializar, un objeto de cualquier tipo, convirtiéndolo en un objeto de tipo bytes, el cual sólo el lenguaje de python puede interpretar. Para hacer uso de este paquete se hace necesario importarlo al algoritmo de captura de tramas, e invocarlos para convertir la trama obtenida, en una trama susceptible de ser enviada a través por el socket implementado, esto se logra con el comando dumps(). Al serializar una trama se obtiene el resultado mostrado en la siguiente figura:

Figura 40. Forma de trama serializada

Fuente: Autor (2019)

Para comprobar el envío de 10 tramas se revisa el resultado del algoritmo del servidor que se muestra en la siguiente figura:

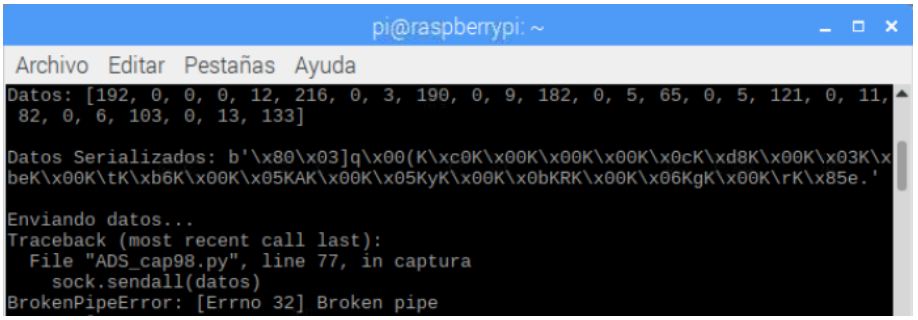
Figura 41. Llegada de tramas enviadas

```
Se creo el Socket Correctamente

Levantando servidor 192.168.43.5, por el puerto 10000

Se asocio correctamente la direccion

Se conecto correctamente con la db
El socket esta escuchando

Esperando Mensaje

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rKaK\x00K\x04K5K\x00K\nK,K\x00K\x05K\x81K\x00K\x05K\xd1K\x00K\x0cK.K\x00K\x06K\xb2K\x00K\rK\xfbe.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\x0cK\xe1K\x00K\x03K\xa3K\x00K\tK\xa5K\x00K\x05K&K\x00K\x05KoK\x00K\x0bK\x8dK\x00K\x06KDK\x00K\rK\x99e.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rK\x83K\x00K\x04K<K\x00K\nKGK\x00K\x05K\x91K\x00K\x05K\xcdK\x00K\x0cK.K\x00K\x06K\xb4K\x00K\x0eK\te.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rK\x17K\x00K\x03K\xcfK\x00K\tK\xd8K\x00K\x05K5K\x00K\x05K\x81K\x00K\x0bK\xc4K\x00K\x06KfK\x00K\rK\xbbe.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rKzK\x00K\x04K6K\x00K\nK=K\x00K\x05K\x86K\x00K\x05K\xc6K\x00K\x0cK%K\x00K\x06K\xb5K\x00K\x0eK\x02e.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rKpK\x00K\x04K9K\x00K\nK8K\x00K\x05K\x88K\x00K\x05K\xd3K\x00K\x0cK/K\x00K\x06K\xacK\x00K\x0eK\x00e.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rKvK\x00K\x04K/K\x00K\nKLK\x00K\x05K\x85K\x00K\x05K\xbcK\x00K\x0cK"K\x00K\x06K\xb4K\x00K\x0eK\x05e.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rKtK\x00K\x04K7K\x00K\nK5K\x00K\x05K\x8bK\x00K\x05K\xd6K\x00K\x0cK/K\x00K\x06K\xbdK\x00K\x0eK\x0ce.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rK\x13K\x00K\x03K\xc9K\x00K\tK\xd9K\x00K\x05K?K\x00K\x05K|K\x00K\x0bK\xc2K\x00K\x06KgK\x00K\rK\xb6e.'

Datos recibidos: b'\x80\x03]q\x00(K\xc0K\x00K\x00K\x00K\rK\x13K\x00K\x03K\xdlK\x00K\tK\xdcK\x00K\x05KBK\x00K\x05K\x81K\x00K\x0bK\xc5K\x00K\x06KiK\x00K\rK\xb3e.'

Transferencia terminada
```

Fuente: Autor (2019)

Comprobado el envío de un conjunto reducido de tramas, entre los sockets se hace la prueba de envío de tramas por un tiempo prolongado (1 a 2 minutos), decodificando las tramas para verlas de manera normal. En la siguiente figura se presenta lo sucedido:

Figura 42. Error de socket roto



Fuente: Autor (2019)

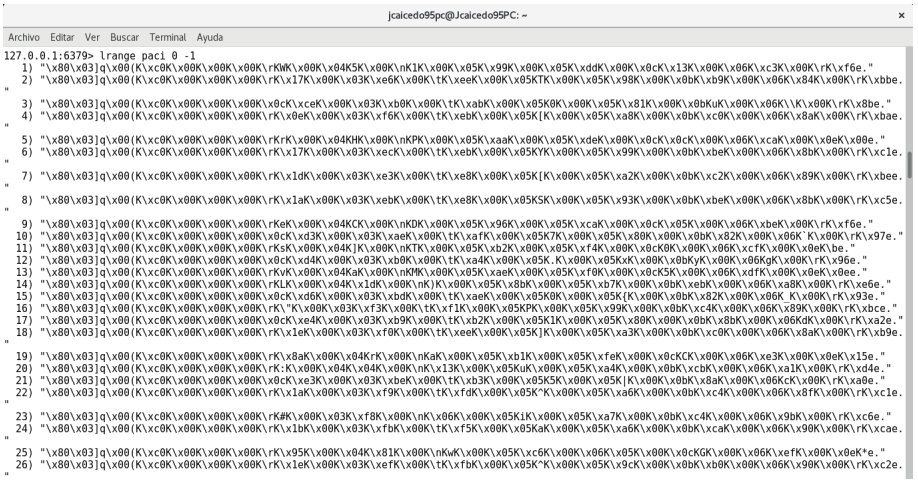
Como se puede ver, al realizar el envió de tramas codificadas del lado del cliente, es decir por parte de la placa de desarrollo del proyecto, luego de un lapso de tiempo en el que las tramas son correctamente recepcionadas y decodificadas por el servidor diseñado, el servidor tiende a cerrarse pues las tramas llegan mas rápido de lo que este puede decodificarlas, por lo cual genera el error presentado en la imagen anterior. Por lo tanto se busca una solución para dicho problema, encontrándose con la opción guardar las tramas que llegan, para luego ser decodificadas por otro algoritmo. Esta es una estrategia válida y eficiente, siempre y cuando el guardado de dichas tramas no afecte su contenido. Para cumplir con este cometido se propone realizar una base de datos que permita el guardado de tramas, a medida que estas

son recepcionadas, para ser más específicos, los gestores de bases de datos que cumplen con la condición de no alterar las tramas, son los gestores de bases de datos no relacionales, pues estos están basados en una filosofía de rápido acceso a la información, y guardado de cualquier tipo de dato, por lo cual se llega al gestor de base de datos no relacionales Redis.

Redis es un gestor de base de datos no relacionales, clave-valor, diseñado en lenguaje C optimizado, persistente en memoria. Esto permite al software acceder de una manera más rápida a la información guardada, debido a que los datos son almacenados en la memoria RAM del dispositivo donde se encuentra instalado el servidor. Entre las distintas ventajas que tiene este gestor de bases de datos, destacan dos, su facilidad para manejar cualquier tipo de dato que ingrese en su órbita, lo cual es muy importante porque el tipo de dato bytes, no es admitido en otros gestores de bases de datos, y además de esto cuenta con la característica de almacenamiento por colas. Este tipo de almacenamiento se basa en una estructura de listas, que permite el fácil ingreso y consulta de datos en la DB. Como última ventaja se encuentra su fácil sintaxis.

Para poder hacer uso de este gestor es necesario hacer la instalación del servidor, y el cliente para el lenguaje de en el cual se manejara el servidor. Luego de esto, se importa la librería de redis, al script del servidor, y se ejecutan las instrucciones de guardado de tramas en la base de datos; para facilidad la base de datos recibirá el nombre de "paci", como muestra la siguiente línea de código: r.rpush('paci',d_recibido). Como se puede apreciar, se tiene el constructor de la clase bautizado r, y de él se hace uso de la función rpush, que tiene por argumentos de entrada, la base de datos donde se ingresará el dato, y el dato a ingresar respectivamente. Al utilizar este comando, se está dando a entender al servidor que la base de datos con nombre 'paci', es una base de datos de colas, que se rellenará con listas. Esto con el fin de mantener la complejidad del manejo de los datos al mínimo, pues el programa encargado de decodificar las tramas, no tiene más acciones que llamar la trama a decodificar, y decodificarla, lo que hace que se aíslen los procesos de recepción y decodificación de tramas, causante del error presentado anteriormente. En la siguiente figura, se presentan las tramas guardadas en el servidor Redis creado:

Figura 43. Tramas Almacenadas en DB Redis



Fuente: Autor (2019)

Comprobado el almacenamiento de tramas en la base de datos ya enunciada, como paso siguiente, atendiendo a la necesidad que se tiene de orden para el correcto guardado de tramas, se implementa un algoritmo de prevención y organización de tramas, teniendo en cuenta dos parámetros, el tamaño de la trama, y la cabecera de la misma. El algoritmo funciona, de manera tal que, si llega una trama completa, es decir, de un tamaño determinado que debe corresponder el tamaño de la trama codificada, que para este caso tiene un tamaño de 62 bytes, esta sea subida a la base de datos inmediatamente, si llega una trama de tamaño menor al indicado, esta sea guardada en una variable aparte a la espera de que la parte que faltante llegue, para que sean unidas, y una vez teniendo la trama requerida esta sea correctamente agregada a la DB de Redis.

En la siguiente figura se muestra el algoritmo de prevención y organización de tramas, diseñado:

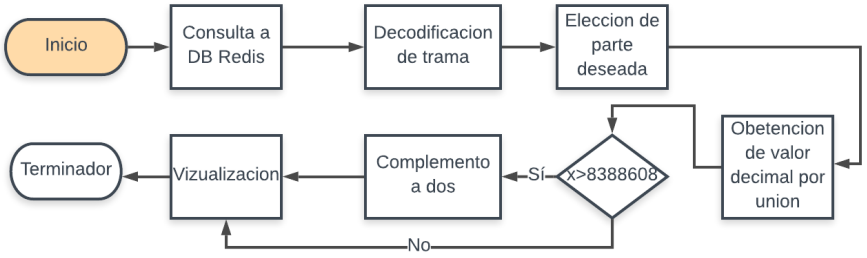
Figura 44. Algoritmo de prevención y organización de tramas

```
d_recibido = co.recv(62)
if len(d_recibido)==6:
    r.rpush('paci', d_recibido)
    print('Transferencia terminada\n')
    break
else:
    if len(d_recibido)==62:
        if d_recibido[0:6]==b'\x80\x03]q\x00(':
            r.rpush('paci',d_recibido)
            print('Indice %s, DatosReci: %s\n' % (contador, d_recibido))
            contador = contador + 1
        else:
            nuevo = espe + d_recibido
            indice = nuevo.find(b'e.')
            indice1 = indice + 2
            r.rpush('paci', nuevo[0:indice1])
            espe = nuevo[indice1:]
            print('Indice %s, DatosReci: %s\n' % (contador, d_recibido))
            contador = contador + 1
    else:
        espe = espe + d_recibido
        print('Indice %s, DatosReci: %s\n' % (contador, d_recibido))
        contador = contador + 1
```

Fuente: Autor (2019)

Asegurados ya los parámetros de envío e integridad de las tramas enviadas, se procede a crear el algoritmo para graficar las tramas recepcionadas para ver la señal cuadrada generada, y dar por completada la prueba. Para ello, se hace uso de la librería PyQtGraph, que es una implementación en python de las librerías de Qt, para la graficación de señales. En la siguiente figura se muestra los pasos a seguir para la graficación de tramas.

Figura 45. Pasos graficación de tramas



Fuente: Autor (2019)

Los dos primeros pasos ya han sido ampliamente explicados en los anteriores apartados, por lo cual de ahora en adelante se explican los pasos siguientes. La elección de parte deseada para la visualización consiste en escoger la parte que se desea visualizar en base a la trama mostrada en la figura 38. Como se puede observar la lista que tiene los datos, se compone de 27 elementos todos estos decimales. Estos elementos corresponden a cada byte capturado por la placa de desarrollo, cuyo contenido ya se enunció anteriormente, correspondiente a una palabra estado y 8 muestras digitalizadas, pertenecientes cada una de ellas a cada uno de los canales del ADS1298. Estas muestras tienen una resolución de 24 bits, es decir que cada conjunto de 3 decimales dentro de la lista presentada, corresponde con una muestra de la señal de entrada en cada canal.

Por lo tanto, se hace necesario definir en el algoritmo, constantes que ayuden a la correcta elección de los limites de cada una de estas muestras. Una vez hecho este proceso se obtiene una lista con los valores deseados, para dar inicio al 4 paso, que consiste en aislar cada uno de estos valores en variables separadas, para poder convertir dicho valor en un valor binario, teniendo en cuanta que este valor binario tiene que estar compuesto obligatoriamente de 8 bits, esto se logra a través de las funciones bin() que permite convertir un valor decimal, en un valor binario y la función zfill() que permite realizar la adición de ceros necesaria para completar los 8 bits necesarios por cada valor decimal. Ya logrado este proceso, se hace la unión de estos valores en una sola variable por medio del operador (+) estos es posible debido a que al completar con ceros cada muestra esta variable pasa a ser de tipo string, lo cual permite concatenar de manera correcta los valores deseados. Concatenados los valores deseados, se hace una última conversión de valores, de este nuevo valor binario con longitud de 24 bits, a un número decimal, que al ser multiplicado por un escalar, dará el valor de la muestra capturada.

Una vez se tiene el número decimal producto de la conversión del numero binario con longitud de 24 bits, se hace necesario determinar la necesidad de realizar el complemento a dos de ese código determinando así, si el código pertenece a una muestra positiva o una muestra negativa, esto resulta de analizar el apartado de formato de palabra de la hoja de datos del ADS1298. La siguiente figura muestra las condiciones de este formato:

Figura 46. ADS1298: Formato de palabra

INPUT SIGNAL, V_{IN} (INxP – INxN)	IDEAL OUTPUT CODE[2]
≥ V_{REF}	7FFFFFh
V_{REF} / (2^{23} – 1)	000001h
0	000000h
–V_{REF} / (2^{23} – 1)	FFFFFFh
≤ –V_{REF} (2^{23} / (2^{23} – 1))	800000h

Fuente: DataSheet ADS1298, pag 54

Como se puede apreciar en la figura anterior, los valores positivos de conversión van desde 000001h (1 en decimal) hasta 7FFFFF (8388607 en decimal), siendo este último valor, el que indica una saturación positiva del canal. Los valores negativos van 800000h (8388608 en decimal) hasta FFFFFFh (16777215 en decimal), siendo el primero de estos valores el que indica una saturación negativa del canal. A estos valores negativos son los que hay

que hacerles el complemento a dos para obtener su valor debido a que esta es su forma normal. Para lograr este cometido, a cada uno de los números que estén por encima del limite inferior (8388608), se les hace la operación XOR, y luego se les adiciona 1 para obtener su version normal, para luego ser multiplicadá el escalar que está determinado por el producto entre valor positivo o negativo de V_{REF}, (según sea la muestra que se desee convertir) y el escalar resultante de dividir la potencia 23 de 2, entre la misma potencia menos 1. Resultado de este proceso, queda el valor de la muestra que se desea graficar.

En la siguiente figura se muestra el algoritmo anteriormente explicado:

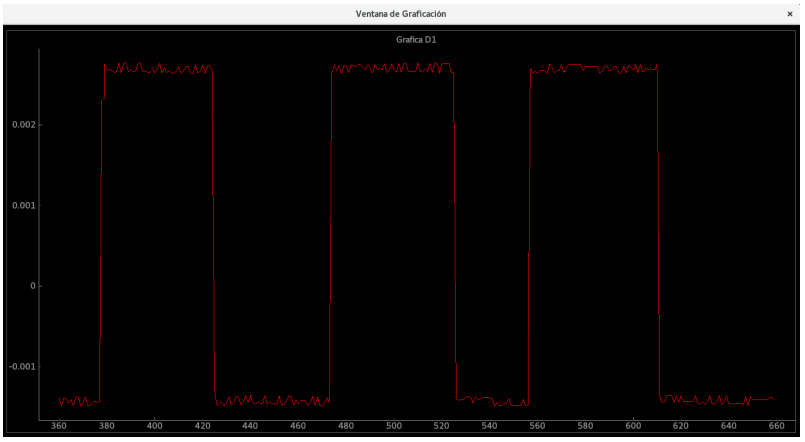
Figura 47. Algoritmo de graficación de tramas

```
db = r.lrange(base,ptr1,ptr1)
dato = db[0]
if len(dato)== 6:
    dataD1 = [0]*300

deco = loads(dato)
D1 = deco[finD2:finD1]
parte1 = bin(D1[0])[2:].zfill(8)
parte2 = bin(D1[1])[2:].zfill(8)
parte3 = bin(D1[2])[2:].zfill(8)
union = parte1 + parte2 +parte3
numero = int(union,2)

if numero>mayor:
    numero1 = numero^comp
    numero2 = numero1 + 1
    gra = -escalar*numero2 + 0.002083333333
    mueD1.append(gra)
else:
    gra = escalar * numero + 0.002083333333
    mueD1.append(gra)
```

Fuente: Autor (2019)

En la siguiente figura se puede apreciar la señal cuadrada de prueba captada por el canal 1:

Figura 48. Gráficas de señal cuadrada canal 1



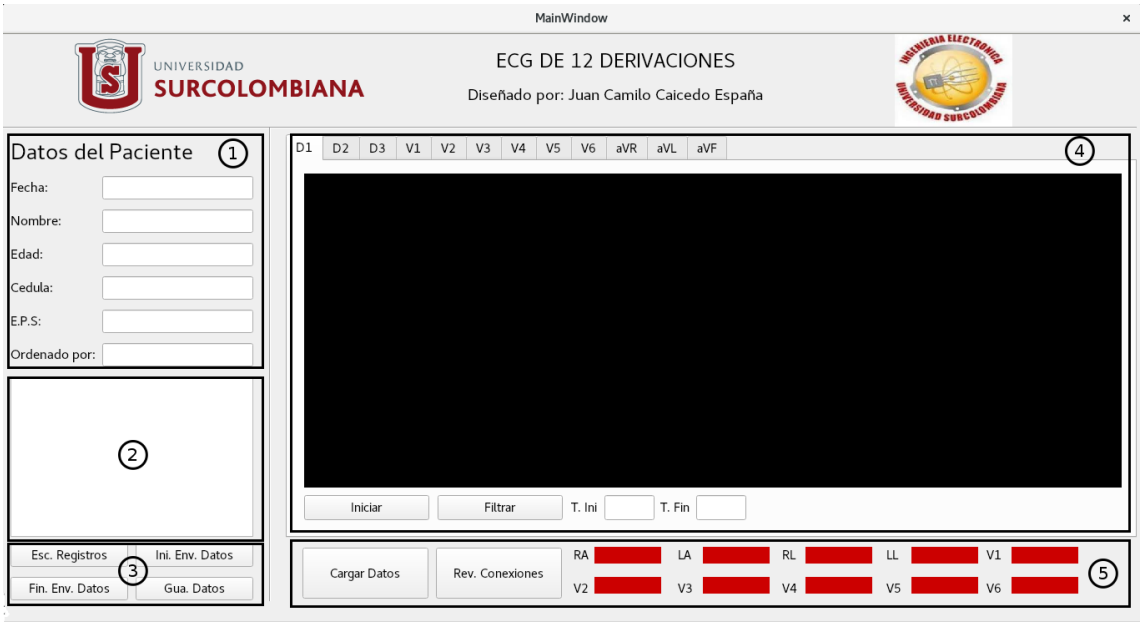
Fuente: Autor (2019)

Como se puede apreciar, en la figura anterior, existe un nivel DC indeseado. Este detalle será tratado en apartados posteriores. Una vez mostrada la gráfica de la señal cuadrada captada por cada el canal 1 y replicada en los 7 restantes, siendo esta generada por el ADS1298, se da por completada la prueba de recolección y transmisión de señal de prueba generada, siendo esta un éxito.

5.3. DISEÑO DE INTERFAZ GRÁFICA PARA PRESENTACIÓN DE DATOS

Ya terminado el proceso de diseño de los algoritmos necesarios para el correcto funcionamiento y control del proyecto, se decide diseñar una interfaz gráfica que cumpla, con estándares de facil de manejo, y rápido entendimiento de funciones presentadas en la misma. En la siguiente figura se mostrará la interfaz gráfica lograda, y se explicará cada una de las funciones que la componen:

Figura 49. Interfaz Gráfica Diseñada



Fuente: Autor (2019)

5.3.1. Panel 1: Datos del paciente

En este panel se encuentran las cajas de texto dedicadas para escribir los datos correspondientes del paciente a saber: Nombre del paciente, Edad, Cédula, E.P.S, y dos campos dedicados para la fecha del examen, y el nombre del médico que ordena el examen. Estos datos serán guardados cuando el practicante del examen así lo desee, este proceso de guardado se describe en apartados posteriores.

5.3.2. Panel 2: Caja de notificaciones

En este panel, aparecerán todos los mensajes pertinentes para el usuario del programa, puesto que cada función presente en el panel 3 esta diseñada para mostrar un mensaje para que el usuario este completamente informado del estado del programa.

5.3.3. Panel 3: Botones

En este panel encuentran los botones que permiten tener control sobre el ADS1298, y el guardado de datos, a continuación se explicarán cada uno.

• Botón Esc.Registros: Es el botón que ejecuta la función hilo_escReg, esta función desciende de la clase QThread debido a que esta función debe ejecutarse en un hilo secundario puesto que el hilo principal del programa es ocupado por la ejecución de la interfaz gráfica. Es la función encargada de escribir los registros del integrado ADS1298 para la correcta recolección de muestras. La configuración de registros se presenta de la siguiente manera, el registro 1 (CONF1) es configurado de la misma manera que en las pruebas anteriores, es decir el valor a escribir en el es 0x86. El registro 2 (CONF2) es configurado de la misma manera que en las pruebas anteriores, es decir el valor a escribir en el es 0x00. El registro 3 (CONF3) se escriben en los primeros 4 bits, los mismos usados en las pruebas (1100), pero los siguientes 3 bits se escriben en (111), dando a entender al integrado que el primer 1 debe activar la señal de referencia para el amplificador de pierna derecha (RLDREF),y este debe ser generado internamente, dando como resultado la división entre 2 de la resta entre AVDD y AVSS. El siguiente 1 indica al integrado que debe encender el Buffer de referencia para el amplificador de pierna derecha (RLD BUFFER), el siguiente 1, indicará al integrado que active la detección de conexión del pin para el amplificador de pierna derecha (RLD SENSE), el último bit se escribe en 1 puesto que no interesa el estado en que se escriba debido a que este bit posee la característica de sólo lectura, e indica si el electrodo de pierna derecha se encuentra conectado o no. Por lo cual este registro se escribe con el valor final de 0xCE.

El siguiente registro a configurar es el registro LOFF, el cual es el encargado de manejar los parámetros que se tendrán en cuenta para realizar la detección de desconexión de los electrodos del paciente. Los bits pertenecientes a este registro se muestran en la siguiente figura:

Figura 50. ADS1298: Registro de LOFF

7	6	5	4	3	2	1	0
COMP_TH2[2:0]			VLEAD_OFF_EN	ILEAD_OFF[1:0]		FLEAD_OFF[1:0]	
R/W-0h			R/W-0h	R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 67

Los primeros 3 bits (7,6,5) se escriben en 0, haciendo que el integrado tenga un comparador positivo del 95 % y un comparador negativo del 5 %. El bit 4 se escribe en 0, haciendo que

el integrado haga la detección por el método de fuente de corriente. Los bits 3 y 2, son escritos con 1, haciendo que la magnitud de la fuente de corriente sea de 24 nA. Por último los bits 1 y 0 son escritos en 1, indicando la frecuencia que tendrá la fuente de corriente, esta se escoge en frecuencia 0 o DC. Por lo tanto el valor a escribir en este registro es 0x0F

Los registros de los canales (CHnSET) son todos escritos con el valor 0x10 indicando que tendrán un funcionamiento normal, y que el factor de amplificación de los amplificadores programables sea de 12, esto con el fin de reducir el ruido generado internamente por el ADS1298. Esta elección se hace teniendo en cuenta la siguiente la figura, que puede ser encontrada en la hoja de datos del mismo integrado:

Figura 51. ADS1298: Tabla de Ruido intrínseco

Input-Referred Noise μV_{RMS} (μV_{PP}) in Low-Power Mode 3-V Analog Supply and 2.4-V Reference(1)

DR BITS OF CONFIG1 REGISTER	OUTPUT DATA RATE (SPS)	–3-dB BANDWIDTH (Hz)	PGA GAIN = 1	PGA GAIN = 2	PGA GAIN = 3	PGA GAIN = 4	PGA GAIN = 6	PGA GAIN = 8	PGA GAIN = 12
000	16000	4193	333 (3481)	166 (1836)	111 (1168)	84 (834)	56 (576)	42 (450)	28 (284)
001	8000	2096	56 (554)	28 (272)	19 (177)	14.3 (133)	9.7 (85)	7.4 (64)	5 (42.4)
010	4000	1048	12.5 (99)	6.5 (51)	4.5 (35)	3.4 (25.9)	2.4 (18.8)	2 (14.5)	1.5 (11.3)
011	2000	524	6.1 (41.8)	3.2 (22.2)	2.3 (15.9)	1.8 (12.1)	1.4 (9.3)	1.2 (7.8)	1 (6.7)
100	1000	262	4.1 (26.3)	2.2 (14.6)	1.6 (9.9)	1.3 (8.1)	1 (6.2)	0.8 (5.4)	0.7 (4.7)
101	500	131	3 (17.9)	1.6 (9.8)	1.1 (6.8)	0.9 (5.7)	0.7 (4.2)	0.6 (3.6)	0.5 (3.4)
110	250	65	2.1 (11.9)	1.1 (6.3)	0.8 (4.6)	0.7 (4)	0.5 (3)	0.5 (2.6)	0.4 (2.4)

Fuente: DataSheet ADS1298, pag 22

El siguiente registro a configurar es el registro RLD_SENSP, que es el encargado de conducir las señales de las extremidades conectadas a las entradas positivas del ADS1298, para así poder generar la señal de retroalimentación RLD. Los bits pertenecientes son los que se muestran en la siguiente figura:

Figura 52. ADS1298: Registro RLD SENSEP

7	6	5	4	3	2	1	0
RLD8P	RLD7P	RLD6P	RLD5P	RLD4P	RLD3P	RLD2P	RLD1P
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 72

Como las señales consideradas son solamente la del pie izquierdo (LL) conectado a la entrada positiva 1, la del brazo izquierdo (LA), el registro se configura con el valor 0x03.

El siguiente registro a configurar es el registro RLD_SENSN, que es el encargado de conducir las señales de las extremidades conectadas a las entradas negativas del ADS1298, para así poder generar la señal de retroalimentación RLD.

Los bits pertenecientes son los que se muestran en la siguiente figura:

Figura 53. ADS1298: Registro RLD SENSEN

7	6	5	4	3	2	1	0
RLD8N	RLD7N	RLD6N	RLD5N	RLD4N	RLD3N	RLD2N	RLD1N
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 73

Como las señales conectadas son solamente la del brazo derecho (RA), el registro se configura con el valor 0x01.

El siguiente registro a configurar es el registro LOFF_SENSP, que es el encargado de activar la detección de desconexión en los canales positivos del ADS1298. Los bits pertenecientes a este registro se muestran la siguiente figura:

Figura 54. ADS1298: Registro LOFF SENSEP

7	6	5	4	3	2	1	0
LOFF8P	LOFF7P	LOFF6P	LOFF5P	LOFF4P	LOFF3P	LOFF2P	LOFF1P
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 74

Como todas las entradas positivas del ADS1298 están conectadas al paciente, se escribe este registro con el valor 0xFF, para activar la detección de desconexión en cada una de ellas.

El siguiente registro a configurar es el registro LOFF_SENSN, que es el encargado de activar la detección de desconexión en los canales negativos del ADS1298. Los bits pertenecientes a este registro se muestran la siguiente figura:

Figura 55. ADS1298: Registro LOFF SENSEN

7	6	5	4	3	2	1	0
LOFF8N	LOFF7N	LOFF6N	LOFF5N	LOFF4N	LOFF3N	LOFF2N	LOFF1N
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 75

Como todas las entradas negativas 1 y 2 del ADS1298 están conectadas al paciente, se escribe este registro con el valor 0x01, para activar la detección de desconexión solamente en las ya mencionadas entradas.

El siguiente registro a configurar es el registro CONF4, que es el encargado de controlar las funciones del circuito de respiración, la conversión por el método Single-Shot, explicado anteriormente, la conexión del bloque WCT para generación de la señal RLD, y por último

activar los comparadores de detección de desconexión de electrodos. Los bits pertenecientes a este registro se muestran en la siguiente figura:

Figura 56. ADS1298: Registro de configuración 4 (CONF4)

7	6	5	4	3	2	1	0
RESP_FREQ[2:0]			0	SINGLE_SHOT	WCT_TO_RLD	$\overline{\text{PD\_LOFF\_COMP}}$	0
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 81

El único bit que interesa de este registro es el bit 1, el que activa los comparadores de desconexión, para ello se escribe este registro con el valor 0x02.

El siguiente registro a configurar es el registro WCT1, encargado de las funciones para generar las derivaciones aumentadas de las señales de las extremidades del paciente, y la activación del amplificador A del bloque WCT junto con el direccionamiento de la señal a este amplificador. La siguiente figura muestra los bits pertenecientes a este registro:

Figura 57. ADS1298: Registro WCT 1

7	6	5	4	3	2	1	0
aVF_CH6	aVL_CH5	aVR_CH7	aVR_CH4	$\overline{\text{PD\_WCTA}}$	WCTA[2:0]		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 82

Como las derivaciones aumentadas serán calculadas digitalmente, se ignoran los bits 7,6,5 y 4. Se activa el amplificador A del bloque WCT, colocando 1 en el bit 3, se conduce la señal del brazo derecho a este amplificador, la cual es ingresada por la entrada negativa 1, por lo cual este registro se escribe con el valor de 0x09.

El último registro a configurar es el registro WCT2, encargado del funcionamiento de los amplificadores B y C del bloque WCT.

Los bits perteneciente a este registro se muestran en la siguiente figura:

Figura 58. ADS1298: Registro WCT 2

7	6	5	4	3	2	1	0
$\overline{\text{PD\_WCTC}}$	$\overline{\text{PD\_WCTB}}$	WCTB[2:0]			WCTC[2:0]		
R/W-0h	R/W-0h	R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Fuente: DataSheet ADS1298, pag 83

Los dos primeros bits (7,6) son puestos en 1 para activar los amplificadores B y C del bloque WCT. Los bits 5,4,3 son los encargados de conducir la señal del brazo izquierdo al amplificador B. Esta es ingresada por la entrada positiva 2, por lo cual estos bits se escriben con el valor 011.Los bits 2,1,0 son los encargados de conducir la señal del pie izquierdo al amplificador C. Esta es ingresada por la entrada positiva 1, por lo cual estos bits se escriben con el valor 000. Por lo tanto este registro queda escrito con el valor de 0xD0.

Las configuraciones anteriormente mencionadas son agregadas al archivo ADS_esc98.py. Este archivo se encuentra en la carpeta raíz de la placa de desarrollo Raspberry Pi Zero W. Este algoritmo es invocado por la interfaz gráfica a través de la función hilo_escReg, que desciende de la clase QThread, para ejecutarse en un hilo aparte del hilo principal. Esta función llama una sesión por medio del protocolo SSH para mandar el comando "sudo python3 ADS_esc98.py", directamente en la raspberry. Una vez ejecutado cierra la sesión SSH, para liberar los recursos y quedar a la espera de mas ordenes.

• Botón Ini.Env.Datos: Este botón ejecuta la función hilo_iniEnvDa, desde la interfaz gráfica que desciende de la clase QThread, para ejecutarse en un hilo aparte del hilo principal. Ésta función llama una sesión por medio del protocolo SSH para mandar el comando "sudo python3 ADS_cap98.py", directamente a la raspberry. Una vez ejecutado cierra la sesión SSH, para liberar los recursos y quedar a la espera de más ordenes.

• Botón Fin.Env.Datos: Este botón ejecuta la función hilo_finEnvDa, desde la interfaz gráfica que desciende de la clase QThread, para ejecutarse en un hilo aparte del hilo principal.Esta función llama una sesión por medio del protocolo SSH para mandar el comando "sudo python3 ADS_rei.py" y el comando "sudo killall python3", directamente a la raspberry. Este último con el fin de parar cualquier ejecución del interprete del lenguaje, para asegurar completamente la cancelación del envío de tramas. Adicionalmente envía una trama codificada al servidor que sirve como señal de parada para este, que provoca que este termine su proceso y quede atento al siguiente examen que se pueda presentar. Una vez ejecutado cierra la sesión SSH, para liberar los recursos y quedar a la espera de más ordenes.

• Botón Gua. Datos: Este botón ejecuta la función GuaDa, desde la interfaz gráfica, que se encarga de captar el contenido de las cajas de texto en la que se encuentra la información del paciente. Esta información es guardada en un objeto tipo lista. Adicionalmente a este objeto, también se crea una lista con todas las tramas que representan el examen del paciente. Ambos objetos son guardados en un archivo binario con extension .dat con la ayuda del modulo pickle para reducir su tamaño. Este archivo resultante, tiene por nombre, la unión de la fecha, con la cédula del paciente, para hacer mas fácil la identificación del archivo

5.3.4. Panel 4: Pestañas de graficación

Este panel se encuentra compuesto por 12 pestañas de graficación, una para cada una de las derivación de un ECG estándar. En cada pestaña se encuentran dos botones y dos cajas

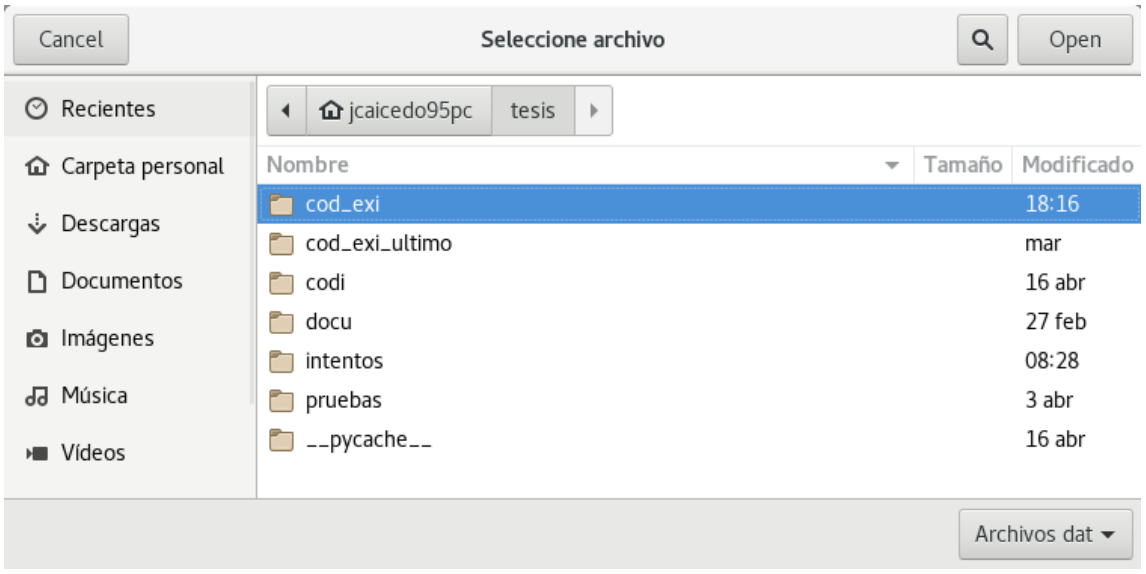
de texto. El primer botón, que tiene de nombre Iniciar, ejecuta el algoritmo de graficación hilo_graf de cada pestaña para realizar la correcta graficación de las tramas recolectadas. La función que se ejecuta desciende de la librería QThread, lo cual le permite ejecutarse en un hilo aparte del hilo principal, dando la ventaja, de que de ser necesario, se creen 12 hilos para graficar, uno por cada derivación del ECG estándar. Este botón sólo debe ser presionado después de que se haya presionado el botón Ini.Env.Datos, pues este es el encargado de enviar las tramas. El botón de filtrar, implementa un algoritmo de filtrado para las señales recolectadas. Estos son filtros digitales, el primero de ellos es un filtro de tipo NOTCH, de frecuencia de corte 60 Hz, y un factor de calidad 0.6, este cumple con el objetivo de eliminar el componente de la red eléctrica, que pueda ser introducido en la señal del ECG. El segundo filtro implementado es un filtro pasa banda Butterworth de orden 4, con frecuencias de corte 0.05 hz y 100 hz, esto con el fin de aislar el ancho de banda de interés de la señal ECG deseada. Los valores escogidos para este filtro están en marcados en la teoría propuesta en el marco teórico del presente proyecto, mas específicamente, en la figura 1. Para lograr el buen funcionamiento de este botón es necesario ingresar en las cajas de texto el tiempo inicio y final en segundos, de la ventana que desee ver el usuario. Al oprimir el botón este abre una ventana aparte, en la cual muestra la señal filtrada en el espacio de tiempo que el usuario escogió.

5.3.5. Panel 5: Carga de datos y revision de conexiones

Este panel se compone de dos secciones a saber:

• Botón Cargar Datos: al oprimir este botón, se abre una ventana de búsqueda, que permite ubicar un archivo .dat que contenga los datos guardados de un examen realizado con anterioridad, con el fin de ser analizado nuevamente. Este carga las tramas a la base de datos, y una vez notificado en la caja de notificaciones, se puede oprimir el botón Iniciar de cualquier pestaña para dar inicio a la graficación de la señal deseada.

Figura 59. Interfaz Grafica: Ventana de busqueda Archivo .dat



Fuente: DataSheet ADS1298, pag 82

• Boton Rev.Conexiones: Este botón ejecuta la función conec, desde la interfaz gráfica que desciende de la clase QThread, para ejecutarse en un hilo aparte del hilo principal. Ésta función llama una sesión por medio del protocolo SSH para mandar el comando "sudo python3 ADS_status.py", directamente a la raspberry. El algoritmo invocado hace la lectura de los registros LOFF_STATP y LOFF_STATN, los cuales muestran un 0 si el canal revisado está conectado, y 1 si no lo está. Esto se traduce en el color verde para indicar que el electrodo deseado esta conectado, y rojo cuando existe una desconexión del electrodo.

6. RESULTADOS

Para comprobar el funcionamiento real del dispositivo se realizaron dos pruebas, la primera de ellas fue la captación de señales por un dispositivo certificado de simulación. La segunda fue la recolección de señales de un sujeto de pruebas real. Adelante se describirán cada una de estas pruebas:

6.1. RECOLECCION DE SEÑAL ECG GENERADA POR SIMULADOR

6.1.1. Montaje de prueba con simulador

Esta prueba consistió en realizar la recolección de las señales ECG generadas por un simulador certificado de calibración de equipos médicos, esto con el fin de determinar la capacidad del diseño presentado en este proyecto, de poder captar de manera correcta, las señales que sean ingresadas a través de sus canales diferenciales. Para tal fin se hace uso del simulador Patient Simulator 217A de la empresa DYNATECH NEVADA INC, que permite generar las señales ECG de cada una de las 12 derivaciones estándar con los valores mencionados en la figura 1 del marco teórico del presente documento, y a una frecuencia de pulsaciones escogida por el usuario, para este caso 80 pulsaciones, lo cual es un estandar normal de un adulto promedio. La siguiente figura se muesran las principales características de dicho aparato.

Figura 60. Caracteristicas: Dynatech Nevada INC Patient Simulator

217A-multiparameter-patient-simulator	
ECG General	
Lead Configuration:	12-lead ECG with nine independent outputs for each signal lead referenced to RL
Output Update Rate:	2 ms
Output Impedance:	1000 \u03a9 to RL
1 V Output:	Lead II at 1 V peak
Amplitude Accuracy:	5 % 2 Hz square wave at 1 mV p-p (Lead II)
Normal Sinus Rhythm	
Rates:	30, 60, 80, 120, 160, 200, 240, and 300 BPM
Amplitudes:	0.5, 1.0, 1.5, 2.0 mV (Lead II)
ST Segment:	0.05, 0.10, 0.15, 0.20, 0.50, and 0.80 mV (positive and negative polarity)
Axis Deviation:	Normal, horizontal, and vertical
Pediatric ECG:	40 ms R-Wave width
ECG Performance	
Square Wave:	2 Hz at 1 mV bipolar
Pulse:	4 s at 1 mV
Sine Waves:	10, 40, 50, 60, and 100 Hz at 1 mV
Triangle Wave:	2 Hz and 3 mV

Fuente: `https://www.flukebiomedical.com/specs/1370?width=80%25&height=100%25`

Figura 61. Dynatech Nevada INC Patient Simulator

Fuente: Autor (2019)

Para llevar acabo esta prueba, se utilizo el cable Schiller 2.400 071, que ofrece las prestaciones necesarias, pues cumple con los estándares de blindado y protección que requiere un examen de este tipo. En la siguiente figura se muestran sus principales características:

Figura 62. Características Cable Schiller 2.400 071

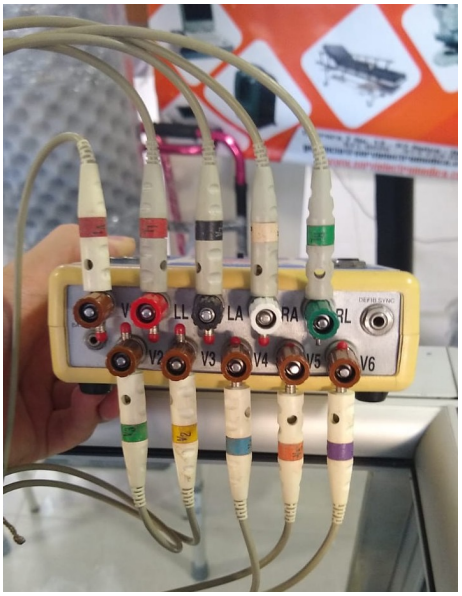
Technical Specifications:

Category	EKG
Certifications	FDA, CE, ISO10993-1, 5, 10:2003E, TUV, RoHS Compliant
Connector Distal	DB-15 Connector with Screws
Connector Proximal	Banana
Latex-free	Yes
Lead Cable Color	Gray
Lead Cable Diameter	3.00 mm
Lead Cable Length Limb	4 ft
Lead Cable Length V	2.5 ft
Lead Cable Material	TPU Jacket
Lead Color Coding	AHA
Lead Number	10
Packaging Type	Bag
Packaging Unit	1
Patient Size	Adult/Pediatric
Resistance	10K Ohms
Sterile	No
Total Cable Length	11 ft
Trunk Cable Color	Gray
Trunk Cable Length	7 ft
Trunk Cable Material	TPU Jacket
Warranty	6 months
Weight	0.9lb

Fuente:https://www.cablesandsensors.com/products/schiller-compatible-direct-connect-ekg-cable-2-400095?variant=33805598344

Las conexiones fueron realizadas conforme lo dicta el siguiente montaje:

Figura 63. Conexiones entre cable y simulador

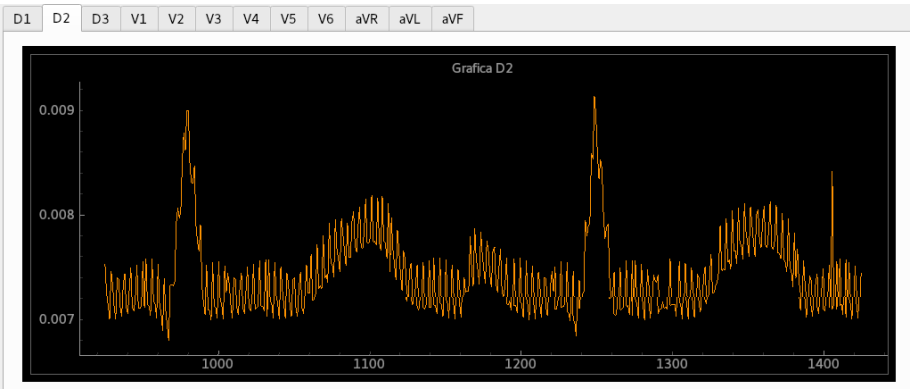


Fuente: Autor (2019)

6.1.2. Señal Recolectada

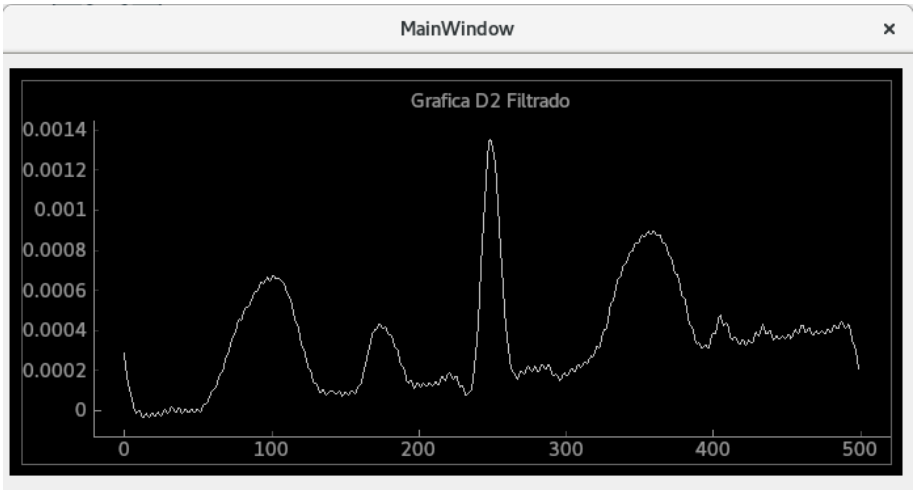
Los resultados que se obtuvieron al recolectar la señal fueron los siguientes:

Figura 64. Señal Recolectada, simulador



Fuente: Autor (2019)

Figura 65. :Señal Recolectada Filtrada, simulador



Fuente: Autor (2019)

Como se observa, la señal recolectada es clara y definida al momento de ser tomada, y mucho mas después de ser esta filtrada. Esto permite inferir que el sistema diseñado es funcional pues el objetivo de esta prueba era comprobar de manera práctica la correcta recolección de señales a través de los canales diferenciales que posee el integrado, logrando la captura se la señal con los parámetros determinados para la prueba mencionados en la parte inicial de esta este apartado, por lo cual se deconsidera que la prueba de recolección de señal ECG generada por simulador ha sido un exito.

6.2. RECOLECCIÓN DE SEÑALES DE UN SUJETO DE PRUEBAS REAL

Esta prueba consiste en extraer las señales ECG de un paciente real. Este sujeto es un varón, con 18 años, 67 kilogramos de peso, y 1 metro con 85 centímetros de altura. Este examen es llevado acabo en la I.P.S IMESS ubicada en en la calle 18A No. 5A - 20. En dicho lugar se realiza la prueba mencionada bajo dos modalidades a saber. La primera es la extracción de señales con un equipo comercial certificado y supervisado por personal idoneo del lugar. El equipo utilizado es el electrocardiografo de la marca CONTEC, modelo ECG100G capaz de proveer un examen ECG estandar de 12 derivaciones como lo solicita la prueba. En la siguiente imagen se muestra dicho elemento.

Figura 66. Electrocardiografo ECG100G

Fuente: Autor (2019)

Seguido de esto, se procede a realizar la comprobación de funcionamiento del cable Schiller 2.400 071, para tal fin se practica la prueba de continuidad en cada una de las vías que conforman el cable, que resulta en el correcto funcionamiento de cada una de las vías. Por lo tanto se procede a conectar al sujeto, al sistema diseñado como muestra la siguiente figura:

Figura 67. Montaje Sujeto real

Fuente: Autor (2019)

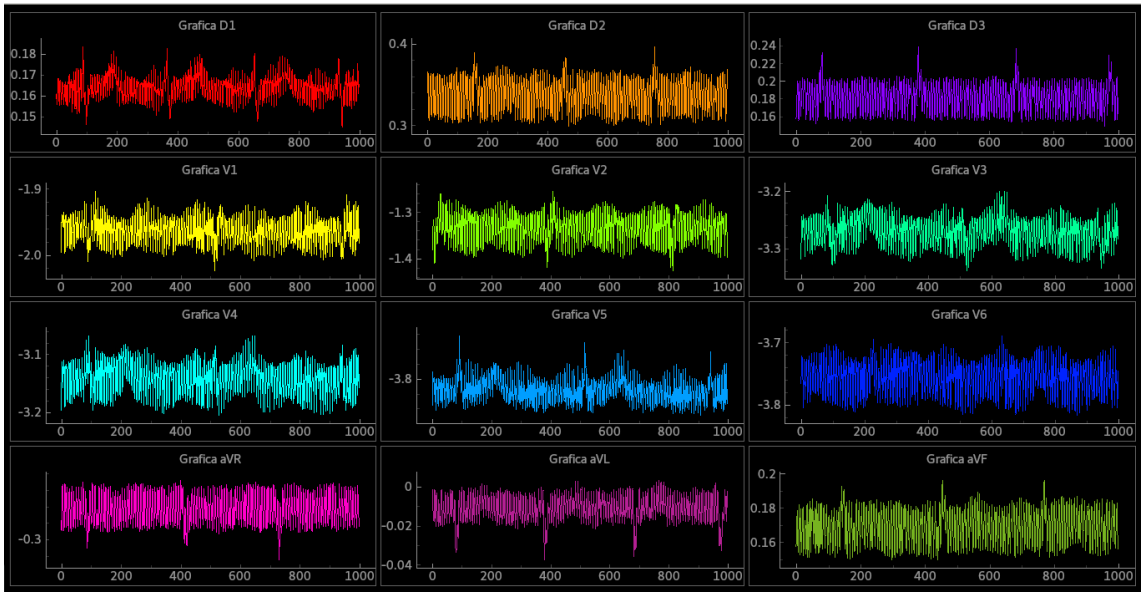
Los resultados obtenidos en este examen utilizando el dispositivo mencionado son generados en una tira larga de papel térmico, por lo cual está se escanea por partes y es presentada en la siguiente imagen:

Figura 68. Resultados obtenidos de ECG100G

Fuente: Autor (2019)

Una vez practicado este examen como base de un análisis, se procede a realizar la toma del examen con el dispositivo diseñado. En la siguiente figura se muestran los resultados obtenidos sin filtrar.

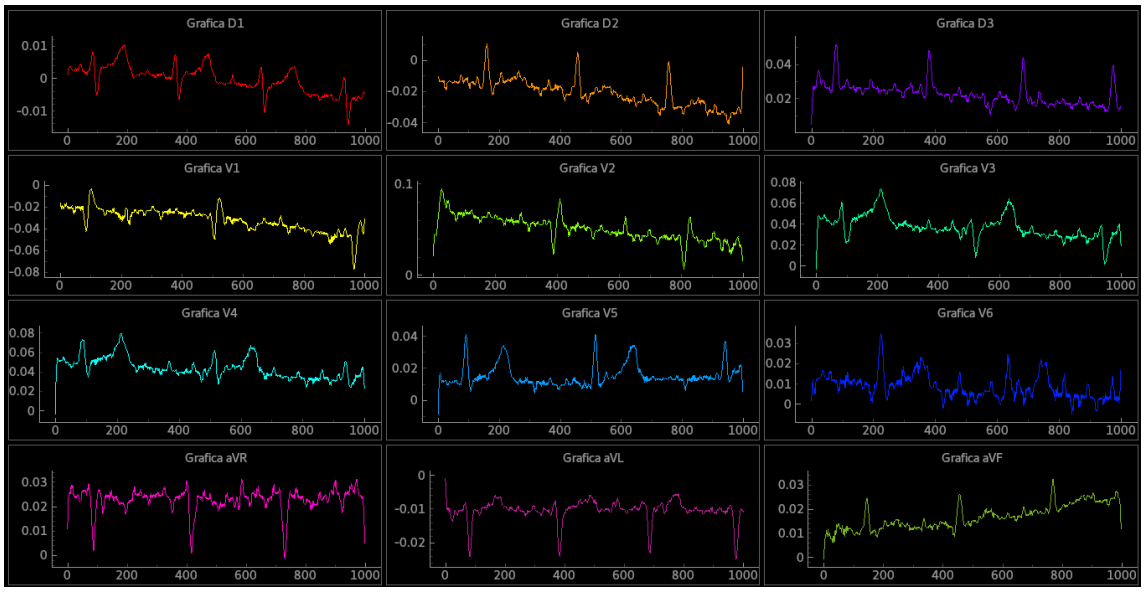
Figura 69. Resultados obtenidos dispositivo diseñado sin filtrar



Fuente: Autor (2019)

A continuación se presentan los resultados obtenidos, ya filtrados para una mejor observación:

Figura 70. Resultados obtenidos dispositivo diseñado filtrados



Fuente: Autor (2019)

En los resultados se pueden apreciar diferencias mínimas en cuanto la inclinación de las señales, esto debido, según el personal especializado del lugar, a movimientos del paciente y los electrodos, provocando así una inserción de ruido, lo cual no llega afectar de manera significativa al resultado final en cuanto a repetibilidad de la señal que es una de los criterios base para calificar como exitoso un examen de este tipo. Por lo cual se permite inferir que los resultados son congruentes con el examen tomado con el dispositivo comercial en la mayoría de los aspectos lo cual permite al personal especializado del lugar concluir que el dispositivo diseñado funciona correctamente y certificar el funcionamiento del mismo para toma de exámenes ECG.

7. CONCLUSIONES

Se realizó un correcto estudio de los dispositivos de tipo SMD existentes en el mercado, con base en las características, que permitieran la correcta adquisición de una señal bioeléctrica, estándares de filtrado, y precio. Encontrándose que la opción correcta para el desarrollo del presente proyecto es el integrado ADS1298, pues permite diseñar un sistema robusto, escalable y de precio reducido para la adquisición de señales ECG, con útiles funciones como lo son bloques dedicados a la generación de señal de manejo de pierna derecha (RLD), y señal de la terminal central de Wilson (WCT) indispensables ambas a la hora de hacer la toma de estas señales.

Se diseñó un dispositivo con el integrado mencionado, lo que permitió estudiar los efectos que se producían al manipular la señal ECG, por medio de la utilización de muestreadores Sigma-delta internos por cada canal, traduciéndose en una útil ventaja, pues al realizar el muestreo de la señal por este método, se encuentra que gran parte del ruido de alta frecuencia que puede invadir el estudio de la señal ECG es considerablemente reducido. Además, al contar con un puerto de comunicación SPI y al hacer la correcta elección de una placa de desarrollo como lo es la Raspberry Pi Zero W, se logró un producto final con comunicación más rápida entre circuitos para tener un óptimo control de las funciones de lectura, escritura y sensado que permite el ADS1298.

Se ejecutó la adquisición adecuada de las señales ECG logrando así la recolección necesaria de tramas para un examen estándar de 12 derivaciones en tiempo real. Todo esto con la facilidad que permite el algoritmo diseñado para transmitir las tramas recolectadas por la placa de desarrollo, haciendo uso del protocolo TCP/IP, dando una garantía al usuario del producto, que la información ha sido transmitida y recepcionada sin ningún tipo de perdida de datos en este proceso, mencionando además la implementación de un algoritmo de prevención de errores en el servidor que reduce aún más esta posibilidad, añadiendo la utilización de un gestor de bases de datos no relacional como lo es Redis, dando al proyecto altas posibilidades de escalamiento rápido, preciso, y adecuado, pues, al guardar las tramas en una base de datos de esta clase, no se altera el tipo de dato que se guardó, añadiendo un acceso más rápido a las mismas, y quitando gran carga al servidor, encargado de recepcionar las tramas, pues como se muestra en el proyecto, se hizo necesario la codificación de tramas para el correcto envió de las mismas, pasando así la tarea de decodificar las tramas, a cada hilo de graficación.

Se implementó una interfaz útil, sin complicaciones, que permite la inclusión de datos de interés acerca de la persona a quien se le practica el procedimiento, como son el nombre, la edad, número de identificación, además de la fecha de realización del examen y el nombre de quien ordena el procedimiento. Cuenta además con un panel de notificaciones que permite al usuario estar permanentemente informado del estado y acciones que realiza el programa al hacer cualquier acción sobre el mismo, contando con un panel de carga de datos para el análisis de exámenes anteriormente guardados, útil al realizar una inspección más detallada por parte del usuario, presentando las señales recolectadas en un panel graficador que permite visualizar adecuadamente las mismas, y de ser necesario realizar un filtrado de una ventana de tiempo, de la señal que el usuario considere importante para un análisis exhaustivo. Esta interfaz, aparte de todo permite el guardado, y carga de datos

encapsulados por la biblioteca pickle, permitiendo reducir su tamaño, y dar un acceso más rápido a la información para el lenguaje empleado python 3.7.

8. RECOMENDACIONES

El programa de interfaz de usuario ésta diseñado para seguir un estricto orden siempre a saber. Primero la escritura de registros, pues al alimentar de corriente por primera vez la placa diseñada, el ADS1298 es puesto en sus configuraciones por defecto. Luego de que los mensajes indiquen que todo se configuró correctamente, puede invocarse el botón de inicio de envío de datos, en este instante las tramas ya se están ingresando en la base de datos, y puede darse click en el botón de iniciar, de cada pestaña, antes no, puesto que cada hilo esta diseñado para esperar que existan en la base de datos al menos 200 tramas para asegurar una correcta transición en la graficación de las mismas. Para filtrar la señal se debe tener en cuenta las tramas graficadas, pues los filtros toman en consideración este parámetro, por lo cual si se intenta filtrar una trama que aún no ha sido graficada, provocará un error. Una vez el examen concluya, se deberá parar el envió de datos con el botón indicado, provocando así un correcto apagado tanto del servidor de recepción, como del programa encargado del envió de tramas. Finalmente una vez realizado este proceso se podrá guardar los datos con el botón indicado para este proceso dando por concluido el uso del programa diseñado.

9. TRABAJOS FUTUROS

El presente proyecto podrá ser utilizado como base para el diseño de un sistema de sensado de todo tipo de señales bioeléctricas necesarias para el diagnostico de enfermedades. Ademas con las tramas aquí recolectadas se puede desarrollar un algoritmo de detección de enfermedades que permita al usuario definir un tipo de tratamiento para la afección detectada, este algoritmo puede ser trabajado por ejemplo con teorías propuestas como la transformada de Wavelet.

BIBLIOGRAFÍA

[1] Varios Autores. *Manual ECG, Electrocardiografía,* AMIR, 2014.

[2] Red De Salud De Cuba. *Las derivaciones del electrocardiograma.* 2008.

[3] Culturacion. *http://culturacion.com/raspberry-pi-que-es-caracteristicas-y-precios/.* 2015.

[4] EcuRed. *Tecnología de Montaje Superficial.* 2009. URL: `https://www.ecured.cu/TecnologÃŋa_de_Montaje_Superficial`.

[5] Jian-Zhi Chen-Liang-Hung Wang-Fa-Xiang Wang-Ming-Hui Fan. *Design of ECG signal acquisition system based on ADS1291.* International Conference On Communication Problem-Solving (ICCP), 2016.

[6] Muhammad Wildan Gifari-Hasballah Zakaria-Richard Mengko. *Design of ECG Homecare:12-lead ECG acquisition using single channel ECG device developed on AD8232 analog front end.* International Conference on Electrical Engineering e Informatics (ICEEI), 2015.

[7] Br. Marttin Oliveri. *Elementos de diseño de circuitos de Amplificación del ECG.* XII Seminario de Ingeniería Biomédica,facultades medicina de ingeniería. Universidad de la república, Montevideo, 2004.

[8] Ángela Patricia Guerrero Castillo. *Diseño de un dispositivo para la adquisición de una señal ECG que incluya la reducción de artefactos oculares.* Escuela Colombiana de Ingeniería Julio Garavito, 2017.

[9] Guillermo Eduardo Vega Picón. *Diseño y construcción de un electrocardiógrafo de 12 derivaciones para el análisis de señales cardiacas.* Universidad Politécnica Salesiana, Sede Cuenca, 2012.

[10] Jesús María Rodríguez Presedo. *Adquisición de señales biológicas.* Universidad de Santiago, 2005.

[11] Luis Enrique Islas Contreras-Oscar Ortiz-Iván Alfredo Hernández Pérez. *Sistema de Adquisición y Transmisión inalámbrica de señales electrocardiográficas.* Instituto Politécnico Nacional, 2015.

[12] Marlon Arturo Pérez Rodas. *Diseño e Implementación de un electrocardiógrafo portátil y del sistema de procesamiento digital de señales eléctricas del corazón, para monitoreo y análisis médico.* Universidad de San Carlos de Guatemala, 2011.

[13] Protocolo SSH. *https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html.* 2017.

[14] Tco. Lic. Javier Sandri. *Electrocardiografía.* Tco. Lic. Javier Sandri, 2008.

[15] Carolina Pascuas B.-Mauro Fernando Vargas Suárez. *Electrocardiógrafo digital portátil.* Universidad Surcolombiana, 2003.

[16] Definición de TCP. *http://www.masadelante.com/faqs/tcp-ip.* 2019.

[17] Protocolo TCP. *https://es.ccm.net/contents/281-protocolo-tcp.* 2017.

[18] Sergio Augusto Tabares Chavarro-Jefferson Perdomo Trujillo. *Prototipo electrocardiógrafo inalámbrico para la detección de enfermedades que desencadenen la muerte súbita, con software de diagnóstico médico aproximado.* Uniersidad Surcolombiana, 2016.

[19] Protocolo UDP. *www.ionos.es/digitalguide/servidores/know-how/udp-user-datagram-protocol/.* 2019.

[20] Santiago VillaFuerte Echeverri-Harby Torres Ávila. *Sistema Portable para la Adquisición y Procesamiento de Señales ECG, con aplicabilidad en dispositivos Móviles.* Uniersidad de San Buenaventura, Cali, 2017.

ANEXOS

Código de interfaz gráfica

Se debe tener en cuenta que para realizar las gráficas V1, V2, V3, V4, V5, y V6, se debe realizar una copia de los algoritmos donde aparece D1 y ser agregado como dicta la lógica del código.

```
from PyQt5.QtWidgets import QFileDialog
from pickle import dumps,loads,dump,load
from ven_graFilD1_ui import Ui_filD1
from ven_graFilD2_ui import Ui_filD2
from ven_graFilD3_ui import Ui_filD3
from ven_graFilV1_ui import Ui_filV1
from ven_graFilV2_ui import Ui_filV2
from ven_graFilV3_ui import Ui_filV3
from ven_graFilV4_ui import Ui_filV4
from ven_graFilV5_ui import Ui_filV5
from ven_graFilV6_ui import Ui_filV6
from ven_graFilVR_ui import Ui_filVR
from ven_graFilVL_ui import Ui_filVL
from ven_graFilVF_ui import Ui_filVF
from threading import Thread
from interfaz_ui import *
from scipy import signal
import pyqtgraph as pg
import paramiko
import socket
import redis
import os

fs = 500
fc = 60
Q = 0.6
bbut,abut = signal.butter(4,[0.05,100],'bandpass',fs=fs)
bnot,anot = signal.iirnotch(fc,Q,fs)
iniD2 = 3
finD2 = 6
finD1 = 9
finV2 = 12
finV3 = 15
finV4 = 18
finV5 = 21
finV6 = 24
finV1 = 27
ptr1 = 0
ptr2 = 0
ptr3 = 0
ptr4 = 0
ptr5 = 0
ptr6 = 0
ptr7 = 0
ptr8 = 0
ptr9 = 0
ptr10 = 0
```

```
ptr11 = 0
ptr12 = 0
dataD2 = [0]*500
dataD3 = [0]*500
dataD1 = [0]*500
dataV2 = [0]*500
dataV3 = [0]*500
dataV4 = [0]*500
dataV5 = [0]*500
dataV6 = [0]*500
dataV1 = [0]*500
dataaVR = [0]*500
dataaVL = [0]*500
dataaVF = [0]*500

base ='paci'

comp = 16777215
mayor = 8388607
d=2**23
d1 = d - 1
d2 = d1*12
escalar =4.972920816/d1

def GuaDa(reci):
    info = []
    fecha = reci[0]
    nombre = reci[1]
    edad = reci[2]
    cedu = reci[3]
    eps = reci[4]
    ordn = reci[5]
    nomAr = reci[6]
    archivo = open(nomAr, 'wb')
    info.append(fecha)
    info.append(nombre)
    info.append(edad)
    info.append(cedu)
    info.append(eps)
    info.append(ordn)
    r = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)
    daticos = r.lrange(base, 0, -1)
    info.append(daticos)
    dump(info,archivo)
    archivo.close()

class conec(QtCore.QThread):
    varconec = QtCore.pyqtSignal(int)
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.r = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)
        self.ssh_client = paramiko.SSHClient()
    def run(self):
        comando = 'echo %s | sudo python3 ADS_status.py'
        self.ssh_client.set_missing_host_key_policy
        (paramiko.AutoAddPolicy())
        self.ssh_client.connect('192.168.43.8', 22, 'pi', 'pi')
        self.entrada, self.salida, self.error
         = self.ssh_client.exec_command(comando)
```

```
            self.salida1 = self.salida.read()
            self.ssh_client.close()
            self.out = self.salida1.decode('ascii')
            self.coma1 = self.out.index(',')
            self.coma2 = self.out.index(',', self.coma1 + 1, -1)
            self.rld = int(self.out[1:self.coma1])
            self.positivos = int(self.out[self.coma1 + 2:self.coma2])
            self.negativos = int(self.out[self.coma2 + 2:-2])
            self.parte1 = bin(self.rld)[2:].zfill(8)
            self.parte2 = bin(self.positivos)[2:].zfill(8)
            self.parte3 = bin(self.negativos)[2:].zfill(8)
            print(self.parte1)

            if self.parte1[-1]==0:
                self.varconec.emit(0)
            else:
                self.varconec.emit(1)
            if self.parte2[0]==0:
                self.varconec.emit(2)
            else:
                self.varconec.emit(3)
            if self.parte2[1]==0:
                self.varconec.emit(4)
            else:
                self.varconec.emit(5)
            if self.parte2[2]==0:
                self.varconec.emit(6)
            else:
                self.varconec.emit(7)
            if self.parte2[3]==0:
                self.varconec.emit(8)
            else:
                self.varconec.emit(9)
            if self.parte2[4]==0:
                self.varconec.emit(10)
            else:
                self.varconec.emit(11)
            if self.parte2[5]==0:
                self.varconec.emit(12)
            else:
                self.varconec.emit(13)
            if self.parte2[6]==0:
                self.varconec.emit(14)
            else:
                self.varconec.emit(15)
            if self.parte2[7]==0:
                self.varconec.emit(16)
            else:
                self.varconec.emit(17)
            if self.parte3[-1]==0:
                self.varconec.emit(18)
            else:
                self.varconec.emit(19)

class hilo_serv(QtCore.QThread):
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.r = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)
```

```
    def run(self):
        db = 'paci'
        #HOST = '192.168.43.5'
        HOST = '192.168.1.11'
        PORT = 10000

        global espe
        global nuevo

        espe = bytes(0)
        dir_servi = (HOST, PORT)
        self.s.bind(dir_servi)
        self.s.listen(1)

        try:
            while True:
                self.co,self.dire = self.s.accept()
                try:
                    while True:
                        self.d_recibido = self.co.recv(62)
                        if len(self.d_recibido) == 6:
                            self.r.rpush(db, self.d_recibido)
                            break
                        else:
                            if len(self.d_recibido) == 62:
                                if self.d_recibido[0:6] == b'\x80\x03]q\x00(':
                                    self.r.rpush(db,self.d_recibido)
                                else:
                                    self.nuevo = espe + self.d_recibido
                                    self.indice = self.nuevo.find(b'e.')
                                    self.indice1 = self.indice + 2
                                    self.r.rpush(db, self.nuevo[0:self.indice1])
                                    espe = self.nuevo[self.indice1:]
                            else:
                                espe = espe + self.d_recibido
                finally:
                    self.co.close()
        finally:
            self.s.close()

class hilo_escReg(QtCore.QThread):
    dataescReg = QtCore.pyqtSignal(int)
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ssh_client = paramiko.SSHClient()

    def run(self):
        comando = 'echo␣%s␣|␣sudo␣python3␣ADS_esc9812.py'
        self.ssh_client.set_missing_host_key_policy
        (paramiko.AutoAddPolicy())
        self.ssh_client.connect('192.168.43.8', 22, 'pi', 'pi')
        self.entrada, self.salida, self.error
         = self.ssh_client.exec_command(comando)
        self.salida1 = self.salida.read()
        self.ssh_client.close()
        self.out = self.salida1.decode('ascii')
        try:
            if self.out[1:4]=='134':
```

```
            self.dataescReg.emit(1)
        else:
            self.dataescReg.emit(-1)
    finally:pass
    try:
        if self.out[6:7]=='0':
            self.dataescReg.emit(2)
        else:
            self.dataescReg.emit(-2)
    finally:pass
    try:
        if self.out[9:12]=='206':
            self.dataescReg.emit(3)
        else:
            self.dataescReg.emit(-3)
    finally:pass
    try:
        if self.out[14:16]=='15':
            self.dataescReg.emit(4)
        else:
            self.dataescReg.emit(-4)
    finally:pass
    try:
        if self.out[18:48]=='96, 96, 96, 96, 96, 96, 96, 96':
            self.dataescReg.emit(5)
        else:
            self.dataescReg.emit(-5)
    finally:pass

    try:
        if self.out[50:51]=='3':
            self.dataescReg.emit(6)
        else:
            self.dataescReg.emit(-6)
    finally:pass
    try:
        if self.out[53:54]=='1':
            self.dataescReg.emit(7)
        else:
            self.dataescReg.emit(-7)
    finally:pass
    try:
        if self.out[56:59]=='255':
            self.dataescReg.emit(8)
        else:
            self.dataescReg.emit(-8)
    finally:pass
    try:
        if self.out[61:62]=='3':
            self.dataescReg.emit(9)
        else:
            self.dataescReg.emit(-9)
    finally:pass
    try:
        if self.out[64:65]=='0':
            self.dataescReg.emit(10)
        else:
            self.dataescReg.emit(-10)
    finally:pass
```

```
            try:
                if self.out[67:68]=='2':
                    self.dataescReg.emit(11)
                else:
                    self.dataescReg.emit(-11)
            finally:pass
            try:
                if self.out[70:71]=='9':
                    self.dataescReg.emit(12)
                else:
                    self.dataescReg.emit(-12)
            finally:pass
            try:
                if self.out[73:76]=='208':
                    self.dataescReg.emit(13)
                else:
                    self.dataescReg.emit(-13)
            finally:pass

class hilo_iniEnvDa(QtCore.QThread):
    datainiEnvDa = QtCore.pyqtSignal(int)
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ssh_client = paramiko.SSHClient()
    def run(self):
        self.datainiEnvDa.emit(1)
        comando = 'echo␣%s␣|␣sudo␣python3␣ADS_cap98.py'
        self.ssh_client.set_missing_host_key_policy
        (paramiko.AutoAddPolicy())
        self.ssh_client.connect('192.168.43.8', 22, 'pi', 'pi')
        self.entrada, self.salida, self.error
        = self.ssh_client.exec_command(comando)
        self.ssh_client.close()
        QtCore.QThread.msleep(10)
        self.datainiEnvDa.emit(2)

class hilo_finEnvDa(QtCore.QThread):
    datafinEnvDa = QtCore.pyqtSignal(int)
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ssh_client = paramiko.SSHClient()
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    def run(self):
        port = 10000
        comando = 'echo␣%s␣|␣sudo␣python3␣ADS_rei.py'
        comando1 = 'echo␣%s␣|␣sudo␣killall␣python3'
        self.ssh_client.set_missing_host_key_policy
        (paramiko.AutoAddPolicy())
        self.ssh_client.connect('192.168.43.8', 22, 'pi', 'pi')
        self.entrada, self.salida, self.error
        = self.ssh_client.exec_command(comando)
        self.datafinEnvDa.emit(1)
        QtCore.QThread.msleep(10)
        self.entrada1, self.salida1, self.error1
        = self.ssh_client.exec_command(comando1)
        self.datafinEnvDa.emit(2)
        QtCore.QThread.msleep(10)
        self.ssh_client.close()
```

```
            self.server_address = ('192.168.43.5',port)
            self.sock.connect(self.server_address)
            self.sock.sendall(dumps([]))
            self.datafinEnvDa.emit(3)
            self.sock.close()
            self.datafinEnvDa.emit(4)

class hilo_grafD1(QtCore.QThread):
    datagrafD1 = QtCore.pyqtSignal(list)
    mueD1 = []
    gra = 0
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.r = redis.StrictRedis(host='127.0.0.1', port=6379, db=0)
        self.timer = pg.QtCore.QTimer()
        self.timer.timeout.connect(self.run)
        self.timer.start(20)
    def run(self):
        while True:
            if self.r.llen(base) == -1:
                pass
            else:
                break
        while True:
            if self.r.llen(base) > 200:
                break
            else:
                pass
        global ptr1, gra, mueD1
        dataD1[:-1] = dataD1[1:]
        self.db = self.r.lrange(base, ptr1, ptr1)
        if (self.db == []):
            self.mueD1.append(0)
            self.datagrafD1.emit(self.mueD1)
        else:
            self.dato = self.db[0]
            if len(self.dato) == 6:
                self.mueD1.append(0)
                self.datagrafD1.emit(self.mueD1)
            else:
                self.deco = loads(self.dato)
                self.D1 = self.deco[finD2:finD1]
                self.parte1 = bin(self.D1[0])[2:].zfill(8)
                self.parte2 = bin(self.D1[1])[2:].zfill(8)
                self.parte3 = bin(self.D1[2])[2:].zfill(8)
                self.union = self.parte1 + self.parte2 + self.parte3
                self.numero = int(self.union, 2)
                if self.numero > mayor:
                    self.numero1 = self.numero ^ comp
                    self.numero2 = self.numero1 + 1
                    self.gra = -escalar * self.numero2 + 0.002083333333
                    self.mueD1.append(self.gra)
                    self.datagrafD1.emit(self.mueD1)
                else:
                    self.gra = escalar * self.numero + 0.002083333333
                    self.mueD1.append(self.gra)
                    self.datagrafD1.emit(self.mueD1)
                ptr1 += 1
```

```
class hilo_grafD3(QtCore.QThread):
    datagrafD3 = QtCore.pyqtSignal(list)
    mueD3 = []
    graD3 = 0
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ptr3 = 0
        self.variD1 = hilo_grafD1.mueD1
        self.variD2 = hilo_grafD2.mueD2
        self.timer = pg.QtCore.QTimer()
        self.timer.timeout.connect(self.run)
        self.timer.start(20)
    def run(self):
        global ptr3, graD3, mueD3
        dataD3[:-1] = dataD3[1:]
        self.graD3 = self.variD2[ptr3] - self.variD1[ptr3]
        self.mueD3.append(self.graD3)
        self.datagrafD3.emit(self.mueD3)
        ptr3 += 1

class hilo_grafaVR(QtCore.QThread):
    datagrafaVR = QtCore.pyqtSignal(list)
    mueaVR = []
    graaVR = 0
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ptr10 = 0
        self.variD3 = hilo_grafD3.mueD3
        self.variD2 = hilo_grafD2.mueD2
        self.timer = pg.QtCore.QTimer()
        self.timer.timeout.connect(self.run)
        self.timer.start(20)
    def run(self):
        global ptr10, graaVR, mueaVR
        dataaVR[:-1] = dataaVR[1:]
        self.graaVR = -(self.variD2[ptr10] + self.variD3[self.ptr10]) / 2
        self.mueaVR.append(self.graaVR)
        self.datagrafaVR.emit(self.mueaVR)
        ptr10 += 1

class hilo_grafaVL(QtCore.QThread):
    datagrafaVL = QtCore.pyqtSignal(list)
    mueaVL = []
    graaVL = 0
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ptr11 = 0
        self.variD1 = hilo_grafD1.mueD1
        self.variD3 = hilo_grafD3.mueD3
        self.timer = pg.QtCore.QTimer()
        self.timer.timeout.connect(self.run)
        self.timer.start(20)
    def run(self):
        global ptr11, graaVL, mueaVL
        dataaVL[:-1] = dataaVL[1:]
        self.graaVL = (self.variD1[ptr11] - self.variD3[ptr11]) / 2
        self.mueaVL.append(self.graaVL)
        self.datagrafaVL.emit(self.mueaVL)
        ptr11 += 1
```

```
class hilo_grafaVF(QtCore.QThread):
    datagrafaVF = QtCore.pyqtSignal(list)
    mueaVF = []
    graaVF = 0
    def __init__(self,*args,**kwargs):
        QtCore.QThread.__init__(self,*args,**kwargs)
        self.ptr12 = 0
        self.variD1 = hilo_grafD1.mueD1
        self.variD3 = hilo_grafD3.mueD3
        self.timer = pg.QtCore.QTimer()
        self.timer.timeout.connect(self.run)
        self.timer.start(20)
    def run(self):
        global ptr12, graaVF, mueaVF
        dataaVF[:-1] = dataaVF[1:]
        self.graaVF = (self.variD1[ptr12] + self.variD3[ptr12]) / 2
        self.mueaVF.append(self.graaVF)
        self.datagrafaVF.emit(self.mueaVF)
        ptr12 += 1

class MainWindow(QtWidgets.QMainWindow,Ui_MainWindow):
    def __init__(self,*args,**kwargs):
        QtWidgets.QMainWindow.__init__(self,*args,**kwargs)
        self.setupUi(self)
        self.pButton_GuaDa.clicked.connect(self.GuaDa)
        self.pButton_escReg.clicked.connect(self.escReg)
        self.pButton_iniEnvDa.clicked.connect(self.iniEnvDa)
        self.pButton_finEnvDa.clicked.connect(self.finEnvDa)
        self.pButton_IniD1.clicked.connect(self.grafD1)
        self.pButton_FilD1.clicked.connect(self.fil_D1)
        self.pButton_IniD2.clicked.connect(self.grafD2)
        self.pButton_FilD2.clicked.connect(self.fil_D2)
        self.pButton_IniD3.clicked.connect(self.grafD3)
        self.pButton_FilD3.clicked.connect(self.fil_D3)
        self.pButton_IniV1.clicked.connect(self.grafV1)
        self.pButton_FilV1.clicked.connect(self.fil_V1)
        self.pButton_IniV2.clicked.connect(self.grafV2)
        self.pButton_FilV2.clicked.connect(self.fil_V2)
        self.pButton_IniV3.clicked.connect(self.grafV3)
        self.pButton_FilV3.clicked.connect(self.fil_V3)
        self.pButton_IniV4.clicked.connect(self.grafV4)
        self.pButton_FilV4.clicked.connect(self.fil_V4)
        self.pButton_finV5.clicked.connect(self.grafV5)
        self.pButton_FilV5.clicked.connect(self.fil_V5)
        self.pButton_IniV6.clicked.connect(self.grafV6)
        self.pButton_FilV6.clicked.connect(self.fil_V6)
        self.pButton_IniaVR.clicked.connect(self.grafaVR)
        self.pButton_FilVR.clicked.connect(self.fil_VR)
        self.pButton_IniaVL.clicked.connect(self.grafaVL)
        self.pButton_FilVL.clicked.connect(self.fil_VL)
        self.pButton_IniaVF.clicked.connect(self.grafaVF)
        self.pButton_FilVF.clicked.connect(self.fil_VF)
        self.pButton_CarDa.clicked.connect(self.CarDa)
        self.pButton_Cone.clicked.connect(self.actual)
        self.r = redis.StrictRedis(host='127.0.0.1',port=6379,db=0)
        self.r.delete('paci')
        self.n = 0
        self.ng = 0
```

```
        self.nD2 = 0
        self.nD3 = 0
        self.nV1 = 0
        self.nV2 = 0
        self.nV3 = 0
        self.nV4 = 0
        self.nV5 = 0
        self.nV6 = 0
        self.naVR = 0
        self.naVL = 0
        self.naVF = 0
        self.datosD1 = hilo_grafD1.mueD1
        self.datosD2 = hilo_grafD2.mueD2
        self.datosD3 = hilo_grafD3.mueD3
        self.datosV1 = hilo_grafV1.mueV1
        self.datosV2 = hilo_grafV2.mueV2
        self.datosV3 = hilo_grafV3.mueV3
        self.datosV4 = hilo_grafV4.mueV4
        self.datosV5 = hilo_grafV5.mueV5
        self.datosV6 = hilo_grafV6.mueV6
        self.datosVR = hilo_grafaVR.mueaVR
        self.datosVL = hilo_grafaVL.mueaVL
        self.datosVF = hilo_grafaVF.mueaVF
        thread_serv = hilo_serv(self)
        thread_serv.start()

    def GuaDa(self):
        self.datosArc = []
        self.listW_texRe.addItem('␣')
        self.listW_texRe.addItem('----------')
        self.fechaPa = self.lineE_fechaPa.text()
        self.datosArc.append(self.fechaPa)
        self.listW_texRe.addItem('Fecha␣Agregada␣correctamente')
        self.nombrePa = self.lineE_nombrePa.text()
        self.datosArc.append(self.nombrePa)
        self.listW_texRe.addItem('Nombre␣agregado␣correctamente')
        self.edadPa = self.lineE_edadPa.text()
        self.datosArc.append(self.edadPa)
        self.listW_texRe.addItem('Edad␣se␣agrego␣correctamente')
        self.ceduPa = self.lineE_ceduPa.text()
        self.datosArc.append(self.ceduPa)
        self.listW_texRe.addItem('Cedula␣agregada␣correctamente')
        self.epsPa = self.lineE_epsPa.text()
        self.datosArc.append(self.epsPa)
        self.listW_texRe.addItem('E.P.S␣agregada␣correctamente')
        self.ordPa = self.lineE_ordPa.text()
        self.datosArc.append(self.ordPa)
        self.listW_texRe.addItem('Orden␣Medica␣agregada␣correctamente')
        self.nombreAr = self.fechaPa + 'C.C' + self.ceduPa + '.dat'
        self.datosArc.append(self.nombreAr)
        hilo_guar =Thread(target=GuaDa,args=(self.datosArc,),daemon=True)
        hilo_guar.start()

    def CarDa(self):
        self.listW_texRe.addItem('␣')
        self.listW_texRe.addItem('----------')
        archivodat, extension = QFileDialog.getOpenFileName
        (self, 'Seleccione␣archivo', os.getcwd(),
        "Archivos␣dat(*.dat)", options=QFileDialog.Options())
```

```
        archivo = open(archivodat,'rb')
        datos = load(archivo)
        self.lineE_fechaPa.setText(datos[0])
        self.lineE_nombrePa.setText(datos[1])
        self.lineE_edadPa.setText(datos[2])
        self.lineE_ceduPa.setText(datos[3])
        self.lineE_epsPa.setText(datos[4])
        self.lineE_ordPa.setText(datos[5])
        self.infor = datos[-1]
        self.con = 0
        while self.con< len(self.infor):
            self.r.rpush('paci',self.infor[self.con])
            self.con = self.con + 1
        self.listW_texRe.addItem('Datos Cargados Correctamente')
        self.listW_texRe.addItem('Puede Empezar a Graficar')

    def update_actual(self,dato):
        if dato == 0:
            self.label_conRL.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 1:
            self.label_conRL.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 2:
            self.label_conV1.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 3:
            self.label_conV1.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 4:
            self.label_conV6.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 5:
            self.label_conV6.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 6:
            self.label_conV5.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 7:
            self.label_conV5.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 8:
            self.label_conV4.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 9:
            self.label_conV4.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 10:
            self.label_conV3.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 11:
            self.label_conV3.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 12:
            self.label_conV2.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 13:
            self.label_conV2.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
```

```
        if dato == 14:
            self.label_conLA.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 15:
            self.label_conLA.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 16:
            self.label_conLL.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 17:
            self.label_conLL.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
        if dato == 18:
            self.label_conRA.setStyleSheet
            ("background-color: rgb(115, 210, 22);")
        if dato == 19:
            self.label_conRA.setStyleSheet
            ("background-color: rgb(204, 0, 0);")
    def actual(self):
        thread_actual = conec(self)
        thread_actual.varconec.connect(self.update_actual)
        thread_actual.start()

    def update_escReg(self,datoescReg):
        if datoescReg == 1:
            self.listW_texRe.addItem
            ('Correcta configuracion para 500 muestras')
        if datoescReg == -1:
            self.listW_texRe.addItem
            ('Incorrecta configuracion para 500 muestras')
        if datoescReg == 2:
            self.listW_texRe.addItem
            ('Correcta configuracion prueba apagada')
        if datoescReg == -2:
            self.listW_texRe.addItem
            ('Incorrecta configuracion prueba encendida')
        if datoescReg == 3:
            self.listW_texRe.addItem
            ('Correcta configuracion RLD y Buffer')
        if datoescReg == -3:
            self.listW_texRe.addItem
            ('Incorrecta configuracion RLD y Buffer')
        if datoescReg == 4:
            self.listW_texRe.addItem
            ('Correcta configuracion Carac. de chupas')
        if datoescReg == -4:
            self.listW_texRe.addItem
            ('Incorrecta configuracion Carac. de chupas')
        if datoescReg == 5:
            self.listW_texRe.addItem
            ('Correcta configuracion de canales')
        if datoescReg == -5:
            self.listW_texRe.addItem
            ('Incorrecta configuracion de canales')
        if datoescReg == 6:
            self.listW_texRe.addItem
            ('Correcta configuracion RLD Positivo')
        if datoescReg == -6:
            self.listW_texRe.addItem
```

```
            ('Incorrecta_configuracion_RLD_Positivo')
        if datoescReg == 7:
            self.listW_texRe.addItem
            ('Correcta_configuracion_RLD_Negativo')
        if datoescReg == -7:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_RLD_Negativo')
        if datoescReg == 8:
            self.listW_texRe.addItem
            ('Correcta_configuracion_Deteccion_Positivos')
        if datoescReg == -8:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_Deteccion_Positivos')
        if datoescReg == 9:
            self.listW_texRe.addItem
            ('Correcta_configuracion_Deteccion_Negativos')
        if datoescReg == -9:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_Deteccion_Negativos')
        if datoescReg == 10:
            self.listW_texRe.addItem
            ('Correcta_configuracion_Normal_deteccion')
        if datoescReg == -10:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_Normal_deteccion')
        if datoescReg == 11:
            self.listW_texRe.addItem
            ('Correcta_activacion_Deteccion_de_Chupas')
        if datoescReg == -11:
            self.listW_texRe.addItem
            ('Incorrecta_activacion_Deteccion_de_Chupas')
        if datoescReg == 12:
            self.listW_texRe.addItem
            ('Correcta_configuracion_WCT1')
        if datoescReg == -12:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_WCT1')
        if datoescReg == 13:
            self.listW_texRe.addItem
            ('Correcta_configuracion_WCT2')
        if datoescReg == -13:
            self.listW_texRe.addItem
            ('Incorrecta_configuracion_WCT2')
    def escReg(self):
        thread_escReg = hilo_escReg(self)
        thread_escReg.dataescReg.connect(self.update_escReg)
        thread_escReg.start()

    def update_iniEnvDa(self,datoiniEnvDa):
        if datoiniEnvDa == 1:
            self.listW_texRe.addItem('_')
            self.listW_texRe.addItem('----------------------')
            self.listW_texRe.addItem('Envio_de_datos_iniciado...')
        if datoiniEnvDa == 2:
            self.listW_texRe.addItem('_')
            self.listW_texRe.addItem
            ('Recibiendo_datos,_puede_graficar_ahora!')
    def iniEnvDa(self):
        thread_iniEnvDa = hilo_iniEnvDa(self)
```

```
        thread_iniEnvDa.datainiEnvDa.connect(self.update_iniEnvDa)
        thread_iniEnvDa.start()

    def update_finEnvDa(self,datofinEnvDa):
        if datofinEnvDa == 1:
            self.listW_texRe.addItem('')
            self.listW_texRe.addItem('----------------------')
            self.listW_texRe.addItem
            ('Iniciando detencion de Envio de Datos...')
        if datofinEnvDa == 2:
            self.listW_texRe.addItem('')
            self.listW_texRe.addItem
            ('Iniciando cierre del programa en el Disp. Remoto')
        if datofinEnvDa == 3:
            self.listW_texRe.addItem('')
            self.listW_texRe.addItem
            ('Enviado senal de Parada al Servidor')
        if datofinEnvDa == 4:
            self.listW_texRe.addItem('')
            self.listW_texRe.addItem
            ('Proceso de Detencion completado')
    def finEnvDa(self):
        thread_finEnvDa = hilo_finEnvDa(self)
        thread_finEnvDa.datafinEnvDa.connect(self.update_finEnvDa)
        thread_finEnvDa.start()

    def update_grafD1(self,datografD1):
        dataD1[-1] = datografD1[-1]
        self.n +=1
        self.curve1.setData(dataD1)
        self.curve1.setPos(self.n,0)
    def grafD1(self):
        self.lD1 = pg.GraphicsLayout(border=(100,100,100))
        self.gra_D1.setCentralItem(self.lD1)
        self.gra_D1.show()
        self.gra_D1.setWindowTitle('Grafica D1')
        self.p1 = self.lD1.addPlot(title='Grafica D1')
        #self.p1.setYRange(-0.02, 0.02)
        self.curve1 = self.p1.plot(dataD1,pen='#ff0000')
        thread_lD1 = hilo_grafD1(self)
        thread_lD1.datagrafD1.connect(self.update_grafD1)
        thread_lD1.start()
    def fil_D1(self):
        self.ven_fil_D1 = QtWidgets.QMainWindow()
        self.uiD1 = Ui_filD1()
        self.uiD1.setupUi(self.ven_fil_D1)
        self.ven_fil_D1.show()
        self.TiD1 = int(self.lineE_TiD1.text())
        self.TfD1 = int(self.lineE_TfD1.text())
        self.mueIniD1 = self.TiD1*500
        self.mueFinD1 = self.TfD1*500
        self.datosfilD1 = self.datosD1[self.mueIniD1:self.mueFinD1]
        self.yD1 = signal.filtfilt(bbut, abut, self.datosfilD1)
        self.filD1 = signal.filtfilt(bnot, anot, self.yD1)
        self.fD1 = pg.GraphicsLayout(border=(100,100,100))
        self.uiD1.gra_filD1.setCentralItem(self.fD1)
        self.uiD1.gra_filD1.show()
        self.uiD1.gra_filD1.setWindowTitle('Grafica D1 Filtrado')
        self.pf1 = self.fD1.addPlot(title='Grafica D1 Filtrado')
```

```
        self.curvafD1 = self.pf1.plot(self.filD1)

    def update_grafD3(self,datografD3):
        dataD3[-1]=datografD3[-1]
        self.nD3 +=1
        self.curve3.setData(dataD3)
        self.curve3.setPos(self.nD3,0)
    def grafD3(self):
        self.lD3 = pg.GraphicsLayout(border=(100,100,100))
        self.gra_D3.setCentralItem(self.lD3)
        self.gra_D3.show()
        self.gra_D3.setWindowTitle('Grafica_D3')
        self.p3 = self.lD3.addPlot(title='Grafica_D3')
        self.curve3 = self.p3.plot(dataD3,pen='#8b00ff')
        thread_lD3 = hilo_grafD3(self)
        thread_lD3.datagrafD3.connect(self.update_grafD3)
        thread_lD3.start()
    def fil_D3(self):
        self.ven_fil_D3 = QtWidgets.QMainWindow()
        self.uiD3 = Ui_filD3()
        self.uiD3.setupUi(self.ven_fil_D3)
        self.ven_fil_D3.show()
        self.TiD3 = int(self.lineE_TiD3.text())
        self.TfD3 = int(self.lineE_TfD3.text())
        self.mueIniD3 = self.TiD3 * 500
        self.mueFinD3 = self.TfD3 * 500
        self.datosfilD3 = self.datosD3[self.mueIniD3:self.mueFinD3]
        self.yD3 = signal.filtfilt(bbut, abut, self.datosfilD3)
        self.filD3 = signal.filtfilt(bnot, anot, self.yD3)
        self.fD3 = pg.GraphicsLayout(border=(100, 100, 100))
        self.uiD3.gra_filD3.setCentralItem(self.fD3)
        self.uiD3.gra_filD3.show()
        self.uiD3.gra_filD3.setWindowTitle('Grafica_D3_Filtrado')
        self.pf3 = self.fD3.addPlot(title='Grafica_D3_Filtrado')
        self.curvafD3 = self.pf3.plot(self.filD3)

if __name__=='__main__':
    app=QtWidgets.QApplication([])
    window = MainWindow()
    window.show()
    app.exec_()
```

Diseño E Implementación De Un Dispositivo Para La Adquisición, Acondicionamiento, Transmisión Inalámbrica Y Almacenamiento De Señales Bioeléctricas ECG Optimizado Para Su Uso Portable

Design and implementation of a device for the acquisition, conditioning, wireless transmission and storage of bioelectric signals ECG optimized for portable use

Juan Camilo Caicedo España[1*] y José de Jesús Salgado Patrón[2]

Resumen

El dispositivo para la adquisición, acondicionamiento, transmisión inalámbrica y almacenamiento de señales bioeléctricas ECG diseñado, dispone del circuito integrado (IC) de montaje superficial (SMD), ADS1298, el cual permite el correcto sensado de la señales producidas por el corazón, las cuales son ingresadas a través de sus entradas diferenciales, logrando una resolución por muestra de 24 bits, gracias a los convertidores Sigma-Delta a una frecuencia de muestreo que varía desde 250 muestras por segundo (SPS), hasta 32.000 muestras por segundo (KSPS), siendo manipuladas por una interfaz de comunicación SPI. Al diseño final obtenido se adiciono la placa de desarrollo, Raspberry Pi Zero W, para permitir la comunicación entre el usuario y el dispositivo diseñado vía WIFI. Cuenta con una interfaz gráfica, diseñada sobre el lenguaje Python 3.7, que permite al usuario hacer una correcta visualización de un examen ECG estándar de 12 derivaciones en tiempo real, además de contar con la función de guardado de datos del paciente. La interfaz posee un panel de notificaciones y la función de filtrado digital, para un análisis minucioso de las señales recolectadas.

Palabras clave: ADS1298; derivaciones unipolares; derivaciones bipolares; protocolo TCP/IP; Redis.

Abstract

Device for the acquisition, conditioning, wireless transmission, and storage of bioelectrical signals designed ECG has integrated circuit (IC) surface mount (SMD), ADS1298, which allows the correct sensing of signals produced by the heart, which are entered through its differential inputs, achieving a resolution by a sample of 24-bit, thanks to the Sigma-Delta converters at a sampling rate that varies from 250 samples per second (SPS), up to 32,000 samples per second (KSPS), being manipulated by an SPI communication interface. The final design was obtained on the development board, Raspberry Pi Zero W, to allow communication between the user and the device designed via WIFI. The project has a graphical interface, designed on the language Python 3.7, which allows the user to make a correct display of ECG examination standard 12 leads in real-time, as well as having the function of saving data of the patient. The interface has a panel of notifications and the function of digital filtering, for a thorough analysis of the collected signals.

Keywords: ADS1298; unipolar leads, bipolar leads, TCP/IP protocol; Redis.

[1]Estudiante de ingeniería electrónica, Universidad Surcolombiana, Neiva, Colombia, u20131115407@usco.edu.co *(Autor para correspondencia)
[2] Mag. en Ingeniería Electrónica y de Computadores, Universidad Surcolombiana, Neiva, Colombia, josesalgadop@usco.edu.co
Fecha de recibo: *Fecha de revisión:* *Fecha de aprobación:*

1. Introducción

Las señales bioeléctricas han sido desde los años 50, un tema de absoluta importancia en el campo de la medicina, esto respondiendo a la necesidad de los especialistas, de obtener una rápida y fiable respuesta a un problema presentado en un paciente, reduciendo así un amplio espectro de posibles tratamientos para su pronta mejoría. Estas señales "son el resultado de la conductividad eléctrica que se produce en el cuerpo debido al movimiento de iones. La adquisición de estas señales implica transformar estas corrientes de iones en corrientes eléctricas susceptibles de ser manejadas por la instrumentación electrónica" Rodríguez (2005) que se tiene hoy en día. Entre las múltiples señales que produce el cuerpo humano destaca para este proyecto la señal electrocardiograma (ECG), esto debido a que el resultado del análisis de esta señal tiene un gran potencial para la detección de enfermedades y afecciones de un órgano tan importante como lo es el corazón. Desde que los métodos no invasivos de análisis se volvieron más confiables y rápidos se han realizado una cantidad extensa de trabajos e investigaciones para realizar mejoras sustanciales en los dispositivos de los cuales merecen especial mención los siguientes: "Diseño de un electrocardiógrafo digital portátil etapa por etapa compuesto de una interfaz de conversión analógica/digital para la adquisición de señales dieléctricas, una interfaz digitalizadora que establecen la comunicación con el PC" (Pascuas & Vargas, 2003). "Diseño de un ECG de 12 derivaciones de uso doméstico, el cual hace uso de un solo canal para la adquisición de las señales desarrollado con el IC AD8232" (Wildan, *et al.* 2015). "Prototipo electrocardiógrafo inalámbrico para la detección de enfermedades que desencadenen la muerte súbita, con software de diagnóstico médico aproximado" (Tabares & Perdomo, 2016). "Diseño de un sistema de sensado de señales ECG basado en IC ADS1291, MSP430 y CC25" (Chen, *et al.* 2016). "Diseño de un dispositivo para la adquisición de una señal EEG que incluya la reducción de artefactos oculares" Castillo (2017). "Sistema portable para la adquisición y procesamiento de señales ECG" (Villafuerte & Torres, 2017). "Diseño de un circuito con esquemas de ganancia y ancho de banda ajustables para la toma de un examen ECG estándar de 12 derivaciones" (Wu, *et al*, 2017). "Sistema portable y de bajo costo para la práctica de un ECG estándar de 12 derivaciones, para un cuidado remoto de la salud" (Choudhari, *et al.* 2018).

Revisados cada uno de los casos expuestos anteriormente, el presente proyecto cobra una importancia mayúscula pues su principal razón de ser, es diseñar una alternativa de bajo costo, confiable y de tamaño reducido para la práctica de un examen ECG estándar de 12 derivaciones, en tiempo real, que permita almacenar la información en un servidor remoto, para así, en el caso de ser necesario, enviar dichos datos a un centro de mayor complejidad para un análisis exhaustivo con el fin de dar una herramienta accesible a centros de salud de zonas apartadas del país que no poseen ni los recursos ni el personal idóneo para el diagnóstico de enfermedades cardiacas

2. Materiales y métodos

Para la realización del proyecto se siguieron 3 etapas a saber, adquisición, preprocesamiento, guardado y presentación como se muestra en el siguiente diagrama.

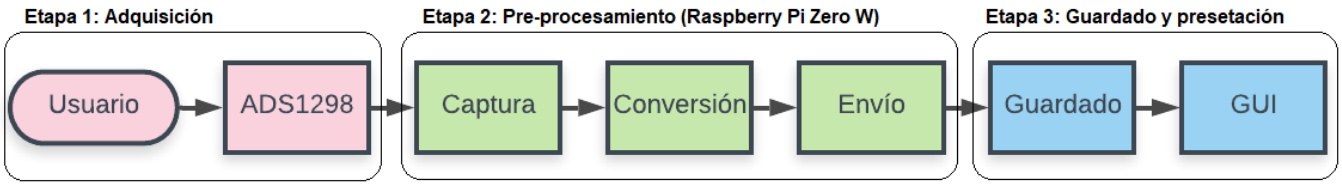


Figura 1. Diagrama de bloques

2.1 Etapa 1: Adquisición (Usuario, ADS1298)

Para la realización del diseño del dispositivo propuesto, se hizo de uso del IC ADS1298 de la empresa Texas Instruments, como se aprecia en la figura 1 y la placa de desarrollo Raspberry Pi Zero W.

Figura 2. ADS1298.

Este IC SMD, fue escogido por las prestaciones que ofrece para simplificar el proceso de adquisición y acondicionamiento de señales bioeléctricas, en especial la señal ECG, pues cuenta una característica de relación de rechazo en modo común (CMRR) de -115 dB, posee 8 canales en configuración de modo diferencial, multiplexor de entrada que permite la realización de pruebas de calibración del mismo IC, amplificadores de ganancia programable en factores de 1,2,3,4,6,12, todo esto controlado por una interfaz con protocolo SPI, además de contar con bloques específicos para la captura de señales ECG como lo son el circuito manejador de pierna derecha (RLD), y el terminal generador de la señal de Wilson (WCT). (Instruments, 2015).

Para el correcto diseño del proyecto se tomaron en cuenta las recomendaciones del fabricante del IC, optando por una alimentación eléctrica en modo bipolar (2.5 voltios, -2.5 voltios), logrando una polarización de IC más estable y recomendable. Esta polarización es lograda gracias alimentación de la placa de desarrollo y reguladores IC TPS60403DBVR, TPS73201MDBVREP, TPS72301DBVR. En cada entrada diferencial se incorporaron filtros pasivos de primer orden en cascada, uno pasa-alto con frecuencia de corte 153 kHz y un pasa-bajos de 338 kHz. Para finalizar, se implementó en el diseño un puerto VGA en paralelo para el ingreso de las señales directamente del paciente, y un cristal de oscilación de 2.048 MHz.

2.2 Etapa 2: Preprocesamiento (Raspberry Pi Zero W)

La Raspberry Pi Zero W es una placa que permite el desarrollo de distintos proyectos en el área de ingeniería electrónica. Está compuesta por un procesador BCM2835 con frecuencia de 1Ghz, memoria RAM de 512 MB, lector de tarjetas Micro SD, conexiones de 40 pines GPIO. Esta placa fue escogida porque cuenta con una interfaz de comunicación SPI para tener permanente control del ADS1298, y posee un interfaz wifi, la cual es la elegida para realizar la transmisión de datos recolectados al servidor diseñado.

2.3 Etapa 3: Guardado y presentación (Servidor y GUI)

El servidor encargado de la recepción de tramas que representan las muestras recolectadas de la señal ECG, es programado para que funcione como un socket de recepción, al cual se le asigna una IP estática, a la cual le serán enviadas las tramas. Estas tramas por la naturaleza de la placa implementada, son procesadas como un objeto lista, con números decimales que representan las muestras en el lenguaje Python, a dichas tramas se les hace un proceso de serialización lo cual agrega una primera capa de seguridad a la información. Para el envío de tramas, se hace uso del protocolo TCP/IP, el cual asegura que la información enviada es recepcionada de manera correcta sin pérdida de la misma. Cuando estas tramas son recibidas por el servidor, este implementa un algoritmo de detección y organización de tramas, para luego guardarlas en una base de datos no relacional. Este tipo de base de datos es escogida por el tipo de dato que se está tratando, es decir, un cumulo de listas de bytes que representan las muestras deseadas, para el tratamiento de estos datos, se hace uso de la herramienta Redis, que es un gestor de base de datos no relacional con características de guardado en memoria RAM del servidor reduciendo el tiempo de acceso a la información que requiere el servidor

La interfaz gráfica es diseñada con el lenguaje Python 3.7, haciendo uso de la librería PyQt 5, la cual ofrece prestaciones adecuadas para la presentación, manipulación y guardado de datos necesarios para el proyecto. Esta se compone de 5 paneles como se muestra en la figura 3.

Panel 1. **Datos del paciente:** En este panel se ingresa la información de interés del paciente al cual se le practicara el examen ECG tales como, nombre, edad y cedula del paciente; fecha de realización del examen, E.P.S a la cual se encuentra afiliado el paciente, y el nombre del médico que ordena el procedimiento.

Panel 2. **Notificaciones:** Este panel es el encargado de mostrar las notificaciones necesarias para que el usuario este permanentemente enterado de los procesos que se llevan a cabo y del estado del programa diseñado. Tales notificaciones son mostradas al momento de pulsar los botones del panel 3.

Panel 3. **Botones:** En este panel se encuentran los botones encargados del funcionamiento de la interfaz gráfica y de todo el proyecto en general. El botón Esc. Registros, es el botón encargado de ejecutar el código de ADS_esc9812.py. Este código es creado para escribir los registros del ADS1298 con el fin de lograr la captura de señales de manera correcta. El botón de Ini.Env.Datos, es encargado de ejecutar el código ADS_cap98.py, que es el algoritmo necesario para la correcta captura de tramas e inmediato envío de la información al servidor diseñado para tal fin. El botón de Fin.Env.Datos es el encargado de detener todas las funciones del lenguaje Python en la placa de desarrollo, esto con el fin de detener cualquier envío de información que pueda existir y que cause algún error en la toma del examen ECG. El botón de Gua. Datos, es el encargado de crear un archivo .dat, que contiene los datos ingresados en el panel 1, y las muestras recolectadas que se encuentran en la base de datos expuesta en el apartado anterior. Una vez creado el archivo, nombrado con la fecha de realización del examen y la cedula del paciente, la información de la base de datos es eliminada, y puesta en marcha nuevamente para el siguiente examen.

Panel 4. **Ventanas de Graficación:** En este panel se encuentran las ventanas de graficación. Cada una de ellas dedicada exclusivamente a mostrar una de las 12 derivaciones el ECG estándar. Las gráficas mostradas corresponden a las señales D1, D2, D3, V1, V2, V3, V4, V5, V6, aVR, aVL, aVF respectivamente. Estas graficas son mostradas una vez se pulse el botón de Iniciar. Si el usuario así lo desea, puede ingresar una ventana de tiempo especificada en rango de tiempo de inicio y tiempo final, para realizar un filtrado de la señal en esa ventana de tiempo para un análisis minucioso de la señal recolectada.

Panel 5. **Carga de datos, y revisión de conexiones:** En este panel se encuentran dos botones, el primero de ellos el de carga de datos, el cual se encarga de buscar un archivo .dat, que el usuario desee, y cargar la información contenida en el mismo para realizar un análisis de las señales recolectadas de un paciente con anterioridad. El botón de Rev. Conexiones, es el encargado de actualizar el estado de las conexiones del paciente, cuando el electrodo del paciente se encuentre correctamente conectado, este se pondrá en color verde, de lo contrario, el color mostrado será rojo.

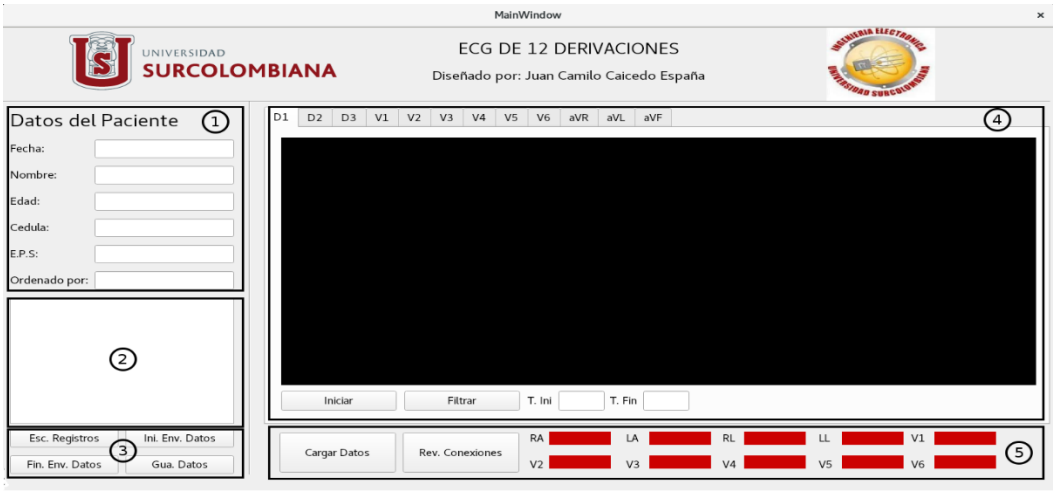


Figura 3. Interfaz gráfica diseñada

3. Resultados

El diseño final obtenido se muestra a continuación:

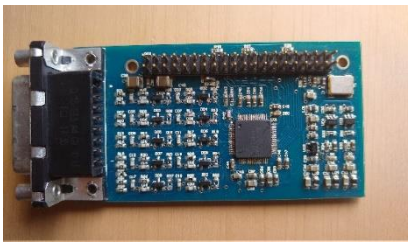


Figura 4. Diseño Final

Figura 5. Diseño final con Raspberry Pi Zero W

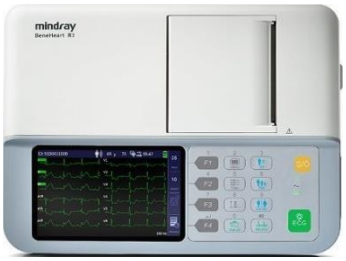


Figura 6. Dispositivo Comercial Mindry R3

Como se puede apreciar el dispositivo logrado es considerablemente mas pequeño con una medida de 91x40x14 milímetros en contra prestación de su competencia comercial, el dispositivo Mindry R3, Shop (2018) con dimensiones 250x194x56 milímetros. A su vez cumple con las funciones de sensado en tiempo real de las 12 derivaciones de un ECG estándar de este último, y adiciona el envío de información de forma inalámbrica hacia un servidor. Para comprobar la idoneidad, y buen funcionamiento tanto del diseño propuesto, como de los scripts diseñados para control del ADS1298, junto con el servidor e interfaz gráfica, se realizaron 3 pruebas como se muestran a continuación:

3.1 Prueba de señal cuadrada:

Esta prueba consiste en adquirir y mostrar una señal cuadrada de prueba con amplitud de 2.04 mV y frecuencia de 1 Hz por cada canal, que el mismo IC ADS1298 es capaz de generar, con fines de calibración y comprobación de buen funcionamiento de cada canal. Esta señal generada será muestreada a una tasa de 500 SPS en cada uno de los 8 canales. Como se puede apreciar, los canales del ADS1298 funcionan de manera adecuada para la toma de datos en pacientes, por lo cual se considera la prueba de la señal cuadrada como exitosa.

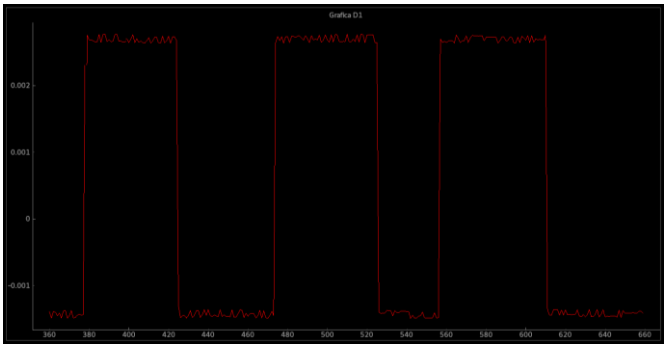


Figura 7. Señal Cuadrada Canal 1

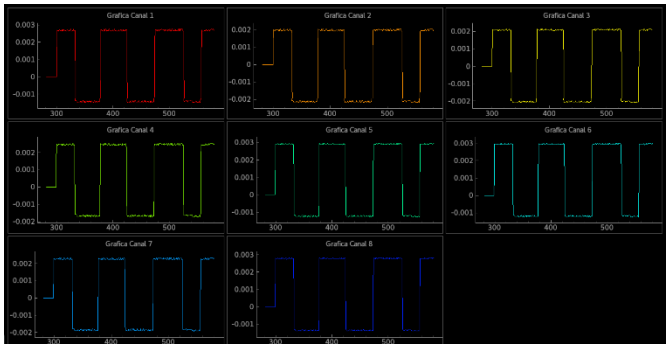


Figura 8. Señal Cuadrada 8 canales

3.2 Prueba de toma de datos con simulador

Esta prueba consiste en adquirir las señales desde un simulador especializado de señales ECG. Para ello, se hace uso del simulador "Patient Simulator 217A" de la empresa Dynatech Nevada. Al realizar la prueba con este tipo de simulador, se asegura que el equipo diseñado sea calibrado por un dispositivo certificado para la calibración de electrocardiógrafos comerciales.

Figura 8. "Patient Simulator 217A". Dynatech Nevada.

Al realizar la configuración de los canales para la adquisición de la señal, se procede a presentar los resultados obtenidos:

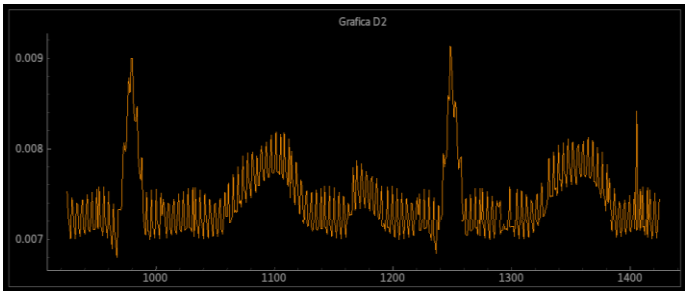


Figura 9. Señal ECG sin filtrar

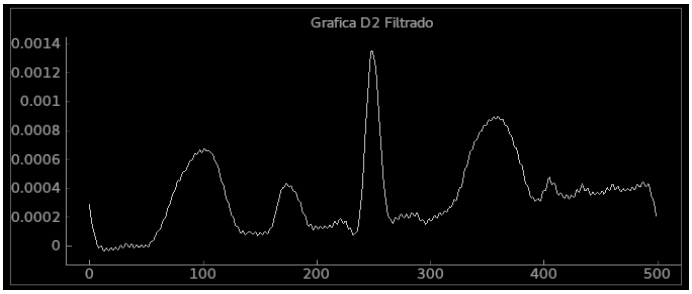


Figura 10. Señal ECG filtrada

Como se puede observar en la figura 9 se obtiene la señal ECG de la derivación D2 de manera correcta y clara. En la figura 10 se muestra la señal antes mencionada, ya filtrada digitalmente, por medio de dos filtros puestos en cascada, el primero de ellos es un filtro NOTCH de 60 Hz, con un factor de calidad de 0.6, el segundo de ellos es un filtro pasa banda Butterworth de 4 orden, la frecuencia de corte inferior es 0.05 Hz y la frecuencia de corte superior es 100 Hz. Este mismo comportamiento se ve replicado en todos los demás canales. Como se puede apreciar se elimina el nivel DC detectado en la prueba de la señal cuadrada, por lo cual se considera que la prueba practicada es exitosa.

3.2 Prueba de toma de datos en paciente

Esta prueba consiste en adquirir las señales de un paciente. El paciente escogido es un varón, con una edad de 18 años, con antecedentes buena alimentación. A continuación, se muestran los resultados de las 12 derivaciones.

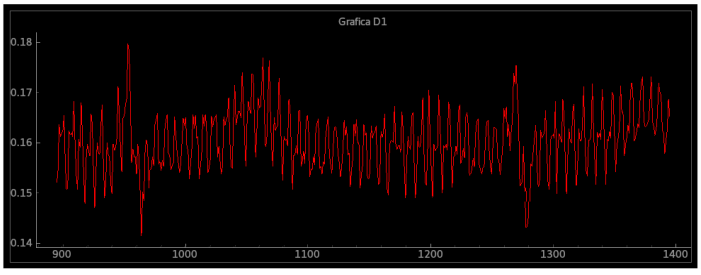


Figura 11. Señal D1 de Paciente.

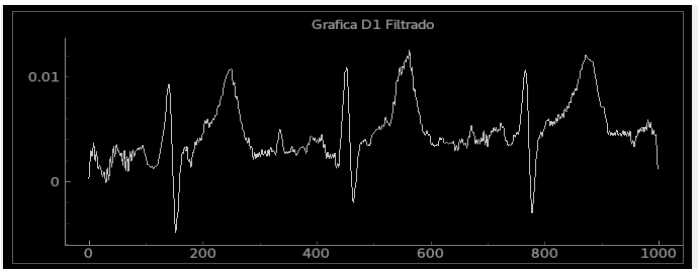


Figura 12. Señal D1 Filtrada de Paciente.

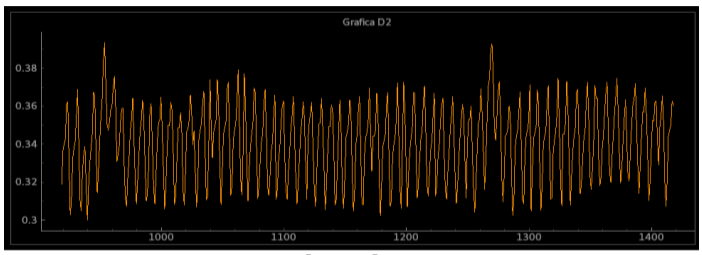


Figura 13. Señal D2 de Paciente.

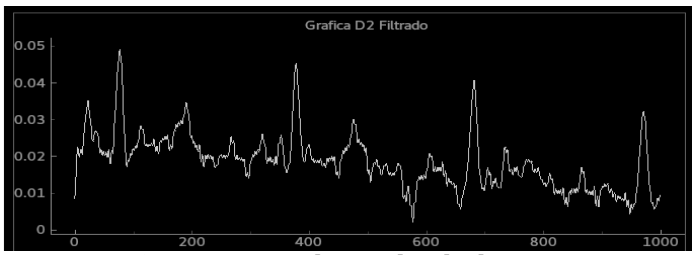


Figura 14. Señal D2 Filtrada de Paciente.

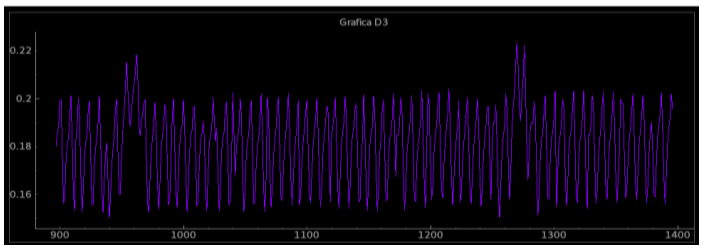


Figura 15. Señal D3 de Paciente.

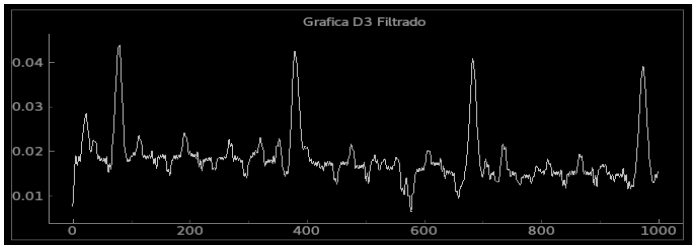


Figura 16. Señal D3 Filtrada de Paciente.

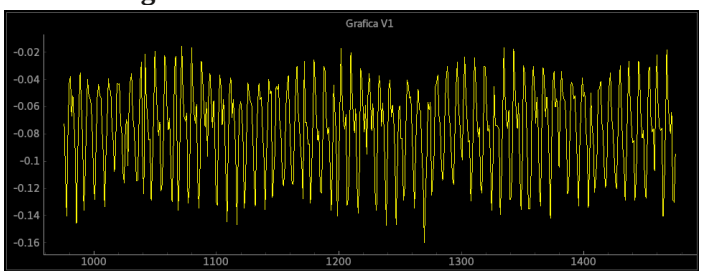


Figura 17. Señal V1 de Paciente.

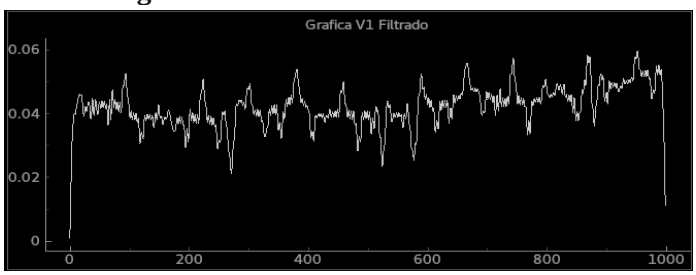


Figura 18. Señal V1 Filtrada de Paciente.

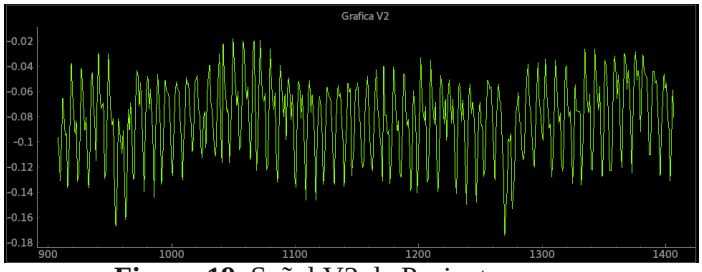


Figura 19. Señal V2 de Paciente.

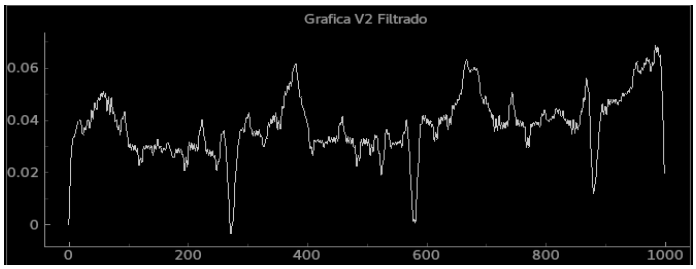


Figura 20. Señal V2 Filtrada de Paciente.

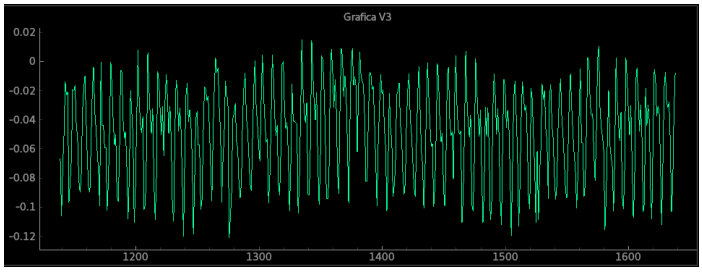


Figura 21. Señal V3 de Paciente.

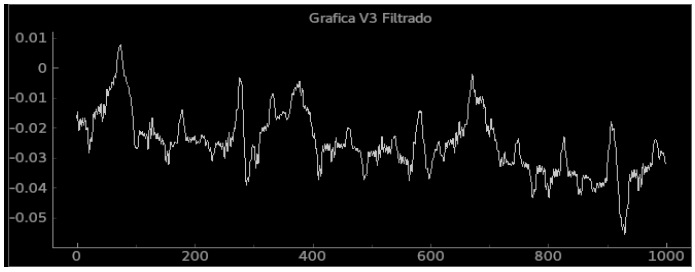


Figura 22. Señal V3 Filtrada de Paciente.

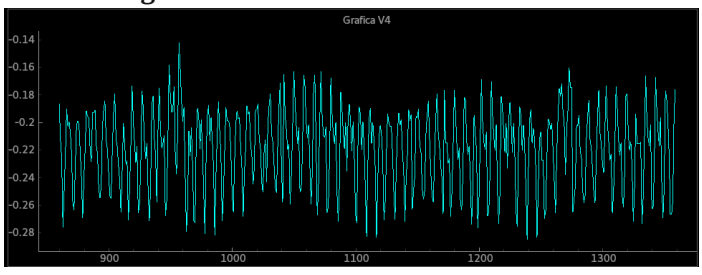


Figura 23. Señal V4 de Paciente.

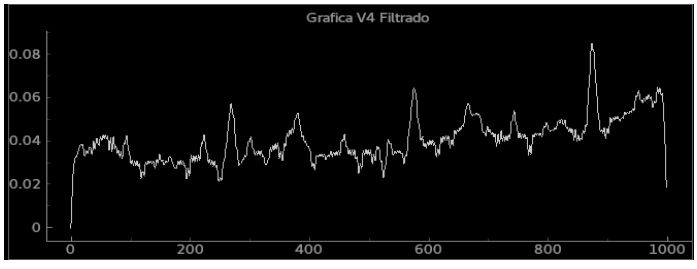


Figura 25. Señal V4 Filtrada de Paciente.

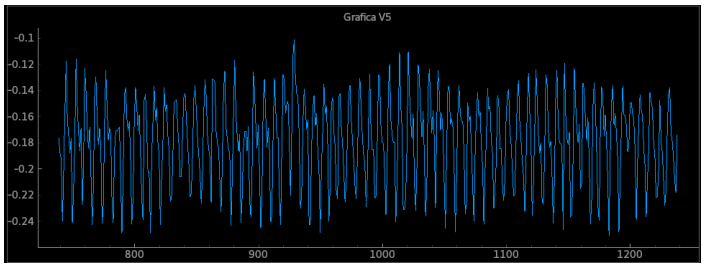


Figura 26. Señal V5 de Paciente.

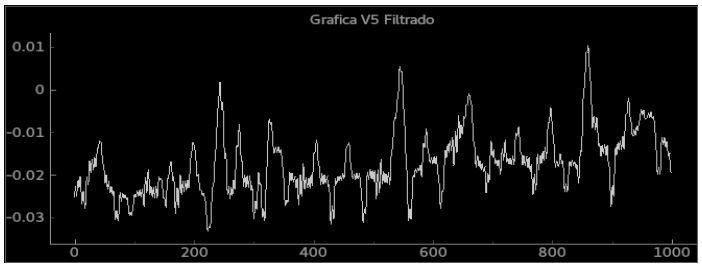


Figura 27. Señal V5 Filtrada de Paciente.

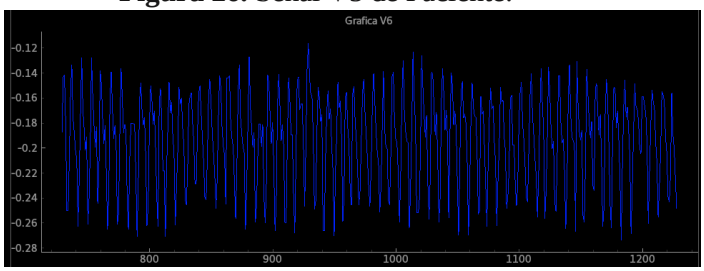


Figura 28. Señal V6 de Paciente.

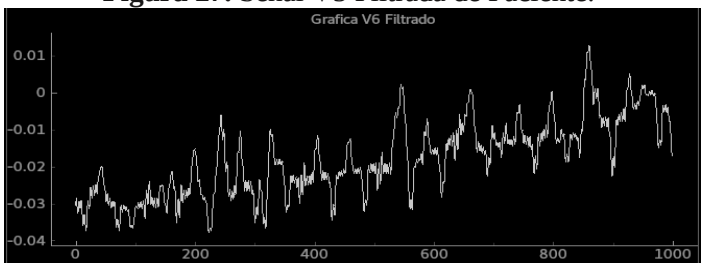


Figura 29. Señal V6 Filtrada de Paciente.

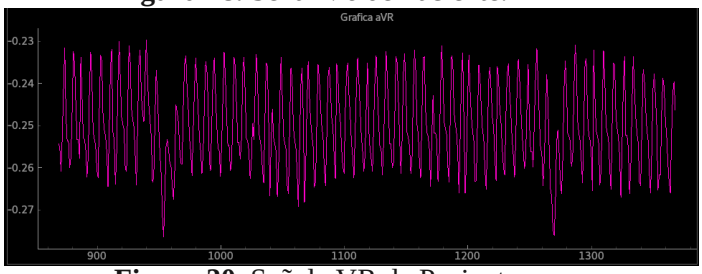


Figura 30. Señal aVR de Paciente.

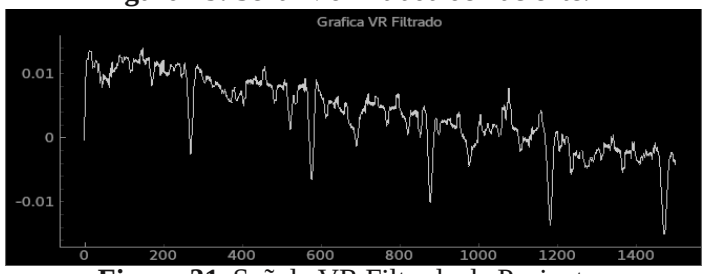


Figura 31. Señal aVR Filtrada de Paciente.

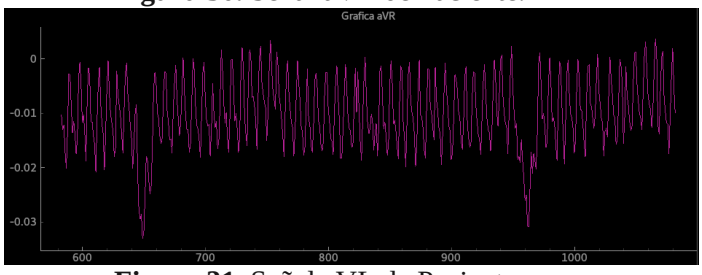


Figura 31. Señal aVL de Paciente.

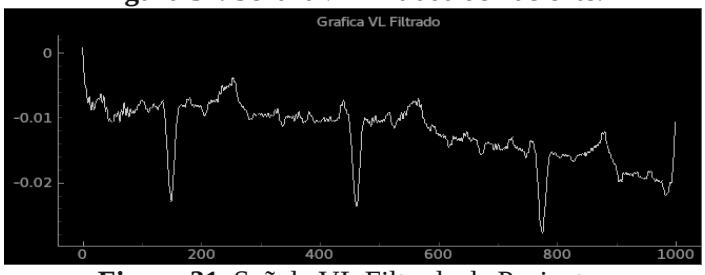


Figura 31. Señal aVL Filtrada de Paciente.

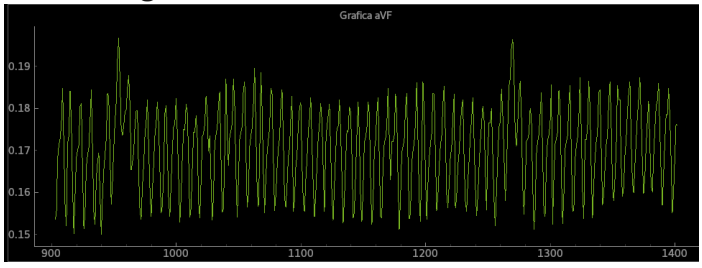


Figura 31. Señal aVF de Paciente.

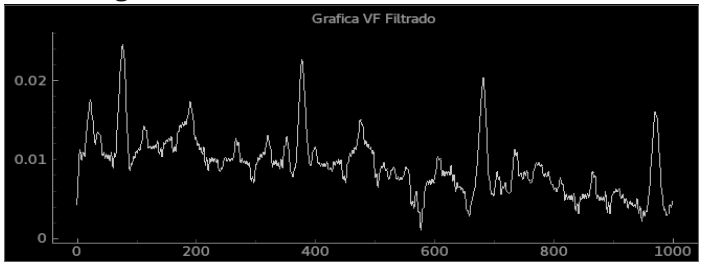


Figura 31. Señal aVF Filtrada de Paciente.

Como se puede apreciar, los resultados muestran una correspondencia entre las señales recolectadas por el simulador, y las señales adquiridas en un paciente real, lo cual no permite afirmar que la prueba practicada en el paciente, ha sido exitosa.

4. Conclusiones

Se realizo un correcto estudio de los dispositivos de tipo SMD existentes en el mercado, con base en las características, que permitieran la correcta adquisición de una señal bioeléctrica, estándares de filtrado, y precio. Encontrándose que la opción correcta para el desarrollo del presente proyecto es el integrado ADS1298, pues permite diseñar un sistema robusto, escalable y de precio reducido para la adquisición de señales ECG, con útiles funciones como lo son bloques dedicados a la generación de señal de manejo de pierna derecha (RLD), y señal de la terminal central de Wilson (WCT) indispensables ambas a la hora de hacer la toma de estas señales.

Se diseño un dispositivo con el integrado mencionado, lo que permitió estudiar los efectos que se producían al manipular la señal ECG, por medio de la utilización de muestreadores Sigma-delta internos por cada canal, traduciéndose en una útil ventaja, pues al realizar el muestreo de la señal por este método, se encuentra que gran parte del ruido de alta frecuencia que puede invadir el estudio de la señal ECG es considerablemente reducido. Además, al contar con un puerto de comunicación SPI y al hacer la correcta elección de una placa de desarrollo como lo es la Raspberry Pi Zero W, se logró un producto final con comunicación más rápida entre circuitos para tener un óptimo control de las funciones de lectura, escritura y sensado que permite el ADS1298.

Se ejecuto la adquisición adecuada de las señales ECG logrando así la recolección necesaria de tramas para un examen estándar de 12 derivaciones en tiempo real. Todo esto con la facilidad que permite el algoritmo diseñado para transmitir las tramas recolectadas por la placa de desarrollo, haciendo uso del protocolo TCP/IP, dando una garantía al usuario del producto, que la información ha sido transmitida y recepcionada sin ningún tipo de perdida de datos en este proceso, mencionando además la implementación de un algoritmo de prevención de errores en el servidor que reduce aún más esta posibilidad, añadiendo la utilización de un gestor de bases de datos no relacional como lo es Redis, dando al proyecto altas posibilidades de escalamiento rápido, preciso, y adecuado, pues, al guardar las tramas en una base de datos de esta clase, no se altera el tipo de dato que se guardó, añadiendo un acceso más rápido a las mismas, y quitando gran carga al servidor, encargado recepcionar las tramas, pues como se muestra en el proyecto, se hizo necesario la codificación de tramas para el correcto envió de las mismas, pasando así la tarea de decodificar las tramas, a cada hilo de graficación.

Se implemento una interfaz útil, sin complicaciones, que permite la inclusión de datos de interés acerca de la persona a quien se le practica el procedimiento, como son el nombre, la edad, número de identificación, además de la fecha de realización del examen y el nombre de quien ordena el procedimiento. Cuenta además con un panel de notificaciones que permite al usuario estar permanentemente informado del estado y acciones que realiza el programa al hacer cualquier acción sobre el mismo, contando con un panel de carga de datos para el análisis de exámenes anteriormente guardados, útil al realizar una inspección más detallada por parte del usuario, presentando las señales recolectadas en un panel graficador que permite visualizar adecuadamente las mismas, y de ser necesario realizar un filtrado de una ventana de tiempo, de la señal que el usuario considere importante para un análisis exhaustivo. Esta interfaz, aparte de todo permite el guardado, y carga de datos encapsulados por la biblioteca pickle, permitiendo reducir su tamaño, y dar un acceso más rápido a la información para el lenguaje empleado python 3.7.

5. Referencias Bibliograficas

Castillo, A.P., 2017. Diseño De Un Dispositivo Para La Adquisición De Una Señal EEG Que Incluya La Reducción De Artefactos Oculares. Escuela Colombiana de Ingeniería Julio Garavito.

Chen, J.Z., Wang, L.H.,Wang, F.X., Fan, M.H., 2016. Design of ECG signal acquisition system based on ADS1291. International Conference On Communication Problem-Solving – ICCP 2016, septiembre. DOI: 10.1109/ICCPS.2016.7751123

Shop, D. 2018. ECG Mindray BeneHeart R3. https://www.doctorshop.es/Prodotti/equipos-PP/electrocardiografos-ecgs-PP-1/ecg-mindray-beneheart-r3-107221 Consultado 24 de mayo 2019

Choudhari, V., Dandge, V., Choudhary, N., Sutar, R., 2018. A portable and low-cost 12-lead ECG device for sustainable remote healthcare. International Conference on Communication information and Computing Technology – ICCICT 2018, marzo. DOI: 10.1109/ICCICT.2018.8325879

Instruments, T., 2015. Data Sheet ADS1298. http://www.ti.com/lit/ds/symlink/ads1298.pdf. Consultado 24 de mayo de 2019.

Pascuas, C., Vargas, M.F., 2003. Electrocardiógrafo Digital Portátil. Universidad Surcolombiana.

Rodríguez, J.M, 2005. Adquisición de señales biológicas. http://www.usc.es/catedras/telemedicina/2005/materialAsignatura/AdquisicionSenalesBiologicas.pdf. Consultado el 23 de mayo de 2019.

Tabares, S. A., Perdomo, J., 2016. Prototipo electrocardiógrafo inalámbrico para la detección de enfermedades que desencadenen la muerte súbita, con software de diagnóstico médico aproximado. Universidad Surcolombiana.

Wildan, M., Zakaria, H., Mengko, R., 2015. Design of ECG Homecare:12-lead ECG acquisition using single channel ECG device developed on AD8232 analog front end. International Conference on Electrical Engineering and Informatics – ICEEI 2015, agosto. DOI: 10.1109/ICEEI.2015.7352529

Wu, C. H., Chen, C.B., Zhou, J.C., 2017. A 12-lead ECG integrated circuit design with adjustable gain and bandwidth schemes. International Conference on Fuzzy Theory and Its Applications – iFUZZY 2017, noviembre. DOI: 10.1109/iFUZZY.2017.8311814

Villafuerte, S., Torres H, 2017. Sistema Portable Para La Adquisición Y Procesamiento De Señales ECG, Con Aplicabilidad En Dispositivos Móviles. Universidad San Buenaventura de Cali.