DISEÑO E IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA EL RASTREO DE DATOS OBTENIDOS DE SENSORES INERCIALES.

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

DISEÑO E IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA EL RASTREO DE DATOS OBTENIDOS DE SENSORES INERCIALES.

Trabajo de graduación en modalidad de trabajo de tesis presentado por
Erick Eduardo De Mata Calderón
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2018

Vo.Bo.:

(f) _____

Msc. José Eduardo Morales Espinoza

Tribunal Examinador:

(f) _____

Msc. José Eduardo Morales Espinoza

(f) _____

Ing. Pablo Daniel Mazariegos de la Cerda

(f) _____

Msc. Carlos Alberto Esquit Hernández

Fecha de aprobación: Guatemala, 20 de junio del 2018.

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

En el presente trabajo de graduación se explica el proceso que se llevó acabo para realizar una interfaz gráfica de usuario capaz de medir posición, velocidad y aceleración lineal en los tres ejes coordenados, para atletas de alto rendimiento. Para el desarrollo e implementación del trabajo de graduación se requirió hacer una investigación previa para estudiar y comprender los principios básicos de la ciencia aplicada al deporte. Posteriormente se realizaron visitas técnicas a diferentes instituciones que ayudan al desarrollo del deporte guatemalteco, en el cual se obtuvo una mejor perspectiva de las necesidades y requerimientos de los deportistas nacionales.

La implementación de los sensores inerciales se realiza de manera individual a través de la interfaz gráfica, lo cual brinda versatilidad en la aplicación de diversos deportes. Cualquier sensor de forma individual puede ser vinculado a cualquier extremidad o parte del cuerpo de interés, logrando de esta manera crear una red de sensores que se comunican inalámbricamente.

La finalidad de obtener la información generada por los sensores es proporcionar a los atletas retroalimentación de un entrenamiento, que ayude a mejorar la técnica y rendimiento. Se generó una base de datos confiable y sencilla, la cual es apta para realizar estudios biomecánicos de los atletas, creando un archivo identificado por usuario y fecha.

Dicha interfaz gráfica es capaz de realizar funciones similares a otros softwares y dispositivos comerciales. La presente interfaz gráfica de usuario fue desarrollada con el fin de reducir costos, utilizando software gratuito.

I. INTRODUCCIÓN

El deporte es una actividad física de carácter competitivo, que se practica de forma recreativa o profesional. La práctica del mismo debe ser constante a través del entrenamiento, y estar sujeto a ciertas normas dependiendo del deporte que se practique. Durante los últimos años es común que tanto entrenadores de alto rendimiento como investigadores del deporte, se formulen interrogantes tales como: ¿Cuáles son las principales causas del bajo rendimiento de un atleta?, ¿Qué efecto tendrá en el desarrollo de los atletas una buena alimentación?, ¿Cómo hacer para que el deportista presente un alto rendimiento y evite lesiones por una mala práctica? Todas estas interrogantes se han ido solucionando con la implementación de la tecnología y la ciencia aplicada en el deporte (Suárez, 2009).

En la actualidad la implementación de tecnología en el deporte es una práctica común, ya que desde su implementación los resultados y el avance de los deportistas se ha manifestado de forma exponencial. Por dicho motivo en la actualidad la ciencia aplicada al deporte es una herramienta útil, ya que conforman un conjunto de disciplinas que se integran para ayudar al desarrollo óptimo del deportista (Ruiz, 2012).

La implementación de tecnología en el deporte ha significado un crecimiento y desarrollo en disciplinas como la psicología, la biomecánica, la biología, la nutrición, la sociología, la física, las matemáticas, entre otras (Suárez, 2009).

La implementación de estudios biomecánicos en las actividades deportivas ha mejorado la eficiencia del deporte de alto rendimiento, implementando una técnica de alta calidad en los movimientos a partir de conocimientos científicos y equipos de diagnóstico (Ruiz, 2012).

En el deporte guatemalteco se desea implementar tecnología que ayude al desarrollo de los deportistas de alto rendimiento, con el fin de lograr obtener mejores resultados a nivel internacional. Con el objetivo de ayudar a que nuestros deportistas alcancen un nivel óptimo se ha planteado realizar una serie de sensores inerciales, que sean funcionales, económicamente accesibles y que presenten una interfaz agradable al usuario. En el presente trabajo de graduación se realizó el desarrollo e implementación de una red de sensores inalámbricos con un alcance máximo de 100 metros, en los cuales se obtiene el cambio lineal de la posición, velocidad y aceleración en tiempo real, generando una base de datos con la información obtenida por los sensores desarrollados. La interfaz gráfica de usuario es la encargada de procesar la información y mostrar al usuario resultados puntuales de los datos obtenidos por los sensores inerciales.

II. OBJETIVOS

A. OBJETIVO GENERAL

1. Diseñar e implementar una interfaz gráfica que sea capaz de representar por medio de gráficas lineales los cambios en posición, velocidad y aceleración de los sensores inerciales, generando una base de datos de las variables mencionadas para brindar al usuario una retroalimentación que ayude a mejorar el desempeño de los atletas.

B. OBJETIVOS ESPECÍFICOS

1. Representar por medio de gráficas lineales los cambios en posición, velocidad y aceleración del usuario en tiempo real.

2. Proporcionar al usuario un reporte de datos puntuales de posición, velocidad y aceleración máxima, mínima y promedio por sensor de movimiento de rastreo en 3D.

3. Implementar una red de comunicación de radio frecuencias entre sensores de rastreo de movimiento en 3D para la obtención de posición, velocidad y aceleración.

4. Representar en un espacio de tres dimensiones el cambio de la posición obtenido por el sensor inercial a lo largo de una trayectoria.

5. Generar una base de datos en la cual se almacene la posición, velocidad y aceleración de los sensores inerciales.

III. JUSTIFICACIÓN

En la actualidad en el deporte guatemalteco se cuentan con pocos recursos tecnológicos accesibles a los deportistas en el área de biomecánica. Según el Comité Olímpico Internacional potencias mundiales en el deporte como Estados Unidos, Alemania, Rusia e Inglaterra incluyen recursos tecnológicos de biomecánica en sus atletas para mejorar día con día su rendimiento y seguir en la cúspide del deporte mundial. Un ejemplo de esto se vio reflejado en los últimos juegos olímpicos realizados en Río de Janeiro en el año 2016; en el cual Estados Unidos se adjudicó 46 medallas de oro, Inglaterra 27, Rusia 19 y Alemania 17 respectivamente. Debido a esta situación se ha planteado el objetivo de desarrollar sensores de rastreo de movimiento en 3D de bajo costo. Al realizar sensores de movimiento de bajo costo se estará brindando una mayor disponibilidad de recursos tecnológicos a los deportistas, debido a que existe una demanda alta de atletas que hoy en día no implementan tecnología para mejorar su rendimiento, por ejemplo, en disciplinas como: marcha, judo, levantamiento de pesas, entre otros.

Con la implementación de los sensores de rastreo de movimiento en 3D, tanto deportistas como entrenadores tendrán la opción de realizar un análisis biomecánico durante un entrenamiento, para obtener resultados tangibles en tiempo real de posiciones, velocidades y aceleraciones. Para un correcto uso de las variables de medición se debe contar con una interfaz gráfica de usuario que realice un análisis y procesamiento de información adecuado y sencillo, el cual debe ser capaz de obtener resultados puntuales y plasmarlos de forma efectiva, sencilla y amigable al usuario, con el fin de mejorar la técnica y el desempeño de los atletas guatemaltecos.

La sencillez y efectividad en el módulo de interfaz gráfica de usuario es de suma importancia, ya que el usuario debe contar con poca o nula capacitación para poder interactuar con la interfaz gráfica e interpretar y transmitir de forma efectiva los resultados obtenidos por los sensores de rastrero de movimiento en 3D. La finalidad del presente módulo se basa en desarrollar e implementar una interfaz gráfica de usuario que brinde a un entrenador una herramienta de retroalimentación utilizada durante una sesión de entrenamiento, para proporcionar información valiosa a los atletas, con el objetivo de mejorar el rendimiento del deporte guatemalteco, trazándose nuevas metas y objetivos.

IV. MARCO TEÓRICO

A. BIOMECÁNICA

La biomecánica es una disciplina científica que tiene como principal objetivo el estudio de las estructuras de carácter mecánico que existen en los seres vivos, específicamente en el cuerpo humano. La biomecánica se apoya de diversas ramas de la ciencia como la mecánica, ingeniería, anatomía, fisiología, entre otras con el objetivo de estudiar el comportamiento del cuerpo humano y resolver problemas derivados de un comportamiento anómalo o una mala praxis (Ruiz, 2012).

En la actualidad la biomecánica es una de las principales herramientas deportivas, ya que todos los deportes involucran movimiento, y los resultados en todos los deportes dependen de una buena técnica. Gracias a la implementación de la tecnología en el deporte y de los estudios biomecánicos los atletas han logrado un desarrollo exponencial, ya que se llevan las capacidades físico atléticas al límite (Izquierdo, 2008).

La biomecánica está presente en diversos ámbitos y aplicaciones, en la actualidad las ramas más desarrolladas de la biomecánica son las siguientes:

- Biomecánica médica
- Biomecánica deportiva
- Biomecánica ocupacional

1. Biomecánica médica. Evalúa diversas patologías que aquejan al cuerpo humano, específicamente del sistema musculo esquelético, con la finalidad de generar soluciones capaces de evaluar, corregir y rehabilitar lesiones musculares, fracturas, entre otros. Dentro de la biomecánica médica existen diversas divisiones que se especializan en brindar una solución eficiente a los problemas que aquejan al sistema muso esquelético (Ruiz, 2012).

a. Biomecánica aplicada a la traumatología. Estudia los principios mecánicos del movimiento para determinar cómo estos pueden causar lesiones en los deportistas, por ejemplo, las cargas máximas que pueden soportar huesos, tendones y ligamentos para no exceder los límites de tolerancia permitidos en el cuerpo humano (Izquierdo, 2008).

b. Biomecánica aplicada a la rehabilitación. Estudia todos aquellos ejercicios que poseen un carácter de rehabilitación, en el cual se toma en cuenta la dirección de fuerzas ejercidas y los momentos generados en torno a las articulaciones. El fin primordial de la rehabilitación es la restauración de todas aquellas funciones deterioradas y la compensación de las mismas (Izquierdo, 2008).

c. Biomecánica aplicada a la fisiología. En esta subdivisión se aborda el estudio de la mecánica de fluidos, así como la relación que existe entre un nervio motor y el músculo en la coordinación de movimientos, además de las implicaciones de los procesos fisiológicos del cuerpo humano sobre las habilidades motoras. La ergometría es sumamente importante en las actividades físicas, ya que mide el esfuerzo biológicamente normado utilizando una metodología de medición mucho más exacta y comparable para realizar un análisis de las funciones de esfuerzo del cuerpo humano (Izquierdo, 2008).

d. Biomecánica ortopédica. Estudia la implantación y adaptación de prótesis al cuerpo humano. En ortopedia uno de los problemas más comunes y con tratamientos invasivos es la degeneración de cartílago, lo cual conlleva a una artroplastia. Los problemas articulares en el cuerpo humano plantean desafíos científicos importantes ya que modifican toda la biomecánica articular y por ende la funcionalidad del cuerpo (Izquierdo, 2008).

Las posibilidades que estas ciencias ofrecen para mejorar la salud y la calidad de vida han generado una continua expansión, siendo capaces de aportar soluciones científicas para la capacidad motora humana, técnicas de rehabilitación a los discapacitados, prótesis, entre otras (Izquierdo, 2008).

Figura 1. Biomecánica médica.

(Biomec, 2017)

En la Figura 1 se puede observar la aplicación de técnicas de rehabilitación para personas que poseen discapacidad física en las extremidades inferiores.

2. Biomecánica deportiva. Son estudios enfocados a métodos mecánicos designados a un análisis de la estructura y función del sistema de locomoción del cuerpo humano. Dentro de los objetivos de la biomecánica deportiva está el evaluar la calidad de los movimientos realizados durante la actividad física, evaluar la técnica deportiva ejecutada y reconocer actividades o movimientos potencialmente peligrosos para los atletas.

Para estudiar el movimiento del cuerpo humano, la biomecánica deportiva utiliza dos procedimientos: el análisis cualitativo y cuantitativo (Suárez, 2009).

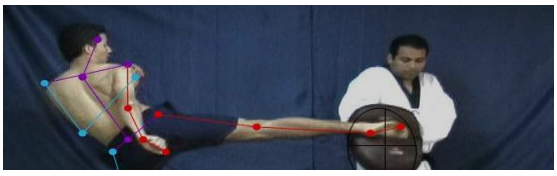
a. Análisis cuantitativo. En este tipo de análisis se realiza una descripción de los movimientos del cuerpo o sus extremidades en términos numéricos. La cuantificación de las características del movimiento del cuerpo humano ayuda a eliminar descripciones subjetivas o estimadas, ya que los datos obtenidos en el presente análisis se realizan mediante instrumentos de medición. Gracias a esto el observador puede explicar y describir con mucha certeza el estado específico de un movimiento. Usualmente el análisis cuantitativo tiene el inconveniente de no ser un análisis económico por la implementación de instrumentos de medición. Dentro de los instrumentos de medición más utilizados para este tipo de análisis se encuentran sensores inerciales, sensores de electromiografía, plataformas de presión, entre otros (Suárez, 2009).

b. Análisis cualitativo. Análisis en el cual se describe un movimiento en términos no cuantitativos; sin embargo, los datos de un análisis cualitativo pueden ser sustentados o aprobados por un análisis cuantitativo. Por dicha razón estos dos tipos de análisis generalmente están relacionados entre sí.

Muchos de los proyectos de investigación son planteados desde un punto de vista cualitativo como se muestra en la Figura 2, la evaluación de este tipo de análisis se basa en la habilidad de un entrenador u observador para reconocer los movimientos críticos que pueden existir en la ejecución de un movimiento o gesto deportivo.

Tanto los análisis cuantitativos como cualitativos proveen información valiosa acerca de la ejecución de un movimiento; sin embargo, el análisis cualitativo es el método predominante utilizado por los entrenadores en el estudio de los movimientos ejecutados por los atletas. Los videos o filmaciones son empleadas en un análisis cualitativo, debido a que es una herramienta que provee a los atletas una retroalimentación visual de sus movimientos, ayudando corregir errores ejecutados durante la práctica. Un equipo de video es una herramienta indispensable en este tipo de análisis debido a que durante la ejecución de movimientos existen múltiples detalles que pueden ser obviados por el observador debido a la velocidad en la que suceden los movimientos y a la baja velocidad de captación del ojo humano (Suárez, 2009).

Figura 2. Análisis biomecánico en entrenamiento de atletas.

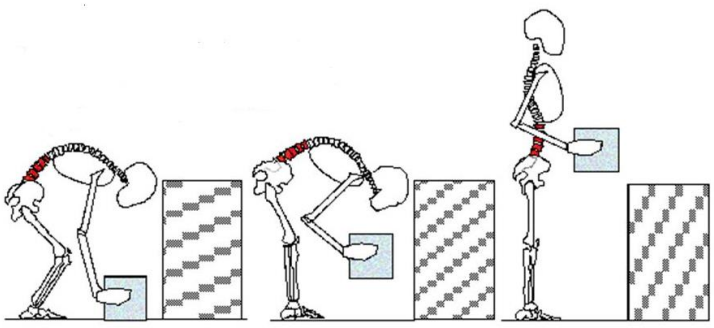


(Suárez, 2009)

3. Biomecánica ocupacional. Estudia la relación mecánica que el cuerpo humano realiza o sostiene con los elementos que interactúan en diversos ámbitos para adaptarlos a sus necesidades y capacidades. La biomecánica ocupacional se encarga de analizar la relación que existe entre la ergonomía y la biomecánica. En un principio la biomecánica tuvo muy poca influencia en el terreno laboral, ya que disciplinas como la fisiología y la psicología tuvieron un mayor peso en los estudios ergonómicos y esto se debió a que la biomecánica no poseía una metodología bien definida para estudios ergonómicos. Sin embargo, con el transcurso de los años esto ha ido evolucionando y la biomecánica ocupacional tiene mucho que aportar en el estudio de la fatiga, bajo rendimiento y lesión en un trabajador, como se muestra en la Figura 3 (Daza, 2007).

Cuando los movimientos requeridos por un medio externo como herramientas, máquinas y utensilios no son compatibles con los movimientos naturales del cuerpo humano, surge la fatiga, posteriormente el trabajador obtiene un bajo rendimiento, lo cual en la mayoría de los casos puede ocasionar una lesión (Daza, 2007).

Figura 3. Biomecánica ocupacional.



(Daza, 2007)

B. IMPLEMENTACIÓN DE SOFTWARE E INTERFAZ GRÁFICA

1. Python. Python es un lenguaje programación de alto nivel, interpretado y multipropósito, lo cual lo hace uno de los lenguajes de programación más versátiles y utilizados en la actualidad. Python puede ser implementado en diversas plataformas y sistemas operativos, entre los cuales podemos destacar Windows, Mac OS y Linux. La principal ventaja que posee Python es que su lenguaje de programación es de código abierto, es decir que cualquier persona puede contribuir a su desarrollo y divulgación. Por lo mencionado anteriormente el intérprete de Python se distribuye de forma gratuita para las plataformas mencionadas anteriormente (Coutinho, 2016).

Python es un lenguaje de escritura independiente de plataforma, preparado y diseñado para realizar cualquier tipo de programa, por ejemplo, aplicaciones científicas, comunicaciones de red, aplicaciones de escritorio con interfaz gráfica de usuario, juegos, aplicaciones para teléfonos inteligentes y aplicaciones web.

Dicho lenguaje de programación es interpretado, lo que significa que no es necesario compilar el código fuente para poder ejecutarlo, lo cual proporciona una mayor rapidez en el desarrollo de software. Otra de las grandes ventajas que presenta Python es que posee diversas librerías, tipos de datos y funciones incorporadas en el propio lenguaje, ayudando a los programadores a minimizar el tiempo de desarrollo. El lenguaje de programación Python es interactivo, ya que dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible al programador, lo cual ayuda a entender de mejor manera el lenguaje y probar ciertos resultados de la ejecución por tramos de código fuente rápidamente (Gonzáles, 2007).

La programación orientada a objetos de Python es un paradigma en el cual se propone modelar todo en base a clase y objetos. El paradigma consiste en implementar el uso de conceptos de herencia, cohesión, abstracción, polimorfismo y encapsulamiento (Coutinho, 2016).

2. Interfaz gráfica de usuario.

La interfaz gráfica de usuario es un método utilizado para facilitar la interacción del usuario con el ordenador o computadora a través de la implementación de un conjunto de imágenes y objetos pictóricos como ventanas, íconos, menús, botones, entre otros además del texto. Las interfaces gráficas surgieron como la evolución de las interfaces de las líneas de comandos que se utilizaban en los primeros sistemas operativos. Los entornos de escritorio como Windows, Linux y Mac OS son algunos de los ejemplos más conocidos de interfaz gráfica de usuario (Albornoz, Miranda y Berón, 2014).

Una interfaz es el dispositivo que permite establecer comunicación entre dos sistemas que no hablan el mismo lenguaje, es decir es un juego de conexiones y dispositivos que facilitan la comunicación entre dos sistemas como se presenta en la interacción humano computadora (Luna, 2004).

Al momento de utilizar una interfaz gráfica todo usuario requiere un tiempo de adaptación a la misma, mientras menor sea el tiempo dedicado a realizar esta acción las posibilidades de éxito en la interfaz aumentarán. Generalmente un usuario percibe la interfaz gráfica de manera no grata cuando los errores se hacen presentes y crean cierta confusión en los usuarios. Dentro de los errores más comunes que se perciben en una interfaz gráfica mal diseñada se encuentran: mensajes poco claros, dificultad al encontrar la información relevante y la inexistencia de jerarquías o temáticas que mantengan una secuencia lógica del funcionamiento de la interfaz gráfica de usuario (Luna, 2004).

Para poder diseñar y desarrollar una interfaz gráfica de manera exitosa, se debe tomar en cuanta diversas características, cualidades y particularidades que estén relacionados con un óptimo funcionamiento de la interfaz gráfica, con la estética y con el despliegue de información. Dichas características se presentarán a continuación:

a. Familiaridad del usuario. La interfaz gráfica debe utilizar términos o imágenes que sean conocidos por el usuario, es decir que estén relacionados con el ámbito de trabajo de los potenciales usuarios. La interfaz gráfica no debe ser diseñada en función de una conveniente implementación, sino debe ir diseñada y orientada a usuarios que pueden tener la libertad de elegir y adaptarse a la interfaz gráfica. Por ejemplo, en un sistema diseñado para oficinas los conceptos más recurrentes y utilizados son cartas, documentos, folders, entre otros, entonces se diseña e implementa una interfaz que tenga congruencia con los conceptos más utilizados, en la Figura 4 se puede observar una interfaz gráfica que cumple con el concepto de familiaridad de usuario (Vásquez, Yánez y Pro Concepción, 2011).

Figura 4. Skype ejemplo de una interfaz gráfica que posee familiaridad con el usuario.

(Skype, 2018)

b. Uniformidad de la interfaz. Toda interfaz gráfica debe poseer menús, acciones y comandos con el mismo formato, ya que las interfaces uniformes reducen el tiempo de aprendizaje, lo cual hace que la interfaz sea efectiva. Por lo tanto, el aprendizaje en un comando o acción dentro de la aplicación debe ser aplicable en otras partes del sistema o en acciones relacionadas. Un ejemplo de uniformidad en una interfaz gráfica se muestra en la Figura 5, la cual corresponde a la interfaz gráfica de LG, mostrando sus opciones en forma de íconos con su respectivo nombre, la promoción de cada producto se realiza de la misma forma, lo cual agrega uniformidad. De manera similar, el comportamiento del botón de regresar funciona de la misma manera en todas las pantallas, ya está acción siempre regresará a la acción anterior (Vásquez, Yánez y Pro Concepción, 2011).

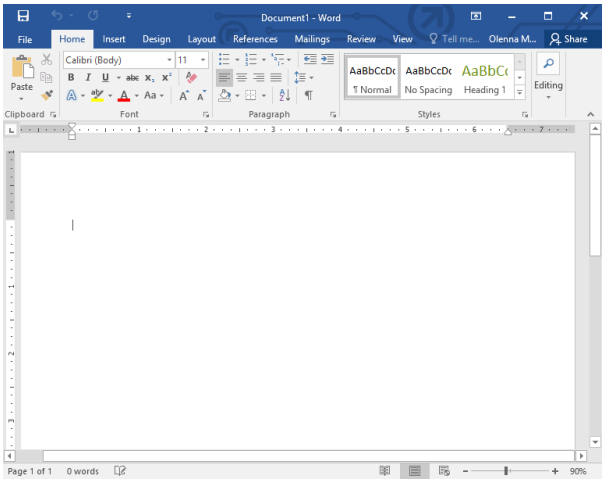
Figura 5. Aplicación de LG para promocionar sus aplicaciones.

(LG Electronics, 2017)

c. Mínima sorpresa. El comportamiento de la interfaz gráfica no debe mostrar situaciones inesperadas o anómalas, ya que ante este tipo de situaciones el usuario puede mostrar cierta irritabilidad o confusión, y por lo tanto perder interés en utilizar la aplicación. La solución para evitar la mínima sorpresa es asegurar que las acciones comparables tengan aspectos comparables para crear un modelo mental de la forma en que trabaja el sistema (Vásquez, Yánez y Pro Concepción, 2011).

d. Funcionalidad. La interfaz debe permitir realizar más de una tarea al mismo tiempo, por ejemplo, imprimir y buscar un archivo al mismo tiempo, además de brindar una respuesta inmediata al accionar del usuario, ofrecer actualizaciones de la aplicación, implementación de atajos o accesos rápidos, entre otros. Un ejemplo puntual de la funcionalidad en una interfaz gráfica de usuario es la implementada por el software Word como se muestra en la Figura 6, ya que permite todas las características mencionadas previamente (Albornoz, 2014).

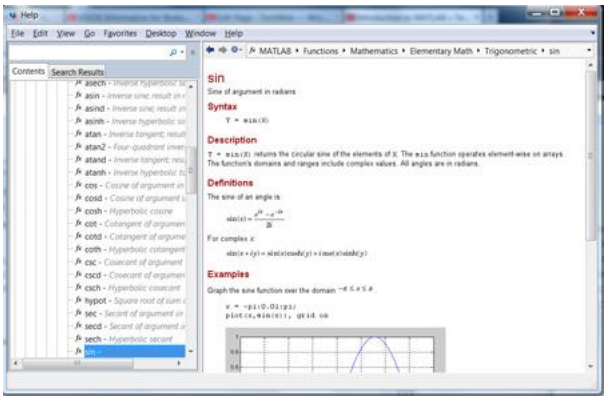
Figura 6. Ejemplo de funcionalidad de una interfaz gráfica de usuario.



e. Efectividad. Una interfaz gráfica de usuario es efectiva si el usuario encuentra entre todas las opciones las tareas que desea realizar, sintiendo conformidad al utilizar la misma. Dicho concepto se refiere al grado en el cual una interfaz gráfica realiza sus funciones con un mínimo uso de recursos de la máquina en donde se ejecuta la aplicación. En consecuencia, las variables de medición utilizadas para dicha característica se enfocan en análisis cuantitativos. Si se realizó una correcta selección de hardware y software para el desarrollo e implementación de una interfaz gráfica de usuario la experiencia del mismo será satisfactoria (Vásquez, Yánez y Pro Concepción, 2011).

f. Guía de usuario. Toda aplicación debe proporcionar al usuario una asistencia o ayuda. Esta es una herramienta que ayuda al usuario a saber qué hacer cuando se cometen errores al utilizar una aplicación, o brindar más información acerca de las tareas que se pueden ejecutar dentro de la interfaz. En la Figura 7 se muestra la guía de usuario del software Matlab, la cual contiene toda la documentación pertinente para desarrollar análisis de datos de manera eficiente (Albornoz, 2014).

Figura 7. Ayuda a los usuarios en la interfaz gráfica de Matlab.



g. Portabilidad. La interfaz gráfica de usuario debe poseer la misma apariencia en cualquier sistema operativo (Albornoz, 2014).

h. Robustez. Es un parámetro fundamental dentro de cualquier aplicación, ya que debe saber responder ante diversas situaciones; ya sean inesperadas o incluso destructivas. Un claro ejemplo de ello es el manejo de errores y alertas (Albornoz, 2014).

i. Estética. Es sumamente importante que la interfaz gráfica de usuario posea un diseño visualmente agradable. Esto se consigue eligiendo un buen tamaño, color y tipo de fuente que ayuden a la legibilidad e interpretación adecuada de la información presentada al usuario final. Los colores y la

combinación de los mismos son muy importante, ya que se debe resaltar la información relevante sin sobrecargar la visión del usuario (Albornoz, 2014). La Figura 8 es un ejemplo de una correcta implementación de la estética en la interfaz gráfica.

Figura 8.Ejemplo de una interfaz gráfica de usuario con excelente estética.

(Aple Inc., 2018)

3. QT. Qt es un entorno de trabajo multiplataforma para el desarrollo de interfaces gráficas de usuario. Qt es desarrollada como un software libre y de código abierto a través de QtProyect, donde tanto programadores como desarrolladores participan para mejorar el rendimiento de la librería. La librería utiliza el lenguaje de programación C++ de forma nativa, aunque existen diversas interfaces para otros lenguajes de programación. Las plataformas soportadas por Qt incluyen Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry entre otros (Qt Company, 2017).

Qt crea interfaces de usuario intuitivas, fluidas y del alto rendimiento. Dicho entorno de trabajo ofrece una estrategia tecnológica que permite crear fácilmente, diseñar y desarrollar aplicaciones a bajo costo, capaces de abarcar diversas áreas de implementación como dispositivos empotrados, interfaces gráficas de usuario, software, internet de las cosas, desarrollo de tecnologías móviles, automoción, automatización e incluso implementación en el área médica. Debido a la diversidad de ámbitos en donde se puede desarrollar e implementar el entorno de trabajo Qt, empresas como LG, Panasonic, Mercedes Benz, AMD, Medec entre otros desarrollan su tecnología en dicho software (Qt Company, 2017).

A continuación, se mostrarán las principales fortalezas que el entorno de trabajo Qt proporciona a sus clientes:

a. Prototipado, desarrollo y despliegue rápido de información. Qt brinda una serie de APIs desarrolladas, exhaustivas y sumamente intuitivas con un entorno de desarrollo profesional que

provee un soporte multiplataforma, lo cual asegura mejor productividad para cualquier tipo de proyecto (Qt Company, 2017).

b. Fácil implementación. Qt es un entorno de trabajo fácil de utilizar y flexible en su desarrollo, posee herramientas de diseño sumamente útiles en las cuales se incluyen controles desarrollados y funcionalidades prediseñadas para un diseño e implementación de interfaces de usuario eficiente (Qt Company, 2017).

c. IDE y herramientas de desarrollo para programadores y diseñadores. El entorno de trabajo Qt permite crear interfaces tanto en 2D como en 3D de manera eficiente, ya que reaccionan rápido al lenguaje interpretado de programación declarativa, diseño imperativo o un desarrollo hibrido de tecnologías. Esto se elige dependiendo de las necesidades del proyecto y del equipo de trabajo con el que se cuente (Qt Company, 2017).

d. Tecnología preparada para el futuro. La plataforma Qt es un entorno de trabajo abierto, extensible y modular, que cuenta con más de un millón de desarrolladores y socios profesionales que habitualmente comparten nuevas tecnologías, en la cual todos pueden tener acceso libre de forma transparente, debido a que no está vinculada únicamente a una plataforma (Qt Company, 2017).

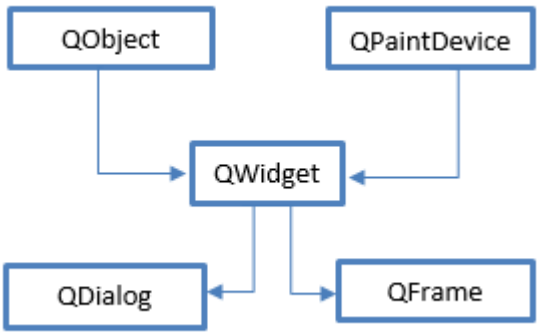
e. Vinculación con otros lenguajes de programación. La mayoría de aplicaciones que usan Qt suelen estar desarrolladas en C++ y QML, sin embargo, existen diferentes interfaces para vincularse con otros lenguajes de programación como Python. La interfaz que Python implementa para vincularse con el entorno de trabajo Qt se denomina PyQt (Qt Company, 2017).

4. PyQt. Es una interfaz de Python para la implementación de la biblioteca gráfica Qt, dicha biblioteca está disponible para Windows, Linux y Mac OS, cada sistema operativo requiere de diferentes licencias para su implementación. PyQt se estructura en Python con un conjunto de módulos que posee más de cuatrocientas clases y seis mil métodos.

PyQt es un conjunto de herramientas diseñada para el desarrollo de interfaz gráfica de usuario. En la Figura 9 se muestran los niveles de jerarquías que poseen las clases principales del software. Dentro del API de PyQt se encuentra la clase QObjetct, dicha clase es la base de todos los objetos Qt y por lo tanto es la clase superior en la jerarquía de clases. QPaintDevice es la clase base para implementar todos los objetos que se pueden pintar o dibujar en la interfaz gráfica. La clase QWidget derivada de la clase QObject y QPaintDevice es la clase encargada de administrar todos los objetos de la interfaz de usuario, de esta clase se derivan las clases QDialog y QFrame, clases cuyo principal fin es contener a objetos como botones, cuadros de texto,

cajas de selección, entre otros. A continuación, se mostrará un diagrama de la jerarquía de las clases principales de PyQt (Blanchette, Summerfield, 2008).

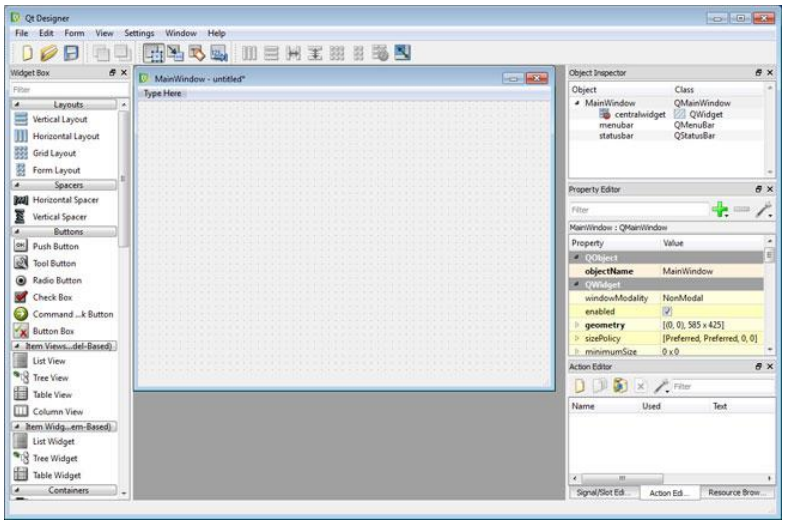
Figura 9.Jerarquía de clases principales PyQt.



5. QtDesigner. Herramienta que contiene PyQt para el desarrollo y creación de interfaz gráfica de usuario. Esta herramienta se caracteriza por ser una herramienta intuitiva y sumamente fácil de utilizar, debido a que, si se desea implementar algún elemento a la interfaz gráfica de usuario, basta con arrastrar y soltar el elemento en el contenedor principal de la interfaz. Esto evita que los componentes se tengan que generar por medio de código fuente, facilitando el diseño al programador (Blanchette, Summerfield, 2008).

El formulario o la interfaz gráfica diseñada en QtDesigner se guarda como un archivo con extensión ".ui" en representación XML de los widgets, es decir elementos que conforman la interfaz gráfica y las propiedades de los mismos. Posteriormente se debe traducir el archivo XML a código Python, esto se realiza a través de la implementación del comando pyuic4 (Blanchette, Summerfield, 2008).

Figura 10. Herramienta utilizada para el desarrollo de interfaces gráficas.

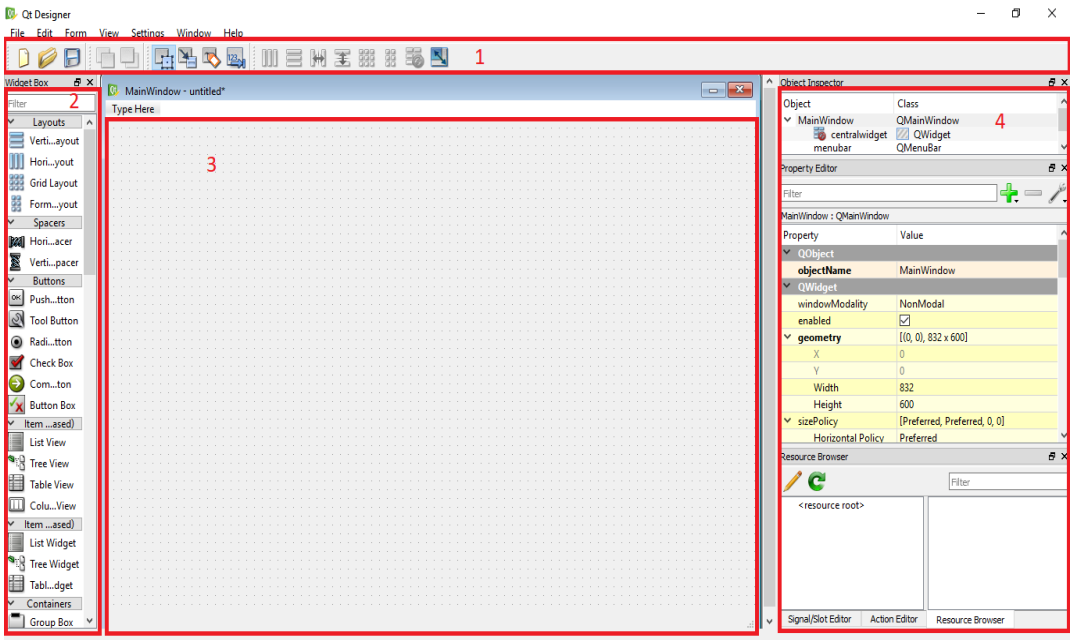


En la Figura 10 se muestra la ventana principal del software QtDesigner para el diseño de interfaz gráfica. Como se puede observar en la Figura 11, en la opción 1 están incluidos íconos de accesos rápidos y funciones especiales que implementa el software para los objetos utilizados dentro de la interfaz gráfica.

En la opción 2 se muestran todos los objetos que pueden ser implementados por la interfaz gráfica, dicha cinta de opciones contiene botones, contenedores, cuadros de texto, tablas, gráficas, temporizadores, entre otros.

En la opción 3 se muestra la ventana principal, es decir el contenedor de todos los objetos que se agreguen al diseño de la interfaz, por último, en la opción 4 se muestran todas las propiedades que se pueden configurar en los objetos implementados a la interfaz gráfica.

Figura 11. Funciones principales del software QtDesigner.



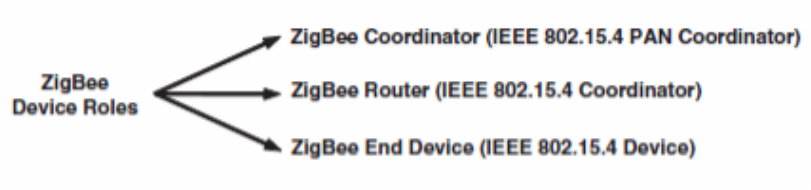
C. COMUNICACIÓN INALÁMBRICA

1. Ziggbee. Es el nombre que recibe el conjunto de protocolos de alto nivel de comunicación inalámbrica basados en radio difusión digital de bajo consumo energético, las cuales utilizan el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN). El objetivo principal son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximizar la vida útil de las baterías (Zigbee Alliance, 2017).

En el estándar ZigBee se definen tres bandas de radio frecuencia diferentes, la banda de 868 MHz utilizada en Europa, la banda de 915 MHz utilizada en América del Norte y la banda de 2.4 GHz se utiliza en el resto del mundo. En términos generales se puede concluir que las aplicaciones de 2.4 GHz permiten un ancho de banda más grande y con más canales. Sin embargo, la coexistencia con otros sistemas que funcionan a 2.4 GHz como WLAN y Bluetooth debe ser considerados para evitar errores de interferencia (Girod, 2012).

El estándar IEEE 802.15.4 y ZigBee definen diferentes tipos de dispositivos como se muestra en la Figura 12, entre los cuales se encuentra el coordinador, router y dispositivo final. El coordinador es el dispositivo más completo, ya que controla el ruteado y la administración de la red. Únicamente hay uno por red. El dispositivo router se encarga de interconectar los dispositivos mediante técnicas de direccionamiento. Finalmente se encuentra el dispositivo final, el cual es un elemento pasivo de la red, ya que no transmite información de manera autónoma; simplemente dispone de la funcionalidad mínima para ser capaz de responder a peticiones de dispositivos superiores como lo son el coordinador y el router (Girod, 2012).

Figura 12. Tipos de dispositivos ZigBee/IEE 802.15.4.



(Girod, 2012)

2. Xbee. Los módulos Xbee son fabricados por la empresa Digi, los cuales proporcionan soluciones integradas que brindan un medio inalámbrico para su interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red IEEE 802.15.4 basado en ZigBee para crear redes punto a punto, punto a multipunto o red tipo malla. Dichos dispositivos fueron diseñados para aplicaciones que requieren un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible (Girod, 2012).

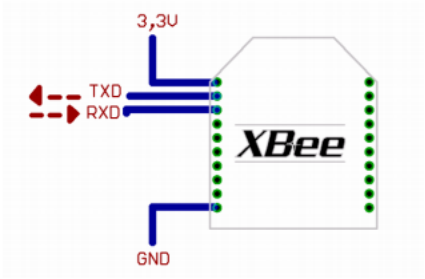
Digi ofrece dos series diferentes de dispositivos, serie 1 y serie 2. Los módulos de la serie 1 y la serie 2 poseen el mismo *pin out*, sin embargo, no son compatibles entre sí, ya que trabajan con diferentes protocolos de comunicación. Los módulos de serie 1 están fabricados para implementar redes punto a punto o punto a multipunto, mientras que los módulos de serie 2 están diseñados para implementar repetidores o una red tipo malla, donde la complejidad es mayor. Sin embargo, los módulos serie 2 también pueden implementar redes punto a punto o punto a multipunto. Ambos módulos pueden ser utilizados en modos AT y API. En la Figura 13 se muestra un módulo Xbee serie 2 (MCI Electronics, 2017).

Figura 13. Xbee serie 2.

(Digi, 2017)

Para realizar una comunicación entre un microcontrolador como Arduino y un módulo Xbee se puede utilizar el módulo UART del microcontrolador, para que en el pin *OUT* del módulo Xbee se conecte con el pin RX del microcontrolador y el pin *IN* del módulo Xbee se conecte con el pin TX del microcontrolador, logrando de esta manera un intercambio de información. En la Figura 14 se puede observar los pines de comunicación del módulo Xbee (Digi, 2017).

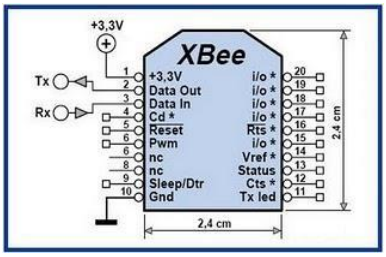
Figura 14. Conexiones mínimas para establecer comunicación entre un módulo Xbee y otro dispositivo.



(Digi, 2017)

Según el fabricante Digi Internacional el módulo de comunicación inalámbrica posee 20 pines como se muestra en la Figura 15. De los 20 pines posee 4 pines son entradas o salidas analógicas, 1 pin de voltaje, 1 pin de referencia o tierra, 1 salida pwm, 1 pin de reset, 2 módulos de comunicación UART, 1 pin de encendido y el resto de pines son entradas digitales (Digi, 2017).

Figura 15. Pines del módulo de radio frecuencia Xbee.



(Digi, 2017)

3. Topologías de red. La topología de una red se define como una cadena de comunicación implementada por nodos que conforman una serie de interconexiones para intercambiar información y comunicarse entre sí. Los módulos Xbee soportan cuatro topologías de red diferente, entre los cuales se encuentran la topología punto a punto, estrella, malla y red tipo árbol como indica la hoja de datos del fabricante Xbee.

a. Topología punto a punto. La topología punto a punto fue la primera topología de red desarrollada, ya que dentro de la misma únicamente existen dos nodos. En una red punto a punto todos los dispositivos son considerados como socios iguales o pares entre sí. Una de las principales ventajas que presenta este tipo de topología es su fácil instalación y operación. Su principal desventaja radica en que a medida que las redes crecen las relaciones punto a punto son más difíciles de operar y coordinar, por lo tanto, la eficiencia de la red decrece a medida que la cantidad de dispositivos en una red aumenta. En el caso particular de los módulos de radio frecuencia Xbee uno de los nodos debe ser el coordinador y el otro puede ser un router o un dispositivo final (Dordoigne, 2015).

Los enlaces que interconectan a una red punto a punto se pueden clasificar en tres tipos según el sentido de comunicación que se desee. En la configuración simplex la transmisión de información se efectúa en un solo sentido. El tipo de comunicación half-duplex se caracteriza por transmitir la información en ambos sentidos, pero de forma intermitente, es decir que sólo uno de los dos nodos puede transmitir en un intervalo de tiempo dado, evitando que ambos dispositivos transmitan información al mismo tiempo (Halberg, 2007).

Full-duplex es un tipo de configuración en el cual se puede transmitir información en ambos sentidos de forma simultánea.

b. Topología tipo estrella. En la red tipo estrella las estaciones auxiliares como un router o un dispositivo final están conectados directamente a un nodo central denominado coordinador. La característica principal de esta topología es que toda la información debe pasar por el coordinador para transmitirla. Este tipo de configuración generalmente es implementado para redes locales. Las principales ventajas que posee dicha topología es que está preparada para prevención de daños o conflictos, permite la comunicación entre todos los nodos y el mantenimiento es económico. La principal desventaja que presenta esta configuración ocurre cuando el nodo central se desconecta o deja de funcionar apropiadamente (Halberg, 2007).

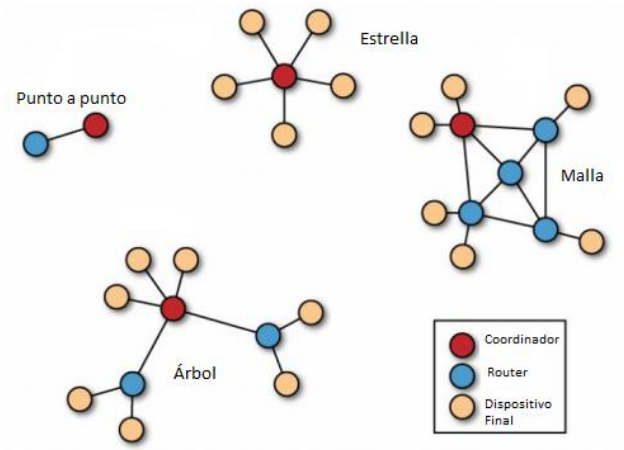
En la tipología tipo estrella para los módulos de comunicación de radio frecuencia Xbee el coordinador es el centro de la red, por lo tanto, es el elemento al cual se conectan los dispositivos finales. Toda la información es dirigida al coordinador y los dispositivos finales no pueden comunicarse entre sí (Halberg, 2007).

c. Topología tipo malla. En este tipo de topología de red cada nodo está conectado a todos los nodos de la red, de esta manera es posible intercambiar información de un nodo a otro por diversos caminos. Esta topología a diferencia de una configuración en estrella no requiere de un nodo central, con lo que se reduce el mantenimiento y se evita que la red se desconecte, por este motivo dicha configuración es más robusta. Otra de las características que agregan valor a este tipo de topología, ocurre cuando dentro de la red la conexión con un nodo falla, ya que el resto de los nodos evitan el paso por el nodo defectuoso (Halberg, 2007).

En el caso particular de los módulos de comunicación por radio frecuencia Xbee, la topología tipo malla es una de las configuraciones más complejas que se puede implementar, dicho tipo de red cuenta con nodos router y coordinador. La topología tipo malla no es jerárquica en el sentido de que cualquier dispositivo de la red puede interactuar con otro. La topología tipo árbol es una variación de una red tipo malla, en la cual los routers forman una columna vertebral con los dispositivos finales, que están agrupados a los routers.

En la Figura 16 se muestra una descripción gráfica de las diversas topologías de red implementables con los módulos de radiofrecuencia Xbee.

Figura 16. Topologías de red.

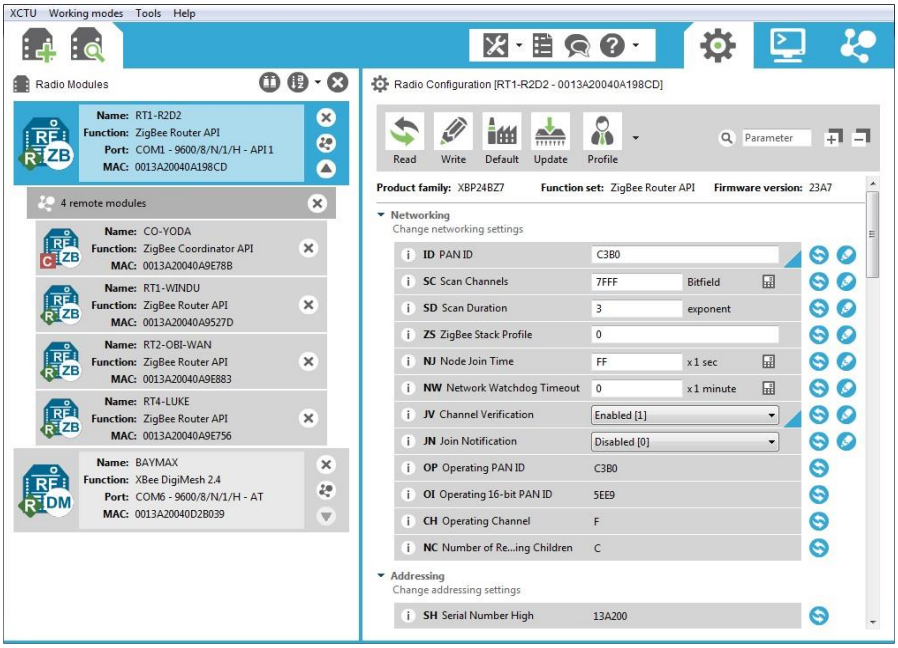


(Dordoigne, 2015)

4. Software XCTU. Es una herramienta multiplataforma gratuita diseñada para la comunicación entre módulos de radio frecuencia, dicho software pertenece a la compañía Digi. El programa posee una interfaz gráfica de usuario sencilla de utilizar, en la cual se puede establecer la configuración inicial de los módulos Xbee. La interfaz gráfica de usuario tiene a disposición diversas opciones dentro de las cuales se puede administrar y configurar dispositivos de radio frecuencia, realizar pruebas en consola para verificar la

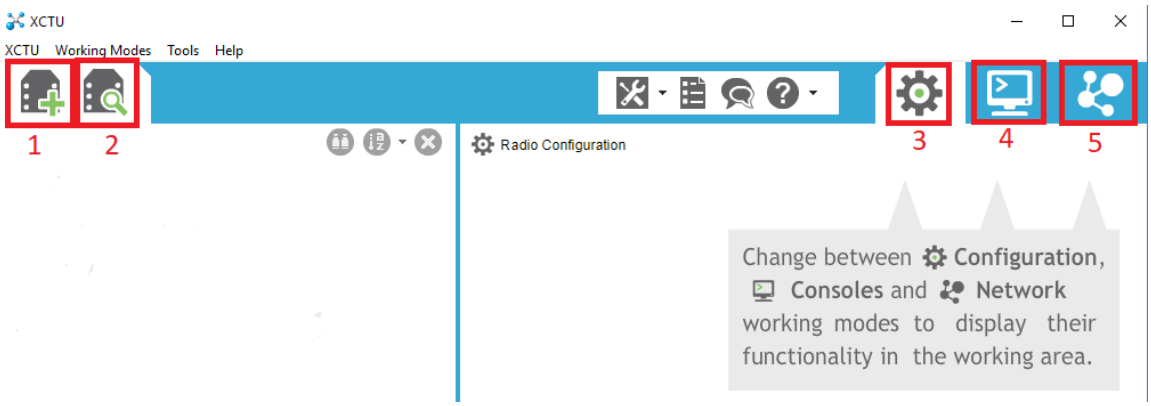
comunicación entre todos los dispositivos de una red, visualización grafica de una red, intensidad de las señales de conexión, entre otras, como se muestra en la Figura 17.

Figura 17. Interfaz gráfica de usuario del software XCTU.



El software XCTU posee 5 funciones principales, las cuales se describirán a continuación haciendo referencia a la Figura 18.

Figura 18. Opciones principales del software XCTU.



En la opción número 1 se pueden agregar nuevos dispositivos Xbee para realizar su configuración, en esta opción se debe seleccionar el puerto de comunicación serial al que se desea conectar. La opción número 2 realiza una búsqueda automática de todos los dispositivos que están conectados en ese momento, indicando

el número de puerto de comunicación serial designado, además de esto se puede seleccionar que dispositivo se desea agregar a una lista para poder realizar la configuración del mismo.

En la opción número 3 se encuentran todas las opciones de configuración que poseen los módulos de comunicación de radio frecuencia Xbee, en la cual se puede seleccionar el número de identificación de la red, seleccionar el tipo del dispositivo, por ejemplo, coordinador, router o un dispositivo final, además de esto se puede seleccionar el número de baudios, opciones de ahorro de energía, entre otros.

La opción número 4 muestra la comunicación serial entre dos o más dispositivos que estén configurados dentro de una misma red, acá se podrá enviar información en tiempo real y verificar si la red funciona como se esperaba. Por último, en la opción número 5 se muestra la topología implementada para la red de comunicación entre los dispositivos, en el cual se puede evaluar la intensidad de señales de conexión.

V. METODOLOGÍA

A. INVESTIGACIÓN

1. Identificar la necesidad. Se realizaron diversas visitas al Comité Olímpico Guatemalteco y a la Confederación Deportiva Autónoma de Guatemala para establecer los objetivos que se desean alcanzar y las necesidades que se requieren satisfacer con el desarrollo de los sensores inerciales.

Se realizaron visitas a diversas federaciones para determinar cuáles son las necesidades más recurrentes en diversos deportes como atletismo, taekwondo, levantamiento de pesas y tiro con armas de caza, con el motivo de implementar un análisis biomecánico.

2. Investigación preliminar. Previo al desarrollo e implementación de la interfaz gráfica de usuario se realizó una investigación acerca de la biomecánica, selección de software y comunicación inalámbrica.

a. Biomecánica. Se realizó un proceso de investigación previa acerca de biomecánica deportiva, para determinar cuáles son sus beneficios, desventajas y los recursos necesarios para realizar dichos análisis.

Se determinó que los análisis biomecánicos están presentes en la actualidad en diversos deportes a nivel mundial y son utilizados para optimizar el rendimiento de deportistas, prevenir lesiones e incluso ofrecer programas de rehabilitación luego de sufrir una lesión.

b. Implementación de software e interfaz gráfica. Proceso de investigación previa acerca del desarrollo, diseño e implementación de una interfaz gráfica de usuario. Se investigó acerca de los leguajes de programación en los cuales se puede desarrollar una interfaz gráfica de usuario intuitiva y de software libre.

c. Comunicación inalámbrica. Se realizó una investigación sobre tecnologías de comunicación inalámbricas para definir topologías de red existentes, topologías implementables para las diversas tecnologías, consumo de potencia y accesibilidad.

d. Especificaciones del trabajo de graduación. Se determinaron los requerimientos de la interfaz gráfica, así como el análisis y procesamiento de datos obtenidos por los sensores inerciales. La

interfaz gráfica debe ser capaz de presentar datos en tiempo real de la posición, velocidad y aceleración; a través de gráficas lineales en donde se muestre el cambio de las variables mencionadas.

Crear una red de comunicación de radio frecuencia entre los sensores y una estación de trabajo, para realizar el análisis y procesamientos de datos obtenidos por los sensores inerciales.

Generar un reporte y una base de datos sencilla que muestre al deportista valores máximos, mínimos y promedio de la posición, velocidad y aceleración, con el fin de mejorar el rendimiento de los atletas.

B. IDEACIÓN Y CREACIÓN

Se realizaron diferentes bocetos y lluvia de ideas que ayudaran a resolver el problema planteado de la mejor manera, cumpliendo con todos los requerimientos establecidos para el módulo de interfaz gráfica, análisis y procesamiento de información, base de datos y red de comunicación inalámbrica.

1. Implementación de software. Para la implementación de software se propusieron tres conceptos o leguajes de programación diferentes, entre los cuales se encuentran: Java, Python y C#.

2. Interfaz gráfica. Para el diseño de la interfaz gráfica de usuario se propusieron tres conceptos o diseños diferentes, los cuales se muestran a continuación:

Figura 19. Diseño de primer prototipo de la interfaz gráfica de usuario.

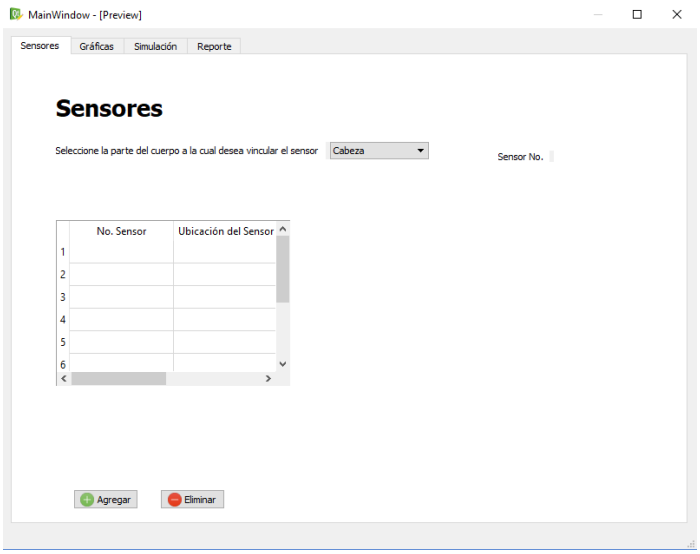


Figura 20. Diseño de segundo prototipo de la interfaz gráfica de usuario.

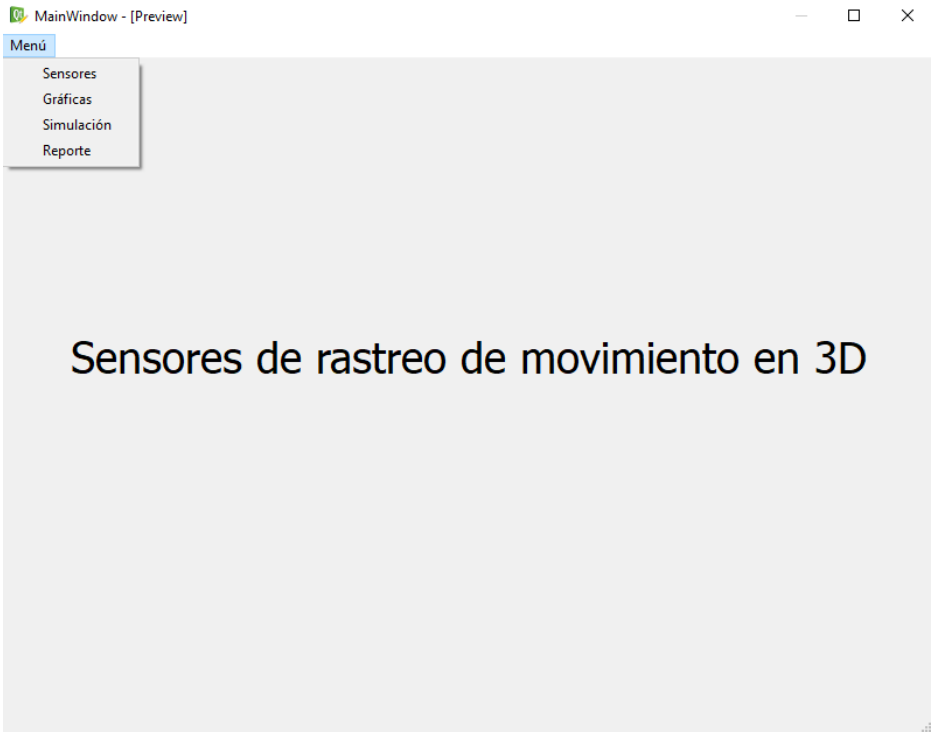


Figura 21. Diseño de tercer prototipo de la interfaz gráfica de usuario.

3. Comunicación inalámbrica. Para la implementación de la comunicación inalámbrica de los sensores inerciales se propusieron tres topologías de red diferentes, entre los cuales se encuentran: topología punto a punto, estrella y malla.

C. ANÁLISIS

1. Evaluación de conceptos. Se realizará un análisis técnico de las mejores ideas obtenidas en la fase de ideación y creación, con el fin de obtener los mejores diseños que sean potencialmente factibles para su desarrollo e implementación.

2. Selección de diseños. En la selección de diseños se evaluaron todos los conceptos relacionados con la implementación en software, interfaz gráfica de usuario y comunicación inalámbrica. Por cada tema mencionado anteriormente se realizó una tabla de decisión en las cuales se evaluaron tres conceptos diferentes por cada categoría, con el fin de seleccionar el concepto que más se adapte a las necesidades y requerimientos para la implementación de los sensores inerciales.

a. Implementación en software. Para determinar el software que será utilizado en el desarrollo e implementación de la interfaz gráfica de usuario se realizó el siguiente cuadro de decisión, en el cual cada característica evaluada en los tres lenguajes de programación propuestos obtuvo una ponderación entre 0 y 10; siendo 0 el valor mínimo y 10 el valor máximo. Las características evaluadas en el Cuadro 1 y su respectiva ponderación se eligieron en base a las necesidades y requerimientos planteadas para el desarrollo de la interfaz gráfica. Dicha interfaz debe ser multiplataforma, es decir que debe ser capaz de ejecutarse en cualquier sistema operativo. Además de no requerir ningún tipo de licencia pagada para la implementación y desarrollo de la misma.

Cuadro 1. Comparación de lenguajes de programación.

Lenguaje de Programación	Java	Python	C#
Multiplataforma	10	10	7
Orientado a objetos	10	10	10
Tipos de datos	10	10	9
Sintaxis	8	9	9
Lenguaje interpretado	7	6	5
Documentación	9	9	8
Librerías externas	9	10	8
Código abierto	9	10	9
Total	72	74	65

Con los resultados obtenidos mostrados en el Cuadro 1, se eligió a Python como el lenguaje de programación utilizado para la implementación y desarrollo de la interfaz gráfica de usuario. Python se caracteriza por ser un lenguaje multiplataforma, que posee una excelente implementación de interfaces y librerías externas, fácil sintaxis, programación interpretada y orientada a objetos.

b. Interfaz gráfica. Con las tres propuestas de prototipos mostradas en la fase de ideación y creación para la interfaz gráfica de usuario se realizó el siguiente cuadro de decisión, en el cual cada característica evaluada por los tres prototipos de diseño propuestos se obtuvo una ponderación entre 0 y 10; siendo 0 la más baja y 10 la más alta. La ponderación designada a cada característica evaluada mostrada en el Cuadro 2 se realizó en base a entrevistas realizadas a entrenadores, requerimientos de la interfaz gráfica y criterio propio del diseñador.

Cuadro 2. Comparación de diseños para la implementación de la interfaz gráfica.

Diseño	No. 1	No. 2	No. 3
Funcionalidad	10	10	8
Efectividad	10	10	10
Portabilidad	10	10	10
Uniformidad	8	8	8
Estética	8	7	6
Total	46	45	42

Con los resultados obtenidos en el Cuadro 2 se decidió realizar el diseño número uno, ya que es el diseño que posee una mejor estética y uniformidad, haciendo que la interfaz gráfica de usuario sea intuitiva y fácil de manejar para cualquier usuario.

c. Comunicación inalámbrica. En el siguiente cuadro de decisión cada característica evaluada por los tres tipos de tecnología diferente se ponderó entre 0 y 10; siendo 0 la ponderación mínima y 10 la ponderación máxima. La ponderación designada a cada característica evaluada mostrada en el Cuadro 3 se realizó con base en las necesidades y requerimientos de la red de comunicación inalámbrica y criterio propio del diseñador.

Cuadro 3. Comparación de tecnologías inalámbricas.

Tecnología	Wifi	Bluetooth	Xbee
Velocidad de transmisión	7	6	8
Número de nodos	8	5	10
Duración de batería	7	8	9
Consumo en transmisión	5	6	6
Consumo en reposo	7	8	10
Precio	6	7	8
Configuración	5	6	9
Accesibilidad	7	7	7
Aplicaciones	8	8	8
Total	60	61	75

En el Cuadro 3 se muestra que la mejor tecnología de comunicación inalámbrica para la implementación en los sensores inerciales es la tecnología Xbee, debido a que ofrece una buena accesibilidad, fácil configuración, consumo energético bajo y un gran número de nodos dentro de una red de comunicación inalámbrica.

D. DISEÑO

1. Diseño detallado. Se llevó a cabo el diseño detallado de la interfaz gráfica de usuario que muestre los resultados de interés en el análisis biomecánico de un deportista.

Se realizaron entrevistas y visitas a los entrenadores para obtener una retroalimentación de los requerimientos básicos que deben de tener los sensores inerciales, con el propósito de implementar una interfaz gráfica intuitiva para los usuarios.

Se realizó un diseño del protocolo de comunicación entre la red de sensores de comunicación inalámbrica para la obtención y procesamiento de los datos brindados por los sensores inerciales.

2. Proceso de verificación y validación. En dicho proceso se determinó si la solución desarrollada cumple con todos los requerimientos planteados de manera satisfactoria, realizando pruebas y obteniendo resultados durante un entrenamiento deportivo.

E. IMPLEMENTACIÓN

Programa ejecutable que contiene la interfaz gráfica de usuario y la base de datos donde se almacenan resultados puntuales de los datos recolectados por el sensor inercial.

Se implementó una red de comunicación por radio frecuencia para la comunicación y obtención de posición, velocidad y aceleración de los sensores inerciales.

VI. RESULTADOS

En la Figura 22 se muestra la temática de usuario correspondiente a la primera pestaña de la interfaz gráfica, en la cual el usuario debe ingresar la información requerida. El objetivo de esta pestaña se basa en personalizar los datos obtenidos por el sensor inercial. En la Figura 23 se muestra el flujo principal del programa correspondiente a la temática de usuario.

Figura 22. Pestaña usuario interfaz gráfica.

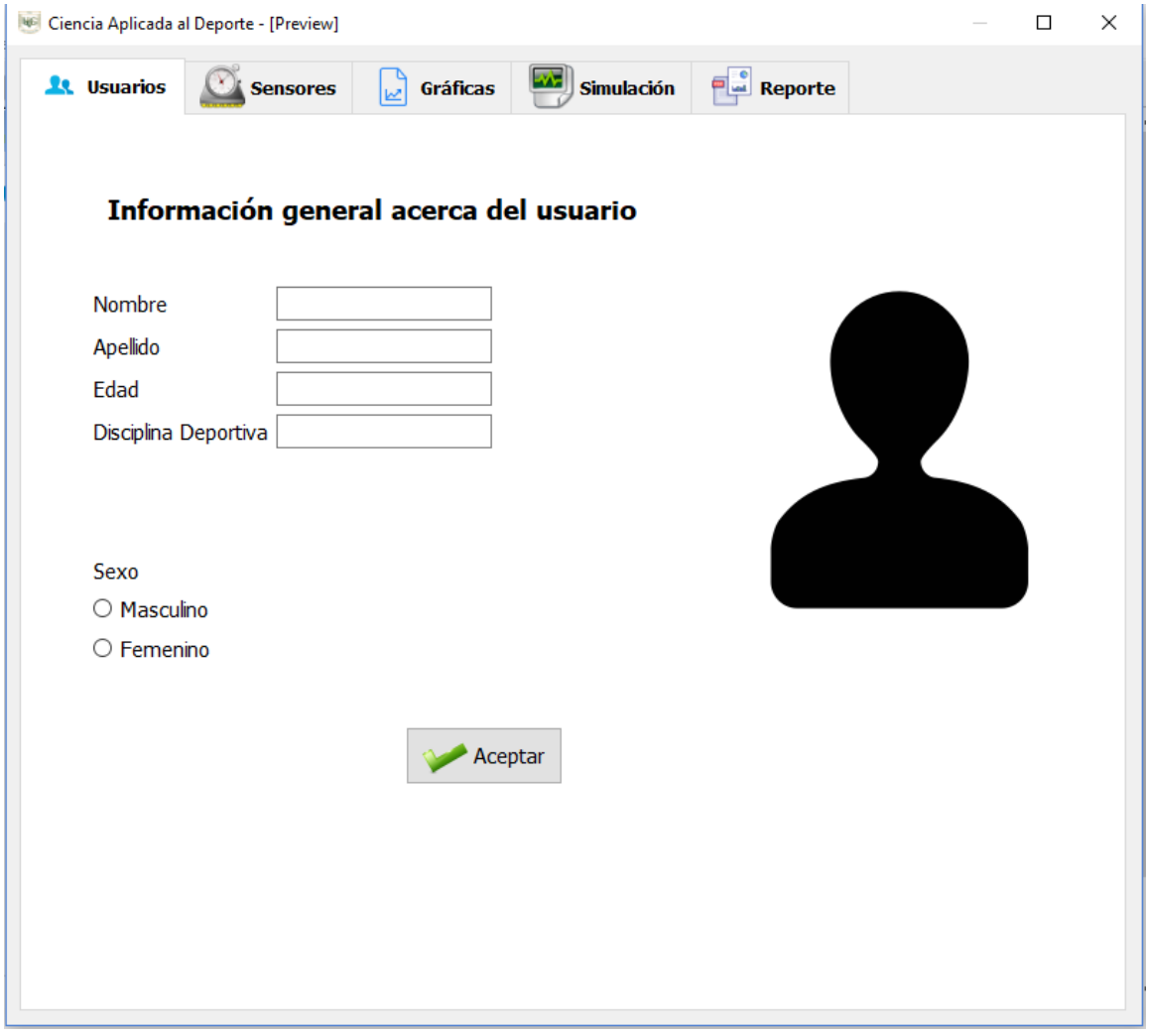
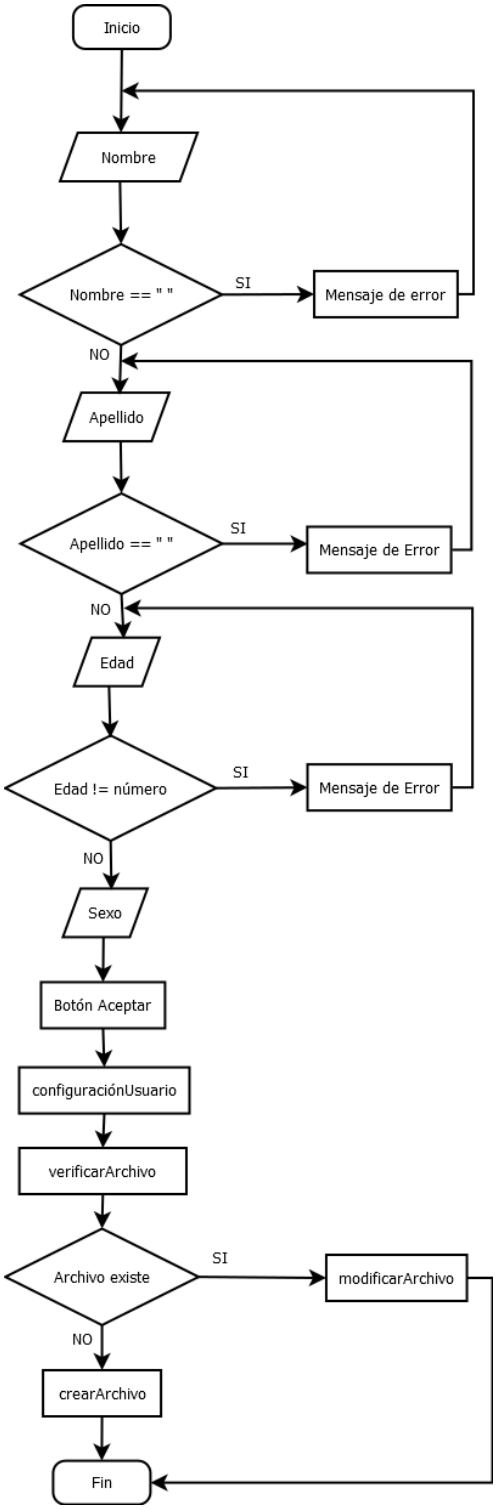


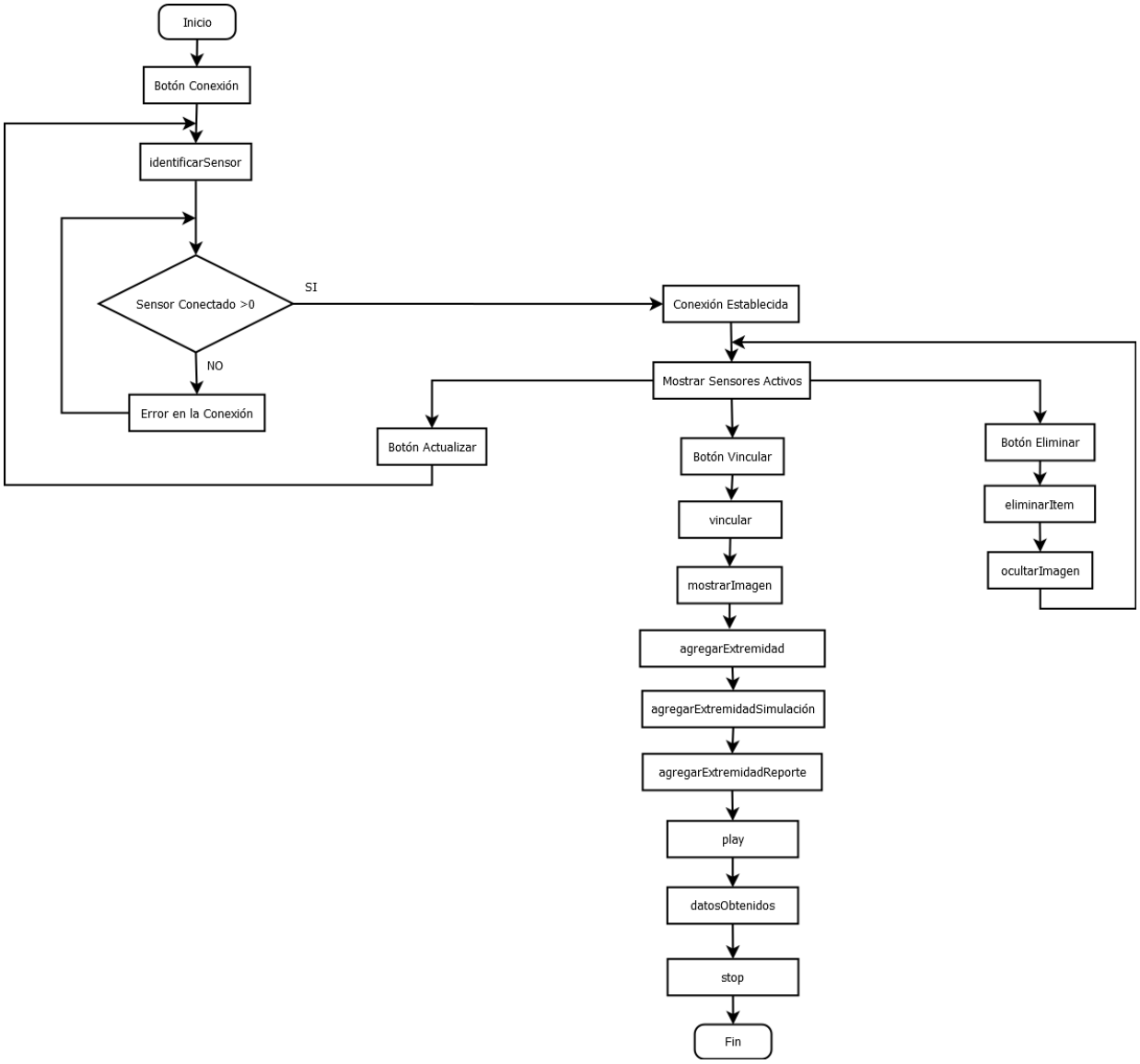
Figura 23. Diagrama de Flujo de la pestaña usuario de la interfaz gráfica.



La Figura 24 muestra la temática de sensores correspondiente a la segunda pestaña de la interfaz gráfica de usuario, esta pestaña es la que realiza la conexión de los sensores inerciales con la estación central, además de vincular y eliminar sensores a una extremidad del cuerpo. En la Figura 25 se muestra el diagrama de flujo que describe el código de programación desarrollado para su implementación.

Figura 24. Pestaña sensores interfaz gráfica.

Figura 25. Diagrama de flujo de la pestaña sensores de la interfaz gráfica.



La temática de gráficas corresponde a la tercera pestaña como se muestra en la Figura 26, en dicha temática se grafica en tiempo real la posición, velocidad y aceleración de la extremidad seleccionada, al seleccionar una extremidad se realiza la obtención de datos del sensor al que se encuentre vinculado como se muestra en la temática de sensores, correspondiente a la segunda pestaña. En la Figura 27 se puede observar el diagrama de flujo que describe el código de programación desarrollado para su implementación.

Figura 26. Pestaña gráficas interfaz gráfica.

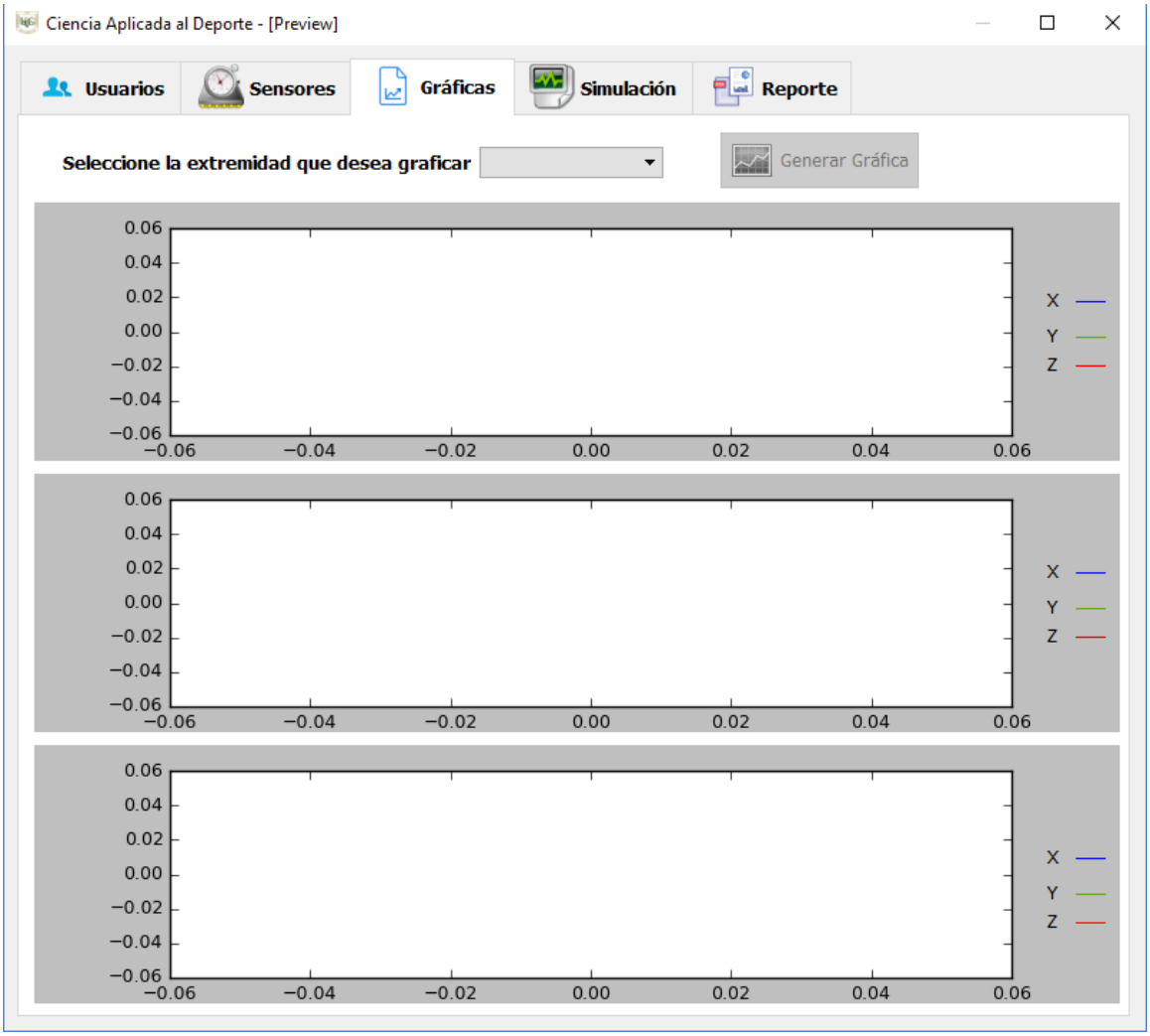
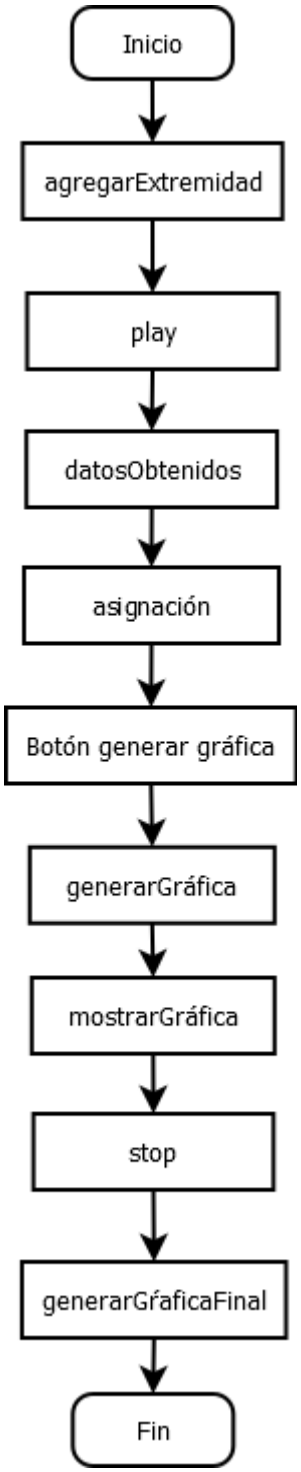


Figura 27. Diagrama de flujo de la pestaña gráficas de la interfaz gráfica.



La temática de reporte corresponde a la quinta pestaña como se muestra en la Figura 28, en dicha temática se sintetizan los datos más relevantes de la posición, velocidad y aceleración de la extremidad seleccionada, al seleccionar una extremidad se realiza la obtención de datos del sensor al que se encuentre vinculado como se muestra en la temática de sensores, correspondiente a la segunda pestaña. En la Figura 29 se puede observar el diagrama de flujo que describe el código de programación desarrollado para la implementación del reporte generado.

Figura 28. Pestaña reporte interfaz gráfica.

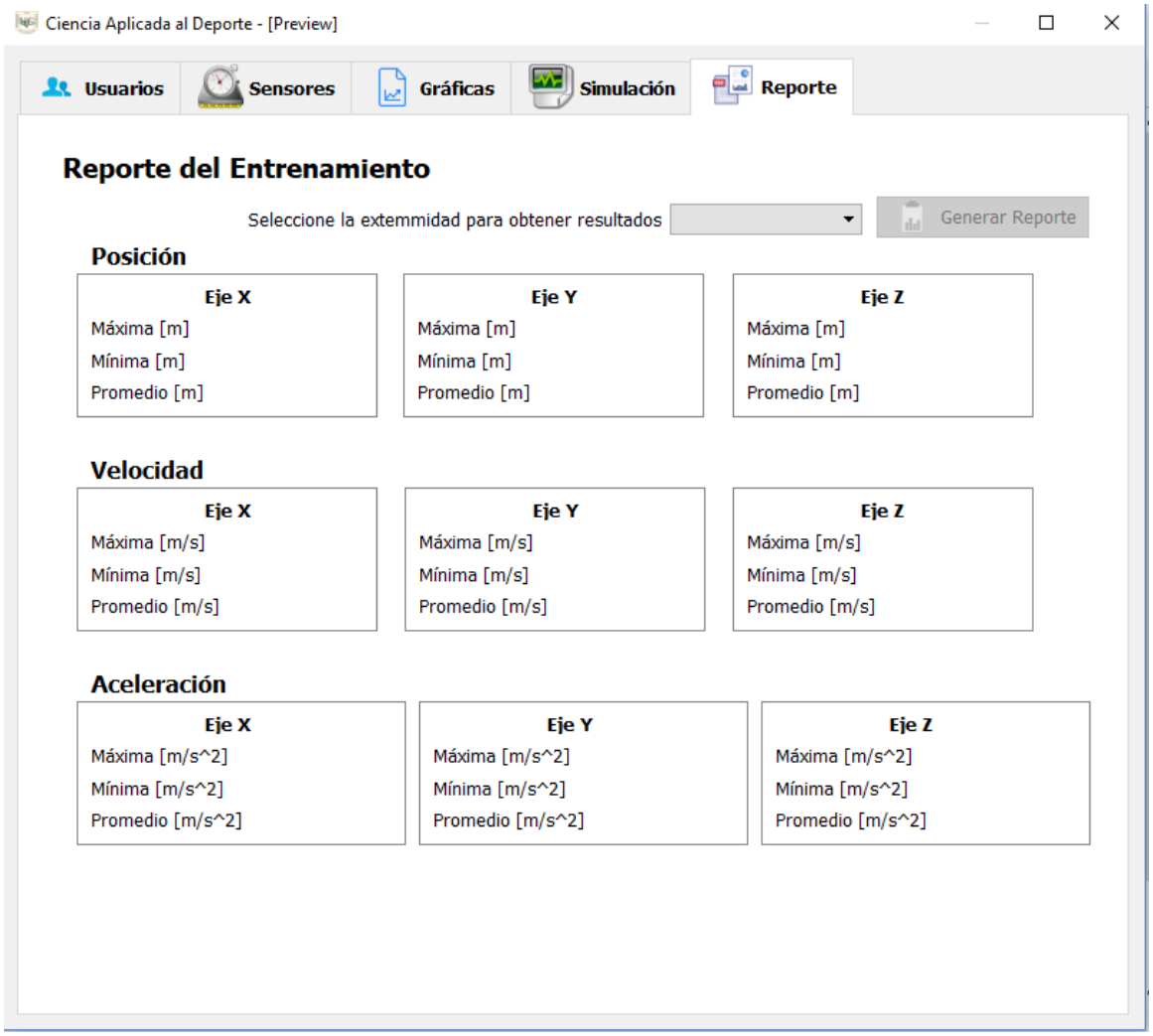
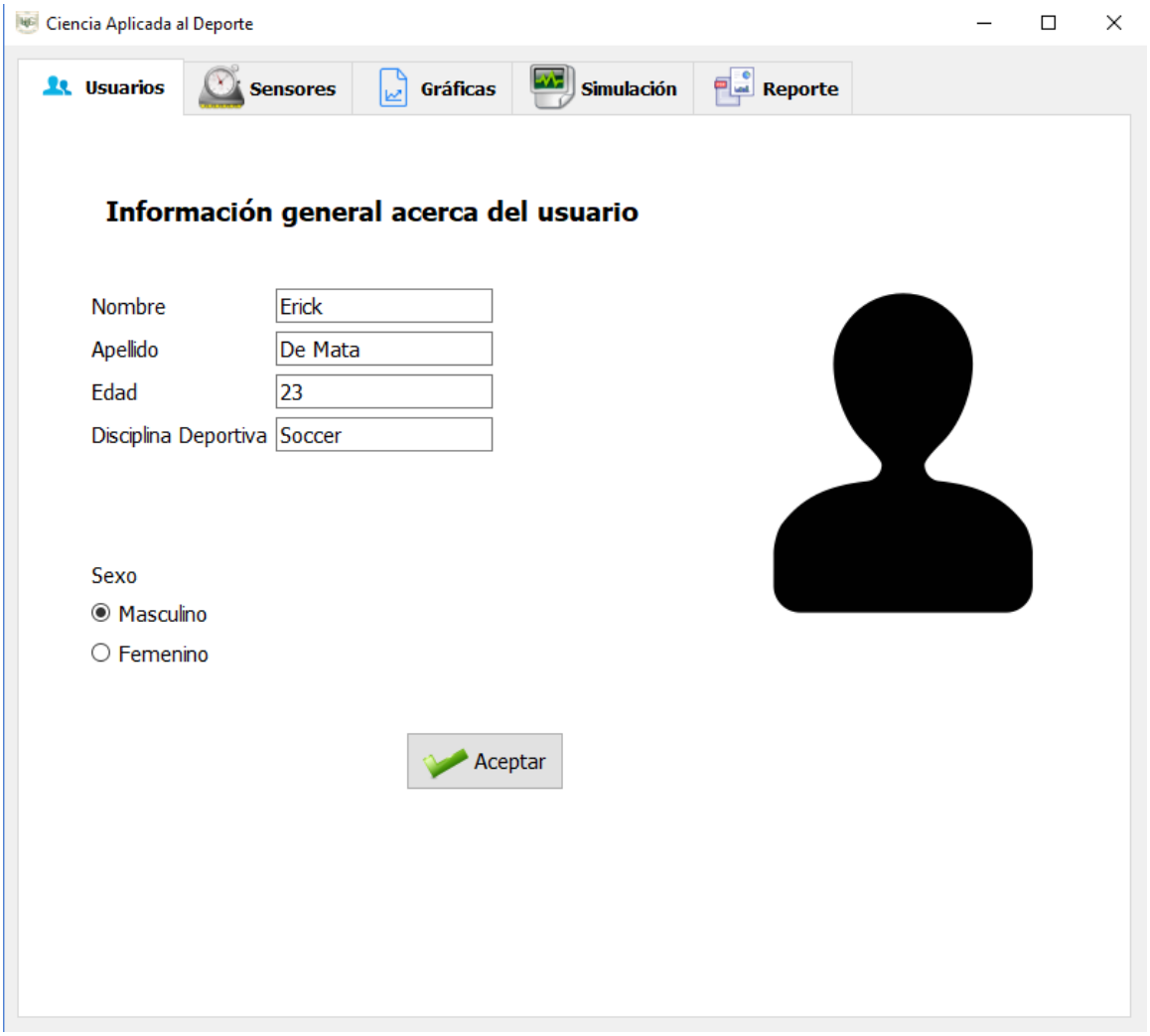


Figura 29. Diagrama de flujo de la pestaña reporte de la interfaz gráfica.

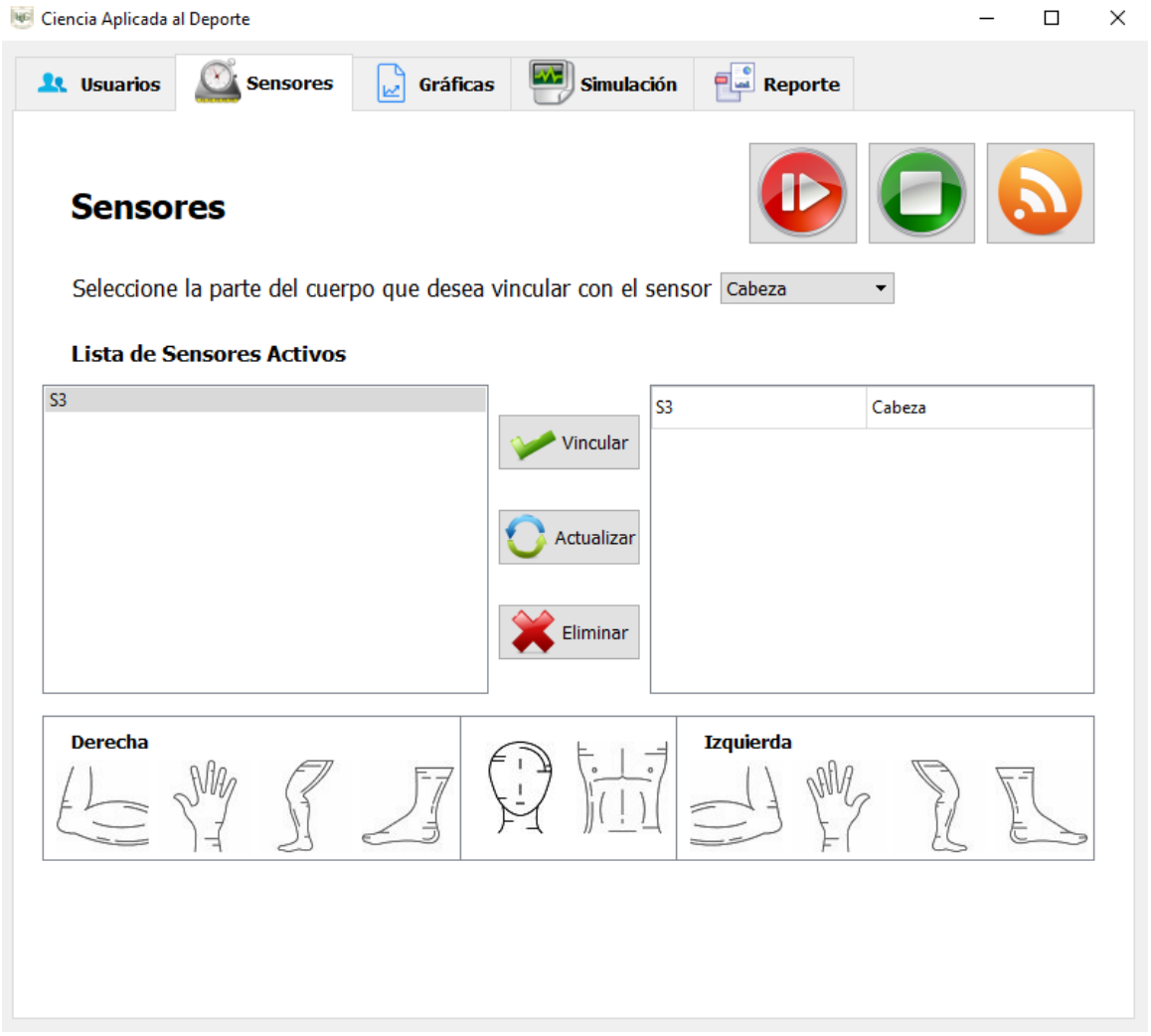
La Figura 30 muestra un ejemplo del funcionamiento de la temática usuarios, la cual correspondiente a la primera pestaña.

Figura 30. Información general del usuario.



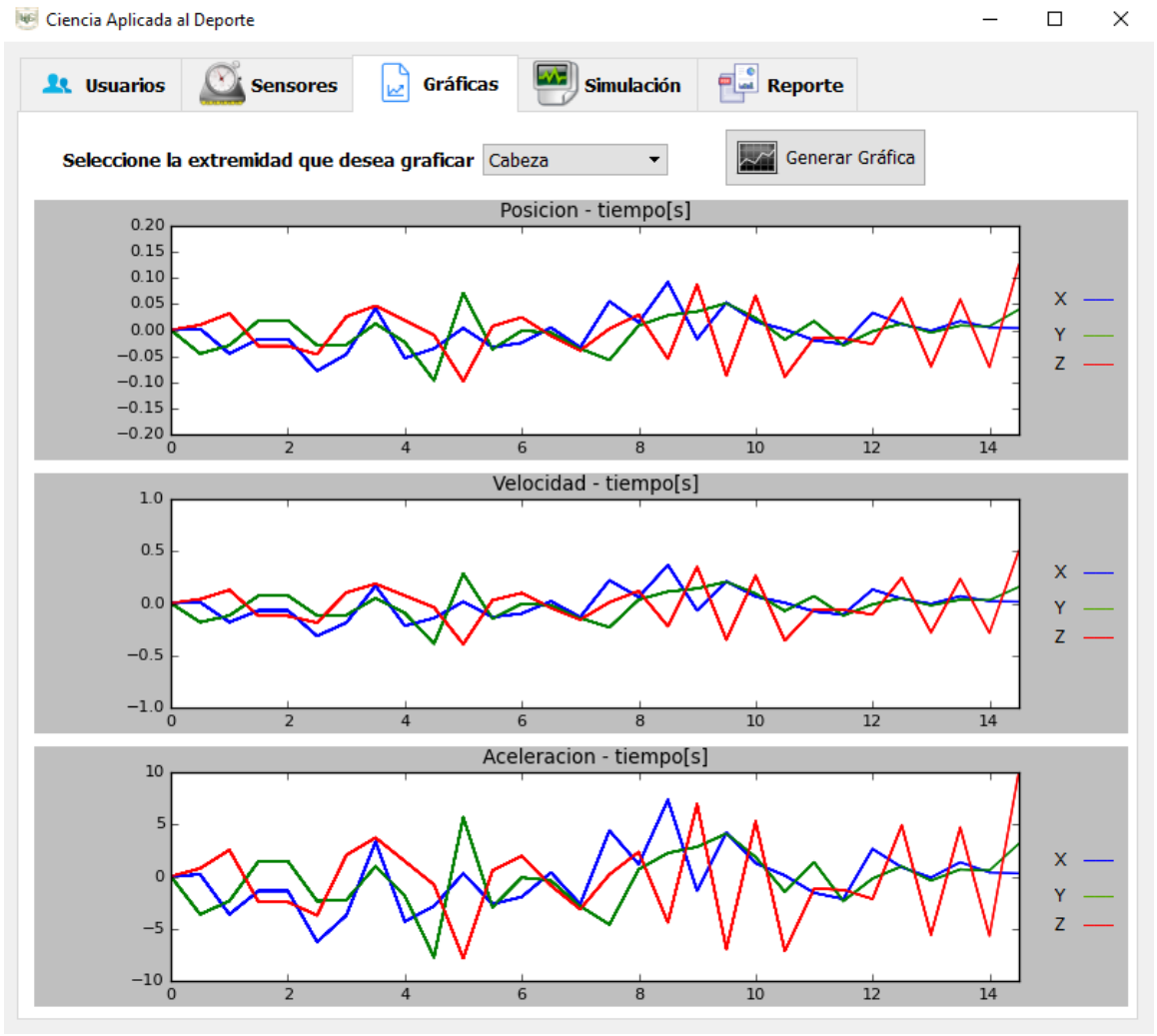
En la Figura 31 se observa una lista con los sensores conectados a la estación central de módulos de radiofrecuencia Xbee y la extremidad vinculada a dicho sensor. Al momento de vincular un sensor con una extremidad se habilitan las opciones de play y stop para iniciar y terminar la grabación de datos.

Figura 31. Lista de sensores activos y sensores vinculados a una extremidad.



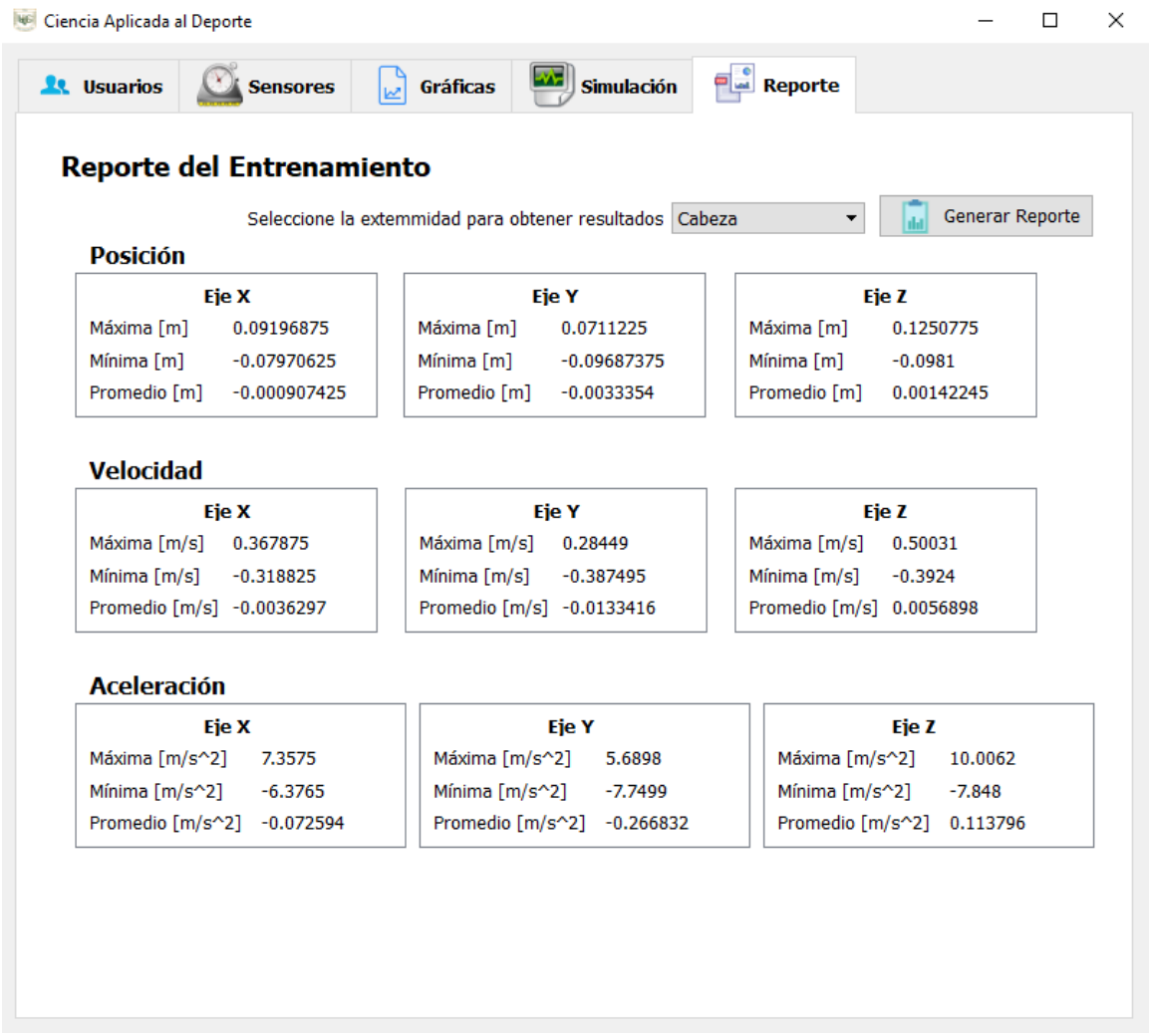
La Figura 32 muestra un ejemplo de las gráficas finales obtenidas por el sensor de movimiento vinculado a la cabeza, dichas gráficas representan la posición, velocidad y aceleración.

Figura 32. Gráficas finales de posición, velocidad y aceleración obtenidas por el sensor de rastreo de movimiento en 3D asignados a la cabeza.



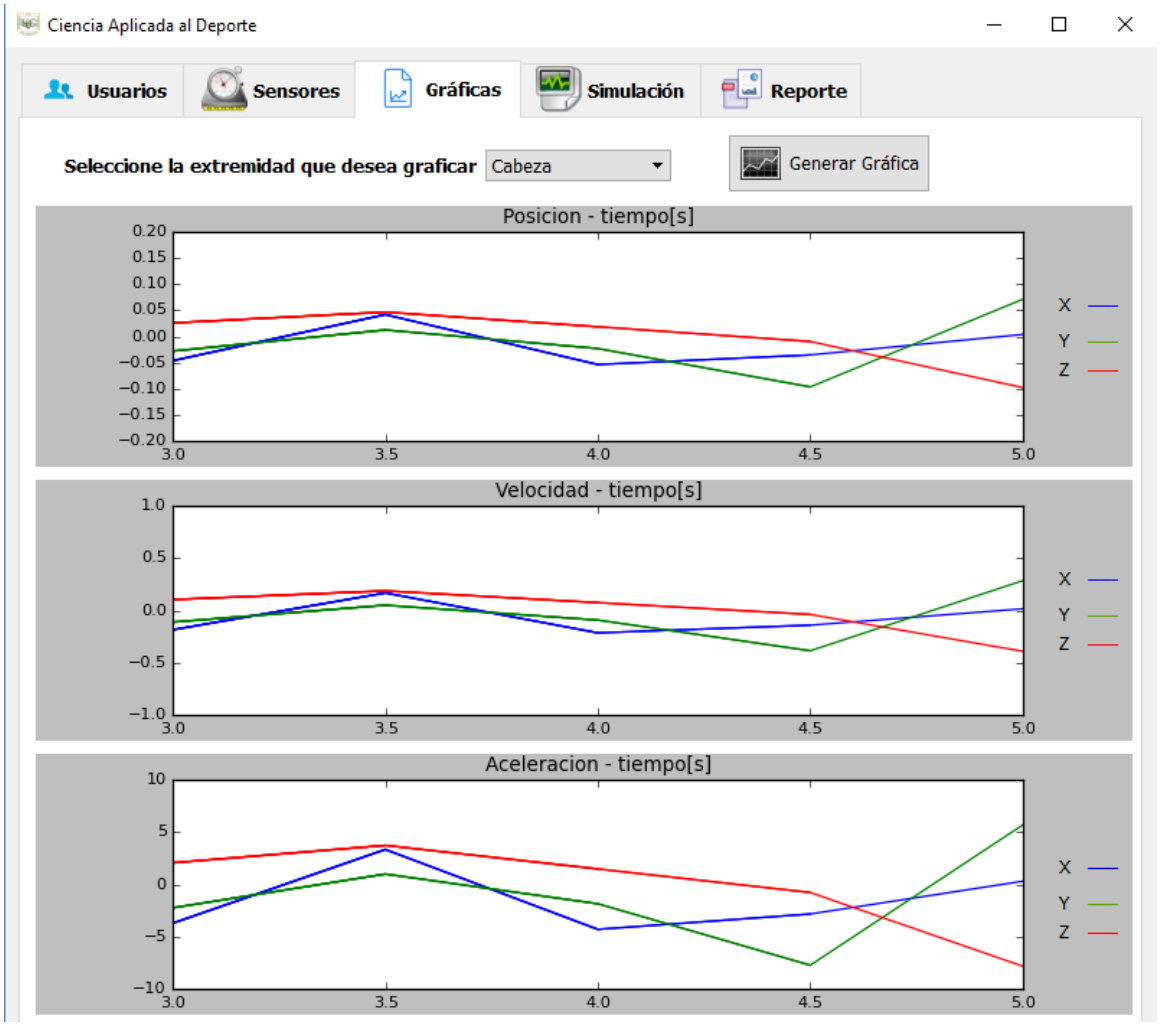
El reporte generado por el sensor vinculado a la cabeza se muestra en la Figura 33, en dicho reporte se muestra los máximos, mínimos y promedio de las posiciones, velocidades y aceleraciones correspondientes a los ejes y, x y z.

Figura 33. Reporte generado al finalizar la grabación de datos los obtenidos por el sensor de rastreo de movimiento de 3D asignado a la cabeza.



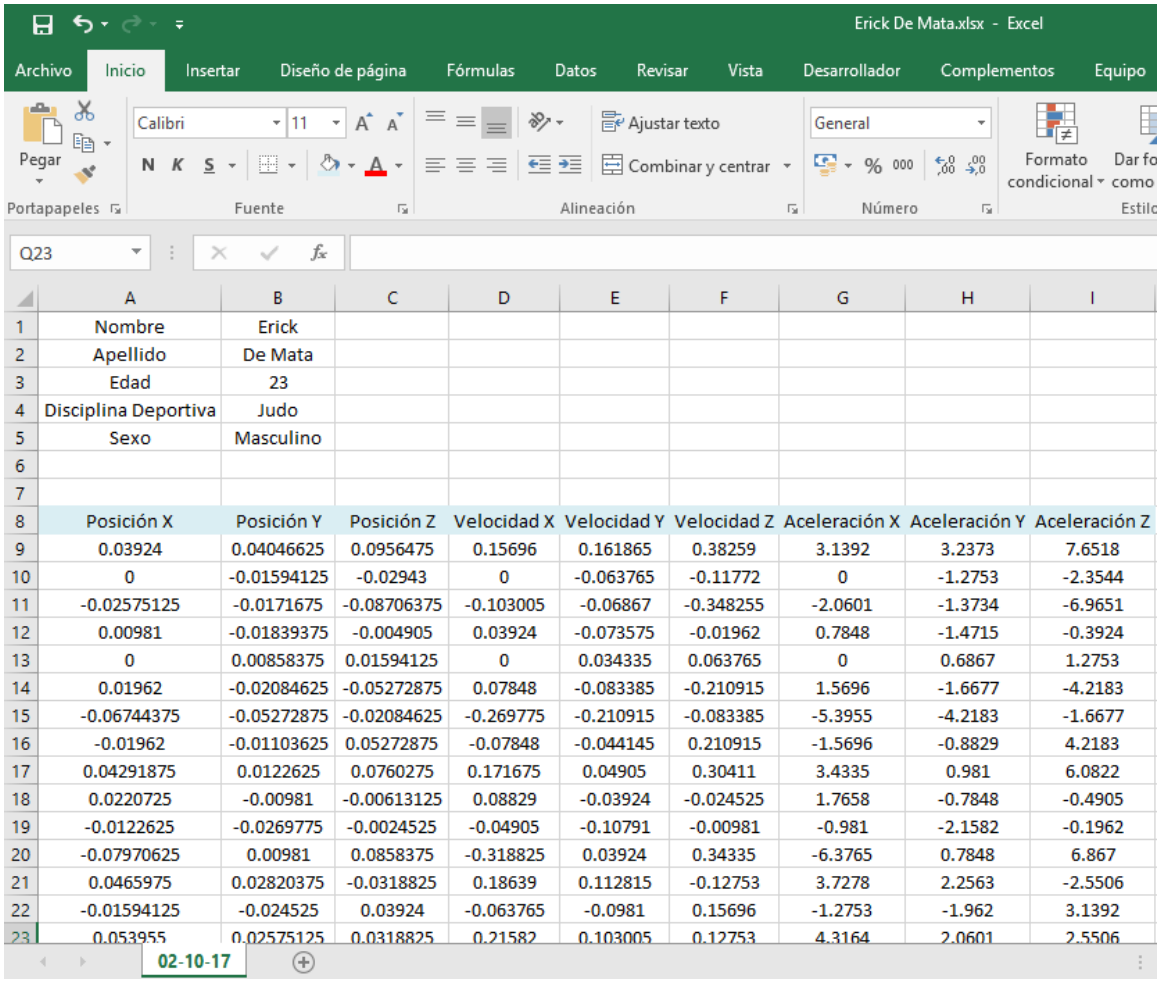
En la Figura 34 se puede observar un ejemplo de las gráficas en tiempo real obtenidas por el sensor de movimiento vinculado a la cabeza, dichas gráficas representan la posición, velocidad y aceleración.

Figura 34. Gráficas de posición, velocidad y aceleración en tiempo real.



Luego de haber finalizado la obtención y grabación de datos se genera un archivo de Excel con el nombre y apellido del usuario, una nueva hoja con la fecha en la que se realizó la prueba, un encabezado con sus datos personales y los datos obtenidos durante la prueba o sesión de entrenamiento.

Figura 35. Archivo de Excel generado con el nombre y apellido del usuario, en donde se genera una base de datos de las variables de medición obtenidas por el sensor.



	A	B	C	D	E	F	G	H	I
1	Nombre	Erick							
2	Apellido	De Mata							
3	Edad	23							
4	Disciplina Deportiva	Judo							
5	Sexo	Masculino							
6									
7									
8	Posición X	Posición Y	Posición Z	Velocidad X	Velocidad Y	Velocidad Z	Aceleración X	Aceleración Y	Aceleración Z
9	0.03924	0.04046625	0.0956475	0.15696	0.161865	0.38259	3.1392	3.2373	7.6518
10	0	-0.01594125	-0.02943	0	-0.063765	-0.11772	0	-1.2753	-2.3544
11	-0.02575125	-0.0171675	-0.08706375	-0.103005	-0.06867	-0.348255	-2.0601	-1.3734	-6.9651
12	0.00981	-0.01839375	-0.004905	0.03924	-0.073575	-0.01962	0.7848	-1.4715	-0.3924
13	0	0.00858375	0.01594125	0	0.034335	0.063765	0	0.6867	1.2753
14	0.01962	-0.02084625	-0.05272875	0.07848	-0.083385	-0.210915	1.5696	-1.6677	-4.2183
15	-0.06744375	-0.05272875	-0.02084625	-0.269775	-0.210915	-0.083385	-5.3955	-4.2183	-1.6677
16	-0.01962	-0.01103625	0.05272875	-0.07848	-0.044145	0.210915	-1.5696	-0.8829	4.2183
17	0.04291875	0.0122625	0.0760275	0.171675	0.04905	0.30411	3.4335	0.981	6.0822
18	0.0220725	-0.00981	-0.00613125	0.08829	-0.03924	-0.024525	1.7658	-0.7848	-0.4905
19	-0.0122625	-0.0269775	-0.0024525	-0.04905	-0.10791	-0.00981	-0.981	-2.1582	-0.1962
20	-0.07970625	0.00981	0.0858375	-0.318825	0.03924	0.34335	-6.3765	0.7848	6.867
21	0.0465975	0.02820375	-0.0318825	0.18639	0.112815	-0.12753	3.7278	2.2563	-2.5506
22	-0.01594125	-0.024525	0.03924	-0.063765	-0.0981	0.15696	-1.2753	-1.962	3.1392
23	0.053955	0.02575125	0.0318825	0.21582	0.103005	0.12753	4.3164	2.0601	2.5506

VII. ANÁLISIS DE RESULTADOS

En el desarrollo de la interfaz gráfica de usuario se implementó un método de diseño para elegir las opciones que cumplían mejor con los requerimientos y necesidades de los sensores inerciales. El motivo por el cual se eligió seguir un proceso de diseño fue para estructurar de una manera ordenada las ideas principales del proyecto y obtener el diseño óptimo que cumpliera con las necesidades de la interfaz gráfica, análisis y procesamiento de datos. Durante el proceso de diseño en la fase de ideación y creación se propusieron tres prototipos de interfaz gráfica como se muestra en la Figura 19, Figura 20 y Figura 21. Posteriormente se realizó una tabla de decisión como se muestra en el Cuadro 2, en la cual se eligió el prototipo número uno, correspondiente a la Figura 19, dicho prototipo cumple con tener la mejor funcionalidad y efectividad de las tres opciones, lo cual garantiza que la interfaz gráfica de usuario permite realizar más de una tarea al mismo tiempo, además de permitir al usuario encontrar entre todas las opciones las tareas que desea realizar, sintiendo conformidad al utilizar la interfaz. Otro aspecto sumamente importante que ayudó a determinar la selección del prototipo número uno fue la estética, ya que este prototipo muestra un diseño visualmente agradable al usuario. Esto se consigue eligiendo un buen tamaño, color y tipo de fuente que ayuden a la legibilidad e interpretación adecuada de la información presentada al usuario final, ayudando a que la interfaz sea intuitiva, fácil de utilizar para cualquier persona, cumpliendo con una excelente relación entre humano y computadora.

Posteriormente de haber elegido el diseño de prototipo que cumple con las necesidades del presente trabajo de graduación, se eligió el software para la implementación de la interfaz gráfica. Para ello se realizó una tabla de decisión como se muestra en el Cuadro 1, con los resultados obtenidos se determinó que el software Python fue el indicado. Esta decisión se basó principalmente en que posee un lenguaje de programación interpretado, orientado a objetos y su facilidad para implementar librerías externas, como es el caso de Qt. Qt es una interfaz de código libre sumamente popular utilizada para el desarrollo de interfaces gráficas de usuario.

Como se muestra en la Figura 22, la interfaz gráfica se dividió en cinco temáticas diferentes; las cuales son: usuario, sensores, gráficas, simulación y reporte. En la temática de usuario se solicita al usuario información general del mismo, esto con el fin de crear una base de datos que posea toda la información generada a lo largo de un periodo de tiempo, para poder evaluar el progreso y evolución de los deportistas durante las sesiones de entrenamiento. Para la implementación de la base de datos se utilizó el software Excel, ya que es una herramienta intuitiva, fácil de utilizar y muestra, de manera ordenada, los datos obtenidos por el usuario. En la Figura 35 se observa el archivo de Excel creado con el nombre y apellido que el usuario ingresó en la temática de usuario como se muestra en la Figura 30. Dentro de este archivo se crea un

encabezado con los datos generales del usuario y una hoja nueva nombrada con la fecha en la cual se realizó la sesión de entrenamiento para la obtención de datos generados por el sensor inercial.

La Figura 24 muestra la temática de sensores desarrollada en la interfaz gráfica de usuario, la cual es el centro de toda la interfaz gráfica. La importancia de esta pestaña radica en el reconocimiento de sensores activos, es decir todos aquellos sensores que se están comunicando de forma remota con la estación base de sensores de radio frecuencia. Al existir un sensor activo dentro de la red de sensores se puede vincular una extremidad del cuerpo al sensor, al ejecutar esta acción se habilita una figura correspondiente a la extremidad, además de mostrar un cuadro con el identificador del sensor y su extremidad vinculada. Esto ayudó a que el usuario identifique rápidamente qué sensores se encuentran activos y qué extremidades del cuerpo están seleccionadas, ya que de esta forma se evitó generar confusión en el usuario y brindar buena efectividad en la interfaz gráfica. Para la identificación de los sensores se desarrolló un protocolo de comunicación que consta de cuatro parámetros iniciales, en la cual el primer parámetro corresponde al indicador del sensor y los tres parámetros restantes a las aceleraciones en los ejes y, x y z. En el primero de los cuatro parámetros, el primer carácter corresponde a una letra “S” mayúscula; la cual hace referencia a que la información obtenida proviene de un sensor. El segundo carácter corresponde al número de sensor que envía la información obtenida; con esto se asegura que no exista ningún tipo de confusión o pérdida de datos en el instante en el cual más de un sensor establezca comunicación con la estación base del módulo de comunicación inalámbrica por radio frecuencia.

Para el módulo de comunicación inalámbrica se implementó una tipología de red tipo estrella debido a que es la topología que mejor se adaptaba a los requerimientos y necesidades del módulo, ya que en la red de comunicación inalámbrica implementada por los sensores únicamente se requiere un enlace de comunicación entre los sensores y una estación base. Por dicho motivo es irrelevante la comunicación entre sensores y se descartan las topologías tipo malla o tipo árbol. El análisis y procesamiento de señales se realiza en la estación base del módulo de comunicación inalámbrica, es aquí donde radica la mayor desventaja de la topología de red seleccionada ya que si la estación base falla toda la red de sensores es inútil. Sin embargo, las ventajas que presenta la topología tipo estrella son mayores que sus desventajas, debido a que dicha topología está preparada para daños o conflictos generados en los dispositivos finales, es decir los sensores. El mantenimiento de la red es económico comparado con las topologías tipo malla o árbol, ya que si un sensor o dispositivo no funciona correctamente es necesario cambiar únicamente el sensor que funciona de forma anómala.

En la Figura 34 se muestran las gráficas en tiempo real de la posición, velocidad y aceleración del sensor vinculado a la cabeza, mostrar las gráficas en tiempo real ayuda a los entrenadores a proporcionar una retroalimentación instantánea del desempeño de los atletas, lo cual ayudará a mejorar su técnica de manera progresiva hasta alcanzar un nivel óptimo. Dentro de la interfaz gráfica se implementó un proceso multi-hilo,

que consiste en generar en paralelo más de una acción a la vez por parte de la interfaz gráfica, esto contribuye a que el flujo del programa no se vea afectado por realizar más de una tarea al mismo tiempo y las gráficas muestren un desfase y se actualicen de forma incorrecta brindando al entrenador y atleta información errónea. Con este tipo de procesos implementados se garantiza que la interfaz gráfica sea funcional, ya que pueden realizar más de una tarea al mismo tiempo.

Al finalizar con la grabación de datos en la interfaz gráfica se actualizan las gráficas de posición, velocidad y aceleración como se muestra en la Figura 32, plasmando todos los datos obtenidos durante la grabación. Esto ayudará tanto a entrenadores como atletas a identificar patrones de comportamiento, tiempos en los cuales existe un cambio de ritmo o rendimiento del atleta, entre otras cosas.

Finalmente, como se puede observar en la Figura 33 se genera un reporte final en los cuales se presenta información puntual de las variables posición, velocidad y aceleración. En dicho reporte se presenta el valor máximo, mínimo y promedio de las variables de medición correspondientes a los ejes y, x y z. Esta información será evaluada de forma inmediata por el entrenador para verificar si el atleta cumplió con sus objetivos. La forma en la cual se presentan los resultados es sumamente importante, ya que, si existe orden, tamaño de letra adecuado y es legible la comunicación de la información generada será transmitida de forma efectiva.

VIII. CONCLUSIONES

1. Con el desarrollo e implementación de la interfaz gráfica es posible graficar en tiempo real las variables de medición correspondientes a posición, velocidad y aceleración, mas no generar una representación gráfica de una extremidad que represente el movimiento en 3D.
2. Se implementó correctamente la red de comunicación inalámbrica entre los módulos de comunicación por radio frecuencia, logrando un alcance máximo de comunicación de cien metros lineales en un ambiente abierto.
3. Se logró establecer la comunicación inalámbrica de los sensores con una topología tipo estrella, mediante la cual se obtuvieron datos de posición, velocidad y aceleración lineal en simultáneo.
4. La interfaz gráfica proporciona al usuario un reporte sobre todos los datos obtenidos por el sensor mostrando los valores máximos, mínimos y promedio de las variables de medición, las cuales corresponden a la posición, velocidad y aceleración.
5. Se recomienda poseer un sensor comercial de rastreo de movimiento inalámbrico en tiempo real, para establecer una comparación entre el sensor inercial desarrollado, estableciendo porcentajes de error y exactitud en las mediciones. Los sensores recomendados son MTw Awinda de la marca Xsens, debido a que son sensores comprobados por atletas profesionales de alto rendimiento.
6. La base de datos fue implementada en un software accesible para todo público, que no requiere de conocimientos técnicos avanzados de tecnología, debido a que fue un requerimiento establecido con el Comité Olímpico Guatemalteco para que facilitara que cualquier entrenador o atleta pudiera realizar una retroalimentación adecuada de una sesión de entrenamiento. Esto se hizo para evitar el aumento del costo del sensor implementando un software de acceso libre o comercial.

IX. BIBLIOGRAFÍA

1. Albornoz, Claudia. 2014. <<Diseño de Interfaz Gráfica de Usuario>>. *Workshop de Investigadores en Ciencias de la Computación* [Argentina]. 16(1): 540-544.
2. Albornoz, Claudia; E. Miranda; M. Berón. 2014. <<Evaluación de Interfaces Gráficas de Usuario usando LSP>>. *Workshop de Investigadores en Ciencias de la Computación* [Argentina]. 16(3): 200-212.
3. Blanchette, Jasmin; M. Summerfield. 2008. *C++ GUI programing with Qt 4*. 2ª ed. New York: Prentice Hall. 752 págs.
4. Coutinho, Nilo. 2016. *Introducción a la programación con Python, algoritmos y lógica de programación para principiantes*. Sao Pablo: Novatec Editora Ltda. 340 págs.
5. Daza, Javier. 2007. *Evaluación clínico-funcional del movimiento corporal humano*. Bogotá: Editorial Médica Panamericana, S.A. 372 págs.
6. Dordoigne, José. 2015. *Redes infromáticas*. 5ª ed. Madrid: Ediciones ENI. 602 págs.
7. Girod, Antón. 2012. *Desarrollo e implementación de una red de sensores Zigbee mediante el dispositivo Xbee de Digi*. Cataluña: Universitat Rovira i Virgili. 103 págs.
8. Gonzáles, Raúl. 2007. *Python para todos*. España: Creative Commons. 160 págs.
9. Halberg, Bruce. 2007. *Fundamentos de redes*. 4ª ed. México DF.; McGraw Hill. 444 págs.
10. Izquierdo, Mikel. 2008. *Biomecánica y bases neuromusculares de la actividad física y el deporte*. Madrid: Editorial Médica Panamericana, S.A. 769 págs.
11. Luna, Lizbeth. 2004. <<El diseño de la interfaz gráfica de usuario para publicaciones digitales>>. *Revista Digital Universitaria* [México]. 5(7): 2-12.
12. MCI Electronics. *¿Qué es Xbee?* http://xbee.cl/que-es-xbee/ [septiembre del 2017].
13. Qt Company. *Acerca de Qt*. https://wiki.qt.io/About_Qt/es [septiembre del 2017].
14. Qt Company. *Desarrollo de software más inteligente*. https://www1.qt.io/es/ [septiembre del 2017].
15. Ruiz, José Antonio, *et al*. 2012. *Análisis del movimiento en el deporte*. Madrid: Editorial Wanceulen, S.L. 491 págs.
16. Suárez, Gustavo Ramón. 2009. *Biomecánica deportiva y control del entrenamiento*. Medellín: Funámbulos Editores. 134 págs.

17. Vásquez, Augusto; C. Yánez y L. Pro Concepción. 2011. <<Técnicas de análisis para el diseño de interface gráfica de usuario>>. *Revista de investigación de sistemas e informática*. VII (1): 53-64.
18. Zigbee Aliance. *Especificaciones de Red*. http://www.zigbee.org/ [septiembre del 2017].

X. ANEXOS

A. DIAGRAMAS DE FLUJO

Figura 36. Subrutina configuración usuario.

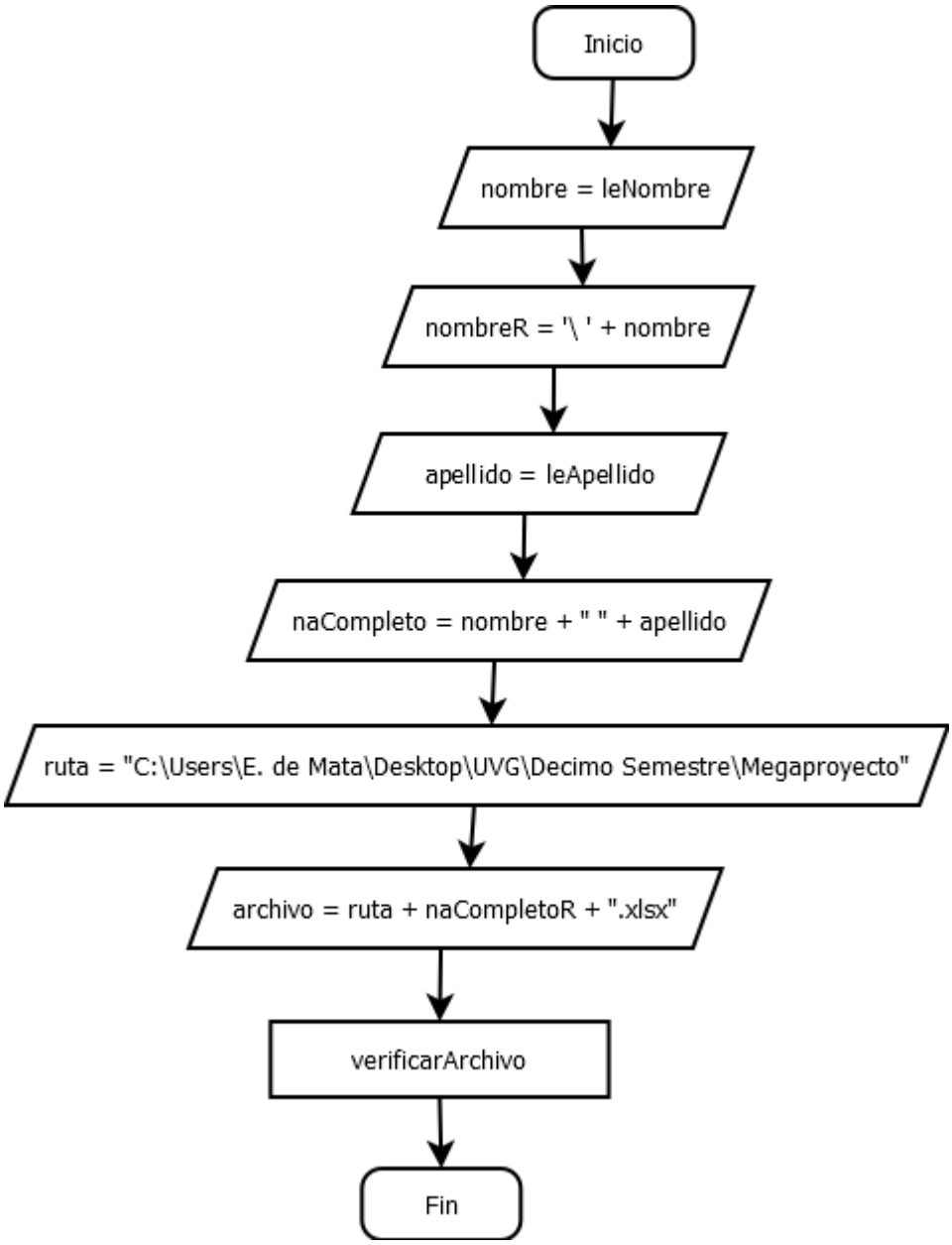


Figura 37. Subrutina crear archivo.

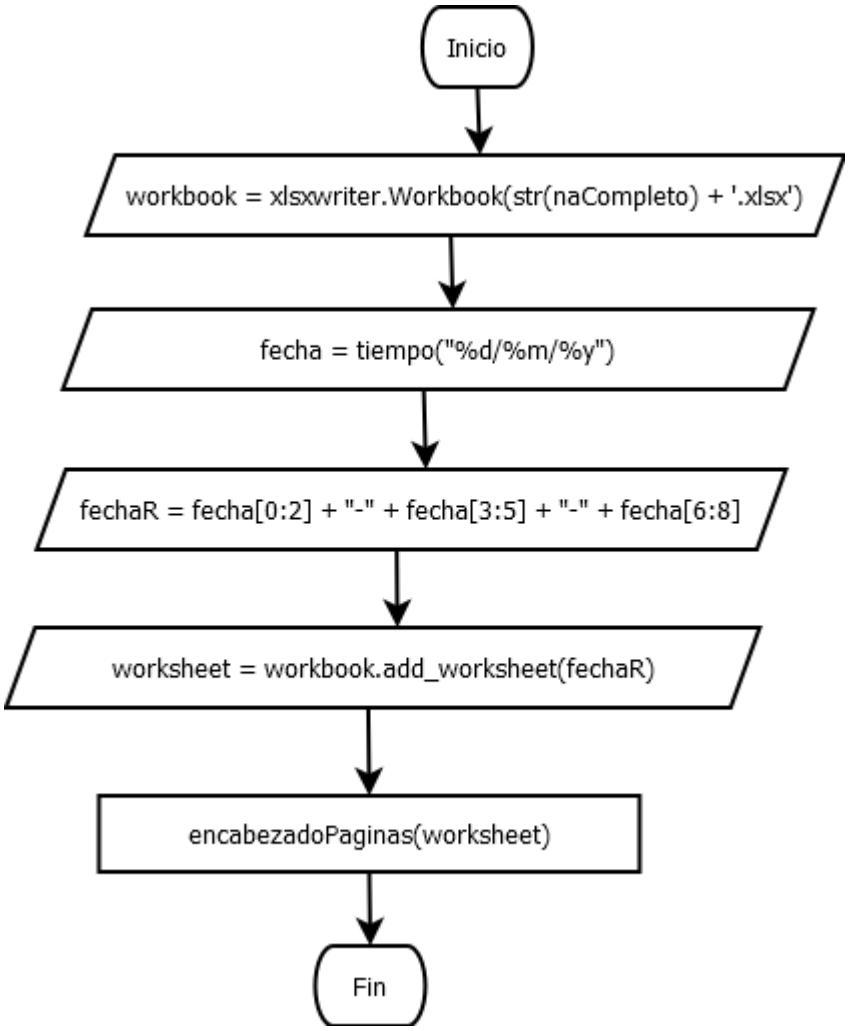


Figura 38. Subrutina encabezado páginas.

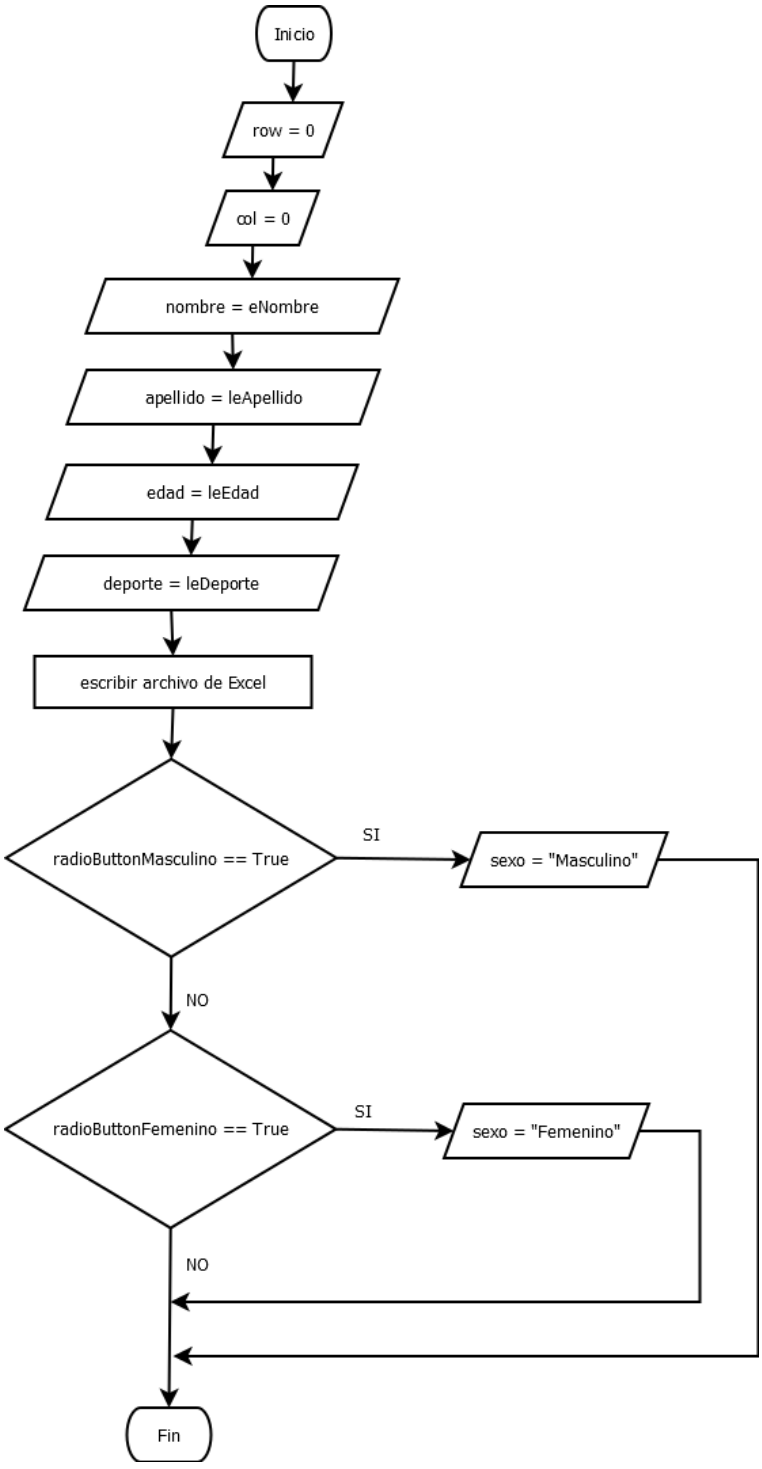


Figura 39. Subrutina modificar archivo.

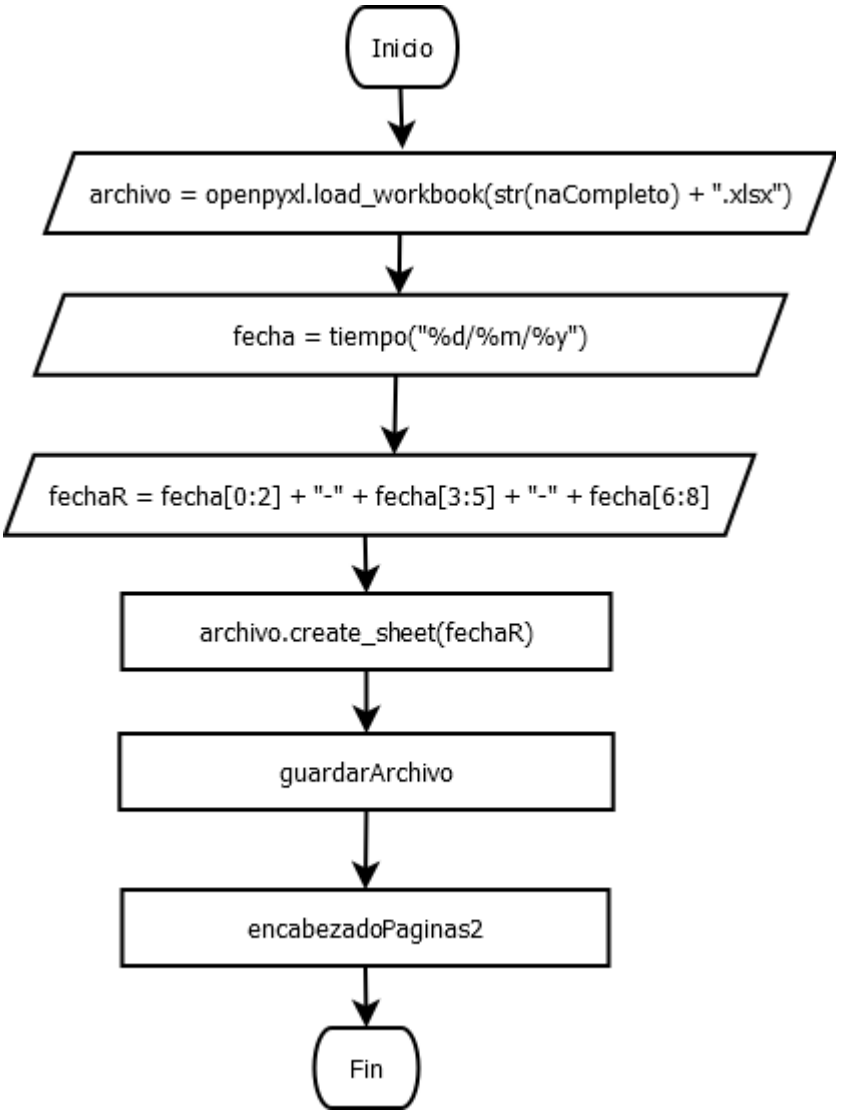


Figura 40. Subrutina asignación.

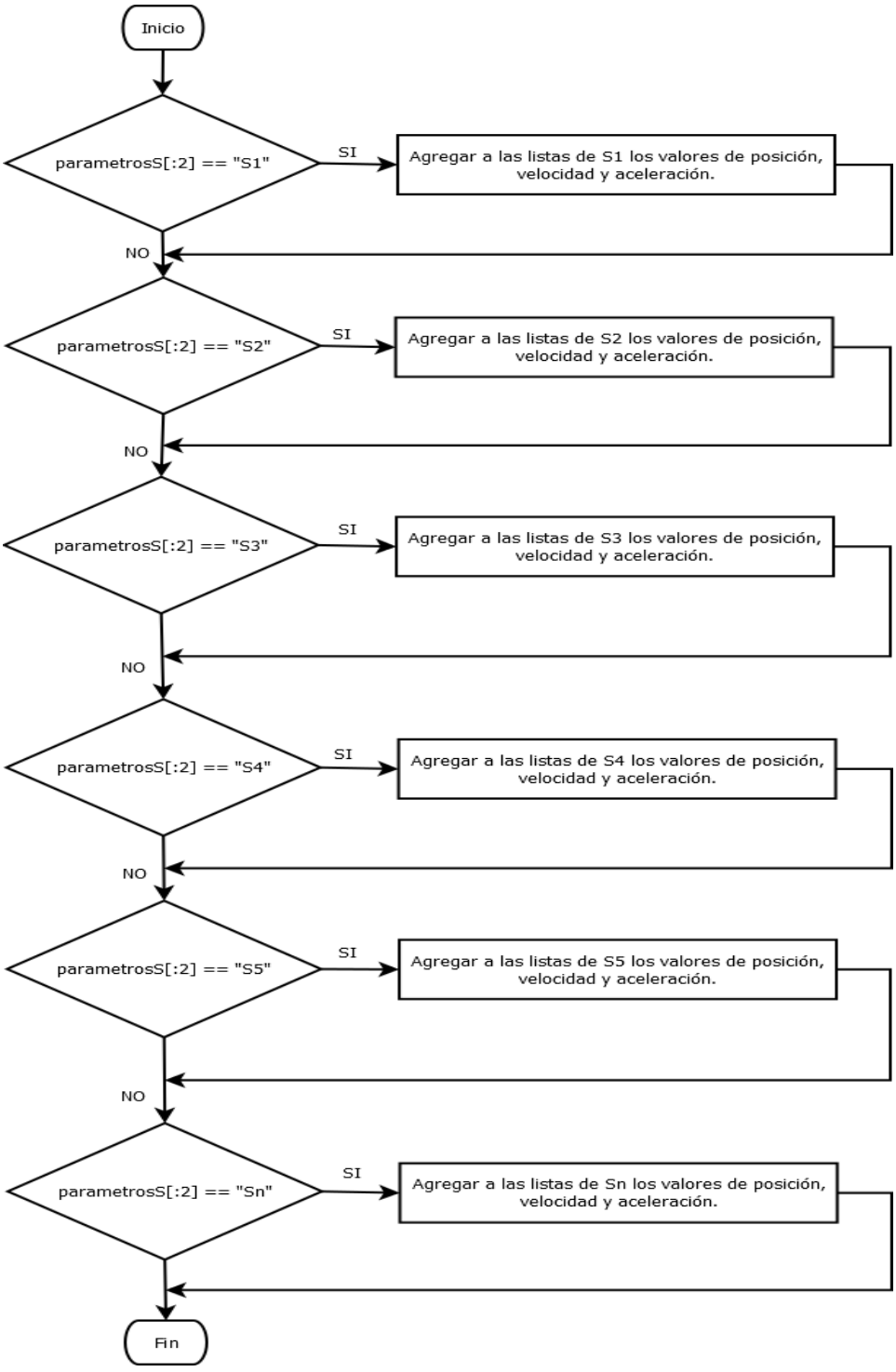


Figura 41. Subrutina datos obtenidos.

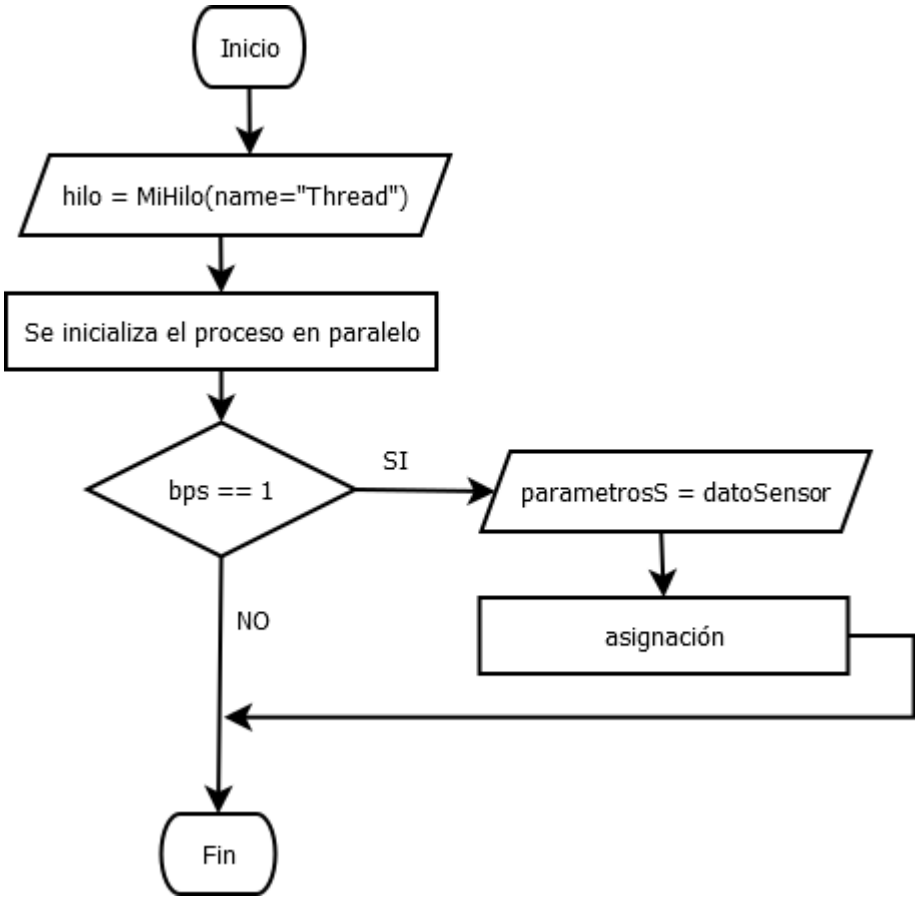


Figura 42. Subrutina guardar datos.

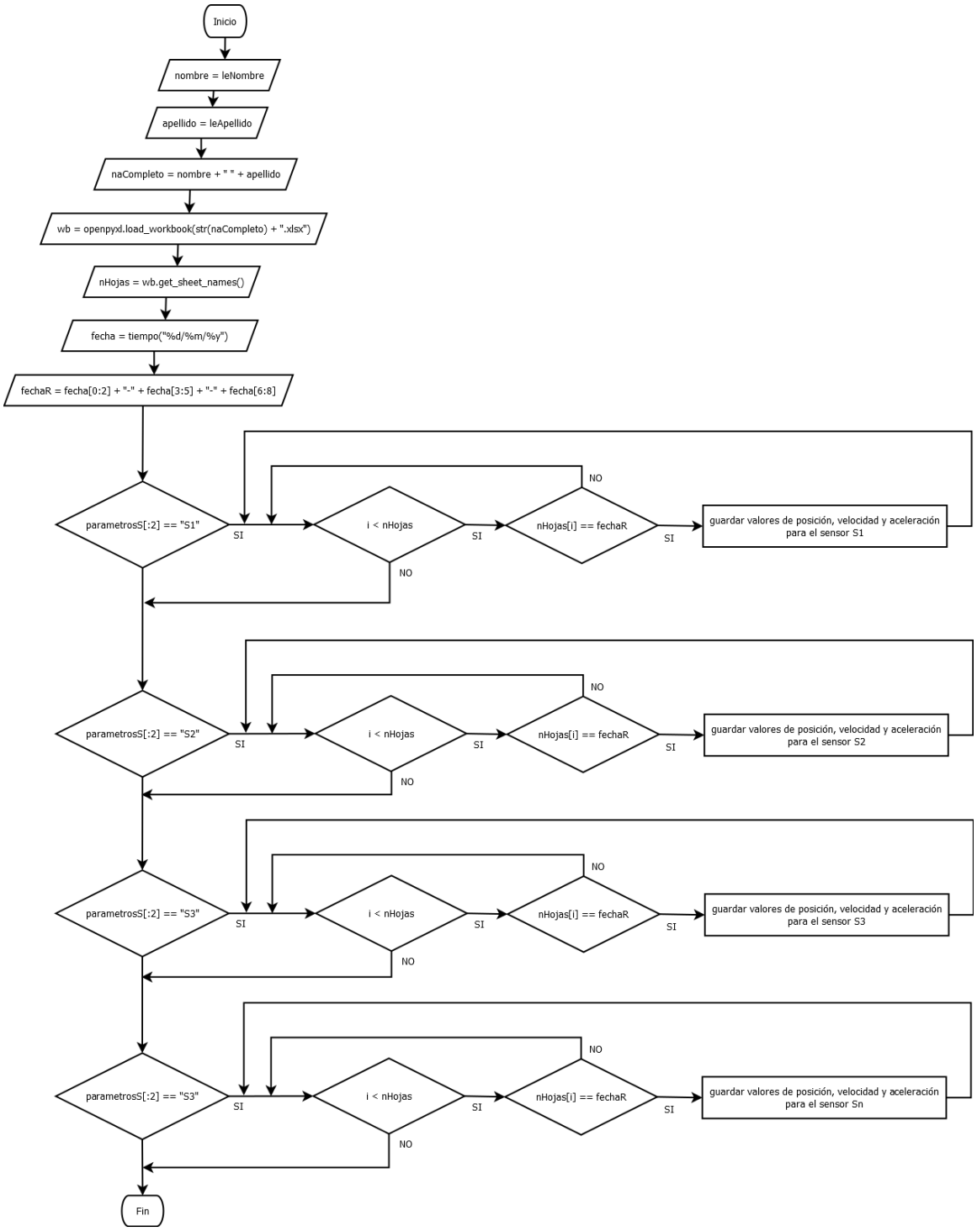


Figura 43. Subrutina identificar sensor.

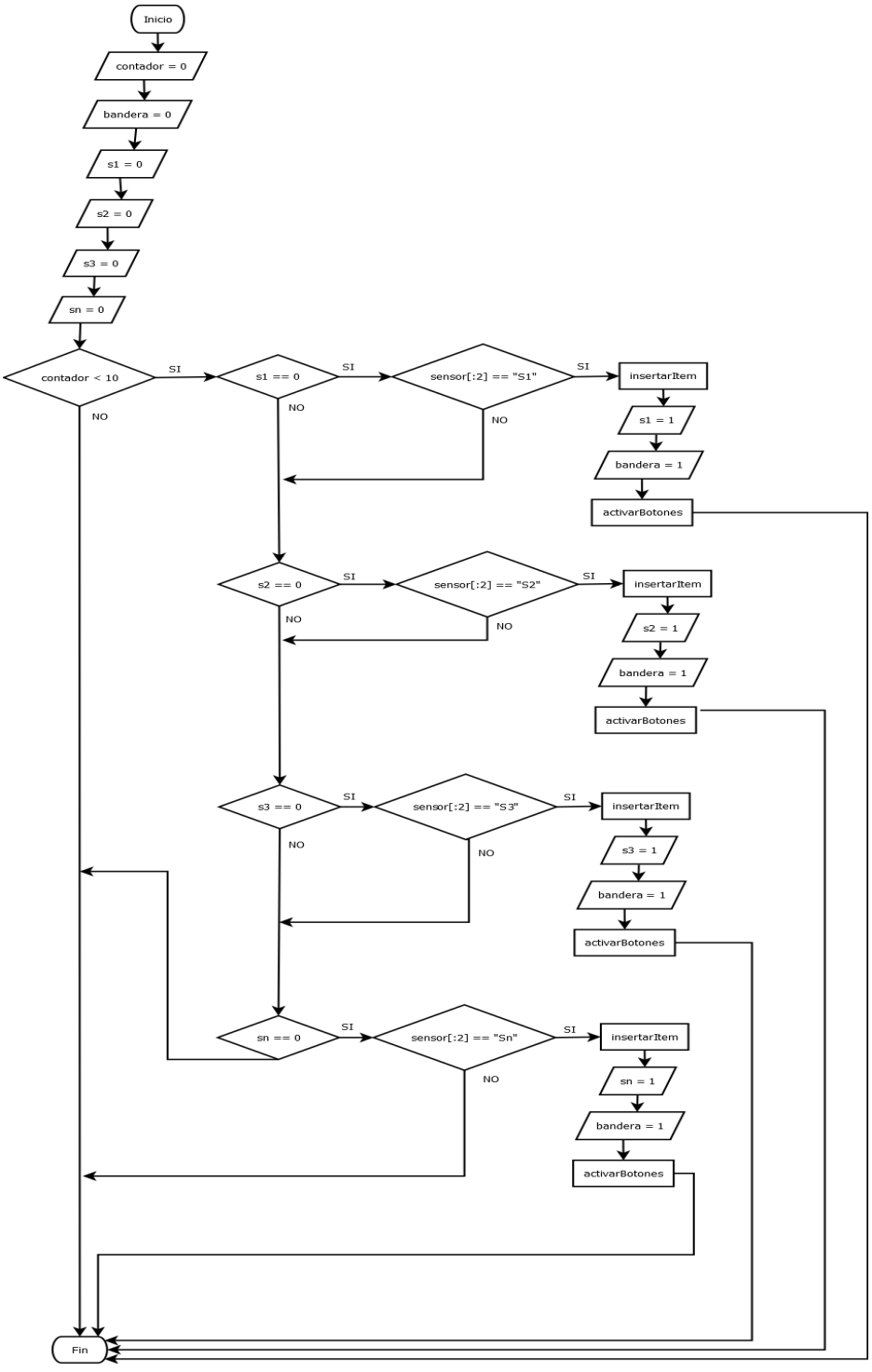


Figura 44. Subrutina play.

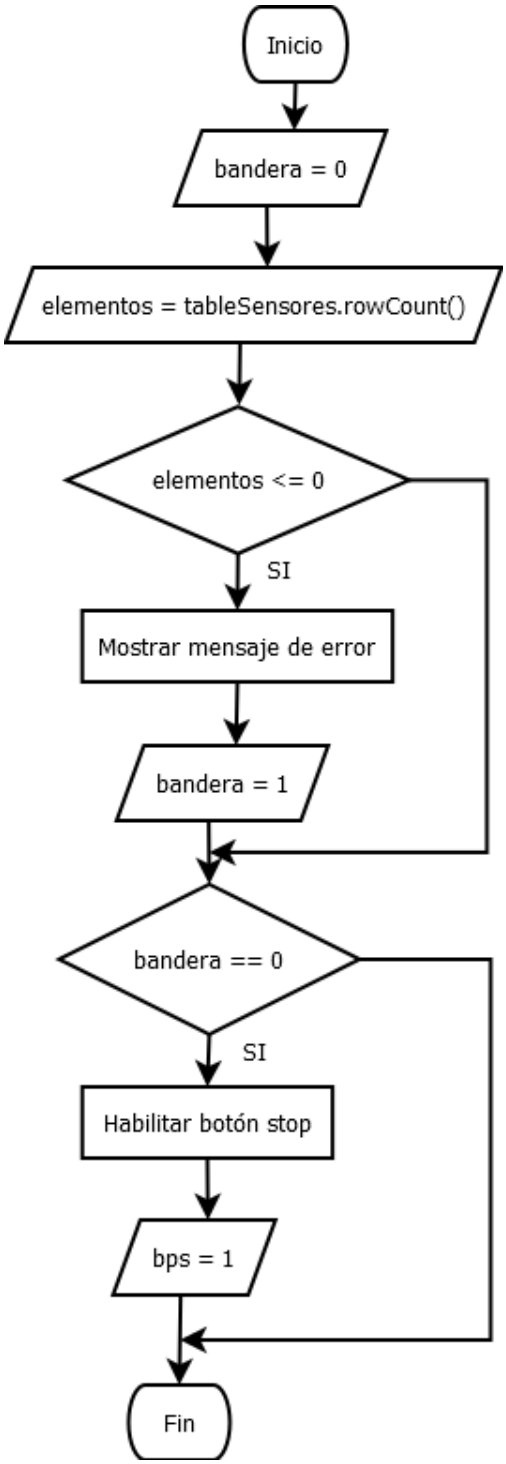
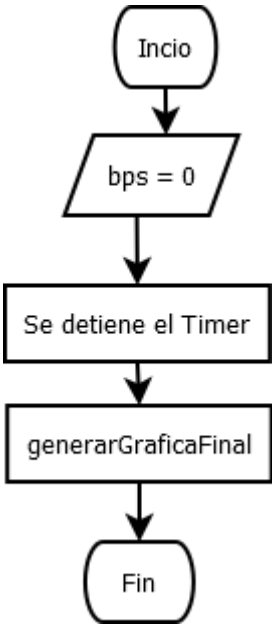


Figura 45. Subrutina stop.



B. CÓDIGO FUENTE

```
#*********************************************************#
#Nombre del archivo: LlamadaProyectoCAP.py                #
#Creado por: Erick Eduardo de Mata Calderón               #
#Fecha: 1/10/2017                                         #
#Descripción: El presente programa realiza el análisis y  #
#procesamiento de los datos obtenidos por los sensores    #
#de rastreo de movimiento en 3D, plasmándolos a través de #
#una interfaz gráfica de usuario                          #
#*********************************************************#

#*********************************************************#
#Se importan las librerias necesarias para la implementación#
#de la interfaz gráfica de usuario                         #
#*********************************************************#
import sys
from ProyectoCAP import *
import openpyxl
import xlsxwriter
import time
import serial
import LecturaSerial
import Parametros
import threading
import numpy as np
import serial
from scipy.signal import lfilter, firwin

#************************************************#
#Inicialización de la clase principal MiVentana  #
#************************************************#
class MiVentana(QtGui.QMainWindow):
    def __init__(self, parent=None):
        QtGui.QWidget.__init__(self, parent)
        self.ui = Ui_VentanaPrincipal()
        self.ui.setupUi(self)
        #Timer para obtener datos del sensor en tiempo real
        GraphTimer = QtCore.QTimer(self)
        GraphTimer.setInterval(50.)
        GraphTimer.start()
        #Variables de Posición del sensor No. 1 (S1)
        self.posxS1 = []
        self.posyS1 = []
        self.poszS1 = []
        self.posxS1.append(0)
        self.posyS1.append(0)
        self.poszS1.append(0)
        #Variables de Velocidad del sensor No. 1 (S1)
        self.velxS1 = []
        self.velyS1 = []
```

```
self.velzS1 = []
self.velxS1.append(0)
self.velyS1.append(0)
self.velzS1.append(0)
#Variables de Aceleracion del sensor No. 1 (S1)
self.acexS1 = []
self.aceyS1 = []
self.acezS1 = []
self.acexS1.append(0)
self.aceyS1.append(0)
self.acezS1.append(0)
#Variables de Posicion del sensor No. 2 (S2)
self.posxS2 = []
self.posyS2 = []
self.poszS2 = []
self.posxS2.append(0)
self.posyS2.append(0)
self.poszS2.append(0)
#Variables de Velocidad del sensor No. 2 (S2)
self.velxS2 = []
self.velyS2 = []
self.velzS2 = []
self.velxS2.append(0)
self.velyS2.append(0)
self.velzS2.append(0)
#Variables de Aceleracion del sensor No. 2 (S2)
self.acexS2 = []
self.aceyS2 = []
self.acezS2 = []
self.acexS2.append(0)
self.aceyS2.append(0)
self.acezS2.append(0)
#Variables de Posicion del sensor No. 3 (S3)
self.posxS3 = []
self.posyS3 = []
self.poszS3 = []
self.posxS3.append(0)
self.posyS3.append(0)
self.poszS3.append(0)
#Variables de Velocidad del sensor No. 3 (S3)
self.velxS3 = []
self.velyS3 = []
self.velzS3 = []
self.velxS3.append(0)
self.velyS3.append(0)
self.velzS3.append(0)
#Variables de Aceleracion del sensor No. 3 (S3)
self.acexS3 = []
self.aceyS3 = []
self.acezS3 = []
```

```
self.acexS3.append(0)
self.aceyS3.append(0)
self.acezS3.append(0)
#Variables de Posicion S1 Gráfica
self.posxS1G = []
self.posyS1G = []
self.poszS1G = []
self.posxS1G.append(0)
self.posyS1G.append(0)
self.poszS1G.append(0)
#Variables de Velocidad S1 Gráfica
self.velxS1G = []
self.velyS1G = []
self.velzS1G = []
self.velxS1G.append(0)
self.velyS1G.append(0)
self.velzS1G.append(0)
#Variables de Aceleracion S1 Gráfica
self.acexS1G = []
self.aceyS1G = []
self.acezS1G = []
self.acexS1G.append(0)
self.aceyS1G.append(0)
self.acezS1G.append(0)
#Variables de Posicion S2 Gráfica
self.posxS2G = []
self.posyS2G = []
self.poszS2G = []
self.posxS2G.append(0)
self.posyS2G.append(0)
self.poszS2G.append(0)
#Variables de Velocidad S2 Gráfica
self.velxS2G = []
self.velyS2G = []
self.velzS2G = []
self.velxS2G.append(0)
self.velyS2G.append(0)
self.velzS2G.append(0)
#Variables de Aceleracion S2 Gráfica
self.acexS2G = []
self.aceyS2G = []
self.acezS2G = []
self.acexS2G.append(0)
self.aceyS2G.append(0)
self.acezS2G.append(0)
#Variables de Posicion S3 Gráfica
self.posxS3G = []
self.posyS3G = []
self.poszS3G = []
self.posxS3G.append(0)
```

```
            self.posyS3G.append(0)
            self.poszS3G.append(0)
            #Variables de Velocidad S3 Gráfica
            self.velxS3G = []
            self.velyS3G = []
            self.velzS3G = []
            self.velxS3G.append(0)
            self.velyS3G.append(0)
            self.velzS3G.append(0)
            #Variables de Aceleracion S3 Gráfica
            self.acexS3G = []
            self.aceyS3G = []
            self.acezS3G = []
            self.acexS3G.append(0)
            self.aceyS3G.append(0)
            self.acezS3G.append(0)
            #Inicialización del timer 1
            self.GraphTimer1 = QtCore.QTimer(self)
            #Lista que acumula el tiempo transcurrido de una grabación
            self.tiempo  = []
            self.tiempo.append(0)
            #Variable que indica cantidad de divisiones en el eje y
            self.yshow= 10
            #Variable que indica el sensor que está enviando datos
            self.sensor = ""
            #Variable que indica el estado de los botones play-stop
            self.bps = 0
            #Identificador de sensor y las variables de velocidad,
posición y aceleración
            self.parametrosS = ""
            #Variable que cuenta número de ciclos
            self.contador = 0
            #Eventos ejecutados por los botones dentro de la interfaz
gráfica de usuario
            QtCore.QObject.connect(self.ui.pbVincular,
QtCore.SIGNAL("clicked()"), self.vincular)
            QtCore.QObject.connect(self.ui.pbEliminar,
QtCore.SIGNAL("clicked()"), self.eliminarItem)
            QtCore.QObject.connect(self.ui.tabGraficas,
QtCore.SIGNAL("clicked()"), self.agregarExtremidad)
            QtCore.QObject.connect(self.ui.pbAceptar,
QtCore.SIGNAL("clicked()"), self.configuracionUsuario)
            QtCore.QObject.connect(self.ui.pbGrafica,
QtCore.SIGNAL("clicked()"), self.generarGrafica)
            QtCore.QObject.connect(self.ui.pbConexion,
QtCore.SIGNAL("clicked()"), self.identificarSensor)
            QtCore.QObject.connect(self.ui.pbActualizar,
QtCore.SIGNAL("clicked()"), self.identificarSensor)
            QtCore.QObject.connect(self.ui.pbPlay,
QtCore.SIGNAL("clicked()"), self.play)
```

```
          QtCore.QObject.connect(self.ui.pbStop,
QtCore.SIGNAL("clicked()"), self.stop)
          QtCore.QObject.connect(self.ui.pbGenerarSimulacion,
QtCore.SIGNAL("clicked()"), self.generarSimulacion)
          QtCore.QObject.connect(self.ui.pbGenerarReporte,
QtCore.SIGNAL("clicked()"), self.generarReporte)boton

QtCore.QObject.connect(GraphTimer,QtCore.SIGNAL("timeout()"),self.datos
Obtenidos)

      #*********************************************************#
      #Nombre Subrutina: vincular                               #
      #Parámetros de entrada: 0                                 #
      #Descripción: Vincula el sensor seleccionado con la       #
      #extremidad seleccionada por el usuario.                  #
      #*********************************************************#
      def vincular(self):
          codSensor = self.ui.listSensores.selectedItems()[0].text()
#Hacer que en la interfaz grafica se seleciones un item por defecto
          PCuerpo = self.ui.comboBox.currentText()
          contFilas = self.ui.tableSensores.rowCount()
          contColumnas = 0
          bandera = 0
          for i in range(contFilas):
              elemento = self.ui.tableSensores.item(i, 1).text()
              codigo =  self.ui.tableSensores.item(i, 0).text()
              if elemento == PCuerpo:
                  QtGui.QMessageBox.warning(self, "Advertencia", "La
parte del cuerpo que desea vincular se encuentra activa en otro
sensor.")
                  bandera = 1
                  break
              if codigo == codSensor:
                  QtGui.QMessageBox.warning(self, "Advertencia", "El
sensor " + codigo +" que desea agregar ya ha sido seleccionado.")
                  bandera = 1
                  break
          if bandera == 0:
              self.ui.tableSensores.insertRow(contFilas)
              self.ui.tableSensores.setItem(contFilas, contColumnas,
QtGui.QTableWidgetItem(codSensor))
              self.ui.tableSensores.setItem(contFilas, contColumnas+1,
QtGui.QTableWidgetItem(PCuerpo))
              self.mostrarImagen()
              self.agregarExtremidad()
              self.agregarExtremidadSimulacion()
              self.agregarExtremidadReporte()

      #*********************************************************#
      #Nombre Subrutina: eliminarItem                           #
```

```
#Parámetros de entrada: 0                                          #
#Descripción: Elimina de la lista sensor extremidad la fila        #
#seleccionada por el usuario.                                      #
#*****************************************************************#
def eliminarItem(self):
    numElemento = self.ui.tableSensores.currentRow()
    itemText = self.ui.tableSensores.item(numElemento, 1).text()
    self.ocultarImagen(itemText)
    self.ui.tableSensores.removeRow(numElemento)
    for i in range (self.ui.comboBoxSeleccionE.count()):
        if itemText == self.ui.comboBoxSeleccionE.itemText(i):
            self.ui.comboBoxSeleccionE.removeItem(i)
            self.ui.comboBoxSeleccionES.removeItem(i)
            self.ui.comboBoxSeleccionER.removeItem(i)

#*****************************************************************#
#Nombre Subrutina: mostrarImagen                                   #
#Parámetros de entrada: 0                                          #
#Descripción: Resalta la imagen correspondiente a la               #
#extremidad seleccionada y vinculada por el usuario.               #
#*****************************************************************#
def mostrarImagen(self):
    parteCuerpo = self.ui.comboBox.currentText()
    if parteCuerpo == "Cabeza":
        self.ui.labelCabeza.setEnabled(True)
    if parteCuerpo == "Torso":
        self.ui.labelTorso.setEnabled(True)
    if parteCuerpo == "Brazo Derecho":
        self.ui.labelCodo.setEnabled(True)
    if parteCuerpo == "Brazo Izquierdo":
        self.ui.labelCodoIzq.setEnabled(True)
    if parteCuerpo == "Pierna Derecha":
        self.ui.labelPierna.setEnabled(True)
    if parteCuerpo == "Pierna Izquierda":
        self.ui.labelPiernaIzq.setEnabled(True)
    if parteCuerpo == "Pie Derecho":
        self.ui.labelPie.setEnabled(True)
    if parteCuerpo == "Pie Izquierdo":
        self.ui.labelPieIzq.setEnabled(True)
    if parteCuerpo == "Mano Derecha":
        self.ui.labelMano.setEnabled(True)
    if parteCuerpo == "Mano Izquierda":
        self.ui.labelManoIzq.setEnabled(True)



#*****************************************************************#
#Nombre Subrutina: ocultarImagen                                   #
```

```
#Parámetros de entrada: 1                                          #
#Descripción: Pone una apariencia opaca a la extremidad            #
#eliminada o desvinculada por el usuario.                          #
#*****************************************************************#
def ocultarImagen(self,parteCuerpo):
    if parteCuerpo == "Cabeza":
        self.ui.labelCabeza.setEnabled(False)
    if parteCuerpo == "Torso":
        self.ui.labelTorso.setEnabled(False)
    if parteCuerpo == "Brazo Derecho":
        self.ui.labelCodo.setEnabled(False)
    if parteCuerpo == "Brazo Izquierdo":
        self.ui.labelCodoIzq.setEnabled(False)
    if parteCuerpo == "Pierna Derecha":
        self.ui.labelPierna.setEnabled(False)
    if parteCuerpo == "Pierna Izquierda":
        self.ui.labelPiernaIzq.setEnabled(False)
    if parteCuerpo == "Pie Derecho":
        self.ui.labelPie.setEnabled(False)
    if parteCuerpo == "Pie Izquierdo":
        self.ui.labelPieIzq.setEnabled(False)
    if parteCuerpo == "Mano Derecha":
        self.ui.labelMano.setEnabled(False)
    if parteCuerpo == "Mano Izquierda":
        self.ui.labelManoIzq.setEnabled(False)

#*****************************************************************#
#Nombre Subrutina: agregarExtremidad                                #
#Parámetros de entrada: 0                                           #
#Descripción: Agrega la extremidad seleccionada por el              #
#usuario al combobox ubicado en la pestaña gráficas.                #
#*****************************************************************#
def agregarExtremidad(self):
    contFilas = self.ui.tableSensores.rowCount()
    elemento = self.ui.tableSensores.item(contFilas-1, 1).text()
    self.ui.comboBoxSeleccionE.addItem(elemento)

#*****************************************************************#
#Nombre Subrutina: agregarExtremidadSimulacion                      #
#Parámetros de entrada: 0                                           #
#Descripción: Agrega la extremidad seleccionada por el              #
#usuario al combobox ubicado en la pestaña Simulación.              #
#*****************************************************************#
def agregarExtremidadSimulacion(self):
    contFilas = self.ui.tableSensores.rowCount()
    elemento = self.ui.tableSensores.item(contFilas-1, 1).text()
    self.ui.comboBoxSeleccionES.addItem(elemento)

#*****************************************************************#
#Nombre Subrutina: agregarExtremidadReporte                         #
```

```
        #Parámetros de entrada: 0                                      #
        #Descripción: Agrega la extremidad seleccionada por el         #
        #usuario al combobox ubicado en la pestaña Reporte.            #
        #***************************************************************#
        def agregarExtremidadReporte(self):
            contFilas = self.ui.tableSensores.rowCount()
            elemento = self.ui.tableSensores.item(contFilas-1, 1).text()
            self.ui.comboBoxSeleccionER.addItem(elemento)

        #***************************************************************#
        #Nombre Subrutina: configuracionUsuario                         #
        #Parámetros de entrada: 0                                       #
        #Descripción: Obtiene los datos del usuario para generar un     #
        #archivo de Excel en donde se guarden los datos del Reporte.    #
        #***************************************************************#
        def configuracionUsuario(self):
            nombre = self.ui.leNombre.text()
            nombreR = '\ ' + nombre
            nombreR = nombreR[0]+nombreR[2:]
            apellido = self.ui.leApellido.text()
            naCompleto = nombre + " " + apellido
            naCompletoR = nombreR + " " + apellido
            ruta = "C:\Users\E. de Mata\Desktop\UVG\Decimo
Semestre\Megaproyecto"
            archivo = ruta + naCompletoR + ".xlsx"
            self.verificarArchivo(archivo,naCompleto)

        #***************************************************************#
        #Nombre Subrutina: verificarArchivo                             #
        #Parámetros de entrada: 2                                       #
        #Descripción: Verifica si el archivo existe para modificarlo    #
        #o crear uno nuevo con el nombre que ingresó el usuario.        #
        #***************************************************************#
        def verificarArchivo(self,archivo,naCompleto):
            try:
                fichero = open(archivo)
                fichero.close()
                print "El fichero existe"
                self.modificarArchivo(naCompleto)
                print "guardado"
            except:
                print "El fichero no existe"
                print naCompleto
                self.crearArchivo(naCompleto)



        #***************************************************************#
        #Nombre Subrutina: crearArchivo                                 #
```

```
#Parámetros de entrada: 1                                          #
#Descripción: Crea un nuveo archivo de Excel con el nombre  #
#ingresado por el usuario.                                         #
#*****************************************************#
def crearArchivo(self,naCompleto):
    workbook = xlsxwriter.Workbook(str(naCompleto) + '.xlsx')
    fecha = time.strftime("%d/%m/%y")
    fechaR = fecha[0:2] + "-" + fecha[3:5] + "-" + fecha[6:8]
    worksheet = workbook.add_worksheet(fechaR)
    self.encabezadoPaginas(worksheet)
    workbook.close()

#*****************************************************#
#Nombre Subrutina: modificarArchivo                          #
#Parámetros de entrada: 1                                    #
#Descripción: Agrega una pestaña nueva con la fecha actual  #
#al archivo de Excel existe con el nombre de usuario.       #
#*****************************************************#
def modificarArchivo(self,naCompleto):
    archivo = openpyxl.load_workbook(str(naCompleto) + ".xlsx")
    fecha = time.strftime("%d/%m/%y")
    fechaR = fecha[0:2] + "-" + fecha[3:5] + "-" + fecha[6:8]
    archivo.create_sheet(fechaR)
    archivo.save(str(naCompleto) + ".xlsx")
    self.encabezadoPaginas2()

#*****************************************************#
#Nombre Subrutina: encabezadoPaginas                         #
#Parámetros de entrada: 1                                    #
#Descripción: Crea el encabezado de una pestaña de Excel con#
#los datos ingresados por el usuario.                        #
#*****************************************************#
def encabezadoPaginas(self,worksheet):
    row = 0
    col = 0
    nombre = self.ui.leNombre.text()
    apellido = self.ui.leApellido.text()
    edad = self.ui.leEdad.text()
    deporte = self.ui.leDeporte.text()
    worksheet.write(row, col, 'Nombre')
    worksheet.write(row, col+1, str(nombre))
    worksheet.write(row+1, col, 'Apellido')
    worksheet.write(row+1, col+1, str(apellido))
    worksheet.write(row+2, col, 'Edad')
    worksheet.write(row+2, col+1, int(edad))
    worksheet.write(row+3, col, 'Disciplina Deportiva')
    worksheet.write(row+3, col+1, str(deporte))
    if self.ui.radioButtonMasculino.isChecked()== True:
        sexo = "Masculino"
    if self.ui.radioButtonFemenino.isChecked()== True:
```

```
            sexo = "Femenino"
        worksheet.write(row+4, col, 'Sexo')
        worksheet.write(row+4, col+1, sexo)

    #*************************************************************#
    #Nombre Subrutina: encabezadoPaginas2                         #
    #Parámetros de entrada: 0                                     #
    #Descripción: Crea el encabezado de una pestaña nueva de      #
    #un archivo de Excel existente con los datos ingresados       #
    #por el usuario.                                              #
    #*************************************************************#
    def encabezadoPaginas2(self):
        nombre = self.ui.leNombre.text()
        apellido = self.ui.leApellido.text()
        edad = self.ui.leEdad.text()
        deporte = self.ui.leDeporte.text()
        dep = deporte
        if self.ui.radioButtonMasculino.isChecked()== True:
            sexo = "Masculino"
            sex = sexo
        if self.ui.radioButtonFemenino.isChecked()== True:
            sexo = "Femenino"
            sex = sexo
        naCompleto = nombre + " " + apellido
        wb = openpyxl.load_workbook(str(naCompleto) + ".xlsx")
        print wb.get_sheet_names()
        nHojas = wb.get_sheet_names()
        fecha = time.strftime("%d/%m/%y")
        fechaR = fecha[0:2] + "-" + fecha[3:5] + "-" + fecha[6:8]
        for i in range (len(nHojas)):
            if nHojas[i] == fechaR:
                hoja = wb.get_sheet_by_name(fechaR)
                hoja.append(["Nombre"])
                hoja.append(["Apellido"])
                hoja.append(["Edad"])
                hoja.append(["Disciplina Deportiva"])
                hoja.append(["Sexo",sex])
        hoja.append([""])
        hoja.append(["Posicion X","Posicion Y","Posicion Z","Velocidad
X","Velocidad Y","Velocidad Z","Aceleracion X","Aceleracion
Y","Aceleracion Z"])
        wb.save(str(naCompleto) + ".xlsx")

    #*************************************************************#
    #Nombre Subrutina: identificarSensor                          #
    #Parámetros de entrada: 0                                     #
    #Descripción: Identifica el sensor que está enviando datos a#
    #la estación central de sensores.                             #
    #*************************************************************#
    def identificarSensor(self):
```

```
            contador = 0
            bandera = 0
            s1 = 0
            s2 = 0
            s3 = 0
            s4 = 0
            s5 = 0
            s6 = 0
            s7 = 0
            s8 = 0
            s9 = 0
            s10 = 0
            QtGui.QMessageBox.information(self, "Conexion", "La
comunicacion se esta estableciendo, espere un momento.")
            for i in range(10):
                self.ui.listSensores.takeItem(0)
            while contador < 10:
                print self.sensor
                if s1 == 0:
                    if self.sensor[:2] == "S1":
                        self.ui.listSensores.insertItem(0,"S1")
                        s1 = 1
                        bandera = 1
                if s2 == 0:
                    if self.sensor[:2] == "S2":
                        self.ui.listSensores.insertItem(1,"S2")
                        s2 = 1
                        bandera = 1
                if s3 == 0:
                    if self.sensor[:2] == "S3":
                        self.ui.listSensores.insertItem(1,"S3")
                        s3 = 1
                        bandera = 1
                if s4 == 0:
                    if self.sensor[:2] == "S4":
                        self.ui.listSensores.insertItem(1,"S4")
                        s4 = 1
                        bandera = 1
                if s5 == 0:
                    if self.sensor[:2] == "S5":
                        self.ui.listSensores.insertItem(1,"S5")
                        s5 = 1
                        bandera = 1
                if s6 == 0:
                    if self.sensor[:2] == "S6":
                        self.ui.listSensores.insertItem(1,"S6")
                        s6 = 1
                        bandera = 1
                if s7 == 0:
                    if self.sensor[:2] == "S7":
```

```
                    self.ui.listSensores.insertItem(1,"S7")
                    s7 = 1
                    bandera = 1
            if s8 == 0:
                if self.sensor[:2] == "S8":
                    self.ui.listSensores.insertItem(1,"S8")
                    s8 = 1
                    bandera = 1
            if s9 == 0:
                if self.sensor[:2] == "S9":
                    self.ui.listSensores.insertItem(1,"S9")
                    s9 = 1
                    bandera = 1
            if s10 == 0:
                if self.sensor[:2] == "S10":
                    self.ui.listSensores.insertItem(1,"S10")
                    s10 = 1
                    bandera = 1
            contador = contador + 1
        if bandera == 1:
            QtGui.QMessageBox.information(self, "Conexion", "La
conexion ha sido establecida.")
            self.ui.pbPlay.setEnabled(True)
            self.ui.pbVincular.setEnabled(True)
            self.ui.pbActualizar.setEnabled(True)
            self.ui.pbEliminar.setEnabled(True)
            self.ui.pbEliminar.setEnabled(True)
            self.ui.pbGrafica.setEnabled(True)
            self.ui.pbGenerarSimulacion.setEnabled(True)
            self.ui.pbGenerarReporte.setEnabled(True)
        if bandera == 0:
            QtGui.QMessageBox.critical(self, "Conexion", "No se ha
podido establecer conexion, intente de nuevo.")

    #*********************************************************#
    #Nombre Subrutina: play                                   #
    #Parámetros de entrada: 0                                 #
    #Descripción: Inicializa la bandera play-stop en 1 para   #
    #iniciar la grabación de datos.                           #
    #*********************************************************#
    def play(self):
        bandera = 0
        elementos = self.ui.tableSensores.rowCount()
        if elementos <= 0:
            QtGui.QMessageBox.warning(self, "Play", "No se puede
iniciar la grabacion de datos, porque no se ha asignado ningun sensor a
una parte del cuerpo.")
            bandera = 1
        if bandera == 0:
            self.ui.pbStop.setEnabled(True)
```

```
        self.bps = 1

#************************************************************#
#Nombre Subrutina: asignacion                                #
#Parámetros de entrada: 1                                    #
#Descripción: Identifica el o los sensores que se están      #
#comunicando con la estación base para asignar los           #
#parámetros de posición, velocidad y aceleración             #
#correspondientes.                                           #
#************************************************************#
def asignacion(self,parametrosS):
    if parametrosS[:2] == "S1":
        print "poner pxS1 = pos_vel_ace[2:4]"
        datos = parametrosS.split(',')
        self.posxS1.append(float(datos[1]))
        self.posyS1.append(float(datos[2]))
        self.poszS1.append(float(datos[3]))
        self.velxS1.append(float(datos[4]))
        self.velyS1.append(float(datos[5]))
        self.velzS1.append(float(datos[6]))
        self.acexS1.append(float(datos[7]))
        self.aceyS1.append(float(datos[8]))
        self.acezS1.append(float(datos[9]))
    if parametrosS[:2] == "S2":
        print "poner pxS1 = pos_vel_ace[2:4]"
        datos = parametrosS.split(',')
        self.posxS2.append(float(datos[1]))
        self.posyS2.append(float(datos[2]))
        self.poszS2.append(float(datos[3]))
        self.velxS2.append(float(datos[4]))
        self.velyS2.append(float(datos[5]))
        self.velzS2.append(float(datos[6]))
        self.acexS2.append(float(datos[7]))
        self.aceyS2.append(float(datos[8]))
        self.acezS2.append(float(datos[9]))
    if parametrosS[:2] == "S3":
        print "poner pxS3 = pos_vel_ace[2:4]"
        datos = parametrosS.split(',')
        self.posxS3.append(float(datos[1]))
        self.posyS3.append(float(datos[2]))
        self.poszS3.append(float(datos[3]))
        self.velxS3.append(float(datos[4]))
        self.velyS3.append(float(datos[5]))
        self.velzS3.append(float(datos[6]))
        self.acexS3.append(float(datos[7]))
        self.aceyS3.append(float(datos[8]))
        self.acezS3.append(float(datos[9]))
        self.guardarDatos(parametrosS)

#************************************************************#
```

```
        #Nombre Subrutina: guardarDatos                                  #
        #Parámetros de entrada: 1                                        #
        #Descripción: Identifica el o los sensores que se están          #
        #comunicando con la estación base para guardar los               #
        #parámetros de posición, velocidad y aceleración                 #
        #correspondientes.                                               #
        #*****************************************************************#
        def guardarDatos(self,parametrosS):
            nombre = self.ui.leNombre.text()
            apellido = self.ui.leApellido.text()
            naCompleto = nombre + " " + apellido
            wb = openpyxl.load_workbook(str(naCompleto) + ".xlsx")
            nHojas = wb.get_sheet_names()
            fecha = time.strftime("%d/%m/%y")
            fechaR = fecha[0:2] + "-" + fecha[3:5] + "-" + fecha[6:8]
            if parametrosS[:2] == "S1":
                datos = parametrosS.split(',')
                for i in range (len(nHojas)):
                    if nHojas[i] == fechaR:
                        hoja = wb.get_sheet_by_name(fechaR)

hoja.append([float(datos[1]),float(datos[2]),float(datos[3]),float(dato
s[4]),float(datos[5]),float(datos[6]),float(datos[7]),float(datos[8]),f
loat(datos[9])])
                wb.save(str(naCompleto) + ".xlsx")
            if parametrosS[:2] == "S2":
                datos = parametrosS.split(',')
                for i in range (len(nHojas)):
                    if nHojas[i] == fechaR:
                        hoja = wb.get_sheet_by_name(fechaR)

hoja.append([float(datos[1]),float(datos[2]),float(datos[3]),float(dato
s[4]),float(datos[5]),float(datos[6]),float(datos[7]),float(datos[8]),f
loat(datos[9])])
                wb.save(str(naCompleto) + ".xlsx")
            if parametrosS[:2] == "S3":
                datos = parametrosS.split(',')
                for i in range (len(nHojas)):
                    if nHojas[i] == fechaR:
                        hoja = wb.get_sheet_by_name(fechaR)

hoja.append([float(datos[1]),float(datos[2]),float(datos[3]),float(dato
s[4]),float(datos[5]),float(datos[6]),float(datos[7]),float(datos[8]),f
loat(datos[9])])
                wb.save(str(naCompleto) + ".xlsx")



        #*****************************************************************#
        #Nombre Subrutina: stop                                           #
```

```
        #Parámetros de entrada: 0                                          #
        #Descripción: Inicializa la bandera play-stop en 0 para            #
        #detener la grabación de datos.                                    #
        #*****************************************************************#
        def stop(self):
            self.bps = 0
            self.GraphTimer1.stop()
            QtGui.QMessageBox.information(self, "Stop", "La grabacion de
datos se ha detenido.")
            self.generarGraficaFinal()

        #*****************************************************************#
        #Nombre Subrutina: generarGrafica                                  #
        #Parámetros de entrada: 0                                          #
        #Descripción: Inicializa el timer correspondiente para             #
        #iniciar las gráficas de la extremidad seleccionada.               #
        #*****************************************************************#
        def generarGrafica(self):
            if self.bps == 1:
                self.GraphTimer1 = QtCore.QTimer(self)
                self.GraphTimer1.setInterval(500.)
                self.GraphTimer1.start()

QtCore.QObject.connect(self.GraphTimer1,QtCore.SIGNAL("timeout()"),self
.mostrarGrafica)

        #*****************************************************************#
        #Nombre Subrutina: mostrarGrafica                                  #
        #Parámetros de entrada: 0                                          #
        #Descripción: Muestra la gráfica en tiempo real de los datos       #
        #obtenidos por el sensor vinculado a la extremidad                 #
        #selecionada.                                                      #
        #*****************************************************************#
        def mostrarGrafica(self):
            self.tiempo.append(self.tiempo.__getitem__(-1)+((500.)/1000))
#/1000 para pasar de ms a s
            parteCuerpo = self.ui.comboBoxSeleccionER.currentText()
            contFilas = self.ui.tableSensores.rowCount()
            for i in range(contFilas):
                elemento = self.ui.tableSensores.item(i, 1).text()
                if elemento == parteCuerpo:
                    codigo =  self.ui.tableSensores.item(i, 0).text()
                    break
            if codigo == "S1":
                datos = self.parametrosS.split(',')
                self.posxS1G.append(float(datos[1]))
                self.posyS1G.append(float(datos[2]))
                self.poszS1G.append(float(datos[3]))
                self.velxS1G.append(float(datos[4]))
                self.velyS1G.append(float(datos[5]))
```

```
                self.velzS1G.append(float(datos[6]))
                self.acexS1G.append(float(datos[7]))
                self.aceyS1G.append(float(datos[8]))
                self.acezS1G.append(float(datos[9]))

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS1G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS1G,"g",label=
'y')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS1G,"r",label=
'z')
            self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")
            self.ui.mpGraficaPosicion.axes.hold(True)
            self.ui.mpGraficaPosicion.axes.set_ylim([-0.2, 0.2])
            if (len(self.tiempo)>self.yshow):

self.ui.mpGraficaPosicion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
            self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS1G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS1G,"g",label='y')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS1G,"r",label='z')
            self.ui.mpVelocidad.axes.set_title("Velocidad - tiempo[s]")
            self.ui.mpVelocidad.axes.hold(True)
            self.ui.mpVelocidad.axes.set_ylim([-1, 1])
            if (len(self.tiempo)>self.yshow):

self.ui.mpVelocidad.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
            self.ui.mpVelocidad.draw()


self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS1G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS1G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS1G,"r",label='z')
            self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
            self.ui.mpAceleracion.axes.hold(True)
            self.ui.mpAceleracion.axes.set_ylim([-10, 10])
            if (len(self.tiempo)>self.yshow):
```

```
self.ui.mpAceleracion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
            self.ui.mpAceleracion.draw()
            if codigo == "S2":
                datos = self.parametrosS.split(',')
                self.posxS2G.append(float(datos[1]))
                self.posyS2G.append(float(datos[2]))
                self.poszS2G.append(float(datos[3]))
                self.velxS2G.append(float(datos[4]))
                self.velyS2G.append(float(datos[5]))
                self.velzS2G.append(float(datos[6]))
                self.acexS2G.append(float(datos[7]))
                self.aceyS2G.append(float(datos[8]))
                self.acezS2G.append(float(datos[9]))

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS2G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS2G,"g",label=
'y')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS2G,"r",label=
'z')
            self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")
            self.ui.mpGraficaPosicion.axes.hold(True)
            self.ui.mpGraficaPosicion.axes.set_ylim([-0.2, 0.2])
            if (len(self.tiempo)>self.yshow):

self.ui.mpGraficaPosicion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
            self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS2G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS2G,"g",label='y')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS2G,"r",label='z')
            self.ui.mpVelocidad.axes.set_title("Velocidad - tiempo[s]")
            self.ui.mpVelocidad.axes.hold(True)
            self.ui.mpVelocidad.axes.set_ylim([-1, 1])
            if (len(self.tiempo)>self.yshow):

self.ui.mpVelocidad.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
            self.ui.mpVelocidad.draw()
```

```
self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS2G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS2G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS2G,"r",label='z')
          self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
          self.ui.mpAceleracion.axes.hold(True)
          self.ui.mpAceleracion.axes.set_ylim([-10, 10])
          if (len(self.tiempo)>self.yshow):

self.ui.mpAceleracion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
          self.ui.mpAceleracion.draw()
          if codigo == "S3":
              datos = self.parametrosS.split(',')
              self.posxS3G.append(float(datos[1]))
              self.posyS3G.append(float(datos[2]))
              self.poszS3G.append(float(datos[3]))
              self.velxS3G.append(float(datos[4]))
              self.velyS3G.append(float(datos[5]))
              self.velzS3G.append(float(datos[6]))
              self.acexS3G.append(float(datos[7]))
              self.aceyS3G.append(float(datos[8]))
              self.acezS3G.append(float(datos[9]))

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS3G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS3G,"g",label=
'y')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS3G,"r",label=
'z')
          self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")
          self.ui.mpGraficaPosicion.axes.hold(True)
          self.ui.mpGraficaPosicion.axes.set_ylim([-0.2, 0.2])
          if (len(self.tiempo)>self.yshow):

self.ui.mpGraficaPosicion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
          self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS3G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS3G,"g",label='y')
```

```
self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS3G,"r",label='z')
          self.ui.mpVelocidad.axes.set_title("Velocidad - tiempo[s]")
          self.ui.mpVelocidad.axes.hold(True)
          self.ui.mpVelocidad.axes.set_ylim([-1, 1])
          if (len(self.tiempo)>self.yshow):

self.ui.mpVelocidad.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
          self.ui.mpVelocidad.draw()


self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS3G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS3G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS3G,"r",label='z')
          self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
          self.ui.mpAceleracion.axes.hold(True)
          self.ui.mpAceleracion.axes.set_ylim([-10, 10])
          if (len(self.tiempo)>self.yshow):

self.ui.mpAceleracion.axes.set_xlim([self.tiempo.__getitem__(-
self.yshow+5),self.tiempo.__getitem__(-1)])
          self.ui.mpAceleracion.draw()

      #*********************************************************#
      #Nombre Subrutina: generarGraficaFinal                    #
      #Parámetros de entrada: 0                                 #
      #Descripción: Muestra la gráfica generada durante toda la #
      #grabación.                                               #
      #*********************************************************#
      def generarGraficaFinal(self):
          parteCuerpo = self.ui.comboBoxSeleccionER.currentText()
          contFilas = self.ui.tableSensores.rowCount()
          for i in range(contFilas):
              elemento = self.ui.tableSensores.item(i, 1).text()
              if elemento == parteCuerpo:
                  codigo =  self.ui.tableSensores.item(i, 0).text()
                  break
          if codigo == "S1":

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS1G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS1G,"g",label=
'y')
```

```
self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS1G,"r",label=
'z')
                self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")

self.ui.mpGraficaPosicion.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS1G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS1G,"g",label='y')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS1G,"r",label='z')
                self.ui.mpVelocidad.axes.set_title("Velocidad -
tiempo[s]")
                self.ui.mpVelocidad.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpVelocidad.draw()


self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS1G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS1G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS1G,"r",label='z')
                self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
                self.ui.mpAceleracion.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpAceleracion.draw()
            if codigo == "S2":

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS2G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS2G,"g",label=
'y')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS2G,"r",label=
'z')
                self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")

self.ui.mpGraficaPosicion.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS2G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS2G,"g",label='y')
```

```
self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS2G,"r",label='z')
                self.ui.mpVelocidad.axes.set_title("Velocidad -
tiempo[s]")
                self.ui.mpVelocidad.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpVelocidad.draw()


self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS2G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS2G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS2G,"r",label='z')
                self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
                self.ui.mpAceleracion.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpAceleracion.draw()
            if codigo == "S3":

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posxS3G,"b",label=
'x')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.posyS3G,"g",label=
'y')

self.ui.mpGraficaPosicion.axes.plot(self.tiempo,self.poszS3G,"r",label=
'z')
                self.ui.mpGraficaPosicion.axes.set_title("Posicion -
tiempo[s]")

self.ui.mpGraficaPosicion.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpGraficaPosicion.draw()


self.ui.mpVelocidad.axes.plot(self.tiempo,self.velxS3G,"b",label='x')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velyS3G,"g",label='y')

self.ui.mpVelocidad.axes.plot(self.tiempo,self.velzS3G,"r",label='z')
                self.ui.mpVelocidad.axes.set_title("Velocidad -
tiempo[s]")
                self.ui.mpVelocidad.axes.set_xlim([0,max(self.tiempo)])
                self.ui.mpVelocidad.draw()


self.ui.mpAceleracion.axes.plot(self.tiempo,self.acexS3G,"b",label='x')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.aceyS3G,"g",label='y')

self.ui.mpAceleracion.axes.plot(self.tiempo,self.acezS3G,"r",label='z')
```

```
            self.ui.mpAceleracion.axes.set_title("Aceleracion -
tiempo[s]")
            self.ui.mpAceleracion.axes.set_xlim([0,max(self.tiempo)])
            self.ui.mpAceleracion.draw()


    def generarSimulacion(self):
        bandera = 0
        elementos = self.ui.comboBoxSeleccionES.count()
        if elementos <= 0:
            QtGui.QMessageBox.warning(self, "Advertencia", "La
simulacion no se puede iniciar debido a que no se ha vinculado un
sensor con una parte del cuerpo.")
            bandera = 1
        if bandera == 0:
            print "iniciar simulacion"

    #*************************************************************#
    #Nombre Subrutina: generarReporte                             #
    #Parámetros de entrada: 0                                     #
    #Descripción: Genera el reporte de la extremidad              #
    #seleccionada.                                                #
    #*************************************************************#
    def generarReporte(self):
        bandera = 0
        elementos = self.ui.comboBoxSeleccionER.count()
        if elementos <= 0:
            QtGui.QMessageBox.warning(self, "Advertencia", "El reporte
no se puede generar debido a que no se ha vinculado un sensor con una
parte del cuerpo.")
            bandera = 1
        if bandera == 0:
            parteCuerpo = self.ui.comboBoxSeleccionER.currentText()
            contFilas = self.ui.tableSensores.rowCount()
            for i in range(contFilas):
                elemento = self.ui.tableSensores.item(i, 1).text()
                if elemento == parteCuerpo:
                    codigo =  self.ui.tableSensores.item(i, 0).text()
                    break
            if codigo == "S1":
                sumPx = 0
                sumPy = 0
                sumPz = 0
                sumVx = 0
                sumVy = 0
                sumVz = 0
                sumAx = 0
                sumAy = 0
                sumAz = 0
```

```
for i in range(len(self.posxS1)):
    sumPx = sumPx + self.posxS1[i]
    sumVx = sumVx + self.velxS1[i]
    sumAx = sumAx + self.acexS1[i]
promPx = sumPx/len(self.posxS1)
promVx = sumVx/len(self.velxS1)
promAx = sumAx/len(self.acexS1)

for i in range(len(self.posyS1)):
    sumPy = sumPy + self.posyS1[i]
    sumVy = sumVy + self.velyS1[i]
    sumAy = sumAy + self.aceyS1[i]
promPy = sumPy/len(self.posyS1)
promVy = sumVy/len(self.velyS1)
promAy = sumAy/len(self.aceyS1)

for i in range(len(self.poszS1)):
    sumPz = sumPz + self.poszS1[i]
    sumVz = sumVz + self.velzS1[i]
    sumAz = sumAz + self.acezS1[i]
promPz = sumPz/len(self.poszS1)
promVz = sumVz/len(self.velzS1)
promAz = sumAz/len(self.acezS1)

#Eje X
self.ui.labelVmax1.setText(str(max(self.posxS1)))
self.ui.labelVmax2.setText(str(min(self.posxS1)))
self.ui.labelVmax3.setText(str(promPx))

self.ui.labelVmax1_2.setText(str(max(self.velxS1)))
self.ui.labelVmax2_2.setText(str(min(self.velxS1)))
self.ui.labelVmax3_2.setText(str(promVx))

self.ui.labelVmax1_3.setText(str(max(self.acexS1)))
self.ui.labelVmax2_3.setText(str(min(self.acexS1)))
self.ui.labelVmax3_3.setText(str(promAx))

#Eje Y
self.ui.labelVmax1_4.setText(str(max(self.posyS1)))
self.ui.labelVmax2_4.setText(str(min(self.posyS1)))
self.ui.labelVmax3_4.setText(str(promPy))

self.ui.labelVmax1_7.setText(str(max(self.velyS1)))
self.ui.labelVmax2_7.setText(str(min(self.velyS1)))
self.ui.labelVmax3_7.setText(str(promVy))

self.ui.labelVmax1_8.setText(str(max(self.aceyS1)))
self.ui.labelVmax2_8.setText(str(min(self.aceyS1)))
self.ui.labelVmax3_8.setText(str(promAy))
```

```
            #Eje z
            self.ui.labelVmax1_6.setText(str(max(self.poszS1)))
            self.ui.labelVmax2_6.setText(str(min(self.poszS1)))
            self.ui.labelVmax3_6.setText(str(promPz))

            self.ui.labelVmax1_9.setText(str(max(self.velzS1)))
            self.ui.labelVmax2_9.setText(str(min(self.velzS1)))
            self.ui.labelVmax3_9.setText(str(promVz))

            self.ui.labelVmax1_10.setText(str(max(self.acezS1)))
            self.ui.labelVmax2_10.setText(str(min(self.acezS1)))
            self.ui.labelVmax3_10.setText(str(promAz))

        if codigo == "S2":
            sumPx = 0
            sumPy = 0
            sumPz = 0
            sumVx = 0
            sumVy = 0
            sumVz = 0
            sumAx = 0
            sumAy = 0
            sumAz = 0

            for i in range(len(self.posxS2)):
                sumPx = sumPx + self.posxS2[i]
                sumVx = sumVx + self.velxS2[i]
                sumAx = sumAx + self.acexS2[i]
            promPx = sumPx/len(self.posxS2)
            promVx = sumVx/len(self.velxS2)
            promAx = sumAx/len(self.acexS2)

            for i in range(len(self.posyS2)):
                sumPy = sumPy + self.posyS2[i]
                sumVy = sumVy + self.velyS2[i]
                sumAy = sumAy + self.aceyS2[i]
            promPy = sumPy/len(self.posyS2)
            promVy = sumVy/len(self.velyS2)
            promAy = sumAy/len(self.aceyS2)

            for i in range(len(self.poszS2)):
                sumPz = sumPz + self.poszS2[i]
                sumVz = sumVz + self.velzS2[i]
                sumAz = sumAz + self.acezS2[i]
            promPz = sumPz/len(self.poszS2)
            promVz = sumVz/len(self.velzS2)
            promAz = sumAz/len(self.acezS2)

            #Eje X
            self.ui.labelVmax1.setText(str(max(self.posxS2)))
```

```
            self.ui.labelVmax2.setText(str(min(self.posxS2)))
            self.ui.labelVmax3.setText(str(promPx))

            self.ui.labelVmax1_2.setText(str(max(self.velxS2)))
            self.ui.labelVmax2_2.setText(str(min(self.velxS2)))
            self.ui.labelVmax3_2.setText(str(promVx))

            self.ui.labelVmax1_3.setText(str(max(self.acexS2)))
            self.ui.labelVmax2_3.setText(str(min(self.acexS2)))
            self.ui.labelVmax3_3.setText(str(promAx))

            #Eje Y
            self.ui.labelVmax1_4.setText(str(max(self.posyS2)))
            self.ui.labelVmax2_4.setText(str(min(self.posyS2)))
            self.ui.labelVmax3_4.setText(str(promPy))

            self.ui.labelVmax1_7.setText(str(max(self.velyS2)))
            self.ui.labelVmax2_7.setText(str(min(self.velyS2)))
            self.ui.labelVmax3_7.setText(str(promVy))

            self.ui.labelVmax1_8.setText(str(max(self.aceyS2)))
            self.ui.labelVmax2_8.setText(str(min(self.aceyS2)))
            self.ui.labelVmax3_8.setText(str(promAy))

            #Eje z
            self.ui.labelVmax1_6.setText(str(max(self.poszS2)))
            self.ui.labelVmax2_6.setText(str(min(self.poszS2)))
            self.ui.labelVmax3_6.setText(str(promPz))

            self.ui.labelVmax1_9.setText(str(max(self.velzS2)))
            self.ui.labelVmax2_9.setText(str(min(self.velzS2)))
            self.ui.labelVmax3_9.setText(str(promVz))

            self.ui.labelVmax1_10.setText(str(max(self.acezS2)))
            self.ui.labelVmax2_10.setText(str(min(self.acezS2)))
            self.ui.labelVmax3_10.setText(str(promAz))

        if codigo == "S3":
            sumPx = 0
            sumPy = 0
            sumPz = 0
            sumVx = 0
            sumVy = 0
            sumVz = 0
            sumAx = 0
            sumAy = 0
            sumAz = 0

            for i in range(len(self.posxS3)):
                sumPx = sumPx + self.posxS3[i]
```

```
        sumVx = sumVx + self.velxS3[i]
        sumAx = sumAx + self.acexS3[i]
    promPx = sumPx/len(self.posxS3)
    promVx = sumVx/len(self.velxS3)
    promAx = sumAx/len(self.acexS3)

    for i in range(len(self.posyS3)):
        sumPy = sumPy + self.posyS3[i]
        sumVy = sumVy + self.velyS3[i]
        sumAy = sumAy + self.aceyS3[i]
    promPy = sumPy/len(self.posyS3)
    promVy = sumVy/len(self.velyS3)
    promAy = sumAy/len(self.aceyS3)

    for i in range(len(self.poszS3)):
        sumPz = sumPz + self.poszS3[i]
        sumVz = sumVz + self.velzS3[i]
        sumAz = sumAz + self.acezS3[i]
    promPz = sumPz/len(self.poszS3)
    promVz = sumVz/len(self.velzS3)
    promAz = sumAz/len(self.acezS3)

    #Eje X
    self.ui.labelVmax1.setText(str(max(self.posxS3)))
    self.ui.labelVmax2.setText(str(min(self.posxS3)))
    self.ui.labelVmax3.setText(str(promPx))

    self.ui.labelVmax1_2.setText(str(max(self.velxS3)))
    self.ui.labelVmax2_2.setText(str(min(self.velxS3)))
    self.ui.labelVmax3_2.setText(str(promVx))

    self.ui.labelVmax1_3.setText(str(max(self.acexS3)))
    self.ui.labelVmax2_3.setText(str(min(self.acexS3)))
    self.ui.labelVmax3_3.setText(str(promAx))

    #Eje Y
    self.ui.labelVmax1_4.setText(str(max(self.posyS3)))
    self.ui.labelVmax2_4.setText(str(min(self.posyS3)))
    self.ui.labelVmax3_4.setText(str(promPy))

    self.ui.labelVmax1_7.setText(str(max(self.velyS3)))
    self.ui.labelVmax2_7.setText(str(min(self.velyS3)))
    self.ui.labelVmax3_7.setText(str(promVy))

    self.ui.labelVmax1_8.setText(str(max(self.aceyS3)))
    self.ui.labelVmax2_8.setText(str(min(self.aceyS3)))
    self.ui.labelVmax3_8.setText(str(promAy))

    #Eje z
    self.ui.labelVmax1_6.setText(str(max(self.poszS3)))
```

```
                self.ui.labelVmax2_6.setText(str(min(self.poszS3)))
                self.ui.labelVmax3_6.setText(str(promPz))

                self.ui.labelVmax1_9.setText(str(max(self.velzS3)))
                self.ui.labelVmax2_9.setText(str(min(self.velzS3)))
                self.ui.labelVmax3_9.setText(str(promVz))

                self.ui.labelVmax1_10.setText(str(max(self.acezS3)))
                self.ui.labelVmax2_10.setText(str(min(self.acezS3)))
                self.ui.labelVmax3_10.setText(str(promAz))
            print len(self.posxS3)
            print len(self.posxS2)
            print len(self.posxS1)
            print len(self.posyS3)
            print max(self.posxS3)
            print max(self.posyS3)
            print max(self.poszS3)
            print min(self.posxS3)
            print min(self.posyS3)
            print min(self.poszS3)

    #*********************************************************#
    #Nombre Subrutina: datosObtenidos                         #
    #Parámetros de entrada: 0                                 #
    #Descripción: Crea un proceso en paralelo o multihilo para  #
    #la obtención de información brindada por el sensor.      #
    #*********************************************************#
    def datosObtenidos(self):
        hilo = MiHilo(name="Thread")
        hilo.start()
        self.sensor = hilo.rnombre()
        time.sleep(.1)
        if self.bps == 1:
            self.parametrosS = hilo.parametros()
            self.asignacion(self.parametrosS)
            time.sleep(.2)

    #*********************************************************#
    #Nombre Clase: MiHilo                                     #
    #Parámetros de entrada: 0                                 #
    #Descripción: Crea una nueva clase para el proceso en     #
    #paralelo o multihilo la cual se encargará de obtener los   #
    #datos del sensor en tiempo real                          #
    #*********************************************************#
class MiHilo(threading.Thread):
    arduino = serial.Serial('COM12', 9600, timeout= 0.2)
    idSensor = ""
    parametro = ""

    def rnombre(self):
```

```
            self.idSensor = self.arduino.readline()
            return self.idSensor

        #*************************************************************#
        #Nombre Subrutina: parametros                                 #
        #Parámetros de entrada: 0                                     #
        #Descripción: Genera los datos de posición, velocidad y       #
        #aceleración de un sensor en específico en tiempo real.       #
        #*************************************************************#
        def parametros(self):
            def filtrar(l,h,x):
                fs = 20.0
                lowcut = l
                highcut = h
                ntaps = 128

                taps_hamming = bandpass_firwin(ntaps, lowcut, highcut,
fs=fs)

                for i in range(0,63):
                    x.append(0)

                filtrado = lfilter(taps_hamming, 1.0, x)
                filtFin=[]
                for i in range(0,len(filtrado)-63):
                    filtFin.append(filtrado[i+63])
                return filtFin

            def bandpass_firwin(ntaps, lowcut, highcut, fs,
window='hamming'):
                nyq = 0.5 * fs
                taps = firwin(ntaps, [lowcut, highcut], nyq=nyq,
pass_zero=False,
                              window=window, scale=False)
                return taps


            open('Datos.txt','w')

            dataSerial = self.arduino.readline()
            cont =0
            while cont<5:
                while self.arduino.readline() == dataSerial:
                    a=0
                cont+=1
                dataSerial = self.arduino.readline()
            print("Midiendo")

            xs = []
            ys = []
```

```
            zs = []
            tMuestreo = 0.05
            aX =[]
            aY =[]
            aZ =[]
            aX1 =[]
            aY1 =[]
            aZ1 =[]
            aX2 =[]
            aY2 =[]
            aZ2 =[]
            aX3 =[]
            aY3 =[]
            aZ3 =[]
            vX =[]
            vY =[]
            vZ =[]
            posX = []
            posY = []
            posZ = []
            cont = 0
            aX.append(0)
            aY.append(0)
            aZ.append(0)
            vX.append(0)
            vY.append(0)
            vZ.append(0)
            posX.append(0)
            posY.append(0)
            posZ.append(0)
            accPrev1 = [0,0,0]
            accPrev2 = [0,0,0]
            vPrev1 = [0,0,0]
            vPrev2 = [0,0,0]
            loop = 1
            while loop==1:
                cont+=1
                dataSerial = self.arduino.readline()
                datos = dataSerial.split(',')
                sensor = datos[0]
                print sensor
                accPrevia =
[float(datos[1]),float(datos[2]),float(datos[3])]
                accFin =
[accPrevia[0]*9.81,accPrevia[1]*9.81,accPrevia[2]*9.81]
                aX.append(accFin[0])
                aY.append(accFin[1])
                aZ.append(accFin[2])
                accMag = (accFin[0]**2+accFin[1]**2+accFin[2]**2)**0.5
                if(abs(accMag)<0.5):
```

```
                    vX.append(0)
                    vY.append(0)
                    vZ.append(0)
                    posX.append(posX[cont-1])
                    posY.append(posY[cont-1])
                    posZ.append(posZ[cont-1])
                else:
                    vX.append(vX[cont-1] + (aX[cont])*tMuestreo)
                    vY.append(vY[cont-1] + (aY[cont])*tMuestreo)
                    vZ.append(vZ[cont-1] + (aZ[cont])*tMuestreo)
                    posX.append(posX[cont-1] + (vX[cont])*tMuestreo*5)
                    posY.append(posY[cont-1] + (vY[cont])*tMuestreo*5)
                    posZ.append(posZ[cont-1] + (vZ[cont])*tMuestreo*5)

datoPva=sensor+","+str(posX[cont])+","+str(posY[cont])+","+str(posZ[con
t])+","+str(vX[cont])+","+str(vY[cont])+","+str(vZ[cont])+","+str(aX[co
nt])+","+str(aY[cont])+","+str(aZ[cont])
                print datoPva
                datos = open('Datos.txt','a')
                datos.write(datoPva)
                datos.write('\n')
                return datoPva


  if __name__ == '__main__':
      app = QtGui.QApplication(sys.argv)
      myapp = MiVentana()
      myapp.show()
      sys.exit(app.exec_())
```