UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA

"INTERFAZ GRÁFICA PARA MEDICIÓN DE ÁNGULOS DE EXTREMIDADES INFERIORES DEL CUERPO HUMANO POR MEDIO DE SENSORES INERCIALES"

AUTOR

Alejandro David López Pozo

DIRECTOR

MSc. Iglesias Navarro Iván

Ibarra – Ecuador

2020-2021

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1722929476		
APELLIDOS Y NOMBRES:	Alejandro David López Pozo		
DIRECCIÓN:	Palora y Lago agrio		
EMAIL:	Adlopez150@gmail.com		
TELÉFONO FIJO:	2959455	**TELÉFONO MÓVIL:**	0999084698

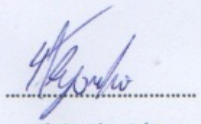
DATOS DE LA OBRA	
TÍTULO:	"Interfaz gráfica para medición de ángulos de extremidades inferiores del cuerpo humano por medio de sensores inerciales"
AUTOR (ES):	Alejandro David López Pozo
FECHA: DD/MM/AAAA	28/06/2021
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA
ASESOR /DIRECTOR:	MSc. Ing. Iván Iglesias Navarro

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de junio de 2021

EL AUTOR:



Alejandro López

1722929476

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICO

Que la Tesis previa a la obtención del título de Ingeniero en Mecatrónica con el tema: "INTERFAZ GRÁFICA PARA MEDICIÓN DE ÁNGULOS DE EXTREMIDADES INFERIORES DEL CUERPO HUMANO POR MEDIO DE SENSORES INERCIALES", ha sido desarrollado y terminado en su totalidad por el Sr. Alejandro David López Pozo, con cédula de identidad: 1722929476, bajo mi supervisión para lo cual firmo en constancia.

IVAN IGLESIAS NAVARRO

Digitally signed by IVAN IGLESIAS NAVARRO
Date: 2021-06-28 12:52-05:00

.....

MSc. Ing. Iván Iglesias Navarro

DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTO

Agradezco a mis padres y abuelos por su apoyo incondicional a lo largo de mis estudios, a mis amigos por los cuales el camino fue más fácil, a todos los docentes que me transmitieron su conocimiento y sabiduría.

Gracias por todo a todos aquellos que participaron en mi formación académica y personal.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

Este trabajo está dedicado a todo aquel que me apoyo y a todo aquel al que le sea de utilidad.

Tabla de contenido

Índice de figuras

Listado de Anexos

- Código módulos esclavos y maestro.
- Código de la interfaz

Resumen

En la actualidad existe la necesidad de nuevos sistemas de adquisición y procesamiento de datos angulares corporales, ya sea aplicado en la rehabilitación, detección de lesiones o estudio de movimiento, esto se evidencia como una necesidad cuando las herramientas usadas en este proceso son anticuadas o costosas, lo cual dificulta su uso cotidiano ya sea por falta de eficiencia o por la dificulta en la adquisición de nuevos sistemas.

En el siguiente trabajo de grado se presenta el desarrollo una interfaz gráfica para medición de ángulos de extremidades inferiores del cuerpo humano por medio de sensores inerciales, dicha interfaz fue desarrollada en *Python* con la herramienta *Tkinter*, además de poseer las siguientes características: permite: el ingreso de múltiples registros, modificarlos, eliminarlos, también, permite la toma de datos angulares una pierna a la vez, con indicadores de estado dinámico de fácil comprensión. Todos los datos ingresados serán guardados en una base de datos *PostgreSQL* con un hostlocal, también permite graficar la señal de los datos obtenidas y realizar una simulación 2D, toda la programación se encuentra dividida por métodos con la finalidad de permitir que el programa pueda ser escalable, el software se centra en el estudio de la marcha humana.

También se diseñó el sistema de adquisición de datos, el cual contiene 4 módulos inerciales, cuyo funcionamiento se basa en la toma de los datos del acelerómetro y giroscopio propiciados por el un sensor inercial para posteriormente ser procesados, dando como resultado ángulos de navegación del segmento de estudio, para posteriormente ser guardarlos y posteriormente transmitidos a un módulo maestro, cuya función consiste en realizar una conexión directa con cada módulo inercial, ya sea para autorizar el inicio de toma de muestras o para su recolección, dichas muestras son enviadas por medio de conexión serial a la interfaz la cual se encargará de su procesamiento, guardado y todas las funciones anteriormente nombradas.

Abstract

In the Currently there is a need for new systems for the acquisition and processing of body angular data, whether applied in rehabilitation, injury detection or movement study, this is evidenced as a need when the tools used in this process are outdated or expensive. , which makes its daily use difficult either due to lack of efficiency or due to the difficulty in acquiring new systems.

In the following degree project, the development of a graphical interface for measuring angles of the lower extremities of the human body by means of inertial sensors is presented, said interface was developed in *Python* with the *Tkinter* tool, in addition to having the following characteristics: it allows: Entering multiple records, modifying them, deleting them, also, allows angular data collection one leg at a time, with easy-to-understand dynamic status indicators. All the data entered will be stored in a *PostgreSQL* database with a local host, it also allows to graph the signal of the data obtained and perform a 2D simulation, all the programming is divided by methods in order to allow the program to be scalable, the software focuses on the study of human gait.

The data acquisition system was also designed, which contains 4 inertial modules, whose operation is based on taking data from the accelerometer and gyroscope provided by an inertial sensor to later be processed, resulting in segment navigation angles. of study, to later be saved and later transmitted to a master module, whose function consists of making a direct connection with each inertial module, either to authorize the start of sampling or for their collection, said samples are sent by means of serial connection to the interface which will be in charge of processing, saving and all the functions previously mentioned.

Introducción

Antecedentes

En el libro la Goniometría se intenta reunir en un solo volumen el fundamento teórico-práctico de la medición clínica de los ángulos que se forman a nivel de las articulaciones para aplicarlo a la valoración de las incapacidades laborales. El libro es un manual de uso teórico del goniómetro el cual es una herramienta barata pero poco precisa con la que la mayoría de fisioterapeutas están relacionados la herramienta en si tiene una eficiencia baja y requiere de una gran cantidad de tiempo usarla. [1]

En la Universidad de Cauca en el año 2018 se realizó un proyecto de grado referente a la estimación de los ángulos articulares por medio de un arreglo simple de sensores inerciales el trabajo se centró en el tratamiento de las señales de los sensores inerciales con el uso del filtro de Karman, los resultados de este trabajo indicaron la necesidad software de bajo costo que permita la adquisición de los datos de los sensores y su procesamiento puesto que este trabajo se utilizó el software Xsens. [2]

En el año 2017 Álvaro Zárraga realizó un trabajo de grado relacionado con sensores inerciales pero enfocado en la movilidad de la muñeca mediante sensores inerciales.

Los resultados de los movimientos articulares fueron verificados con un goniómetro dando como resultado una gran efectividad en cuanto al factor de tiempo. [3]

En la Universidad técnica del norte en el año 2020 se realizó un sistema para tomar datos de la marcha humana con la herramienta de Kinect en la cual se cumple con lo propuesto, pero a su vez deja varias recomendaciones sobre esta tecnología, como puntos más importantes fueron: la falla en la detección del cuerpo u objetos al usar y otro fue el error detectado de 15%, la autora de dicho trabajo plantea buscar una mejora o un nuevo método para la adquisición de los datos en la marcha humana. [4]

En la Universidad politécnica nacional del Ecuador se realizó una tesis de maestría referente al tema de visualización de la marcha humana enfocado en la detección de gonartrosis en la rodilla, el proyecto tiene un muy buen enfoque y servirá de referente. Al detectar patrones en la marcha de personas sanas como de personas con gonartrosis realiza todo un proceso comparativo, se denota la necesidad de un sistema de medición de ángulos articulares puesto que un sistema que permita interactuar con una base de datos sería de gran apoyo para futuras investigaciones similares. [5]

Descripción del problema

Los problemas físicos pueden ser tratados mediante procesos de rehabilitación y terapia, las cuales se pueden apoyar de ayudas técnicas diseñadas para problemas específicos, sin embargo, no todas las personas afectadas tienen acceso a estas terapias o ayudas técnicas [6].

La Organización mundial de la salud plantea que la rehabilitación ayuda a potenciar al máximo la capacidad de vivir y normalmente a reforzar la independencia [7], en el mismo artículo se expone la necesidad de métodos más económicos que permitan un fácil acceso a rehabilitaciones más eficientes, esto se puede lograr con una combinación de tecnología y conocimiento en el cual la biomecánica es fundamental, ya que no solo se enfoca en la rehabilitación sino también en la maximización de otras actividades. El objetivo de la biomecánica en las actividades deportivas es la caracterización y la mejora de las técnicas del movimiento a partir de conocimientos científicos [8]

Una de las ramas más utilizadas en la rehabilitación es la lectura de ángulos articulares los cuales por medio de rangos específicos se pueden detectar ciertos problemas físicos originados por posturas erróneas, accidentes y/o alteraciones físicas congénitas dichas herramientas dependen mucho de la vista del operario y requiere mucho de su tiempo. [1]

El uso de la tecnología para la toma de ángulos se ha venido realizando de varias maneras ya sea con sensores de movimiento, cámaras u otros dispositivos técnicos los cuales son o de costo elevado o a su vez no poseen la precisión adecuada, una solución a estos inconvenientes y su posible uso complementario es el uso de sensores inerciales para la lectura de ángulos

articulares. En varios estudios se ha planteado su uso, cada uno con ciertos problemas en el procesamiento y uso de los datos [2]y. [3]

Teniendo en cuenta estos precedentes se concluye que existe la necesidad de una herramienta que apoye a una fácil obtención de datos del cuerpo humano ya sea para la rehabilitación o para mejorar el desempeño físico con una interfaz amigable con datos procesables y útiles para posteriores estudios.

Objetivos

Objetivo General

Diseñar una interfaz gráfica para medición de ángulos de extremidades inferiores del cuerpo humano por medio de sensores inerciales.

Objetivos Específicos

- Definir el método correcto para la toma de ángulos de la marcha humana y procesamiento de señales de sensores inerciales.
- Programar el sistema de adquisición y procesamiento de las señales provenientes de los sensores inerciales.
- Implementar la interfaz gráfica y la estructura de la base de datos.
- Validar el sistema y analizar los resultados.

Justificación

El uso de la tecnología en la rehabilitación o detección de lesiones se evidencia como una necesidad cuando se requiere el uso diario de herramientas anticuadas y poco efectivas como el goniómetro, el cual requiere de un entrenamiento y un arduo trabajo por parte del fisioterapista; además de una gran cantidad de uso de tiempo [1].

Se han propuesto en varios trabajos que proponen: el uso de sensores, sistemas de filtrado de información y algunos métodos para el procesamiento de la información con interfaz

graficas funcionales, pero todas ellas tienen inconvenientes en ciertos aspectos como: el costo ya sea en los sensores o en el software a utilizar, inclusive en la forma de tratar las señales. Por otro lado, también existen propuestas con diversos métodos, pero estos procedimientos usados tradicionalmente generan gran incomodidad a los participantes, en el proceso de adecuación del equipo y en la posterior toma de datos, limitando la autonomía del paciente y provocando que se alteren sus patrones de caminata, lo cual genera valores incorrectos de las variables a usar. [9]

Se debe tomar en cuenta que la adquisición de datos debe ser de manera efectiva y rápida, para una correcta lectura de ángulos articulares ya que el estudio de estos se realiza con ángulos específicos de movimiento o con rangos de movimiento ya establecidos por el fisioterapista por ende se debe asegurar que los datos obtenidos sean fiables y a su vez que sean datos en tiempo real de movimiento. [3] y [10]

Esto evidencia una necesidad de implementar los resultados positivos de varias investigaciones más recientes, además de dar una solución que sea posible de aplicar con pocos recursos y que posea una confiabilidad acorde con la necesidad de los profesionales del medio.

Alcance

Se realizará una interfaz gráfica que permita visualizar por medio de representaciones una aproximación de los ángulos articulares de cadera, rodilla y tobillo, también permitirá ingresar información del usuario y guardar todo en una base de datos.

Los datos serán tomados por medio de sensores inerciales luego serán enviados a una tarjeta de desarrollo para su normalización, posterior a eso se enviarán a un CPU el cual por medio de Software permitirá su procesamiento, manipulación y visualización.

Capítulo 1

Biomecánica de la marcha humana y procesamiento de señales de sensores inerciales.

1.1 Introducción

En este capítulo se estudian los factores fundamentales de la marcha humana, sus articulaciones y su rango de movimiento, con la finalidad de tener una mayor comprensión de esta y sus principales características. También se plantea la importancia y uso de la polimetría especialmente el plano sagital para una posterior comprensión de las fases y sub-fases de la marcha humana con lo cual se comprendera de mejor forma la cinemática de esta, al final del capítulo se exponen los componentes con su respectiva explicación de los sensores inerciales, de igual manera se expondrán los métodos usados para tratar las señales de dichos sensores.

1.1 Miembro inferior humano

El miembro inferior humano se especializa en mantener la locomoción y asegura el equilibrio, sus articulaciones son amplias y están unidas por potentes ligamentos.

En la articulación de la cadera se unen la cabeza del fémur con la cavidad cotiloidea. La articulación de la rodilla además de la flexión y de la extensión sólo permite limitados movimientos de rotación. La articulación tobillo formada entre el astrágalo, la tibia y el peroné, y sólo permite movimientos de flexión y de extensión. El eje longitudinal del pie forma un ángulo recto con la pierna, que es efectiva para mantener la postura erecta, en la Figura 1 se muestra una representación de la pierna. [11]

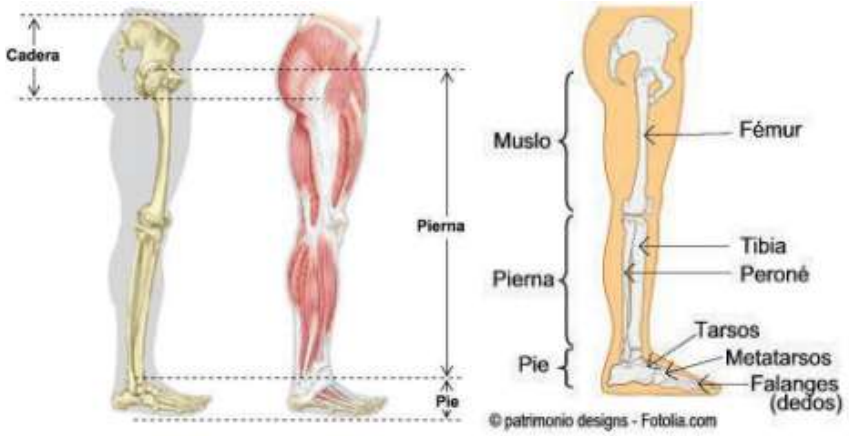


Figura 1: Estructura ósea, muscular de la pierna y segmentos de la pierna de estructura ósea. [12]

1.1.1 Anatomía de la cadera

La cadera cumple un rol fundamental en la marcha humana puesto que es para proveer movilidad y estabilidad al cuerpo humano, ver Figura 2.

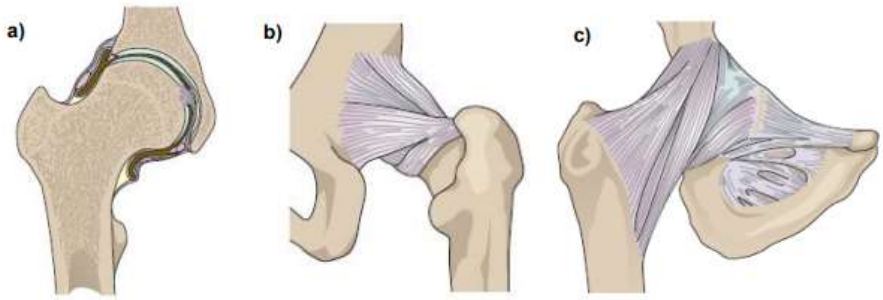


Figura 2: Articulación de la cadera humana. [12]

La articulación presente en la cadera por ser una unión esférica permite a toda la extremidad inferior moverse por los tres planos del cuerpo humano. La pierna puede moverse libremente en la cadera o a su vez esta puede estar fija y ser la cadera la que se mueva respecto a ella, en la marcha dichas condiciones van alternando, dependiendo de la fase en la que se encuentre. [12]

Los movimientos básicos de la pierna respecto a la cadera están representados en la Figura 3 y son:

- **Extensión y flexión.** -Se genera en el plano sagital, la extensión hacia atrás tiene un máximo de 15º, mientras la extensión respecto al tórax tiene un ángulo máximo de 140º a 130º
- **Abducción y Aducción.** – Se genera en el plano transversal, la abducción tiene un rango de movimiento de 30º a 45º alejándose del cuerpo, mientras que la aducción tiene un rango de 20º a 30º cruzando sobre la otra pierna.
- **Rotación. -** Se genera alrededor del eje vertical, la rotación interna tiene un rango de 30º a 45º mientras que la rotación externa posee un rango de 40º a 50º. [12]

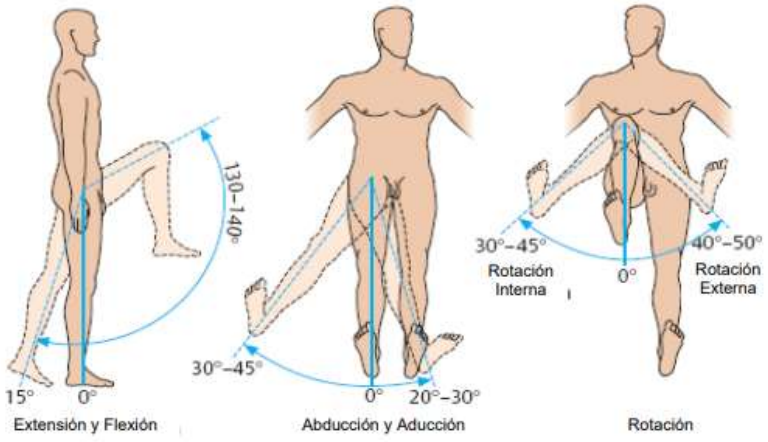


Figura 3: Rangos de movimiento articulación de la cadera. [2]

1.1.2 Anatomía de la rodilla

La rodilla es la articulación más grande del cuerpo humano en ella se puede apreciar la unión de 3 huesos (fémur, tibia y rotula) de suma importancia para la marcha humana los cuales se pueden apreciar en la Figura 4, esta articulación soporta el peso corporal y estabilidad al poseer extensión completa, además de estar dotada para una movilidad en función del tipo de terreno. [12] y [5].

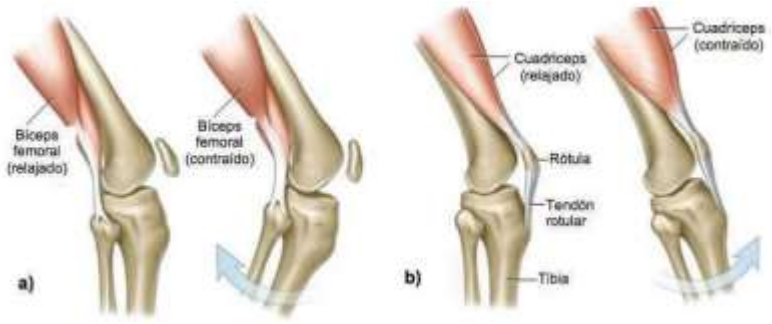


Figura 4: Extensión de la rodilla. [3]

El ángulo de la rodilla es el ángulo generado entre el fémur y la rodilla, dicho ángulo se observa equivalente a 0º cuando la pierna está extendida, el ángulo de una flexión total puede llegar a los 155º mientras que si se fuerza el cuádriceps se puede llegar a un ángulo de -10º, en la Figura 5 se puede apreciar dichos rangos de movimiento. [5]

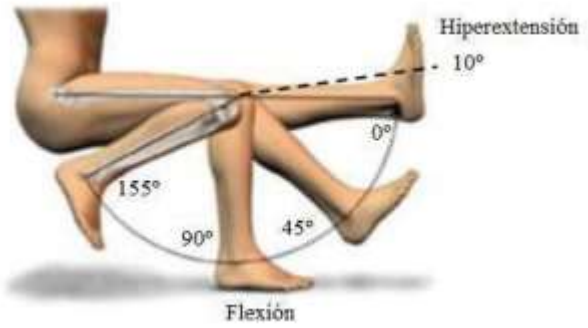


Figura 5: Movimiento de la rodilla plano sagital. [3]

1.1.3 Anatomía del tobillo

El tobillo está conformado por 2 articulaciones, ver representación en la Figura 6, la articulación del tobillo conformada por la tibia, peroné y astrágalo, mientras que la articulación intertarsal se encuentra conformada por astrágalo del pie, hueso calcáneo y el escafoides, dichas articulaciones pueden realizar el movimiento de flexión y extensión. [12] y [5]

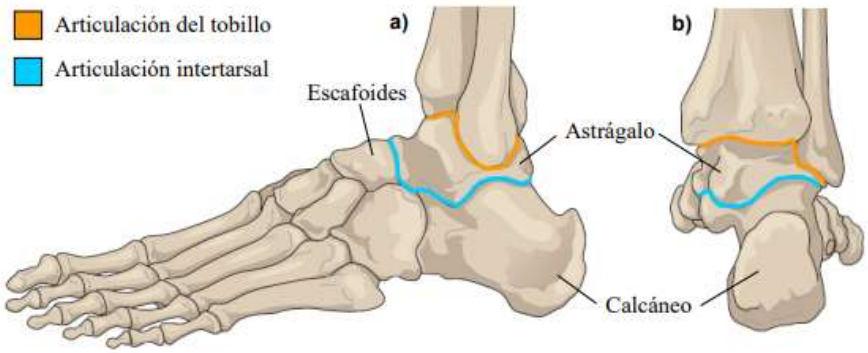


Figura 6: Articulaciones del tobillo. [2]

El ángulo de la articulación del tobillo es formado por la tibia y el pie, cuando el pie se encuentra en posición neutral el ángulo formado es de 90º, dependiendo de la flexibilidad de la persona el pie en dorsiflexión (la punta del pie en dirección a la tibia) tiene un ángulo máximo de 30º y la flexión plantar (pararse de puntas) tiene un ángulo máximo de 50º, se puede apreciar de mejor manera el rango de movimiento en la Figura 7. [12]

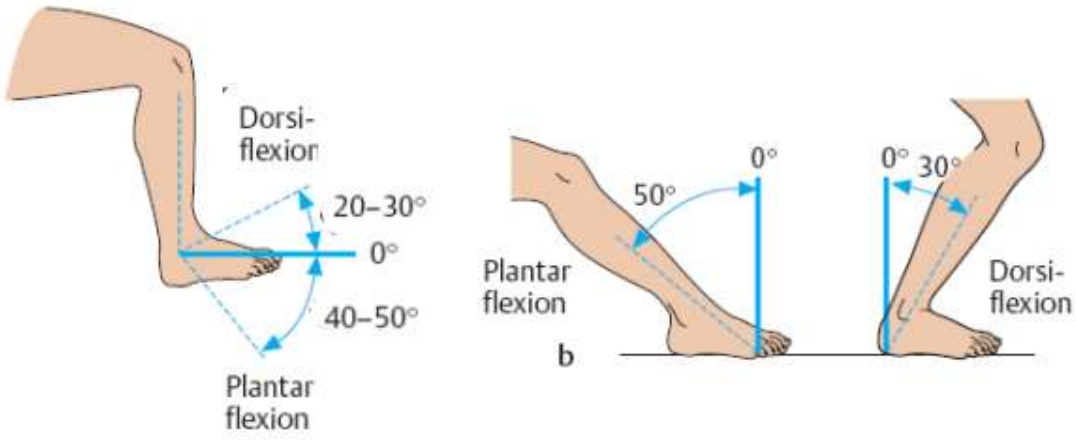


Figura 7: Rangos de movimiento tobillo humano. [2]

1.2 Polimetría

Es el conjunto de coordenadas, términos y puntos de orientación convencionales empleados para describir la posición de una estructura anatómica dentro del cuerpo, así como su relación con el resto de los elementos anatómicos presentes en el organismo, dichos planos y ejes se encuentran ilustrados en la Figura 8.

El conocimiento de los planos, ejes y sistemas orientación anatómica resulta fundamental para permitir la comunicación fluida y sin errores entre los equipos médicos. [13]

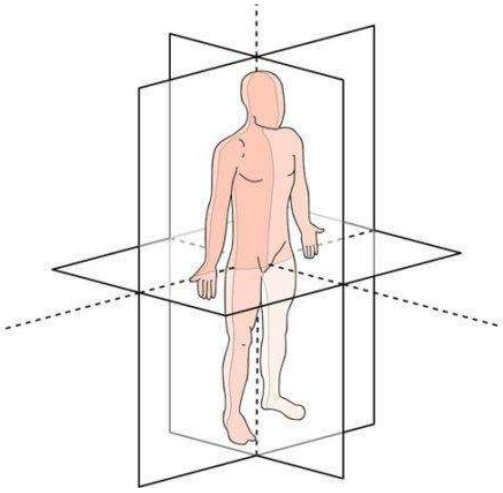


Figura 8: Representación de planos y ejes del cuerpo humano. [4]

1.2.1 Plano sagital

El plano sagital es aquel que divide el cuerpo humano en partes equivalentes: derecha e izquierda, es el plano paralelo al movimiento de la pierna al caminar en línea recta, por lo que los movimientos más importantes de la pierna se realizan paralelamente a este plano, ver Figura 9. Los movimientos de los eslabones en el plano sagital son de extensión y flexión, los cuales son medidos desde la articulación de la cadera y la rodilla. [14]

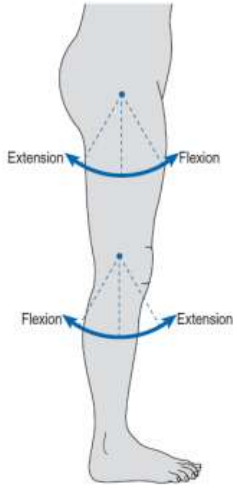


Figura 9: Movimiento de la pierna en torno a las articulaciones de la cadera y rodilla [5]

En este plano es en donde se puede apreciar de manera más eficiente todo el proceso de marcha tomando en cuenta todas sus subfases, por ende, en este plano se realizará la toma de ángulos aciculares presentes en cada una.

1.3 Marcha humana

Se puede definir la marcha como la forma de desplazamiento en posición bípeda en la que suceden apoyos bipodales y los monopodales**,** a su vez la locomoción humana normal se ha descrito como una serie de movimientos alternantes, rítmicos, de las extremidades y del tronco que determinan un desplazamiento hacia delante del centro de gravedad, en la Figura 10 se ilustra un ciclo de marcha completo. La locomoción normal puede describirse con algunas características comunes. Aunque existen pequeñas diferencias en la forma de la marcha de un individuo a otro, estas diferencias caen dentro de pequeños límites. El ciclo de la marcha comienza cuando el pie contacta con el suelo y termina con el siguiente contacto con el suelo del mismo pie. Los dos mayores componentes del ciclo de la marcha son: la fase de apoyo y la fase de balanceo. Una pierna está en fase de apoyo cuando está en contacto con el suelo y está en fase de balanceo cuando no contacta con el suelo. [15]

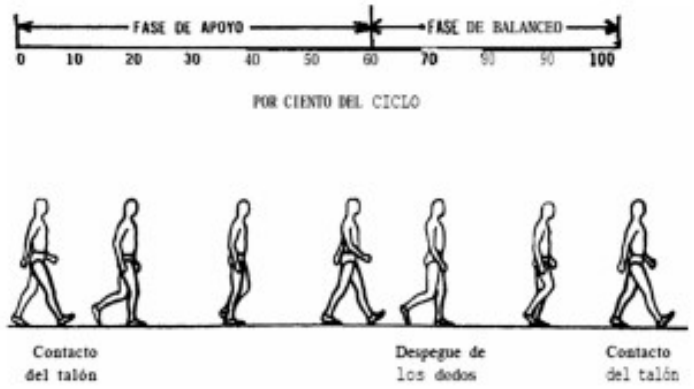


Figura 10: Representación de ciclo de marcha humana. [6]

1.3.1 Fase de apoyo

Esta fase puede subdividirse en 5 momentos de suma utilidad ilustrados en la Figura 11 y son:

- **Contacto del talón** (hace referencia al instante en el cual el talón de la pierna de referencia hace contacto con el suelo)

- **Apoyo plantar** (hace referencia al instante en el cual la planta anterior del pie hace contacto con el suelo)
- **Apoyo medio** (hace referencia al instante en el cual la articulación mayor de la cadera se encuentra alineada verticalmente al centro del pie)
- **Elevación del talón** (hace referencia al instante en el cual el talón de la pierna contraria a la de referencia se eleva)
- **Despegue del pie** (hace referencia al instante en el cual los dedos se despegan del suelo)

Todas estas faces pueden ser reconocida por la cantidad de peso distribuido en el pie respecto al suelo. [15]

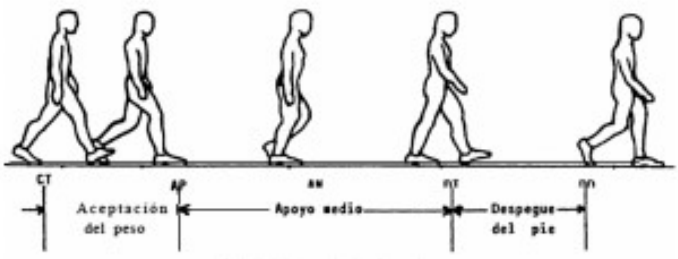


Figura 11: Fase de apoyo. [6]

1.3.2 Fase de balanceo

Esta fase puede subdividirse en 3 momentos de suma utilidad ilustrados en la Figura 12 y son:

- **Periodo de aceleración** (Se caracteriza por su rápida aceleración del extremo de la pierna de manera inmediata en cuanto los dedos dejan el suelo)
- **Periodo de balanceo medio** (En esta etapa la pierna balanceada se mueve hacia adelante pasando a la pierna de apoyada.)
- **Periodo de desaceleración** (Se caracteriza por la desaceleración de la pierna cuando se acerca el final del intervalo.) [15]

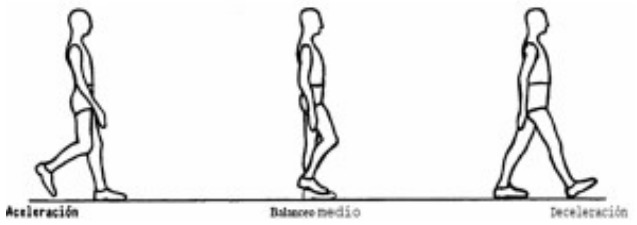


Figura 12: Fase balanceo. [6]

1.4 Cinemática de la marcha humana

En el análisis cinemático de la marcha posee una gran relevancia el estudio de las variaciones angulares de las distintas articulaciones inferiores, cuya movilidad es imprescindible para el normal desarrollo de la marcha. [16]

El análisis cinemático completo de la marcha humana toma en cuenta el ángulo de cada articulación respecto a cada plano corporal, además de las reacciones musculares y fuerzas físicas externas, varios autores toman en consideración únicamente el plano sagital puesto que aporta más información del posicionamiento articular y a su vez dividen la marcha en 3 intervalos de movimiento. [5], [12] y [14]

1.4.1 Intervalo 1

Este intervalo se produce en la fase de apoyo entre las subdivisiones del contacto del talón hasta apoyo medio.

En la Tabla 1 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento del tobillo en el intervalo 1.

Tabla 1: Ángulos de movimiento del estudio cinemático del tobillo intervalo 1.

Sub-Fase	Posición
Contacto del talón	En posición neutral 0º. Justo entre la dorsiflexión y la flexión plantar.
Apoyo plantar	Se mueve 15º respecto a la posición neutral de la flexión plantar.
Apoyo medio	Pasa rápidamente a aproximadamente 5º de dorsiflexión.

Fuente: [5] y [12].

En la Figura 13 se encuentran ilustrados los ángulos articulares expuestos en la tabla 1 [12].

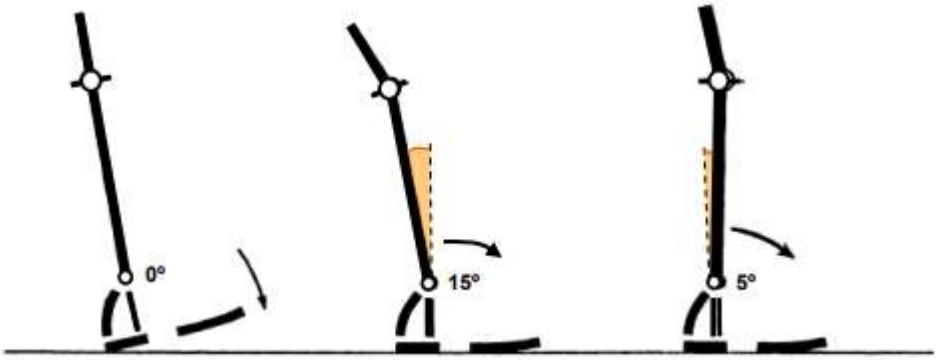


Figura 13: Estudio cinemático del tobillo intervalo 1. [2]

En la Tabla 2 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la rodilla en el intervalo 1.

Tabla 2: Ángulos de movimiento del estudio cinemático de la rodilla intervalo 1.

Sub-Fase	Posición
Contacto del talón	En posición neutral 0º. En completa extensión.
Apoyo plantar	Tiene aproximadamente 20º de flexión y comienza a extenderse.
Apoyo medio	Tiene aproximadamente 10º de flexión y continúa extendiéndose.

Fuente: [12]y [5].

En la Figura 14 se encuentran ilustrados los ángulos articulares expuestos en la Tabla 2 [12].

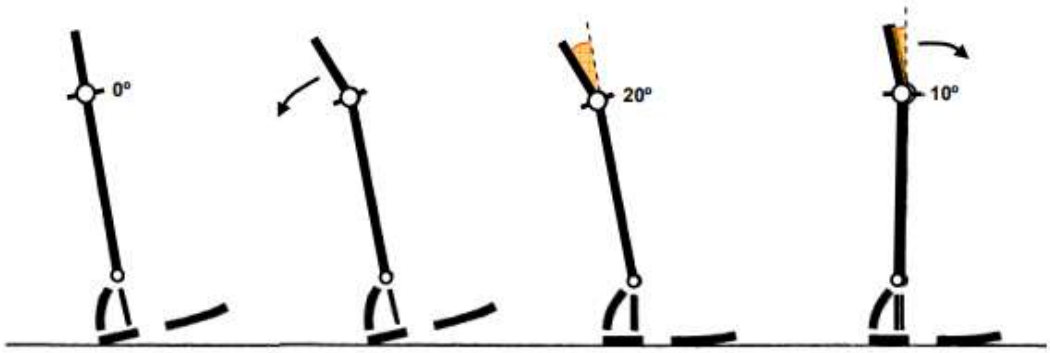


Figura 14: Estudio cinemático de la rodilla intervalo 1. [2]

En la Tabla 3 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la cadera en el intervalo 1.

Tabla 3: Ángulos de movimiento del estudio cinemático de la cadera intervalo 1.

Sub-Fase	Posición
Contacto del talón	Está a aproximadamente 30º de flexión.
Apoyo plantar	El ángulo de flexión disminuye alrededor de 20º.
Apoyo medio	Se mueve a posición neutral 0º.

Fuente: [12]y [5].

En la Figura 15 se encuentran ilustrados los ángulos articulares expuestos en la Tabla 3 [12].

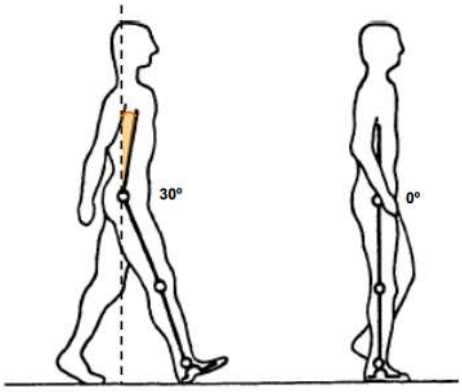


Figura 15:Estudio cinemático de la cadera intervalo 1.[2]

1.4.2 Intervalo 2

Este intervalo se produce en la fase de apoyo entre las subdivisiones de apoyo medio hasta el despegue del pie.

En la Tabla 4 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento del tobillo en el intervalo 2.

Tabla 4: Ángulos de movimiento del estudio cinemático del tobillo intervalo 2.

Sub-Fase	Posición
Apoyo medio	Pasa rápidamente a aproximadamente 5º de dorsiflexión.
Elevación del talón	Está aproximadamente a 15º de dorsiflexión
Despegue del pie	Se mueve rápidamente 35º, con lo que el ángulo final del tobillo es de 25º aproximados en flexión

Fuente: [12]y [5].

En la Figura 16 se encuentran ilustrados los ángulos articulares expuestos en la tabla 4 [12].

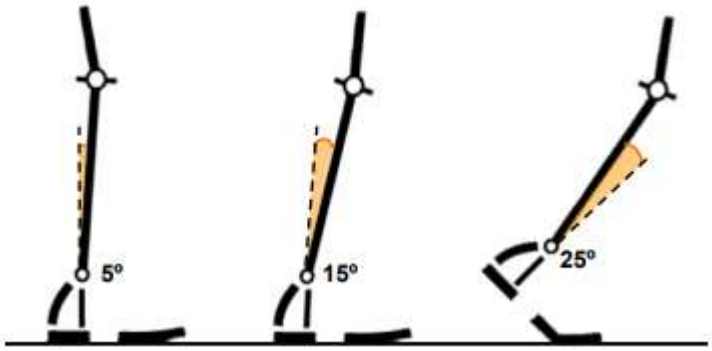


Figura 16: Estudio cinemático del tobillo intervalo 2. [2]

En la Tabla 5 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la rodilla en el intervalo 2.

Tabla 5: Ángulos de movimiento del estudio cinemático de la rodilla intervalo 2.

Sub-Fase	Posición
Apoyo medio	Tiene aproximadamente 10º de flexión y continúa extendiéndose.
Elevación del talón	Está aproximadamente a 4º de flexión de la extensión completa.
Despegue del pie	Se mueve de una extensión casi completa a aproximada 40º de flexión.

Fuente: [12]y [5].

En la Figura 17 se encuentran ilustrados los ángulos articulares expuestos en la Tabla 5 [12].

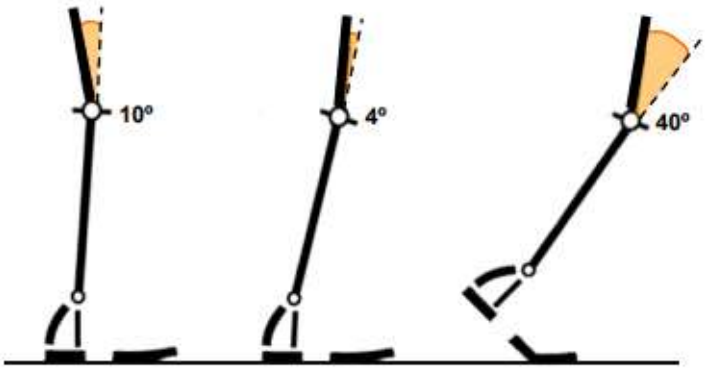


Figura 17: Estudio cinemático de la rodilla intervalo 2. [2]

En la Tabla 6 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la cadera en el intervalo 2.

Tabla 6: Ángulos de movimiento del estudio cinemático de la cadera intervalo 2

Sub-Fase	Posición
Apoyo medio	Se mueve a posición neutral 0º.
Elevación del talón	Alcanza un máximo de hiperextensión de 20º
Despegue del pie	Se encuentra cerca de una posición neutral y se mueve en dirección de la flexión.

Fuente: [12]y [5].

En la Figura 18 se encuentran ilustrados los ángulos articulares expuestos en la Tabla 6 [12].

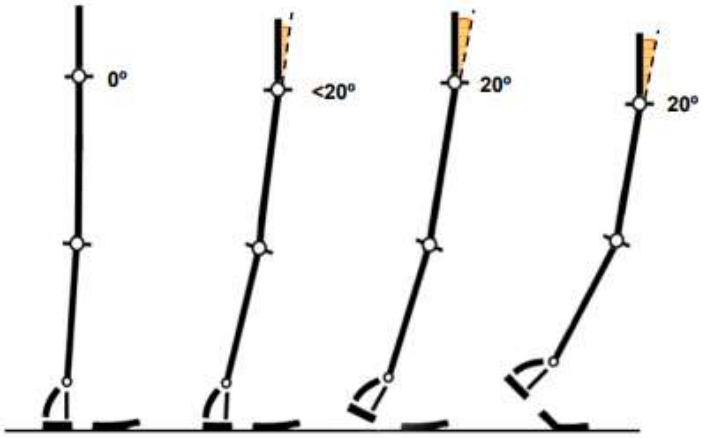


Figura 18: Estudio cinemático de la cadera intervalo 2. [2]

1.4.3 Intervalo 3

Este intervalo se produce en la fase de balanceo en la Tabla 7 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento del tobillo en el intervalo 3.

Tabla 7: Ángulos de movimiento del estudio cinemático del tobillo intervalo 3.

Sub-Fase	Posición
Todas las etapas de balanceo.	En posición neutral 0º. Justo entre la dorsiflexión y la flexión plantar.

Fuente: [12]y [5].

En la Tabla 8 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la rodilla en el intervalo 3.

Tabla 8: Ángulos de movimiento del estudio cinemático de la rodilla intervalo 3

Sub-Fase	Posición
Periodo de aceleración y balanceo medio.	Se flexiona de una posición inicial de 40º hasta un ángulo máximo de 65º aproximadamente.
Periodo de desaceleración y balanceo medio.	Se extiende casi completamente hasta el último instante de la etapa.

Fuente: [12]y [5].

En la Tabla 9 se puede observar un análisis más detallado de los ángulos comúnmente realizados en el movimiento de la cadera en el intervalo 3.

Tabla 9: Ángulos de movimiento del estudio cinemático de la cadera intervalo 3

Sub-Fase	Posición
Todas las etapas de balanceo.	Se flexiona 30º aproximadamente y se mantiene hasta el final de la etapa.

Fuente: [12]y [5].

En la Figura 19 se encuentran ilustrados los ángulos articulares expuestos en la tabla 7,8 y 9 [12].

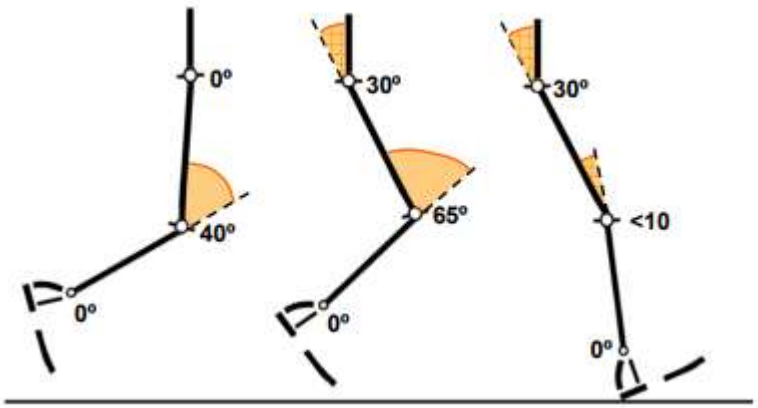


Figura 19: Estudio cinemático de la pierna intervalo 3. [2]

La evolución angular de las articulaciones puede ser representadas gráficamente respecto al tiempo, siendo una referencia de una marcha humana normal representado en la Figura 20.

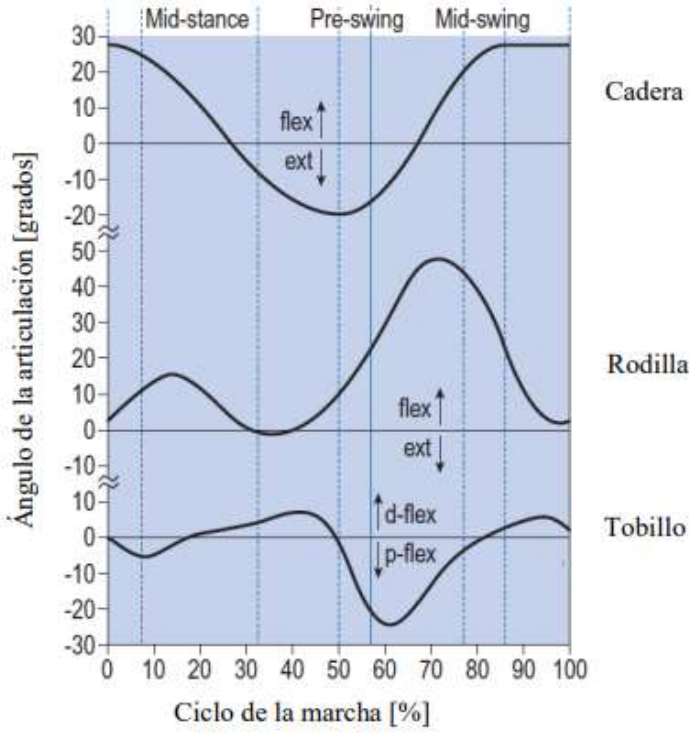


Figura 20: Evolución angular del tobillo, rodilla y cadera durante el ciclo de marcha en el plano sagital. [12]

Dicha evolución articular se puede representar de manera digital por lo cual se pueden usar varios métodos de adquisición de datos.

1.5 Sensores Inerciales

Son utilizados principalmente para el estudio y análisis de movimiento, en base a las variables de aceleración y velocidad angular, que son obtenidas mediante acelerómetros y giroscopios respectivamente. [17]

1.5.1 Acelerómetro

La medida de la aceleración se basa en la variación de una capacitancia en el interior del chip. Se trata de un *MEMS* (Sistemas microelectromecánicos) que suspende partículas de silicio, ancladas en un punto fijo, que se mueven libremente en el eje de medición. Cuando una aceleración actúa, toda esta masa de partículas se desplaza respecto a su posición de origen y

crea un desequilibrio en la capacitancia, que se mide y da información de la aceleración que está actuando en ese eje, este dispositivo se encuentra ilustrado en la Figura 21. [18]

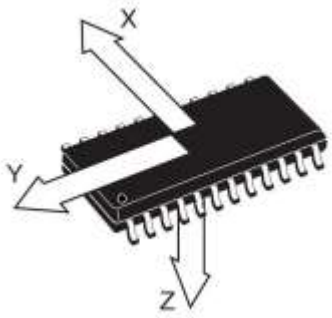


Figura 21: Esquema de orientación de ejes de medida. [9]

1.5.2 Magnetómetro

Es un sensor que mide la intensidad de campo magnético en 3 ejes. Gracias a estas 3 medidas se obtiene un vector de campo que da información del ángulo del movimiento o de su actitud. Se podría decir que es la versión ampliada a 3 dimensiones de la brújula, pero este instrumento es capaz de distinguir giros en los ejes de roll y pitch. El campo se mide a través unas magneto-resistencias que cambian su valor en función del campo que las atraviesa en su dirección, este dispositivo se encuentra ilustrado en la Figura 22. [18]

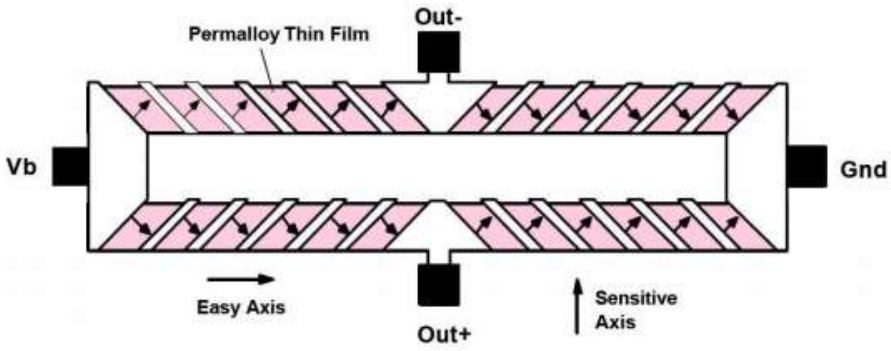


Figura 22: Magneto-Resistencia [9]

1.5.3 Giroscopio

El giroscopio embarcado transforma la fuerza generada por un movimiento angular en una señal eléctrica proporcional a ella. Gracias a estos sensores se puede conocer la velocidad

de giro de un móvil y, por tanto, también su orientación [19], este dispositivo se encuentra ilustrado en la Figura 23.

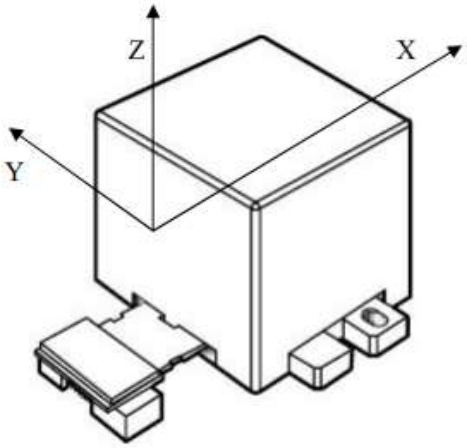


Figura 23: Esquema giroscopio. [9]

1.6 Ángulos de navegación

Los ángulos de navegación son utilizados para describir la orientación de un objeto en tres dimensiones de un sistema de coordenadas móvil respecto a un sistema fijo, dichos ángulos proporcionan la posición del sistema móvil en un momento determinado los cuales son: dirección (ψ), elevación (θ) y ángulo de alabeo o roll(φ) mejor apreciados en la Figura 24. [20]

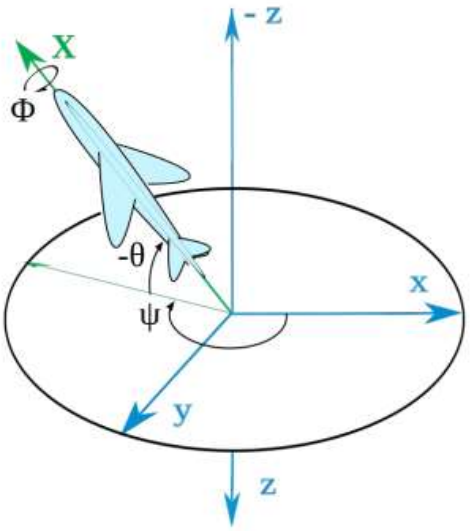


Figura 24: Referencia de ángulos de navegación. [11]

Dichos ángulos también son conocidos como ángulos de Euler ψ, θ y φ.

1.6.1 Ángulos de Euler

Euler probó que para especificar la rotación de un sistema de coordenadas en el espacio se necesitan 3 parámetros reales, y que toda rotación en R3 puede obtenerse como una composición de tres rotaciones elementales consecutivas: se selecciona el orden de los ejes coordenados y se efectua la rotación alrededor de ese eje de forma consecutiva. Los tres ángulos correspondientes forman los ángulos de Euler representados en la Figura 25. [21]

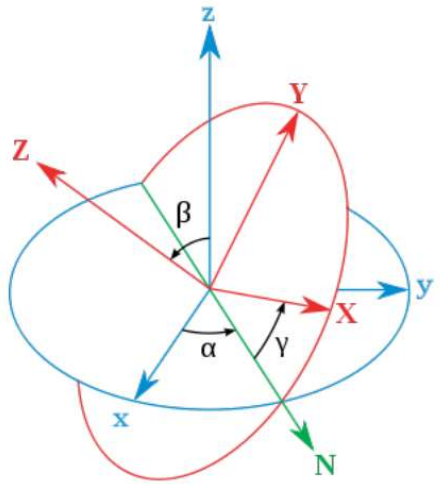


Figura 25: Representación ángulos de Euler. [21]

Estos ángulos se pueden representar como cuaterniones para representar las orientaciones y las rotaciones de objetos en tres dimensiones. Tiene componentes en i,j,k que representan los ejes alrededor del cual se producirán una rotación. También tiene componente q0 o eje referencial el cual representará la cantidad de rotación que se producirá alrededor del mismo. [20]

1.6.2 Cuaterniones

Los cuaterniones son números hipercomplejos de cuatro componentes; una componente real y tres componentes imaginarias. Se puede denotar de la forma q = w + x.i + y.j + z.k con (x, y, z) vector en el espacio, w parte real y (i, j, k) base imaginaria. Los cuaterniones tienen una notación compacta y son muy fáciles de componer. Las operaciones usando cuaterniones son muy eficientes en comparación con las que usan matrices, ya que requieren menor cantidad de operaciones básicas y menor espacio de almacenamiento. Conforman una representación no

singular y solucionan el problema de “pérdida de dimensionalidad”. Gracias a todo esto se usa en aplicaciones que requieren rotaciones o ubicaciones en el espacio bajo demanda en tiempo real. [22]

Capítulo 2

Marco metodológico.

2.1 Introducción

En este capítulo se expondrán los materiales y métodos a usar en el desarrollo del Sistema de adquisición de datos e interfaz gráfica de la marcha humana, de esta forma se revisarán las características y beneficios del hardware, sus protocolos de conexión, la ubicación general del sistema, las ecuaciones necesarias para la detección de ángulos y las normativas generales para el desarrollo de la interfaz gráfica, además de plantear una idea más detallada del sistema y su función.

2.2 Vista general del Sistema

Para analizar la marcha de un usuario, el sistema debe ser capaz de recolectar los cambios de posición de sus extremidades inferiores por medio de una detección continua de ángulos acordes a la fase de marcha en la que se encuentre. Los segmentos de sus miembros inferiores son equipados con módulos inerciales, circuitos impresos encargados de medir y transmitir varias señales de interés la información de los módulos inerciales será recibida por un dispositivo llamado módulo maestro, el cual se encarga de retransmitirla periódicamente a un equipo externo (computador) que ejecuta una aplicación virtual.

Además de incorporar los algoritmos de procesamiento de datos, esta aplicación presenta la información del sistema a su operador, por medio de una interfaz gráfica. El sistema de medición proporciona las señales necesarias por cada segmento de los miembros inferiores y calcula los ángulos de flexión-extensión de cadera, rodilla y tobillo de ambos miembros inferiores un miembro a la vez. [23]

2.3 Módulo Inercial

Para diseñar un módulo inercial se requieren de 3 partes principales: el sensor el cual es el encargado de la toma de datos, el cerebro o controlador y un transmisor el cual está encargado de enviar los datos al dispositivo maestro.

2.3.1 Sensor *MPU 6050*

Este módulo contiene un giroscopio de 3 ejes y un acelerómetro de 3 ejes, permitiendo así medir el movimiento en 6 grados de libertad, combinando dichos componentes además integra un *DMP* (Procesador digital de movimiento) capaz de realizar complejos algoritmos de captura de movimiento en 9 ejes, ver la Figura 26. [24]

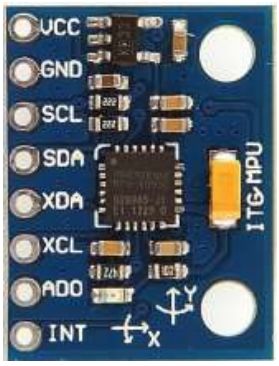


Figura 26 Sensor MPU6050. [24]

Se comunica a través del protocolo de comunicación I2C. además posee una librería para su uso inmediato. Este sensor incorpora un regulador de tensión a 3.3V y resistencias pull-up para su uso directo por I2C. [24]

Este sensor fue seleccionado por tener una facilidad de acceso en el Mercado nacional, además de ser el mayormente usado en este tipo de proyectos con una mayor cantidad de documentación, ejemplos y proyectos.

2.3.2 *Arduino Nano*

Según la página oficial de Arduino su versión "*Arduino Nano*" es una placa de desarrollo de tamaño compacto, completa y compatible con protoboards, basada en el microcontrolador *ATmega328P*. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, un cristal de 16Mhz, conexión Mini-USB, terminales para conexión *ICSP* y un botón de reseteo ver la Figura 27. [25]

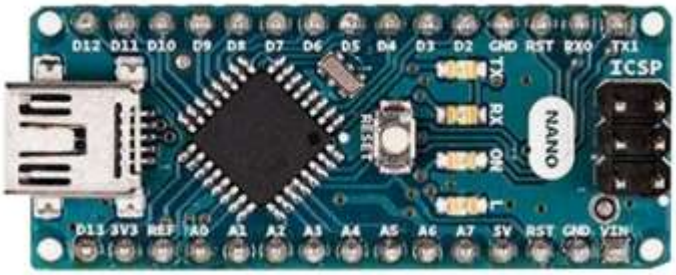


Figura 27 Arduino Nano. [25]

Posee las mismas capacidades que un Arduino UNO, tanto en potencia del microcontrolador como en conectividad, solo se ve recortado en su conector USB, conector jack de alimentación y los pines cambian a un formato de pines header. [25]

La mayoría de las tarjetas de desarrollo son basadas en Arduino UNO por ello se escogió su versión más compacta siendo igual de eficiente, además de ser OpenSourse y poseer las librerías necesarias para el uso del sensor *MPU 6050* además de ejemplos prácticos en los cuales se basará el código posterior. [5]

2.3.3 *HC-05* Módulo *Bluetooth*

El *HC-05* es un módulo *Bluetooth* V2.0+EDR configurable mediante comandos AT, el cual puede funcionar como dispositivo maestro o esclavo, con una frecuencia de operación: 2.4 GHz Banda ISM, a una distancia de hasta 10 metros en condiciones óptimas con un voltaje de operación de 3.6 VDC a 6 VDC y con un consumo de 30 mA a 50Ma, ver la Figura 28. [26]

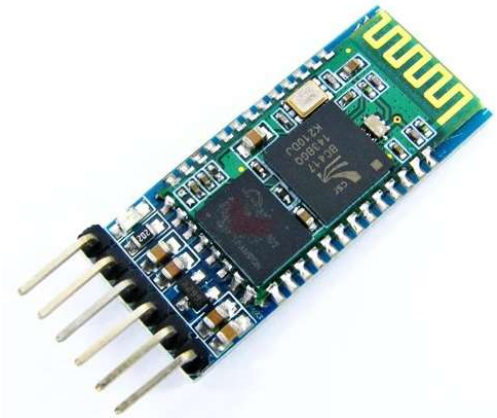


Figura 28 HC-05 Módulo Bluetooth. [26]

El módulo de *Bluetooth HC-05* ofrece una buena relación de precio y características, ya que es un módulo Maestro-Esclavo, quiere decir que además de recibir conexiones desde una

PC o tablet, también es capaz de generar conexiones hacia otros dispositivos *Bluetooth*. Esto permite, por ejemplo, conectar dos módulos de *Bluetooth* y formar una conexión punto a punto para transmitir datos entre dos microcontroladores o dispositivos, de igual manera dicho módulo puede lograr una conexión múltiple con varios módulos por medio de un arreglo de código posteriormente indicado. [26]

Se seleccionó un módulo bluethooth puesto que varios autores señalan que el uso de cables puede llegar a interferir en la toma de correctas mediciones ya sea por la generación de ruido o interfiriendo con la normal forma de locomoción del usuario y como ya se mencionó anteriormente el módulo *HC-05* es el mejor en cuanto calidad y precio. [23]y [24]

2.3.4 Lector de adaptador de tarjeta Micro SD para Arduino.

Es un módulo lector de tarjetas Micro SD para lectura y escritura a través del sistema de archivos y el controlador de interfaz SPI, admite tarjeta Micro SD y tarjeta Micro SDHC, se alimenta con 3.3 V, posee una Interfaz de control de seis pines (GND, VCC, MISO, MOSI, SCK, CS) GND a tierra, VCC es la fuente de alimentación, MISO, MOSI, SCK para el bus SPI, CS es el pin de señal de selección de chip, ver Figura 29. [27]

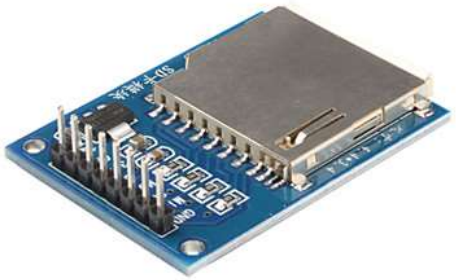


Figura 29 Lector de adaptador de tarjeta Micro SD Micro SDHC Mini TF. [27]

2.4 Protocolos de Comunicación

Un protocolo de comunicación está formado por un conjunto de reglas y formatos de mensajes establecidas a priori para que la comunicación entre el emisor y un receptor sea posible Las reglas definen la forma en que deben de efectuarse las comunicaciones de las redes, incluyendo la temporización, la secuencia, la revisión y la corrección de errores. [28]

2.4.1 Protocolo I2C

El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz. La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos, ver Figura 30. [29]

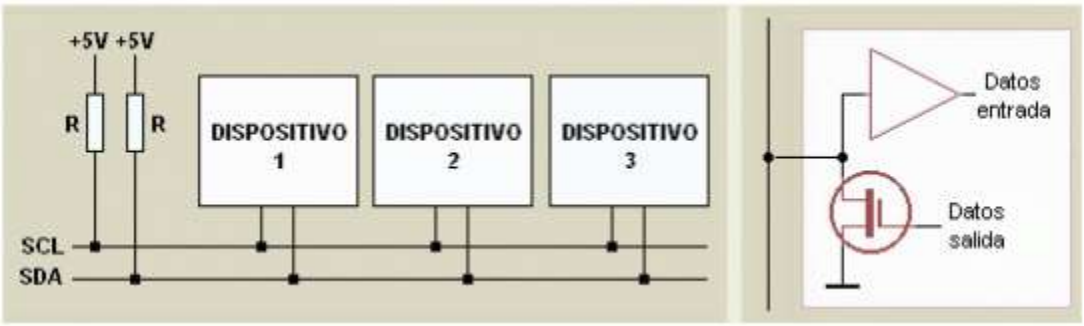


Figura 30 Esquema de conexión y funcionamiento protocolo I2C. [29]

Este es el protocolo que se usa para la comunicación del microcontrolador (*Arduino Nano*) y el sensor inercial (*MPU 6050*).

2.4.2 Redes *Bluetooth*

La topología de las redes *Bluetooth* puede ser punto-a-punto o punto-a-multipunto ver Figura 31, los dispositivos, se comunican en redes denominadas *piconets*. Estas redes tienen posibilidad de crecer hasta tener 8 conexiones punto a punto. Además, se puede extender la red mediante la formación de sctternets. Una scatternet es la red producida cuando dos dispositivos pertenecientes a dos *piconets* diferentes, se conectan.

En una *piconets*, un dispositivo debe actuar como máster, enviando la información del reloj y la información de los saltos de frecuencia.

El resto de los dispositivos actúan como esclavos. [30]

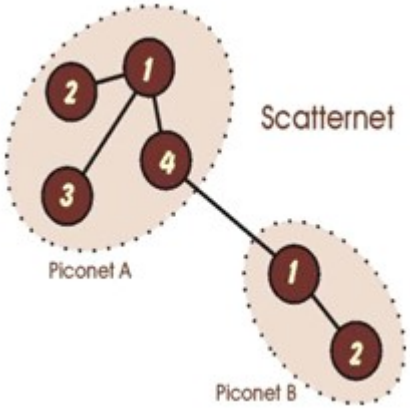


Figura 31 Red piconet Bluetooth. [30]

El método usado para la comunicación *Bluetooth* es una red *piconets* simple la cual consiste en una conexión e intercambio de datos de dispositivo maestro esclavo uno a la vez, la programación del módulo maestro servirá para ir variando su conexión con los diferentes esclavos con la finalidad de recolectar los datos esclavo por esclavo, los módulos *HC-05* dan la facilidad de modificar sus conexiones desde el módulo cerebro (*Arduino Nano*) con un arreglo en la conexión y comandos AT.

2.5 Técnicas utilizadas para el cálculo de posición y orientación de la unidad de procesamiento múltiple (MUP).

Los datos comúnmente entregados por los sensores inerciales son:

- Aceleración y velocidad rotacional en el eje X
- Aceleración y velocidad rotacional en el eje Y
- Aceleración y velocidad rotacional en el eje Z.

El arreglo de sensores inerciales es utilizado para determinar las trayectorias del ciclo de marcha humana, donde los ángulos y posiciones son calculados utilizando los siguientes parámetros:

- Aceleración
- Velocidad angular
- Filtro complementario y Filtro Kalman. [19]

2.5.1 Cálculo por Aceleración

Este parámetro permite conocer la orientación del sensor haciendo uso de la aceleración de la gravedad natural que actúa sobre el circuito integrado en todo momento. La desventaja de este método es que es insensible a rotaciones sobre el eje en el cual actúa el vector de gravedad, así mismo para que el cálculo sea efectivo, la única aceleración que debe estar afectando al sensor deberá ser la gravedad. [19]

Para el cálculo de la aceleración en un espacio tridimensional se utilizan las Ecuaciones 1 y 2:

$$\theta x = tan^{-1}\left(\frac{ax}{\sqrt{ay^2 + az^2}}\right)$$

Ecuación 1 aceleración de espacio tridimensional eje X [19]

$$\theta y = tan^{-1}\left(\frac{ay}{\sqrt{ax^2 + az^2}}\right)$$

Ecuación 2 aceleración de espacio tridimensional eje Y [19]

Tener en cuenta que se esta calculando el ángulo de inclinación, si se desea el ángulo de rotación es decir por ejemplo el ángulo que rota el eje x en su mismo eje, entonces en las ecuaciones anteriores se necesita cambiar el ay por el ax y viceversa, donde ax, ay, az son los datos entregados por el acelerómetro, es decir sus aceleraciones respecto a cada eje. [20]

2.5.2 Cálculo de la velocidad angular

Los giroscopios son capaces de proporcionar lecturas directamente proporcionales a la rotación angular en cada eje. Realizando una integración de la velocidad angular respecto al

tiempo y conociendo el ángulo inicial es posible conocer la orientación del sensor, esto se realiza utilizando las Ecuaciones 3 y 4 [19]:

$$\theta x = \theta x0 + \omega x \Delta t$$

Ecuación 3 Velocidad angular eje X [19]

$$\theta y = \theta y0 + \omega y \Delta t$$

Ecuación 4 Velocidad angular eje Y [19]

La desventaja de este método es que es difícil conocer los intervalos de tiempo exacto sobre los cuales se realiza la integración por una serie de factores como el tiempo de ejecución en el procesamiento computacional, y variaciones en la frecuencia del reloj de los dispositivos debido a cambios de temperatura o variaciones en las fuentes de alimentación de voltaje. Este error llamado *DRIFT* es acumulativo en cada iteración, lo que hace a este método poco efectivo en cortos periodos de tiempo. [19]

2.5.3 Filtro complementario

Este método combina los resultados de los ángulos calculados por los acelerómetros y por los giroscopios con la finalidad de compensar el *DRIFT* y a su vez las fuerzas externas que pueden llegar a actuar sobre la IMU y que afectan la medición en los acelerómetros. Las proporciones recomendadas oscilan entre un 98% para el giroscopio y un 2% para los acelerómetros. De esta forma el ángulo calculado responderá rápido ante las rotaciones y no será tan sensible a las aceleraciones ocasionadas al desplazar al sensor. [19]

Este filtro es una unión de dos diferentes: un pasa altas para el giroscopio y un pasa bajas para el acelerómetro. La fórmula que viene de esta combinación se observa en la Ecuación 5 [19]:

$$\theta = 0.96(\acute{a}ngulo + \omega gyro\ dt) + 0.04(\alpha acel)$$

Ecuación 5 Combinación filtros pasa altas y pasa bajas. [19]

2.5.4 Filtro Kalman

El filtro de Kalman es un algoritmo de filtrado de datos discretos, utilizado en sistemas en los cuales existe cierta incertidumbre de los datos entregados por los sensores, realiza una

predicción del sistema $f(t+1)$ y posteriormente se efectúa una corrección con los valores reales medidos de forma iterativa. Tiene como desventaja que requiere de más capacidad de procesamiento computacional, por lo tanto, su implementación en sistemas con recursos limitados como la plataforma Arduino debe ser realizada tomando en cuenta el uso de recursos que esto implica. [19]

2.6 Estimación de Ángulos entre segmentos

Con la finalidad de estimar los ángulos de las articulaciones de cadera y tobillo durante la marcha de un usuario, se deberá ubicar cada módulo inercial en una zona que caracteriza a un segmento corporal específico: uno en la zona lumbar específicamente a la altura de L5, uno en el muslo, uno en la canilla y uno en el dorso del pie. De esta forma se asegura que se registren los datos necesarios de aceleración y velocidad angular de cada segmento de interés. [31]

Para realizar el cálculo de los ángulos de las articulaciones a partir de los datos entregados por el módulo inercial, primero se deben calcular los cuaterniones correspondientes a cada segmento. Posteriormente, se deben alinear los módulos inerciales con un sistema de referencia denominado q0. Para ello, se debe solicitar al usuario que mantenga su posición referencia mientras se adquieren los datos necesarios para definir el sistema de referencia y de esta manera se puede determinar la posición por la Ecuación 5. [31]

$$Qs = Qg * Qc^{-1}$$

Ecuación 5 Ecuación de orientación del objeto. [31]

Donde Qs es el cuaternión de la orientación del segmento en el sistema de referencia global, Qg es el cuaternión de la orientación del sensor en el sistema de referencia Qc.

En la Figura 32 se presenta un esquema del sistema de referencia utilizado para alinear los ejes de los sensores al inicio de la adquisición. Se muestra los cuatro módulos inerciales en sus ubicaciones respectivas sobre los segmentos S1, S2, S3 Y S4, que representan a las diferentes partes de la pierna. [31]

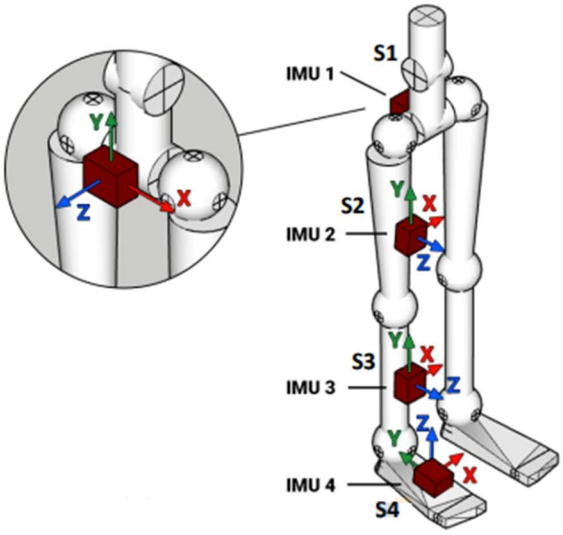


Figura 32 Sistema de referencia para la ubicación de los módulos inerciales. [31]

Posterior a alinear los módulos inerciales al sistema de referencia en cada segmento de la extremidad representado por un vector es rotado y los ángulos entre ellos se obtiene con la Ecuación 6, como el arcocoseno de producto punto de los vectores V1 Y V2 correspondientes a los segmentos adyacentes. [31]

$$\alpha = \arccos(V_1 . V_2)$$

Ecuación 6 Ecuación de ángulos entre segmentos.

Por ejemplo, para calcular el ángulo de flexión y extensión de la rodilla (plano sagital) se debe utilizar los datos de los módulos IMUs 2 y 3 y los segmentos adyacentes son S2 Y S3 correspondientes al muslo y pierna. Por lo tanto, el ángulo de la rodilla se obtiene usando la Ecuación 7. [31]

$$\alpha(rodilla) = \arccos(S_2 . S_3)$$

Ecuación 7 ángulo de flexión y extensión de la rodilla. [31]

2.7 Interfaces gráficas de usuario (GUI)

Son un conjunto de pantallas también conocidas como interfaces gráficas de usuario la cual permite el control de botones, menús y ventanas de manera sencilla, con lo cual se elimina la necesidad de aprender un lenguaje de programación, evitando la tarea de escribir comandos para ejecutar una acción o evento, ya que por medio de la *GUI* se puede realizar de manera

sencilla e intuitiva. Comparando con los scripts, en estos los comandos están en un orden preestablecido, mientras que en la *GUI* no lo están. Una vez que se ejecuta la GUI, esta permanece en la pantalla, aunque se haya terminado la ejecución del script. La interacción con el usuario continúa hasta que se cierra la *GUI*. [4]

El diseñador de la interfaz del usuario puede hacer uso de importantes recursos visuales y multimediales, tales como imágenes, videos, sonido, fotos. Pero, será indispensable analizar y aplicar normas de diseño específicas para lograr un mejor aprovechamiento del espacio, forma, color y contenido. [32]

2.7.1 Objetivos de la Interfaz del Usuario

Por ser una componente que dialoga con seres humanos, se le exige cualidades como simpleza, amigabilidad, naturalidad, flexibilidad y otras características que afectan el grado de utilidad de la interfaz y por consiguiente la productividad general del resto del sistema interactivo. La conjunción de todas las cualidades exigidas a la componente de diálogo es sintetizada con el término de "Usabilidad". El estándar ISO/IEC 9241 define a la usabilidad como la efectividad, eficiencia y satisfacción con la que un producto alcanza sus objetivos específicos para los usuarios. El ISO/IEC 9126 hace énfasis en las características internas y externas de las aplicaciones que actúan en la usabilidad. [32]

Se debe tomar en cuenta para lograr un alto grado de satisfacción y conformidad por parte del cliente o usuario los siguientes objetivos expuestos en la Tabla 10. [32]

Tabla 10 Objetivos de la Interfaz del Usuario. [32]

Objetivo	Descripción
Simplicidad	El sistema interactivo debe ser simple de instalar, de aprender, de usar, de configurar. Al usuario no se le puede exigir un entrenamiento sobre Computación para poder utilizar el sistema, ni que se someta a posteriores cursos de aprendizaje para interactuar con la interfaz.
Confiabilidad	Este objetivo hace referencia a que la interfaz del usuario debe ser consistente tanto en su forma de expresión como de presentación. No puede contener diálogos ambiguos, que lleven a falsas interpretaciones o que genere dudas al usuario. Tampoco puede existir

	desorganización en la visualización de los datos ni heterogeneidad en los mecanismos de utilización de los servicios provistos.
Flexibilidad	La misma debe ser tolerante a un cierto grado de error que pueda presentar el usuario y condescendiente ante equivocaciones frecuentes efectuadas por él. Puede ayudar brindando mecanismos de sugerencias o de corrección automática, más que informarle un mensaje de error. No puede comportarse como un agente interlocutor controlador, rígido e inflexible, debe ser abierta y aceptar los tiempos y las formas que el usuario requiera.
Transparencia	En todo momento, la interfaz debe comunicarle al usuario sobre los detalles de la transacción y de los cómputos realizados, debe indicar el porcentaje de realización, índices del progreso e información del tiempo que falta para culminar la función que esté ejecutando. También debe proveer mecanismos para poder interrumpir, suspender o deshacer lo efectuado por el sistema y alcanzar un estado previo deseado por el usuario.
Ergonomía	Este objetivo se refiere a la capacidad que presente la interfaz para amoldarse, adaptarse al estilo propio del usuario, brindando mecanismos de configuración y personalización.

Capítulo 3

Diseño y Programación.

3.1 Introducción

En este capítulo se expondrá el diseño y código usados en el desarrollo del Sistema de adquisición de datos e interfaz gráfica de la marcha humana, de esta forma se revisarán las principales características del sistema y sus funciones al igual que se realizará una explicación más detallada a nivel de función interna y restricciones.

3.2 Módulos

Los elementos físicos usados en este proyecto constan de 4 módulos inerciales los cuales toman los datos y ángulos articulares de cada sección de la pierna una pierna a la vez y un módulo maestro siempre conectado al computador el cual se encarga de la recolección, activación, comunicación y envió de los datos obtenidos por los módulos iniciales o esclavos.

3.2.1 Módulo maestro

Este módulo está compuesto por un *Arduino Nano* y un módulo *HC-05*, ya que este módulo es únicamente encargado de la comunicación entre los módulos inerciales y la computadora, ver Figura 33.

Figura 33 Módulo Mastro(Fuente propia)

El módulo maestro es el encargado de dar la activación a cada módulo inercial, por medio de comandos AT provenientes desde programación en el *Arduino Nano*, conectando

módulo a módulo hasta recibir una confirmación de funcionamiento de todos los módulos implicados en la recolección de datos.

Posteriormente el módulo maestro se encargará de la recolección y envió al ordenador de los datos provenientes de cada módulo inercial, el módulo requiere estar conectado directamente en la computadora en la cual se ejecutará la interfaz, por ello no requiere alimentación externa ni módulo de memoria externa.

Sus conexiones son: del módulo *HC-05* respecto al *Arduino Nano* son pin del módulo RXD a pin digital 10, TXD a pin digital 11, GND a GND, VCC a pin digital 8 y EN a pin digital 4. Estos dos últimos serán los que permitirán la activación y configuración del módulo por comandos AT, ver Figura 34.

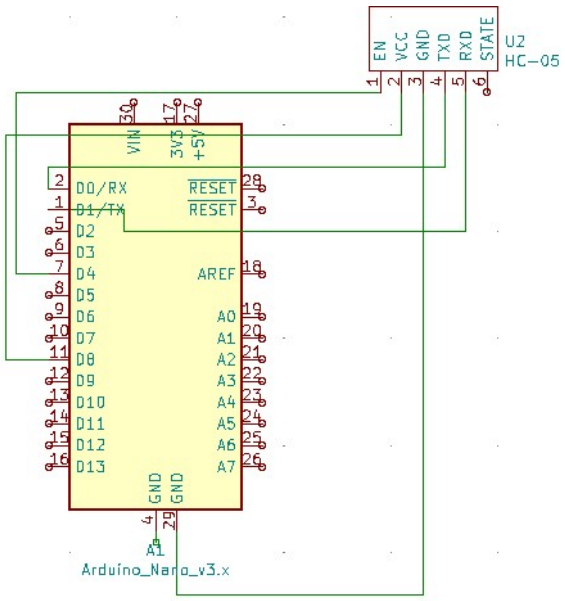


Figura 34 Esquema de conexión módulo maestro. (Fuente propia).

El código se divide en 3 partes fundamentales, la primera es la declaración de variables e inclusión de librerías estas librerías permitirán el correcto funcionamiento de los elementos. Se usaron las librerías: *SoftwareSerial.h* la cual realiza la comunicación Serial del módulo *HC-05* con el *Arduino Nano* por medio de comunicación serial, ver Figura 35.

Figura 35 Datos iniciales y librerías módulo maestro. (Fuente propia)

La segunda parte del código se encarga de la inicialización del módulo *HC-05* el cual permanece apagado hasta que se emita una activación por parte del usuario de la interfaz, el cual por medio de una selección de datos envía el tiempo de función de los módulos esclavos, ver Figura 36.

Figura 36 Detección y selección inicial. (Fuente propia)

La tercera y última parte del código se divide en varios procesos el primero es la configuración por comandos AT módulo por módulo al realizar una especificación de la dirección ADDR la cual es única para cada *HC-05*, de igual manera el módulo *HC-05* es controlado tanto su alimentación como su pin de habilitación para permitir una fácil entrada y salida del modo AT de este ver Figura 37.

Figura 37 Comandos AT y cambio de modo. (Fuente propia)

Posterior a este proceso el módulo espera una confirmación por parte del módulo esclavo con la finalidad de guardar el tiempo en el cual se conectó para mantener de manera controlada los tiempos de inicio de trabajo de cada módulo, dicha información será enviada a el programa ejecutable en el ordenador, ver Figura 38.

Figura 38 Código modo 0, estado de conexión inicial. Fuente propia)

El módulo repetirá los eventos especificados anteriormente módulo por módulo hasta llegar al último esclavo con el cual mantendrá una conexión fija mientras el tiempo de función culmine, una vez dicho tiempo haya terminado el módulo maestro recibirá un carácter único por parte del módulo inercial, el cual además de indicar el fin de la toma de datos también cambiará el modo de función del maestro en el cual se procederá a la recepción de datos y tiempos, ver Figura 39.

Figura 39 Código modo 1, recolección y reconexión módulo a módulo(Fuente propia)

Una vez el módulo maestro termina la recolección de datos de un módulo regresa en el orden inverso en el cual se conectó repitiendo el proceso hasta terminar con el primer módulo.

3.2.2 Módulo Inercial

Este módulo está compuesto por un *Arduino Nano*, un módulo *HC-05*, un sensor *MPU6050*, un módulo lector/escritor de microSD, una memoria Kingston de 8G y una batería de 9V la cual se encargará de alimentar todo el sistema, ver Figura 40.

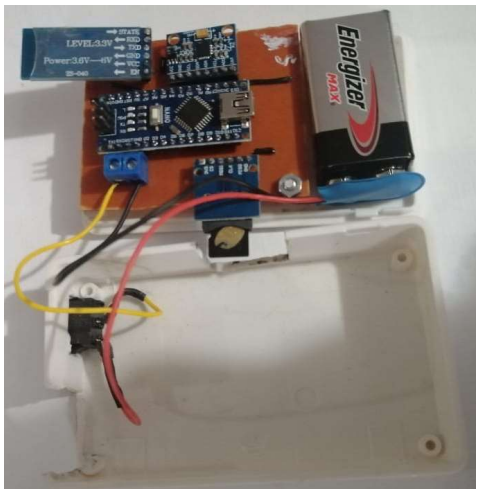


Figura 40 Módulo Inecial. (Fuente propia)

Sus conexiones son, ver Figura 41:

- Módulo *HC-05* respecto al *Arduino Nano* son pin del módulo RXD a pin TX1, TXD a pin RX1, GND a GND , VCC a pin digital 8 y EN a pin digital 4.
- Sensor MPU60-50 respecto al *Arduino Nano* son VCC a pin 5V, GND a GND, SLC a pin analógico A5, SDA a pin analógico A4, ADO a pin digital 2 y INT a GND.
- Módulo lector/escrito microSD respecto al *Arduino Nano* son VCC a pin 3.3V , GND a GND, CS a pin digital 10, MOSI a pin digital 1, CLK a pindigital 13 y MISO a pin digital 12.

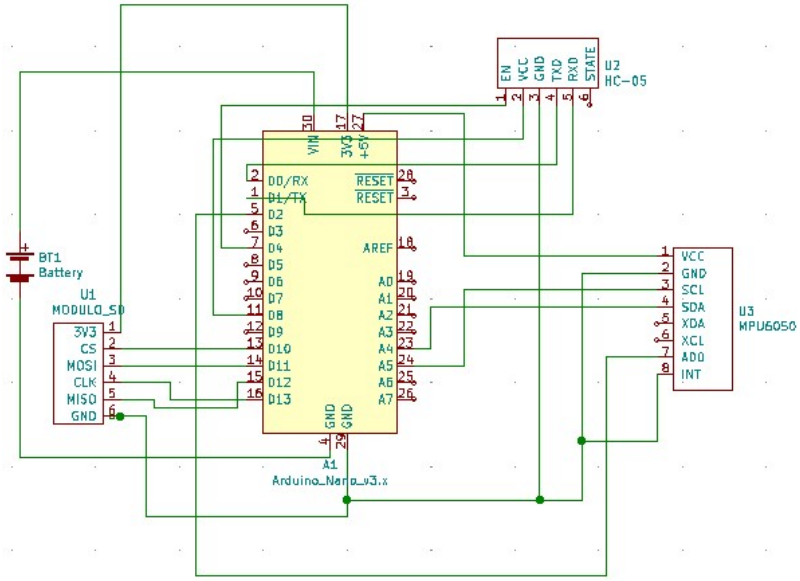


Figura 41 Esquema de conexión módulo inercial. (Fuente propia)

Este módulo es el encargado de procesar los datos del sensor de manera parcial, transformado los datos de aceleración que entrega al sensor en ángulos de navegación, por medio de las ecuaciones presentadas en capítulos anteriores, posterior a esto dichos datos serán almacenados en la tarjeta microSD para su posterior envío.

El código se divide en 3 partes fundamentales, la primera es la declaración de variables e inclusión de librerías estas librerías permitirán el correcto funcionamiento de los elementos. Se usaron las librerías: *SoftwareSerial.h* la cual realiza la comunicación Serial del módulo *HC-05* con el *Arduino Nano* por medio de comunicación serial, las librerías *SPI.h*, *SD.h* para la comunicación entre el módulo lector/escritor microSD y el *Arduino Nano*, las librerías *I2Cdev.h*, *MPU6050.h* y *Wire.h* para la conexión entre el sensor *MPU6050* y el *Arduino Nano*, Todas las librerías se encuentran incluidas en el desarrollador de Arduino, ver Figura 41.

Una parte fundamental es la variable muestras puesto que esta definirá el tiempo de función de cada módulo dicha variable cambiará dependiendo del módulo a tratar con la finalidad de que todos los módulos tomen los datos en el mismo lapso de tiempo, ver Figura 42.

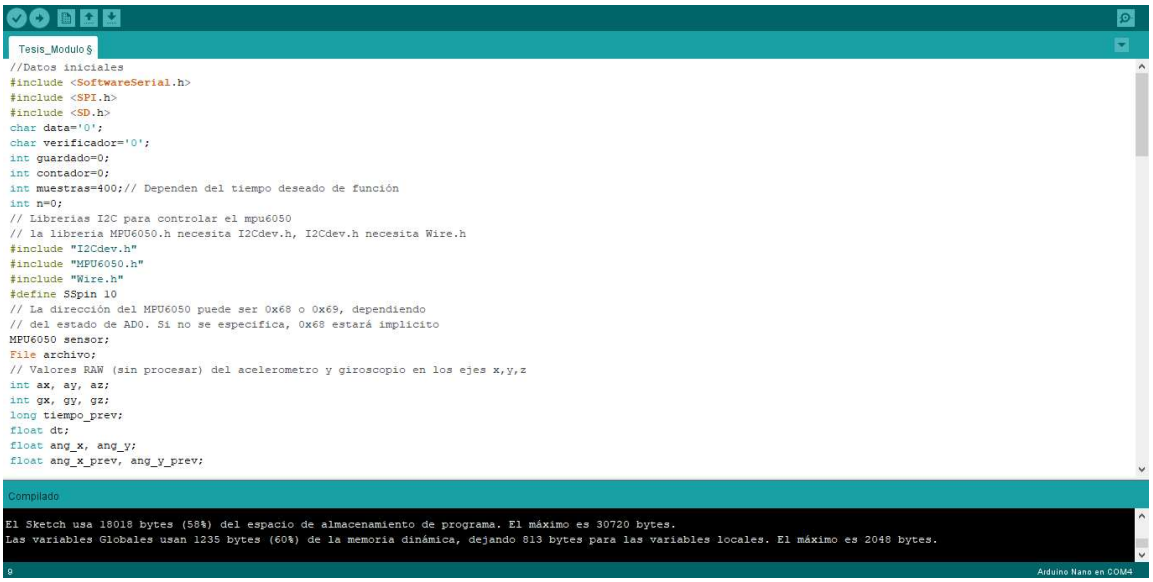


Figura 42 Datos iniciales y librerías. (Fuente propia)

La segunda sección del código es el void setup en el cual se activarán y se comprobará la correcta función de todos los elementos utilizados en los módulos, ver Figura 43.

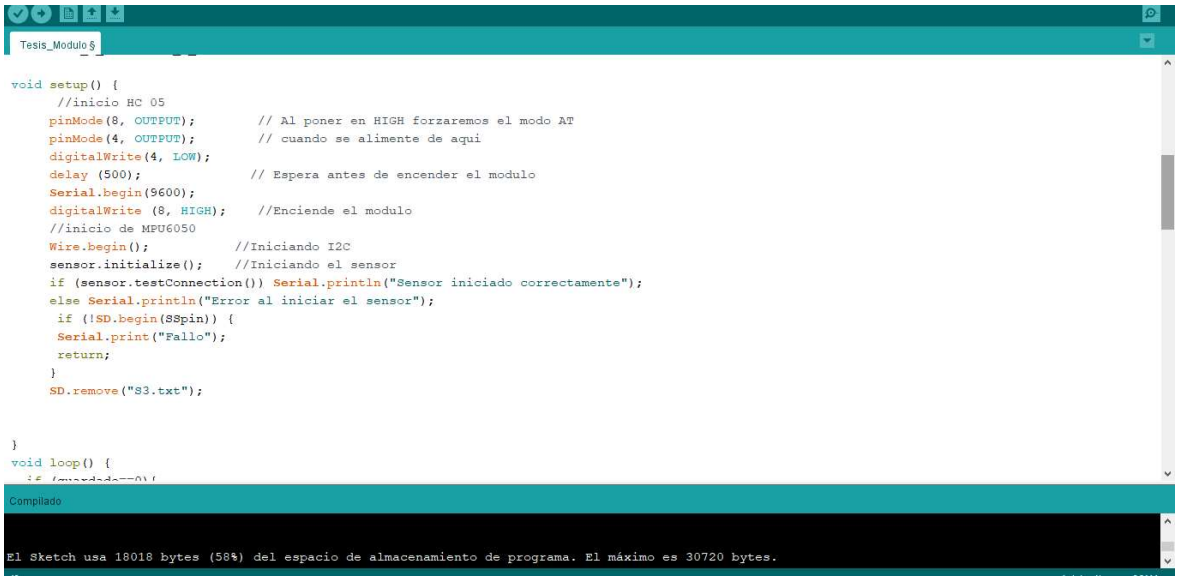


Figura 43 Inicialización y comprobación de elementos. (Fuente propia).

La última sección es el void loop de función continua el código, comienza con una comprobación de conexión con el módulo maestro y posterior a esto toma los datos y los procesa, los datos de las aceleraciones se convierten en ángulos de navegación, pasado una cantidad de muestras especificadas por módulo empieza a guardar los datos en la memoria

externa por medio del módulo microSD, esas muestras que no se consideran en el estudio es debido a que al iniciar el módulo siempre requiere de un pequeño lapso de estabilización y esas muestras son desechables puesto son datos erróneos. Posterior a la finalización del guardado el módulo entra a un modo de espera hasta recibir una confirmación del módulo maestro para el envió de los datos, el cual se realiza carácter por carácter, ver Figura 44 y 45.

Figura 44 Código de toma y manejo de datos angulares parte 1. (Fuente propia)

Figura 45 Código de toma y manejo de datos angulares parte 2. (Fuente propia)

3.3 Interfaz gráfica

La interfaz gráfica denominada *TMOVE* se desarrolló en lenguaje *Python* con la extensión *Tkinter* la cual permite desarrollar programas de escritorio con un grado de complejidad bajo o medio, la interfaz se puede dividir en 2 partes fundamentales, ver Figura 46.

Figura 46 Interzasgrafica TMOVE v.1.0(Fuente propia)

El código de función y ubicación de los elementos fue basado en programación orientada a objetos, además de emplearse el uso de varias librerías usadas en diversas áreas del programa para el diseño de elementos se usó la librería *Tkinter* la cual viene por defecto en *Python*, ver Figura 47 y 48.

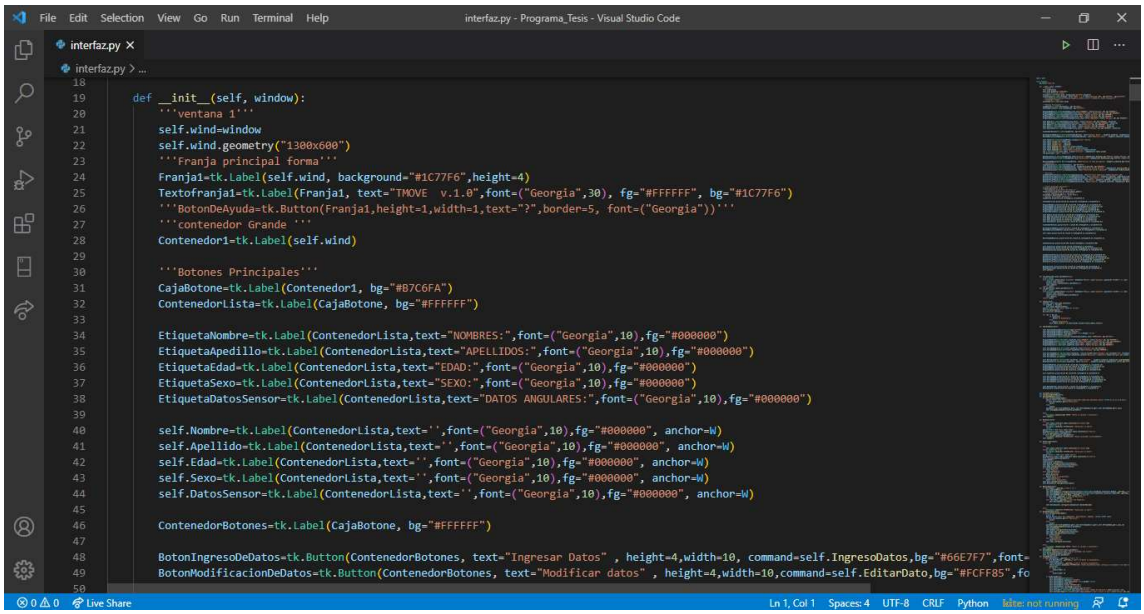


Figura 47 Declaración de elementos ventana principal. (Fuente propia)

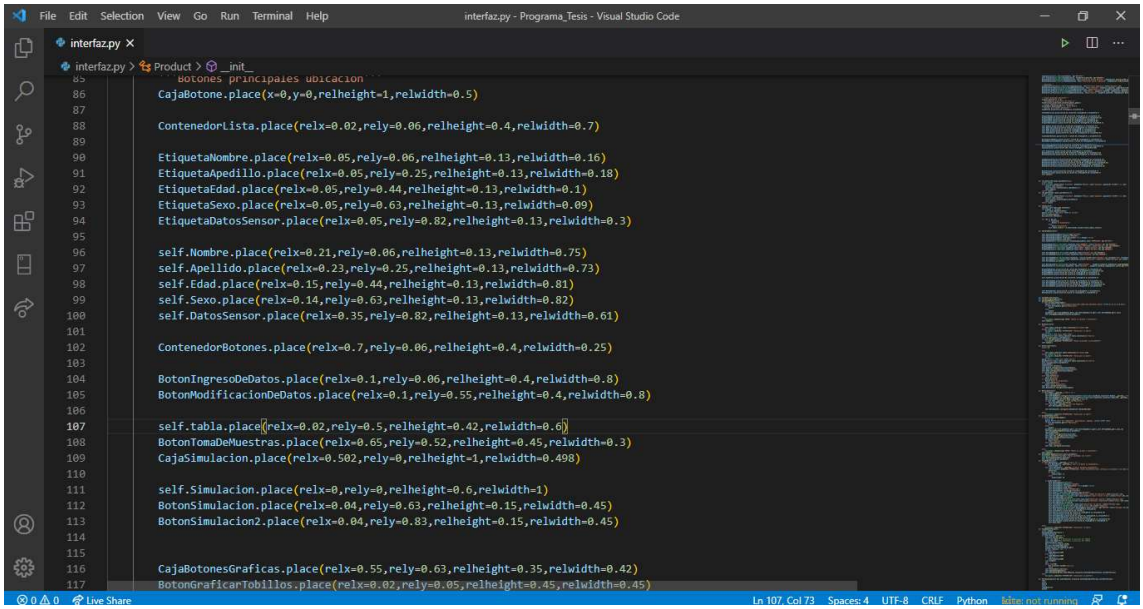


Figura 48 Ubicación de elementos de la ventana principal. (Fuente propia)

La interfaz se puede dividir en 2 partes para una mejor comprensión del programa y función, la parte izquierda es encargada de la gestión de datos tanto los datos del sujeto de estudio como la toma de datos angulares cada botón tiene una función específica y única con un orden que se puede apreciar de manera lógica. Como primer punto se aprecia el botón de ingreso

de datos el cual despliega un formulario en una nueva ventana, en la cual el usuario podrá ingresar los nombres, apellidos, edad y sexo del sujeto de estudio, ver Figura 49.

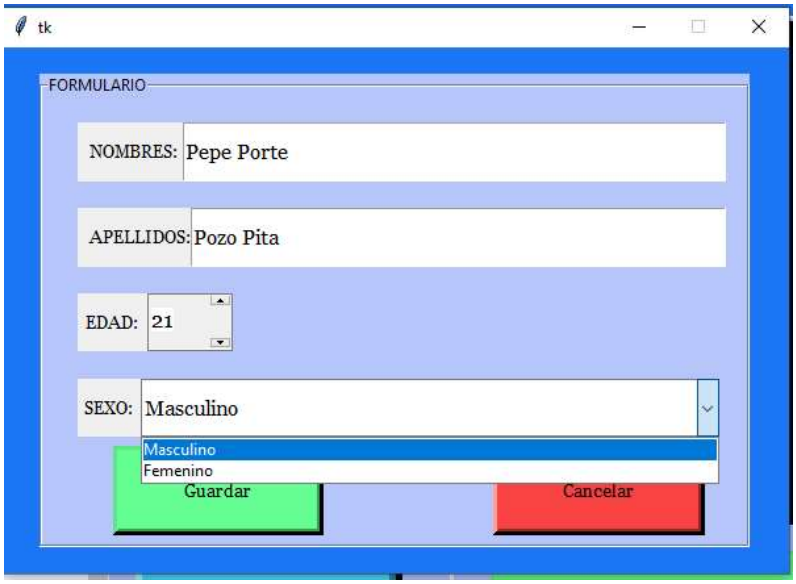


Figura 49 Formulario ingreso de datos sujeto de estudio. (Fuente Propia)

Una vez ingresado los datos estos son procesados y enviados a una base de datos *PostgreSQL* con un hostlocal, y a su vez en la parte grafica se indicará un mensaje de guardado exitoso de ser el caso, ver Figura 48.

Figura 50 Mensaje guardado exitoso. (Fuente propia)

En cuanto al código de función se divide en la parte gráfica y la parte lógica la parte gráfica se encargará de todos los elementos que ve el usuario y permitirá la interacción, mientras la parte lógica se encarga del guardado y procesamiento de los datos, ver Figura 51 y 52.

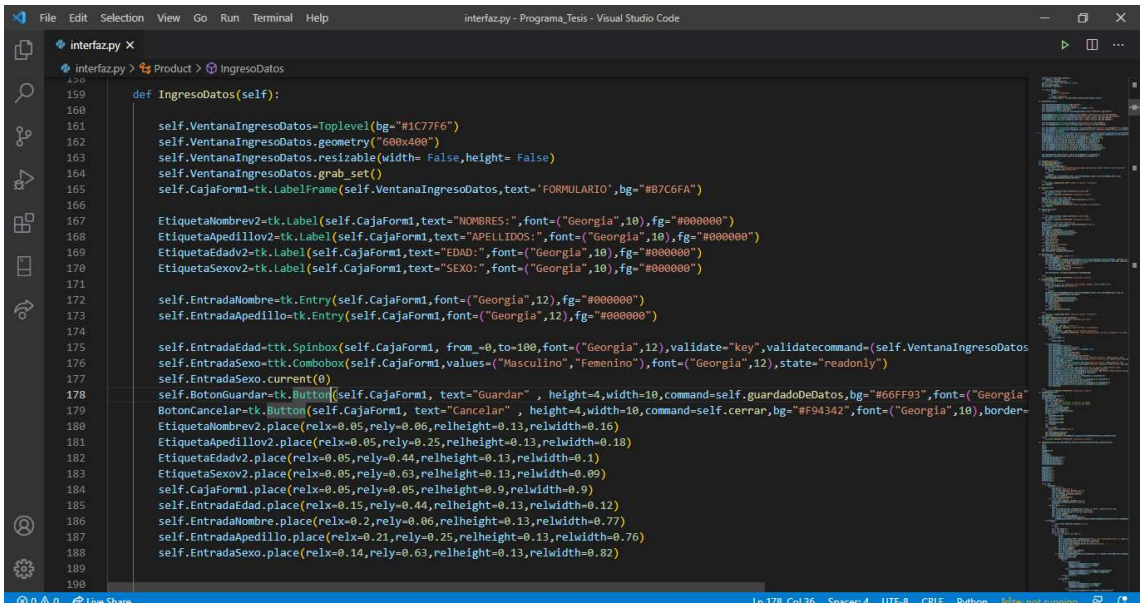


Figura 51 Declaración de elementos ventana de ingreso de datos. (Fuente propia)

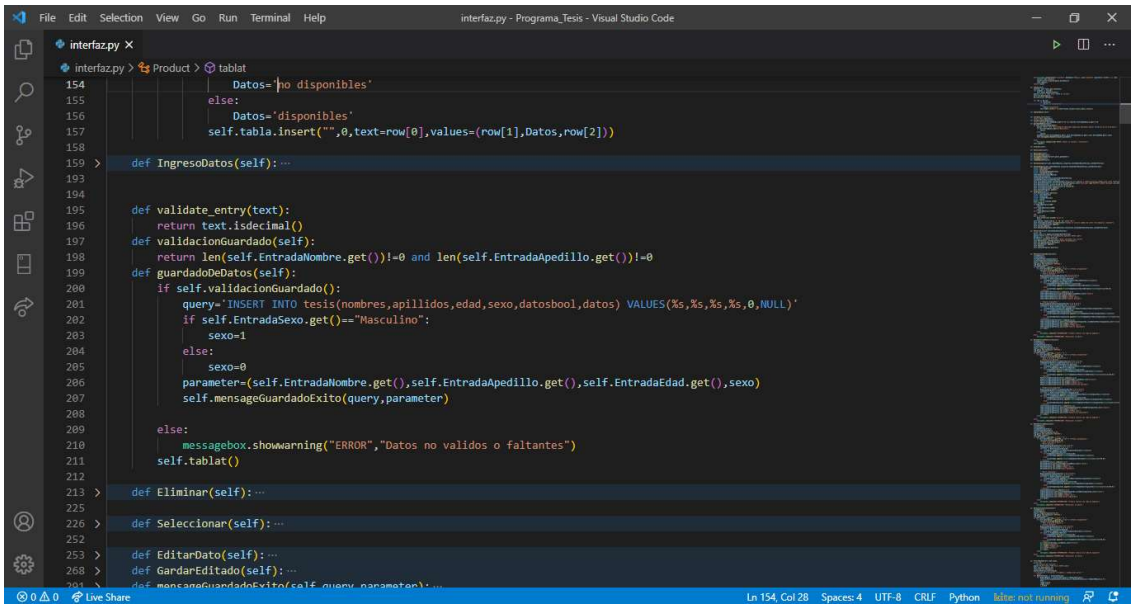


Figura 52 Procesamiento de datos y guardado. (Fuente propia)

Todos los datos ingresados por el usuario son mostrados por una tabla de datos ubicada en la parte izquierda de la ventana principal la cual será actualizada cada que un evento relacionado con los datos del sujeto de prueba cambie o se ingresen nuevos datos, ver Figura 53.

Figura 53 Tabla de datos de sujetos de prueba. (Fuente propia)

El código de la tabla datos se centra en traer todos los datos disponibles de la base de datos con el fin de mostrar cualquier modificación o cambio que estos sufran, ver Figura 54.

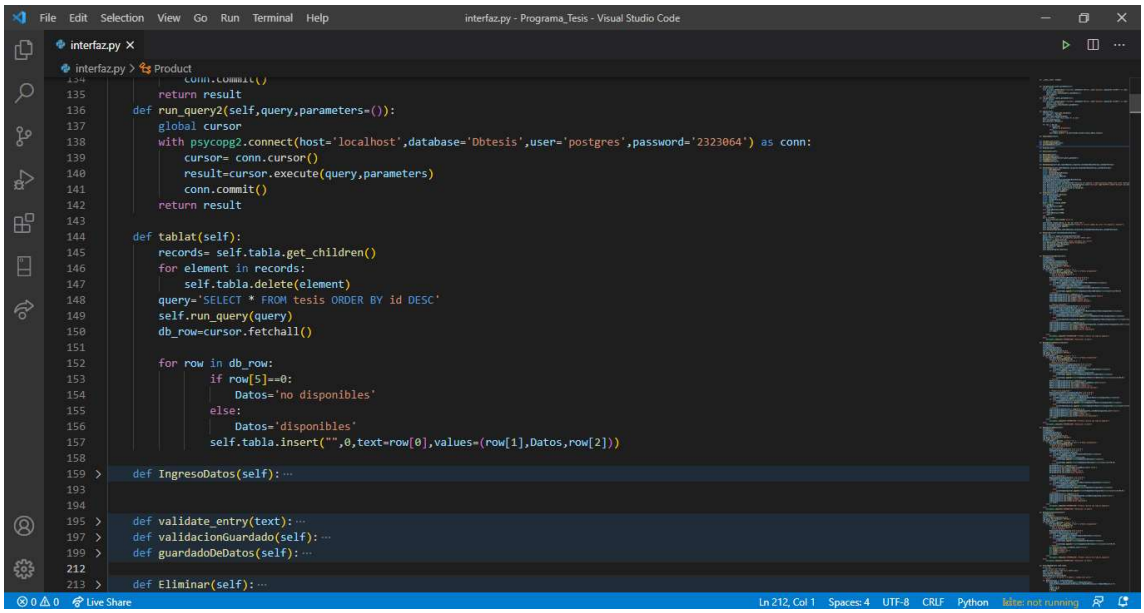


Figura 54 Código tabla de datos. (Fuente propia)

En la tabla de datos se puede apreciar 2 botones uno con una función de eliminar datos y el otro de seleccionar la información de un sujeto de prueba, la cual se mostrará en la parte superior izquierda y a su vez habilitará el resto de componentes, si no se selecciona ningún

registro el programa no permitirá que ningún botón se ejecute, evitando errores de especificación, ver Figura 55 y 56.

Figura 55 Datos de registro seleccionado. (Fuente Propia)

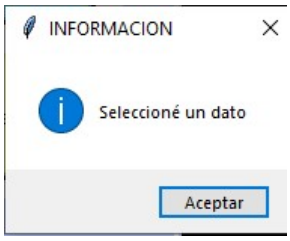


Figura 56 Mensaje datos no seleccionados. (Fuente propia)

El código del botón de selección se centra en captar el ID especificado por registro y posteriormente hace una consulta a la base de datos y actualiza la información en la ventana principal con el fin de hacer el programa más dinámico y entendible, mientras el botón de eliminar simplemente elimina por completo el registro, ver Figura 57.

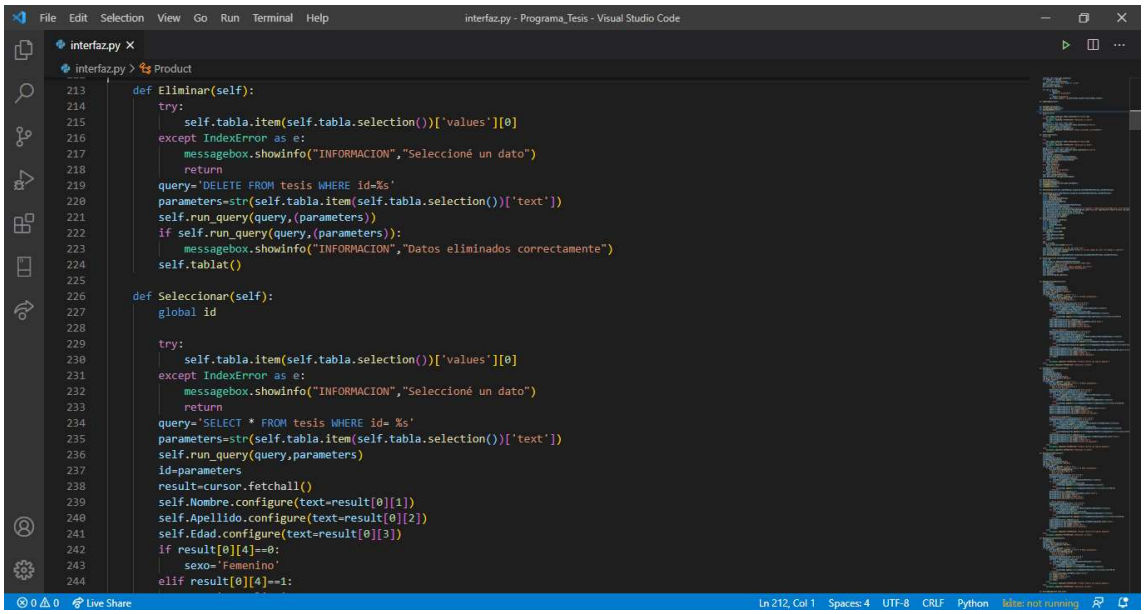


Figura 57 Código de eliminar y seleccionar un registro. (Fuente propia)

Siguiendo con la secuencia de función, si el usuario requiere una modificación en el registro y una vez haya seleccionado el registro objetivo el botón de modificar registro se habilitará, desplegando el mismo formulario de ingreso de registro, pero con los datos pre cargados permitiendo así una fácil y rápida modificación, ver Figura 58.

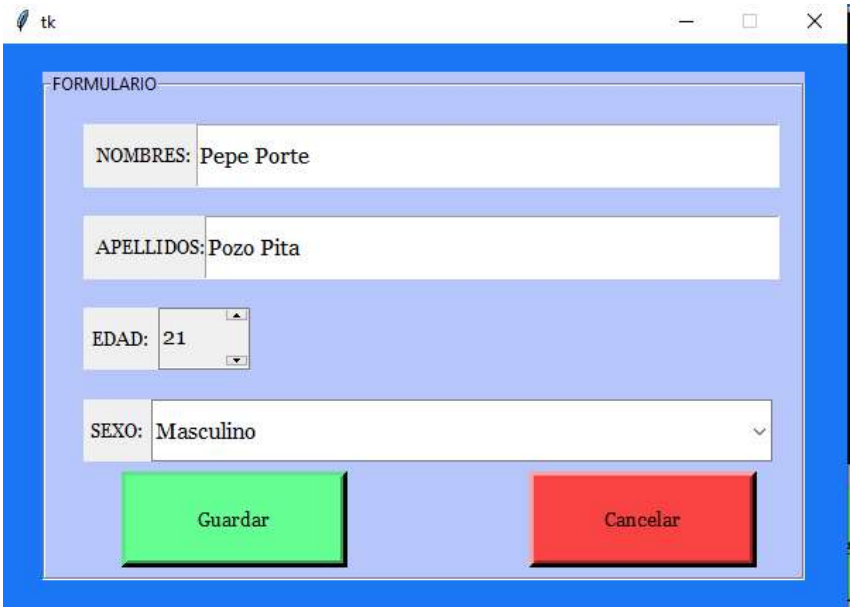


Figura 58 Ventana modificación de registro. (Fuente propia)

El código de modificación de datos se encarga en primera instancia de realizar una consulta con el fin d presentar los datos en el formulario, para su posterior modificación y a su vez reutiliza el método de guardado del registro del formulario solo que modifica el query de inserción de nuevo dato a modificación de este, ver Figura 59.

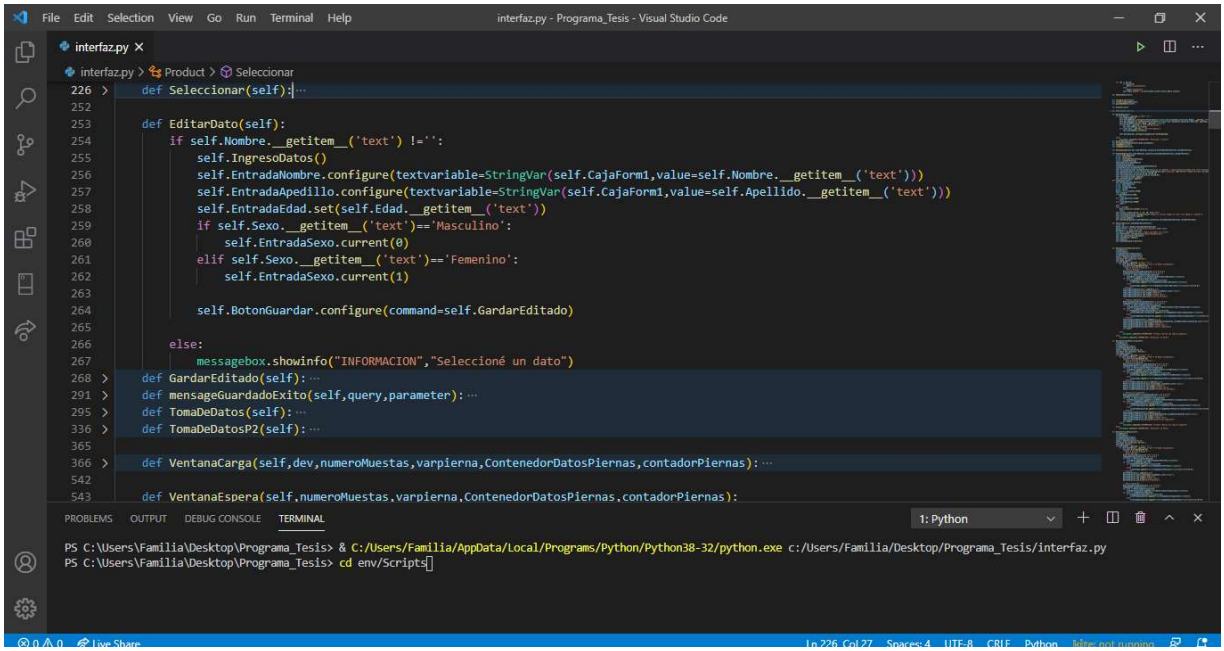


Figura 59 Código modificación de registro. (Fuente propia)

Una vez terminada en ingreso del registro y su selección ya puede realizarse a toma de los datos angulares, para esto se debe conectar el módulo maestro a un puerto de la computadora, el programa realizará una detección automática de puertos conectados, si no conecta el maestro en un puerto USB el programa no lo dejará continuar, una vez cumplidos los requisitos se desplegará una nueva ventana en la cual podrá seleccionar el tiempo de función, la pierna con la cual iniciará la recolección de los datos y el puerto USB (deberá seleccionar uno si tiene más de 1 elemento conectado), ver Figura 60 y 61.

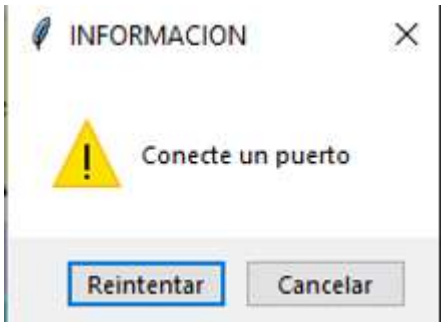


Figura 60 Mensaje maestro desconectado. (Fuente propia)

Figura 61 Ventana selección de puerto y parámetros. (Fuente propia)

El código de comprobación de puertos y el que permite la conexión seria con el módulo maestro fue posible gracia a la librerías *serial.tools*.list_ports y serial las cuales se pueden descargar libremente de la red, ver Figura 62.

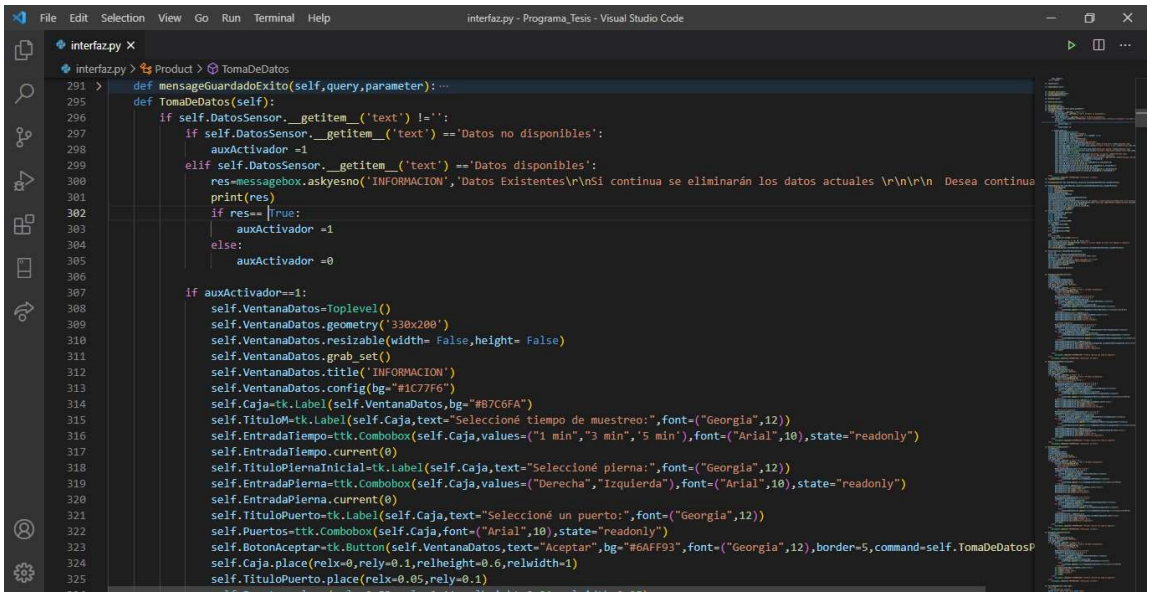


Figura 62 Código ventana de selección de puerto y parámetros de función. (Fuente propia)

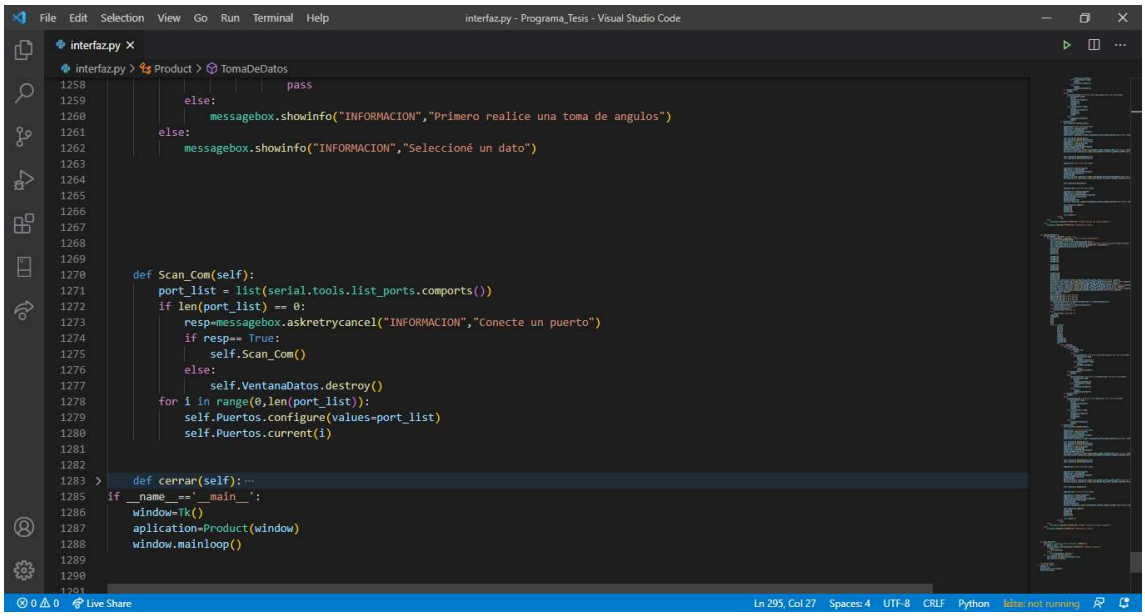


Figura 63 Código de escaneo de puertos. (Fuente propia)

Posterior a la selección de tiempo, pierna y puerto el programa mostrará una nueva ventana de carga, la cual mostrará las indicaciones o estado del proceso por medio de un gráfico y un texto junto a una barra de carga.

Dicha ventana cambiar sus elementos dependiendo del estado del proceso, permitiendo al usuario un mayor control de este, ver Figura 64, 65 ,66 y 67.

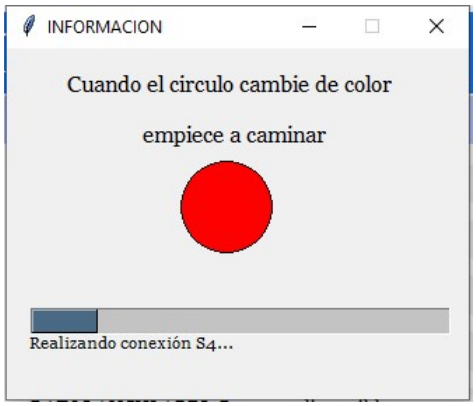


Figura 64 Ventana de carga confirmación de conexiones módulos inerciales. (Fuente propia)

Figura 65 Ventana de carga inicio de la actividad. (Fuente propia)

Figura 66 Ventana de carga lectura de datos. (Fuente propia)

Todos los indicadores gráficos mostrados anterior mente son productos de cambios de eventos en el módulo maestro, dado que la comunicación es vía serial todos los indicadores bajo la barra de carga son recibidos directamente del módulo maestro en tiempo real.

La primera etapa es la encargada de la confirmación de conexión de los módulos inerciales, posterior a la comunicación con cada uno, el maestro entra en un estado de espera permitiendo realizar la actividad del sujeto de estudio, una vez pasado el tiempo de muestreo el módulo maestro empieza un estado de recepción de datos por parte de los módulos inerciales y a su vez envía dichos datos al ordenador para su posterior tratamiento y guardado.

El tiempo de la etapa de transición de datos variara dependiendo del tiempo de estudio, dando como resultado que por cada minuto de toma de ángulos se requiere 2 minutos aproximadamente de transición por módulo, posterior a la recolección de los datos provenientes

de los módulos el programa entra en un estado de pausa mostrando un nuevo indicador, esta vez con un botón de continuación.

En este nuevo estado de pausa se da las indicaciones al usuario para que apague los módulos inerciales y los reubique de manera segura en la otra pierna del sujeto de prueba, una vez la reubicación se realiza el usuario podrá continuar con la toma de datos de la pierna contraria repitiendo el proceso anteriormente especificado, ver Figura 67.

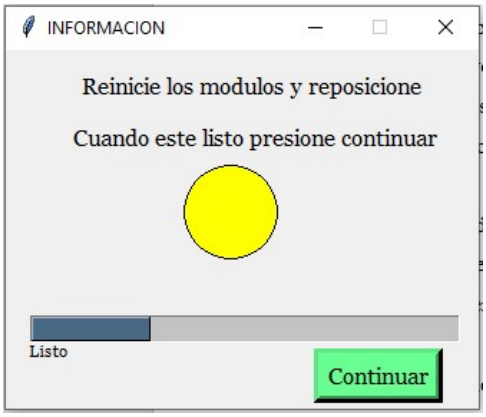


Figura 67 Ventana carga estado espera cambio de pierna. (Fuente propia)

A nivel código ocurren varios hechos importantes, el programa entra en un ciclo de toma de datos mientras los indicadores del módulo maestro sirven como indicadores de cambios de eventos permitiendo que el programa realice las recolecciones y clasificación de datos al igual que se empieza a calcular el retraso entre módulos cortando la señal de los datos entregados con la Ecuación 8.

$$M_D = (T_F - T_M) * 0.02666$$

Ecuación 8 Ecuación muestras sobrantes. (Fuente propia)

Donde:

M_D: número de muestras desechables

T_f: tiempo conexión del primer módulo (último en transmitir las muestras)

T_m: tiempo conexión de los módulos a calcular

La constante 0,026666 fue calculada con una regla de 3 teniendo como parámetros que 1 minuto de función equivale a 1500 muestras, dichos parámetros se guardan en un diccionario para su posterior uso, ver Figura 68.

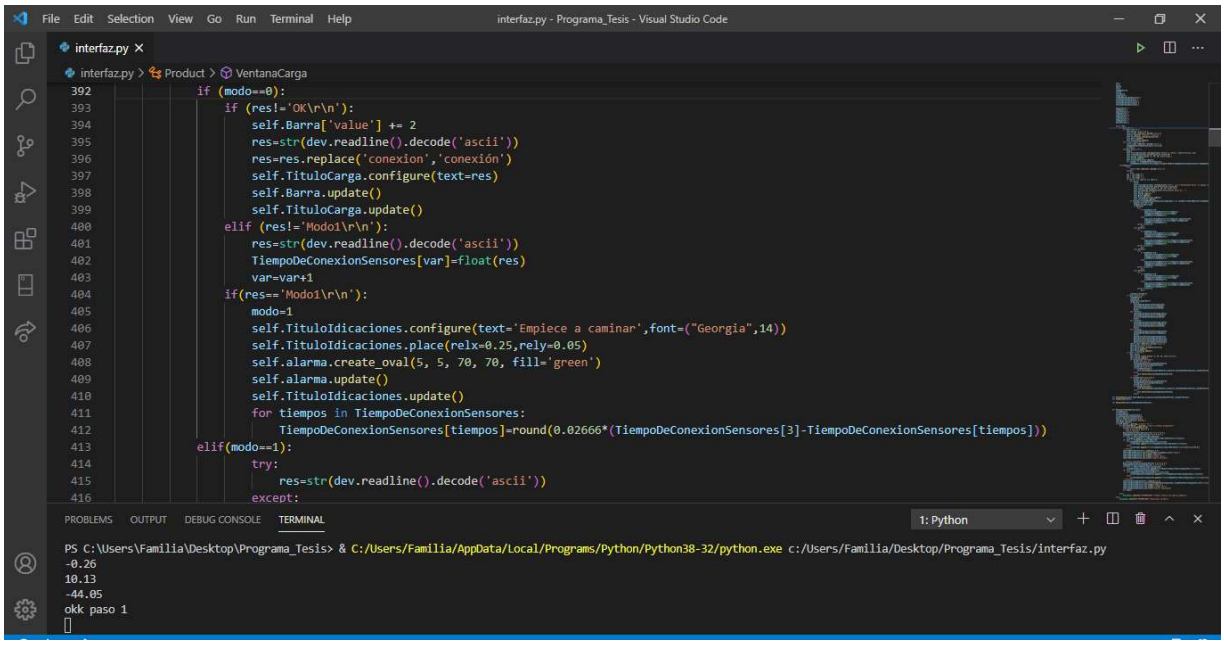


Figura 68 Código modo 0, comprobación de conexión y cálculo de diferencia de muestras. (Fuente propia)

Posterior a la comprobación de conexión empieza un estado de pausa en el cual únicamente se modifica los títulos y color de la señal, en la recepción de datos el programa entra en un ciclo de comprobación de estructura de los datos recibidos desechando datos corruptos o con inconsistencias de formato, además de empezar a guardar los datos de los módulos en un arreglo de diccionarios, ver Figura 69 y 70.

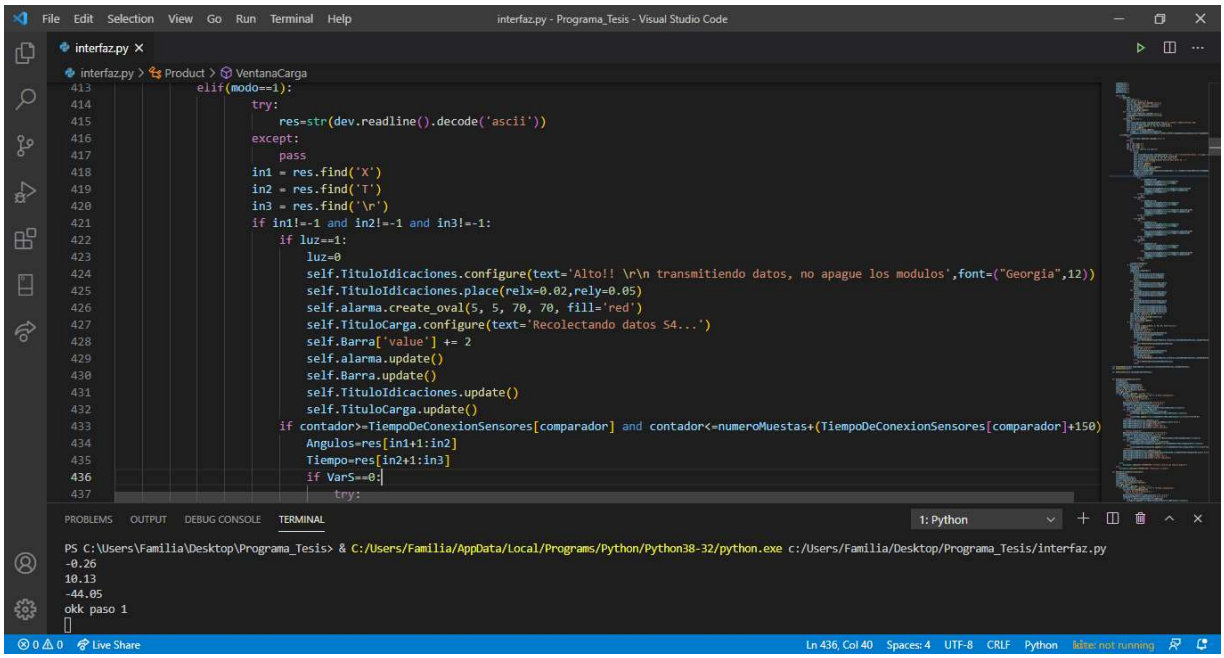


Figura 69 Código verificación de estructura de datos y cambio de indicaciones. (Fuente propia)

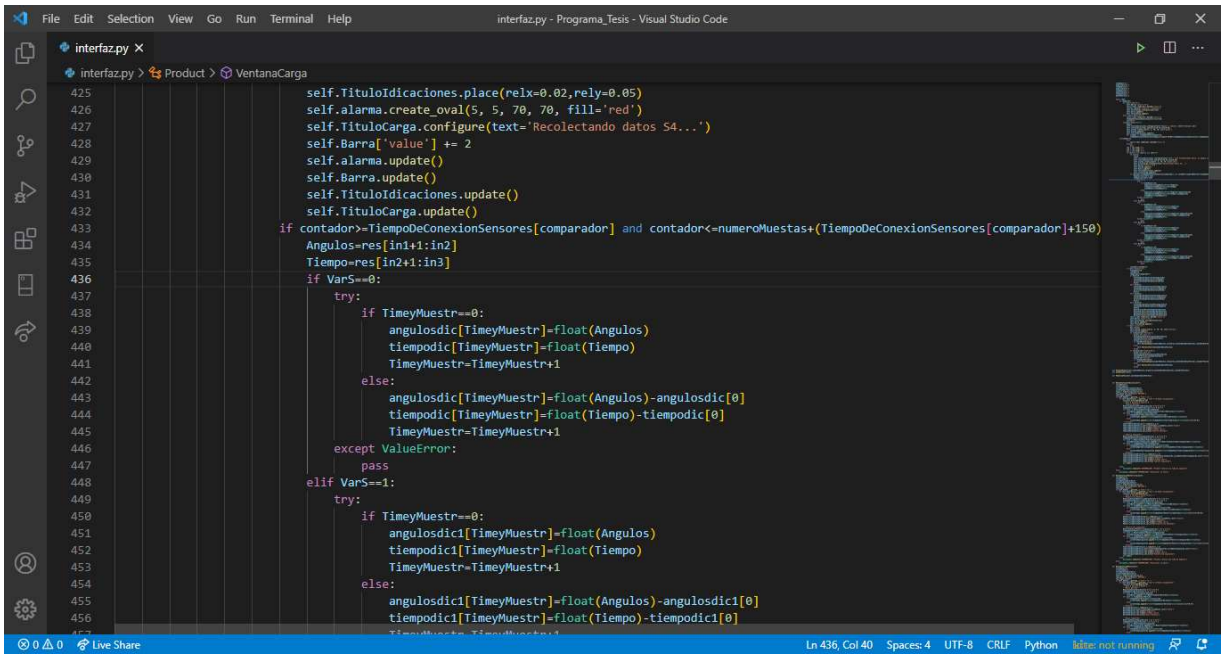


Figura 70 Código guardado y clasificación de datos . (Fuente propia)

Al recibir la señal de finalización de la recepción de datos del último sensor el programa realiza una verificación, la cual consiste en revisar que sea la primera toma de datos y especifica de que pierna son, si es la primera vez en el proceso se redirecciona a la ventana de espera de la

Figura 67, caso contrario los datos son transformados en formato Json para su posterior guardado en la base de datos, ver Figura 71, 72 y 73.

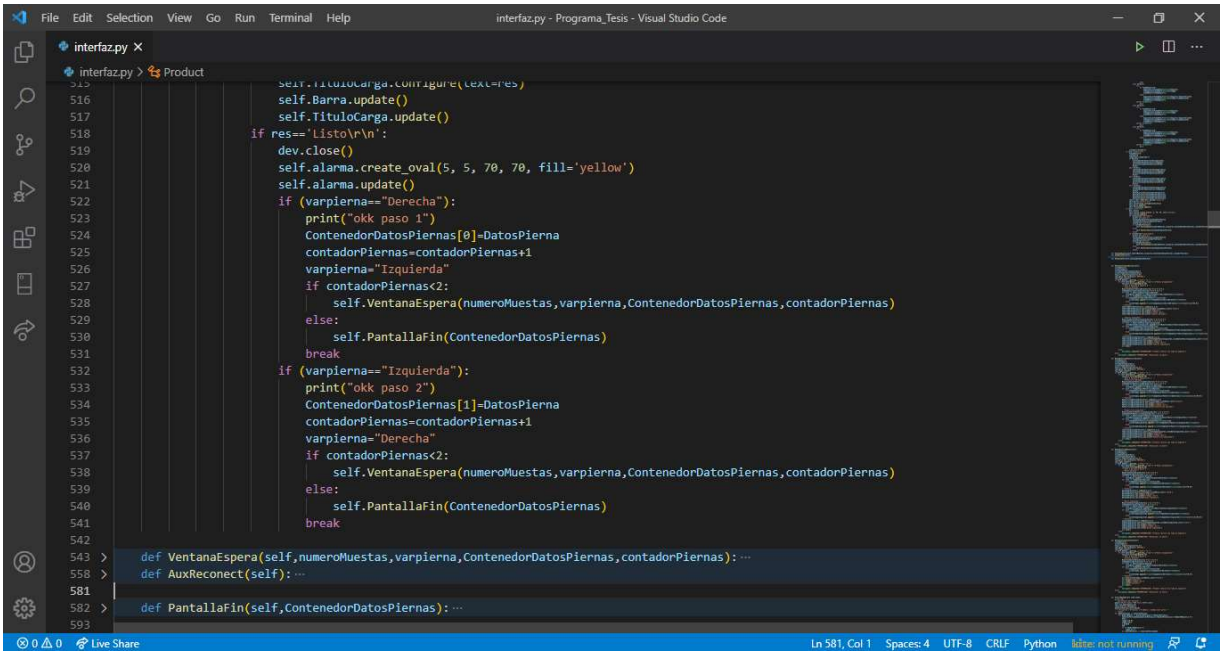


Figura 71 Código verificación de toma de datos. (Fuente propia)

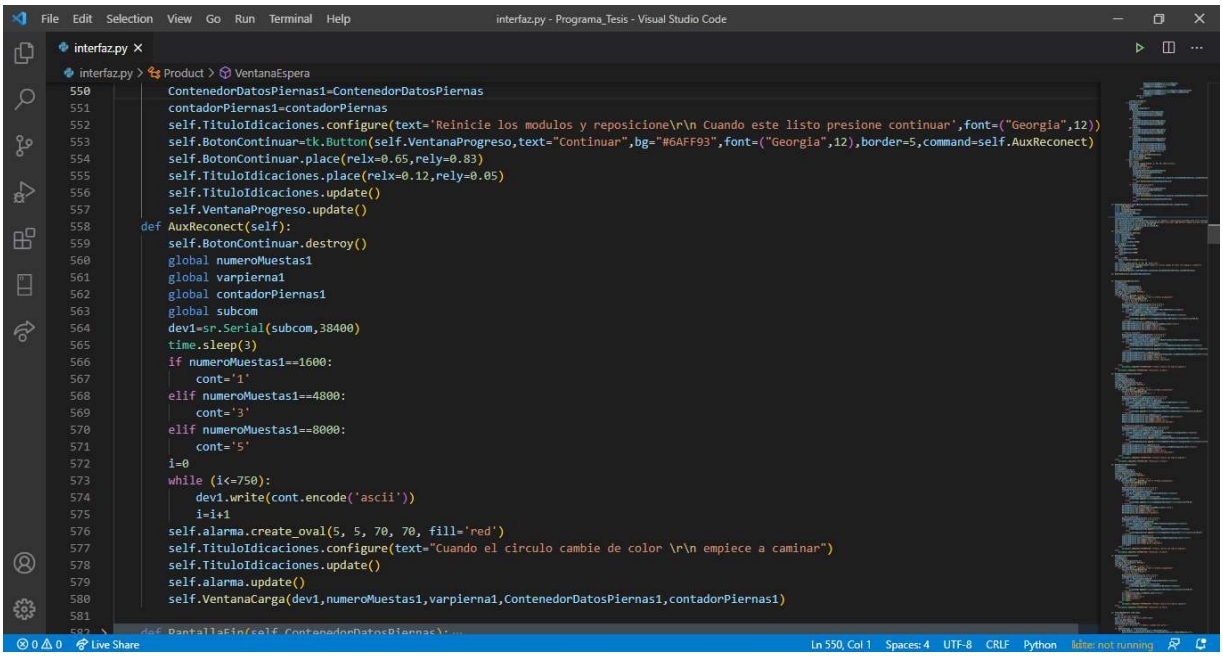


Figura 72 Código diseño ventana de cambio de pierna. (Fuente propia)

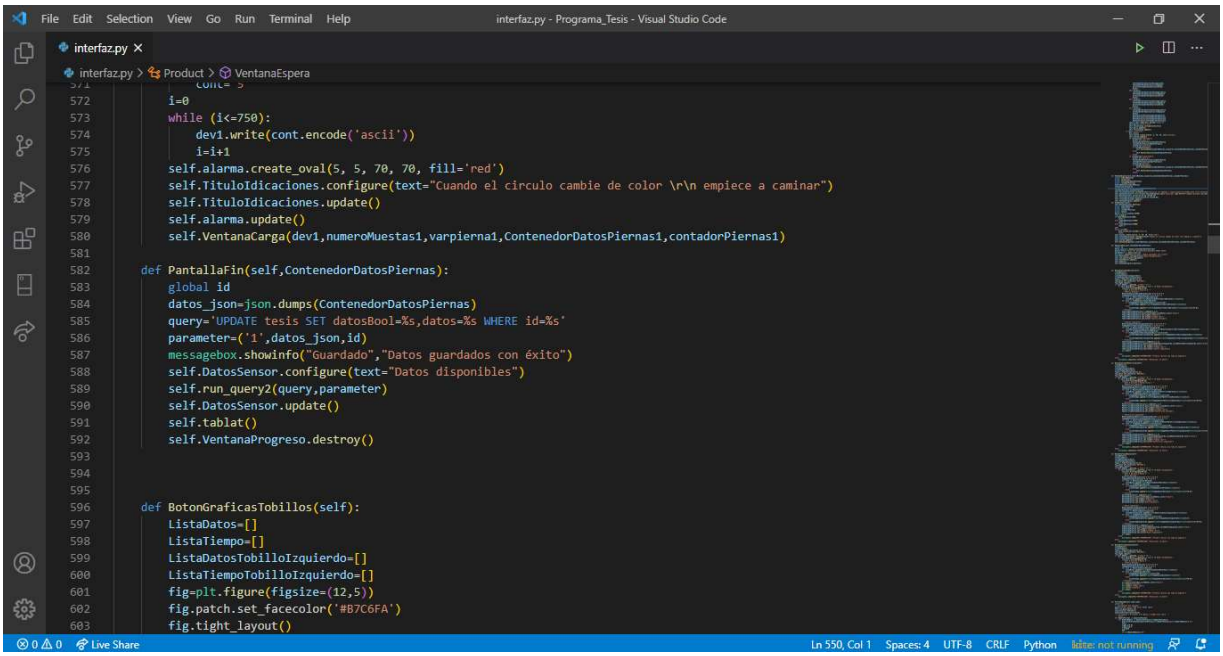


Figura 73 Código cambio de formato y guardado. (Fuente propia)

Una vez se realiza el guardado de datos los botones de las simulaciones y los gráficos se habilitan caso contrario mostrará un mensaje de datos no disponibles, tanto los botones de las gráficas como las simulaciones al ser ejecutados proceden a un filtrado de datos anteriormente guardados.

El filtrado es realizado localmente y no afecta a los datos guardados, el filtro aplicado es el filtro de Kalmad, a su vez se limita la señal en un rango especifico de función puesto que los datos en ocasiones superan el filtro del formato, dando resultados no aceptables para el estudio, ver Figura 74 y 75.

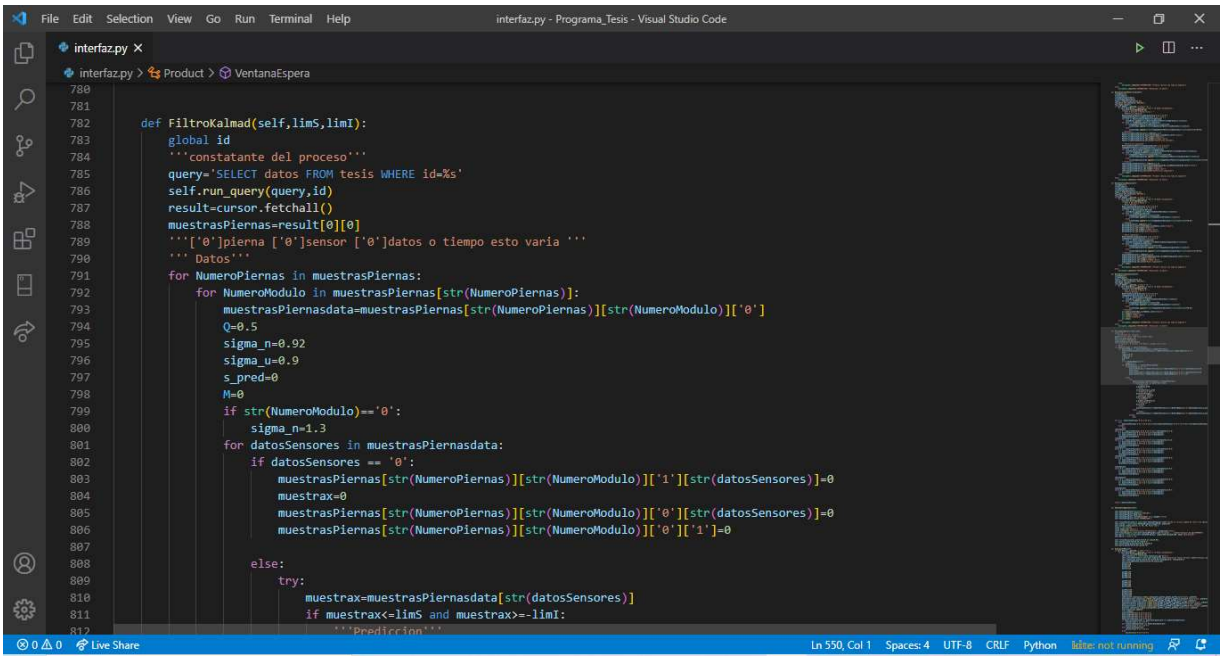


Figura 74 Código filtro de Kalmad. (Fuente Propia)

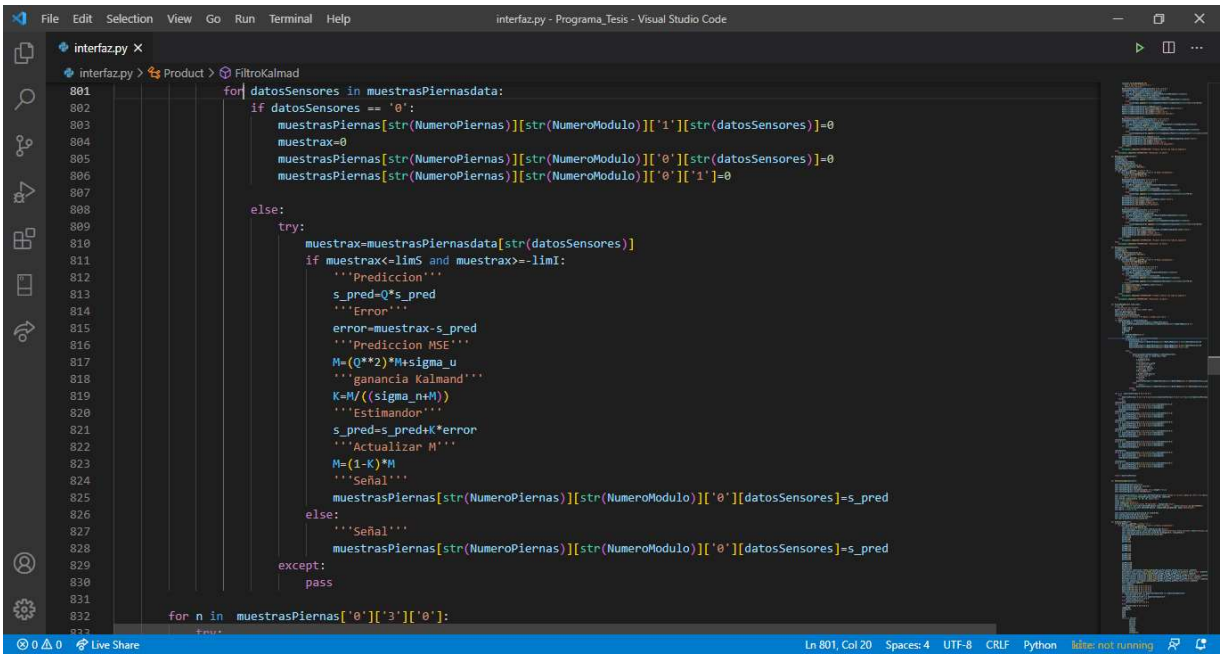


Figura 75 Código filtro de Kalmad. (Fuente propia)

Los datos usados en el filtro fueron ajustados mediante observación excepto el sigmaN el cual fue calculado en una hoja Excel con las muestras sin procesar de uno de los módulos dando como resultado 0.92.

Una vez terminado de procesar los datos de todos los sensores se realiza un corte de señales con la finalidad de disminuir aún más el error producido por interferencias, ruido y retraso entre señales, ver Figura 76.

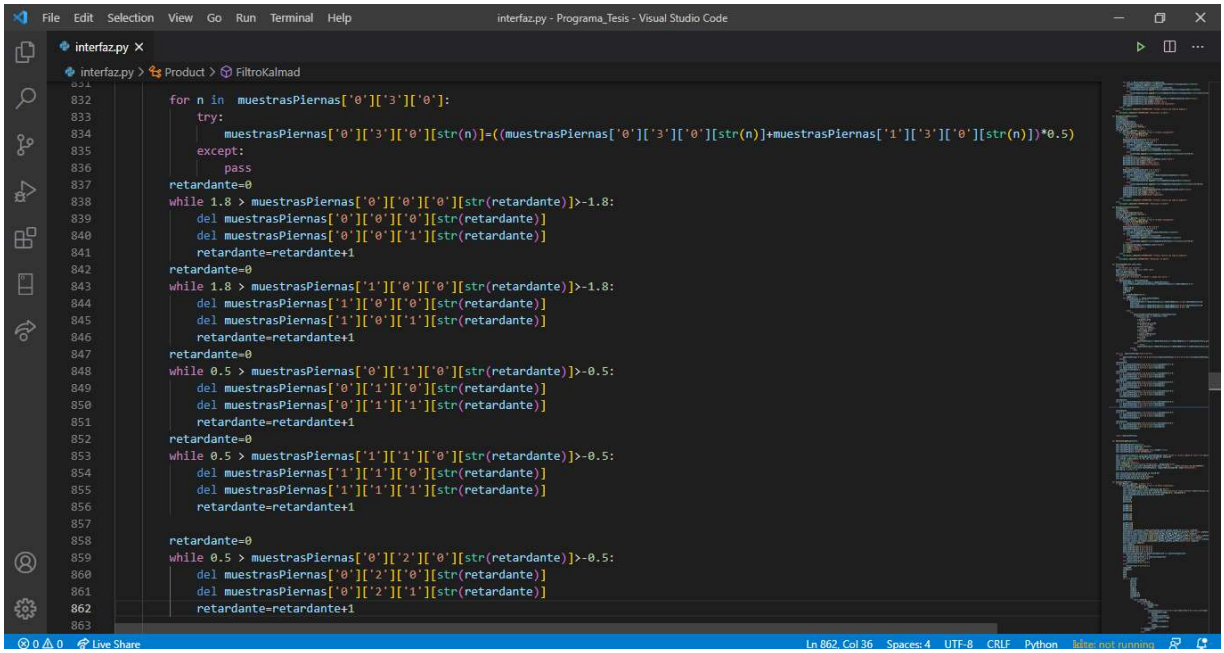


Figura 76 Código corte de señales. (Fuente propia)

Los rangos de las distintas señales fueron tomados de varias pruebas de coordinación de sensores, se realizó una ejecución del programa y se movieron todos los sensores a la vez y en el mismo ángulo dando como resultados los rangos propuestos en el programa.

Las gráficas se pueden realizar de manera sencilla gracias a la librería Matplot la cual se encuentra de manera libre en internet, todas las gráficas llevan el mismo precepto una vez presionado el botón el programa realiza el filtrado de la señal, retornando una señal más clara y precisa, posterior a esto se realiza un cambio de estructura de diccionario a listado con la finalidad de facilitar el uso de las librerías, cada botón imprime los resultados de ambas piernas, ver Figura 77, 78 y 79.

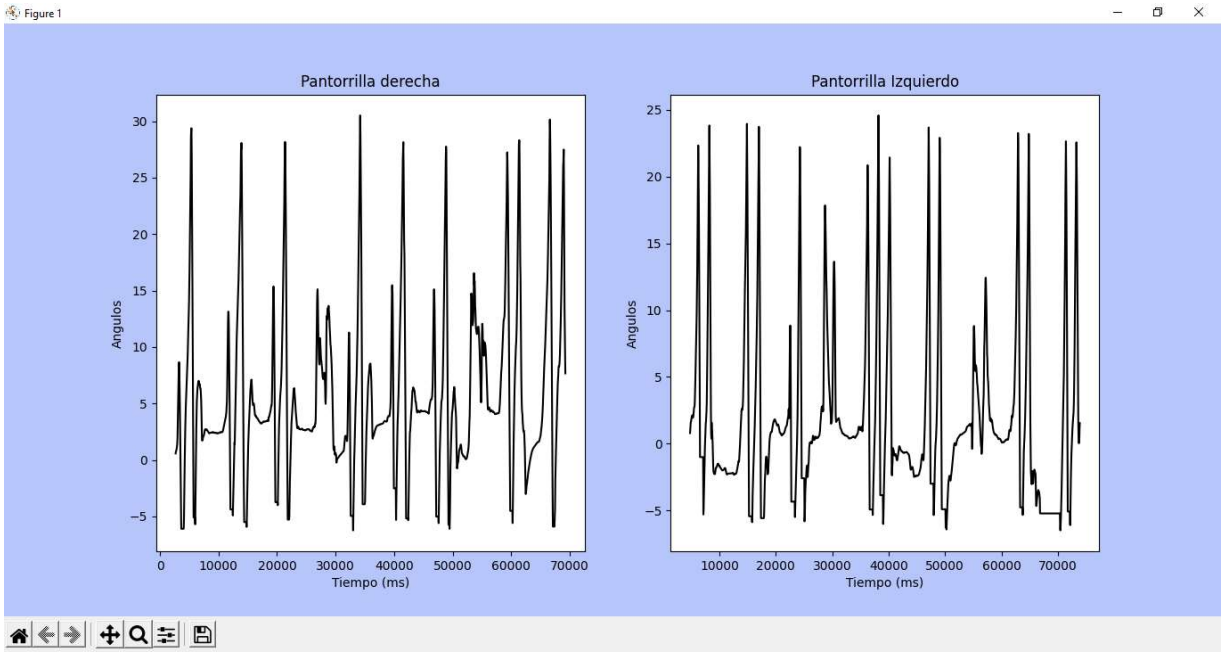


Figura 77 Graficas pantorrilla Izquierda y derecha en caminata. (Fuente propia)

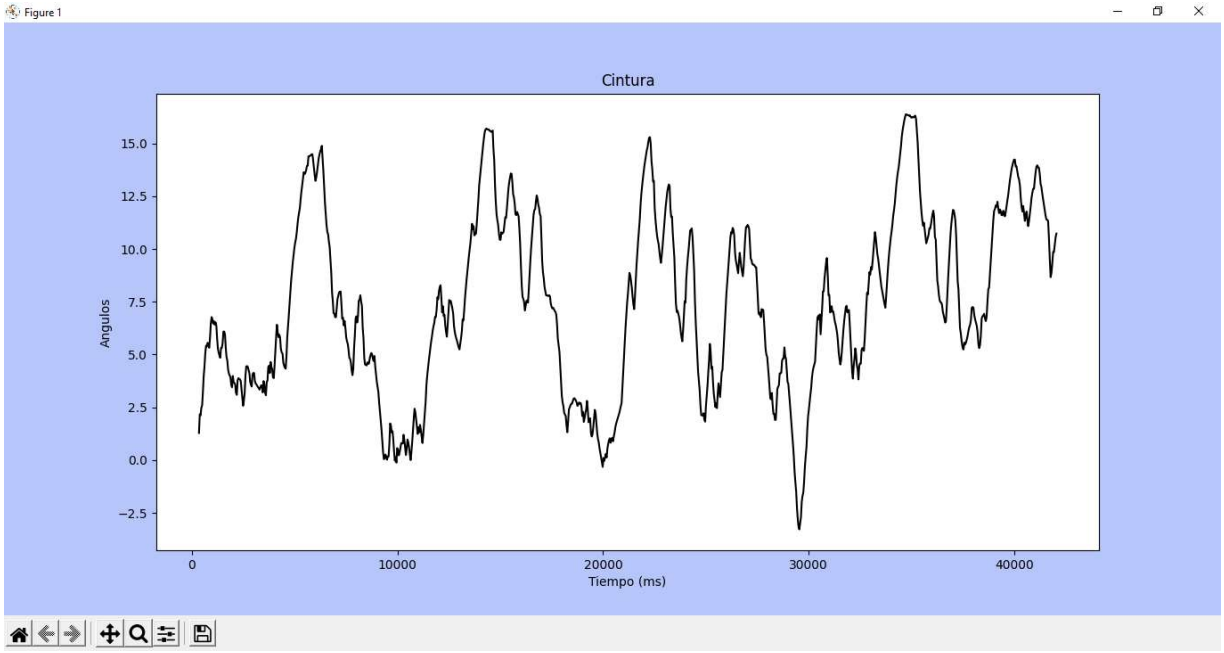


Figura 78 Grafica cintura en caminata. (Fuente propia)

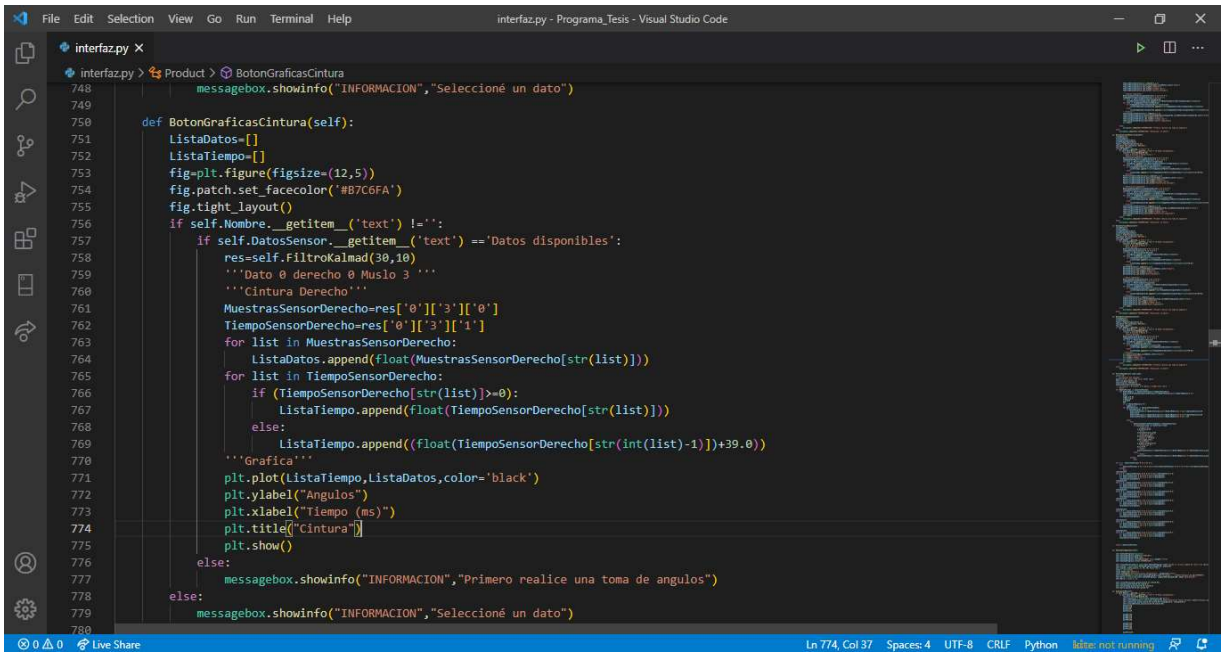


Figura 79 Código grafica de cintura. (Fuente propia)

Finalmente, los botones de simulación muestran una animación en 2D del proceso realizado gracias a los mismos elementos de *Python,* se puede crear un canvas dentro de un escenario controlado en el cual por medio de trigonometría básica se transforma los datos angulares filtrados en posiciones vectoriales, dicho arreglo solo detectará movimientos ascendentes de las piernas dando una visión aproximada de la caminata real del sujeto de prueba, ver Figura 80 y 81.

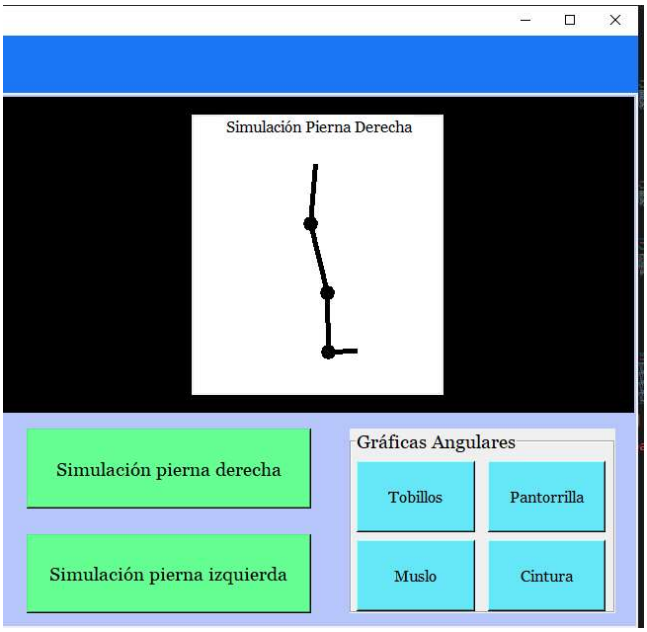


Figura 80 Simulación Pierna Derecha en caminata. (Fuente propia)

En primera instancia el programa trata de igualar de una manera comparativa los tiempos de los sensores dando como resultado un movimiento más armónico y acorde al real, posteriormente se trasforman los grados a puntos vectoriales tomando en cuenta un punto de referencia fijo y una distancia específica, ver Figura 81 y 82.

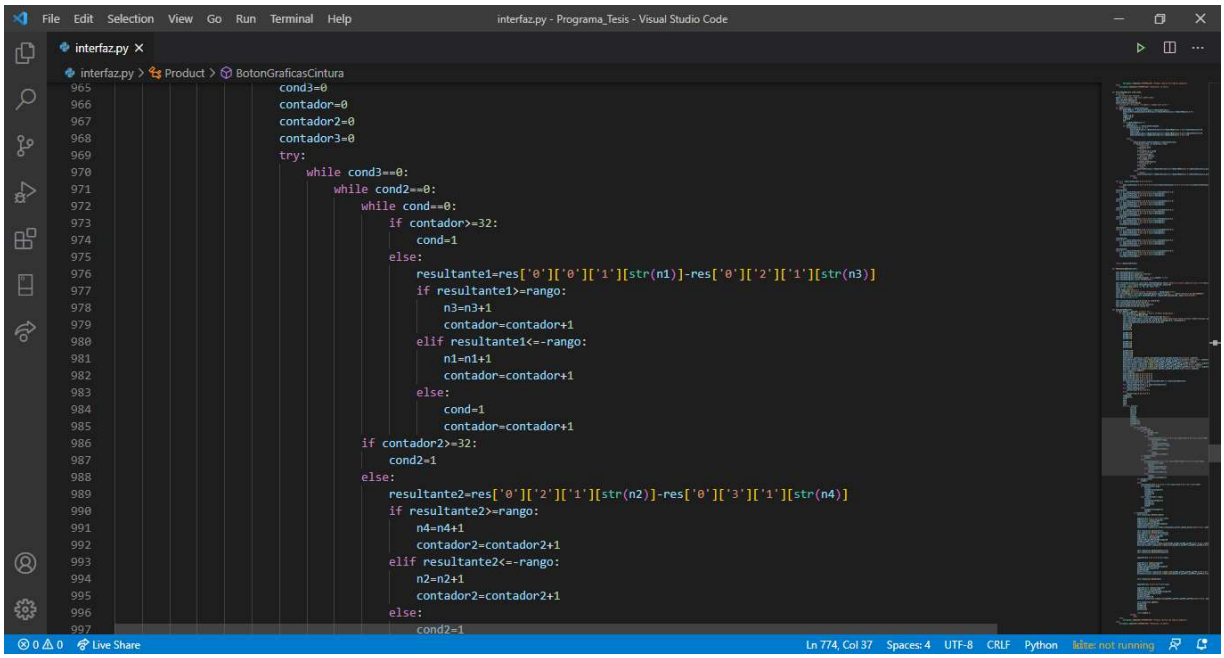


Figura 81 Código Simulación pierna derecha. (Fuente propia)

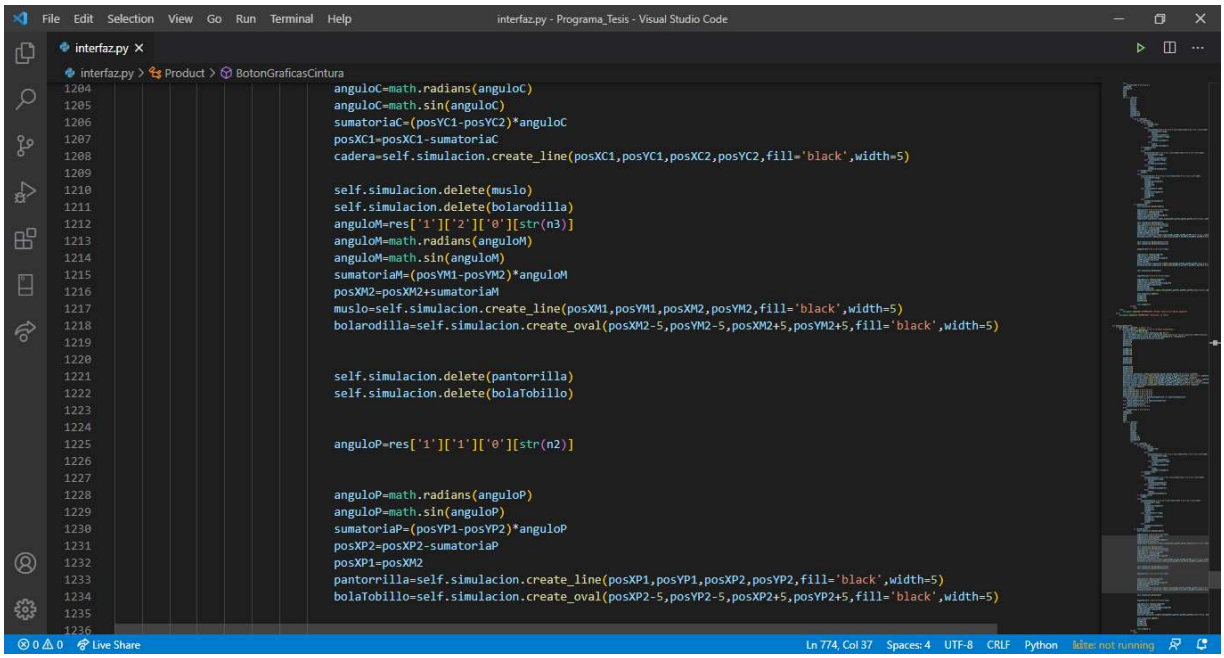


Figura 82 Código Simulación pierna derecha. (Fuente propia)

Capítulo 4

Análisis de los resultados y validación.

4.1 Introducción

En este capítulo se explicará las pruebas realizadas al sistema al igual que sus resultados y los métodos usados para la corrección de ciertas anomalías no contempladas, también se presentan las conclusiones y recomendaciones del proyecto.

4.2 Pruebas del sistema

Una vez diseñado y programado el sistema, se realizó varios tipos de pruebas, las cuales se podrían dividir en:

- Pruebas de funciones relacionadas con la base de datos.
- Pruebas de funciones relacionadas con la toma de muestras y verificación de estas.
- Pruebas de funciones relacionadas con las simulaciones.

4.2.1 Pruebas de funciones relacionadas con la base de datos

Estas pruebas se realizaron con finalidad de comprobar el correcto guardado de los registros en la base de datos, para ello se realizó una ejecución rápida de todos los botones relacionados a este proceso, los cuales constan de: ingreso de registros, modificación de datos, eliminación de registros y guardado de datos angulares todo esto se logró gracias al programa *DBeaver* el cual permite visualizar de manera fácil y didáctica los cambios o sucesos producidos en la base de datos utilizada en este proyecto.

El primer proceso probado y verificado fue el de ingreso de datos, se ejecutó el botón relacionado a este evento, posteriormente se rellenó los datos y se ejecutó la acción de guardando, dando como resultado una correcta inserción de los registros en la base de datos, ver Figura 83 y 84.

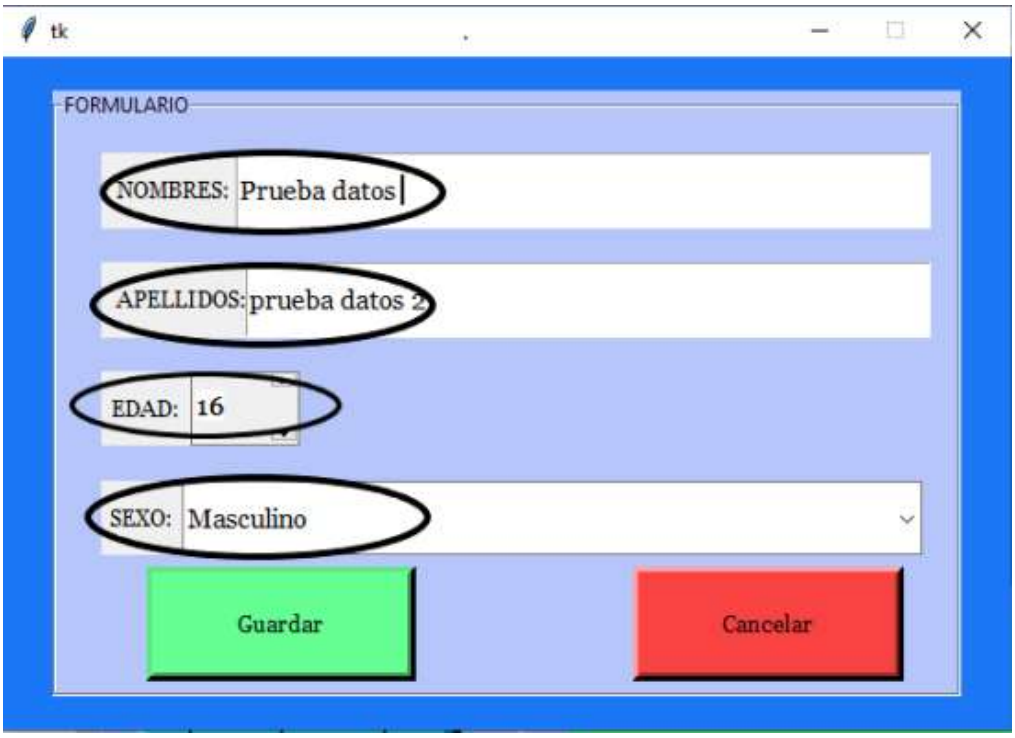


Figura 83 Ingreso de datos de prueba. (Fuente propia)

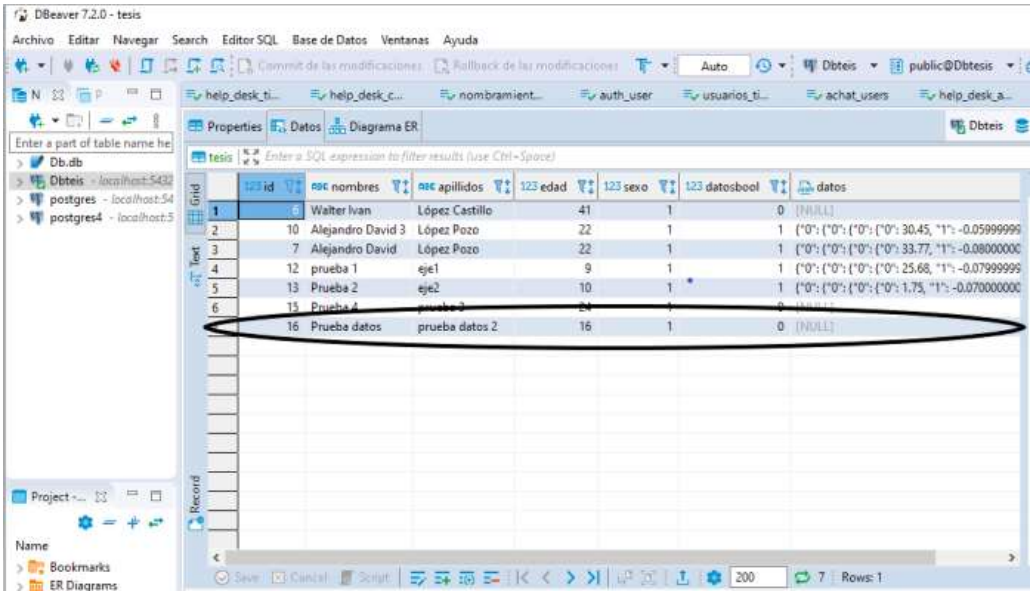


Figura 84 Registro ingresado en la base de datos. (Fuente propia)

Posteriormente se realizó la prueba de modificación de datos y de igual forma se comprobó dichos cambios en la base de datos, dicha modificación se puede comprobar por medio de la comparación de ID entre la prueba de ingreso y la de modificación puesto que dicho ID no debe modificarse, ver Figura 85 y 86.

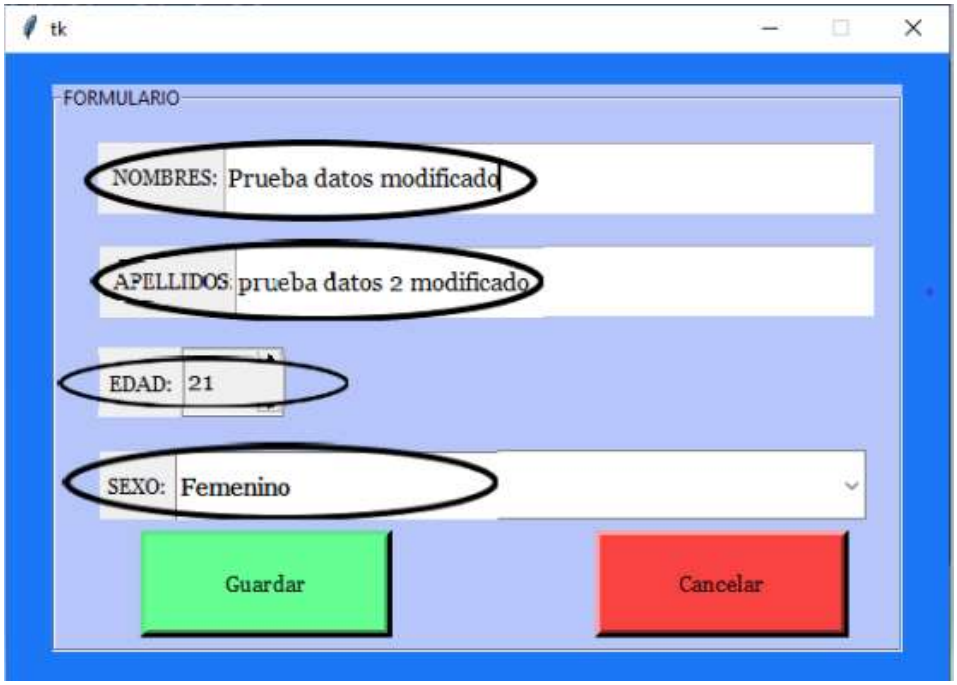


Figura 85 Modificación de registro. (Fuente propia)

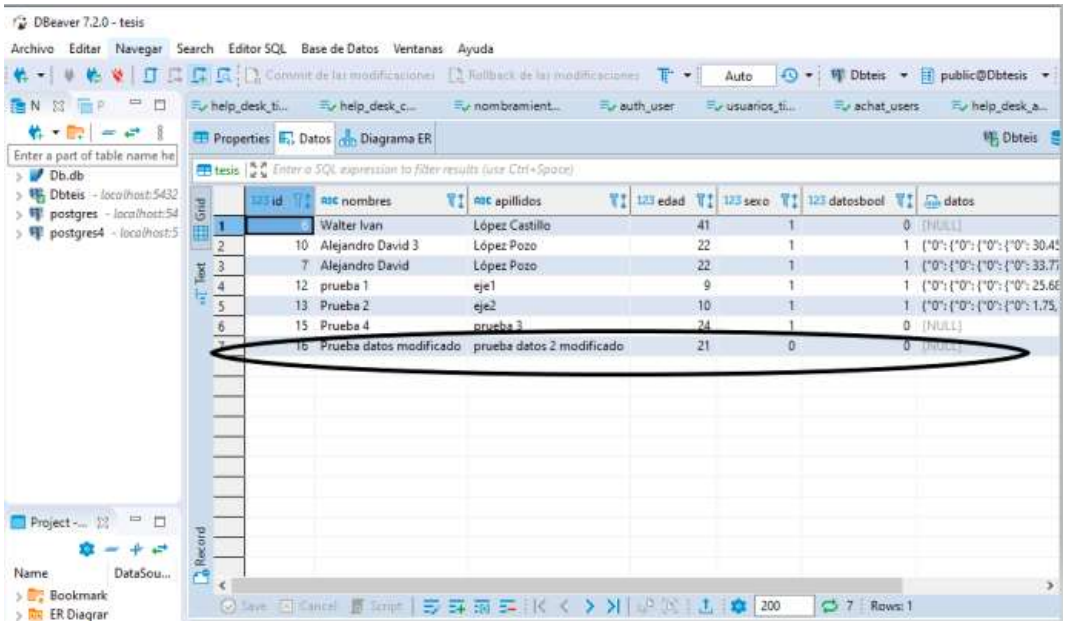


Figura 86 Modificación de registro base de datos. (Fuente propia)

Posteriormente se realizó la prueba de la toma de ángulos y guardado de los mismos dando como resultado una correcta inserción de datos, ver Figura 87.

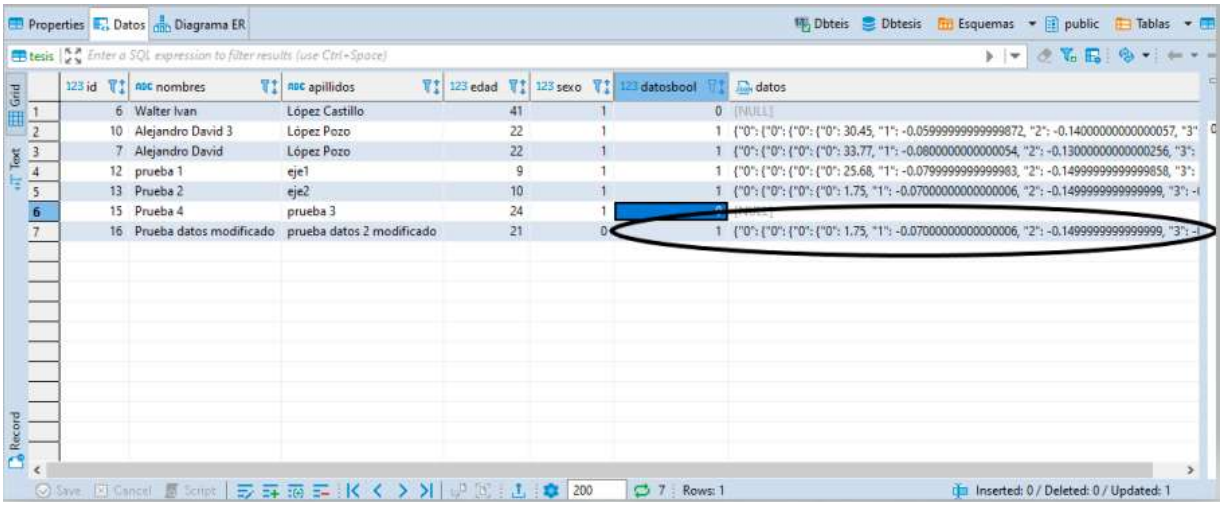


Figura 87 Guardado de muestras de los módulos inerciales en la base de datos. (Fuente propia)

En el proceso de recolección de datos, ver Figura 66, se debe tomar en cuenta ciertas consideraciones, en primera instancia el tiempo que se demora cada módulo inercial en transmitir los datos será de 2 minutos por cada 1500 muestras y a su vez dicho número de muestras será mayor conforme cambie el módulo. Puesto como se explicó en el capítulo anterior existe un aumento de muestras entre módulos, la cual se realiza para evitar la pérdida de datos.

A su vez a nivel código existe un primer recorte de señal la cual trata de igualar el punto de partida de los datos de cada módulo para evitar desfases, esto no se logra del todo puesto que las aproximaciones y la transformación de tiempos a muertas, ver Ecuación 8, no es del todo precisa, ya que dicha ecuación parte de la premisa de que 1500 muestras equivale a 1 minuto de trabajo.

El tiempo de desface se calculó por medio de un cronometro y repitiendo el proceso varias veces, pero no es del todo exacta a nivel código por lo tanto los datos adquiridos por cada módulo llevan un desfase de entre 2 a 3 segundos, de manera acumulativa respecto el primero del último, esto se debe a que los módulos se basan en una conexión 1 a 1 por las características del módulo *HC-05*, explicadas en el capítulo 2.

Por último, se comprobó el funcionamiento del evento eliminar el cual funciona de manera efectiva, borrando el registro seleccionado con todos los datos que este pueda poseer, ver Figura 88.

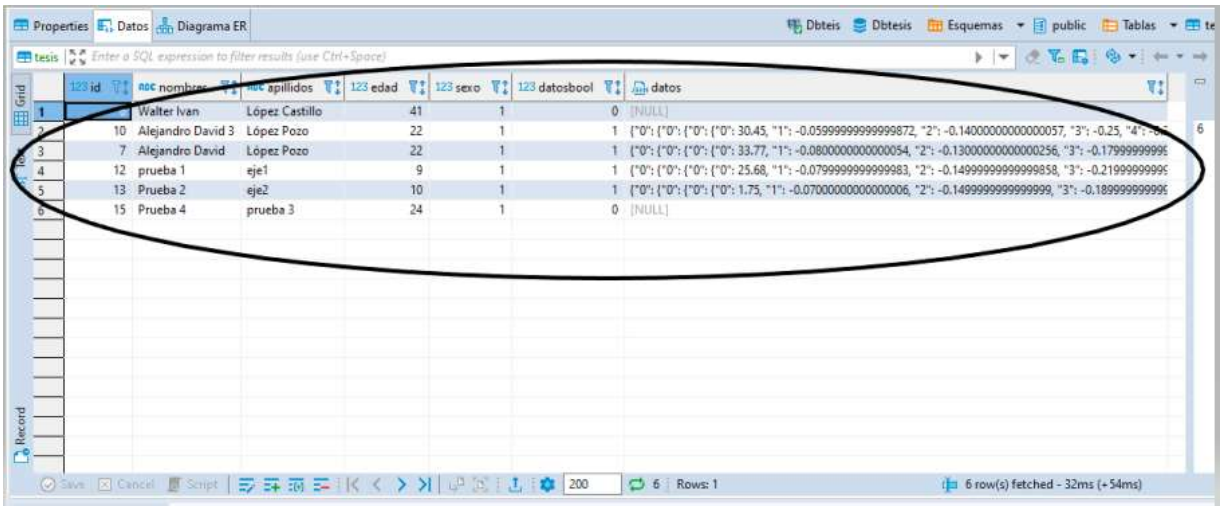


Figura 88 Eliminación de registro base de datos. (Fuente propia)

4.2.2 Pruebas de funciones relacionadas con la toma de muestras y verificación de datos

Estas pruebas se realizaron con la finalidad de comprobar la fiabilidad de los datos adquiridos tanto a los elementos de los módulos como a nivel de código usado para el control de estos, mientras se realizaban estas pruebas ocurrieron varias correcciones a nivel código, las cuales las se encargarán de minimizar los imprevistos producidos por los elementos y tratamiento de señal de los módulos inerciales, las cuales se explicarán a continuación.

La primera prueba se realizó para comprobar el tiempo de desfase entre módulos, dicha prueba se realizó bajo las siguientes condiciones:

- Se ingresó un registro llamado prueba 1.
- Posteriormente se procedió a ubicar todos los módulos inerciales en una superficie plana.
- Se realizó el procedimiento indicado en el capítulo 3 para la adquisición de datos.
- Una vez el programa indico el inicio de la adquisición de datos se procedió a levantar la superficie de apoyo de los módulos dándoles un cambio de ángulo de 38º, medido por graduador, a todos los módulos.
- Una vez realizado el proceso anterior 2 veces se procedió a comprobar por medio de las gráficas el desfase de las señales y de igual forma la fiabilidad de los datos dando como resultado la Figura 89 y 90.

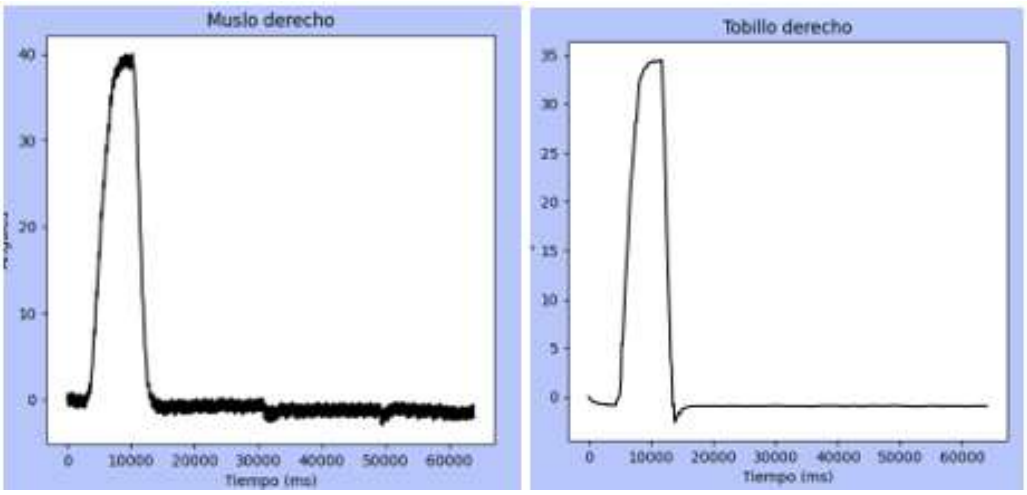


Figura 89 Comparación señales angulares muslo tobillo prueba 1. (Fuente propia)

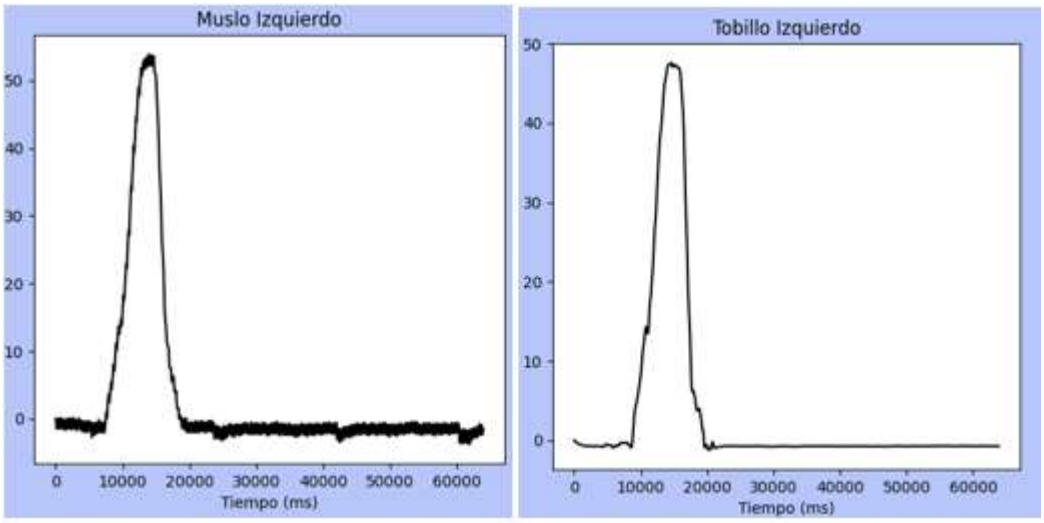


Figura 90 Comparación señales angulares muslo tobillo prueba 2. (Fuente propia)

En las Figuras 89 y 90, se puede apreciar varias consideraciones, en primera instancia el ruido producido en el sensor del muslo es mayor al del tobillo esto se debe a los elementos usados en los módulos inerciales.

Apreciablemente el sensor del muslo posee una mayor inestabilidad que su equivalente de tobillo a pesar de pasar por el mismo tratamiento de señal, el mismo circuito y el uso de los mismos elementos, esto ocurre de una manera concurrente al ir revisando los sensores, algunos son más estables que otros, pero este es el caso más apreciable.

De igual manera se puede apreciar un desfase de señal entre dichos módulos esto se debe a la inexactitud de las aproximaciones en cuanto a muestras por cantidad de tiempo o por muestras dañadas por la transición de datos las cuales son descartadas por el programa ya que dichos elementos aumentarían el ruido y la estabilidad de las señales.

También se aprecia un error de medición del ángulo puesto que el ángulo medido en la primera `prueba era de 38º mientras las señales indican un ángulo máximo de 40º y 35º, al repetir la prueba con un ángulo conocido de 50º los datos variaron de 53º y 47º respectivamente. Gracias a los resultados se pudo establecer un error aproximado de ±3º, esto se produce por el error acumulativo presente en cada módulo, el cual continúa afectando a pesar del uso de los filtros complementario y de Kalmad.

Se intentó reducir el error y el ruido aumentando la cantidad de muestras por minuto, ya que esto produce una reducción significativa de ruido y aumenta la exactitud, pero el resultado fue adverso ya que un aumento de muestras afecta directamente el guardado de los datos en los lectores de memoria, puesto que estos requieren un tiempo mínimo para guardar la información de manera correcta, caso contrario los datos se sobrescriben o son incongruentes provocando que sean completamente inservibles para el proyecto.

La segunda prueba tubo como finalidad comprobar la función del sistema de aducción de datos por medio de un ejercicio más dinámico y aplicable, como es la caminata humana se tomaron en cuenta varias consideraciones:

- Se posicionaron los sensores de una manera específica puesto que la única medición tomada en cuenta para este proyecto se centra en los movimientos del plano sagital, por ello solo se transmiten se procesan y se guardan los datos del eje X de los módulos inerciales, cuyos datos poseen una mayor estabilidad y menor cantidad de ruido que los datos del eje Y por lo tanto el posicionamiento de los módulos debe ser frontal respecto al sector de estudio, ver Figura 91.

- Una vez ejecutado el programa de toma de datos el sujeto de prueba debe mantener una posición recta y estable, mínima de 5 segundos una vez el indicador de movimiento se allá activado puesto que el programa toma el primer

dato como referencia o punto de partida del cual se empezará a medir los cambios angulares producidos en el segmento de estudio.

- Se intentó hacer un movimiento pausado, natural y recto, pero por falta de espacio se realizaron algunos giros los cuales afectaron de manera significativa las señales entregadas por los sensores, ver Figura 92.

Figura 91 Posicionamiento de módulos inerciales. (Fuente propia)

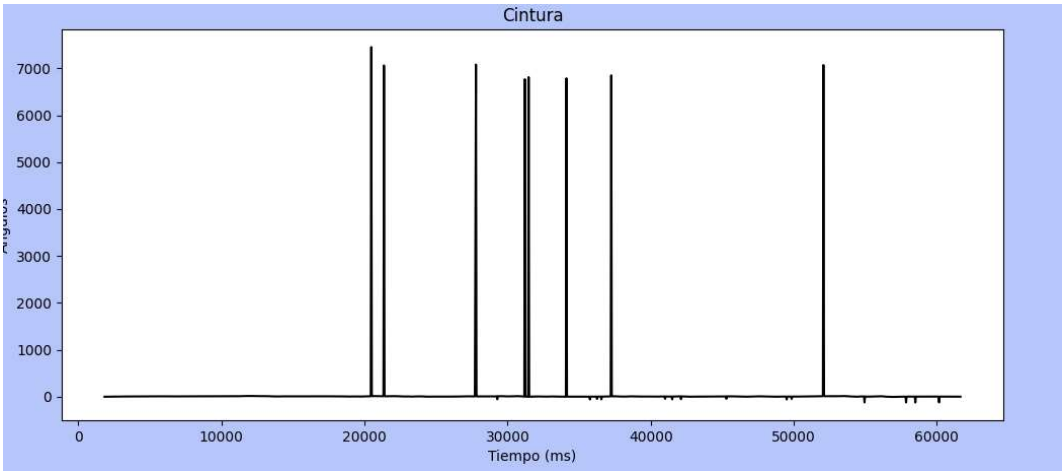


Figura 92 Gráfica de la cintura movimientos circulares y bruscos. (Fuente propia)

Como se puede observar en la Figura 92 existen datos que pueden salirse del rango de medición ya sea por giros o movimientos bruscos, esto se produce puesto que al realizar un giro o algún movimiento brusco existe un cambio de estado de los módulos inerciales lo que produce que algunos datos cambien radicalmente su valor aumentando de manera significativa la cantidad de muestras inestables o fuera de formato.

El formato se encuentra para realizar una correcta identificación de cambio de muestra o separación de ángulo/tiempo de cada cadena, su estructura es: “X2.98T4500”(Fuente propia).

Al pasar por el proceso de guardado se omiten datos relevantes como el punto decimal o los caracteres de separación dando como resultado datos inestables y fuera de contexto, esto se podría evitar alargando el tiempo de transición de datos entre módulo maestro y esclavos, pero esto provocaría que el tiempo de transición aumente significativamente, por ello se implementó un filtro de datos, evitando que dichos datos dañados se tomen en cuenta dando como resultado un cambio significativo en la señal, ver Figura 93.

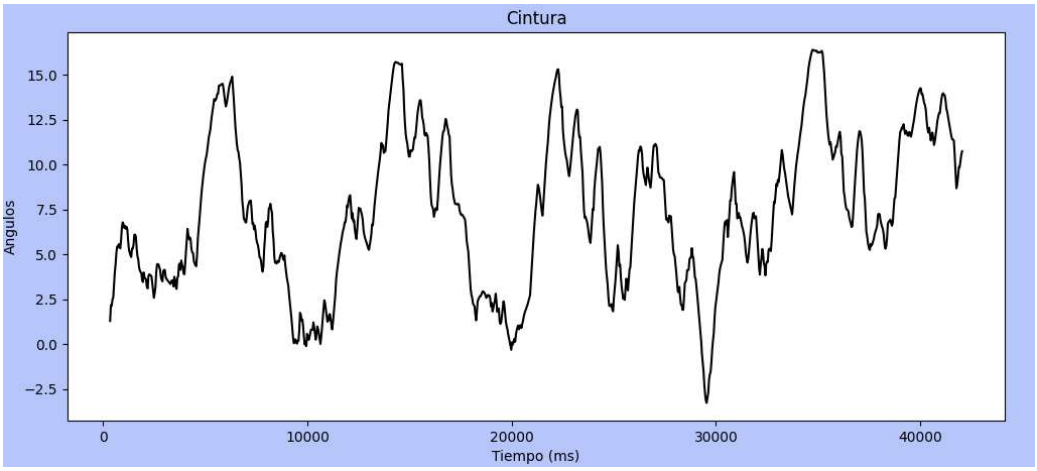


Figura 93 Gráfica de la cintura movimientos circulares y bruscos filtrada. (Fuente propia)

Como se puede observar en la Figura 93, aún existe una cierta cantidad de ruido la cual dada la selección de elementos de los módulos y modelo de procesamiento de datos no puede ser disminuida ni eliminada, ya que para ello se debería realizar un cambio significativo de elementos y métodos provocando una restructuración total del proyecto, lo cual por cuestión de tiempo y presupuesto no es factible.

En la misma figura se puede observar que el recorte de señales o eliminación de datos provoca un recorte significativo en señales, esto se puede evitar con movimientos más fluidos y lineales, pero esto dependerá del tipo de estudio objetivo.

La tercera prueba tubo como finalidad la comprobación de la fiabilidad de los datos tomados en cuanto a caminata se refiere, por lo tanto, se tomaron partes de las diferentes señales las cuales posean una menor cantidad de ruido y se comparó con señales de estudios posteriores, dando como resultado que todos los datos tomados por los sensores se encuentran dentro de los rangos pre-establecidos, ver Figura 94 y 95.

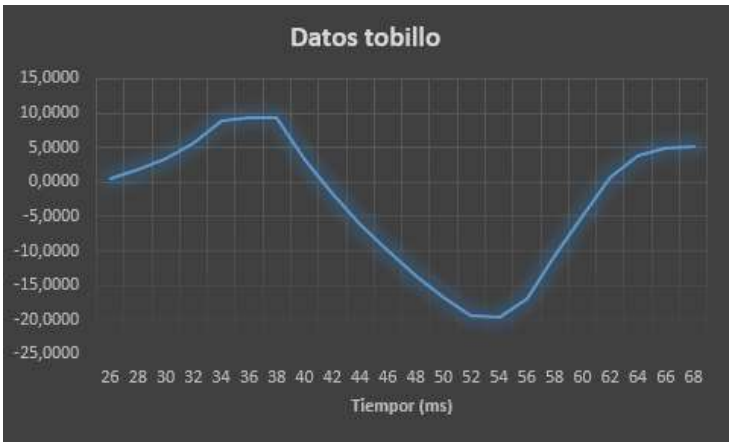


Figura 94 Datos tomados por los módulos tobillo derecho. (Fuente propia)

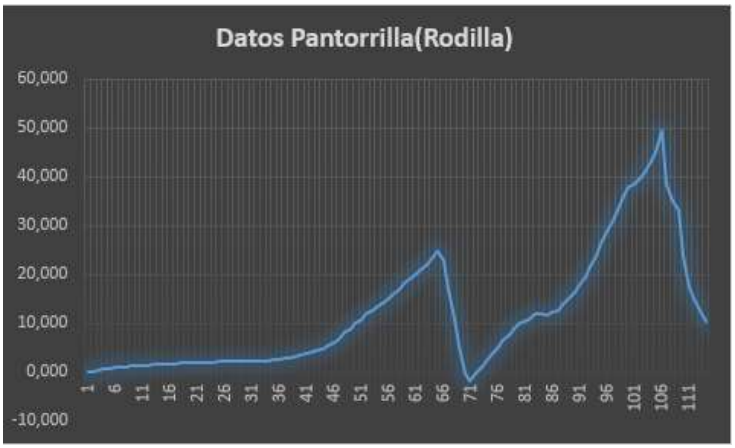


Figura 95 Datos tomados por los módulos pantorrilla(rodilla) derecha. (Fuente propia)

Las Figuras 94 y 95, fueron comparadas con la Figura 20, tomada como objetivo y planteada en el capítulo 1 dichas figuras son similares y se manejan rangos similares, de igual forma el resto de los módulos fueron comparadas con los datos planteados en dicho capitulo dando como resultado que los movimientos medidos en la prueba están dentro del rango normalmente adquirido en estudios posteriores.

La cuarta y la última prueba de referente a este tema fue la comprobación de los tiempos de función en la cual se ejecutó 3 veces la toma de datos con los respectivos tiempos de función de 1,3 y 5 minutos dando como resultado un aumento de 10 segundos por tiempo esto se produce nuevamente por la equivalencia entre muestras y tiempo, puesto que el eje fundamental de la toma de datos es la cantidad de muestras.

4.2.3 Pruebas de funciones relacionadas con las simulaciones

Estas pruebas se realizaron para comprobar la correcta simulación de la marcha humana, en las pruebas se usaron los mismos datos angulares usados anteriormente dando como resultado una simulación con ciertos desfases entre segmentos, ver Figura 96.

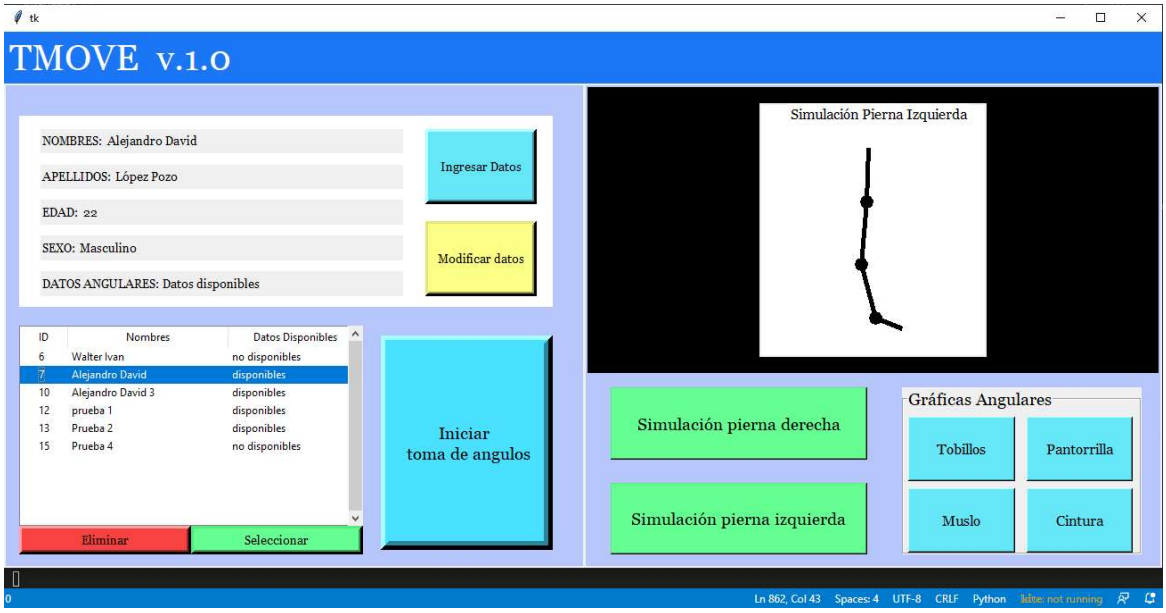


Figura 96 Simulación de caminata datos desfasados. (Fuente propia)

A pesar de los 2 arreglos anteriormente planteados para evitar el desfase de las señales estas continúan con un error de coordinación por lo cual la simulación no es armónica ni presenta datos útiles, por ello se implementó un arreglo en el programa el cual se encarga de una comparación de tiempos, dando un rango mínimo de 2 veces el tiempo por arriba y debajo de una muestra específica, esto se realiza puesto a que cada muestra posee un tiempo específico, y que a su vez es medido en el mismo tiempo que es tomada una muestra, ver Figura 97.

Figura 97 Simulación de caminata datos corregidos. (Fuente propia)

El resultado de la corrección es una simulación más armónica y que permite visualizar pasos más detalladamente, pero aun con un desfase mínimo entre segmentos el cual no se puede corregir puesto es una afectación directa de la eliminación de datos dañados producida por la transición.

Conclusiones

- El método y elementos usados para el desarrollo del sistema no es el óptimo, aunque los resultados obtenidos se encuentran dentro de los rangos propuestos por proyectos similares, lo cual muestra la posibilidad que con un ajuste posterior al sistema se podría optimizar de tal manera que sea más preciso y aplicable en el área médica.
- Una adquisición optima dentro de los parámetros permisibles de los módulos usados, evito datos erroneos, redujo el desfase entre señales y proporciono datos útiles dentro de rangos admisibles.
- El desarrolló de una interfaz gráfica amigable con el usuario, la cual brinda una rápida comprensión en cuanto a flujo y acciones del programa. Además, tomar en cuenta varios factores de diseño expuestos en el capitulo 2 con lo cual se evita que el usuario canse su vista o vea datos innecesarios.
- La validación de todas las acciones permisibles en la interfaz tanto en su parte gráfica como a nivel de código permitió conocer los resultados ya expuestos en la sección de pruebas del sistema.

Recomendaciones

- Para trabajos futuros se recomienda el cambio de módulo Bluetooth puesto que el usado en el presente proyecto tarda en la trasmisión de datos y provoca desfases de señales entre módulos.
- Se recomienda usar otro método de proceso de las señales entregadas por el sensor puesto que la usada sería mucho más efectiva en una toma de datos en tiempo real o con una conexión directa al ordenador ya que esto aumentaría su velocidad de muestreo disminuyendo significativamente su error acumulado.
- Se recomienda mantener el formato de interfaz actual puesto es agradable a la vista y no la cansa y de igual manera se entiende el proceso con facilidad.

Bibliografía

[1] C. H, Goniometría, Buenos Aires: ASOCIART SA ART. ©, 2007.

[2] K. Quijano, *Estimación de la cinemática de las articulaciones de mienbro a partir de un arreglo reducido de sensores,* Popayán: Universidad del Cauca , 2018.

[3] Á. Z. López Quiles, *DISEÑO Y DESARROLLO DE UN SISTEMA PARA LA VALORACIÓN DE LA MOVILIDAD ARTICULAR DE LA MUÑECA MEDIANTE TECNOLOGÍA INERCIAL,* Valéncia: Universidad de Valéncia, 2017.

[4] K. J. R. Benavides, "SISTEMA DE ADQUISICIÓN DE DATOS DE LAS CARACTERÍSTICAS CINEMÁTICAS DE LA MARCHA HUMANA NORMAL NIVEL 1 SIN PENDIENTE," 02 Febrero 2020. [Online]. Available: http://repositorio.utn.edu.ec/bitstream/123456789/10166/2/04%20MEC%20296%20TRABAJO%20GRADO.pdf.

[5] D. F. Terán, "Diseño e implementación de un sistema para visualizar la marcha humana biomecánica en la afectación de rodilla ante una gonartosis," 20 abril 2017. [Online]. Available: https://bibdigital.epn.edu.ec/bitstream/15000/17259/1/CD-7763.pdf.

[6] B. Miranda, "CULCYT," UACJ Revistas electrónicas, 2007. [Online]. Available: https://erevistas.uacj.mx/ojs/index.php/culcyt/article/view/3126. [Accessed 18 07 2020].

[7] Organización Mundial de la Salud, 11 2017. [Online]. Available: https://www.who.int/features/factfiles/disability/es/. [Accessed 18 07 2020].

[8] W. S. S. Leite, "Dialnet," Universidad de la Rioja, 07 2012. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=4741932. [Accessed 18 07 2020].

[9] C. M. F. A. C. Vincent Bonnet, "Un algoritmo de identificación de mínimos cuadrados para estimar la mecánica del ejercicio de sentadillas usando una sola unidad de medida inercial," 11 mayo 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0021929012001145?via%3Dihub.

[10] G. V. Vincent Bonnet, "Determinación rápida de los parámetros inerciales del segmento corporal plano mediante sensores asequibles," 4 Julio 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7052399.

[11] C. Y. E. l.-D. Johanes W. Rohen, Atlas de anatomía humana, Santiado de Chile: MEDITERRANEO, 2002.

[12] F. Stengele, *Diseño y construcción de prototipo neumatico de prótesis de pierna humana,* Puebla: México, 2008.

[13] S. Sosa, "LIFEDER.com," LIFEDER.com, 20 Junio 2019. [Online]. Available: https://www.lifeder.com/planimetria-anatomica/. [Accessed 16 Septiembre 16].

[14] B. N. T. CABRERA, *DISEÑO Y CONSTRUCCIÓN DE UNA PIERNA EXOESQUELÉTICA PARA LA ASISTENCIA DE LA MARCHA,* Santiago de Chile: UNIVERSIDAD DE CHILE, 2017.

[15] M. H. V. M. José Henry Osorio PhD, *Bases para el entendimiento del proceso de la marcha humana.,* Caldas: Universidad de Manizales, 2013.

[16] S. C. Vázquez, *ANÁLISIS DE LA MARCHA HUMANA CON PLATAFORMAS DINAMOMÉTRICAS.,* MADRID : UNIVERSIDAD COMPLUTENSE DE MADRID , 2002.

[17] D. F. Pozo, *Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio.,* Quito : Universidad Politecnica Nacional, 2010.

[18] G. F. Mínguez, *Integración Kalman de sensores inerciales INS con GPS en un UAV,* 2009.
[19] H.-T. S. HERNANDEZ-SANTOS, *Diseño de un sistema de adquisición de variables articulares mediante sensores IMU,* Nuevo León: Instituto Tecnológico de Nuevo León, 2017.
[20] V. F. Byron Contreras, *Diseño construcción e implementación de un sistema enbebido de adquisión de parámetros cinemáticos de la marchahumana en tobillo, rodilla y cadera,* Cuenca: Univesidad politécnic salesiana, 2015.
[21] P. P. López, *Un método de calibración de sensores inerciales,* 2018.
[22] B. M. L. Kamlofsky Jorge A., *LOS CUATERNIONES EN VISIÓN ROBOTICA,* Buenos Aires: Universidad Abierta Interamericana, 2018.
[23] A. A. E. R. F. G. Juan Luna, *Silvereye, sistema electrónico para el análisis de la marcha humana en el plano sagital a partir de sensores inerciales,* Cali: Universidad del Valle, 2018.
[24] H. Maria, *Diseño de un Sistema de Adquisición de Variables Articulares Mediante Sensores IMU,* México D.F: CONACYT, 2017.
[25] Arduino, *Arduino Nano,* 2020.
[26] Greek Factory, *HC-05 Módulo Bluetooth maestro esclavo,* Greek Factory, 2020.
[27] Greek Factory, *Lector de adaptador de tarjeta Micro SD Micro SDHC Mini TF,* 2016.
[28] G. Tolosa, *Protocolos y Modelo OSI,* Univesidad Nacional de Luan , 2015.
[29] E. Carletti, *Comunicación - Bus I2C,* 2015.
[30] S. Gonzales, *Tegnoligía Bluetooth,* México D.F: Instituto Politécnico Nacional , 2008.
[31] S. Yáñez, *Estudio comparativo de sistema de análisis de marcha basados en sensores inerciales y cámaras infrarrojas,* Concepción : Universidad de concepción, 2018.
[32] J. Díaz, *Guía de recomendaciones para diseño de software centrado en el usuario,* Buenos Aires: Universidad Nacional de la Plata , 2013.

Anexos

Anexo I :Código de Arduino módulo maestro

```
#include <SoftwareSerial.h >
int contador=1;//Realiza la conexion por módulo
char confirm='y';// Confirmacion de conexion por módulo
char auxconfirm='2';// Auxiliar de confirm
int modo=0;//modo del sistema // conexion inicial:0 // recoleccion de datos:1
char numero='1';// Envia tiempos y confirma conexiones
char inicio='0';//Habilita inicio de programa
int on=1;
int espera=700;
float tiempo[4];
int activador=0;
String tx1="Realizando conexion S1.....";
String tx2="Realizando conexion S2.....";
String tx3="Realizando conexion S3.....";
int paus= 30;
SoftwareSerial BT1(11, 10); // RX | TX
void setup() {
    pinMode(8, OUTPUT);       // Al poner en HIGH forzaremos el modo AT
    pinMode(4, OUTPUT);       // cuando se alimente de aqui
    digitalWrite(4,  LOW);
    delay (500);              // Espera antes de encender el módulo
    Serial.begin(38400);
    digitalWrite (8, LOW);    //Enciende el módulo
    BT1.begin(38400);
}
void loop() {
 if(inicio=='0'){
   if (Serial.available()){
   inicio=Serial.read();
   numero=inicio;
  }}
  if(inicio!='0'){
   //Confirmar conexión
   if(activador==0){
    BT1.print(numero);
     }
    delay(paus);
    if (BT1.available()>0){
      confirm=BT1.read();
    }
    if(modo==1 and activador==1 and confirm!=auxconfirm){
      delay(4);
      Serial.print(confirm);
      }
    else if (modo==1 and confirm=='X'){
      paus=3;
```

```
  activador=1;
  }
  //Conexión Inicial
  //Sensor 1 Cintura
 if(contador==1 and on==1) {
  Serial.println(tx1);
  digitalWrite (8, LOW);
  delay(espera);
  digitalWrite(4, HIGH);
  delay(espera);
  digitalWrite (8,HIGH);
  delay(espera);
  BT1.write("AT+BIND=98d3,c1,fd9d61\r\n");
  delay(espera-300);
  BT1.write("AT+INIT\r\n");
  delay(espera-300);
  BT1.write("AT+BIND=98d3,c1,fd9d61\r\n");
  delay(espera);
  on=0;
  digitalWrite(4, LOW);
 }
 //Sensor 2 Muslo
 if(contador==2 and on==1) {
  Serial.println(tx2);
  digitalWrite (8, LOW);
  delay(espera);
  digitalWrite(4, HIGH);
  delay(espera);
  digitalWrite (8,HIGH);
  delay(espera);
  BT1.write("AT+BIND=98d3,b1,fd5145\r\n");
  delay(espera-300);
  BT1.write("AT+INIT\r\n");
  delay(espera-300);
  BT1.write("AT+BIND=98d3,b1,fd5145\r\n");
  delay(espera);
  on=0;
  digitalWrite(4, LOW);
 }
//Sensor 3
if(contador==3 and on==1) {
 Serial.println(tx3);
 digitalWrite (8, LOW);
 delay(espera);
 digitalWrite(4, HIGH);
 delay(espera);
 digitalWrite (8,HIGH);
 delay(espera-300);
 BT1.write("AT+BIND=98D3,71,F5BF04\r\n");
 delay(espera-300);
```

```
BT1.write("AT+INIT\r\n");
delay(espera-300);
BT1.write("AT+BIND=98D3,71,F5BF04\r\n");
delay(espera);
on=0;
digitalWrite(4, LOW);
}
//Sensor 4 tobillo
if(contador==4 and on==1) {
  Serial.println("Realizando conexion S4...");
  digitalWrite (8, LOW);
  delay(espera);
  digitalWrite(4, HIGH);
  delay(espera);
  digitalWrite (8,HIGH);
  delay(espera);
  BT1.write("AT+BIND=98D3,41,F5D149\r\n");
  delay(espera-300);
  BT1.write("AT+INIT\r\n");
  delay(espera-300);
  BT1.write("AT+BIND=98D3,41,F5D149\r\n");
  delay(espera);
  on=0;
  digitalWrite(4, LOW);
}
//Cambia de sensor
if(confirm==auxconfirm){
  if(modo==1){
    Serial.println("Cambio");
    delay(10);
    if(contador==5){
      contador=4;
    }
    on=1;
    activador=0;
    paus=30;
    contador=contador-1;
    if(contador==0){
      numero='1';// Envia tiempos y confirma conexiones
      inicio='0';//Habilita inicio de programa
      on=1;
      Serial.println("Listo");
      delay(10);
    }
    else if(contador==3){
      auxconfirm='p';
      tx3="Recolectando datos S3...";
      delay(10);
    }
    else if(contador==2){
```

```
        auxconfirm='m';
        tx2="Recolectando datos S2...";
        delay(10);
       }
       else if(contador==1){
        auxconfirm='c';
        tx1="Recolectando datos S1...";
        delay(10);
       }
       }
     if(modo==0){
       tiempo[contador-1]=millis();
       Serial.println("OK");
       delay(20);
       Serial.println(tiempo[contador-1]);
       on=1;
       contador=contador+1;
       auxconfirm=auxconfirm+1;
       if(auxconfirm=='6'){
        numero='7';
        auxconfirm='t';
        modo=1;
        Serial.println("Modo1");
    delay(20);
       }
    }
    }
}}
```

Anexo II :Código de Arduino módulo inercial

```
//Datos iniciales
#include <SoftwareSerial.h >
#include <SPI.h>
#include <SD.h>
char data='0';
char verificador='0';
int guardado=0;
int contador=0;
int muestras=400;// Dependen del tiempo deseado de función
int n=0;
int confir=0;
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#define SSpin 10
// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
```

```
MPU6050 sensor;
File archivo;
// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;
long tiempo_prev;
float dt;
float ang_x, ang_y;
float ang_x_prev, ang_y_prev;
void setup() {
     //inicio HC 05
    pinMode(8, OUTPUT);        // Al poner en HIGH forzaremos el modo AT
    pinMode(4, OUTPUT);        // cuando se alimente de aqui
    digitalWrite(4, LOW);
    delay (500);              // Espera antes de encender el módulo
    Serial.begin(9600);
    digitalWrite (8, HIGH);    //Enciende el módulo
    //inicio de MPU6050
    Wire.begin();           //Iniciando I2C
    sensor.initialize();    //Iniciando el sensor
    if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
    else Serial.println("Error al iniciar el sensor");
     if (!SD.begin(SSpin)) {
     Serial.print("Fallo");
     return;
    }
    SD.remove("S3.txt");
}
void loop() {
  if (guardado==0){
    if (Serial.available()>0){
      data=Serial.read();
      delay(500);
      }
     if(data=='1'){
      muestras=2925;// 1 minuto+300muestras por módulo
      confir=1;
     }
     if(data=='2'){
      muestras=6125;// 3 minutos
      confir=1;
     }
     if(data=='3'){
      muestras=9325;// 5 minutos
      confir=1;
     }
     if(confir==1){
      Serial.print('2');//cambiar los datos dependiendo del módulo
      guardado=1;
      delay(50);
```

```
  data='0';
 }
}
if(guardado==1){
  if (contador<=muestras){
    sensor.getAcceleration(&ax, &ay, &az);
    sensor.getRotation(&gx, &gy, &gz)
    dt = (millis()-tiempo_prev)/1000.0;
    tiempo_prev=millis()
    //Calcular los ángulos con acelerometro
    float accel_ang_x=atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
    if(az<0){
      accel_ang_x=180-accel_ang_x;
      }
    float accel_ang_y=atan(-ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
    if(ax<0){
      accel_ang_y=180-accel_ang_y;
      }
    //Calcular angulo de rotación con giroscopio y filtro complemento
    ang_x = 0.94*(ang_x_prev+(gx/131)*dt) + 0.06*accel_ang_x;
    ang_y = 0.94*(ang_y_prev+(gy/131)*dt) + 0.06*accel_ang_y;
    ang_x_prev=ang_x;
    ang_y_prev=ang_y;
    if (contador>700){
      delay(20);
      archivo=SD.open("S3.txt",FILE_WRITE);
      if (archivo){
        archivo.print("X");
        archivo.print(ang_x);
        archivo.print("T");
        archivo.println(millis());
        archivo.close();
      }
    contador =contador+1;
  }
  else{
    guardado=2;
  }
 }
 if(guardado==2){
  if (Serial.available()>0){
  data=Serial.read();
  delay(500);
  }
 if(data=='7'){
  archivo=SD.open("S3.txt");
      if (archivo){
        while (archivo.available()) {
          char guard=archivo.read();
          Serial.print(guard);
```

```
            delay(7);
          }
         }
      guardado=3;
      Serial.print('c');//Confirma fin del envio de datos y comunicacion
      delay(50);
    }
    }
}
```

Anexo III: Código de la Interfaz grafica

```
import Tkinter as tk
from Tkinter.ttk import Progressbar
from Tkinter.constants import LEFT, TOP
from Tkinter import *
from Tkinter import ttk
from Tkinter import messagebox
import time
import serial as sr
import psycopg2
import json
from typing import DefaultDict
import serial.tools.list_ports
import matplotlib.pyplot as plt
import math
class Product:
    db_tesis='tesis.db'
    def __init__(self, window):
        '''ventana 1'''
        self.wind=window
        self.wind.geometry("1300x600")
        '''Franja principal forma'''
        Franja1=tk.Label(self.wind, background="#1C77F6",height=4)
        Textofranja1=tk.Label(Franja1, text="TMOVE  v.1.0",font=("Georgia",30), fg="#FFFFFF",
bg="#1C77F6")
        '''BotonDeAyuda=tk.Button(Franja1,height=1,width=1,text="?",border=5, font=("Georgia"))'''
        '''contenedor Grande '''
        Contenedor1=tk.Label(self.wind)
        '''Botones Principales'''
        CajaBotone=tk.Label(Contenedor1, bg="#B7C6FA")
        ContenedorLista=tk.Label(CajaBotone, bg="#FFFFFF")
EtiquetaNombre=tk.Label(ContenedorLista,text="NOMBRES:",font=("Georgia",10),fg="#000000")
EtiquetaApedillo=tk.Label(ContenedorLista,text="APELLIDOS:",font=("Georgia",10),fg="#000000")
        EtiquetaEdad=tk.Label(ContenedorLista,text="EDAD:",font=("Georgia",10),fg="#000000")
        EtiquetaSexo=tk.Label(ContenedorLista,text="SEXO:",font=("Georgia",10),fg="#000000")
        EtiquetaDatosSensor=tk.Label(ContenedorLista,text="DATOS
ANGULARES:",font=("Georgia",10),fg="#000000")
        self.Nombre=tk.Label(ContenedorLista,text=",font=("Georgia",10),fg="#000000", anchor=W)
        self.Apellido=tk.Label(ContenedorLista,text=",font=("Georgia",10),fg="#000000", anchor=W)
```

```
        self.Edad=tk.Label(ContenedorLista,text='',font=("Georgia",10),fg="#000000", anchor=W)
        self.Sexo=tk.Label(ContenedorLista,text='',font=("Georgia",10),fg="#000000", anchor=W)
        self.DatosSensor=tk.Label(ContenedorLista,text='',font=("Georgia",10),fg="#000000",
anchor=W)
        ContenedorBotones=tk.Label(CajaBotone, bg="#FFFFFF")
        BotonIngresoDeDatos=tk.Button(ContenedorBotones, text="Ingresar Datos" , height=4,width=10,
command=self.IngresoDatos,bg="#66E7F7",font=("Georgia",10),border=5)
        BotonModificacionDeDatos=tk.Button(ContenedorBotones, text="Modificar datos" ,
height=4,width=10,command=self.EditarDato,bg="#FCFF85",font=("Georgia",10),border=5)
        self.tabla=ttk.Treeview(CajaBotone,columns=["uno","dos"])
        self.tabla.column("#0",width=-70)
        self.tabla.column("uno", width=55)
        self.tabla.column("dos", width=25)
        self.tabla.heading("#0",text="ID",anchor=CENTER)
        self.tabla.heading("uno",text="Nombres",anchor=CENTER)
        self.tabla.heading("dos",text="Datos Disponibles",anchor=CENTER)
        vsb = ttk.Scrollbar(self.tabla, orient="vertical", command=self.tabla.yview)
        vsb.pack(side='right', fill='y')
BotonEliminar=tk.Button(CajaBotone,text="Eliminar",command=self.Eliminar,bg="#F94342",font=("
Georgia",10),border=5)
BotonSeleccionar=tk.Button(CajaBotone,text="Seleccionar",command=self.Seleccionar,bg="#66FF93
",font=("Georgia",10),border=5)
        BotonTomaDeMuestras=tk.Button(CajaBotone, text="Iniciar \n toma de angulos",
height=4,width=10,bg="#4AE0FF",font=("Georgia",14),border=10,command=self.TomaDeDatos)
        '''Simulación'''
        CajaSimulacion=tk.Label(Contenedor1, bg="#B7C6FA")
        self.Simulacion=tk.Label(CajaSimulacion, height=20,width=100, bg="#000000")
        BotonSimulacion=tk.Button(CajaSimulacion, text="Simulación pierna derecha"
,command=self.SimulacionPD,font=("Georgia",14),bg="#66FF93")
        BotonSimulacion2=tk.Button(CajaSimulacion, text="Simulación pierna izquierda"
,command=self.SimulacionPI,font=("Georgia",14),bg="#66FF93")
        '''Graficas'''
        CajaBotonesGraficas=tk.LabelFrame(CajaSimulacion, text="Gráficas
Angulares",font=("Georgia",14))
        BotonGraficarTobillos=tk.Button(CajaBotonesGraficas, text="Tobillos", height=4,width=50,
command=self.BotonGraficasTobillos,font=("Georgia",12),bg="#66E7F7")
        BotonGraficarPantorrilla=tk.Button(CajaBotonesGraficas, text="Pantorrilla", height=4,width=50,
command=self.BotonGraficasPantorrilla,font=("Georgia",12),bg="#66E7F7")
        BotonGraficarMuslo=tk.Button(CajaBotonesGraficas, text="Muslo", height=4,width=50,
command=self.BotonGraficasMuslo,font=("Georgia",12),bg="#66E7F7")
        BotonGraficarCintura=tk.Button(CajaBotonesGraficas, text="Cintura", height=4,width=50,
command=self.BotonGraficasCintura,font=("Georgia",12),bg="#66E7F7")
        '''Franja principal ubicacion'''
        Franja1.pack(fill= tk.X)
        '''BotonDeAyuda.place(relx=.97,rely=.1)'''
        Textofranja1.grid(row=0,column=0,padx=1,pady=1)
        '''Franja segundaria parte 1 ubicacion'''
        Contenedor1.pack(expand=True, fill="both")
        '''Botones principales ubicacion'''
        CajaBotone.place(x=0,y=0,relheight=1,relwidth=0.5)
```

```
        ContenedorLista.place(relx=0.02,rely=0.06,relheight=0.4,relwidth=0.7)
        EtiquetaNombre.place(relx=0.05,rely=0.06,relheight=0.13,relwidth=0.16)
        EtiquetaApedillo.place(relx=0.05,rely=0.25,relheight=0.13,relwidth=0.18)
        EtiquetaEdad.place(relx=0.05,rely=0.44,relheight=0.13,relwidth=0.1)
        EtiquetaSexo.place(relx=0.05,rely=0.63,relheight=0.13,relwidth=0.09)
        EtiquetaDatosSensor.place(relx=0.05,rely=0.82,relheight=0.13,relwidth=0.3)
        self.Nombre.place(relx=0.21,rely=0.06,relheight=0.13,relwidth=0.75)
        self.Apellido.place(relx=0.23,rely=0.25,relheight=0.13,relwidth=0.73)
        self.Edad.place(relx=0.15,rely=0.44,relheight=0.13,relwidth=0.81)
        self.Sexo.place(relx=0.14,rely=0.63,relheight=0.13,relwidth=0.82)
        self.DatosSensor.place(relx=0.35,rely=0.82,relheight=0.13,relwidth=0.61)
        ContenedorBotones.place(relx=0.7,rely=0.06,relheight=0.4,relwidth=0.25)
        BotonIngresoDeDatos.place(relx=0.1,rely=0.06,relheight=0.4,relwidth=0.8)
        BotonModificacionDeDatos.place(relx=0.1,rely=0.55,relheight=0.4,relwidth=0.8)
        self.tabla.place(relx=0.02,rely=0.5,relheight=0.42,relwidth=0.6)
        BotonTomaDeMuestras.place(relx=0.65,rely=0.52,relheight=0.45,relwidth=0.3)
        CajaSimulacion.place(relx=0.502,rely=0,relheight=1,relwidth=0.498)
        self.Simulacion.place(relx=0,rely=0,relheight=0.6,relwidth=1)
        BotonSimulacion.place(relx=0.04,rely=0.63,relheight=0.15,relwidth=0.45)
        BotonSimulacion2.place(relx=0.04,rely=0.83,relheight=0.15,relwidth=0.45)
        CajaBotonesGraficas.place(relx=0.55,rely=0.63,relheight=0.35,relwidth=0.42)
        BotonGraficarTobillos.place(relx=0.02,rely=0.05,relheight=0.45,relwidth=0.45)
        BotonGraficarPantorrilla.place(relx=0.52,rely=0.05,relheight=0.45,relwidth=0.45)
        BotonGraficarMuslo.place(relx=0.02,rely=0.55,relheight=0.45,relwidth=0.45)
        BotonGraficarCintura.place(relx=0.52,rely=0.55,relheight=0.45,relwidth=0.45)
        BotonEliminar.place(relx=0.02,rely=0.92,relheight=0.06,relwidth=0.3)
        BotonSeleccionar.place(relx=0.32,rely=0.92,relheight=0.06,relwidth=0.3)
        self.tablat()
    def run_query(self,query,parameters=()):
        global cursor
        with psycopg2.connect(host='localhost',database='Dbtesis',user='postgres',password='2323064') as
conn:
            cursor= conn.cursor()
            result=cursor.execute(query,(parameters,))
            conn.commit()
        return result
    def run_query2(self,query,parameters=()):
        global cursor
        with psycopg2.connect(host='localhost',database='Dbtesis',user='postgres',password='2323064') as
conn:
            cursor= conn.cursor()
            result=cursor.execute(query,parameters)
            conn.commit()
        return result
    def tablat(self):
        records= self.tabla.get_children()
        for element in records:
            self.tabla.delete(element)
        query='SELECT * FROM tesis ORDER BY id DESC'
        self.run_query(query)
```

```
        db_row=cursor.fetchall()
        for row in db_row:
            if row[5]==0:
                Datos='no disponibles'
            else:
                Datos='disponibles'
            self.tabla.insert("",0,text=row[0],values=(row[1],Datos,row[2]))
    def IngresoDatos(self):
        self.VentanaIngresoDatos=Toplevel(bg="#1C77F6")
        self.VentanaIngresoDatos.geometry("600x400")
        self.VentanaIngresoDatos.resizable(width= False,height= False)
        self.VentanaIngresoDatos.grab_set()
       self.CajaForm1=tk.LabelFrame(self.VentanaIngresoDatos,text='FORMULARIO',bg="#B7C6FA")
EtiquetaNombrev2=tk.Label(self.CajaForm1,text="NOMBRES:",font=("Georgia",10),fg="#000000")
EtiquetaApedillov2=tk.Label(self.CajaForm1,text="APELLIDOS:",font=("Georgia",10),fg="#000000"
)
        EtiquetaEdadv2=tk.Label(self.CajaForm1,text="EDAD:",font=("Georgia",10),fg="#000000")
        EtiquetaSexov2=tk.Label(self.CajaForm1,text="SEXO:",font=("Georgia",10),fg="#000000")
        self.EntradaNombre=tk.Entry(self.CajaForm1,font=("Georgia",12),fg="#000000")
        self.EntradaApedillo=tk.Entry(self.CajaForm1,font=("Georgia",12),fg="#000000")
        self.EntradaEdad=ttk.Spinbox(self.CajaForm1,
from_=0,to=100,font=("Georgia",12),validate="key",validatecommand=(self.VentanaIngresoDatos.reg
ister(self.validate_entry), "%S"),state="readonly")
self.EntradaSexo=ttk.Combobox(self.CajaForm1,values=("Masculino","Femenino"),font=("Georgia",1
2),state="readonly")
        self.EntradaSexo.current(0)
        self.BotonGuardar=tk.Button(self.CajaForm1, text="Guardar" ,
height=4,width=10,command=self.guardadoDeDatos,bg="#66FF93",font=("Georgia",10),border=5)
        BotonCancelar=tk.Button(self.CajaForm1, text="Cancelar" ,
height=4,width=10,command=self.cerrar,bg="#F94342",font=("Georgia",10),border=5)
        EtiquetaNombrev2.place(relx=0.05,rely=0.06,relheight=0.13,relwidth=0.16)
        EtiquetaApedillov2.place(relx=0.05,rely=0.25,relheight=0.13,relwidth=0.18)
        EtiquetaEdadv2.place(relx=0.05,rely=0.44,relheight=0.13,relwidth=0.1)
        EtiquetaSexov2.place(relx=0.05,rely=0.63,relheight=0.13,relwidth=0.09)
        self.CajaForm1.place(relx=0.05,rely=0.05,relheight=0.9,relwidth=0.9)
        self.EntradaEdad.place(relx=0.15,rely=0.44,relheight=0.13,relwidth=0.12)
        self.EntradaNombre.place(relx=0.2,rely=0.06,relheight=0.13,relwidth=0.77)
        self.EntradaApedillo.place(relx=0.21,rely=0.25,relheight=0.13,relwidth=0.76)
        self.EntradaSexo.place(relx=0.14,rely=0.63,relheight=0.13,relwidth=0.82)
        self.BotonGuardar.place(relx=0.1,rely=0.78,relheight=0.2,relwidth=0.3)
        BotonCancelar.place(relx=0.64,rely=0.78,relheight=0.2,relwidth=0.3)
    def validate_entry(text):
        return text.isdecimal()
    def validacionGuardado(self):
        return len(self.EntradaNombre.get())!=0 and len(self.EntradaApedillo.get())!=0
    def guardadoDeDatos(self):
        if self.validacionGuardado():
            query='INSERT INTO tesis(nombres,apillidos,edad,sexo,datosbool,datos)
VALUES(%s,%s,%s,%s,0,NULL)'
            if self.EntradaSexo.get()=="Masculino":
```

```
            sexo=1
        else:
            sexo=0
        parameter=(self.EntradaNombre.get(),self.EntradaApedillo.get(),self.EntradaEdad.get(),sexo)
        self.mensageGuardadoExito(query,parameter)
    else:
        messagebox.showwarning("ERROR","Datos no validos o faltantes")
    self.tablat()
  def Eliminar(self):
    try:
        self.tabla.item(self.tabla.selection())['values'][0]
    except IndexError as e:
        messagebox.showinfo("INFORMACION","Seleccioné un dato")
        return
    query='DELETE FROM tesis WHERE id=%s'
    parameters=str(self.tabla.item(self.tabla.selection())['text'])
    self.run_query(query,(parameters))
    if self.run_query(query,(parameters)):
        messagebox.showinfo("INFORMACION","Datos eliminados correctamente")
    self.tablat()
  def Seleccionar(self):
    global id
    try:
        self.tabla.item(self.tabla.selection())['values'][0]
    except IndexError as e:
        messagebox.showinfo("INFORMACION","Seleccioné un dato")
        return
    query='SELECT * FROM tesis WHERE id= %s'
    parameters=str(self.tabla.item(self.tabla.selection())['text'])
    self.run_query(query,parameters)
    id=parameters
    result=cursor.fetchall()
    self.Nombre.configure(text=result[0][1])
    self.Apellido.configure(text=result[0][2])
    self.Edad.configure(text=result[0][3])
    if result[0][4]==0:
        sexo='Femenino'
    elif result[0][4]==1:
        sexo='Masculino'
    if result[0][5]==0:
        datos='Datos no disponibles'
    elif result[0][5]==1:
        datos='Datos disponibles'
    self.Sexo.configure(text=sexo)
    self.DatosSensor.configure(text=datos)
  def EditarDato(self):
    if self.Nombre.__getitem__('text') !='':
        self.IngresoDatos()
self.EntradaNombre.configure(textvariable=StringVar(self.CajaForm1,value=self.Nombre.__getitem__
('text')))
```

```
self.EntradaApedillo.configure(textvariable=StringVar(self.CajaForm1,value=self.Apellido.__getitem_
_('text')))
            self.EntradaEdad.set(self.Edad.__getitem__('text'))
            if self.Sexo.__getitem__('text')=='Masculino':
                self.EntradaSexo.current(0)
            elif self.Sexo.__getitem__('text')=='Femenino':
                self.EntradaSexo.current(1)
            self.BotonGuardar.configure(command=self.GardarEditado)
        else:
            messagebox.showinfo("INFORMACION","Seleccioné un dato")
    def GardarEditado(self):
        if self.validacionGuardado():
            global id
            query='UPDATE tesis SET nombres=%s, apillidos=%s, edad=%s, sexo=%s WHERE id=%s'
            if self.EntradaSexo.get()=="Masculino":
                sexo=1
            else:
                sexo=0
        parameter=(self.EntradaNombre.get(),self.EntradaApedillo.get(),self.EntradaEdad.get(),sexo,id)
            self.mensageGuardadoExito(query,parameter)
            result=parameter
            self.Nombre.configure(text=result[0])
            self.Apellido.configure(text=result[1])
            self.Edad.configure(text=result[2])
            if result[3]==0:
                sexo='Femenino'
            elif result[3]==1:
                sexo='Masculino'
            self.Sexo.configure(text=sexo)
        else:
            messagebox.showwarning("ERROR","Datos no validos o faltantes")
        self.tablat()
    def mensageGuardadoExito(self,query,parameter):
        messagebox.showinfo("Guardado","Datos guardados con éxito")
        self.VentanaIngresoDatos.destroy()
        self.run_query2(query,parameter)
    def TomaDeDatos(self):
        if self.DatosSensor.__getitem__('text') !='':
            if self.DatosSensor.__getitem__('text') =='Datos no disponibles':
                auxActivador =1
            elif self.DatosSensor.__getitem__('text') =='Datos disponibles':
                res=messagebox.askyesno('INFORMACION','Datos Existentes\r\nSi continua se eliminarán
los datos actuales \r\n\r\n  Desea continuar??')
                print(res)
                if res== True:
                    auxActivador =1
                else:
                    auxActivador =0
            if auxActivador==1:
                self.VentanaDatos=Toplevel()
```

```
            self.VentanaDatos.geometry('330x200')
            self.VentanaDatos.resizable(width= False,height= False)
            self.VentanaDatos.grab_set()
            self.VentanaDatos.title('INFORMACION')
            self.VentanaDatos.config(bg="#1C77F6")
            self.Caja=tk.Label(self.VentanaDatos,bg="#B7C6FA")
            self.TituloM=tk.Label(self.Caja,text="Seleccioné tiempo de muestreo:",font=("Georgia",12))
            self.EntradaTiempo=ttk.Combobox(self.Caja,values=("1 min","3 min",'5
min'),font=("Arial",10),state="readonly")
            self.EntradaTiempo.current(0)
            self.TituloPiernaInicial=tk.Label(self.Caja,text="Seleccioné pierna:",font=("Georgia",12))
self.EntradaPierna=ttk.Combobox(self.Caja,values=("Derecha","Izquierda"),font=("Arial",10),state="r
eadonly")
            self.EntradaPierna.current(0)
            self.TituloPuerto=tk.Label(self.Caja,text="Seleccioné un puerto:",font=("Georgia",12))
            self.Puertos=ttk.Combobox(self.Caja,font=("Arial",10),state="readonly")
self.BotonAceptar=tk.Button(self.VentanaDatos,text="Aceptar",bg="#6AFF93",font=("Georgia",12),b
order=5,command=self.TomaDeDatosP2)
            self.Caja.place(relx=0,rely=0.1,relheight=0.6,relwidth=1)
            self.TituloPuerto.place(relx=0.05,rely=0.1)
            self.Puertos.place(relx=0.53,rely=0.11,relheight=0.21,relwidth=0.25)
            self.TituloM.place(relx=0.05,rely=0.34)
            self.EntradaTiempo.place(relx=0.75,rely=0.34,relheight=0.21,relwidth=0.2)
            self.TituloPiernaInicial.place(relx=0.05,rely=0.57)
            self.EntradaPierna.place(relx=0.46,rely=0.57,relheight=0.21,relwidth=0.25)
            self.BotonAceptar.place(relx=0.35,rely=0.62,relheight=0.3,relwidth=0.3)
            self.Scan_Com()
        else:
            messagebox.showinfo("INFORMACION","Seleccioné un dato")
    def TomaDeDatosP2(self):
        global subcom
        ContenedorDatosPiernas={ }
        contadorPiernas=0
        if (self.Puertos.get()!=''):
            com=self.Puertos.get()
            in1 = com.index('C') #obtenemos la posición del PUERTO
            in2 = com.index('M') #obtenemos la posición del PUERTO
            subcom=com[in1:in2+2]
            dev = sr.Serial(subcom,38400)
            var=self.EntradaTiempo.get()
            varpierna=self.EntradaPierna.get()
            var=var.replace(' min','')
            if var=='1':
                numeroMuestas=1600
            elif var=='3':
                numeroMuestas=4800
            elif var=='5':
                numeroMuestas=8000
            time.sleep(2)
            i=0
```

```
    while (i<=750):
        dev.write(var.encode('ascii'))
        i=i+1
    self.VentanaDatos.destroy()
    self.VentanaCargaDiseño()
    self.VentanaCarga(dev,numeroMuestas,varpierna,ContenedorDatosPiernas,contadorPiernas)
else:
    messagebox.showinfo("INFORMACION","seleccioné un puerto")
def VentanaCarga(self,dev,numeroMuestas,varpierna,ContenedorDatosPiernas,contadorPiernas):
res=''
var=0
modo=0
luz=1
TimeyMuestr=0
VarS=0
contador=0
comparador=3
TiempoDeConexionSensores={ }
ContenedorVarSensores={ }
ContenedorVarSensores1={ }
ContenedorVarSensores2={ }
ContenedorVarSensores3={ }
angulosdic={ }
tiempodic={ }
angulosdic1={ }
tiempodic1={ }
angulosdic2={ }
tiempodic2={ }
angulosdic3={ }
tiempodic3={ }
DatosPierna={ }
while TRUE:
    if (modo==0):
        if (res!='OK\r\n'):
            self.Barra['value'] += 2
            res=str(dev.readline().decode('ascii'))
            res=res.replace('conexion','conexión')
            self.TituloCarga.configure(text=res)
            self.Barra.update()
            self.TituloCarga.update()
        elif (res!='Modo1\r\n'):
            res=str(dev.readline().decode('ascii'))
            TiempoDeConexionSensores[var]=float(res)
            var=var+1
        if(res=='Modo1\r\n'):
            modo=1
            self.TituloIdicaciones.configure(text='Empiece a caminar',font=("Georgia",14))
            self.TituloIdicaciones.place(relx=0.25,rely=0.05)
            self.alarma.create_oval(5, 5, 70, 70, fill='green')
            self.alarma.update()
```

```
                self.TituloIdicaciones.update()
                for tiempos in TiempoDeConexionSensores
TiempoDeConexionSensores[tiempos]=round(0.02666*(TiempoDeConexionSensores[3]-
TiempoDeConexionSensores[tiempos]))
        elif(modo==1):
                try:
                   res=str(dev.readline().decode('ascii'))
                except:
                   pass
                in1 = res.find('X')
                in2 = res.find('T')
                in3 = res.find('\r')
                if in1!=-1 and in2!=-1 and in3!=-1:
                   if luz==1:
                      luz=0
                      self.TituloIdicaciones.configure(text='Alto!! \r\n transmitiendo datos, no apague los
módulos',font=("Georgia",12))
                      self.TituloIdicaciones.place(relx=0.02,rely=0.05)
                      self.alarma.create_oval(5, 5, 70, 70, fill='red')
                      self.TituloCarga.configure(text='Recolectando datos S4...')
                      self.Barra['value'] += 2
                      self.alarma.update()
                      self.Barra.update()
                      self.TituloIdicaciones.update()
                      self.TituloCarga.update()
                   if contador>=TiempoDeConexionSensores[comparador] and
contador<=numeroMuestas+(TiempoDeConexionSensores[comparador]+150):
                      Angulos=res[in1+1:in2]
                      Tiempo=res[in2+1:in3]
                      if VarS==0:
                         try:
                            if TimeyMuestr==0:
                               angulosdic[TimeyMuestr]=float(Angulos)
                               tiempodic[TimeyMuestr]=float(Tiempo)
                               TimeyMuestr=TimeyMuestr+1
                            else:
                               angulosdic[TimeyMuestr]=float(Angulos)-angulosdic[0]
                               tiempodic[TimeyMuestr]=float(Tiempo)-tiempodic[0]
                               TimeyMuestr=TimeyMuestr+1
                         except ValueError:
                            pass
                      elif VarS==1:
                         try:
                            if TimeyMuestr==0:
                               angulosdic1[TimeyMuestr]=float(Angulos)
                               tiempodic1[TimeyMuestr]=float(Tiempo)
                               TimeyMuestr=TimeyMuestr+1
                            else:
                               angulosdic1[TimeyMuestr]=float(Angulos)-angulosdic1[0]
                               tiempodic1[TimeyMuestr]=float(Tiempo)-tiempodic1[0]
```

```
                TimeyMuestr=TimeyMuestr+1
            except ValueError:
                pass
        elif VarS==2:
            try:
                if TimeyMuestr==0:
                    angulosdic2[TimeyMuestr]=float(Angulos)
                    tiempodic2[TimeyMuestr]=float(Tiempo)
                    TimeyMuestr=TimeyMuestr+1
                else:
                    angulosdic2[TimeyMuestr]=float(Angulos)-angulosdic2[0]
                    tiempodic2[TimeyMuestr]=float(Tiempo)-tiempodic2[0]
                    TimeyMuestr=TimeyMuestr+1
            except ValueError:
                pass
        elif VarS==3:
            try:
                if TimeyMuestr==0:
                    angulosdic3[TimeyMuestr]=float(Angulos)
                    tiempodic3[TimeyMuestr]=float(Tiempo)
                    TimeyMuestr=TimeyMuestr+1
                else:
                    angulosdic3[TimeyMuestr]=float(Angulos)-angulosdic3[0]
                    tiempodic3[TimeyMuestr]=float(Tiempo)-tiempodic3[0]
                    TimeyMuestr=TimeyMuestr+1
            except ValueError:
                pass
    contador=contador+1
if res=='Cambio\r\n':
    TimeyMuestr=0
    contador=0
    comparador=comparador-1
    if VarS==0 :
        ContenedorVarSensores[0]=angulosdic
        ContenedorVarSensores[1]=tiempodic
        print(ContenedorVarSensores[0][0])
        VarS=1
    elif VarS==1:
        ContenedorVarSensores1[0]=angulosdic1
        ContenedorVarSensores1[1]=tiempodic1
        print(ContenedorVarSensores1[0][0])
        VarS=2
    elif VarS==2:
        ContenedorVarSensores2[0]=angulosdic2
        ContenedorVarSensores2[1]=tiempodic2
        print(ContenedorVarSensores2[0][0])
        VarS=3
    elif VarS==3:
        ContenedorVarSensores3[0]=angulosdic3
        ContenedorVarSensores3[1]=tiempodic3
```

```
                    VarS=4
                    DatosPierna[0]=ContenedorVarSensores
                    DatosPierna[1]=ContenedorVarSensores1
                    DatosPierna[2]=ContenedorVarSensores2
                    DatosPierna[3]=ContenedorVarSensores3
                res=str(dev.readline().decode('ascii'))
                self.Barra['value'] += 2
                self.TituloCarga.configure(text=res)
                self.Barra.update()
                self.TituloCarga.update()
            if res=='Listo\r\n':
                dev.close()
                self.alarma.create_oval(5, 5, 70, 70, fill='yellow')
                self.alarma.update()
                if (varpierna=="Derecha"):
                    print("okk paso 1")
                    ContenedorDatosPiernas[0]=DatosPierna
                    contadorPiernas=contadorPiernas+1
                    varpierna="Izquierda"
                    if contadorPiernas<2:
self.VentanaEspera(numeroMuestas,varpierna,ContenedorDatosPiernas,contadorPiernas)
                    else:
                        self.PantallaFin(ContenedorDatosPiernas)
                    break
                if (varpierna=="Izquierda"):
                    print("okk paso 2")
                    ContenedorDatosPiernas[1]=DatosPierna
                    contadorPiernas=contadorPiernas+1
                    varpierna="Derecha"
                    if contadorPiernas<2:
self.VentanaEspera(numeroMuestas,varpierna,ContenedorDatosPiernas,contadorPiernas)
                    else:
                        self.PantallaFin(ContenedorDatosPiernas)
                    break
    def VentanaEspera(self,numeroMuestas,varpierna,ContenedorDatosPiernas,contadorPiernas):
        global numeroMuestas1
        global varpierna1
        global ContenedorDatosPiernas1
        global contadorPiernas1
        numeroMuestas1=numeroMuestas
        varpierna1=varpierna
        ContenedorDatosPiernas1=ContenedorDatosPiernas
        contadorPiernas1=contadorPiernas
        self.TituloIdicaciones.configure(text='Reinicie los módulos y reposicione\r\n Cuando este listo
presione continuar',font=("Georgia",12))
self.BotonContinuar=tk.Button(self.VentanaProgreso,text="Continuar",bg="#6AFF93",font=("Georgia
",12),border=5,command=self.AuxReconect)
        self.BotonContinuar.place(relx=0.65,rely=0.83)
        self.TituloIdicaciones.place(relx=0.12,rely=0.05)
        self.TituloIdicaciones.update()
```

```
        self.VentanaProgreso.update()
    def AuxReconect(self):
        self.BotonContinuar.destroy()
        global numeroMuestas1
        global varpierna1
        global contadorPiernas1
        global subcom
        dev1=sr.Serial(subcom,38400)
        time.sleep(3)
        if numeroMuestas1==1600:
            cont='1'
        elif numeroMuestas1==4800:
            cont='3'
        elif numeroMuestas1==8000:
            cont='5'
        i=0
        while (i<=750):
            dev1.write(cont.encode('ascii'))
            i=i+1
        self.alarma.create_oval(5, 5, 70, 70, fill='red')
        self.TituloIdicaciones.configure(text="Cuando el circulo cambie de color \r\n empiece a caminar")
        self.TituloIdicaciones.update()
        self.alarma.update()
self.VentanaCarga(dev1,numeroMuestas1,varpierna1,ContenedorDatosPiernas1,contadorPiernas1)
    def PantallaFin(self,ContenedorDatosPiernas):
        global id
        datos_json=json.dumps(ContenedorDatosPiernas)
        query='UPDATE tesis SET datosBool=%s,datos=%s WHERE id=%s'
        parameter=('1',datos_json,id)
        messagebox.showinfo("Guardado","Datos guardados con éxito")
        self.DatosSensor.configure(text="Datos disponibles")
        self.run_query2(query,parameter)
        self.DatosSensor.update()
        self.tablat()
        self.VentanaProgreso.destroy()
    def BotonGraficasTobillos(self):
        ListaDatos=[]
        ListaTiempo=[]
        ListaDatosTobilloIzquierdo=[]
        ListaTiempoTobilloIzquierdo=[]
        fig=plt.figure(figsize=(12,30))
        fig.patch.set_facecolor('#B7C6FA')
        fig.tight_layout()
        if self.Nombre.__getitem__('text') !='':
            if self.DatosSensor.__getitem__('text') =='Datos disponibles':
                res=self.FiltroKalmad(15,20)
                '''Dato 0 derecho 0 tobillo 0 '''
                '''Tobillo Derecho'''
                MuestrasSensorTobilloDerecho=res['0']['0']['0']
                TiempoSensorTobilloDerecho=res['0']['0']['1']
```

```
            for list in MuestrasSensorTobilloDerecho:
                ListaDatos.append(float(MuestrasSensorTobilloDerecho[str(list)]))
            for list in TiempoSensorTobilloDerecho:
                if (TiempoSensorTobilloDerecho[str(list)]>=0):
                    ListaTiempo.append(float(TiempoSensorTobilloDerecho[str(list)]))
                else:
                    ListaTiempo.append((float(TiempoSensorTobilloDerecho[str(int(list)-1)])+39.0))
            '''Grafica'''
            TobilloDerechoGrafica=plt.subplot(1,2,1)
            TobilloDerechoGrafica.plot(ListaTiempo,ListaDatos,color='black')
            TobilloDerechoGrafica.set_ylabel("Angulos")
            TobilloDerechoGrafica.set_xlabel("Tiempo (ms)")
            TobilloDerechoGrafica.set_title("Tobillo derecho")
            '''Tobillo Izquierdo'''
            MuestrasSensorTobilloIzquierdo=res['1']['0']['0']
            TiempoSensorTobilloIzquierdo=res['1']['0']['1']
            for list in MuestrasSensorTobilloIzquierdo:
                ListaDatosTobilloIzquierdo.append(float(MuestrasSensorTobilloIzquierdo[str(list)]))
            for list in TiempoSensorTobilloIzquierdo:
                if (TiempoSensorTobilloIzquierdo[str(list)]>=0):
                    ListaTiempoTobilloIzquierdo.append(float(TiempoSensorTobilloIzquierdo[str(list)]))
                else:
                    ListaTiempoTobilloIzquierdo.append((float(TiempoSensorTobilloIzquierdo[str(int(list)-
1)])+39.0))
            '''Grafica'''
            TobilloIzquierdoGrafica=plt.subplot(1,2,2)
TobilloIzquierdoGrafica.plot(ListaTiempoTobilloIzquierdo,ListaDatosTobilloIzquierdo,color='black')
            TobilloIzquierdoGrafica.set_ylabel("Angulos")
            TobilloIzquierdoGrafica.set_xlabel("Tiempo (ms)")
            TobilloIzquierdoGrafica.set_title("Tobillo Izquierdo")
            plt.show()
        else:
            messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
    else:
        messagebox.showinfo("INFORMACION","Seleccioné un dato")
  def BotonGraficasPantorrilla(self):
    ListaDatos=[]
    ListaTiempo=[]
    ListaDatosIzquierdo=[]
    ListaTiempoIzquierdo=[]
    fig=plt.figure(figsize=(12,5))
    fig.patch.set_facecolor('#B7C6FA')
    fig.tight_layout()
    if self.Nombre.__getitem__('text') !='':
        if self.DatosSensor.__getitem__('text') =='Datos disponibles':
            res=self.FiltroKalmad(60,10)
            '''Dato 0 derecho 0 Pantorrilla 1 '''
            '''Pantorrilla Derecho'''
            MuestrasSensorPantorrillaDerecho=res['0']['1']['0']
            TiempoSensorPantorrillaDerecho=res['0']['1']['1']
```

```
            for list in MuestrasSensorPantorrillaDerecho:
                ListaDatos.append(float(MuestrasSensorPantorrillaDerecho[str(list)]))
            for list in TiempoSensorPantorrillaDerecho:
                if (TiempoSensorPantorrillaDerecho[str(list)]>=0):
                    ListaTiempo.append(float(TiempoSensorPantorrillaDerecho[str(list)]))
                else:
                    ListaTiempo.append((float(TiempoSensorPantorrillaDerecho[str(int(list)-1)])+39.0))
            '''Grafica'''
            PantorrillaDerechoGrafica=plt.subplot(1,2,1)
            PantorrillaDerechoGrafica.plot(ListaTiempo,ListaDatos,color='black')
            PantorrillaDerechoGrafica.set_ylabel("Angulos")
            PantorrillaDerechoGrafica.set_xlabel("Tiempo (ms)")
            PantorrillaDerechoGrafica.set_title("Pantorrilla derecha")
            '''Pantorrilla Izquierdo'''
            MuestrasSensorPantorrillaIzquierdo=res['1']['1']['0']
            TiempoSensorPantorrillaIzquierdo=res['1']['1']['1']
            for list in MuestrasSensorPantorrillaIzquierdo:
                ListaDatosIzquierdo.append(float(MuestrasSensorPantorrillaIzquierdo[str(list)]))
            for list in TiempoSensorPantorrillaIzquierdo:
                if (TiempoSensorPantorrillaIzquierdo[str(list)]>=0):
                    ListaTiempoIzquierdo.append(float(TiempoSensorPantorrillaIzquierdo[str(list)]))
                else:
                    ListaTiempoIzquierdo.append((float(TiempoSensorPantorrillaIzquierdo[str(int(list)-
1)])+39.0))
            '''Grafica'''
            TobilloIzquierdoGrafica=plt.subplot(1,2,2)
            TobilloIzquierdoGrafica.plot(ListaTiempoIzquierdo,ListaDatosIzquierdo,color='black')
            TobilloIzquierdoGrafica.set_ylabel("Angulos")
            TobilloIzquierdoGrafica.set_xlabel("Tiempo (ms)")
            TobilloIzquierdoGrafica.set_title("Pantorrilla Izquierdo")
            plt.show()
        else:
            messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
    else:
        messagebox.showinfo("INFORMACION","Seleccioné un dato")
  def BotonGraficasMuslo(self):
    ListaDatos=[]
    ListaTiempo=[]
    ListaDatosIzquierdo=[]
    ListaTiempoIzquierdo=[]
    fig=plt.figure(figsize=(12,5))
    fig.patch.set_facecolor('#B7C6FA')
    fig.tight_layout()
    if self.Nombre.__getitem__('text') !='':
        if self.DatosSensor.__getitem__('text') =='Datos disponibles':
            res=self.FiltroKalmad(40,20)
            '''Dato 0 derecho 0 Muslo 2 '''
            '''Muslo Derecho'''
            MuestrasSensorDerecho=res['0']['2']['0']
            TiempoSensorDerecho=res['0']['2']['1']
```

```
            for list in MuestrasSensorDerecho:
                ListaDatos.append(float(MuestrasSensorDerecho[str(list)]))
            for list in TiempoSensorDerecho:
                if (TiempoSensorDerecho[str(list)]>=0):
                    ListaTiempo.append(float(TiempoSensorDerecho[str(list)]))
                else:
                    ListaTiempo.append((float(TiempoSensorDerecho[str(int(list)-1)])+39.0))
            '''Grafica'''
            DerechoGrafica=plt.subplot(1,2,1)
            DerechoGrafica.plot(ListaTiempo,ListaDatos,color='black')
            DerechoGrafica.set_ylabel("Angulos")
            DerechoGrafica.set_xlabel("Tiempo (ms)")
            DerechoGrafica.set_title("Muslo derecho")
            '''Muslo Izquierdo'''
            MuestrasSensorIzquierdo=res['1']['2']['0']
            TiempoSensorIzquierdo=res['1']['2']['1']
            for list in MuestrasSensorIzquierdo:
                ListaDatosIzquierdo.append(float(MuestrasSensorIzquierdo[str(list)]))
            for list in TiempoSensorIzquierdo:
                if (TiempoSensorIzquierdo[str(list)]>=0):
                    ListaTiempoIzquierdo.append(float(TiempoSensorIzquierdo[str(list)]))
                else:
                    ListaTiempoIzquierdo.append((float(TiempoSensorIzquierdo[str(int(list)-1)])+39.0))
            '''Grafica'''
            IzquierdoGrafica=plt.subplot(1,2,2)
            IzquierdoGrafica.plot(ListaTiempoIzquierdo,ListaDatosIzquierdo,color='black')
            IzquierdoGrafica.set_ylabel("Angulos")
            IzquierdoGrafica.set_xlabel("Tiempo (ms)")
            IzquierdoGrafica.set_title("Muslo Izquierdo")
            plt.show()
        else:
            messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
    else:
        messagebox.showinfo("INFORMACION","Seleccioné un dato")
def BotonGraficasCintura(self):
    ListaDatos=[]
    ListaTiempo=[]
    fig=plt.figure(figsize=(12,5))
    fig.patch.set_facecolor('#B7C6FA')
    fig.tight_layout()
    if self.Nombre.__getitem__('text') !='':
        if self.DatosSensor.__getitem__('text') =='Datos disponibles':
            res=self.FiltroKalmad(30,10)
            '''Dato 0 derecho 0 Muslo 3 '''
            '''Cintura Derecho'''
            MuestrasSensorDerecho=res['0']['3']['0']
            TiempoSensorDerecho=res['0']['3']['1']
            for list in MuestrasSensorDerecho:
                ListaDatos.append(float(MuestrasSensorDerecho[str(list)]))
            for list in TiempoSensorDerecho:
```

```
            if (TiempoSensorDerecho[str(list)]>=0):
                ListaTiempo.append(float(TiempoSensorDerecho[str(list)]))
            else:
                ListaTiempo.append((float(TiempoSensorDerecho[str(int(list)-1)])+39.0))
        '''Grafica'''
        plt.plot(ListaTiempo,ListaDatos,color='black')
        plt.ylabel("Angulos")
        plt.xlabel("Tiempo (ms)")
        plt.title("Cintura")
        plt.show()
      else:
        messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
    else:
      messagebox.showinfo("INFORMACION","Seleccioné un dato")
def FiltroKalmad(self,limS,limI):
    global id
    '''constatante del proceso'''
    query='SELECT datos FROM tesis WHERE id=%s'
    self.run_query(query,id)
    result=cursor.fetchall()
    muestrasPiernas=result[0][0]
    '''['0']pierna ['0']sensor ['0']datos o tiempo esto varia '''
    ''' Datos'''
    for NumeroPiernas in muestrasPiernas:
      for NumeroMódulo in muestrasPiernas[str(NumeroPiernas)]:
        muestrasPiernasdata=muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['0']
        Q=0.5
        sigma_n=0.92
        sigma_u=0.9
        s_pred=0
        M=0
        if NumeroMódulo=='0':
          sigma_n=1.2
        for datosSensores in muestrasPiernasdata:
          if datosSensores == '0':
            muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['1'][str(datosSensores)]=0
            muestrax=0
            muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['0'][str(datosSensores)]=0
            muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['0']['1']=0
              else:
            try:
              muestrax=muestrasPiernasdata[str(datosSensores)]
              if muestrax<=limS and muestrax>=-limI:
                '''Prediccion'''
                s_pred=Q*s_pred
                '''Error'''
                error=muestrax-s_pred
                '''Prediccion MSE'''
                M=(Q**2)*M+sigma_u
                '''ganancia Kalmand'''
```

```
                        K=M/((sigma_n+M))
                        '''Estimandor'''
                        s_pred=s_pred+K*error
                        '''Actualizar M'''
                        M=(1-K)*M
                        '''Señal'''
muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['0'][datosSensores]=s_pred
                    else:
                        '''Señal'''
muestrasPiernas[str(NumeroPiernas)][str(NumeroMódulo)]['0'][datosSensores]=s_pred
                except:
                    pass
    for n in  muestrasPiernas['0']['3']['0']:
        try:
muestrasPiernas['0']['3']['0'][str(n)]=((muestrasPiernas['0']['3']['0'][str(n)]+muestrasPiernas['1']['3']['0'][s
tr(n)])*0.5)
        except:
            pass
    retardante=0
    while 1.8 > muestrasPiernas['0']['0']['0'][str(retardante)]>-1.8:
        del muestrasPiernas['0']['0']['0'][str(retardante)]
        del muestrasPiernas['0']['0']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 1.8 > muestrasPiernas['1']['0']['0'][str(retardante)]>-1.8:
        del muestrasPiernas['1']['0']['0'][str(retardante)]
        del muestrasPiernas['1']['0']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 0.5 > muestrasPiernas['0']['1']['0'][str(retardante)]>-0.5:
        del muestrasPiernas['0']['1']['0'][str(retardante)]
        del muestrasPiernas['0']['1']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 0.5 > muestrasPiernas['1']['1']['0'][str(retardante)]>-0.5:
        del muestrasPiernas['1']['1']['0'][str(retardante)]
        del muestrasPiernas['1']['1']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 0.5 > muestrasPiernas['0']['2']['0'][str(retardante)]>-0.5:
        del muestrasPiernas['0']['2']['0'][str(retardante)]
        del muestrasPiernas['0']['2']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 0.5 > muestrasPiernas['1']['2']['0'][str(retardante)]>-0.5:
        del muestrasPiernas['1']['2']['0'][str(retardante)]
        del muestrasPiernas['1']['2']['1'][str(retardante)]
        retardante=retardante+1
    retardante=0
    while 0.7 > muestrasPiernas['0']['3']['0'][str(retardante)]>-0.7:
```

```
            del muestrasPiernas['0']['3']['0'][str(retardante)]
            del muestrasPiernas['0']['3']['1'][str(retardante)]
            retardante=retardante+1
        return (muestrasPiernas)
    def VentanaCargaDiseño(self)
        self.VentanaProgreso=Toplevel()
        self.VentanaProgreso.geometry('330x250')
        self.VentanaProgreso.grab_set()
        self.VentanaProgreso.resizable(width= False,height= False)
        self.VentanaProgreso.title('INFORMACION')
        self.TituloIdicaciones=tk.Label(self.VentanaProgreso,text="Cuando el circulo cambie de color
\r\n empiece a caminar",font=("Georgia",12),fg="#000000")
        self.alarma = Canvas(self.VentanaProgreso,height=70, width=70)
        self.alarma.create_oval(5, 5, 70, 70, fill='red')
        style = ttk.Style()
        style.theme_use('default')
        style.configure("black.Horizontal.TProgressbar", background='blue')
self.TituloCarga=tk.Label(self.VentanaProgreso,text="Cargando....",font=("Georgia",8),fg="#000000"
)
        self.Barra = Progressbar(self.VentanaProgreso, length=300,maximum=100, mode='determinate')
        self.Barra ['value'] = 0
        self.TituloIdicaciones.place(relx=0.12,rely=0.05)
        self.alarma.place(relx=0.36,rely=0.3)
        self.TituloCarga.place(relx=0.04,rely=0.8)
        self.Barra.place(relx=0.05,rely=0.74)
    def SimulacionPD(self):
        if self.Nombre.__getitem__('text') !='':
            if self.DatosSensor.__getitem__('text') =='Datos disponibles':
                res=self.FiltroKalmad(60,40)
                self.simulacion = Canvas(self.Simulacion,bg='White')
                self.tituloSDerecha=tk.Label(self.Simulacion,text="Simulación Pierna
Derecha",font=("Georgia",12),fg="#000000", bg="#FFFFFF")
                self.simulacion.place(relx=0.30,rely=0.05,relheight=0.9, relwidth=0.4)
                self.tituloSDerecha.place(relx=0.35,rely=0.05)
                posXC1=120
                posXC2=120
                posYC1=50
                posYC2=110
                posXM1=120
                posXM2=120
                posYM1=110
                posYM2=180
                posXP1=120
                posXP2=120
                posYP1=180
                posYP2=240
                posXPi1=120
                posXPi2=150
                posYPi1=240
                posYPi2=240
```

```
            cadera=self.simulacion.create_line(posXC1,posYC1,posXC2,posYC2,fill='black',width=5)
            bolacadera=self.simulacion.create_oval(posXC1-5,posYC2-
5,posXC1+5,posYC2+5,fill='black',width=5)
            muslo=self.simulacion.create_line(posXM1,posYM1,posXM2,posYM2,fill='black',width=5)
            bolarodilla=self.simulacion.create_oval(posXM2-5,posYM2-
5,posXM2+5,posYM2+5,fill='black',width=5)
pantorrilla=self.simulacion.create_line(posXP1,posYP1,posXP2,posYP2,fill='black',width=5)
            bolaTobillo=self.simulacion.create_oval(posXP2-5,posYP2-
5,posXP2+5,posYP2+5,fill='black',width=5)
            pie=self.simulacion.create_line(posXPi1,posYPi1,posXPi2,posYPi2,fill='black',width=5)
            self.Simulacion.update()
            time.sleep(1)
            muestrasC=len(res['0']['3']['0'])
            muestrasM=len(res['0']['2']['0'])
            muestrasP=len(res['0']['1']['0'])
            muestrasT=len(res['0']['0']['0'])
            if muestrasC<muestrasM and muestrasC<muestrasP and muestrasC<muestrasT:
               selctor=res['0']['3']['0']
            elif muestrasM<muestrasP and muestrasC<muestrasT:
               selctor=res['0']['2']['0']
            elif muestrasP<muestrasT:
               selctor=res['0']['1']['0']
            else:
               selctor=res['0']['0']['0']
            rango=300
            contador=0
            n1=0
            n2=0
            n3=0
            n4=0
            for n in selctor:
                  n1=1+n1
                  n2=1+n2
                  n3=1+n3
                  n4=1+n4
                  cond=0
                  cond2=0
                  cond3=0
                  contador=0
                  contador2=0
                  contador3=0
                  try:
                     while cond3==0:
                        while cond2==0:
                           while cond==0:
                              if contador>=32:
                                 cond=1
                              else:
                                 resultante1=res['0']['0']['1'][str(n1)]-res['0']['2']['1'][str(n3)]
                                 if resultante1>=rango:
```

```
                        n3=n3+1
                        contador=contador+1
                    elif resultante1<=-rango:
                        n1=n1+1
                        contador=contador+1
                    else:
                        cond=1
                        contador=contador+1
                if contador2>=32:
                    cond2=1
                else:
                    resultante2=res['0']['2']['1'][str(n2)]-res['0']['3']['1'][str(n4)]
                    if resultante2>=rango:
                        n4=n4+1
                        contador2=contador2+1
                    elif resultante2<=-rango:
                        n2=n2+1
                        contador2=contador2+1
                    else:
                        cond2=1
                        contador2=contador2+1
            if contador3>=16:
                cond3=1
            else:
                resultante3=res['0']['0']['1'][str(n1)]-res['0']['2']['1'][str(n3)]
                if resultante3>=rango:
                    n3=n3+1
                    contador3=contador3+1
                    contador=0
                    contador2=0
                    cond2=0
                elif resultante3<=-rango:
                    n1=n1+1
                    contador3=contador3+1
                    contador=0
                    contador2=0
                    cond=0
                else:
                    contador3=contador3+1
                    cond3=1
        if contador3<=2:
            self.simulacion.delete(cadera)
            anguloC=res['0']['3']['0'][str(n4)]
            anguloC=math.radians(anguloC)
            anguloC=math.sin(anguloC)
            sumatoriaC=(posYC1-posYC2)*anguloC
            posXC1=posXC1-sumatoriaC
cadera=self.simulacion.create_line(posXC1,posYC1,posXC2,posYC2,fill='black',width=5)
            self.simulacion.delete(muslo)
            self.simulacion.delete(bolarodilla)
```

```
                        anguloM=res['0']['2']['0'][str(n3)]
                        anguloM=math.radians(anguloM)
                        anguloM=math.sin(anguloM)
                        sumatoriaM=(posYM1-posYM2)*anguloM
                        posXM2=posXM2+sumatoriaM
muslo=self.simulacion.create_line(posXM1,posYM1,posXM2,posYM2,fill='black',width=5)
                        bolarodilla=self.simulacion.create_oval(posXM2-5,posYM2-
5,posXM2+5,posYM2+5,fill='black',width=5)
                        self.simulacion.delete(pantorrilla)
                        self.simulacion.delete(bolaTobillo)
                        anguloP=res['0']['1']['0'][str(n2)]
                        anguloP=math.radians(anguloP)
                        anguloP=math.sin(anguloP)
                        sumatoriaP=(posYP1-posYP2)*anguloP
                        posXP2=posXP2-sumatoriaP
                        posXP1=posXM2
pantorrilla=self.simulacion.create_line(posXP1,posYP1,posXP2,posYP2,fill='black',width=5)
                        bolaTobillo=self.simulacion.create_oval(posXP2-5,posYP2-
5,posXP2+5,posYP2+5,fill='black',width=5)
                        self.simulacion.delete(pie)
                        anguloPi=res['0']['0']['0'][str(n1)]
                        anguloPi=math.radians(anguloPi)
                        anguloPi=math.sin(anguloPi)
                        sumatoriaPi=(posXPi1-posXPi2)*anguloPi
                        posYPi2=posYPi2-sumatoriaPi
                        posXPi1=posXP2
                        posXPi2=posXPi1+30
pie=self.simulacion.create_line(posXPi1,posYPi1,posXPi2,posYPi2,fill='black',width=5)
                        self.Simulacion.update()
                        posXC1=120
                        posXM2=120
                        posXP2=120
                        posYPi2=240

                        time.sleep(0.1)
                    except:
                        pass
            else:
                messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
        else:
            messagebox.showinfo("INFORMACION","Seleccioné un dato")
    def SimulacionPI(self):
        if self.Nombre.__getitem__('text') !='':
            if self.DatosSensor.__getitem__('text') =='Datos disponibles':
                res=self.FiltroKalmad(80,60)
                self.simulacion = Canvas(self.Simulacion,bg='White')
                self.tituloSDerecha=tk.Label(self.Simulacion,text="Simulación Pierna
Izquierda",font=("Georgia",12),fg="#000000", bg="#FFFFFF")
                self.simulacion.place(relx=0.30,rely=0.05,relheight=0.9, relwidth=0.4)
                self.tituloSDerecha.place(relx=0.35,rely=0.05)
```

```
            posXC1=120
            posXC2=120
            posYC1=50
            posYC2=110
            posXM1=120
            posXM2=120
            posYM1=110
            posYM2=180
            posXP1=120
            posXP2=120
            posYP1=180
            posYP2=240
            posXPi1=120
            posXPi2=150
            posYPi1=240
            posYPi2=240
            cadera=self.simulacion.create_line(posXC1,posYC1,posXC2,posYC2,fill='black',width=5)
            bolacadera=self.simulacion.create_oval(posXC1-5,posYC2-
5,posXC1+5,posYC2+5,fill='black',width=5)
            muslo=self.simulacion.create_line(posXM1,posYM1,posXM2,posYM2,fill='black',width=5)
            bolarodilla=self.simulacion.create_oval(posXM2-5,posYM2-
5,posXM2+5,posYM2+5,fill='black',width=5)
pantorrilla=self.simulacion.create_line(posXP1,posYP1,posXP2,posYP2,fill='black',width=5)
            bolaTobillo=self.simulacion.create_oval(posXP2-5,posYP2-
5,posXP2+5,posYP2+5,fill='black',width=5)
            pie=self.simulacion.create_line(posXPi1,posYPi1,posXPi2,posYPi2,fill='black',width=5)
            self.Simulacion.update()
            time.sleep(1)
            muestrasC=len(res['1']['3']['0'])
            muestrasM=len(res['1']['2']['0'])
            muestrasP=len(res['1']['1']['0'])
            muestrasT=len(res['1']['0']['0'])
            if muestrasC<muestrasM and muestrasC<muestrasP and muestrasC<muestrasT:
               selctor=res['1']['3']['0']
            elif muestrasM<muestrasP and muestrasC<muestrasT:
               selctor=res['1']['2']['0']
            elif muestrasP<muestrasT:
               selctor=res['1']['1']['0']
            else:
               selctor=res['1']['0']['0']
            rango=300
            contador=0
            n1=0
            n2=0
            n3=0
            n4=0
            for n in selctor:
                  n1=1+n1
                  n2=1+n2
                  n3=1+n3
```

```
n4=1+n4
cond=0
cond2=0
cond3=0
contador=0
contador2=0
contador3=0
try:
    while cond3==0:
        while cond2==0:
            while cond==0:
                if contador>=32:
                    cond=1
                else:
                    resultante1=res['1']['0']['1'][str(n1)]-res['1']['2']['1'][str(n3)]
                    if resultante1>=rango:
                        n3=n3+1
                        contador=contador+1
                    elif resultante1<=-rango:
                        n1=n1+1
                        contador=contador+1
                    else:
                        cond=1
                        contador=contador+1
            if contador2>=32:
                cond2=1
            else:
                resultante2=res['1']['2']['1'][str(n2)]-res['1']['3']['1'][str(n4)]
                if resultante2>=rango:
                    n4=n4+1
                    contador2=contador2+1
                elif resultante2<=-rango:
                    n2=n2+1
                    contador2=contador2+1
                else:
                    cond2=1
                    contador2=contador2+1
        if contador3>=16:
            cond3=1
        else:
            resultante3=res['1']['0']['1'][str(n1)]-res['1']['2']['1'][str(n3)]
            if resultante3>=rango:
                n3=n3+1
                contador3=contador3+1
                contador=0
                contador2=0
                cond2=0
            elif resultante3<=-rango:
                n1=n1+1
                contador3=contador3+1
```

```
                        contador=0
                        contador2=0
                        cond=0
                    else:
                        contador3=contador3+1
                        cond3=1
                if contador3<=2:
                    self.simulacion.delete(cadera)
                    anguloC=res['1']['3']['0'][str(n4)]
                    anguloC=math.radians(anguloC)
                    anguloC=math.sin(anguloC)
                    sumatoriaC=(posYC1-posYC2)*anguloC
                    posXC1=posXC1-sumatoriaC
cadera=self.simulacion.create_line(posXC1,posYC1,posXC2,posYC2,fill='black',width=5)
                    self.simulacion.delete(muslo)
                    self.simulacion.delete(bolarodilla)
                    anguloM=res['1']['2']['0'][str(n3)]
                    anguloM=math.radians(anguloM)
                    anguloM=math.sin(anguloM)
                    sumatoriaM=(posYM1-posYM2)*anguloM
                    posXM2=posXM2+sumatoriaM
muslo=self.simulacion.create_line(posXM1,posYM1,posXM2,posYM2,fill='black',width=5)
                    bolarodilla=self.simulacion.create_oval(posXM2-5,posYM2-
5,posXM2+5,posYM2+5,fill='black',width=5)
                    self.simulacion.delete(pantorrilla)
                    self.simulacion.delete(bolaTobillo)
                    anguloP=res['1']['1']['0'][str(n2)]
                    anguloP=math.radians(anguloP)
                    anguloP=math.sin(anguloP)
                    sumatoriaP=(posYP1-posYP2)*anguloP
                    posXP2=posXP2-sumatoriaP
                    posXP1=posXM2
pantorrilla=self.simulacion.create_line(posXP1,posYP1,posXP2,posYP2,fill='black',width=5)
                    bolaTobillo=self.simulacion.create_oval(posXP2-5,posYP2-
5,posXP2+5,posYP2+5,fill='black',width=5)
                    self.simulacion.delete(pie)
                    anguloPi=res['1']['0']['0'][str(n1)]
                    anguloPi=math.radians(anguloPi)
                    anguloPi=math.sin(anguloPi)
                    sumatoriaPi=(posXPi1-posXPi2)*anguloPi
                    posYPi2=posYPi2-sumatoriaPi
                    posXPi1=posXP2
                    posXPi2=posXPi1+30
pie=self.simulacion.create_line(posXPi1,posYPi1,posXPi2,posYPi2,fill='black',width=5)
                    self.Simulacion.update()
                    posXC1=120
                    posXM2=120
                    posXP2=120
                    posYPi2=240
                    time.sleep(0.1)
```

```
                except:
                    pass
            else:
                messagebox.showinfo("INFORMACION","Primero realice una toma de angulos")
        else:
            messagebox.showinfo("INFORMACION","Seleccioné un dato")
    def Scan_Com(self):
        port_list = list(serial.tools.list_ports.comports())
        if len(port_list) == 0:
            resp=messagebox.askretrycancel("INFORMACION","Conecte un puerto")
            if resp== True:
                self.Scan_Com()
            else:
                self.VentanaDatos.destroy()
        for i in range(0,len(port_list)):
            self.Puertos.configure(values=port_list)
            self.Puertos.current(i)
    def cerrar(self):
        self.VentanaIngresoDatos.destroy()
if __name__=='__main__':
    window=Tk()
    aplication=Product(window)
    window.mainloop()
```