UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"DISEÑO E IMPLEMENTACIÓN DE UN ELECTROCARDIÓGRAFO CON AUTODIAGNÓSTICO PARA REDUCIR EL ÍNDICE DE MORTALIDAD OCASIONADAS POR PATOLOGÍAS CARDÍACAS QUE AFECTAN A ZONAS CON POBLACIONES VULNERABLES EN EL PERÚ"

PARA OBTENER EL TÍTULO DE INGENIERO ELECTRÓNICO

AUTORES:

Bach. HUAMAN YRIGOIN, DENNIS

Bach. VALDEZ DE LA CRUZ, ALEX JUNIOR

ASESOR: **Dr. Ing. RAÚL NICANOR BENITES SARAVIA**

CO-ASESOR: **Dr. Ing. JUAN HERBER GRADOS GAMARRA**

Callao, 2022

PERÚ

HOJA DE REFERENCIA DEL JURADO Y APROBACIÓN

PRESIDENTE	**:**	**Mg. Ing. Luis Ernesto Cruzado Montañez**
SECRETARIO	**:**	**Dr. Ing. Santiago Linder Rubiños Jiménez**
VOCAL	**:**	**Mg. Ing. Jorge Elías Moscoso Sánchez**
ASESOR	**:**	**Dr. Ing. Raúl Nicanor Benites Saravia**
CO–ASESOR	**:**	**Dr. Ing. Juan Herber Grados Gamarra**

DEDICATORIA

A nuestros padres y amigos, que siempre estuvieron brindándonos su apoyo incondicional. Por motivarnos y ayudarnos a poder continuar y terminar esta tesis pese a las adversidades.

AGRADECIMIENTO

A nuestro asesor Dr. Raúl Benites Saravia y co-asesor Dr. Juan Grados Gamarra, por brindarnos su apoyo en la revisión, paciencia y sugerencias, son ellos a quienes debemos gran parte de nuestro aprendizaje para poder culminar esta presente investigación.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de tesis tiene como objetivo principal el diseño e implementación de un dispositivo electrocardiógrafo que pueda reducir el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú.

Para ello se profundizará en el análisis y procesamiento de las señales de electrocardiograma mediante el uso de filtros analógicos y digitales y así determinar el método óptimo de filtrado de las señales de ECG. Luego se procede a la comparación de estas señales adquiridas con una base de datos de PhysioNet para el entrenamiento de un algoritmo de inteligencia artificial, el cual para esta investigación se consideró el empleo de un árbol de decisiones, en busca de un modelo con mayor precisión para el autodiagnóstico de enfermedades cardiovasculares basándonos en parámetros estadísticos a fin de obtener mejores resultados.

Palabras clave: electrocardiógrafo, autodiagnóstico, inteligencia artificial, árbol de decisiones.

INTRODUCCIÓN

En el Perú, actualmente las enfermedades cardiovasculares son consideradas unas de las principales causas de muerte según de la Sociedad Peruana de Cardiología y el Ministerio de Salud. Estas son ocasionadas generalmente por malos hábitos alimenticios, tabaquismo, hipertensión arterial, tenencia de un grado elevado de colesterol, diabetes, inactividad física (sedentarismo), estrés, herencia genética, etc. Entre las principales enfermedades cardiovasculares comunes en el Perú son: arritmias, infarto de miocardio, insuficiencia cardiaca, hipertensión arterial, fibrilación auricular e infarto de miocardio.

Dada a la geomorfología de nuestro país, las masas migratorias se dan de manera constante, por consiguiente, deriva con ello cambios en sus hábitos, estilos de vida y alimentación. Esto conlleva a que existan diferencias epidemiológicas variables entre los habitantes de un determinado lugar. Además, las enfermedades cardiovasculares, la morbilidad y desarrollo las mismas en cada persona se dan por la existencia de diferencias educativas y socioeconómicas.

Se estima que para el 2030, casi 23.6 millones de personas morirán por alguna enfermedad cardiovascular, principalmente por cardiopatías y accidentes cerebrovasculares. Y además se prevé que estas enfermedades sigan siendo la principal causa de muerte a nivel mundial [1].

Viendo la necesidad de un dispositivo que permita ayudar en la prevención frente a estas enfermedades, la finalidad de esta investigación es la de diseñar, desarrollar y construir un dispositivo que permita el monitoreo de las señales de electrocardiograma y mediante el uso de inteligencia artificial, poder clasificar las patologías existentes en un paciente determinado, brindado así un diagnóstico que pueda ser contrastado con la de un especialista en la materia.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción de la realidad problemática

Hoy en día, el número de personas que mueren por enfermedades cardíacas es mucho mayor que en años anteriores y esta cifra tiende a incrementar, afectando directamente el índice de mortalidad en el Perú.

Esto se debe por la poca conciencia de las personas por el cuidado de su salud, el no llevar una vida con actividad física, una alimentación balanceada el cual ayudaría a prevenir y evitar el desarrollo de enfermedades cardiovasculares. Las poblaciones vulnerables que se encuentran en zonas remotas no tienen ciertas facilidades para poder realizarse una revisión o un análisis de manera periódica pues los centros hospitalarios no tienen los recursos necesarios, sea por personal o por equipamiento, para compensar esa gran demanda de atención médica.

1.2 Formulación del problema

1.2.1 Problema general

¿La implementación de un dispositivo electrocardiógrafo con diagnóstico automático puede reducir el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú?

1.2.2 Problemas específicos

a. ¿Cuál es el método óptimo para la adquisición de datos y procesamiento de señales electrocardiográficas?
b. ¿Cómo un algoritmo de I.A. puede influir en la prevención ante enfermedades cardiovasculares que afecten a zonas con poblaciones vulnerables en el Perú?
c. ¿Qué tan confiable es el diagnóstico emitido por un algoritmo de IA comparado con uno estándar?

1.3 Objetivos

1.3.1 Objetivo general

Diseñar e implementar un electrocardiógrafo con diagnóstico automático para reducir el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú.

1.3.2 Objetivos específicos

a. Aplicar el óptimo método para la adquisición de datos y procesamiento de las señales electrocardiográficas.
b. Desarrollar un algoritmo de I.A para influir en la prevención ante enfermedades cardiovasculares que afecten a las zonas con poblaciones vulnerables en el Perú.
c. Contrastar el diagnóstico emitido por el algoritmo de I.A y se comparará con uno estándar.

1.4 Justificación

Ante el incremento de las muertes por enfermedades cardiovasculares en los últimos años en el Perú, resulta notable conocer cuáles son los parámetros fundamentales que determinan las patologías cardiovasculares, con el fin de establecer medidas que permitan prevenirlos y/o controlarlos.

La escasez y/o bajo desarrollo de tecnología asequible, confiable y de calidad para el rubro médico, tecnología orientada al uso en zonas rurales donde no existen centro especializados para el diagnóstico de patologías cardíacas.

La presente investigación se origina de la necesidad de prevenir las enfermedades cardiovasculares, con el propósito de poder reducir el índice de mortalidad en el Perú mediante el diseño e implementación de un electrocardiógrafo que permita un diagnóstico automático de patologías.

1.5 Limitantes de la investigación

1.5.1 Teórica

Esta investigación profundiza en el estudio de las señales de ECG frente a la deficiencia de investigaciones en el Perú que profundicen en el análisis y procesamiento de señales de electrocardiograma utilizando algún tipo de inteligencia artificial.

1.5.2 Temporal

La investigación se vio afectada por la pandemia del COVID-19 en su desarrollo y requirió una extensión en el tiempo para su ejecución por la cuarentena y debido a que la posibilidad para acceder a los hospitales del departamento de Lima para la toma de datos era inviable por cuestiones sanitarias.

1.5.3 Espacial

La carencia de recursos económicos para realizar viajes a las distintas ciudades del país para realizar la adquisición de datos de las señales de electrocardiograma y así generar nuestra propia base de datos.

2. MARCO TEÓRICO

2.1 Antecedentes del estudio

2.1.1 Nacionales

En el 2007, Huaman [2] nos menciona que el objetivo principal de su tesis fue preparar un simulador de señales electrocardiográficas (ECG) para la estimación funcional de monitores cardiológicos. Donde se concluyó que “se pudo elaborar un simulador de señales de ECG para la evaluación de monitores, el cual utiliza microcontroladores para procesar la frecuencia del ritmo cardiaco y poder utilizarlo en dispositivos que admiten señales de 3, 5 y 10 derivaciones [2].

En el 2018, García y Quino [3] nos comentó que el objetivo de su trabajo de investigación consistió en intervenir en la problemática del Hospital Nacional Arzobispo Loayza y desarrollar un equipo de bajo costo tipo Holter de una derivación para el monitoreo de señales cardiovasculares. Donde concluyeron que sus resultados fueron satisfactorios y obtuvieron una buena calidad de señal para el descarte de arritmias y fibrilación auricular [3].

En el 2018, Yupanqui y Roncal [4] tuvieron por objetivo el diseñar e implementar un dispositivo de monitoreo de señales cardiovasculares portátil cuyo costo sea asequible, el cual transmite la data obtenida hacia una computadora. Donde demostraron que su prototipo diseñado fue de bajo costo, el cual contribuye al diseño de dispositivos médicos para poder ser adquiridos por centro de salud que no dispongan de mucho presupuesto [4].

En 2019, Chanta [5] nos comentó que el objetivo de su tesis fue el predecir la frecuencia de interpretación correcta del ECG de pacientes con SCA por médicos de emergencia de los establecimientos de nivel II y III – Lambayeque 2017. Donde concluyó que la frecuencia de interpretación adecuada fue menor a la mitad del personal médico,

mientras que la frecuencia del diagnóstico correcto fue mayor a la mitad del personal médico [5].

En el 2015, Vásquez [6] sostuvo que el objetivo principal de su tesis fue la de realizar el diseño y la implementación de un equipo que capte las señales eléctricas del corazón, las trate, y que sea capaz de identificar en tiempo real y emitir una alerta remota ante la ocurrencia de FA en pacientes postrados en cama. Donde concluyó que se pudo implementar un sistema capaz de adquirir una señal electrocardiográfica en la derivación DII, con la capacidad de detectar y alertar anomalías en esta señal que puedan ser atribuible a eventos de fibrilación auricular enviando una alerta vía SMS [6].

2.1.2 Internacionales

En el 2017, Gutiérrez [7] tuvo como principal objetivo en su tesis la de verificar hasta qué punto es factible desarrollar un sistema completo, desde que se recibe la señal hasta que se transforma y trata y está preparada para ser transmitida al sistema operativo. Donde concluyó que la decisión de utilizar el menor número de componentes externos posibles, tratando de aprovechar lo que la Zynq-7000 tiene que ofrecer es lo que ha determinado que se optara por el convertidor analógico – digital que incluye el FPGA [7].

En 2018, Burbano [8] mencionó que el objetivo de su tesis fue el diseño e implementación de un simulador de electrocardiografía (ECG), el cual permite generar las diferentes señales de las derivaciones bipolares, unipolares, precordiales y dos señales de patologías. Donde concluyó que el propósito general era dejar un prototipo funcional para la empresa Innovatec, utilizando instrumentos económicamente asequibles pero que garantizaran una funcionalidad y respuesta favorable a la hora de cumplir con las funciones básicas de dicho simulador ECG para fines de mantenimiento preventivo [8].

En 2018, Carrera [9] tuvo como objetivo principal en su tesis la del desarrollo de un prototipo de adquisición de señales Electrocardiográficas, el prototipo se basa principalmente en la adquisición de estas señales por medio de electrodos y transmitirlas a una tarjeta de adquisición desarrollada mediante amplificadores de instrumentación. Concluyó que el método de monitoreo de pacientes cardiacos actualmente en su país se realiza de forma tradicional y aún no se utilizan las nuevas tecnologías de comunicación y tampoco va ligado a sistemas de telemedicina [9].

En 2017, Tsampi [10] tuvo como objetivo principal en la tesis la de crear un sistema que pueda proporcionar el procesamiento adecuado y producir los resultados del ECG. Se concluyó que su sistema analiza la señal y evalúa algunas métricas que ayudan al diagnóstico preciso contra las enfermedades del corazón [10].

En 2018, Pérez [11] menciona como objetivo principal en su tesis la del diseño de un dispositivo de adquisición de ECG utilizando un microcontrolador y el circuito integrado ADAS1000 de Analog Devices para mejorar la prevención secundaria de cardiopatías. Donde se concluyó que la monitorización de la actividad cardiaca en tiempo real de un paciente es una tarea de gran relevancia en los últimos años debido al incremento considerable de muertes a causa de patologías cardiovasculares [11].

2.2 Bases Teóricas

2.2.1 Electrocardiógrafo y señales fisiológicas del corazón

El electrocardiógrafo es un instrumento de medición el cual tiene por objetivo brindarnos información respecto a la dirección y magnitud de las corrientes eléctricas que son emitidas por el órgano del corazón, este dispositivo es ampliamente utilizado a nivel mundial para el monitoreo correcto de las señales fisiológicas del corazón en tiempo real además de ser usados para poder diagnosticar patologías

cardiacas. El dispositivo utiliza la dirección resultante de la corriente del músculo del corazón, obteniendo la resultante mediante el uso electrodos que son posicionados en diferentes partes del cuerpo. Los electrodos, derivan la señal eléctrica al electrocardiógrafo, y a través de un galvanómetro, se obtiene la corriente que pasa por el dispositivo y se transmite a un receptor que registra la señal ECG y permite observar los complejos [12].

Este registro es documentado a partir de reflexiones tanto positivas como negativas en relación a una línea isoeléctrica puede ser sobre papel o en un monitor para comprender la actividad eléctrica. Cada actividad anatómica del sistema de conducción del corazón corresponde a una porción del registro eléctrico, esta puede ser en forma de ondas, segmentos e intervalos. La línea isoeléctrica corresponde a la ausencia de actividad eléctrica, y se considera como punto de base para determinar si un evento sucede por arriba de esta línea siendo positivo, o si ocurre por debajo de ella siendo negativo.

Una onda se define como una deflexión ya sea positiva o negativa a partir de la línea isoeléctrica. Un segmento se define como la línea isoeléctrica entre dos ondas dentro de un mismo latido y un intervalo se define como el complejo de una o más ondas con un segmento, tal como muestra la Figura N° 1.

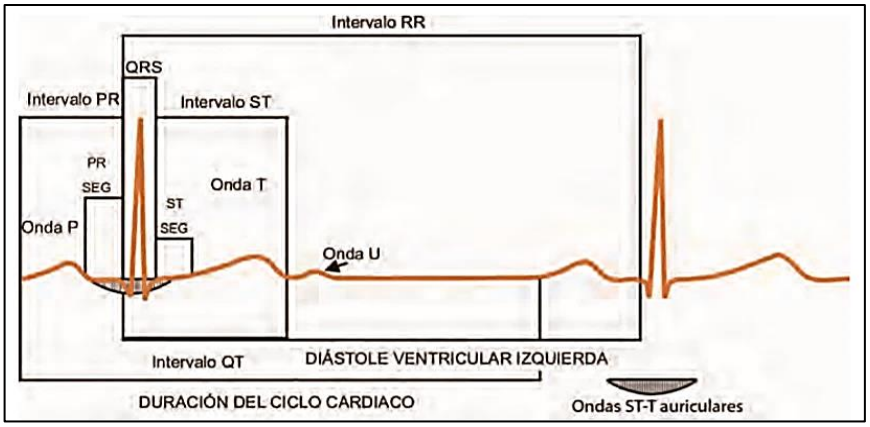


Figura N° 1. Relaciones temporales entre las diferentes ondas del ECG y nomenclatura de los intervalos y segmentos.

Fuente: Bayés de Luna, Antoni. Manual de electrocardiografía básica, 2014 [13].

2.2.1.1 Onda P

La primera deflexión es la onda P, la primera mitad de la onda P corresponde a la actividad de la aurícula derecha, y la segunda mitad a la aurícula izquierda, esta deflexión debe ser redondeada y generalmente esta onda es menor de los 0.3 mV. Cuando esta onda P es mayor que los 0.3 mV, se correlaciona con un crecimiento de la aurícula derecha. Cuando la morfología de la onda P no es redondeada, sino que tiene la forma de m se correlaciona al crecimiento de la aurícula izquierda.

2.2.1.2 Complejo QRS

La primera deflexión de este complejo es negativa, denominada onda q, esta onda viene dada por la despolarización del septum interventricular, la onda positiva (u onda R) corresponde a la despolarización del ventrículo izquierdo y la deflexión final de este complejo (onda S) corresponde a la despolarización del ventrículo derecho.

2.2.1.3 Onda T

Esta deflexión positiva (Onda T), corresponde a la repolarización ventricular, el cual es necesario para que pueda producirse una nueva despolarización, esta onda es generalmente positiva o negativa, un ejemplo se muestra en la Figura N° 2.

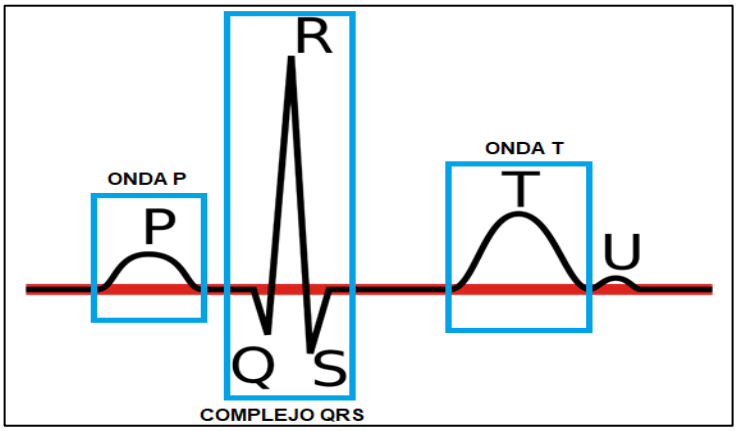


Figura N° 2. Señal fisiológica del corazón con sus respectivas ondas P, T y el complejo QRS.

Fuente: Autoría propia, 2022.

2.2.2 Derivaciones

Existen 6 derivaciones situadas en el plano frontal llamadas I, II, III, VR, VL, VF, que captan la actividad eléctrica con los electrodos en las extremidades del cuerpo y seis en el plano horizontal de V1 a V6 como se observa en la Figura N° 3, que registran la actividad cardiaca con electrodos colocados en el precordio. Cada derivación se coloca en un cierto lugar específico (en ángulos) y se tiene por cada una de ellas una línea de derivación el cual está opuesto en 180°, pasando por el eje del corazón, cada derivación se divide en una parte positiva y en otra negativa [13].

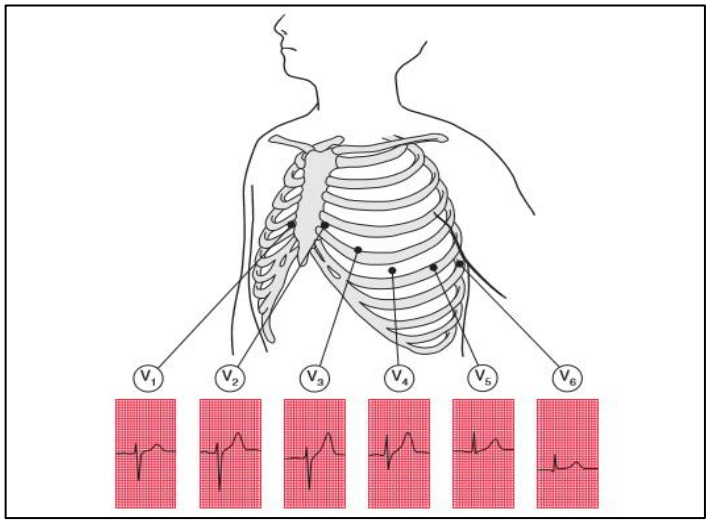


Figura N° 3. Patrones de ECG registrados por derivaciones en el pecho.
Fuente: R. Hampton, John. The ECG made easy, 2013 [14].

2.2.3 Inteligencia Artificial

La inteligencia artificial (I.A.) es una rama de la ciencia que se enfoca en el aprendizaje y/o desarrollo de comportamientos inteligentes en elementos artificiales. La I.A. tiene por objetivo el de crear sistemas y dispositivos que puedan discernir como si la acción fuera llevada por una persona. Estas decisiones y la capacidad de adaptación en diferentes entornos son comportamientos que guardan concordancia con un comportamiento inteligente [15].

Actualmente existen diferentes tipos de aprendizajes y enfoques de estudio en desarrollo el cual se enfocan en perfeccionar dichos comportamientos y/o decisiones autónomas.

2.2.3.1 Machine Learning

El campo de Machine Learning se refiere a la cuestión de cómo construir programas informáticos que mejoren automáticamente con la experiencia. Desde haber desarrollado programas que detectan transacciones fraudulentas de tarjeta de crédito, hasta vehículos autónomos que pueden circular por autopistas. Ha habido un importante avance en la teoría y el desarrollo de algoritmos que permiten profundizar y ahondar en este campo de investigación [16].

2.2.3.2 Deep Learning

El Deep Learning es un subcampo específico del Machine Learning el cual se enfoca en el estudio del aprendizaje a través de datos y pone énfasis en el aprendizaje sucesivo en capas. En el Deep learning, estas capas son representadas a través de modelos llamados redes neuronales, estructuradas en capas una encima de la otra. Este nombre último se tomó en referencia al entendimiento del cerebro. Teniendo en cuenta que el Deep learning es matemáticamente un “framework” para el aprendizaje desde representaciones de datos [17].

2.2.4 Tipos de aprendizajes de Inteligencia Artificial

2.2.4.1 Aprendizaje supervisado

Consiste en aprender a asignar datos de entrada a objetivos conocidos (también llamados anotaciones), datos un conjunto de ejemplos (a menudo anotaciones por humanos). Generalmente, casi todas las aplicaciones de Deep learning que están en el centro de atención actualmente pertenecen a esta categoría, como el

reconocimiento de carácter óptico, reconocimiento de voz, clasificaciones de imágenes y traducción de idiomas [17].

2.2.4.2 Aprendizaje no supervisado

Esta rama del Machine learning consiste en encontrar transformaciones interesantes del ingreso de datos sin la ayuda de ningún objetivo, para fines de visualización de datos, datos de compresión, eliminación de ruido de datos, o para comprender mejor las correlaciones presentes en los datos disponibles. El aprendizaje no supervisado es la base del análisis de datos, y a menudo es un paso necesario para comprender mejor un conjunto de datos antes de intentar resolver un problema de aprendizaje supervisado. La reducción de la dimensionalidad y el agrupamiento son categorías bien conocidos de aprendizaje no supervisado [17].

2.2.5 Transformada Wavelet

El análisis de la señal Wavelet se basa en una operación de tipo de correlación que incluye una propiedad de escala en términos de amplitud y extensión temporal del núcleo de correlación bajo el supuesto que la señal no es estacionaria. Proporciona una alternativa a la clásica transformada de Fourier de corta duración (la transformada de Gabor). La teoría de Wavelet ha sido desarrollada como un marco unificador para el análisis solamente de señales no estacionarias recientemente, aunque ideas similares han existido por muchos años. La idea de mirar en una señal a varias escalas y analizarlo con varias resoluciones surgió de forma independiente en muchos campos diferentes de las matemáticas, la física y la ingeniería [18].

$$W(a,\tau) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{a}} \Psi^* \left(\frac{t-\tau}{a} \right) \cdot s(t) \cdot dt \qquad (1)$$

2.3 Marco conceptual

El empleo de filtros digitales en el procesamiento de señales de electrocardiograma ha sido ampliamente utilizados y desarrollados desde el desarrollo de los amplificadores operacionales. Estos contribuyeron a mejorar y profundizar en el estudio de señales biopotenciales. Entre los diferentes tipos de filtros tenemos los analógicos y digitales teniendo entre sí varias ventajas y desventajas en su aplicación. Desde el uso de filtros activos con componentes sean activos (amplificadores operacionales, transistores, circuitos integrados, diodos, etc) y componentes pasivos (resistencias, condensadores, transformadores, etc.) en comparación con filtros con que utilizan potencia de procesamiento para la no dependencia de circuitos físicos.

Con la señal correctamente filtrada se procede a comparar con una base de datos de señales ECG del MIT-BIH en PhysioNet para el entrenamiento del algoritmo de I.A. y determinar el mejor modelo. Tomando como base el diagrama de la Figura N° 4.

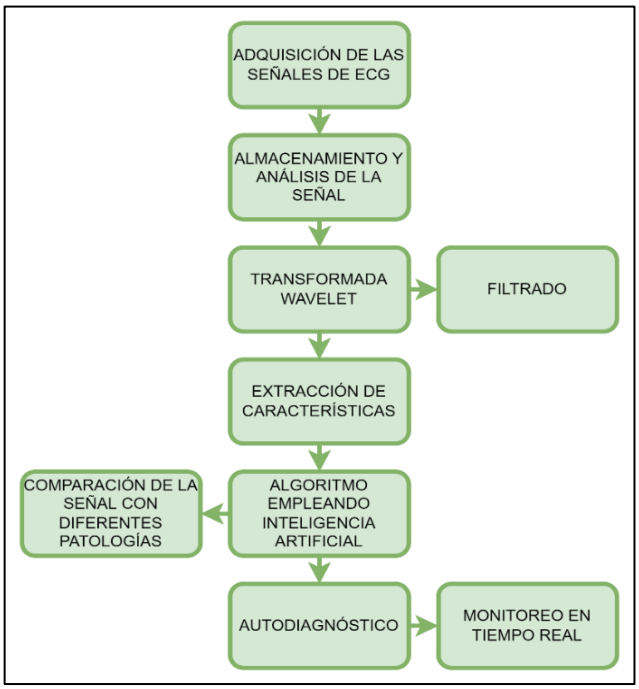


Figura N° 4. Diagrama de flujo del procesamiento de la señal de ECG.

Fuente: "Diseño e implementación de un dispositivo ECG para el autodiagnóstico de patologías mediante redes neuronales en Python y monitoreo inalámbrico en tiempo real" [19].

2.4 Definición de términos básicos

Ciclo Cardíaco

El ciclo cardiaco consiste en diástole (o relajación, que incluye la relajación isovolumínica y las fases de llenado diastólico) y sístole (o contracción, que incluye las fases de contracción y eyección isovolumínica). El ciclo cardiaco generalmente dura aproximadamente 800 ms, con la mayor parte de eso (aproximadamente 500 ms) en la fase diastólica o de relajación. La sístole (o contracción) generalmente dura unos 300 ms [20].

Conversión AD y DA

La conversión análogo-digital y digital-análogo se basa principalmente en la toma o recepción de información (datos o muestras) obtenidas de una señal de voltaje continua, una vez captada esta señal y convertida a valores discretos, permites procesarlas y almacenarlas con la finalidad de que la señal pueda ser replicada de manera fiable y exacta a la original [21]. El uso de estas conversiones es de gran utilidad para realizar procesamiento digital de señales.

Resolución del ADC

La resolución del ADC se determina por los números de bits del ADC que se utilizan para representar cada muestra de la señal, a mayor número de bits del ADC, el dispositivo será capaz de detectar pequeñas variaciones en la señal. La variedad de niveles en que se divide la señal a convertir viene dada por la ecuación de los Conversores AD y DA respectivamente [22].

$$\mathrm{ADC} = \frac{V_{IN} * (2^{n}\text{-}1)}{V_{REF}} \tag{2}$$

Espectro de frecuencia

El espectro de frecuencia de una señal es el conjunto de frecuencias que la constituyen. Además, mediante el análisis de Fourier se puede demostrar que cualquier señal está constituida por componentes

senoidales de distintas frecuencias [23]. El espectro de frecuencia nos permite obtener las señales de ruido presentes en nuestra muestra a analizar para así elegir el correcto procesamiento posterior de la misma.

$$x(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \tag{3}$$

Microprocesador

Un microprocesador es el componente fundamental de un sistema que procese información. De manera interna el microprocesador trabaja recibiendo instrucciones y procesándolas para enviarlas a su destino correspondiente. Todas las instrucciones del ordenador pasan por el microprocesador, por lo que el funcionamiento principal es determinar la velocidad a la que nuestro ordenador o sistema va a realizar las órdenes que nosotros le establecemos de acuerdo a cada tarea asignada.

Covarianza

La covarianza es una medida que nos brinda la información sobre la relación de semejanza entre dos variables. El signo que afecta a este valor nos indica si es positiva o negativa. Esto es, si es positiva, existe una correspondencia directamente proporcional; si es negativa, la correspondencia es inversamente proporcional, en caso sea cero es posible que la correspondencia pueda ser inexistente y en caso de acercarse a la unidad nos indica la cercana semejanza entre la señal original y la señal reconstruida [24].

$$Cov(X,Y) = \frac{\sum_{1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n} \tag{4}$$

Error cuadrático medio (MSE)

Este valor nos permite conocer las desviaciones cuadradas que existe entre los valores a analizar y su media, tal como muestra la Figura N° 5 como ejemplo de desviación. Esto es, nos describe una estimación

y siempre es un valor mayor que cero, el valor del error cuadrático medio disminuye a medida que tiende a cero. Está representado por la ecuación mostrada a continuación.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i)^2 \quad (5)$$

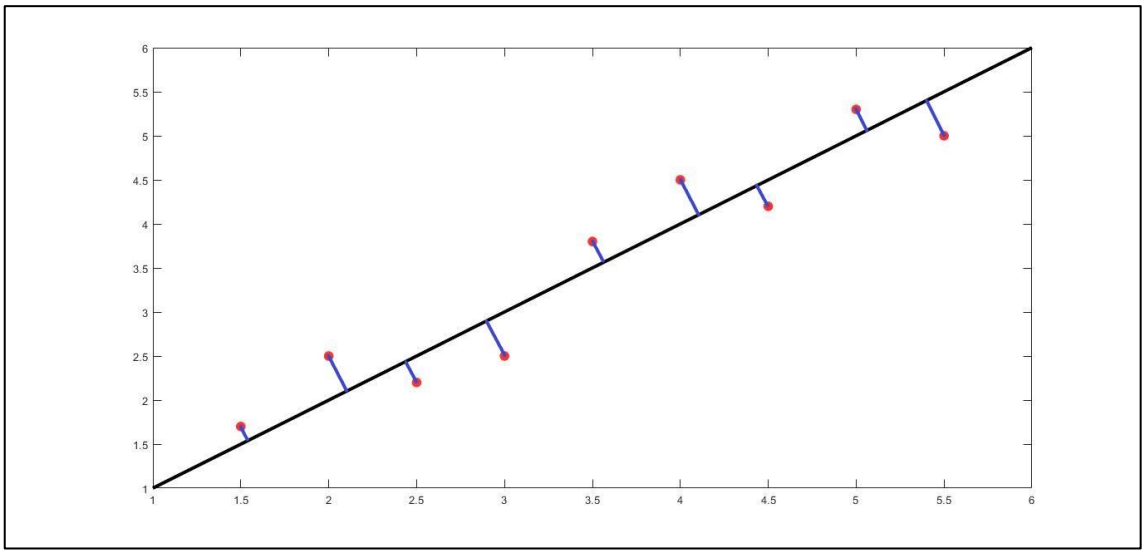


Figura N° 5. Valores cercanos a la variable estudiada (función lineal).

Fuente: Autoría propia, 2022.

Relación señal a ruido (SNR)

Este valor nos proporciona la relación de la calidad de una señal respecto al ruido que interfiere con la misma, su medida está dada en decibelios (dB), un valor por encima de la unidad nos indica que la señal prevalece sobre el ruido generado. El SNR se obtiene mediante la Transformada de Fourier.

3. HIPÓTESIS Y VARIABLES

3.1 Hipótesis

3.1.1 Hipótesis General

¿Es posible realizar el diseño y la implementación de un electrocardiógrafo con diagnóstico automático que puede reducir el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú?

3.1.2 Hipótesis Específicas

a. ¿Es posible la aplicación del método utilizado para la adquisición de datos de las señales electrocardiográficas sea el óptimo?

b. ¿Es posible que el desarrollo del algoritmo pueda influir en la prevención ante enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú?

c. ¿Es posible la contrastación del diagnóstico emitido por el algoritmo de IA será confiable comparado con uno estándar?

3.2 Definición conceptual de las variables

Variable independiente: Dispositivo electrocardiógrafo con autodiagnóstico.

Variable dependiente: Índice de mortalidad por enfermedades cardiovasculares.

3.3 Operacionalización de variables

Variable Dependiente:

- Índice de mortalidad por enfermedades cardiovasculares.

Indicadores:

- Cantidad de enfermedades que se puede detectar.
- Confiabilidad del diagnóstico.

Variable Independiente:

- Dispositivo de electrocardiógrafo que permite reducir la mortalidad por enfermedades cardiovasculares.

Indicadores:

- Portabilidad.
- Precio.
- Robustez.
- Independencia energética.

4. DISEÑO METODOLÓGICO

4.1 Tipo y diseño de investigación

Investigación explicativa, correlacional causal la esencia de este trabajo es la de resolver un problema en particular de nuestra sociedad a través de la aplicación de resultados de los conocimientos de investigaciones básicas.

4.2 Método de investigación

Cuantitativa, esto es, porque se describen las variables existentes de una determinada muestra o población. La finalidad de este método es la de evaluar y analizar posteriormente los datos para cumplir los requerimientos de la presente investigación.

4.3 Población y muestra

La población y muestra de esta investigación está conformada por personas de diferentes edades de la base de datos del PhysioNet (MIT – BIH).

4.4 Lugar de estudio

Centro de investigación “Ingeniería para la Salud”, Facultad de Ingeniería Eléctrica y Electrónica, Universidad Nacional del Callao.

4.5 Técnicas e instrumentos para la recolección de la información

La técnica empleada en la presente investigación es la de recopilación de datos secundarios e instrumentos electrónicos.

La recolección de datos fue realizada a través de los siguientes instrumentos:

a. AD8232, este sensor utilizado para mediciones de señales de electrocardiograma. Este sensor permite la extracción, amplificación y filtraje de pequeñas señales biopotenciales en condiciones de ruido. Este dispositivo tiene implementado un filtro pasa-altas y un pasa-bajas de dos polos calibrados para ayudar reducir el ruido en la señal medida, el instrumento ha formado parte de varias investigaciones anteriores

contrastando y validando el funcionamiento del mismo al momento de ser comparado con instrumentos de medición más especializados [25].

b. BASE DE DATOS DEL MIT – BIH

Las bases de datos requeridos para la realización de este proyecto se descargaron de Physionet. Se tiene que Seleccionar ECG en Waveform Databases del PhysioBank para la obtención de toda la data de señales ECG [26].

4.6 Análisis y procesamiento de datos

Se muestra en la Figura N° 6 la interfaz web de PhysioNet.

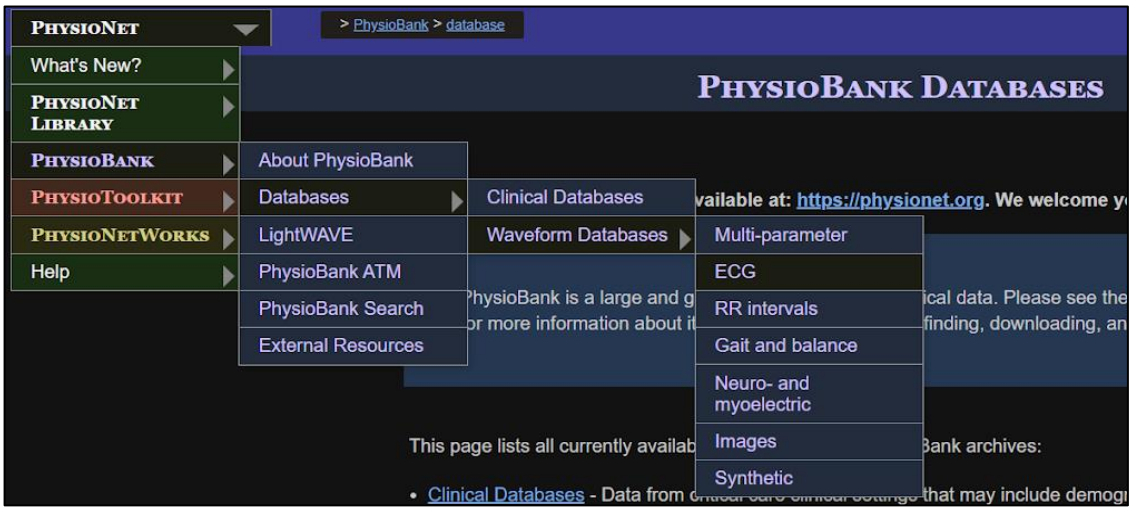


Figura N° 6. Página de la base de datos PhysioNet.

Fuente: PhysioNet, 2022.

Para tener una referencia con respecto al autodiagnóstico para las anomalías cardíacas fue necesario elegir a la Arritmia dentro de la base de datos ECG en PhysioNet, así como muestra la Figura N° 7.

Figura N° 7. Colección de archivos, MIT – BITH Arrythmia Database.

Fuente: PhysioNet, 2022.

Fue necesario descargar todos los archivos tanto las señales ECG y los datos de los pacientes mediante los siguientes comandos en la consola Command Prompt (CMD) que se muestra a continuación.

rdsamp -r mitdb/100>100.txt

rdsamp -r mitdb/500 -f 5:0 -t 10:30 -p -v>500.txt

Se obtuvo una carpeta con los siguientes archivos con la data que se necesita para su posterior análisis (ver Figura N° 8).

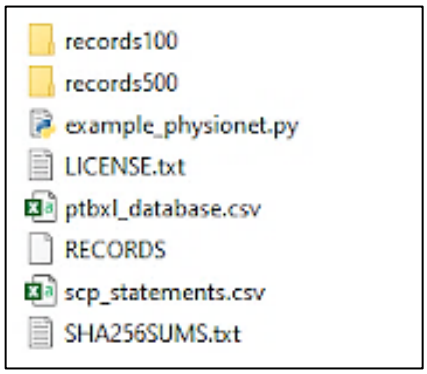


Figura N° 8. Archivos descargados con el comando ejecutado en CMD.

Fuente: Autoría propia, 2022.

Las grabaciones de los datos de las señales ECG de cada paciente se almacenan en el formato WaveForm DataBase (WFDB), por ello fue necesario realizar un código para adaptarlo al formato que podamos visualizar.

```
import pandas as pd

import numpy as np

import wfdb

import ast

def load_data(df, sampling_rate, path):

    if sampling_rate == 100:

        data = [wfdb.rdsamp(path+f) for f in alldata_clean.filename_lr]
```

```
    else:
        data = [wfdb.rdsamp(path+f) for f in alldata_clean.filename_hr]
    data = np.array([signal for signal, meta in data])
    return data
path = 'database2/'
sampling_rate=100
X = load_data(alldata_clean, sampling_rate, path)
```

La data el ECG viene comprimida y dividida en 2 archivos con formatos distintos como .dat y .hea (ver Figura N° 9).

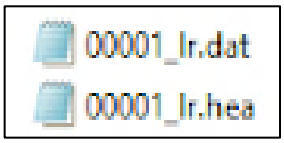


Figura N° 9. Archivos comprimidos con formatos .dat y .hea.

Fuente: Autoría propia, 2022.

Para elegir y mostrar la primera tabla de toda la base de datos se realizó la siguiente codificación, el cual nos permite organizar los datos para una mayor facilidad al combinar con otras tablas. Se obtuvo la Tabla N° 1 con dimensiones 9352x1000 donde el número de filas indica la cantidad de pacientes por analizar y las columnas, cantidad de datos según el tiempo de muestreo:

```
X_df=pd.DataFrame(X[0])
X[0][:,1]
len(X[0][:,1]) //1000
X.shape //9352,1000,12)
a=[]
a=np.append(a,X[0][:,1])
```

```
len(a)  //1000

b=a.ravel()

b.reshape((2,1000))

c=[]

for i in range(0,9351+1):

    c=np.append(c,X[i][:,10], axis=0)

c.shape //9352000

d=c.ravel()

df10=pd.DataFrame(d.reshape((9352,1000)))

df10=df10.T
```

TABLA N° 1

Base de datos extraída de los archivos .dat y .hea.

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	
0	-0.039	-0.034	-0.029	-0.022	-0.018	-0.012	-0.007	-0.001	0.005	0.009	...	0.166	0.138	0.087	0.037	-0.003	-0.025	-0.036	-0.
1	0.083	0.057	0.013	-0.006	-0.008	-0.012	-0.016	-0.019	-0.024	-0.027	...	-0.181	-0.229	-0.219	-0.104	0.686	1.656	0.511	-0.
2	-0.076	-0.079	-0.066	-0.088	-0.021	0.097	0.389	0.696	0.182	-0.175	...	0.021	0.007	0.024	-0.010	0.041	-0.013	0.095	0.2
3	-0.184	-0.185	-0.193	-0.196	-0.198	-0.202	-0.204	-0.207	-0.212	-0.210	...	0.731	-0.675	-0.456	-0.082	0.147	0.184	0.164	0.2
4	0.135	0.134	0.130	0.127	0.122	0.119	0.115	0.109	0.106	0.101	...	-0.171	-0.170	-0.170	-0.165	-0.164	-0.167	-0.171	-0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9347	-0.018	0.005	0.047	0.075	0.071	0.064	0.058	0.051	0.050	0.050	...	0.050	0.050	0.051	0.048	0.046	0.044	0.050	0.0
9348	0.040	0.048	0.053	0.044	0.037	0.018	-0.002	-0.009	-0.016	-0.022	...	-0.070	-0.070	-0.089	-0.118	-0.120	-0.122	-0.142	-0.
9349	-0.323	-0.244	-0.133	-0.078	-0.066	-0.063	-0.058	-0.054	-0.049	-0.044	...	-0.031	-0.028	-0.024	-0.016	-0.012	-0.006	0.001	0.0
9350	0.084	0.074	0.062	0.054	0.045	0.036	0.024	0.013	-0.010	-0.032	...	-0.065	-0.067	-0.057	-0.074	-0.127	-0.165	0.024	0.1
9351	-0.235	-0.234	-0.237	-0.254	-0.251	-0.244	-0.240	-0.232	-0.230	-0.230	...	-0.141	-0.130	-0.146	-0.108	-0.175	-0.130	0.047	0.7

Fuente: Autoría propia, 2022.

5. RESULTADOS

5.1 Resultados Descriptivos

Diseño de la fuente de alimentación del dispositivo

Se propuso para la alimentación general del dispositivo en primera instancia dos tipos de fuentes diferentes. Estas fuentes de alimentación permiten brindar la energía necesaria para que pueda funcionar el Raspberry PI 3B+, el AD8232, el ADS1115 y la pantalla TFT LCD de interfaz gráfica de 2.4”. El consumo promedio del dispositivo con estos componentes mencionados oscila entre los 2.5 y 3 amperios. La potencia consumida por el dispositivo no sobrepasa los 13 Watts como muestra la TABLA N° 2. Tomando aquello como referencia y en base a los consumos máximos según indican sus hojas de datos se procedió a diseñar las fuentes de alimentación para las cargas de baterías con el amperaje necesario para el dispositivo.

TABLA N° 2

Valores máximos de consumos de los componentes del dispositivo.

	Volts	Amps	Potencia (W)
RASPBERRY PI 3B+	5 V	2.5 A	12.5
PANTALLA TFT 2.4"	3.3 V	80 mA	0.264
AD8232	3.3 V	230 uA	0.000759
ADS1115	3.3 V	300 uA	0.00099
TOTAL			12.765749

Fuente: Autoría propia, 2022.

Fuente lineal con el LM7809

La fuente lineal fue realizada con una etapa de adecuamiento de la señal de 220VAC, se colocó un fusible de 4A a la entrada del transformador de 220/12VAC y un switch de encendido y apagado respectivamente.

La señal de salida del transformador en voltaje AC se pasó a través de un puente de diodos rectificador para después atenuar el rizado y estabilizar

la señal a través de dos capacitores C1 y C2. El voltaje obtenido a la salida de esta etapa pasa al integrado LM7809, este integrado tiene la limitación de trabajar a 1A según su hoja de datos. Es por ello que se le adecuó una etapa de ganancia de corriente de hasta 5A con un transistor en configuración PNP y se le añadió una protección contra corrientes parásitas a través del diodo D1 para que no perjudique el ciclo de trabajo del circuito LM7809, el circuito se muestra en la Figura N° 10.

La fuente de alimentación presentó pérdidas de potencia con el circuito LM7809, siendo no ideal para su implementación al tratar de optimizar el consumo de potencia para el dispositivo.

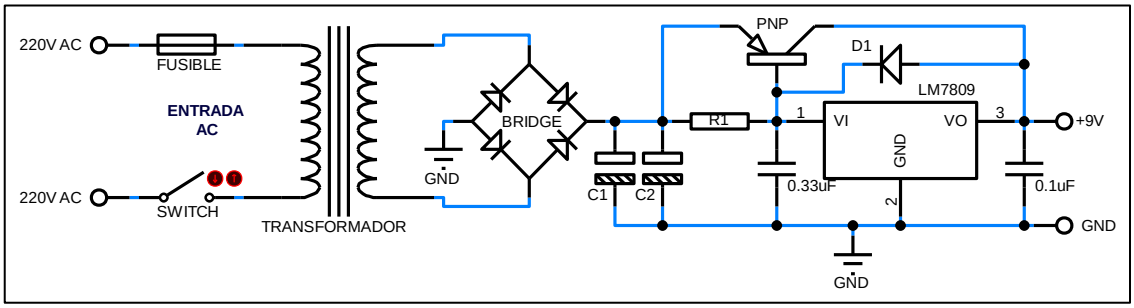


Figura N° 10. Fuente de alimentación con el integrado LM7809.

Fuente: Autoría propia, 2022.

Fuente de alimentación con el LM2596

La fuente desarrollada con el LM2596, este integrado permite manejar cargas de hasta 3A, pero en caso existan picos de corriente de consumo del dispositivo se le añadió la etapa de ganancia de corriente con un transistor en configuración PNP y un diodo de protección para el circuito integrado LM2596.

El circuito LM2596, al ser un regulador Switching Step – Down, permite una mejor eficiencia energética respecto al integrado LM7809. En el diseño de la presente fuente se utilizó el diodo Schottky 1N5284 de acuerdo al amperaje de consumo en la hoja de datos para el voltaje determinado de alimentación tanto para el circuito BMS como para el Raspberry Pi 3B+ [27]. El circuito se muestra en la Figura N° 11.

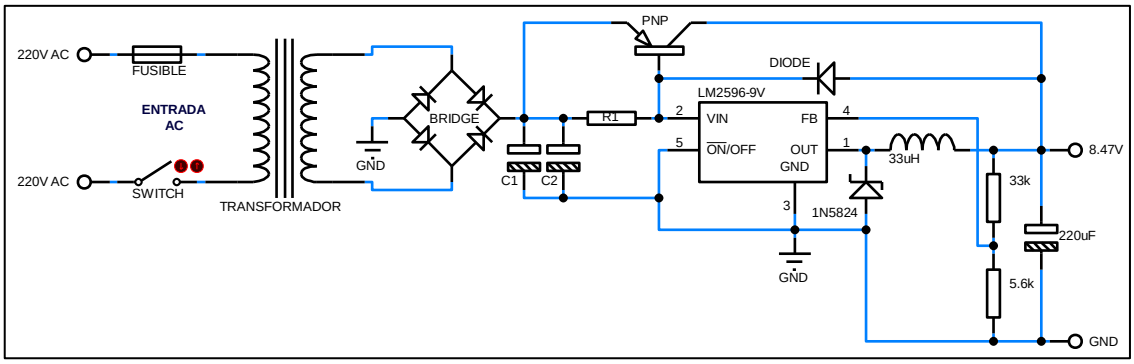


Figura N° 11. Fuente de alimentación con el integrado LM2596.

Fuente: Autoría propia, 2022.

ETAPA DE FUNCIONAMIENTO CON BATERÍAS

Se añadió una etapa para el funcionamiento del dispositivo con baterías de 3.7 V utilizando un circuito integrado BMS. Este circuito se le añadió una combinación de switches que le permite cargar las baterías o utilizarlas para que alimenten al dispositivo ECG. El switch 1C alimenta al Rasberry PI 3B+ con un voltaje aproximado de 4.92 V cuando se conecta con el pin 3C, siendo el circuito final el mostrado en la Figura N° 12.

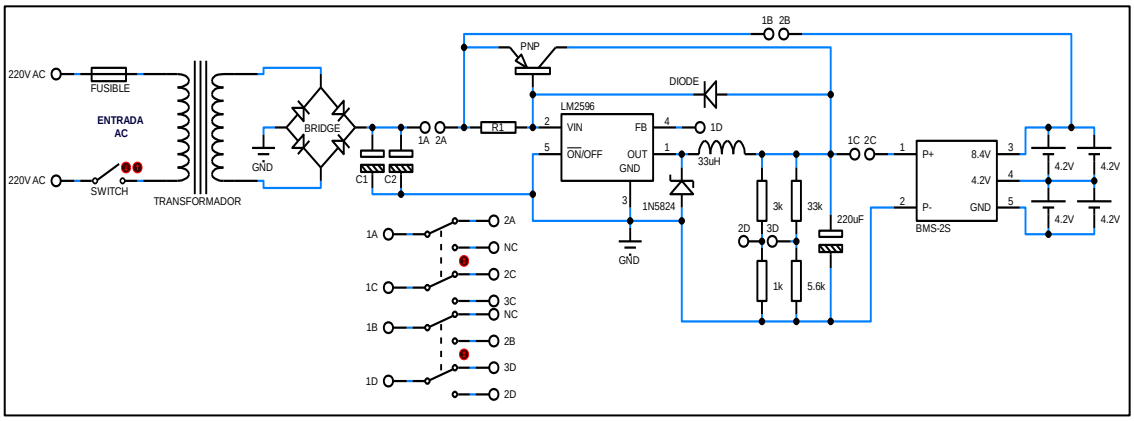


Figura N° 12. LM2596 en configuración para carga de baterías o uso del Raspberry PI3B+.

Fuente: Autoría propia.

ETAPA DE ADQUISICIÓN Y TRANSICIÓN DE LA SEÑAL DE ECG

Para la etapa de adquisición de la señal de ECG se acopló la salida de señal del AD8232 al ADS1115, este último circuito integrado es un ADC de 16 bits de tipo delta-sigma de precisión y bajo consumo que opera mediante interfaz I2C. Posee un muestreo a alta frecuencia y alta rapidez

de conversión gracias al oscilador interno de 1 MHz [28]. Los electrodos utilizados se conectan directamente al AD8232 a través de un conector tipo plug de 3.5mm, mostrado en la Figura N° 13.

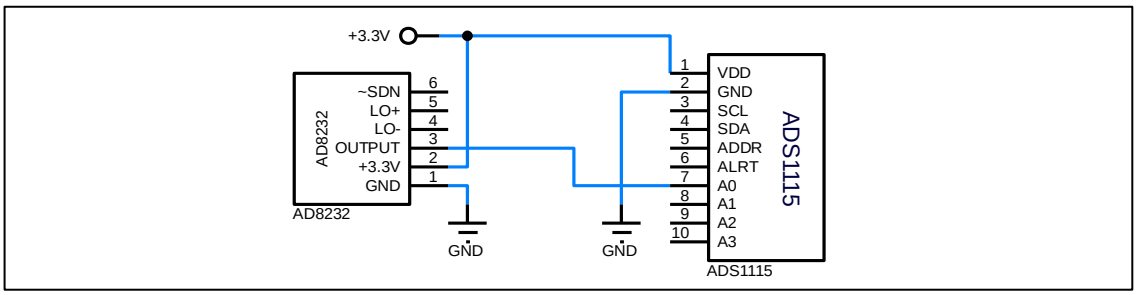


Figura N° 13. Etapa de acoplamiento de la señal del AD8232 y el ADS1115.

Fuente: Autoría propia, 2022.

ETAPA DE CONEXIÓN DEL ADS1115 AL RASPBERRY PI 3B+

La Figura N° 14 muestra la etapa de conexión en la que se le añadieron resistencias para polarizar en una configuración pull-up al bus I2C del ADS1115 y para regular los niveles lógicos de envío de la señal hacia el Raspberry PI 3B+.

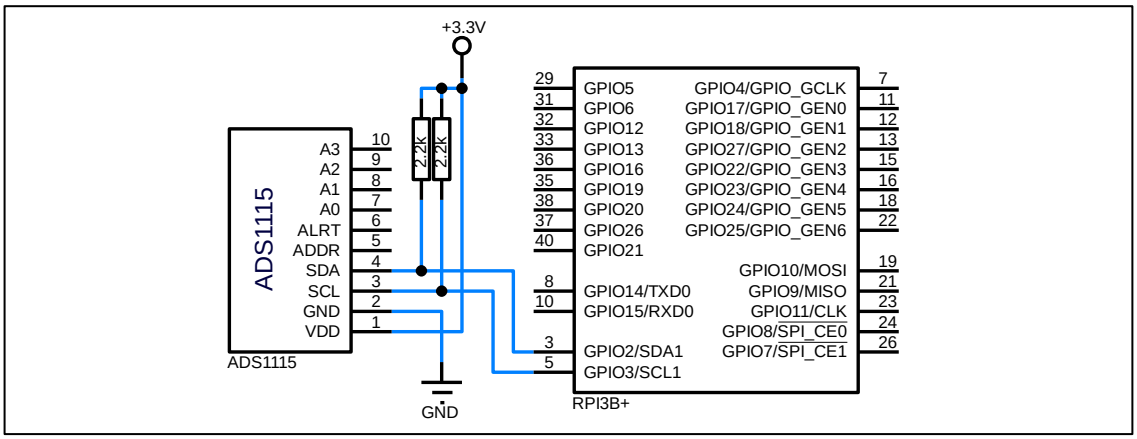


Figura N° 14. Etapa de acoplamiento del ADS1115 al Raspberry PI 3B+.

Fuente: Autoría propia, 2022.

ETAPA DE INTERFAZ GRÁFICA (GUI)

La etapa final del dispositivo es la conexión del Raspberry PI 3B+ con la pantalla TFT 2.4” para la muestra de la señal con diagnóstico y la interfaz de usuario. La pantalla TFT 2.4” viene como un módulo acoplable que redujo el tamaño de implementación del dispositivo.

El circuito de conexión se aprecia en la Figura N° 15.

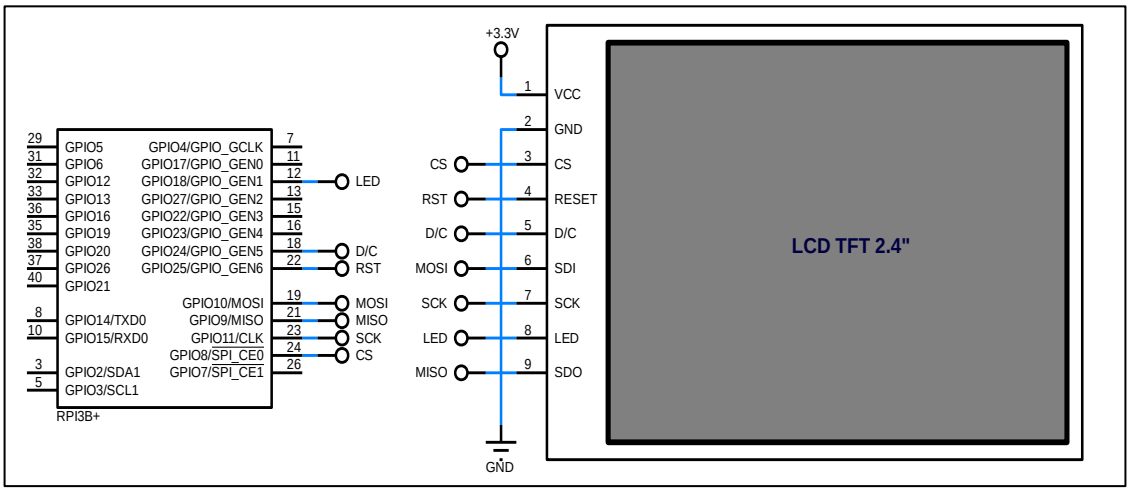


Figura N° 15. Etapa de conexión con pantalla TFT 2.4".

Fuente: Autoría propia. 2022.

Los datos que se obtuvieron directamente del dispositivo, operado por el AD8232 que es un sensor que permite las obtener señales analógicas de ECG para su posterior procesamiento. En la Figura N° 16 se visualiza la señal tomada del sensor, se ingresa con ruido debido a que la alimentación del sensor proviene de la red eléctrica el cual se introduce con un ruido de 60 Hz. Además del ruido proveniente del movimiento del paciente, y de la calidad de los electrodos, también del tipo de cable usado para la conexión de los electrodos.

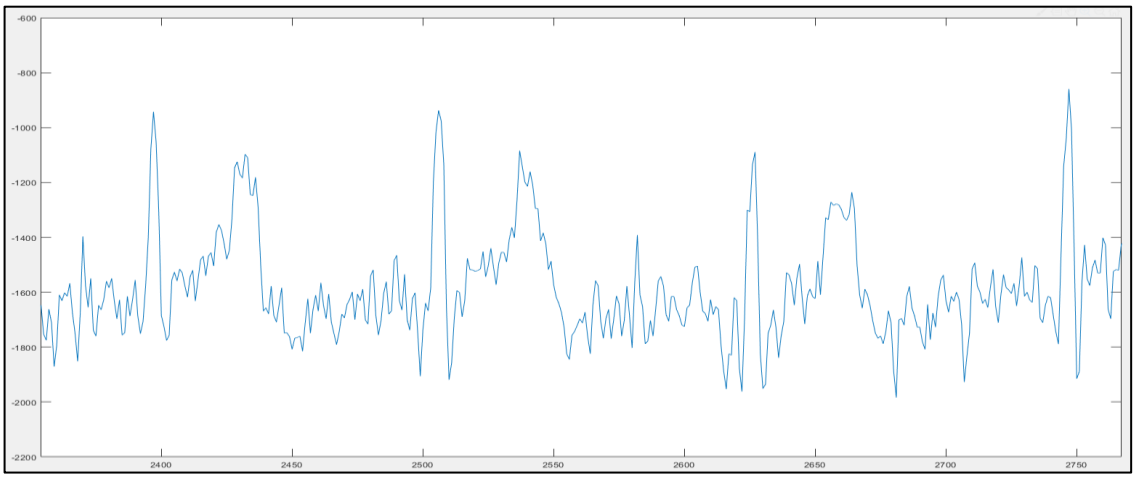


Figura N° 16. Señal sin procesar del AD8232.

Fuente: Autoría propia, 2022.

Se utilizó el sensor AD8232 debido al fácil uso e implementación en nuestro diseño, así como las ventajas que ofrece de acuerdo a sus especificaciones técnicas. Este sensor a su vez permite la reducción pasiva del ruido presente en la señal analizada, se decantó también por la elección de este sensor es por el amplio rango de temperatura que el sensor resiste, desde -40°C hasta +85°C [25].

La señal ECG del paciente se obtuvo evitando el uso una fuente de alimentación conectada a la red eléctrica, ver Figura N° 17, en su caso, se emplearon baterías y se guardó la señal obtenida como referencia para poder realizar el análisis y la comparación de la efectividad y eficacia de los filtros que se utilizaron para así determinar cuál de ellos se ajusta más a nuestras necesidades para la obtención de señales fiables y válidas para su posterior análisis.

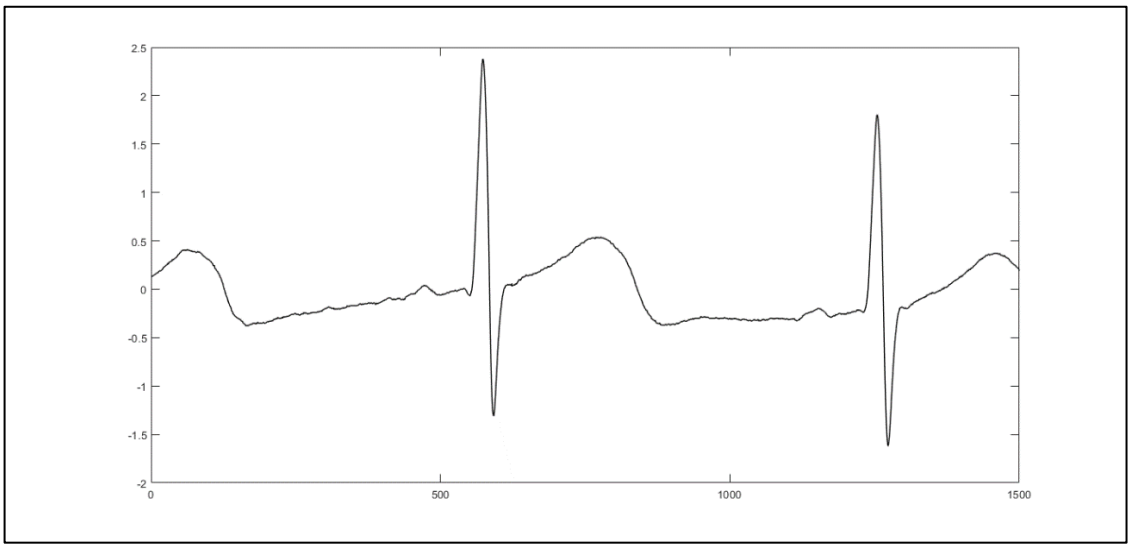


Figura N° 17. Señal AD8232 en el ambiente controlado.

Fuente: Autoría propia, 2022.

Para la elección adecuada del filtro que reduzca el ruido de 60 Hz como se aprecia en la Figura N° 18, que proviene de la red eléctrica y sin procesar, esta señal fue analizada para reducir su presencia en la señal original de ECG.

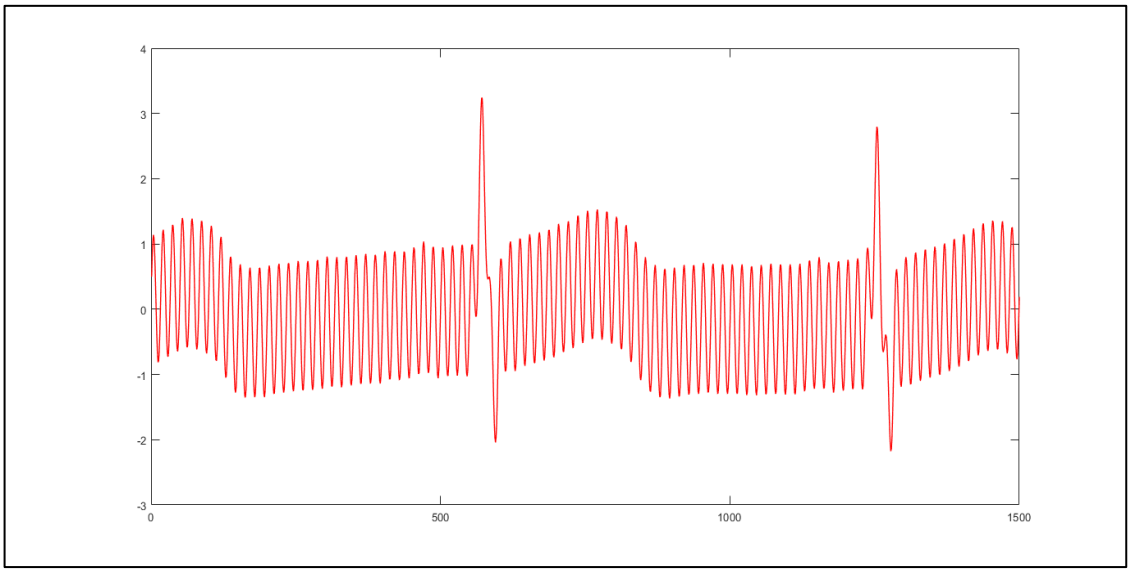


Figura N° 18. Señal del ECG con ruido de 60 Hz.

Fuente: Autoría propia, 2022.

La Transformada Rápida de Fourier (FFT) nos indica la señal senoidal de 60 Hz predominante en nuestra muestra de ECG, tal como se muestra en la Figura N° 19. Esta señal de electrocardiograma no nos permite poder analizar correctamente los diferentes tipos de ondas presentes como las ondas P, complejo QRS y la onda T.

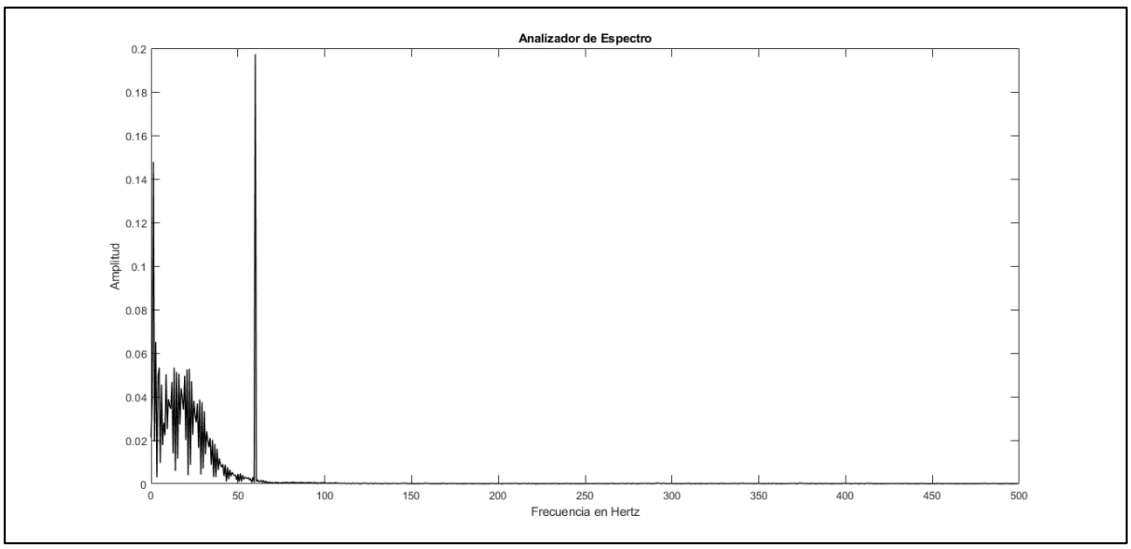


Figura N° 19. Transformada de Fourier para la señal de ECG con ruido de 60 Hz.

Fuente: Autoría propia, 2022.

Para el caso de los filtros analógicos, se estudiaron el impacto de la aplicación de ellos en nuestra señal para su posterior uso.

Por cada filtro se realizó una tabla con los valores del SNR (Relación señal a Ruido), Error Cuadrático Medio y la Covarianza para analizar qué tan cercana es la señal de salida de nuestro filtro implementado con la señal obtenida en el ambiente controlado sin la presencia de ruido de la red alterna. Esta señal fue sometida a diferentes tipos de filtros como se verá a continuación para elegir el que más se adapta y favorece en el análisis de la señal en esta presente investigación.

A continuación, se muestra el análisis realizado para cada tipo de filtro, estos fueron analizados en el programa MATLAB en la etapa de prueba para luego ser implementados en el Raspberry Pi 3B+ de manera integrada.

FILTRO BUTTERWORTH

Este filtro tiene una gran variedad de aplicaciones, entre ellas dentro el filtrado de señales biopotenciales, la ventaja de este filtro es la de tener una respuesta plana a medida que se incrementa el orden, esto es, la oscilación inicial en la banda pasante o de rechazo no es muy pronunciada. Para el filtro Butterworth se muestran los siguientes valores del SNR, MSE y la covarianza en la TABLA N° 3.

TABLA N° 3

Valores del SNR, MSE y la Covarianza para un filtro Butterworth rechaza-banda.

VALORES SNR				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
BUTTER-RB	13.3126558	13.7968981	11.5636649	9.67017891
ERROR CUADRÁTICO MEDIO				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
BUTTER-RB	0.08871309	0.08392785	0.10853455	0.13497151
COVARIANZA				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
BUTTER-RB	0.97690572	0.97920998	0.96521445	0.94618737

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza óptimos del filtro en la Figura N° 20.

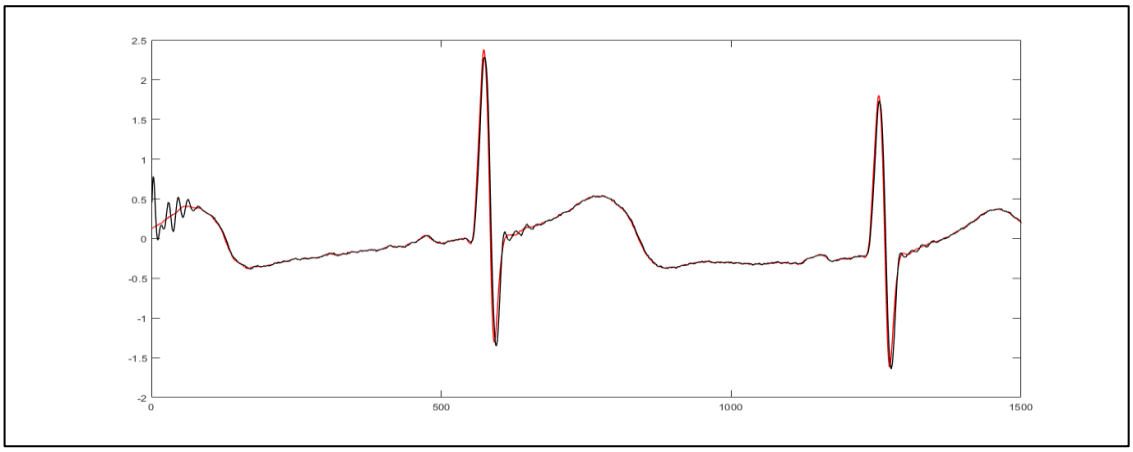


Figura N° 20. Señal del ECG de 60 Hz con filtro Butterworth de 4to Orden.

Fuente: Autoría propia, 2022.

Se observó que la señal filtrada tiene una leve oscilación inicial al inicio del filtrado y una atenuación respecto a la señal original.

FILTRO CHEBYSHEV

Los filtros Chebyshev, sea de Tipo I, el cual tienen más ripple en la banda de paso o de Tipo II, el cual posee más ripple en la banda de rechazo [29].

Para el caso del filtro de Chebyshev se optó por uno de tipo II y se tomaron muestras del filtro hasta un orden 8 y los valores obtenidos se muestran la TABLA N° 4 con los datos del SNR, MSE y la covarianza, respectivamente.

TABLA N° 4

Valores del SNR, MSE y la Covarianza para un filtro Chebyshev rechaza-banda.

VALORES SNR				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CHEBY-RB	13.3071601	9.93375973	8.52558664	7.76073414
ERROR CUADRÁTICO MEDIO				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CHEBY-RB	0.08876917	0.13093065	0.15398231	0.16815403
COVARIANZA				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CHEBY-RB	0.97687936	0.98225376	0.92759844	0.92966233

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza óptimos del filtro Chebyshev en la Figura N° 21.

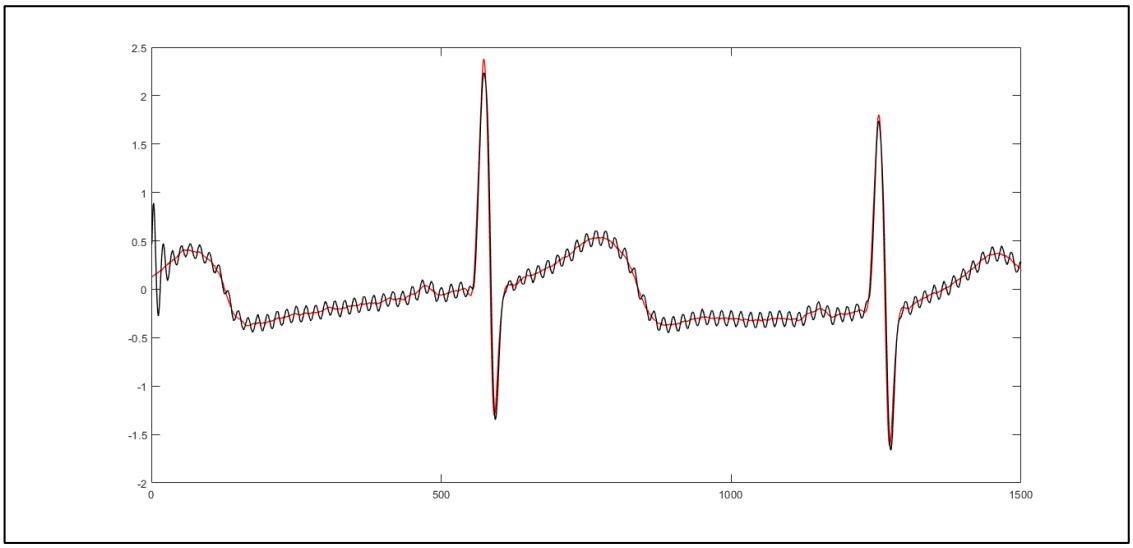


Figura N° 21. Señal del ECG de 60 Hz con filtro Chebyshev de 2do Orden.

Fuente: Autoría propia, 2022.

Se observó que la señal presenta mucho ruido oscilatorio cuando se aplica el filtro de orden 2, con un Ripple de 3 dB, para filtros Chebyshev de mayor orden tal cual se apreció en la TABLA N° 4, no son recomendados debido a los valores obtenidos se alejan de la señal deseada.

FILTRO ELÍPTICO O DE CAUER

Este tipo de filtro no requiere de un orden superior para satisfacer las necesidades de diseño, pero tienen una transición más oscilatoria en la banda de paso. [30]

En el filtro elíptico o de Cauer, se muestra que el filtro ideal a utilizar sería el de 6to orden, siendo no viable para su implementación debido a la gran cantidad de materiales y componentes que se deben utilizar que se traduce en un mayor costo de elaboración del dispositivo.

Los valores correspondientes al SNR, MSE y la Covarianza se muestran en la TABLA N° 5.

TABLA N° 5

Valores del SNR, MSE y la Covarianza para un filtro Cauer rechaza-banda.

VALORES SNR				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CAUER-RB	13.3071601	-0.1094675	13.5400232	-0.1098408
ERROR CUADRÁTICO MEDIO				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CAUER-RB	0.08876917	0.41612297	0.08642872	0.4161408
COVARIANZA				
ORDEN	2(2n)	4(2n)	6(2n)	8(2n)
CAUER-RB	0.97687936	0.58588849	0.97788627	0.58587884

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza óptimos del filtro Cauer en la Figura N° 22.

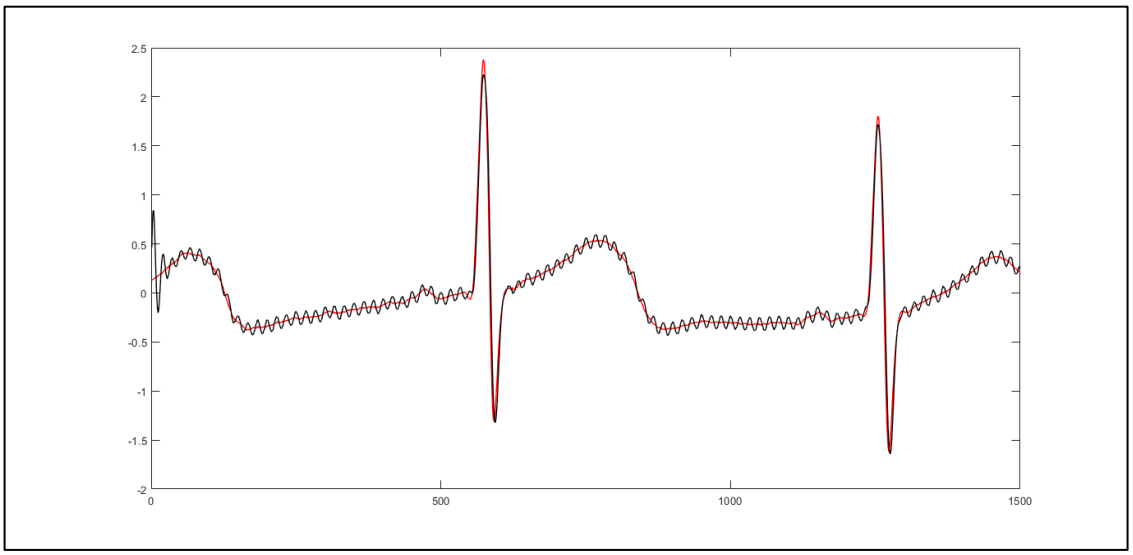


Figura N° 22. Señal del ECG de 60 Hz con filtro de Cauer de 6to Orden.

Fuente: Autoría propia, 2022.

La señal presenta oscilaciones sobre la señal original de forma similar al filtro de Chebyshev, El filtro se implementó con un Ripple de 3 dB y una atenuación en la banda rechazada de 5 dB.

FILTRO NOTCH

Este tipo de filtro es ampliamente utilizado en técnicas de filtrado de señales ECG, generalmente para filtras señales de interferencia de la red eléctrica. [31]

La implementación del filtro Notch nos brindó los siguientes valores del SNR, MSE y la covarianza, mostrados en la TABLA N° 6.

TABLA N° 6

Valores del SNR, MSE y la covarianza para un filtro Notch con Q = 4.

VALORES SNR								
Q	1	2	3	4	5	6	7	8
NOTCH-RB	10.4258094	14.2902967	15.4938063	15.5365244	15.1604697	14.6633751	14.1522005	13.6630694
ERROR CUADRÁTICO MEDIO								
Q	1	2	3	4	5	6	7	8
NOTCH-RB	0.12372551	0.0792932	0.06903369	0.06869507	0.07173458	0.07595974	0.08056423	0.08523126
COVARIANZA								
Q	1	2	3	4	5	6	7	8
NOTCH-RB	0.9534846	0.98121844	0.9858829	0.98611986	0.98495653	0.9832223	0.98121798	0.97907383

Fuente: Autoría propia, 2022.

La señal procesada con un Q=4 en MATLAB se muestra a continuación en la Figura N° 23.

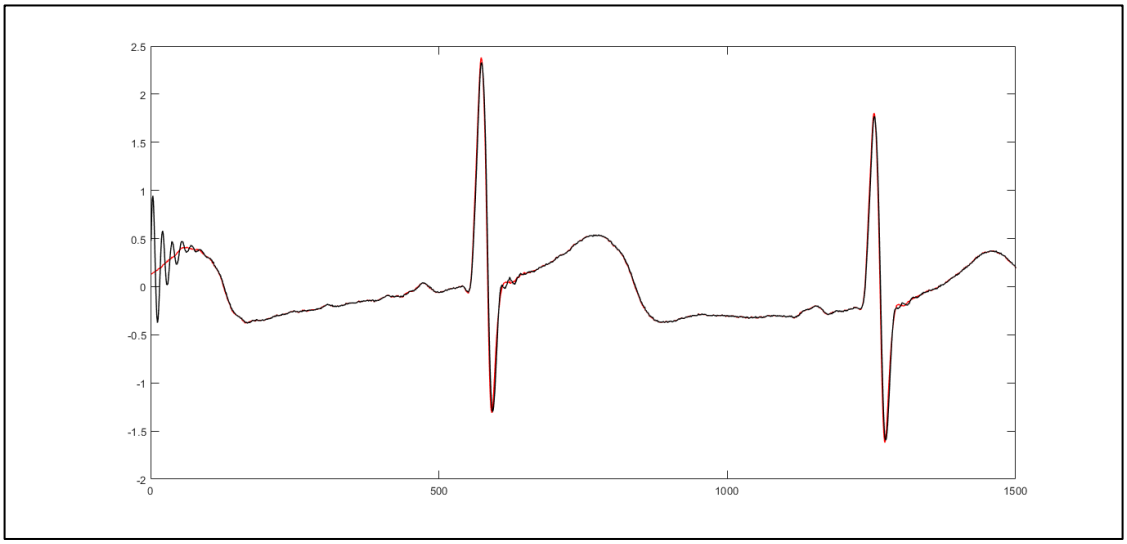


Figura N° 23. Señal del ECG de 60 Hz con filtro Notch con un Q = 4.

Fuente: Autoría propia, 2022.

Se observó valores favorables para el filtro Notch, con lo apreciado según la TABLA N° 6, presenta una fuerte atenuación a la señal de 60 Hz de la red eléctrica además de adquirir de manera correcta la señal del ECG.

A continuación, se muestra la Transformada de Fourier para el filtro Notch en la Figura N° 24.

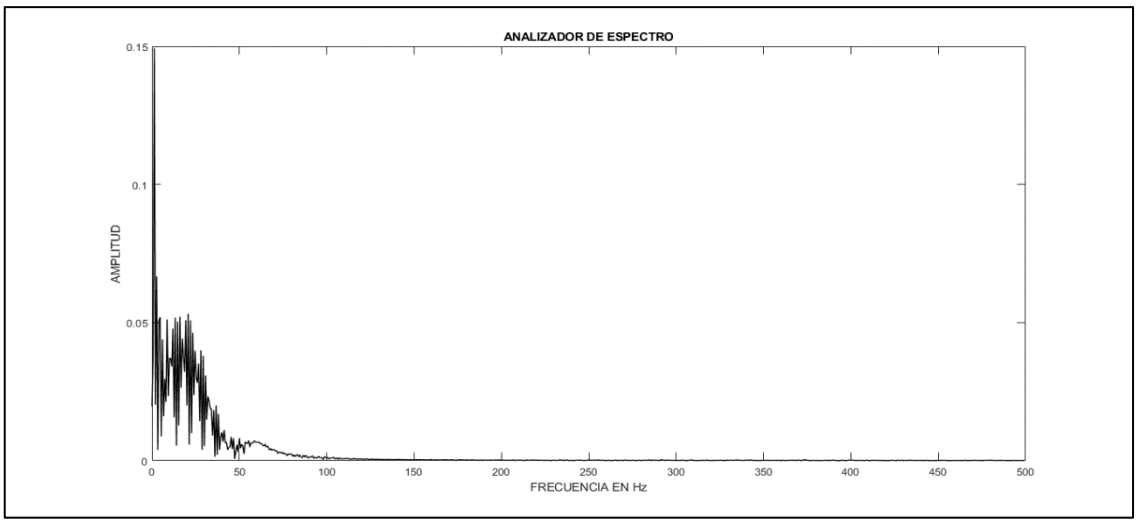


Figura N° 24. Transformada Rápida de Fourier para el filtro Notch con un Q = 4.

Fuente: Autoría propia, 2022.

Se observó la FFT y se apreció la reducción se la componente de 60 Hz.

A continuación, se muestra el análisis para los filtros digitales.

FILTRO VENTANA DE BLACKMAN

Se notó un incremento sustancial al aplicar los filtros digitales, en el caso del filtro de Blackman, el orden ideal para filtrar la señal de ECG es de orden 300 como muestra la TABLA N° 7.

TABLA N° 7

Valores del SNR, MSE y la covarianza para un filtro Blackman rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
BLACKM-RB	-4.916437697	-4.611857171	-0.3968586	5.882534015	21.181352	22.60279046	22.52182669	21.51818541
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
BLACKM-RB	0.72426723	0.700101293	0.432040954	0.210147567	0.036477438	0.031525172	0.032371335	0.032384309
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
BLACKM-RB	0.482025512	0.494231432	0.669407021	0.888315613	0.996163838	0.997231032	0.99718426	0.996453344

Fuente: Autoría propia, 2022.

La señal procesada con un N = 300 en del filtro de Blackman en MATLAB se muestra a continuación en la Figura N° 25.

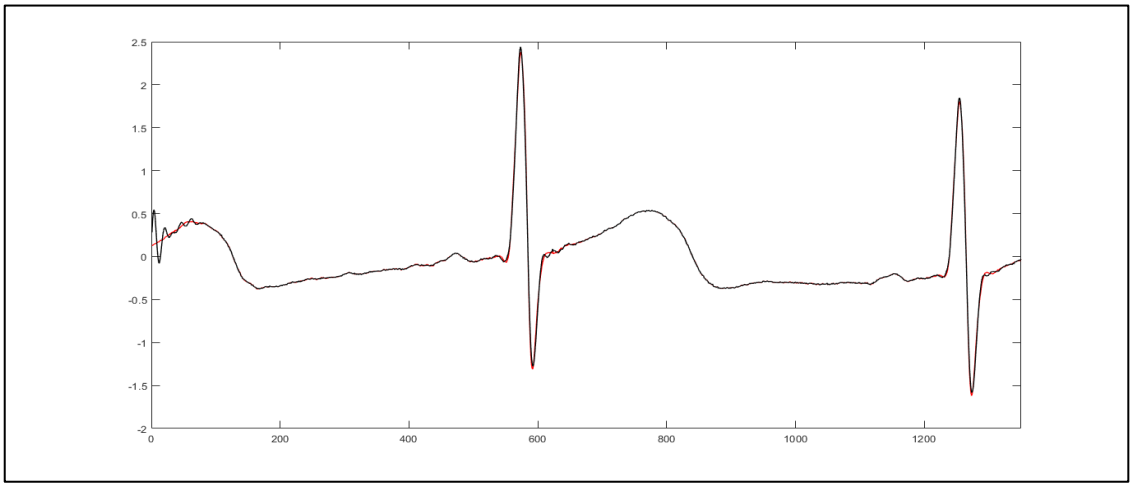


Figura N° 25. Salida del filtro de Blackman de orden N = 300.

Fuente: Autoría propia, 2022.

La señal se observó con menor rizado inicial respecto a los filtros analógicos.

FILTRO VENTANA DE GAUSS

Este tipo de ventana tiene un mejor desempeño cuando se aplica para el suavizado de la señal de ECG, el complejo QRS presenta mayor fiabilidad, pues esta ventana representa la señal correctamente en las respectivas pendientes de la señal [32].

Para el filtro de Gauss se obtuvo los siguientes valores mencionados en la TABLA N° 8.

TABLA N° 8

Valores del SNR, MSE y la covarianza para un filtro Gauss rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
GAUSS-RB	-4.949784686	-4.043468784	0.650081086	8.293147709	21.87646606	22.61194827	22.40512016	21.50848185
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
GAUSS-RB	0.727053191	0.655754773	0.383028421	0.159218557	0.033671978	0.031491951	0.032809222	0.032420508
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
GAUSS-RB	0.480688798	0.51582878	0.721536478	0.931393685	0.996731333	0.997236382	0.997108423	0.996445101

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza óptimos del filtro ventana de Gauss en la Figura N° 26.

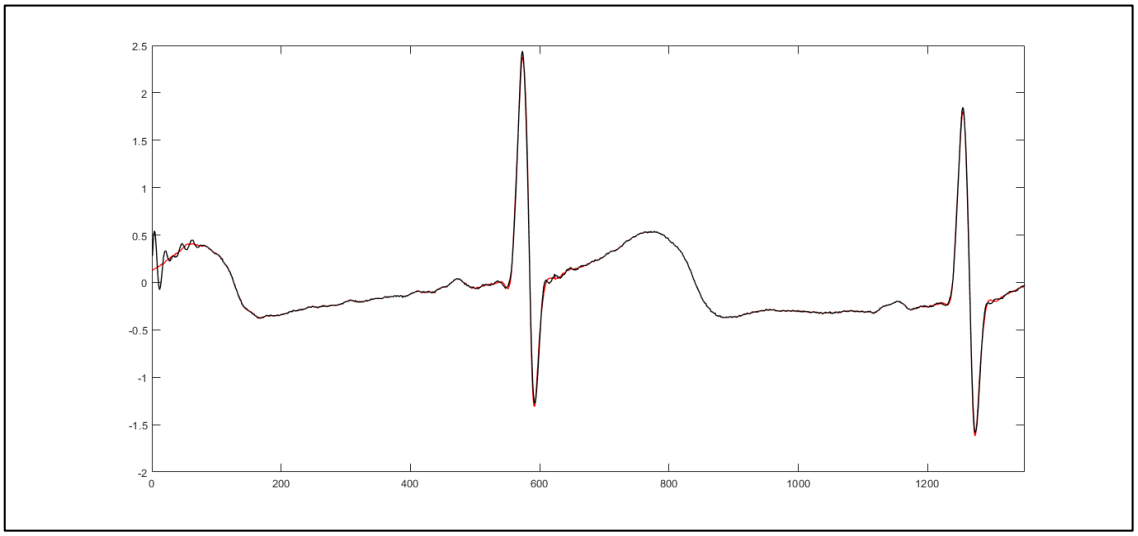


Figura N° 26. Salida del filtro de Gauss de orden N = 300.

Fuente: Autoría propia, 2022.

Filtro con Ventana de Hamming

Para el filtro de Hamming se obtuvo los siguientes valores mencionados en la TABLA N° 9.

TABLA N° 9.

Valores del SNR, MSE y la covarianza para un filtro Hamming rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
HAMM-RB	-4.969356664	-3.77952817	1.380890189	10.3660377	22.75473219	22.56415843	22.32595937	21.48202136
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
HAMM-RB	0.728693312	0.636127879	0.35211988	0.125414914	0.03043374	0.031665698	0.033109603	0.032519423
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
HAMM-RB	0.479902028	0.525986716	0.751146738	0.955919166	0.997330534	0.99720548	0.997055986	0.99642312

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza óptimos del filtro ventana de Hamming en la Figura N° 27.

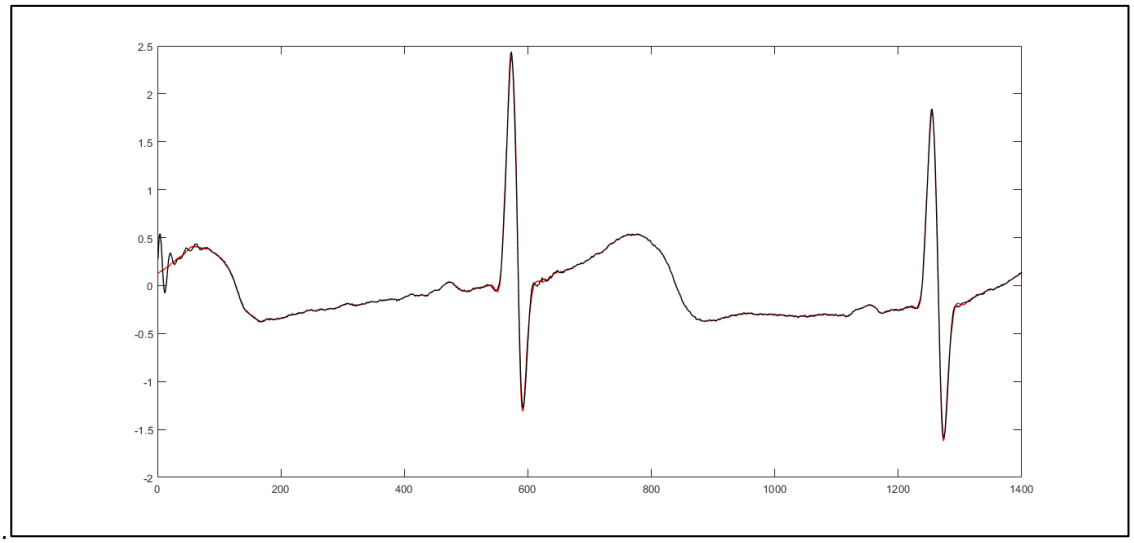


Figura N° 27. Salida del filtro de Hamming de orden N = 200

Fuente: Autoría propia, 2022.

Filtro con Ventana de Kaiser

Para el filtro de Kaiser se obtuvo los siguientes valores mencionados en la TABLA N° 10.

TABLA N° 10

Valores del SNR, MSE y la Covarianza para un filtro Kaiser rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
KAISER-RB	-4.903569206	1.315202801	12.80010182	10.70286475	14.98035041	18.87057679	18.80536125	19.96704155
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
KAISER-RB	0.723194993	0.353915672	0.094576468	0.120664785	0.074485959	0.04844714	0.049657389	0.03871612
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
KAISER-RB	0.482348448	0.728686701	0.974440874	0.958962704	0.984405919	9.935E-01	0.9934378	0.994932296

Fuente: Autoría propia, 2022.

Como se observa en la Figura N° 28 la señal senoidal aún se encuentra presente en la señal de ECG, no permitiendo el análisis correcto de la misma, pese a ser de orden N = 500. Esto determina que, para filtrar correctamente la señal, al usar ese filtro se tendrá más uso de procesamiento.

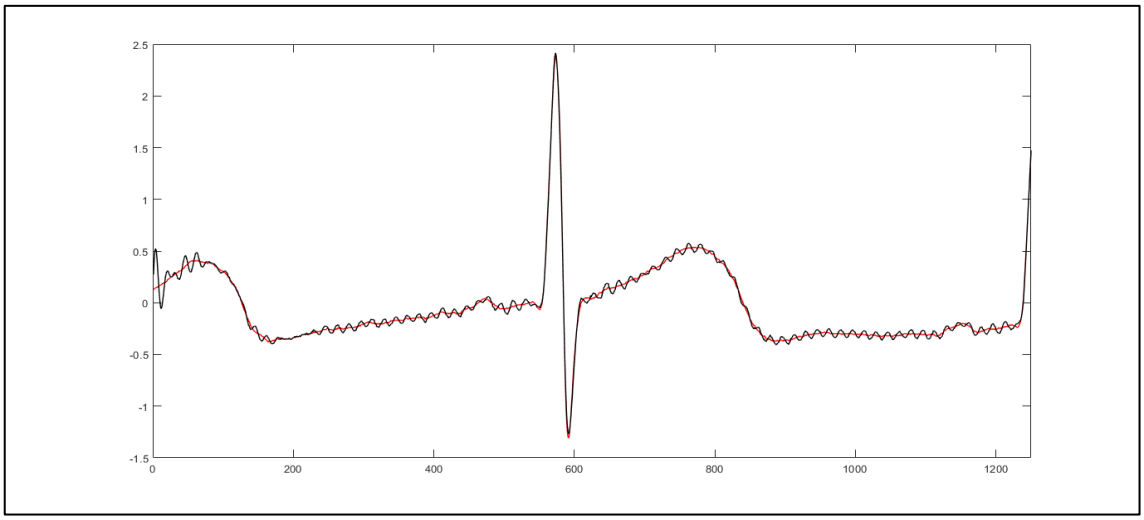


Figura N° 28. Salida del filtro de Kaiser de orden N = 500.

Fuente: Autoría propia, 2022.

A continuación, se muestra la Transformada de Fourier para el filtro Kaiser en la Figura N° 29.

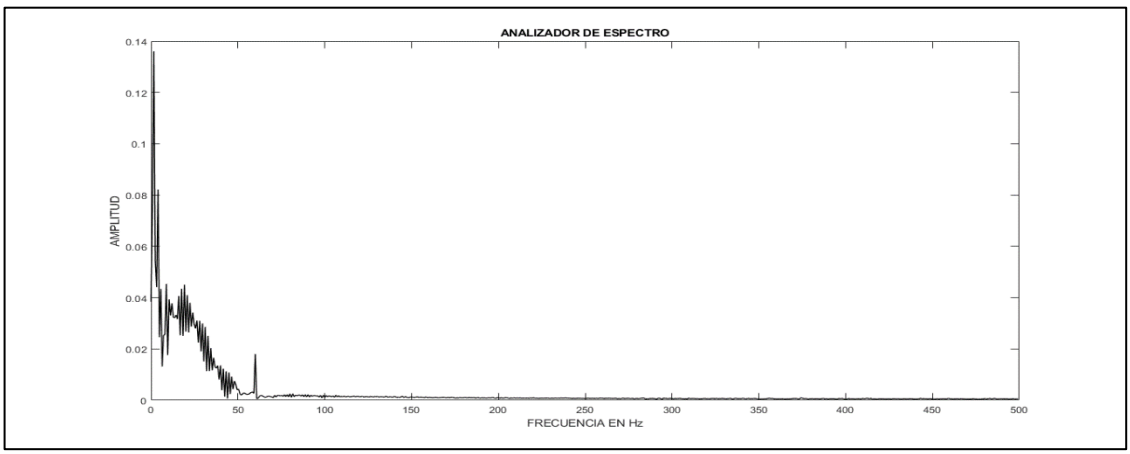


Figura N° 29. Transformada Rápida de Fourier para el filtro Kaiser con orden N = 500.

Fuente: Autoría propia, 2022.

FILTRO VENTANA RECTANGULAR

Para el filtro con ventana Rectangular se obtuvo los siguientes valores correspondientes al SNR, MSE y la Covarianza mencionados correspondientemente en la TABLA N° 11.

TABLA N° 11

Valores del SNR, MSE y la Covarianza para un filtro Rectangular rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
RECTANG-RB	0.48280186	1.587953242	13.88969593	10.17604945	14.54575404	18.50872349	18.54076538	19.77812328
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
RECTANG-RB	0.72222345	0.342974848	0.083426377	0.128209825	0.078307657	0.050508074	0.051193365	0.039567421
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
RECTANG-RB	0.48280186	0.738619714	0.9798901	0.953999945	0.982813841	9.929E-01	0.993031762	0.994708

Fuente: Autoría propia, 2022.

A continuación, se muestra la señal del filtro para los valores de SNR, MSE y Covarianza en la Figura N° 30.

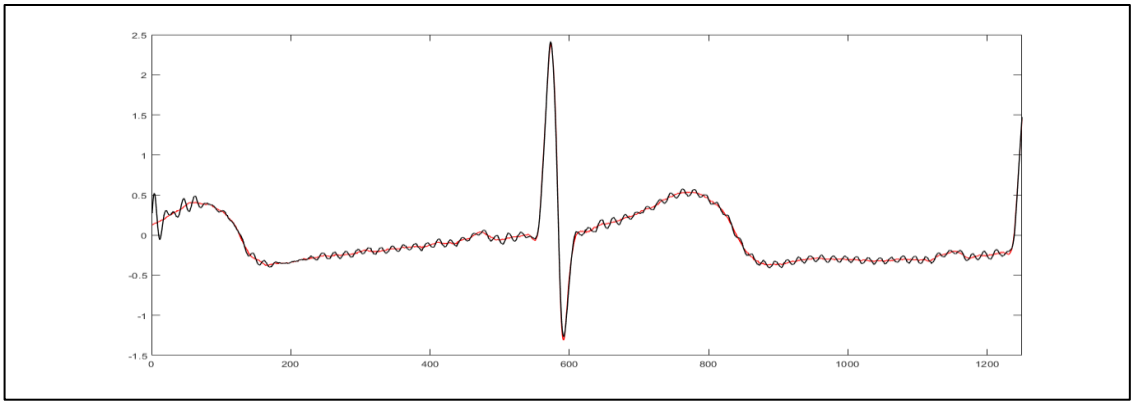


Figura N° 30. Salida del filtro Rectangular de orden N = 500

Fuente: Autoría propia, 2022.

Filtro con ventana de Hanning

Para el filtro con ventana de Hanning se obtuvo los siguientes valores mencionados en la TABLA N° 12.

TABLA N° 12

Valores del SNR, MSE y la Covarianza para un filtro Hanning rechaza-banda.

VALORES SNR								
ORDEN	10	20	50	100	200	300	400	500
Hann-RB	-4.975826207	-4.230308245	0.817619232	9.153410739	22.754624	22.38188459	22.47808858	21.42756159
ERROR CUADRÁTICO MEDIO								
ORDEN	10	20	50	100	200	300	400	500
Hann-RB	0.72923627	0.670013315	0.375711158	0.14420507	0.030434119	0.032337225	0.032534753	0.032723958
COVARIANZA								
ORDEN	10	20	50	100	200	300	400	500
Hann-RB	0.479660307	0.508966993	0.728949951	0.942911101	0.997329164	9.971E-01	0.99715607	0.99637868

Fuente: Autoría propia, 2022.

Se muestra la salida del filtro de Hanning en la Figura N° 31.

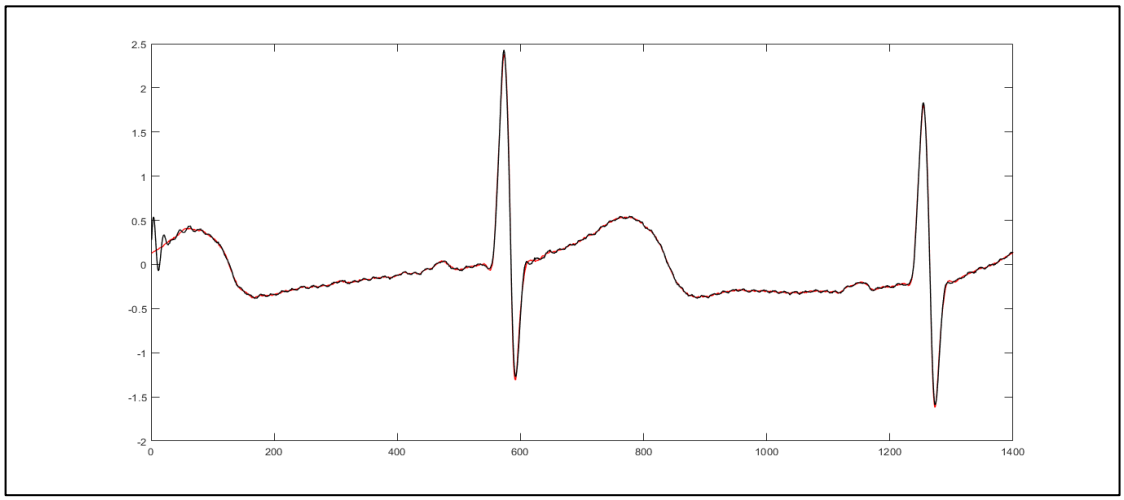


Figura N° 31. Salida del filtro Hanning de orden N = 200

Fuente: Autoría propia, 2022.

Filtro Wavelet

El análisis con filtros wavelet se realizó considerando los diferentes daubechies y niveles del filtro, los resultados se revelan en las Tablas N° 13, 14 y 15 correspondientemente.

TABLA N° 13

Valores del SNR de diferentes daubechies y niveles del filtro Wavelet.

VALORES SNR								
Dubechies	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**	**Nivel 6**	**Nivel 7**	**Nivel 8**
Db2	3.1396	4.162	13.9291	14.4795	11.8287	8.9916	8.171	7.5591
Db3	3.1514	3.9345	16.221	14.5034	9.5834	9.0872	8.224	7.5193
Db4	3.1413	3.7357	18.1954	20.4345	11.8289	9.2775	8.5332	7.5454
Db5	3.1164	3.619	19.4714	16.6255	12.3871	9.0517	8.5582	7.7547
Db6	3.139	3.5212	20.1225	16.9915	10.0452	9.1091	8.2413	7.7694
Db7	3.1481	3.4429	20.6462	21.9742	10.7137	9.2918	8.253	7.693
Db8	3.2007	3.4154	21.2544	16.748	12.5554	9.1727	8.5213	7.6242
Db9	3.1753	3.3577	21.8431	18.4956	10.9445	9.0736	8.5253	7.8254
Db10	3.0727	3.2936	22.834	21.7591	10.2423	9.1785	8.2839	7.8195

Fuente: Autoría propia, 2022.

TABLA N° 14

Valores del MSE de diferentes daubechies y niveles del filtro Wavelet.

ERROR CUADRÁTICO MEDIO								
Daubechies	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**	**Nivel 6**	**Nivel 7**	**Nivel 8**
Db2	0.6712	0.5909	0.1658	0.153	0.2267	0.3278	0.3654	0.3994
Db3	0.6748	0.6276	0.1148	0.1524	0.3005	0.3206	0.3585	0.3923
Db4	0.6771	0.6295	0.0846	0.0559	0.2224	0.3126	0.3445	0.391
Db5	0.6792	0.6386	0.0677	0.1102	0.206	0.3221	0.3434	0.3808
Db6	0.6773	0.6463	0.0596	0.1037	0.2827	0.3195	0.3576	0.38
Db7	0.6766	0.6526	0.0533	0.04	0.2587	0.3121	0.3572	0.3838
Db8	0.6722	0.6547	0.0464	0.1076	0.2013	0.3169	0.345	0.3871
Db9	0.6743	0.6594	0.0409	0.0807	0.2508	0.3212	0.3449	0.3774
Db10	0.6828	0.6646	0.0315	0.0417	0.2754	0.3168	0.3558	0.3776

Fuente: Autoría propia, 2022.

TABLA N° 15

Valores de la covarianza de diferentes daubechies y niveles del filtro Wavelet.

COVARIANZA								
Daubechies	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**	**Nivel 6**	**Nivel 7**	**Nivel 8**
Db2	0.5423	0.5849	0.9181	0.914	0.8151	0.593	0.4613	0.2939
Db3	0.5377	0.5688	0.954	0.9128	0.6665	0.6126	0.4837	0.3195
Db4	0.5413	0.5586	0.9698	0.984	0.8215	0.6346	0.5363	0.3185
Db5	0.5382	0.5635	0.98	0.95	0.8462	0.6079	0.5401	0.3792
Db6	0.5416	0.549	0.984	0.955	0.7089	0.6154	0.4874	0.3843
Db7	0.5485	0.5524	0.9863	0.9911	0.7585	0.6367	0.4888	0.3643
Db8	0.5446	0.556	0.989	0.9519	0.8529	0.6225	0.5343	0.346
Db9	0.5431	0.5495	0.9913	0.9706	0.7735	0.611	0.535	0.3976
Db10	0.5408	0.5469	0.9948	0.9905	0.7249	0.6235	0.4943	0.3958

Fuente: Autoría propia, 2022.

Luego de tener la señal filtrada y libre de la componente de 60 Hz tal como muestra la Figura N° 32, se procesó con un filtro de Savitsky-Golay para que realice la nivelación de la tensión de referencia de la señal de ECG.

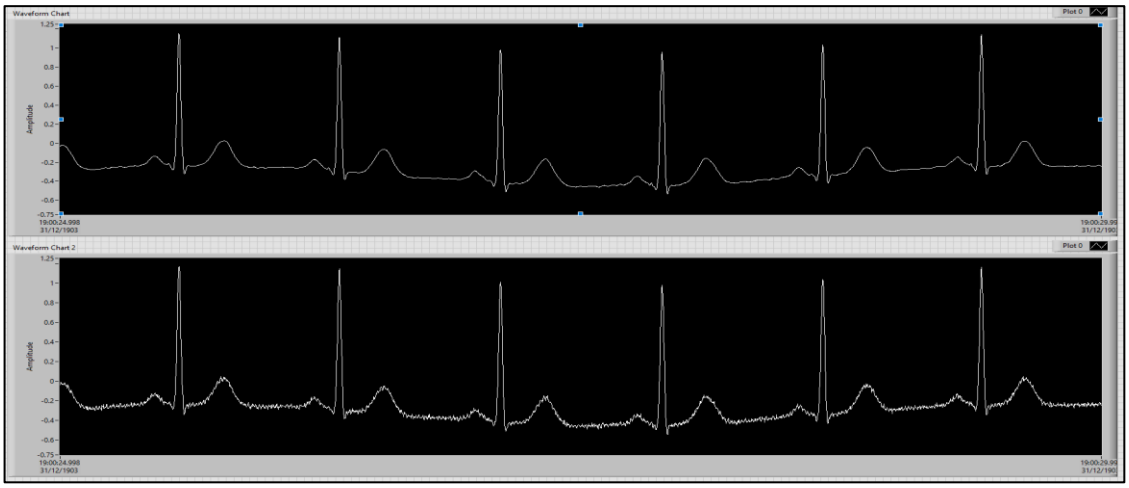


Figura N° 32. Señal de una onda de ECG antes de la aplicación del filtro Savitsky-Golay.

Fuente: Autoría propia, 2022.

La señal al ser pasada por el filtro de Savitsky-Golay nos permite normalizar la señal de ECG para valores de tensión tal como se presenta en la Figura N° 33, y así analizar correctamente los tipos de onda presentes en la señal.

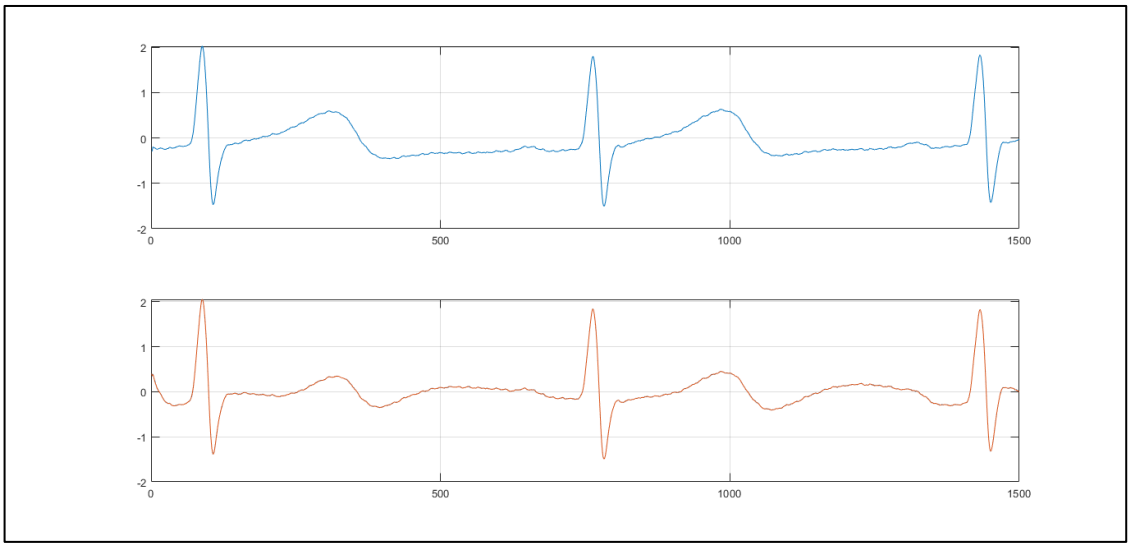


Figura N° 33. Señal de una onda ECG por un filtro Savitsky-Golay.

Fuente: Autoría propia, 2022.

Luego de determinar al filtro Wavelet como el mejor para el procesamiento de las señales de ECG, se procedió al procesamiento de datos y la realización del algoritmo de inteligencia artificial con el fin de predecir un diagnóstico.

Se tuvo que insertar los datos obtenidos de la adquisición de datos del dispositivo cuyo intervalo es 5 segundos tanto la señal original (ver Tabla N° 16), normalizada (ver Tabla N° 17) y filtrada mediante transformada wavelet (ver Tabla N° 18) con sus respectivas líneas de código mostradas a continuación.

```
#importar Señal ECG

df1=pd.read_excel('datos5s.xlsx')

df1=df1.rename(columns= {308: 'Señal'})
```

TABLA N° 16

Señal ECG original

	Señal
0	308
1	317
2	317
3	315
4	315

Fuente: Autoría propia, 2022.

```
#Importar Señal ECG normalizada

df2=pd.read_excel('datosN5s.xlsx')

df2=df2.rename(columns={-0.28:'Señal_N'})
```

TABLA N° 17

Señal ECG Normalizada

	Señal_N
0	-0.29
1	-0.20
2	-0.20
3	-0.22
4	-0.22

Fuente: Autoría propia, 2022.

```
#Importar Señal ECG filtrada por Wavelet

df3=pd.read_excel('datosN5s_wav.xlsx')

df3=df3.rename(columns={-0.0737859829315331:'Señal_N_wav'})
```

TABLA N° 18

Señal ECG filtrada por transformada wavelet

	Señal_N_wav
0	-0.049848
1	-0.026120
2	-0.004491
3	0.014393
4	0.030803

Fuente: Autoría propia, 2022.

El siguiente código muestra la unión de las 3 tablas que contiene la señal ECG original, normalizada y filtrada transformada wavelet para su posterior análisis.

```
#Unir tablas para obtener uno general

tabla1=pd.merge(df1, df2, how='inner', left_index=True, right_index=True)

tabla2=pd.merge(tabla1, df3, how='inner', left_index=True,
right_index=True)
```

Para determinar los valores de BPM, es necesario identificar los picos de la señal y ciertas regiones donde están presentes dichas señales. Lo mencionado se verifica en el siguiente programa en Python.

```
#Determinar los picos de la señal ECG

peak_index = peakutils.indexes(tabla2['Señal_N'], thres=0.7,
min_dist=100)

fig, ax=plt.subplots()

ax.plot(tabla2['Señal_N'])

for peak in peak_index:

ax.axvline(x=peak, color='r')
```

Graficar las regiones y los picos de la señal (ver Figura N° 34) es gracias a las siguientes líneas de código:

```
#Graficar las regiones de las ondas en la señal ECG

fig1, ax1=plt.subplots()

for peak in peak_index:

    ax1.axvline(x=peak, color='r')

p_min_distance=-80

p_max_distance=-140
```

```
for peak in peak_index:
    ax1.axvspan(peak+p_max_distance, peak+p_min_distance, alpha=0.2, color='r')
 t_min_distance=100
t_max_distance=300
for peak in peak_index:
    ax1.axvspan(peak+t_max_distance, peak+t_min_distance, alpha=0.2, color='blue')
ax1.plot(tabla2['Señal_N'])
```

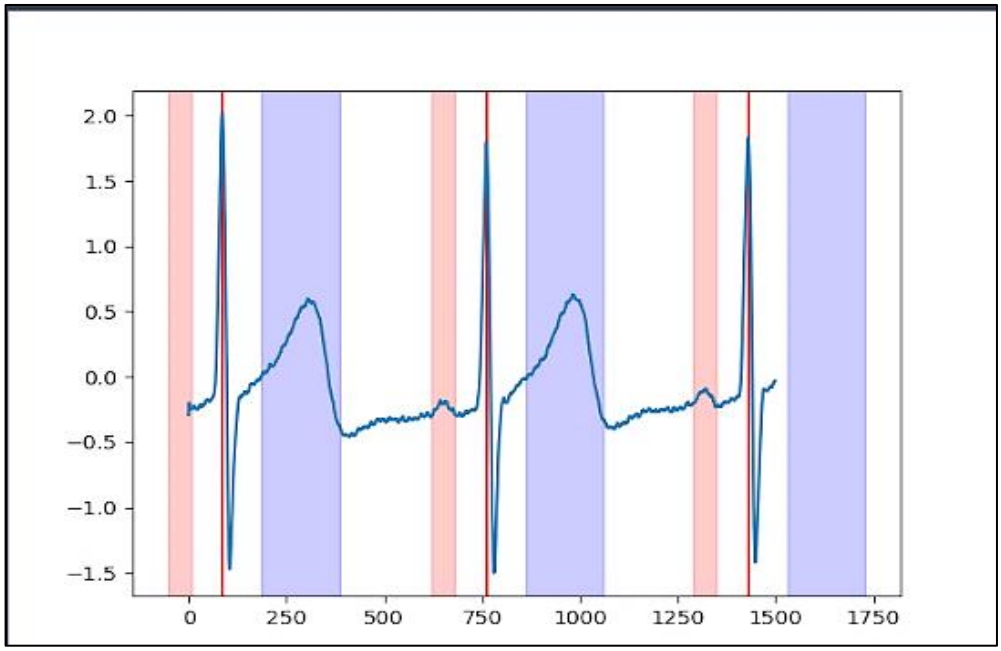


Figura N° 34. Señal ECG delimitada en regiones por su tipo de onda.

Fuente: Autoría propia, 2022.

En el código siguiente se muestra la fórmula para determinar los BPM para ese intervalo (ver Figura N° 35), el cual generó 2 valores porque se obtuvo 3 picos en la señal ECG:

```
#Calcular los BPM
RR_intervals=np.diff(peak_index)/1000
```

HR=60/RR_intervals

```
array([88.88888889, 89.68609865])
```

Figura N° 35. Vector de valores de BPM para el intervalo de la señal ECG.

Fuente: Autoría propia, 2022.

Para aplicar el filtro wavelet es necesario dividir la señal en diferentes partes con su determinada frecuencia con el fin de identificar la señal de ruido (ver Figura N° 36) y así eliminarla al momento de reconstruir la señal. En el algoritmo se visualiza la configuración del filtro de wavelet.

```
#Configuración del filtro Wavelet

wavelets=pywt.wavedec(tabla2['Señal_N'],'db4',level=10)

fig2, ax2=plt.subplots(len(wavelets)+1)

ax2[0].plot(tabla2['Señal_N'])

for i, wavelet in enumerate(wavelets):

    ax2[i+1].plot(wavelet)
```

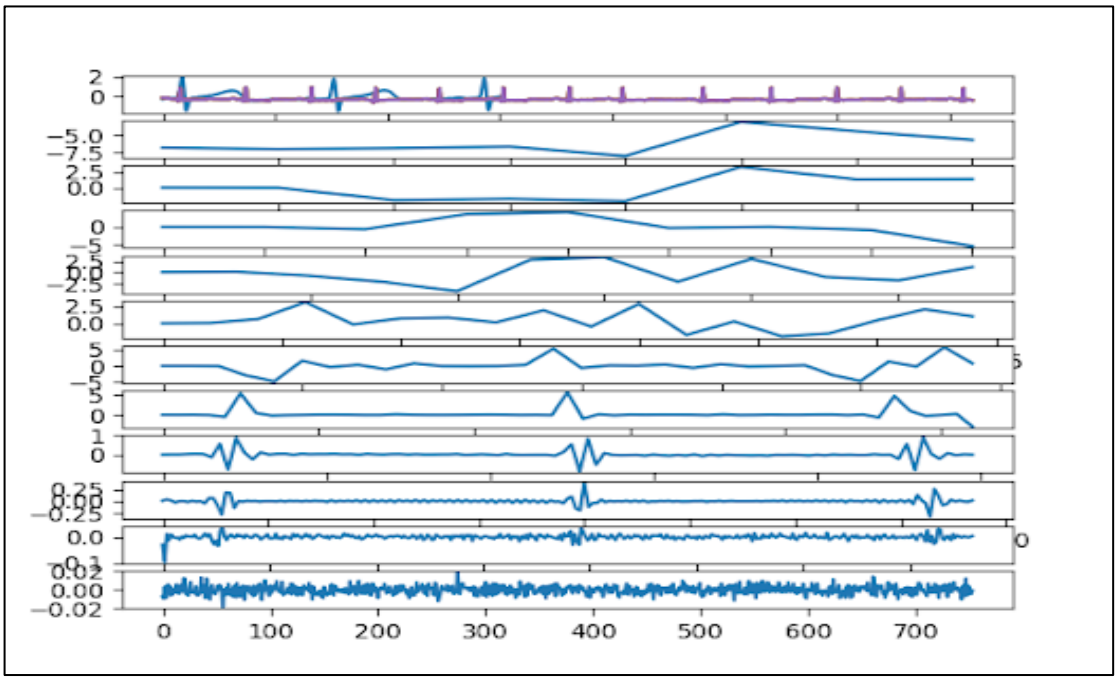


Figura N° 36. Señal ECG divida en partes según su frecuencia.

Fuente: Autoría propia, 2022.

Reconstrucción de la señal con una respectiva ventana de tipo de filtrado wavelet eliminando la señal de ruido (ver Figura N° 37).

```
#Reconstrucción de la señal filtrada

fig3, ax3=plt.subplots()

ax3.plot(pywt.waverec(wavelets[0:7],'db4'))
```

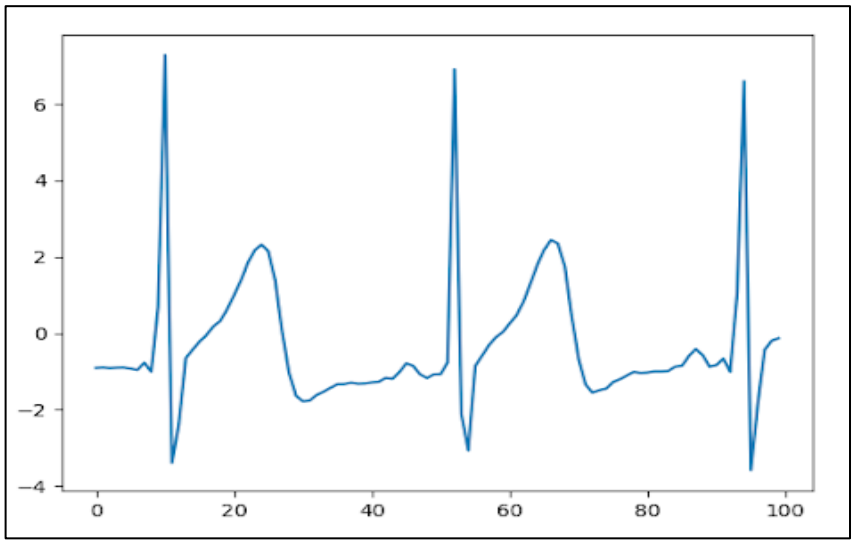


Figura N° 37. Señal ECG reconstruida después de aplicar la transformada wavelet.

Fuente: Autoría propia, 2022.

Con esto vemos la variación de los BPM durante el tiempo que dura la señal. Es normal tener variación entre 8 a 10 bpm por cada 30 segundos (ver Figura N° 38).

```
peak_index = peakutils.indexes(tabla2['Señal_N'], thres=0.7,
min_dist=100)

RR_intervals=np.diff(peak_index)/1000

HR=60/RR_intervals

HR_max=200

HR_min=40

HR=np.where((HR<HR_max)&(HR>HR_min), HR, 0)
```

```
fig4, ax4=plt.subplots()

ax4.plot(HR)
```

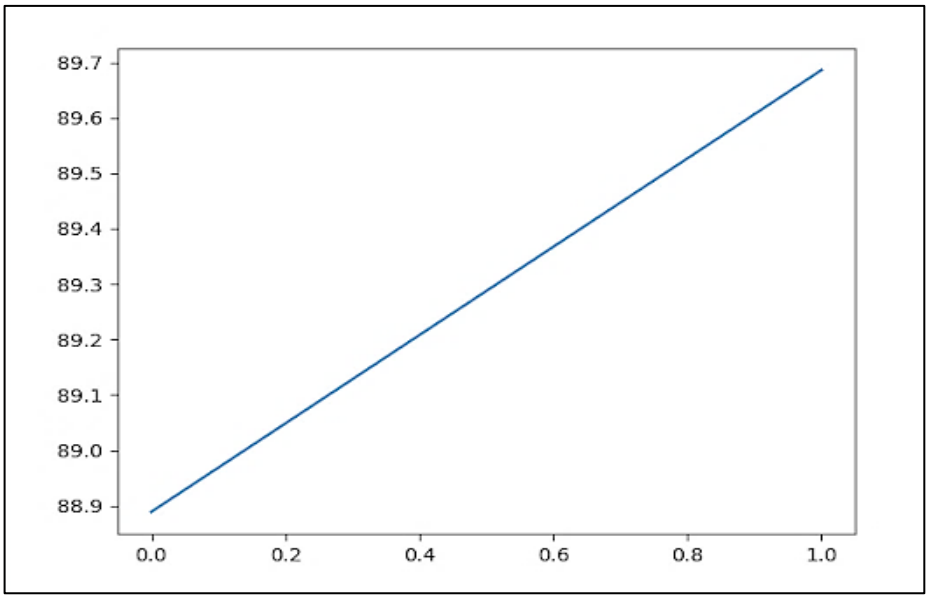


Figura N° 38. Gráfica de la variación en valores de BPM en la señal ECG.

Fuente: Autoría propia, 2022.

Extraer datos de las señales de ECG (ver Figura N° 39) de la base de datos de PhysioNet con el fin de verificar y comparar con los datos que se tiene.

```
df_mit=pd.read_csv('MIT1.csv', names=['time','mv'],skiprows=2)

#times=np.linspace(0, len(df_mit['mv']),len(df_mit['mv']))*(0.003)

#times_pd=pd.DataFrame(times)

fig10, ax10=plt.subplots()

ax10.plot(df_mit.time, df_mit.mv)
```

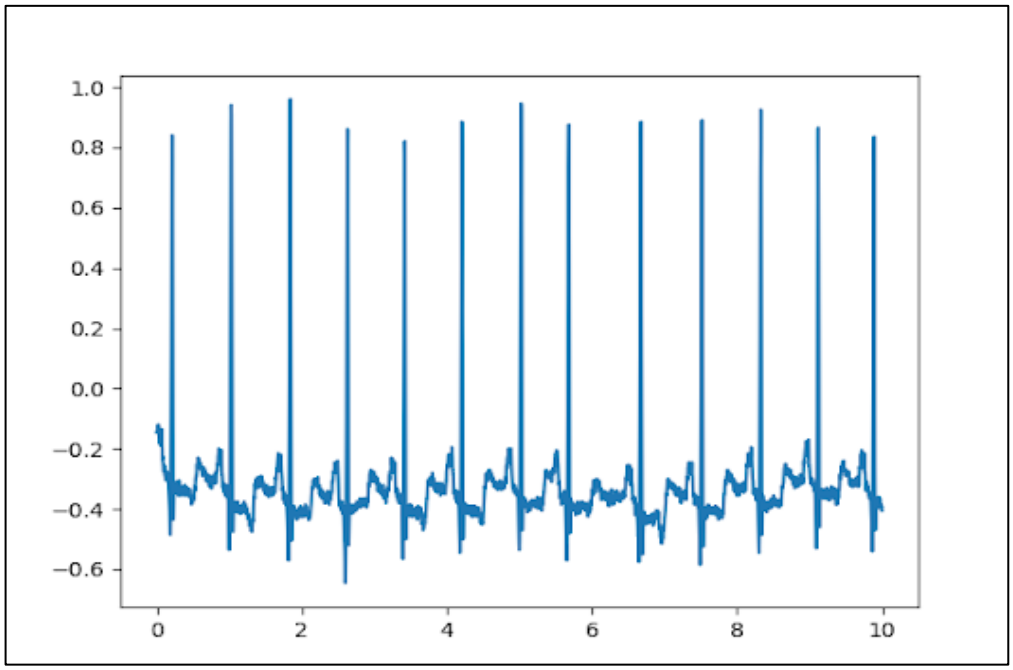


Figura N° 39. Señal ECG de la base de datos MIT.

Fuente: Autoría propia, 2022.

Graficar e identificar los picos para la determinación de BPM (ver Figura N° 40).

```
peak_mit = peakutils.indexes(df_mit['mv'], thres=0.5, min_dist=100)

fig11, ax11=plt.subplots()

ax11.plot(df_mit['mv'])

for peak in peak_mit:

    ax11.axvline(x=peak, color='r')

f=360 #frecuencia de muestreo segun su base de datos

RR_mit=np.diff(peak_mit)/f

HR_mit=60/RR_mit
```

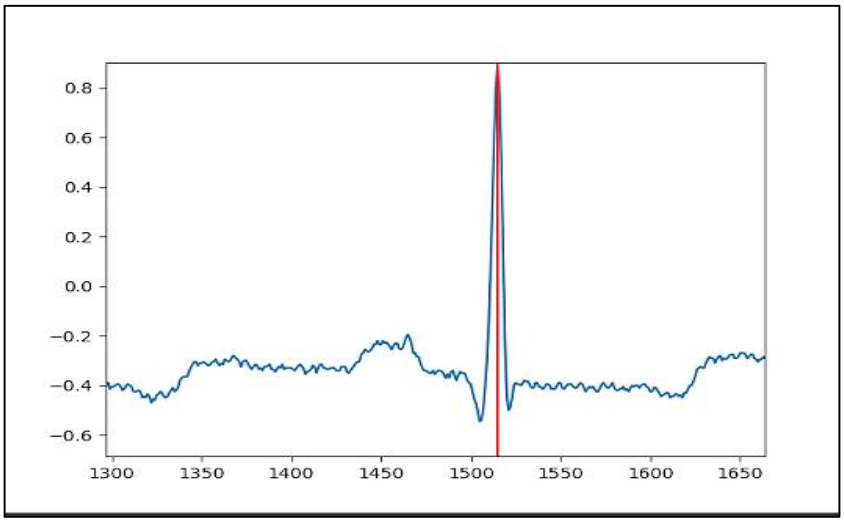


Figura N° 40. Señal ECG de la base de datos del MIT con sus picos identificados.

Fuente: Autoría propia, 2022.

Dividir la señal para identificar la señal ruido (ver Figura N° 41).

```
wavelets_mit=pywt.wavedec(df_mit['mv'],'db3',level=5)
fig12, ax12=plt.subplots(len(wavelets_mit)+1)
ax12[0].plot(df_mit['mv'])
for i, wavelet in enumerate(wavelets_mit):
    ax12[i+1].plot(wavelet)
```

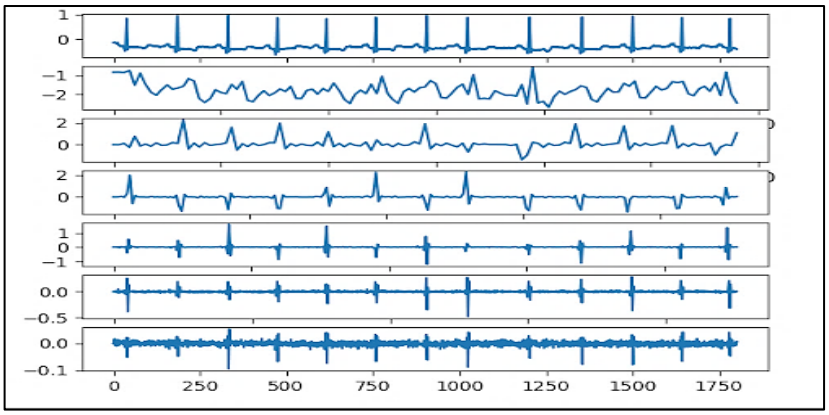


Figura N° 41. Señal ECG de la base de datos del MIT dividida según su frecuencia.

Fuente: Autoría propia, 2022.

Al reconstruir la Señal se ha notado la eliminación del ruido (ver Figura N° 42):

```
fig13, ax13=plt.subplots()

ax13.plot(pywt.waverec(wavelets_mit[0:3],'db3'))
```

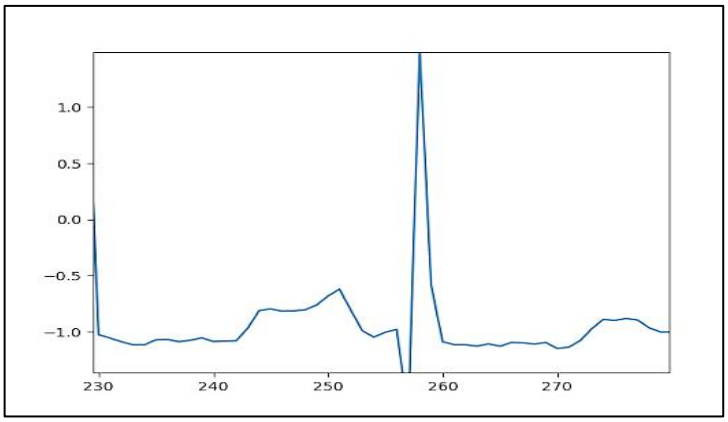


Figura N° 42. Señal ECG de la base de datos del MIT reconstruida sin ruido.

Fuente: Autoría propia, 2022.

Luego se procedió a descargar los archivos con una gran cantidad de datos de muchos pacientes. En PhysioNet los datos de la señal ECG, el cual se muestra en la Tabla N° 19, y del paciente están en archivos distintos por lo que es necesario abrirlos por separado y luego juntarlos.

```
columns=['time','I','II','III','AVR','AVL','AVF','V1','V2','V3','V4','V5','V6']

lr01=pd.read_csv('01.txt',names=columns, sep='\t', skiprows=2)
```

TABLA N° 19

Señal ECG con sus 12 derivaciones de 1000 pacientes

	time	I	II	III	AVR	AVL	AVF	V1	V2	V3	V4	V5	V6
0	0.00	-0.119	-0.055	0.064	0.086	-0.091	0.004	-0.069	-0.031	0.000	-0.026	-0.039	-0.079
1	0.01	-0.116	-0.051	0.065	0.083	-0.090	0.006	-0.064	-0.036	-0.003	-0.031	-0.034	-0.074
2	0.02	-0.120	-0.044	0.076	0.082	-0.098	0.016	-0.058	-0.034	-0.010	-0.028	-0.029	-0.069
3	0.03	-0.117	-0.038	0.080	0.077	-0.098	0.021	-0.050	-0.030	-0.015	-0.023	-0.022	-0.064
4	0.04	-0.103	-0.031	0.072	0.066	-0.087	0.021	-0.045	-0.027	-0.020	-0.019	-0.018	-0.058
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	9.95	0.106	0.028	-0.078	-0.067	0.092	-0.025	0.012	-0.008	0.025	0.020	-0.025	0.026
996	9.96	0.090	0.021	-0.069	-0.055	0.079	-0.023	-0.014	-0.009	-0.023	-0.027	-0.036	-0.008
997	9.97	0.069	0.000	-0.069	-0.034	0.069	-0.035	-0.001	-0.026	0.000	0.024	-0.041	-0.058
998	9.98	0.086	0.004	-0.081	-0.044	0.083	-0.038	0.001	-0.001	-0.025	0.242	-0.046	-0.098
999	9.99	0.022	-0.031	-0.054	0.005	0.038	-0.042	-0.001	0.107	-0.149	0.143	-0.035	-0.120

Fuente: Autoría propia, 2022.

Gráfica de la señal de la V5 (ver Figura N° 43).

```
fig20, ax20=plt.subplots()

ax20.plot(lr01.V5)
```

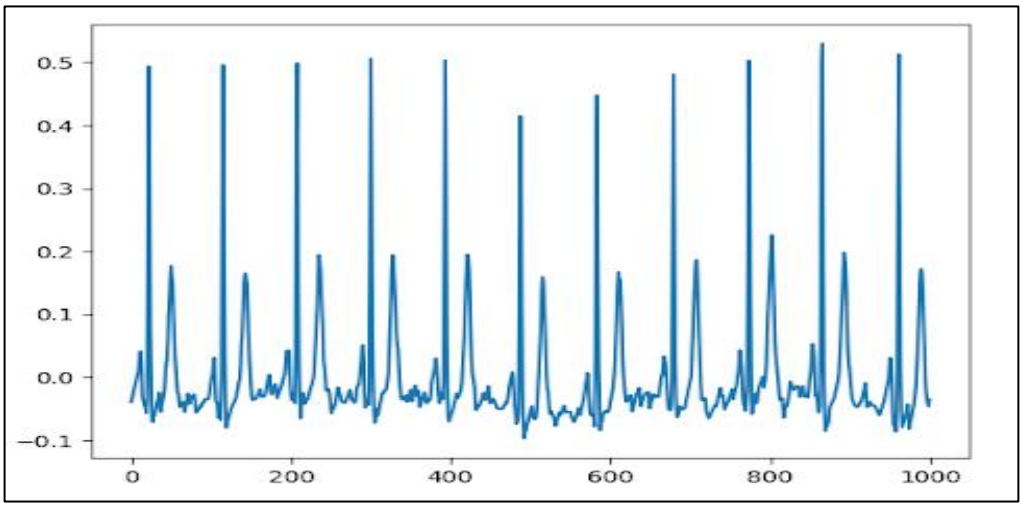


Figura N°43 Gráfica de la derivación V5 de la señal ECG.

Fuente: Autoría propia, 2022.

Leer el archivo donde se encuentra información más específica del paciente (ver Tabla N° 20):

```
alldata=pd.read_csv('database2/ptbxl_database.csv')
```

TABLA N° 20

Datos de los pacientes y procedimiento al obtener la señal ECG

	ecg_id	patient_id	age	sex	height	weight	nurse	site	device	recording_date	...	validated_by_human	baseline_drif
0	1	15709.0	56.0	1	NaN	63.0	2.0	0.0	CS-12 E	1984-11-09 09:17:34	...	True	NaN
1	2	13243.0	19.0	0	NaN	70.0	2.0	0.0	CS-12 E	1984-11-14 12:55:37	...	True	NaN
2	3	20372.0	37.0	1	NaN	69.0	2.0	0.0	CS-12 E	1984-11-15 12:49:10	...	True	NaN
3	4	17014.0	24.0	0	NaN	82.0	2.0	0.0	CS-12 E	1984-11-15 13:44:57	...	True	, II,III,AVF
4	5	17448.0	19.0	1	NaN	70.0	2.0	0.0	CS-12 E	1984-11-17 10:43:15	...	True	, III,AVR,AVF
...	...	...	...	...	...	...	...	...	...	...	...	...	...
21832	21833	17180.0	67.0	1	NaN	NaN	1.0	2.0	AT-60 3	2001-05-31 09:14:35	...	True	NaN
21833	21834	20703.0	93.0	0	NaN	NaN	1.0	2.0	AT-60 3	2001-06-05 11:33:39	...	True	NaN
21834	21835	19311.0	59.0	1	NaN	NaN	1.0	2.0	AT-60 3	2001-06-08 10:30:27	...	True	NaN
21835	21836	8873.0	64.0	1	NaN	NaN	1.0	2.0	AT-60 3	2001-06-09 18:21:49	...	True	NaN
21836	21837	11744.0	68.0	0	NaN	NaN	1.0	2.0	AT-60 3	2001-06-11 16:43:01	...	True	NaN

Fuente: Autoría propia, 2022.

Leer todas las columnas (ver Figura N°44):

alldata.columns

```
Index(['ecg_id', 'patient_id', 'age', 'sex', 'height', 'weight', 'nurse',
       'site', 'device', 'recording_date', 'report', 'scp_codes', 'heart_axis',
       'infarction_stadium1', 'infarction_stadium2', 'validated_by',
       'second_opinion', 'initial_autogenerated_report', 'validated_by_human',
       'baseline_drift', 'static_noise', 'burst_noise', 'electrodes_problems',
       'extra_beats', 'pacemaker', 'strat_fold', 'filename_lr', 'filename_hr'],
      dtype='object')
```

Figura N° 44. Nombres de las columnas de la tabla con información de los pacientes.

Fuente: Autoría propia, 2022.

En la figura N° 45 se muestra los tipos de datos de cada columna de la tabla.

alldata.dtypes

```
ecg_id                        int64
patient_id                  float64
age                         float64
sex                           int64
height                      float64
weight                      float64
nurse                       float64
site                        float64
device                       object
recording_date               object
report                       object
scp_codes                    object
heart_axis                   object
infarction_stadium1          object
infarction_stadium2          object
validated_by                float64
second_opinion                 bool
initial_autogenerated_report   bool
validated_by_human             bool
baseline_drift               object
static_noise                 object
burst_noise                  object
electrodes_problems          object
extra_beats                  object
pacemaker                    object
strat_fold                    int64
filename_lr                  object
filename_hr                  object
dtype: object
```

Figura N° 45. Tipo de datos que se tiene en cada columna de los datos de los pacientes.

Fuente: Autoría propia, 2022.

Se eliminan las columnas que en base al propósito del proyecto no son de importancia:

```
#Limpiar la base de datos del paciente para no tener valores nulos o
#vacíos

alldata_clean=alldata.drop(['patient_id','height','nurse','site','device','recor
ding_date','heart_axis',
'infarction_stadium1','infarction_stadium2','validated_by',
'baseline_drift','static_noise','burst_noise','electrodes_problems',
'extra_beats','pacemaker'], axis=1)

alldata_clean.dropna(subset=['ecg_id','age','sex','weight','report',
'scp_codes','second_opinion','initial_autogenerated_report',
'validated_by_human','strat_fold'], axis=0, inplace=True)

final_df=alldata_clean.copy()
```

De la cual se obtuvo la siguiente TABLA N° 21 como resultado.

TABLA N° 21

Datos de los pacientes con columnas no requeridas eliminadas

	ecg_id	age	sex	weight	report	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold
0	1	56.0	1	63.0	sinusrhythmus periphere niederspannung	False	False	True	3
1	2	19.0	0	70.0	sinusbradykardie sonst normales ekg	False	False	True	2
2	3	37.0	1	69.0	sinusrhythmus normales ekg	False	False	True	5
3	4	24.0	0	82.0	sinusrhythmus normales ekg	False	False	True	3
4	5	19.0	1	70.0	sinusrhythmus normales ekg	False	False	True	4
...	...	...	...	...	...	...	...	...	...
9347	20970	58.0	0	74.0	sinustachykardie p-sinistrocardiale linkstyp p...	False	True	True	6
9348	21040	58.0	0	74.0	sinusrhythmus a-v block i p-sinistrocardiale l...	False	True	True	6

Fuente: Autoría propia, 2022.

Combinar tablas df10 el cual contiene las señales ECG de cada paciente y final_df con datos del paciente (ver Tabla N° 22):

```
#Tabla final
```

tabla_final=pd.merge(final_df, df10, how='inner', left_index=True, right_index=True)

TABLA N° 22

Tabla unificada con información de la señal ECG y datos del paciente

	ecg_id	age	sex	weight	report	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold
0	1	56.0	1	63.0	sinusrhythmus periphere niederspannung	False	False	True	3
1	2	19.0	0	70.0	sinusbradykardie sonst normales ekg	False	False	True	2
2	3	37.0	1	69.0	sinusrhythmus normales ekg	False	False	True	5
3	4	24.0	0	82.0	sinusrhythmus normales ekg	False	False	True	3
4	5	19.0	1	70.0	sinusrhythmus normales ekg	False	False	True	4
...	...	...	...	...	...	...	...	...	...
9347	20970	58.0	0	74.0	sinustachykardie p-sinistrocardiale linkstyp p...	False	True	True	6
9348	21040	58.0	0	74.0	sinusrhythmus a-v block i p-sinistrocardiale l...	False	True	True	6

Fuente: Autoría propia, 2022.

Luego se procedió a limitar los diagnósticos a normal y anormal para determinar si existe alguna enfermedad o no, lo mencionado se visualiza en la Tabla N° 23.

Tabla_final['report'].replace({'sinusrhythmus normales ekg':'normal', }, inplace=True)

TABLA N° 23

Determinar si existe una anomalía cardíaca en el reporte de cada paciente

	ecg_id	age	sex	weight	report	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold	0_x
0	1	56.0	1	63.0	anormal	False	False	True	3	0.064
1	2	19.0	0	70.0	normal	False	False	True	2	0.134
2	3	37.0	1	69.0	normal	False	False	True	5	-0.049
3	4	24.0	0	82.0	normal	False	False	True	3	-0.083
4	5	19.0	1	70.0	normal	False	False	True	4	-0.540
...	...	...	...	...	...	...	...	...	...	...
13732	20873	46.0	0	80.0	anormal	0	1	1	10	0.088
13733	20943	52.0	0	80.0	anormal	0	0	1	10	-0.056
13734	20970	58.0	0	74.0	anormal	0	1	1	6	-0.008
13735	21040	58.0	0	74.0	anormal	0	1	1	6	0.065
13736	21255	58.0	1	52.0	anormal	0	1	0	5	-0.030

Fuente: Autoría propia, 2022.

Luego se determinó la cantidad de “normal” y “anormal” en la columna de reportes (ver Figura N° 46), además nos encontraremos con valores nulos (nan)

tabla_final ['report'].value_counts()

```
anormal    5281
normal     4071
Name: report, dtype: int64
```

Figura N° 46. Cantidad de reportes con resultados normal y anormal.

Fuente: Autoría propia, 2022.

Luego se tuvo que saber cuántas filas existe con valores nulos y únicos (ver Figura N° 47).

tabla_final ['report'].unique()

```
array(['anormal', 'normal', nan], dtype=object)
```

Figura N° 47. Vector con valores únicos en la columna de reporte.

Fuente: Autoría propia, 2022.

Después se eliminó las filas con valores nulos con el fin de limpiar la data antes de entrenar el algoritmo de inteligencia artificial (ver Tabla N° 24).

tabla_final = tabla_final.dropna()

TABLA N° 24

Tabla general sin valores nulos en la columna “report”

	ecg_id	age	sex	weight	report	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold	0_x
0	1.0	56.0	1.0	63.0	anormal	False	False	True	3.0	0.064
1	2.0	19.0	0.0	70.0	normal	False	False	True	2.0	0.134
2	3.0	37.0	1.0	69.0	normal	False	False	True	5.0	-0.049
3	4.0	24.0	0.0	82.0	normal	False	False	True	3.0	-0.083
4	5.0	19.0	1.0	70.0	normal	False	False	True	4.0	-0.540
...	...	...	...	...	...	...	...	...	...	...
13732	20873.0	46.0	0.0	80.0	anormal	0	1	1	10.0	0.088
13733	20943.0	52.0	0.0	80.0	anormal	0	0	1	10.0	-0.056
13734	20970.0	58.0	0.0	74.0	anormal	0	1	1	6.0	-0.008
13735	21040.0	58.0	0.0	74.0	anormal	0	1	1	6.0	0.065
13736	21255.0	58.0	1.0	52.0	anormal	0	1	0	5.0	-0.030

Fuente: Autoría propia, 2022.

Normalizar los valores booleanos a 0 y 1 en todas las columnas como se aprecia en la Tabla N° 25.

```
tabla_final['initial_autogenerated_report'].replace({False:0,True:1}, inplace=True)

tabla_final['validated_by_human'].replace({False:0,True:1}, inplace=True)

tabla_final['report'].replace({'normal':0,'anormal':1}, inplace=True)
```

TABLA N° 25

TABLA FINAL PREPARADA PARA REALIZAR EL ENTRENAMIENTO

	ecg_id	age	sex	weight	report	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold	0_x
0	1.0	56.0	1.0	63.0	1	0	0	1	3.0	0.064
1	2.0	19.0	0.0	70.0	0	0	0	1	2.0	0.134
2	3.0	37.0	1.0	69.0	0	0	0	1	5.0	-0.049
3	4.0	24.0	0.0	82.0	0	0	0	1	3.0	-0.083
4	5.0	19.0	1.0	70.0	0	0	0	1	4.0	-0.540
...	...	...	...	...	...	...	...	...	...	...
9347	20873.0	46.0	0.0	80.0	1	0	1	1	10.0	0.088
9348	20943.0	52.0	0.0	80.0	1	0	0	1	10.0	-0.056
9349	20970.0	58.0	0.0	74.0	1	0	1	1	6.0	-0.008
9350	21040.0	58.0	0.0	74.0	1	0	1	1	6.0	0.065
9351	21255.0	58.0	1.0	52.0	1	0	1	0	5.0	-0.030

Fuente: Autoría propia, 2022**.**

Para el entrenamiento del algoritmo es necesario determinar las entradas y salidas del sistema por lo que se tiene que eliminar las columnas de ecg_id y report porque no se les considera como entrada y datos importantes para el entrenamiento (ver Tabla N° 26).

```
X= tabla_final.drop(['ecg_id','report'], axis=1).copy()
```

TABLA N° 26

Información necesaria para la variable de entrada X sin la columna "Report"

	age	sex	weight	second_opinion	initial_autogenerated_report	validated_by_human	strat_fold	0_x	1_x	2_x	...
0	56.0	1.0	63.0	0	0	1	3.0	0.064	0.065	0.076	...
1	19.0	0.0	70.0	0	0	1	2.0	0.134	0.136	0.145	...
2	37.0	1.0	69.0	0	0	1	5.0	-0.049	-0.035	-0.003	...
3	24.0	0.0	82.0	0	0	1	3.0	-0.083	-0.103	-0.090	...
4	19.0	1.0	70.0	0	0	1	4.0	-0.540	-0.537	-0.527	...
...	...	...	...	...	...	...	...	...	...	...	...
9347	46.0	0.0	80.0	0	1	1	10.0	0.088	0.062	0.038	...
9348	52.0	0.0	80.0	0	0	1	10.0	-0.056	-0.047	-0.032	...
9349	58.0	0.0	74.0	0	1	1	6.0	-0.008	-0.036	-0.051	...
9350	58.0	0.0	74.0	0	1	1	6.0	0.065	0.066	0.041	...
9351	58.0	1.0	52.0	0	1	0	5.0	-0.030	-0.023	-0.012	...

Fuente: Autoría propia, 2022.

Luego determinar la salida del sistema (ver Figura N° 48):

Y=tabla_final['report'].copy()

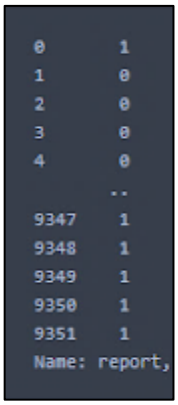


Figura N° 48. Vector de salida “report”.

Fuente: Autoría propia, 2022.

Para datos de entrada con valores limitados y repetitivos se requirió dividir en columnas y transformar en binario los valores de dicha columna, todo ello se visualiza en la Tabla N° 27.

X_final=pd.get_dummies(X, columns=['strat_fold'])

TABLA N° 27

Normalizar algunas columnas de la entrada a binario

	age	sex	weight	second_opinion	initial_autogenerated_report	validated_by_human	0_x	1_x	2_x	3_x	...	strat
0	56.0	1.0	63.0	0	0	1	0.064	0.065	0.076	0.080	...	0
1	19.0	0.0	70.0	0	0	1	0.134	0.136	0.145	0.145	...	0
2	37.0	1.0	69.0	0	0	1	-0.049	-0.035	-0.003	-0.042	...	0
3	24.0	0.0	82.0	0	0	1	-0.083	-0.103	-0.090	-0.101	...	0
4	19.0	1.0	70.0	0	0	1	-0.540	-0.537	-0.527	-0.525	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9347	46.0	0.0	80.0	0	1	1	0.088	0.062	0.038	0.038	...	0
9348	52.0	0.0	80.0	0	0	1	-0.056	-0.047	-0.032	-0.056	...	0
9349	58.0	0.0	74.0	0	1	1	-0.008	-0.036	-0.051	-0.037	...	0
9350	58.0	0.0	74.0	0	1	1	0.065	0.066	0.041	0.007	...	0
9351	58.0	1.0	52.0	0	1	0	-0.030	-0.023	-0.012	-0.003	...	0

Fuente: Autoría propia, 2022.

Ahora se tiene que entrenar el algoritmo con los datos que se tienen con rango de 30% de toda la data para realizar el procedimiento de prueba y el resto para el entrenamiento.

x_trainset, x_testset, y_trainset, y_testset = train_test_split(X_final, Y, test_size=0.3, random_state=3)

Confirmamos los rangos tanto del trainset (ver Figura N° 49) y testset (ver Figura N° 50).

x_trainset.shape

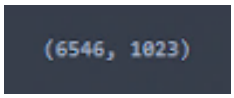


Figura N° 49. Dimensión del Trainset.

Fuente: Autoría propia, 2022.

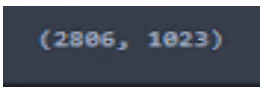


Figura N° 50. Dimensión del Testset.

Fuente: Autoría propia, 2022.

Insertar el algoritmo de árbol de decisiones y configurarlo:

model_tree=DecisionTreeClassifier(random_state=3)

model_tree.fit(x_trainset, y_trainset)

Ahora se tiene que obtener los valores de predicción del testset

prediction_tree=model_tree.predict(x_testset)

Para determinar la precisión del algoritmo con parámetros estadísticos (ver Figura N° 51):

score_model=metrics.accuracy_score(y_testset, prediction_tree)

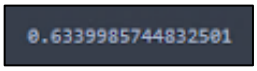


Figura N° 51. Parámetro de Exactitud para el modelo 1.

Fuente: Autoría propia, 2022.

Con esos valores vemos la matriz de confusión y ver falsos negativos y positivos (ver Figura N° 52):

```
plot_confusion_matrix(model_tree, x_testset, y_testset,
display_labels=['Not HD', 'HD']);
```

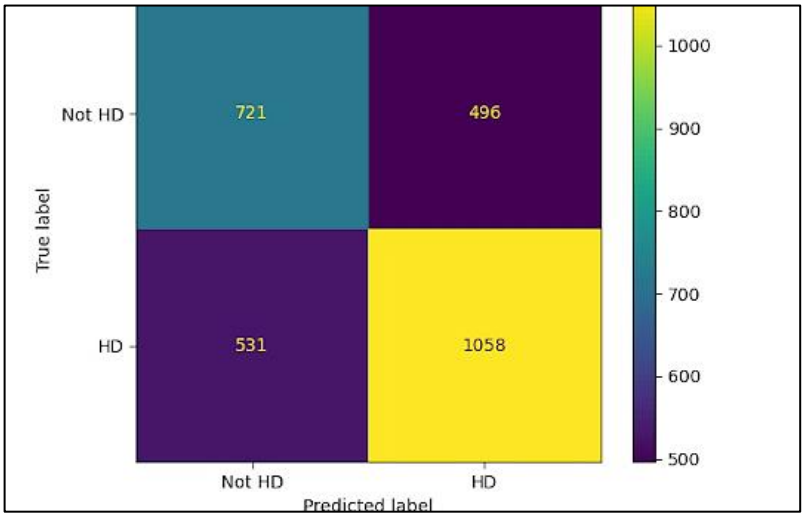


Figura N° 52. Matriz de confusión del modelo 1.

Fuente: Autoría propia, 2022.

Para mejorar la precisión del algoritmo es necesario tener el rango de valores del parametró “alpha” (ver Figura N° 53):

```
path = model_tree.cost_complexity_pruning_path(x_trainset, y_trainset)

ccp_alphas, impurities = path.ccp_alphas, path.impurities

ccp_alphas = ccp_alphas[:-1]

print(ccp_alphas)
clf_dts = []

for ccp_alpha in ccp_alphas:

    model_tree = DecisionTreeClassifier(random_state=0,
ccp_alpha=ccp_alpha)

    model_tree.fit(x_trainset, y_trainset)

    clf_dts.append(model_tree)
```

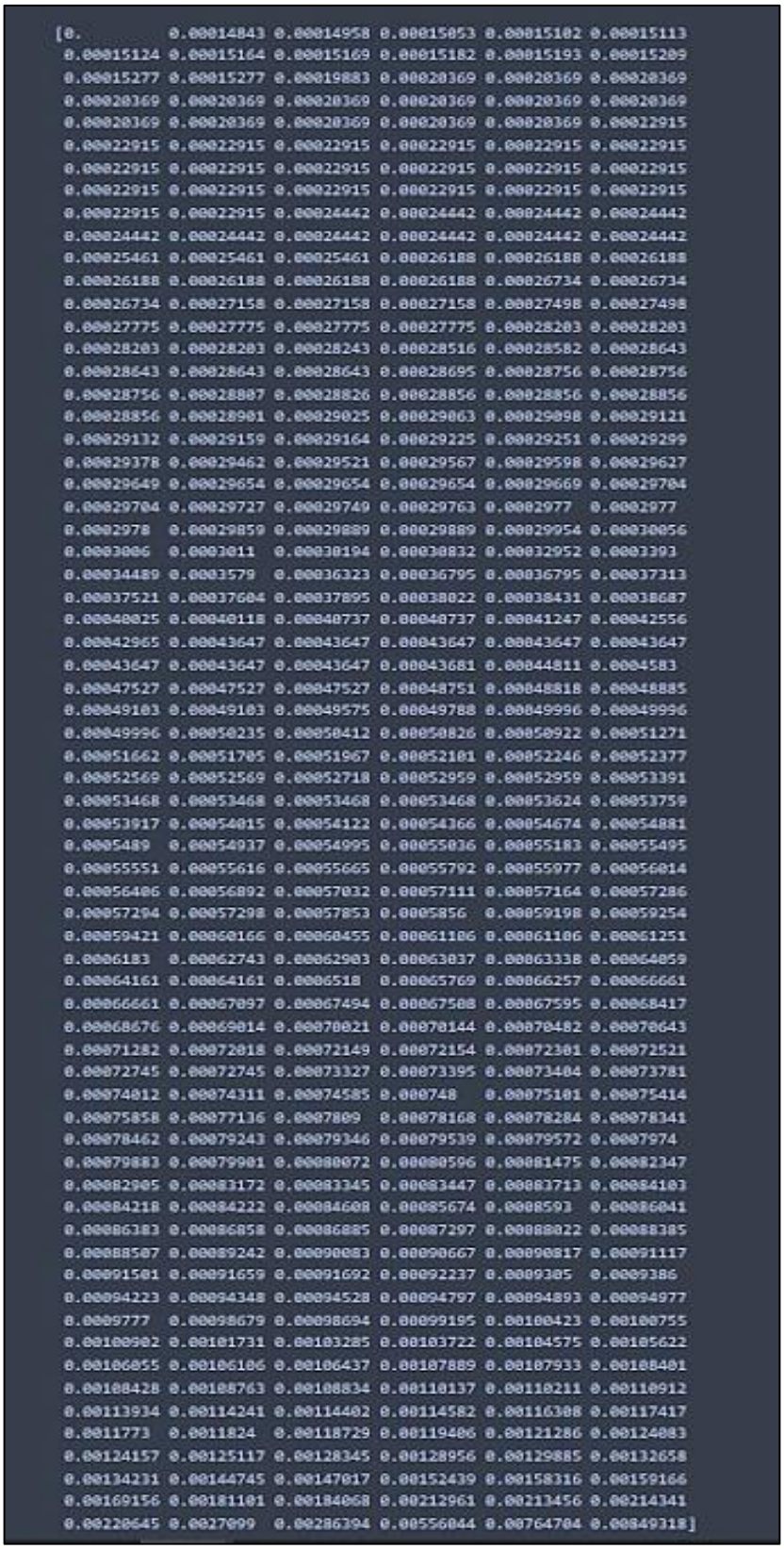


Figura N° 53. Vector de valores que puede tomar Alpha.

Fuente: Autoría propia, 2022.

Luego en la Figura N° 54 se muestra la gráfica los valores que pueda tomar tanto el trainset y testset para determinar el óptimo valor para una mayor exactitud.

```
train_scores = [model_tree.score(x_trainset, y_trainset) for i in clf_dts]

test_scores = [model_tree.score(x_testset, y_testset) for i in clf_dts]

fig100, ax100 = plt.subplots()

ax100.set_xlabel("alpha")

ax100.set_ylabel("accuracy")

ax100.set_title("Accuracy vs alpha for training and testing sets")

ax100.plot(ccp_alphas, train_scores, marker='o', label="train", drawstyle="steps-post")

ax100.plot(ccp_alphas, test_scores, marker='o', label="test", drawstyle="steps-post")

ax100.legend()

plt.show()
```

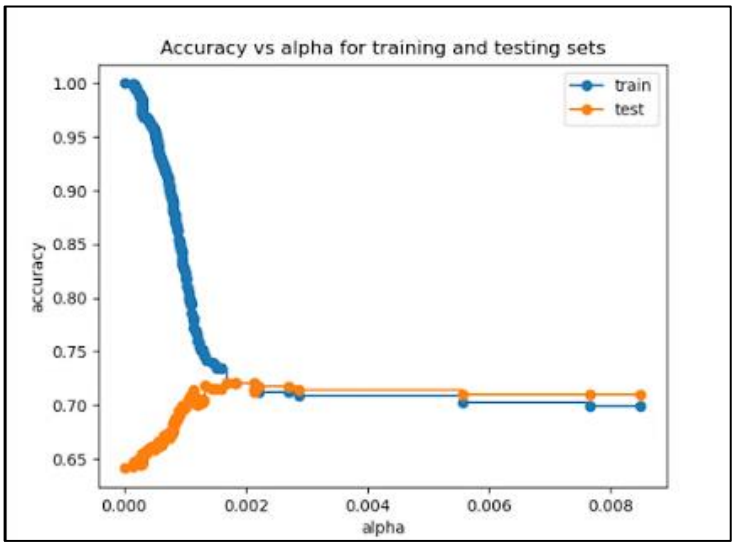


Figura N° 54. Gráfica de Exactitud vs alpha.

Fuente: Autoría propia, 2022.

Luego cuando ya se tiene que con el valor de Alpha=0.00168 se tiene la mayor exactitud se opta para ver la exactitud en base al tipo de árbol de decisiones (ver Figura N° 55):

```
model_tree = DecisionTreeClassifier(random_state=42,
ccp_alpha=0.00168)

scores = cross_val_score(model_tree, x_trainset, y_trainset, cv=10)

df = pd.DataFrame(data={'tree': range(10), 'accuracy': scores})

df.plot(x='tree', y='accuracy', marker='o', linestyle='--')
```

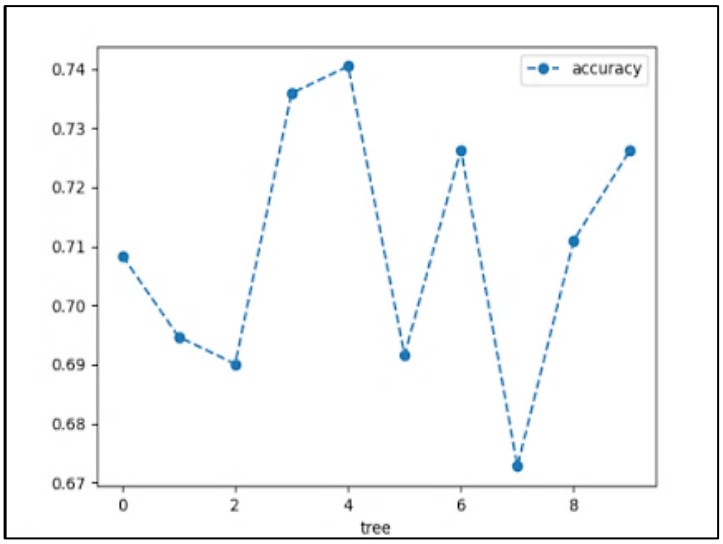


Figura N° 55. Gráfica de Exactitud vs tipo de árbol de decisiones.

Fuente: Autoría propia, 2022.

Ahora en la Figura N° 56 veremos otra gráfica como referencia para determinar el valor del Alpha el cual nos indica la exactitud promedio calculado con el trainset y testset:

```
alpha_loop_values = []

for ccp_alpha in ccp_alphas:

    model_tree = DecisionTreeClassifier(random_state=0,
ccp_alpha=ccp_alpha)

    scores = cross_val_score(model_tree, x_trainset, y_trainset, cv=5)
```

```
    alpha_loop_values.append([ccp_alpha, np.mean(scores),
np.std(scores)])

alpha_results = pd.DataFrame(alpha_loop_values,

                             columns=['alpha', 'mean_accuracy', 'std'])

alpha_results.plot(x='alpha',

                   y='mean_accuracy',

                   yerr='std',

                   marker='o',

                   linestyle='--')
```

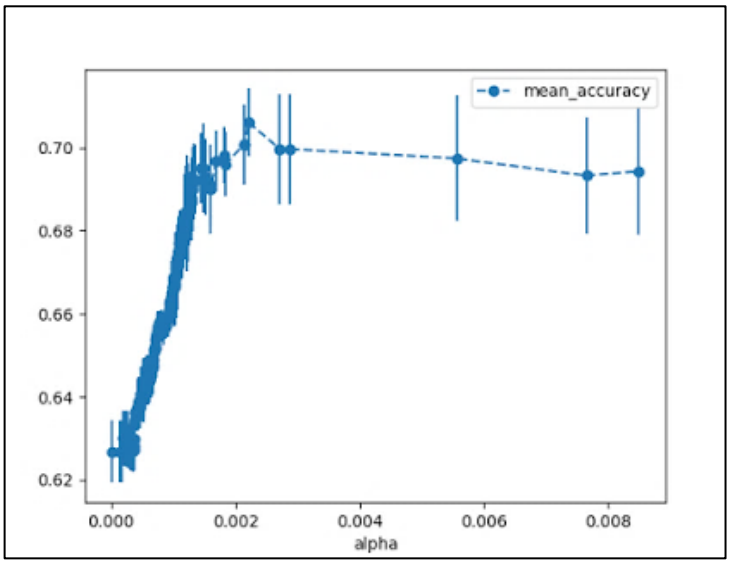


Figura N° 56. Parámetro de Exactitud para el modelo 1.

Fuente: Autoría propia, 2022.

Viendo de la gráfica determinamos un rango y determinamos el valor exacto de Alpha para una mayor precisión (ver Tabla N° 28).

```
alpha_results[np.logical_and(alpha_results['alpha']>0.00215,
alpha_results['alpha']<0.00230)]
```

TABLA N° 28

El óptimo valor de Alpha para la mayor exactitud

	alpha	mean_accuracy	std
396	0.002206	0.70608	0.008084

Fuente: Autoría propia, 2022**.**

Mediante el siguiente comando veremos el valor ideal y con todos los decimales posible requeridos (ver Figura N° 57).

```
ideal_ccp_alpha = alpha_results[np.logical_and(alpha_results['alpha']>
0.00215, alpha_results['alpha']<0.00230)]['alpha']
ideal_ccp_alpha = float(ideal_ccp_alpha)
```

```
0.0022064460052452321
```

Figura N° 57. Valor de Alpha con más decimales.

Fuente: Autoría propia, 2022.

Finalmente se vuelve a entrenar el algoritmo con los parámetros óptimos obtenidos y en la Figura N° 58 se visualiza la matriz de confusión de confusión del modelo 2:

```
model_tree_2 = DecisionTreeClassifier(random_state=42,
ccp_alpha=ideal_ccp_alpha)

model_tree_2 = model_tree_2.fit(x_trainset, y_trainset)

plot_confusion_matrix(model_tree_2,
                      x_testset,
                      y_testset,
                      display_labels=["not HD", "Has HD"])
```

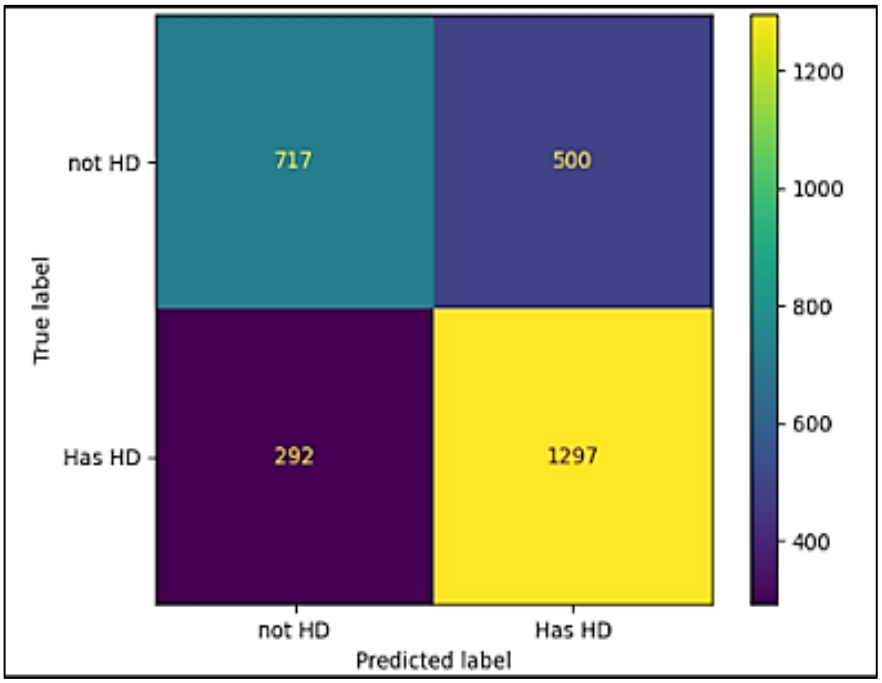


Figura N° 58. Matriz de confusión para el modelo final.

Fuente: Autoría propia, 2022.

Mostrar todo el árbol de decisiones del algoritmo (ver Figura N° 59):

```
plt.figure(figsize=(10,7.5))
plot_tree(model_tree_2,
          filled=True,
          rounded=True,
          class_names=["No HD", "Yes HD"],
          feature_names=X_final.columns);
```

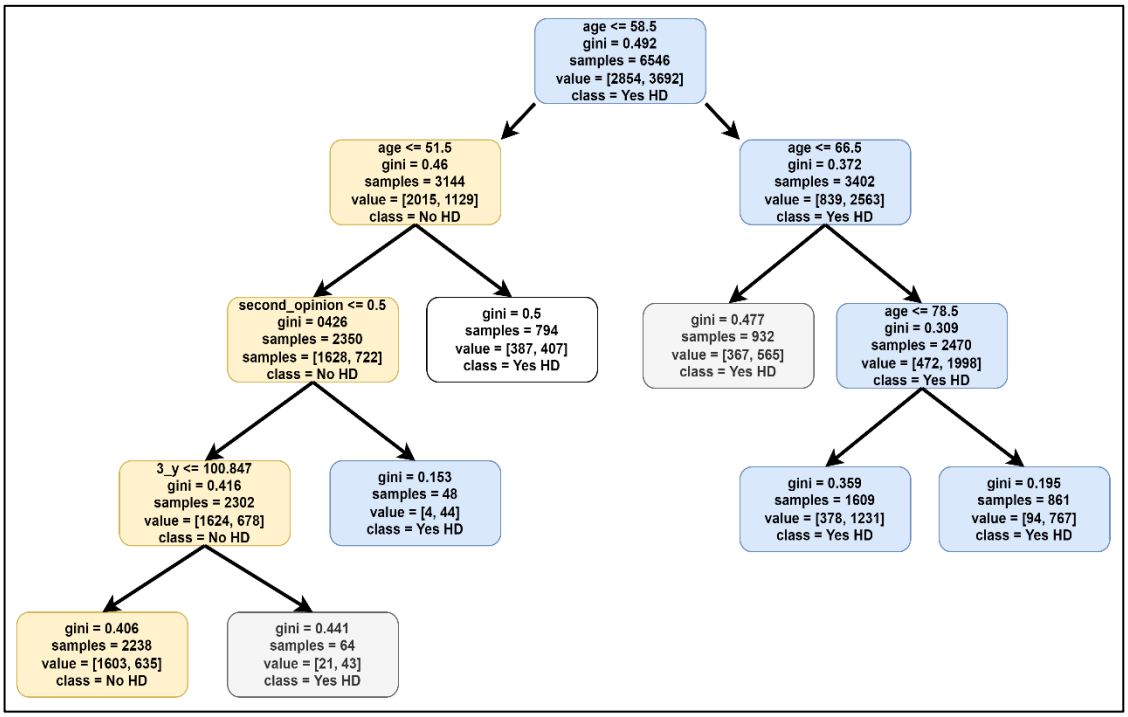


Figura N° 59. Esquema del modelo final de árbol de decisiones.

Fuente: Autoría propia, 2022.

Obtener la precisión del modelo 2 (ver Figura N° 60):

```
prediction_tree_2=model_tree_2.predict(x_testset)

score_model_2=metrics.accuracy_score(y_testset, prediction_tree_2)
```

```
0.7177476835352815
```

Figura N° 60. Parámetro de Exactitud para el modelo 2.

Fuente: Autoría propia, 2022.

Se procedió a obtener más parámetros estadísticos del modelo 2 para tener una mayor comprensión de la eficiencia del entrenamiento (ver Figura N° 61).

```
print('accuracy: {:.2f}'.format(accuracy_score(y_testset,
prediction_tree_2)))

print('precision: {:.2f}'.format(precision_score(y_testset,
prediction_tree_2)))
```

```
print('recall: {:.2f}'.format(recall_score(y_testset, prediction_tree_2)))

print('f1: {:.2f}'.format(f1_score(y_testset, prediction_tree_2)))
```

```
accuracy: 0.72
precision: 0.72
recall: 0.82
f1: 0.77
```

Figura N° 61. Todos los parámetros estadísticos para el modelo 2.

Fuente: Autoría propia, 2022.

Finalmente, en la Tabla N° 29 se muestra un cuadro general de los parámetros estadísticos y la matriz de confusión.

```
print(classification_report(y_testset, prediction_tree_2, target_names=['
No HD ', ' HD ']))
```

TABLA N° 29

Parámetros estadísticos vs Matriz de confusión

	precision	recall	f1-score	support
No HD	0.71	0.59	0.64	1217
HD	0.72	0.82	0.77	1589
accuracy			0.72	2806
macro avg	0.72	0.70	0.71	2806
weighted avg	0.72	0.72	0.71	2806

Fuente: Autoría propia, 2022.

6. DISCUSIÓN DE RESULTADOS

6.1 Contrastación y demostración de la hipótesis con los resultados.

6.1.1 Contrastación de la hipótesis general

La implementación de un electrocardiógrafo con diagnóstico automático tuvo una precisión de diagnóstico del 72% validado con una base de datos obtenida de PhysioNet con una muestra de 9 352 pacientes de diferentes edades y sexo, con esto se asegura que es posible hacer descartes de enfermedades cardiovasculares de manera rápida y segura. La prueba de campo no se pudo concretar debido a la emergencia nacional por la pandemia del SARS-COV2.

Asimismo, se pudo crear un dispositivo de bajo costo para que pueda ser adquirido por hospitales y centros de salud de escasos recursos en zonas rurales y vulnerables del Perú.

6.1.2 Contrastación de las hipótesis específicas

- H1. La aplicación del método utilizado con filtros analógicos y digitales no fue óptima, por lo cual se optó por el uso del filtro Wavelet para el procesamiento y adecuamiento de la señal de ECG, basándonos en datos estadísticos y comparativos los valores del SNR (22.834), MSE (0.0318) y la Covarianza (0.9948). Para después ser pasado por un filtro Savitsky-Golay para eliminar la oscilación de la señal.
- H2. De acuerdo al algoritmo de inteligencia artificial aplicando un árbol de decisiones, se obtuvo los parámetros estadísticos como precisión y exactitud con un valor de 71% y 72% respectivamente comparado con una base de datos de una población de 9 352 pacientes tomando en cuenta su señal de ECG, edad, peso, sexo, etc. Con lo mencionado anteriormente, es posible influir en la prevención de enfermedades cardiovasculares que afecten a zonas vulnerables del Perú.

- H3. Los diagnósticos emitidos por la I.A. después de su correcto entrenamiento comparado con un diagnóstico estándar el cual nos brinda la base de datos del MIT – PhysioNet se pudo reducir la frecuencia de falsos positivos y negativos en los diagnósticos de las enfermedades cardiovasculares para obtener una precisión de 72%.

6.2 Responsabilidad ética de acuerdo a los reglamentos vigentes

Los autores de la presente investigación son responsables de la información brindada en el documento de Tesis de título “Diseño e implementación de un electrocardiógrafo con autodiagnóstico para reducir el índice de mortalidad ocasionadas por patologías cardíacas que afectan a zonas con poblaciones vulnerables en el Perú”, de acuerdo con las normativas actuales de la Universidad Nacional del Callao.

CONCLUSIONES

- Se diseñó una fuente conmutada que posee una mejor eficiencia energética respecto a una fuente lineal para la alimentación del circuito en general, además de la adición de un circuito para el manejo de baterías con la finalidad de hacerlo portable. Se obtuvo el método óptimo para el procesamiento de señales de electrocardiograma para la presente investigación. Se determinó al filtro wavelet como el filtro ideal frente a los demás filtros (analógicos o filtros digitales por ventanas).

- Se generó un óptimo modelo de inteligencia artificial utilizando el algoritmo árbol de decisiones y maximizando los valores de cada parámetro como precisión y exactitud para obtener como resultado 71% y 72 % respectivamente.

- El modelo final responde a varios datos del paciente (peso, edad, sexo, etc). Extraídos de la base de datos MIT- PysioNet con la finalidad de comparar los autodiagnósticos brindados por el algoritmo con los que verificamos en la base de datos

RECOMENDACIONES

- En el momento del uso del dispositivo presentado en este estudio se recomienda que no se use con la fuente conectada a la red eléctrica, sino con las baterías, debido a que existe un riesgo que el paciente pueda sufrir un choque eléctrico.
- Algunos filtros en este informe no se explayaron, por lo cual se recomienda validar otros tipos de filtros y comparar sus resultados obtenidos para que se pueda validar la eficacia de los mismos.
- En la etapa de entrenamiento del algoritmo de I.A. es necesario realizar variaciones en los valores del trainset y testset para mejorar la precisión del diagnóstico.
- Utilizar otros tipos de algoritmos de I.A. para la comparación con el árbol de decisiones que se aplicó en la presente investigación con la finalidad de buscar el óptimo algoritmo para la detección de enfermedades cardiovasculares.

REFERENCIAS BIBLIOGRÁFICAS

[1] "Cardiovascular diseases (cvds)," World Health Organization. [Online]. Available:https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). [Accessed: 15-Oct-2019].

[2] K. M. M. Huamán Guzmán, "Implementación de un simulador de señales electrocardiográficas para la evaluación funcional de monitores," tesis, Universidad de Ciencias y Humanidades, Los Olivos, 2017.

[3] G. M. García Gamarra and K. F. Quino Omonte, "Desarrollo de un equipo tipo Holter de una derivación para monitoreo de señales cardiacas y una aplicación web de procesamiento de señales ECG para detección y estudio de arritmias de tipo fibrilación auricular," Universidad Peruana de Ciencias Aplicadas (UPC), Lima, Perú, 2018.
doi: https://doi.org/10.19083/tesis/624461

[4] J. R. Yupanqui Lizana and S. M. Roncal Loyola, "Diseño e implementación de un módulo de monitoreo cardíaco portátil para zonas rurales," tesis, Pontificia Universidad Católica del Perú, Lima, 2018.

[5] L. E. Chanta Chunga, "Interpretación adecuada del electrocardiograma de un paciente con síndrome coronario agudo por personal médico de emergencia de los establecimientos nivel II y III – Lambayeque 2017," tesis, Universidad de San Martín de Porres, Chiclayo, 2019.

[6] A. E. Vásquez Cahua, "Diseño e implementación de un sistema inalámbrico de detección y alerta de fibrilación auricular en pacientes postrados en cama," tesis, Pontificia Universidad Católica del Perú, Lima, 2015.

[7] P. Gutiérrez Lora, "Diseño de un electrocardiograma portátil en la FPGA Zynq-7000," tesis, Universidad de Sevilla, Sevilla, 2017.

[8] P. A. BURBANO ROJAS, "Diseño e implementación de un equipo simulador de la señal electrocardiográfica para el mantenimiento

preventivo de electrocardiógrafos realizado por la empresa Innovatec S.A," tesis, Universidad Santo Tomás, Bogotá, 2018.

[9] J. C. Carrera Valle, "Sistema de telemedicina para monitorear señales electrocardiográficas en pacientes con enfermedades cardíacas," tesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Carrera de Ingeniería en Electrónica y Comunicaciones, Ambato, Ecuador, 2018.

[10] K. Tsampi, S. Panagiotakis, E. Hatzakis, E. Lakiotakis, G. Atsali, K. Vassilakis, G. Mastorakis, C. X. Mavromoustakis, and A. Malamos, "Extending the SANA mobile healthcare platform with features providing ECG analysis," *Mobile Big Data*, pp. 289–321, 2017.

[11] A. Pérez Talens, "Diseño de un electrocardiógrafo inalámbrico de reducidas dimensiones y elevada autonomía para uso diario," tesis, Universitat Politècnica de València, Valencia, España, 2017.

[12] W. Uribe Arango, M. Duque Ramírez, and E. Medina Arango, *"Electrocardiografía y arritmias",* Rev. Iberoam. Arritmología, 2010.

[13] Luna A. Bayés de, *Manual de Electrocardiografía Básica*. Barcelona, España: Caduceo Multimedia, 2014.

[14] J. R. Hampton, *The ECG made easy*, 8th ed. Londres, Reino Unido: Elsevier Health Sciences UK, 2013.

[15] J. J. Romero, C. Dafonte, Á. Gòmez, and F. J. Penousal, *Inteligencia Artificial Y Computación Avanzada*. Santiago de Compostela, España: Fundación Alfredo Brañas. 2007.

[16] T. M. Mitchell, *Machine learning*. New York, Estados Unidos: McGraw-Hill Book Company, 1997.

[17] Chollet François, *Deep learning with python*. Shelter Island, NY, Estados Unidos: Manning Publications Co., 2018.

[18] J. M. Blackledge, *Digital signal processing: Mathematical and computational methods, software development and applications*, 2a ed. Cambridge, Inglaterra: Woodhead Publishing, 2006.

[19] J. H. Grados Gamarra, N. R. Benites Saravia, D. Huaman Yrigoin, y A. J. Valdez De la Cruz, “Diseño e implementación de un dispositivo ECG para el autodiagnóstico de patologías mediante redes neuronales en Python y monitoreo inalámbrico en tiempo real”, en Proceedings of the 17th LACCEI International Multi-Conference for Engineering, Education, and Technology: “Industry, Innovation, and Infrastructure for Sustainable Cities and Communities”, 2019.

[20] M. Labrosse, Ed., *Cardiovascular Mechanics*. Londres, Inglaterra: CRC Press, 2018.

[21] J. R. Clavijo Mendoza, *Diseño y simulación de sistemas microcontrolados en lenguaje C*, 1st ed. Bogotá, Colombia: Clavijo Mendoza, Juan Ricardo, 2011.

[22] “Sistemas de adquisición y Procesamiento de datos”. [PDF]. Disponible: https://rua.ua.es/dspace/bitstream/10045/19119/1/Sistemas%20de%20adquisici%C3%B3n%20y%20Procesamiento%20de%20datos.pdf. [Accessed: 22-Feb-2022].

[23] J. E. Dombald, “Series de Fourier Aplicación: Análisis de señales,” http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-Juan%20Dombald.pdf, Aug-2011. [Online]. Available: http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-Juan%20Dombald.pdf. [Accessed: 23-Feb-2022].

[24] R. E. Walpole, R. H. Myers, and S. L. Myers, *Probabilidad y estadística para Ingeniería y Ciencias*, 9th ed. Distrito Federal, México: Pearson Educación, 2012.

[25] Analog Devices, “AD8232: Single-Lead, Heart Rate Monitor Front End Data Sheet (Rev. D).” Analog Devices, Inc, Estados Unidos, 2020.

[26] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, “Physiobank, PhysioToolkit, and PhysioNet”, *Circulation*, vol. 101, no. 23, 2000.

[27] Texas Instruments, “LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator.” Texas Instruments, Texas, Estados Unidos, Apr-2021.

[28] Texas Instruments, “ADS111x Ultra-Small, Low-Power, I2C-Compatible, 860-SPS, 16-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator.” Texas Instruments, Estados Unidos, Jan-2018.

[29] N. Rastogi and R. Mehra, “Analysis of butterworth and Chebyshev filters for ECG denoising using wavelets,” IOSR Journal of Electronics and Communication Engineering, vol. 6, no. 6, pp. 37–44, 2013.

[30] J. A. Peralta Arroyo, “Diseño de un algoritmo modular para evaluar la conectividad funcional en señales EEG durante el desarrollo de tareas cognitivas utilizando MATLAB,” tesis, Universidad Tecnológica del Perú, Lima, Perú, 2021.

[31] F. A. Ali, M. Wali, and T. H. Abd, “Design and implementation novel filter to de-noising the electrocardiogram signals,” *IOP Conference Series: Materials Science and Engineering*, vol. 928, no. 2, p. 022057, 2020.

[32] A. Hashemi, M. Rahimpour and M. R. Merati, "Dynamic Gaussian filter for muscle noise reduction in ECG signal," 2015 23rd Iranian Conference on Electrical Engineering, 2015, pp. 120-124, doi:10.1109/IranianCEE.2015.7146194.

ANEXO A

Matriz de Consistencia

Problema	Objetivo	Hipótesis	Variables	Metodología
Principal	**Principal**	**Principal**	Las variables que definen el modelo de la presente investigación quedan expresadas en los términos siguientes: **Variable Independiente** X: Dispositivo electrocardiógrafo con autodiagnóstico **Variable Dependiente** Y: Índice de mortalidad por enfermedades cardiovasculares	**Tipo de estudio y diseño:** Investigación explicativa, correlacional causal **Metodología:** Cuantitativa **Área de estudio:** Universidad Nacional del Callao. **Población y muestra:** Base de datos del MIT – BIH. **Instrumentos:** Datos secundarios (bases de datos). Dispositivos Electrónicos de medición.
¿Es posible diseñar e implementar un dispositivo electrocardiógrafo con diagnóstico automático para reducir el índice de mortalidad ocasionadas por enfermedades cardio-vasculares que afectan a zonas con poblaciones vulnerables en el Perú?	Diseñar e implementar un electrocardiógrafo con diagnóstico automático que pueda reducir el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú.	El diseño y la implementación de un electrocardiógrafo con diagnóstico automático reduce el índice de mortalidad ocasionadas por enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú.		
Específicos	**Específicos**	**Específicos**	**Operacionalización de las variables**	
¿Cuál es el método óptimo para la adquisición y procesamiento de datos de señales de electrocardiógrafo?	Aplicar el óptimo método para la adquisición de datos de las señales electrocardiográficas.	La aplicación del método utilizado para la adquisición de datos de las señales electrocardiográficas es el óptimo.	**Indicadores:** X1: Portabilidad X2: Precio X3: Robustez X4: Independencia energética Y1: Cantidad de enfermedades que se pueden detectar Y2: Confiabilidad del diagnóstico.	
¿Cómo un algoritmo de I.A. puede influir en la prevención ante enfermedades cardio-vasculares que afecten a zonas con poblaciones vulnerables en el Perú?	Desarrollar un algoritmo de I.A para influir en la prevención ante enferme-dades cardiovasculares que afecten a las zonas con poblaciones vulnerables en el Perú.	El desarrollo del algoritmo influirá en la prevención de enfermedades cardiovasculares que afectan a zonas con poblaciones vulnerables en el Perú.		
¿Qué tan confiable es el diagnóstico emitido por un algoritmo de IA?	Contrastar el diagnóstico emitido por el algoritmo de I.A.	La contrastación del diagnóstico emitido por el algoritmo de I.A será confiable.		

ANEXO B

Programa en MATLAB

ADQUISICIÓN DE DATOS

```
clc
clear all
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end
close all
clc

s = serial('COM6','BaudRate',250000);
fopen(s);
% datos directos del puerto serial
data_raw = [];
i = 1;
muestras = 1500;
pause(1);
while(1)
    data_raw(i) = str2double(fscanf(s));
    if i<(muestras + 1)
        plot(data_raw);
    else
        plot(data_raw(i-muestras:i));
    end
    axis([0 muestras 0 600]);
    %pause(0.001);
    if i == muestras
        break
    end
    i = i+1;
end

% datos normalizados
data_N = (data_raw-337)/100;
plot(data_N);
```

FILTRO WAVELET

```
%Generar Ruido Gussiano a 10db
rsn = 10*log10((1/0.2)^2);

%Generar el ruido por la red eléctrica a 60hz
l = 1:1:1500;
A = 1; %Amplitud a variar
f = 60;
x = A*sin(2*pi*f*l*0.0015);

%Agregar el Ruido Gaussiano y senoidal a la señal
normalizada
data_N1 = data_N;
data_seno = data_N1 + x;
data_rsn = awgn(data_seno,rsn);

%Configurar los parámetros para el filtro Wavelet (tipo y
nivel)
%w='haar';
%w='coif4';
w='db10';
%w='bior3.3';
%w='sym8';
m=4;

[c,l]=wavedec(data_rsn,m,w);
data_filtrada_wavelet=wrcoef('a',c,l,w,m);

%Parámetro SNR que indica la eficiencia del filtro
(mientras más alto el
%valor es mejor
numSNR=snr(data_N,data_filtrada_wavelet-data_N);

%Cálculo del parámetro Error cuadrático medio (MSE),
mientras más %cercano a 0 es mejor

error=0;
i=1;
N=1500;
while(1)
    error_acumulado=((data_N(i)-
data_filtrada_wavelet(i))^2)/N;
    error=error_acumulado+error;
    i=i+1;
    if i==1500
        break
    end
end
```

```
error_total=sqrt(error);

%Cálculo del parámetro de Covarianza entre la señal ruido y
la %original, mientras más cercano a 1 o -1 es mejor
C=cov(data_N,data_filtrada_wavelet);
R=(C(1,2)/sqrt(C(1,1)*C(2,2)));

%Graficar los resultados
figure(1);
subplot(3,1,1);
plot(data_N);
title('Señal original');
subplot(3,1,2);
plot(data_rsn);
title('Señal con ruido gaussiano');
subplot(3,1,3);
plot(data_filtrada_wavelet);
title('Señal filtrada por wavelet');
```

ANEXO C

PROGRAMA EN PYTHON

CÓDIGO RASPBERRY PI 3B+ (PYTHON)

```
import pandas as pd
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
import scipy.signal
import pywt
import peakutils
import wfdb
import ast
import wfdb
import ast
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.metrics import classification_report

#importar Señal ECG
df1=pd.read_excel('datos5s.xlsx')
df1=df1.rename(columns= {308: 'Señal'})

#Importar Señal ECG normalizada
df2=pd.read_excel('datosN5s.xlsx')
df2=df2.rename(columns={-0.28:'Señal_N'})

#Importar Señal ECG filtrada por Wavelet
df3=pd.read_excel('datosN5s_wav.xlsx')
df3=df3.rename(columns={-0.0737859829315331:'Señal_N_wav'})

#Unir tablas para obtener uno general
tabla1=pd.merge(df1, df2, how='inner', left_index=True,
right_index=True)
tabla2=pd.merge(tabla1, df3, how='inner', left_index=True,
right_index=True)

#Determinar los picos de la señal ECG
peak_index = peakutils.indexes(tabla2['Señal_N'], thres=0.7,
min_dist=100)
fig, ax=plt.subplots()
ax.plot(tabla2['Señal_N'])
```

```
for peak in peak_index:
ax.axvline(x=peak, color='r')

#Calcular los BPM
RR_intervals=np.diff(peak_index)/1000
HR=60/RR_intervals

#Graficar las regiones de las ondas en la señal ECG
fig1, ax1=plt.subplots()
for peak in peak_index:
    ax1.axvline(x=peak, color='r')
p_min_distance=-80
p_max_distance=-140
for peak in peak_index:
    ax1.axvspan(peak+p_max_distance, peak+p_min_distance,
alpha=0.2, color='r')
 t_min_distance=100
t_max_distance=300
for peak in peak_index:
    ax1.axvspan(peak+t_max_distance, peak+t_min_distance,
alpha=0.2, color='blue')
ax1.plot(tabla2['Señal_N'])

#Configuración del filtro Wavelet
wavelets=pywt.wavedec(tabla2['Señal_N'],'db4',level=10)
fig2, ax2=plt.subplots(len(wavelets)+1)
ax2[0].plot(tabla2['Señal_N'])
for i, wavelet in enumerate(wavelets):
    ax2[i+1].plot(wavelet)

#Reconstrucción de la señal filtrada
fig3, ax3=plt.subplots()
ax3.plot(pywt.waverec(wavelets[0:7],'db4'))

#Importar señales ECG de la base de datos del MIT-Physionet
def load_data(df, sampling_rate, path):
    if sampling_rate == 100:
        data = [wfdb.rdsamp(path+f) for f in
alldata_clean.filename_lr]
    else:
        data = [wfdb.rdsamp(path+f) for f in
alldata_clean.filename_hr]
    data = np.array([signal for signal, meta in data])
    return data
path = 'database2/'
sampling_rate=100
database = load_data(alldata_clean, sampling_rate, path)
```

```
#Darle formato a la tabla de datos de señales ECG para su
análisis y comparación
database_df=pd.DataFrame(database[0])
database[0][:,1]
len(database[0][:,1]) //1000
database.shape //9352,1000,12)
a=[]
a=np.append(a,database[0][:,1])
len(a)  //1000
b=a.ravel()
b.reshape((2,1000))
c=[]
for i in range(0,9351+1):
    c=np.append(c,database[i][:,10], axis=0)
c.shape //9352000
d=c.ravel()
df10=pd.DataFrame(d.reshape((9352,1000)))
df10=df10.T

#Importar datos del paciente
alldata=pd.read_csv('database2/ptbxl_database.csv')

#Limpiar la base de datos del paciente para no tener valores
nulos o vacios
alldata_clean=alldata.drop(['patient_id','height','nurse','site'
,'device','recording_date','heart_axis',
'infarction_stadium1','infarction_stadium2','validated_by',
'baseline_drift','static_noise','burst_noise',
'electrodes_problems', 'extra_beats','pacemaker'], axis=1)
alldata_clean.dropna(subset=['ecg_id','age','sex','weight','repo
rt', 'scp_codes','second_opinion',
'initial_autogenerated_report',
'validated_by_human','strat_fold'], axis=0, inplace=True)
final_df=alldata_clean.copy()

#Tabla final
tabla_final=pd.merge(final_df, df10, how='inner',
left_index=True, right_index=True)

#Adecuamos el formato de la Tabla final para empezar el
entrenamiento
tabla_final['report'].replace({'sinusrhythmus normales
ekg':'normal', }, inplace=True)
tabla_final.dropna()
tabla_final['initial_autogenerated_report'].replace({False:0,Tru
e:1}, inplace=True)
```

```
tabla_final['validated_by_human'].replace({False:0,True:1},
inplace=True)
tabla_final['report'].replace({'normal':0,'anormal':1},
inplace=True)

#Establecer las entradas del algoritmo para el entrenamiento y
prueba
X= Tabla_final.drop(['ecg_id','report'], axis=1).copy()
X_final=pd.get_dummies(X, columns=['strat_fold'])

#Establecer las salidas del algortimos para el entrenamiento y
prueba
Y=df60['report'].copy()

#Seccionar tanto las entradas y salida para la realización del
entranamiento y prueba
x_trainset, x_testset, y_trainset, y_testset =
train_test_split(X_final, Y, test_size=0.3, random_state=3)

#Configuración inicial del modelo 1 usando el algoritmo árbol de
decisiones
model_tree=DecisionTreeClassifier(random_state=3)
model_tree.fit(x_trainset, y_trainset)

#Predicción del mdelo 1 con las entradas de prueba
prediction_tree=model_tree.predict(x_testset)

#Parámetro de exactitud del modelo 1
score_model=metrics.accuracy_score(y_testset, prediction_tree)

#Graficar la Matriz de confusión del modelo 1
plot_confusion_matrix(model_tree, x_testset, y_testset,
display_labels=['Not HD', 'HD']);

#Extraer los valores del parámetro alpha que pueda tomar
path = model_tree.cost_complexity_pruning_path(x_trainset,
y_trainset)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
ccp_alphas = ccp_alphas[:-1]
print(ccp_alphas)
clf_dts = []
for ccp_alpha in ccp_alphas:
    model_tree = DecisionTreeClassifier(random_state=0,
ccp_alpha=ccp_alpha)
    model_tree.fit(x_trainset, y_trainset)
    clf_dts.append(model_tree)
```

```
#Graficar los múltiples casos de combinación entre el parámetro
alpha con el rango de división
#de las entradas y salida
train_scores = [model_tree.score(x_trainset, y_trainset) for i
in clf_dts]
test_scores = [model_tree.score(x_testset, y_testset) for i in
clf_dts]
fig100, ax100 = plt.subplots()
ax100.set_xlabel("alpha")
ax100.set_ylabel("accuracy")
ax100.set_title("Accuracy vs alpha for training and testing
sets")
ax100.plot(ccp_alphas, train_scores, marker='o', label="train",
drawstyle="steps-post")
ax100.plot(ccp_alphas, test_scores, marker='o', label="test",
drawstyle="steps-post")
ax100.legend()
plt.show()

#Para un valor determinado de alpha se gráfica que tantas
ramificaciones se debe
#tener en el modelo
model_tree = DecisionTreeClassifier(random_state=42,
ccp_alpha=0.00168)
scores = cross_val_score(model_tree, x_trainset, y_trainset,
cv=10)
df = pd.DataFrame(data={'tree': range(10), 'accuracy': scores})
df.plot(x='tree', y='accuracy', marker='o', linestyle='--')

#Considerando lo analizado anteriormente se determina el valor
exacto de alpha
alpha_loop_values = []
for ccp_alpha in ccp_alphas:
    model_tree = DecisionTreeClassifier(random_state=0,
ccp_alpha=ccp_alpha)
    scores = cross_val_score(model_tree, x_trainset, y_trainset,
cv=5)
    alpha_loop_values.append([ccp_alpha, np.mean(scores),
np.std(scores)])
alpha_results = pd.DataFrame(alpha_loop_values,
    columns=['alpha', 'mean_accuracy', 'std'])
alpha_results.plot(x='alpha',
    y='mean_accuracy',
    yerr='std',
    marker='o',
    linestyle='--')
```

```
alpha_results[np.logical_and(alpha_results['alpha']>0.00215,
alpha_results['alpha']<0.00230)]
ideal_ccp_alpha =
alpha_results[np.logical_and(alpha_results['alpha']> 0.00215,
alpha_results['alpha']<0.00230)]['alpha']
ideal_ccp_alpha = float(ideal_ccp_alpha)

#Se entrena el modelo 2 considerando los parámetros conseguidos
model_tree_2 = DecisionTreeClassifier(random_state=42,
ccp_alpha=ideal_ccp_alpha)
model_tree_2 = model_tree_2.fit(x_trainset, y_trainset)

#Se grafica la Matriz de confusión del modelo 2
plot_confusion_matrix(model_tree_2,
    x_testset,
    y_testset,
    display_labels=["not HD", "Has HD"])

#Se grafica el esquema del modelo 2
plt.figure(figsize=(10,7.5))
plot_tree(model_tree_2,
    filled=True,
    rounded=True,
    class_names=["No HD", "Yes HD"],
    feature_names=X_final.columns);

#Se predice y muestra los parámetros estadísticos de los
resultados
prediction_tree_2=model_tree_2.predict(x_testset)
score_model_2=metrics.accuracy_score(y_testset,
prediction_tree_2)
print('accuracy: {:.2f}'.format(accuracy_score(y_testset,
prediction_tree_2)))
print('precision: {:.2f}'.format(precision_score(y_testset,
prediction_tree_2)))
print('recall: {:.2f}'.format(recall_score(y_testset,
prediction_tree_2)))
print('f1: {:.2f}'.format(f1_score(y_testset,
prediction_tree_2)))

#Se muestra en tabla un resumen de la matriz de confusión con
los
#parámetros estadísticos
print(classification_report(y_testset, prediction_tree_2,
target_names=[' No HD ', ' HD ']))
```

ANEXO D

AD8232 DATASHEET

Single-Lead, Heart Rate Monitor Front End

Data Sheet **AD8232**

FEATURES

- Fully integrated single-lead ECG front end
- Low supply current: 170 µA (typical)
- Common-mode rejection ratio: 80 dB (dc to 60 Hz)
- Two or three electrode configurations
- High signal gain (G = 100) with dc blocking capabilities
- 2-pole adjustable high-pass filter
- Accepts up to ±300 mV of half cell potential
- Fast restore feature improves filter settling
- Uncommitted op amp
- 3-pole adjustable low-pass filter with adjustable gain
- Leads off detection: ac or dc options
- Integrated right leg drive (RLD) amplifier
- Single-supply operation: 2.0 V to 3.5 V
- Integrated reference buffer generates virtual ground
- Rail-to-rail output
- Internal RFI filter
- 8 kV HBM ESD rating
- Shutdown pin
- 20-lead, 4 mm × 4 mm LFCSP and LFCSP_SS package
- Qualified for automotive applications

APPLICATIONS

- Fitness and activity heart rate monitors
- Portable ECG
- Remote health monitors
- Gaming peripherals
- Biopotential signal acquisition

FUNCTIONAL BLOCK DIAGRAM

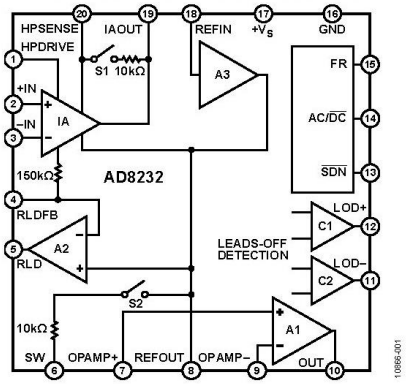


Figure 1.

GENERAL DESCRIPTION

The AD8232 is an integrated signal conditioning block for ECG and other biopotential measurement applications. It is designed to extract, amplify, and filter small biopotential signals in the presence of noisy conditions, such as those created by motion or remote electrode placement. This design allows for an ultralow power analog-to-digital converter (ADC) or an embedded microcontroller to acquire the output signal easily.

The AD8232 can implement a two-pole high-pass filter for eliminating motion artifacts and the electrode half-cell potential. This filter is tightly coupled with the instrumentation architecture of the amplifier to allow both large gain and high-pass filtering in a single stage, thereby saving space and cost.

An uncommitted operational amplifier enables the AD8232 to create a three-pole low-pass filter to remove additional noise. The user can select the frequency cutoff of all filters to suit different types of applications.

To improve common-mode rejection of the line frequencies in the system and other undesired interferences, the AD8232 includes an amplifier for driven lead applications, such as right leg drive (RLD).

The AD8232 includes a fast restore function that reduces the duration of otherwise long settling tails of the high-pass filters. After an abrupt signal change that rails the amplifier (such as a leads off condition), the AD8232 automatically adjusts to a higher filter cutoff. This feature allows the AD8232 to recover quickly, and therefore, to take valid measurements soon after connecting the electrodes to the subject.

The AD8232 is available in a 4 mm × 4 mm, 20-lead LFCSP and a LFCSP_SS package. Performance for the A grade models is specified from 0°C to 70°C and the models are operational from −40°C to +85°C. Performance for the W grade models is specified over the automotive temperature range of −40°C to +105°C.

Rev. D Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
Technical Support www.analog.com

SPECIFICATIONS

V_S = 3 V, V_{REF} = 1.5 V, V_{CM} = 1.5 V, T_A = 25°C, operating temperature (T_{OPR}) = −40°C to +105°C for the W grade, FR = low, SDN = high, and AC/$\overline{DC}$ = low, unless otherwise noted.

Table 1.

Parameter	Symbol	Test Conditions/ Comments	A Grade			W Grade			Unit
			Min	Typ	Max	Min	Typ	Max	
INSTRUMENTATION AMPLIFIER									
Common-Mode Rejection Ratio, DC to 60 Hz	CMRR	V_{CM} = 0.35 V to 2.85 V, V_{DIFF} = 0 V	80	86					dB
		T_{OPR}				80			dB
		V_{CM} = 0.35 V to 2.85 V, V_{DIFF} = ±0.3 V		80			80		dB
Power Supply Rejection Ratio	PSRR	V_S = 2.0 V to 3.5 V	76	90					dB
		T_{OPR}				76	90		dB
Offset Voltage (RTI)	V_{OS}								
Instrumentation Amplifier Inputs				3	8				mV
		T_{OPR}					3	8	mV
DC Blocking Input[1]				5	50		5	50	µV
Average Offset Drift									
Instrumentation Amplifier Inputs				10					µV/°C
		T_{OPR}					10		µV/°C
DC Blocking Input[1]				0.05			0.05		µV/°C
Input Bias Current	I_B			50	200		50	200	pA
		T_A = 0°C to 70°C		1					nA
		T_{OPR}						3.75	nA
Input Offset Current	I_{OS}			25	100		25	100	pA
		T_A = 0°C to 70°C		1					nA
		T_{OPR}						1	nA
Input Impedance									
Differential				10‖7.5			10‖7.5		GΩ‖pF
Common Mode				5‖15			5‖15		GΩ‖pF
Input Voltage Noise (RTI)									
Spectral Noise Density		f = 1 kHz		100			100		nV/√Hz
Peak-to-Peak Voltage Noise		f = 0.1 Hz to 10 Hz		12			12		µV p-p
		f = 0.5 Hz to 40 Hz		14			14		µV p-p
Input Voltage Range		T_A = 0°C to 70°C	0.2		+V_S				V
		T_{OPR}				0.2		+V_S	V
DC Differential Input Range	V_{DIFF}		−300		+300				mV
		T_{OPR}				−300		+300	mV
Output									
Output Swing		R_L = 50 kΩ	0.1		+V_S − 0.1				V
		T_{OPR}				0.1		+V_S − 0.1	V
Short-Circuit Current	I_{OUT}			6.3			6.3		mA
Gain	A_V			100			100		V/V
Gain Error		V_{DIFF} = 0 V		0.4			0.4		%
		V_{DIFF} = −300 mV to +300 mV		1	3.5			3.5	%
		T_{OPR}						7.9	%
Average Gain Drift		T_A = 0°C to 70°C		12					ppm/°C
		T_{OPR}					12		ppm/°C
Bandwidth	BW			2			2		kHz
RFI Filter Cutoff (Each Input)				1			1		MHz

Parameter	Symbol	Test Conditions/ Comments	A Grade Min	A Grade Typ	A Grade Max	W Grade Min	W Grade Typ	W Grade Max	Unit
OPERATIONAL AMPLIFIER (A1)									
Offset Voltage	V_{OS}			1	5				mV
		T_{OPR}					1	5	mV
Average TC		T_A = 0°C to 70°C		5					µV/°C
		T_{OPR}					5		µV/°C
Input Bias Current	I_B			100			100		pA
		T_A = 0°C to 70°C		1					nA
		T_{OPR}					2.5		nA
Input Offset Current	I_{OS}			100			100		pA
		T_A = 0°C to 70°C		1					nA
		T_{OPR}					1		nA
Input Voltage Range			0.1		$+V_S$ – 0.1	0.1		$+V_S$ – 0.1	V
Common-Mode Rejection Ratio	CMRR	V_{CM} = 0.5 V to 2.5 V		100			100		dB
Power Supply Rejection Ratio	PSRR			100			100		dB
Large Signal Voltage Gain	A_{VO}			110			110		dB
Output Voltage Range		R_L = 50 kΩ	0.1		$+V_S$ – 0.1				V
		T_{OPR}				0.1		$+V_S$ – 0.1	V
Short-Circuit Current Limit	I_{OUT}			12			12		mA
Gain Bandwidth Product	GBP			100			100		kHz
Slew Rate	SR			0.02			0.02		V/µs
Voltage Noise Density (RTI)	e_n	f = 1 kHz		60			60		nV/√Hz
Peak-to-Peak Voltage Noise (RTI)	$e_{n\ p-p}$	f = 0.1 Hz to 10 Hz		6			6		µV p-p
		f = 0.5 Hz to 40 Hz		8			8		µV p-p
RIGHT LEG DRIVE AMPLIFIER (A2)									
Output Swing		R_L = 50 kΩ	0.1		$+V_S$ – 0.1	0.1		$+V_S$ – 0.1	V
Short-Circuit Current	I_{OUT}			11			11		mA
Integrator Input Resistor			120	150	180	120	150	180	kΩ
Gain Bandwidth Product	GDP			100			100		kHz
REFERENCE BUFFER (A3)									
Offset Error	V_{OS}	R_L > 50 kΩ		1			1		mV
Input Bias Current	I_B			100			100		pA
Short-Circuit Current Limit	I_{OUT}			12			12		mA
Voltage Range		R_L = 50 kΩ	0.1		$+V_S$ – 0.7				V
		T_{OPR}				0.1		$+V_S$ – 0.7	V
DC LEADS OFF COMPARATORS									
Threshold Voltage				$+V_S$ – 0.5			$+V_S$ – 0.5		V
Hysteresis				60			60		mV
Propagation Delay				0.5			0.5		µs
AC LEADS OFF DETECTOR									
Square Wave Frequency	f_{AC}		50	100	175	50	100	175	kHz
Square Wave Amplitude	I_{AC}			200			200		nA p-p
Impedance Threshold		Between +IN and –IN	10	20		10	20		MΩ
Detection Delay				110			110		µs
FAST RESTORE CIRCUIT									
Switches		S1 and S2							
On Resistance	R_{ON}		8	10	12	8	10	12	kΩ
Off Leakage				100			100		pA
Window Comparator									
Threshold Voltage		From either rail		50			50		mV
Propagation Delay				2			2		µs
Switch Timing Characteristics									
Feedback Recovery Switch On Time	t_{SW1}			110			110		ms
Filter Recovery Switch On Time	t_{SW2}			55			55		ms
Fast Restore Reset	t_{RST}			2			2		µs

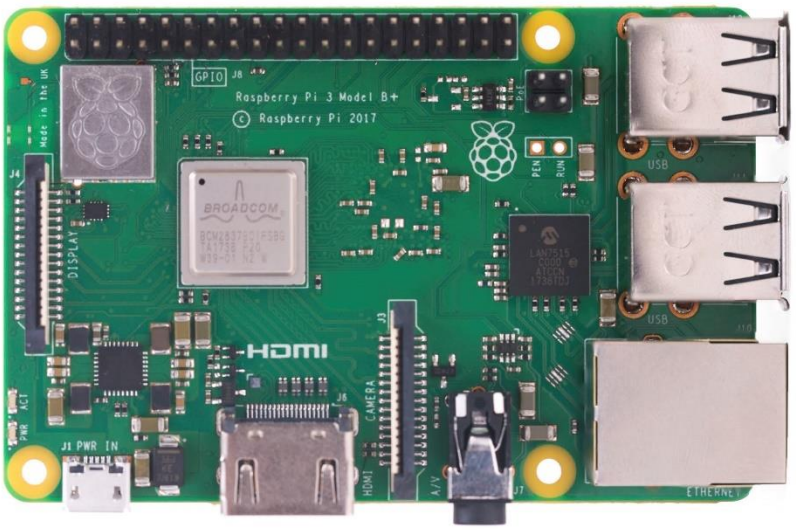
Parameter	Symbol	Test Conditions/ Comments	A Grade Min	A Grade Typ	A Grade Max	W Grade Min	W Grade Typ	W Grade Max	Unit
LOGIC INTERFACE									
Input Characteristics									
Input Voltage (AC/$\overline{DC}$ and FR)									
Low	V_{IL}			1.24			1.24		V
High	V_{IH}			1.35			1.35		V
Input Voltage ($\overline{SDN}$)									
Low	V_{IL}			2.1			2.1		V
High	V_{IH}			0.5			0.5		V
Output Characteristics		LOD+ and LOD− terminals							
Output Voltage									
Low	V_{OL}			0.05			0.05		V
High	V_{OH}			2.95			2.95		V
SYSTEM SPECIFICATIONS									
Quiescent Supply Current				170	230		170	230	µA
		T_A = 0°C to 70°C		210					µA
		T_{OPR}						270	µA
Shutdown Current				40	500		40	500	nA
		T_A = 0°C to 70°C		100					nA
		T_{OPR}						612	nA
Supply Range			2.0		3.5	2.0		3.5	V
Specified Temperature Range			0		+70	−40		+105	°C
Operational Temperature Range			−40		+85	−40		+105	°C

[1] Offset referred to the input of the instrumentation amplifier inputs. See the Input Referred Offsets section for additional information.

ANEXO E

RASPBERRY PI 3B+ DATASHEET

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5V/2.5A DC via micro USB connector ■ 5V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

raspberrypi.org