UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

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

TEMA:

"PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS: SOFTWARE DE ADQUISICIÓN Y TRATAMIENTO DE DATOS"

AUTOR: JOSSELINE VALERIA DORADO JÁCOME

DIRECTOR: CARLOS XAVIER ROSERO

IBARRA-ECUADOR
DICIEMBRE 2018

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

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto *Repositorio Digital Institucional*, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

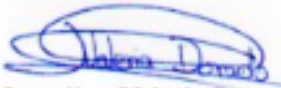
Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD:	0401739727		
APELLIDOS Y NOMBRES:	JOSSELINE VALERIA DORADO JÁCOME		
DIRECCIÓN:	La Victoria		
EMAIL:	jvdoradoj@utn.edu.ec - valeria.dorado154@gmail.com		
TELÉFONO FIJO:	062291221	**TELÉFONO MÓVIL:**	0996997483
DATOS DE LA OBRA			
TÍTULO:	”PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS:SOFTWARE DE ADQUISICIÓN Y TRATAMIENTO DE DATOS”		
AUTOR:	JOSSELINE VALERIA DORADO JÁCOME		
FECHA (AAAA-MM-DD):	2018-12-19		
SÓLO PARA TRABAJOS DE GRADO			
PROGRAMA:	PREGRADO		
TÍTULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA		
ASESOR/DIRECTOR:	CARLOS XAVIER ROSERO C.		

2. CONSTANCIAS

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

Ibarra, a los 19 días del mes de diciembre de 2018.



Josseline Valeria Dorado Jácome
C.I.: 0401739727

.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado "PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS: SOFTWARE DE ADQUISICIÓN Y TRATAMIENTO DE DATOS", presentado por el egresado JOSSELINE VALERIA DORADO JÁCOME, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, diciembre de 2018

Carlos Xavier Rosero
DIRECTOR DE TESIS

.

Agradecimiento

Agradezco a Dios y a la virgen María por bendecirme en cada paso de mi vida, por guiarme a lo largo de mi existencia y cuidar de mí estos años que estuve fuera de mi hogar.

A mis padres Vinicio y Maritza por ser mi apoyo y fortaleza en los momentos de dificultad, por siempre estar a mi lado brindandome todo su amor, inculcándome buenos valores y haciendo de mí una mejor persona. A esos dos seres que entregaron todo por verme salir adelante, que con mucha dedicación contribuyeron para que pudiera culminar mi carrera universitaria. Gracias, mil gracias pues las palabras no son suficientes para expresar mi gratitud, los amo.

A mi hermana Diana por estar siempre a mi lado, dándome su apoyo en cada decisión tomada, desde que llegaste todo ha sido más sencillo, sin ti esta meta hubiese sido mucho más difícil de conseguir, te quiero con todo mi corazón.

A Kevin por ser mi apoyo y mi mejor amigo, por cada palabra y gesto de amor, la distancia ha sido difícil pero a pesar de eso supimos crecer juntos y alcanzar lo que cada uno sueña, gracias por ser mi empuje y dejarme ser el tuyo, sabes llenarme el corazón y dar alegría a mi vida.

A mi familia por cada palabra de aliento y estar al pendiente de mí, siempre fueron fuente de motivación.

A mi director de tesis Carlos Xavier Rosero un gran profesional, por brindarme sus conocimientos, tiempo y experiencia, pero sobre todo su amistad. A los docentes de la carrera de ingeniería en mecatrónica, que fueron parte de mi formación profesional, gracias por brindarme todo su conocimiento.

A cada uno de mis amigos y compañeros quienes compartieron un pedacito de su vida a mi lado, haciendo mi estadía en la universidad más llevadera.

Valeria

Dedicatoria

Dedico este trabajo a mis padres, quienes son el pilar fundamental en mi vida, por su incansable amor y dedicación, por apoyarme en cada decisión. Han estado a mi lado en los momentos más difíciles y no me bastará la vida para agradecérselos. Ustedes son mi ejemplo y de quien aprendí los valores más importantes, mami gracias por enseñarme a tener responsabilidad, tenacidad, respeto y a luchar siempre por lo que quiero, papi gracias por inculcarme, la bondad, la nobleza y el amor, por demostrarme que a pesar de todos los obstáculos y momentos difíciles no podemos dejar de ver lo bonito de la vida, ayudar a los demás sin esperar nada a cambio y que si nos lo proponemos los días pueden ser mejores. Dios los bendiga y me permita siempre tenerlos a mi lado. Hoy puedo decirles que lo que una vez era nuestro sueño, ahora es una realidad y todo es más bonito porque lo logramos juntos.

Valeria

Resumen

En Ecuador el uso de la bicicleta es muy extendido debido a que permite una movilidad sostenible. Sin embargo, la seguridad del ciclista es un problema que crece a diario, hecho que se evidencia a través del alto índice de accidentes de tránsito con este medio de transporte. Este trabajo detalla el desarrollo de una plataforma para experimentación naturalística en bicicletas, específicamente el software de adquisición y tratamiento de datos. Para el diseño del trabajo mencionado, en la descripción general del sistema, se establecen requerimientos funcionales y no funcionales, acorde a las necesidades del sistema y de los usuarios. La información de las variables obtenidas del sistema embebido se guarda en una base de datos que luego se extrae y convierte en una hoja electrónica para mejorar su manipulación. Con el fin de facilitar el manejo de estas funciones, se procede a integrarlas en una interfaz gráfica, y son acompañadas de otras actividades que permiten visualizar la información de forma gráfica. Para la verificación del sistema se emplea la evaluación de usabilidad que valora la efectividad, eficiencia y satisfacción del usuario al interactuar con la interfaz gráfica. Los resultados obtenidos son favorables, existiendo una aceptación positiva de la interfaz, el tiempo empleado para realizar las diferentes tareas está dentro del rango estimado. Al final de la evaluación los datos arrojan que los usuarios ven a la interfaz, intuitiva, rápida, cómoda y fácil de usar.

Términos del Índice — Bicicletas, plataforma para experimentación naturalística, adquisición y tratamiento de datos e interfaz gráfica.

Abstract

In Ecuador, the use of bicycles is very widespread because it allows sustainable mobility. However, the safety of the cyclist is a problem that grows daily, a fact that is evidenced by the high index of traffic accidents with this means of transport. This paper details the development of a platform for naturalistic experimentation in bicycles, specifically the data acquisition and processing software. For the design of the aforementioned work, in the general description of the system, functional and non-functional requirements are established, according to the needs of the system and the users. The information of the variables obtained from the embedded system is stored in a database that then is extract and convert into an electronic spreadsheet to improve its manipulation. In order to facilitate the management of these functions, they have been integrated into a graphical interface, which are accompanied by other activities that allow visualization of the information in graphic form. For the verification of the system, the usability evaluation is used, which assesses the effectiveness, efficiency and user satisfaction when interacting with the graphic interface. The results obtained are favorable, there being a positive acceptance of the interface, the time used to perform the different tasks is within the estimated range. At the end of the evaluation the data show that users see the interface, intuitive, fast, convenient and easy to use.

Index Terms — Bicycles, platform for naturalistic experimentation, data acquisition and processing and graphic interface.

Índice general

Índice de figuras

Índice de cuadros

Lista de Programas

Capítulo 1

Introducción

Este trabajo de grado ha sido realizado en conjunto con el *Grupo de Investigación en Sistemas Inteligentes de la Universidad Técnica del Norte (GISI-UTN)*.

1.1. Problema

La bicicleta es una alternativa de movilidad menos contaminante, menos costosa, más eficaz y más amigable [1]. En Ecuador tres de cada diez hogares tienen al menos una bicicleta y su uso se incrementa cada año. Según INEN en [2] el 49,83 % de las personas usan bicicleta al menos una vez a la semana, el 34,09 % la usa a diario.

En nuestro entorno la seguridad del ciclista es una preocupación constante debido al alto índice de accidentes de tránsito a causa de impericia o imprudencia de los conductores de vehículos (51,9 %), seguida del irrespeto a las señales de tránsito (13,4 %) y en tercer lugar debido al exceso de velocidad (12,4 %) [2]. La bicicleta surge como alternativa viable tomando en cuenta los problemas que ocasiona un parque motor abultado [4].

Según cifras de la Comisión de Tránsito del Ecuador (CTE), en los tres primeros meses del año, los accidentes registrados con bicicletas y triciclos han aumentado en un 16.67 % con respecto a todo el 2016 [3]. Lo anterior se debe a que el ciclista irrespeta el sentido de la vía, realiza maniobras peligrosas al rebasar a los autos, invade las aceras y no respeta las señales de tránsito [5].

No existen maneras formales para el análisis del comportamiento del ciclista en las vías, que permita determinar las causas precisas de los accidentes de tránsito involucrando bicicletas. Además, hacen falta estudios previos y herramientas de monitoreo de las maniobras realizadas por el ciclista [6].

1.2. Objetivos

1.2.1. Objetivos general:

Desarrollar una aplicación para la adquisición y tratamiento de variables naturalísticas obtenidas desde bicicletas.

1.2.2. Objetivos específicos:

- Determinar los requerimientos y requisitos del sistema computacional.
- Sintetizar los algoritmos para tratamiento y almacenamiento de la información.
- Implementar el sistema computacional en software libre para someterlo a condiciones reales de funcionamiento.

1.3. Justificación

La información obtenida servirá como plataforma para realizar estudios de conducción de ciclistas. Podría también usarse para el análisis de la siniestralidad que involucra el uso inadecuado de la bicicleta a través de información capturada en el momento del accidente, de la misma forma que una caja negra o registrador de vuelo en los aviones.

El mismo sistema podría ser replicado para compartir inalámbricamente estados de conducción en una comunidad de ciclistas, permitiendo la exploración de las condiciones de las rutas. Además, con la información obtenida a través del sistema se podría incrementar la eficiencia en el manejo deportivo de bicicletas e incluso aumentar la seguridad del ciclista.

1.4. Alcance

En el presente proyecto se desarrollará un software que permita la adquisición de variables naturalísticas obtenidas desde un módulo embebido en una bicicleta. Esta información será almacenada en una base de datos. Además, se realizará una interfaz gráfica que permita al usuario interactuar con la información.

1.5. Estructura del documento

El siguiente trabajo consta de cinco capítulos y el apéndice. El primer capítulo presenta la introducción. El capítulo 2 hace referencia a la revisión bibliográfica donde se desarrolla el sustento teórico en base a fuentes bibliográficas y descripción de trabajos similares.

El capítulo 3 contiene la descripción general del sistema y abarca los requerimientos, características y diagramas de bloques.

Posteriormente en el capítulo 4 se expone el desarrollo del software de adquisición, el tratamiento e interacción con la información, se presenta un diagrama de bloques para una mejor compresión.

La implementación de la adquisición, de la base de datos y de la interfaz gráfica, son el centro del capítulo 5. En este apartado también, se somete al software desarrollado a pruebas bajo condiciones reales de funcionamiento para validar su desempeño.

El Capítulo 6 presenta las conclusiones y recomendaciones, además se propone posibles líneas de trabajo futuro.

Finalmente se encuentra el Apéndice que muestra el código del trabajo.

Capítulo 2

Revisión Literaria

2.1. Bases de Datos

Una base de datos es un almacenamiento de información organizado, estructurado y categorizado, que comparten entre sí un vínculo o relación en común al momento de acceder a ella. Las bases permiten integrar los datos y tener coherencia en los mismos, proporcionando un corto tiempo de respuesta para su acceso [7]. Dependiendo de las circunstancias, se puede usar un formato de base de datos compleja o un archivo de texto sencillo [8].

Los sistemas gestores de bases de datos DBMS (Database Management Systems), son software que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Cada gestor dispone de distintas interfaces con sus propias características [9].

Según la relación entre los datos se diferencian dos tipos de gestores de bases de datos que son: relacionales y no relacionales.

2.1.1. Bases de datos relacionales

Una base de datos relacional consta de tablas que poseen información que las vincula, relacionándolas unas con otras. Esto permite almacenar datos eficientemente y sin redundancias. Poseen una estructura bidimensional donde constan varias filas y columnas llamadas registros y campos respectivamente [8].

En 1970 Edgar Frank Codd realiza la postulación de sus fundamentos, los que no tardaron en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de relaciones. En este tipo de modelo, la forma para almacenar los datos no tiene relevancia. Los datos pueden ser recuperados o almacenados mediante consultas, ofreciendo flexibilidad para administrar la información [9].

El lenguaje más utilizado para construir bases de datos relacionales es SQL (Structured Query Language o Lenguaje Estructurado de Consultas), declarado estándar internacional de comunicación dentro de las bases de datos [9]. Los gestores de bases de datos relacionales más destacados son:

- MySQL: Es una base de datos Open Source desarrollada por Oracle. Es un amplio subconjunto del lenguaje SQL, permite seleccionar distintos tipos de almacenamiento, replicación de los datos, búsqueda e indexación de los campos [9]. Poco a poco fue dotada de componentes esenciales del algebra relacional. En la actualidad, es una verdadera base de datos relacional que ofrece un buen rendimiento [7].

- PostgreSQL: Es un evolucionado sistema de administración de bases de datos objeto-relacionales de código libre, entre sus características se encuentran: está basado en lenguaje C, posee interoperabilidad con SQL, de almacenamiento confiable y muy robusto, de manipulación potente, flexible y eficiente, además ofrece muchas funcionalidades para responder a necesidades muy avanzadas de manera eficaz [10].

- SQLite: Es un gestor de base de datos relacional escrito en C de código abierto, se diferencia de MySQL y PostgreSQL, ya que no funciona según el modelo cliente-servidor, sino que está embebida. Tiene una rápida configuración y sólo se necesita incluir una librería en la aplicación. Es ideal para pequeñas aplicaciones, pero el rendimiento puede disminuir si el número de datos es excesivamente elevado [7].

2.1.2. Bases de datos no relacionales

Este tipo de bases de datos se diferencian de las bases de datos relacionales porque no usan SQL como lenguaje principal para consultas, llamándolas NoSQL. Las bases de datos NoSQL almacenan la información sin cumplir el esquema entidad–relación. Se las usa principalmente

donde las bases de datos relacionales tienen problemas de escalabilidad y rendimiento, principalmente donde existen miles de usuarios concurrentes con millones de consultas [11].

NoSQL hace referencia a las bases de datos, en las que priman el manejo de grandes conjuntos de datos y la velocidad. Estas bases de datos permiten tener escalabilidad (ampliar rápidamente) y desligar el hardware del tipo de datos haciéndolo eficiente. Las bases de datos NoSQL son utilizadas cuando el volumen de datos es demasiado grande, mientras que SQL se utiliza para realizar análisis más detallados [9].

Se clasifican según su forma de almacenar los datos pues no lo hacen en forma de tabla, usan otros formatos como clave-valor, BigTable, bases de datos documentales y orientadas a grafos [11]. Entre las bases de datos no relacionales destacan las siguientes:

- Redis: Es una base de datos clave-valor, escrita en C, que provee un excelente rendimiento como base de datos en memoria. Útil para almacenar datos que no poseen relaciones complejas entre sí, pero que están indexados por una clave. Sus valores pueden ser del tipo: cadena de caracteres, listas, diccionarios y conjuntos [7]. Su uso es recomendable cuando los datos cambian rápidamente y tienen un tamaño predecible, como el análisis en tiempo real [9].

- MongoDB: Es una base de datos orientada a gestionar y almacenar documentos, escrita en C++ de código abierto, se trata de un tipo de base clave-valor donde el valor es un documento. Así, la eficacia de esta tecnología radica en dar una buena estructura al valor y tener un uso correcto de claves [7]. Su uso es útil si se hacen consultas dinámicas [9].

- Cassandra: Apache Cassandra es Open Source implementado en Java. Desarrollado por Facebook que en 2010 fue donada a Apache para proyectos de primer nivel como software libre. Es una base de datos NoSQL importante y muy utilizada a nivel mundial, haciendo su uso, tecnologías relevantes como Netflix, eBay, Spotify, Twitter, Urban Airship, Constant Contact, entre otros. Cassandra es capaz de trabajar con varios terabytes de datos. La información es almacenada en columnas en modelo clave-valor [12].

2.2. Comunicación

2.2.1. Clases de redes

2.2.1.1. Redes personales (PAN)

El alcance de esta red inalámbrica es más restringido, esta direccionada al uso de un solo usuario, posee un alcance máximo de diez metros. Comúnmente se la usa para comunicar un celular con auriculares o una computadora con el mouse o teclado [13]. Entre los protocolos más conocidos de redes PAN son el bluetooth, infrarroja y ZigBee [14].

2.2.1.2. Redes de Área Local (LAN)

Son redes privadas que se usan para conectar computadores personales en un hogar o estaciones de trabajo en una oficina y fábricas. Las redes LAN operan a una velocidad de 10 a 1000 Mbps y tiene un alcance de hasta cien metros, estos límites varían dependiendo del medio que se use para su transmisión. El medio para su conexión puede ser cable coaxial un cable de dos hilos, fibra óptica o cable UTP. Estas redes utilizan protocolos para intercambiar la información, evitando una colisión de los datos; entre estos protocolos están Ethernet o Token Ring [15].

2.2.1.3. Redes de Área Metropolitana (MAN)

Se extienden a lo largo de una ciudad, a ellas se puede conectar un cierto número de redes LAN. Por ejemplo, haciendo uso de una red MAN se puede conectar todas las LAN dispersas de una oficina. Tiene una cobertura de decenas de kilómetros [15].

2.2.1.4. Redes de Área Amplia (WAN)

Geográficamente se extiende en un país o en todo un continente, utiliza Hosts conectados por una subred de comunicaciones. Conecta redes LAN y MAN [15].

A continuación en la Tabla 2.1 se realiza una comparación entre los protocolos más utilizados, para la construccion de esta tabla se compila información de las referencias [13], [14], [15], [18] y [21].

Tabla 2.1: Comparación de Protocolos [13], [14], [15], [18] y [21]

Características	Bluetooth	Zigbee	Ethernet	Wi-Fi
Especificación IEEE	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.3	IEEE 802.11
Alcance	10 m	100 m	100 m	100 m
Velocidad	10 Mbps	250 Kbps	100 Mbps	11 Mbps
Frecuencia	2.4 GHz	2.4 GHz	100 MHz	2.4 GHz
Conexión	Inalámbrica	Inalámbrica	100 base T categoria 5	Inalámbrica

2.2.2. Ethernet

En la actualidad Ethernet es una tecnología indispensable al llevar a cabo actividades cotidianas, pues es económico y es un protocolo ampliamente difundida con aceptación mundial. Ethernet es el protocolo más popular de las redes LAN con gran futuro a medio plazo. Este protocolo crece rápidamente, en los últimos años, se establece la norma Fast Ethernet que aumenta la velocidad de comunicación a 100 Mbps tan solo con pocos cambios en el cableado. Entre las funciones que realiza Ethernet está el empaquetado y desempaquetado de datagramas, el control del enlace, la codificación y decodificación de los datos y el acceso al canal [16].

Ethernet posee una clara ventaja por encima del WiFi y es el aprovechamiento del ancho de banda, ya que la conexión directa con el router proporciona una alta velocidad. Wifi no lo aprovecha completamente aún estando el ordenador junto al router debido a que existen pérdidas y si se lo aleja más, la calidad del ancho de banda desciende claramente. La conexión Ethernet brinda más seguridad en la transferencia de información; la misma va del ordenador al router y luego al destino, mientras que en WiFi, aunque la conexión esté cifrada no se puede saber si alguien más está intentando acceder a los datos. La conexión cableada Ethernet tiene un alcance máximo de 100 metros, menor que la conexión WiFi que llega a los 300 metros pero en ésta, se tiene la pérdida de calidad de señal mientras que en Ethernet son solo 100 metros pero en toda la distancia existe una calidad similar. El cableado que incomoda a muchas personas se puede solucionar realizando una buena distribución eléctrica [17].

2.2.3. Arquitectura de Redes

Existen dos importantes arquitecturas de redes: el modelo de referencia OSI y el modelo de referencia TCP/IP. El protocolo OSI ya no es muy utilizado pero el modelo sí, ya que sirve de referencia para otros modelos por las características de cada uno de sus niveles. Lo contra-

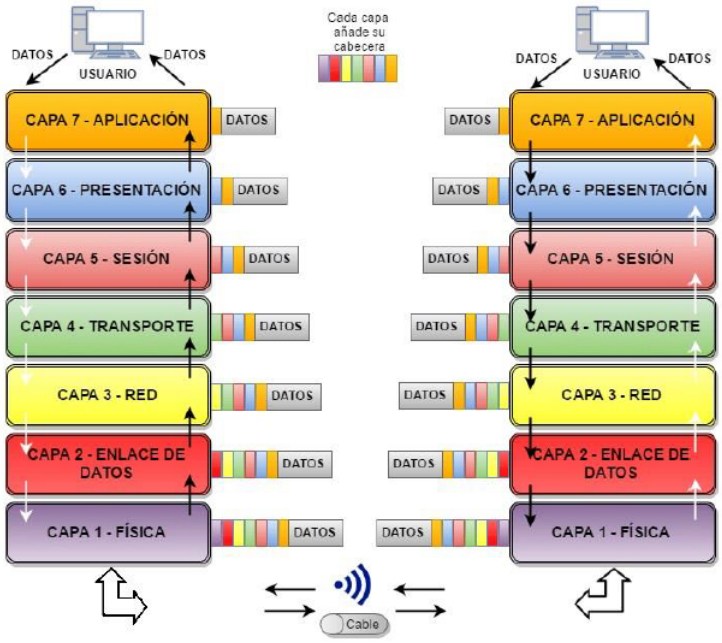


Figura 2.1: Capas del modelo de referencia OSI [19].

rio ocurre con el modelo TCP/IP puesto que no es muy usado pero su protocolo se lo utiliza frecuentemente en la actualidad [15].

2.2.3.1. Modelo OSI

El modelo OSI fue desarrollado a partir de una propuesta presentada por la Organización Internacional de Normas (ISO), para establecer una estandarización internacional de todos los protocolos existentes en las diferentes capas, esto permitió comunicar fácilmente computadores de diferentes fabricantes [18]. En 1983 el modelo fue revisado, se lo llamó Modelo de Referencia OSI (Interconexión de Sistemas Abiertos) y se lo ubicó como un estándar internacional. El modelo propuesto divide en siete niveles todas las tareas que realiza cuando una computadora se comunica con otra, esto con el fin de que si en alguna capa existe problemas las demás capas no se vean afectadas. Las primeras cuatro capas del modelo son para comunicación y las tres siguientes para proceso [15], como se puede apreciar en la Fig. 2.1. A continuación se describen las siete capas del modelo OSI en base a [15].

- Capa física: A este nivel pertenecen los medios eléctricos y mecánicos por donde se va establecer y mantener la conexión física entre la computadora y la red, los medios pueden

ser cables, conectores y tipos de señales.

- Capa de enlace de datos: Aquí se establece y mantiene el flujo de datos que se transmiten entre los usuarios, manteniendo una distribución ordenada de tramas; si se producen errores esta capa los controla y corrige.
- Capa de red: Este nivel decide por dónde se transmiten los datos dentro de una red, se incluye la administración y gestión de los mismos y regula el tráfico de la red.
- Capa de transporte: Se encarga de la transferencia segura de la información, aún tomando en cuenta los problemas que puedan ocurrir en las otras capas. Los protocolos más usados en esta capa son: el Protocolo TCP y UDP, el primero está orientado a la conexión y el segundo no necesita conexión.
- Capa de sesión: Ejecuta las funciones (usuarios, contraseñas y administración del sistema) que permitan a dos usuarios comunicarse a través de la red, manteniendo y controlando el enlace.
- Capa de presentación: En esta capa se traduce toda la información del formato de varios tipos de máquinas a un formato que el usuario pueda comprender.
- Capa de aplicación: Es el programa entre el usuario y el sistema operativo que se encarga del intercambio de información, el mismo se conecta con esta capa y realiza las funciones indicadas.

2.2.3.2. Modelo TCP/IP

ARPANET fue desarrollada por el departamento de defensa de Estados Unidos, era usada como una red de investigación que luego pasó a unir varias universidades e instalaciones del estado usando líneas telefónicas rentadas. Luego, se desarrollaron las redes satelitales y de radio con lo que los protocolos de ese momento tuvieron problemas. Desde ese momento se buscó una arquitectura que permitiera interactuar varios tipos de redes sin dificultad. Así, surgió una arquitectura llamada Modelo de Referencia TCP/IP [18].

Su nombre proviene de sus dos primeros protocolos, fue concebido por primera vez por Cerf y Kahn en 1974, posteriormente a cargo de Braden en 1989 se redefinió como estándar en el

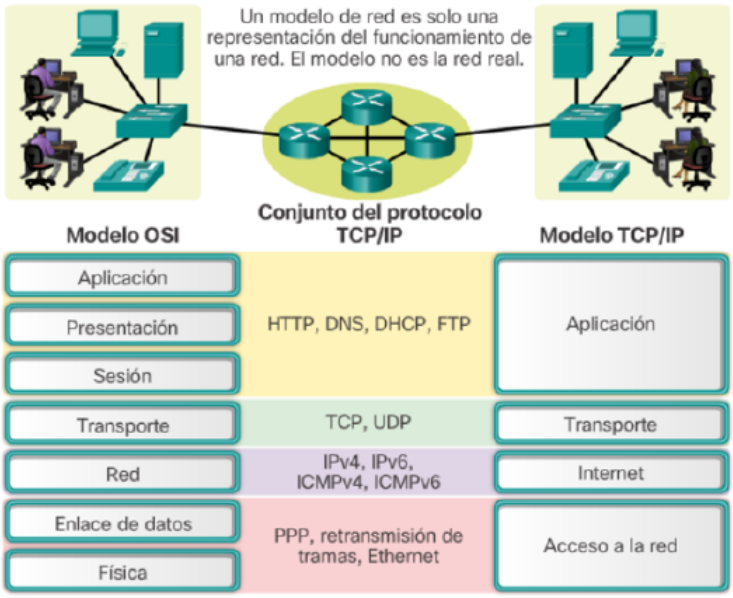


Figura 2.2: Comparación modelo OSI con modelo TCP/IP [20].

mundo del Internet [18].

El modelo TCP/IP presenta cuatro capas distribuidas como se muestra en la Fig. 2.2. A continuación se describen las capas que conforman el modelo TCP/IP en base a [18].

- Capa de enlace: En esta capa se permite el acceso a la red por medio de una conexión por hosts, así se vuelve el medio físico para la transmisión. Existen dos categorías para realizar un enlace de red: conexión punto a punto y canales de difusión.
- Capa de interred (Internet): En esta capa se define un protocolo llamado IP (Protocolo de Internet). Todos los dispositivos poseen una dirección IP propia la cual los identifica, mediante estas y con ayuda de los switches y routers se decide la red por la cual serán enviados los datos.
- Capa de transporte: Se encarga de la transferencia de los datos, en esta capa se definen dos protocolos.
    - UDP: Es el Protocolo de Datagrama de Usuario, no usa conexión y no es muy confiable, se usa en aplicaciones donde es más importante la entrega rápida que una entrega precisa; pues solo se encarga de enviar paquetes y deja que las mismas

aplicaciones construyan sus propios protocolos en las capas superiores según sea necesario.

- TCP: Protocolo de Control de la Transmisión, este protocolo permite la transmisión sin errores de un flujo de bytes de una máquina a otra, mediante una conexión segura. En este protocolo el flujo de bytes es segmentado y pasa de uno en uno a la capa de interred, cuando llega al destino, esta misma capa en la máquina receptora vuelve a unir la información recibida. Además, controla el flujo de tal manera que, si el emisor es rápido y el receptor es lento no ocurra una saturación que produzca errores.

- Capa de aplicación: En esta capa se encuentra la interfaz del usuario, utiliza principalmente el protocolo FTP para la transferencia de archivos, SMTP para correo electrónico, entre otros protocolos. Aquí además se validan algunos requisitos para iniciar la sesión o conexión.

2.2.4. Protocolo SSH

SSH es un intérprete de órdenes seguro (Secure SHell). Este protocolo se usa para acceder a máquinas remotas por medio de una red, permite manipular por completo la computadora remota y lo hace mediante un intérprete de comandos. Este protocolo proporciona la posibilidad de gestionar claves RSA, donde no es necesario el ingreso de las mismas al conectar los dispositivos, además permite copiar datos de forma segura [21].

2.2.5. Socket

Los segmentos TCP contienen el número de puerto de la fuente y el destino, sirven para identificar el dispositivo emisor y el receptor. Estos dos valores, seguidos de las direcciones IP de la fuente y el destino, identifican cada conexión de forma única. A la combinación del número de puerto con la dirección IP se denomina socket. Así, el cuarteto compuesto por el número de puerto y la dirección IP del Cliente, el número de puerto y la dirección IP del Servidor; forman la pareja de sockets para identificar la conexión TCP entre dos computadores permitiendo interactuar entre ellos y realizar transferencia de datos [18].

2.3. Interfaz Gráfica

El ser humano interactúa diariamente con interfaces gráficas, pues las encuentra al usar el celular, computadoras, cajeros automáticos, etc. En la actualidad las aplicaciones forman parte importante en la vida diaria, más usuarios con o sin experiencia interactúan con interfaces y es pensando en usuarios inexpertos que se ha desarrollado interés por el diseño de interfaces [22]. Una interfaz debe ser amigable y de fácil uso, en tanto más amigable será más fácil de usar para una mayor proporción de usuarios. Una buena interfaz es interactiva ya que permite al usuario dialogar con ella, proporcionándole facilidad de comunicación [23].

Un término muy utilizado en el diseño de interfaces gráficas es el diseño de interacción que se dedica a crear productos interactivos, lo que se refiere a desarrollar entornos que sean fáciles de comprender, eficaces de usar y sobre todo que proporcionen al usuario una experiencia agradable. Este tipo de diseño además busca apoyar a las personas en su vida laboral y cotidiana. Para el diseño de productos interactivos se deben identificar las debilidades y fortalezas del mismo, además de tomar en cuenta quien y donde se usarán. El objetivo principal de este tipo de diseño es, que el producto final sea cien porciento utilizable por los usuarios; esto se puede lograr mediante el diálogo con los mismos, observando su desempeño y evaluando su interacción con el trabajo.

En 1997, Winograd [24] describió al diseño interactivo como *el diseño de espacios para la comunicación y la interacción humana*. El diseño de interacción debe estar siempre relacionado con la ingeniería de software, la compara con la relación de un arquitecto, con un ingeniero civil al momento de construir una vivienda, el primero se encarga de velar la interacción que ocurre entre las personas y la casa. Por ejemplo: la combinación de los espacios; mientras que el ingeniero civil se encarga de los aspectos estructurales y construcción. Así, ambas disciplinas son necesarias. Para que el diseño de interacción sea exitoso se necesita del apoyo de muchas disciplinas, para comprender como interactúan, se comunican y reaccionan los usuarios, para esto, se toman en cuenta disciplinas como la psicología y sociología. Cuando ya se ha comprendido la interacción, es necesario diseñar productos estéticamente agradables; aquí intervienen profesionales como diseñadores gráficos, artistas, fotógrafos y diseñadores de productos [24].

Para identificar las necesidades de los usuarios se plantean dos objetivos: El primero trata

de comprender a los usuarios y su desempeño en el trabajo, de manera que pueda ayudarlos en sus labores. El segundo objetivo es construir y desarrollar la solución a partir de las necesidades del usuario. En la práctica, las necesidades evolucionan y se van produciendo a medida que los usuarios interactúan con el diseño [24].

2.3.1. Proceso de diseño de interacción

Para llevar a cabo el proceso de diseño de interacción se deben llevar a cabo cuatro importantes actividades, en base a [24]:

1. Identificar y establecer los requisitos.
2. Desarrollar varios diseños que cumplan con los requisitos.
3. Crear diseños interactivos que permitan ser comunicativos con el usuario.
4. Evaluar todo el proceso de diseño.

Las cuatro actividades deben interactuar entre sí, esto con el fin de proporcionar información sobre cambios en los requisitos de diseño. Además, existen otras tres características claves en el proceso, como muestra [24]:

1. Los usuarios participan directamente en el proceso de desarrollo.
2. La usabilidad y experiencia del usuario deben ser identificados y acordados al inicio del proceso.
3. La iteración entre las actividades es fundamental.

La usabilidad asegura que los productos del diseño interactivo sean fáciles de aprender y usar, además de que el usuario disfrute usándolos en cualquier lugar que se encuentre, la usabilidad se divide en varios subobjetivos: efectivo, eficiente, seguro al usar, fácil de aprender y recordar [24].

2.3.2. Características de una buena interfaz gráfica

El mundo digital ofrece un sin número de recursos multimedia, pero recargar una interfaz de ellas, la satura y se vuelve poco atractiva, si se los usa debe ser en beneficio del usuario, como para reforzar las ideas que se tratan de transmitir, haciendo a la interfaz eficaz. Una interfaz como se demuestra claramente en [25] debe ser:

- Sencilla: Los elementos que tenga la interfaz son para guiar y ayudar al usuario, no para confundir, hay que evitar la saturación y cosas innecesarias
- Clara: La información debe ser localizable fácilmente, tener lógica y estar de manera ordenada.
- Predecible: Todas las acciones que se realicen deben tener un resultado igual, por ejemplo: si das un doble click en un título lleve a su índice y esto debe repetirse en todos los documentos.
- Flexible: La interfaz debe ser compatible con la mayoría de navegadores y plataformas. Pensar en opciones que siempre nos devuelvan a la página principal, proporcionar varios formatos del mismo documento, además permitir una actualización constante de la misma.
- Consistente: Se debe lograr una similitud entre todas las secciones o capítulos.
- Intuitiva: Debe permitir al usuario sentirse seguro con su uso, donde no tenga que adivinar como ejecutar ciertas acciones.
- Coherente: Para aportar al contenido de la publicación se debe adoptar textos, gráficos, colores y demás elementos que vayan acorde a lo que se está intentando transmitir.

2.3.3. Problemas frecuentes en interfaces gráficas

Muchas veces al usar una computadora los usuarios pueden llegar a sentirse muy satisfechos y cómodos o sumamente frustrados, en la referencia [24] se explica cómo evitar que una interfaz resulte poco atractiva para el usuario:

- Aplicaciones que no funcionan correctamente o se bloquean.

- Sistemas que no hacen lo que el usuario desea o no cumplen sus expectativas.
- Programas que no proporcionan suficiente información sobre su uso.
- Interfaces con apariencia ruidosa o engañosa, con mensajes vagos y que sean tediosas al realizar tareas.

Capítulo 3

Descripción general del sistema

3.1. Requerimientos del sistema

Los requerimientos son las necesidades de los usuarios; definen las funciones que el sistema será capaz de realizar. Los requerimientos o requisitos describen los servicios que ofrece el sistema y las restricciones asociadas a su funcionamiento. Los requerimientos son funcionales y no funcionales [26].

3.1.1. Requerimientos funcionales

Permiten conocer la naturaleza del funcionamiento del sistema. Describen cualquier actividad que el sistema realiza, el comportamiento o función particular cuando se cumplen ciertas condiciones [27]. Con respecto a la Plataforma para Experimentación Naturalística en Bicicletas: Software de Adquisición y Tratamiento de Datos se presentan los siguientes requerimientos funcionales:

- El módulo posee un subsistema para manipulación de hora y fecha, el cual actualiza la hora y fecha del micro computador remoto, para realizarlo necesita del subsistema de transferencia remota de datos.
- El software cuenta con un subsistema para el manejo de hojas electrónicas y bases de datos. La información obtenida desde los sensores por el módulo embebido son almacenados en una base de datos.

- La aplicación en el computador extrae la base desde el micro computador remoto mediante el subsistema de transferencia remota de datos.
- El subsistema para el manejo de hojas electrónicas y bases de datos se encarga de convertir la base en una hoja electrónica.
- Por último, el software posee un subsistema para la interfaz gráfica de usuario, la que permite presentar las variables almacenadas en la base de datos de forma gráfica, permitiendo al usuario interactuar con los mismos. Esta se encuentra ligada con la consola de comandos de Linux e informa al usuario todas las actividades o acciones que la interfaz realiza.

3.1.2. Requerimientos no funcionales

Son características o cualidades que los usuarios requieren como parte del comportamiento y calidad del software. Las características proporcionan información del sistema. Este tipo de requerimientos no son parte fundamental del sistema, pero sí son necesarios para hacerlo funcionar de la manera deseada [28]. Para seleccionar los atributos que especifican las características útiles sobre la calidad del sistema, se toma en cuenta los criterios que se encuentran en [29], que muestra el estandar para la evaluación de calidad expuesto por la ISO/IEC 9126 y las siete categorias de atributos de calidad definidos por Bass, Clements y Kazman. Entre los requerimientos no funcionales de la Plataforma para Experimentación Naturalística en Bicicletas: Software de Adquisición y Tratamiento de Datos se proponen:

Eficiencia:

- El sistema es capaz de guardar hasta 50 secuencias de datos por segundo, con una base de datos de hasta 4 gigabytes.
- La comunicación entre la aplicación del computador y la del micro computador remoto responde al usuario en menos de 5 segundos.

Disponibilidad:

- La base de datos en el micro computador puede ser usada aproximadamente 2 horas hasta que la batería se descargue por completo y la interfaz gráfica puede ser usada las 24 horas, pues ésta no depende de la misma fuente de alimentación.
- El sistema se conecta a una red de infraestructura para realizar la comunicación entre la computadora y el micro computador, si se tiene buena disponibilidad de red el sistema estará disponible para funcionamiento en un 99,99

Seguridad lógica y de datos:

- Las IP y números de puerto para la comunicación del sistema son fijos y pueden solo ser cambiados directamente en el código de programación del software.
- Todos los archivos de la base de datos deben transferirse luego de cada adquisición para mayor seguridad. Los respaldos serán almacenados en una localidad segura ubicada en la computadora donde se encuentra la interfaz gráfica.

Usabilidad:

- El aprendizaje del sistema por el usuario debe ser menor a un tiempo de 20 minutos ya que la interfaz es intuitiva.
- El software posee interfaces gráficas interactivas.
- La aplicación es de sencillo uso.

Software: El sistema está basado en software libre, debido a que la mayoría de las herramientas están disponibles sin costo. Además, las diferentes aplicaciones pueden ser copiadas y compartidas sin ninguna restricción de uso.

Hardware: El sistema de adquisición y tratamiento de datos para experimentación naturalística en bicicletas necesita el micro computador de adquisición de variables y una computadora para la interfaz gráfica.

Soporte: El software tiene una sencilla instalación, es gratuito y tiene facilidad de mantenimiento. Es claro con la finalidad de permitir en un futuro desarrollar nuevos requerimientos, aislar y corregir los errores y atender las demandas del entorno cambiante.

Modificable: Proporciona facilidad para hacer cambios en los requerimientos.

Escalabilidad: El sistema puede tomar a consideración un crecimiento futuro, solo se necesitará una pequeña modificación en los campos de la base de datos y en la presentación de la interfaz gráfica.

Extensibilidad: El sistema toma en cuenta un crecimiento en el futuro, realizando los códigos de programación sencillos y comprensibles para facilitar que el software crezca en funcionalidades.

3.2. Selección de Variables Naturalísticas obtenidas desde bicicletas

Para la selección de las variables que se obtendrán desde las bicicletas, se realiza un análisis de diferentes estudios que relacionan la adquisición de datos para definir estados de conducción de ciclistas, además de trabajos de instrumentación de bicicletas, permitiendo saber cuales son las variables más importantes que debe conocer el ciclista luego del recorrido. Así, se decide basar este trabajo en la selección de variables realizada por [30]

Este estudio tiene como propósito investigar los factores que afectan el comportamiento de los ciclistas en las intersecciones señalizadas, para ello se realiza un experimento de ciclismo naturalista; se desarrolla un sistema de adquisición de datos que incluye sensores como [30]:

- Velocidad.
- Acelerómetro.
- Giroscopio.
- GPS.

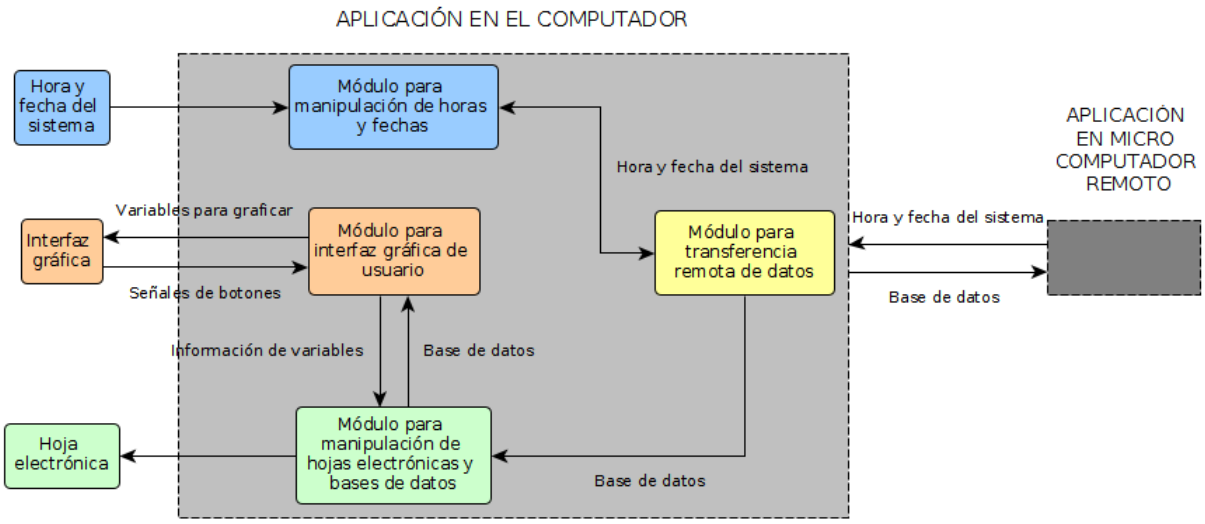


Figura 3.1: Diagrama de bloques del sistema.

Finalmante se adopta varios algoritmos de aprendizaje automático para desarrollar modelos de predicción de acciones en intersecciones [30].

3.3. Diagrama de bloques

El diagrama de bloques que describe el sistema y subsistemas se muestra en la Fig. 3.1. Como se puede observar, la aplicación cuenta con cuatro subsistemas que se detallan en la sección 4.1.

Capítulo 4

Software de adquisición, tratamiento e interacción con la información

4.1. Subsistemas del software de adquisición y tratamiento de datos

El software de adquisición y tratamiento de datos usa Ubuntu del sistema operativo Linux y el lenguaje de programación Python. Ubuntu es un software libre, gratuito, robusto y más estable que otros sistemas [31]. Por otro lado Python es un lenguaje interpretado de alto nivel, es multiplataforma , de tipado dinámico y multiparadigma; estas caracteristicas hacen que escribir en este lenguaje tenga código más legible y estandarizado [32]. Para instalar la aplicación solo basta con descargar el software de internet.

La aplicación en el computador está compuesta por cuatro principales subsistemas, los que interactúan entre sí para cumplir con las diversas funcionalidades establecidas y son:

- Subsistema para manipulación de horas y fechas
- Subsistema para manipulación de hojas electrónicas y bases de datos
- Subsistema para interfaz gráfica de usuario
- Subsistema para transferencia remota de datos

4.1.1. Subsistema para manipulación de horas y fechas

Este subsistema se encarga de actualizar la hora y fecha del sistema en la aplicación del micro computador remoto, estos datos los obtiene del módulo *time* [33] de python, el cual contiene varias funciones vinculadas con el tiempo, en el sistema se extrae la hora y la fecha actual con la aplicación en el computador para luego transferirla al micro computador remoto, aquí, con la ayuda del módulo *os* [34] y *sys* [35] se logra manipular la funcionalidad del sistema operativo reseteando la hora y estableciéndola en la hora enviada.

4.1.2. Subsistema para manipulación de hojas electrónicas y bases de datos

Para definir el módulo de base de datos en python a usar en la aplicación, se analiza la mejor alternativa según la utilidad y funcionalidad que tiene y se escoge a las bases de datos *SQLite*, por su posibilidad de estar embebida y rápida configuración. Para hacer uso de la base de datos se importa el módulo *sqlite3* [36]. En la aplicación, en el micro computador remoto mediante una función del módulo mencionado, se crea automáticamente un archivo db, posteriormente se origina una tabla con los campos necesarios para todas las variables leídas por los sensores. Cuando el sistema se encuentra funcionando, cada lectura de datos es insertada como registro en la base de datos.

Este subsistema también se encarga de transformar el archivo de base de datos a una hoja electrónica, para esto, se decide tomar la opción de hacerlo en un archivo *csv* por su forma sencilla de presentar la información y que es compatible con otros lenguajes de programación. Este tipo de archivo coloca los datos en forma de tablas donde las columnas están separadas por una coma, mientras que las filas lo están por saltos de línea. Para convertir a este formato la base de datos se importa los módulos sqlite3 y *csv* [37], donde la base de datos es abierta y leída, posteriormente se crea un archivo csv vacío. Todo el contenido del archivo db es copiado en el archivo csv colocando los campos y los registros con las características mencionadas.

4.1.3. Subsistema para transferencia remota de datos

Este subsistema se encarga de transferir dos tipos de datos: el primero es la fecha y la hora desde el computador hasta el micro computador y el segundo es copiar la base de datos del

micro computador y transferirla al computador.

Para el primer parámetro, el sistema realiza una *conexión TCP/IP* debido a que proporciona una conexión segura; para usar esta conexión en python se importa el módulo *socket* [38]. Para hacer uso del protocolo se necesita un socket servidor y un socket cliente, esta comunicación se usa para transmitir la hora y fecha desde el computador hasta el micro computador. El servidor es el micro computador y el cliente es la computadora. El sistema controlará la conexión TCP/IP y lo permitirá solamente a usuarios autorizados, mediante la IP y el número de puerto del emisor y el receptor, una vez establecida la conexión el servidor envía la hora actual en forma de un mensaje y lo transmite al cliente. Una vez reseteada la hora la conexión TCP/IP se cierra.

Para la segunda parte se realiza una conexión por *protocolo SSH* para acceder de forma remota al micro computador desde la aplicación en el computador, esta comunicación por medio de un intérprete de comandos puede controlar al computador remoto, en este caso se lo usa para transferir la base de datos. Para realizar la comunicación se importa el módulo *paramiko* [39], luego se establece la conexión SSH al servidor, validando los parámetros del micro computador como usuario, contraseña, IP y el puerto, la conexión se mantiene abierta y se usa el módulo *warnings* [40] para emitir mensajes de alerta en situaciones que se requiere advertir al usuario como el cierre repentino de la comunicación. Luego interviene el módulo *md5* [41] que compara la base de datos en el micro computador y la base de datos en el computador, si los datos son iguales el archivo, no es copiado y permanece el mismo, si los datos son diferentes; el contenido es modificado, aquí paramiko transfiere el archivo db con la ayuda del módulo *os* para poder acceder a la ubicación remota del archivo en el microcomputador copiarlo y pegarlo en la ubicación indicada en el computador local. Posteriormente paramiko cierra la comunicación.

4.1.4. Subsistema para interfaz gráfica de usuario

Este subsistema es el encargado de realizar la directa interacción del software con el usuario. La interfaz gráfica esta desarrolla en *Tkinter* la cual es un toolkit para el desarrollo de interfaces gráficas (GUIs) que acompaña a la distribución de Python. Para el diseño y distribución de estas características se hace uso del módulo de python *Tkinter* [42], que permite tener las ventanas necesarias para la interfaz, la distribución de botones y textos.

En la interfaz gráfica se encuentra distribuidos varios botones, cada uno de ellos realiza una función sobre los subsistemas anteriormente nombrados, así el usuario no necesita llevar a cabo varias acciones, pues solo necesita presionar el botón de la actividad que desea ejecutar.

Existe también el botón *Verifique Trajectory* el cual grafica el recorrido efectuado por el ciclista mediante un mapa e indica la distancia recorrida en kilómetros. Para obtener el mapa de localización de la ruta efectuada se necesita la latitud y longitud de la región recorrida, además de los puntos que describen la trayectoria, este mapa permite interactuar con la información, pues al estar conectado a internet se actualiza cada instante, dejando al usuario explorar de cerca toda la ruta; para esto se usa el módulo *folium* [43], de la librería de Python que facilita el trazo de mapas de folletos, el cual crea un archivo html en la carpeta personal, este archivo puede ser abierto por el usuario desde la carpeta pero este proceso no es el más conveniente para el usuario, por esto se usa el módulo *webbrowser* [44] que accede a páginas web, siendo necesario solo un clic en el botón Verifique Trajectory y la ventana del navegador con el mapa emerge por sí sola mostrando la trayectoria, además este botón brinda la distancia recorrida en la consola mediante la fórmula matemática de *Haversine* [45] que utiliza la longitud y latitud del punto de partida, proporcionando un valor muy aproximado de la distancia entre dos puntos, para esto se utiliza la siguiente fórmula:

$$D = 2 * R * arcsen(\sqrt{sin^2(\frac{\Delta lat}{2}) + cos(lat1) * cos(lat2) * sin^2(\frac{\Delta lon}{2})}) \tag{4.1}$$

Donde D es la distancia recorrida, R es el radio de la tierra y vale $6372{,}795477598 Km$, Δlat es la diferencia entre la latitud 1 y la latitud 2, $lat1$ es la latitud en el punto 1, $lat2$ es la latitud en el punto 2 y Δlon es la diferencia entre la longitud 1 y la longitud 2.

Además, contiene botones que permiten graficar las variables obtenidas del módulo embebido, se encuentra el botón *Speed 2D* que grafica la velocidad y proporciona en la consola la velocidad máxima y media obtenidas en todo el recorrido, el botón *Spin 2D* que permite visualizar el giro y el botón *Acelerometer 2D* que presenta la aceleración, cada variable respecto al tiempo. Para graficar las tres variables se utiliza gráficos 2D, donde se necesita definir los datos para el eje x y para el eje y, aquí se usa el módulo *matplotlib* [46] y *pylab* [47], que se encargan de trazar el lienzo para graficar y presentar el gráfico final, además se tiene un submenú de herramientas que posee un zoom, guardar la gráfica, entre otros y solo es necesario colocar

el curso en un cierto punto para mostrar los valores exactos de los ejes en ese punto. Como se mencionó anteriormente para los valores de velocidad máxima y media del botón speed, se emplea el módulo *math* [48] que desarrolla varias funciones matemáticas. Seguido de los botones de gráficas 2D se encuentran las gráficas 3D en las que se localizan dos botones *Spin 3D* y *Acelerometer 3D* estos botones usan la librería *mplot3d* [49] de matplotlib que realiza gráficos 3D y usa los datos de las mediciones en los ejes *x y z*, además posee el submenú de herramientas antes mencionado.

Por último se muestran dos botones *Clear Screen* y *Quit*, el primero limpia el lienzo de gráficos 2D y 3D permitiendo graficar una sola variable a la vez, si se desea mostrar otra variable el lienzo debe estar en blanco, el segundo botón cierra todas ventanas emergentes de la interfaz gráfica.

4.1.5. Diagrama de bloques del funcionamiento de la interfaz gráfica de usuario

El diagrama de bloques que se muestra en la Fig. 4.1 describe el funcionamiento de la interfaz gráfica y su interacción con el usuario.

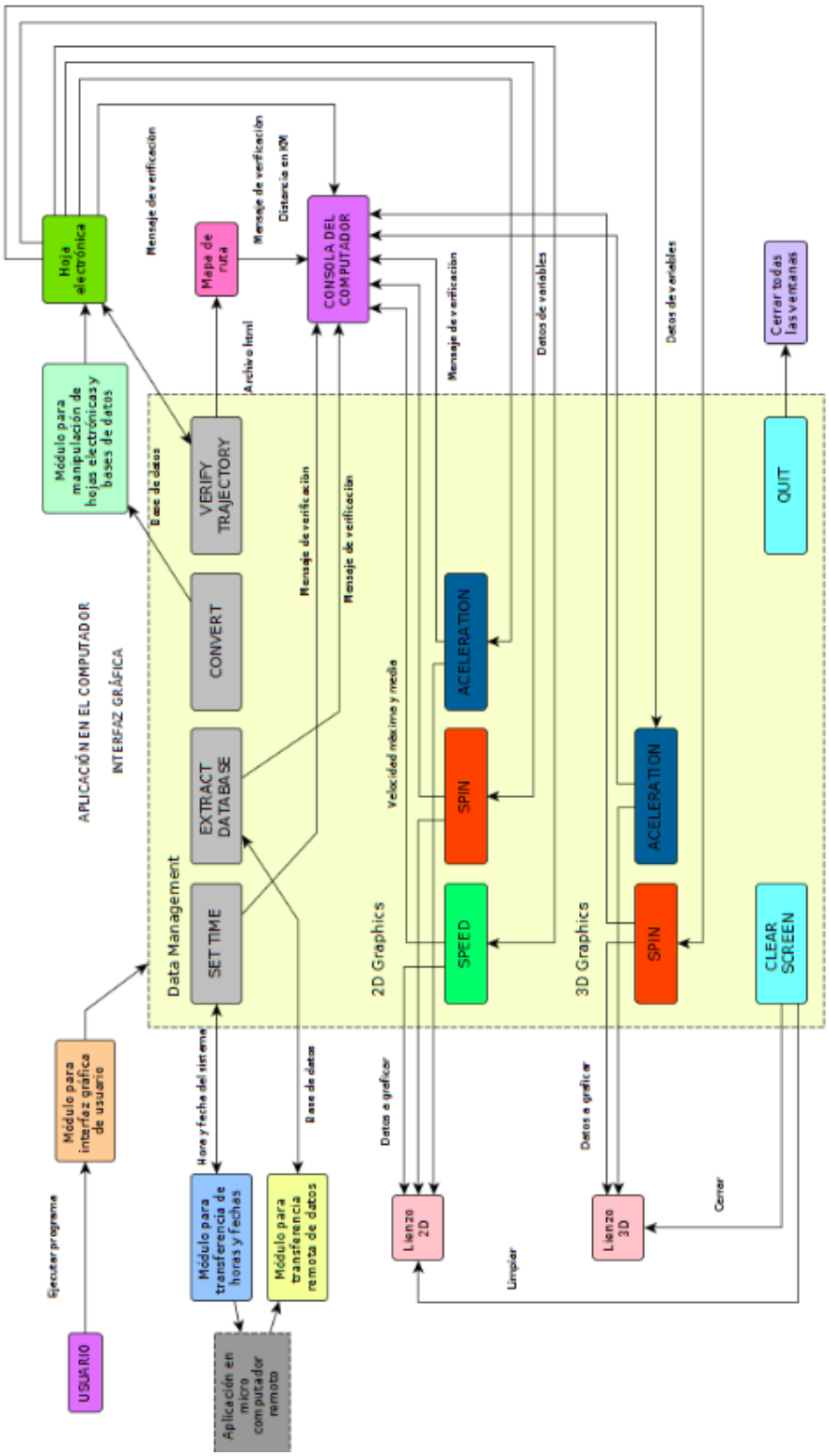


Figura 4.1: Diagrama de bloques del funcionamiento de la interfaz gráfica.

Capítulo 5

Implementación, pruebas y validación

5.1. Implementación de la adquisición, base de datos e interfaz gráfica

Para realizar la implementación de todo el sistema se hace uso de una interfaz gráfica, como herramienta factible y sencilla para que el usuario pueda realizar las diferentes actividades con tan solo un click, sin la necesidad de comprender a fondo la estructura del software, la interfaz consta de tres ventanas: a) Panel de botones que realizan las funciones implementadas, b) El lienzo para gráficos 2D y 3D y c) La consola (terminal de Linux) del computador para notificaciones sobre la ejecución de las actividades realizadas en la aplicación y para mostrar los diferentes valores numéricos resultado de la trayectoria recorrida, como se puede observar en la Fig. 5.1.

Entre las acciones que permite realizar la interfaz gráfica están; actualizar la hora y fecha del micro computador de forma remota, extraer la base de datos que contienen las variables medidas por el sistema embebido desde el micro computador hasta la computadora, convertir la base de datos en una hoja electrónica para su sencilla manipulación, trazar la trayectoria y mostrar la distancia recorrida por el ciclista, graficar en 2D las variables obtenidas de velocidad, giro y aceleración con respecto al tiempo, graficar en 3D las variables de giro y aceleración con sus tres ejes, por último acciones auxiliares como limpiar el lienzo de gráficos y cerrar las ventanas emergentes de la aplicación.

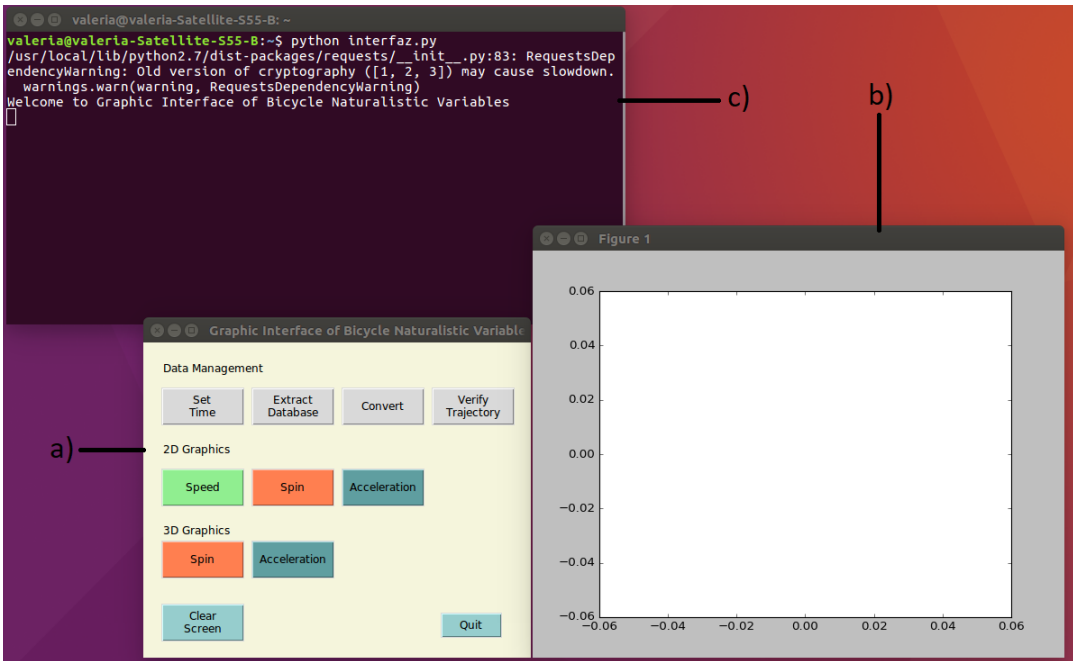


Figura 5.1: Interfaz gráfica.

Para usar todas las funciones que posee la interfaz es necesario conectar inalámbricamente el computador al micro computador para esto se necesita:

- Encender el micro computador.
- Conectar a la misma red inalámbrica el micro computador y la computadora.
- Abrir el código de programación de la interfaz gráfica, ingresar la IP del servidor, su usuario y contraseña en la función *baseextraida* del mismo.
- En el mismo código de programación verificar el número de puerto para la conexión por socket en el computador y el micro computador.

Una vez ingresados todos estos parámetros se procede a compilar la interfaz gráfica para esto se requiere:

- Acceder a la terminal de Ubuntu
- Ejecutar el software de la interfaz, escribiendo en la terminal *python interfaz.py* seguido de la tecla enter.

Tabla 5.1: Botones de la Interfaz Graficá con sus Funciones

Botones	Funciones
Set Time (Establecer el tiempo)	Actualiza la fecha y la hora del micro computador
Extract Database (Extraer base de datos)	Extrae la base de datos desde el micro computador hasta el computador
Convert (Convertir)	Convierte la base de datos en una hoja electrónica
Verify trajectory (Verificar trayectoria)	Traza la trayectoria recorrida por la bicicleta en un mapa
Speed 2D (Velocidad 2D)	Grafica la velocidad con respecto al tiempo en un plano 2D
Spin 2D (Giro 2D)	Grafica el giro con respecto al tiempo en un plano 2D
Aceleration 2D (Aceleración 2D)	Grafica la aceleración con respecto al tiempo en un plano 2D
Spin 3D (Giro 3D)	Grafica el giro con sus tres ejes x y z en un plano 3D
Aceleration 3D (Aceleración 3D)	Grafica la aceleración con sus tres ejes x y z en un plano 3D
Clear Screen (Limpiar pantalla)	Limpia los lienzos de gráficos 2D y 3D
Quit (Salir)	Cierra la interfaz gráfica

Si todos los pasos anteriormente descritos fueron realizados exitosamente el usuario puede interactuar con la interfaz. En ella se encuentran varios botones con su respectivo nombre que describe la actividad que realiza, para una mejor comprensión se muestra la Tabla 5.1.

5.1.1. Funcionamiento de la interfaz gráfica

A continuación se describe el funcionamiento de la interfaz gráfica.

5.1.1.1. Verify trajectory - Verificar trayectoria

Presionando el botón verify trajectory de la Fig. 5.2 a) se obtiene un mapa con la trayectoria realizada por el ciclista mostrando el punto de partida y el de llegada como se presenta en la Fig. 5.2 b), además en la consola del computador se muestra la distancia recorrida en kilómetros como se puede apreciar en la Fig. 5.2 c).

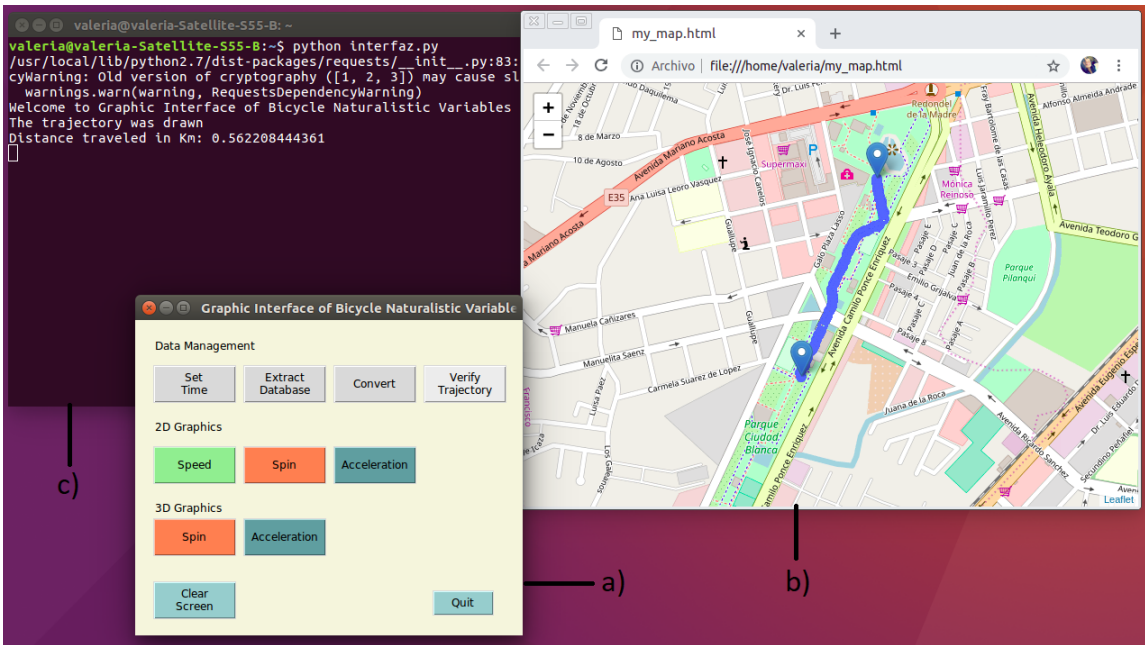


Figura 5.2: Trayectoria del ciclista.

5.1.1.2. Speed - Velocidad

Presionando el botón speed en el panel de botones de la Fig. 5.3 a), se obtiene la gráfica de la velocidad respecto al tiempo, graficado en el lienzo de gráficos 2D como se muestra en la Fig. 5.3 b), además se presenta en la consola la velocidad máxima y media obtenida en todo el recorrido como se aprecia en la Fig. 5.3 c).

5.1.1.3. Spin 2D - Giro 2D

Oprimiendo el botón spin de las gráficas 2D de la Fig. 5.4 a), se obtiene los giros en el eje transversal X, el eje longitudinal Y y en el eje sagital Z, presentando en el lienzo de gráficos cada giro con respecto al tiempo en dos dimensiones como se muestra en la Fig. 5.4 b), por último se presenta la verficación de la acción en la consola como en la Fig. 5.4 c).

5.1.1.4. Aceleration 2D - Aceleración 2D

Presionando el botón aceleration en las gráficas 2D en la Fig. 5.5 a), se presenta la gráfica de la aceleración en dos dimensiones de los ejes transversal X, longitudinal Y y sagital Z cada uno

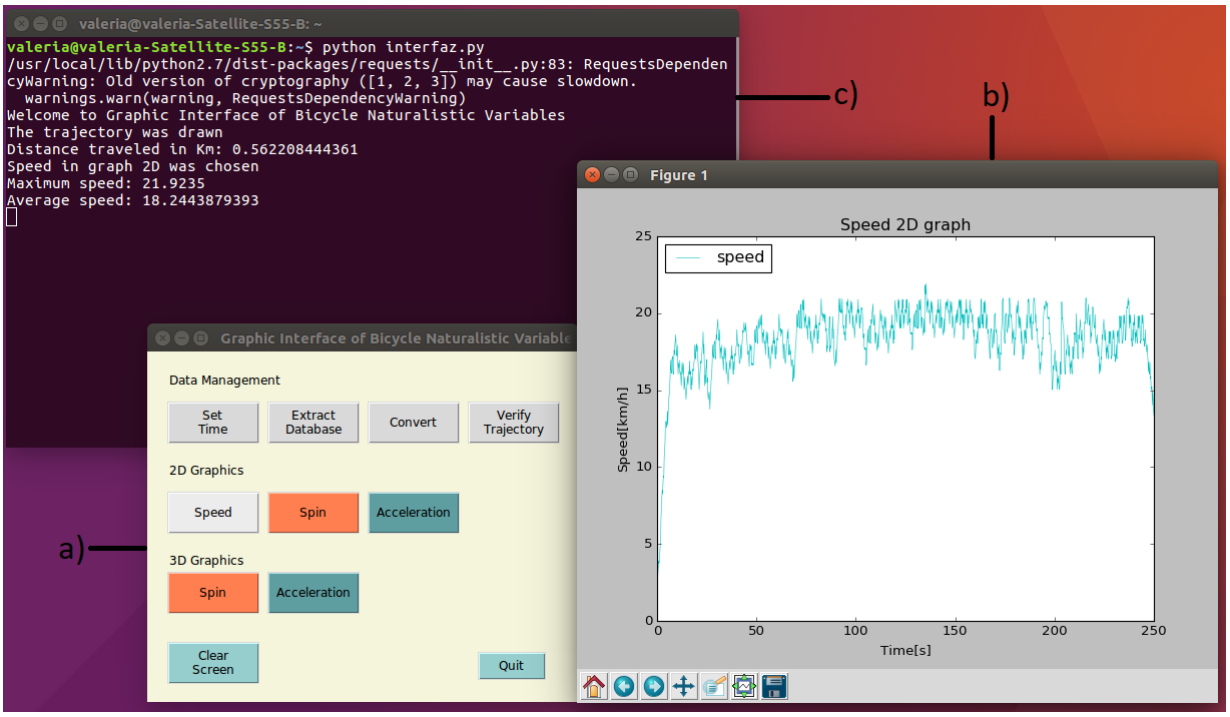


Figura 5.3: Figura 2D de la velocidad.

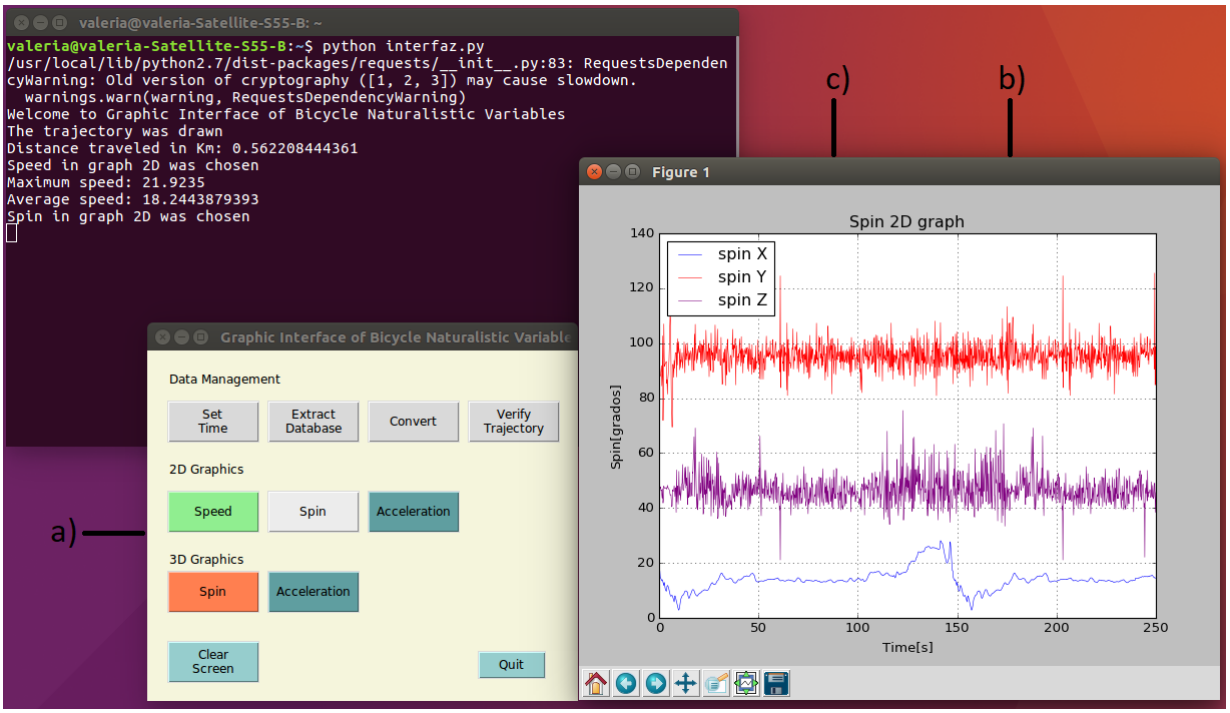


Figura 5.4: Figura 2D del giro.

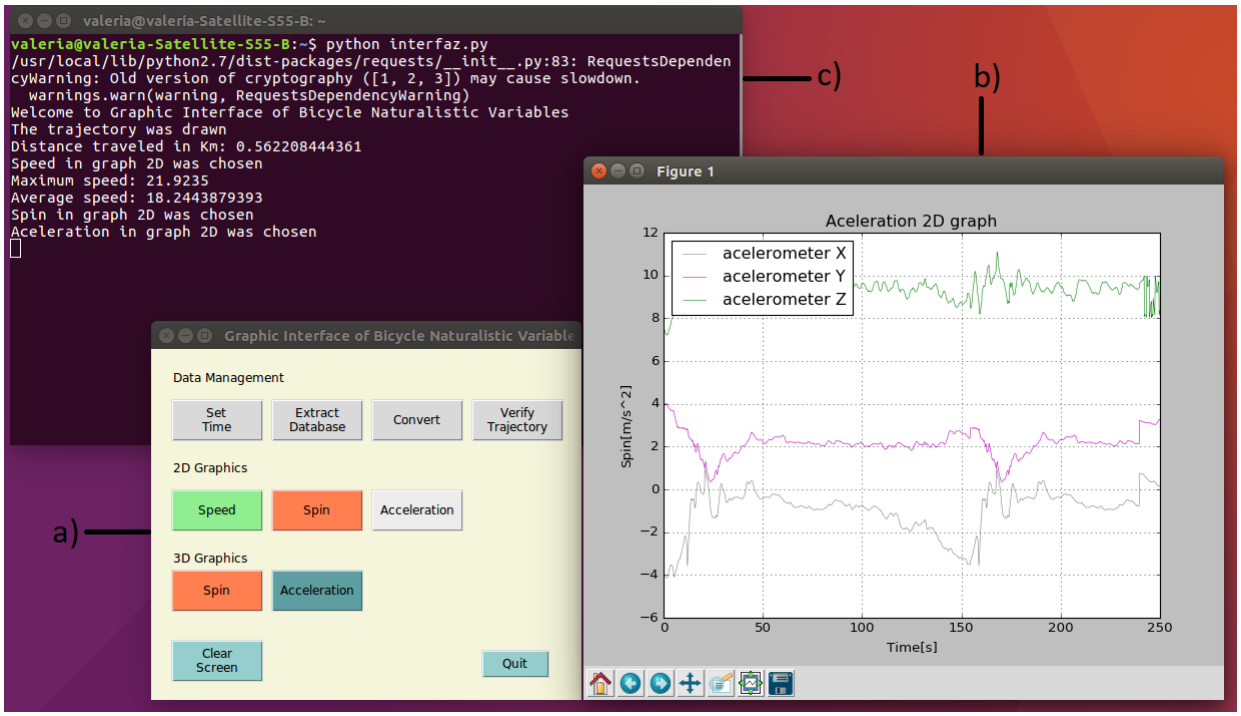


Figura 5.5: Figura 2D de la aceleración.

con respecto al tiempo como se aprecia en la Fig. 5.5 b), finalmente se presenta la confirmación de la actividad realizada en la consola como en la Fig. 5.5 c) .

5.1.1.5. Spin 3D - Giro 3D

Oprimiendo el botón spin en las gráficas 3D de la Fig. 5.6 a), se obtiene la representación tridimensional de los giros en los ejes transversal X, longitudinal Y y sagital Z como se presenta en la Fig. 5.6 b), además se muestra en la consola el mensaje de verificación de que la actividad fue realizada como en la Fig. 5.6 c).

5.1.1.6. Aceleration 3D - Aceleración 3D

Presionando el botón aceleration de las gráficas 3D de la Fig. 5.7 a), se presenta la representación tridimensional de las aceleraciones en los ejes transversal X, longitudinal Y y sagital Z como se muestra en la Fig. 5.7 b), finalmente se presenta un mensaje de confirmación por la actividad realizada como en la Fig. 5.7 c).

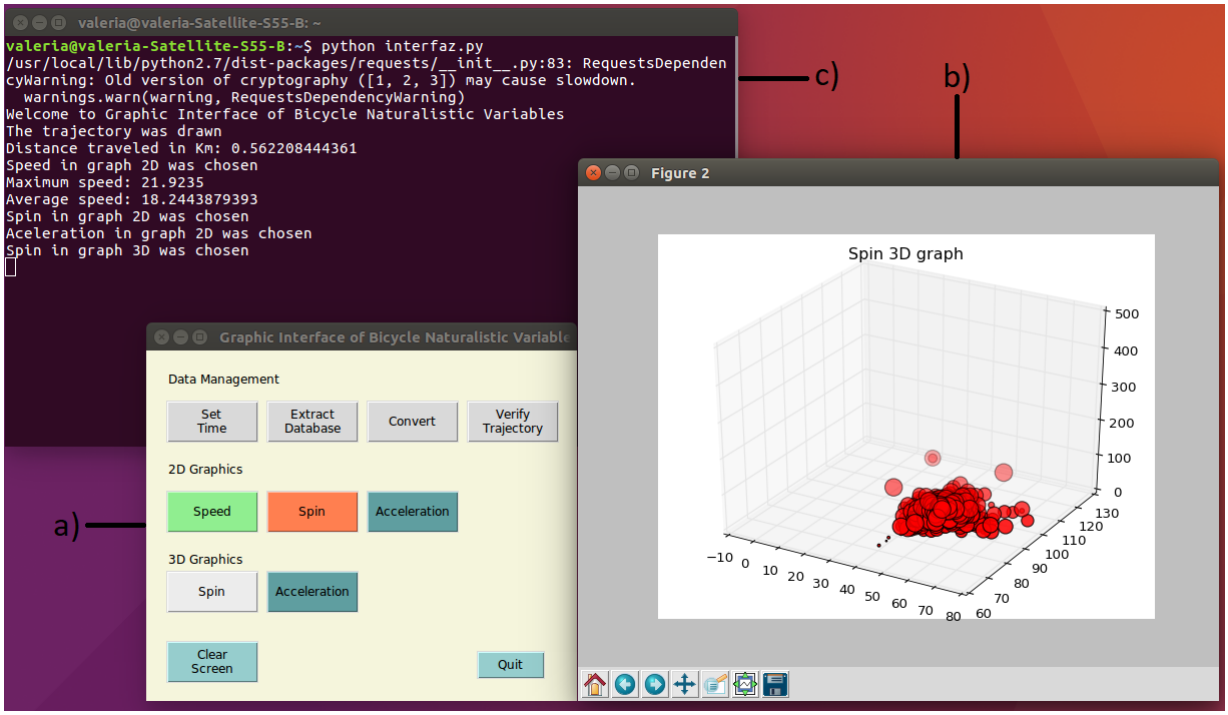


Figura 5.6: Figura 3D del giro.

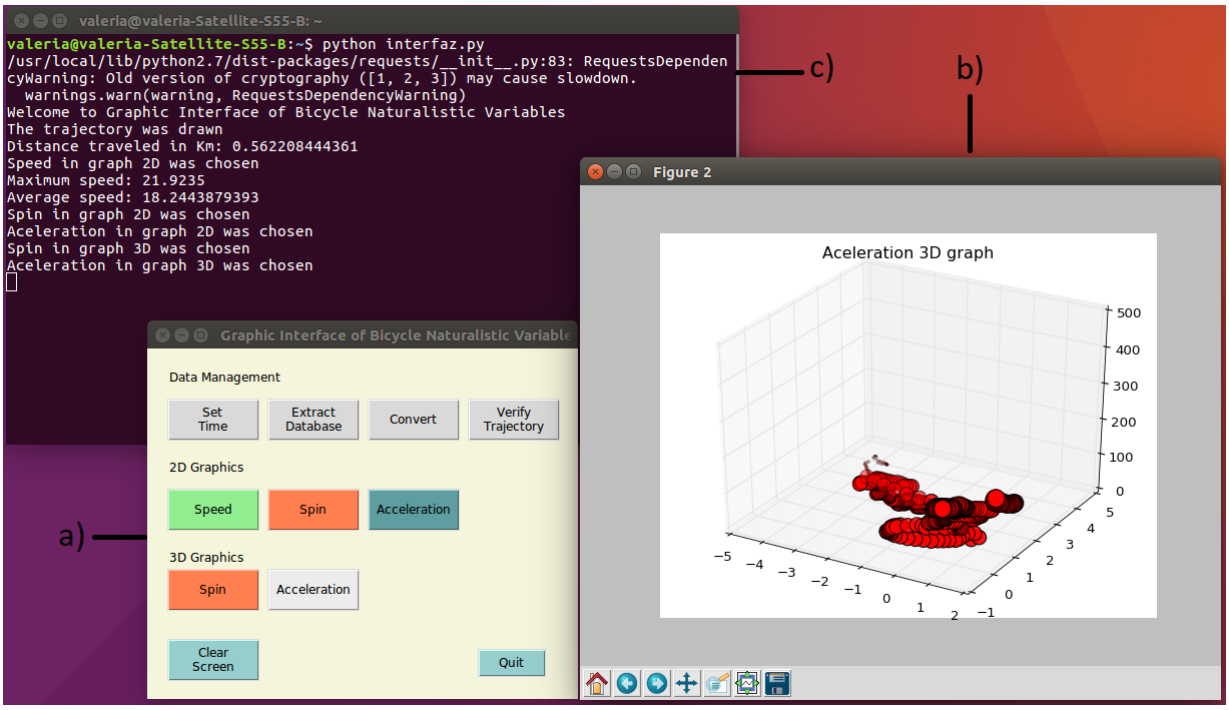


Figura 5.7: Figura 3D de la aceleración.

5.2. Validación de la interfaz gráfica

Se someterá al software desarrollado a pruebas bajo condiciones reales de funcionamiento para determinar su desempeño. Para evaluar el software se toma como referencia la evaluación de usabilidad de entornos integrados con respecto al rendimiento y la satisfacción del usuario propuesto en [50].

5.2.1. Evaluación de usabilidad

La usabilidad es una característica muy relevante de los productos de software, esta define como el usuario emplea o interactúa con el software, logrando usarlo con *eficiencia, eficacia y satisfacción* [51]. Una buena usabilidad garantiza que el software es fácil de aprender, interactivo y agradable para el usuario. Los atributos de usabilidad que se evalúan son [52]:

- Eficacia: Realizar tareas; como el usuario logra cumplir los objetivos de manera precisa y completa.
- Eficiencia: Cantidad de esfuerzo necesario para que el usuario consiga los objetivos.
- Satisfacción: Qué tan agradable y cómodo se siente el usuario al usar la interfaz gráfica.

La metodología usada para evaluar la usabilidad es un método de investigación cuantitativo-experimental, que recopilan información sobre los tres atributos antes mencionados de usabilidad. La eficacia y la eficiencia se estiman por medio de observación directa, mientras que para valorar la satisfacción se aplica una encuesta [50].

Para realizar el método experimental [50] recomienda enlistar tres tareas, estas son propuestas por el diseñador en base a las actividades que se pueden realizar con la interfaz, por esto se decide escoger una actividad con alto grado de dificultad, una de medio nivel y la tercera de bajo nivel. Posteriormente se escoge a 10 usuarios voluntarios que deben ejecutar estas actividades dentro de un rango de tiempo establecido. La población evaluada está conformada por personas inexpertas en la manipulación de software de interfaces gráficas, se pidió la colaboración de estudiantes de la FICAYA. Así la evaluación a los usuarios será fiable ya que no cuentan con previa experiencia.

Finalmente, para evaluar se registra, analiza e interpreta los resultados considerando los atributos de usabilidad expuestos en [50]. Las tareas a realizar son:

- Tarea 1: Conectar inalámbricamente la interfaz con el micro computador. Se valora el proceso de conexión del microcomputador a la computadora para garantizar el funcionamiento correcto.
- Tarea 2: Compilar la interfaz gráfica.
- Tarea 3: Interactuar con la interfaz gráfica. Se requiere que el usuario realice el proceso de adquisición y tratamiento de la información; incluye la actualización de la hora y fecha del micro computador, extraer la base de datos, convertir la base a una hoja electrónica, trazar la trayectoria de recorrido y graficar las variables de velocidad, aceleración y giro.

Los pasos de cómo realizar las diferentes tareas se indicaron a los usuarios voluntarios y se los pueden encontrar en la sección 5.1 de este trabajo. Al realizar las tareas no se permite el acceso a internet u otras fuentes de consulta.

El tiempo límite para realizar las actividades se establece mediante el tiempo usando por el diseñador para ejecutarlas más el 50 % de ese tiempo. Para considerar que un usuario voluntario completó una tarea, debe cumplir con el objetivo solicitado y entregarlo antes del tiempo límite. Si una tarea no fue desarrollada con éxito dentro del tiempo planteado, se la toma como insatisfactoria con un tiempo de desarrollo igual al tiempo límite [50].

Los tiempos límites para la ejecución de las tareas por los voluntarios inexpertos son 240, 180, 100 segundos, respectivamente en cada tarea.

5.2.1.1. Prueba de efectividad

Para esta prueba se considera el cumplimiento de las tareas (si o no). La eficiencia e_x de cada tarea está dentro del rango $[0,1]$ siendo: bajo $0 \leq e_x \leq 0{,}33$, medio $0{,}33 < e_x \leq 0{,}67$, y alto $0{,}67 < e_x \leq 1$. Para encontrar este valor se usa la relación [50]

$$e_x = \frac{V_x}{n} \tag{5.1}$$

donde V_x es el número de voluntarios que completo la tarea, y n el total de voluntarios evaluados.

5.2.1.2. Prueba de eficiencia

Esta prueba se basa en comparar los tiempos de ejecución s_x de las tareas obtenidas con la interfaz gráfica por usuarios sin experiencia previa y con el desarrollo de las mismas actividades pero con usuarios expertos. El tiempo promedio $\bar{s_x}$ se calcula usando la ecuación [50]

$$\bar{s_x} = \frac{1}{n}\sum_{i=1}^{n} s_{x_i} \tag{5.2}$$

donde $s_{x_i} \varepsilon \mathbb{R}$ es el tiempo que el voluntario usa para completar la tarea y n es el total de voluntarios. La relación r_{s_x} se calcula con la división entre $\bar{s_x}$ de las dos interfaces gráficas.

5.2.1.3. Prueba de satisfacción

Para esta prueba se toma en cuenta dos evaluaciones: satisfacción y facilidad de uso. La escala de calificación es del rango $[1,5]$. Para definirlas, se consideran los valores estadísticos como valor medio y la desviación estándar. El valor promedio $\bar{Y_q}$ se calcula mediante [50]

$$\bar{Y_q} = \frac{1}{n}\sum_{i=1}^{n} Y_{q_i} \tag{5.3}$$

donde q representa cada pregunta, la evaluación de satisfacción se define como $q \varepsilon \mathbb{R} | 1 \leq q \leq 10$, y la evaluación de la facilidad de uso es $q \varepsilon \mathbb{R} | 1 \leq q \leq 4$, Y_{q_i} es la calificación que el usuario da a cada pregunta y n es el número de usuarios voluntarios [50].

La desviación estándar $\bar{S_q}$ se calcula como

$$\bar{S_q} = \sqrt{\frac{\sum_{i=1}^{n}(\bar{Y_q} - Y_{q_i})^2}{n}} \tag{5.4}$$

Para evaluar la satisfacción de los voluntarios al usar la interfaz gráfica se emplea el cuestionario de escala de usabilidad del sistema. Para determinar una puntuación en cada pregunta se usa un intervalo de $[1,5]$ el número 1 que indica desacuerdo total y 5 acuerdo total. El cuestionario de usabilidad del sistema en base a [51] es:

1. Me gustaría utilizar esta interfaz gráfica con frecuencia
2. Encuentro esta interfaz gráfica innecesariamente complejo
3. Creo que esta interfaz gráfica es fácil de usar
4. Necesitaría ayuda para usar esta interfaz gráfica
5. Las diversas funciones están bien integradas (constituyen un todo)
6. Hay demasiada incoherencia en la interfaz gráfica
7. La mayoría de las personas aprendería a usar interfaz gráfica muy rápidamente
8. Me resulta muy engorroso / difícil de usar
9. Lleno con mucha confianza usándolo
10. Necesito aprender muchas cosas antes de poder comenzar a trabajar con esta interfaz gráfica

Para medir la facilidad de uso y comodidad de la interfaz gráfica, [53] propone las siguientes preguntas:

1. El esfuerzo mental requerido para el desarrollo de las tareas ha sido [1 gran esfuerzo, 5 pequeños esfuerzos]
2. La velocidad de funcionamiento es [1 muy lento, 5 muy rápido]
3. La comodidad es [1 muy incómodo, 5 muy cómodo]
4. En general, la gestión de la interfaz gráfica es [1 muy difícil, 5 muy fácil]

5.2.2. Resultados de la evaluación

Los resultados de las pruebas de efectividad, eficiencia y satisfacción basándose en las tres tareas antes presentadas se muestran a continuación.

Tabla 5.2: Eficacia de los usuarios inexpertos para desarrollar las tareas

Usuarios	Tarea 1	Tarea 2	Tarea 3
1	si	si	si
2	no	si	si
3	no	no	si
4	si	si	si
5	no	no	si
6	si	no	si
7	no	si	si
8	si	si	si
9	si	si	si
10	no	no	si

Tabla 5.3: Efectividad para realizar las tres tareas

Tareas	Valor de eficacia	Efectividad
Tarea 1	0.5	Media
Tarea 2	0.6	Media
Tarea 3	1	Alta

5.2.2.1. Resultados de la prueba de efectividad

La Tabla 5.2 presenta los resultados del cumplimiento o no de las tareas y la Tabla 5.3 muestra la efectividad de los usuarios para desarrollar las actividades.

5.2.2.2. Resultados de la prueba de eficiencia

Para llevar a cabo esta prueba se desconoce de otra aplicación que realice actividades similares a las de esta, por lo que se compara los tiempos de desarrollo $\bar{s_x}$ de las tareas obtenidas con interacción de la interfaz gráfica por personas inexpertas con el desempeño de la misma interfaz pero con personas que ya conocen el software, los resultados se muestran en la Tabla 5.4.

Tabla 5.4: Relación de tiempos de eficiencia para el desarrollo de las tareas

Tarea	$\bar{s_{x1}}$	$\bar{s_{x2}}$	r_{s_x}
1	231.2	196.4	1.17
2	169.8	156	1.09
3	63.2	59.8	1.06

Tabla 5.5: Resultados del cuestionario de la escala de usabilidad

Pregunta	Valor Medio	Desviación Estándar
1	4	0.63
2	2.2	0.4
3	4.2	0.4
4	1.6	0.49
5	4.2	0.4
6	1	0
7	4.4	0.49
8	1.4	0.49
9	4.2	0.4
10	3	0.63

Tabla 5.6: Resultados del cuestionario de comodidad / facilidad de uso

Pregunta	Valor Medio	Desviación Estándar	Evaluación final
1	4.2	0.74	Pequeño esfuerzo
2	4.8	0.4	Rápido
3	4.4	0.49	Cómodo
4	4.4	0.49	Fácil de usar

5.2.2.3. Resultados de la prueba de satisfacción

Los resultados del valor medio y la desviación estándar de la encuesta de escala de usabilidad del sistema se pueden observar en la Tabla 5.5.

Los valores medios y desviaciones estándar obtenidos de la encuesta de comodidad / facilidad de uso se encontran en la Tabla 5.6. Para esta característica se realiza una evaluación final en cada pregunta.

5.2.3. Discusión sobre la evaluación de la interfaz gráfica

La Tabla 5.3 muestra la efectividad del voluntario al usar la interfaz gráfica aumenta de 0.4 a 1 a medida que disminuye la complejidad de las tareas, pues la tarea 1 es considerada la más difícil y 3 la más fácil e intuitiva para el usuario. Teniendo en cuenta que los usuarios evaluados no tienen experiencia en el manejo de interfaces gráficas, es previsible que la actividad con más dificultad haya tenido un valor tan bajo, pero es satisfactorio que la tarea 3 logró una efectividad de 1, valorando que la interfaz si es intuitiva y permite interactuar con los datos.

Los resultados de eficiencia se muestran en la Tabla 5.4, los que presentan los tiempos de

desarrollo; $s_{\bar{x}1}$ de la interacción con usuarios sin experiencia y $s_{\bar{x}2}$ para usuarios con experiencia. El valor de la relación permanece mayor a 1, lo que indica que los tiempos de los usuarios sin experiencia son mayores que los de con experiencia. Este resultado era evidente, pues no había un conocimiento previo sobre interfaces, pero a pesar de esto, las relaciones no son muy elevadas permitiendo concluir que la respuesta de desarrollo con la interfaz es eficiente.

La prueba de satisfacción permite evaluar la apreciación de la interfaz gráfica por el usuario, además proporciona datos cuantitativos sobre la satisfacción del usuario describiéndola en los valores medios y las desviaciones estándar. Estos valores permiten observar directamente los niveles de valoración sobre la interfaz, a través de los valores medios $\bar{Y}_q$ que dieron los usuarios se puede verificar que las respuestas dadas son de aceptación y satisfacción por la interfaz gráfica, mientras que las desviaciones estándar $\bar{S}_q$ muestran las dispersiones en las respuestas como se muestra en la Tabla 5.5, al ser valores bajos demuestran que los usuarios estuvieron en concordancia entre ellos y de existir valores ciertamente altos comprueba que los usuarios tenían respuestas dispersas derivadas de la falta efectividad y eficiencia al llevar a cabo las tareas.

Por último están los resultados al cuestionario de comodidad / facilidad de uso situados en la Tabla 5.6, donde la columna de evaluación final, evidencia que los usuarios sienten comodidad al interactuar con la interfaz gráfica, además la consideran rápida y fácil de usar.

5.3. Validación del software de adquisición, tratamiento de datos e interfaz gráfica con sistemas comerciales

Para validar la utilidad de la información presentada sobre el software, se procede a compararla con sistemas comerciales.

Entre los dispositivos utilizados por ciclistas se encuentran los relojes multideporte, que resultan relativamente incomodos al consultar los datos en la pantalla cuando el usuario se encuentra subido en la bicicleta, además estos no proporcionan suficiente información, ya que en su mayoría solo muestran la velocidad, distancia recorrida y ritmo cardiaco [54].

Para tener mayor comodidad y una más amplia gama de información existen los ciclo-

computadores que se colocan en el manillar y permiten tener un mejor acceso a la pantalla y a los botones del dispositivo, entre las variables que ofrecen están la velocidad, GPS, frecuencia cardiaca, potencia de la pedaleada, consumo de calorias entre otras, el inconveniente con este tipo de dispositivos es su alto costo [55].

Finalmente se hace uso de los smartphone, hoy en día la mayor parte de estos dispositivos poseen varios sensores por lo que el uso de aplicaciones que proporcionen información sobre el recorrido en bicicleta es la opción más fácil. Entre las aplicaciones más destacadas para ciclismo están: Strava que permite registrar estadísticas sobre la distancia, el ritmo cardiaco, velocidad, aumento de la altitud y estimación de las calorías consumidas durante la trayectoria y Runtastic que registra la distancia, duración, velocidad, altitud y ritmo cardiaco [56].

Luego de analizar las tres opciones antes expuestas podemos valorar el software desarrollado en este trabajo, los relojes multideporte son los menos aptos para usarse en la adquisición de variables ya que no proporcionan comodidad al usuario y no permite obtener las variables necesarias para el software. Por otro lado las ciclocomputadoras son elevabadamente costosas por lo que no son de interés en este estudio. Finalmente la opción que más se acerca a los requerimientos del sistema son las aplicaciones en smartphone sin embargo no cubren todas las variables que se necesita adquirir, además no permiten obtener una base de datos, pues en su mayoría solo brindan valores promedio.

Capítulo 6

Conclusiones y Recomendaciones

En este capítulo se presenta las conclusiones y recomendaciones del trabajo, además se propone posibles líneas para un trabajo futuro.

6.1. Conclusiones

En este trabajo se ha presentado una aplicación para la adquisición y tratamiento de variables naturalísticas obtenidas desde bicicletas, para esto se establecieron requerimientos funcionales y no funcionales sobre el sistema a partir de las necesidades que demanda la aplicación y los usuarios, estos marcaron el diseño y desarrollo del sistema y las funciones finales que el mismo será capaz de realizar.

Se implementó algoritmos que permitieron la transferencia de datos, usando protocolos como TCP—IP y SSH. Para el tratamiento y presentación de los mismos se optó por usar hojas electrónicas que facilitaron su manipulación.

En la implementación total del sistema computacional se eligió usar una interfaz gráfica, que integró todas las funciones que es capaz de realizar el sistema, así como la adquisición y tratamiento de datos, además se agregaron funciones que facilitaron el manejo y la interacción con la información, haciendo una interfaz con varias opciones para visualizar los datos, que es cómoda y fácil de usar para el usuario.

Para someter el sistema a condiciones reales de funcionamiento se usó la evaluación de usabilidad que permitió verificar la efectividad, eficiencia y satisfacción del usuario al manipular la interfaz gráfica. Para llevar a cabo la prueba se tomó dos grupos de estudio: el primer grupo y más importante para la validación lo conformaron cinco usuarios inexpertos con la interfaz, y el segundo cinco usuarios expertos. Los resultados arrojados demostraron que la efectividad del voluntario al usar la interfaz aumentó a medida que disminuyó la complejidad de las tareas, pero si se toma en cuenta que los usuarios evaluados no tienen experiencia, su alta puntuación en las últimas tareas valoró a la interfaz como muy intuitiva y permitió interactuar con los datos. Los datos de la eficiencia evidenciaron que a pesar de que los tiempos de los usuarios sin experiencia son mayores que los de con experiencia, las relaciones no son elevadas permitiendo concluir que la respuesta de desarrollo con la interfaz es eficiente. En la prueba de satisfacción a través de los valores medios, se pudo verificar que las respuestas dadas son de aceptación y satisfacción a la interfaz y que con el cuestionario de comodidad / facilidad de uso los usuarios sintieron comodidad y la consideraron rápida y fácil de usar.

6.2. Recomendaciones

- Para un buen servicio de la interfaz gráfica se debe tener una buena red a internet.
- Verificar que los diferentes parámetros necesarios para la conexión por protocolos sean correctos.
- Se podrían presentar más datos en la interfaz gráfica que por el momento no son adquiridas.
- Comprobar el nivel de carga de la batería que alimenta al micro computador.

6.3. Trabajo futuro

La información obtenida y almacenada en bases de datos servirán para llevar a cabo estudios de como conducen los ciclistas, así determinar estados de conducción. Por otro lado, la información recopilada podrá usarse para el análisis de la siniestralidad que involucra el uso inadecuado de la bicicleta a través de información capturada en el momento del accidente, de

la misma forma que una caja negra o registrador de vuelo en los aviones.

El sistema podrá ser replicado para compartir inalámbricamente estados de conducción en una comunidad de ciclistas, permitiendo la exploración de las condiciones de las rutas, incrementar la eficiencia en el manejo deportivo de bicicletas e incluso aumentar la seguridad del ciclista.

Bibliografía

[1] M. Proaño, "Cultura ciclera en Quito, políticas de movilidad: estudio de caso ciclópolis y al sur en Bici", trabajo de fin de máster, Universidad Andina Simón Bolívar. Quito, 2012.

[2] J. García y L. Naranjo, "Anuario de estadística de transporte", Instituto Nacional de Estadísticas y Censos, Quito, Ecuador, 2016.

[3] N. Pinto, F. Fuentes y D. Alcivar, "La situación de la bicicleta en Ecuador: avances, retos y perspectivas", *Friedrich Ebert Stiftung*, Quito, Ecuador, ISBN: 978-9978-94-147-8, mar.2015.

[4] A. García, "Los accidentes con bicicletas aumentaron 16 % en primer trimestre del 2017", J. *El Comercio*. May., 2017 [En línea]. Disponible en: https://www.elcomercio.com/actualidad/accidentes–bicicletas–guayas–aumentaron–ciclopaseo.html. [Accedido: 15-may-2018]

[5] El Comercio, "Imágenes de las faltas más comunes en las ciclovías de Quito", J. *El Comercio*, 2014. [En línea]. Disponible en: https://www.elcomercio.com/actualidad/quito/imagenes–de–faltas–mas–comunes.html. [Accedido: 15-may-2018]

[6] F. Molina, G. Alvarez y J. Torres, "Determinación del ciclo típico de conducción de una bicicleta en las ciclovías de la ciudad de Cuenca", trabajo de fin de grado, Universidad del Azuay, Cuenca, 2016.

[7] S. Chazallet, *Python 3 los Fundamentos del Lenguaje.* F. Piqueres, Ed. Barcelona: ENI, 2015.

[8] H. Spona, *Programación de Bases de Datos con MySQL y PHP.* Marcombo, Ed. México: Alfaomega, 2012.

[9] J. Rea, “Diseño, desarrollo e implementación de un protocolo de comunicaciones entre sensores en red y computadores”, trabajo de fin de grado, Universidad Autónoma de Madrid, Madrid, 2006.

[10] M. Sabana, *PHP con PostgreSQL 8.* Lima: Megabyte, 2006.

[11] A. Castro, J. González y M. Callejas, “Utilidad y funcionamiento de las bases de datos NoSQL”, *J. CEDEC*, vol. 21, no. 33, pp. 21-32, dic. 2012.

[12] R. Herranz, “Bases de datos NoSQL: arquitectura y ejemplos de aplicación”, trabajo de fin de grado, Universidad Carlos III de Madrid, Madrid, 2014.

[13] J. Dordoigne, *Redes Informáticas*. J. Musset, Ed. Barcelona: ENI, 2013.

[14] D. Comer, *Redes de Computadoras e Internet*. A. Romero y C. Enríquez, Ed. México: Pearson, 2015.

[15] J. Raya y L. Raya, *Redes Locales*. 4Ta. Ed. México: Alfaomega, 2006.

[16] F. Reyes y J. Cid, *Arduino. Aplicaciones en Robótica, Mecatrónica e Ingenierías*. M. Grillo, Ed. México: Alfaomega, 2015.

[17] J. Márquez, K. Pardo y S. Pizarro, “Ethernet: Su origen, funcionamiento y rendimiento”, *Ingeniería y Desarrollo*, no. 9, pp. 22-34, jul. 2001.

[18] A. Tanenbaum y D. Wetherall, *Redes de Computadoras*. A. Romero y C. Enríquez, Ed. México: Pearson, 2012.

[19] E. Rico, “Puntos básicos del modelo OSI – parte 1: aspectos generales”, 2018. [En línea]. Disponible: http://www.ermesh.com/modelo-osi-parte-1-aspectos-generales/. [Accedido: 31-may-2018]

[20] S. Szmandiuk, “Modelo OSI y TCP/IP”, 2017. [En línea]. Disponible: http://silvanaszmandiuk.blogspot.com/2017/10/modelo-osi-y-tcpip.html. [Accedido: 31-may-2018]

[21] H. de la Cruz, *Redes: Instalación, Administración y Soporte*. Lima: MACRO, 2013.

[22] C. Albornoz, “Diseño de interfaz gráfica de usuario”, en *WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación*, Argentina, pp. 540-542.

[23] M. Fernández, J. Angós y J. Salvador, “Interfaces de usuario: diseño de la visualización de la información como medio para mejorar la gestión del conocimiento y los resultados obtenidos por el usuario”, en *V Congreso ISKO*, España, pp. 2-5.

[24] J. Preece, Y. Rogers y H. Sharp, *Interaction Design*. G. Crockett, Ed. Estados Unidos de América: John Wiley y Sons, Inc., 2002.

[25] L. Luna, “El diseño de interfaz gráfica de usuario para publicaciones digitales”, *Revista Digital Universitaria*, vol. 5, no. 7, pp. 4-7, ago. 2004.

[26] M. Tabares, R. Anaya y F. Arango, “La ingeniería de requisitos orientada a aspectos: una experiencia de aplicación en un sistema de ayuda en línea”, *Scielo*, no. 153, pp. 285-299, nov. 2007. doi: ISSN 0012-7353

[27] V. Hernández y Y. Machado, “Describiendo requisitos verificables”, *Innovation and Development for the Americas*, jun. 2010.

[28] J. Rumbaugh, I. Jacobson y J. Booch, *El Lenguaje Unificado de Modelado. Manual de Referencia*. Madrid: Addison Wesley, 2000.

[29] H. Cervantes, P. Velasco y L. Castro, *Arquitectura de Software, Conceptos y Ciclo de Desarrollo*. R. Kinney, Ed. México: CENGAGE Learning, 2016.

[30] A. Jahangiri, M. Elhenawy, H. Rakha y T. Dingus, “Investigating Cyclist Violations at Signal-Controlled Intersections using Naturalistic Cycling Data”, en *19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016, pp. 2619-2624.

[31] P. Martinez, “SOFTWARE LIBRE, LiNUX Y UBUNTU”, *EUBacteria*, vol. 1, no. 17, pp. 23-24, 2006. doi: ISSN-1697-0454.

[32] E. Bahit, “Curso: Python para principiantes”, 2012. [En línea]. Disponible en: https://docs.python.org/3/library/time.html. [Accedido: 30-may-2018].

[33] Python, “Time - Time access and conversions”, 2018. [En línea]. Disponible en: https://docs.python.org/3/library/time.html. [Accedido: 13-jun-2018].

[34] Python, “os - Miscellaneous operating system interfaces”, 2018. [En línea]. Disponible en: https://docs.python.org/3/library/os.html. [Accedido: 13-jun-2018].

[35] Python, “sys - System-specific parameters and functions”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/sys.html. [Accedido: 13-jun-2018].

[36] Python, “sqlite3 - DB-API 2.0 interface for SQLite databases”, 2018. [En línea]. Disponible en: https://docs.python.org/3.4/library/sqlite3.html. [Accedido: 13-jun-2018].

[37] Python, “csv - CSV file reading and writing”, 2018. [En línea]. Disponible en: https://docs.python.org/3/library/csv.html. [Accedido: 13-jun-2018].

[38] Python, “socket - Low-level networking interface”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/socket.html. [Accedido: 13-jun-2018].

[39] Paramiko, “Welcome to paramiko”, 2018. [En línea]. Disponible en: http://www.paramiko.org/. [Accedido: 13-jun-2018].

[40] Python, “warnings - Warning control”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/warnings.html. [Accedido: 13-jun-2018].

[41] Python, “md5 - Algoritmo de resumen de mensaje MD5”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/md5.html. [Accedido: 13-jun-2018].

[42] Python, “Graphical user interfaces with Tk”, 2018. [En línea]. Disponible en: https://docs.python.org/3/library/tk.html. [Accedido: 21-jul-2018].

[43] Folium, “Folium 0.6.0 ”, 2018. [En línea]. Disponible en: https://pypi.org/project/folium/. [Accedido: 21-jul-2018].

[44] Python, “webbrowser — convenient web-browser controller”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/webbrowser.html. [Accedido: 21-jul-2018].

[45] A. Diaz, “Calculadora Geoespacial”, Universidad de la Plata, Argentina, ISSN: 1850-2946, 2012. [En línea]. Disponible en: http://41jaiio.sadio.org.ar/sites/default/files/41-EST-2012.pdf. [Accedido: 21-jul-2018].

[46] Matplotlib, “Matplotlib version 2.2.2.”, 2018. [En línea]. Disponible en: https://matplotlib.org/. [Accedido: 21-jul-2018].

[47] Matplotlib, “Pyplot tutorial”, 2018. [En línea]. Disponible en: https://matplotlib.org/users/pyplot/tutorial.html. [Accedido: 21-jul-2018].

[48] Python, “math — Mathematical functions”, 2018. [En línea]. Disponible en: https://docs.python.org/2/library/math.html. [Accedido: 21-jul-2018].

[49] Matplotlib, “Matplotlib mplot3d toolkit”, 2018. [En línea]. Disponible en: https://matplotlib.org/mpl-toolkits/mplot3d/index.html. [Accedido: 21-jul-2018].

[50] C. Vaca, C. Rosero, A. Basantes y D. Ortiz, “Usability of integrated environments for programming embedded systems with respect to performance and user satisfaction in teaching/learning scenarios”, jul. 2018.

[51] J. Brooke, “SUS-A quick and dirty usability scale,” en *Usability evaluation in industry*, vol. 189, no. 194, pp. 4-7, 1996.

[52] J. Preece, Y. Rogers y H. Sharp, *Interaction Design Beyond Human Computer-Interaction*, 4th ed. Chichester: John Wiley and Sons Inc, 2015.

[53] C. Manresa-Yee, P. Ponsa, J. Varona y F. Perales, “User experience to improve the usability of a vision-based interface” en *Interacting with computers, special issue on inclusion and interaction: designing interaction for inclusive populations*, vol. 22, no. 6, pp. 594-605, Nov. 2010.

[54] Garmin, “FORERUNNER Manual del usuario”, 2017. [En línea]. Disponible en: http://static.garmin.com/pumac/Forerunner935-OM-ES.pdf. [Accedido: 11-dic-2018].

[55] E. Martínez, “Los mejores ciclocomputadores de 2018”, 2018. [En línea]. Disponible en: https://deporte10.top/mejores-ciclocomputadores/. [Accedido: 11-dic-2018].

[56] S. Borja, “Aplicaciones Android”, 2017. [En línea]. Disponible en: https://elandroidelibre.elespanol.com/2017/03/mejores-aplicaciones-bicicleta.html. [Accedido: 11-dic-2018].

Apéndice

Este apéndice presenta el código de la interfaz gráfica que incluye todas las funciones que realiza la Plataforma para Experimentación Naturalística en Bicicletas: Software de Adquisición y Tratamiento de Datos desarrollados en el proyecto.

.A. Software

.A.1. Interfaz gráfica (interfaz.py)

Programa 1: Interfaz gráfica

```
from Tkinter import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import warnings
import os
import paramiko
import md5
import sqlite3
import csv
import matplotlib.pyplot as pltcsv
import sys
import pylab as pl
import socket
import time
import folium
import webbrowser
import numpy as np
import math



gen = 0

def speed():
        global gen
        if gen == 0:
                gen = 1
                print 'Speed in graph 2D was chosen'
                entrada = open('base.csv')
                tabla = []
```

```
                for fila in csv.reader(entrada):
                        tabla.append(fila)
                entrada.close()
                x=[0]
                y=[0]
                for fila in range(1, len(tabla)):
                        x.append(float(tabla[fila][0]))
                        y.append(float(tabla[fila][1]))
                maximo=max(y)
                num=len(y)
                suma=(sum(y))/num

                print("Maximum speed: %s" %maximo)
                print("Average speed: %s" %suma)

                pl.plot(x,y,color="c",  linewidth=0.5, linestyle="-", label="speed")
                pl.legend(loc='upper left')
                pl.xlabel('Time[s]')
                pl.ylabel('Speed[km/h]')
                pl.title('Speed 2D graph')
                pl.show()



def spin2():
        global gen
        if gen == 0:
                gen = 1
                print 'Spin in graph 2D was chosen'
                entrada2 = open('base.csv')
                tabla2 = []
                for fila2 in csv.reader(entrada2):
                        tabla2.append(fila2)
                entrada2.close()
                x2=[0]
                y2=[0]
                z2=[0]
                u2=[0]
                for fila2 in range(1, len(tabla2)):
                        x2.append(float(tabla2[fila2][0]))
                        y2.append(float(tabla2[fila2][7]))
                        z2.append(float(tabla2[fila2][6]))
                        u2.append(float(tabla2[fila2][5]))
                pl.plot(x2,y2,color="blue", linewidth=0.5, linestyle="-", label="spin X")
                pl.plot(x2,z2,color="red",  linewidth=0.5, linestyle="-", label="spin Y")
                pl.plot(x2,u2,color="purple",  linewidth=0.5, linestyle="-", label="spin Z")
                pl.legend(loc='upper left')
                pl.xlabel('Time[s]')
```

```
                pl.ylabel('Spin[grados]')
                pl.title('Spin 2D graph')
                pl.grid(True)
                pl.show()


def spin3():
        global gen
        if gen == 0:
                gen = 1
                plt.close()
                pl.close()
                print 'Spin in graph 3D was chosen'
                fig = plt.figure()
                ax = fig.gca(projection='3d')

                entrada = open('base.csv')
                tabla = []
                for columna in csv.reader(entrada):
                        tabla.append(columna)
                entrada.close()
                T=[]
                X=[]
                Y=[]
                Z=[]
                for columna in range(1, len(tabla)):
                        T.append(float(tabla[columna][0]))
                        X.append(float(tabla[columna][5]))
                        Y.append(float(tabla[columna][6]))
                        Z.append(float(tabla[columna][7]))

                ax.scatter(X, Y, Z, marker='o', c='r',s=T)
                ax.set_zlim3d(0, 500)
                plt.title('Spin 3D graph')
                plt.show()


def aceleration2():
        global gen
        if gen == 0:
                gen = 1
                print 'Aceleration in graph 2D was chosen'
                entrada2 = open('base.csv')
                tabla2 = []
                for fila2 in csv.reader(entrada2):
                        tabla2.append(fila2)
                entrada2.close()
                x2=[0]
                y2=[0]
```

```
                z2=[0]
                u2=[0]
                for fila2 in range(1, len(tabla2)):
                        x2.append(float(tabla2[fila2][0]))
                        y2.append(float(tabla2[fila2][2]))
                        z2.append(float(tabla2[fila2][3]))
                        u2.append(float(tabla2[fila2][4]))
                pl.plot(x2,y2,color="grey", linewidth=0.5, linestyle="-", label="acelerometer
                    X")
                pl.plot(x2,z2,color="m",  linewidth=0.5, linestyle="-", label="acelerometer Y"
                    )
                pl.plot(x2,u2,color="green",  linewidth=0.5, linestyle="-", label="
                    acelerometer Z")
                pl.legend(loc='upper left')
                pl.xlabel('Time[s]')
                pl.ylabel('Spin[m/s^2]')
                pl.title('Aceleration 2D graph')
                pl.grid(True)
                pl.show()

def aceleration3():
        global gen
        if gen == 0:
                gen = 1
                plt.close()
                pl.close()
                print 'Aceleration in graph 3D was chosen'
                fig = plt.figure()
                ax = fig.gca(projection='3d')

                entrada = open('base.csv')
                tabla = []
                for columna in csv.reader(entrada):
                        tabla.append(columna)
                entrada.close()
                T=[]
                X=[]
                Y=[]
                Z=[]
                for columna in range(1, len(tabla)):
                        T.append(float(tabla[columna][0]))
                        X.append(float(tabla[columna][2]))
                        Y.append(float(tabla[columna][3]))
                        Z.append(float(tabla[columna][4]))

                ax.scatter(X, Y, Z, marker='o', c='r',s=T)
                ax.set_zlim3d(0, 500)
                plt.title('Aceleration 3D graph')
                plt.show()
```

```
def horafecha():
        HOST = '192.168.0.154'     # The remote host
        PORT = 50007               # The same port as used by the server
        s = socket . socket ( socket . AF_INET , socket . SOCK_STREAM )
        s.connect((HOST, PORT))

        mensaje='sudo date --set "2018-11-27 17:05"'
        s.sendall(mensaje)
        data = s.recv(1024)

        s.close()
        print 'Set time'
def baseextraida():
        print 'Database was extracted'
        serv = ['192.168.0.154']        # The remote host
        usern = ['pi']
        passw = ['raspberry']
        remoteP = ["/home/pi/Desktop/datos.db"]
        localP = ["/home/valeria/datos.db"]

        def copyDataBase(Server,Username,Password,LocalPath,RemotePath):
                filesCopied = 0
                try:
                        print 'Establish connection SSH:', Username,'@', Server

                        warnings.filterwarnings("ignore")

                        ssh = paramiko.SSHClient()
                        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                        ssh.connect(Server, username=Username, password=Password)
                        sftp = ssh.open_sftp()

                        try:
                                if sftp.stat(RemotePath):
                                        local_file_data = open(LocalPath, "rb").read()
                                        remote_file_data = sftp.open(RemotePath).read()
                                        md1 = md5.new(local_file_data).digest()
                                        md2 = md5.new(remote_file_data).digest()
                                        if md1 == md2:
                                                print "Local file is the same as remote file:"
                                                    , os.path.basename(LocalPath), 'remains
                                                    unchanged'
                                        else:
                                                sftp.get(RemotePath, LocalPath)
                                                print LocalPath, "has been modified"
                                                filesCopied += 1
                        except:
                                sftp.get(RemotePath, LocalPath)
```

```
                                print 'Copying', RemotePath, 'to ', LocalPath
                                filesCopied += 1

                        try:
                                sftp.close()
                                ssh.close()
                        except:
                                pass

                except Exception, e:
                        print '*** Caught exception: %s: %s' %(e.__class__, e)
                        try:
                                ssh.close()
                        except:
                                pass
                return filesCopied

        totalFiles = 0
        for i in range(0,1):
                totalFiles += copyDataBase(serv[i],usern[i],passw[i],localP[i],remoteP[i])
        print 'Total files copied:',totalFiles
        print 'All operations complete!'


def db_csv():
        print 'Database convert spreadsheet '
        inpsql3 = sqlite3.connect('datos.db')
        sql3_cursor = inpsql3.cursor()
        sql3_cursor.execute('SELECT * FROM posiciones')
        with open('datos.csv','w') as out_csv_file:
                csv_out = csv.writer(out_csv_file)
                csv_out.writerow([d[0] for d in sql3_cursor.description])
                for result in sql3_cursor:
                        csv_out.writerow(result)
        inpsql3.close()

def grafica():
        print 'The trajectory was drawn'

        my_map4 = folium.Map(location = [0.340487,-78.135422], zoom_start = 16)
        entrad = open('base.csv')
        tab = []
        for colum in csv.reader(entrad):
                tab.append(colum)
        entrad.close()
        latitude_list=[]
        longitude_list=[]
        for colum in range(1, len(tab)):
                latitude_list.append(float(tab[colum][8]))
```

```
                longitude_list.append(float(tab[colum][9]))


        for i in range(len(latitude_list)):
                folium.CircleMarker(location = [latitude_list[i], longitude_list[i]], radius =
                    4, popup = ' FRI ').add_to(my_map4)

        lat1= latitude_list[2]
        lon1= longitude_list[2]
        lat2= latitude_list[1250]
        lon2= longitude_list[1250]
        folium.Marker([lat1, lon1], popup = 'Start').add_to(my_map4)
        folium.Marker([lat2, lon2], popup = 'Finish').add_to(my_map4)

        my_map4.save("my_map.html")
        webbrowser.open_new("file:///home/valeria/my_map.html")

        rad=math.pi/180
        dlat=lat2-lat1
        dlon=lon2-lon1
        R=6372.795477598
        a=(math.sin(rad*dlat/2))**2+math.cos(rad*lat1)*math.cos(rad*lat2)*(math.sin(rad*dlon
            /2))**2
        dis=2*R*math.asin(math.sqrt(a))
        print("Distance traveled in Km: %s" %dis)


def _quit():
        w.quit()
        w.destroy()
        plt.close()
        pl.close()

def limpiar():
        global gen

        gen = 0
        plt.close()
        x=[0]
        y=[0]
        pl.plot(x,y)
        pl.show()


w = Tk()
w.geometry('430x350')
w.configure(bg = 'beige')
w.title('Graphic Interface of Bicycle Naturalistic Variables')
print 'Welcome to Graphic Interface of Bicycle Naturalistic Variables'
```

```
L = Label(w, text='Data Management',bg = 'beige')
L.place(x=20, y=20)

L2 = Label(w, text='3D Graphics',bg = 'beige')
L2.place(x=20, y=200)

L3 = Label(w, text='2D Graphics',bg = 'beige')
L3.place(x=20, y=110)

b5 = Button(w, text='Speed', bg = 'light green', width=8, height=2, command=speed)
b5.place(x=20, y=140)

b6 = Button(w, text='Spin', bg = 'coral',width=8, height=2,command=spin2)
b6.place(x=120, y=140)

b6 = Button(w, text='Spin', bg = 'coral', width=8, height=2, command=spin3)
b6.place(x=20, y=220)

b7= Button(w, text='Acceleration', bg = 'CadetBlue', width=8, height=2, command=aceleration2)
b7.place(x=220, y=140)

b9= Button(w, text='Acceleration', bg = 'CadetBlue', width=8, height=2, command=aceleration3)
b9.place(x=120, y=220)

b8 = Button(w, text='Verify\nTrajectory', width=8, height=2, command=grafica)
b8.place(x=320, y=50)

b0 = Button(w, text='Quit', bg = 'PaleTurquoise3', width=5, height=1, command=_quit)
b0.place(x=330, y=300)

b1 = Button(w, text='Set\nTime', width=8, height=2,command=horafecha)
b1.place(x=20, y=50)

b2 = Button(w, text='Extract\nDatabase',width=8, height=2, command=baseextraida)
b2.place(x=120, y=50)

b3 = Button(w, text='Convert', width=8, height=2, command=db_csv)
b3.place(x=220, y=50)

b4 = Button(w, text='Clear\nScreen', bg = 'PaleTurquoise3', width=8, height=2, command=limpiar
    )
b4.place(x=20, y=290)

x=[0]
y=[0]
pl.plot(x,y)
pl.show()
```

```
w.mainloop()
```