Neiva, 13 de Marzo de 2024

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Byron Hernando Galindo Suarez, con C.C. No. 1.003.810.784,

Juan Esteban Narváez Carvajal, con C.C. No. 1.075.322.604,

_____, con C.C. No. _____,

_____, con C.C. No. _____,

Autor(es) de la tesis y/o trabajo de grado o _____

Titulado Interfaz Gráfica Para Reconocimiento Y Detección De Características En Electrocardiogramas Usando Python

presentado y aprobado en el año 2024 como requisito para optar al título de

Ingeniero electrónico;

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

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

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

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

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

EL AUTOR/ESTUDIANTE:

Firma: Byron H. Galindo

EL AUTOR/ESTUDIANTE:

Firma:

EL AUTOR/ESTUDIANTE:

Firma: _____

EL AUTOR/ESTUDIANTE:

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: Interfaz Gráfica Para Reconocimiento Y Detección De Características En Electrocardiogramas Usando Python

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Galindo Suarez Narváez Carvajal	Byron Hernando Juan Esteban

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Salgado Patrón	José de Jesús

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
Diaz Franco Ramírez Gutiérrez	Fernand Julián Adolfo

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2024 **NÚMERO DE PÁGINAS:** 108

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

Diagramas_**X**__ Fotografías___ Grabaciones en discos___ Ilustraciones en general_**X**__ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_**X**_

SOFTWARE requerido y/o especializado para la lectura del documento: Microsoft Word / Adobe Reader

MATERIAL ANEXO: Código Fuente en formato .pdf

Electrocardiogramas Fuente de datos

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

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Ingles	Español	Ingles
1.	Python	Python	6. Kardia	Kardia
2.	Electrocardiograma	Electrocardiography	7. Procesamiento de imagen	Image Processing
3.	Píxel	Pixel	8. Recorte de vectores	Array Slicing
4.	TkInter	TkInter	9. Numpy	Numpy
5.	Interfáz gráfica	GUI	10. Transformada de Hough	Hough's Transform

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

La finalidad de este trabajo es la elaboración de una interfaz gráfica capaz de realizar la detección de enfermedades cardiacas en electrocardiogramas producidos por el dispositivo Kardia 6L a través del lenguaje de programación Python orientado a la visión artificial o por computador. El presente proyecto busca una mejora importante en la entrega de diagnósticos precisos y efectivos relacionados a cardiopatías.

Para realizar de manera correcta lo anteriormente planteado es necesario llevar a cabo un proceso de digitalización de las señales eléctricas suministradas por el electrocardiógrafo. Mediante la aplicación de filtros, operaciones morfológicas y demás técnicas de visión por computador a la imagen que contiene las mediciones cardiacas se obtienen finalmente señales equivalentes registradas de manera digital para un posterior análisis matemático el cual permitirá la detección de los posibles padecimientos en el corazón que pueda presentar un paciente.

La interfaz gráfica consta de una ventana con cuatro vistas conectadas entre sí mediante botones; la primera ventana contiene la información general de la interfaz, en la segunda se encuentra la ventana donde se carga el archivo en formato pdf del electrocardiograma, en la tercera la visualización del electrocardiograma ya digitalizado y los valores de onda calculados y en la cuarta los resultados con las posibles enfermedades diagnosticadas.

ABSTRACT: (Máximo 250 palabras)

The purpose of this work is to develop a graphical interface capable of detecting cardiac diseases in electrocardiograms produced by the Kardia 6L device using the Python programming language oriented to artificial or computer vision. The present project seeks a significant improvement in the delivery of accurate and effective diagnoses related to heart disease.

In order to correctly perform the above mentioned, it is necessary to carry out a digitization process of the electrical signals supplied by the electrocardiograph. By applying filters, morphological operations and other computer vision techniques to the image containing the cardiac measurements, equivalent digitally recorded signals are finally obtained for subsequent mathematical analysis, which will allow the detection of possible heart conditions that a patient may present.

The graphical interface consists of a window with four views connected to each other by means of buttons; the first window contains the general information of the interface, in the second is the window where the file in pdf format of the electrocardiogram is loaded, in the third the visualization of the already digitized electrocardiogram and the calculated wave values and in the fourth the results with possible diagnosed diseases.

APROBACION DE LA TESIS

Nombre Presidente Jurado: José de Jesús Salgado Patá

Firma:

Nombre Jurado: Fernando Diaz Franco

Firma: Fernand Diaz F.

Nombre Jurado: Julian Adolfo Ramirez Gutierrez

Firma:

INTERFAZ GRÁFICA PARA RECONOCIMIENTO Y DETECCIÓN DE CARACTERÍSTICAS EN ELECTROCARDIOGRAMAS USANDO PYTHON

BYRON HERNANDO GALINDO SUÁREZ
JUAN ESTEBAN NARVÁEZ CARVAJAL

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2024

INTERFAZ GRÁFICA PARA RECONOCIMIENTO Y DETECCIÓN DE CARACTERÍSTICAS EN ELECTROCARDIOGRAMAS USANDO PYTHON

BYRON HERNANDO GALINDO SUÁREZ Cod. 20171155352
JUAN ESTEBAN NARVÁEZ CARVAJAL Cod. 20171159625

Trabajo de grado para aplicar
al título de ingeniero electrónico

Director:
Mag. José de Jesús Salgado Patrón

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2024

Notas de aceptación

_____
_____
_____
_____
_____
_____
_____

_____
Firma del director de Tesis

_____
Firma del Jurado

_____
Firma del Jurado

Neiva, 05 de febrero de 2024.

Quiero extender mi más sincero agradecimiento a las personas que desempeñaron roles fundamentales en la realización de esta tesis. A Esteban, mi compañero de tesis, por su constante dedicación, colaboración y apoyo a lo largo de este desafiante pero gratificante camino.

A Daniela Rendón, cuya guía como médica asesora ha sido esencial para dar dirección y rigor a nuestro trabajo. Tus conocimientos han sido un pilar invaluable.

A Daniel Alaguna, mi mentor, quien generosamente compartió su experiencia y sabiduría, y cuyo aliento me impulsó a superar obstáculos y crecer tanto académica como personalmente.

También quiero agradecer a todos aquellos que brindaron palabras de aliento y comprensión en momentos clave, amigos y conocidos que, mediante sus contribuciones, grandes y pequeñas, han dejado una huella perdurable en esta tesis y a lo largo de toda mi fase académica.

“Mar…”

Byron

Primeramente, quisiera darles las gracias a mi familia, quienes son el motor y el factor indispensable para lograr este objetivo, a mis padres por guiar, apoyar e incentivar mi proceso formativo, por sus esfuerzos y sacrificios sin los cuales no sería posible la culminación de esta etapa. A mi hermano por ser mi mentor y consejero en esta fase académica siendo mi ejemplo a seguir, brindándome su ayuda incondicional y los motivos para superarme a nivel personal y académico.

A Byron, mi compañero de tesis, por ser parte de este trabajo, por su gran dedicación y apoyo para la construcción de este proyecto de grado. A Daniela Rendon por aconsejarnos y darnos a mi compañero y a mi sus conceptos médicos para llevar adelante este trabajo.

A las amistades que he logrado formar en esta etapa de mi vida, en especial a Joseph, Cesar, Roberth, Paola y Tannia que me acompañaron y estuvieron presentes en el transcurso de estos años formativos.

Esteban

AGRADECIMIENTOS

Agradecemos a Dios y a nuestras familias por acompañarnos, apoyarnos y aconsejarnos en el transcurso de esta etapa de nuestras vidas y la elaboración de este proyecto de grado que busca contribuir y ser de ayuda para la sociedad.

Agradecemos a nuestros compañeros y amigos que fueron un aporte importante para culminar nuestro objetivo, por dejar unas agradables memorias en este trayecto.

Agradecemos a los ingenieros docentes del programa de ingeniería electrónica por impartir y proporcionar sus conocimientos y experiencias para desarrollarnos como profesionales al servicio de la sociedad, principalmente al ingeniero José de Jesús Salgado por ser nuestro guía y por su deber como profesor y director de nuestro proyecto de grado.

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

ECG: (Electrocardiograma) Es un examen médico que mide la actividad eléctrica del corazón, cuando este órgano late la señal eléctrica que circula a través de él es registrada, este procedimiento muestra si el corazón late a un ritmo y una fuerza adecuados[1].

OPENCV: (Open Source Computer Vision Library) es una librería de código abierto que incluye varios cientos de algoritmos de visión por computador y machine Learning; permite entre otras cosas, detectar y reconocer caras, identificar objetos, clasificar acciones humanas en video, capturar movimientos de cámara y objetos en movimiento, entre otras tantas aplicaciones[2].

GUI: (Graphic User Interface) Es un tipo de interfaz de usuario que permite a los usuarios interactuar con un sistema computacional usando elementos gráficos como lo pueden ser ventanas, iconos, menús y botones, en lugar de interfaces basadas en texto. Facilita a los usuarios la interacción con aplicaciones de software, la navegación a través del sistema, y llevando a cabo tareas de una manera más intuitiva[3].

PX o **px:** El píxel es la menor unidad básica de una imagen digitalizada en pantalla, con base de puntos de color o en escala de grises. Las imágenes se forman como una sucesión de píxeles; esta marca la coherencia de la información visualizada. Es la unidad más pequeña de una imagen digital; son pequeños cuadrados o rectángulos de un color homogéneo, cuando se juntan forman una imagen compleja. Los píxeles se pueden distinguir de forma fácil, solo tienes que hacer zoom a una imagen[4].

---

[1] MedlinePlus. (s.f.). Electrocardiograma (ECG). MedlinePlus. https://medlineplus.gov/spanish/pruebas-de-laboratorio/electrocardiograma/#:~:text=Un%20electrocardiograma%20(ECG)%20es%20un,y%20con%20una%20fuerza%20normal

[2] OpenCV. (s.f.). Acerca de OpenCV. OpenCV. https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,perception%20in%20the%20commercial%20products

[3] ONOS. (s.f.). ¿Qué es una GUI? IONOS Digital Guide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-gui/#:~:text=Una%20graphical%20user%20interface%20o,el%20manejo%20del%20usuario%20humano

[4] Rosvel. (s.f.). ¿Qué es un píxel? Rosvel Blog. https://www.rosvel.com/blog/que-es-un-pixel/

RESUMEN

La finalidad de este trabajo es la elaboración de una interfaz gráfica capaz de realizar la detección de enfermedades cardiacas en electrocardiogramas producidos por el dispositivo Kardia 6L a través del lenguaje de programación Python orientado a la visión artificial o por computador. El presente proyecto busca una mejora importante en la entrega de diagnósticos precisos y efectivos relacionados a cardiopatías.

Para realizar de manera correcta lo anteriormente planteado es necesario llevar a cabo un proceso de digitalización de las señales eléctricas suministradas por el electrocardiógrafo. Mediante la aplicación de filtros, operaciones morfológicas y demás técnicas de visión por computador a la imagen que contiene las mediciones cardiacas se obtienen finalmente señales equivalentes registradas de manera digital para un posterior análisis matemático el cual permitirá la detección de los posibles padecimientos en el corazón que pueda presentar un paciente.

La interfaz gráfica consta de una ventana con cuatro vistas conectadas entre sí mediante botones; la primera ventana contiene la información general de la interfaz, en la segunda se encuentra la ventana donde se carga el archivo en formato pdf del electrocardiograma, en la tercera la visualización del electrocardiograma ya digitalizado y los valores de onda calculados y en la cuarta los resultados con las posibles enfermedades diagnosticadas.

Este trabajo representa un avance significativo en el área de la digitalización de señales a través de la visión por computador y en la detección de enfermedades en el ámbito medico; impulsando de esta manera el uso de nuevas e innovadoras tecnologías en la región huilense y el país.

ABSTRACT

The purpose of this work is to develop a graphical interface capable of detecting cardiac diseases in electrocardiograms produced by the Kardia 6L device using the Python programming language oriented to artificial or computer vision. The present project seeks a significant improvement in the delivery of accurate and effective diagnoses related to heart disease.

In order to correctly perform the above mentioned, it is necessary to carry out a digitization process of the electrical signals supplied by the electrocardiograph. By applying filters, morphological operations and other computer vision techniques to the image containing the cardiac measurements, equivalent digitally recorded signals are finally obtained for subsequent mathematical analysis, which will allow the detection of possible heart conditions that a patient may present.

The graphical interface consists of a window with four views connected to each other by means of buttons; the first window contains the general information of the interface, in the second is the window where the file in pdf format of the electrocardiogram is loaded, in the third the visualization of the already digitized electrocardiogram and the calculated wave values and in the fourth the results with possible diagnosed diseases.

This work represents a significant advance in the area of digitalization of signals through computer vision and in the detection of diseases in the medical field, thus promoting the use of new and innovative technologies in the region of Huelva and the country.

INTRODUCCIÓN

La medicina es una rama científica de estudio que se encarga de prevenir, detectar y curar padecimientos que afectan el correcto funcionamiento del cuerpo y la calidad de vida humana. Debido al desarrollo tecnológico que ha tenido la misma a través de los años, existen dispositivos como el electrocardiógrafo, capaces de monitorear el desempeño del corazón.

Las enfermedades relacionadas con el corazón son, desde hace más de dos décadas, la causa principal de muertes alrededor del mundo[5], por lo que es necesaria la realización de estudios e investigaciones que permitan mitigar, prevenir y detectar tales padecimientos presentes en la población. Los electrocardiogramas permiten identificar anomalías en la actividad cardiaca de una persona para posteriormente ser diagnosticada y tratada de manera correcta y eficaz.

Debido a la sencillez y eficacia de este examen médico, con el paso del tiempo se ha optimizado de tal manera que existen dispositivos portátiles capacitados para realizar dichas mediciones, como la serie de instrumentos elaborados por la empresa Alivecor llamados “Kardia”, los cuales suministran resultados de electrocardiogramas de hasta 6 derivaciones[6].

La necesidad de diagnosticar de manera adecuada estos resultados es imprescindible para que los pacientes reciban la atención y los cuidados adecuados, para de esta forma disminuir los índices elevados de tasas de mortalidad que representan estas causas. Es por esto por lo que, debido a la gran variedad de herramientas tecnológicas que ha desarrollado el ser humano en todos los ámbitos, se han creado técnicas y procedimientos para la detección y digitalización de señales físicas de la naturaleza, como los pulsos eléctricos del corazón, para un posterior estudio y análisis[7].

El trabajo que se presenta en este documento busca ser de apoyo a trabajos previos relacionados, con la adición de la utilización de tecnología moderna, planteando como objetivo primordial la creación de una interfaz gráfica para el reconocimiento y detección de características en electrocardiogramas mediante técnicas de visión por computador, desarrollando una interacción que beneficia de manera importante el diagnostico efectivo de padecimientos relacionados a cardiopatías.

---

[5] OPS/OMS. (2020, Diciembre 9). OMS revela las principales causas de muerte y discapacidad en el mundo para 2000-2019. Organización Panamericana de la Salud. https://www.paho.org/es/noticias/9-12-2020-oms-revela-principales-causas-muerte-discapacidad-mundo-2000-2019

[6] AliveCor. (s.f.). KardiaMobile 6L. AliveCor España. https://www.alivecor.es/kardiamobile6l

[7] A. BENHAMIDA and M. KOZLOVSZKY, "Human ECG data collection, digitalization, streaming and storing," 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2020, pp. 105-110, doi: 10.1109/SAMI48414.2020.9108765.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Desarrollar e implementar una interfaz gráfica para dar soporte a diagnósticos médicos reales, realizando reconocimiento y detección de características de señales de electrocardiogramas reales en formato PDF mediante el uso de Python en sistemas de código cerrado.

1.2 OBJETIVOS ESPECÍFICOS

- Diseñar un algoritmo capaz de digitalizar resultados de electrocardiogramas presentados en formato PDF mediante el uso de herramientas de código abierto como OpenCV.
- Crear algoritmos aptos para la detección de enfermedades cardiacas tales como taquicardias, bradicardias, dilataciones, pericarditis, isquemias entre otras.
- Realizar un esquema de validación de resultados que permita probar la efectividad de los algoritmos elaborados.

2. FUNDAMENTOS BÁSICOS

2.1 ELECTROCARDIOGRAMA

El electrocardiograma es un registro que refleja la actividad eléctrica del corazón, realizado con la ayuda de un aparato conocido con el nombre de electrocardiógrafo[8]. Este examen proporciona información sobre la parte del corazón que se activa con cada latido cardiaco, las vías de conducción nerviosa del corazón y también la frecuencia y ritmo cardiacos. Estas pruebas se realizan cuando se tienen sospechas de alguna dolencia cardiaca. También se suelen practicar como chequeo o exploración física normal en personas de mediana edad y edad avanzada, aun cuando no se tenga evidencia de alguna dolencia coronaria. Desde su invención en 1902 por Willem Einthoven, hace más de un siglo, se ha constituido en la herramienta diagnostica más utilizada para dar un enfoque inicial de las enfermedades cardiovasculares[9].

Para llevar a cabo el ECG, el examinador coloca electrodos (sensores redondos que se adhieren a la piel) sobre el tórax, los brazos y las piernas del paciente, a cada una de las ubicaciones donde son posicionados los electrodos se les conoce como derivaciones. Estos elementos miden la magnitud y la dirección de las corrientes eléctricas del corazón durante cada latido. Estos se encuentran conectados por cables a una máquina que produce un registro especifico (trazo), que varía según la ubicación del electrodo. Cada trazo muestra el desempeño del corazón desde distintos ángulos; estos registros constituyen el electrocardiograma[10].

Si bien el examen representa un riesgo nulo y un buen suministro de información acerca del funcionamiento del corazón, es necesario un profesional de la salud que interprete y diagnostique los trazos registrados, por lo que existe la posibilidad de que existan errores o fallos humanos a la hora de dictaminar un resultado. El presente proyecto busca brindar un apoyo a estas valoraciones medicas para así dar un tratamiento adecuado a los pacientes.

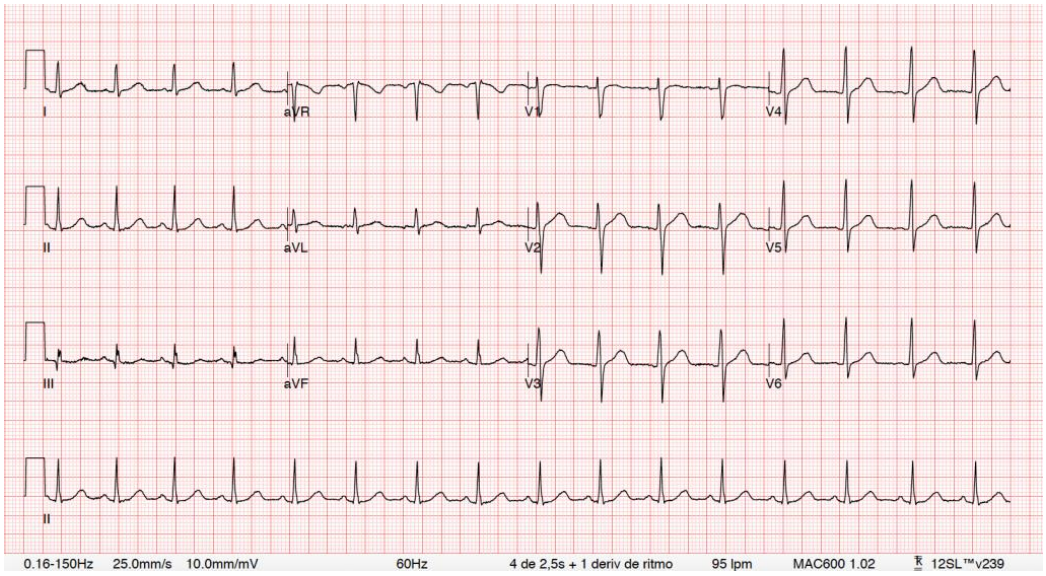
---

[8] Uríbe Arango, W., Duque Ramírez, M., & Medina Durango, E. (s.f.). Electrocardiografía y Arritmias. Siocardio. URL: https://www.siacardio.com/wp-content/uploads/2015/01/Libro-EKG-y-Arritmias-WU.pdf

[9] Muñoz V., A. (s.f.). Electrocardiografía Básica. Repositorio Universidad del Rosario. Disponible en: https://repository.urosario.edu.co/server/api/core/bitstreams/b891abb3-2690-4551-a0c1-b16f41a17427/content

[10] Manual Merck. (s.f.). Electrocardiografía. Manual Merck. https://www.msdmanuals.com/es/hogar/trastornos-del-coraz%C3%B3n-y-los-vasos-sangu%C3%ADneos/diagn%C3%B3stico-de-las-enfermedades-cardiovasculares/electrocardiograf%C3%ADa

Figura 1. Hoja de electrocardiograma



Fuente: Arteris Médica[11]

2.2 VISIÓN POR COMPUTADOR

La visión por computador es un campo de estudio que se enfoca en permitir que los computadores interpreten y entiendan información visual del mundo que los rodean. Implica el desarrollo de algoritmos y técnicas que permiten a las maquinas reconocer y clasificar objetos, personas y otras características visuales en imágenes y/o videos[12].

La visión por computador opera de manera similar a la visión humana. A diferencia de tener retinas, nervios ópticos, una corteza visual y una vida de contexto para entrenar la diferenciación de objetos, su proximidad, si se encuentran en movimiento o si hay alguna falla en los mismos; las computadoras se entrenan con cámaras, datos y algoritmos en un tiempo mucho menor, por lo que eventualmente estas técnicas superan rápidamente las habilidades y capacidades humanas[13].

En la visión por computador es necesario el suministro de información adecuada y correcta para que las maquinas, de la misma forma, alcancen un alto grado de exactitud en sus respectivas tareas.

[11] Arteris Médica. (s.f.). Electrocardiograma. [Imagen]. En: ¿Qué es un electrocardiograma? Arteris Médica. https://www.arterismedica.com/que-es-un-electrocardiograma/
[12] DeepAI. (s.f.). Computer Vision. DeepAI Machine Learning Glossary and Terms. https://deepai.org/machine-learning-glossary-and-terms/computer-vision
[13] IBM. (s.f.). Computer Vision. IBM. https://www.ibm.com/topics/computer-vision

Esta área de estudio es necesaria para llevar a cabo el proceso de digitalización de los electrocardiogramas y la posterior extracción de los datos contenidos en los mismos, por lo que es crucial para llevar a cabo este proyecto.

2.3 KARDIA 6L

Es una versión más avanzada del electrocardiógrafo portable Kardia desarrollado por AliveCor. El dispositivo Kardia 6L provee lecturas de ECG de 6 derivaciones, lo que es una mejora con respecto a las lecturas de ECG de una sola derivación proporcionadas por el dispositivo Kardia original.

Es similar en tamaño y diseño al Kardia original, pero este tiene dos sensores adicionales que se usan para capturar las derivaciones adicionales. Para tomar una lectura de ECG, el usuario coloca sus dedos sobre los sensores del dispositivo, y los dos sensores adicionales son colocados a la izquierda y a la derecha del pecho. Luego el aparato registra una tira de electrocardiograma de 30 segundos.

El dispositivo y la aplicación que lo acompaña proporcionan lecturas de ECG más detalladas y precisas, que se pueden usar para detectar una gama más amplia de anomalías cardiacas que el dispositivo Kardia original. Puede detectar arritmias como fibrilación auricular, bradicardia, taquicardia y más, y puede usarse para monitorear la salud del corazón a lo largo del tiempo.

Al igual que el aparato original, el Kardia 6L es un dispositivo portátil y fácil de usar que pueden utilizar pacientes, médicos y proveedores de atención médica para el control y la gestión remotos de la salud del corazón. Está aprobado por la Administración de Drogas y Alimentos de los Estados Unidos (FDA) como un dispositivo médico para detectar y monitorear arritmias cardiacas[14].

Figura 2. Kardia Mobile 6L

Fuente: AliveCor[15]

[14] Kardia. (s.f.). KardiaMobile 6L. Kardia Store. https://store.kardia.com/products/kardiamobile6l
[15] CE-TekMed. (s.f.). AliveCor KardiaMobile 6-Lead ECG for iPhone and Android. CE-TekMed. https://www.ce-tekmed.ie/product-page/alivecor-kardia-mobile-6-lead-ecg-for-iphone-and-android

2.4 PYTHON

Python es un lenguaje de programación interpretado de alto nivel que se usa ampliamente para una gran variedad de propósitos, incluido desarrollo web, análisis de datos, inteligencia artificial, computación científica y más[16]. Algunos conceptos claves de Python son:

- Tiene una sintaxis simple y sencilla de leer que hace énfasis en la legibilidad y una barrera baja de entrada para nuevos programadores.
- Permite una alta gama de tipos de datos integrados, incluidos números, cadenas de texto, listas, tuplas y diccionarios.
- Tiene construcciones para declaraciones condicionales, bucles y manejo de excepciones, que permitan que los desarrolladores controlar el flujo de sus programas.
- Soporta funciones, las cuales permiten a los desarrolladores encapsular fracciones de código reutilizables.
- Tiene un vasto ecosistema de bibliotecas y módulos que brindan funcionalidades adicionales para una amplia gama de casos de uso, como Numpy para computación científica, OpenCV para visión artificial, Pandas para análisis de datos, Django para desarrollo web y TensorFlow para aprendizaje automático.
- Soporta la programación orientada a objetos, la cual permite a los desarrolladores organizar su código en clases y objetos los cuales pueden ser reutilizados y ampliados.
- El código se interpreta en tiempo de ejecución, en lugar de compilarse de antemano, lo que permite la creación rápida de prototipos y desarrollo rápido.

2.5 OPENCV

Es una librería de software de aprendizaje automático y visión artificial de código abierto. Proporciona una gran variedad de funcionalidades y algoritmos para el procesamiento de imágenes y videos, detección de características, reconocimiento de objetos y más[17]. Algunos conceptos claves y características de OpenCV incluyen:

- Proporciona una variedad de funciones de procesamiento de imágenes, incluido el filtrado de imágenes, la mejora de imágenes y la transformación de imágenes.
- Puede detectar características en imágenes, como bordes, esquinas y manchas, utilizando varios algoritmos como el detector de esquinas de Harris.
- Es usado para reconocer objetos es imágenes y videos. Proporciona modelos reentrenados para el reconocimiento de objetos.

[16] Python Software Foundation. (s.f.). About Python. Python Software Foundation. https://www.python.org/about/

[17] OpenCV. (s.f.). Acerca de OpenCV. OpenCV. https://opencv.org/about/

- Proporciona una variedad de algoritmos de aprendizaje automático, que incluyen clasificación, regresión, agrupamiento y reducción de dimensionalidad.
- Se puede utilizar para desarrollar diversas aplicaciones de visión artificial, como realidad aumentada, reconocimiento facial, seguimiento de objetos y vehículos autónomos.

2.6 TKINTER

Tkinter es una biblioteca de Python incorporada para crear interfaces graficas de usuarios (GUI). Proporciona un conjunto de herramientas y widgets que permiten a los desarrolladores crear ventanas, menús, botones, cuadros de texto y otros elementos para crear aplicaciones de escritorio[18]. Algunos conceptos y características de Tkinter incluyen:

- Proporciona un conjunto de widgets que se pueden usar para crear varios elementos GUI, como botones, etiquetas, cuadros de texto, casillas de verificación, botones redondos, menús y muchos más.
- Provee varios métodos para gestionar el diseño de los elementos de la GUI, como pack, grid y place.
- Suministra un mecanismo para manejar eventos generados por el usuario o el sistema. Los desarrolladores pueden adjuntar una función, conocida como función de devolución de llamada, a un evento para realizar ciertas acciones.
- Proporciona cuadros de dialogo y cuadros de mensaje integrados que se pueden usar para mostrar información u obtener información del usuario.
- Provee muchas opciones para personalizar la apariencia de los widgets y las ventanas, como fuentes, colores, tamaños e imágenes.

[18] Autor. (s.f.). What Is Tk? McGill University. https://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/#WhatIsTk

3. TRATAMIENTO DEL ARCHIVO

Para el desarrollo de la interfaz se parte de un lote de datos entregados por el Kardia Mobile 6L, los datos son entregados por el dispositivo en un archivo en formato .PDF, al cual posteriormente se le realiza un tratamiento para realizar la extracción de las señales del electrocardiograma.

La base del proceso radica en el documento que se le entrega al programa en la vista dos, donde se realiza la entrada de datos. Este documento se encuentra en formato .PDF y su longitud es de 5 páginas, las cuales se encuentran estructuradas de la siguiente manera.

Figura 3. Vista rápida del documento

Kardia

Registro de ECG

Fecha de nacimiento:
SEXO: Hombre

ECG Resumen de la grabación

Determinación de Kardia:
Ritmo sinoauricular normal
*Se ha realizado la determinación de Kardia en la derivación I.

Grabado en: miércoles, 24 de mayo de 2023, 2:17:42 p.m.
Frecuencia cardiaca: 94 LPM
Duración: 30s

Información adicional

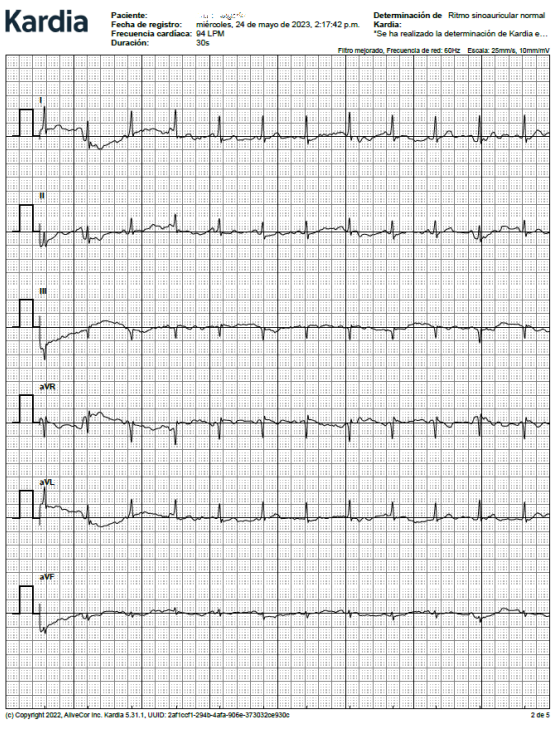
No hay información adicional que mostrar

Kardia does not check for heart attack. If you believe you are having a medical emergency, call emergency services. AliveCor does not provide medical advice or services, and any information from AliveCor is provided to assist you and your doctor with your medical care and not as a replacement for consulting with your doctor.

2022 AliveCor, Inc.
Confidential patient information

AliveCor

1 de 5



Kardia

Paciente:
Fecha de registro: miércoles, 24 de mayo de 2023, 2:17:42 p.m.
Frecuencia cardiaca: 94 LPM
Duración: 30s

Determinación de Kardia: Ritmo sinoauricular normal
*Se ha realizado la determinación de Kardia e...

Filtro mejorado, Frecuencia de red: 60Hz Escala: 25mm/s, 10mm/mV

I
II
III
aVR
aVL
aVF

(c) Copyright 2022, AliveCor Inc. Kardia 5.31.1, UUID: 2af1ccf1-294b-4afa-906e-373032de930c

2 de 5

Fuente: Autores

La primera página ofrece al lector una descripción de la información del paciente en caso de que la toma del ECG se realice desde una cuenta *(Figura 3)*, para el caso de un perfil de invitado se omite esta sección y solo se agrega que el origen de la información es desconocido *(Figura 4)*, posteriormente se encuentra el resumen del

ECG, donde se muestra el diagnóstico aproximado que realiza Kardia, así como también la derivación utilizada para la emisión de dicho diagnóstico.

Figura 4. Sección descriptiva de la página uno

Fuente: Autores

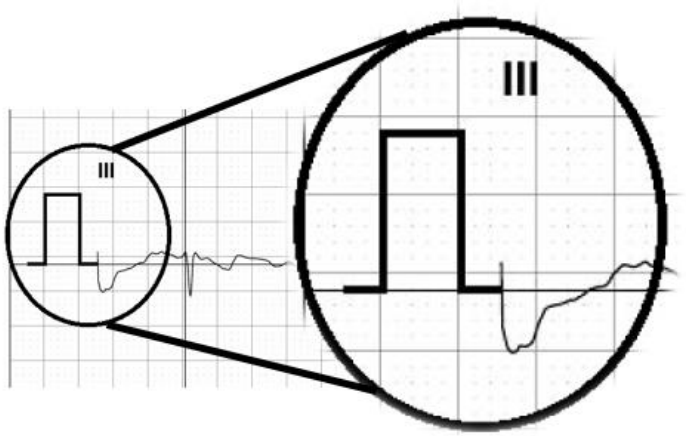
También, dentro de esta misma sección se dan algunos parámetros adicionales, tales como la fecha de la toma del ECG, la frecuencia cardiaca del paciente y la duración del ECG.

Este último parámetro es de gran importancia porque a partir de él se realizan algunas consideraciones dentro del procesamiento de la señal capturada en la digitalización.

A partir de la segunda página y hasta el final del documento, se encuentra una cuadrícula en la cual se contienen 6 gráficas, las cuales son la representación de las seis derivaciones cardiacas *(Figura 3)*. Al inicio de cada una de las gráficas, se encuentra un pulso rectangular también conocido como calibración estándar, el cual consta de 10 mm de altura por 5 mm de ancho, este trazo se obtiene cuando el sistema simula un pulso de 1 mV por 0,2 segundos[19] como se puede observar en la Figura 5.

[19] My-EKG. (s.f.). Papel EKG - My-EKG.com. https://www.my-ekg.com/generalidades-ekg/papel-ekg.html

Figura 5. Calibración estándar del ECG

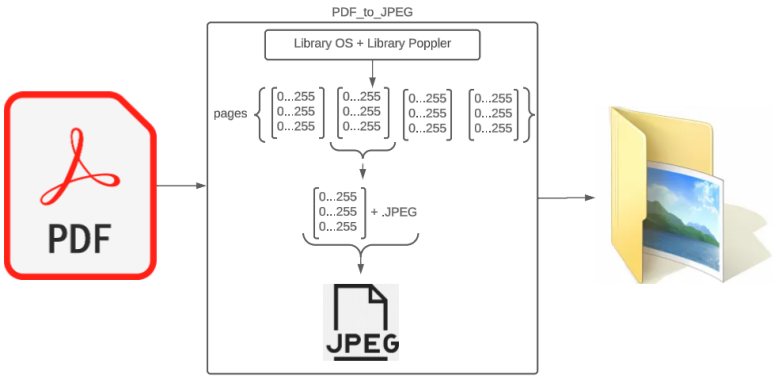


Fuente: Autores

Una vez identificada la estructura del documento, se procede a realizar un proceso de conversión en la estructura de los datos, en la cual se tiene por objetivo convertir un documento de formato PDF en un grupo estructurado de imágenes, desde la cuales se pueda realizar la extracción de la información.

Para esta finalidad nace la función “PDF_to_jpeg”, la cual va a recibir como argumento de entrada la ruta del archivo, para posteriormente y con ayuda de la librería poppler, convertir cada página del PDF en un archivo de formato .JPEG como se muestra en el flujo de la Figura 6.

Figura 6. Diagrama funcional de la función PDF_to_JPEG

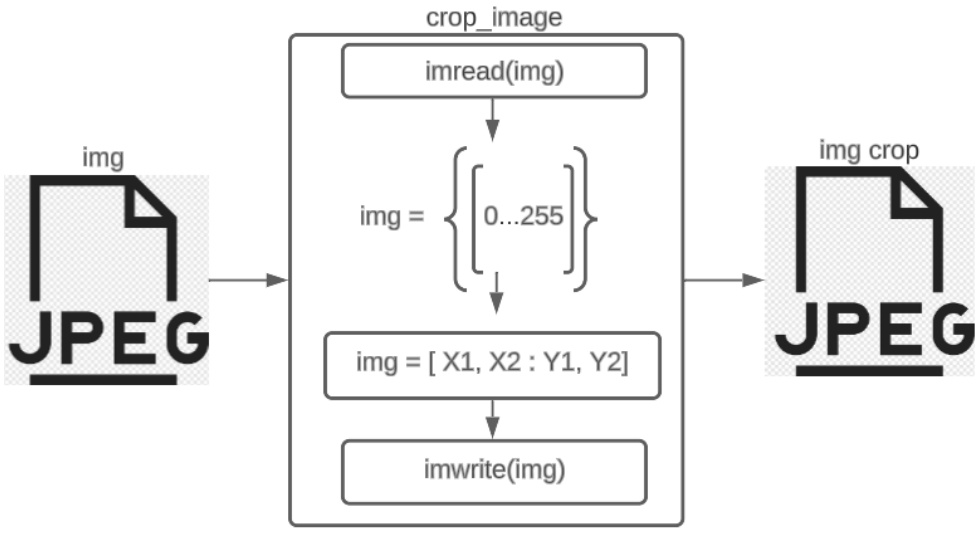


Fuente: Autores

Una vez se finaliza la ejecución de la función, se procede a realizar el recorte de las imágenes, para este proceso se encontraban dos alternativas disponibles; mediante el primera, se realiza el reconocimiento del recuadro general mediante herramientas de la librería OpenCV, pero al momento de analizar los resultados de salida, se genera gran variabilidad en los valores que encierran las esquinas.

La segunda opción llevada a cabo consistió en el proceso de inspección manual y detección visual de los pixeles que representaban las esquinas del recuadro, a lo cual, tras analizar cada una de las 4 imágenes se obtiene información concluyente que apunta a que cada los vértices del recuadro se encuentran en la misma posición para cada caso, por lo cual, para evitar carga computacional realizando la detección de vértices para cada página, se opta por esta opción, definir de manera manual los vértices dentro de una función llamada “crop_image” *(Figura 7)*.

Figura 7. Diagrama funcional de la función crop_image



Fuente: Autores

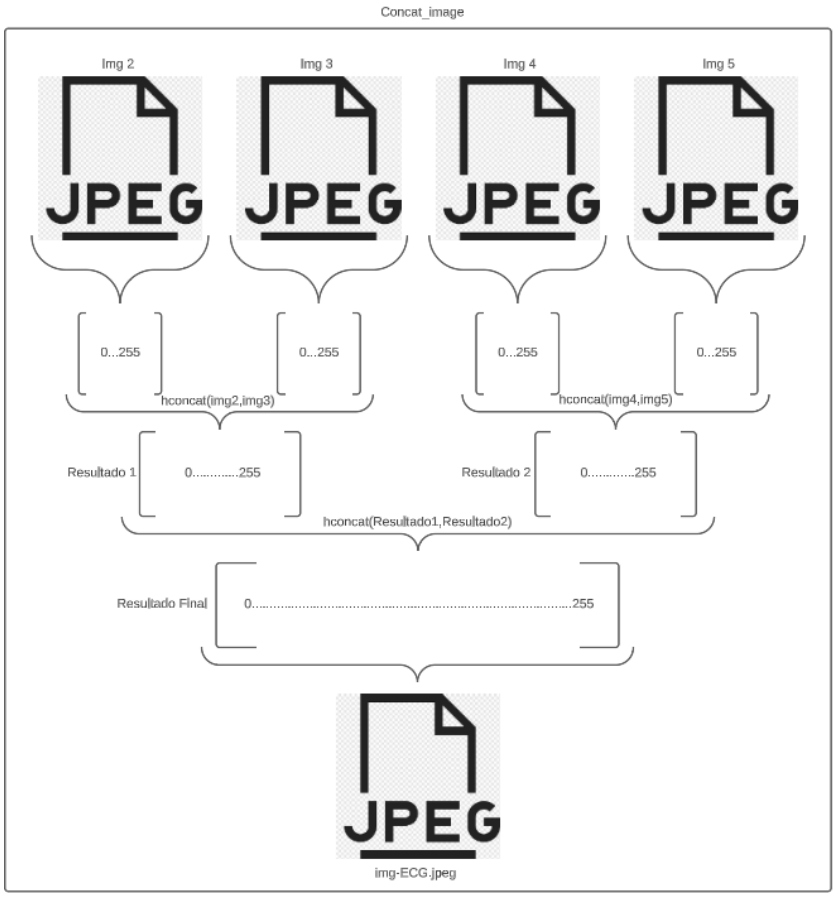
Una vez realizados todos los recortes de las imágenes, se puede realizar el proceso de concatenación de estas, con este proceso se concatenan una a una las imágenes mediante la función “hconcat” de la librería de OpenCV.

La función hconcat realiza la fusión de imágenes de forma horizontal, es decir, una imagen junto a la otra, bajo la única condición de que las dos imágenes deben de compartir el mismo alto (height)[20].

[20] OpenCV. (s.f.). Documentación de OpenCV: group__core__array. OpenCV. https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga4676b1376cdc4e528dab6bd9edc51c1a

Con base en este principio, se realiza la construcción de la función “Concat_image” (*Figura 8),* dentro de la cual se realiza el cargue de las cuatro imágenes y se hacen tres procesos de concatenación, en el primero se realiza la concatenación del recorte de la página dos con la página tres, en el segundo se realiza la concatenación del recorte de la página cuatro con la página cinco, y por último se realiza la concatenación de los resultados de los dos procesos anteriores.

Figura 8. Diagrama funcional de la función Concat_image



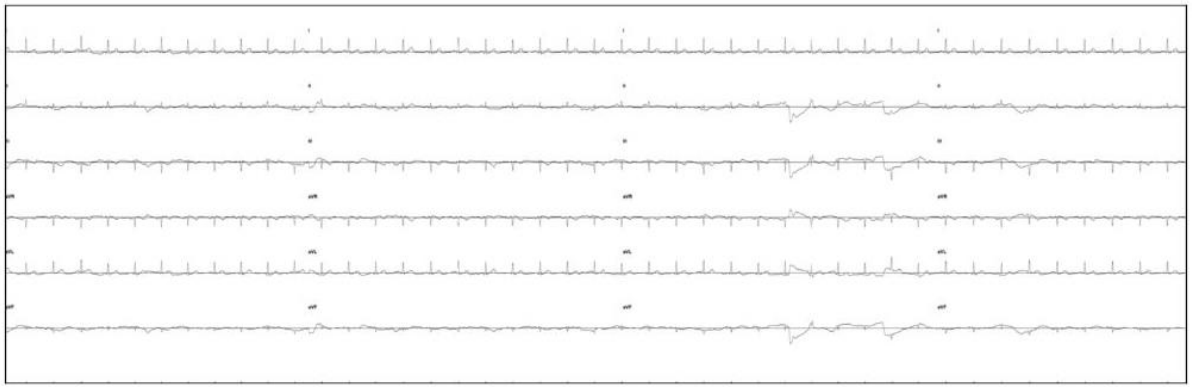
Fuente: Autores

4. DIGITALIZACIÓN DE LAS DERIVACIONES CARDIACAS

Hasta el punto actual del desarrollo, solo se ha realizado el tratamiento de los datos a un nivel básico, en el cual no se ha llevado a cabo una modificación que los afecte significativamente. A partir de este punto se inicia el tratamiento de la imagen final con el fin de separar la información relevante de la misma como se ve en la Figura 9.

El primer lote de información residual son las leyendas de la imagen, para este proceso se desarrolla una función de nombre “Limpiar imagen”, con esta función se realiza la eliminación de cada uno de los 24 bloques de texto existentes en la imagen.

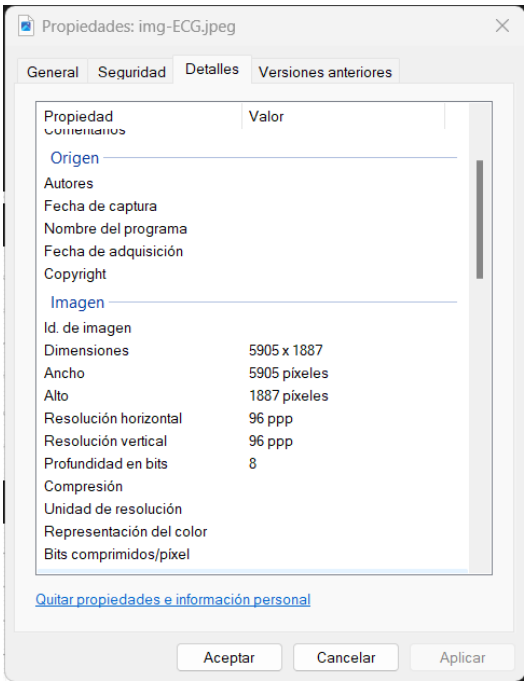
Figura 9. Vista de la imagen concatenada



Fuente: Autores

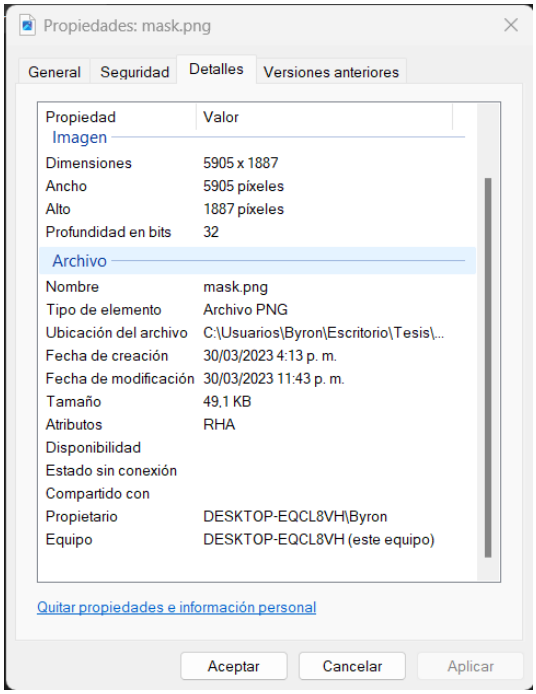
Para este caso se desarrolla una máscara de tamaño definido *(Figura 11)* y general para cualquier imagen de entrada. Se sabe que la capacidad de la máscara para ser global radica en el hecho de que el tamaño de la imagen final está determinado por constantes definidas durante el proceso de corte, lo que genera como resultado final una imagen de 5905x1887 pixeles como se ve en la Figura 10.

Figura 10. Metadatos de la imagen final



Fuente: Autores

Figura 11. Metadatos de la imagen máscara



Fuente: Autores

Ya teniendo definida la máscara *(Figura 12)*, se puede pasar sobre la imagen final, lo que da como resultado una imagen completamente libre de los bloques de texto.

Figura 12. Máscara definida para la eliminación de leyendas

Fuente: Autores

El proceso de pasar una imagen sobre la otra se hace mediante una evaluación de la máscara, manejando un espacio de color en escala de grises[21] como se muestra en la Figura 13.

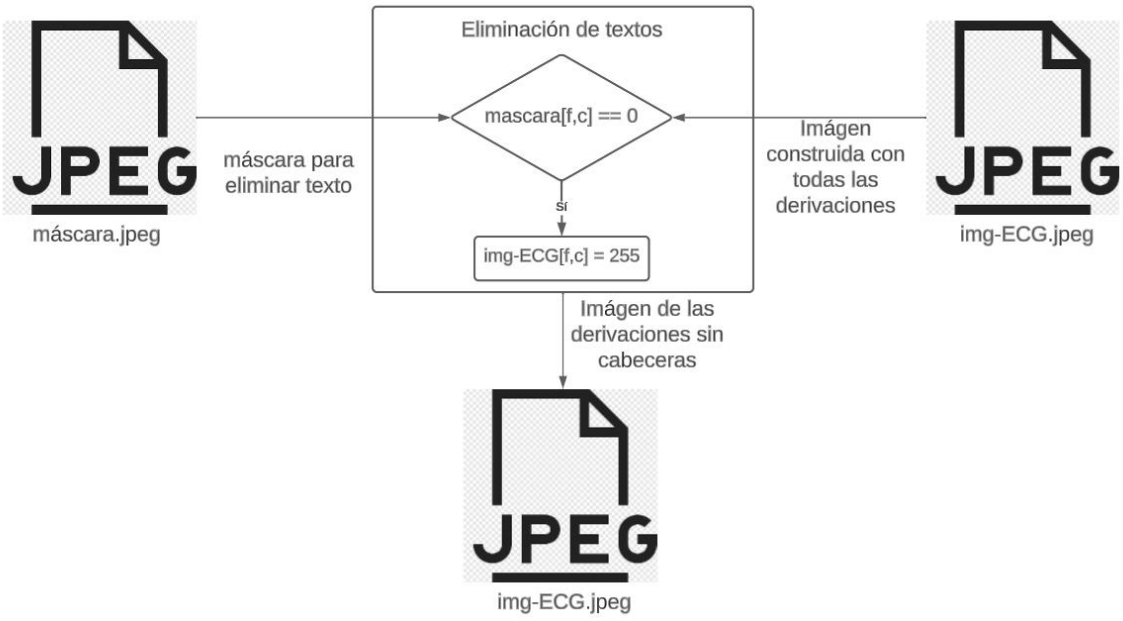
Figura 13. Escala de grises

Fuente: Eduspace[21]

La máscara se recorre mediante dos índices, uno para la fila (f) y otro para la columna (c), y se realiza el condicional para determinar si el píxel de la máscara ubicado en (f, c) corresponde a un píxel negro o con un valor de 0 en la imagen objetivo, el píxel con estas mismas coordenadas es reemplazado por un píxel blanco o un valor de 255, como se ilustra en la siguiente imagen.

[21] Agencia Espacial Europea (ESA). (s.f.). Eduspace: Unidad de Recursos de Educación Espacial. ESA. https://www.esa.int/SPECIALS/Eduspace_ES/SEMCDX3FEXF_2.html

Figura 14. Diagrama de flujo de la eliminación de textos



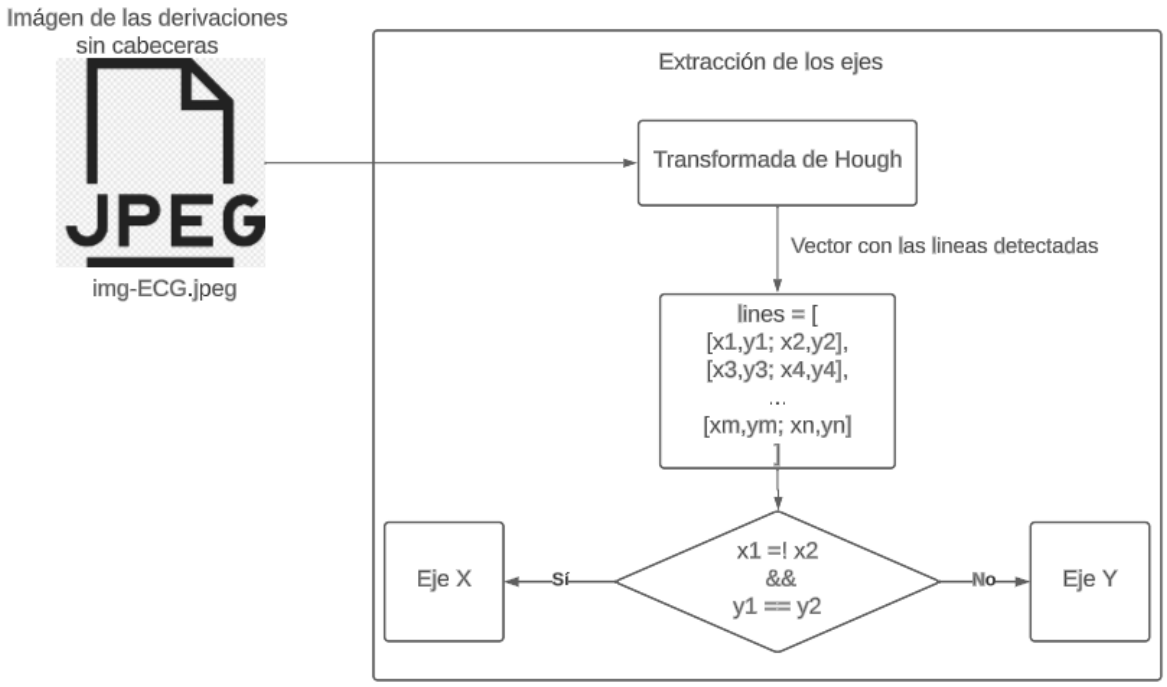
Fuente: Autores

Posterior a la eliminación de los textos, se procede a realizar la detección de los ejes de cada señal, este proceso se hace mediante la aplicación de la transformada de Hough, la cual se usa para encontrar puntos alineados que puedan existir en la imagen, es decir, puntos en la misma que satisfagan la ecuación de la recta, para distintos valores de rho (ρ) y theta (θ)[22].

Este proceso, retorna un conjunto de posibles líneas, que para este caso se consideran como los ejes de las derivaciones, tanto verticales como horizontales, cabe aclarar que no retorna el valor exacto o el valor del pixel en el que se ubica el eje, lo que entrega es un conjunto de puntos aproximados al valor donde se encuentra el eje, para redondear este proceso primero se realiza la separación de los posibles ejes en dos vectores, donde se acumularan los puntos que presentan variaciones en y (ejes x) y los que presentan cambios en x se acumularan en el vector ejes y como se muestra a continuación.

[22] Universidad de Jaén. (s.f.). Práctica 4: Visión por Computadora. [Documento PDF]. Universidad de Jaén. http://www4.ujaen.es/~satorres/practicas/practica4_vc.pdf

Figura 15: Diagrama de flujo de la extracción de los ejes



Fuente: Autores

Luego de este proceso ya se tienen separados los valores de los ejes, mas no están precisados, para subsanar esta falla, se usa el mismo proceso para los dos vectores, lo primero que se hace es aplicarle el método ".sorted()"[23] que se encarga de ordenarlos de menor a mayor, ya obtenido el vector organizado se crea un vector auxiliar que se usará más adelante.

Ya teniendo el vector ordenado, se pasa por un doble ciclo while, con el primero se realiza el recorrido del vector y con el segundo se evalúa que la distancia medida en pixeles desde el píxel n hasta el píxel n+1 no sea superior a 10 pixeles, de cumplirse esto, el píxel n es agregado al vector auxiliar y en caso de que no se cumpla, se calcula el promedio del vector auxiliar y se redondea al entero por encima del mismo, y es este dato es el que se almacena en el vector final de los ejes.

Una vez determinados los ejes de las derivaciones, se hace uso de los ejes en x para realizar el corte de la señal, este corte se realiza aplicando los principios de

[23] Python Software Foundation. (s.f.). Cómo hacerlo: Ordenar. Documentación de Python 3. https://docs.python.org/es/3/howto/sorting.html

slicing[24] que tiene Python, según los cuales para extraer un segmento de un vector basta con usar la instrucción vector[a:b], donde vector corresponde al nombre del vector objetivo, y haciendo uso de las llaves cuadradas se indican dos números enteros en los cuales, el primero (a), será el punto de inicio del slicing, cabe recalcar que este índice es de tipo incluyente, el cual se encuentra separado del segundo índice (b) por medio de dos puntos (":"), para el caso del índice b, se está señalando el punto final del slicing, sabiendo que el índice b es de tipo excluyente.

En el recorte se da como objetivo que la imagen final tenga por centro el eje x y 150 pixeles por encima y por abajo, los cuales abarcarían el espacio comprendido por el trazado de cada derivación cardiaca, dichos recortes se almacenan en imágenes separadas que llevan por nombre la derivación que contienen. Con esto, al finalizar el proceso, de una imagen se obtienen 6 imágenes.

Una vez individualizadas las derivaciones, se procede a realizar la vectorización, para este proceso, lo primero que se hace es una etapa de limpieza, dentro de esta se realiza la eliminación de las líneas que representan los ejes, en esta etapa se usan dos métodos diferentes, uno para el eje X y otro para el eje Y, para la eliminación de los ejes X, se aplica un análisis de la concentración de pixeles por columna en la imagen, este proceso carga la imagen y la convierte en un numpy array, lo cual hace que sea más fácil realizar la manipulación de esta gracias a los métodos que ofrecen.

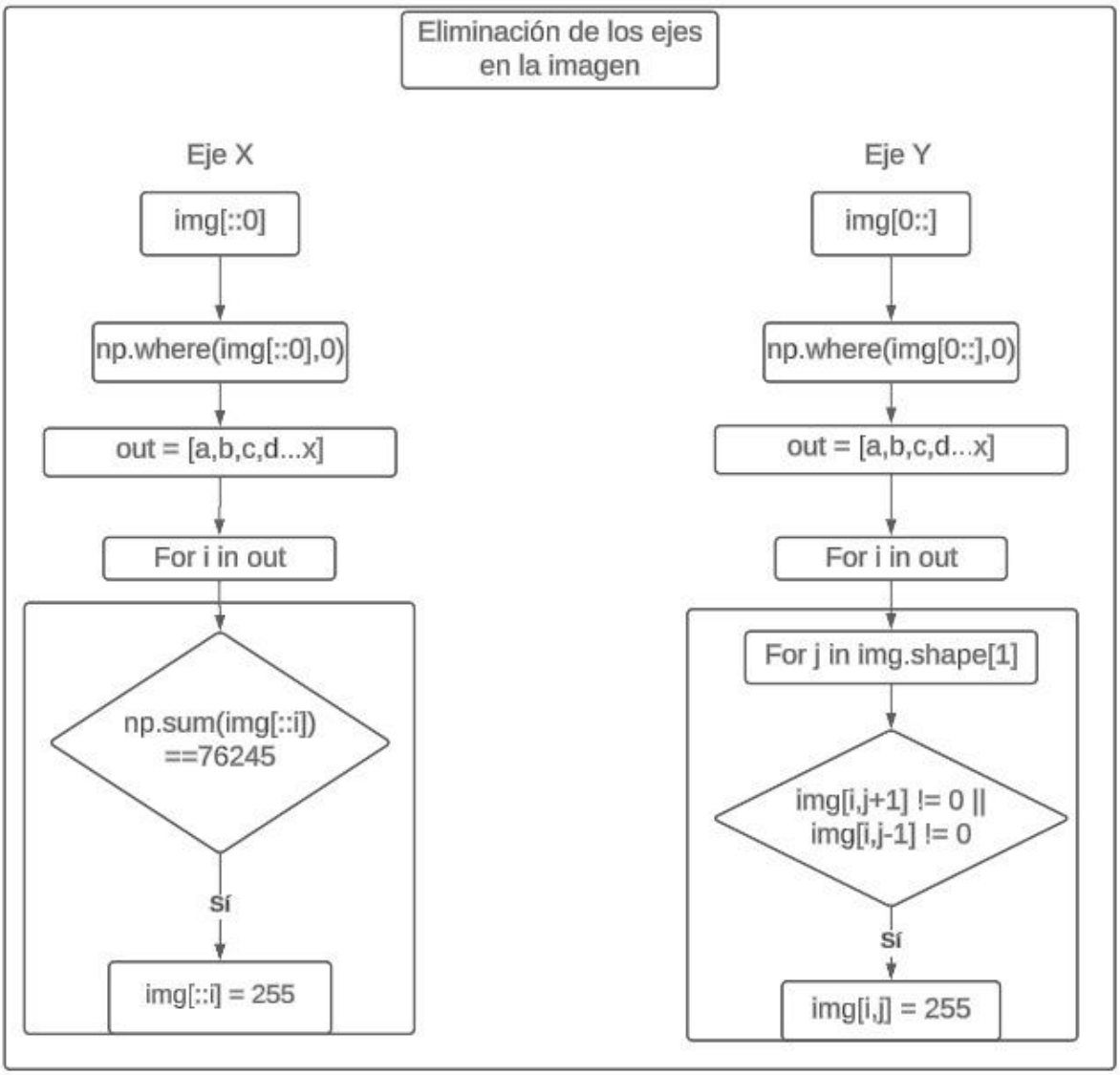
Una vez se tiene la imagen como un numpy array, se realiza el conteo de los ceros que hay en la primera fila de la imagen, donde cada cero idealmente corresponde con cada eje Y de la señal, una vez contados mediante el método np.where(), se realiza un condicional en el cual se evalúa si la cantidad de elementos contados es igual a 30, si se cumple, se hace la eliminación de la línea reemplazando sus valores de 0 (negro) a 255 (blanco), para este proceso se parte del vector de ceros detectados en la primera fila, donde se ingresa a un condicional que considera si la suma (np.sum()) de la columna donde se detectó el cero es de 76245, si es así se realiza el cambio.

Y para la eliminación del Eje X, se realiza la consideración de la posición del eje x mediante la búsqueda de los ceros dentro de la primera columna de la imagen, al detectar ese valor, se procede a recorrer el vector a lo largo de este punto indagando si en algún momento se acerca un pixel con valor cero, esto para saber si en algún momento la señal se pone al nivel del eje y no realizar cortes durante el proceso de borrado del mismo como se aprecia expone en la Figura 16, con esto se realiza la individualización de la señal, obteniendo que solo los pixeles de equivalencia 0 serían los representativos de la señal. Para recoger estos valores basta con barrer la imagen por columnas detectando la posición de los ceros y almacenándola en un

[24] Cupi2-IP. (s.f.). Sección 3.4: Slicing. [Documento web]. https://cupi2-ip.github.io/IPBook/nivel3/seccion3-4.html

vector final denominado signal, este proceso se complementa con una serie de condicionales que ayudan a determinar por cual extremo se realiza la aproximación a la señal para detectar los picos de la forma más detallada posible, ya luego de tener recogidas las posiciones de la señal, basta con restarle la posición del eje X, para poder determinar la concentración de pixeles que hay entre el eje y el punto determinado en la misma.

Figura 16: Diagrama de flujo de la eliminación de los ejes

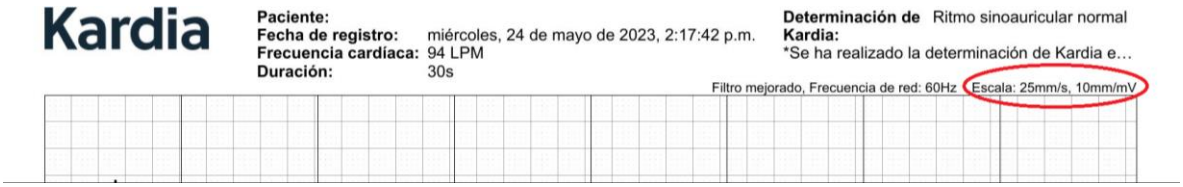


Fuente: Autores

Una vez se tiene el vector final con la consideración de la densidad de pixeles, se puede decir que se ha obtenido la señal exitosamente, pero falta un paso de gran importancia, el cual es la conversión de los valores de los pixeles a la unidad

deseada. Para llegar a esto se realiza un análisis de una de las primeras imágenes creadas, donde se buscará la resolución que ofrece el electrocardiograma *(Figura 17)*, mediante la cual se despejará el valor de los pixeles.

Figura 17. Resolución / Escala del ECG

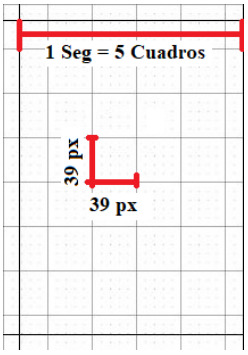


Fuente: Autores

Ya con la resolución del ECG determinada, se puede hacer uso de la misma imagen y se realiza la medición de la distancia en pixeles que hay en uno de los cuadrados pequeños que conforman el papel del trazado.

Del proceso de medición se realizaron múltiples medidas obteniendo una media de 39x39 pixeles *(Figura 18)*. Una vez obtenidas las dimensiones se pueden realizar las respectivas reglas de tres para determinar los valores en tiempo y voltaje de cada píxel.

Figura 18. Parámetros del papel del trazado



Fuente: Autores

Para determinar el valor del píxel en tiempo:

$$25\ mm \rightarrow 1\ Seg \rightarrow 5\ cuadros$$ Ecuación (1)

$$\frac{25\ mm}{1\ \cancel{Seg}} \rightarrow \frac{1\ \cancel{Seg}}{5\ cuadros} = \frac{\cancel{25}mm}{\cancel{5}\ cuadros} \rightarrow \frac{5\ mm}{1\ cuadro}$$

Con este despeje, ya se conoce que cada 39 pixeles nos representan una longitud de 5 mm.

$$\frac{5\ \cancel{mm}}{39\ px} \rightarrow \frac{1\ Seg}{25\ \cancel{mm}} = \frac{\cancel{5}\ Seg}{\cancel{975}\ px} \rightarrow \frac{1\ Seg}{195\ px} \rightarrow 5.128\ ms/px$$ Ecuación (2)

Para determinar el valor del píxel en voltaje se parte de conocer que un cuadro tiene de lado 5x5 mm, con base a esto se realiza el siguiente despeje.

$$\frac{5\ \cancel{mm}}{39\ px} \rightarrow \frac{1\ mV}{10\ \cancel{mm}} = \frac{\cancel{5}\ mV}{\cancel{390}\ px} \rightarrow \frac{1\ mV}{78\ px} \rightarrow 12.820\ \mu V/px$$ Ecuación (3)

Una vez obtenida la resolución en pixeles *(Tabla 1)*, se procede a multiplicar la densidad de pixeles del vector de la señal por su respectivo valor en voltios, con lo cual ya se tendrá construida la señal. Para el eje del tiempo, se crea un espacio lineal de la misma dimensión de la señal en cuanto a columnas, y se multiplica por su respectivo equivalente en segundos.

Tabla 1. Resolución medida en Pixeles

Resolución Medida en Pixeles	
1 píxel vertical	12.820 uV
1 píxel horizontal	5.128 mS

Fuente: Autores

En la finalización de todos los tratamientos de la información en bruto, se obtiene una serie de 7 vectores, que por longitud tienen la dimensión en x de la imagen.

Seis de los vectores serán la representación en amplitud de las señales en función del eje determinado para la señal, considerando la diferencia por arriba o por abajo del eje tomando como resolución el tamaño de un píxel.

El séptimo vector será la representación del eje temporal, que en un futuro será el eje x de la señal cardiaca.

El objetivo del proceso de vectorización es obtener la serie de datos tratables para una serie de procesos que permitan la caracterización de la señal en aras de identificar patrones y puntos de inflexión que apunten a detectar afecciones cardiacas.

5. DETECCION DEL PICO CARACTERÍSTICO

El proceso de detección del pico característico es de vital importancia, este pico es seleccionado en base al análisis de las señales, y se busca dentro de la derivación DI, la cual presenta la mejor afinidad y fidelidad a lo largo del electrocardiograma.

De los resultados de la búsqueda de este pico, depende en gran medida la efectividad del análisis, este pico que se determinó como característico es el pico en el cual se fijará el marcador para extraer las señales características tales como la amplitud y duración de los picos y segmentos de la señal.

Para el proceso de detección del pico, se asume el criterio de que un pico ideal esta alrededor del valor de 0.5 mV. Con este criterio en mente, se inicia una serie de condicionales para evaluar la señal y detectar el punto de valor optimo.

Lo primero que se hace es evaluar la señal para determinar si presenta valores arriba de 1.5 mV, si presenta estos niveles de voltaje, la señal es descartada y el proceso se finaliza. Si no se detectan esos niveles de amplitud, el código procede con un segundo condicional el cual evalúa si al menos un pico máximo de la señal es superior a 0.5 mV, si la condición se cumple se procede a seguir la elección del pico, de lo contario se descarta la señal y se finaliza el proceso, esto debido a que ningún pico alcanza el umbral esperado y la razón puede llegar a ser una mala toma de los electrocardiogramas que generalmente está asociada a un mal posicionamiento de los electrodos.

Si este condicional es superado, se ingresa a un filtro adicional, en el cual se evalúa que la cantidad de los picos detectados sean inferior de 150, esto con tal de limitar la cantidad de picos, si se detectan demasiados puntos, puede que el ECG presente demasiado ruido.

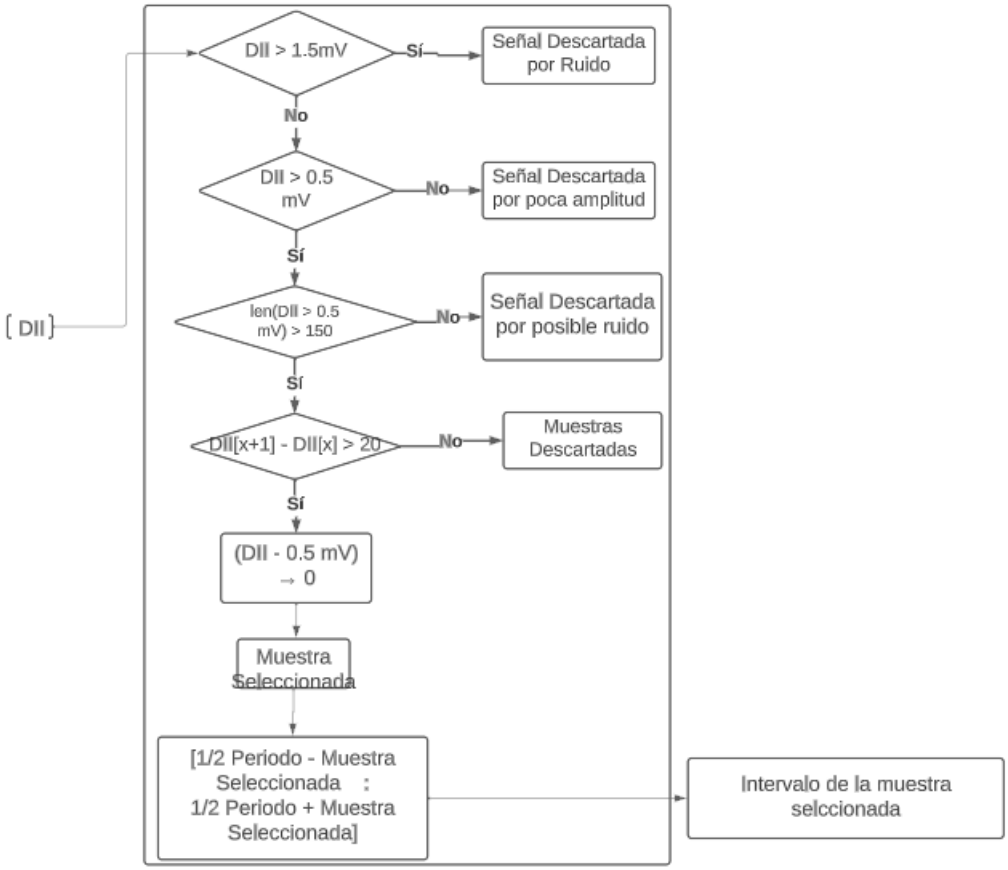
Si esta última condición se cumple, se procede a capturar todos los puntos máximos de la señal, dentro de estos máximos, se hace la diferenciación y la determinación de la distancia entre cada uno de los picos detectados, descartando los que se encuentran a menos de 20 muestras de distancia.

Una vez filtrado el vector de picos máximos, y calculadas las diferencias entre cada uno de estos, se calcula la media de la distancia entre estos picos y se logra determinar el periodo de la señal cardiaca. Basados en el vector se puede determinar de forma indirecta la frecuencia cardiaca medida en latidos por minuto teniendo en cuenta que el vector representa la cantidad de latidos en 30 segundos.

Para determinar el pico que se desea usar, al vector general se le restan 0.5 mV, y del vector resultante de este proceso se selecciona el valor más cercano a 0, de esta manera se asegura que el valor seleccionado sea el valor más próximo a los 0.5 mV que representan el pico óptimo.

Como se ve en la Figura 19, la salida del proceso es un segmento el cual contiene el pico determinado, para este proceso de corte de la señal se asume que la longitud del segmento es equivalente con el periodo de la onda cardiaca. Para construirlo, se suma la mitad del periodo cardiaco previo al momento donde se registra el pico, y la otra mitad posterior al pico y bajo estos mismos criterios se realiza el recorte del segmento del tiempo.

Figura 19. Diagrama de flujo del proceso de selección del pico



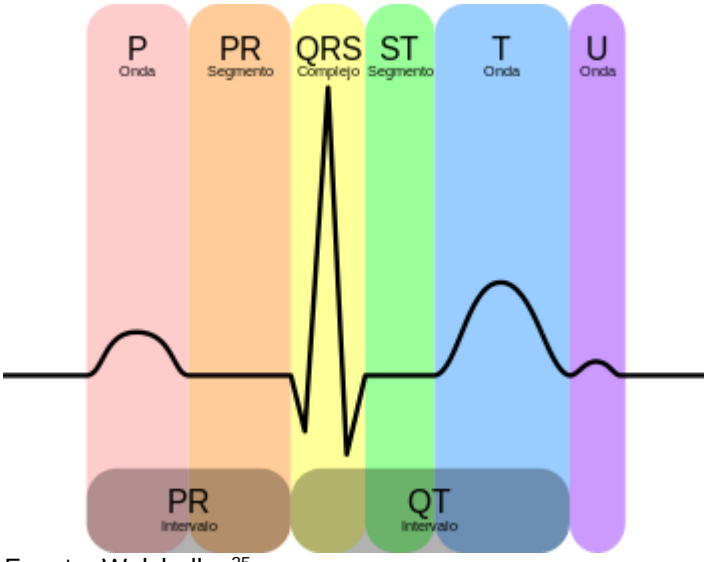
Fuente: Autores

6. CARACTERIZACIÓN DE LA SEÑAL EXTRAIDA

La etapa de mayor importancia y relevancia llega a ser el proceso de caracterización, dentro de esta etapa se realiza el análisis completo del segmento extraído, esta tiene por finalidad extraer cada uno de los parámetros característicos de una señal cardiaca. Los detalles de esta señal son los usados para determinar o realizar el cruce de los datos para determinar si existe alguna afección cardiaca detectable por la variabilidad de los parámetros.

La caracterización de la señal puede ser llevada a cabo de diversas maneras. En este caso se hace uso de dos técnicas distintas, mediante la primera se realiza el análisis de los máximos y mínimos característicos de la señal, y en base al conocimiento de la anatomía de la señal del electrocardiograma *(Figura 20)* asignar cada uno de los resultados a uno de los puntos característicos del mismo, y en el segundo se realiza un análisis de la segunda derivada de la señal para determinar los puntos descritos anteriormente.

Figura 20. Anatomía de la señal del ECG



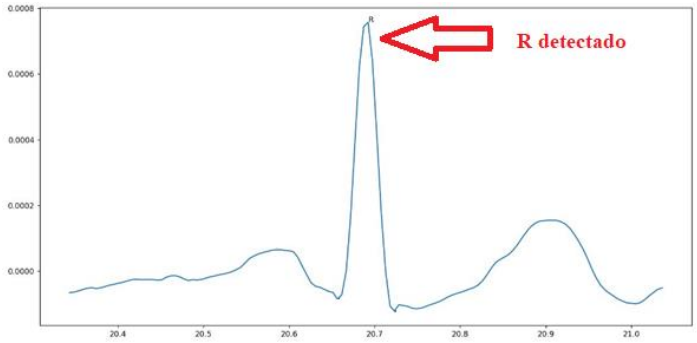
Fuente: Welchallyn[25]

[25] welchallyn, PC-based resting ECG - Hill-ROM, https://www.welchallyn.com/content/dam/welchallyn/documents/upload-docs/Catalogs/Full-Line-Catalog/Cardiopulomonary.pdf

6.1 MÉTODO 1: EXTRACCIÓN DE MÁXIMOS Y MÍNIMOS MEDIANTE EL MÉTODO NP.MAX Y NP.MIN

A través de este método se lleva a cabo la determinación de las características de la señal, lo primero que se determina es el pico representativo de la onda R. Para determinar el punto de R se aplica np.max() al segmento de señal extraído, mediante este método se encuentra el valor máximo *(Figura 21)*, de esta manera, en el punto donde se ubica este máximo, se aloja el pico de la onda R.

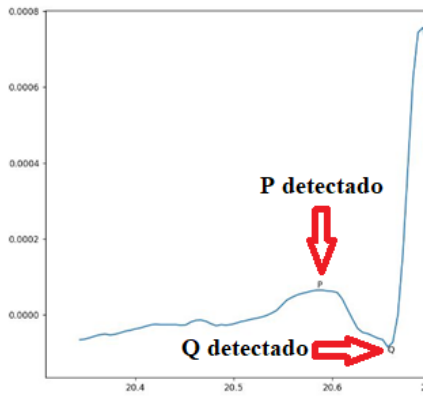
Figura 21. Detección del pico de la onda R



Fuente: Autores

Una vez detectado el pico se divide la señal en dos segmentos, uno previo y uno posterior al pico R. En el segmento previo al pico de R se detectan los momentos de las dos ondas antecesoras, P y Q, siendo P detectado, mediante la captura del pico máximo del segmento previo y Q siendo detectado como el mínimo de este segmento como se aprecia en la Figura 22.

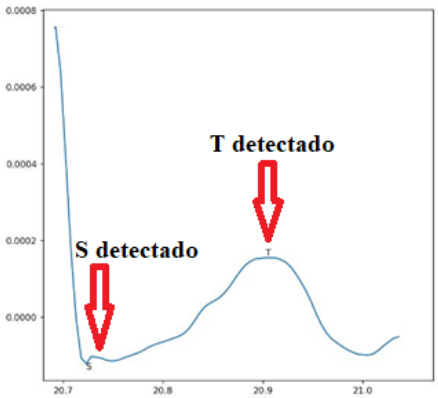
Figura 22. Ondas P y Q detectadas



Fuente: Autores

Para la detección de las dos componentes restantes, se realiza el mismo análisis en el segmento posterior, mediante la detección del mínimo, se detecta el pico correspondiente al momento de la onda S y al detectar el valor máximo se halla el punto de la onda T *(Figura 23)*.

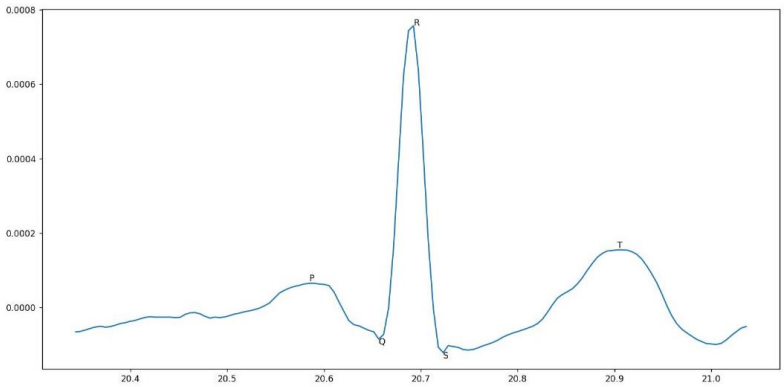
Figura 23. Ondas S y T detectadas



Fuente: Autores

Con este análisis, se pueden obtener los puntos característicos de la señal, y las amplitudes respectivas a cada punto, con estos datos se obtiene un análisis superficial pero susceptible a errores en la detección de las ondas Q y T. Dado que mediante este método se detectan los puntos máximos dentro de los segmentos, pero no se toma en cuenta la posible existencia de una onda U, la cual en su gran mayoría de apariciones apunta hacia una anomalía cardiaca.

Figura 24. Señal caracterizada por el método 1



Fuente: Autores

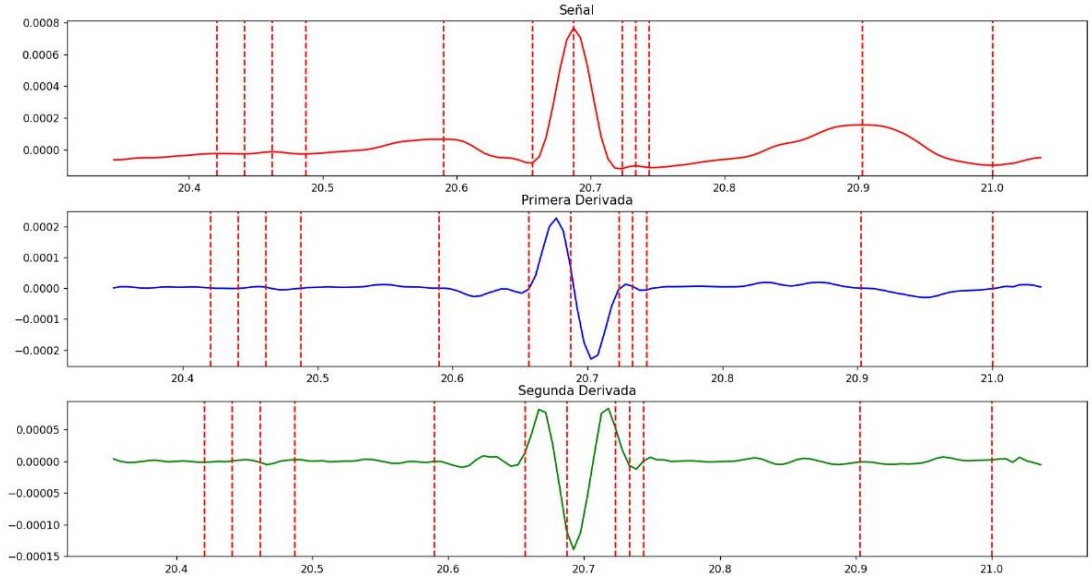
Como se puede apreciar en la Figura 24, la señal caracterizada por el método 1, se tienen fallas en la detección, por ende, se hace necesario un segundo método, mediante el cual se pueda realizar el análisis de los puntos iniciales y finales de las ondas P y T.

6.2 MÉTODO 2: ANÁLISIS DE LOS PUNTOS CARACTERÍSTICOS MEDIANTE LA 1° Y 2° DERIVADA

Mediante la implementación del segundo método se busca aumentar la precisión, así como también encontrar algunos parámetros restantes.

Lo primero que se hace es extraer la primera y la segunda derivada del vector, y en este se buscan los cambios de signo de la señal, estos a su vez son separados en un vector que contiene los puntos en los que la señal crece y empieza a decrecer o resumiendo, donde la pendiente es negativa; y en un segundo vector se almacenan los puntos donde la señal deja de decrecer y empieza a crecer o los puntos con una pendiente positiva *(Figura 25)*.

Figura 25. Gráfico de los puntos de inflexión en 1° y 2° derivada



Fuente: Autores

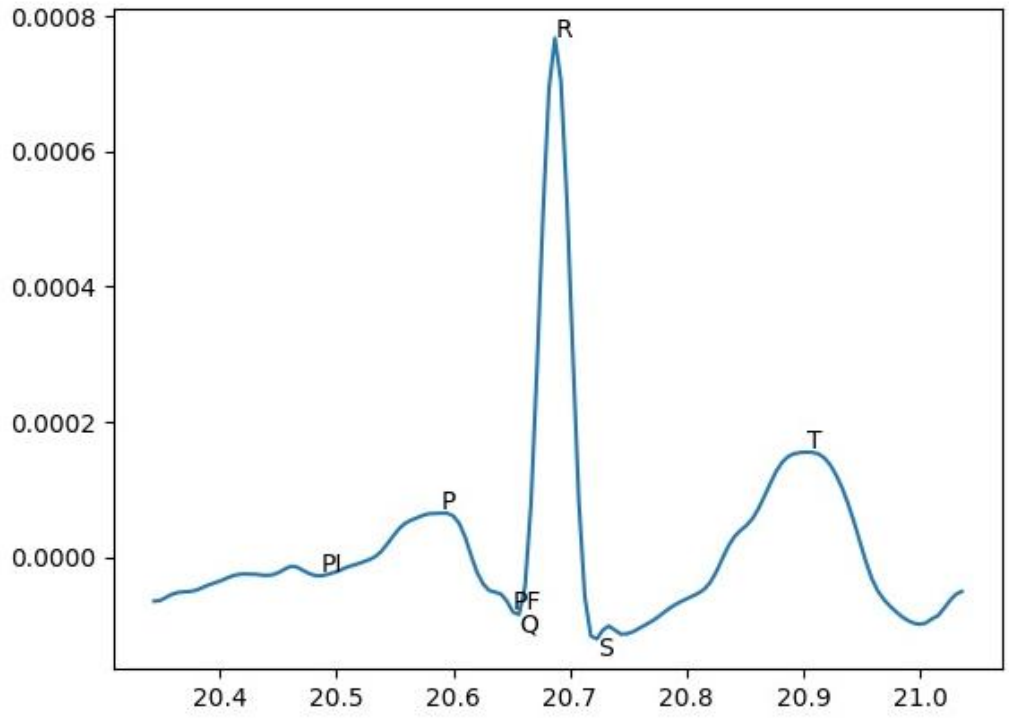
Inicialmente se determina el valor de R, sabiendo que R es un punto donde la derivada deja de crecer y empieza su decrecimiento. Por ende, para encontrar este punto se busca en el vector CaD (crece y empieza a decrecer), y dentro de este se determina el punto en el cual el vector tiene su punto máximo.

Para la determinación del punto Q se usan dos alternativas, mediante la primera se determina Q analizando los valores donde se presenta un mínimo en la señal antes del momento R, y en la segunda, se hace el análisis de los puntos en el vector DaC (decrece y empieza a crecer), mediante este análisis se busca el punto inmediatamente anterior al momento R donde la señal tenga el comportamiento mencionado.

Una vez se obtienen los valores de cada una de las dos opciones se comparan para determinar la efectividad en el proceso de detección, para complementar este análisis, se busca el valor que presente menos diferencia significativa con el momento de la onda R. En caso de no ser iguales los resultados, se deja el precedente de la divergencia en la determinación de los datos.

Este proceso se aplica de igual forma para determinar el momento de la onda P, S y T. Una vez terminado el proceso para cada uno de los parámetros se obtiene la señal completamente mapeada y caracterizada.

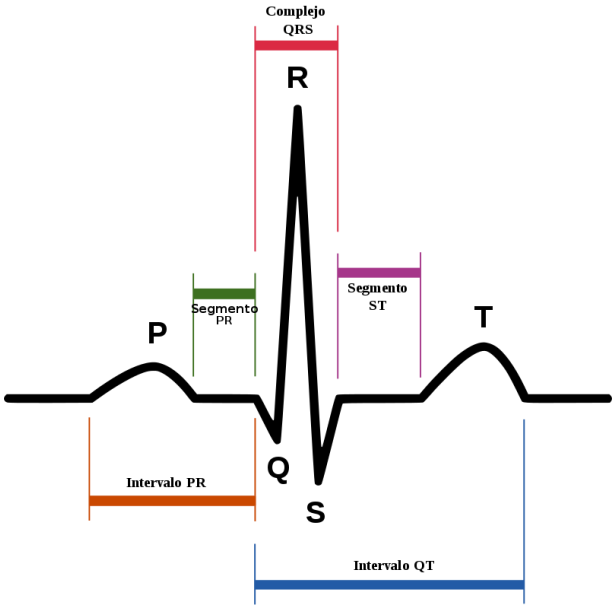
Figura 26. Señal caracterizada mediante el método 2



Fuente: Autores

Con la señal caracterizada *(Figura 26)* solo resta calcular la duración de los complejos y segmentos, de esta manera se completaría la gama de parámetros necesarios para la detección de las enfermedades cardiacas propuestas

Figura 27. Complejos cardiacos y sus puntos



Fuente: welchallyn[25]

El calculo de los complejo es llevado a cabo mediante las ecuaciones presentadas en la siguiente tabla.

Tabla 2. Cálculos de los Segmentos y Complejos

Cálculo de Intervalos y Complejos	
Complejo QRS	PF - S
Segmento PR	PF - Q
Segmento ST	S - TI
Intervalo PR	PI – Q
Intervalo QT	Q - TF

Fuente Autores

7. CONSTRUCCÍON DEL DIAGNÓSTICO

En este punto del desarrollo del código del programa se tienen ya los valores de onda necesarios para la elaboración del diagnóstico con los posibles padecimientos cardiacos abarcados en este trabajo, por lo que se procede a determinar la existencia o no de los mismos a través del cumplimiento de condiciones relacionadas a estas enfermedades.

7.1 HIPERPOTASEMIA

Esta afección se genera por el alto nivel de potasio en la sangre, lo que causa una disminución de la contractilidad cardiaca y favorece la aparición de arritmias ventriculares; este padecimiento suele ser causado por insuficiencia renal, el uso de determinados fármacos, entre otros factores[26].

Entre los valores anormales de onda que identifican a la hiperpotasemia se tiene una onda P plana, un ensanchamiento del complejo QRS y un estrechamiento de la onda T, de esta forma:

$$T\ (s) < 0.1\ s \qquad \text{Ecuación (4)}$$

$$P\ (mV) < 0.05\ mV \qquad \text{Ecuación (5)}$$

$$QRS\ (s) > 0.12\ s \qquad \text{Ecuación (6)}$$

7.2 HIPERTROFIA AURICULAR DERECHA

La hipertrofia auricular derecha suele presentarse a causa de enfermedades pulmonares, enfermedades valvulares derechas o cardiopatías congénitas.

Este padecimiento se encuentra fuertemente relacionado con el comportamiento de la onda P, por lo que se considera como un valor anómalo[27].

$$P\ (mV) < 0.25\ mV \qquad \text{Ecuación (7)}$$

[26] My-EKG. (s.f.). Hiperpotasemia en el EKG. My-EKG.com. https://www.my-ekg.com/metabolicas-drogas/hiperpotasemia-ekg.html

[27] My-EKG. (s.f.). Dilatación de la Aurícula Derecha. My-EKG.com. https://www.my-ekg.com/hipertrofia-dilatacion/dilatacion-auricula-derecha.html

7.3 DILATACION AURICULAR

Es un padecimiento en el cual se presenta un agrandamiento anormal de las aurículas del corazón, suelen ser un signo de una enfermedad subyacente o una condición médica que afecta el funcionamiento del corazón; entre las causas más comunes de esta afección se encuentran la insuficiencia cardiaca, enfermedad valvular cardiaca, hipertensión arterial, fibrilación auricular, cardiopatías congénitas, enfermedades pulmonares, entre otras.

Entre los valores en los cuales ya se considera que el paciente potencialmente puede presentar esta condición se encuentra[28]:

$$P\ (s) > 0.1\ s \qquad \text{Ecuación (8)}$$

7.4 BLOQUEO AUROVENTRICULAR

También llamado Bloqueo AV, es un trastorno en el sistema de conducción cardiaco que provoca que el estímulo eléctrico que se genera en las aurículas sea conducido, ya sea con retraso o no sea conducido, hacia los ventrículos[29]. Suele ser causado por problemas en otras estructuras cardiacas o por alteraciones metabólicas[30].

Esta enfermedad suele presentar ondas P, seguidas del complejo QRS, pero con un intervalo PR prolongado, por lo que se tiene como valor anormal:

$$PR\ (s) > 0.2\ s \qquad \text{Ecuación (9)}$$

[28] Cleveland Clinic. (s.f.). Left Atrial Enlargement. Cleveland Clinic. https://my.clevelandclinic.org/health/diseases/23967-left-atrial-enlargement#:~:text=Left%20atrial%20enlargement%20is%20when,pumps%20blood%20to%20your%20aorta

[29] My-EKG. (s.f.). Bloqueos AV (Bloqueos de la Conducción Aurículo-Ventricular). My-EKG.com. https://www.my-ekg.com/arritmias-cardiacas/bloqueos-av.html

[30] My-EKG. (s.f.). Bloqueos AV de Primer Grado. My-EKG.com. https://www.my-ekg.com/arritmias-cardiacas/bloqueos-av-primer-grado.html

7.5 BLOQUEO DE RAMA

Los bloqueos de rama son afectaciones en la conducción eléctrica que provocan cambios en la forma en que los ventrículos se despolarizan, esto se puede causar por un valor de tensión arterial alto o a otra enfermedad cardiaca[31].

Esta condición provoca alteraciones importantes en el complejo QRS del electrocardiograma, causando aumentos es su duración y cambios en su morfología; por lo que se tiene como un valor irregular de QRS:

$$QRS\ (s) > 0.12\ s \qquad \text{Ecuación (10)}$$

7.6 MIOCARDITIS

La miocarditis es una inflamación en el miocardio, el musculo del corazón; puede afectar la capacidad del corazón de bombear sangre de manera eficiente y, en casos graves, puede conllevar a complicaciones cardiacas importantes. Suele producirse por infecciones virales, bacterianas, fúngicas o parasitarias, reacciones alérgicas, toxinas o sustancias químicas, quimioterapia, entre otras[32].

Este padecimiento puede ser detectado en un electrocardiograma a través de los siguientes valores de onda anormales:

$$QT\ (s) > 0.42\ s \qquad \text{Ecuación (11)}$$

$$R\ (mV) > 2\ mV \qquad \text{Ecuación (12)}$$

7.7 HIPERCALCEMIA

Se define como un incremento en los niveles de calcio en el plasma, entre las causas principales de este trastorno se tiene el hiperparatiroidismo (primario, secundario y terciario), cáncer, enfermedades granulomatosas, síndrome de leche y alcalinos, entre otras[33].

---

[31] My-EKG. (s.f.). Bloqueos de Rama. My-EKG.com. URL: https://www.my-ekg.com/bloqueos-rama/bloqueos-rama.html

[32] MSD Manuals. (s.f.). Miocarditis. MSD Manuals Profesional. https://www.msdmanuals.com/es-co/professional/trastornos-cardiovasculares/miocarditis-y-pericarditis/miocarditis

[33] My-EKG. (s.f.). Hipercalcemia en el EKG. My-EKG.com. https://www.my-ekg.com/metabolicas-drogas/hipercalcemia-ekg.html

Suele provocar cambios significativos en el electrocardiograma, sobre todo en la duración del segmento ST y del intervalo QT, por lo que se tiene como valor de parámetro anormal:

$$QT\ (s) < 0.32\ s \qquad \text{Ecuación (13)}$$

7.8 SINDROME DEL QT PROLONGADO

El intervalo QT prolongado se suele asociar a un mayor riesgo de arritmias cardiacas dado que puede causar una fibrilación ventricular y una muerte súbita. Puede ser causada por diferentes factores hereditarios o por el síndrome QT largo adquirido por el uso de fármacos, alteraciones electrolíticas, hipertrofia ventricular izquierda, isquemia miocárdica, entre otras[34].

Esta enfermedad, asociada directamente a la duración del intervalo QT, presenta como un valor ya anormal:

$$QT\ (s) > 0.42\ s \qquad \text{Ecuación (14)}$$

7.9 PERICARDITIS

La pericarditis es la inflamación de la membrana que envuelve al corazón (el pericardio), este problema suele causar cambios en el electrocardiograma y puede ser causado por afecciones autoinmunes, infecciosas, metabólicas, traumáticas, inducidas por fármacos, entre otras[35].

Este padecimiento suele estar asociado al segmento ST de la señal cardiaca a través de una duración prolongada discorde y a la amplitud de la onda T. Dentro de los valores anormales que sirven para identificar esta enfermedad se encuentran:

$$ST\ (s) > 0.15\ s \qquad \text{Ecuación (15)}$$

$$T\ (mV) > \frac{R}{3}\ mV \qquad \text{Ecuación (16)}$$

---

[34] My-EKG. (s.f.). Cómo leer un EKG: Intervalo QT. My-EKG.com. https://www.my-ekg.com/como-leer-ekg/intervalo-qt.html

[35] My-EKG. (s.f.). Pericarditis Aguda en el EKG. My-EKG.com. https://www.my-ekg.com/enfermedades/pericarditis-aguda-ekg.html

7.10 HIPERTROFIA AURICULAR IZQUIERDA

La hipertrofia auricular izquierda es el agrandamiento o aumento anormal de la aurícula izquierda del corazón. Las causas más comunes de la hipertrofia auricular izquierda incluyen: hipertensión arterial, cardiopatías crónicas, trastornos del ritmo cardiaco, enfermedades pulmonares, factores genéticos, entre otras[36].

Esta afección suele estar asociada a la onda T, en la cual se presenta un ensanchamiento irregular, teniendo como valor anómalo:

$$T\ (s) > 0.12\ s \qquad \text{Ecuación (17)}$$

7.11 ISQUEMIA

Es una enfermedad ocasionada por la falta de flujo sanguíneo adecuado hacia el musculo cardiaco (miocardio). Esta condición ocurre cuando las arterias coronarias que suministran sangre al corazón se estrechan o se bloquean parcial o completamente debido a la acumulación de placa en sus paredes. Entre las causas más comunes se encuentran: edad, genero (los hombres poseen un mayor riesgo que las mujeres antes de que estas desarrollen la menopausia), herencia, antecedentes personales, dislipemia, tabaquismo, diabetes mellitus, hipertensión arterial, sedentarismo, entre otras[37].

Este padecimiento suele estar ligado al comportamiento de la onda T en el electrocardiograma, teniendo como valores anormales en este:

$$T\ (s) > 0.12\ s \qquad \text{Ecuación (18)}$$

$$T\ (mV) > \frac{R}{3}\ mV \qquad \text{Ecuación (19)}$$

7.12 TAQUICARDIA SINUSUAL

Es un tipo de taquicardia que se genera en el nodo sinusual que aparece en reposo o en situaciones de esfuerzo leve, es una arritmia poco frecuente que produce frecuencias cardiacas elevadas y suele observarse más en el sexo femenino y en

[36] My-EKG. (s.f.). Dilatación de la Aurícula Izquierda. My-EKG.com. https://www.my-ekg.com/hipertrofia-dilatacion/dilatacion-auricula-izquierda.html

[37] My-EKG. (s.f.). Cardiopatía Isquémica. My-EKG.com. https://www.my-ekg.com/infarto-ekg/cardiopatia-isquemica.php

rango de edad entre los 15 y los 45 años, entre las causas se encuentra la realización de ejercicio leve, ansiedad o estrés psicológico, fiebre, consumo de cafeína, alcohol u otros estimulantes, entre otros[38].

Este padecimiento se encuentra relacionado de manera directa con la frecuencia cardiaca que presente el paciente, teniendo como un valor ya anómalo:

$$Frecuencia\ Cardiaca\ (LPM) > 100\ LPM \qquad \text{Ecuación (20)}$$

7.13 BRADICARDIA SINUSUAL

Es una condición médica en la que el corazón late más lentamente de lo normal debido a la disminución en la frecuencia de los impulsos eléctricos que provienen del nodo sinusual. La bradicardia sinusual no significa patología cardiaca, es frecuente observarla en los deportistas de alto rendimiento, en los pacientes con tratamientos con fármacos que enlentecen la frecuencia cardiaca, hipotiroidismo, efectos del envejecimiento, entre otras[39].

Esta afección se encuentra relacionada directamente con el ritmo cardiaco que tenga el paciente, teniendo como un valor irregular:

$$Frecuencia\ Cardiaca\ (LPM) < 60\ LPM \qquad \text{Ecuación (21)}$$

[38] My-EKG. (s.f.). Taquicardia Sinusal Inapropiada. My-EKG.com. https://www.my-ekg.com/arritmias-cardiacas/taquicardia-sinusal-inapropiada.html
[39] My-EKG. (s.f.). Cómo leer un EKG: Ritmo Cardiaco. My-EKG.com. https://www.my-ekg.com/como-leer-ekg/ritmo-cardiaco.html#tab

8. INTERFAZ (GUÍA DE USO)

Una interfaz gráfica de usuario, también conocido como GUI es un grupo de componentes visuales e interactivos que sirven para facilitar la comunicación entre una persona y un sistema computacional. Cumple con la tarea de simplificar la interacción de los usuarios haciendo que un programa sea ejecutado de manera sencilla e intuitiva[40].

Figura 28. Ideología de un GUI

Fuente: Periódico X[41]

Un GUI se constituye de diversos objetos tales como ventanas, botones, iconos, menús, cuadros de texto, entre otros componentes que realizan y reciben información de manera eficiente. Son utilizadas en un sinfín de herramientas de software, tales como aplicaciones móviles y web y sistemas operativos.

Un GUI se constituye de diversos objetos tales como ventanas, botones, iconos, menús, cuadros de texto, entre otros componentes que realizan y reciben información de manera eficiente. Son utilizadas en un sinfín de herramientas de software, tales como aplicaciones móviles y web y sistemas operativos, en la Figura 28 se puede apreciar la interacción entre el aplicativo manejado por un sistema computacional y el usuario.

[40] INTERFAZ GRÁFICA de usuario gui [Anónimo]. Arimetrics [página web]. Disponible en Internet: <https://www.arimetrics.com/glosario-digital/interfaz-grafica-usuario-gui>.
[41] CHACÓN, Paola Villegas. Interfaz gráfica de usuario. Tecnología.cr [página web]. (12, mayo, 2016) Disponible en Internet: <https://periodicox.bigpress.net/texto-diario/mostrar/436057/interfaz-grafica-usuario>.

El desarrollo de la interfaz gráfica se realizó por medio de la librería Tkinter de Python, utilizando programación secuencial y definiendo cada vista de esta a través de funciones.

Para esto es necesario el uso de los componentes, también llamados widgets, que proporciona la biblioteca, como lo son Frame, PhotoImage, Button, Filedialog, FigureCanvasTkAgg, Treeview, Combobox y Checkbutton.

La interfaz se compone en principio de una ventana que contiene cuatro Frames principales los cuales contienen a su vez el resto de los componentes del programa.

8.1 VISTA 1: PRESENTACIÓN

El primer Frame abarca la información general del proyecto, como lo son el titulo y el nombre de los autores (para esto se hace uso del widget Label[42], también la imagen del logotipo de la universidad (utilizando el componente PhotoImage[43]), y por último un botón de continuar (por medio del elemento Button[44]) que sirve para navegar hacia la siguiente vista. En la Figura 29, se puede apreciar la vista de presentación anteriormente descrita.

Figura 29. Vista 1 / Presentación

Fuente: Autores

[42] GeeksforGeeks. (s.f.). Python Tkinter Label. GeeksforGeeks. https://www.geeksforgeeks.org/python-tkinter-label/

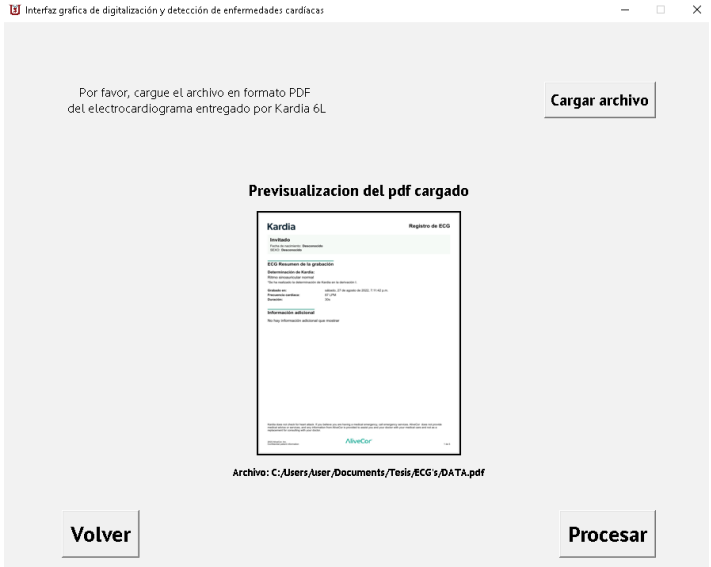
[43] CodersLegacy. (s.f.). Tkinter PhotoImage in Python. CodersLegacy. https://coderslegacy.com/python/tkinter-photoimage/

[44] Anzeljg. (s.f.). Button - Tkinter 8.6. [Documento web]. https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/button.html

8.2 VISTA 2: ENTRADA DE DATOS Y PROCESAMIENTO

En el segundo Frame se plasma el proceso pertinente para llevar a cabo el procesamiento del electrocardiograma, a través del botón “Cargar archivo” (con la función askopenfile[45]) se pide al usuario la búsqueda del archivo en formato pdf del ECG a través del explorador de archivos del equipo, una vez se seleccione el archivo correctamente, se presenta la vista previa del documento cargado en el recuadro que se encuentra debajo del botón previamente mencionado, posterior a la vista previa se encuentran dos botones, el botón “Volver” realiza la tarea de regresar a la primera vista del programa y el botón “Procesar” que conecta el Frame actual al siguiente, y a su vez, realiza la ejecución del proceso de digitalización y procesamiento del electrocardiograma cargado. Es importante mencionar que si la cantidad de páginas del pdf no supera o excede a las de un resultado estándar entregado por el dispositivo Kardia 6L se mostrará en pantalla una ventana emergente advirtiendo esta inconsistencia y se pedirá nuevamente el cargue de un archivo acorde a lo solicitado. La Figura 30, muestra la ventana de entrada de datos y procesamiento.

Figura 30. Vista 2 / Entrada de datos y procesamiento



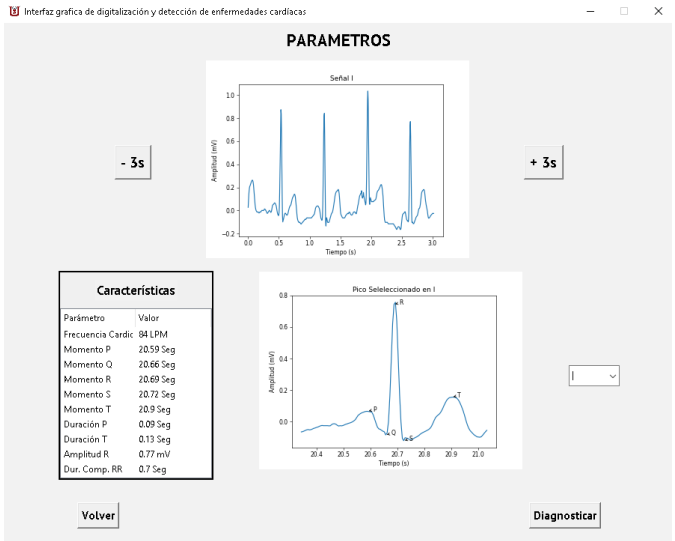
Fuente: Autores

[45] GeeksforGeeks. (s.f.). Python askopenfile() function in tkinter. GeeksforGeeks. https://www.geeksforgeeks.org/python-askopenfile-function-in-tkinter/

8.3 VISTA 3: RESULTADOS

Para la vista 3 se tiene el ECG ya digitalizado y adicionalmente los valores de ondas cardiacas que determinan la posibilidad de un padecimiento coronario. En la parte superior se encuentra la gráfica del electrocardiograma digitalizado muestreado cada tres segundos (A través del widget FigureCanvasTkAgg[46]) y dos botones, uno a la izquierda y otro a la derecha, estos se utilizan para desplazar en el tiempo la señal y apreciarla de una mejor manera. Inmediatamente debajo se encuentra la tabla de valores de onda con los que se realiza el análisis de las enfermedades cardiacas (usando el componente Treeview[47]), el pico al cual se le extrae esta información y una lista desplegable que permite navegar entre las derivaciones del examen cardiaco (utilizando la herramienta Combobox[48]). Finalmente se tienen dos botones que cumplen con la labor de darle navegación a la interfaz: a través del botón “Volver” se cambia de vista hacia el segundo Frame, y con el botón “Diagnosticar” se da paso hacia la última vista. Esta vista también despliega una ventana emergente en caso de que se detecte una divergencia en el cálculo de los parámetros de onda. Esta ventana se aprecia mejor en la Figura 31, que contiene los resultados del programa.

Figura 31. Vista 3 / Resultados



Fuente: Autores

[46] CodersLegacy. (s.f.). FigureCanvasTkAgg - Matplotlib & Tkinter. CodersLegacy. https://coderslegacy.com/figurecanvastkagg-matplotlib-tkinter/

[47] RecursosPython. (s.f.). Vista de Árbol (TreeView) en Tkinter. RecursosPython. https://recursospython.com/guias-y-manuales/vista-de-arbol-treeview-en-tkinter/

[48] RecursosPython. (s.f.). Lista Desplegable (Combobox) en Tkinter. RecursosPython. https://recursospython.com/guias-y-manuales/lista-desplegable-combobox-en-tkinter/

8.4 VISTA 4: DIAGNOSTICO

Por último, en la vista de diagnóstico se tiene la información relacionada con las enfermedades detectadas y la justificación en base a discordancias entre los valores de onda obtenidos con valores medicamente aceptados o normales. A la izquierda de este se encuentran las cardiopatías analizadas en este proyecto (donde se emplea el widget Checkbutton[49]), dependiendo de si difieren los resultados a valores normales de onda se da que dicha enfermedad pueda estar presente en el examen del usuario, a la derecha se encuentra el motivo por el cual la persona puede tener el padecimiento en cuestión y en la parte inferior tres botones que cumplen con la función de darle navegación a la interfaz: a través del botón “Volver” se regresa a la vista inmediatamente anterior, con “Inicio” se vuelve a la primera ventana del programa y si se selecciona “Salir” se finaliza la ejecución de la interfaz. La Figura 32 muestra la ventana de diagnóstico explicada anteriormente.

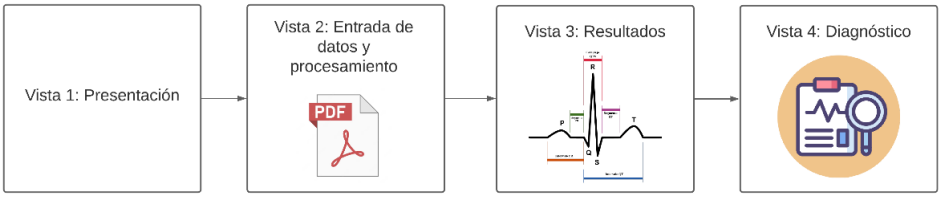
Figura 32. Vista 4 / Diagnostico

Fuente: Autores

A grandes rasgos, el flujo de la interfaz sigue la secuencia de la Figura 33, donde cada cuadro representa las vistas anteriormente descritas:

[49] RecursosPython. (s.f.). Checkbox (Checkbutton) en Tcl/Tk (Tkinter). RecursosPython. https://recursospython.com/guias-y-manuales/checkbox-checkbutton-en-tcltk-tkinter/

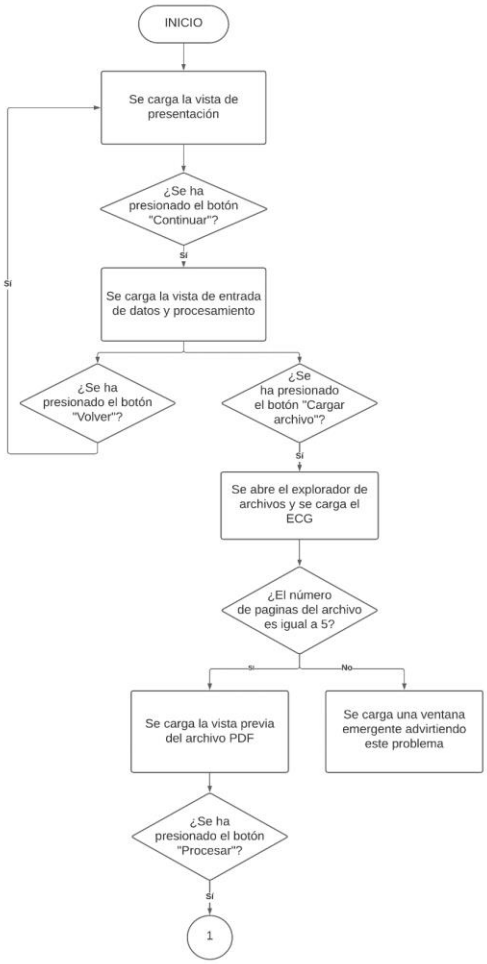
Figura 33. Diagrama de vistas



Fuente: Autores

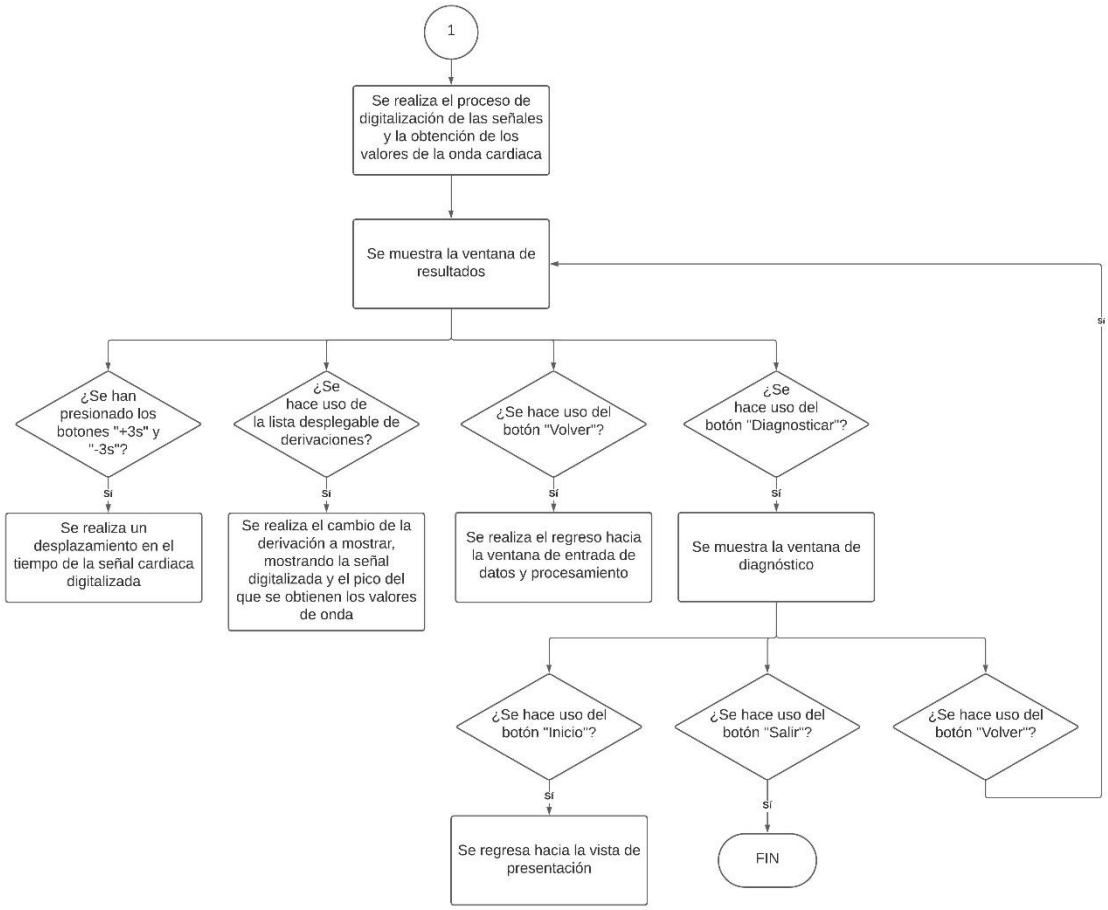
De manera más detallada, la Figura 34 y la Figura 35 muestra el proceso completo que se lleva a cabo en la interfaz, mostrando las decisiones que se llevan a cabo para navegar entre las diferentes vistas de esta, la entrada de datos y la visualización de los resultados y el diagnóstico:

Figura 34. Diagrama de flujo de la interfaz, parte 1



Fuente: Autores

Figura 35. Diagrama de flujo de la interfaz, parte 2



Fuente: Autores

9. RESULTADOS

Partiendo de un banco de datos proveniente de un grupo de sujetos de prueba determinados, se realiza la caracterización de los electrocardiogramas mediante el uso de la aplicación.

El lote de prueba está construido con 8 electrocardiogramas provenientes de 4 sujetos de prueba, de los cuales, uno confirma no poseer falencias cardiacas.

Al usar la aplicación para caracterizar se obtiene que, de los 8 electrocardiogramas tomados, solo 2 fueron admitidos por el software *(Tabla 3)*, los demás fueron descartados por diversas causas, siendo la más frecuente la captación de ruido al momento de tomar la lectura.

Tabla 3. Validación de los ECG's por parte del software

Nombre	Validez	Causal
ecg-20230322-145823	No Procesable	La señal fue cortada durante el proceso de individualización de las derivaciones, debido a valores de voltaje anormales en las derivaciones unipolares de las extremidades (aVR, aVL, aVF), incorporadas por posibles fallas en el momento de la captura del ECG.
ecg-20230524-141500	No Procesable	La señal de la derivación I es demasiado baja (<0.5 mV), por lo cual es imposible realizar la elección de un pico de estudio.
ecg-20230524-142047	No Procesable	En el ECG leído, se encuentra gran cantidad de ruido, generando picos demasiado altos, al realizar el escaneo de la señal se excede la limitante de 150 picos, por lo cual la señal se descarta.
ecg-20230322-145416	No Procesable	El ECG presenta una cantidad de ruido inusual para un resultado aceptable, lo que genera una gran cantidad de picos que desbordan el umbral de los 150.
ecg-20230524-141742	Procesable	Divergencia determinando S y T.
ecg-20230322-141514	No Procesable	El pico determinado como viable, se encuentra en un punto final del ECG, en el cual no se puede tomar la captura de tiempos futuros más allá de los 30 segundos.

ecg-20220827-191142	Procesable	Divergencia estimando T.
ecg-20230322-145221	No Procesable	Error en la detección de R.

Fuente: Autores

Con base a los resultados obtenidos de la aplicación, se genera cierta incertidumbre respecto al nivel de fiabilidad de la aplicación, por lo cual se genera la necesidad de buscar la verificación y caracterización de los electrocardiogramas bajo criterio médico. Para este proceso se le suministran los mismos lotes de datos que se le entregan al software, para realizar el mismo proceso de caracterización y determinación de los datos validables.

De este modo, luego de validar la información medicamente y bajo el criterio humano se encontró que los descartes realizados por el software no son concluyentes desde un punto de vista médico como se muestra en la tabla 4 a continuación.

Tabla 4. Validación de ECG's medicamente

Nombre	**Validez**	**Causal**
ecg-20230322-145823	No Procesable	Es buen electro, lo único es que en una de las hojas hay un bajón enorme, horrible y eso no es patológico es como si se hubiera movido el paciente.
ecg-20230524-141500	No Procesable	La entiendo, pero no la usaría porque AVL no es legible, el trazado no es confiable, no hay ondas y pues uno pensaría entonces una patología, pero es que el trazo entonces no concuerda con los demás trazos y pues uno diría listo, está afectada entonces sólo la cara lateral alta, pero es que esa también está definida por DI y DI si está bien, entonces pues si tengo que mirar todos los ejes, no me confiaría por AVL

ecg-20230524-142047	No Procesable	Yo no lo veo como ritmo normal, los complejos QRS no son regulares, no en todas hay una onda que me indicaría bloqueo de rama, pero no es posible porque no se ve exactamente así, yo pediría que lo volvieran a tomar porque para mí está mal tomado, hubo movimiento también, de pronto (eso si no estoy segura) mala ubicación de electrodos
ecg-20230322-145416	Procesable	Sin quejas ni observaciones
ecg-20230524-141742	Procesable	Tiene muy buena claridad, se puede leer cómodamente
ecg-20230322-141514	Procesable	Sin quejas ni observaciones
ecg-20220827-191142	Procesable	Lo único es que en la página 4 parece como que se movió, esta página me generaría confusiones.
ecg-20230322-145221	No Procesable	Hay unas páginas donde se entiende, otras donde no tanto, hay como que diferente trazo en unas, o sea si lo entiendo, pero no podría definirlo completamente.

Fuente: Autores

Una vez se obtuvo una muestra médica, se procedió a realizar el análisis de esta mediante el software, en el cual, al momento de realizar el diagnóstico aparecen ciertas alarmas en cuanto a la detección de algunos puntos característicos, pero llevando a cabo el análisis general de la señal.

Con la señal caracterizada mediante el software, se solicita al médico evaluador que realice la caracterización de la señal cardiaca, sin que este conozca previamente los resultados suministrados por el software para no alterar el criterio médico. Partiendo de la primera caracterización médica, la cual se puede realizar en cualquier punto de las 6 derivaciones cardiacas presentadas en el ECG se realiza una comparación de los parámetros más relevantes como lo son:

- La frecuencia cardiaca
- La duración del intervalo RR
- La duración de la onda P
- Intervalo PR
- Segmento PR
- Intervalo QT
- Segmento ST
- Amplitud de la onda P
- Amplitud del complejo QRS
- Amplitud de T

Con estos parámetros se solicita al médico evaluador que realiza una segunda caracterización empleando la misma gamma de parámetros en un segmento específico de la derivación I, el segmento especifico consta de una duración de 3 segundos dentro de la cual se encuentra contenido el pico que uso el software para realizar la determinación de los parámetros.

Una vez obtenidos los dos análisis médicos, se realiza la muestra del software y la parametrización que entrega el programa del mismo ECG caracterizado por el médico, permitiéndole al médico interactuar con el software y probar el desempeño de este.

Para finalizar el proceso de validación médica, se realiza una comparación de los parámetros determinados por el software y las dos parametrizaciones entregadas por el médico *(**¡Error! No se encuentra el origen de la referencia.**)*.

El primer cruce de datos se realiza entre el primer análisis del médico y los resultados del aplicativo, con este cruce se busca probar la validez de la caracterización realizada a través de la derivación I. Con el segundo cruce de datos se busca obtener el porcentaje de fidelidad y precisión que puede llegar a presentar el software al momento de parametrizar el ECG.

En la siguiente tabla se presentan los datos recabados en el proceso, en la tabla se consignan los datos respectivos de la caracterización de los electrocardiogramas asumidos como validos por el software.

Tabla 5. Resultados de las caracterizaciones

Parámetro	ecg-20230524-141742					ecg-20220827-191142				
	Valor	Val. 1	Val. 2	Diff.	Error	Valor	Val. 1	Val. 2	Diff.	Error
Frecuencia Cardiaca (lpm)	97	88	92	4	4.5	87	90	84	6	6.66
Duración P (S)	0.04	0.08	0.1	0.02	25	0.04	0.08	0.09	0.01	12.5
Duración T (S)	N.C	N.C	0.02	N.C	N.C	N.C	N.C	0.13	N.C	N.C
Amplitud R (mV)	N.C	N.C	0.6	N.C	N.C	N.C	N.C	0.77	N.C	N.C
Duración complejo RR (S)	0.68	0.68	0.65	0.03	4.4	0.68	0.66	0.7	0.04	6.06
Duración complejo PR (S)	0.12	0.12	0.14	0.02	16.67	0.12	0.12	0.13	0.01	8.33
Duración complejo QRS (S)	N.C	0.04	0.06	0.02	50	N.C	0.04	0.07	0.03	75
Duración complejo QT (S)	0.32	0.2	0.16	0.04	20	0.3	0.32	0.35	0.03	9.37
Duración Complejo ST (S)	0.12	0.12	0.05	0.07	58.33	0.16	0.1	0.1	0.00	0

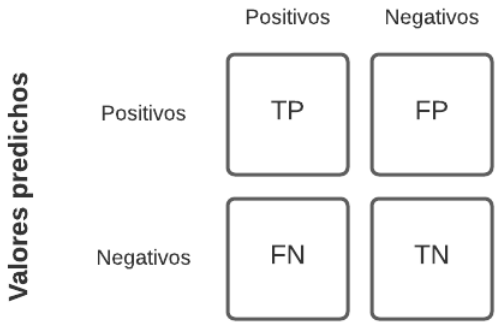
Fuente: Medico Validador, Autores.

Nomenclatura de la tabla	
Valor	Características del ECG considerando todas sus derivaciones (Criterio Médico)
Val. 1	Características del ECG considerando solo la derivación I (Criterio Médico)
Val. 2	Características del ECG considerando solo la derivación I (Criterio Software)
Diff.	Diferencia entre Val.1 y Val. 2
Error	Error relativo del Val. 2 con respecto a Val. 1

10. ANÁLISIS DE RESULTADOS

La matriz de confusión corresponde a una tabla que resume el número de predicciones correctas e incorrectas de un conjunto de datos[50]. La matriz de confusión se representa de la siguiente manera:

Figura 36: Representación de la matriz de confusión



Fuente: DataSource[50]

Las predicciones se establecen como se especifica a continuación:

- Verdaderos positivos (TP): Es la cantidad de predicciones verdaderas y correctas.
- Falsos negativos (FN): Es la cantidad de predicciones falsas correctas.
- Falsos positivos (FP): Es la cantidad de predicciones verdaderas incorrectas.
- Verdaderos negativos (TN): Es la cantidad de predicciones falsas incorrectas.

Partiendo de la definición de la matriz, se puede trazar la matriz *(Tabla 6)* para medir la efectividad del software para determinar cuándo un ECG presenta validez médica o es relevante para un diagnóstico.

[50] DataSource.ai. (s.f.). Comprensión de la Matriz de Confusión y Cómo Implementarla en Python. DataSource.ai. https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

Tabla 6. Matriz de confusión del proyecto

Matriz de Confusión		Software	
		Inválido	Válido
Médico	Inválido	0	4
	Válido	2	2

Fuente: Autores

A. EXACTITUD: La exactitud representa el número total de predicciones correctas respecto al número total de muestras clasificadas. Se calcula de la siguiente manera:

$$exactitud = \frac{\#\ predicciones\ correctas}{\#\ total\ de\ predicciones} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Ecuación (22)}$$

Esta métrica se usa solo en escenarios donde se presenta una cantidad de datos balanceados en el entrenamiento para cada clase.

B. PRECISIÓN: La precisión representa el número total de clasificaciones correctas en cada clase. Se calcula de la siguiente manera:

$$precision = \frac{TP}{TP + FP} \qquad \text{Ecuación (23)}$$

C. SENSIBILIDAD: O también llamado recall representa el número de clasificaciones que corresponde a una clase respecto a las clases evaluadas. Se calcula de la siguiente manera:

$$sensibilidad = \frac{TP}{TP + FN} \qquad \text{Ecuación (24)}$$

Con base en la distribución generada por la matriz de confusión se pueden usar los datos para evaluar el rendimiento *(Tabla 7)* del software en cuanto al proceso de discernir a partir de que documentos se pueden procesas y extraer los datos.

Tabla 7. Medidas de la matriz de confusión

Parámetro	Valor Porcentual
Sensibilidad	50%
Especifidad	0%
Precisión	66.67%
Exactitud	25%

Fuente: Autores

Basados en los resultados de precisión del software, se puede dar por sentado que el criterio del software para decidir si un ECG es válido para su lectura es parcialmente valido, desde un punto de vista genérico, el cual esta presto a realizar un proceso de validación mediante métodos estadísticos los cuales apunten realmente al nivel de relevancia de los datos presentados debido a la precisión tan baja que presenta el software, esto obliga a que la primer parte del desarrollo sea sometida a procesos estadísticos para determinar a ciencia cierta la efectividad del clasificador, ya que con un valor tan cercano al 50% de precisión, los resultados podrían llegar a ser asumidos como resultados de un proceso de adivinación y suerte. El software logra determinar características relevantes para el descarte de un ECG basado en los puntos comunes considerados por un médico. Sin embargo, no llega a ser del todo preciso, como de evidencia en la Tabla 3 y la Tabla 4, donde medicamente los criterios de descarte son similares, pero en el software son un tanto más rigurosos debido a la carencia de comprensión del panorama completo del ECG como si lo puede desarrollar una persona.

Con esto en mente, se sugiere en trabajos futuros realizar un robustecimiento del banco de datos aplicando teoremas estadísticos que apunten a resultados verdaderamente fiables y con un sustento estadístico adecuado y como refuerzo de esta hipótesis son los demás resultados presentados en la matriz de confusión, los cuales solo apuntan a una gran incertidumbre siendo esto reforzado por la limitada cantidad de datos los cuales dan cabida a que pequeños errores sean abruptos considerando la desviación estándar que puede presentar un grupo tan pequeño como el presentado.

Estos datos se ven reflejados en la información reflejada por la matriz de confusión en la Tabla 7 donde la exactitud no supera un 25% y donde la precisión no obtiene los niveles adecuados para un sistema aceptable donde la precisión supera el 90%.

Para esta segunda etapa, como se propuso anteriormente, el objetivo es evaluar la calidad de los datos que emite el software respecto a los ECG’s que este acepta, para esto manejamos el porcentaje de error como parámetro de medida.

El porcentaje de error es una operación que permite determinar de forma clara el margen de error entre las diferencias existentes entre el valor estimado y el valor real, por lo que, si se quiere obtener dicha respuesta, se hace indispensable tener ambos valores para realizar la operación matemática[51].

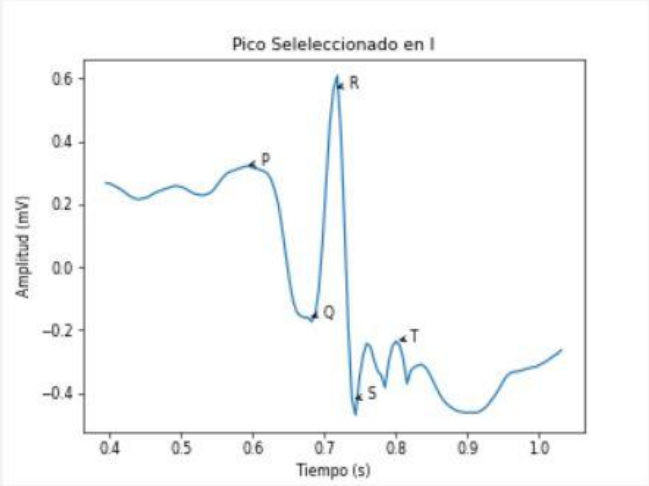
Basados en la definición y en la fórmula que especifica la manera de calcular el porcentaje de error, este se calcula y se consigna en la columna “error” de la Tabla 5, esto con el fin de determinar la efectividad y calidad del proyecto desarrollado.

Dejando de lado la matriz de confusión y el análisis de la validez de cada ECG, el paso siguiente es evaluar los resultados obtenidos de los ECG considerados válidos por el software. Mediante el primer proceso de validación se obtiene resultados dentro de parámetros duramente aceptables, pero al dar un vistazo al segundo proceso de validación, se observa que la parametrización que realiza el software es bastante cercana a los datos medicamente considerados.

La razón de esta variabilidad se puede obtener mediante una inspección de los picos determinados por el software.

Figura 37. Pico seleccionado del documento #1

Características	
Parámetro	Valor
Frecuencia Cardiaca	92 LPM
Momento P	0.59 Seg
Momento Q	0.68 Seg
Momento R	0.71 Seg
Momento S	0.74 Seg
Momento T	0.8 Seg
Duración P	0.1 Seg
Duración T	0.02 Seg
Amplitud R	0.6 mV
Dur. Comp. RR	0.65 Seg



Fuente: Autores

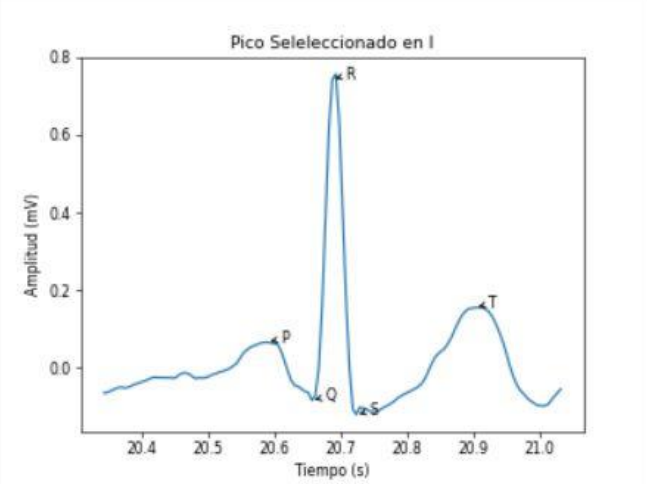
Si se realiza una inspección del pico posterior a la onda S, se encuentra un doble pico el cual puede generar confusión en el software, lo que inserta una gran cantidad de error en las detecciones del documento 1, de no ser así, la señal posterior al pico

[51] Web y Empresas. (s.f.). ¿Qué es el Porcentaje de Error? Web y Empresas. https://www.webyempresas.com/porcentaje-de-error/#%C2%BFQue_es_el_porcentaje_de_error

S es más limpia, por lo cual el pico puede ser más limpio y los resultados mucho más claros.

Figura 38. Pico seleccionado del documento #2

Características	
Parámetro	Valor
Frecuencia Cardiaca	84 LPM
Momento P	20.59 Seg
Momento Q	20.66 Seg
Momento R	20.69 Seg
Momento S	20.72 Seg
Momento T	20.9 Seg
Duración P	0.09 Seg
Duración T	0.13 Seg
Amplitud R	0.77 mV
Dur. Comp. RR	0.7 Seg



Fuente: Autores

El pico evaluado para el documento 2 de la Figura 38 muestra una señal más acorde y que puede ser evaluada y analizada de manera normal, no presenta picos adicionales y se logra distinguir fácilmente las ondas de la señal cardiaca; esto también se ve evidenciado en el porcentaje de error reducido que se obtiene de contrastar las validaciones medicas con los valores calculados por el software.

En contraste, los ECG's no aceptados por el software, son electrocardiogramas que presentan demasiado ruido o fallas en la toma de este como se menciona en la Tabla 3, donde se encuentran las fallas que presenta cada ECG para no ser validado por el software. En la

Figura *39* y la Figura 40 se pueden apreciar algunas secciones de algunos de los electrocardiogramas presentados al software, donde se evidencia la dispersión de una señal normal al presentado en la prueba. Como ECG's normales podemos ver los presentados en la Figura *37* y la Figura *38*, en los cuales se pueden apreciar claramente las componentes de un ciclo cardiaco en contraste con las señales en donde se aprecia total distorsión de la señal.

Figura 39. Segmento intermedio del ecg-20230322-145221.pdf

Fuente: Autores

Figura 40. Segmento intermedio del ecg-20230322-145221.pdf

Fuente: Autores

Condensando la información recabada a lo largo de los resultados se encuentra que el software aún carece de fiabilidad en cuanto a la naturaleza de los datos y resultados que puede llegar a presentar, para la primera etapa se tiene gran

cantidad de entropía, la cual está siendo aumentada por la carencia de bancos de datos con una varianza y dispersión adecuada para el proceso determinado, para la segunda parte, partiendo de las fallas del primera sección se tienen pocos datos para validar, que aunque los resultados presentados para los dos casos particulares es muy cercano al resultado médico, no llegan a ser del todo fiables, partiendo de la idea que con solo dos datos no se puede definir ni validar un sistema como el presentado, donde la cantidad de muestras disponibles en un entorno idónea podría llegar a ser infinitas. La naturaleza de los hallazgos llega a ser desconcertante en la medida de se llegaron a esperar mejores resultados, pero se espera que en futuros avances e investigaciones se llegue a mejorar la data y se puedan trazar resultados estadísticamente reales.

11. DISCUSIONES Y TRABAJOS FUTUROS

Es de vital importancia que el avance y las mejoras tecnológicas en este campo de investigación se sigan ampliando, por lo que se considera que este trabajo puede mejorarse y optimizarse de manera considerable. Se tienen en cuenta los siguientes aspectos que pueden incluirse o llevarse a cabo a futuro:

- Utilización de electrocardiogramas de 12 derivaciones, ya que las derivaciones obtenidas en las inmediaciones del tórax suministran información de suma importancia de la condición cardiaca del paciente.
- Realizar el análisis de la señal en un pico que pertenezca a la derivación II, ya que los profesionales de la salud suelen basar su criterio y realizar los cálculos de los valores de onda en esta derivación.
- Mejorar la calidad y la eficacia de la interfaz gráfica, ya que genera gastos importantes de recursos en el procesador y la memoria RAM del equipo. Por lo que se debe tener en cuenta la reducción de los recursos computacionales.
- Implementación de inteligencia artificial para el análisis y el diagnostico de los ECG.

12. CONCLUSIONES

A través del trabajo de grado propuesto se detalla el proceso para el desarrollo y elaboración de una interfaz gráfica para el reconocimiento y detección de características en electrocardiogramas usando Python. Este proyecto evidencia el uso de diferentes herramientas tecnológicas para lograr un correcto funcionamiento y la respectiva documentación, investigación y estudio para lograr un manejo optimo. Así pues, se tiene a continuación las siguientes conclusiones del presente trabajo.

- A través de las matrices de confusión se obtuvo que el algoritmo tiene una precisión de 67%. Esto se debe al sesgo existente entre la cantidad de electrocardiogramas descartados y los que se pueden analizar correctamente.
- Para realizar cualquier análisis utilizando una matriz de confusión es necesario tener un banco de datos mucho más extenso y también de mayor dispersión, de esta manera se obtienen resultados mucho más sólidos y no susceptibles a binarizaciones.
- Por medio del porcentaje de error calculado para cada parámetro de la señal cardiaca se puede apreciar una alta congruencia en la mayoría de los resultados, y una elevada discordancia en algunos otros, esto se debe a que el paciente o la persona a la que se le realiza el examen debe encontrarse en un estado de calma y sin estar en posesión de dispositivos o aparatos eléctricos y electrónicos que alteren o añadan ruido a la señal pues esta se encuentra en el orden de los milivoltios.
- Se creó un algoritmo de detección y análisis de los valores de onda presentes en electrocardiogramas digitales a través de bibliotecas de código abierto como Numpy y OpenCV de Python que favorecieron la manipulación de las imágenes y los vectores de datos generados gracias a la digitalización. Estas librerías proporcionan una amplia variedad de funciones y alternativas para manejar este tipo de información.
- Se logró implementar una interfaz gráfica dinámica y de fácil uso gracias a la librería Tkinter de Python, que permite cargar el examen cardiaco, posteriormente entregar datos detallados de los valores de onda y las señales digitalizadas de las seis derivaciones y finalmente un diagnóstico estimado a partir de información contrastada previamente, lo que genera en el usuario cierta comodidad a la hora de utilizarla.
- La interfaz presenta algunos inconvenientes de consumo de recursos computacionales, debido a la cantidad de información que debe ser extraída a partir del pdf cargado y también a la manipulación de los gráficos a través de los

botones. Teniendo esto en cuenta, se recomienda el uso del programa en máquinas con una memoria RAM superior a 4 GB.

- La aplicación exitosa de análisis de señales en la detección de enfermedades cardiacas representa un potencial transformador en la tecnología de la atención médica. Al agilizar este proceso se traduce en una disminución de carga laboral para los profesionales de la salud y en la mejora de la calidad de vida de los pacientes. Este trabajo contribuye y sienta las bases para sistemas de apoyo a decisiones clínicas más accesibles y efectivas.
- A partir de recomendaciones médicas, se recomienda para trabajos futuros, la utilización de electrocardiógrafos capaces de entregar las doce derivaciones las cuales suministran una mayor cantidad de información importante sobre el estado del corazón del paciente y, así mismo, la ampliación en la cobertura de la gama de enfermedades.
- Este trabajo significa una aportación importante en lo que se refiere al campo médico y el desarrollo tecnológico. Es una base sólida para futuras investigaciones relacionadas a la visión por computador, aplicados a Colombia y el departamento del Huila que está haciendo uso de herramientas y tecnología recientes.
- El proyecto cuenta con una buena orientación, pero carece de validez por la carencia de bancos de datos con una buen densidad o desviación estándar, lo cual hace que los resultados no sean concluyentes desde un punto de vista estadístico. Al momento de realizar una construcción de bancos de datos es necesario contar con una aprobación estadística de los datos para que se consideren válidos para la determinación de resultados

BIBLIOGRAFÍA

A. BENHAMIDA and M. KOZLOVSZKY, "Human ECG data collection, digitalization, streaming and storing," 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2020, pp. 105-110, doi: 10.1109/SAMI48414.2020.9108765. [Conferencia]

Agencia Espacial Europea (ESA). Eduspace: Unidad de Recursos de Educación Espacial. ESA. [Página web]. [consultado: 01 Agosto 2023]. Disponible en: https://www.esa.int/SPECIALS/Eduspace_ES/SEMCDX3FEXF_2.html

AliveCor. KardiaMobile 6L. AliveCor España. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://www.alivecor.es/kardiamobile6l

Anzeljg. Button - Tkinter 8.6. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/button.html

Arteris Médica. Electrocardiograma. [Imagen]. En: ¿Qué es un electrocardiograma? Arteris Médica. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.arterismedica.com/que-es-un-electrocardiograma/

Anonim. What Is Tk? McGill University. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/#WhatIsTk

CE-TekMed. AliveCor KardiaMobile 6-Lead ECG for iPhone and Android. CE-TekMed. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.ce-tekmed.ie/product-page/alivecor-kardia-mobile-6-lead-ecg-for-iphone-and-android

CHACÓN, Paola Villegas. Interfaz gráfica de usuario. Tecnología.cr [página web]. (12, mayo, 2016) [consultado: 01 Agosto 2023]. Disponible en: https://periodicox.bigpress.net/texto-diario/mostrar/436057/interfaz-grafica-usuario

Cleveland Clinic. Left Atrial Enlargement. Cleveland Clinic. [Página web]. [consultado: 03 Agosto 2023]. Disponible en:

https://my.clevelandclinic.org/health/diseases/23967-left-atrial-enlargement#:~:text=Left%20atrial%20enlargement%20is%20when,pumps%20blood%20to%20your%20aorta

CodersLegacy. FigureCanvasTkAgg - Matplotlib & Tkinter. CodersLegacy. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://coderslegacy.com/figurecanvastkagg-matplotlib-tkinter/

CodersLegacy. Tkinter PhotoImage in Python. CodersLegacy. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://coderslegacy.com/python/tkinter-photoimage/

Cupi2-IP. Sección 3.4: Slicing. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://cupi2-ip.github.io/IPBook/nivel3/seccion3-4.html

DataSource.ai. Comprensión de la Matriz de Confusión y Cómo Implementarla en Python. DataSource.ai. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

DeepAI. Computer Vision. DeepAI Machine Learning Glossary and Terms. [Página web]. [consultado: 02 Agosto 2023]. Disponible en: https://deepai.org/machine-learning-glossary-and-terms/computer-vision

GeeksforGeeks. Python askopenfile() function in tkinter. GeeksforGeeks. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.geeksforgeeks.org/python-askopenfile-function-in-tkinter/

GeeksforGeeks. Python Tkinter Label. GeeksforGeeks. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.geeksforgeeks.org/python-tkinter-label/

IBM. Computer Vision. IBM. [Página web]. [consultado: 03 Septiembre 2023]. Disponible en: https://www.ibm.com/topics/computer-vision

INTERFAZ GRÁFICA de usuario gui [Anónimo]. Arimetrics [página web]. [consultado: 04 Agosto 2023]. Disponible en: https://www.arimetrics.com/glosario-digital/interfaz-grafica-usuario-gui

Kardia. KardiaMobile 6L. Kardia Store. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://store.kardia.com/products/kardiamobile6l

Manual Merck. Electrocardiografía. Manual Merck. [Página web]. [consultado: 01 Agosto 2023]. Disponible en: https://www.msdmanuals.com/es/hogar/trastornos-del-coraz%C3%B3n-y-los-vasos-sangu%C3%ADneos/diagn%C3%B3stico-de-las-enfermedades-cardiovasculares/electrocardiograf%C3%ADa

MedlinePlus. Electrocardiograma (ECG). MedlinePlus. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://medlineplus.gov/spanish/pruebas-de-laboratorio/electrocardiograma/#:~:text=Un%20electrocardiograma%20(ECG)%20es%20un,y%20con%20una%20fuerza%20normal

MSD Manuals. Miocarditis. MSD Manuals Profesional. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.msdmanuals.com/es-co/professional/trastornos-cardiovasculares/miocarditis-y-pericarditis/miocarditis

Muñoz V., A. Electrocardiografía Básica. Repositorio Universidad del Rosario. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://repository.urosario.edu.co/server/api/core/bitstreams/b891abb3-2690-4551-a0c1-b16f41a17427/content

My-EKG. Bloqueos AV (Bloqueos de la Conducción Aurículo-Ventricular). My-EKG.com. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://www.my-ekg.com/arritmias-cardiacas/bloqueos-av.html

My-EKG. Bloqueos AV de Primer Grado. My-EKG.com. [Página web]. [consultado: 01 Agosto 2023]. Disponible en: https://www.my-ekg.com/arritmias-cardiacas/bloqueos-av-primer-grado.html

My-EKG. Bloqueos de Rama. My-EKG.com. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.my-ekg.com/bloqueos-rama/bloqueos-rama.html

My-EKG. Cardiopatía Isquémica. My-EKG.com. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.my-ekg.com/infarto-ekg/cardiopatia-isquemica.php

My-EKG. Cómo leer un EKG: Intervalo QT. My-EKG.com. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.my-ekg.com/como-leer-ekg/intervalo-qt.html

My-EKG. Cómo leer un EKG: Ritmo Cardiaco. My-EKG.com. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.my-ekg.com/como-leer-ekg/ritmo-cardiaco.html#tab

My-EKG. Dilatación de la Aurícula Derecha. My-EKG.com. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.my-ekg.com/hipertrofia-dilatacion/dilatacion-auricula-derecha.html

My-EKG. Dilatación de la Aurícula Izquierda. My-EKG.com. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://www.my-ekg.com/hipertrofia-dilatacion/dilatacion-auricula-izquierda.html

My-EKG. Hipercalcemia en el EKG. My-EKG.com. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.my-ekg.com/metabolicas-drogas/hipercalcemia-ekg.html

My-EKG. Hiperpotasemia en el EKG. My-EKG.com. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.my-ekg.com/metabolicas-drogas/hiperpotasemia-ekg.html

My-EKG. Papel EKG - My-EKG.com. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.my-ekg.com/generalidades-ekg/papel-ekg.html

My-EKG. Pericarditis Aguda en el EKG. My-EKG.com. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.my-ekg.com/enfermedades/pericarditis-aguda-ekg.html

My-EKG. Taquicardia Sinusal Inapropiada. My-EKG.com. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.my-ekg.com/arritmias-cardiacas/taquicardia-sinusal-inapropiada.html

ONOS. ¿Qué es una GUI? IONOS Digital Guide. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-

gui/#:~:text=Una%20graphical%20user%20interface%20o,el%20manejo%20del%20usuario%20humano

OpenCV. Acerca de OpenCV. OpenCV. [Página web]. [consultado: 01 Agosto 2023]. Disponible en: https://opencv.org/about/

OpenCV. Acerca de OpenCV. OpenCV. [Página web]. [consultado: 01 Agosto 2023]. Disponible en: https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,perception%20in%20the%20commercial%20products

OpenCV. Documentación de OpenCV: group__core__array.OpenCV. [Página web]. [consultado: 03 Septiembre 2023]. Disponible en: https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga4676b1376cdc4e528dab6bd9edc51c1a

OPS/OMS. (2020, Diciembre 9). OMS revela las principales causas de muerte y discapacidad en el mundo para 2000-2019. Organización Panamericana de la Salud. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://www.paho.org/es/noticias/9-12-2020-oms-revela-principales-causas-muerte-discapacidad-mundo-2000-2019

Python Software Foundation. About Python. Python Software Foundation. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.python.org/about/

Python Software Foundation. Cómo hacerlo: Ordenar. Documentación de Python 3. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://docs.python.org/es/3/howto/sorting.html

RecursosPython. Checkbox (Checkbutton) en Tcl/Tk (Tkinter). RecursosPython. [Página web]. [consultado: 01 Septiembre 2023]. Disponible en: https://recursospython.com/guias-y-manuales/checkbox-checkbutton-en-tcltk-tkinter/

RecursosPython. Lista Desplegable (Combobox) en Tkinter. RecursosPython. [Página web]. [consultado: 04 Agosto 2023]. Disponible en: https://recursospython.com/guias-y-manuales/lista-desplegable-combobox-en-tkinter/

RecursosPython. Vista de Árbol (TreeView) en Tkinter. RecursosPython. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://recursospython.com/guias-y-manuales/vista-de-arbol-treeview-en-tkinter/

Rosvel. ¿Qué es un píxel? Rosvel Blog. [Página web]. [consultado: 02 Septiembre 2023]. Disponible en: https://www.rosvel.com/blog/que-es-un-pixel/

Universidad de Jaén. Práctica 4: Visión por Computadora. [Documento PDF]. [consultado: 02 Septiembre 2023]. Disponible en: http://www4.ujaen.es/~satorres/practicas/practica4_vc.pdf

Uríbe Arango, W., Duque Ramírez, M., & Medina Durango, E. Electrocardiografía y Arritmias. Siocardio. [Documento PDF]. [consultado: 03 Agosto 2023]. Disponible en: https://www.siacardio.com/wp-content/uploads/2015/01/Libro-EKG-y-Arritmias-WU.pdf

Web y Empresas. ¿Qué es el Porcentaje de Error? Web y Empresas. [Página web]. [consultado: 03 Agosto 2023]. Disponible en: https://www.webyempresas.com/porcentaje-de-error/#%C2%BFQue_es_el_porcentaje_de_error

Welchallyn, PC-based resting ECG - Hill-ROM, [Libro web] https://www.welchallyn.com/content/dam/welchallyn/documents/upload-docs/Catalogs/Full-Line-Catalog/Cardiopulomonary.pdf

A. BENHAMIDA and M. KOZLOVSZKY, "Human ECG data collection, digitalization, streaming and storing," 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2020, pp. 105-110, doi: 10.1109/SAMI48414.2020.9108765. [Conferencia]

ANEXO

Anexo A. Carta de parte del médico evaluador

Me complace enormemente recomendar el proyecto de grado titulado "Interfaz Gráfica para Reconocimiento y Deteccion de Caracteristicas en Electrocardiogramas Usando Python" propuesto por los estudiantes Byron Hernando Galindo Suarez y Juan Esteban Narváez Carvajal en el campo de detección de electrocardiogramas. He tenido el privilegio de trabajar de cerca con estos estudiantes y he sido testiga de su dedicación y capacidad excepcionales.

Este proyecto aborda una cuestión de gran relevancia en la actualidad: la detección precisa y eficiente de anomalías en los electrocardiogramas. La detección temprana de problemas cardíacos es de vital importancia para la salud de los pacientes, y la utilización de técnicas de procesamiento de señales y análisis de datos puede tener un impacto significativo en este campo. Durante el desarrollo de su proyecto, estos estudiantes han demostrado una profunda comprensión de los fundamentos teóricos detrás de las señales cardíacas, así como la habilidad para implementar algoritmos sofisticados en plataformas computacionales.

Con base en su desempeño académico y sus habilidades demostradas, estoy segura de que los estudiantes realizadores del trabajo tienen el potencial para contribuir significativamente al avance de la detección de electrocardiogramas y, en última instancia, a la mejora de la atención médica en este campo. Recomiendo encarecidamente que se considere su proyecto ya que estoy convencida de que será un valioso aporte para la comunidad científica y médica.

Atentamente,

Angie Daniela Rendón Carvajal

Médico Cirujano MPPS: 163.516

Hospital Dr. Humberto De Pascuali

Guanare, Portuguesa, Venezuela

+58 4121536045

Anexo B. Código para el desarrollo de la interfaz

```
global ruta_PDF
global emergente
emergente = None


############################################################
###                 SECCION DE LIBRERIAS                 ###
############################################################

# Aqui se realiza la importacion de las librerias

#   Back

import os                                   # libreria para el manejo del sistema
import shutil                               # libreria para el control de archivos
import cv2                                  # libreria para el manejo de las imágenes
from pdf2image import convert_from_path     # libreria de conversion de archivos
import matplotlib.pyplot as plt             # libreria para visualizar imagen
import numpy as np                          # Libreria para manipular vectores de la
señal
import time                                 # Libreria para control de tiempo
import sys                                  # Libreria para puntos de ruptura
from scipy.signal import savgol_filter      # Libreria para suavizar la señal muestreada

#   Front
from tkinter import *
import tkinter as tk
from tkinter import messagebox, ttk
from tkinter import filedialog
from PIL import ImageTk, Image
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import numpy as np
import pandas as pd


############################################################
###         PREPARACION DE LAS RUTAS DE DATOS            ###
############################################################


# Aqui se ajustan las variables de rutas para el proyecto
# además de crear y validar los directorios necesarios
```

```
def Preparacion():
    if os.path.exists("./Code/Data/Output"):
        shutil.rmtree("./Code/Data/Output")
        os.mkdir("./Code/Data/Output")
    else:
        os.mkdir("./Code/Data/Output")


############################################################
####         CUERPO DE FUNCIONES DEL CÓDIGO             ###
############################################################

# Función de conversión de PDF a JPEG
def PDF_to_jpeg(ruta_PDF):
    global pages
    poppler_path = os.path.abspath("./Code/lib/poppler-22.04.0/Library/bin")            #
Configuracion de rutas de los archivos
    pdf_path = os.path.abspath(ruta_PDF)                                                #
Configuracion de rutas de los archivos
    saving_folder =
os.path.abspath("./Code/Data/Output")                                  # Configuracion de
rutas de los archivos
    pages = convert_from_path(pdf_path = pdf_path, poppler_path = poppler_path )    #
Obtención de la cantidad de páginas del documento
    i = 1
    for page in pages:
        img_name = f"img-{i}.jpeg"
        page.save(os.path.join(saving_folder,img_name),"JPEG")                          #
Conversion de las páginas a PDF
        i += 1
# Función de recorte de las imágenes
def crop_image():
    global emergente
    for i in range (2,6,1):
        image="img-"+ str(i) + ".jpeg"
        rimage=
os.path.abspath("./Code/Data/Output/"+image)                                  #
Construción de la ruta de imagen que se desea leer
        try:
            data = cv2.imread(rimage)                                                   #
Lectura de la imagen original
            data = data [204:2091,63:1638]                                              #
Recorte de la imágen
            cv2.imwrite('./Code/Data/Output/img-'+ str(i)+'-
crop.jpeg',data)                # Almacenaje de las immágenes recortadas
        except (TypeError):
            pass
```

```python
# Función para conectar las imágenes recortadas
def Concat_image():
    im2 = cv2.imread("./Code/Data/Output/img-2-crop.jpeg")
    im3 = cv2.imread("./Code/Data/Output/img-3-crop.jpeg")
    im4 = cv2.imread("./Code/Data/Output/img-4-crop.jpeg")
    im5 = cv2.imread("./Code/Data/Output/img-5-crop.jpeg")
    imga = cv2.hconcat([im2, im3])
    imgb = cv2.hconcat([im4, im5])
    img = cv2.hconcat([imga, imgb])
    img = img[:,97:6002]
    cv2.imwrite('./Code/Data/Output/img-ECG.jpeg',img)

# Función para limpiar la imágen
def Limpiar_imagen():
    img = cv2.imread("./Code/Data/Output/img-ECG.jpeg")
   ##########################################################################
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _,img = cv2.threshold(img,100,255,cv2.THRESH_BINARY)
   ##########################################################################
    mask = cv2.imread("./Code/lib/mask.png",0)
    for f in range(img.shape[0]):
        for c in range(img.shape[1]):
            if (int(mask[f,c]) == 0):
                img[f,c] = 255

    ##########################################################################
    cv2.imwrite('./Code/Data/Output/img-ECG.jpeg',img)

#Detecto los ejes y secciono la imágen
def Lines_Hough(img):
    global Ejesy
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150, apertureSize = 3)
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=500, maxLineGap=50)
    Ejesx=[]
    Ejesy=[]
    for line in lines:
        x1, y1, x2, y2 = line[0]
        if (x1 != x2):
            Ejesx.append(y1)
            cv2.line(img, (x1,y1), (x2,y2), (0,255,0), 1, cv2.LINE_AA)
        else:
            Ejesy.append(x1)
            cv2.line(img, (x1,y1), (x2,y2), (0,255,0), 1, cv2.LINE_AA)
    #Determino el valor de los ejes de cada derivación
    temp=sorted(Ejesx)
    Ejesx=[]
    aux=[]
    x=0
```

```
    i=0
    while (i+x)<len(temp)-1:
        while (int(temp[x+i])-int(temp[i]))<=10 and (i+x)<len(temp)-1:
            aux.append(int(temp[x+i]))
            x=x+1
        i=x+i
        x=0
        Ejesx.append(round((sum(aux)/len(aux))))
        aux=[]
    temp=sorted(Ejesy)
    Ejesy=[]
    aux=[]
    x=0
    i=0
    while (i+x)<len(temp)-1:
        while (int(temp[x+i])-int(temp[i]))<=10 and (i+x)<len(temp)-1:
            aux.append(int(temp[x+i]))
            x=x+1
        i=x+i
        x=0
        Ejesy.append(round((sum(aux)/len(aux))))
        aux=[]
    img =  cv2.imread("./Code/Data/Output/img-ECG.jpeg")
    cv2.imwrite('./Code/Data/Output/I.jpeg',img[Ejesx[0]-150:Ejesx[0]+150,:])
    cv2.imwrite('./Code/Data/Output/II.jpeg',img[Ejesx[1]-150:Ejesx[1]+150,:])
    cv2.imwrite('./Code/Data/Output/III.jpeg',img[Ejesx[2]-150:Ejesx[2]+150,:])
    cv2.imwrite('./Code/Data/Output/aVR.jpeg',img[Ejesx[3]-150:Ejesx[3]+150,:])
    cv2.imwrite('./Code/Data/Output/aVL.jpeg',img[Ejesx[4]-150:Ejesx[4]+150,:])
    cv2.imwrite('./Code/Data/Output/aVF.jpeg',img[Ejesx[5]-150:Ejesx[5]+150,:])


############################################################
####          CONSIDERACIONES DE RESOLUCIÓN            ####
##################                      ##################
# TIEMPO                                                   #
#                                                          #
# ------------  1 Cuadro -> 5 mm -> 0.2 Seg --------------#
#                                                          #
# VOLTAJE                                                  #
# ------------  1 Cuadro -> 5 mm -> 0.5 mV ---------------#
#                                                          #
# DIMENSIONES                                              #
#                                                          #
# ------------  1 Cuadro -> 39px X 39px  ---------------- #
# ---------  1 px Vertical -> 12.82051282 uV ------------ #
# --------  1 px Horizontal -> 5.128205128 mS  ---------- #
############################################################


# Vectorización de la Señal
def Vectorizacion_Señal():
```

```
    global aVF, aVL, aVR, I, II, III, Vtiempo
    global fallaCorte
    fallaCorte = False
    Signals = ['aVF','aVL','aVR','I','II','III']
    for signalstr in Signals:
        strtemp = './Code/Data/Output/' + signalstr + '.jpeg'
        img = cv2.imread(strtemp)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        _,img = cv2.threshold(gray,10,255,cv2.THRESH_BINARY)
        aux = np.asarray(img)
        aux1 = np.sum(aux, axis=0)
        posceros = np.where(aux1==76245)
        aux2 = aux[:,posceros[0][0]]
        Ejex = np.where(aux2==0)
        Ejesx = Ejex[0][0]
        aux2 = aux[0,:]
        posceros = np.where(aux2==0)
        Ejesy = posceros[0][:]
        if len(Ejesy) == 30 :
            for i in range(len(aux1)):
                if aux1[i] < 76245:
                    img[Ejesx, i] = 255
            for i in range(img.shape[0]):
                for ejey in Ejesy:
                    temp = (((img[i,(ejey+1)]==255) or (img[i,(ejey-1)]==255)) and
i!=Ejesx)
                    if temp:
                        img[i,ejey]=255
            signal=[]
            for i in range(img.shape[1]):
                temp=aux[0:img.shape[0], i]
                ceros =  np.where(temp==0)
                ceros = np.asanyarray(ceros[0])
                if len(ceros) == 0:
                    val = -10000
                    signal.append(val)
                else:
                    temp = np.where(abs(ceros - Ejesx)==np.amax(abs(ceros - Ejesx)))
                    if len(temp[0][:])==1:
                        signal.append(ceros[temp][0])
                    else:
                        temp = ceros[temp]
                        try:
                            postemp=np.where(abs(temp-signal[-1])==np.amax(abs(temp-
signal[-1])))
                            signal.append(temp[postemp][0])
                        except IndexError:
                            fallaCorte = True
            signal = np.array(signal)
            huecos = np.where(signal==-10000)
            for hueco in huecos:
```

```
                signal[hueco] = ((signal[hueco + 1]) + (signal[hueco - 1]))/2
            signal =(signal - Ejesx)*-0.000012820512819999999
            signal = savgol_filter(signal,7,2)
            if signalstr == 'aVF':
                aVF = signal
            elif signalstr == 'aVR':
                aVR = signal
            elif signalstr == 'aVL':
                aVL = signal
            elif signalstr == 'I':
                I = signal
            elif signalstr == 'II':
                II = signal
            else:
                III = signal
            Vtiempo = (np.arange(0,int(signal.shape[0]),1))*0.005128205128000001
        else:
            fallaCorte = True

#Caracterización de la señal
def ObtencionPico():
    global aVF, aVL, aVR, I, II, III, Vtiempo
    global SegmentoSeñal, SegmentoTiempo
    global Maximos, Periodo
    global FrecuenciaCardiaca

    if (len(np.where(I>0.0015)[0])>0):
        print('Descarte la señal')
        print('La señal presenta mucho ruido y picos demasiado altos')
        sys.exit()
    else:
        Maximos = np.where(I>0.0005)[0]
        if len(Maximos) == 0:
            print('Desarte la señal, la onda R se encuentra bajo el estandar de 0.5 mV')
            sys.exit()
        else:
            maximos=np.asarray(Maximos[:])
            if len(maximos) < 150:
                print('Error en la detección')
                print('La onda R no se detecta como marcador')
                sys.exit()
            else:
                temp = []
                aux = []
                Maximos = []
                for i in range(len(maximos)):
                    if i >=1 :
                        ja = maximos[i] - maximos[i-1]
                        if (maximos[i] - maximos[i-1]) <= 20 :
                            temp.append(maximos[i])
                        else:
```

```
                            aux = np.asarray(I[temp][:])


                            indice = np.where(aux == indice)[:][0][0]
                            Maximos.append(temp[indice])
                            aux = []
                            temp = []
                    else:
                        temp.append(maximos[i])

                temp = []
                for i in range(len(Maximos)):
                    if i >= 1:
                        temp.append(Maximos[i] - Maximos[i-1])
                Periodo = round(np.mean(temp))
                FrecuenciaCardiaca = len(Maximos)*2
                Picos = I[Maximos]-0.0005
                indice = [np.where(Picos==np.min(Picos))][:][0][0][0]
                indice = Maximos[indice]
                Pico = I[indice]
                Time = Vtiempo[indice]

                SegmentoSeñal = I[indice-round(Periodo/2):indice+round(Periodo/2)]
                SegmentoTiempo = Vtiempo[indice-round(Periodo/2):indice+round(Periodo/2)]

#Obtener características del pico
def Características():
    global SegmentoSeñal, SegmentoTiempo, Periodo
    global VdatoTiempo, VdatoAmplitud, VdatoString
    global divergencias

    divergencias = []
    SegmentoSeñal = savgol_filter(SegmentoSeñal,10,5)  # Filtro la señal

    d1x = np.diff(SegmentoSeñal)        # Extraigo las derivadas
    d2x = np.diff(d1x)


    ##############################################################################################
    ####            CALCULO LAS ONDAS POR OBSERVACIONES EN LAS DERIVADAS            ####
    ##############################################################################################


    picos_indices = np.where((np.diff(np.sign(d1x)) < 0) & (d1x[:-1] > 0))[0] +
1
    CaD = SegmentoTiempo[picos_indices]
    picos_indices = np.where((np.diff(np.sign(d1x)) > 0) & (d1x[:-1] < 0))[0] + 1
    DaC = SegmentoTiempo[picos_indices]
    picos_indices = np.where(np.diff(np.sign(d1x)))[0] + 1
    Picos = SegmentoTiempo[picos_indices]
```

```
    TiempoR = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal))])][:][0]

    TiempoQa = DaC[np.where(DaC == SegmentoTiempo[np.where(SegmentoSeñal ==
np.min(SegmentoSeñal[0:np.where(SegmentoTiempo == TiempoR)[:][0][0]]))])][:][0]
    TiempoQb = Picos[np.where(Picos == TiempoR)[:][0][0]-1]
    if TiempoQa == TiempoQb:
        TiempoQ = TiempoQa
    else:
        divergencias.append('Q')
        if ((TiempoR - TiempoQa > 0.13) and (TiempoR - TiempoQb < 0.13)) :
            TiempoQ = TiempoQb
        elif ((TiempoR - TiempoQa < 0.13) and (TiempoR - TiempoQb > 0.13)):
            TiempoQ = TiempoQa
        elif SegmentoSeñal[np.where(SegmentoTiempo ==TiempoQa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoQb)[:][0][0]]:
            TiempoQ = TiempoQb
        else:
            TiempoQ = TiempoQa

    TiempoSa = DaC[np.where(DaC == SegmentoTiempo[np.where(SegmentoSeñal ==
np.min(SegmentoSeñal[np.where(SegmentoTiempo == TiempoR)[:][0][0]:]))])][:][0]
    TiempoSb = Picos[np.where(Picos == TiempoR)[:][0][0]+1]
    if TiempoSa == TiempoSb:
        TiempoS = TiempoSa
    else:
        divergencias.append('S')
        if ((TiempoSa - TiempoR > 0.13) and (TiempoSb - TiempoR < 0.13)) :
            TiempoS = TiempoSb
        elif ((TiempoSa - TiempoR < 0.13) and (TiempoSb - TiempoR > 0.13)) :
            TiempoS = TiempoSa
        elif SegmentoSeñal[np.where(SegmentoTiempo ==TiempoSa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoSb)[:][0][0]]:
            TiempoS = TiempoSb
        else:
            TiempoS = TiempoSa

    TiempoPa = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal[:np.where(SegmentoTiempo == TiempoQ)[:][0][0]]))])][:][0]
    TiempoPb = Picos[np.where(Picos == TiempoQ)[:][0][0]-1]
    if TiempoPa == TiempoPb:
        TiempoP = TiempoPa
    else:
        divergencias.append('P')
        if SegmentoSeñal[np.where(SegmentoTiempo ==TiempoPa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoPb)[:][0][0]]:
            TiempoP = TiempoPa
        else:
            TiempoP = TiempoPb
```

```
    TiempoTa = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal[np.where(SegmentoTiempo == TiempoS)[:][0][0]:]))])][:][0]
    TiempoTb = Picos[np.where(Picos == TiempoS)[:][0][0]+1]
    if TiempoTa == TiempoTb:
        TiempoT = TiempoTa
    else:
        divergencias.append('T')
        if SegmentoSeñal[np.where(SegmentoTiempo ==TiempoTa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoTb)[:][0][0]]:
            TiempoT = TiempoTa
        else:
            TiempoT = TiempoTb

    xD = d2x[np.where(SegmentoTiempo == Picos[np.where(Picos == TiempoP)[:][0][0]-
1])[:][0][0] : np.where(SegmentoTiempo == TiempoP)[:][0][0]]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == Picos[np.where(Picos ==
TiempoP)[:][0][0]-1])[:][0][0] : np.where(SegmentoTiempo == TiempoP)[:][0][0]]
    TiempoPI =  aux[np.where(xD == np.max(xD))[:][0][0]]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == TiempoP)[:][0][0] :
np.where(SegmentoTiempo == TiempoQ)[:][0][0]]
    TiempoPF = aux[int(len(aux)/2)]

    aux = SegmentoTiempo[np.where(SegmentoTiempo == Picos[np.where(Picos ==
TiempoT)[:][0][0]-1])[:][0][0] : np.where(SegmentoTiempo == TiempoT)[:][0][0]]
    TiempoTI = aux[int(len(aux)/2)]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == TiempoT)[:][0][0] :
np.where(SegmentoTiempo == Picos[np.where(Picos == TiempoT)[:][0][0]+1])[:][0][0] ]
    TiempoTF = aux[int(len(aux)/2)]

    R = SegmentoSeñal[np.where(SegmentoTiempo == TiempoR)][:][0]
    S = SegmentoSeñal[np.where(SegmentoTiempo == TiempoS)][:][0]
    T = SegmentoSeñal[np.where(SegmentoTiempo == TiempoT)][:][0]
    Q = SegmentoSeñal[np.where(SegmentoTiempo == TiempoQ)][:][0]
    P = SegmentoSeñal[np.where(SegmentoTiempo == TiempoP)][:][0]
    PI = SegmentoSeñal[np.where(SegmentoTiempo == TiempoPI)][:][0]
    PF = SegmentoSeñal[np.where(SegmentoTiempo == TiempoPF)][:][0]
    TI = SegmentoSeñal[np.where(SegmentoTiempo == TiempoTI)][:][0]
    TF = SegmentoSeñal[np.where(SegmentoTiempo == TiempoTF)][:][0]

    VdatoTiempo = [TiempoPI, TiempoP, TiempoPF, TiempoQ, TiempoR, TiempoS, TiempoTI,
TiempoT, TiempoTF]
    VdatoAmplitud = [PI, P, PF, Q, R, S, TI, T, TF]
    VdatoString = ['PI', 'P', 'PR', 'Q', 'R', 'S', 'TI', 'T', 'TF']

# Obtencion de los segmentos y datos específicos
def ObtencionParametros():
    global VdatoAmplitud, VdatoTiempo, VdatoString, Periodo
    global VdataSegmento, VdatoSegmento

    SegmentoRR = Periodo*0.005128205128000001
    SegmentoPR = VdatoTiempo[3] - VdatoTiempo[0]
```

```
    SegmentoQRS = VdatoTiempo[5] - VdatoTiempo[3]
    SegmentoQT = ((VdatoTiempo[8] - VdatoTiempo[3])/np.sqrt(SegmentoRR))
    SegmentoST = VdatoTiempo[6] - VdatoTiempo[5]

    VdatoSegmento = ['RR', 'PR', 'QRS', 'QT', 'ST']
    VdataSegmento = [SegmentoRR, SegmentoPR, SegmentoQRS, SegmentoQT, SegmentoST]


############################################################
###        DESARROLLO DE LA INTERFAZ DE USUARIO        ###
############################################################
# Parametros de personalizacion
fondo = "#FFF"
fuente = "PT Sans"
ancho = 900
alto = 700
color_letra_titulo = "#8F141B"
color_letra_texto = "#000"

# Declaracion de la ventana
ventana = Tk()

#Configuracion para pantalla centrada y dimensiones de la ventana
ancho_ventana = ventana.winfo_screenwidth() // 2 - ancho // 2
alto_ventana = ventana.winfo_screenheight() // 2 - alto // 2
posicion = str(ancho) + "x" + str(alto) + "+" + str(ancho_ventana) + "+" +
str(alto_ventana)
ventana.geometry(posicion)

#Personalizacion de la ventana
ventana.resizable(0,0)
       # No puede maximizarse
icono = PhotoImage(file =
"./Code/lib/usco.png")                                                    # Icono
de la ventana
ventana.iconphoto(True, icono)
ventana.title("Interfaz grafica de digitalización y detección de enfermedades
cardíacas")       # Titulo de la ventana
ventana.config(bg = fondo)

frameMayor = Frame(ventana, bg = fondo)
frameMayor.pack(fill = 'both', expand = True)

imagenUsco = ImageTk.PhotoImage(Image.open("./Code/lib/universidad-surcolombiana.png"))
archivo = None

def inicio():
    Preparacion()
    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    global inicioFrame
```

```
    inicioFrame = Frame(frameMayor, bg = "#F2F2F2")
    inicioFrame.pack(fill = 'both', expand = 1)

    frameHeader = Frame(inicioFrame, bg = "#F2F2F2")
    frameMid = Frame(inicioFrame, bg = "#F2F2F2")
    frameFooter = Frame(inicioFrame, bg = "#F2F2F2")

    # Declaracion de etiquetas
    label1 = Label(frameHeader, text = "Interfaz de digitalización y detección de
enfermedades cardíacas", fg = color_letra_titulo, bg = "#F2F2F2", font = (fuente, 20,
"bold"))
    label2 = Label(frameMid, text = "Byron Hernando Galindo Suárez - 20171155352", fg =
"black", bg = "#F2F2F2", font = (fuente, 17, "bold"))
    label3 = Label(frameMid, text = "Juan Esteban Narváez Carvajal - 20171159625", fg =
"black", bg = "#F2F2F2", font = (fuente, 17, "bold"))
    label4 = Label(frameMid, image = imagenUsco, bg ="#F2F2F2")

    # Declaracion de botones
    btnContinuar = Button(frameFooter, text = "Continuar", font = (fuente, 20, "bold"),
command = datos, width = 20, height = 20)

    # Posicionamiento de frames
    frameHeader.pack(side = "top", fill = "both")
    frameMid.pack(fill = "both", expand = True)
    frameFooter.pack(side = "bottom", fill = "both")

    # Posicionamiento de etiquetas
    label1.pack(pady = 100)
    label2.pack()
    label3.pack()
    label4.pack(pady = 75)

    # Posicionamiento de botones
    btnContinuar.pack(pady = 50)

def datos():
    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    global datosFrame
    datosFrame = Frame(frameMayor, bg = "#F2F2F2")
    datosFrame.pack(fill = "both", expand = 1)



    frameHeader = Frame(datosFrame, bg = "#F2F2F2")
    frameMid = Frame(datosFrame, bg = "#F2F2F2")
    frameFooter = Frame(datosFrame, bg = "#F2F2F2")

    # Posicionamiento de frames
    frameHeader.pack(side = "top", fill = "both")
    frameMid.pack(fill = "both", expand = True)
```

```
    frameFooter.pack(side = "bottom", fill = "both")

    lbl7 = Label(frameMid, text = "Archivo: ", fg = "black", bg = "#F2F2F2", font =
(fuente, 10, "bold"))

    def abrirArchivo():
        btnProcesar['state'] = DISABLED
        lbl7['text'] = "Archivo: "
        global archivo, pages
        try:
            archivo = filedialog.askopenfile(title = "Abrir", initialdir =
os.path.abspath(os.getcwd()), filetypes = (("Formato PDF", "*.pdf"),))
            if archivo.name != None :
                PDF_to_jpeg(archivo.name)
                lbl7['text'] = lbl7['text'] + str(archivo.name)
                ventana.imgtk = ImageTk.PhotoImage((Image.open("./Code/Data/Output/img-
1.jpeg")).resize((250,300)))
                labelvista.configure(image=ventana.imgtk)
                labelvista.pack()
                if len(pages)==5:
                    btnProcesar['state'] = NORMAL
                else:
                    emergente = tk.Toplevel(ventana)
                    emergente.title('File Error!!!')
                    emergente.geometry(str('290x180+' +
str(ventana.winfo_screenwidth()//2 - 290 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                    emergente.configure(background = 'white')
                    emergente.attributes("-toolwindow", True)
                    emergente.resizable(False,False)
                    emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                    etiqueta = tk.Label(emergente, text = "Error!!!")
                    etiqueta.pack( anchor = CENTER)
                    etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                    labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                    ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta.png")))
                    labelAlerta.configure(image=ventana.alerta)
                    labelAlerta.pack()
                    etiqueta2 = tk.Label(emergente, text = "El archivo seleccionado no
cumple con las \ncaracterísticas de un ECG de Kardia")
                    etiqueta2.pack(anchor = CENTER)
                    etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
                    btn_aceptar = tk.Button(emergente, text = "Aceptar",  command =
emergente.destroy,  font = (fuente, 10, "bold"))
                    btn_aceptar.pack(pady = 10)
                    btnProcesar['state'] = DISABLED
        except AttributeError:
            pass
```

```
    lbl5 = Label(frameHeader, text = "Por favor, cargue el archivo en formato PDF\n del
electrocardiograma entregado por Kardia 6L", fg = "black", bg = "#F2F2F2", font =
(fuente, 12))

    btnCargar = Button(frameHeader, text = "Cargar archivo", font = (fuente, 15, "bold"),
command = abrirArchivo)
    lbl6 = Label(frameMid, text = "Previsualizacion del pdf cargado", fg = "black", bg =
"#F2F2F2", font = (fuente, 15, "bold"))
    frameVisualizacion = Frame(frameMid, bg = "white", width = 250, height = 300,
highlightbackground = "black", highlightthickness = 2)
    labelvista = Label(frameVisualizacion, bg = "#F2F2F2", image = None)
    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 20, "bold"), command
= inicio)
    btnProcesar = Button(frameFooter, text = "Procesar", font = (fuente, 20, "bold"),
command = resultados, state = DISABLED)

    lbl5.pack(side = "left", padx = 75, pady = 75)
    btnCargar.pack(side = "right", padx = 75, pady = 75)
    lbl6.pack()
    frameVisualizacion.pack(pady = 10)
    lbl7.pack()
    btnVolver.pack(side = "left", padx = 75, pady = 25)
    btnProcesar.pack(side = "right", padx = 75, pady = 25)

def resultados():
    global fallaCorte, ancho_ventana, alto_ventana, divergencias
    global resultadosFrame, SignalPlot, Clicks, Capsulasegmento, Capsulapico
    global I, II, III, aVR, aVL, aVF, Vtiempo, SegmentoTiempo, SegmentoSeñal
    global Inicio_pico_Seg, Fin_pico_Seg
    global Maximos, Periodo
    global FrecuenciaCardiaca

    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    crop_image()
    Concat_image()
    Limpiar_imagen()
    Lines_Hough(cv2.imread('./Code/Data/Output/img-ECG.jpeg'))
    Vectorizacion_Señal()
    if fallaCorte == True:
        emergente = tk.Toplevel(ventana)
        emergente.title('Process Error!!!')
        emergente.geometry(str('390x250+' + str(ventana.winfo_screenwidth()//2 - 390 //
2) + "+" + str(ventana.winfo_screenheight() // 2 - 250 // 2)))
        emergente.configure(background = 'white')
        emergente.attributes("-toolwindow", True)
        emergente.resizable(False,False)
        emergente.protocol("WM_DELETE_WINDOW", lambda: None)
        etiqueta = tk.Label(emergente, text = "Error")
        etiqueta.pack( anchor = CENTER)
        etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
```

```
        labelAlerta = tk.Label(emergente, bg = fondo, image = None)
        ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
        labelAlerta.configure(image=ventana.alerta)
        labelAlerta.pack()
        etiqueta2 = tk.Label(emergente, text = "La señal detectada fue cortada durante la
digitalización, \nla señal puede tener mucho ruido o \n es una señal con muy poca
amplitud. \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
        etiqueta2.pack(anchor = CENTER)
        etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
        def alfa():
            emergente.destroy()
            datos()
        btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,  font =
(fuente, 10, "bold"))
        btn_aceptar.pack(pady = 10)
    else:
        if (len(np.where(I>0.0015)[0])>0):
            emergente = tk.Toplevel(ventana)
            emergente.title('Process Error!!!')
            emergente.geometry(str('310x210+' + str(ventana.winfo_screenwidth()//2 - 310
// 2) + "+" + str(ventana.winfo_screenheight() // 2 - 210 // 2)))
            emergente.configure(background = 'white')
            emergente.attributes("-toolwindow", True)
            emergente.resizable(False,False)
            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
            etiqueta = tk.Label(emergente, text = "Error")
            etiqueta.pack( anchor = CENTER)
            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
            labelAlerta.configure(image=ventana.alerta)
            labelAlerta.pack()
            etiqueta2 = tk.Label(emergente, text = "La señal detectada presenta demasido
ruido \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
            etiqueta2.pack(anchor = CENTER)
            etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
            def alfa():
                emergente.destroy()
                datos()
            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,  font =
(fuente, 10, "bold"))
            btn_aceptar.pack(pady = 10)

        else:
            Maximos = np.where(I>0.0005)[0]
            if len(Maximos) == 0:
                emergente = tk.Toplevel(ventana)
```

```
                emergente.title('Process Error!!!')
                emergente.geometry(str('250x210+' + str(ventana.winfo_screenwidth()//2 -
250 // 2) + "+" + str(ventana.winfo_screenheight() // 2 - 210 // 2)))
                emergente.configure(background = 'white')
                emergente.attributes("-toolwindow", True)
                emergente.resizable(False,False)
                emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                etiqueta = tk.Label(emergente, text = "Error")
                etiqueta.pack( anchor = CENTER)
                etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
                labelAlerta.configure(image=ventana.alerta)
                labelAlerta.pack()
                etiqueta2 = tk.Label(emergente, text = "En la señal detectada presenta
poca amplitud \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
                etiqueta2.pack(anchor = CENTER)
                etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
                def alfa():
                    emergente.destroy()
                    datos()
                btn_aceptar = tk.Button(emergente, text = "Aceptar",  command =
alfa,  font = (fuente, 10, "bold"))
                btn_aceptar.pack(pady = 10)
            else:
                maximos=np.asarray(Maximos[:])
                if len(maximos) < 100:
                    emergente = tk.Toplevel(ventana)
                    emergente.title('Process Error!!!')
                    emergente.geometry(str('250x210+' +
str(ventana.winfo_screenwidth()//2 - 250 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 210 // 2)))
                    emergente.configure(background = 'white')
                    emergente.attributes("-toolwindow", True)
                    emergente.resizable(False,False)
                    emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                    etiqueta = tk.Label(emergente, text = "Error")
                    etiqueta.pack( anchor = CENTER)
                    etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                    labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                    ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
                    labelAlerta.configure(image=ventana.alerta)
                    labelAlerta.pack()
                    etiqueta2 = tk.Label(emergente, text = "Error en la detección de la
onda R \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
                    etiqueta2.pack(anchor = CENTER)
                    etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
```

```
                    def alfa():
                        emergente.destroy()
                        datos()
                    btn_aceptar = tk.Button(emergente, text = "Aceptar",  command =
alfa,  font = (fuente, 10, "bold"))
                    btn_aceptar.pack(pady = 10)
                else:
                    temp = []
                    aux = []
                    Maximos = []
                    for i in range(len(maximos)):
                        if i >=1 :
                            ja = maximos[i] - maximos[i-1]
                            if (maximos[i] - maximos[i-1]) <= 20 :
                                temp.append(maximos[i])
                            else:
                                aux = np.asarray(I[temp][:])
                                indice = np.max(aux)
                                indice = np.where(aux == indice)[:][0][0]
                                Maximos.append(temp[indice])
                                aux = []
                                temp = []
                        else:
                            temp.append(maximos[i])

                    temp = []
                    for i in range(len(Maximos)):
                        if i >= 1:
                            temp.append(Maximos[i] - Maximos[i-1])
                    Periodo = round(np.mean(temp))
                    FrecuenciaCardiaca = len(Maximos)*2
                    Picos = I[Maximos]-0.0005
                    indice = [np.where(Picos==np.min(Picos))][:][0][0][0]
                    indice = Maximos[indice]
                    Pico = I[indice]
                    Time = Vtiempo[indice]

                    SegmentoSeñal = I[indice-round(Periodo/2):indice+round(Periodo/2)]
                    SegmentoTiempo = Vtiempo[indice-
round(Periodo/2):indice+round(Periodo/2)]

                    Características()
                    ObtencionParametros()


                    SignalPlot = I
                    Clicks = 0

                    # Vista por defecto  es con la señal I
```

```
                    Inicio_pico_Seg = Vtiempo[np.where (Vtiempo ==
SegmentoTiempo[0])][:][0]
                    Fin_pico_Seg = Vtiempo[np.where (Vtiempo == SegmentoTiempo[-
1])][:][0]

                    Datospicoplottiempo = Vtiempo[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]
                    Datospicoplotseñal = SignalPlot[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]*1000

                    Datossegmentoplottiempo = Vtiempo[(Clicks*590):((Clicks+1)*590)]
                    Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000

                    resultadosFrame = Frame(frameMayor, bg = "#F2F2F2")
                    resultadosFrame.pack(fill = "both", expand = 1)
                    frameHeader = Frame(resultadosFrame, height = 50, bg = "#F2F2F2")
                    frameParametros = Frame(resultadosFrame, height = 100, bg =
"#F2F2F2")
                    frameFooter = Frame(resultadosFrame, height = 50, bg = "#F2F2F2")
                    ecgFrame = Frame(frameParametros, height = 50, bg = "#F2F2F2")

                    def res_3s():
                        global Clicks, SignalPlot, Capsulasegmento, Vtiempo
                        if Clicks > 0:
                            Capsulasegmento.get_tk_widget().destroy()
                            Clicks = Clicks - 1
                            Datossegmentoplottiempo =
Vtiempo[(Clicks*590):((Clicks+1)*590)]
                            Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000
                            figure = plt.Figure(dpi=55)
                            Segmento = figure.add_subplot(1, 1, 1)
                            Segmento.set_ylabel('Amplitud (mV)')
                            Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                            Segmento.set_xlabel('Tiempo (s)')
                            Segmento.set_title(' Señal I')
                            Capsulasegmento = FigureCanvasTkAgg(figure, master =
ecgFrame)
                            Capsulasegmento.get_tk_widget().pack(pady = 5)
                            # aqui debe de actualizar el plot

                        # Debe de desplazar la señal hacia atras 6seg, ojo para clicks >
0

                    def sum_3s():
                        global Clicks, SignalPlot, Capsulasegmento, Vtiempo
                        if Clicks < 9:
                            Capsulasegmento.get_tk_widget().destroy()
                            Clicks = Clicks + 1
```

```
                            Datossegmentoplottiempo =
Vtiempo[(Clicks*590):((Clicks+1)*590)]
                            Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000
                            figure = plt.Figure(dpi=55)
                            Segmento = figure.add_subplot(1, 1, 1)
                            Segmento.set_ylabel('Amplitud (mV)')
                            Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                            Segmento.set_xlabel('Tiempo (s)')
                            Segmento.set_title(' Señal I')
                            Capsulasegmento = FigureCanvasTkAgg(figure, master =
ecgFrame)
                            Capsulasegmento.get_tk_widget().pack(pady = 5)
                            # aqui debe de actualizar el plot

                        # debe de desplazar la señal hacia adelante 6seg, ojo para clicks
< 5

                    def Cambiar_grafica(event):
                        global SignalPlot, Capsulasegmento, Capsulapico, Clicks
                        global Inicio_pico_Seg, Fin_pico_Seg
                        global Vtiempo, VdatoTiempo, VdataSegmento, VdatoSegmento
                        global VdatoTiempo, VdatoAmplitud, VdatoString,
FrecuenciaCardiaca

                        Clicks = 0

                        SignalSelect = desplegable.get()
                        SignalPlot = globals()[SignalSelect]

                        Datospicoplottiempo = Vtiempo[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]
                        Datospicoplotseñal = SignalPlot[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]*1000

                        Datossegmentoplottiempo = Vtiempo[(Clicks*590):((Clicks+1)*590)]
                        Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000

                        Capsulasegmento.get_tk_widget().destroy()
                        figure = plt.Figure(dpi=55)
                        Segmento = figure.add_subplot(1, 1, 1)
                        Segmento.set_ylabel('Amplitud (mV)')
                        Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                        Segmento.set_xlabel('Tiempo (s)')
                        Segmento.set_title(' Señal '+ SignalSelect)
                        Capsulasegmento = FigureCanvasTkAgg(figure, master = ecgFrame)
                        Capsulasegmento.get_tk_widget().pack(pady = 5)
                        Capsulapico.get_tk_widget().destroy()
                        P = (SignalPlot[np.where(Vtiempo == VdatoTiempo[1])][:][0])*1000
                        Q = (SignalPlot[np.where(Vtiempo == VdatoTiempo[3])][:][0])*1000
```

```
                        R = (SignalPlot[np.where(Vtiempo ==
VdatoTiempo[4])][:][0])*1000
                        S = (SignalPlot[np.where(Vtiempo == VdatoTiempo[5])][:][0])*1000
                        T = (SignalPlot[np.where(Vtiempo ==
VdatoTiempo[7])][:][0])*1000
                        figure = plt.Figure(dpi=55)
                        Pico = figure.add_subplot(1, 1, 1)
                        Pico.set_ylabel('Amplitud (mV)')
                        Pico.plot(Datospicoplottiempo,Datospicoplotseñal)
                        Pico.annotate("P", xy = (VdatoTiempo[1],P), xytext =
(VdatoTiempo[1],P+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("Q", xy = (VdatoTiempo[3],Q), xytext =
(VdatoTiempo[3],Q-0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("R", xy = (VdatoTiempo[4],R), xytext =
(VdatoTiempo[4],R+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("S", xy = (VdatoTiempo[5],S), xytext =
(VdatoTiempo[5],S-0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("T", xy = (VdatoTiempo[7],T), xytext =
(VdatoTiempo[7],T+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.set_xlabel('Tiempo (s)')
                        Pico.set_title('Pico Seleleccionado en '+ SignalSelect)
                        Capsulapico = FigureCanvasTkAgg(figure, master = picoFrame)
                        Capsulapico.get_tk_widget().pack(pady = 5)

                        #actualizar el graficos el segmento y el pico

                    btnAtras = Button(ecgFrame, text = "- 3s", font = (fuente, 15,
"bold"), command = res_3s)
                    btnAdelante = Button(ecgFrame, text = "+ 3s", font = (fuente, 15,
"bold"), command = sum_3s)

                    btnAtras.pack(side = 'left', padx = (150, 0), pady = 50)
                    btnAdelante.pack(side = 'right', padx = (0, 150), pady = 50)

                    figure = plt.Figure(dpi=55)
                    Segmento = figure.add_subplot(1, 1, 1)
                    Segmento.set_ylabel('Amplitud (mV)')
                    Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                    Segmento.set_xlabel('Tiempo (s)')
                    Segmento.set_title(' Señal I')
                    Capsulasegmento = FigureCanvasTkAgg(figure, master = ecgFrame)
                    Capsulasegmento.get_tk_widget().pack(pady = 5)
                    picoFrame = Frame(frameParametros, height = 50, bg = "#F2F2F2")
                    frameComplejos = Frame(picoFrame, highlightbackground="black",
highlightthickness=2)
                    frameComplejos.pack(side = "left", padx = (75,0))
                    labelCabecera = Label(frameComplejos,fg = color_letra_texto, text =
'Características', bg = "#F2F2F2", font = (fuente, 13, "bold"))
                    labelCabecera.pack(pady=10)
                    tabla = ttk.Treeview(frameComplejos)
                    tabla['columns'] = ('Parámetro', 'Valor')
```

```
                    tabla.column('#0', width=0, stretch=tk.NO)  # Columna de índice
oculta
                    tabla.column('Parámetro', width=115)
                    tabla.column('Valor', width=100)
                    tabla.heading('#0', text='', anchor=tk.W)
                    tabla.heading('Parámetro', text='Parámetro', anchor=tk.W)
                    tabla.heading('Valor', text='Valor', anchor=tk.W)
                    tabla.insert('', 'end', text='1', values=('Frecuencia Cardiaca
',str(FrecuenciaCardiaca)+' LPM' ))
                    tabla.insert('', 'end', text='2', values=('Momento ' +
VdatoString[1],str(round(VdatoTiempo[1],2))+' Seg' ))
                    tabla.insert('', 'end', text='3', values=('Momento ' +
VdatoString[3],str(round(VdatoTiempo[3],2))+' Seg' ))
                    tabla.insert('', 'end', text='4', values=('Momento ' +
VdatoString[4],str(round(VdatoTiempo[4],2))+' Seg' ))
                    tabla.insert('', 'end', text='5', values=('Momento ' +
VdatoString[5],str(round(VdatoTiempo[5],2))+' Seg' ))
                    tabla.insert('', 'end', text='6', values=('Momento ' +
VdatoString[7],str(round(VdatoTiempo[7],2))+' Seg' ))
                    tabla.insert('', 'end', text='7', values=('Duración ' +
VdatoString[1],str(round(VdatoTiempo[2]-VdatoTiempo[0],2))+' Seg' ))
                    tabla.insert('', 'end', text='8', values=('Duración ' +
VdatoString[7],str(round(VdatoTiempo[8]-VdatoTiempo[6],2))+' Seg' ))
                    tabla.insert('', 'end', text='9', values=('Amplitud ' +
VdatoString[4],str(round(VdatoAmplitud[4]*1000,2))+' mV' ))
                    tabla.insert('', 'end', text='10', values=('Dur. Comp. ' +
VdatoSegmento[0],str(round(VdataSegmento[0],2))+' Seg' ))
                    tabla.insert('', 'end', text='11', values=('Dur. Comp. ' +
VdatoSegmento[1],str(round(VdataSegmento[1],2))+' Seg' ))
                    tabla.insert('', 'end', text='12', values=('Dur. Comp. ' +
VdatoSegmento[2],str(round(VdataSegmento[2],2))+' Seg' ))
                    tabla.insert('', 'end', text='13', values=('Dur. Comp. ' +
VdatoSegmento[3],str(round(VdataSegmento[3],2))+' Seg' ))
                    tabla.insert('', 'end', text='14', values=('Dur. Comp. ' +
VdatoSegmento[4],str(round(VdataSegmento[4],2))+' Seg' ))
                    tabla.pack()

                    desplegable = ttk.Combobox(picoFrame, values = ["I", "II", "III",
"aVR", "aVL", "aVF"], font = (fuente, 13), width = 5)
                    desplegable.set("I")
                    desplegable.bind("<<ComboboxSelected>>", Cambiar_grafica)
                    desplegable.pack(side = "right", padx = (0,75))

                    P = (SignalPlot[np.where(Vtiempo == VdatoTiempo[1])][:][0])*1000
                    Q = (SignalPlot[np.where(Vtiempo == VdatoTiempo[3])][:][0])*1000
                    R = (SignalPlot[np.where(Vtiempo ==
VdatoTiempo[4])][:][0])*1000
                    S = (SignalPlot[np.where(Vtiempo == VdatoTiempo[5])][:][0])*1000
                    T = (SignalPlot[np.where(Vtiempo == VdatoTiempo[7])][:][0])*1000

                    figure = plt.Figure(dpi=55)
```

```
                    Pico = figure.add_subplot(1, 1, 1)
                    Pico.set_ylabel('Amplitud (mV)')
                    Pico.plot(Datospicoplottiempo,Datospicoplotseñal)
                    Pico.annotate("P", xy = (VdatoTiempo[1],P), xytext =
(VdatoTiempo[1]+0.02,P), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("Q", xy = (VdatoTiempo[3],Q), xytext =
(VdatoTiempo[3]+0.02,Q), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("R", xy = (VdatoTiempo[4],R), xytext =
(VdatoTiempo[4]+0.02,R), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("S", xy = (VdatoTiempo[5],S), xytext =
(VdatoTiempo[5]+0.02,S), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("T", xy = (VdatoTiempo[7],T), xytext =
(VdatoTiempo[7]+0.02,T), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.set_xlabel('Tiempo (s)')
                    Pico.set_title('Pico Seleleccionado en I')
                    Capsulapico = FigureCanvasTkAgg(figure, master = picoFrame)
                    Capsulapico.get_tk_widget().pack(pady = 5)

                    lbl8 = Label(frameHeader, text = "PARAMETROS", fg = "black", font =
(fuente, 17, "bold"), bg = "#F2F2F2")

                    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 12,
"bold"), command = datos)
                    btnDiagnosticar = Button(frameFooter, text = "Diagnosticar", font =
(fuente, 12, "bold"), command = diagnostico)

                    frameHeader.pack(side = 'top', fill = 'x')
                    frameParametros.pack(fill = 'both', expand = '1')
                    frameFooter.pack(side = 'bottom', fill = 'x')
                    ecgFrame.pack(side = 'top', fill = 'both', expand = True)
                    picoFrame.pack(side = 'bottom', fill = 'both', expand = True)

                    lbl8.pack(side = 'top', fill = 'both', expand = 1, pady = 5)

                    btnVolver.pack(side = 'left', padx = (100, 0), pady = 25)
                    btnDiagnosticar.pack(side = 'right', padx = (0, 100), pady = 25)
                    if len(divergencias)>=1:
                        if len(divergencias)==1:
                            emergente = tk.Toplevel(ventana)
                            emergente.title('Advertencia')
                            emergente.geometry(str('240x180+' +
str(ventana.winfo_screenwidth()//2 - 240 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                            emergente.configure(background = 'white')
                            emergente.attributes("-toolwindow", True)
                            emergente.resizable(False,False)
                            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                            etiqueta = tk.Label(emergente, text = "Advertencia!!!")
                            etiqueta.pack( anchor = CENTER)
                            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg
='white')
```

```
                            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta3.png")))
                            labelAlerta.configure(image=ventana.alerta)
                            labelAlerta.pack()
                            etiqueta2 = tk.Label(emergente, text =("Divergencia en la
estimación de \n" + "la onda " + str(divergencias[0])))
                            etiqueta2.pack(anchor = CENTER)
                            etiqueta2.configure( font = (fuente, 10, "bold"), bg
='white')
                            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command
= emergente.destroy,  font = (fuente, 10, "bold"))
                            btn_aceptar.pack(pady = 10)
                        else:
                            temp = ""
                            for i in range(len(divergencias)):
                                if i == len(divergencias)-1:
                                    temp = temp + str(divergencias[i]) + "."
                                else:
                                    temp = temp + str(divergencias[i]) + ","
                            emergente = tk.Toplevel(ventana)
                            emergente.title('Advertencia')
                            emergente.geometry(str('240x180+' +
str(ventana.winfo_screenwidth()//2 - 240 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                            emergente.configure(background = 'white')
                            emergente.attributes("-toolwindow", True)
                            emergente.resizable(False,False)
                            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                            etiqueta = tk.Label(emergente, text = "Advertencia!!!")
                            etiqueta.pack( anchor = CENTER)
                            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg
='white')
                            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta3.png")))
                            labelAlerta.configure(image=ventana.alerta)
                            labelAlerta.pack()
                            etiqueta2 = tk.Label(emergente, text =("Divergencia en la
estimación de \n" + "las ondas " + temp))
                            etiqueta2.pack(anchor = CENTER)
                            etiqueta2.configure( font = (fuente, 10, "bold"), bg
='white')
                            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command
= emergente.destroy,  font = (fuente, 10, "bold"))
                            btn_aceptar.pack(pady = 10)

def diagnostico():
    for widgets in frameMayor.winfo_children():
        widgets.destroy()
```

```
    global diagnosticoFrame
    global VdataSegmento, VdatoTiempo, VdatoAmplitud, FrecuenciaCardiaca
    global Diagnostico
    diagnosticoFrame = Frame(frameMayor, bg = "#F2F2F2")
    diagnosticoFrame.pack(fill = "both", expand = 1)

    frameHeader = Frame(diagnosticoFrame, height = 50, bg = "#F2F2F2")
    lbl25 = Label(frameHeader, text = "DIAGNOSTICO", fg = "black", font = (fuente, 20,
"bold"), bg = "#F2F2F2")
    lbl25.pack(side = 'top', fill = 'both', expand = 1, pady = 30)
    frameHeader.pack(side = 'top', fill = 'x')

    frameEnfermedades = Frame(diagnosticoFrame, height = 100, bg = "#F2F2F2")
    frameEnfermedades.pack(fill = 'both', expand = '1')
    listaFrame = Frame(frameEnfermedades, bg = "#F2F2F2", highlightbackground = 'black',
highlightthickness = 2)

    lbl9 = Label(listaFrame, text = "Posibles enfermedades: ", fg = "black", font =
(fuente, 13, "bold"), bg = "#F2F2F2")

    lbl9.pack(side = 'top', fill = 'both', pady = 15, padx = 50, anchor = "w")

    listaFrame.pack(side = 'left', fill = 'both', padx = 5)

    frameSoporte = Frame(frameEnfermedades, bg = "#F2F2F2")
    lbl26 = Label(frameSoporte, text = "Soporte de diagnóstico", fg = "black", font =
(fuente, 15, "bold"), bg = "#F2F2F2")
    lbl26.pack(side = 'top', fill = 'both', pady = 12, anchor = 'w')
    lbl27 = Label(frameSoporte, text = "", fg = "black", font = (fuente, 12, "bold"), bg
= "#F2F2F2")
    lbl27.pack(fill = 'both', pady = 5, anchor = 'w')
    frameSoporte.pack(side = 'right', fill = 'both', expand = '1')

    ventana.a = tk.BooleanVar()
    ventana.b = tk.BooleanVar()
    ventana.c = tk.BooleanVar()
    ventana.d = tk.BooleanVar()
    ventana.m = tk.BooleanVar()
    ventana.f = tk.BooleanVar()
    ventana.e = tk.BooleanVar()
    ventana.g = tk.BooleanVar()
    ventana.h = tk.BooleanVar()
    ventana.i = tk.BooleanVar()
    ventana.k = tk.BooleanVar()
    ventana.l = tk.BooleanVar()
    ventana.j = tk.BooleanVar()

    if (VdatoTiempo[8]-VdatoTiempo[6] < 0.1 and VdatoAmplitud[1] < 0.00005 and
VdataSegmento[2] > 0.12):
            ventana.a.set(True)
            lbl27['text'] = lbl27['text'] + "\n  Hiperpotacemia: \n"
```

```
            if VdatoTiempo[8]-VdatoTiempo[6] < 0.1:
                lbl27['text'] = lbl27['text'] + "- La duración de la onda T es inferior a
0.1 Seg. \n"
            if  VdatoAmplitud[1] < 0.00005:
                lbl27['text'] = lbl27['text'] + "- La amplitud de la onda P es inferior a
0.05 mV. \n"
            if VdataSegmento[2] > 0.12:
                lbl27['text'] = lbl27['text'] + "- La duración del complejo QRS es
superior a 0.12 Seg. \n"
    else:
            ventana.a.set(False)

    if (VdatoAmplitud[1] > 0.00025):
        ventana.b.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipertrofia auricular derecha: \n"
        lbl27['text'] = lbl27['text'] + "- La amplitud de la onda P es inferior a 0.25
mV. \n"
    else:
        ventana.b.set(False)



    if (VdatoTiempo[2]-VdatoTiempo[0] > 0.1):
        ventana.c.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Dilatacion auricular: \n"
        lbl27['text'] = lbl27['text'] + "- La duración de la onda P es superior a 0.1
Seg. \n"
    else:
        ventana.c.set(False)


    if (VdataSegmento[1] > 0.2):
        ventana.d.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bloqueo auroventricular: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo PR es superior a 0.2
Seg. \n"
    else:
        ventana.d.set(False)

    if (VdataSegmento[2] > 0.12):
        ventana.m.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bloqueo de rama: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QRS es superior a
0.12 Seg. \n"
    else:
        ventana.m.set(False)

    if (VdataSegmento[3] > 0.42 and VdatoAmplitud[4] < 0.002):
        ventana.f.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Miocarditis: \n"
        if VdataSegmento[3] > 0.42:
```

```
            lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es superior a
0.42 Seg. \n"
        if VdatoAmplitud[4] < 0.002:
            lbl27['text'] = lbl27['text'] + "- La amplitud de la onda R es inferior a 0.2
mV. \n"
    else:
        ventana.f.set(False)

    if (VdataSegmento[3] < 0.32):
        ventana.e.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipercalcemia: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es inferior a 0.32
Seg. \n"
    else:
        ventana.e.set(False)

    if (VdataSegmento[3] > 0.42):
        ventana.g.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Sindrome del QT Prolongado: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es superior a 0.42
Seg. \n"
    else:
        ventana.g.set(False)

    if (VdataSegmento[4] > 0.15 and VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003)):
        ventana.h.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Pericarditis: \n"
        if VdataSegmento[4] > 0.15:
            lbl27['text'] = lbl27['text'] + "- La duración del complejo ST es superior a
0.15 Seg. \n"
        if  VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003):
            lbl27['text'] = lbl27['text'] + "- la amplitud de la onda T es superior a la
Amplitud de R / 3mV. \n"
    else:
        ventana.h.set(False)

    if (VdatoTiempo[8]-VdatoTiempo[6] > 0.12):
        ventana.j.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipertrófia Auricular Izquierda: \n"
        lbl27['text'] = lbl27['text'] + "- La duración de la onda T es superior a 0.12
Seg. \n"
    else:
        ventana.j.set(False)


    if (VdatoTiempo[8]-VdatoTiempo[6] > 0.12 and VdatoAmplitud[7] >
(VdatoAmplitud[4]/0.003)):
        ventana.i.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Isquemia: \n"
        if VdatoTiempo[8]-VdatoTiempo[6] > 0.12:
```

```
            lbl27['text'] = lbl27['text'] + "- La duración de la onda T es superior a
0.12 Seg. \n"
        if VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003):
            lbl27['text'] = lbl27['text'] + "- la amplitud de la onda T es superior a la
Amplitud de R / 3mV. \n"
    else:
        ventana.i.set(False)

    if (FrecuenciaCardiaca > 100):
        ventana.k.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Taquicardia Sinusual: \n"
        lbl27['text'] = lbl27['text'] + "- La frecuencia cardiaca es superior a 100 LPM.
\n"
    else:
        ventana.k.set(False)

    if (FrecuenciaCardiaca < 60):
        ventana.l.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bradicardia Sinusual: \n"
        lbl27['text'] = lbl27['text'] + "- La frecuencia cardiaca es inferior a 60 LPM.
\n"
    else:
        ventana.l.set(False)

    lblDisclaimer = Label(frameSoporte, text = "Los resultados de este análisis se basan
en la derivación I y no son concluyentes, \n por lo que se recomienda consultar con un
médico.", bg = "#F2F2F2", fg = color_letra_texto, font = (fuente, 10, "bold"))
    lblDisclaimer.pack(side = 'bottom', pady = 10)

    lbl10 = Checkbutton(listaFrame, state = "disabled", variable = ventana.a , text =
"Hiperpotacemia", fg = color_letra_texto, font = (fuente, 10))
    lbl11 = Checkbutton(listaFrame, state = "disabled", variable = ventana.b , text =
"Hipertrofia auricular derecha", fg = color_letra_texto, font = (fuente, 10))
    lbl12 = Checkbutton(listaFrame, state = "disabled", variable = ventana.c , text =
"Dilatacion auricular", fg = color_letra_texto, font = (fuente, 10))
    lbl15 = Checkbutton(listaFrame, state = "disabled", variable = ventana.d , text =
"Bloqueo Auroventricular", fg = color_letra_texto, font = (fuente, 10))
    lbl16 = Checkbutton(listaFrame, state = "disabled", variable = ventana.e , text =
"Hipercalcemia", fg = color_letra_texto, font = (fuente, 10))
    lbl17 = Checkbutton(listaFrame, state = "disabled", variable = ventana.f , text =
"Miocarditis", fg = color_letra_texto, font = (fuente, 10))
    lbl18 = Checkbutton(listaFrame, state = "disabled", variable = ventana.g , text =
"Sindrome del QT Prolongado", fg = color_letra_texto, font = (fuente, 10))
    lbl19 = Checkbutton(listaFrame, state = "disabled", variable = ventana.h , text =
"Pericarditis", fg = color_letra_texto, font = (fuente, 10))
    lbl20 = Checkbutton(listaFrame, state = "disabled", variable = ventana.i , text =
"Isquemia", fg = color_letra_texto, font = (fuente, 10))
    lbl21 = Checkbutton(listaFrame, state = "disabled", variable = ventana.j , text =
"Hipertrofia auricular izquierda", fg = color_letra_texto, font = (fuente, 10))
    lbl22 = Checkbutton(listaFrame, state = "disabled", variable = ventana.k , text =
"Taquicardia", fg = color_letra_texto, font = (fuente, 10))
```

```
    lbl23 = Checkbutton(listaFrame, state = "disabled", variable = ventana.l , text =
"Bradicardia", fg = color_letra_texto, font = (fuente, 10))
    lbl24 = Checkbutton(listaFrame, state = "disabled", variable = ventana.m , text =
"Hipertrofia Ventricular o Bloqueo de Rama", fg = color_letra_texto, font = (fuente, 10))
    lbl10.pack(anchor = "w")
    lbl11.pack(anchor = "w")
    lbl12.pack(anchor = "w")
    lbl15.pack(anchor = "w")
    lbl16.pack(anchor = "w")
    lbl17.pack(anchor = "w")
    lbl18.pack(anchor = "w")
    lbl19.pack(anchor = "w")
    lbl20.pack(anchor = "w")
    lbl21.pack(anchor = "w")
    lbl22.pack(anchor = "w")
    lbl23.pack(anchor = "w")
    lbl24.pack(anchor = "w")

    frameFooter = Frame(diagnosticoFrame, height = 50, bg = "#F2F2F2")
    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 15, "bold"), command
= resultados)
    btnVolver.pack(side = 'left', padx = (100, 0), pady = 25)
    btnSalir = Button(frameFooter, text = "Salir", font = (fuente, 15, "bold"), command =
ventana.destroy)
    btnSalir.pack(side = 'right', padx = (0, 100), pady = 25)
    btnHome = Button(frameFooter, text = "Inicio", font = (fuente, 15, "bold"), command =
inicio)
    btnHome.pack(pady = 25)
    frameFooter.pack(side = 'bottom', fill = 'x')

inicio()
ventana.mainloop()

############################################################
###              SECCION DE PRUEBAS SIN UI               ###
############################################################


# inicio = time.time()
# Preparacion()
# PDF_to_jpeg(ruta_PDF)
# crop_image()
# Concat_image()
# Limpiar_imagen()
# Lines_Hough(cv2.imread('./Code/Data/Output/img-ECG.jpeg'))
# Vectorizacion_Señal()
# ObtencionPico()
# Características()
# ObtencionParametros()
# Diagnosticar()
# fin = time.time()
```

```
# print(str(fin-inicio))
```

```
global ruta_PDF
global emergente
emergente = None


############################################################
###                 SECCION DE LIBRERIAS                 ###
############################################################

# Aqui se realiza la importacion de las librerias

#   Back

import os                                   # libreria para el manejo del sistema
import shutil                               # libreria para el control de archivos
import cv2                                  # libreria para el manejo de las imágenes
from pdf2image import convert_from_path     # libreria de conversion de archivos
import matplotlib.pyplot as plt             # libreria para visualizar imagen
import numpy as np                          # Libreria para manipular vectores de la
señal
import time                                 # Libreria para control de tiempo
import sys                                  # Libreria para puntos de ruptura
from scipy.signal import savgol_filter      # Libreria para suavizar la señal muestreada

#   Front
from tkinter import *
import tkinter as tk
from tkinter import messagebox, ttk
from tkinter import filedialog
from PIL import ImageTk, Image
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import numpy as np
import pandas as pd


############################################################
###         PREPARACION DE LAS RUTAS DE DATOS            ###
############################################################


# Aqui se ajustan las variables de rutas para el proyecto
# además de crear y validar los directorios necesarios

def Preparacion():
    if os.path.exists("./Code/Data/Output"):
        shutil.rmtree("./Code/Data/Output")
        os.mkdir("./Code/Data/Output")
    else:
        os.mkdir("./Code/Data/Output")
```

```
############################################################
####         CUERPO DE FUNCIONES DEL CÓDIGO             ###
############################################################

# Función de conversión de PDF a JPEG
def PDF_to_jpeg(ruta_PDF):
    global pages
    poppler_path = os.path.abspath("./Code/lib/poppler-22.04.0/Library/bin")            #
Configuracion de rutas de los archivos
    pdf_path = os.path.abspath(ruta_PDF)                                                #
Configuracion de rutas de los archivos
    saving_folder = os.path.abspath("./Code/Data/Output")
 # Configuracion de rutas de los archivos
    pages = convert_from_path(pdf_path = pdf_path, poppler_path = poppler_path )    #
Obtención de la cantidad de páginas del documento
    i = 1
    for page in pages:
        img_name = f"img-{i}.jpeg"
        page.save(os.path.join(saving_folder,img_name),"JPEG")                          #
Conversion de las páginas a PDF
        i += 1
# Función de recorte de las imágenes
def crop_image():
    global emergente
    for i in range (2,6,1):
        image="img-"+ str(i) + ".jpeg"
        rimage= os.path.abspath("./Code/Data/Output/"+image)
    # Construción de la ruta de imagen que se desea leer
        try:
            data = cv2.imread(rimage)                                                   #
Lectura de la imagen original
            data = data [204:2091,63:1638]                                              #
Recorte de la imágen
            cv2.imwrite('./Code/Data/Output/img-'+ str(i)+'-crop.jpeg',data)
    # Almacenaje de las immágenes recortadas
        except (TypeError):
            pass




# Función para conectar las imágenes recortadas
def Concat_image():
    im2 = cv2.imread("./Code/Data/Output/img-2-crop.jpeg")
    im3 = cv2.imread("./Code/Data/Output/img-3-crop.jpeg")
    im4 = cv2.imread("./Code/Data/Output/img-4-crop.jpeg")
    im5 = cv2.imread("./Code/Data/Output/img-5-crop.jpeg")
```

```
    imga = cv2.hconcat([im2, im3])
    imgb = cv2.hconcat([im4, im5])
    img = cv2.hconcat([imga, imgb])
    img = img[:,97:6002]
    cv2.imwrite('./Code/Data/Output/img-ECG.jpeg',img)

# Función para limpiar la imágen
def Limpiar_imagen():
    img = cv2.imread("./Code/Data/Output/img-ECG.jpeg")
    ###########################################################################
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _,img = cv2.threshold(img,100,255,cv2.THRESH_BINARY)
    ###########################################################################
    mask = cv2.imread("./Code/lib/mask.png",0)
    for f in range(img.shape[0]):
        for c in range(img.shape[1]):
            if (int(mask[f,c]) == 0):
                img[f,c] = 255

    ###########################################################################
    cv2.imwrite('./Code/Data/Output/img-ECG.jpeg',img)

#Detecto los ejes y secciono la imágen
def Lines_Hough(img):
    global Ejesy
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150, apertureSize = 3)
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=500, maxLineGap=50)
    Ejesx=[]
    Ejesy=[]
    for line in lines:
        x1, y1, x2, y2 = line[0]
        if (x1 != x2):
            Ejesx.append(y1)
            cv2.line(img, (x1,y1), (x2,y2), (0,255,0), 1, cv2.LINE_AA)
        else:
            Ejesy.append(x1)
            cv2.line(img, (x1,y1), (x2,y2), (0,255,0), 1, cv2.LINE_AA)
    #Determino el valor de los ejes de cada derivación
    temp=sorted(Ejesx)
    Ejesx=[]
    aux=[]
    x=0
    i=0
    while (i+x)<len(temp)-1:
        while (int(temp[x+i])-int(temp[i]))<=10 and (i+x)<len(temp)-1:
            aux.append(int(temp[x+i]))
            x=x+1
        i=x+i
        x=0
        Ejesx.append(round((sum(aux)/len(aux))))
```

```
        aux=[]
    temp=sorted(Ejesy)
    Ejesy=[]
    aux=[]
    x=0
    i=0
    while (i+x)<len(temp)-1:
        while (int(temp[x+i])-int(temp[i]))<=10 and (i+x)<len(temp)-1:
            aux.append(int(temp[x+i]))
            x=x+1
        i=x+i
        x=0
        Ejesy.append(round((sum(aux)/len(aux))))
        aux=[]
    img =  cv2.imread("./Code/Data/Output/img-ECG.jpeg")
    cv2.imwrite('./Code/Data/Output/I.jpeg',img[Ejesx[0]-150:Ejesx[0]+150,:])
    cv2.imwrite('./Code/Data/Output/II.jpeg',img[Ejesx[1]-150:Ejesx[1]+150,:])
    cv2.imwrite('./Code/Data/Output/III.jpeg',img[Ejesx[2]-150:Ejesx[2]+150,:])
    cv2.imwrite('./Code/Data/Output/aVR.jpeg',img[Ejesx[3]-150:Ejesx[3]+150,:])
    cv2.imwrite('./Code/Data/Output/aVL.jpeg',img[Ejesx[4]-150:Ejesx[4]+150,:])
    cv2.imwrite('./Code/Data/Output/aVF.jpeg',img[Ejesx[5]-150:Ejesx[5]+150,:])


############################################################
####          CONSIDERACIONES DE RESOLUCIÓN            ####
##################                    ######################
# TIEMPO                                                   #
#                                                          #
# ------------  1 Cuadro -> 5 mm -> 0.2 Seg --------------#
#                                                          #
# VOLTAJE                                                  #
# ------------  1 Cuadro -> 5 mm -> 0.5 mV ---------------#
#                                                          #
# DIMENSIONES                                              #
#                                                          #
# ------------  1 Cuadro -> 39px X 39px  ---------------- #
# ---------  1 px Vertical -> 12.82051282 uV ------------ #
# --------  1 px Horizontal -> 5.128205128 mS  ---------- #
############################################################


# Vectorización de la Señal
def Vectorizacion_Señal():
    global aVF, aVL, aVR, I, II, III, Vtiempo
    global fallaCorte
    fallaCorte = False
    Signals = ['aVF','aVL','aVR','I','II','III']
    for signalstr in Signals:
        strtemp = './Code/Data/Output/' + signalstr + '.jpeg'
        img = cv2.imread(strtemp)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
        _,img = cv2.threshold(gray,10,255,cv2.THRESH_BINARY)
        aux = np.asarray(img)
        aux1 = np.sum(aux, axis=0)
        posceros = np.where(aux1==76245)
        aux2 = aux[:,posceros[0][0]]
        Ejex = np.where(aux2==0)
        Ejesx = Ejex[0][0]
        aux2 = aux[0,:]
        posceros = np.where(aux2==0)
        Ejesy = posceros[0][:]
        if len(Ejesy) == 30 :
            for i in range(len(aux1)):
                if aux1[i] < 76245:
                    img[Ejesx, i] = 255
            for i in range(img.shape[0]):
                for ejey in Ejesy:
                    temp = (((img[i,(ejey+1)]==255) or (img[i,(ejey-1)]==255)) and
i!=Ejesx)
                    if temp:
                        img[i,ejey]=255
            signal=[]
            for i in range(img.shape[1]):
                temp=aux[0:img.shape[0], i]
                ceros =  np.where(temp==0)
                ceros = np.asanyarray(ceros[0])
                if len(ceros) == 0:
                    val = -10000
                    signal.append(val)
                else:
                    temp = np.where(abs(ceros - Ejesx)==np.amax(abs(ceros - Ejesx)))
                    if len(temp[0][:])==1:
                        signal.append(ceros[temp][0])
                    else:
                        temp = ceros[temp]
                        try:
                            postemp=np.where(abs(temp-signal[-1])==np.amax(abs(temp-
signal[-1])))
                            signal.append(temp[postemp][0])
                        except IndexError:
                            fallaCorte = True
            signal = np.array(signal)
            huecos = np.where(signal==-10000)
            for hueco in huecos:
                signal[hueco] = ((signal[hueco + 1]) + (signal[hueco - 1]))/2
            signal =(signal - Ejesx)*-0.00001282051281999999
            signal = savgol_filter(signal,7,2)
            if signalstr == 'aVF':
                aVF = signal
            elif signalstr == 'aVR':
                aVR = signal
            elif signalstr == 'aVL':
```

```
                aVL = signal
            elif signalstr == 'I':
                I = signal
            elif signalstr == 'II':
                II = signal
            else:
                III = signal
            Vtiempo = (np.arange(0,int(signal.shape[0]),1))*0.005128205128000001
        else:
            fallaCorte = True

#Caracterización de la señal
def ObtencionPico():
    global aVF, aVL, aVR, I, II, III, Vtiempo
    global SegmentoSeñal, SegmentoTiempo
    global Maximos, Periodo
    global FrecuenciaCardiaca

    if (len(np.where(I>0.0015)[0])>0):
        print('Descarte la señal')
        print('La señal presenta mucho ruido y picos demasiado altos')
        sys.exit()
    else:
        Maximos = np.where(I>0.0005)[0]
        if len(Maximos) == 0:
            print('Desarte la señal, la onda R se encuentra bajo el estandar de 0.5 mV')
            sys.exit()
        else:
            maximos=np.asarray(Maximos[:])
            if len(maximos) < 150:
                print('Error en la detección')
                print('La onda R no se detecta como marcador')
                sys.exit()
            else:
                temp = []
                aux = []
                Maximos = []
                for i in range(len(maximos)):
                    if i >=1 :
                        ja = maximos[i] - maximos[i-1]
                        if (maximos[i] - maximos[i-1]) <= 20 :
                            temp.append(maximos[i])
                        else:
                            aux = np.asarray(I[temp][:])


                            indice = np.where(aux == indice)[:][0][0]
                            Maximos.append(temp[indice])
                            aux = []
                            temp = []
                    else:
```

```
                        temp.append(maximos[i])

                temp = []
                for i in range(len(Maximos)):
                    if i >= 1:
                        temp.append(Maximos[i] - Maximos[i-1])
                Periodo = round(np.mean(temp))
                FrecuenciaCardiaca = len(Maximos)*2
                Picos = I[Maximos]-0.0005
                indice = [np.where(Picos==np.min(Picos))][:][0][0][0]
                indice = Maximos[indice]
                Pico = I[indice]
                Time = Vtiempo[indice]

                SegmentoSeñal = I[indice-round(Periodo/2):indice+round(Periodo/2)]
                SegmentoTiempo = Vtiempo[indice-round(Periodo/2):indice+round(Periodo/2)]

#Obtener características del pico
def Características():
    global SegmentoSeñal, SegmentoTiempo, Periodo
    global VdatoTiempo, VdatoAmplitud, VdatoString
    global divergencias

    divergencias = []
    SegmentoSeñal = savgol_filter(SegmentoSeñal,10,5)  # Filtro la señal

    d1x = np.diff(SegmentoSeñal)        # Extraigo las derivadas
    d2x = np.diff(d1x)


    ##############################################################################
    ####            CALCULO LAS ONDAS POR OBSERVACIONES EN LAS DERIVADAS          ####
    ##############################################################################


    picos_indices = np.where((np.diff(np.sign(d1x)) < 0) & (d1x[:-1] > 0))[0] + 1

    CaD = SegmentoTiempo[picos_indices]
    picos_indices = np.where((np.diff(np.sign(d1x)) > 0) & (d1x[:-1] < 0))[0] + 1
    DaC = SegmentoTiempo[picos_indices]
    picos_indices = np.where(np.diff(np.sign(d1x)))[0] + 1
    Picos = SegmentoTiempo[picos_indices]

    TiempoR = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal))])][:][0]

    TiempoQa = DaC[np.where(DaC == SegmentoTiempo[np.where(SegmentoSeñal ==
np.min(SegmentoSeñal[0:np.where(SegmentoTiempo == TiempoR)[:][0][0]]))])][:][0]
    TiempoQb = Picos[np.where(Picos == TiempoR)[:][0][0]-1]
    if TiempoQa == TiempoQb:
        TiempoQ = TiempoQa
```

```
    else:
        divergencias.append('Q')
        if ((TiempoR - TiempoQa > 0.13) and (TiempoR - TiempoQb < 0.13)) :
            TiempoQ = TiempoQb
        elif ((TiempoR - TiempoQa < 0.13) and (TiempoR - TiempoQb > 0.13)):
            TiempoQ = TiempoQa
        elif SegmentoSeñal[np.where(SegmentoTiempo ==TiempoQa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoQb)[:][0][0]]:
            TiempoQ = TiempoQb
        else:
            TiempoQ = TiempoQa

    TiempoSa = DaC[np.where(DaC == SegmentoTiempo[np.where(SegmentoSeñal ==
np.min(SegmentoSeñal[np.where(SegmentoTiempo == TiempoR)[:][0][0:]))])][:][0]
    TiempoSb = Picos[np.where(Picos == TiempoR)[:][0][0]+1]
    if TiempoSa == TiempoSb:
        TiempoS = TiempoSa
    else:
        divergencias.append('S')
        if ((TiempoSa - TiempoR > 0.13) and (TiempoSb - TiempoR < 0.13)) :
            TiempoS = TiempoSb
        elif ((TiempoSa - TiempoR < 0.13) and (TiempoSb - TiempoR > 0.13)) :
            TiempoS = TiempoSa
        elif SegmentoSeñal[np.where(SegmentoTiempo ==TiempoSa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoSb)[:][0][0]]:
            TiempoS = TiempoSb
        else:
            TiempoS = TiempoSa

    TiempoPa = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal[:np.where(SegmentoTiempo == TiempoQ)[:][0][0]]))])][:][0]
    TiempoPb = Picos[np.where(Picos == TiempoQ)[:][0][0]-1]
    if TiempoPa == TiempoPb:
        TiempoP = TiempoPa
    else:
        divergencias.append('P')
        if SegmentoSeñal[np.where(SegmentoTiempo ==TiempoPa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoPb)[:][0][0]]:
            TiempoP = TiempoPa
        else:
            TiempoP = TiempoPb

    TiempoTa = CaD[np.where(CaD == SegmentoTiempo[np.where(SegmentoSeñal ==
np.max(SegmentoSeñal[np.where(SegmentoTiempo == TiempoS)[:][0][0:]]))])][:][0]
    TiempoTb = Picos[np.where(Picos == TiempoS)[:][0][0]+1]
    if TiempoTa == TiempoTb:
        TiempoT = TiempoTa
    else:
        divergencias.append('T')
        if SegmentoSeñal[np.where(SegmentoTiempo ==TiempoTa)[:][0][0]] >=
SegmentoSeñal[np.where(SegmentoTiempo ==TiempoTb)[:][0][0]]:
```

```
            TiempoT = TiempoTa
        else:
            TiempoT = TiempoTb

    xD = d2x[np.where(SegmentoTiempo == Picos[np.where(Picos == TiempoP)[:][0][0]-
1])[:][0][0] : np.where(SegmentoTiempo == TiempoP)[:][0][0]]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == Picos[np.where(Picos ==
TiempoP)[:][0][0]-1])[:][0][0] : np.where(SegmentoTiempo == TiempoP)[:][0][0]]
    TiempoPI =  aux[np.where(xD == np.max(xD))[:][0][0]]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == TiempoP)[:][0][0] :
np.where(SegmentoTiempo == TiempoQ)[:][0][0]]
    TiempoPF = aux[int(len(aux)/2)]

    aux = SegmentoTiempo[np.where(SegmentoTiempo == Picos[np.where(Picos ==
TiempoT)[:][0][0]-1])[:][0][0] : np.where(SegmentoTiempo == TiempoT)[:][0][0]]
    TiempoTI = aux[int(len(aux)/2)]
    aux = SegmentoTiempo[np.where(SegmentoTiempo == TiempoT)[:][0][0] :
np.where(SegmentoTiempo == Picos[np.where(Picos == TiempoT)[:][0][0]+1])[:][0][0] ]
    TiempoTF = aux[int(len(aux)/2)]

    R = SegmentoSeñal[np.where(SegmentoTiempo == TiempoR)][:][0]
    S = SegmentoSeñal[np.where(SegmentoTiempo == TiempoS)][:][0]
    T = SegmentoSeñal[np.where(SegmentoTiempo == TiempoT)][:][0]
    Q = SegmentoSeñal[np.where(SegmentoTiempo == TiempoQ)][:][0]
    P = SegmentoSeñal[np.where(SegmentoTiempo == TiempoP)][:][0]
    PI = SegmentoSeñal[np.where(SegmentoTiempo == TiempoPI)][:][0]
    PF = SegmentoSeñal[np.where(SegmentoTiempo == TiempoPF)][:][0]
    TI = SegmentoSeñal[np.where(SegmentoTiempo == TiempoTI)][:][0]
    TF = SegmentoSeñal[np.where(SegmentoTiempo == TiempoTF)][:][0]

    VdatoTiempo = [TiempoPI, TiempoP, TiempoPF, TiempoQ, TiempoR, TiempoS, TiempoTI,
TiempoT, TiempoTF]
    VdatoAmplitud = [PI, P, PF, Q, R, S, TI, T, TF]
    VdatoString = ['PI', 'P', 'PR', 'Q', 'R', 'S', 'TI', 'T', 'TF']

# Obtencion de los segmentos y datos específicos
def ObtencionParametros():
    global VdatoAmplitud, VdatoTiempo, VdatoString, Periodo
    global VdataSegmento, VdatoSegmento

    SegmentoRR = Periodo*0.005128205128000001
    SegmentoPR = VdatoTiempo[3] - VdatoTiempo[0]
    SegmentoQRS = VdatoTiempo[5] - VdatoTiempo[3]
    SegmentoQT = ((VdatoTiempo[8] - VdatoTiempo[3])/np.sqrt(SegmentoRR))
    SegmentoST = VdatoTiempo[6] - VdatoTiempo[5]

    VdatoSegmento = ['RR', 'PR', 'QRS', 'QT', 'ST']
    VdataSegmento = [SegmentoRR, SegmentoPR, SegmentoQRS, SegmentoQT, SegmentoST]


############################################################
```

```
###        DESARROLLO DE LA INTERFAZ DE USUARIO        ###
#############################################################
# Parametros de personalizacion
fondo = "#FFF"
fuente = "PT Sans"
ancho = 900
alto = 700
color_letra_titulo = "#8F141B"
color_letra_texto = "#000"

# Declaracion de la ventana
ventana = Tk()

#Configuracion para pantalla centrada y dimensiones de la ventana
ancho_ventana = ventana.winfo_screenwidth() // 2 - ancho // 2
alto_ventana = ventana.winfo_screenheight() // 2 - alto // 2
posicion = str(ancho) + "x" + str(alto) + "+" + str(ancho_ventana) + "+" +
str(alto_ventana)
ventana.geometry(posicion)

#Personalizacion de la ventana
ventana.resizable(0,0)
       # No puede maximizarse
icono = PhotoImage(file = "./Code/lib/usco.png")
                  # Icono de la ventana
ventana.iconphoto(True, icono)
ventana.title("Interfaz grafica de digitalización y detección de enfermedades cardíacas")
      # Titulo de la ventana
ventana.config(bg = fondo)

frameMayor = Frame(ventana, bg = fondo)
frameMayor.pack(fill = 'both', expand = True)

imagenUsco = ImageTk.PhotoImage(Image.open("./Code/lib/universidad-surcolombiana.png"))
archivo = None

def inicio():
    Preparacion()
    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    global inicioFrame
    inicioFrame = Frame(frameMayor, bg = "#F2F2F2")
    inicioFrame.pack(fill = 'both', expand = 1)

    frameHeader = Frame(inicioFrame, bg = "#F2F2F2")
    frameMid = Frame(inicioFrame, bg = "#F2F2F2")
    frameFooter = Frame(inicioFrame, bg = "#F2F2F2")

    # Declaracion de etiquetas
```

```
    label1 = Label(frameHeader, text = "Interfaz de digitalización y detección de
enfermedades cardíacas", fg = color_letra_titulo, bg = "#F2F2F2", font = (fuente, 20,
"bold"))
    label2 = Label(frameMid, text = "Byron Hernando Galindo Suárez - 20171155352", fg =
"black", bg = "#F2F2F2", font = (fuente, 17, "bold"))
    label3 = Label(frameMid, text = "Juan Esteban Narváez Carvajal - 20171159625", fg =
"black", bg = "#F2F2F2", font = (fuente, 17, "bold"))
    label4 = Label(frameMid, image = imagenUsco, bg ="#F2F2F2")

    # Declaracion de botones
    btnContinuar = Button(frameFooter, text = "Continuar", font = (fuente, 20, "bold"),
command = datos, width = 20, height = 20)

    # Posicionamiento de frames
    frameHeader.pack(side = "top", fill = "both")
    frameMid.pack(fill = "both", expand = True)
    frameFooter.pack(side = "bottom", fill = "both")

    # Posicionamiento de etiquetas
    label1.pack(pady = 100)
    label2.pack()
    label3.pack()
    label4.pack(pady = 75)

    # Posicionamiento de botones
    btnContinuar.pack(pady = 50)

def datos():
    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    global datosFrame
    datosFrame = Frame(frameMayor, bg = "#F2F2F2")
    datosFrame.pack(fill = "both", expand = 1)



    frameHeader = Frame(datosFrame, bg = "#F2F2F2")
    frameMid = Frame(datosFrame, bg = "#F2F2F2")
    frameFooter = Frame(datosFrame, bg = "#F2F2F2")

    # Posicionamiento de frames
    frameHeader.pack(side = "top", fill = "both")
    frameMid.pack(fill = "both", expand = True)
    frameFooter.pack(side = "bottom", fill = "both")

    lbl7 = Label(frameMid, text = "Archivo: ", fg = "black", bg = "#F2F2F2", font =
(fuente, 10, "bold"))

    def abrirArchivo():
        btnProcesar['state'] = DISABLED
        lbl7['text'] = "Archivo: "
```

```
        global archivo, pages
        try:
            archivo = filedialog.askopenfile(title = "Abrir", initialdir =
os.path.abspath(os.getcwd()), filetypes = (("Formato PDF", "*.pdf"),))
            if archivo.name != None :
                PDF_to_jpeg(archivo.name)
                lbl7['text'] = lbl7['text'] + str(archivo.name)
                ventana.imgtk = ImageTk.PhotoImage((Image.open("./Code/Data/Output/img-
1.jpeg")).resize((250,300)))
                labelvista.configure(image=ventana.imgtk)
                labelvista.pack()
                if len(pages)==5:
                    btnProcesar['state'] = NORMAL
                else:
                    emergente = tk.Toplevel(ventana)
                    emergente.title('File Error!!!')
                    emergente.geometry(str('290x180+' +
str(ventana.winfo_screenwidth()//2 - 290 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                    emergente.configure(background = 'white')
                    emergente.attributes("-toolwindow", True)
                    emergente.resizable(False,False)
                    emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                    etiqueta = tk.Label(emergente, text = "Error!!!")
                    etiqueta.pack( anchor = CENTER)
                    etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                    labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                    ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta.png")))
                    labelAlerta.configure(image=ventana.alerta)
                    labelAlerta.pack()
                    etiqueta2 = tk.Label(emergente, text = "El archivo seleccionado no
cumple con las \ncaracterísticas de un ECG de Kardia")
                    etiqueta2.pack(anchor = CENTER)
                    etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
                    btn_aceptar = tk.Button(emergente, text = "Aceptar",  command =
emergente.destroy,  font = (fuente, 10, "bold"))
                    btn_aceptar.pack(pady = 10)
                    btnProcesar['state'] = DISABLED
        except AttributeError:
            pass

    lbl5 = Label(frameHeader, text = "Por favor, cargue el archivo en formato PDF\n del
electrocardiograma entregado por Kardia 6L", fg = "black", bg = "#F2F2F2", font =
(fuente, 12))

    btnCargar = Button(frameHeader, text = "Cargar archivo", font = (fuente, 15, "bold"),
command = abrirArchivo)
    lbl6 = Label(frameMid, text = "Previsualizacion del pdf cargado", fg = "black", bg =
"#F2F2F2", font = (fuente, 15, "bold"))
```

```
    frameVisualizacion = Frame(frameMid, bg = "white", width = 250, height = 300,
highlightbackground = "black", highlightthickness = 2)
    labelvista = Label(frameVisualizacion, bg = "#F2F2F2", image = None)
    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 20, "bold"), command
= inicio)
    btnProcesar = Button(frameFooter, text = "Procesar", font = (fuente, 20, "bold"),
command = resultados, state = DISABLED)

    lbl5.pack(side = "left", padx = 75, pady = 75)
    btnCargar.pack(side = "right", padx = 75, pady = 75)
    lbl6.pack()
    frameVisualizacion.pack(pady = 10)
    lbl7.pack()
    btnVolver.pack(side = "left", padx = 75, pady = 25)
    btnProcesar.pack(side = "right", padx = 75, pady = 25)

def resultados():
    global fallaCorte, ancho_ventana, alto_ventana, divergencias
    global resultadosFrame, SignalPlot, Clicks, Capsulasegmento, Capsulapico
    global I, II, III, aVR, aVL, aVF, Vtiempo, SegmentoTiempo, SegmentoSeñal
    global Inicio_pico_Seg, Fin_pico_Seg
    global Maximos, Periodo
    global FrecuenciaCardiaca

    for widgets in frameMayor.winfo_children():
        widgets.destroy()
    crop_image()
    Concat_image()
    Limpiar_imagen()
    Lines_Hough(cv2.imread('./Code/Data/Output/img-ECG.jpeg'))
    Vectorizacion_Señal()
    if fallaCorte == True:
        emergente = tk.Toplevel(ventana)
        emergente.title('Process Error!!!')
        emergente.geometry(str('390x250+' + str(ventana.winfo_screenwidth()//2 - 390 //
2) + "+" + str(ventana.winfo_screenheight() // 2 - 250 // 2)))
        emergente.configure(background = 'white')
        emergente.attributes("-toolwindow", True)
        emergente.resizable(False,False)
        emergente.protocol("WM_DELETE_WINDOW", lambda: None)
        etiqueta = tk.Label(emergente, text = "Error")
        etiqueta.pack( anchor = CENTER)
        etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
        labelAlerta = tk.Label(emergente, bg = fondo, image = None)
        ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
        labelAlerta.configure(image=ventana.alerta)
        labelAlerta.pack()
```

```
        etiqueta2 = tk.Label(emergente, text = "La señal detectada fue cortada durante la
digitalización, \nla señal puede tener mucho ruido o \n es una señal con muy poca
amplitud. \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
        etiqueta2.pack(anchor = CENTER)
        etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
        def alfa():
            emergente.destroy()
            datos()
        btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,  font =
(fuente, 10, "bold"))
        btn_aceptar.pack(pady = 10)
    else:
        if (len(np.where(I>0.0015)[0])>0):
            emergente = tk.Toplevel(ventana)
            emergente.title('Process Error!!!')
            emergente.geometry(str('310x210+' + str(ventana.winfo_screenwidth()//2 - 310
// 2) + "+" + str(ventana.winfo_screenheight() // 2 - 210 // 2)))
            emergente.configure(background = 'white')
            emergente.attributes("-toolwindow", True)
            emergente.resizable(False,False)
            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
            etiqueta = tk.Label(emergente, text = "Error")
            etiqueta.pack( anchor = CENTER)
            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
            labelAlerta.configure(image=ventana.alerta)
            labelAlerta.pack()
            etiqueta2 = tk.Label(emergente, text = "La señal detectada presenta demasido
ruido \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
            etiqueta2.pack(anchor = CENTER)
            etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
            def alfa():
                emergente.destroy()
                datos()
            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,  font =
(fuente, 10, "bold"))
            btn_aceptar.pack(pady = 10)

        else:
            Maximos = np.where(I>0.0005)[0]
            if len(Maximos) == 0:
                emergente = tk.Toplevel(ventana)
                emergente.title('Process Error!!!')
                emergente.geometry(str('250x210+' + str(ventana.winfo_screenwidth()//2 -
250 // 2) + "+" + str(ventana.winfo_screenheight() // 2 - 210 // 2)))
                emergente.configure(background = 'white')
                emergente.attributes("-toolwindow", True)
                emergente.resizable(False,False)
```

```
                emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                etiqueta = tk.Label(emergente, text = "Error")
                etiqueta.pack( anchor = CENTER)
                etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
                labelAlerta.configure(image=ventana.alerta)
                labelAlerta.pack()
                etiqueta2 = tk.Label(emergente, text = "En la señal detectada presenta
poca amplitud \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
                etiqueta2.pack(anchor = CENTER)
                etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
                def alfa():
                    emergente.destroy()
                    datos()
                btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,
 font = (fuente, 10, "bold"))
                btn_aceptar.pack(pady = 10)
            else:
                maximos=np.asarray(Maximos[:])
                if len(maximos) < 100:
                    emergente = tk.Toplevel(ventana)
                    emergente.title('Process Error!!!')
                    emergente.geometry(str('250x210+' +
str(ventana.winfo_screenwidth()//2 - 250 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 210 // 2)))
                    emergente.configure(background = 'white')
                    emergente.attributes("-toolwindow", True)
                    emergente.resizable(False,False)
                    emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                    etiqueta = tk.Label(emergente, text = "Error")
                    etiqueta.pack( anchor = CENTER)
                    etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg ='white')
                    labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                    ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta2.png")).resize((80,80),
resample=Image.LANCZOS))
                    labelAlerta.configure(image=ventana.alerta)
                    labelAlerta.pack()
                    etiqueta2 = tk.Label(emergente, text = "Error en la detección de la
onda R \nEl procesamiento no es válido \nReintente con un nuevo archivo!")
                    etiqueta2.pack(anchor = CENTER)
                    etiqueta2.configure( font = (fuente, 10, "bold"), bg ='white')
                    def alfa():
                        emergente.destroy()
                        datos()
                    btn_aceptar = tk.Button(emergente, text = "Aceptar",  command = alfa,
 font = (fuente, 10, "bold"))
                    btn_aceptar.pack(pady = 10)
```

```
                else:
                    temp = []
                    aux = []
                    Maximos = []
                    for i in range(len(maximos)):
                        if i >=1 :
                            ja = maximos[i] - maximos[i-1]
                            if (maximos[i] - maximos[i-1]) <= 20 :
                                temp.append(maximos[i])
                            else:
                                aux = np.asarray(I[temp][:])
                                indice = np.max(aux)
                                indice = np.where(aux == indice)[:][0][0]
                                Maximos.append(temp[indice])
                                aux = []
                                temp = []
                        else:
                            temp.append(maximos[i])

                    temp = []
                    for i in range(len(Maximos)):
                        if i >= 1:
                            temp.append(Maximos[i] - Maximos[i-1])
                    Periodo = round(np.mean(temp))
                    FrecuenciaCardiaca = len(Maximos)*2
                    Picos = I[Maximos]-0.0005
                    indice = [np.where(Picos==np.min(Picos))][:][0][0][0]
                    indice = Maximos[indice]
                    Pico = I[indice]
                    Time = Vtiempo[indice]

                    SegmentoSeñal = I[indice-round(Periodo/2):indice+round(Periodo/2)]
                    SegmentoTiempo = Vtiempo[indice-
round(Periodo/2):indice+round(Periodo/2)]

                    Características()
                    ObtencionParametros()


                    SignalPlot = I
                    Clicks = 0

                    # Vista por defecto  es con la señal I

                    Inicio_pico_Seg = Vtiempo[np.where (Vtiempo ==
SegmentoTiempo[0])][:][0]
                    Fin_pico_Seg = Vtiempo[np.where (Vtiempo == SegmentoTiempo[-
1])][:][0]

                    Datospicoplottiempo = Vtiempo[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]
```

```
                    Datospicoplotseñal = SignalPlot[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]*1000

                    Datossegmentoplottiempo = Vtiempo[(Clicks*590):((Clicks+1)*590)]
                    Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000

                    resultadosFrame = Frame(frameMayor, bg = "#F2F2F2")
                    resultadosFrame.pack(fill = "both", expand = 1)
                    frameHeader = Frame(resultadosFrame, height = 50, bg = "#F2F2F2")
                    frameParametros = Frame(resultadosFrame, height = 100, bg =
"#F2F2F2")
                    frameFooter = Frame(resultadosFrame, height = 50, bg = "#F2F2F2")
                    ecgFrame = Frame(frameParametros, height = 50, bg = "#F2F2F2")

                    def res_3s():
                        global Clicks, SignalPlot, Capsulasegmento, Vtiempo
                        if Clicks > 0:
                            Capsulasegmento.get_tk_widget().destroy()
                            Clicks = Clicks - 1
                            Datossegmentoplottiempo =
Vtiempo[(Clicks*590):((Clicks+1)*590)]
                            Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000
                            figure = plt.Figure(dpi=55)
                            Segmento = figure.add_subplot(1, 1, 1)
                            Segmento.set_ylabel('Amplitud (mV)')
                            Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                            Segmento.set_xlabel('Tiempo (s)')
                            Segmento.set_title(' Señal I')
                            Capsulasegmento = FigureCanvasTkAgg(figure, master =
ecgFrame)
                            Capsulasegmento.get_tk_widget().pack(pady = 5)
                            # aqui debe de actualizar el plot

                        # Debe de desplazar la señal hacia atras 6seg, ojo para clicks >
0

                    def sum_3s():
                        global Clicks, SignalPlot, Capsulasegmento, Vtiempo
                        if Clicks < 9:
                            Capsulasegmento.get_tk_widget().destroy()
                            Clicks = Clicks + 1
                            Datossegmentoplottiempo =
Vtiempo[(Clicks*590):((Clicks+1)*590)]
                            Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000
                            figure = plt.Figure(dpi=55)
                            Segmento = figure.add_subplot(1, 1, 1)
                            Segmento.set_ylabel('Amplitud (mV)')
                            Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
```

```
                            Segmento.set_xlabel('Tiempo (s)')
                            Segmento.set_title(' Señal I')
                            Capsulasegmento = FigureCanvasTkAgg(figure, master =
ecgFrame)
                            Capsulasegmento.get_tk_widget().pack(pady = 5)
                            # aqui debe de actualizar el plot

                        # debe de desplazar la señal hacia adelante 6seg, ojo para clicks
< 5

                    def Cambiar_grafica(event):
                        global SignalPlot, Capsulasegmento, Capsulapico, Clicks
                        global Inicio_pico_Seg, Fin_pico_Seg
                        global Vtiempo, VdatoTiempo, VdataSegmento, VdatoSegmento
                        global VdatoTiempo, VdatoAmplitud, VdatoString,
FrecuenciaCardiaca

                        Clicks = 0

                        SignalSelect = desplegable.get()
                        SignalPlot = globals()[SignalSelect]

                        Datospicoplottiempo = Vtiempo[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]
                        Datospicoplotseñal = SignalPlot[np.where(Vtiempo ==
Inicio_pico_Seg)[:][0][0] : np.where(Vtiempo == Fin_pico_Seg)[:][0][0]]*1000

                        Datossegmentoplottiempo = Vtiempo[(Clicks*590):((Clicks+1)*590)]
                        Datossegmentoplotseñal =
SignalPlot[(Clicks*590):((Clicks+1)*590)]*1000

                        Capsulasegmento.get_tk_widget().destroy()
                        figure = plt.Figure(dpi=55)
                        Segmento = figure.add_subplot(1, 1, 1)
                        Segmento.set_ylabel('Amplitud (mV)')
                        Segmento.plot(Datossegmentoplottiempo,Datossegmentoplotseñal)
                        Segmento.set_xlabel('Tiempo (s)')
                        Segmento.set_title(' Señal '+ SignalSelect)
                        Capsulasegmento = FigureCanvasTkAgg(figure, master = ecgFrame)
                        Capsulasegmento.get_tk_widget().pack(pady = 5)
                        Capsulapico.get_tk_widget().destroy()
                        P = (SignalPlot[np.where(Vtiempo == VdatoTiempo[1])][:][0])*1000
                        Q = (SignalPlot[np.where(Vtiempo == VdatoTiempo[3])][:][0])*1000
                        R = (SignalPlot[np.where(Vtiempo == VdatoTiempo[4])][:][0])*1000

                        S = (SignalPlot[np.where(Vtiempo == VdatoTiempo[5])][:][0])*1000
                        T = (SignalPlot[np.where(Vtiempo == VdatoTiempo[7])][:][0])*1000

                        figure = plt.Figure(dpi=55)
                        Pico = figure.add_subplot(1, 1, 1)
                        Pico.set_ylabel('Amplitud (mV)')
```

```
                        Pico.plot(DatospicoplottiempoDatospicoplotseñal)
                        Pico.annotate("P", xy = (VdatoTiempo[1],P), xytext =
(VdatoTiempo[1],P+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("Q", xy = (VdatoTiempo[3],Q), xytext =
(VdatoTiempo[3],Q-0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("R", xy = (VdatoTiempo[4],R), xytext =
(VdatoTiempo[4],R+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("S", xy = (VdatoTiempo[5],S), xytext =
(VdatoTiempo[5],S-0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.annotate("T", xy = (VdatoTiempo[7],T), xytext =
(VdatoTiempo[7],T+0.05), arrowprops=dict(facecolor='black', arrowstyle='->'))
                        Pico.set_xlabel('Tiempo (s)')
                        Pico.set_title('Pico Seleleccionado en '+ SignalSelect)
                        Capsulapico = FigureCanvasTkAgg(figure, master = picoFrame)
                        Capsulapico.get_tk_widget().pack(pady = 5)

                        #actualizar el graficos el segmento y el pico

                    btnAtras = Button(ecgFrame, text = "- 3s", font = (fuente, 15,
"bold"), command = res_3s)
                    btnAdelante = Button(ecgFrame, text = "+ 3s", font = (fuente, 15,
"bold"), command = sum_3s)

                    btnAtras.pack(side = 'left', padx = (150, 0), pady = 50)
                    btnAdelante.pack(side = 'right', padx = (0, 150), pady = 50)

                    figure = plt.Figure(dpi=55)
                    Segmento = figure.add_subplot(1, 1, 1)
                    Segmento.set_ylabel('Amplitud (mV)')
                    Segmento.plot(DatossegmentoplottiempoDatossegmentoplotseñal)
                    Segmento.set_xlabel('Tiempo (s)')
                    Segmento.set_title(' Señal I')
                    Capsulasegmento = FigureCanvasTkAgg(figure, master = ecgFrame)
                    Capsulasegmento.get_tk_widget().pack(pady = 5)
                    picoFrame = Frame(frameParametros, height = 50, bg = "#F2F2F2")
                    frameComplejos = Frame(picoFrame, highlightbackground="black",
highlightthickness=2)
                    frameComplejos.pack(side = "left", padx = (75,0))
                    labelCabecera = Label(frameComplejos,fg = color_letra_texto, text =
'Características', bg = "#F2F2F2", font = (fuente, 13, "bold"))
                    labelCabecera.pack(pady=10)
                    tabla = ttk.Treeview(frameComplejos)
                    tabla['columns'] = ('Parámetro', 'Valor')
                    tabla.column('#0', width=0, stretch=tk.NO)  # Columna de índice
oculta
                    tabla.column('Parámetro', width=115)
                    tabla.column('Valor', width=100)
                    tabla.heading('#0', text='', anchor=tk.W)
                    tabla.heading('Parámetro', text='Parámetro', anchor=tk.W)
                    tabla.heading('Valor', text='Valor', anchor=tk.W)
```

```
                    tabla.insert('', 'end', text='1', values=('Frecuencia Cardiaca
',str(FrecuenciaCardiaca)+' LPM' ))
                    tabla.insert('', 'end', text='2', values=('Momento ' +
VdatoString[1],str(round(VdatoTiempo[1],2))+' Seg' ))
                    tabla.insert('', 'end', text='3', values=('Momento ' +
VdatoString[3],str(round(VdatoTiempo[3],2))+' Seg' ))
                    tabla.insert('', 'end', text='4', values=('Momento ' +
VdatoString[4],str(round(VdatoTiempo[4],2))+' Seg' ))
                    tabla.insert('', 'end', text='5', values=('Momento ' +
VdatoString[5],str(round(VdatoTiempo[5],2))+' Seg' ))
                    tabla.insert('', 'end', text='6', values=('Momento ' +
VdatoString[7],str(round(VdatoTiempo[7],2))+' Seg' ))
                    tabla.insert('', 'end', text='7', values=('Duración ' +
VdatoString[1],str(round(VdatoTiempo[2]-VdatoTiempo[0],2))+' Seg' ))
                    tabla.insert('', 'end', text='8', values=('Duración ' +
VdatoString[7],str(round(VdatoTiempo[8]-VdatoTiempo[6],2))+' Seg' ))
                    tabla.insert('', 'end', text='9', values=('Amplitud ' +
VdatoString[4],str(round(VdatoAmplitud[4]*1000,2))+' mV' ))
                    tabla.insert('', 'end', text='10', values=('Dur. Comp. ' +
VdatoSegmento[0],str(round(VdataSegmento[0],2))+' Seg' ))
                    tabla.insert('', 'end', text='11', values=('Dur. Comp. ' +
VdatoSegmento[1],str(round(VdataSegmento[1],2))+' Seg' ))
                    tabla.insert('', 'end', text='12', values=('Dur. Comp. ' +
VdatoSegmento[2],str(round(VdataSegmento[2],2))+' Seg' ))
                    tabla.insert('', 'end', text='13', values=('Dur. Comp. ' +
VdatoSegmento[3],str(round(VdataSegmento[3],2))+' Seg' ))
                    tabla.insert('', 'end', text='14', values=('Dur. Comp. ' +
VdatoSegmento[4],str(round(VdataSegmento[4],2))+' Seg' ))
                    tabla.pack()

                    desplegable = ttk.Combobox(picoFrame, values = ["I", "II", "III",
"aVR", "aVL", "aVF"], font = (fuente, 13), width = 5)
                    desplegable.set("I")
                    desplegable.bind("<<ComboboxSelected>>", Cambiar_grafica)
                    desplegable.pack(side = "right", padx = (0,75))

                    P = (SignalPlot[np.where(Vtiempo == VdatoTiempo[1])][:][0])*1000
                    Q = (SignalPlot[np.where(Vtiempo == VdatoTiempo[3])][:][0])*1000
                    R = (SignalPlot[np.where(Vtiempo == VdatoTiempo[4])][:][0])*1000

                    S = (SignalPlot[np.where(Vtiempo == VdatoTiempo[5])][:][0])*1000
                    T = (SignalPlot[np.where(Vtiempo == VdatoTiempo[7])][:][0])*1000

                    figure = plt.Figure(dpi=55)
                    Pico = figure.add_subplot(1, 1, 1)
                    Pico.set_ylabel('Amplitud (mV)')
                    Pico.plot(Datospicoplottiempo,Datospicoplotseñal)
                    Pico.annotate("P", xy = (VdatoTiempo[1],P), xytext =
(VdatoTiempo[1]+0.02,P), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("Q", xy = (VdatoTiempo[3],Q), xytext =
(VdatoTiempo[3]+0.02,Q), arrowprops=dict(facecolor='red', arrowstyle='->'))
```

```
                    Pico.annotate("R", xy = (VdatoTiempo[4],R), xytext =
(VdatoTiempo[4]+0.02,R), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("S", xy = (VdatoTiempo[5],S), xytext =
(VdatoTiempo[5]+0.02,S), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.annotate("T", xy = (VdatoTiempo[7],T), xytext =
(VdatoTiempo[7]+0.02,T), arrowprops=dict(facecolor='red', arrowstyle='->'))
                    Pico.set_xlabel('Tiempo (s)')
                    Pico.set_title('Pico Seleleccionado en I')
                    Capsulapico = FigureCanvasTkAgg(figure, master = picoFrame)
                    Capsulapico.get_tk_widget().pack(pady = 5)

                    lbl8 = Label(frameHeader, text = "PARAMETROS", fg = "black", font =
(fuente, 17, "bold"), bg = "#F2F2F2")

                    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 12,
"bold"), command = datos)
                    btnDiagnosticar = Button(frameFooter, text = "Diagnosticar", font =
(fuente, 12, "bold"), command = diagnostico)

                    frameHeader.pack(side = 'top', fill = 'x')
                    frameParametros.pack(fill = 'both', expand = '1')
                    frameFooter.pack(side = 'bottom', fill = 'x')
                    ecgFrame.pack(side = 'top', fill = 'both', expand = True)
                    picoFrame.pack(side = 'bottom', fill = 'both', expand = True)

                    lbl8.pack(side = 'top', fill = 'both', expand = 1, pady = 5)

                    btnVolver.pack(side = 'left', padx = (100, 0), pady = 25)
                    btnDiagnosticar.pack(side = 'right', padx = (0, 100), pady = 25)
                    if len(divergencias)>=1:
                        if len(divergencias)==1:
                            emergente = tk.Toplevel(ventana)
                            emergente.title('Advertencia')
                            emergente.geometry(str('240x180+' +
str(ventana.winfo_screenwidth()//2 - 240 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                            emergente.configure(background = 'white')
                            emergente.attributes("-toolwindow", True)
                            emergente.resizable(False,False)
                            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                            etiqueta = tk.Label(emergente, text = "Advertencia!!!")
                            etiqueta.pack( anchor = CENTER)
                            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg
='white')
                            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta3.png")))
                            labelAlerta.configure(image=ventana.alerta)
                            labelAlerta.pack()
                            etiqueta2 = tk.Label(emergente, text =("Divergencia en la
estimación de \n" + "la onda " + str(divergencias[0])))
```

```
                            etiqueta2.pack(anchor = CENTER)
                            etiqueta2.configure( font = (fuente, 10, "bold"), bg
='white')
                            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command
= emergente.destroy,  font = (fuente, 10, "bold"))
                            btn_aceptar.pack(pady = 10)
                        else:
                            temp = ""
                            for i in range(len(divergencias)):
                                if i == len(divergencias)-1:
                                    temp = temp + str(divergencias[i]) + "."
                                else:
                                    temp = temp + str(divergencias[i]) + ","
                            emergente = tk.Toplevel(ventana)
                            emergente.title('Advertencia')
                            emergente.geometry(str('240x180+' +
str(ventana.winfo_screenwidth()//2 - 240 // 2) + "+" + str(ventana.winfo_screenheight()
// 2 - 180 // 2)))
                            emergente.configure(background = 'white')
                            emergente.attributes("-toolwindow", True)
                            emergente.resizable(False,False)
                            emergente.protocol("WM_DELETE_WINDOW", lambda: None)
                            etiqueta = tk.Label(emergente, text = "Advertencia!!!")
                            etiqueta.pack( anchor = CENTER)
                            etiqueta.config(font = (fuente, 10, "bold"), fg = 'red', bg
='white')
                            labelAlerta = tk.Label(emergente, bg = fondo, image = None)
                            ventana.alerta =
ImageTk.PhotoImage((Image.open("./Code/lib/alerta3.png")))
                            labelAlerta.configure(image=ventana.alerta)
                            labelAlerta.pack()
                            etiqueta2 = tk.Label(emergente, text =("Divergencia en la
estimación de \n" + "las ondas " + temp))
                            etiqueta2.pack(anchor = CENTER)
                            etiqueta2.configure( font = (fuente, 10, "bold"), bg
='white')
                            btn_aceptar = tk.Button(emergente, text = "Aceptar",  command
= emergente.destroy,  font = (fuente, 10, "bold"))
                            btn_aceptar.pack(pady = 10)

def diagnostico():
    for widgets in frameMayor.winfo_children():
        widgets.destroy()

    global diagnosticoFrame
    global VdataSegmento, VdatoTiempo, VdatoAmplitud, FrecuenciaCardiaca
    global Diagnostico
    diagnosticoFrame = Frame(frameMayor, bg = "#F2F2F2")
    diagnosticoFrame.pack(fill = "both", expand = 1)

    frameHeader = Frame(diagnosticoFrame, height = 50, bg = "#F2F2F2")
```

```
    lbl25 = Label(frameHeader, text = "DIAGNOSTICO", fg = "black", font = (fuente, 20,
"bold"), bg = "#F2F2F2")
    lbl25.pack(side = 'top', fill = 'both', expand = 1, pady = 30)
    frameHeader.pack(side = 'top', fill = 'x')

    frameEnfermedades = Frame(diagnosticoFrame, height = 100, bg = "#F2F2F2")
    frameEnfermedades.pack(fill = 'both', expand = '1')
    listaFrame = Frame(frameEnfermedades, bg = "#F2F2F2", highlightbackground = 'black',
highlightthickness = 2)

    lbl9 = Label(listaFrame, text = "Posibles enfermedades: ", fg = "black", font =
(fuente, 13, "bold"), bg = "#F2F2F2")

    lbl9.pack(side = 'top', fill = 'both', pady = 15, padx = 50, anchor = "w")

    listaFrame.pack(side = 'left', fill = 'both', padx = 5)

    frameSoporte = Frame(frameEnfermedades, bg = "#F2F2F2")
    lbl26 = Label(frameSoporte, text = "Soporte de diagnóstico", fg = "black", font =
(fuente, 15, "bold"), bg = "#F2F2F2")
    lbl26.pack(side = 'top', fill = 'both', pady = 12, anchor = 'w')
    lbl27 = Label(frameSoporte, text = "", fg = "black", font = (fuente, 12, "bold"), bg
= "#F2F2F2")
    lbl27.pack(fill = 'both', pady = 5, anchor = 'w')
    frameSoporte.pack(side = 'right', fill = 'both', expand = '1')

    ventana.a = tk.BooleanVar()
    ventana.b = tk.BooleanVar()
    ventana.c = tk.BooleanVar()
    ventana.d = tk.BooleanVar()
    ventana.m = tk.BooleanVar()
    ventana.f = tk.BooleanVar()
    ventana.e = tk.BooleanVar()
    ventana.g = tk.BooleanVar()
    ventana.h = tk.BooleanVar()
    ventana.i = tk.BooleanVar()
    ventana.k = tk.BooleanVar()
    ventana.l = tk.BooleanVar()
    ventana.j = tk.BooleanVar()

    if (VdatoTiempo[8]-VdatoTiempo[6] < 0.1 and VdatoAmplitud[1] < 0.00005 and
VdataSegmento[2] > 0.12):
            ventana.a.set(True)
            lbl27['text'] = lbl27['text'] + "\n  Hiperpotacemia: \n"
            if VdatoTiempo[8]-VdatoTiempo[6] < 0.1:
                lbl27['text'] = lbl27['text'] + "- La duración de la onda T es inferior a
0.1 Seg. \n"
            if  VdatoAmplitud[1] < 0.00005:
                lbl27['text'] = lbl27['text'] + "- La amplitud de la onda P es inferior a
0.05 mV. \n"
            if VdataSegmento[2] > 0.12:
```

```
                lbl27['text'] = lbl27['text'] + "- La duración del complejo QRS es
superior a 0.12 Seg. \n"
    else:
            ventana.a.set(False)

    if (VdatoAmplitud[1] > 0.00025):
        ventana.b.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipertrofia auricular derecha: \n"
        lbl27['text'] = lbl27['text'] + "- La amplitud de la onda P es inferior a 0.25
mV. \n"
    else:
        ventana.b.set(False)



    if (VdatoTiempo[2]-VdatoTiempo[0] > 0.1):
        ventana.c.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Dilatacion auricular: \n"
        lbl27['text'] = lbl27['text'] + "- La duración de la onda P es superior a 0.1
Seg. \n"
    else:
        ventana.c.set(False)


    if (VdataSegmento[1] > 0.2):
        ventana.d.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bloqueo auroventricular: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo PR es superior a 0.2
Seg. \n"
    else:
        ventana.d.set(False)

    if (VdataSegmento[2] > 0.12):
        ventana.m.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bloqueo de rama: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QRS es superior a
0.12 Seg. \n"
    else:
        ventana.m.set(False)

    if (VdataSegmento[3] > 0.42 and VdatoAmplitud[4] < 0.002):
        ventana.f.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Miocarditis: \n"
        if VdataSegmento[3] > 0.42:
            lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es superior a
0.42 Seg. \n"
        if VdatoAmplitud[4] < 0.002:
            lbl27['text'] = lbl27['text'] + "- La amplitud de la onda R es inferior a 0.2
mV. \n"
    else:
        ventana.f.set(False)
```

```
    if (VdataSegmento[3] < 0.32):
        ventana.e.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipercalcemia: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es inferior a 0.32
Seg. \n"
    else:
        ventana.e.set(False)

    if (VdataSegmento[3] > 0.42):
        ventana.g.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Sindrome del QT Prolongado: \n"
        lbl27['text'] = lbl27['text'] + "- La duración del complejo QT es superior a 0.42
Seg. \n"
    else:
        ventana.g.set(False)

    if (VdataSegmento[4] > 0.15 and VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003)):
        ventana.h.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Pericarditis: \n"
        if VdataSegmento[4] > 0.15:
            lbl27['text'] = lbl27['text'] + "- La duración del complejo ST es superior a
0.15 Seg. \n"
        if  VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003):
            lbl27['text'] = lbl27['text'] + "- la amplitud de la onda T es superior a la
Amplitud de R / 3mV. \n"
    else:
        ventana.h.set(False)

    if (VdatoTiempo[8]-VdatoTiempo[6] > 0.12):
        ventana.j.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Hipertrófia Auricular Izquierda: \n"
        lbl27['text'] = lbl27['text'] + "- La duración de la onda T es superior a 0.12
Seg. \n"
    else:
        ventana.j.set(False)


    if (VdatoTiempo[8]-VdatoTiempo[6] > 0.12 and VdatoAmplitud[7] >
(VdatoAmplitud[4]/0.003)):
        ventana.i.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Isquemia: \n"
        if VdatoTiempo[8]-VdatoTiempo[6] > 0.12:
            lbl27['text'] = lbl27['text'] + "- La duración de la onda T es superior a
0.12 Seg. \n"
        if VdatoAmplitud[7] > (VdatoAmplitud[4]/0.003):
            lbl27['text'] = lbl27['text'] + "- la amplitud de la onda T es superior a la
Amplitud de R / 3mV. \n"
    else:
        ventana.i.set(False)
```

```
    if (FrecuenciaCardiaca > 100):
        ventana.k.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Taquicardia Sinusual: \n"
        lbl27['text'] = lbl27['text'] + "- La frecuencia cardiaca es superior a 100 LPM.
\n"
    else:
        ventana.k.set(False)

    if (FrecuenciaCardiaca < 60):
        ventana.l.set(True)
        lbl27['text'] = lbl27['text'] + "\n  Bradicardia Sinusual: \n"
        lbl27['text'] = lbl27['text'] + "- La frecuencia cardiaca es inferior a 60 LPM.
\n"
    else:
        ventana.l.set(False)

    lblDisclaimer = Label(frameSoporte, text = "Los resultados de este análisis se basan
en la derivación I y no son concluyentes, \n por lo que se recomienda consultar con un
médico.", bg = "#F2F2F2", fg = color_letra_texto, font = (fuente, 10, "bold"))
    lblDisclaimer.pack(side = 'bottom', pady = 10)

    lbl10 = Checkbutton(listaFrame, state = "disabled", variable = ventana.a , text =
"Hiperpotacemia", fg = color_letra_texto, font = (fuente, 10))
    lbl11 = Checkbutton(listaFrame, state = "disabled", variable = ventana.b , text =
"Hipertrofia auricular derecha", fg = color_letra_texto, font = (fuente, 10))
    lbl12 = Checkbutton(listaFrame, state = "disabled", variable = ventana.c , text =
"Dilatacion auricular", fg = color_letra_texto, font = (fuente, 10))
    lbl15 = Checkbutton(listaFrame, state = "disabled", variable = ventana.d , text =
"Bloqueo Auroventricular", fg = color_letra_texto, font = (fuente, 10))
    lbl16 = Checkbutton(listaFrame, state = "disabled", variable = ventana.e , text =
"Hipercalcemia", fg = color_letra_texto, font = (fuente, 10))
    lbl17 = Checkbutton(listaFrame, state = "disabled", variable = ventana.f , text =
"Miocarditis", fg = color_letra_texto, font = (fuente, 10))
    lbl18 = Checkbutton(listaFrame, state = "disabled", variable = ventana.g , text =
"Sindrome del QT Prolongado", fg = color_letra_texto, font = (fuente, 10))
    lbl19 = Checkbutton(listaFrame, state = "disabled", variable = ventana.h , text =
"Pericarditis", fg = color_letra_texto, font = (fuente, 10))
    lbl20 = Checkbutton(listaFrame, state = "disabled", variable = ventana.i , text =
"Isquemia", fg = color_letra_texto, font = (fuente, 10))
    lbl21 = Checkbutton(listaFrame, state = "disabled", variable = ventana.j , text =
"Hipertrofia auricular izquierda", fg = color_letra_texto, font = (fuente, 10))
    lbl22 = Checkbutton(listaFrame, state = "disabled", variable = ventana.k , text =
"Taquicardia", fg = color_letra_texto, font = (fuente, 10))
    lbl23 = Checkbutton(listaFrame, state = "disabled", variable = ventana.l , text =
"Bradicardia", fg = color_letra_texto, font = (fuente, 10))
    lbl24 = Checkbutton(listaFrame, state = "disabled", variable = ventana.m , text =
"Hipertrofia Ventricular o Bloqueo de Rama", fg = color_letra_texto, font = (fuente, 10))
    lbl10.pack(anchor = "w")
    lbl11.pack(anchor = "w")
    lbl12.pack(anchor = "w")
    lbl15.pack(anchor = "w")
```

```
    lbl16.pack(anchor = "w")
    lbl17.pack(anchor = "w")
    lbl18.pack(anchor = "w")
    lbl19.pack(anchor = "w")
    lbl20.pack(anchor = "w")
    lbl21.pack(anchor = "w")
    lbl22.pack(anchor = "w")
    lbl23.pack(anchor = "w")
    lbl24.pack(anchor = "w")

    frameFooter = Frame(diagnosticoFrame, height = 50, bg = "#F2F2F2")
    btnVolver = Button(frameFooter, text = "Volver", font = (fuente, 15, "bold"), command
= resultados)
    btnVolver.pack(side = 'left', padx = (100, 0), pady = 25)
    btnSalir = Button(frameFooter, text = "Salir", font = (fuente, 15, "bold"), command =
ventana.destroy)
    btnSalir.pack(side = 'right', padx = (0, 100), pady = 25)
    btnHome = Button(frameFooter, text = "Inicio", font = (fuente, 15, "bold"), command =
inicio)
    btnHome.pack(pady = 25)
    frameFooter.pack(side = 'bottom', fill = 'x')

inicio()
ventana.mainloop()

#############################################################
###              SECCION DE PRUEBAS SIN UI                ###
#############################################################


# inicio = time.time()
# Preparacion()
# PDF_to_jpeg(ruta_PDF)
# crop_image()
# Concat_image()
# Limpiar_imagen()
# Lines_Hough(cv2.imread('./Code/Data/Output/img-ECG.jpeg'))
# Vectorizacion_Señal()
# ObtencionPico()
# Características()
# ObtencionParametros()
# Diagnosticar()
# fin = time.time()
# print(str(fin-inicio))
```