Universidad de Cuenca

Facultad de Ingeniería

Carrera de
Electrónica y Telecomunicaciones

Tecnologías IoT aplicadas a la monitorización de ciclovías en la ciudad de Cuenca

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autores :

Eduardo Alejandro Lima Huayllas — C.I. 010572420-7
eduardo.limah16@gmail.com

Guido Fernando Parapi Patiño — C.I. 010485274-4
guidofernandoparapi@gmail.com

Director :

Ing. Santiago Renán González Martínez, PhD — C.I. 010389593-4

Cuenca - Ecuador

12 de octubre de 2020

Resumen

El presente trabajo de tesis se enfoca en analizar la problemática referente a los niveles de confort del ciclista. La monitorización se lleva a cabo utilizando las tecnologías *Internet of Things* (*IoT*). Esta propuesta tecnológica se centra en la recolección de datos referentes al estado físico y ambiental de las ciclovías, donde las variables a capturar son: el estado de la ciclovía, el nivel de contaminación ambiental de la zona, la capacidad de maniobra de la que dispone el ciclista y los niveles de ruido que percibe el mismo respecto al tráfico vehicular. El estado de la vía se analiza según los niveles de vibración que experimenta el ciclista durante la trayectoria. El nivel de contaminación ambiental se medirá usando el incremento en volumen en *ppm* de la presencia de Dióxido de carbono (*CO2*) y el volumen en $\mu g/m^3$ de la presencia del material particulado fino y grueso. Cabe indicar que las primeras corresponden a partículas generadas por procesos mecánicos como obras de construcción, la suspensión del polvo y el viento, mientras que las segundas son partículas generadas por cualquier fuente de combustión, estas son las más peligrosas para la salud de las personas. Finalmente La capacidad de maniobrar de la que dispone el ciclista, se mide usando la separación entre la bicicleta y un posible obstáculo lateral (v.g. vehículos, personas, paredes etc).

En concreto nuestra propuesta tecnológica consiste en la creación de una estación móvil, la cual integrará diferentes sensores para capturar las variables mencionadas. La estación dispondrá de conexión a Internet usando la red móvil, permitiendo de esta manera dar seguimiento a los datos capturados por el ciclista y la posición del mismo en tiempo real, así como también almacenar los datos registrados durante toda la ruta. Además, localmente se guardará un vídeo por cada recorrido que posteriormente se podrá transferir a un dispositivo de almacenamiento externo liberando la memoria de la estación, con este sistema se brindará un respaldo de seguridad ante posibles eventos adversos que pueda sufrir el ciclista. Adicionalmente, se implementa dos tipos de aplicación que permiten visualizar e interactuar con los datos registrados por la estación, siendo una de ellas una aplicación móvil nativa para dispositivos con sistema operativo *Android* y otra una aplicación web multiplataforma. En cada una de ellas se podrá visualizar los usuarios activos en tiempo real y los valores capturados por los mismos, así como también, permiten la revisión de todas las rutas almacenadas y sus respectivos valores asociados.

Finalmente, se muestra un análisis del rendimiento de la estación en la captura de datos del entorno, los valores de carga y temperatura del procesador central, así como también el consumo energético que utiliza la estación. Por otro lado, también se analiza los parámetros correspondientes a la red celular que afectan en la trasmisión de datos, siendo estos el *throughput*, el retardo y la pérdida de datos por intermitencia de la red. El rendimiento de la estación se analiza en varias rutas integradas por diferentes tipos de ciclovías (compartida, segregada, integrada, reservada y ciclo-vereda).

Palabras clave: Ciudad inteligente. Movilidad sostenible. Ciclismo. Ciclovías. Monitorización. IoT. Sensores. Tiempo real

Abstract

The present thesis focuses on analyzing the problem regarding comfort levels of the cyclist. The monitoring is carried out using the *IoT* technologies. This technological proposal focuses on the collection of data regarding the physical and environmental state of bicycle lanes, where variables to be captured are: the state of the bike lane, the level of environmental pollution in the area, the ability to maneuver available to the cyclist and the noise levels perceived by the cyclist regard to vehicular traffic. The state of the bike lane will be analyzed according to the vibration levels that experiences the cyclist during the trajectory. The level of environmental pollution in the area will be measured using the increase in volume in *ppm* of *CO2* which is one of the gases with the greatest presence in the emissions of internal combustion vehicles and the volume in $\mu g/m3$ of the presence of the particulate material with diameter aerodynamic between 2.5 microns and 10 microns and less than 2.5 microns. It should be noted that the first correspond to coarse particles generated by mechanical processes such as construction sites , the suspension of dust and wind, while the latter are fine particles generated by any combustion source, these are the most dangerous for the health of the person. The ability to maneuver available to the cyclist will be measured by the separation between the center frame of the bicycle and a possible lateral obstacle (e.g. vehicles, people, walls, other cyclists etc).

Specifically, our technological proposal consists whit creation of a mobile station, which It will integrate different sensors to capture the mentioned variables. The station will have connection to the Internet using the mobile network, allowing in this way to track the data captured by the rider and his position in real time, as well as storing the recorded data throughout to whole route. In addition, a video will be saved locally for each route that later can be transferred to an external storage device freeing the station memory, with this system a security backup will be provided against possible adverse events that the cyclist could suffer from. Additionally, two types of application are implemented that allow you to view and interact with the data recorded by the station, one of them being mobile application for devices with Android operating system and another a multiplatform web application. In each of them you can view active users in real time and the values captured by cyclist, as well as also, they allow the review of all stored routes and their respective associated values.

Finally, an analysis of the behavior of the station is shown, in terms of performance in data capture for each of the variables; processor load and temperature settings central, as well as the energy consumption used by the station. On the other hand, it is also analyzes the parameters corresponding to the mobile network that affect the transmission of data, being these are the throughput, delay and data loss due to intermittent network. The performance of the station is analyzed in several routes integrated by different types of bike lanes (shared, segregated, integrated, reserved and cycle-path).

Keywords: Smart city. Sustainable mobility. Cycling. Bike lanes. Monitoring. IoT. Sensors. Real time

Índice general

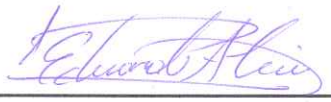
Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Yo, Eduardo Alejandro Lima Huayllas, autor de la tesis "Tecnologías IoT aplicadas a la monitorización de ciclovías en la ciudad de Cuenca", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 12 de octubre de 2020

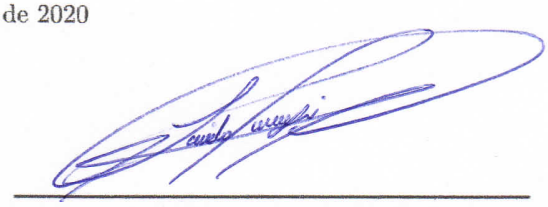


Eduardo Alejandro Lima Huayllas
010572420-7

Cláusula de Propiedad Intelectual

Yo, Guido Fernando Parapi Patiño, autor de la tesis "Tecnologías IoT aplicadas a la monitorización de ciclovías en la ciudad de Cuenca", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 12 de octubre de 2020



Guido Fernando Parapi Patiño
010485274-4

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Eduardo Alejandro Lima Huayllas en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Tecnologías IoT aplicadas a la monitorización de ciclovías en la ciudad de Cuenca", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 12 de octubre de 2020

Eduardo Alejandro Lima Huayllas
010572420-7

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Guido Fernando Parapi Patiño en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Tecnologías IoT aplicadas a la monitorización de ciclovías en la ciudad de Cuenca", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 12 de octubre de 2020

Guido Fernando Parapi Patiño
010485274-4

Dedicatoria

Este trabajo de titulación se la dedico a mis padres Ángel y Leonila pilares fundamentales de mi vida, quienes con su amor y sacrificio me han llevado a cumplir mis metas. A mi tío Luis quién nunca dejo de apoyarme. A mis hermanos, primos y familiares que siempre estuvieron presentes con sus consejos y aliento.

Eduardo Alejandro Lima Huayllas

Dedicatoria

Este trabajo de titulación se lo dedico a mi padre, quien fue un ejemplo de dedicación, esfuerzo y superación, además de ser la primera persona en enseñarme muchas de las duras lecciones de la vida e impulsar mi pasión por la ingeniería. A mi madre por sus consejos, su apoyo incondicional y su paciencia, quien, junto con mi padre, me ha dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, todo esto de manera desinteresada y llena de amor. A mis hermanos y demás familia en general por el apoyo que me brindaron siempre durante el transcurso de cada año de la carrera.

Guido Fernando Parapi Patiño

Agradecimientos

Queremos expresar nuestro agradecimiento a todas aquellas personas que durante el trayecto de la carrera nos han apoyado de una u otra manera.

También agradecemos a todos los docentes quienes nos han formado con sus conocimientos, en especial queremos agradecer a nuestro director de tesis el Dr. Santiago González, quien ha sido parte fundamental en el desarrollo de este proyecto.

Finalmente, queremos agradecer a la Universidad de Cuenca por todo el apoyo y darnos la oportunidad de ser parte de su comunidad estudiantil.

LOS AUTORES

Abreviaciones y Acrónimos

ADC Convertidor Analógico a Digital. 33, 59, 64
AMQP *Advanced Message Queuing Protocol*. 31
API *Application Programming Interface*. 31, 32, 40, 42, 46, 74, 78, 106, 165
APS Alimentación Por Suelo. 94
AWS *Amazon Web Services*. 29, 31
CDN *Content Delivery Network*. 72, 78
CE *Chip Enable*. 54, 59
CO2 Dióxido de carbono. 1, 2, 26, 45, 103, 105
CPU *Central Processing Unit*. 23, 56, 100, 104
CSI *Camera Serial Interface*. 54
CSS *Cascading Style Sheets*. 42, 78
CS *Chip Select*. 37
DSA *Debt Sustainability Analyses*. 31
EDGE *Enhanced Data Rates for GSM Evolution*. 41
ESH *Eclipse SmartHome*. 31
GPIO *General Purpose Input Output*. 4, 53, 54, 63, 64, 66, 68, 84, 85, 107
GPRS *General Packet Radio Service*. 41, 45
GPS *Global Positioning System*. 5, 33, 44, 45, 51, 65–67, 83, 84, 131
GSM *Global System For Mobile Comunication*. 41
HSPA *High-Speed Packet Access*. 41
HTML *HyperText Markup Language*. 42
HTTP *Hypertext Transfer Protocol*. 42
I2C *Inter Integrated Circuits*. 4, 8, 36, 37, 43, 53, 60, 61
IMS *Internet Media Services*. 41
IoT *Internet of Things*. 1–4, 8, 24, 27–32, 39, 43, 50
JVM *Java Virual Machine*. 31
LTE *Long Term Evolution*. 41, 45, 65, 98, 108
MISO *Master In-Slave Out*. 37, 54, 59
MOSI *Master Out-Slave In*. 37, 54, 59
MP Materia Particulada. 4, 8, 48, 49, 55, 56, 89
MQTT *Message Queing Telemetry Transport*. 31, 46
MVC Modelo Vista Controlador. 39, 40, 43
NPM *Node Package Manager*. 40
OMS Organización Mundial de la Salud. 89, 105

PAC Presentación Abstracción Control. 39
PCB Placa de Circuito Integrado. 5, 8, 10, 68, 69, 85, 121
PWM *Pulse Width Modulation*. 54, 56
REST *Representational state transfer*. 4, 31, 32, 40, 43, 78
SBC *Single Board Computer*. 4, 5, 8, 10, 11, 51–54, 56, 58, 60, 61, 63, 66, 84, 122
SCLK *Serial Clock*. 37, 54, 59
SCL *Serial Clock*. 36, 60
SDA *Serial Data Analizer*. 36, 60
SOA *Service Oriented Architecture*. 31
SPI *Serial Peripheral Interface*. 4, 8, 37, 43, 53, 54, 59, 61
SQL *Structured Query Language*. 29, 41, 46, 72, 78
SSL *Secure Sockets Layer*. 72
TIC Tecnologías Información y Comunicación. 24, 43
UART *Universal Asynchronous Receiver-Trasnsmitter*. 4, 38, 54, 56, 65, 66
UE *User Equipment*. 98
UMTS *Universal Mobile Telecommunications System*. 41
URI *Uniform Resource Identifier*. 40, 43
URL *Uniform Resource Locator*. 78, 80, 106
WSN *Wireless Sensor Network*. 4, 24, 32–35, 46
eNodeB *envolved Node B*. 98

Capítulo 1

Introducción y Objetivos de la Tesis

1.1. Identificación del problema

El ciclismo es una importante alternativa de movilidad que tiene un alto valor social, ambiental y recreativo, además de ser un componente clave para un sistema de transporte sostenible [12]. En la ciudad de Cuenca los gobiernos de turno han promovido cada vez más el uso de la bicicleta como alternativa de movilidad, proporcionando infraestructura para el ciclismo y fomentando el mismo con la implementación de servicios de bicicletas públicas. A pesar de ello, el uso de bicicletas en la ciudad de Cuenca, se ve limitado por la falta de conectividad entre las ciclovías y el estado de las mismas. Otro factor a tener en cuenta es la inseguridad que sufren los ciclistas al utilizar caminos compartidos con vehículos y peatones provocando que la percepción de confort se vea afectado por parámetros tales como, la exposición a rutas cubiertas de polvo, el mal estado de las diferentes estructuras de ciclovías, los niveles de ruido generado por el tráfico vehicular, el incremento de exposición a gases contaminantes en rutas de alto tráfico y la poca capacidad de movilidad que se puede encontrar por lo general en las rutas compartidas con vehículos.

.

1.2. Justificación

En la actualidad en la ciudad de Cuenca no existe un método efectivo para monitorizar los diferentes parámetros que afectan la percepción del confort que experimenta un ciclista durante el recorrido de una ruta. Esto ha provocado que los intentos por promover dicho medio de transporte no cuente con una mayor aceptación y difusión entre los ciudadanos. Por tal motivo se prefiere alternativas como el uso de vehículos o autobuses, incluso para realizar trayectos cortos. En tal contexto, en este trabajo de tesis se propone implementar una estación móvil con capacidad de adquisición de datos en tiempo real, cuya importancia radica en monitorizar las principales características que afectan la percepción de confort de un ciclista, siendo estas, los niveles de contaminación ambiental; el volumen de exposición

a material particulado que puede llegar a ser nocivo para la salud de las personas, los niveles de contaminación auditiva, la capacidad de maniobra; el estado de la vía; y la contaminación visual a la que el ciclista se enfrenta durante toda la trayectoria. La información capturada esta dirigida a la propia comunidad ciclista, la mismas que se beneficiará con la identificación de las zonas con los mejores y peores resultados. Adicionalmente, la información capturada resulta de gran utilidad para la mejora y la planificación de nuevas rutas.

1.3. Alcance

En este trabajo de tesis, se implementará una estación con capacidad de ser instalada sobre una bicicleta sin afectar su normal operación. La estación tiene por objetivo monitorizar parámetros de interés durante el desplazamiento en las ciclovías de la ciudad, parámetros tales como el confort de los ciclistas, tracking (seguimiento) de las rutas empleadas, monitorización ambiental de las rutas y aspectos de seguridad. La estación tendrá como función la recolección de datos utilizando diferentes tipos de sensores, dentro de los que podemos encontrar, sensores que miden la calidad del aire, el nivel de ruido, la velocidad, la distancia entre un vehículo y la bicicleta, así como también se implementará una cámara que permita registrar el trayecto del usuario por motivos de seguridad. Los datos adquiridos serán almacenados, empleando servicios *cloud* y se habilitará la conectividad de la estación utilizando las redes móviles (3G y 4G) desplegadas por los operadores locales. En cuanto a la visualización de los datos, se plantea el desarrollo de una aplicación para dispositivos *Android*, donde los usuarios puedan consultar sus datos más relevantes. Adicionalmente, se propone el desarrollo de una aplicación web, para la monitorización de todos los nodos. Para el trabajo de tesis, se implementarán dos estaciones. De esta manera el objetivo del presente proyecto es el diseño e implementación de un dispositivo registrador de datos, con capacidad de transmisión en tiempo real. El dispositivo permitirá la recolección de datos de niveles de contaminación ambiental, sonoro, velocidad, distancia de separación vehículo-bicicleta y la ruta seguida.

1.4. Objetivos

1.4.1. Objetivo general

Implementar una solución tecnológica para la monitorización de variables de confort, seguridad, seguimiento del trayecto y estado de las ciclovías.

1.4.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Diseñar e implementar una estación con capacidad de integrarse en la estructura mecánica de una bicicleta.
- Integrar sensores en la estación con capacidad de capturar variables de posición, parámetros ambientales y seguridad de los ciclistas.
- Desarrollar un sistema de gestión y/o aplicación para la visualización y análisis remoto de los datos capturados por la estación con acceso desde Internet.

- Desarrollar experimentos de campo en las ciclovías de la ciudad para la captura de las variables de interés.

1.5. Estructura del Documento y Contribuciones

El documento esta estructurado por seis capítulos. En el Capítulo 2, se presenta el sustento teórico de las principales herramientas, conceptos y tecnologías que se utilizarán para la implementación. Así también, en el Capítulo 3, se realiza una revisión bibliográfica de los trabajos que abordan una problemática similar y plantean diferentes soluciones planteadas. Los trabajos relacionados son extraídos de diferentes revistas tendiendo en consideración las soluciones más actuales. Por otro lado, en el Capítulo 4, se presenta la arquitectura del sistema, donde se muestra los componentes que se usarán para la implementación de los nodos, el diseño físico del nodo para integrar en la bicicleta y los elementos de comunicación usados para la transmisión de datos, así también se presenta el desarrollo de las aplicaciones para la monitorización de los datos. En el Capítulo 5, se detallan las pruebas realizadas y los resultados obtenidos, dentro de los que podemos encontrar variables como el *throughput* o tasa de transferencia efectiva, la pérdida de datos por intermitencia de la red, el retardo, la carga del *Central Processing Unit (CPU)* entre otras variables. En el Capítulo 6, se presenta las principales conclusiones obtenidas al desarrollar este proyecto de tesis en función de los objetivos planteados. Finalmente en los Anexos se incluye los códigos de los todos los programas realizados, así como, el registro de los fotogramas de vídeo de las diferentes ciclovías analizadas donde se puede apreciar los principales eventos que afectan en el confort del ciclista.

Capítulo 2

Marco teórico

2.1. Introducción

En este capítulo se presenta el sustento teórico del trabajo de titulación. El presente trabajo tiene como propósito contribuir al desarrollo tecnológico de la ciudad. En este contexto en la sección 2.2 se describe el concepto de Ciudad inteligente (*Smart City*) definiendo sus características, elementos y sobre todo haciendo énfasis en la movilidad urbana.

Dentro de las *Smart City* se encuentra integrada el *IoT*, que ha tenido un enorme crecimiento en los últimos años y se espera que crezca aún más. Cuando se habla de *IoT* básicamente se refiere a millones de sensores u objetos conectados a Internet. En la Sección 2.3 se explica los conceptos básicos y se describe la importancia de los *frameworks* en el desarrollo de aplicaciones para el *IoT*. Además, se indica algunas plataformas *software* disponibles para el manejo de la información obtenida a través de dispositivos *IoT*.

Para empezar con el desarrollo del trabajo es necesario describir la estructura base para comenzar a construir la estación, para esto, en la Sección 2.4 se muestra las *WSN* que tienen como función obtener datos a través de sensores y transmitirlos para su interpretación hacia alguna estación base o a Internet. Por otra parte, debido a la diversidad de sensores que se plantea implementar es necesario realizar una revisión de los protocolos que utilizan distintos sensores para comunicar los datos obtenidos hacia un elemento central, el cual se describe en la Sección 2.5.

Por último, en la Sección 2.6, se detallan las aplicaciones desarrolladas para la monitorización y visualización de los datos.

2.2. *Smart Cities*

El concepto *Smart City* engloba un amplio campo de investigación y desarrollo orientado a la inclusión de las Tecnologías Información y Comunicación (*TIC*) para mejorar la dinámica y fluidez de una ciudad. Para solventar sus necesidades se requiere plantear diferentes esquemas de acción, sin

embargo, los ejes en los que suele incidir un proyecto de *Smart City* tiene que ver con la movilidad urbana, la eficiencia energética y en general la gestión sostenible de los recursos [13]. Estos ejes de desarrollo permiten a las *Smart Cities* manejar adecuadamente el incremento de población en las áreas urbanas. Cabe recalcar, que para el año 2050 se prevee que el 70 % de la población mundial vivirá en zonas urbanas [14], por lo tanto, tendrán que enfrentar grandes retos en cuanto planificación, administración y gobernanza de ciudades cada vez mas pobladas.

Las *Smart Cities* no deben ser vista como proyectos futuros, más bien son realidades que las vivimos hoy gracias al constante desarrollo de las redes de banda ancha, las cuales permiten la recolección de datos de redes de sensores, el desarrollo de aplicaciones móviles, medios sociales y portales web.

Por citar alguno ejemplos, en Europa se lleva algunos años trabajando en el desarrollo de las ciudades inteligentes plateando aplicaciones como:

- El uso de sensores distribuidos para optimizar las operaciones de la ciudad
- Paradas de autobús que ofrecen información en tiempo real.
- Sistemas de redes de iluminación pública que manipula el nivel de corriente para ahorrar energía.
- Contenedores de basura que ofrece datos para mejorar las rutas de recolección.
- El manejo de cámaras de seguridad para detectar acciones sospechosas y prevenir delitos, etc...

Todas estas tecnologías permiten que una *Smart City* sea sostenible, inclusiva, transparente y genere riquezas, estas características mejoran la calidad de vida de las personas y dan acceso rápido a servicios públicos más eficientes.

2.2.1. Elementos de una *Smart City*

Existen diferentes elementos que conforma una *Smart City* dependiendo del tipo de ciudad, la población, la arquitectura y las tecnologías. En este punto nos centraremos en describir los cuatro elementos básicos de los que debe disponer una *Smart City* desde un enfoque tecnológico, independientemente de la aplicación.

- Interfaces de comunicación: Este elemento consiste en el uso de los servicios, portales web y aplicaciones móviles, con el fin de trasmitir y recibir información que pueda ser usada para mejorar la gestión y transparentar los datos de la estructura publica.
- Centro de operaciones y control: Procesa los datos recolectados por los sensores mediante el uso de aplicaciones de *software* que permiten mostrar de manera amigable los resultados.
- Sensores y dispositivos conectados: Son los equipos físicos que permiten captar las diferentes señales del medio ambiente y trasformarlas a valores digitales para luego ser transportadas y procesadas.
- Infraestructura de conectividad: Comprende las redes de Internet de banda ancha necesarias para recibir y enviar datos, se debe mencionar que estas redes deben brindar la capacidad de soportar la trasmisión de datos en tiempo real.

2.2.2. Modelo de capas de una *Smart City*

El modelo de capas para mejorar la administración de una *Smart City*, propuesta en [15] por ®Cisco, una de las empresas lideres en telecomunicaciones, esta compuesto por 4 capas que desarrollan las siguientes funciones:

- *Street Layer*: En esta capa se colocan dispositivos y sensores en varias partes de la ciudad para recopilar datos y tomar acciones automatizadas u ordenadas que resultan de los datos analizados. Los sensores utilizados dependerán de la ubicación y la función que se espera de ellos.
- *City Layer*: Esta capa es la encargada de permitir que los dispositivos conectados intercambien datos, para ello maneja los enrutadores y conmutadores de red así como con los protocolos de comunicaciones usados entre ellos.
- *Data Center Layer*: Esta capa se encarga de almacenar los datos procesados para su posterior uso en la capa de servicios, cabe indicar que, dicha capa juega un papel importante en la nube al proporcionar almacenamiento y potencia de procesamiento.
- Capa de servicios: Esta capa se encarga de proporcionar los resultados de los datos recolectados a las aplicaciones que los utilizan. Por ejemplo, una herramienta de visualización para mostrar el estado del tráfico en tiempo real en la ciudad.

2.2.3. Movilidad urbana sostenible

Uno de los grandes retos de los países de América Latina para transformar sus ciudades en *Smart City*, es la capacidad de administrar la movilidad, sector donde se proyecta que para 2025 se cuente con 140 millones de vehículos [14]. Esto contribuiría al aumento del congestionamiento vehicular, mayor probabilidad de accidentes de tráfico e incremento de emisiones en gases contaminantes y de efecto invernadero.

Uno de los mayores problemas de las ciudades tradicionales es la congestión vehicular que tiene un impacto negativo en la calidad de vida, reduciendo la productividad, la calidad del aire y aumentando la contaminación acústica. Según [13], en las grandes ciudades este impacto ronda entre el 1.4 % y 4 % del PIB, generando pérdidas de hasta 78.000 millones de dólares por las 4.200 millones de horas pérdidas en ellas y los 11.000 millones de litros de combustible gastados. En un entorno urbano solo para conseguir aparcamiento se efectúa un gasto de 178.000 litros de combustible y se genera 730 toneladas de *CO2*, el equivalente a 38 viajes alrededor de la tierra. En una *Smart City*, se puede solucionar esta problemática con diferentes tipos de sistemas que hacen uso de las redes de sensores para mejorar la movilidad, sistemas como:

- Gestión de tráfico en tiempo real.
- Gestión de medios de transporte público.
- Gestión de sitios de aparcamiento.
- Fomentar el uso de bicicletas.
- Pago de peajes.
- Servicio de vehículo compartido, entre otros.

De todas las soluciones expuestas, el uso de la bicicleta como medio de trasporte alternativo es uno de los métodos mas usados por las ciudades en proceso de trasformación de ciudad tradicional a *Smart City*. Debido a que es una solución simple y efectiva que no incluye el uso de un vehículo lo que genera grandes beneficios en cuanto a reducción de emisiones de *CO2*, mejorando así la calidad de vida de las personas y creando un entorno más habitable.

2.2.4. Movilidad en la ciudad de Cuenca

En la ciudad de Cuenca el estudio realizado en el PMEP (Plan de Movilidad y Espacios Públicos) de 2015 presentado en [16], establece que el 69 % de los viajes con origen y destino al interior de la ciudad se realizan utilizando vehículos motorizados. Mientras que el 31 % corresponde a peatones y ciclistas. El PMEP realizó 1023 encuestas, donde el 1.6 % indicó la bicicleta como medio de transporte principal, mientras que el 83.5 % indicó algún medio de transporte motorizado como principal medio de movilidad. Sin embargo, un 58 % de los actuales usuarios de auto estarían dispuestos a dejar este medio de transporte por uno más amigable con la ciudad y el ambiente. Las principales razones por la que las personas dejarían de usar el vehículo como principal medio de transporte, son los beneficios en la salud, la reducción de problemas de tráfico y el ahorro económico que representan otros modos de movilidad como el uso de bicicletas.

Del mismo modo en el PMEP se muestra que 53 % de los hogares disponen de una bicicleta, sin embargo, únicamente la cuarta parte la utiliza ocasional o regularmente. Pese a estos antecedentes el ciclismo no ha ganado fuerza como medio de movilidad en la ciudad. Debido a los problemas relacionados con la falta de respeto existente por conductores de vehículos motorizados frente a los ciclistas, la poca o nula seguridad de este medio de transporte contra robos y peligros que representa al utilizarlo en zonas donde no existe estructura orientada al ciclista. Para hacer frente a estos problemas en junio del 2020 el concejo cantonal de la ciudad de Cuenca emitió la ordenanza [17], la misma que contiene varios artículos orientados a fomentar el uso de la bicicleta como principal medio de movilidad por la población. Para este propósito se usa diferentes metodología como incrementar el número de bicicletas publicas, el número de ciclovías, mejorar la señalización, aumentar el presupuesto para el análisis de la movilidad, incorporar áreas de estacionamiento en entidades publicas y privadas, entre otras actividades que impulsan el ciclismo en la ciudad.

Finalmente, se debe mencionar que la ciudad de Cuenca hoy en día no dispone de una metodología para monitorizar el estado de las ciclovías y las principales razones que afectan la percepción del confort del ciclistas al realizar un recorrido. Así también, no dispone de un método efectivo para monitorizar el comportamiento del ciclista frente a estas adversidades. Una metodología efectiva para monitorizar el confort que percibe el ciclista permitiría generar nuevas rutas en función de las necesidades y comportamientos de los ciclistas.

2.3. Tecnología *IoT* y Servicios de *Frontend*

Debido al rápido crecimiento del *IoT*, hoy en día existe diferentes puntos de vista para entender esta tecnología, sin embargo, se puede definir *IoT* como una red integrada de sensores que permiten capturar datos del mundo real con la finalidad de integrarlos en el mundo digital de la Internet. *IoT* se puede entender además, como un sistema integrado de cuatro elementos fundamentales que se describen a continuación.

- *Things*.- Este elemento engloba todos los dispositivos físicos con capacidad de conexión a Internet que se usan en una red *IoT* para interactuar con el mudo real (sensores y actuadores).
- *Date*.- Los datos son todos valores digitales capturados del mundo real mediante el uso de los sensores, así también son todos los comandos usados para accionar el funcionamiento de los actuadores.

- *People.*- Las personas se pueden entender como actuadores de la red y beneficiaros de los datos recolectados. Las personas deben interactuar entre sí para dar sentido a los datos adquiridos.
- *Process.*- Este elemento permite generar resultados de las lecturas y acciones realizadas en la red *IoT*, con la finalidad que los beneficiarios de la red puedan mejorar el control y toma de decisiones al brindar la información correcta.

2.3.1. Impacto, Beneficios y Retos del *IoT*

Acorde al estudio descrito en [15] se estima que en 2020 se contará con 50 mil millones de cosas conectadas a Internet. Por consiguiente, con tal cantidad de elementos, la forma que interactuamos con tales objetos permite que la toma de decisiones sea más eficiente fundamentada en análisis de los datos recolectados. En la Figura 2.1, se presenta algunos de los diferentes elementos de *software* y *hardware* que conforman los ecosistemas *IoT*, los cuales interactúan y se complementan para generar un entorno enriquecido de datos que permita innovar nuevos servicios, incrementando las fuentes ingresos y mejorando la calidad de vida de los ciudadanos.

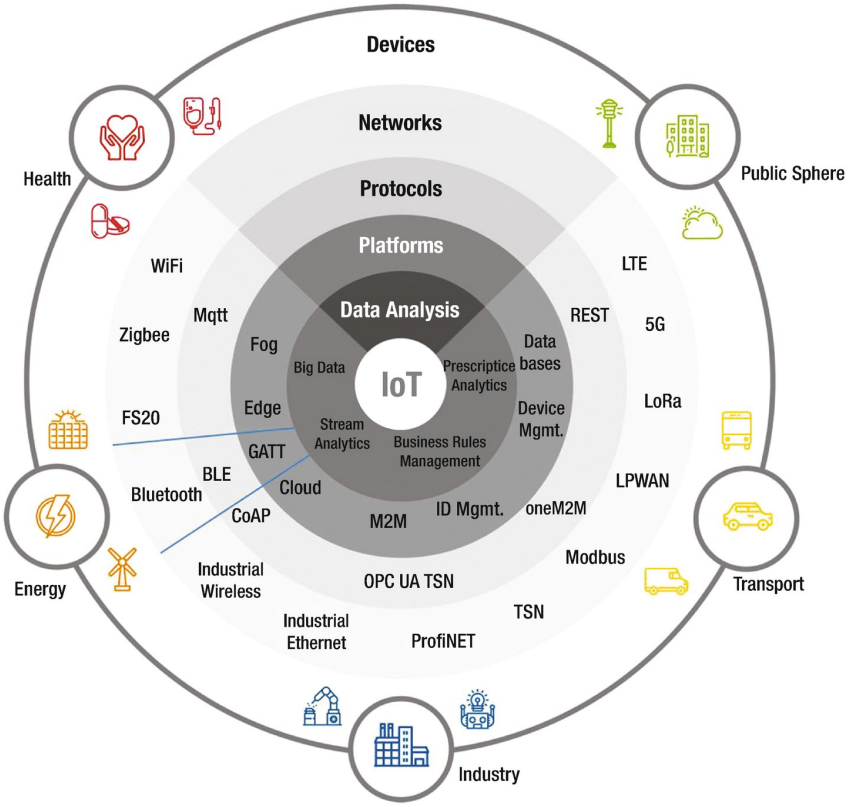


Figura 2.1: Ecosistema *IoT* [1]

Algunos de los beneficios que ofrece los servicios integrados de un ecosistema *IoT* son:

- Seguridad y confort de los usuarios.
- Eficiencia en manejo de equipos e instrumentos.
- Transparencia en los datos.
- Automatización y control de procesos.

- Exactitud en los datos.
- Monitoreo de entornos empresariales y no empresariales.
- Información en tiempo real.
- Optimización de costos.
- Conciencia ambiental, etc.

Por otro lado los principales retos con los que un sistema *IoT* debe lidiar son: la escalabilidad, la heterogeneidad, la privacidad de los datos, la seguridad, la intemporalidad, ect.

2.3.2. *IoT* y *Big Data*

Internet es una red que cuenta con miles de millones de dispositivos conectados de forma simultánea solicitando y enviado datos. Estas grandes cantidades de datos generan una necesidad de mejorar el tratamiento de los mismo buscando la manera de no congestionar la red. Como una alternativa para mejorar el manejo de datos surge la *Big Data*, que se define como los resultados de analizar grandes flujos de datos, donde estos pueden ser estructurados, no estructurados y semi estructurados. Adicionalmente en [15] se define *Big Data* como el sistema que implementar las tres V (Volumen, Velocidad y Variedad). Existe diferentes enfoques para tratar los datos dentro de los que podemos encontrar:

- *Analizing date*: Transforma el análisis manual por análisis automatizados.
- *Machine learning*: Consiste en un modelo de aprendizaje automático con datos almacenados o en tiempo real.
- *Edge analytics*: Se enfoca en analizar los datos cerca de los nodos para reducir el tráfico de la red.
- *Real-time analytics*: Mejoran el tiempo de respuesta en aplicaciones donde el retraso reduce el valor de los datos.
- *Distributed analytics*: Permite manejar grandes cantidades de datos distribuyendo la carga en varios nodos.

2.3.3. *IoT* y *Cloud Computing*

El almacenamiento de datos es otro problema desafiante en la era de los grandes datos. Las técnicas tradicionales (bases de datos *Structured Query Language (SQL)*) generalmente no son factibles debido a la cantidad de datos que se almacenan, con su naturaleza variada y a menudo no estructurada. Las bases de datos *NoSQL* ofrecen alto rendimiento, baja latencia de almacenamiento y recuperación inmediata. En este caso, puesto que no existe un esquema, se permiten nuevos tipos de datos dinámicos. *Couch Base*, *Apache Cassandra*, *Apache Couched*, *MongoDB* y *Apache HBase (Hadoop)*, son ejemplos de *framework* que usan *NoSQL* con almacenamiento local. Por otro lado, *Cloudant* de IBM (una base de datos distribuida), *DynamoDB* de *Amazon Web Services (AWS)* y *Firebase* de *Google* son soluciones *NoSQL* con capacidad de almacenamiento y computación en la nube. Una base de datos de series temporales también puede ser una base de datos *NoSQL* o incluso una base de datos relacional. La indexación y las consultas se basan en marcas de tiempo en los datos. Algunos *framework* que utilizan bases de datos de series temporales son *InfluxDB*, *Prometheus* y *Graphite*. [15]

La nube se define como un grupo de servidores y computadoras conectados a través de Internet en una gran infraestructura distribuida, esta característica de la nube permiten que la computación en

la nube se muestre como una solución a la creciente demanda de almacenamiento y procesamiento. El concepto es ofrecer servicios a pedido a través de Internet logrando generar ventajas para los consumidores en costo de cómputo. Algunos de los servicios más importantes que ofrece la integración de *IoT* en el *cloud computing* son, administración del dispositivo, comunicación, agrupación de recursos, almacenamiento, procesamiento y respaldo.

2.3.4. Servicios de *Backend*

Los servicios de *backend* representa uno de los componentes más importantes en el desarrollo de aplicaciones *IoT*, dado que los mismos pueden ser implementados por los propios desarrolladores o integrados gracias a un *framework* de *IoT*. Es decir, una capa de *middleware* debajo de una o más aplicaciones de *IoT*, la cual representa una interfaz de aplicación orientada a la red, a través de la que interactúan los nodos [1]. Los *frameworks* pueden manejar diferentes tecnologías de comunicación y técnicas de transmisión, además de brindar capacidades de seguridad. Estos cumplen con cuatro objetivos principales, mejorar la capacidad de servicio, la confiabilidad, la mantenibilidad, reducir la complejidad de implementación y operación, mejorar la portabilidad e interoperabilidad y reducir el tiempo de desarrollo.

Un *framework* como *backend* sigue el modelo de tres capas que se muestra en la Figura 2.2, donde la capa datos de objeto define las estructuras de información que exponen los nodos y sus capacidades utilizando un lenguaje de definición de datos como *JavaScript*. La capa de interacción de nodos contiene semántica de mensajería y define las interfaces utilizadas para la interacción de los nodos. Finalmente la capa de abstracción de plataforma define los puntos de conexión lógica disponibles para los nodos de *framework*.

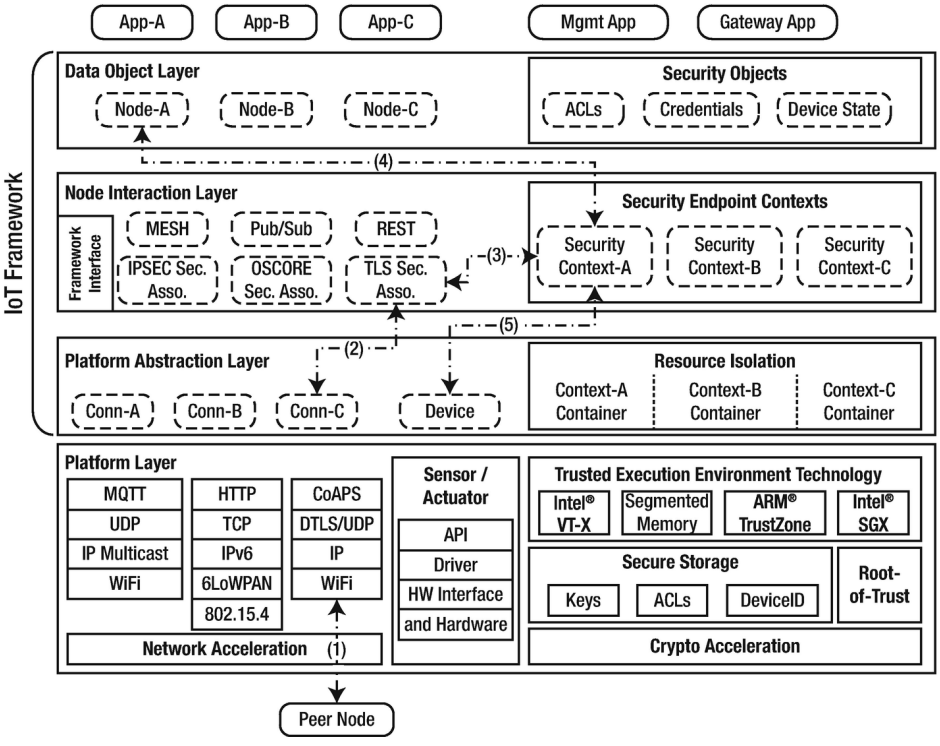


Figura 2.2: Arquitectura del *backend* [1]

A continuación se describe algunas de las principales plataformas que se puede usar como un *framework* de *backend* para dispositivos *IoT*.

- *FIWARE*: Plataforma de código abierto, donde cualquiera puede conectar dispositivos a un catálogo alojado en la nube. La idea es simplificar la tarea de integrar dispositivos en sistemas *IoT* y permitir una economía basada en datos.[15]
- *SmartThings*: Plataforma de *Samsung* que se enfoca en construir y ejecutar una *Smart Home* impulsada por *IoT*. El sistema de gestión de aplicaciones se utiliza para procesar suscripciones desde controladores. Más de 300 dispositivos diferentes son compatibles con el sistema.[15]
- *AWS IoT*: Plataforma de *Amazon* que está dirigido a usuarios domésticos y usuarios industriales con su combinación de *software* de dispositivo, control y servicios de datos.
- *Azure IoT*: Es una colección de servicios que es capaz de conectar, controlar y rastrear miles de millones de dispositivos *IoT*. Los servicios disponibles en *Microsoft IoT* incluyen [15]:
  - *Azure Internet of Things IoT Hub*
  - *AzureIoT Edge*
  - *Azure Stream Analytics*
  - *Azure Machine Learning*
  - *Azure Logic Apps*
- *SiteWhere*: Plataforma de código abierto, esta solución está diseñada como un sistema multi-inquilino en el que los inquilinos son responsables de la mayor parte de la lógica de procesamiento. *SiteWhere* admite comunicaciones *Message Queing Telemetry Transport (MQTT)*, *Advanced Message Queuing Protocol (AMQP)* y *REST* [18].
- *Eclipse SmartHome (ESH)*: Esta plataforma está diseñada para facilitar la resolución del sistema *IoT* y los problemas de los desarrolladores que se benefician de sus interfaces, reglas de automatización, mecanismo de persistencia y su implementación *Service Oriented Architecture (SOA)*. El *ESH* es un marco de conexión e integración para el dominio doméstico inteligente de *IoT* y es independiente de las características de conectividad del *hardware*.[19]
- *Zetta*: Plataforma orientada al servidor que se ha creado en torno a *Node.js*, *REST* y una filosofía de desarrollo de programación reactiva basada en flujo vinculada con la *Application Programming Interface (API) Hipermedia Siren*.
- *Thingspeak*: Plataforma que permite analizar y visualizar los datos en *MATLAB* y elimina la necesidad de comprar una licencia. Le ayuda a recopilar y almacenar datos de sensores en canales privados mientras le da la libertad de compartirlos en canales públicos. Se utiliza principalmente para el registro de sensores, el seguimiento de la ubicación y las alertas y análisis.
- *Debt Sustainability Analyses (DSA)*: Plataforma código abierto que unifica los dispositivos, servicios y aplicaciones separados en el modelo de datos estructurados y en tiempo real y facilita la comunicación, lógica y aplicaciones descentralizadas de dispositivos.
- *OpenHAB*: Plataforma capaz de ejecutarse en cualquier dispositivo que pueda usar *Java Virual Machine (JVM)*. La pila modular abstrae todas las tecnologías de *IoT* en elementos y ofrece reglas, *scripts* y soporte para la persistencia, la capacidad de retener los estados del dispositivo durante un período de tiempo. Ofrece una variedad de interfaces de usuario basadas en web y es compatible con las principales placas de *hackers* de *Linux*. Se implementa en las instalaciones y se conecta a dispositivos y servicios de diferentes proveedores.

- *Firebase*: A diferencia de las plataforma anteriores *Firebase* no se creó pensando en satisfacer las necesidades de los desarrolladores *IoT*, sin embargo, sus capacidades de integración como servicios de *backend* y uno de sus puntos claves el soporte de almacenamiento y consulta de datos en tiempo real, donde cualquier hardware que pueda comunicarse con su *API REST*, ha logrado convertir a esta plataforma en una de las más utilizadas para el desarrollo *IoT* de sistemas en los que no se requiere actualizaciones fuera de red ni herramientas de visualización de datos.

2.4. Redes de Sensores

Una red de sensores inalámbrica *WSN* (*Wireless Sensor Network*) es un conjunto de nodos sensores capaces de comunicarse entre si con el fin de trasmitir información obtenida del entorno en el que están ubicados. Estos nodos son dispositivos electrónicos que hacen seguimiento de un evento o capturan variables físicas presentes en el ambiente.

2.4.1. Nodos Sensores

Un nodo cumple con una función específica, obtener información del medio, con el fin de permitir la toma de decisiones, identificar la necesidad de aplicar un cambio, detectar un comportamiento anómalo, etc. El nodo puede ser configurado a nivel de *hardware* y *software*, es decir estos dispositivos cuentan con elementos de *hardware* que se adaptan a la aplicación donde se utilicen y cuentan con un código de programación, que les permite ejecutar la acción que se haya planeado [20].

La red de sensores puede estar compuesta por uno y varios nodos sensores, que se despliegan alrededor del área de seguimiento. La posición de los nodos sensores no necesita ser predeterminada o diseñada permitiendo un mejor despliegue. Sin embargo, para esto se requiere la implementación de protocolos y algoritmos capaces de realizar auto organización de los nodos [2].

2.4.2. Componentes de un nodo sensor

El nodo sensor básicamente tiene cuatro componentes, la unidad de detección, unidad de procesamiento, unidad de transceptor y unidad de potencia. Además, pueden contener componentes adicionales como un sistema de geolocalización, un generador de potencia y un movilizador. En la Figura 2.3, se muestra los componentes y la conexión entre unidades.

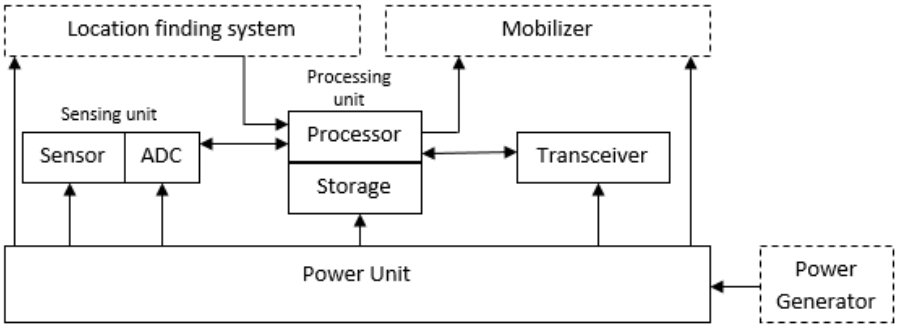


Figura 2.3: Los componentes de un nodo sensor [2]

- La unidad de detección esta formada por los sensores y en algunos casos contienen un Convertidor Analógico a Digital (*ADC*). Generalmente los sensores generan señales analógicas basadas en el fenómeno observado. La información obtenida por los sensores es enviada hacia la unidad de procesamiento. Sin embargo, en ocasiones los elementos de esta unidad solo admiten señales digitales por lo que es necesario el uso de un *ADC*. En este contexto, existen sensores que entregan señales analógicas y digitales haciendo mas fácil su integración con cualquier sistema [2].
- La unidad de procesamiento también denominado coloquialmente el cerebro electrónico de un nodo sensor, típicamente esta compuesta por un microprocesador y una memoria flash. Esta unidad almacena los datos de los sensores en la memoria hasta recopilar suficientes datos. Luego estos datos son empaquetados para su posterior transmisión. Por otro lado, la unidad gestiona procedimientos para comunicarse con otros nodos para mantener la estructura de red. Además, es la única unidad conectada con las demás para gestionar el correcto funcionamiento de cada una [21].
- La unidad de transceptor es la encargada de conectar al nodo a la red a través de una comunicación inalámbrica. Tiene principalmente cuatro estados operativos que son recibir, transmitir, inactivo y suspender [21].
- La unidad de potencia es la fuente de alimentación del nodo sensor y es la encargada de suministrar energía al los componentes. Generalmente es la mas critica en una *WSN*. La potencia se disipa en la detección, la comunicación, y el procesamiento de datos. La unidad de potencia mas común es una batería. En muchas aplicaciones, recargar o reemplazar la batería de miles de nodos es una tarea costosa que lleva mucho tiempo y no podría ser posible para los nodos implementados en un entorno hostil. Por lo tanto, es importante preservar la vida útil de la batería durante meses o años [22].
- En algunas aplicaciones se requiere el conocimiento de la ubicación del nodo, por lo tanto, es común que un sensor cuente con un sistema de localización, en muchos casos se implementa un *GPS* en el nodo sensor. Por otra parte, en ocasiones es necesario un movilizador para cambiar de posición a los sensores cuando sea requerido. Por ultimo un nodo sensor puede contener una fuente externa de energía como la solar, termoeléctrica y mecánica para recargar la batería del nodo.

2.4.3. Difusión de datos obtenidos por los nodos sensores

Cuando múltiples nodos sensores cooperan para la monitorización de un entorno, estos forma una red de sensores. Los nodos sensores aparte de comunicarse con los demás nodos, también lo hacen con una estación base usando comunicación inalámbrica. A la estación llegan los datos de los nodos permitiendo así realizar un procesamiento, visualización, análisis y almacenamiento de todo el conjunto de datos [3]. Además, de la estación base se puede realizar una conexión a Internet para una mayor difusión de los datos obtenidos. Por ejemplo, en la Figura 2.4, se muestra la comunicación de los datos de dos redes de sensores hacia la estación base y posteriormente a Internet para la interpretación de la información obtenida.

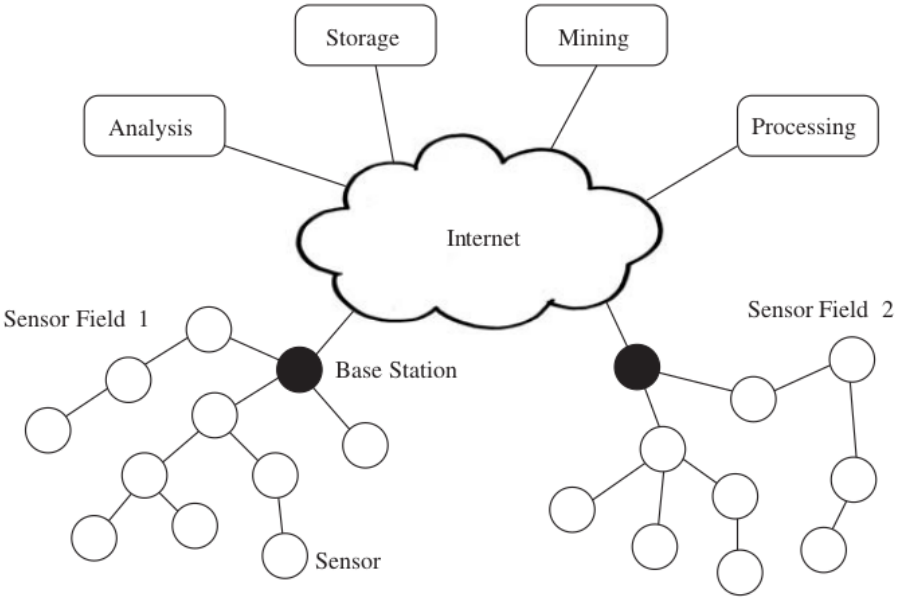


Figura 2.4: Difusión de datos en la red de sensores [3]

En muchas aplicaciones los nodos están ubicados a grandes distancias haciendo imposible la comunicación directa con la estación base. Además, dependiendo de la tecnología usada en la unidad de transceptor del nodo sensor se puede conocer el rango de comunicación inalámbrica. Cuando existe una comunicación directa de un nodo a la base se forma una topología en estrella, como se muestra en la Figura 2.5.

Las redes de sensores a menudo cubren grandes áreas geográficas y es necesario conservar la energía, sobre todo la utilizada para transmisión de radio. En este contexto, se puede aplicar la comunicación por saltos, la cual consiste en que los datos obtenidos por un sensor son enviados hacia un nodo sensor de tal manera que los datos sean enrutados hasta la estación base [3]. Esta topología indicada a la derecha de la Figura 2.5 es el caso mas común para las *WSN*.

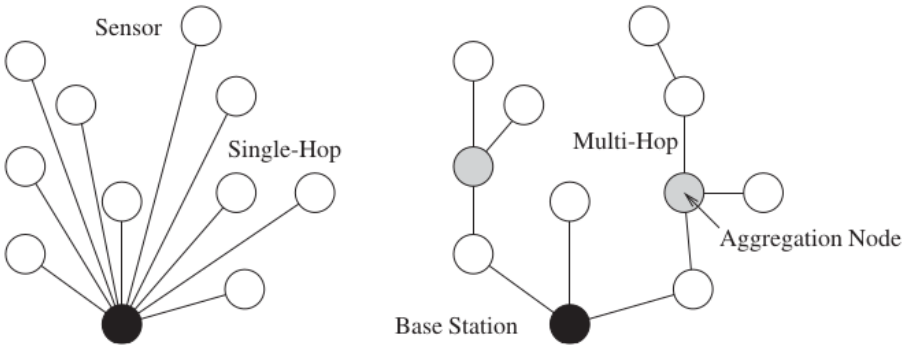


Figura 2.5: Topologías de una red de sensores [3]

2.4.4. Seguridad en las *WSN*

Un problema en las redes de sensores es la restricción de recursos en los nodos. Debido a ello, algunas aplicaciones funcionan sin seguridad, lo que disminuye la calidad de servicio. Los ataques más

comunes son los que consisten en la denegación de servicio. Si un nodo atacado continua intercambiando información y esto lleva a disminuir su rendimiento, entonces el nodo se declara como nodo muerto, que es el peor de los casos.

Un ataque puede ser realizado para obtener acceso ilegal a un servicio, información o el análisis de la integridad, la confidencialidad o la disponibilidad de un sistema. Los ataques pueden ser [23]:

- Pasivo.- Una persona u otra entidad que solo monitorea el canal de comunicación que amenaza la confidencialidad de los datos.
- Activo.- El atacante puede agregar, eliminar o alterar la transmisión en el canal que amenaza la confidencialidad, la autenticación y la integridad de los datos.
- *Insider*.- Persona con acceso a información que puede robar material clave y ejecutar código malicioso comprometiendo algunos nodos autorizados de la red.
- *Outsider*.- El atacante no tiene acceso particular a la red.
- Atacante de clase del nodo.- Tiene acceso a los nodos minoritarios con capacidades similares.
- Atacantes de clase portátil.- Tienen acceso a dispositivos potentes que tiene ventajas superiores a los nodos legales, por ejemplo, un procesador más capaz, una mayor batería y una antena de alta potencia.

Por otra parte, existen algunos requisitos mínimos que se debe cumplir en una red de sensores [23]:

- Autenticación de datos.- Asegurar que los datos se inicien desde una fuente exacta.
- Confidencialidad de los datos.- Asegurar de que solo los nodos sensores autorizados pueden obtener el contenido de los mensajes.
- Integridad de datos.- Asegurar que cualquier mensaje recibido no haya sido modificado al enviarlo por partes no autorizadas.
- Disponibilidad.- Asegurar de que los servicios ofrecidos por una WSN o por un solo nodo deben estar disponibles siempre que sea necesario.
- Actualización de datos.- Asegura de que no se hayan reproducido datos antiguos.

2.5. Protocolos e Interfaces de Comunicación

En una *WSN* los nodos sensores son el elemento central ya que estos realizan la detección de eventos, almacenamiento y procesamiento de los datos obtenidos y ejecutan los protocolos y algoritmos de comunicación. La calidad, el tamaño y la frecuencia de los datos detectados que se pueden extraer de un nodo dependen de los recursos disponibles en los componentes. Por lo tanto, el diseño e implementación del nodo sensor involucra una cierta complejidad [3].

En la sección anterior se mostró las cuatro unidades básicas que debe tener un nodo sensor las cuáles deben ser construidas y juntadas para formar un nodo unificado, y programable. Una vez realizada la unión, la unidad de procesamiento sera capaz de manejar las demás unidades mediante programación. En este contexto la comunicación entre las unidades se realiza mediante protocolo e interfaces como se muestra en la Figura 2.6.

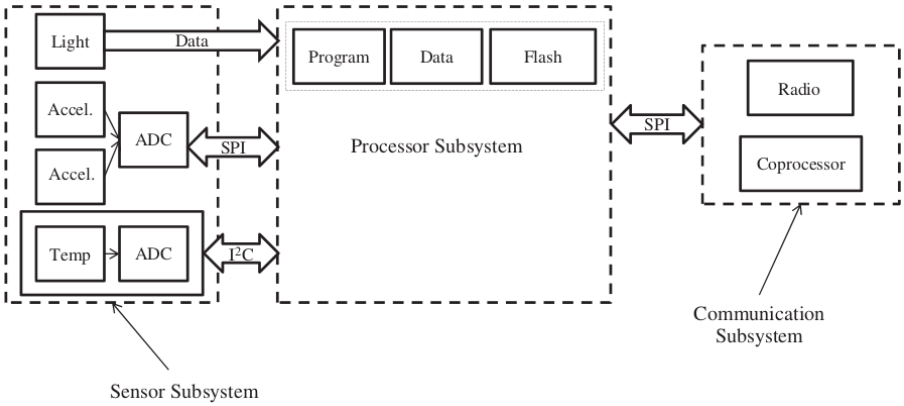


Figura 2.6: Arquitectura del nodo sensor [3]

2.5.1. *I2C*

El Circuito inter-integrado es un bus serie síncrono semi-dúplex multi-maestro como se esquematiza en la Figura 2.7. El objetivo de *I2C* es minimizar los costos para conectar dispositivos dentro de un sistema con velocidades de transmisión bajas. Existen dos modos de velocidad: modo rápido hasta $400Kbps$ y modo de alta velocidad hasta $3{,}4Mbps$. El dispositivo que usa *I2C* debe tener una dirección única de 10 bits.

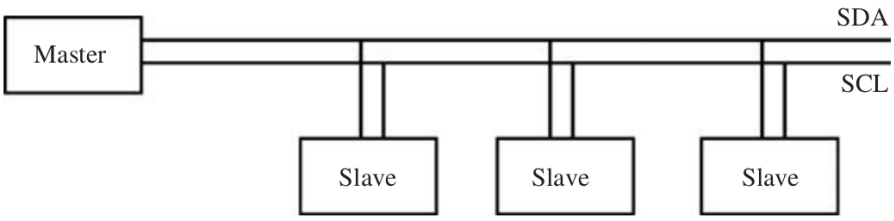


Figura 2.7: Conexión serial con *I2C* [3]

La interfaz *I2C* usa dos lineas bidireccionales, estas son *Serial Clock (SCL)* y *Serial Data Analizer (SDA)*. Cada maestro genera su propia señal de reloj, los dispositivos deben sincronizar sus velocidad de reloj. En caso de que no lo hagan, un dispositivo esclavo más lento podría detectar erróneamente su dirección en la línea *SDA* mientras un dispositivo maestro más rápido está enviando datos a un tercer dispositivo [3]. Además de la sincronización del reloj, *I2C* requiere arbitraje entre dispositivos maestros que desean enviar o recibir datos al mismo tiempo.

Como se puede observar en la Figura 2.8 el protocolo consiste en que un dispositivo maestro marca las condiciones de inicio y posteriormente envía una dirección de 7 bits que indican la dirección del dispositivo esclavo a cual va dirigido. Entonces el maestro expresa su interés por leer o escribir. En este momento, el esclavo envía un acuse de recibo (ACK). Posteriormente, el transmisor de datos envía datos de 1 byte (8 bits) que luego es aceptado por el receptor. Si aún quedan datos por enviar, el transmisor sigue enviando y el receptor sigue aceptando los datos. Finalmente, el maestro levanta la bandera de parada (condición de parada) para indicar el final de una comunicación [3].

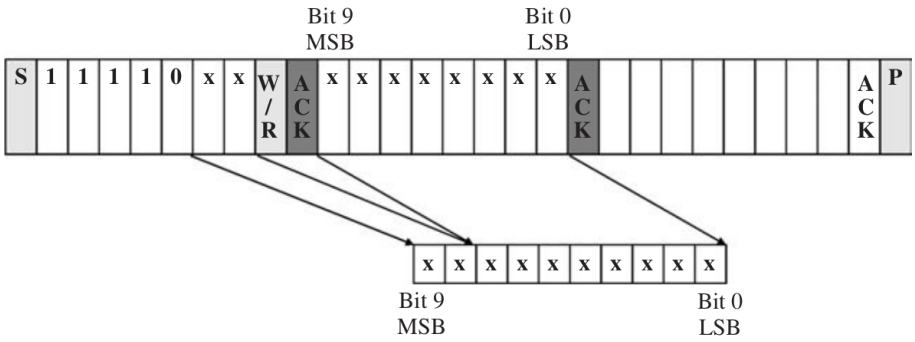


Figura 2.8: Protocolo de comunicación en *I2C* [3]

2.5.2. *SPI*

La interfaz periférica es un bus serie síncrono full-dúplex de alta velocidad. Se define cuatro pines: *Master Out-Slave In (MOSI)* (), *Master In-Slave Out (MISO)*, *Serial Clock (SCLK)* y *Chip Select (CS)*. Algunos fabricantes se refieren a *MOSI* como *SIMO* y a *MISO* como *SOMI*, pero la semántica es la misma. Del mismo modo, *CS* a veces se conoce como *SS*. Como su nombre lo indica, *MOSI* se utiliza para transmitir datos del maestro al esclavo cuando un dispositivo se configura como maestro. En caso de que esté configurado como esclavo, este puerto se usa para recibir datos del maestro correspondiente.

SCLK es utilizado por el maestro para enviar la señal de reloj que se necesita para sincronizar la transmisión; y por el esclavo para leer esta señal. Cada comunicación es iniciada por el maestro. Un dispositivo maestro señala a un esclavo con el que desea comunicarse a través del puerto *CS*. Como *SPI* es un bus maestro único, el microcontrolador es el maestro por defecto en un nodo sensor inalámbrico. Por lo tanto, los componentes no pueden comunicarse directamente entre sí, sino solo a través del microcontrolador [3]. En la Figura 2.9, se muestra la conexión entre dispositivos maestro y esclavos.

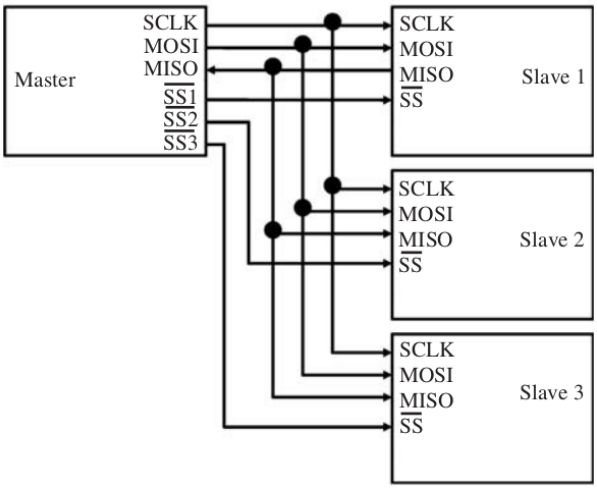


Figura 2.9: Conexión de dispositivos con *SPI* [3]

SPI admite un protocolo de comunicación síncrono. En consecuencia, el maestro y el esclavo deben acordar el momento. Para hacerlo, el maestro ajusta el reloj de acuerdo con la velocidad máxima del esclavo. El generador de baudios del maestro lee el reloj del esclavo y calcula el reloj del maestro

dividiendo la velocidad de lectura con un valor definido internamente.

2.5.3. *UART*

Un receptor y transmisor asíncrono universal es un circuito integrado que esta programado para controlar los puertos y dispositivos serie. Las funciones principales de chip *UART* son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

El *UART* toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo *UART* reensambla los bits en bytes completos. La transmisión serie de la información digital a través de un cable único u otros medios es mucho más efectiva en cuanto a costo que la transmisión en paralelo a través de múltiples cables. Se utiliza un *UART* para convertir la información transmitida entre su forma secuencial y paralela en cada terminal de enlace. Cada *UART* contiene un registro de desplazamiento que es el método fundamental de conversión entre las forma secuencial y paralela. Como no se transmite ningún reloj, el *UART* debe sincronizar el flujo entrante de bits con el reloj local [24].

2.5.4. RS-485

Es un estándar del canal de comunicación y el método de transmisión de la señal. La red de comunicaciones basada en RS-485 consta de transceptores conectados por un cable par trenzado. El principio básico de la interfaz es la transmisión de datos diferencial. Funciona para semi-dúplex, admite comunicaciones de datos multi-punto. Utiliza la estructura maestro-esclavo de la Figura 2.10 y se divide en los siguientes modos [4]:

- Modo maestro-esclavo.- El sistema de comunicación consta de un dispositivo maestro y varios esclavos. El maestro comprueba constantemente si el esclavo necesita comunicación, si es necesario, otorga el derecho de control de Bus a los dispositivos esclavos hasta que se envíen por completo. El dispositivo maestro no solo puede comunicarse con el esclavo por separado, sino que también puede transmitir datos. Si los dispositivos esclavos desean comunicarse entre sí, deben hacerse a través del dispositivo maestro.
- Modo alternativo maestro-esclavo.- El modo significa entregar el control de bus en todos los dispositivos de manera similar a *Token Ring*, el dispositivo que obtiene el control correcto se convierte en maestro y los otros son esclavos. Un dispositivo pasa el control del bus a los adyacentes cuando termina de enviar datos, y los dispositivos adyacentes pasan el control justo después de ocuparse de los requisitos de comunicación, por lo que cualquier punto de control puede inicializar la comunicación con otro controlador.

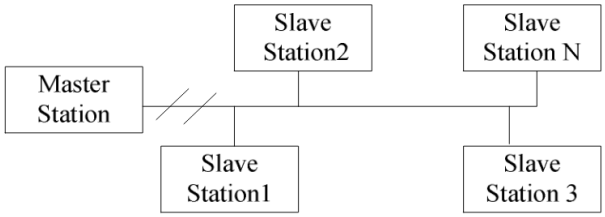


Figura 2.10: Estructura de comunicación con *RS485* [4]

2.6. Tecnologías para desarrollo de Aplicaciones web y Móviles

El desarrollo de servicios web se ha convertido en un estándar en la programación al momento de compartir información entre diferentes aplicaciones de *software*, esto se debe a que para distribuir sus funcionalidades estos servicios son independientes del *hardware*, el sistema operativo, el lenguaje de programación e independientes al momento de ejecutarse y desarrollarse para su uso.[25] Herramientas como Internet, las aplicaciones web y las aplicaciones móviles, que trabajan de la mano con el desarrollo de los sistemas de comunicación, permiten que la información se difunda de forma más rápida y eficiente a nivel mundial, estas características han convertido a las aplicaciones web y móviles en ejes fundamentales para el desarrollo de las aplicaciones *IoT*, mejorando la calidad de vida de las personas al brindar servicios de fácil acceso para realizar consultas sobre el clima, visualizar la ubicación en tiempo real, encontrar sitios turísticos, restaurantes, centros médicos, entre otras aplicaciones.

2.6.1. Aplicaciones web

En la actualidad las aplicaciones web se han vuelto más interactivas y flexibles para abordar diferentes áreas relacionadas con las necesidades de la sociedad y los negocios [26]. Estas características agregan complejidad adicional en el diseño y desarrollo de aplicaciones, donde los usuarios son cada vez más exigentes. Como solución a la complejidad, las aplicaciones web se han vuelto demasiado largas, complejas, distribuidas y divididas en diferentes capas y componentes.

En un escenario en el que varios clientes acceden a servicios de aplicaciones web desde diferentes ubicaciones geográficas al mismo tiempo, los problemas de rendimiento de las aplicaciones se vuelven muy importantes, ya que están asociados con la satisfacción del cliente. Hay muchos factores que determinan el rendimiento de una aplicación web, como, los parámetros del sistema, el protocolo de red, el ancho de banda, latencia y velocidad de transferencia, parámetros de carga de trabajo,entre otros, que influyen en el rendimiento de las aplicaciones web. [26]

2.6.1.1. Arquitectura

Para mejorar el funcionamiento de las aplicaciones web existen diferentes *frameworks* que plantean una arquitectura que divide el funcionamiento de la aplicación en varios componentes. Desde un punto de vista físico, proporcionan una abstracción de la arquitectura cliente servidor y desde un punto de vista lógico proporciona un alto nivel de abstracción. La mayoría de *frameworks* están basados principalmente en dos modelos Presentación Abstracción Control (*PAC*) y Modelo Vista Controlador (*MVC*). *PAC* es más adecuado para *software* interactivo y esta conformado por tres componentes que

tienen una relación padre-hijo. Por otro lado *MVC* se usa popularmente para el diseño de aplicaciones web brindando características de mantenimiento, escalabilidad y eficiencia. La arquitectura *MVC* proporciona una forma de descomponer una aplicación en tres partes:

- Modelo.- Un modelo representa los datos de una aplicación y contiene la lógica para acceder y manipular esos datos.
- Vista.- La vista es responsable de representar el estado del modelo.
- Controlador.- El controlador es responsable de interceptar y traducir la entrada del usuario en acciones a realizar por el modelo.

MVC ha sido adoptado por todos los principales actores tecnológicos del mercado para proporcionar soporte ambiental de programación en el desarrollo de aplicaciones. Proporciona una mejor separación de la lógica de la aplicación [26].

2.6.1.2. Servicios *REST*

Las *API*s *REST*, hoy en día son uno de los ejes fundamentales en desarrollo de las diferentes aplicaciones, sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. El servicio *REST* presenta las siguientes características:

- Satisface las peticiones sin almacenar estados en ninguno de los extremos.
- Simplifica la comunicación mediante los mensajes *POST, GET, PUT* y *DELETE* de HTTP.
- Cada objeto tiene sus propia *Uniform Resource Identifier* (*URI*).
- Arquitectura basada en un modelo de capas jerárquica
- Ejecuta acciones especificas sobre los datos mediante el uso de hipermedios.

Algunas de las ventajas que ofrece el sistema *REST* para el desarrollo de aplicaciones son, la separación entre el cliente y el servidor, la visibilidad, fiabilidad y escalabilidad e independencia del tipo de plataformas o lenguajes.

2.6.1.3. *Node.js*

Node.js es un entorno de ejecución de *JavaScript* orientado a eventos asíncronos, diseñado para construir aplicaciones en red escalables usando muchas conexiones concurrentes, este permanecerá en *sleep* hasta que no detecte una solicitud, a continuación se muestra los beneficios de usar *node.js* como entorno de ejecución [25]:

- Puede generar páginas con contenido dinámico.
- Puede crear, abrir, leer, escribir, eliminar y cerrar archivos en el servidor.
- Puede agregar, eliminar, modificar registros de la base de datos.
- Los archivos *JavaScript* contienen métodos que se ejecutaran de acuerdo a ciertos eventos.
- Fácil y sencillo de aprender e implementar.
- En la actualidad es considerado lo mejor para el desarrollo de *API REST*.
- Poco código para mantener.
- Fácil de testear con aplicaciones con *plugins* como *Mocha/Chai*.
- Empaquetado a través de *Node Package Manager* (*NPM*).
- Modelo de no bloqueo de I/O.

- Trabaja con casos síncronos y asíncronos.
- Fácil integración con bases de datos *NoSQL*.

2.6.2. Aplicaciones Móviles

En América Latina nueve de cada diez personas utilizan un teléfono inteligente según datos recabados por la segunda edición del estudio *Internet Media Services (IMS) Mobile in LatAM* [9]; esto ha generado un aumento considerablemente el desarrollo de aplicaciones móviles que sirvan como medio para solventar las necesidades de los diferentes usuarios de la red.

2.6.2.1. Elementos de la tecnología Móvil

- Redes Celulares.- Son la infraestructura base de la tecnología móvil, que deben cumplir dos funciones principales, mantener una conexión constantes y asegurar una buena velocidad de trasmisión, en la Tabla 2.1, se presenta la velocidad de descarga en usuarios con alta movilidad para cada una de las diferentes tecnologías móviles.

Tabla 2.1: Velocidades de conexión de las redes de transmisión celular más conocidas [9]

RED	Denominación	Velocidad
Global System For Mobile Comunication (GSM)	2G	Hasta 0.009 Mbps de descarga
General Packet Radio Service (GPRS)	2.5G	Hasta 0.08 Mbps de descarga
Enhanced Data Rates for GSM Evolution (EDGE)	2.75G	Hasta 0.236 Mbps de descarga
Universal Mobile Telecommunications System (UMTS)	3G	Hasta 0.384 Mbps de descarga
High-Speed Packet Access (HSPA)	3.5G	Hasta 7.2 Mbps de descarga
Evolved HSPA	3.75G	Hasta 22 Mbps de descarga
Long Term Evolution (LTE)	4G	Hasta 75 Mbps de descarga

- Tipos de dispositivos.- En la actualidad existen una gran variedad de dispositivos móviles dentro de los que podemos encontrar, *Smartphones*, *Tablets* y dispositivos incorporados.
- Tipos de aplicaciones.- Existen tres tipos de aplicaciones, nativas, híbridas y web. A diferencia de las aplicaciones web (basadas en el navegador), las aplicaciones nativas e híbridas están instaladas físicamente en el dispositivo y, por lo tanto, siempre están disponibles para el usuario. En la Tabla 2.2, se realiza una comparación de las características básicas presentes para cada tipo de aplicación.

Tabla 2.2: Comparación de los tipos de aplicaciones móviles.[9]

Características	Nativa	Híbrida	Web
Uso de características específicas del dispositivo	Alta	Media	Baja
Uso de *API*s nativas	Alta	Media	Baja
Uso de gráficos	Alta	Media	Media
Calidad de la interfaz de usuario	Alta	Media	Baja
Capacidad de actualización	Baja	Media	Alta
Portabilidad	Ninguna	Alta	Alta
Instalación en el dispositivo	Si	Si	No

- Plataformas: La plataforma o sistema operativo móvil es el encargado de manejar, controlar y gestionar todo lo relacionado con los recursos de hardware de un dispositivo móvil. Esto trae consigo grandes retos para los fabricantes y desarrolladores, pues deben optimizar la gestión de procesos, el manejo de memoria principal y secundaria y la gestión de archivos, sin sacrificar el rendimiento y la seguridad. Hoy en día el mercado es dominado por dos sistemas operativos móviles: *Android* e *iOS* [9].

2.6.2.2. Tipos de aplicaciones móviles

En los últimos años el mercado de los dispositivos móviles, en especial *Smartphones*, ha mostrado un crecimiento notable. A continuación se presentan tres enfoques para desarrollo de aplicaciones para dispositivos móviles [27].

- **Aplicaciones Web:** Este tipo de aplicaciones usan la misma tecnología de desarrollo de sitios web, por lo general se desarrollan bajo un *stack* *HyperText Markup Language (HTML)*, que integra *Hypertext Transfer Protocol (HTTP)*, *Cascading Style Sheets (CSS)* y *JavaScript*.
  Las ventajas principales de este modelo de desarrollo es que no necesitan ser instalada sobre el dispositivo móvil, ni la aprobación del fabricante para su distribución. Además, las actualizaciones de la aplicación son visualizadas directamente en el dispositivo, ya que los cambios son aplicados sobre el servidor. Sin embargo, esto disminuye la velocidad de ejecución y podrían llegar a ser menos atractivas que las aplicaciones nativas. Además, este tipo de aplicaciones no pueden utilizar todos los elementos de *hardware* del dispositivo.
- **Aplicaciones Nativas:** Este tipo de aplicaciones son las mas usadas en los dispositivos móviles, dado que las mismas permiten usar todas las capacidades del equipo. Su ejecución es rápida, puede ejecutarse en modo *background* y notificar al usuario cuando ocurra un evento que necesite su atención. Sin embargo, requieren mayor costo de desarrollo, pues se debe utilizar un lenguaje de programación diferente según la plataforma. Por ende, si se desea cubrir varias plataformas, se deberá generar una aplicación para cada una de ellas. Esto conlleva a mayores costos de actualización y distribución de nuevas versiones.
  Sin embargo, este tipo de aplicaciones presentan mayor complejidad en el desarrollo multiplataforma, dado que las mismas están restringidas a las características propias de cada dispositivo (Tipo de dispositivo, sistema operativo, Versión).

- **Aplicaciones Híbridas:** Últimamente este tipo de aplicaciones han ganado fuerza entre los desarrolladores debido a que combina las características de cada una de la aplicaciones anteriores. Sin embargo, el rendimiento general sera limitado respecto a los modelos de desarrollo anteriores.

2.7. Conclusiones

Uno de los primeros pasos para transformar las ciudades tradicionales en *smart Cities* es trabajar en el tráfico urbano que genera grandes pérdidas económicas y ambientales, afectando directamente a la calidad de vida de la población. Existen diferentes enfoques con los que se puede solventar esta problemática. En tal sentido, uno de los métodos que genera mayor beneficio a corto plazo es el fomentar el uso de la bicicleta como medio transporte, permitiendo reducir las emisiones de gases contaminantes y el nivel de ruido que generan los vehículos. Todos los beneficios que ofrece una *Smart City* a sus ciudadanos esta directamente relacionado con la inversión y manejo de las *TIC* usadas para transportar grandes flujos de datos de sensores y actuadores, por lo que se necesita una red de banda ancha que pueda soportar elementos móviles y fijos con capacidad de interacción en tiempo real. Las *TIC* además permiten que todas las capas de una ciudad se comuniquen y funcionen como un solo sistema diseñado para mejorar la calidad de vida de la población.

IoT es una de las tecnologías que mayor impacto ha logrado generar en la sociedad, permitiendo capturar datos del mundo real y usarlos en el mundo digital. Los datos recolectados comúnmente son tratados con *Big Data* y *Cloud Computing* para reducir el costo computacional y de almacenamiento de los nodos, así como también garantiza un menor tráfico en la red de forma que no congestione la misma. Debido a la complejidad que involucra un proyecto *IoT*, se han desarrollo una gran número de *frameworks* que permiten integrar diferentes tecnologías, reduciendo la complejidad de implementación, el tiempo de desarrollo y aumentando la confiabilidad.

Un nodo sensor esta compuesto por cuatro partes principales que se estructuran de acuerdo a la aplicación ejecutada. De esta forma al momento de elegir los dispositivos que formarán al nodo se requiere realizar un análisis previo, sobre todo poniendo mayor atención en la fuente de alimentación, ya que es una parte crítica para el buen desempeño del sensor. Los protocolos de comunicación hacen que los sensores y el dispositivo de procesamiento puedan comunicarse de una forma controlada. De esta forma para para la elección de un sensor en particular se debe verificar que las unidades de detección y procesamiento manejen el mismo protocolo de comunicación. Y, además, la mayoría de los sensores usan los protocolos *SPI* e *I2C*.

La variedad de dispositivos finales para visualizar datos y recolectar información, así como la gran cantidad de sistemas operativos usado en los diferentes dispositivos ha generado la necesidad de crear interfaces de visualización independientes del *hardware* y el sistema operativo. Como solución a esta problemática surge las aplicaciones web que cuentan con la posibilidad de integrarse a Internet y ser accedida desde cualquier buscador siempre y cuando se conozca su nombre de dominio. Para simplificar el desarrollo de estas aplicación se usa diferentes arquitecturas como *MVC*. Además de servicios como *REST* que permite el manejo de objetos bajo una única *URI* y la integración de lenguajes de programación como *node.js*, logrando así, generar aplicaciones de red escalables. Sin embargo, este tipo aplicaciones no puede manipular completamente todos las capacidades de los equipos finales, por lo cuál, para solventar esta problemática se ha desarrollado nuevos modelos de aplicaciones que se pueden integrar fácilmente en dispositivos móviles de forma nativa.

Capítulo 3

Estado del Arte: Tecnología IoT y Monitorización de Variables en Tiempo Real aplicado a Ciclovias

El ciclismo es reconocido como uno de los medios de trasporte más amigables con el medio ambiente y es apoyado ampliamente alrededor del mundo por diferentes gobiernos. Sin duda una tarea fundamental es mejorar la experiencia de los usuarios en las ciclovías para incrementar la participación de los ciudadanos. De esta forma es posible desarrollar un sistema de movilidad más sostenible en las ciudades teniendo en consideración las características compartidas y únicas de cada ciudad. Por lo tanto los encargados de la toma de decisiones deben tener un panorama claro de como esta estructurado el sistema de bicicletas, con el fin de generar proyectos que impulsen el ciclismo y muestren beneficios en la movilidad.

Durante las últimas tres décadas, se han realizado muchos estudios en el campo del ciclismo para evaluar diferentes características de este modo de transporte que influyen directamente en como promover este servicio, una de las más utilizadas para el análisis es el confort. El confort que podría definirse como la armonía entre los humanos y el medio ambiente debido al equilibrio de los aspectos físicos, psicológicos y sociológicos. Esto hace que la evaluación sea más compleja porque se basa en las características de cada persona. El confort también se puede considerar como la capacidad para andar en bicicleta, la idoneidad y la amabilidad con la bicicleta.

3.1. Modelo de adquisición de datos

En la actualidad existen diferentes enfoques tecnológicos orientados a la recolección de datos para mejorar e impulsar el ciclismo local, regional y mundial, garantizando que la exposición de los ciclistas a contaminantes sea menor, de tal forma que el ciclismo se desarrolle en un entorno saludable, por ejemplo:

En [28] se propone el análisis de la calidad del aire usando un detector de partículas NOVA SDS0111 integrado a una Raspberry Pi modelo 3B. Se usa un módulo *GPS* Globalsat BU-353-s4 para posicionar los datos recolectados durante el movimiento. Luego, la información se visualizan en tiempo real usando

un aplicación móvil, la cual presenta el valor numérico de concentración de partículas, empleando para ello un esquema de semáforo acorde a los índices definidos en este caso en Europa. Se debe mencionar además, que los datos de toda la trayectoria se almacenan localmente y son usados para generar mapas de contaminación en QGIS (*Quantum Geographic Information System*).

De igual manera en [29], se presenta un dispositivo que consta de un detector de partículas, un sensor de humedad y temperatura, un receptor *GPS* y un módulo de comunicación NB-IoT, los cuales son instalados sobre una bicicleta compartida para monitorizar la calidad del aire en sus trayectorias. El sistema funciona integrando cada uno de los sensores en una plataforma Arduino, donde también se incorpora una tarjeta micro-SD para almacenar los datos recolectados. La placa Arduino a su vez se conecta con un módulo de comunicación NB-IoT el cual permite cargar los datos en el servidor usando las redes *LTE* o *GPRS*. En los resultados presentados se muestra mediciones periódicas cada 30 minutos donde no se puede apreciar mayor cambio en las lecturas.

De forma similar en [30], se presenta un sistema de monitorización de contaminantes del aire, incorporado en un sistema público. En concreto, la estación cuenta con sensores, *GPS*, *Bluetooth* y un módulo *GPRS*. En este caso, en el momento en que la bicicleta es devuelta, los datos se cargan en la estación y son enviados a un centro de datos. También lo hace de forma directa a través del *GPRS*. Posteriormente, los datos recopilados son presentados en un mapa de colores que muestran el nivel de contaminación presente en la trayectoria. Esto se realiza a través de un sitio web disponible para todos los usuarios.

Por otra parte en [31], se plantea el desarrollo de un sistema denominado *CYclAir*, que cosiste en monitorizar los niveles de *CO2* en el recorrido de un ciclista, con la finalidad de mostrar datos en tiempo real y retrospectivamente de los niveles de contaminación. La visualización de los datos en tiempo real se realiza usando indicadores LED siguiendo un modelo semáforo según el nivel de *CO2* de la zona actual. El sistema esta desarrollado sobre una plataforma Arduino que recolecta los datos de niveles *CO2* (Adafruit SGP30) y posicionamiento *GPS* (Neo6M-V2) en intervalos de un segundo. Los datos recolectados se almacenan en una memoria micro SD, permitiendo a los usuarios realizar un análisis del recorrido una vez finalizado el mismo.

De igual manera en [32], se presenta el sistema *Citisense* conformado por cuatro componentes principales, un placa de sensores portátil, un *smarthphone* para monitoreo, un servidor web para brindar mapas de contaminación diaria y un componente social que permite la integración con *facebook* y *twitter* para la publicación de los datos adquiridos. La placa de sensores usa comunicación *bluetooth* para enviar datos de calidad del aire al *smartphone*, donde se visualizan en una aplicación móvil, la cual presenta el numero y color del índice de calidad del aire registrado en un promedio de 6 segundos. Además, la aplicación cumple la función de enviar la información a la nube e incorporarlos en la base de datos. La aplicación web por otro lado permite monitorizar la ruta diaria y los niveles de exposición a contaminación durante el día para cada usuario. Ambas aplicaciones cuentan con la posibilidad de publicar los mapas de contaminación y las lecturas en tiempo real en las redes sociales. Se debe mencionar que la placa de sensores es independiente del medio de trasporte. En el estudio se incorpora sobre la mochila de los usuarios.

En cuanto a seguridad se refiere, en [33], se presenta un sistema móvil para albergar sensores de imagen, sensores de posicionamiento y sensores de calidad de aire, los mismos que se implementan sobre una bicicleta con la finalidad de recolectar datos que permitan describir las características tanto del estado de las ciclovías, así como los niveles de calidad de aire por zonas. En dicha propuesta

cada sensor funciona de forma independiente, es decir, tiene su propio sistema de control y los datos recolectados se almacenan individualmente. Al finalizar el recorrido la recopilación y gestión de los datos adquiridos se analiza mediante un sistema tipo GIS (*Geographic Information System*).

Un estudio similar se realiza en [34], donde se desarrolla un sistema que provee de servicios de ubicación de la bicicleta en tiempo real, servicios de anti-robo, información sobre la ruta recorrida y monitoreo de la contaminación de aire. El prototipo desarrollado en dicho estudio trabaja bajo la condición de que el usuario solo posee una bicicleta y un dispositivo móvil. Los datos recolectados relacionados a la calidad del aire están influenciados únicamente por el nivel de concentración de monóxido de carbono. Los datos de calidad del aire y movilidad son enviados mediante conexión *bluetooth* a la aplicación móvil, la cual se encarga de subir los datos a los servidores centrales usando la redes móviles. A continuación, la información se visualiza tanto en la aplicación móvil como en la aplicación web, esta última se emplea además para brindar mapas de calidad de aire de las rutas recorridas y mostrar las ubicaciones donde se generaron alertas de robo.

En cuanto al manejo de sensores, en [35], se propone el uso de una *WSN* para registrar, almacenar y visualizar datos adquiridos por sensores incorporados en un bicicleta. El sistema usa una plataforma Raspberry Pi 3 para el almacenamiento, comunicación *MQTT* para interconectar diferentes nodos, mientras que la información se almacena en una base de datos *SQL* y se presenta en una aplicación móvil. La arquitectura del sistema se compone de 3 elementos un *Gateway*, sensores y un *smartphone*, donde cada uno se conectan con una plataforma WeMos D1 ESP8266 la cual se configura como *access point* para brindar la conexión *wifi* dedicada con la plataforma Raspberry pi. Finalmente, la información se gestiona mediante una base de datos y se visualiza mediante una interfaz de usuario desarrollada con el *framework node red*.

Por otra parte en [36], se presenta una propuesta que se enfoca en el uso de bicicletas eléctricas para reducir la inhalación de contaminantes. Con este objetivo se plantea reducir el volumen de aire inhalado por minuto, mediante el uso de un sistema de control, que aumenta o reduce la potencia del motor eléctrico en función de datos de contaminación adquiridos. El control del sistema se realiza usando comunicación *bluetooth* entre el *smartphone* y una plataforma Arduino.

Al momento se han detallado varios estudios relacionados en el contexto del presente trabajo de tesis, sin embargo, aún falta mencionar la forma en que los datos son almacenados y presentados al usuario. En tal sentido, en [37] se muestra el uso de la plataforma *Firebase* de *Google* para el almacenamiento de datos. En concreto, se presenta una aplicación que permite guardar la ruta de los autobuses. Dicha aplicación ha sido desarrollada en *Android Studio* y mediante una *API* se accede a la base de datos de *Firebase*. Así mismo, en [38], se ha desarrollado un software para la seguridad de los usuarios el cual ha sido realizado usando *Android*, *node.js* y *Firebase*.

Por otra parte, *Firebase* también demuestra ser muy amigable para adaptarse con diferentes tarjetas de desarrollo, en [39] se muestra un sistema que usa una Raspberry para la seguridad de los vehículos y que almacena los datos en *Firebase*. El sistema almacena los datos para monitorizar la ruta y además se usa para el control remoto del encendido del vehículo. Para el manejo de los datos se usa una aplicación web que requiere un inicio de sesión. Finalmente, se resalta que para dicha implementación se ha empleado la herramienta *AngularFire*.

3.2. Confort en el ciclismo

En lo que respecta a estudios previos enfocados en el análisis del confort de los ciclistas, en [5], se discute el concepto nivel de servicio de las bicicletas BLOS (*Bycicle Label-Of-Service*). Dicho concepto se centra en las percepciones de comodidad de los usuarios mediante el análisis de tres factores, flujo de bicicleta, infraestructura y las variables exógenas, como se esquematiza en la figura 3.1. A continuación, se detallan dichos factores.

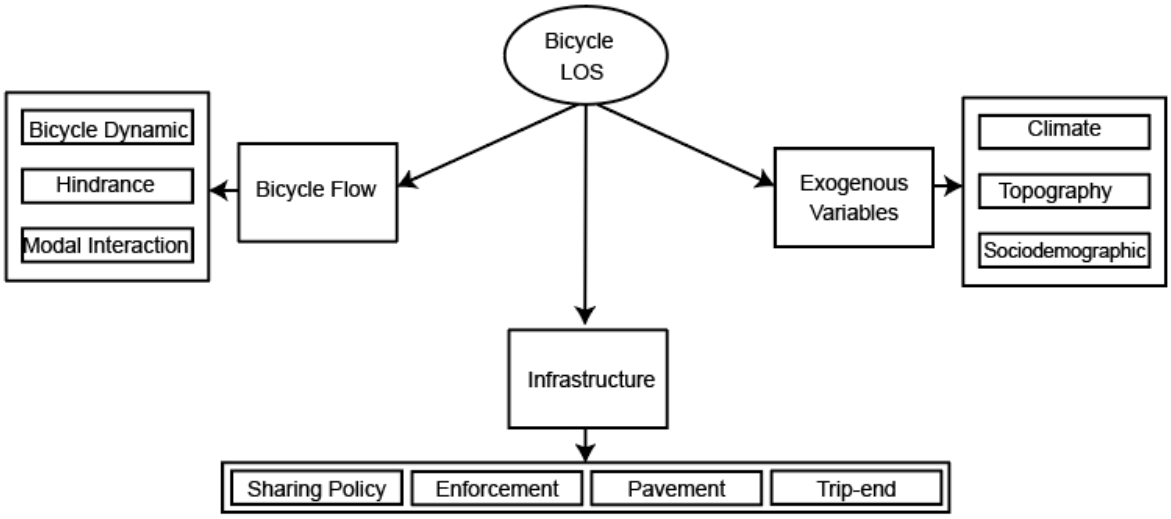


Figura 3.1: Parámetros que influyen en el confort del ciclista [5]

Flujo de bicicleta: Permite comprender el movimiento de la bicicleta y como esta afecta la percepción del usuario, para ello [5] enfoca su investigación en tres variables:

- Dinámicas de bicicletas.- Se centra en describir los factores de movimiento dentro y fuera de la carretera, establece tres zonas ovales, zonas de colisión, confort y circulación, En la zona de colisión (0,9m x 2,4m), se llevan a cabo acciones evasivas. La zona de confort (1,7m $\times$ 3,4m) es donde la bicicleta no es interrumpida por otras bicicletas, y la zona de circulación (2,3m $\times$ 4m) es el área dentro de la cual los ciclistas pueden moverse libremente.
- Fenómenos de impedimento.- Se analiza la libertad de maniobra frente a obstáculos y reuniones, los cuales, generalmente ocurren cuando el ciclista necesita rebasar a un ciclista más lento y cuando dos ciclistas se encuentra en una zona de la ciclovía viajando en direcciones opuestas. En los estudios realizados se concluye que la probabilidad de reunión es mayor que la probabilidad de obstáculo aumentando el temor a sufrir accidentes.
- Interacción modal.- Analiza como interactúan los diferentes modos de transporte en las instalaciones viales. El diseño de las ciclovías en términos de acomodar diferentes tipos de medios de transporte requiere una consideración integral, ya que esto puede afectar tanto la comodidad como la seguridad de los ciclistas.

Infraestructura: Los análisis de costo beneficio [40, 41] demuestran que el aumento del ciclismo beneficia en 4 o 5 veces los costos de invertir en nueva infraestructura para el ciclismo, dentro de la estructura se puede distinguir 5 variables principales:

- Política de intercambio: Se enfoca en el análisis de donde pueden viajar los ciclistas, planteando instalaciones en la calle y fuera de la calle. A partir de los estudios, se establece que los ciclistas prefieren instalaciones fuera de la calle por seguridad y confort. Sin embargo, esto genera roses entre los peatones y ciclistas. Un diseño de las ciclovías alejado de las carreteras con mucho tráfico reduce significativamente la ingesta acumula de contaminantes permitiendo que el ciclismo sea un medio de transporte más atractivo para los usuarios.
- Aplicación de tráfico: Las señales de tráfico, las señales y las marcas viales son una parte vital de la infraestructura de una carretera dada, sin embargo el exceso de señalización puede jugar un rol negativo en el confort de los ciclistas.
- Condiciones del pavimento: Existen diferentes características del pavimento que afectan el confort del ciclista, por ejemplo los montículos, reductores de velocidad, la mala calidad del pavimento, la infraestructura de acceso a los alcantarillados, etc. Dichos factores ocasionan una tasa de cambio de carril de 51,38 %, impactando directamente en la comodidad del ciclista y la capacidad de del carril.
- Instalación de fin de viaje: En este caso se analiza la influencia de la ubicación de las instalaciones de aparcamiento en la comodidad de los ciclistas. Los estudios indican que una buena elección de estas estaciones podría aumentar hasta en tres cuartos la población de ciclistas.

Factores Exógenos: Al respecto, en [5], se analiza varios estudios donde se muestran que para determinar el confort del ciclista se requiere una evaluación no solo de los factores que pueden controlarse por el ciclismo en sí, sino también de los factores que no depende del ciclismo como, el clima, la topografía y los aspectos socio-demográficos (edad, el género y el nivel educativo), factores importantes que se deben considerar en el diseño de nuevas rutas, ya que los mismos pueden reducir hasta en 47 % la población de ciclistas.

3.2.1. Materia Particulada

La *MP* se genera al combinarse partículas sólidas que pueden provenir del polvo, del humo, del hollín, o de procesos de producción con partículas líquidas que se encuentran suspendidas en el aire. Debido a su tamaño y densidad, estas partículas permanecen suspendidas, por lo que no se sedimenta en períodos de tiempo cortos, lo que afecta directamente la calidad de aire en la ciudad y por ende la calidad de vida. Estas partículas pueden llegar a formar complejas mezclas de productos químicos y/o elementos biológicos, como metales, sales, materiales carbonosos, orgánicos volátiles, compuestos volátiles, hidrocarburos, aromáticos policíclicos y endotoxinas que pueden interactuar entre si formando otros compuestos [42], que afectan al medio ambiente y a la salud de las personas.

Debido a que el *MP* tiene diversas formas, composición y tamaño para su identificación se han clasificado en función de su diámetro como finas y gruesas, donde las partículas finas tiene un diámetro menor de 2,5 micras y las partículas gruesas un diámetro entre 2,5 micras y 10 micras. Las primeras se forman básicamente por medio de procesos mecánicos, como las obras de construcción, la suspensión del polvo de los caminos y el viento, mientras que las segundas se proceden sobre todo de fuentes de combustión. En la mayor parte de los entornos urbanos están presentes ambos tipos de partículas, gruesas y finas, pero la proporción correspondiente a cada uno de los dos tipos de tamaños es probable que varíe de manera sustancial entre las ciudades en todo el mundo, en función de la geografía, la meteorología y las fuentes específicas de *MP* de cada lugar [43].

3.2.1.1. Efectos de la *MP* en la salud

La *MP* afecta a más personas que cualquier otro tipo de contaminante en el mundo, al ser partículas muy pequeñas nuestros cuerpos no pueden detectarlas, por lo que no activa sus mecanismos de defensa, permitiendo el paso libre del *MP* hacia los pulmones e inclusive hacia el torrente sanguíneo. Dicho efecto resulta en enfermedades cardiovasculares y respiratorias, que pueden causar muertes prematuras, ataques cardíacos no mortales, irregularidades en el ritmo cardíaco e inclusive cáncer pulmonar. Debido al gran impacto que tiene sobre la salud, los gobiernos de todo el mundo han planteado diferentes leyes que regulan la exposición de las personas al *MP*, así en Ecuador, la Normativa Ecuatoriana de Calidad del Aire establece los límites máximos permisibles de contaminantes en el aire ambiente a nivel del suelo como se muestra en la Tabla 3.1.

Tabla 3.1: Niveles máximos permisibles de contaminantes en el aire ambiente a nivel del suelo [10]

Contaminante y Período de tiempo	**Norma Vigente**
Monóxido de Carbono Concentración promedio en una hora	10000 $\mu g/m^3$
Ozono Concentración promedio en ocho horas	100 $\mu g/m^3$
Dióxido de Nitrógeno Concentración promedio en una hora	200 $\mu g/m^3$
Dióxido de Azufre Concentración promedio en 10 minutos	500 $\mu g/m^3$
Material Particulado con diámetro menor a 10 micras (24H)	100 $\mu g/m^3$
Material Particulado con diámetro menor a 2,5 micras (24H)	50 $\mu g/m^3$

Capítulo 4

Arquitectura del Sistema

4.1. Introducción

Acorde a la descripción realizada en el Capitulo 3, las bicicletas son ampliamente usadas en aplicaciones para la recolección de datos que posteriormente son usados para resolver algún problema específico. Siempre será más factible resolver problemas si se tiene conocimiento de las causas reales que lo provocan. En este contexto, el presente proyecto busca construir un prototipo capaz de registrar datos que estén presentes en la ruta de un ciclista.

Para cumplir con el objetivo propuesto, en el Capitulo 2, se describieron las herramientas necesarias para la realización de este trabajo. Uniendo todos estos conceptos y tecnologías se pretende formar el sistema de monitorización.

El presente capítulo describe el proceso realizado para el diseño e implementación del sistema. En primer lugar se describe la arquitectura general del sistema, luego se expondrá todos los componentes usados para formar la estación móvil también llamada “nodo". Después, se mostrará el diseño de las aplicación web y móvil para la visualización de los datos. Finalmente, se muestra la funcionalidad del sistema de monitorización.

4.2. Descripción de la Arquitectura

En esta sección se presenta la arquitectura del sistema formado con tecnologías *IoT* para la monitorización de las variables físicas. El sistema consta de tres partes principales: la adquisición, transmisión y análisis de las variables obtenidas en el recorrido de la bicicleta.

En la Figura 4.1, se muestra el esquema general del sistema. El nodo realiza las funciones de adquisición y transmisión, mientras que el analisis de los datos se realiza mediante una aplicación *android* y web.

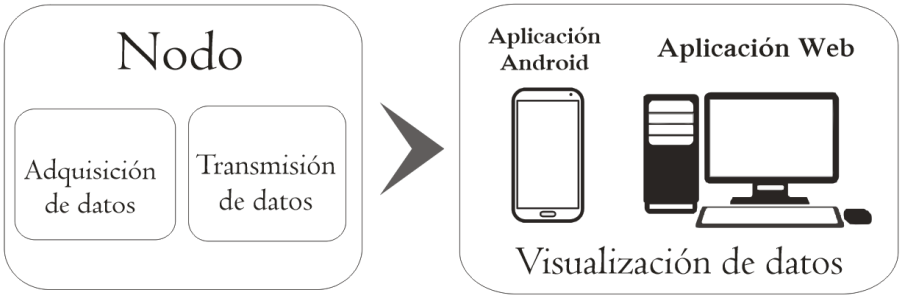


Figura 4.1: Arquitectura General

4.2.1. Arquitectura del nodo

La adquisición de datos es una de las funciones principales del nodo. En concreto, se trata de un conjunto de sensores encargados de detectar las variables de interés. También se encuentra el *GPS*, que estará obteniendo las coordenadas de posición actual de la bicicleta. En la Figura 4.2, se muestra los elementos usados para obtener los datos.

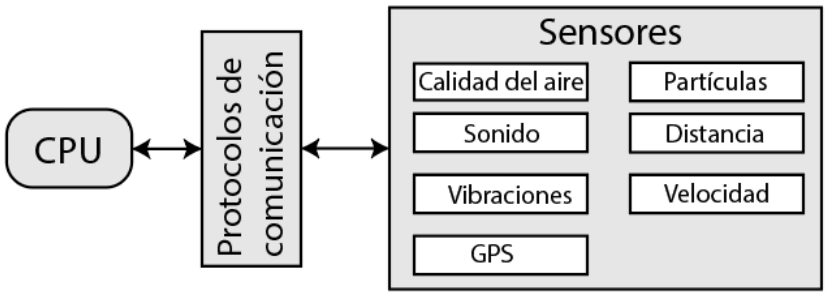


Figura 4.2: Elementos que integran la adquisición de datos

La unidad de control y procesamiento se encarga de registrar los datos detectados por cada sensor. Además, en el caso de que algún dato necesite un procesamiento adicional se realiza en la misma unidad. Esta debe ser capaz de adaptarse al protocolo e interfaz que use cada sensor. En la siguiente sección se describe los elementos seleccionados para la implementación del nodo.

La transmisión de datos es la función que complementa al nodo. Una vez que se ha obtenido los datos es necesario la transmisión de los mismos, los cuáles podrán ser visualizados en tiempo real en una aplicación móvil y web. Además, tendrá otra función para observar todos los datos obtenidos en un recorrido.

4.3. Componentes de la Estación e Implementación

4.3.1. Plataforma *SBC* Raspberry Pi

La plataforma *SBC* de la fundación Raspberry Pi, se lanzó por primera vez en febrero del 2012 con su Raspberry Pi 1 modelo A y modelo B, debido al increíble éxito que tuvieron en el mercado en 2014 lanzaron las Raspberry Pi 1 Modelo A+ y Raspberry Pi Modelo B+, que remplazarían a sus predecesores. En la actualidad los modelos A y B han sido descontinuados y hoy en día la fundación

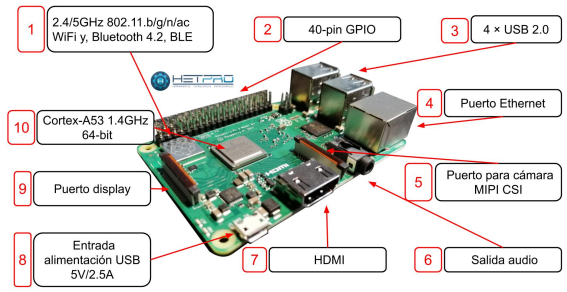
cuenta con cuatro series en producción que suman un total de 9 computadores de tamaño reducido, donde la más actual es la Raspberry Pi 4 modelo B lanzada en Junio del 2019. En la Tabla 4.1, se presenta las características principales de cada uno de los modelos que actualmente se encuentran en producción.

Tabla 4.1: Características de los *SBC* en producción de la fundación Raspberry Pi.

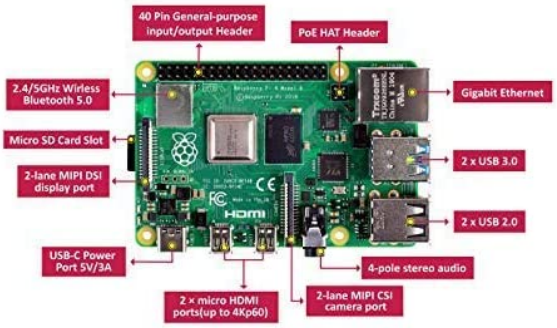
Plataforma	**Procesador**	**RAM**	**Costo**	**Conectividad**	**Vídeo y sonido**
Rpi 1 modelo A+; B+	BCM2835 a 700MHz	512MB	$20; 25	Ninguna y Fast Ethernet	1P USB; y 4P USB HDMI Puerto CSI y DSI
Rpi Zero; Zero W	BCM2835 Single-Core 1GHz	512MB	$5; $10	Ninguna; Wireless LAN (Band 2.4GHz), Bluetooth 4.1	Mini HDMI 1P Micro USB Puerto CSI
Rpi 2 modelo B	BCM2836/7 Quad-Core 900MHz	1GB	$35	Fast Ethernet	Puerto compuesto RCA-HDMI Puerto CSI y DSI 4P USB 2.0
Rpi 3 modelo A+	BCM2837B0 Quad-Core 1.4GHz	512MB	$25	Wireless LAN (Dual-band) Bluetooth 4.2	HDMI norma Puerto CSI y DSI 1P USB 2.0
Rpi 3 modelo B	BCM2837A0/B0 Quad Core 1.2GHz 64bit	1GB	$35	Wireless LAN (Band 2.4GHz) Bluetooth 4.1 Fast Ethernet	HDMI normal Puerto CSI y DSI 4P USB 2.0
Rpi 3 modelo B+	Quad Core 1.4GHz Broadcom 64-bit	1GB	$35	Wireless LAN (Dual-band) Bluetooth 4.2 Giga-Ethernet	Puerto HDMI normal Puerto CSI y DSI 4P USB 2.0
Rpi 4 modelo B	BCM2711 Quad-core de 1.5GHz	2GB 4GB 8GB	$35 $55 $75	Wireless LAN (Dual-band) Bluetooth 5.0 Giga-Ethernet	2P micro HDMI(4Kp60) Puerto DSI y CSI. 2P 3.0 2P USB 2.0

La Raspberry Pi, es el elemento central de nuestro proyecto, dado que esta plataforma contiene la lógica necesaria para controlar todo el conjunto de sensores, procesar los datos adquiridos y almacenarlos. En base a las características de cada modelo presentadas en la Tabla 4.1, se decidió escoger la Raspberry Pi 3 modelo B+ (Figura 4.3 (a)) para uno de los nodos y la Raspberry Pi 4 modelo B (Figura 4.3 (b)) de 4GB para el otro nodo, esto debido a que estos *SBC* cumplen con los requisitos necesarios para su uso en el desarrollo del proyecto, gracias a características como por ejemplo, el número de puertos USB, los codecs y resolución de vídeo que soporta (v.g. H.264 1080p60), el precio, la velocidad

de procesamiento, etc. Se debe mencionar además, que cada plataforma de la fundación Raspberry Pi cuenta con 40 pines que permiten la conexión de algunos de los sensores a la plataforma. En la Figura 4.3(c), se muestra la distribución de los *GPIO* y los diferentes tipos de entradas y salidas de las que disponen los *SBC* escogidos para el desarrollo de este proyecto.



(a) Raspberry Pi 3B +



(b) Raspberry Pi 4B

BOARD	GPIO		GPIO	BOARD
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

(c) *GPIO*

Figura 4.3: Componentes del los *SBC*

4.3.1.1. Características de los *GPIO*

Como se puede observar en la Figura 4.3(c), los pines cuentan con dos conexiones de alimentación de de 5V y dos de 3.3V, 8 puertos de Tierra, 2 puertos reservados para la comunicación con la EEPROM de la *SBC* y 26 puertos *GPIO* que se pueden usar como entradas y salidas digitales. La *SBC* permite además, funciones adicionales que se pueden desarrollar en puertos *GPIO* específicos por ejemplo:

- *GPIO* 02 y 03 (Pin 3 y 5): Permiten usar la comunicación I^2C, para controlar varios periféricos en conexión paralelo, cabe recalcar que los periféricos también deben soportar la comunicación *I2C*.
- *GPIO* 04, 05 y 06 (PIN 7 29 y 31): Se puede configurar como reloj de uso general para emitir una frecuencia fija sin ningún control de software continuo.
- *GPIO* 9, 10, 11 +7/8 (PIN 21, 19, 23 + 25/24): Conforman el Bus *SPI* 0, que permite una función de trasmisión de datos en serie síncrono que se usa para la comunicación con los circuitos

integrados usando las señales *SCLK* (11) , *MOSI* (10), *MISO* (9) y *Chip Enable (CE)* (7 y 8), los *GPIO* 19, 20 y 21 +16/17/18 se pueden usar como un Bus *SPI* 1.

- *GPIO* 12 y 13 (PIN 32 y 33): Pines generalmente usados para realizar *Pulse Width Modulation (PWM)*.
- *GPIO* 14 Y 15 (PIN 8 y 10): Se usan para brindar una comunicación *UART*, que es un protocolo de comunicación serie asíncrono, lo que significa que toma bytes de datos y transmite los bits individuales de forma secuencial. La transmisión asíncrona permite que los datos se transmitan sin que el emisor tenga que enviar una señal de reloj al receptor. En cambio, el emisor y el receptor acuerdan los parámetros de temporización por adelantado y se agregan bits especiales llamados "bits de inicio.ª cada palabra y se utilizan para sincronizar las unidades de envío y recepción.

4.3.1.2. Sistema Operativo

La plataforma Raspberry Pi, es compatible con una gran variedad de sistemas operativos que usan bajos recursos, como: *Kali Linux*, *Pidora*, *Windows 10 IoT core*, *Ubuntu Core*, *Arch Linux ARM*, *FreeBSD*, *Raspbian X Nighthawk*, *iRaspbian y Raspbian* que es el sistema operativo oficial para Rasberry pi.

- *Raspbian.*- El sistema operativo *Raspbian* fue diseñado específicamente para ser usado en las plataformas Raspberry Pi, es una distribución de *Linux* basado en *Debian*, por lo cual es de *software* libre y código abierto. Características fundamentales que han permitido que los desarrolladores sigan optimizando este *software* con la finalidad de mejorar el manejo de paquetes para el control de los elementos de *hardware*. El sistema operativo es de 32 bits y puede ser instalado de dos formas diferentes, siendo *NOOBS* la forma más sencilla, dado que únicamente requiere descargar y descomprimir los archivos de instalación sobre una memoria micro SD formateada como FAT. Se debe mencionar además, que el sistema operativo cuenta con su versión recomendada y su versión *lite*.
  Para el desarrollo de este proyecto utilizaremos la versión recomendada de *Raspbian*, debido a que tiene integrada la interfaz gráfica del escritorio que simplifica el manejo de la *SBC* y para la instalación del sistema operativo se usará el modo *NOOBS*.

4.3.2. Cámara de vídeo

Para elección de la cámara de vídeo se empleó dos criterios claves, el primero es la capacidad para integrarse al puerto *Camera Serial Interface (CSI)* del *SBC* y la segunda es la resolución que dispone. Bajo estas condiciones se decidió utilizar una versión genérica del módulo cámara propio de la fundación Raspberry Pi versión q1. Dicho módulo dispone de 5 megapíxeles, lo que permite una visualización de calidad de imágenes de 20x25cm y con resolución de 2592x1944 píxeles. En cuanto al vídeo, se cuenta con una resolución de vídeo de hasta 1800p, en la Tabla 4.2, se muestra las características principales del módulo usado.

Tabla 4.2: Características de la cámara

Parámetro	Característica
Modelo	CMT-5MP-RP-OV5647-X010
Sensor	Omnivision OV4647 de 1/4
Pixel	5 megapíxeles
Píxeles más efectivos	2592 x 1944
Tamaño de píxel	1,4 um x 1,4 um
Tipo de enfoque	50 Foco fijo
FOV	62 °
Formato de imagen	VGA
Color	Color verdadero de 24 bits
Construcción de lentes	4G + IR650
Abertura	F2.4
Temperatura de funcionamiento	- 20 ° C t o 70 ° C
Dimensión	25 mm x 24 mm
El consumo de energía	90Wm / 30uW
Voltaje	1,7 V-3,0 V
Número de PIN	15 pines
Señal de salida	YCBCR4: 2: 2, RGB565, RAWRGB
Cuadros por segundo	30 fps codec H264 (AVC)

4.3.3. Sensor de *MP*

El análisis del *MP* a los que un ciclista está expuesto es uno de los puntos claves a la hora de analizar su confort en una ruta, dado que como se menciona en el estado del arte, estas partículas son perjudiciales para la salud de las personas y estas debidamente reglamentadas en la norma ecuatoriana de calidad del aire. Para el desarrollo de este trabajo se empleó el sensor Nova SDS 011 que se muestra en la Figura 4.4, se eligió este sensor debido a las características de las que dispone, como el costo, la alimentación, la forma de recolección de los datos, el porcentaje de error relativo, las dimensiones, etc. En la Tabla 4.3 se presenta las características principales del sensor.

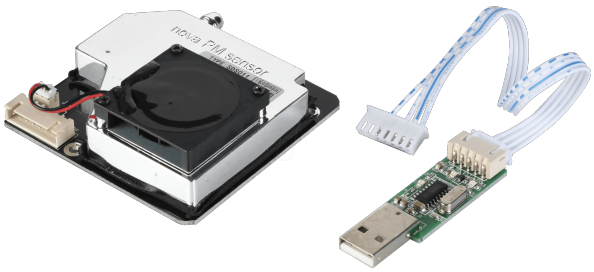


Figura 4.4: Sensor de *MP* Nova SDS 011

Tabla 4.3: Características del sensor SDS011 [11]

Tipo de comunicación	UART
Parámetros de medición	2.5PM,10PM
Rango	0.0-999.9 ug/m^3
Voltaje de alimentación	5V
Corriente requerida	70mA $\pm 10mA$
Tiempo de respuesta	1s
Frecuencia de salida	1Hz
Resolución mínima	0.3um
Rendimiento de conteo	70 % a 0.3um 98 % a 0.5um
Error relativo	$\pm 15\,\%y \pm 10$ug/m^3
Dimensiones	71*70*23mm
Costo	$28

El sensor Nova SDS011 utiliza el principio de detección de partículas por dispersión láser, es decir, detecta la dispersión de la luz cuando el *MP* atraviesa el área de detección. El porcentaje de luz dispersa se trasforma en señales eléctricas que posteriormente se amplifican y procesa. Para detectar el número y el diámetro de las partículas se usa la forma de onda de la señal resultante. Se debe mencionar que este sensor cuenta con la certificación CE, FCC y RoHS, las mismas que garantizan que la electrónica del producto no es nociva y cumple con los requisitos de conformidad. Además, como se puede observar en la Figura 4.4, el sensor puede ser controlado por el puerto USB. En cuanto a sus limitaciones, la principal desventaja es su vida útil, la cual el fabricante estima en un promedio de 8000 horas.

El sensor contiene siete pines, el primero no tiene conexión, el segundo es la salida *PWM* de las lecturas de las *MP* de 2.5um, el tercero es la alimentación de 5V, el cuarto es la salida *PWM* de las lecturas de las *MP* de 10um, el pin 5 es la tierra, el pin 6 y 7 se usan para la comunicación *UART* a 3.3V siendo el pin 6 el receptor y el pin 7 el transmisor. Se debe mencionar que la comunicación *UART* se realiza a un *bit rate* de 9600 *baudios* y el equipo completo está controlado por un *CPU* de 8 bits.

4.3.3.1. Integración del sensor Nova SDS-011 al *SBC*

Para utilizar el sensor SDS 011 con la Raspberry Pi se requiere instalar librerías previas, como, *Git* que es una herramienta fundamental para el control de versiones de *software*, *Python Serial* que permite establecer las características de la comunicación entre el sensor y el *SBC* y *Python enum* para la detección del sensor cuando este se conecta a uno de los puertos del *SBC*. La instalación de las librerías, se realiza mediante el comando **sudo apt install git-core python-serial python-enum**. Una vez finalizado el proceso anterior se conecta el sensor a uno de sus puertos y con el comando **dmesg** se analiza el ID del puerto al cual se conectó y mediante la librería "py-sds011" se inicia el sensor y se comienza a recibir los datos correspondientes. La librería trabaja con los registros del controlador mediante comandos en ensamblador.

4.3.4. Sensor de calidad del aire

Otro de los sensores que se utiliza para el desarrollo del proyecto es el sensor MQ-135 el cual permite detectar el nivel de diferentes gases como el Amoníaco (NH_3), Dióxido de nitrógeno (NO_2), Alcohol, Benzeno (C_6H_6), Dióxido de carbono (CO_2) y Monóxido de carbono (CO). Se debe mencionar que en el desarrollo de proyecto se requiere medir únicamente el nivel de CO_2 el mismo que permitirá establecer un incremento de trafico vehicular, dado que este, es el segundo gas con mayor presencia en las emisiones de un vehículo a combustión interna. Si bien, existe muchos sensores que desarrollan funciones similares en el mercado, estos no se acoplan adecuadamente al desarrollo del proyecto debido a su tamaño, la velocidad de respuesta y por la forma en la que se entrega los datos capturados, características esenciales para su integración en la estructura del nodo y comunicación con el SBC. El MQ-135 es un sensor electroquímico que varía su resistencia cuando es expuesto a los gases mencionados anteriormente. En su interior el sensor tiene un calentador que funciona con 5V y es usado para detectar los gases a través de los cambios de temperatura. El módulo que lo contiene, integra la resistencia de carga R_L usada para crear un divisor de voltaje que permita leer los datos desde un microcontrolador. Se debe mencionar además que para su correcto funcionamiento el sensor debe realizar un período de calentamiento hasta que su salida sea estable. En la Figura 4.5, se muestra el sensor MQ-135 integrado sobre un módulo, que es la forma, como generalmente se consigue en el mercado. El módulo simplifica el manejo del sensor mediante el uso de 4 pines, donde el primero corresponde al voltaje de alimentación (5v), el segundo sera la tierra y el tercero y cuarto serán los pines de salida, que, como se puede observar en la Figura 4.5, una es analógica y la otra digital.

Figura 4.5: Sensor de calidad del aire MQ-135

4.3.4.1. Salida Digital y Analógica

La salida digital del sensor MQ-135 toma únicamente dos valores que indican la ausencia de un gas con un 1 lógico (5V) y la presencia del mismo con un 0 lógico (0V). La sensibilidad del sensor se puede manipular a través de la resistencia variable permitiendo que el sensor se pueda activar con diferentes niveles de gas, se debe mencionar que el rango de captura del sensor es de $10ppm$ hasta las $1000ppm$.

La salida analógica del sensor MQ-135, puede mostrar diversos niveles de concentración de gas, dado que el sensor tiene una resistencia variable (R_s) incorporada, que cambia su valor de acuerdo a los niveles de mismo. Si la concentración es alta, el valor de R_s disminuye y viceversa. Debido a este comportamiento, para funcionar el sensor requiere de una resistencia de carga (R_L), la cuál en este

caso esta integrada en el módulo y sirve para ajustar sensibilidad del sensor. Según las recomendaciones del fabricante la resistencia debe variar entre $10k\Omega$ a $47K\Omega$. Se debe tener en cuenta que, a mayor resistencia más sensibilidad tendrá el sensor, dado que el sensor tiene la capacidad de detectar diferentes tipos de gases y concentraciones de los mismos. El ajuste de la sensibilidad es un punto fundamental para poner en funcionamiento el sensor de forma analógica.

4.3.4.2. Integración al *SBC*

Para calibrar el sensor MQ-135, el fabricante recomienda que se realice para $100ppm$ de NH_3 en el aire limpio y usar un valor de resistencia de R_L de aproximadamente $20K\Omega$, bajo estas condiciones, en la Figura 4.6, se muestra los valores de sensibilidad típicos para varios gases. Esta gráfica, además permite determinar los niveles de concentración de gas en partes por millón de acuerdo a la relación R_s/R_o, donde R_o es la resistencia del sensor a una concentración conocida sin la presencia de otros gases o en el aire fresco.

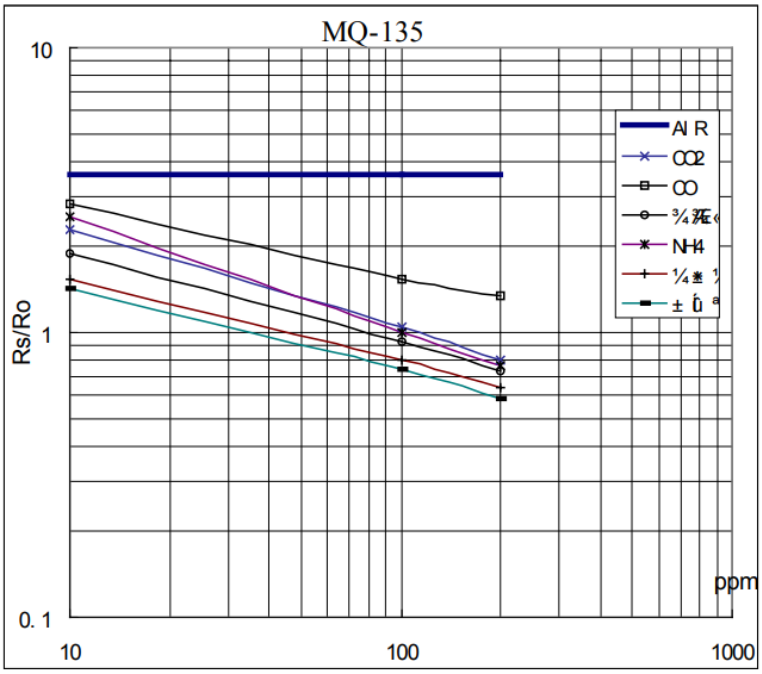


Figura 4.6: Tabla de sensibilidad del sensor MQ-135 [6]

Debido al complejo proceso de calibración del sensor, se ha buscado otra forma de ajustar al sensor para obtener datos veraces. En general, lo que se busca con este sensor es detectar la contaminación presente en el aire causado por los vehículos principalmente. Además, este sensor percibe al CO_2 el cuál es el gas que mayormente producen los vehículos de combustión interna. En particular, la calibración del sensor se realizó en base a la metodología recomendada en [44]. El primer paso para calibrar es estabilizar el sensor en cero, para ello, se realizó mediciones en aire puro y mediante la variación de la resistencia R_L se obtuvo un valor aproximado. Este modo de calibración permite que cuando el sensor perciba saturación de gases en el aire, los valores entregados sean superiores a cero, permitiendo de esta manera registrar el aumento de gas contaminante. Al realizar las pruebas de campo su verificó el correcto funcionamiento, el cual se mostrará en el capítulo 5 .

Dado que, se usará como procesador central la Raspberry Pi, se necesita un *ADC* que permita muestrear los valores analógicos a digitales, para este proceso se utiliza el integrado MCP3208 de 8 bits que se integra en los pines *MOSI*, *MISO*, *SCLK* y *CE*0 de la Raspberry Pi, con lo cual se logra una comunicación serial síncrona por el Bus *SPI*. Al utilizar el *ADC*, se debe tener en consideración que los valores recibidos por la Raspberri Pi son valores cuantizados del nivel de voltaje de salida del módulo, para conocer los valores reales se debe aplica aplicar la ecuación:

$$V_{RL} = \frac{Valoranalógico}{4095} * 5 \tag{4.1}$$

4.3.5. Sensor de corriente

El análisis del consumo energético es un punto clave para el desarrollo de nuestro proyecto, esto nos permitirá dimensionar la capacidad mínima de la batería necesaria para que funcione cada uno de los nodos por al menos 30 minutos, tiempo promedio de uso en el sistema de bicicletas públicas de la ciudad de Cuenca. Los datos permitirán además determinar el nivel de consumo de potencia eléctrica en cada recorrido del usuario, el tiempo máximo de funcionamiento del nodo. Con estos datos se podrá detectar anomalías en el funcionamiento del sistema, lo que permitirá brindar una respuesta rápida antes de que se vea comprometido todo el nodo.

Teniendo en consideración los puntos anteriores y las recomendaciones de la fundación Raspberry Pi respecto al uso de una fuente de alimentación de $5V$ y $2{,}5A$ para el modelo 3B+ y de $3A$ para el modelo 4B. Para realizar las lecturas del consumo energético de los nodos se eligió usar el sensor INA219 (Figura 4.7) debido a sus características propias como:

- Rango de medición de corriente: $\pm$ $3{,}2A$
- Rango de medición de voltaje: $26V$
- Voltaje de alimentación: $3{,}3V$ o $5V$
- Tipo de comunicación: I^2C
- Resolución de corriente: $\pm$ $0{,}8mA$
- Dimensiones: 2,3 x 2,1cm ($09x08in$)
- Costo: $12

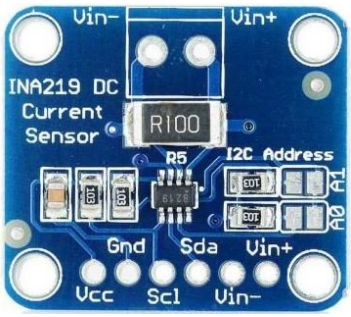


Figura 4.7: Sensor de corriente INA219

4.3.5.1. Integración al *SBC*

Como se mencionó el sensor dispone de la comunicación *I2C*, esta característica permite usar los canales *SDA* (Pin 3) y *SCL* (Pin 5) de la Raspberry Pi para el manejo de este elemento. El primer paso en la integración consiste en determinar si el sensor es reconocido por el *SBC*, para ello se aplica el comando **sudo i2cdetect -y 1**, donde el elemento debe ser visualizado con una dirección 40 en hexadecimal sobre el Bus 0 o 1, una ves detectado se usa la librería pi_ina219 que permite controlar el sensor simplificando la complejidad de configuración. Esta librería se puede integrar en python 2 y python 3, para su instalación sobre el *SBC* se debe aplicar el comando **sudo pip3 instalar pi-ina219**.

Una vez instalada la librería pi_ina219, se configura los siguientes parámetros para poder lograr los resultados con la mejor resolución de corriente posible, se debe mencionar que la librería esta diseñada para emitir mensajes de *DeviceRangeError* cuando las lecturas superan los valores configurados. A continuación se muestra algunas características del sensor.

Identificación: Permite identificar los elementos físicos y lógicos del sensor.

- *shunt_ohms*: Valor de resistencia de derivación en *ohms*.
- *max_expected_amps*: Valor de máximo esperado de corriente en Amperios.
- *busnum*: El número de Bus I^2C donde se detecta el sensor.
- *address*: La dirección I^2C del sensor (por defecto 0x40).

Configuración: Permite configurar los diferentes parámetros de medición.

- *voltage_range*: Establece el rango de voltaje máximo a 16V o 32V.
- *gain*: Controla el rango máximo de tensión de derivación en 40mV, 80mV, 160mV, 320mV y automática
- *bus*adc*: Establece la resolución del ADC en 9 bits con tiempo de convergencia de 84us, 10 bits con tiempo de convergencia de 148us, 11 bits con tiempo de convergencia de 276us y 12 bits con tiempo de 532us, o establecer el número de muestras usadas al promediar los resultados en 2, 4, 8, 16, 32, 64 y 128 muestras.
- *shunt*adc*: Al igual que en el caso anterior establece la resolución o el número de muestras utilizadas para promediar los resultados.

Salida: Son las funciones usadas para obtener los resultados.

- *voltaje()*: Voltaje del bus (V)
- *supply_voltage()*: Voltaje de alimentación del bus (V).
- *current()*: Corriente del bus (mA)
- *power()*: Consumo de energía del bus (mW)
- *shunt_voltage()*: Voltaje de derivación (mV)
- *current_overflow()*: Permite visualizar si se produce un desbordamiento en las lecturas (Verdadero).
- *sleep()*: Apaga el sensor
- *wake()*: Despierta el sensor
- *reset()*: Restablecer la configuración del sensor.

Para el desarrollo del proyecto únicamente se usará (*current()*) la corriente y la potencia (*power()*), por lo cual la configuración realizada sobre la librería es:

- *shunt_ohms* = 0,1Ω
- *max_expected_amps* = 3*A*
- *busnum* = 1
- *address*: = 0*x*40
- *voltage_range*=RANGE*16V
- *gain*=GAIN*8_320MV
- *bus*adc*=ADC*12BIT
- *shunt*adc*=ADC*12BIT

4.3.6. Sensor de vibración

Para esta parte se usa un acelerómetro, el cual es un dispositivo que mide la vibración o la aceleración del movimiento de una estructura. La fuerza generada por la vibración o el cambio en el movimiento (aceleración) hace que la masa comprima el material piezoeléctrico, generando una carga eléctrica que es proporcional a la fuerza ejercida sobre el mismo. El hecho de que la carga sea proporcional a la fuerza y que la masa sea constante hace que la carga también sea proporcional a la aceleración.

Se ha elegido el ADXL345 que es un módulo acelerómetro MEMS de 3 ejes de baja potencia con interfaces *I2C* y *SPI*, puede ser alimentado con 3.3 o 5V. El ADXL345 presenta 4 rangos de sensibilidad desde +/- 2G a +/- 16G, y admite velocidades de datos de salida que van desde 10 Hz a 3200 Hz. En la Figura 4.8, se observa el sensor.

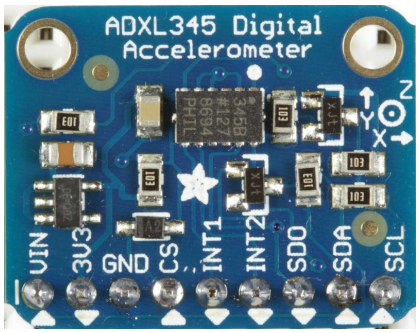


Figura 4.8: Acelerómetro Adxl345

4.3.6.1. Integración al *SBC*

El ADXL345 Breakout tiene una dirección *I2C* de 0x53. Este sensor comparte el Bus *I2C* con el sensor de corriente descrito anteriormente. Cabe recalcar se puede compartir el Bus, dado que, ambos sensores disponen de una dirección única I2C. Se requieren 4 conexiones para la comunicación *I2C* con la Raspberry Pi:

- GND: GND
- VIN: 5V
- SDA: SDA (GPIO 4)
- SCL: SCL (GPIO 3)

Existen múltiples librerías para detectar movimiento, caída libre, vibraciones, etc. Luego de verificar cada una de las librerías, se decidió realizar una calibración personalizada para detectar las vibraciones.

El acelerómetro proporciona valores en los ejes "X", "Y", "Z", como se muestra en la Figura 4.9, se puede observar que la gravedad actúa sobre el eje "Z". La orientación del sensor es el punto de partida de la calibración, ya que se trabajará con el eje que se encuentre hacia abajo. En nuestra aplicación se usará el eje "Y" (perpendicular al suelo), para calibrar el sensor. Se realizaron múltiples pruebas sobre diferentes calzadas, en las cuales se van tomando los datos del sensor en el eje "Y". El eje está "Y" ubicado en posición perpendicular y cuando se mantiene en reposo entrega valores de $9{,}8m/s^2$ aproximadamente. Mientras que cuando se mueve de manera brusca este valor tiene variaciones en entre $0m/s^2$ y $12m/s^2$. Basándonos en este hecho, se realiza un muestreo de los cambios que realiza el eje "Y" en un intervalo de tiempo, para posteriormente interpretar esos datos y establecer el estado de la vía. Entonces, cuando el nodo es puesto en la bicicleta, mientras realice un recorrido por diferentes vías, se toma un registro de las variaciones de aceleración entregadas por el eje "Y".

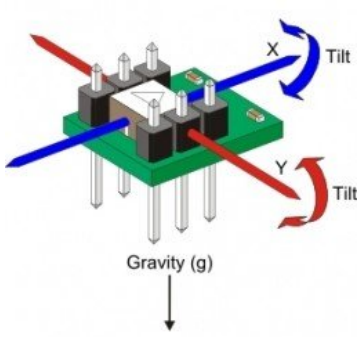


Figura 4.9: Ejes del acelerómetro [7]

4.3.7. Sensor de Distancia

Los sensores de distancia permitirán establecer el espacio entre un obstáculo lateral y bicicleta en el recorrido. En este contexto, se colocará un sensor a la izquierda y derecha del nodo, los cuales retornaran un valor en centímetros indicando la distancia a la que se encontró un obstáculo. Este puede ser un vehículo, bicicleta o alguna pared.

El sensor de distancia debe tener características que se ajusten al funcionamiento del nodo. Al ser un registrador de datos que actúa en tiempo real y registra datos continuamente se requiere un dispositivo sensor que tenga una repuesta rápida. El rango de medición es un parámetro esencial para la elección del sensor. En este caso, no se requiere medir distancias largas. Al contrario se necesita detectar el momento en el que un obstáculo se encuentra cerca de la bicicleta. Por último se requiere que el sensor pueda comunicarse con la Raspberry Pi.

De acuerdo a los requerimientos mencionados se ha elegido el sensor ultrasónico. El sensor emite ultrasonidos, los cuales se propagan a velocidad constante del sonido $340m/s$. Estas señales de sonido emitidas tienen una frecuencia de $40KHz$ y están fuera del rango de frecuencias percibidos por el oído humano. Además, este tipo de ondas pueden sufrir fenómenos de reflexión, refracción y difusión. Aprovechando la reflexión de la onda se puede percibir cuando existe un obstáculo en el medio, ya que se produce un rebote de la onda a la misma velocidad y en dirección contraria. Con el sensor ultrasónico usando esta propiedad se puede calcular el tiempo que tardó la onda en ir y regresar [45].

Los sensores ultrasónicos generalmente están formados por dos transductores de forma cilíndrica

como se observa en la Figura 4.10, los cuales son emisor y receptor, es decir un transductor emite el ultrasonido y el otro recepta el mismo. Además, cuenta con un pin de entrada y salida digital. Cuando se requiera emitir una señal de sonido el pin de salida debe tomar un nivel lógico alto por un tiempo de 10 a 600us, para evitar grandes trenes de señales. Luego, el pin de entrada cambiará a estado lógico alto cuando se haya recibido la señal rebotada. Por último, se calcula la distancia con el tiempo y la velocidad ya conocidos.

Figura 4.10: Sensor ultrasónico HC-SR04

Las características principales del sensor se resumen a continuación:

- Voltaje: 5VDC, máx 5,5V y mín 4,5V
- Corriente: 15mA, máx 20mA, mín 1mA
- Detección de distancia: 2cm a 4cm
- Precisión: variación hasta 3mm
- Ángulo de medición: 30^o, efectivo inferior a 15^o

4.3.7.1. Integración al *SBC*

Los pines de conexión del HC-SR04 son cuatro y se muestran en Figura 4.10. El pin VCC se conectará al pin de 5V de la *GPIO* del Raspberry Pi, mientras que el pin GND se conectará al cualquier pin GND de la Raspberry Pi. El pin Trig, es el terminal de salida, deberá ser conectado a un pin digital de la Raspberry Pi (previamente declarado como salida). El pin Echo es el pin de entrada, se conectará a un pin digital de la Raspberry Pi así mismo previamente declarado como entrada.

Una vez que se tenga conectado el sensor al Raspberry Pi se realiza la comunicación entre ellos. Como se indicó anteriormente se necesita emitir y receptar el ultrasonido. Esto se logra cambiando el estado del pin de salida a un nivel lógico alto que activa el Trig y se genera la señal ultrasónica. El tiempo de duración del pin de salida debe ser de 1 us y luego el pin de entrada deberá esperar hasta que capte la señal de retorno como resultado del rebote. Además, en todo este proceso se debe estar registrando el tiempo de salida y entrada de la señal.

El escenario que se busca capturar es cuando un obstáculo moleste al ciclista. Por lo tanto la lógica usada para obtener los datos es registrar la distancia mínima a la que haya estado expuesto el ciclista durante un intervalo de tiempo. Mientras que cuando se capten valores que no afecten al confort del ciclista se entregara un valor constante que indique que no hay molestias en la vía. Entonces, el nodo estará enviado los datos cada 2 segundos en los cuales se tomarán 10 muestras con un intervalo de 0,2 s y se enviará la distancia más corta detectada.

4.3.8. Sensor de ruido ambiental

La mayoría de los sensores de sonido presentan circuitos integrados, que cuentan con una etapa de pre-amplificación de la señal, sin embargo se requiere otra etapa de amplificación que en muchos de los casos se requiere de un circuito aparte. Por tal razón se ha elegido el sensor MAX9814 descrito en la Figura 4.11, ya que este sensor cuenta con las dos etapas de amplificación en el mismo dispositivo.

Figura 4.11: Sensor de sonido MAX9814

El sensor micrófono MAX9814 posee 5 pines (*Vdd, Gnd, Gain, Out, Ar*) de los cuales para el proyecto utilizaremos: Vdd para la alimentación del sensor con $5v$, *Gnd* para la conmutación a tierra y *Out* es la señal amplificada con valores analógicos censados del micrófono que serán receptadas por el *ADC* mcp3208 y posteriormente enviadas a la Raspberry Pi. Esta configuración nos permite utilizar el sensor micrófono para que trabaje en el valor de ganancia de $60dB$. Sin embargo, esto puede causar saturaciones, por lo que se conecta el pin *Gain* a *Gnd* para trabajar con $50dB$ de ganancia. A continuación se muestra las características de este sensor:

- Ganancia: 60/50/40 dBA
- Dimensión: $26mm \times 14mm$
- Rango de frecuencia: $20Hz - 20kHz$
- Corriente de trabajo: $< 3mA$
- Amplificación: $X100$

La señal de audio de la salida del amplificador es un voltaje variable. Para medir el nivel de sonido, se necesita tomar múltiples muestras para encontrar el valor mínimo y máximo o amplitud pico a pico de la señal. Para este proyecto se eligió una ventana de muestra de 50 milisegundos. Eso es suficiente para medir niveles de sonido de frecuencias tan bajas como 20 Hz, el límite inferior del oído humano. Después de encontrar las muestras mínima y máxima, se calcula la diferencia y se la convierte a voltios, una vez obtenida estos valores se realiza un mapeo a dB. La calibración del sensor se la realizó de forma manual, usando un sonómetro de un dispositivo móvil.

4.3.9. Modulo 4G, GPS y Velocímetro

En esta parte se describe tres componentes de la estación. La razón para describirlos juntos es que estas funciones se incluyen en un solo módulo llamado SIM7600A-H. El módulo cuenta con una cabecera de pines que calzan en la *GPIO* del Raspberry Pi. Entonces debido a lo mencionado y demás características que se describen a continuación tomadas del *datasheet* [8], lo hacen el ideal para este proyecto.

El SIM7600A-H es un módulo para la comunicación $4G/3G/2G$ y posicionamiento global. Admite *LTE* de hasta $150Mbps$ para la transferencia de datos hacia Internet. También puede ser conectado a la Raspberry Pi o al ordenador para ser usado como módem con salida a Internet. Otra de las funciones es hacer llamadas telefónicas y enviar mensajes. Además es de bajo consumo de energía. Sus características son:

- Conectividad compatible con Raspberry Pi Zero/Zero W/Zero WH/2B/3B/3B+
- Admite acceso telefónico, llamadas telefónicas, SMS, MMS, correo, TCP, UDP, DTMF, HTTP, FTP, etc.
- Admite posicionamiento de estación base GPS
- Interfaz USB integrada, para probar comandos AT, obtener datos de posicionamiento *GPS*, etc.
- Convertidor CP2102 USB a *UART* integrado, para depuración en serie
- Pines de control *UART* de ruptura, para conectar con placas de *host* como *Arduino* / STM32
- Ranura para tarjeta SIM, admite tarjeta SIM de 1.8V / 3V
- Ranura para tarjeta TF para almacenar datos como archivos, mensajes, etc.
- Conector de audio integrado y decodificador de audio para realizar llamadas telefónicas
- 2 indicadores LED, fáciles de controlar el estado de trabajo
- Traductor de voltaje a bordo, el voltaje de funcionamiento se puede configurar en 3.3V o 5V a través de un puente
- Velocidad en baudios: $300bps$ a $4Mbps$ (predeterminado: $115200bps$)
- Velocidad de transmisión automática: $9600bps$ a $115200bps$
- Control mediante comandos AT (3GPP TS 27.007, 27.005 y conjunto de comandos V.25TER)

En la Figura 4.12, se muestra las partes que componen al módulo, de la cual se partirá para indicar el proceso de conexión y control del módulo.

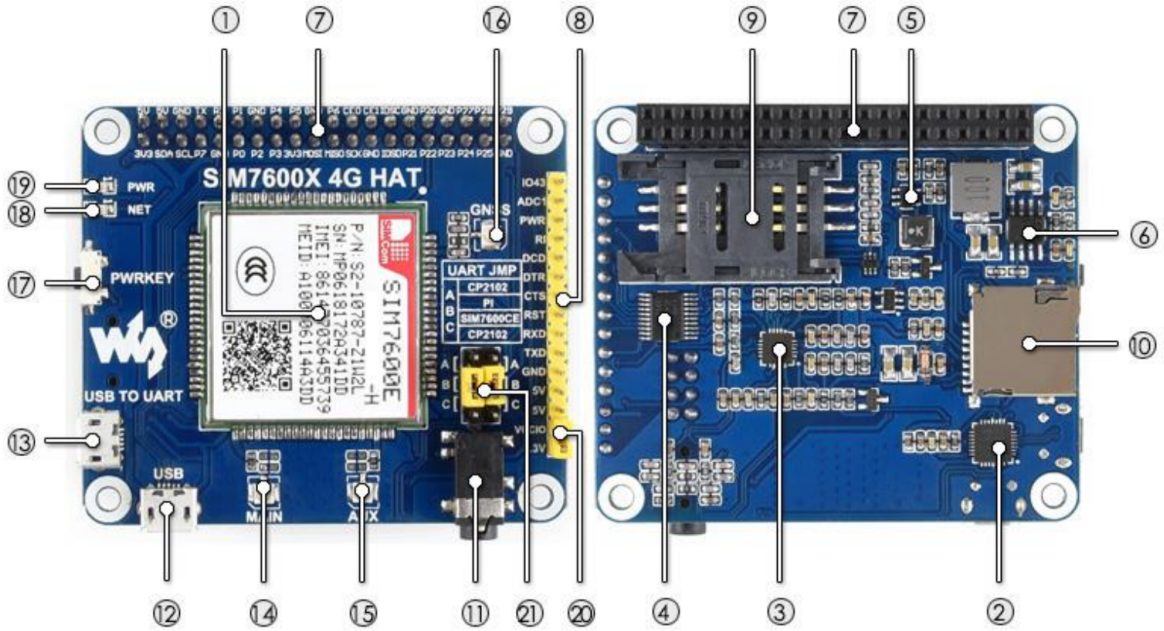


Figura 4.12: Partes del módulo SIM7600 [8]

1. SIM7600E-H
2. Convertidor CP2102 USB a *UART*
3. Decodificador de audio NAU8810
4. Traductor de voltaje TXS0108EPWR: traduce 3.3V / 5V en 1.8V
5. Chip de potencia MP2128DT
6. Chip de potencia MP1482
7. Cabecera *GPS* de Raspberry Pi: para conectar con Raspberry Pi
8. Interfaz de control SIM7600: para conectar con placas de host como Arduino / STM32
9. Ranura para tarjeta SIM: admite tarjeta SIM de 1.8V / 3V
10. Ranura para tarjeta TF: permite almacenamiento de archivos / SMS / ...
11. Conector para auriculares / micrófono de 3,5 mm
12. Interfaz USB: para probar comandos AT, obtener datos de posicionamiento GPS, etc.
13. Interfaz USB a *UART*: para depuración en serie o inicie sesión en Raspberry Pi
14. Conector de antena PRINCIPAL
15. Conector de antena AUX
16. Conector de antena GNSS
17. Interruptor de encendido
18. Indicador de estado de la red
19. Indicador de encendido
20. Puente de selección de voltaje de funcionamiento:
    VCCIO - 3,3 V: establezca el voltaje de funcionamiento en 3,3 V
    VCCIO - 5V: establezca el voltaje de funcionamiento en 5V
21. Puente de selección *UART*:
    A: acceda a Raspberry Pi a través de USB a *UART*
    B: controla el SIM7600 por Raspberry Pi
    C: controla el SIM7600 a través de USB a *UART*

4.3.9.1. Integración al *SBC*

La configuración del *hardware* inicia por la colocación de la tarjeta SIM en la ranura del módulo ubicada en la parte de abajo. Esta tarjeta tiene dimensiones de $25mm$ x $15mm$ (Mini-SIM) y puede ser de cualquier operadora disponible. Luego, el módulo SIM7600 se conecta a la Raspberry Pi a través de la *GPIO* y dos puertos USB. La cabecera *GPIO* da acceso a todos los pines de la Raspberry Pi, mientras que la conexión USB proporciona interacción entre módulo y la Raspberry Pi.

La señal de la red móvil y *GPS* son receptadas a través del conector principal y GNSS respectivamente indicados en la Figura 4.12. Las antenas se conectan mediante un cable de extensión ufl a SMA. Cabe recalcar que el conector auxiliar también sirve para la red móvil. Por otro lado, hay que verificar que el puente de selección *GPIO* indicado en el numeral 21 de la Figura 4.12, este en B, para controlar el SIM7600 por medio de la Raspberry Pi.

4.3.9.2. Configuración *GPS* y Velocímetro

Los datos de posicionamiento *GPS* se obtiene a través de la interfaz USB mediante comandos AT. La comunicación se realiza por el puerto ttyS0 de la Raspberry Pi. La velocidad en baudios es 115200 por defecto. Para comprobar la conexión entre los dispositivos se envía comandos AT mediante un script

realizado en *Python*. En [8] se lista los comandos usados para obtener los datos de posicionamiento. El primer paso es activar el *GPS* con el comando $AT + CGPS = 1$ y su respuesta es OK, en caso de que el GPS ya este activado la respuesta será $ERROR$. Luego la antena GPS debe ser colocada en una área abierta al aire libre. La posición del *GPS* se obtiene a través del comando $AT + CGPSINFO$. Al inicio el posicionamiento puede tardar varios segundos y hasta minutos dependiendo del área. Sin embargo una vez que el *GPS* detecte su posición, en los siguientes datos no habrá retardos. El la Tabla 4.4, se resume el comando y su respuesta.

AT+CGPS=1	OK (Encender *GPS*)
AT+CGPS=0	OK (Apagar *GPS*)
AT+CGPSINFO	[<lat>],[<N/S>],[<log>], [<E/W>],[<fecha>],[<hora UTC>], [<alt>],[<velocidad>],[<course>]

Tabla 4.4: Comandos para obtener datos del GPS

- <lat>Latitud de la posición actual. El formato de salida es ddmm.mmmmmm
- <N/S>Indicador N / S, N = norte o S = sur
- <log>Longitud de la posición actual. El formato de salida es dddmm.mmmmmm
- <E/W>Indicador E / W, E = este o W = oeste
- <fecha>Fecha. El formato de salida es ddmmyy
- <Hora UTC>Hora UTC. El formato de salida es hhmmss.s
- <alt>Altitud MSL. La unidad son metros.
- <velocidad>Velocidad sobre el suelo. La unidad son nudos.
- <Inclinación>Inclinación del *GPS* en grados.

En la Tabla anterior se observa que los datos obtenidos son varios de los cuáles se usa la latitud, longitud y velocidad. Sin embargo hay que realizar una conversión de unidades ya que se requiere las coordenadas en grados y la velocidad en km/h.

4.3.9.3. Configuración Red Móvil

La configuración de la red móvil se hace a través de la interfaz *usb to uart*. Antes de iniciar la configuración se requiere la instalación de varias librerías en el sistema operativo. Esto se realiza con el comando *"sudo apt-get install libqmi-utils udhcpc"*. La librería *libqmi* es una biblioteca basada en glib que permite conectarse con *módems* y dispositivos WWAN que emplean el protocolo *Qualcomm MSM Interface (QMI)*. Con este protocolo instalado se realiza la comunicación con el módulo 4G. Además se instalo la librería *udhcpc* usado para que el cliente udhcp negocie una concesión con el servidor DHCP y notifica a un conjunto de *scripts* cuando se obtiene o se pierde una concesión.

Por otro lado, la tarjeta SIM usada pertenece a la operadora Tuenti, la cuál debe tener activada datos móviles o tener disponible saldo. Al encender la Raspberry Pi la red móvil permanecerá apagada hasta que se presione el botón de encendido, sin embargo esto no se puede realizar ya que el módulo quedará dentro de la caja. En este caso se hace uso de la interfaz *usb to uart* para mediante el protocolo QMI enviar comandos para iniciar la red.

El módulo indica la conexión a la operadora mediante el parpadeo del *LED NET*, en caso de no estar conectado el botón se mantendrá apagado. Mediante *libqmi* se puede verificar la conexión a la

red y el nivel de potencia de la antena, con esto se puede constatar la recepción de señal. Una vez que se tenga el módulo conectado a la red, se activa al módulo como una interfaz de red en la Raspberry Pi. Para esto es necesario activar la interfaz wwan0. Las librerías instaladas anteriormente sirven para el manejo de esta interfaz de red. Es así que cuando el módulo esta conectado correctamente con el comando ïfconfig"se puede verificar la interfaz de red. El modulo SIM7600 actúa con un módem al cual se debe pedir una dirección IP mediante *udhcpc*. En el anexo A.1.1, se muestra el proceso completo para la conexión hacia Internet.

4.4. Diseño y construcción de la *PCB*

Una placa de circuito integrado o *PCB* es una superficie que soporta y conecta componentes a través de pistas o buses de material conductor laminados sobre una base no conductora. Existen varios simuladores de circuitos electrónicos disponibles de forma gratuita y de pago. Para este trabajo se usa la versión gratuita del software EAGLE. El diseño de la *PCB* tiene las siguientes características:

- Puede ser colocada sobre la *GPIO* del módulo SIM7600 de modo que queden las tres placas en paralelo (Raspberry Pi, SIM7600 y PCB).
- Tiene un espacio entre las pista, para agujerar la placa y sacar la cámara hacia el exterior.
- Tamaño semejante a la Raspberry Pi
- Ancho de pista mínimo de 1mm.

El desarrollo de una PCB conlleva dos partes, crear el esquema y diseñar la placa. En *EAGLE* el esquema se realiza en la ventana *schematic*, se selecciona los componentes que van sobre la placa y se realiza la conexión entre ellos. En la Figura 4.13, se puede observar el esquemático de la *PCB* a construir. Es necesario buscar las librerías de los componentes que no aparecen en el software. Uno de los principales componentes es el *header* del *GPIO*, ya que al conectarse directamente al Raspberry Pi las pistas saldrán desde sus pines hacia los sensores.

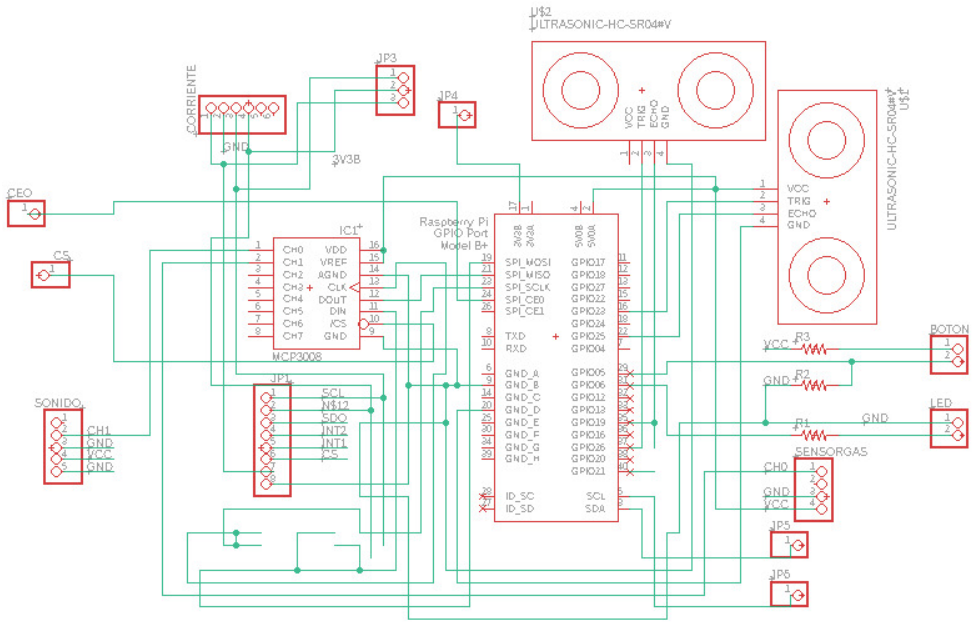


Figura 4.13: Esquema *PCB*

En la Figura 4.14, se muestra el diseño de la *PCB*. El espacio vació que se observa es la parte por donde la cámara saldrá. Alrededor de la *PCB* se ha colocado el *header* de la mayoría de sensores. En la Figura 4.14, se muestra el resultado final de placa construida. Se observa las pistas de 1mm y ya con los elementos soldados. Por otro lado, existen sensores como el acelerómetro y sensor de corriente que se encuentran integrados en la placa dado que no es necesario ubicarles en la estructura. Mientras que los demás elementos se encuentra cableados desde la placa hacia la estructura.

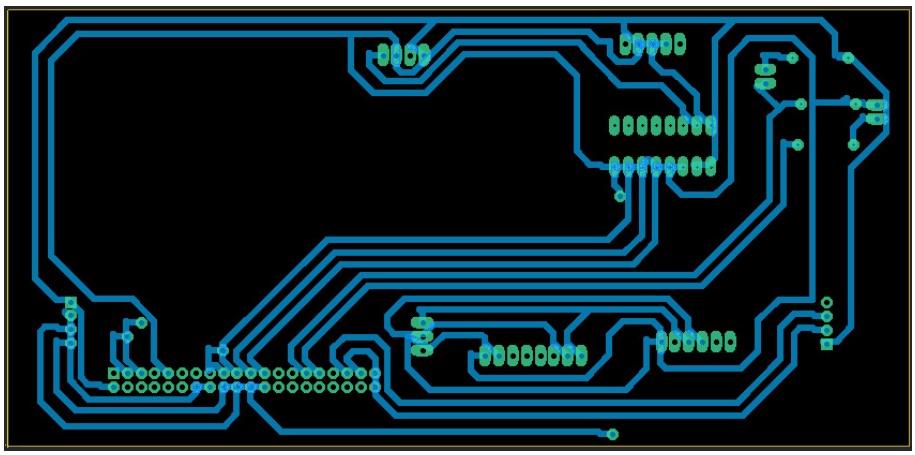


Figura 4.14: Diseño de la *PCB*

4.5. Acoplamiento de la Estación

El diseño de la estructura de acoplamiento de la estación se desarrollo para ser instalada sobre la estructura de la bicicleta sin afectar la movilidad del ciclista. El sistema contención de los diferentes elementos (sensores, controladores, fuente de alimentación) que integra el nodo, se desarrolló de dos formas distintas, el primer modelo consiste en una estructura metálica de acero inoxidable fabricada artesanalmente con materiales de reciclaje, mientras que, el segundo modelo consiste en un diseño impreso en 3D desarrollado con el *software Fusion 360*.

La distribución de los elementos dentro de la estructura se realizó de manera estratégica de tal forma que cada sensor pueda capturar datos reales del entorno en el que se encuentre la bicicleta. Entonces, el sensor de partículas SDS-011, el sensor de sonido y el sensor de niveles de calidad de aire se ubicaron apuntando hacia la base de la estructura, mientras que en las caras laterales se ubicaron los dos sensores de ultrasonido. Por otro lado, la cámara y el pulsante para almacenar los datos recolectados se ubican en la cara frontal, en la parte posterior de la estructura se ubicó el banco de baterías y el puerto USB que permite descargar los datos de vídeo. Finalmente, en la cara superior se ubica el LED indicador del estado de funcionamiento del nodo y el *switch* de encendido y apagado del mismo. Bajo esta distribución de elementos el diseño de la estructura tendrá una dimensión interior de $19cm$ x $9cm$ x $9cm$ (Largo x Ancho x Alto). Las dimensiones del largo ($19cm$) y el alto ($9cm$) están dictadas por las dimensiones de la batería, mientras que el ancho esta regido al modelo de pila en el cual se ubicaron los sensores y el respectivo controlador incluyendo el ancho del banco de baterías. Por otro lado la tapa tendrá dimensión interior de $19{,}6cm$ x $9{,}6cm$ x $1cm$. Lo que permite que la misma pueda ser colocada a presión, de tal forma que soporte las diferentes tipos de ciclovías existentes en ciudad (Ciclo veredas, Reservadas, Compartidas, Integradas y Segregadas).

4.5.1. Partes de la estructura

Vista Frontal.- En la Figura 4.15, se muestra la vista frontal de la estructura, donde, se puede distinguir las aperturas de la cara frontal y la cara lateral derecha, así como también los elementos físico internos de la estructura. Donde 1, es la apertura que permite asegurar la adquisición de datos de imagen mediante una cámara CSI propia de la fundación Raspberry Pi. El elemento 2, es una apertura que permite acoplar un pulsante, el cual cumple con la función de informar al controlador que se ha finalizado una ruta y enviar comandos para apagar el nodo vía software, del mismo modo 3, permitirá acoplar los sensores ultrasónicos usados para medir la distancia de separación entre un obstáculo lateral izquierdo y el centro manubrio de la bicicleta. Por otro lado, los elementos 4 y 5 son elementos físicos internos de la caja que permite acoplar la batería y el sensor de particular de tal forma que no exista movimiento interno de los elementos al recorrer cualquiera de las rutas.

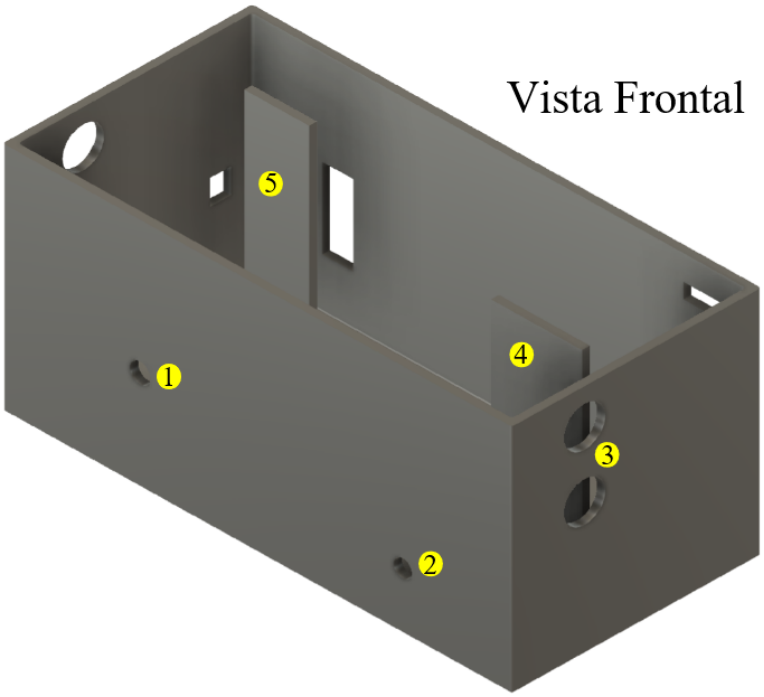


Figura 4.15: Vista frontal de la estructura

Vista Posterior,- Del mismo modo que en el caso anterior en la Figura 4.16, se presenta los elementos, aperturas y estructuras físicas principales de las caras posterior, lateral izquierda e inferior. Donde 6, es la apertura que permitirá integrar un puerto USB mediante el cual se podrá descargar toda la información visual capturada durante la ruta por la cámara. Por otro lado 7 corresponde a la parte física de la estructura que permitirá acoplar el nodo con el cuadro o cuerpo principal de la bicicleta. Con tal propósito se empleo dos platinas de hierro y dos bridas, permitiendo que el nodo se pueda acoplar fácilmente a cualquier bicicleta y por lo tanto sea independiente del modelo que disponga un usuario. La apertura 8 y 9 permitirá visualizar el estado de la batería del nodo y cargar la misma en caso de requerirlo.Las aperturas 10 a la 13, permiten a los sensores interactuar con la ruta ya que acoplan, al sensor ultrasónico usado para medir la distancia de separación entre un obstáculo lateral derecho y el manubrio de la bicicleta, el nivel de partículas por millón, el nivel de ruido y los valores de calidad de aire de la ruta recorrida respectivamente.

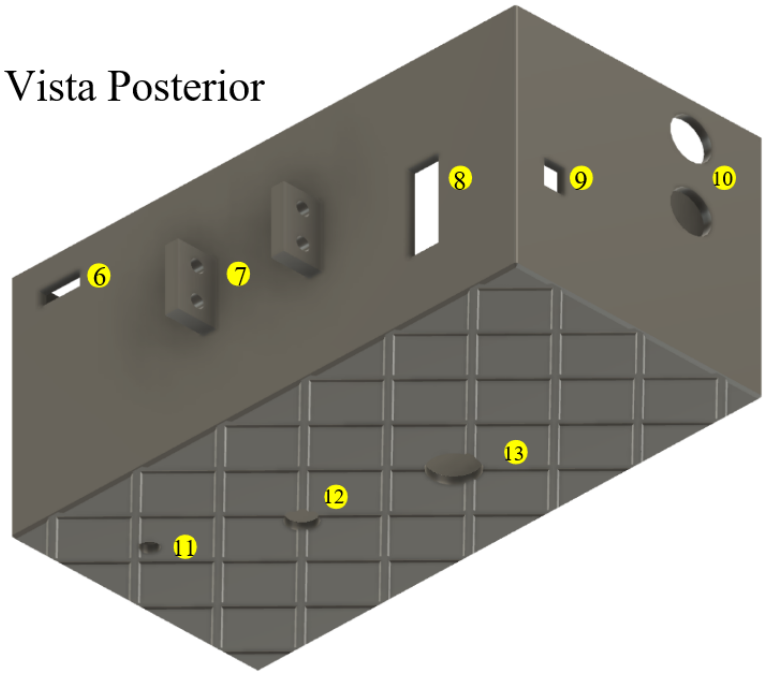


Figura 4.16: Vista posterior de la estructura

Tapa.- Como se pude apreciar en la Figura 4.17, dispone únicamente de dos aperturas, siendo 14, la apertura usada para integrar un LED que indica los diferentes estados del nodo mediante velocidades de parpadeo, mientras que, la apertura 15 se usa para acoplar un *switch* que permite controlar el encendido y apagado físico del nodo. Se debe mencionar que en el diseño 3D la cara superior e inferior se realizó en forma cuadricular para brindar mayor facilidad en la impresión y más estabilidad en la estructura.

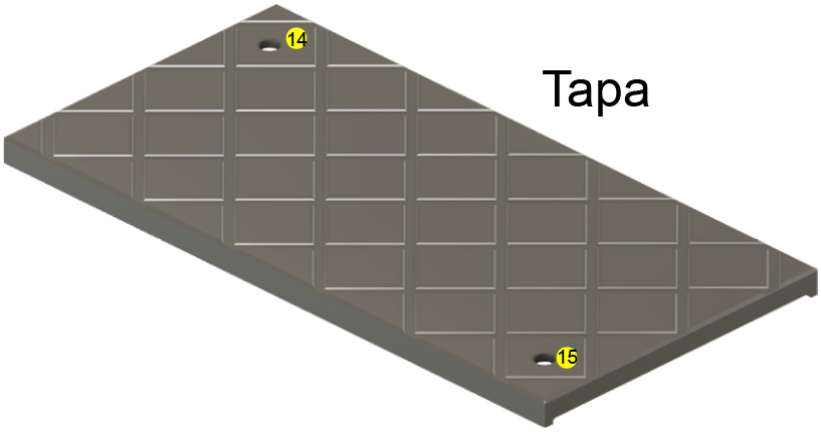


Figura 4.17: Tapa de la estructura

En el anexo C se muestra el proceso de integración de todos los elementos del nodo sobre la estructura de contención y el acoplamiento en dos modelos de bicicletas. Por otro lado, en el anexo G se muestra el costo de implementación del nodo tendiendo en consideración los elementos seleccionados la construcción de la estructura y elementos adicionales.

4.6. *Firebase* como servicio de *backend*

Firebase es una plataforma digital que permite simplificar el desarrollo de aplicaciones web o móviles, dado que integra un amplio campo de funcionalidades como autenticación, base de datos no relacionales con soporte en tiempo real, servicios de *hosting*, almacenamiento de archivos, entre otras características que convierten a la plataforma en una de las mejores aplicaciones de *backend*. El desarrollador de *frontend* integra las funcionalidades según las necesidades de la aplicación. Para el desarrollo del proyecto se implementará dos aplicaciones, donde la primera consiste en una aplicación móvil para dispositivos *Android*. Esta requerirá los servicios de autenticación y acceso a la base de datos *NoSQL*, mientras que la segunda sera una aplicación Web, que de igual manera deberá tener acceso a la base de datos *NoSQL* y además usará el servicios de *hosting* propio de *Firebase*. Esto reduce los tiempos de procesamiento de las solicitudes mejorando así el rendimiento de la aplicación. Otra de las capacidades de las que dispone este plataforma es la fácil administración del servicio donde además se ofrece estadísticas del uso de la plataforma como se muestra en la Figura 4.18.

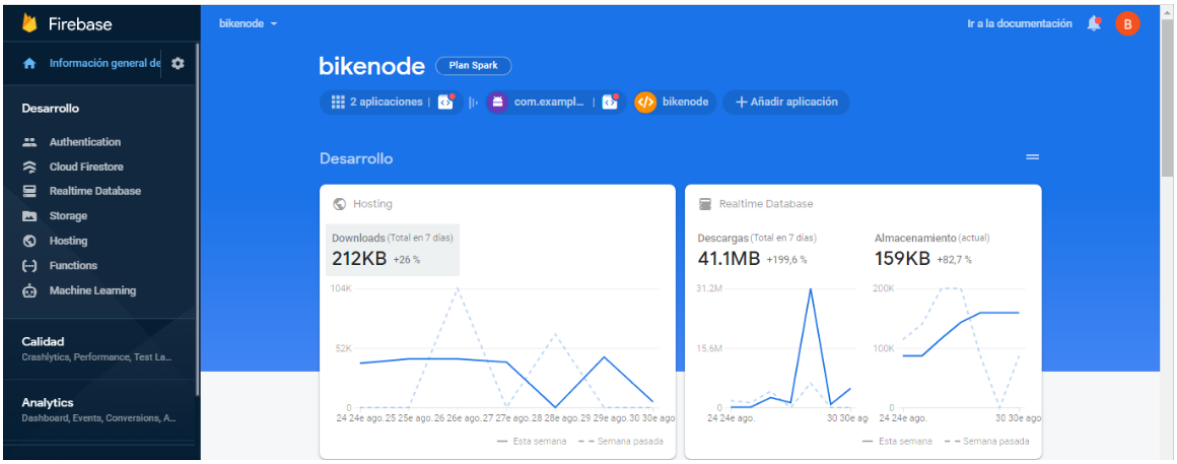


Figura 4.18: *Firebase* página de principal

4.6.1. *Authentication*

Este servicio se empleará en la aplicación móvil donde el usuario debe en primer lugar registrarse en la base de datos usando un correo y una contraseña, de este modo al acceder a las funcionalidades de la aplicación se puede restringir el servicio por usuario, es decir limitar el acceso de los usuarios únicamente a sus datos, brindando de esta manera seguridad al resto de usuarios.

4.6.2. *Hosting*

El servicio de *Hosting* ofrecido por *Firebase* simplifica el desarrollo de la aplicación web y optimiza los tiempos de acceso y de consulta de datos, al ser un servicio de *Google* empleará el mismos servidor para el almacenamiento, el cual, se realiza como si se tratase de una *Content Delivery Network* (*CDN*) del servidor principal. Además, la plataforma permite crear dominios personalizados y agregar certificados *Secure Sockets Layer* (*SSL*) gratuitos a los mismos.

4.6.3. *Realtime Database*

El soporte de la plataforma *firebase* a la administración de datos en tiempo real es uno de los factores claves por los cuales se eligió esta plataforma como servicio de *backend*, dado que uno de los objetivos del trabajo consiste en informar continuamente al usuario sobre los valores censados. Además, al utilizar base de datos no relacionales brinda la posibilidad de simplificar y optimizar el manejo de datos. La estructura desarrollada para este proyecto consiste en dos listas, que almacenan la información de todos los usuarios registrados, como se muestra en la Figura 4.19. En el caso de la lista "*DatosUsuarios*", se almacenarán los datos personales de cada usuario registrado usando la aplicación móvil. Mientras que, la lista "*DatosPosicion*", almacenaran las listas de rutas creadas por los usuarios donde cada item "*Ruta*" contendrá 17 atributos, correspondientes a los valores de cada uno de los sensores, incluyendo las horas en las que se capturaron los datos, la fecha de captura y la hora de inicio y fin de las rutas. Los datos almacenados podrán ser accedidos tanto desde la aplicación web y la aplicación móvil, mientras que, la incorporación de los datos correspondientes a nuevas rutas únicamente se podrá realizar desde el nodo, así como los datos de nuevos usuarios solo se podrá realizar desde la aplicación móvil.

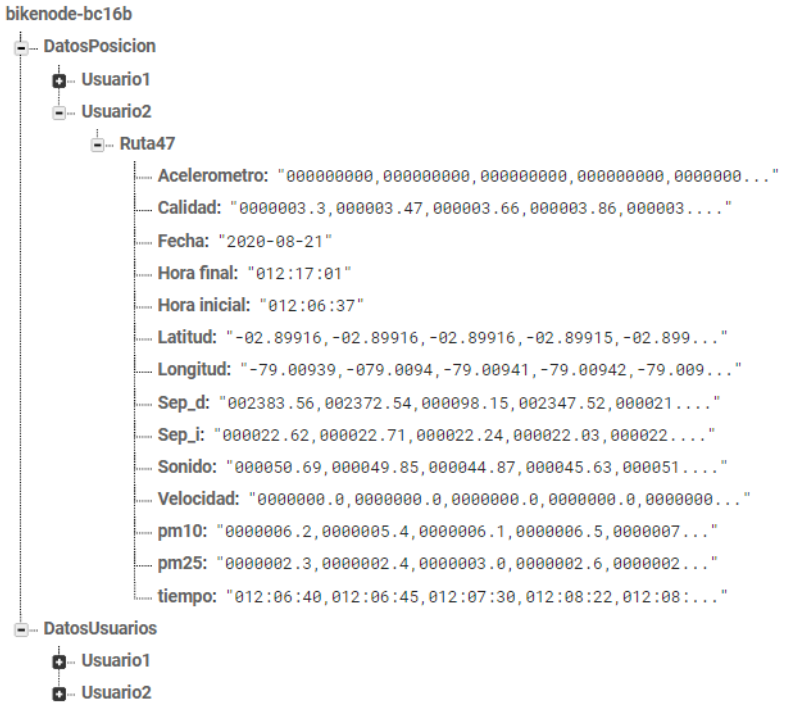


Figura 4.19: Estructura de la base de datos

Se debe mencionar que existen dos planes de uso de *Firebase* el primero consiste en el uso gratis del servicio conocido como plan *spark* mientras que el segundo es de pago y se conoce como plan *blaze*. Para el desarrollo del proyecto únicamente se requirió el uso del plan *spark* donde se ofrece servicio de autenticación de hasta 10000 usuarios por mes, servicios de *hosting* de hasta $10GB$ de almacenamiento y $10GB$ de transferencia de datos y servicios de almacenamiento en tiempo real que soporta hasta 100 conexiones simultaneas, $1GB$ de almacenamiento y $10GB$ de descarga mensual , así como también el servicio de análisis del consumo totalmente libre.

4.7. Software de Gestión y Monitorización

Una de las partes fundamentales del desarrollo del proyecto es el modo de interacción entre el cliente y el nodo, es decir la interfaz de usuario mediante el cual se podrá administrar el funcionamiento de la estación y visualizar los datos adquiridos. Para ello, se desarrollo dos soluciones principales, donde la primera consiste en el desarrollo de una aplicación nativa orientada a dispositivos móviles con sistema operativo *Android*, mientras que la segunda consiste en el desarrollo de una aplicación Web. Cada una de ellas dispone del servicios de captura de datos en tiempo real y la visualización de los datos históricos de las rutas. En el caso de la aplicación móvil, el histórico de rutas que se podrá visualizar serán únicamente las que el propio usuario haya generado, mientras que en el caso de la aplicación web se podrá visualizar las rutas históricas de todos los usuarios. Se debe mencionar que se eligió estas alternativas sobre el desarrollo de una aplicación híbrida, debido a las grandes ventajas que presenta cada una de ellas en sus respectivos campos, ventajas como, la portabilidad, el uso de gráficos, la capacidad de actualización entre otras.

4.7.1. Implementación y Funcionalidades de la Aplicación Móvil Nativa

La aplicación móvil se desarrollo para *smarthphones* con sistema operativo *Android* de versión desde KitKat (4.4) o superiores, esta conformada por con 5 ventanas e integra *API*s externas como *Firebase* para el registro del usuario, la autenticación y la consulta de datos, y *Google Maps* para mostrar al usuario su posición en tiempo real y las rutas que el mismo a generado durante sus viajes.

4.7.1.1. Ventana Inicio

Como su nombre lo indica esta ventana será la primera con la que un usuario se encontrará al abrir la aplicación. La ventana cuenta con dos botones y dos cuadros de entrada de texto como se muestra en la Figura 4.20. Si el usuario es nuevo en la aplicación primero deberá crear una cuenta para lo cual se debe seleccionar el botón crear cuenta, el mismo que iniciará el direccionamiento hacia la ventana de registro. Por otro lado, si el usuario ya esta registrado y desea iniciar una sesión debe ingresar el correo del usuario y la contraseña registrada al crear la cuenta. Completados estos campos, se debe seleccionar el botón Iniciar Sesión, el cual direcciona al usuario a la venta de selección.

4.7.1.2. Ventana de registro

La ventana de registro permitirá ingresar los datos personales del usuario como se muestra en la Figura 4.21, donde los más importantes son el correo y la contraseña, dado que los mismos permitirán realizar la autenticación en la ventana de inicio, el resto de datos son datos complementarios y se almacenan por seguridad del desarrollador. En caso de la pérdida de la cuenta de uno de los usuarios se pueda eliminar la misma de forma manual validando estos datos. Además, estos permiten que la aplicación identifique al usuario y la interacción con la aplicación sea más personalizada. Una vez ingresados todos los campos solicitados, al seleccionar el botón crear cuenta este direcciona al usuario hacia la página de inicio donde podrá validar el registro.

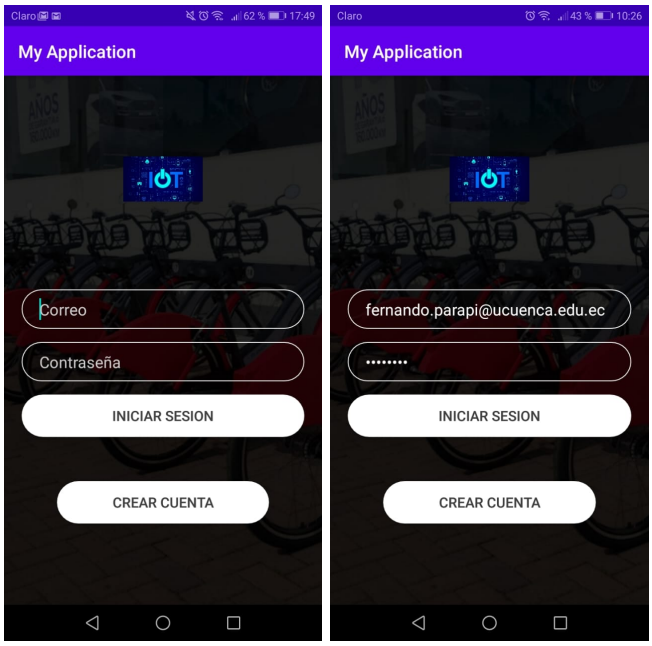


Figura 4.20: Ventana de inicio (Aplicación móvil)

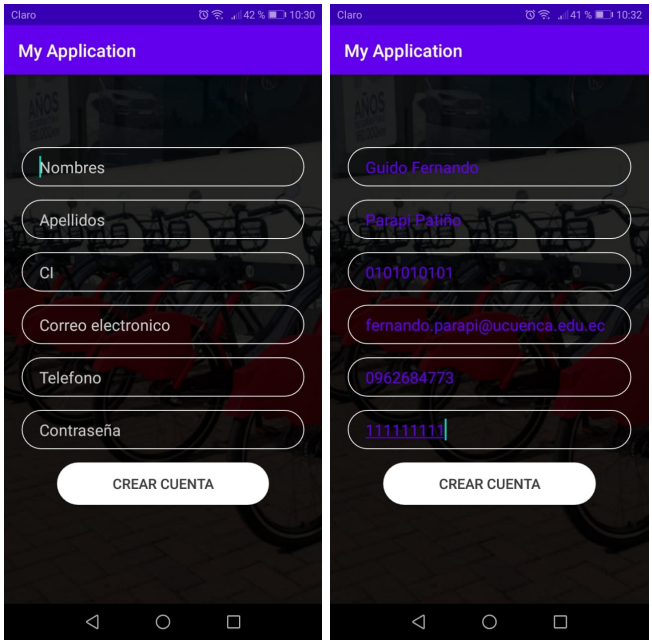


Figura 4.21: Ventana de registro para nuevos usuarios (Aplicación móvil)

4.7.1.3. Ventana de Selección

Luego de autenticar al usuario con el correo y la contraseña se mostrará un mensaje de bienvenida que motiva al usuario a generara nuevas rutas, al mismo tiempo que se direcciona al usuario a la ventana de selección, donde se podrá escoger entre la opción de analizar rutas o usuarios activos como se muestra en la Figura 4.22. Al seleccionar el primer botón se direccionará al usuario a la ventana de selección de rutas, mientras que el segundo botón direccionara al usuario a la ventana *Realtime*.

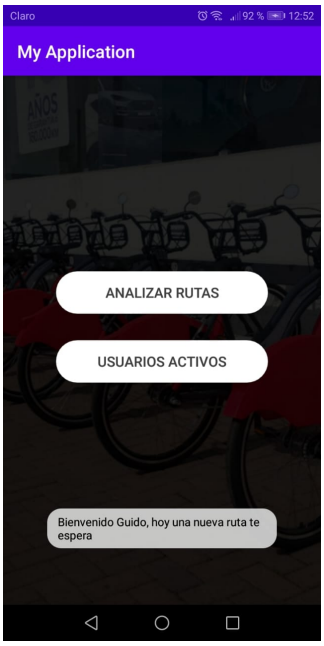


Figura 4.22: Ventana de selección de opciones en la aplicación móvil

4.7.1.4. Ventana *RealTime*

Al iniciar la ventana se mostrará el mapa de *Google Maps* centrado en la ciudad de Cuenca. También se mostrará los nodos que están en ese momento en uso. En la Figura 4.23, se observa que aparecen dos nodos activos en funcionamiento con nombre Usuario1 y Usuario2. Al presionar sobre cualquier nodo aparecerán los datos que se están capturando en tiempo real.

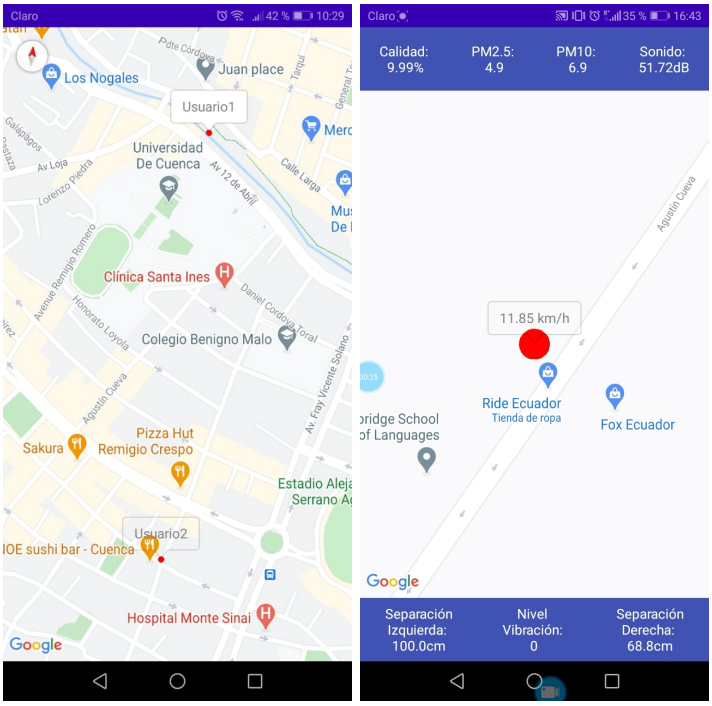


Figura 4.23: Ventana de monitorización en tiempo real (Aplicación Móvil)

La ventana en tiempo real se irá actualizando cada vez que el nodo envié un dato, y el mapa se irá moviendo automáticamente con el nodo seleccionado. En los casos en que no haya datos nuevos se mantendrán los últimos datos capturados. Una vez que el ciclista haya terminado con su recorrido el nodo correspondiente a ese usuario desaparecerá hasta que sea activado nuevamente. Por otro lado, si se requiere realizar seguimiento a otro nodo, se deberá presionar sobre el nodo actual y la ventana volverá a visualizar a todos los nodos en funcionamiento y se podrá elegir otro nodo.

4.7.1.5. Ventana de selección de rutas

Esta ventana se visualiza cuando el usuario escoge la opción analizar rutas en la opción de selección, se debe mencionar que en esta aplicación el usuario únicamente podrá revisar las rutas que el mismo ha generado, la ventana esta compuestas de cinco elementos, como se muestra en la Figura 4.24.

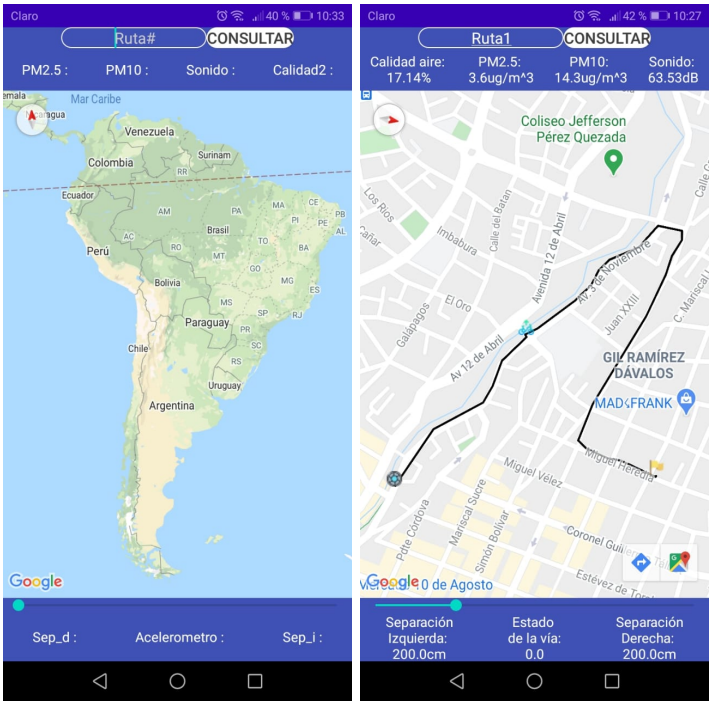


Figura 4.24: Ventana para el análisis de rutas almacenas (Aplicación móvil)

- Entrada de texto: Este elemento permite al usuario ingresar el número de ruta que se desea visualizar, para ello se requiere ingresar la palabra clave Ruta y el número correspondiente de misma.
- Botón: Con el botón consultar se realiza la *query* a la base de datos NoSQL de Firebase, la cual analiza la existencia de la ruta y devuelve los datos.
- Mapa: Este elemento por otro lado permitirá visualizar la ruta completa seleccionada por el usuario, donde la posición del nodo sobre la ruta se visualiza con el ícono de una bicicleta, para la construcción de la ruta se usa polilíneas cuyo salto dependerá del tiempo en el que se registraron los datos, a menor tiempo mayor presión.
- Slider: Este elemento permite manipular el desplazamiento del ícono de posición del usuario en la ruta.

- Datos: Por último, esta funcionalidad permite visualizar los valores capturados por el nodo del usuario durante el trayecto.

Finalmente, se debe mencionar que cada una de las ventanas cuenta con sus mensajes de error correspondientes, como por ejemplo en la ventana de inicio en caso de que los datos de autenticación no coincidan se envía un mensaje para que el usuario verifique el correo y la contraseña. En la ventana de registro se muestra estos mensajes cuando ya existe una cuenta registrada con el correo ingresado o cuando existe algún campo vació, así mismo, se visualiza un mensaje informativo que indica al usuario que se ha registrado correctamente. En el anexo E se muestra los programas desarrollados para permitir las funcionalidad de las diferentes ventanas de aplicación móvil y las correspondientes interfaces de usuario de cada una de ellas.

4.7.2. Implementación y Funcionalidades de la Aplicación Web

A diferencia de la aplicación móvil nativa la aplicación Web se desarrolló como una aplicación de administrador, es decir, una aplicación que permita analizar los datos generados por todos los usuarios que disponen de un nodo. El desarrollo de la aplicación Web se implementó, con especial atención en el *frontend*, dado que, como *backend* se emplea los servicios disponibles en *Firebase*. La comunicación entre el *backend* y *frontend* se realiza usando la arquitectura *REST* donde cada servicio tendrá su propia *Uniform Resource Locator* (*URL*). Se debe mencionar que la aplicación es de dominio público, es decir que no se requiere autenticación para interactuar con la aplicación, por lo cual el cliente únicamente tendrá acceso a la visualización de los datos, mas no podrá actualizar ni eliminar los mismos de la base de datos *NoSQL* de *Firebase*.

4.7.2.1. *Stack* para el desarrollo *frontend*

Para el desarrollo de la aplicación Web como frontend se uso HTML5, que implementa HTTP en el manejo de la estructura de la página, CSS en el diseño de los estilos de presentación que se usan en la pantalla del dispositivo final y JavaScript para agregar las funcionalidades de interacción. Todo ello en un entorno multi-dispositivo y con soporte a una lógica más compleja, además, permite la integración de *CDN*s, como las de *Boostrap* que agrega ventajas adicionales, como mejorar la latencia, reduce del uso de ancho de banda y tráfico cliente-servidor, logrando disminuir la carga en servidor de *backend*.

4.7.2.2. Estructura de la aplicación

La estructura de la aplicación esta dividida en dos páginas, donde la página principal permitirá visualizar los datos en tiempo real, miestras que, la página secundaria se usa para realizar revisiones de todas las rutas completadas por los diferentes usuarios. La estructura HTTP de las páginas esta divida en encabezado y cuerpo. El encabezado se integra el nombre de la página y los diferentes estilos *CSS* y las *CDN*s de estilos de *Boostrap*. Estos se emplean para la visualización en el dispositivo final incluyendo los estilos usados por *Google Maps*. Mientras que, en el *body* se crea la estructura del contenido de la página web y se integra las diferentes *API*s que interviene para brindar la interacción cliente-servidor, para el caso de este proyecto se usa *Firebase* como base de datos y *Google Maps* para presentar los mismos de forma amigable para el usuario. Para solicitar los mapas, consultar los datos y construir las rutas se usa *Java Script* bajo un entorno *Node.js*. En la página secundaria se integra

además la aplicación *morris.js*, mediante la cual se muestra los gráficos lineales y de área de los valores censados en las diferentes rutas.

4.7.2.3. Página Principal

Esta página es la primera con la que se encontrara un usuario al acceder a la aplicación Web, esta conformada por 3 partes principales: Encabezado, Mapa y Datos.

- **Encabezado:** Este elemento permite que el usuario pueda seleccionar entre la página principal con datos en tiempo real o la página secundaria.
- **Mapa:** Este componente constituye la mayor parte de la página, dado que el mismo se podrá visualizar la posición exacta de los usuario activos, el cual se representa con un ícono de una bicicleta verde, donde, al seleccionarla se muestra el número de usuario con el que el nodo esta registrado en la base de datos y la posición de latitud y longitud como se muestra en la Figura 4.25. Se debe mencionar que no se visualizará los datos personales de los usuarios dado que la aplicación web es de dominio publico.
- **Datos** Finalmente este apartado muestra en una Tabla los datos en tiempo real que se están registrando con el nodo, siendo estos el nivel de CO2, la cantidad de partículas por millón de 10 micras y 2.5 micras, el nivel de ruido al que se encuentra expuesto el ciclista, la distancia de separación entre el manubrio de la bicicleta hacia los laterales derecho e izquierdo, el estado de la vía según estimaciones del nivel de vibración y la velocidad con la que se esta movilizando.

En la Figura 4.25, se muestra la interfaz de usuario de la página principal donde se puede distinguir cada uno de los partes mencionadas anteriormente.

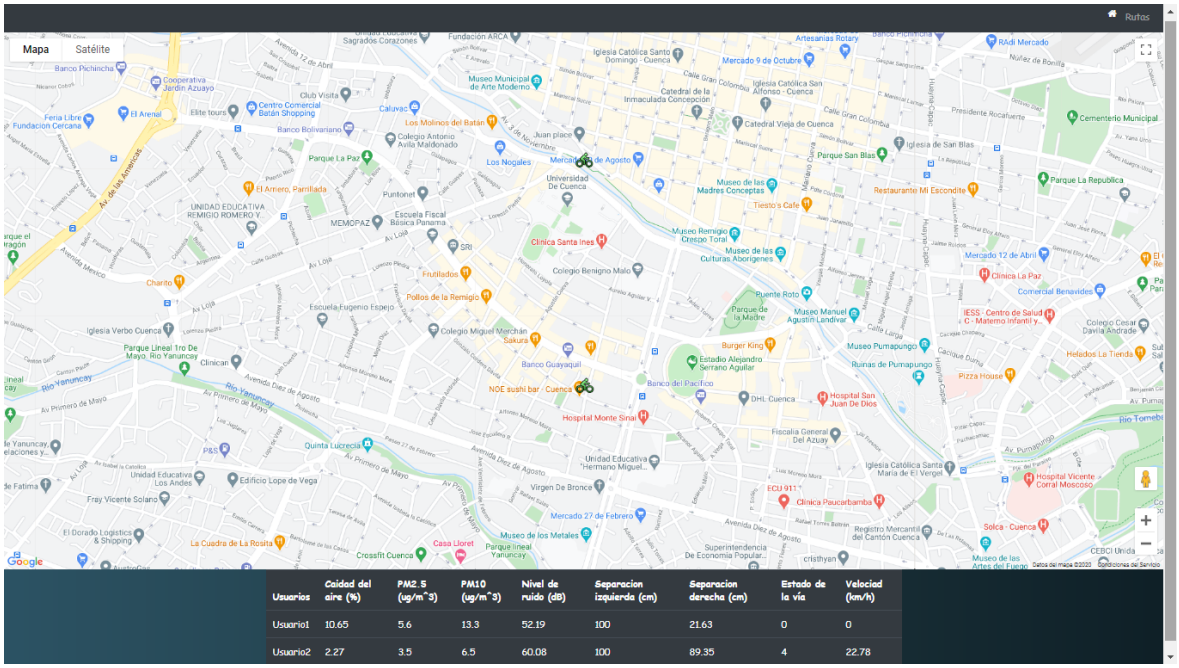


Figura 4.25: Página principal de la interfaz de usuario (Aplicación Web)

4.7.2.4. Página Secundaria

La página secundaria se puede seleccionar desde la página principal o se puede acceder desde su dirección *URL* propia, entregada por el servidor de *hosting*, esta página está desarrollada para brindar al cliente un acceso a todas las rutas almacenadas por los diferentes nodos registrados en la base de datos de *Firebase*, la página secundaria esta conformada de 7 elementos: Encabezado, Mensajes informativo, Selector, Mapa, *Slider*, Gráficas y un Mensaje de error.

- Encabezado: Del mismo modo que en el caso anterior este elemento permitirá que el usuario pueda desplazarse desde la página secundaria hacia la página principal
- Mensaje informativo: Este elemento permite informar al cliente cuantos usuario registrados existen en la base de datos, para ello al iniciar la página se realiza una consulta al servidor de *firebase* usando *JavaScript* y se visualiza en la pantalla mediante HTML.
- Selector: Como su nombre lo indica este elemento permite seleccionar el usuario y la ruta que se desea analizar, se debe mencionar que los datos de entrada son numéricos como se muestra en la Figura 4.26.
- Mapa: De igual manera que en el caso anterior en esta página también se integra el mapa de *Google Maps*, el mismo que permite trazar la ruta generada por el usuario mediante el uso de polilíneas y visualizar el punto de partida y punto de finalización de la ruta. El punto de partida se mostrará con mismo ícono usado para mostrar a los usuarios activos. Mientras que para señalar el punto de finalización se usa el ícono similar a la sombra de una bicicleta que se muestra en la Figura 4.26. Se debe mencionar que, al seleccionar el ícono de partida se puede visualizar los valores censados por los diferentes elementos en el punto de interés.
- *Slider*: La opción tipo dial, permitirá desplazar el ícono de inicio de ruta por toda la trayectoria generada siguiendo los puntos donde se han realizado las lecturas, además este elemento muestra la hora exacta en la que se capturó cada uno de los datos.

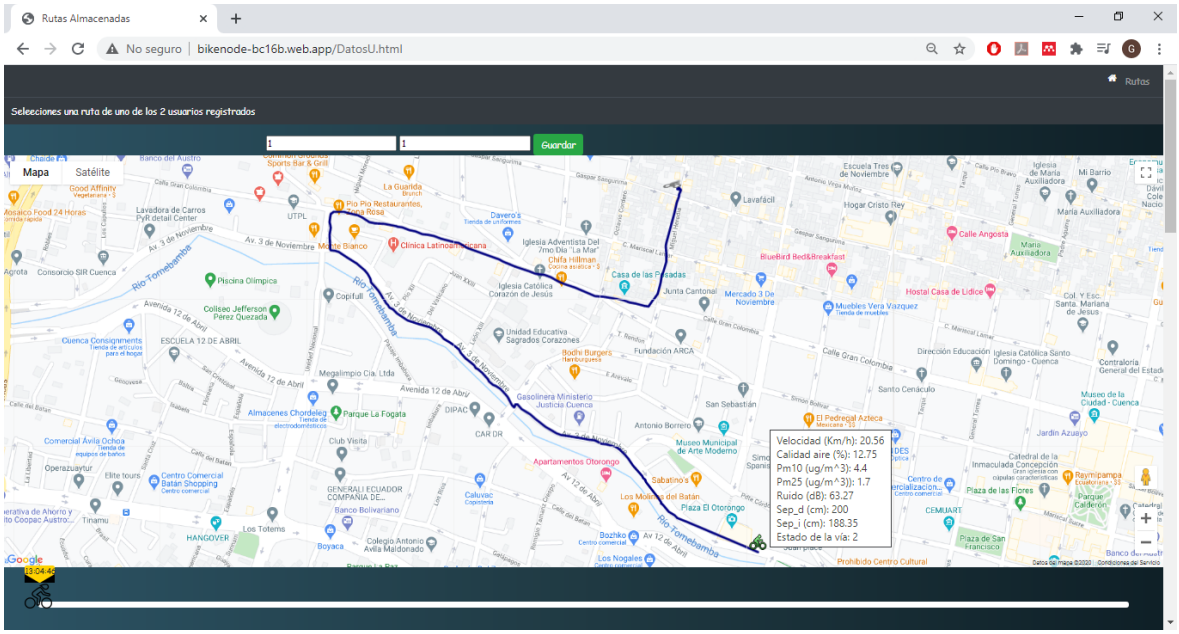


Figura 4.26: Página secundaria de la aplicación web (tracking)

- Gráficas: Este apartado permite mostrar el comportamiento de las variables de interés durante toda la trayectoria, además, se puede seleccionar valores específicos según la hora en la que se obtuvieron los valores, logrando así enlazar los datos con la trayectoria realizada. En la Figura 4.27, se muestra algunas de las gráficas obtenidas por el usuario 1 en su primera ruta.

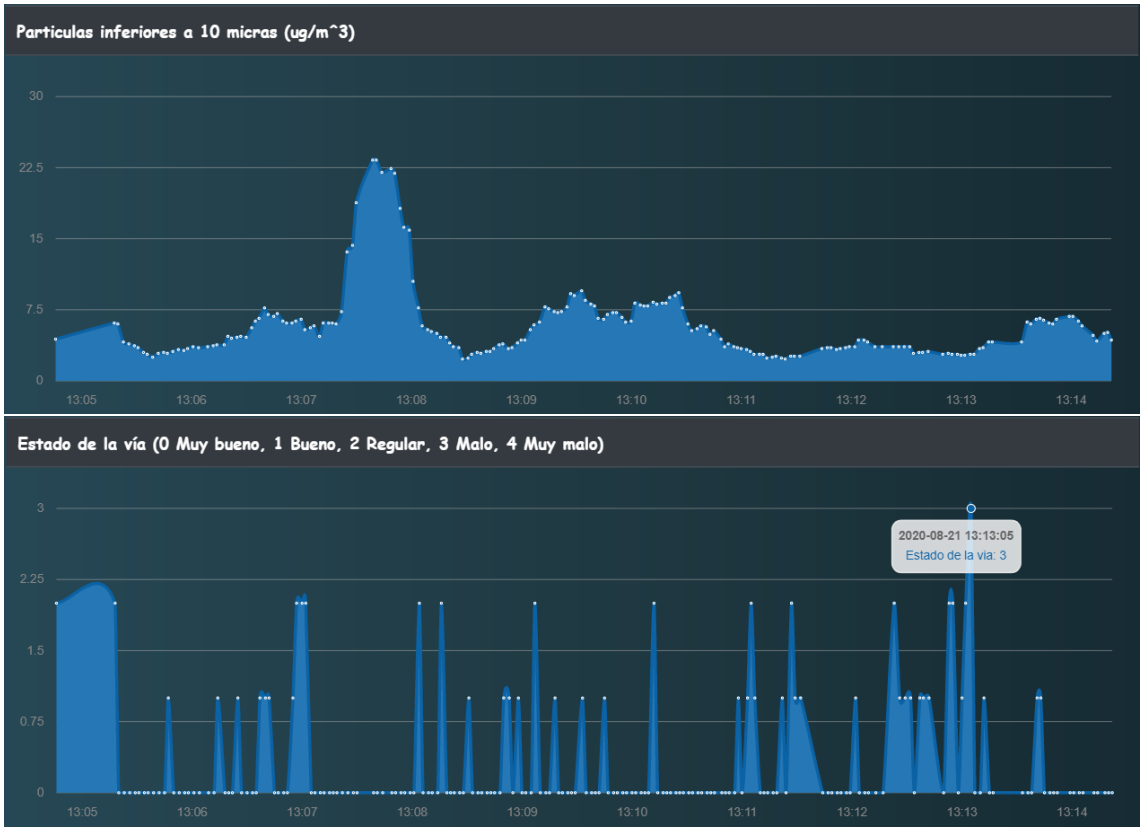


Figura 4.27: Página secundaria de la aplicación web (Gráficas)

- Mensaje de error: Este mensaje surge cuando el cliente de la aplicación web a seleccionado un usuario que no existe o una ruta no almacenada como se muestra en la Figura 4.28.

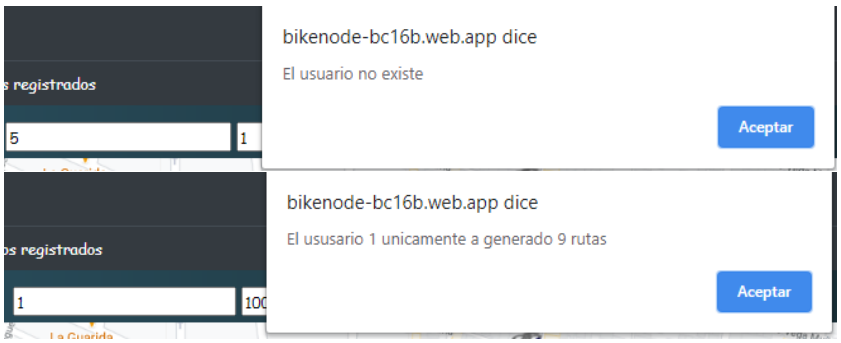


Figura 4.28: página secundaria de la aplicación web (Mensajes de error)

4.8. Funcionamiento del sistema de monitorización

Una vez realizada la descripción de los diferentes elementos que componen el nodo así como las aplicaciones y la plataforma digital que complementan el sistema de monitorización. En la Figura 4.29, se muestra un diagrama general de la arquitectura implementada. En primer lugar se encuentra la bicicleta y en su parte frontal esta ubicado el nodo. Este irá capturando las diferentes variables presentes en el ambiente, la posición actual, para posteriormente transmitir los datos obtenidos hacia la plataforma digital *Firebase*. La transmisión de datos se realiza a través del módulo 4G conectado a la red celular de la operadora Tuenti. También hay que recalcar que la captura de datos se esta tomando cada dos segundos.

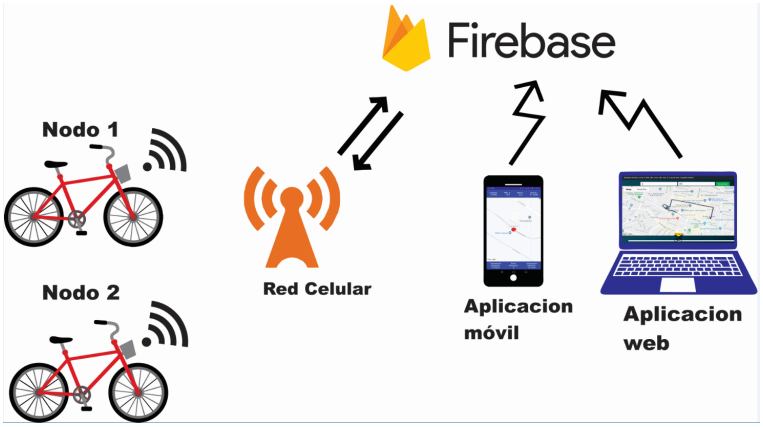


Figura 4.29: Sistema de Monitorización General

Los datos enviados son almacenados en la herramienta *Realtime Database* de *Firebase* y estarán actualizándose conforme el nodo envíe los datos. Al mismo tiempo desde la aplicación móvil es posible acceder a esos datos y presentarlos. No esta demás hacer notar que en la aplicación también se irán actualizando los datos produciéndose así un seguimiento de datos en tiempo real. Así mismo, desde la aplicación web alojada en el *Hosting* de *Firebase* también es posible visualizar los datos tomados al instante.

Hasta ahora se ha descrito las funciones cumplidas en tiempo real, si bien es posible ir visualizando y analizando la información obtenida al momento, es necesario almacenar todos esos datos recolectados durante el trayecto para posteriormente realizar un análisis más profundo de la información. En este contexto, en la misma base de datos se guardará la información de toda la ruta. La aplicación web cumple la función de mostrar toda la información recolectada de una ruta de una forma clara de modo que el ciclista o persona puede realizar un análisis más profundo de datos.

4.8.1. Desarrollo de la monitorización

En esta parte se muestra el proceso que realiza el nodo en la captura de datos. El nodo deberá cumplir algunas condiciones para asegurar una operación correcta. A continuación, se muestra las funciones pre-captura realizadas por el nodo.

1. Realizar la conexión hacia la red móvil.
2. Verificar que los sensores estén en funcionamiento.

3. Asegurar que el *GPS* este entregando datos de posición.
4. Iniciar la captura con la cámara.
5. Identificar el número de ruta actual.

Entonces al presionar el *switch* de encendido, al inicio el nodo realizará las funciones descritas. Hay que decir, que mientras está en proceso el LED que indica los diferentes estados del nodo se mantendrá prendido. Este proceso puede durar hasta dos minutos, mayoritariamente esto depende de la ubicación del *GPS*, como se mencionó anteriormente, éste debe estar en una zona abierta.

La siguiente parte es el inicio de la captura, se obtiene los datos de cada sensor y *GPS*; y son enviados a *Firebase*. Para visualizar los datos en tiempo real se requiere datos enviados al instante, por lo tanto, a falta de conexión de red se esperará por dos segundos, sino se logró enviar, este dato sera descartado solo para la visualización en tiempo real.

Mientras el nodo esta recolectando información, también estará guardando cada dato que tome, para al final enviar toda la información a la base de datos. Hay que indicar que en este conjunto también irán los datos que no se lograron enviar en tiempo real. Además, mientras se esta capturando los datos el LED se prenderá y apagará cuando se tome un dato, dando a conocer al ciclista que el nodo esta funcionando correctamente.

La duración del recorrido depende del ciclista. Sin embargo, el nodo tiene un límite de uso que es de una hora. Esto se realiza con el fin de cuidar al nodo y para casos que el ciclista olvide apagar la estación. El apagado del nodo se realiza mediante el botón ubicado en la estructura. Una vez que se presione el botón, el LED permanecerá encendido unos cuantos segundos y posteriormente se apagará denotando así una correcta funcionalidad del nodo.

4.8.2. Monitorización con/sin acceso a Internet

La conexión a Internet es un de los aspectos importantes del sistema, por esta razón se hace un énfasis en esta parte. A continuación se describe los casos que se pueden dar durante el recorrido.

- Conexión a Internet continua: Este es el mejor de los casos para cumplir con toda las funcionalidades del sistema.
- Conexión intermitente a Internet: En este caso la señal a la red celular podría presentar variaciones de potencia, ocasionando así una retraso en el envío de datos y simplemente que los datos se pierdan, pero como se menciono anteriormente solo se perderán datos en tiempo en real.
- Sin conexión a Internet: Este caso no es el peor de todos porque el nodo realizará todas las funciones excepto enviar datos en tiempo real. Es decir, cuando se inicie el nodo realizará las mismas funciones de captura. Por otra parte cuando acabe una ruta, todos los datos serán guardados, y serán enviados a la base de datos en una próxima ruta que este conectada a Internet, caso contrario se seguirán guardando.

4.8.3. Captura de vídeo

El registro de vídeo inicia al mismo tiempo que la captura. Por motivos de capacidad de almacenamiento del nodo, el vídeo es codificado para evitar altos tamaños de archivos, esto no quiere decir que el vídeo grabado será de mala calidad. La codificación utilizada es la H.264, además se optó por emplear como parámetro de codificación la variable QP (*Quantization Parameter*). Dicho valor oscila

en un rango de 1 a 100 (donde 100 representa la máxima calidad). A partir de pruebas realizadas se trabajó con un valor QP igual a 40. Todo este proceso se realiza con la librería *picamera*. Más adelante se mostrará algunas capturas de vídeo para constatar su calidad.

Los vídeos grabados no puede ser almacenados en la plataforma porque supondría un alto costo de datos móviles. Para solucionar este problema se ha dispuesto una entrada USB en la estructura. Al colocar una flash memory en el nodo, se transferirá todos los vídeos grabados. Esto se puede realizar mientras el led este encendido. Finalmente, los vídeos que ya han sido transferidos serán eliminados para evitar falta de almacenamiento.

En el anexo D se muestra los diferentes programas de control que integra el desarrollo de cada función descrita anteriormente.

4.9. Conclusiones

La elección de los elementos electrónicos adecuados para la implementación del nodo es una de las parte más importantes del desarrollo del proyecto ya que los mismos serán los responsables de capturar y procesar los datos en tiempo real. Dentro de estos elementos podemos distinguir entre los sensores y el controlador, donde, las características esenciales que se analizaron para la elección de los diferentes sensores fueron, el consumo energético, las dimensiones, el error relativo establecido por los fabricantes, el modo de comunicación, la velocidad de respuesta y el costo, mientras que, para la elección del controlador o *SBC*, se analizó el tipo de comunicaciones que soporta, de tal forma que se pueda integrar todos los sensores, el número de puertos de conexión, el consumo energético, las dimensiones y la velocidad de procesamiento.

El enlace de los sensores con la placa central Raspberry Pi supone el uso de varios protocolos de comunicación, la unidad de proceso ha respondido satisfactoriamente al manejo de múltiples dispositivos trabajando al mismo tiempo. Por otra parte, la conexión a Internet establecida a través del módulo 4G, permite una conexión estable, considerando que la antena se encuentra dentro de estructura. Mientras que para el posicionamiento la antena GPS debe ser ubicada en la parte externa de la estructura, sin embargo, no muestra problemas para su colocación en la bicicleta ya que cuenta con un imán en la parte posterior.

El correcto funcionamiento del sistema recae principalmente en la conexión a Internet y el posicionamiento del *GPS*. Para evitar posibles problemas que puedan causar estos dos elementos del sistema, se ha tomado las debidas precauciones. En caso de una alta intermitencia a Internet los datos capturados están siendo guardados localmente y al final de la ruta serán enviados a la base de datos siempre y cuando haya conexión a Internet, caso contrario, se mantendrán almacenadas en el dispositivo. En cuanto a errores del *GPS* que se puedan generar, el nodo esta constantemente verificando el estado del mismo. Entonces, se ha elegido un módulo capaz de realizar ambas operaciones y sobretodo realizarlas al mismo tiempo. El módulo SIM7600 ha demostrado ser el idóneo, por su conexión sencilla con la Raspberry Pi. Solo basta dos comandos AT para obtener datos del GPS y para el manejo de los datos móviles del módulo solo hace falta instalar una librería en Raspbian, para que por medio del protocolo Qualcomm MSM Interface (QMI) se realice la interacción. Además, el SIM7600 también puede obtener datos de velocidad logrando así capturar otro parámetro de interés para la estación.

La construcción de la placa, tiene un rol importante en lo que se refiere a la integración de los elementos la caja. Por está razón se decidió que la placa use los pines *GPIO* para conectarse

directamente a la Raspberry Pi, para así evitar que los sensores se conecten con cables a la *GPIO*. La implementación de placa también se baso en el echo de que la cámara pueda atravesar la *PCB* y además como va sobre los módulos fue necesario ubicar las salidas que van a los sensores en zonas libres, es decir se buscó, diseñar la placa de tal manera que las pistas y sensores no hagan contacto con los circuitos de los módulos.

El modo de presentación de los datos a los usuarios finales, es otro de los puntos claves en el desarrollo del proyecto dado que mientras más amigable sea este, mayor aceptación tendrá. En este contexto, hoy en día los modos de presentación más aceptados por los usuarios son mediante el uso de aplicaciones, ya sean estas nativas en el propio teléfono, aplicaciones web o aplicaciones híbridas. Las aplicaciones web y aplicación nativas son las que mejor se adaptan para el desarrollo de este proyecto dado que permite ventajas como multiplataforma en el primer caso y el uso de todas las capacidades del dispositivo móvil en el segundo caso.

El desarrollo de aplicaciones en la actualidad se divide en desarrollo *frontend* y *backend*, donde el primero consiste en la parte de la aplicación que interactúa con los clientes mientras que el segundo, es la parte que se encarga de la administración de los servidores y las bases de datos, para muchos desarrolladores el uso de la plataforma digital *firebase* como aplicación de *backend* ha permitido enfocar todo el esfuerzo en el desarrollo de la parte de *frontend* o interacción con el usuario.

Capítulo 5

Evaluación Experimental y Resultados

5.1. Introducción

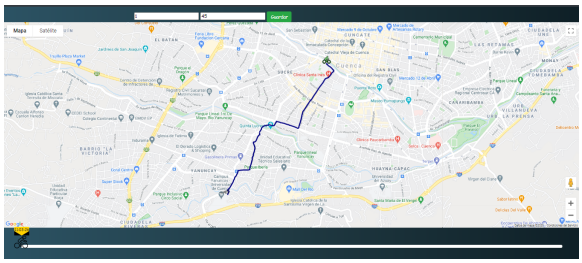
Constantemente los ciclistas se ven expuestos a eventos que reducen su confort. Estos se presentan independientemente del tipo de ciclovía, horario y el tipo de bicicleta que se este utilizando. Es por ello, que en este capítulo se expondrá a los nodos a diferentes escenarios de prueba, donde se evalúa el desempeño de los mismos en la captura de las variables censadas y la comunicación con el servidor central donde se almacenaran los datos. Para ello se analiza los parámetros de la red que afectan la trasmisión de datos y la respuesta general de consumo de corriente, consumo del procesador y aumento de temperatura en el mismo. Así también, se evaluará el rendimiento de cada una de aplicaciones frente a los escenarios expuestos, mediante las cuales se podrá determinar si los datos capturados por los sensores coincide con los principales eventos que afectan el confort. Con tal objetivo, se utilizan los valores capturados en tiempo real y la revisión de las rutas almacenadas. Se compara los valores capturados en cada zona mediante el seguimiento de la ruta y las gráficas del comportamiento de las variables en el tiempo con el registro de vídeo en cada una de las rutas. Se debe mencionar que el análisis se realizó desde el 19 Agosto hasta el 5 Septiembre del 2020, periodo de tiempo que coincide con algunas restricciones de movilidad por emergencia sanitaria. En este contexto, los datos obtenidos están influenciados por el cambio de hábitos de movilidad experimentados durante este periodo, como la disminución de tráfico vehicular y el aumento del ciclismo.

5.2. Escenario de pruebas

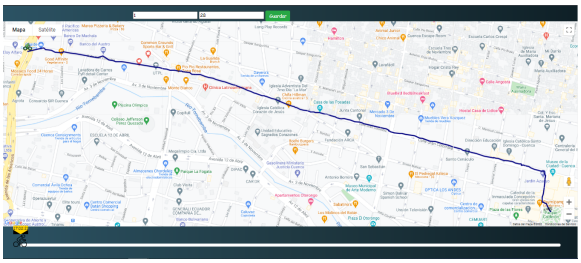
Para evaluar el desempeño de los nodos se utilizó cuatro rutas estratégicas, las mismas que permiten registrar datos de las diferentes ciclovías disponibles en nuestra ciudad mostradas en el anexo A. En particular, rutas transitadas por estudiantes, rutas de paseo, rutas céntricas y rutas de alto tráfico vehicular. Donde cada una de ellas cuenta con sus características propias que enriquecen la información capturada por los nodos.

5.2.1. Rutas de Estudiantes

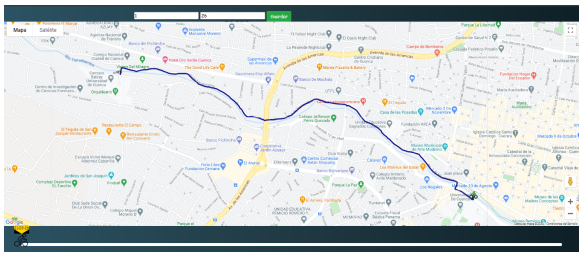
Esta ruta simula la que normalmente toman los estudiantes para desplazarse desde un campus a otro, la sección de la ruta se realiza en función de la trayectoria mas corta. En tal contexto, para verificar la operación de los nodos se seleccionó la ruta desde el Campus Central de la Universidad de Cuenca hacia cada uno de los otros campus de la misma (Campus Yanuncay, Campus Centro Histórico, Eco-Campus y Campus Paraíso). En la Figura 5.1, se muestran las trayectorias empleadas por los ciclistas para interconectar los extremos de las rutas. Se debe mencionar cada una de las rutas esta conformadas por diferentes tipos de ciclovías.



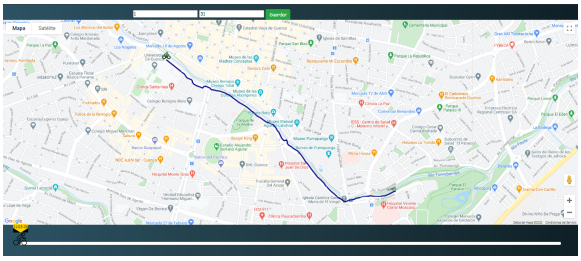
(a) Campus Central - Campus Yanuncay



(b) Campus Central - Campus Centro Histórico



(c) Campus Central - Eco-Campus



(d) Campus Central - Campus Paraíso

Figura 5.1: Rutas de estudiantes (Entre campus de la Universidad de Cuenca)

5.2.2. Ruta de Paseo

Debido a la gran diversidad de rutas de paseo que se puede ejecutar con las ciclovías existentes, se decidió utilizar una única ruta para el análisis. La misma que integrará todos los diferentes tipos de ciclovías con las que un ciclista se puede enfrentar, por ejemplo, ciclo veredas, ciclovías reservadas, compartidas, integradas y segregadas. En la Figura 5.2, se muestra la trayectoria de la ruta de paseo desarrollada por un ciclista.

5.2.3. Ruta Céntrica

La planificación estructural del centro de la ciudad al ser patrimonio cultural, limita la captura de datos a ciclovías compartidas y en el mejor de los casos a ciclovías integradas. Esta zona no cuenta con ningún tipo de estructura vial orientado a ciclistas. Bajo tales condiciones, la ruta escogida seguirá la trayectoria que se muestra en la Figura 5.3. Esta conecta una de las principales avenidas de la ciudad con el centro de la misma. Se debe mencionar que la mayor parte de la trayectoria se realiza utilizando la ruta tranviaria de la ciudad, dado que, al realizar diferentes pruebas se determinó que estas son las que mejores resultados presentan en cuento al confort del ciclista se refiere.

Figura 5.2: Ruta de paseo

Figura 5.3: Ruta hacia el centro de la ciudad

5.2.4. Ruta de alto tráfico vehicular

Las rutas de alto tráfico vehicular son las menos utilizadas por los ciclistas debido a los peligros que representan. Sin embargo, analizarlas representa un alto impacto en el desarrollo del proyecto dado que permite establecer el comportamiento de las variables ante una situación adversa al confort del ciclista. La elección de la ruta consiste en el uso de tres principales avenidas de la parte urbana de la ciudad como se muestra en la Figura 5.4.

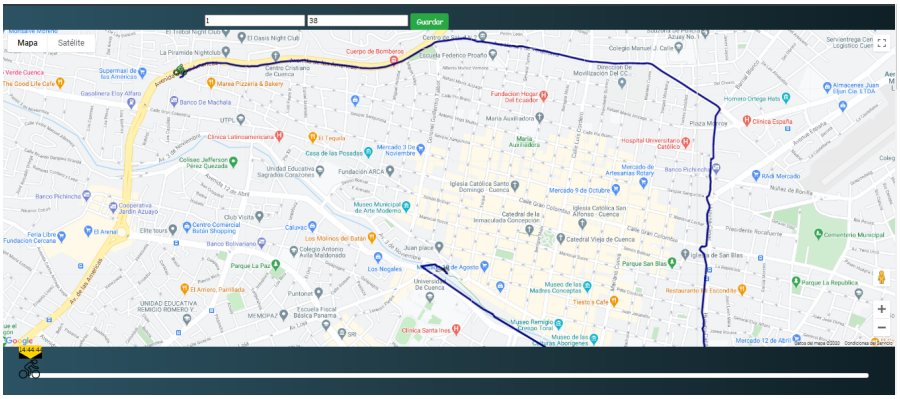


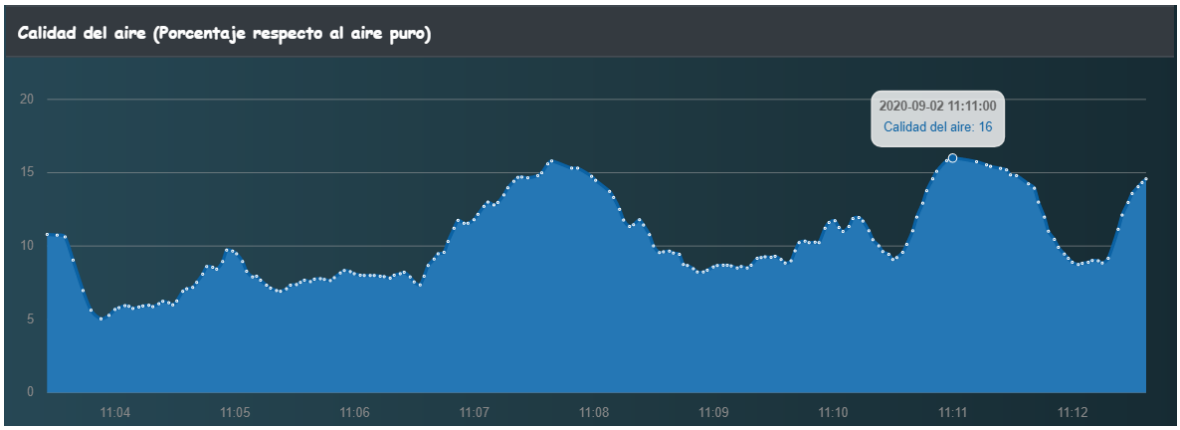
Figura 5.4: Ruta de alto tráfico vehicular

5.3. Evaluación del Software

5.3.1. Captura y Análisis de Variables

En esta sección se analiza el rendimiento de cada una de las aplicaciones frente a los principales eventos que encontramos en las diferentes rutas planteadas en el punto anterior. Además, como se mencionó en capítulos anteriores, las variables capturadas corresponden a cuatro parámetros que afectan directamente el confort del ciclista. En estas podemos encontrar la contaminación ambiental, para lo que analizamos la calidad del aire y el volumen de *MP* de la ruta, la contaminación sonora en decibelios, la capacidad de maniobra y el estado de la vía:

- **Calidad del aire:** Para medir este parámetro se usa el sensor MQ-135, se captura el porcentaje de gases contaminantes respecto al aire puro. Se debe mencionar que las emisiones de gases en un vehículo esta compuesto principalmente por CO_2, por lo cual, al analizar este valor se detecta el aumento de tráfico vehicular como se muestra en la Figura 5.5(a). El desempeño del sensor MQ-135 se analiza en la ruta de estudiantes Campus Central - Campus Paraíso de la Universidad de Cuenca. En la gráfica se observa que durante la mayor parte de la trayectoria el ciclista esta expuesto a valores de concentración de gases de entre el 7 % y 9 % mayor respecto al volumen en aire puro. El fotograma 1 de la Figura 5.5(b), muestra que estos parámetros por lo general se registran cuando el ciclista se encuentra alejado de la vía principal. Mientras que en zonas de alto tráfico, como los redondeles donde el ciclista esta dentro de la vía principal mostrados en los fotogramas 2 y 3, los valores se incrementa hasta en un 15 % y 16 %. Cabe recalcar que en rutas de alto tráfico se puede encontrar valores promedio constantes de entre el 15 % y 20 % de incremento en el volumen de gases presentes en la zona, lo que se traduce como un aumento de tráfico vehicular.
- **Sensor de partículas:** Para analizar el desempeño de este sensor se empleó la ruta de alto tráfico en la que se observa un considerable incremento de material particulado tras el desarrollo de un evento. Como se puede observar en el registro de vídeo en los fotogramas 1 y 2 de la Figura 5.6, el ciclista alcanza y debe conducir detrás de un bus interprovincial por un lapso de tiempo donde se presenta el incremento en las mediciones de *MP*, logrando alcanzar valores de $86\mu g/m^3$. Mientras en el fotograma 3 se muestra que el ciclista cambia de ruta e inmediatamente las lecturas comienzan a bajar a valores normales con un valor promedio de $15\mu g/m^3$. Por otro lado, en el caso de material particulado inferior a 2,5 micras en la misma zona registra incrementos desde un valor promedio de $6\mu g/m^3$ hasta los $25\mu g/m^3$. Los valores registrados por el sensor muestra que los parámetros de emisión no superan los limites de exposición establecidos en la normativa ecuatoriana de calidad del aire. La cual establece en $100\mu g/m^3$ el valor promedio de exposición tolerable para partículas de 10 micras y en $50\mu g/m^3$ el valor tolerable para partículas de 2,5 micras. Sin embargo, sí exceden los valores establecidos en las directrices de la Organización Mundial de la Salud (*OMS*) que recomiendan la explosión a concentración de $50\mu g/m^3$ y $25\mu g/m^3$ para partículas inferiores a 10 micras y a 2,5 micras, respectivamente. Además la *OMS* advierte que la explosión a material particulado conlleva a efectos sanitarios incluso en muy bajas concentraciones, de hecho, no se ha podido identificar ningún umbral por debajo del cual no se hayan observado daños para la salud.



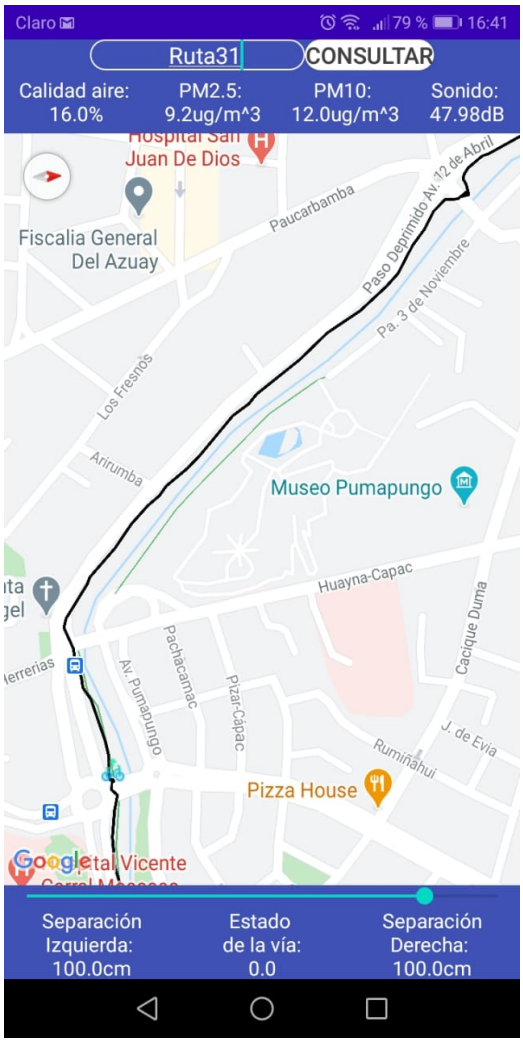
(a) Gráfica de la respuesta en la aplicación web

Fotograma-1

Fotograma-2

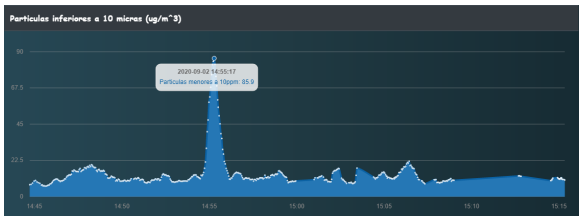
Fotograma-3

(b) Registro de vídeo de la ruta

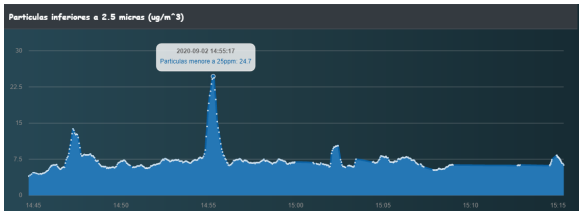


(c) Análisis de los valores usando la aplicación móvil

Figura 5.5: Análisis de los resultados del sensor de calidad del aire (Ruta Campus Central-Campus Paraíso de la Universidad de Cuenca)



(a) Gráfica de la respuesta en la aplicación web (10MP)



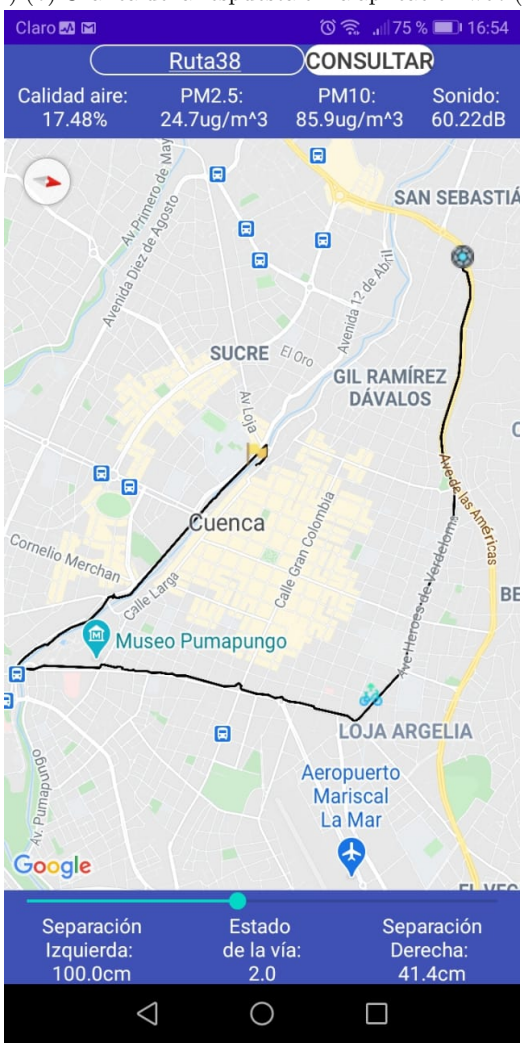
(b) Gráfica de la respuesta en la aplicación web (2.5MP)

Fotograma-1

Fotograma-2

Fotograma-3

(c) Registro de vídeo de la ruta



(d) Análisis de los valores usando la aplicación móvil

Figura 5.6: Análisis de los resultados del sensor de partículas (Ruta de alto tráfico vehicular)

- Niveles de Ruido: Para capturar este parámetro se usó el sensor MAX9814, el cuál permite registrar rangos desde $30dB$ a $83dB$. Este fue el sensor que mayor dificultad presentó al integrarlo en el sistema ya que se debe aislar el sonido ambiente del sonido de vibración de la bicicleta y el sonido del golpe de viento que se crea al iniciar el movimiento. Bajo estas condiciones, la mejor solución que se planteó fue la captura de los datos de contaminación auditiva en un ambiente de baja vibración y alto tráfico. Para el análisis se eligió la ruta de paseo, donde el ciclista se desplaza a bajas velocidad con lo que se logra registrar secciones en las que el usuario experimenta

niveles de ruido entre $40dB$ y $50dB$, cuando hay poco tráfico. Mientras que en secciones de la ruta con tráfico considerable se logra registra valores de $60dB$. En la Figura 5.7, se muestra el comportamiento de los valores registrados por el sensor durante el intervalo de tiempo que dura el recorrido. Cabe recalcar que los valores capturados indican que no se esta cumpliendo la normativa vigente que establece para zonas residenciales máximos de $55dB$ en el día y $45dB$ en la noche. .

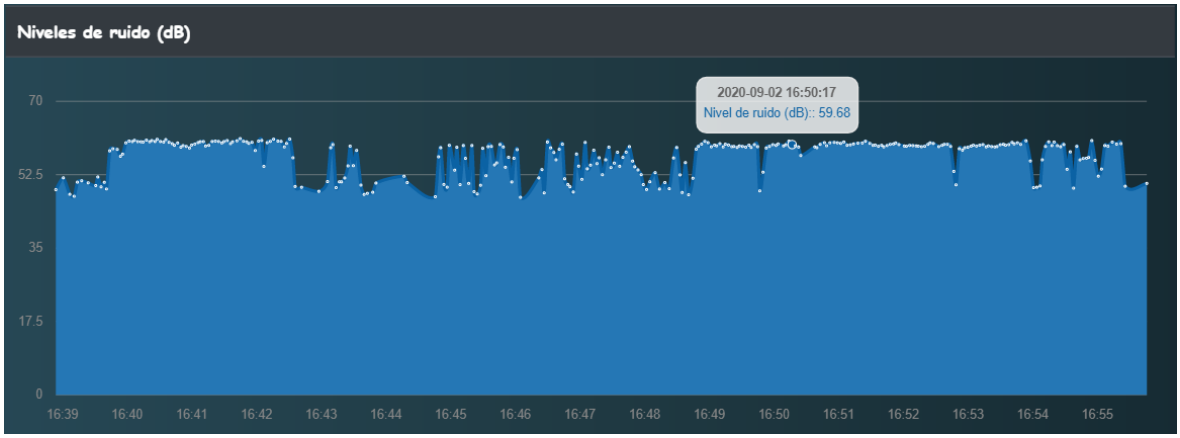
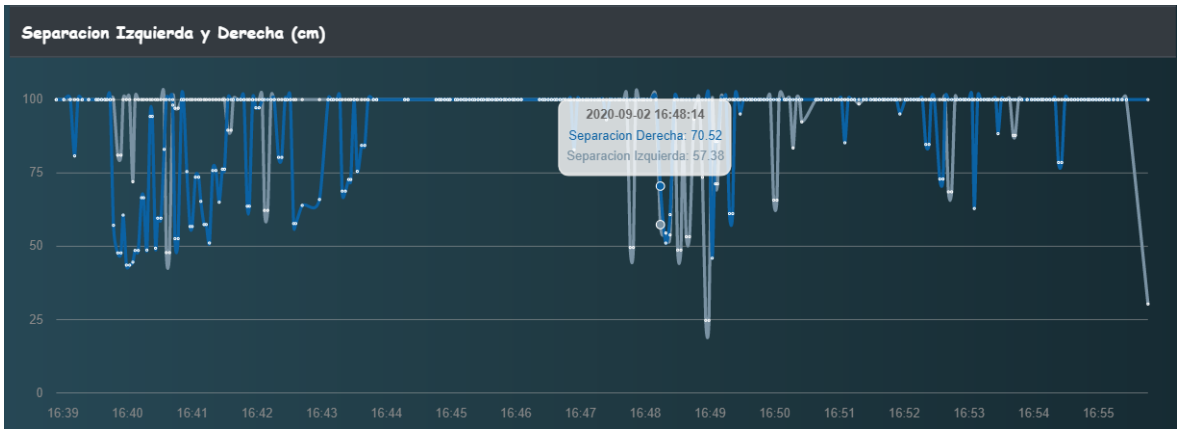


Figura 5.7: Análisis del los niveles de ruido en la Ruta de paseo

- **Separación lateral de obstáculos:** Este parámetro nos permite determinar el espacio de maniobra del que dispone un ciclista al realizar una ruta. Para este caso, se analizó la ruta de estudiantes desde el Campus Central al Campus Yanuncay de la Universidad de Cuenca. En la Figura 5.8, se muestra el resultado de la distancia de separación desde el centro de la bicicleta hacia cada uno de los lados. Se puede apreciar que esta trayectoria esta conformada de tres escenarios en cuanto a separación de obstáculos se refiere. En el primer escenario que se establece desde el inicio de la ruta hasta la hora 16:44, se puede observar que el ciclista se encontró en la necesidad de viajar orillado hacia el lado derecho como se muestra en el fotograma 1 de la Figura 5.8(b), generando así gran cantidad de oscilaciones de distancia menores a 100cm llegando a ser inclusive menores a 50cm. Mientras en el lado izquierdo se registra valores mayores o iguales a 100cm. El segundo escenario por otro lado se establece desde las 16:44 hasta las 16:47 donde se puede observar que todas los valores registrados hacia cada uno de los lados son mayores o iguales a 100cm lo que indica que esta parte del trayecto el usuario goza de una gran espacio para maniobrar y se encuentra dentro de la zona de confort establecida por [5]. Esto se visualiza en el fotograma 2 de la Figura 5.8(b). Finalmente, el tercer escenario se establece desde las 14:48 hasta las 14:50, donde se puede observar que se registra obstáculos tanto en el lado derecho como el lado izquierdo, esto indica que en este punto el ciclista dispone de poco espacio. En la Figura 5.8(a), se puede apreciar que los valores inclusive llegan a ser menores a 25cm lo indica una separación de entre 5cm a 10cm desde el manubrio de la bicicleta hasta el borde del obstáculo. En el fotograma 3 de la Figura 5.8(b), se muestra que el usuario tiene la necesidad de circular entre dos hileras de vehículos, zona en la cual se registran estos valores.



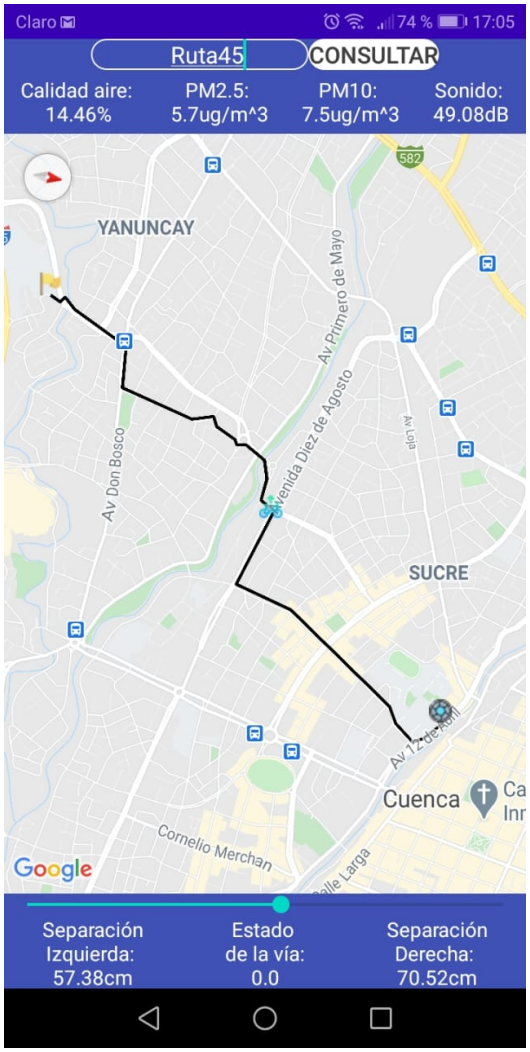
(a) Gráfica de la respuesta en la aplicación web

Fotograma-1

Fotograma-2

Fotograma-3

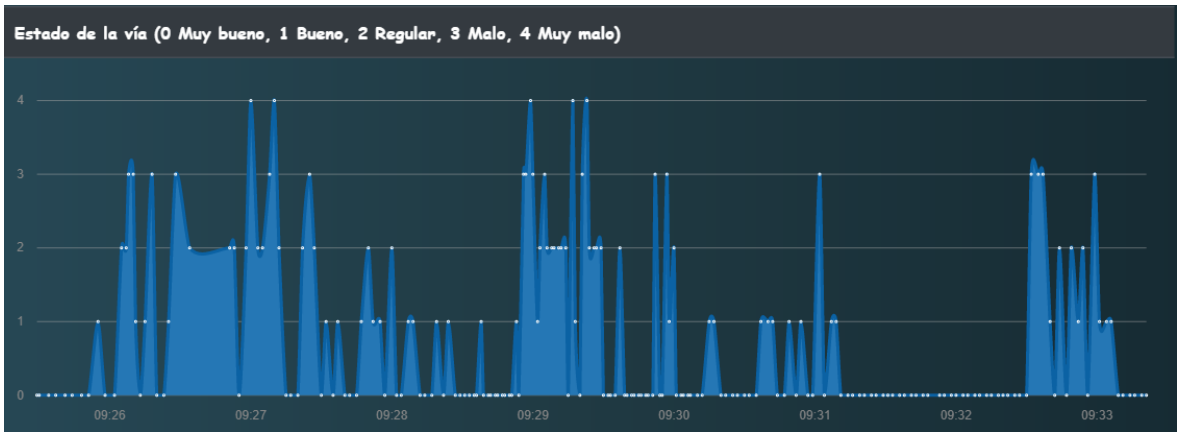
(b) Registro de vídeo de la ruta



(c) Análisis de los valores usando la aplicación móvil

Figura 5.8: Análisis de los resultados de la separación lateral con obstáculos (Ruta Campus Central-Campus Yanuncay de la Universidad de Cuenca)

- **Estado de la vía:** El análisis del estado de la vía se realizó en función del nivel de vibraciones que experimenta un ciclista al recorrer una ruta. Para ello se empleó un acelerómetro que registra los pequeños cambios de aceleración en el eje "Y" (perpendicular al suelo). El uso de este sensor permite que el análisis del estado de la vía sea independiente del tipo de ciclovía y material de construcción de la misma. En este contexto, se estableció 5 rangos de vibración. Estado de la vía 0: representa un camino en muy buen estado donde no se registra vibración o registra vibraciones muy leves, un ejemplo de este tipo de vía es la ruta céntrica siguiendo el camino central del sistema de Alimentación Por Suelo (*APS*) para la alimentación del tranvía. Estado de la vía 1: es un camino en buen estado cuyos intervalos de vibración están entre $-6m/s^2$ y $-8m/s^2$, un ejemplo de este tipo de vías son las ciclovías en buen estado y calles de cemento en buen estado, caminos de tierra planos, entre otros. El estado de la vía 2: representa un camino regular que tiene niveles vibración entre $-4m/s^2$ y $-6m/s^2$, un ejemplo de este tipo de vías son ciclovías con muchos baches, caminos de adoquín en buen estado, entre otros. Estado de la vía 3: se encuentra entre ($-2m/s^2$ y $-4m/s^2$) y representa un camino en mal estado, por ejemplo calles de adoquín normales, ciclovías con baches entre otras. Finalmente el estado de la ruta 4 indica un camino en muy mal estado, donde se establece en valores de vibración entre $0m/s^2$ y $-2m/s^2$, un ejemplo de este tipo de vías son caminos con muchos baches o con relleno de piedras, caminos de montaña y caminos de adoquín en mal estado.
  Para el análisis de este parámetro se usa la ruta céntrica, donde en gran parte se siguió la ruta tranviaria, que esta compuesta por partes de cemento y el sistema *APS*. En la Figura 5.9(a), se muestran los valores de estado del vía capturados por el ciclista al recorrer esta ruta. Los primeros minutos corresponde al usuario siguiendo la parte cementada de la ruta como se muestra en el fotograma 1 de la Figura 5.9(b). A continuación, en el intervalo entre las 9:30 y las 9:32, se muestra que el ciclista inicia una sección del tramo en buen estado y regular, valores que coinciden con el inicio de la zona donde se presenta el sistema *APS* del tranvía como se puede apreciar en el fotograma 2 de la Figura 5.9(b). Por otro lado, en el fotograma 3 de la Figura 5.9(b), se muestra como el usuario cambia de vía hacia una construida con adoquines donde inmediatamente se comienza a registrar mayor cantidad de vibraciones que indican que la vía se encuentra entre un estado regular y malo.



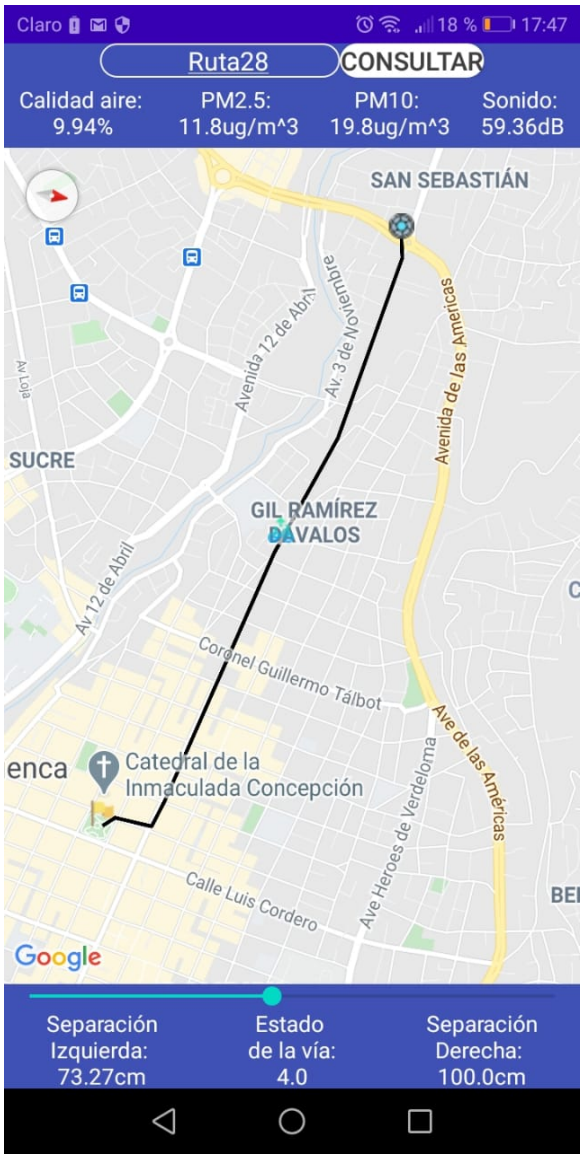
(a) Gráfica de la respuesta en la aplicación web

Fotograma-1

Fotograma-2

Fotograma-3

(b) Registro de vídeo de la ruta

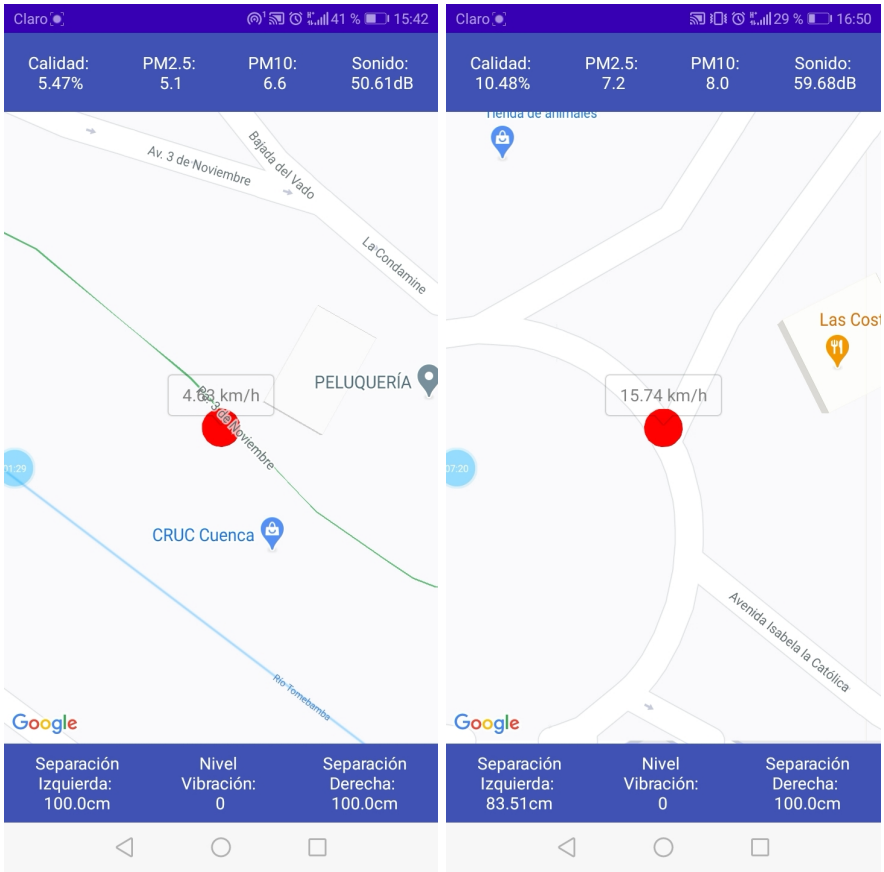


(c) Análisis de los valores usando la aplicación móvil

Figura 5.9: Análisis de los resultados del estado de la vía (Ruta céntrica)

5.3.2. Monitorización en tiempo real

La monitorización en tiempo real se realiza desde la Aplicación móvil o Web. En la Figura 5.10, se muestra capturas realizadas en el momento que los nodos están activos. En la Figura 5.10(a), se observa que el nodo esta recorriendo por la ciclovía cercana al río Tomebamba a una velocidad de 4,63km/h, mientras que en la Figura 5.10(b), se muestra al nodo recorriendo a una velocidad 15,74km/h. En ambos casos se puede decir que las velocidades son comunes en el tipo de vía que se encuentran. Esto se verifico, revisando los vídeos correspondientes a cada ruta, por ejemplo en el caso a) el nodo esta sobre la ciclovía con calzada de adoquín, donde por lo general el usuario disminuye la velocidad debido al incremento en la cantidad de vibraciones percibidas. Mientras en el caso b), en el video se observa que el ciclista pasa el redondel de forma veloz, lo que verifica el dato obtenido. De igual forma los datos correspondientes al ambiente en a) son mas bajos que b) ya que son zonas de bajo y alto tráfico respectivamente.



(a) Recorrido 1 (b) Recorrido 2

Figura 5.10: Capturas realizadas en tiempo real

Por otra parte, en las grabaciones de vídeo de la aplicación móvil, se observa que el nodo actualiza sus datos de manera continua algo que se buscaba que realice esta aplicación. La actualización es estable en la mayor parte tiempo del recorrido, pero también existen ocasiones donde se visualiza datos atrasados o simplemente no llegan los datos y la aplicación permanece estática. Esto es poco frecuente ya que la mayoría de las rutas se puede apreciar un recorrido continuo.

La aplicación web tiene un funcionamiento similar para mostrar los datos en tiempo real. En la Figura 5.11, se muestra una captura del seguimiento que se llevó a cabo en tiempo real. Se observa en el mapa la ubicación de los nodos activos y en la parte inferior se muestra los datos actualizándose de forma continua.

Usuarios	Caidad del aire (%)	PM2.5 (ug/m^3)	PM10 (ug/m^3)	Nivel de ruido (dB)	Separacion izquierda (cm)	Separacion derecha (cm)	Estado de la vía	Velociad (km/h)
Usuario2	4.74	15	18.8	59.43	100	43.62	0	6.11

Figura 5.11: Aplicación web en tiempo real

5.4. Evaluación del Sistema de Comunicación

5.4.1. *Throughput*

El *throughput* es la velocidad de transferencia sobre un canal de comunicación. Es una medida real que es comparada a la teórica. Es decir, un canal de comunicación esta determinado por su ancho de banda el cual indica el número máximo de paquetes que se puede enviar, mientras que el *throughput* indica el número real de paquetes transmitidos. La medida usada es en bits por segundo (b/s), aunque también se puede medir en paquetes por segundo (p/s). El *throughput* de una red se mide con un promedio de las velocidades alcanzadas en un intervalo de tiempo para representar el desempeño de la red.

Generalmente el *throughput* se usa para comparar la velocidad real con la ofrecida. Por ejemplo, cuando se tiene conocimiento del ancho de banda de un canal de comunicación, es posible evaluar ese supuesto, enviando datos a través de la red hasta alcanzar el límite. Aquí es donde se mide la velocidad real de transferencia para posteriormente realizar la respectiva comparación.

Ahora en el sistema implementado, en lo que se refiere a transmisión de datos desde el nodo, los volúmenes de datos enviados a la base de datos a través de la red celular son relativamente bajos. Por lo tanto en teoría la red no tendrá dificultades al momento de la transmisión de datos, sin embargo, hay que tener en cuenta que la red es inalámbrica y aunque se tenga un amplio ancho de banda dado por la operadora, la velocidad de envío de datos puede variar debido al movimiento constante de la bicicleta.

Entonces, se realizó un análisis del *throughput* de la red mientras el nodo esta en movimiento comparando con la velocidad de transferencia de datos mientras el nodo permanece estático. Para la evaluación, se estableció dos escenarios: nodo estático y nodo en movimiento.

En la Figura 5.12 se muestra los resultados obtenidos del *througput* del nodo en movimiento y estático. El método que se utilizó fue en los primeros diez minutos mantener sin movimiento a la bicicleta mientras esta enviando datos a hacia la base de datos. Luego, empezar a recorrer por 10 minutos más, esto se realizó en varias rutas, para luego sacar un promedio del *througput* en cada caso.

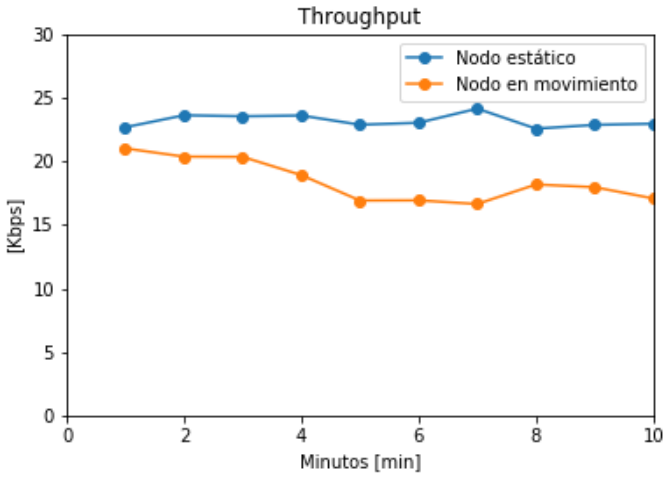


Figura 5.12: *Throughput* de la red móvil

Los resultados obtenidos muestran que cuando el nodo se mantiene estático presenta una mayor velocidad de transferencia de datos. Entonces mientras el nodo esta en movimiento hay disminución de transferencia de datos. Por otra parte, las gráficas indican que la transferencia de datos se mantiene estable y dado que los paquetes enviados son de bajo tamaño para lograr recibirlos en tiempo real, se puede decir que la velocidad de transferencia es adecuado para este sistema. Esto también se puede constatar en la siguiente sección cuando se evalúa la pérdida de paquetes.

5.4.2. Pérdida de datos por intermitencia de la red

El análisis de la pérdida de datos que puede sufrir la estación se realiza sobre el propio nodo, donde se determina cuando un conjunto de muestras censadas no pudo ser enviado en tiempo real al servidor y se contabiliza el mismo. Por lo general, las pérdidas de datos se debe a que el *User Equipment* (*UE*) pierde conexión con el *envolved Node B* (*eNodeB*). Cuando sucede esto, el *UE* inicia un nuevo proceso de conexión y si éste toma más de 3 segundos los valores capturados en este periodo de tiempo serán descartados, mientras que si la conexión se restablece se enviarán los valores almacenados y continuará normalmente con las siguientes lecturas. El análisis de la pérdida de datos se realizó en 13 rutas, las mismas que entregaron los resultados que se muestran en la Figura 5.13, donde se observa que la pérdida esta en un rango de entre el 1 % y 10 % del total de muestras capturadas. La pérdida de datos es independiente del tipo escenario en el que se realiza las pruebas y del tiempo de la muestra. Debido a que no se puede estimar cuando el nodo realiza un salto desde un hacia otro, dado que los mismos en la red *LTE Advance* son auto administrados. Lo que significa que éstos deciden cuando se debe

realizar el *handover* del usuario en base a parámetros como la intensidad de la señal, los valores de saturación entre otros.

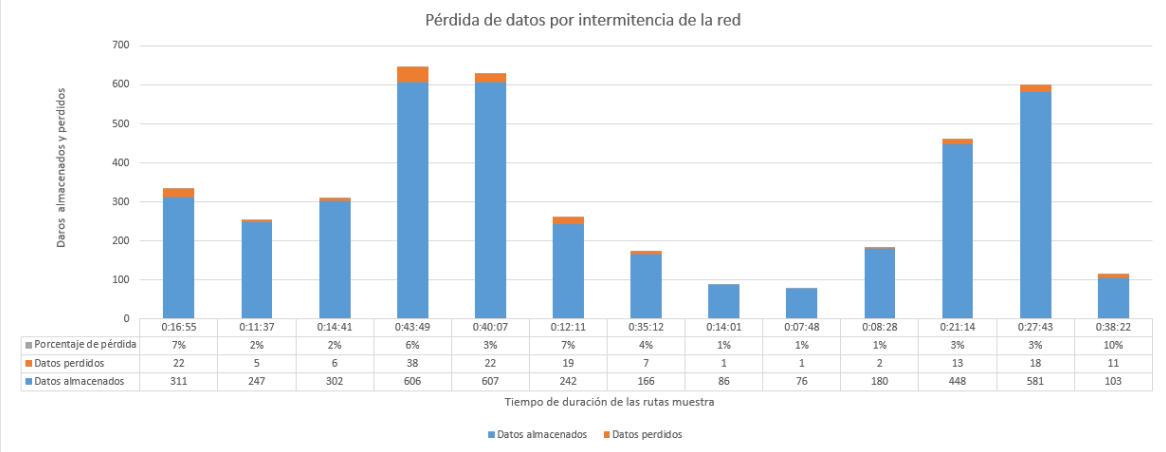


Figura 5.13: Análisis de la pérdida de datos por intermitencia de la red

5.4.3. Retardo

Para medir el retardo que experimentan los datos se utilizó el comando *ping* desde la estación en movimiento hacia el servidor de *firebase* donde esta almacenado nuestra base datos, cuya dirección es bikenode-bc16b.firebaseio.com. Al utilizar esta metodología se obtiene los siguientes resultados, en una ruta cualquiera cuya duración es 27 minutos con 43 segundos se capturan 355 muestras siendo el tiempo máximo de respuesta de 398ms y el tiempo mínimo de 92,1ms, el promedio de todas las muestras se establece en 136ms con desviación estándar de 38,85ms. Sin embargo, se debe tener en consideración este análisis muestra el tiempo de respuesta del servidor, por lo cual, bajo el planteamiento de que el tiempo que le toma a un paquete viajar desde el Cliente al Servidor es el mismo que le toma viajar a otro desde el Servidor al Cliente se puede establecer que el retardo promedio que sufre la estación será de 65ms con desviación estándar de 19,43ms como se muestra en la Figura 5.14.

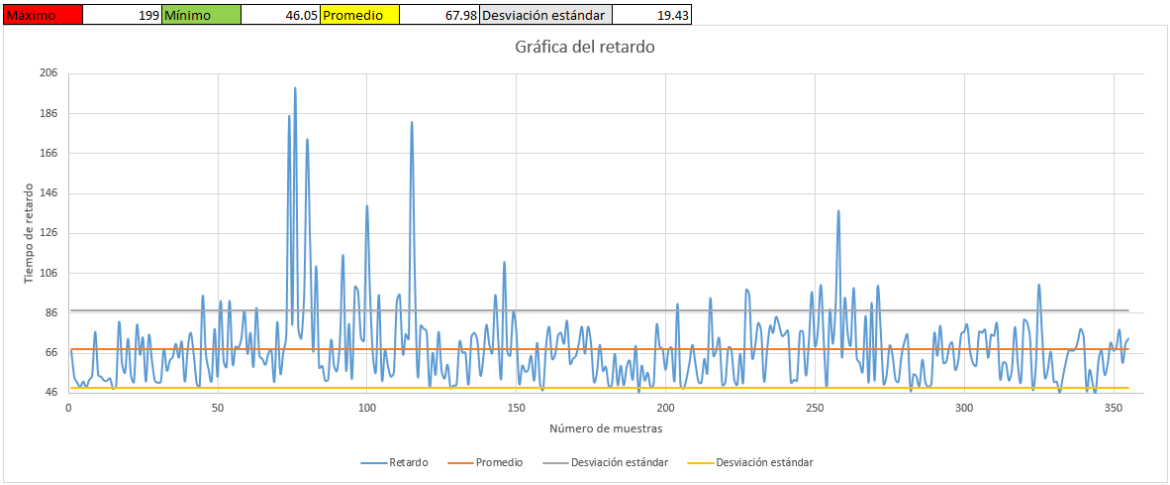


Figura 5.14: Análisis del los valores retardo

5.5. Análisis del rendimiento de la estación

5.5.1. Carga del CPU

La carga de CPU o nivel de uso describe en que medida el procesador esta ocupado en la gestión de programas y procesos que están ejecutándose. Se trata del porcentaje del tiempo de proceso total que la *CPU* usa realmente. El valor máximo llega a 100 %. En particular, la unidad central de proceso del nodo, es la Raspberry Pi. Es así que se realiza el análisis de carga de su procesador.

El sistema operativo de la Raspberry Pi basado en *linux* permite mediante comandos obtener la carga del CPU, el cuál fue tomado cada cierto tiempo. Esto se realizó en algunos recorridos obteniendo resultados similares.

En la Figura 5.15, se muestra el resultado obtenido de la toma de datos en algunas rutas, pero todas muestran el mismo comportamiento, es así que se obtiene los valores máximos y mínimos de cada ruta y se obtiene un promedio. Entonces se puede decir que el nodo esta usando como máximo el 10 % de la *CPU*.

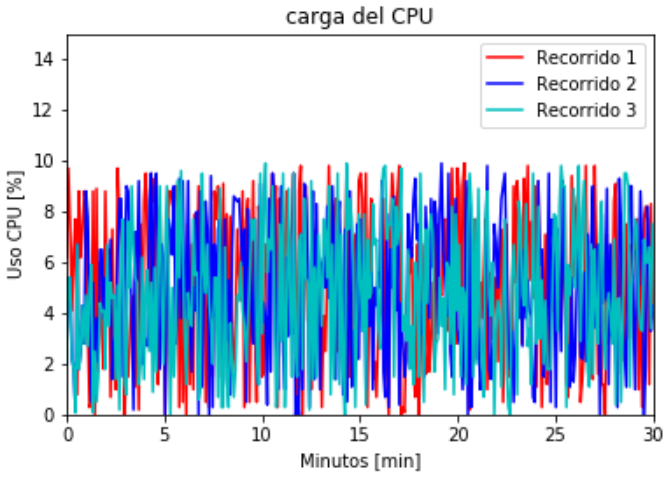


Figura 5.15: Carga del CPU

5.5.2. Temperatura

En esta parten se realiza el análisis de la temperatura del nodo, sobre todo haciendo énfasis en la unidad de procesamiento. De igual manera que la carga de *CPU* se puede aplicar comandos para obtener la temperatura actual de la Raspberry. En este contexto, se ha tomado datos de la temperatura mientras se realiza algunos recorridos.

Para realizar el análisis se tomo en cuenta dos escenarios: recorrido en día soleado y en día nublado. Los resultados obtenidos se muestran el Figura 5.16. Al observar la gráfica se puede inferir que en días de bastante calor el nodo eleva su temperatura con mayor rapidez. Hay que tomar en cuenta que el crecimiento no se da en cuestión de minutos, como se indica en la gráfica tuvieron que pasar 30 minutos para que el nodo eleve su temperatura de 50 a 60^oC.

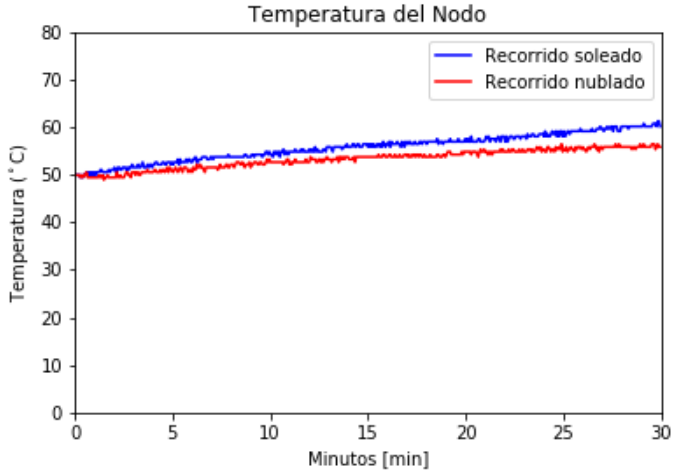


Figura 5.16: Temperatura del nodo

Entonces la temperatura del nodo aumenta lentamente durante el recorrido. En días soleados la temperatura crece más rápido, pero, los resultados demuestran que no se llega a la temperatura máxima que soporta el nodo que es de 85^oC. Entonces se establece que la temperatura del nodo para una ruta realizada alrededor de los 30 minutos, no incide mayormente en el funcionamiento del mismo.

5.5.3. Consumo energético

La fuente de alimentación de la estación consiste en una batería de $26800mAH$, que dispone de 3 puertos USB donde cada uno de ellos puede entregar $2{,}4A$ como máximo al utilizar solo uno de los puertos y $5{,}5A$ como máximo total. Al usar los 3 puertos en paralelo, con estos valores ofrecidos por el fabricante se establece que el banco de baterías puede entregar el máximo de $26800mAH$ en 11 horas y 10 minutos al utilizar solo uno de los puertos y 4 horas con 50 minutos utilizando todos los puertos en su máxima capacidad.

El análisis del consumo energético se realizó usando el sensor de corriente INA2019, el cual mide el nivel de corriente utilizado por toda la estación durante el desarrollo de cada una de las diferentes rutas, para el análisis se usa los datos capturados por el nodo al desarrollar 14 rutas, donde cada una de ellas tiene tiempos de duración distintos, por lo que no se dispone del mismo número de muestras. Bajo este contexto en la tabla 5.1, se muestra las características principales de cada una de las rutas seleccionadas, se debe mencionar que el nivel de voltajes es una constante de $5V$.

Tabla 5.1: Características de las rutas seleccionadas para el análisis del consumo energético

Ruta	Duración	Muestras	Máximo (mA)	Mínimo (mA)	Promedio (mA)
Ruta 45	0:16:55	311	1061	673	809
Ruta 46	0:11:37	247	908	696	765
Ruta 47	0:14:41	302	949	655	731
Ruta 51	0:43:49	606	938	674	754
Ruta 52	0:40:07	607	1080	658	742
Ruta 53	0:12:11	242	990	667	738
Ruta 55	0:35:12	166	1060	695	736
Ruta 56	0:14:01	86	883	691	726
Ruta 57	0:07:48	76	751	682	717
Ruta 59	0:08:28	180	1122	675	738
Ruta 60	0:21:14	448	979	668	744
Ruta 61	0:27:43	581	960	652	760
Ruta 62	0:38:22	103	988	665	771

En la Figura 5.17, se muestra como el comportamiento de consumo de corriente en cada una de las rutas presenta un modelo similar en el tiempo, donde el máximo absoluto se da en la ruta 59 y el mínimo absoluto en la ruta 61. Con los resultados obtenidos, se puede establecer que en el peor de los casos el nodo consumirá 1,122AH, esto permite que con una sola carga el sistema funcione hasta por 23 de viajes de una hora (Tiempo limite de funcionamiento). Del mismo modo, en este caso el nodo consumirá 5,56WH del total de energía de almacenamiento 134W. Se debe mencionar que el análisis se realizo bajo la premisa de que el nodo consume un valor de corriente constantes de 1,122A, valor máximo registrado.

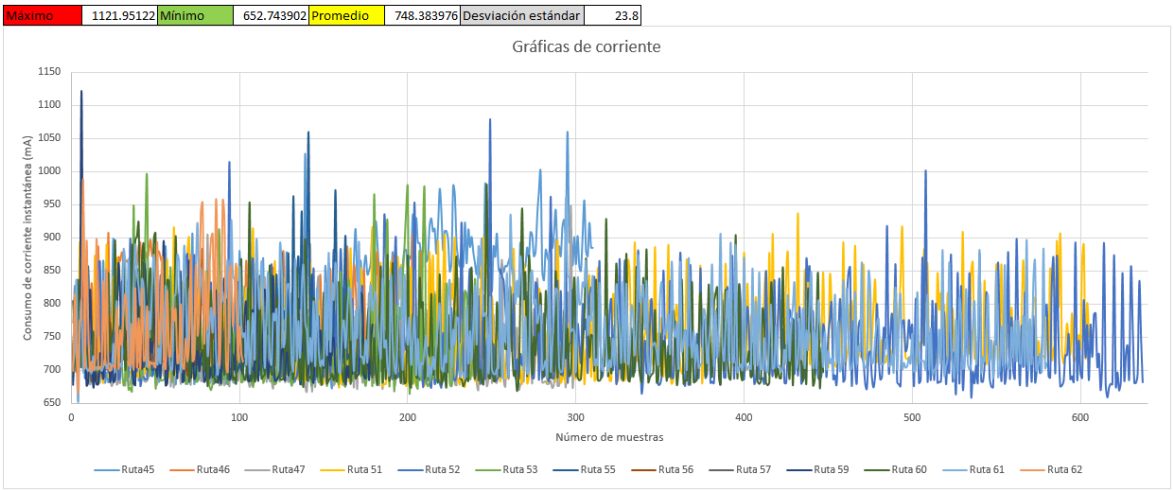


Figura 5.17: Gráficas del consumo de corriente instantánea (Promedio de varias rutas)

Del mismo modo, al analizar el consumo de energético, con el valor promedio de todas las rutas establecidas y la desviación estándar de consumo de corriente, se puede establecer un tiempo de funcionamiento de 34,67H entregando un valor constante de 773mA, lo que da como resultado un consumo de 3,85WH.

Por otro lado, en la Figura 5.18, se muestra el comportamiento del consumo de corriente instantánea en una ruta cualquiera con un periodo de duración de una hora (tiempo limite del funcionamiento del nodo). Para el análisis del consumo energético se establece una linea de tendencia polinomial de sexto grado, como se muestra en la Figura 5.18, la ecuación indica que los valores asociados a la variable polinomial no son relevantes respecto a la constante, con lo cual se puede establecer, que durante este recorrido se consumió aproximadamente 718mAH de energía eléctrica.

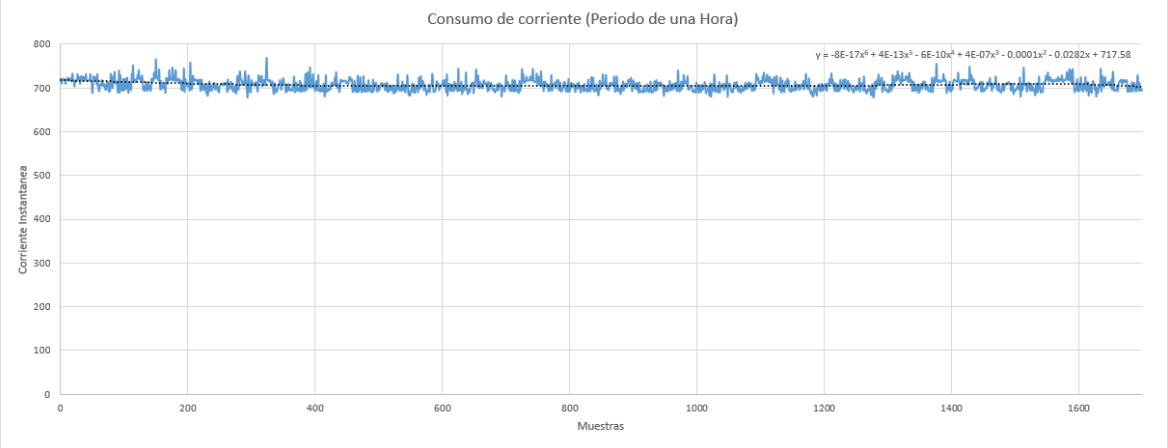


Figura 5.18: Gráficas del consumo de corriente instantánea (Periodo de una hora)

5.6. Conclusiones

Al realizar el análisis del rendimiento de las aplicaciones se observó que cumplen a la perfección con el objetivo de permitir al usuario registrarse, monitorizar y analizar los datos adquiridos al realizar ciclismo en la ciudad. El registro de los usuarios se puede realizar únicamente desde la aplicación móvil para lo cuál se requiere disponer de correo personal y contraseña además de brindar información que permita recuperar o eliminar la misma. El registro se realiza inmediatamente después de que el usuario ingresa los datos, entonces podrá iniciar sesión, visualizar los nodos activos y los diferentes valores que están capturando los nodos en tiempo real. En caso de usuarios que ya disponen de un historial de rutas, podrán analizar las rutas que hayan desarrollado con anterioridad. Por otro lado, la aplicación web permitirá analizar las rutas desarrolladas por todos los usuarios, en este caso la aplicación se desarrolló para la administración de los valores capturados en las diferentes rutas. Para esto se brinda gráficas lineales y de área que permitan al usuario observar el comportamiento de las variables a medida que el ciclista avanzan por la ruta.

Los valores capturados por la estación móvil permiten determinar el nivel de confort de un ciclista en cualquiera de las rutas desarrolladas. Dado que los mismos presentan datos reales de las rutas en cuanto a niveles de calidad del aire, donde se usa la concentración en ppm de *CO2* para detectar la emisión de gases de los vehículos, el volumen de material particulado inferior 10 micras y 2,5 micras (partículas que son nocivas para la salud de las personas), la separación entre un obstáculo lateral y la bicicleta, con lo que se logra detectar la factibilidad de maniobra del ciclista, los valores de vibración en toda la ruta, con lo cual se establece el estado de la vía independientemente del tipo de ciclovía utilizada y los niveles de contaminación sonora, del mismo modo mediante el uso de la cámara se puede registrar

otros eventos que afectan el confort del ciclistas como se muestra en el anexo B. Se debe mencionar que los sensores responsables de la captura de datos de las cuatro primeras variables presentaron un comportamiento completamente funcional. Es decir, se acopla a cualquier ruta, velocidad y maniobra con la que el usuario haga ciclismo. Sin embargo, el sensor MAX9814 no tuvo el mismo rendimiento dado que presentó problemas en la captura a alta velocidad y en vías en mal estado, debido a que se requiere aislar el sonido del golpe de aire y el sonido de vibración de la bicicleta.

Una vez realizadas varias pruebas con diferentes escenarios se establece que la estación móvil puede ser usado por toda la ciudad de Cuenca. Además de analizar los datos entregados por el nodo. También se analizó los niveles de temperatura a los que se encuentra la estación, encontrando que la temperatura promedio se encuentra muy por debajo de la máxima temperatura permitida. Del mismo modo, se observó que el uso del *CPU* es de apenas el 10 % de la capacidad total del procesador. Por otro lado, la estación realiza un consumo energético muy bajo lo que alarga el tiempo de vida de la batería. Finalmente, se analizó el rendimiento de la red móvil obteniendo resultados que demuestran que la conexión es fiable. En definitiva, el nodo construido cumple con sus funciones establecidas: registro de datos, transmisión en tiempo real, acople a la bicicleta y sobre todo entrega datos completos sobre una ruta. Además, con la cámara ubicada en el nodo se puede verificar la veracidad de los datos obtenidos.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

La implementación del sistema de monitorización mostró grandes resultados en cuanto a la captura de datos de los diferentes parámetros que afectan al confort del ciclista, siendo estos: El nivel de calidad del aire para lo cual se usa el sensor MQ135 que permite detectar el incremento en los valores de *CO2* en el ambiente respecto a los niveles en aire puro, este gas es uno de los que mayor presencia tiene en las emisiones producidas por el uso de vehículos de combustión interna; el volumen de material particulado en $\mu g/m3$ de la zona que transita el ciclista, para esto se utilizo el sensor SDS011 que permite detectar material particulado de entre 2.5 micras a 10 micras y menor 2.5 micras de diámetro aerodinámico. Este material es perjudicial para la salud. Los valores de exposición promedio para intervalos de tiempo dados, están debidamente reglamentados por norma Ecuatoriana de calidad del aire. Sin embargo, la *OMS* advierte que no existe rangos por debajo de los cuales no se presenten problemas sanitarios en el caso del material particulado de 2.5 micras; los valores de separación lateral izquierda y derecha desde el centro frontal del cuadro de la bicicleta hacia cualquier obstáculo que se encuentre a una distancia lateral inferior a $100cm$, este valor se estableció en base estudios donde se menciona que el valor de la zona de confort de un ciclista tiene una área de $1{,}7m$ x $3{,}4m$. Para medir este parámetro se usaron sensores ultrasónicos ubicados en cada lado del nodo; el estado de la vía se estableció según los niveles de vibración que presente la bicicleta independientemente del tipo del ciclovía, la captura de los valores se realizo usando el acelerómetro ADXL345 donde se tomó la variación de lecturas en el eje "Y" ubicado perpendicular al suelo; y para los valores de contaminación auditiva se utilizó el sensor MAX9814 el mismo que captura niveles de ruido desde los $0dB$ a los $84dB$, sin embargo, el uso de este sensor no mostró los resultados esperados debido a que registra los valores de ruido generados por las vibraciones de la bicicleta y el sonido del golpe de viento al iniciar el movimiento, por lo que este sensor a diferencia de los otros se puede usar solo en rutas en buen estado y bajas velocidades.

El uso de *Firebase* como servicio de *backend* permitió enfocar todo el esfuerzo del desarrollo de las aplicaciones en servicio de *frontend*, con lo cual se logró crear aplicaciones fáciles de entender y con

alta capacidad para el análisis de los datos y la visualización en tiempo real. Esta plataforma digital se integra tanto; en el nodo, donde usa para almacenar los datos capturados por los diferentes sensores en tiempo real y almacenar las diferentes rutas desarrolladas por los ciclistas; así como también en cada una de las aplicaciones de monitorización, donde se usa para solicitar la información almacenada tanto de los datos en tiempo real como de los datos de las diferentes rutas registras. Sobre la aplicación móvil se integró además el servicio de autenticación, el mismo que permite registrar al usuario y verificar al mismo, mientras que la aplicación web por otro lado emplea el servicio de *hosting* de *Firebase* para almacenar la aplicación.

El desarrollo de las aplicaciones de administración y monitorización fueron realizadas de forma tal, que el manejo por los usuarios sea de forma intuitiva e integre la mayor cantidad de funcionalidades posibles. Bajo este contexto, se desarrolló una aplicación móvil nativa que permite utilizar todas las capacidades del dispositivo móvil en cuanto a la velocidad de procesamiento, uso de gráficas y seguridad. También se realizó una aplicación web que brinda características de interoperabilidad con diferentes dispositivos, independientemente del tipo de sistema operativo o las dimensiones del mismo. El único requisito para el uso de esta aplicación es disponer de conexión a Internet desde la cual se pueda acceder a la *URL* del sitio web y una pantalla donde se visualicen los datos. El desarrollo de cada aplicación se realizó desde un punto de vista diferente, la aplicación móvil enfocada en los usuarios y la aplicación web para analizadores de datos. La aplicación móvil brinda al usuario la posibilidad de registrarse e iniciar sesión, motiva a generar nuevas rutas impulsando el ciclismo y permite el *tracking* de todos los usuarios activos en tiempo real y la visualización de los datos que los mismos están registrando, así también permite al realizar revisiones de todas las rutas almacenas previamente por el propio usuario, donde el mismo podrá dar seguimiento al recorrido desarrollado y observar los valores censados por su estación en los diferentes puntos de la trayectoria. Por otro lado, la aplicación web permite analizar el comportamiento de las variables de interés en cada una de las rutas almacenadas por el conjunto de todos los usuarios. Del mismo modo que en el caso de la aplicación móvil se visualiza el *tracking* desarrollado en cada ruta y los valores censados en los diferentes punto de la ruta. Sin embargo, esta aplicación integra además gráficas lineales y de área del comportamiento de las variables durante la trayectoria, así también se integra la capacidad de monitorización de los usuarios en tiempo real donde se presentan las mismas características de la aplicación móvil. Se debe mencionar que para visualizar el *tracking* se usó la *API Maps* de *Google* debido la fácil integración de la misma sobre cada una de las aplicaciones y el nivel de desarrollo que esta aplicación tiene sobre otras similares.

La integración de la estación sobre la estructura de la bicicleta se desarrolló en dos partes donde la primera consiste en el cuerpo principal que contiene todos los elementos de la estación y la segunda es el sistema de acoplamiento entre el cuerpo principal y la estructura mecánica de la bicicleta permitiendo de esta manera que el sistema se se pueda integrar sobre cualquier tipo de bicicleta, independientemente de la forma y estructura de la misma. El cuerpo principal se desarrolló de dos materiales diferentes, con acero inoxidable de forma artesanal y con poliácido láctico mediante diseño en $3D$, donde se pudo observar que cada uno presenta sus propias ventajas y desventajas. Por ejemplo, la estructura metálica se integra con mayor firmeza y presentan mayor resistencia a golpes, mientras que por otro lado la estructura desarrollada en $3D$ presenta menores valores en los niveles de ruido ante vibración y permite crear diseños que se acoplan perfectamente a cada uno de los sensores con mayor facilidad. Por otro lado, el sistema de acoplamiento se desarrolló usando platinas de hierro de $1/4$ de pulgada de ancho y un grosor de $0{,}5cm$ lo que brinda estabilidad al sistema. Para asegurar el sistema de acoplamiento se

usa dos bridas lo que permite montar y desmontar el sistema de forma rápida y fácil.

La programación del nodo realizada en python3, incluyo varias librerías, el uso de varios subprogramas que trabajan en paralelo con el programa principal, el manejo de la cámara con su respectiva codificación, el uso de la *GPIO* para el manejo de los sensores que además usaron diferentes protocolos de comunicación, la comunicación por comandos AT con el módulo 4G y por último la transferencia de datos hacia Internet. Ante todo esto, el lenguaje de programación se mantuvo estable, entregando las respuestas que se solicitaron en los tiempos correctos. Además, usar este lenguaje de programación generó poco uso del CPU. Cabe destacar que para el manejo de la mayoría de elementos, Python3 ya contaba con librerías.

6.2. Recomendaciones

Si se desea replicar el trabajo se recomienda utilizar un sensor que capture y grabe los sonidos del ambiente, para mediante procesamiento digital de señales identificar el ruido generado por el tráfico vehicular y eliminar ruido generado por las vibraciones de la estructura de la bicicleta y el ruido del golpe de viento. Todo esto utilizando las frecuencias características de cada uno de estos eventos. Además, se recomienda reubicar este sensor en la parte parte posterior de la estructura de contención lo que reduce los niveles de ruido por golpe de viento.

Por otra parte, para trabajos similares a este proyecto se recomienda seguir usando el mismo lenguaje de programación, ya que ha dado buenos resultados en el manejo de múltiples componentes al mismo tiempo. Además, en Internet existe un gran número de librerías de soporte para Python3.

La ubicación de cada uno de los sensores mostró resultados excepcionales en cuanto a la captura de datos salvo por el sensor de sonido. Por otro lado la ubicación del actuador (pulsante) utilizado para finalizar la ruta pese a que funcionó según las expectativas planteadas, resultó un tanto incomodo para el ciclista su ubicación frontal. Se recomienda reubicar este actuador en la parte superior (junto al indicador del estado del nodo) del cuerpo principal.

Para mejorar la calidad del vídeo se recomienda verificar que la cámara que se adquiera sea la versión 2 del módulo que brinda una calidad de hasta 8 megapixeles o se puede adquirir una cámara de alta calidad de hasta 12.3 megapixeles con lentes intercambiables.

El uso de *Firebase* como servicio se *backend* se recomienda ampliamente incluso si el número de nodos se incrementa hasta *superar* el limite del plan *spark*, debido a que los servicios que integra son mas representativos que el costo por consumo del plan *blaze*.

6.3. Trabajos futuros

A continuación se describe algunos trabajos futuros que pueden desarrollarse como un complemento a la propuesta tecnológica presentada en este trabajo de titulación, los mismos que no se profundizaron debido a que superan el alcance de esta tesis, además se sugiere algunos desarrollos específicos para apoyar y mejorar el modelo y la metodología propuesta en esta solución tecnológica.

Crear una segunda versión de la propuesta tecnológica donde se integre todas las recomendaciones mencionadas en el punto anterior. Esto reducirá el tamaño del nodo y mejorará las capacidades de captura de datos en cuanto a niveles de ruido, así también diseñar un sistema de seguridad mecánico que

permita proteger al nodo contra posibles robos. El sistema debería funcionar sin alterar la estructura de la bicicleta.

Analizar el comportamiento de las diferentes variables de la red en un entorno multi-operador, es decir, probar el desempeño del nodo con cada una de las operadoras móviles y crear mapas de cobertura según las lecturas de intensidad de señal de la red *LTE* de cada una de ellas. También realizar un análisis comparativo en el rendimiento del nodo al utilizar el módulo 4G, datos compartidos desde un teléfono móvil y ambos para la conexión a Internet.

Investigar sobra la factibilidad de la integración de algoritmos de redes neuronales para detectar eventos de peligro al desarrollar una ruta y analizar el tipo de ciclovía y el estado de misma mediante el uso de la cámara de la Raspberry Pi.

Analizar el consumo de datos al integrar servicios de transmisión del *streaming* de la ruta en tiempo real y el comportamiento de diferentes parámetros de la red (*Troughput*, retardo, y perdida de paquetes) frente al incremento del consumo de ancho de banda.

Integrar más funcionalidades en las aplicaciones donde se pueda visualizar: El número de horas acumuladas de ciclismo; pequeños cortos de vídeo de las rutas que están siguiendo los usuarios activos en tiempo real; el porcentaje de exposición del ciclista a los contaminantes durante un determinado periodo de tiempo, buscar compañeros que estén desarrollando rutas específicas, entre otras.

Investigar la factibilidad en el uso de redes *Ad-Hoc MultiHop* al incrementar el número de usuarios, lo que permitiría que los datos capturados viajen entre los nodos hasta encontrar una estación base donde se almacenan los datos, eliminando la necesidad del uso de las redes móviles.

Anexo A

Registro de vídeo de las ciclovías

A.1. Tipo de ciclovías existentes en la urbe de la ciudad

En este apartado se muestra los diferentes tipos de ciclovías donde se realizaron las pruebas de los nodos, sin embargo, se debe mencionar que las pruebas también se desarrollaron donde no existe estructura vial debido a la falta de interconexión entre las mismas, obligando al ciclista a transitar por veredas y vías de alta velocidad.

Figura A.1: Ciclo vereda

Figura A.2: Ciclovía reservada

Figura A.3: Ciclovía compartida

Figura A.4: Ciclovía integrada

Figura A.5: Ciclovía segregada

Anexo B

Registro de vídeo de los principales eventos que afecta el confort del ciclista

En esta sección se muestra los diferentes eventos registrados que afectan directamente el desarrollo del ciclismo, lo que reduce el nivel de confort experimentado por los ciclistas al recorrer cada una de las rutas.

B.1. Malestar generado por vehículos

Figura B.1: Vehículo estacionado en la vereda donde no existe infraestructura para el ciclismo

Figura B.2: Vehículos muy apegados a las veredas en ciclovías compartidas

Figura B.3: Vehículos que no respetan los semáforos

Figura B.4: Vehículos estacionados en los conectores de ciclovías

Figura B.5: Vehículos que rebasan al ciclista muy cerca del mismo

Figura B.6: Vehículos que se cruzan en vías compartidas

B.2. Malestar generado por el estado de la vía

Figura B.7: Huecos en aceras de rutas de alto tráfico, donde no hay estructura de ciclovías

Figura B.8: Deformaciones en las aceras

Figura B.9: Calles de adoquín en mal estado que generan alto nivel de vibración

Figura B.10: Obras inconclusas que limitan el tráfico

Figura B.11: Huecos en conectores entre ciclovías

Figura B.12: Veredas demasiado altas

B.3. Otros eventos que generan malestar

Figura B.13: Obstrucción en ciclovías reservadas

Figura B.14: Ciclistas angostas, limitados espacio para rebasar

Figura B.15: Rutas céntricas con alto tráfico y aglomeración de personas

Figura B.16: Falta de planificación para que los ciclistas puedan cruzar los puentes con alto tráfico vehicular de forma segura

Anexo C

Elemento de la estación, estructura mecánica y acoplamiento

En este anexo se muestra todos los componentes electrónicos y mecánicos de la estaciones, así también como se integran los mismos sobre la estructura de las bicicletas.

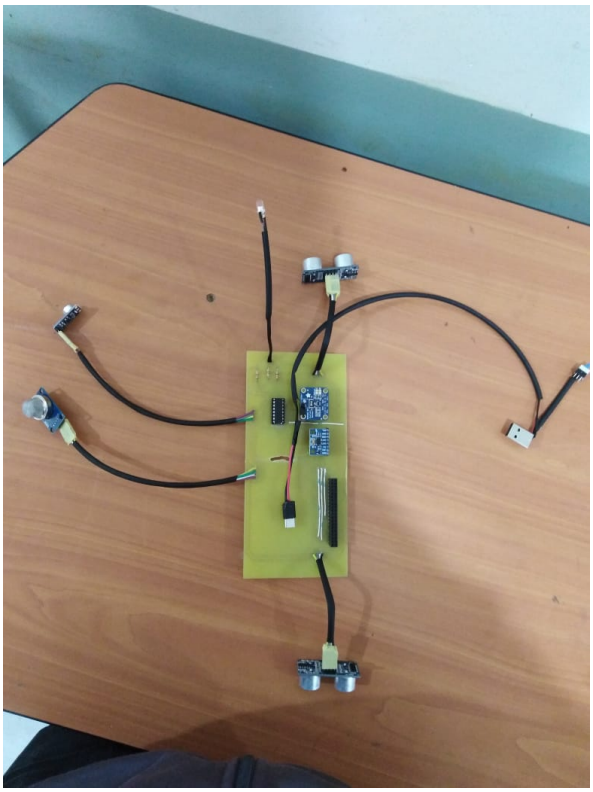


Figura C.1: Integración de los sensores sobre el *PCB*

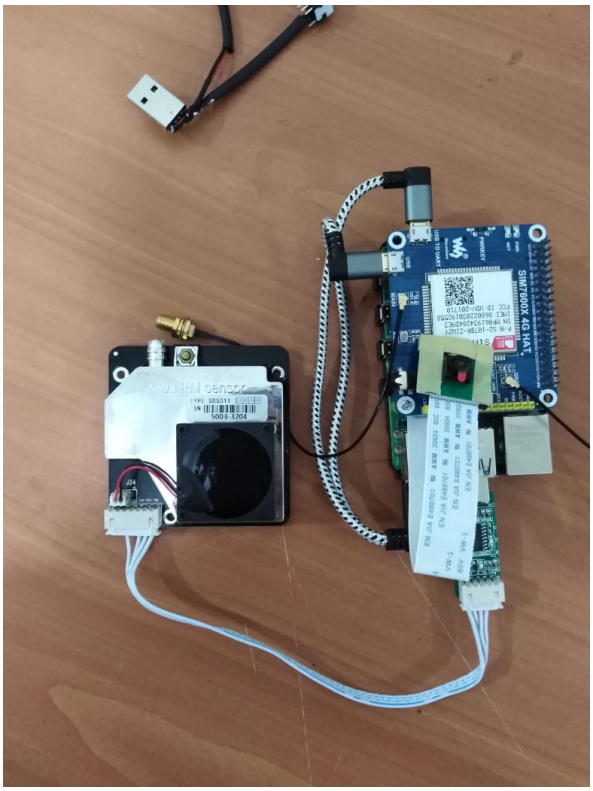


Figura C.2: Conexión del sensor SDS011, el módulo 4G y la cámara sobre el *SBC*

Figura C.3: Conexiones de todo el sistema

Figura C.4: Integración de los elementos en la estructura de contención

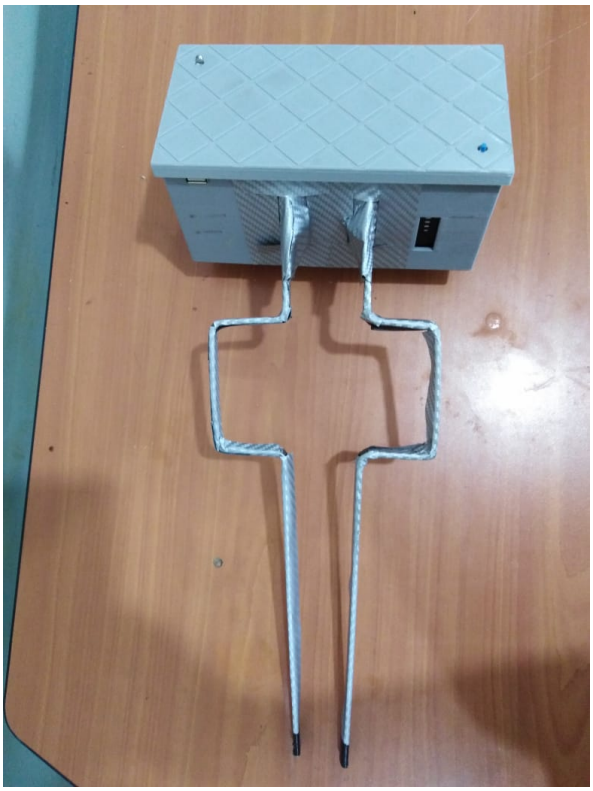


Figura C.5: Estructura de acoplamiento y contención

Figura C.6: Apertura para visualizar el estado de almacenamiento del banco la baterías

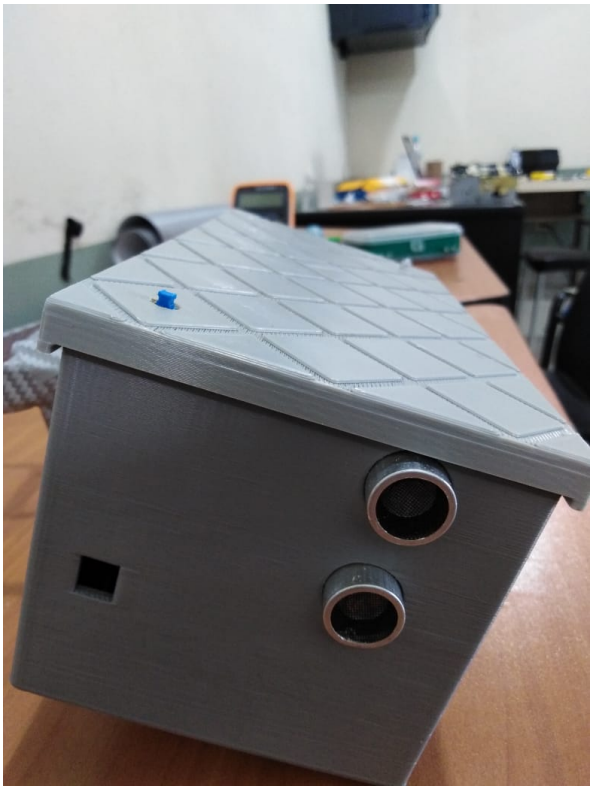


Figura C.7: Vista lateral de la estructura

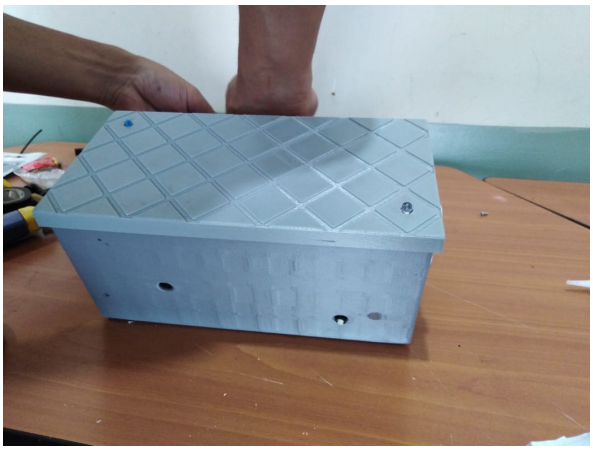


Figura C.8: Vista frontal de la estructura

Figura C.9: Puerto USB para descargar los datos de vídeo

Figura C.10: Acoplamiento de la estación en la estructura de las bicicletas

Figura C.11: Nodos en funcionamiento

Anexo D

Programación del nodo

D.1. Programa Principal

El *script* mostrado a continuación es el programa que se ejecuta al arrancar la Raspberry Pi, cuenta con 496 líneas de código, por esta razón se muestra en partes. Además, existen *scripts* externos que se ejecutan en el programa principal, los cuáles se describen en la siguiente sección. A continuación se muestra las librerías usadas, algunas son instaladas directamente con pip3, mientras que las otras son *scripts* que contienen una función.

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import mcp3208
from mcp3208 import MCP3208
import pyrebase
import time
import verficared
import firebase as fire
import p3
import distanciaboard as distancia
import board
import digitalio
import busio
import adafruit_adxl34x
import queue
import threading
import picamera
import pickle
import time
import sys
import usb
import os
import redmovil
import datetime as dt
from ina219 import INA219
from numpy import interp
```

```
from math import sin, cos, sqrt, atan2, radians
from multiprocessing import Process
cola=queue.Queue()
cola1=queue.Queue()
cola3=queue.Queue()
cola4=queue.Queue()
adc = MCP3208()
camera = picamera.PiCamera()
offhilos=True
R=6373.0
```

Las siguientes líneas de código corresponden a programación de hilos que ejecutarán en segundo plano. Esto se realiza porque algunos sensores requieren estar en constante funcionamiento, como el sensor de sonido capturando los picos alto de sonido, el sensor de vibración tomando muestras de la calzada a cada instante, la cámara que deberá estar capturando en toda la trayectoria, etc. Además se incluyen otras funciones que serán llamadas periódicamente por el programa principal.

```
##############################PINES PARA EL BOTON Y LED#####################################
led = digitalio.DigitalInOut(board.D6)
led.direction = digitalio.Direction.OUTPUT
button = digitalio.DigitalInOut(board.D5)
button.direction = digitalio.Direction.INPUT
########################SENSOR DE SONIDO#########################################################
def funcionA(data):
    try:
        db.child("RealTime").child("Usuario1").set(data)
    except:
        print("no hay internet")
def ejecutarConTiempoLimite(func, args, time):
    p = Process(target=func, args=args)
    p.start()
    p.join(time)
    if p.is_alive():
        p.terminate()
        print("Ha finalizado por timeout")
        return False
    print("Se ha ejecutado correctamente")
    return True
def sensorsonido():
    cola1.put(0)
    nivel = 0
    while (nivel!=-1):
        sampleWindow = 50
        startMillis= time.time()*1000
        peakToPeak = 0
        signalMax = 0
        signalMin = 1024
        while((time.time()*1000 - startMillis) < (sampleWindow)):
           sample=adc.read(0)
           if (sample < 4095):
               if (sample > signalMax):
                   signalMax = sample
               if (sample < signalMin):
                   signalMin = sample
```

```
        peakToPeak = signalMax - signalMin
        db = interp(peakToPeak,[0,4095],[33,83])
        try:
            nivel=cola1.get(block=False,timeout=None)
        except:
            print("La cola de sensorsonido se encuentra vacia")
        if nivel<db:
            cola1.put(db)
        else:
            cola1.put(nivel)
#####################ARCHIVO CON EL NUMERO DE RUTA#####################################
def numeroruta():
    fo=open("/home/pi/Desktop/numeroruta.txt","r+")
    h=fo.read()
    ruta=int(h)+1
    fo.close()
    fo=open("/home/pi/Desktop/numeroruta.txt","w")
    fo.write(str(ruta))
    fo.close()
    return ruta
#####################CAMARA ###########################################################
def grabar_video(ruta):
    nombre="/home/pi/Desktop/videos/"+str(ruta)+".h264"
    camera.start_recording(nombre,quality=40)
    camera.annotate_foreground = picamera.Color('black')
    camera.annotate_background = picamera.Color('white')
def alto_video():
    camera.stop_recording()
#####################APAGADO Y ENCENDIDO DEL LED#######################################
def ledindicador():
    led.value=True
    time.sleep(0.15)
    led.value=False
#########################VERIFICAR CONEXION A INTERNET#################################
def internet():
    cont5=0
    while(cont5!=-1):
        response = os.system("ping -c 1 -w 3 www.google.com")
        if response == 0:
            time.sleep(3)
        else:
            redmovil.red()
            time.sleep(2)
###############ACELEROMETRO DETECTAR VIBRACION ######################################
def dis():
    cont3=0
    cola3.put(200)
    cola4.put(200)
    valor=0
    while (valor!=-1):
        de1=100
        iz1=100
        for i in range(5):
            time.sleep(0.2)
            iz=distancia.calculo("izquierda")
```

```
            de=distancia.calculo("derecha")
            if iz<iz1:
                iz1=iz
            if de<de1:
                de1=de
        try:
            valor=cola3.get(block=False,timeout=None)
            cola3.put(iz1)
        except:
            print("error en izquierda")
        try:
            valor=cola4.get(block=False,timeout=None)
            cola4.put(de1)
        except:
            print("error en derecha")
def vibracion():
    i2c = busio.I2C(board.SCL, board.SDA)
    accelerometer = adafruit_adxl34x.ADXL345(i2c)
    cola.put(0)
    cont=0
    cont1=0
    con=0
    promedio=0
    vi=0
    while (cont1!=-1):
        v=accelerometer.acceleration[1]
        if con<6:
            promedio=promedio+v
            time.sleep(0.18)
            con=con+1
        else:
            promedio=promedio/con
            con=0
            if promedio>=-12 and promedio <=-9:
                vi=0
            if promedio>=-9 and promedio <=-7:
                vi=1
            if promedio>=-7 and promedio <=-4:
                vi=2
            if promedio>=-4 and promedio <0:
                vi=3
            if promedio>0:
                vi=4
            if promedio<=-12:
                try:
                    cont=cola.get(block=False,timeout=None)
                    cola.put(cont)
                except:
                    print("La cola de vibracion se encuentra vacia")
            else:
                try:
                    cont=cola.get(block=False,timeout=None)
                    cola.put(vi)
                except:
                    print("La cola de vibracion se encuentra vacia")
```

```
        try:
            cont1=cola.get(block=False,timeout=None)
        except:
            print("error")
        cola.put(cont1)
################ INCIIAR HILOS ##########################################################
threads = []
vib = threading.Thread(target=vibracion)
threads.append(vib)
ssonido = threading.Thread(target=sensorsonido)
threads.append(ssonido)
distance1 = threading.Thread(target=dis)
threads.append(distance1)
inter = threading.Thread(target=internet)
threads.append(inter)
time.sleep(1)
inter.start()
time.sleep(1)
distance1.start()
time.sleep(1)
vib.start()
time.sleep(1)
ssonido.start()
```

Ahora se muestra el código para iniciar el proceso activación de sensores, *GPS*, y realizar la conexión a la base de datos de *firebase*. Además en esta parte se da la subida de datos de algunas rutas que no fueron enviadas a la *firebase*. Esta parte es la más importante ya que aquí se verifica el correcto funcionamiento de cada elemento.

```
############## ACTIVAR DEL SISTEMA#########################################################
ledindicador()
usb.usb()
config = {
    "apiKey": "AIzaSyD4hf0doODmvPidtgYyWUB5CX4Rhi7yN6Y",
    "authDomain": "bikenode-bc16b.firebaseapp.com",
    "databaseURL": "https://bikenode-bc16b.firebaseio.com",
    "storageBucket": "bikenode-bc16b.appspot.com"
}
firebase = pyrebase.initialize_app(config)
db = firebase.database()
config1 = {
    "apiKey": "AIzaSyCvjc2ILSruw4qxdx2J3jxQ9zNNBNa59xg",
    "authDomain": "bikenodedatos.firebaseapp.com",
    "databaseURL": "https://bikenodedatos.firebaseio.com",
    "storageBucket": "bikenodedatos.appspot.com"
}
ina = INA219(shunt_ohms=0.1,
             max_expected_amps = 2,
             address=0x40)
ina.configure(voltage_range=ina.RANGE_16V,
              gain=ina.GAIN_AUTO,
              bus_adc=ina.ADC_128SAMP,
              shunt_adc=ina.ADC_128SAMP)
i=ina.current()
```

```
firebase1 = pyrebase.initialize_app(config1)
db1 = firebase1.database()
usb1=0
puertos=[1,5,0,6,3,4,2,7,8,9]
while (usb1<9):
    try:
        sensor = p3.SDS011("/dev/ttyUSB"+str(puertos[usb1]), use_query_mode=True)
        sensor.sleep(sleep=False)
        [pm2,pm1]=sensor.query()
        print("sensor de particulas encontrado")
        usb1=10
    except:
        print("Sensor de particulas no encontrado")
    usb1=usb1+1
ledindicador()
a=verficared.get_gps_position()
latitud=1
longitud=1
acc=""
lista=[]
f = open("/home/pi/Desktop/test.pickle", 'rb')
try:
    lista = pickle.load(f)
    for i in range(len(lista)):
        try:
            db.child("DatosPosicion").child("Usuario1").update(lista[0])
            lista.pop(0)
        except:
            print("no hay conexion a internet")
        time.sleep(2)
except:
    print("no hay datos guardado")
f.close()
f = open("test.pickle", 'wb')
pickle.dump(lista,f,protocol=pickle.HIGHEST_PROTOCOL)
f.close()
for i in range(1):
    time.sleep(5)
    led.value=True
    while (latitud==1 and longitud==1):
        time.sleep(0.5)
        [latitud,longitud,vel]=verficared.latlon()
    led.value=False
    latitud=round(latitud,5)
    longitud=round(longitud,5)
    vel=round(vel,2)
    sep_i=distancia.calculo("izquierda")
    sep_d=distancia.calculo("derecha")
    [pm2,pm1]=sensor.query()
    calidad=round(((adc.read(1))/4095*100),2)
    try:
        sonido=cola1.get(block=False,timeout=None)
        cola1.put(0)
    except:
        print("activar:error de cola en el for de sonido")
```

```
    try:
        acc=cola.get(block=False,timeout=None)
        cola.put(acc)
    except:
        print("activar:error de cola en el for de vibracion")
    lat=str(latitud).zfill(9)
    lon=str(longitud).zfill(9)
    sep_i1=str(sep_i).zfill(9)
    sep_d1=str(sep_d).zfill(9)
    pm21=str(pm2).zfill(9)
    pm11=str(pm1).zfill(9)
    ca1=str(calidad).zfill(9)
    so1=str(round(sonido,2)).zfill(9)
    ac=str(acc).zfill(9)
    vel1=str(vel).zfill(9)
    lat2=latitud
    lon2=longitud
```

Para iniciar la captura y subida de datos se ejecuta el siguiente código. Se realiza el censado con cada sensor y al final se suben los datos capturados ese instante, que posteriormente serán visualizados en la aplicación móvil o web.

```
#########################################INICIAR SISTEMA
    ###################################################
boton="iniciar"
rutaactual=""
while boton != "salir":
    if boton=="iniciar":
        sensor.sleep(sleep=False)
        ruta=numeroruta()
        u=True
        while u:
            try:
                grabar_video(ruta)
                u=False
            except:
                print("problemas con la camara")
                time.sleep(2)
        time.sleep(2)
        lista=[]
        horai=str(time.strftime("%H:%M:%S")).zfill(9)
        c=time.localtime(time.time())
        inicio=c[3]*3600+c[4]*60+c[5]
        now=time.localtime(time.time())
        resta=now[3]*3600+now[4]*60+now[5]-inicio
        while (resta<3600):
            try:
                [latitud,longitud,vel]=verficared.latlon()
                if latitud==1 and longitud==1:
                    latitud=lat2
                    longitud=lon2
                    a=verficared.get_gps_position()

                latitud=round(latitud,5)
                longitud=round(longitud,5)
```

```
                vel=round(vel,2)
            except:
                print("error: por sms o por el gps")
                latitud=lat2
                longitud=lon2
                vel=0
            time.sleep(0.025)
            [pm2,pm1]=sensor.query()
            time.sleep(0.025)
            try:
                sep_i=cola3.get(block=False,timeout=None)
                cola3.put(sep_i)
            except:
                cola3.put(100)
            time.sleep(0.025)
            try:
                sep_d=cola4.get(block=False,timeout=None)
                cola4.put(sep_d)
            except:
                cola4.put(100)
            ca=round(((adc.read(1))/4095*100),2)
            time.sleep(0.025)
            try:
                so=cola1.get(block=False,timeout=None)
                cola1.put(0)
            except:
                so=0
            time.sleep(0.025)
            try:
                acc=cola.get(block=False,timeout=None)
                cola.put(acc)
            except:
                acc=0
            time.sleep(0.025)
            camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            time.sleep(0.025)
            lat2=latitud
            lon2=longitud
            latitud=str(latitud).zfill(9)
            longitud=str(longitud).zfill(9)
            pm2=str(pm2).zfill(9)
            pm1=str(pm1).zfill(9)
            sep_i=str(sep_i).zfill(9)
            sep_d=str(sep_d).zfill(9)
            ca=str(ca).zfill(9)
            so=str(round(so,2)).zfill(9)
            acc=str(acc).zfill(9)
            vel=str(vel).zfill(9)
            i=ina.current()
            horatr=str(time.strftime("%H:%M:%S")).zfill(9)
            ledindicador()
            data={"Calidad":ca,"Sonido":so,"Latitud":latitud,"Longitud":longitud,
                "Sep_i":sep_i,"Sep_d":sep_d,"pm25":pm2,"pm10":pm1,"Acelerometro":acc,"
    Velocidad":vel}
            ejecutarConTiempoLimite(funcionA,(data,), 1)
```

```
            if ruta != rutaactual:
                lat1=lat2
                lon1=lon2
                lat=str(latitud).zfill(9)
                lon=str(longitud).zfill(9)
                pm21=str(pm2).zfill(9)
                pm11=str(pm1).zfill(9)
                sep_i1=str(sep_i).zfill(9)
                sep_d1=str(sep_d).zfill(9)
                ca1=str(ca).zfill(9)
                so1=str(so).zfill(9)
                ac=str(acc).zfill(9)
                vel1=str(vel).zfill(9)
                i1=str(i)
                horatr1=horatr
```

Por último, cuando se presiona el botón, el sistema comienza el proceso de apagado. En las siguientes líneas de código se muestra el proceso que se sigue para subir los datos, guardarlos en la memoria interna de la Raspberry Pi en caso de que no haya conexión a Internet y se espera un intervalo de tiempo hasta que usuario ingrese una *flash memory* para transferir el vídeo.

```
boton="iniciar"
rutaactual=""
while boton != "salir":
    if boton=="iniciar":
        sensor.sleep(sleep=False)
        ruta=numeroruta()
        u=True
        while u:
            try:
                grabar_video(ruta)
                u=False
            except:
                print("problemas con la camara")
                time.sleep(2)
        time.sleep(2)
        lista=[]
        horai=str(time.strftime("%H:%M:%S")).zfill(9)
        c=time.localtime(time.time())
        inicio=c[3]*3600+c[4]*60+c[5]
        now=time.localtime(time.time())
        resta=now[3]*3600+now[4]*60+now[5]-inicio
        while (resta<3600):
            try:
                [latitud,longitud,vel]=verficared.latlon()
                if latitud==1 and longitud==1:
                    latitud=lat2
                    longitud=lon2
                    a=verficared.get_gps_position()

                latitud=round(latitud,5)
                longitud=round(longitud,5)
                vel=round(vel,2)
            except:
                print("error: por sms o por el gps")
```

```
                latitud=lat2
                longitud=lon2
                vel=0
            time.sleep(0.025)
            [pm2,pm1]=sensor.query()
            time.sleep(0.025)
            try:
                sep_i=cola3.get(block=False,timeout=None)
                cola3.put(sep_i)
            except:
                cola3.put(100)
            time.sleep(0.025)
            try:
                sep_d=cola4.get(block=False,timeout=None)
                cola4.put(sep_d)
            except:
                cola4.put(100)
            ca=round(((adc.read(1))/4095*100),2)
            time.sleep(0.025)
            try:
                so=cola1.get(block=False,timeout=None)
                cola1.put(0)
            except:
                so=0
            time.sleep(0.025)
            try:
                acc=cola.get(block=False,timeout=None)
                cola.put(acc)
            except:
                acc=0
            time.sleep(0.025)
            camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            time.sleep(0.025)
            lat2=latitud
            lon2=longitud
            latitud=str(latitud).zfill(9)
            longitud=str(longitud).zfill(9)
            pm2=str(pm2).zfill(9)
            pm1=str(pm1).zfill(9)
            sep_i=str(sep_i).zfill(9)
            sep_d=str(sep_d).zfill(9)
            ca=str(ca).zfill(9)
            so=str(round(so,2)).zfill(9)
            acc=str(acc).zfill(9)
            vel=str(vel).zfill(9)
            i=ina.current()
            horatr=str(time.strftime("%H:%M:%S")).zfill(9)
            ledindicador()
            data={"Calidad":ca,"Sonido":so,"Latitud":latitud,"Longitud":longitud,
                "Sep_i":sep_i,"Sep_d":sep_d,"pm25":pm2,"pm10":pm1,"Acelerometro":acc,"
    Velocidad":vel}
            ejecutarConTiempoLimite(funcionA,(data,), 1)
            if ruta != rutaactual:
                lat1=lat2
                lon1=lon2
```

```
        lat=str(latitud).zfill(9)
        lon=str(longitud).zfill(9)
        pm21=str(pm2).zfill(9)
        pm11=str(pm1).zfill(9)
        sep_i1=str(sep_i).zfill(9)
        sep_d1=str(sep_d).zfill(9)
        ca1=str(ca).zfill(9)
        so1=str(so).zfill(9)
        ac=str(acc).zfill(9)
        vel1=str(vel).zfill(9)
        i1=str(i)
        horatr1=horatr
        rutaactual=ruta
        segnuevo=now[3]*3600+now[4]*60+now[5]
        segnuevo1=segnuevo
    else:
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
        c = 2 * atan2(sqrt(a), sqrt(1 - a))
        distance = round(R*c*1000,2)
        now=time.localtime(time.time())
        segnuevo=now[3]*3600+now[4]*60+now[5]
        segunuevo1=segnuevo
        if (distance < 3):
            segnuevo1=segnuevo2
            latitud=lat3
            longitud=lon3
            latitud=str(lat3).zfill(9)
            longitud=str(lon3).zfill(9)
        if (segnuevo-segnuevo2<150):
            lat=lat+","+latitud
            lon=lon+","+longitud
            pm21=pm21+","+pm2
            pm11=pm11+","+pm1
            sep_i1=sep_i1+","+sep_i
            sep_d1=sep_d1+","+sep_d
            ca1=ca1+","+ca
            so1=so1+","+so
            ac=ac+","+acc
            i1=i1+","+str(i)
            vel1=vel1+","+vel
            lon1=lon2
            lat1=lat2
            horatr1=horatr1+","+horatr
    lat3=latitud
    lon3=longitud
    segnuevo2=segnuevo1
    resta=segnuevo-inicio
    if resta>=3600:
        alto_video()
        led.value=True
    a=button.value
    if a:
        led.value = True
```

```
                resta=3600
                alto_video()
#####################APAGAR SISTEMA#########################################################
        horaf=str(time.strftime("%H:%M:%S")).zfill(9)
        sensor.sleep()
        data={"Calidad":ca1,"Sonido":so1,"Latitud":lat,"Longitud":lon,"Sep_i":sep_i1,
            "Sep_d":sep_d1,"pm25":pm21,"pm10":pm11,"Acelerometro":ac,"Hora inicial":
    horai,
            "Hora final":horaf,"Fecha":fecha,"Velocidad":vel1,"tiempo":horatr1}
        datac={("Ruta"+str(ruta)):data}          lista.append(datac)
        time.sleep(2)
        response = os.system("ping -c 1 www.google.com")
        if response == 0:
            db.child("DatosPosicion").child("Usuario1").update(datac)
        else:

            f = open("/home/pi/Desktop/test.pickle", 'wb')
            time.sleep(0.5)
            pickle.dump(lista,f,protocol=pickle.HIGHEST_PROTOCOL)
            time.sleep(2)
            f.close()
        try:
            db.child("RealTime").child("Usuario1").remove()
        except:
            print("nose pudo borrar la ruta en tiempo real")
        ledindicador()
        led.value=True
        archivo=open("/home/pi/Desktop/"+str(ruta)+".txt","w")
        archivo.write(i1+os.linesep)
        archivo.close()
        time.sleep(2)
        led.value = False
        espera=0
        while espera<10:
            espera=espera+1
            time.sleep(1)
            boton=""
            a=button.value
            if a:
                boton="salir"
                espera=120
        boton="salir"
usb.usb()
#############################APAGADO DE LA RASPBERRY#################
for i in range(1,20):
    led.value = True
    time.sleep(0.25)
    led.value = False
    time.sleep(0.25)
```

D.1.1. Función para activar la red móvil

```
import os
```

```
def red():
    os.system("qmicli -d /dev/cdc-wdm0 --dms-set-operating-mode='online'")
#verificacion de que la radio este encendida
    os.system('qmicli -d /dev/cdc-wdm0 --dms-get-operating-mode')
    os.system('qmicli -d /dev/cdc-wdm0 --nas-get-signal-strength')
#devuelve la ID de red LTE si el dispositivo se conect   correctamente.
    os.system('qmicli -d /dev/cdc-wdm0 --nas-get-home-network')
#esto confirma el nombre de la interfaz de red, t  picamente wwan0
    os.system('qmicli -d /dev/cdc-wdm0 -w')
#cambia a wwan0
    os.system('ip link set wwan0 down')
    os.system("echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip")
    os.system('ip link set wwan0 up')
#Iniciando la red movil
    try:
        string = "qmicli -p -d /dev/cdc-wdm0 --device-open-net='net-raw-ip|net-no-qos-
    header' --wds-start-network="
+"apn='internet.tuenti.ec',username='tuenti',password='tuenti',ip-type=4"+" --client
    -no-release-cid"
        os.system(string)
#configurando la ip
        os.system('udhcpc -i wwan0 -n')
#verificar la ip
        os.system('ip r s')
    except:
        print("no se pudo conectar a la red de movil")
```

D.1.2. Función para obtener datos desde el GPS

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import RPi.GPIO as GPIO

import serial
import time

ser = serial.Serial('/dev/ttyS0',115200)
ser.flushInput()
rec_buff = ''
rec_buff2 = ''
time_count = 0
coordenadas=''
coordenadas1=''
def send_at(command,back,timeout):
    rec_buff = ''
    ser.write((command+'\r\n').encode())
    time.sleep(timeout)
    if ser.inWaiting():
        time.sleep(0.01)
        rec_buff = ser.read(ser.inWaiting())
        print(rec_buff.decode())
    if rec_buff != '':
        if back not in rec_buff.decode():
```

```
            #print(command + ' ERROR')
            #print(command + ' back:\t' + rec_buff.decode())
            return 0,rec_buff
        else:
            return 1,rec_buff
    else:
        print('GPS is not ready')
        return 0,rec_buff

def get_gps_position():
    rec_null = True
    answer = 0
    print('Start GPS session...')
    rec_buff = ''
    a,rec_buff2=send_at('AT+CGPS=1,1','OK',1)
    return a
    time.sleep(2)
def latlon():
    lat=0
    lon=0
    vel=0
    rec_null=True
    rec_buff=''
    while rec_null:
        answer,rec_buff2 = send_at('AT+CGPSINFO','+CGPSINFO: ',1)
        if 1 == answer:
            answer = 0
            if ',,,,,,,,' in rec_buff2.decode().split():
                print('GPS is not ready')
                rec_null = False
                time.sleep(1)
                return 1,1,1
            else:
                coordenadas=rec_buff2.decode().split()
                coordenadas1=(coordenadas[2].split(','))
                lath=float(coordenadas1[0][0:2])
                latm=float(coordenadas1[0][2:])
                lat=-lath-latm/60
                lonh=float(coordenadas1[2][0:3])
                lonm=float(coordenadas1[2][3:])
                lon=-lonh-lonm/60
                vel=float(coordenadas1[7])
                vel=vel*1.852
                #print(lat)
                #print(lon)
                #print(rec_buff.decode())
                return lat,lon,vel
        else:
            print('error %d'%answer)
            rec_buff = ''
            send_at('AT+CGPS=1','OK',1)
            return 1,1,1
        time.sleep(1.5)
```

D.1.3. Función para transferencia de vídeo

```
import sys
import os
def usb():
    pathvideos='/home/pi/Desktop/videos'
    try:
        nombreusb=os.listdir('/media/pi')
        try:
            videos=os.listdir(pathvideos)
            videos=sorted(videos)
            print(videos)
            for i in range(len(videos)):
                print(i)
                os.system('mv /home/pi/Desktop/videos/'+str(videos[i])+' /media/pi/'
                          +str(nombreusb[0]))
        except:
            print("no hay videos")
 ##  os.system('mv /home/eduardo/Escritorio/tesis/videos/ /media/eduardo/nombreusb
    [0]')
    except:
        print("no usb")

    sizefile=0
    try:
        videos=os.listdir(pathvideos)
        videos=sorted(videos)
        for i in range(len(videos)):
            sizefile= sizefile + os.path.getsize(pathvideos+'/'+str(videos[i]))
    except:
        print("no hay videos")
```

D.1.4. Función para obtener datos del sensor de distancia

```
import board
import time
import digitalio

def calculo(opcion):
    if opcion == "izquierda":
        triger = digitalio.DigitalInOut(board.D23)
        triger.direction = digitalio.Direction.OUTPUT
        ech = digitalio.DigitalInOut(board.D24)
        ech.direction = digitalio.Direction.INPUT
    else:
        triger = digitalio.DigitalInOut(board.D19)
        triger.direction = digitalio.Direction.OUTPUT
        ech = digitalio.DigitalInOut(board.D26)
        ech.direction = digitalio.Direction.INPUT
    #GPIO.setmode(GPIO.BCM)
    velocidad=3840
    #GPIO.setup(TRIG,GPIO.OUT)
    #GPIO.setup(ECHO,GPIO.IN)
```

```
    triger.value = 0
    time.sleep(0.1)
    triger.value = 1
    time.sleep(0.00001)
    triger.value = 0

    while ech.value == 0:
            start=time.time()
```

Anexo E

Desarrollo de la aplicación móvil

E.1. Ventana de inicio

Dentro de esta ventana se codificará toda la lógica para el inicio de sesión con el usuario y contraseña propios de un usuario registrado, y la lógica de direccionamiento hacia la página de selección o la página de registro de usuarios en caso no constar en la base de datos de usuarios.

- **Lógica de la ventana**

```
package com.example.myapplication;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
    private String iduser;
    //atuadores de la aplicacion
```

```
Button Registrarcuenta;
Button IniciarSesion;
private EditText User;
private EditText Password;
//Integracion del servicio de autenticacion de Firebase
private FirebaseAuth firebaseAuth;
// Acceso a la base de datos
private DatabaseReference db;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    db = FirebaseDatabase.getInstance().getReference("DatosUsuarios");
    setContentView(R.layout.activity_main);
    firebaseAuth = FirebaseAuth.getInstance();
    User = (EditText) findViewById(R.id.Usuario);
    Password = (EditText) findViewById(R.id.Contrase a);
    //Adjuntar boton de escucha
    Registrarcuenta=findViewById(R.id.Registrar);
    Registrarcuenta.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view){
            startActivity(new Intent(MainActivity.this,Ventana_de_Registro.
class));
        }
    });
    IniciarSesion=findViewById(R.id.Login);
    IniciarSesion.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view){
            Start_sesion();
        }
    });
}
private void Start_sesion(){
    //Obtenemos el email y la contrase a desde las cajas de texto
    final String user = User.getText().toString().trim();
    String password  = Password.getText().toString().trim();
    //Verificamos que las cajas de texto no esten vacías
    if(TextUtils.isEmpty(user)){
        Toast.makeText(this,"Ingrese el Correo",Toast.LENGTH_LONG).show();
        return;
    }
    if(TextUtils.isEmpty(password)){
        Toast.makeText(this,"Ingrese la Contrase a",Toast.LENGTH_LONG).show
();
        return;
    }
    //Logear usuario
    firebaseAuth.signInWithEmailAndPassword(user, password)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
 {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    //checking if success
                    if(task.isSuccessful()){
```

```
                        db.addListenerForSingleValueEvent(new
    ValueEventListener() {
                            @Override
                            //Autenticacion
                            public void onDataChange(DataSnapshot
    dataSnapshot) {
                                for(DataSnapshot snapshot: dataSnapshot.
    getChildren()){
                                    String mail = snapshot.child("correo").
    getValue().toString();
                                    if(mail.contains(user)){
                                        iduser=snapshot.child("id").getValue
    ().toString();
                                        String nombre=snapshot.child("nombre"
    ).getValue().toString();
                                        Toast.makeText(MainActivity.this,"
    Bienvenido "+nombre+", hoy una nueva ruta te espera",Toast.LENGTH_LONG).show
    ();
                                    }
                                }
                                //Mantiene el id del usuario para
                                // limitar la busqueda de rutas
                                // solo a las que este haya generado
                                Intent miIntent=new Intent(MainActivity.this,
    menu.class);
                                Bundle miBundle=new Bundle();
                                miBundle.putString("Usuario",iduser);
                                miIntent.putExtras(miBundle);
                                startActivity(miIntent);
                            }
                            @Override
                            public void onCancelled(@NonNull DatabaseError
    databaseError) {

                            }
                        });
                    }else{
                        Toast.makeText(MainActivity.this,"Verificar Usuario o
     contrase a ",Toast.LENGTH_LONG).show();
                    }
                }
            });
    }
```

- **Código de la interfaz**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:srcCompat="@drawable/fig1"
        android:scaleType="centerCrop" />

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:background="#d8000000"
        android:gravity="center"
        android:layout_height="match_parent">

        <ImageView
            android:layout_width="110dp"
            android:layout_height="110dp"
            android:layout_marginBottom="70dp"
            app:srcCompat="@drawable/fig2"
            android:padding="10dp" />

        <EditText
            android:id="@+id/Usuario"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Correo"
            android:layout_marginRight="20dp"
            android:layout_marginLeft="20dp"
            android:layout_marginBottom="15dp"
            android:paddingLeft="20dp"
            android:paddingRight="10dp"
            android:paddingBottom="10dp"
            android:paddingTop="10dp"
            android:background="@drawable/btn_rdn_sin_color"
            android:textColorHint="#ccc"
            android:textColor="#FFF"
            />

        <EditText
            android:id="@+id/Contrase a"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Contrase a"
            android:layout_marginRight="20dp"
            android:layout_marginLeft="20dp"
            android:layout_marginBottom="15dp"
            android:paddingLeft="20dp"
            android:paddingRight="10dp"
            android:paddingBottom="10dp"
            android:paddingTop="10dp"
            android:background="@drawable/btn_rdn_sin_color"
            android:textColor="#FFF"
            android:textColorHint="#ccc" />

        <Button
            android:id="@+id/Login"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="20dp"
            android:layout_marginLeft="20dp"
            android:background="@drawable/btn_redondeado"
            android:textColor="#403f3f"
            android:layout_marginBottom="50dp"
            android:textSize="16dp"
            android:text="Iniciar Sesion"/>

        <Button
            android:id="@+id/Registrar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="60dp"
            android:layout_marginLeft="60dp"
            android:background="@drawable/btn_redondeado"
            android:textColor="#403f3f"
            android:layout_marginBottom="30dp"
            android:textSize="16dp"
            android:text="Crear cuenta"/>



    </LinearLayout>


</RelativeLayout>
```

E.2. Ventana de registro

En esta venta un usuario nuevo podrá crear una cuenta mediante el registro del correo, la contraseña y algunos datos personales.

- **Lógica de la ventana**

```
package com.example.myapplication;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthUserCollisionException;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

public class Ventana_de_Registro extends AppCompatActivity {

    private String lastuser;
    //actuadores
    EditText Name;
    EditText Lastname;
    EditText DNI;
    EditText Email;
    EditText Phone;
    EditText Password;
    Button btnRegistrar;

    //Declaramos un objeto firebaseAuth
    private FirebaseAuth firebaseAuth;
    private DatabaseReference db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ventana_de__registro);
        //inicializamos el objeto firebaseAuth
        firebaseAuth = FirebaseAuth.getInstance();
        db = FirebaseDatabase.getInstance().getReference("DatosUsuarios");
        //Referenciamos los views
        Name = (EditText) findViewById(R.id.Nombres);
        Lastname = (EditText) findViewById(R.id.Apellidos);
        DNI = (EditText) findViewById(R.id.CI);
        Email = (EditText) findViewById(R.id.Correo);
        Phone = (EditText) findViewById(R.id.Telefono);
        Password = (EditText) findViewById(R.id.Contrase a);
        btnRegistrar = (Button) findViewById(R.id.Registrar);
        //attaching listener to button
        btnRegistrar.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                registrarUsuario();
            }
        });
    }

    private void registrarUsuario(){
        //Obtenemos el email y la contrase a desde las cajas de texto
        final String name= Name.getText().toString();
        final String lastname= Lastname.getText().toString();
        final String ci= DNI.getText().toString();
        final String email = Email.getText().toString().trim();
        final String phone= Phone.getText().toString();
        final String password  = Password.getText().toString().trim();
```

```
    //Verificamos que las cajas de texto no esten vacías
    if(TextUtils.isEmpty(password) || TextUtils.isEmpty(name) || TextUtils.
isEmpty(lastname) || TextUtils.isEmpty(ci) || TextUtils.isEmpty(phone) ||
TextUtils.isEmpty(email)){
        Toast.makeText(this,"existe campos vacios",Toast.LENGTH_LONG).show();
        return;
    }else {
        //crea un nuevo usuario
        firebaseAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<
AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        //checking if success
                        if (task.isSuccessful()) {
                            startActivity(new Intent(Ventana_de_Registro.this
, MainActivity.class));
                            db.addListenerForSingleValueEvent(new
ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                    for(DataSnapshot snapshot: dataSnapshot.
getChildren()){
                                        lastuser= snapshot.child("id").
getValue().toString();
                                    }
                                    int nuser=Integer.parseInt(lastuser.
substring(7))+1;
                                    String id="Usuario"+String.valueOf(nuser)
;
                                    //Almacenamiento de informacion personalS
                                    DatosUsuario RegDatos = new DatosUsuario(
id,name, lastname, ci, email, phone);
                                    db.child(id).setValue(RegDatos);
                                    Toast.makeText(Ventana_de_Registro.this,"
Usuario Registrado: " + Name.getText(), Toast.LENGTH_LONG).show();
                                }
                                @Override
                                public void onCancelled(@NonNull
DatabaseError databaseError) {
                                }
                            });
                        } else {
                            if (task.getException() instanceof
FirebaseAuthUserCollisionException) {
                                Toast.makeText(Ventana_de_Registro.this, "Ya
existe un usuario registrado con este correo", Toast.LENGTH_LONG).show();
                                startActivity(new Intent(Ventana_de_Registro.
this, MainActivity.class));
                            } else {
                                Toast.makeText(Ventana_de_Registro.this, "No
se pudo registrar el usuario ", Toast.LENGTH_LONG).show();
                                startActivity(new Intent(Ventana_de_Registro.
```

```
    this, MainActivity.class));
```

- **Código de la interfaz**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Ventana_de_Registro">


    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:srcCompat="@drawable/fig1"
        android:scaleType="centerCrop" />

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:background="#d8000000"
        android:gravity="center"
        android:layout_height="match_parent">


        <EditText
            android:id="@+id/Nombres"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Nombres"
            android:layout_marginRight="20dp"
            android:layout_marginLeft="20dp"
            android:layout_marginBottom="15dp"
            android:paddingLeft="20dp"
            android:paddingRight="10dp"
            android:paddingBottom="10dp"
            android:paddingTop="10dp"
            android:background="@drawable/btn_rdn_sin_color"
            android:textColorHint="#ccc"
            android:textColor="#6200EE"
            />

        <EditText
            android:id="@+id/Apellidos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Apellidos"
            android:layout_marginRight="20dp"
            android:layout_marginLeft="20dp"
            android:layout_marginBottom="15dp"
            android:paddingLeft="20dp"
            android:paddingRight="10dp"
            android:paddingBottom="10dp"
```

```
        android:paddingTop="10dp"
        android:background="@drawable/btn_rdn_sin_color"
        android:textColorHint="#ccc"
        android:textColor="#6200EE"
        />

    <EditText
        android:id="@+id/CI"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="CI"
        android:layout_marginRight="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginBottom="15dp"
        android:paddingLeft="20dp"
        android:paddingRight="10dp"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:background="@drawable/btn_rdn_sin_color"
        android:textColorHint="#ccc"
        android:textColor="#6200EE"
        />

    <EditText
        android:id="@+id/Correo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Correo electronico"
        android:layout_marginRight="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginBottom="15dp"
        android:paddingLeft="20dp"
        android:paddingRight="10dp"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:background="@drawable/btn_rdn_sin_color"
        android:textColorHint="#ccc"
        android:textColor="#6200EE"
        />

    <EditText
        android:id="@+id/Telefono"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Telefono"
        android:layout_marginRight="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginBottom="15dp"
        android:paddingLeft="20dp"
        android:paddingRight="10dp"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:background="@drawable/btn_rdn_sin_color"
        android:textColorHint="#ccc"
        android:textColor="#6200EE"
```

```
            />

        <EditText
            android:id="@+id/Contrase a"
```

E.3. Ventana de selección

En esta sección se muestra el correspondiente que permite al usuario seleccionar entre la visualización de los nodos activos en tiempo real y la revisión de las rutas.

- **Lógica de ventana**

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class menu extends AppCompatActivity {
    Button BuscarR;
    Button BuscarC;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
        BuscarR=findViewById(R.id.Opcion1);
        BuscarC=findViewById(R.id.Opcion2);
        //restriccion de busqueda por usuario
        Bundle miBundle=this.getIntent().getExtras();
        final String iduser=miBundle.getString("Usuario");
        //seleccion de rutas
        BuscarR.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                Intent miIntent=new Intent(menu.this,BuscarRutas.class);
                Bundle miBundle=new Bundle();
                miBundle.putString("Usuario",iduser);
                miIntent.putExtras(miBundle);
                startActivity(miIntent);
            }
        });
        BuscarC.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                startActivity(new Intent(menu.this,MapsActivity.class));
            }
        });
```

```
    }
}
```

- **Código de la interfaz**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".menu">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:srcCompat="@drawable/fig1"
        android:scaleType="centerCrop" />

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:background="#d8000000"
        android:gravity="center"
        android:layout_height="match_parent">

        <Button
            android:id="@+id/Opcion1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="60dp"
            android:layout_marginLeft="60dp"
            android:background="@drawable/btn_redondeado"
            android:textColor="#403f3f"
            android:layout_marginBottom="30dp"
            android:textSize="16dp"
            android:text="Buscar Ruta"/>

        <Button
            android:id="@+id/Opcion2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="60dp"
            android:layout_marginLeft="60dp"
            android:background="@drawable/btn_redondeado"
            android:textColor="#403f3f"
            android:layout_marginBottom="30dp"
            android:textSize="16dp"
            android:text="Buscar Compa eros"/>



    </LinearLayout>
```

```
</RelativeLayout>
```

E.4. Ventana de monitorización de datos en tiempo real

La ventana de monitorización en tiempo real contiene la lógica necesaria para visualizar todos los usuarios activos en tiempo real y los valores de los diferentes parámetros que cada uno de ellos esta registrando.

- **Lógica de ventana**

```
package com.example.myapplication;

import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentActivity;

import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.util.Log;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.Circle;
import com.google.android.gms.maps.model.CircleOptions;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.maps.android.ui.IconGenerator;
import com.google.maps.android.ui.IconGenerator;

import org.w3c.dom.Text;

import java.util.ArrayList;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
    GoogleMap.OnMarkerClickListener {

    private GoogleMap mMap;
    private DatabaseReference mDatabase;
```

```
private ArrayList<Marker> tmpRealTimeMarkers = new ArrayList<>();
private ArrayList<Circle> realTimeMarkers = new ArrayList<>();
private String opc = "vacio";
private int cont = 0;
TextView cal, cal1, son, sepi, sepd, ace, cal3;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
    cal = (TextView) findViewById(R.id.Calidad);
    cal1 = (TextView) findViewById(R.id.Calidad2);
    son = (TextView) findViewById(R.id.Sonido);
    sepd = (TextView) findViewById(R.id.Sep_d);
    sepi = (TextView) findViewById(R.id.Sep_i);
    ace = (TextView) findViewById(R.id.Acelerometro);
    cal3 = (TextView) findViewById(R.id.Calidad3);
    mDatabase = FirebaseDatabase.getInstance().getReference("RealTime");
}

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    googleMap.setOnMarkerClickListener(this);
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(-2.900787, -
79.005612), 12));
    final IconGenerator iconFactory = new IconGenerator(this);
    //myRef1.child("nodo1").addValueEventListener(new ValueEventListener()
    mDatabase.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            for(Marker marker:tmpRealTimeMarkers){
                marker.remove();
            }
            tmpRealTimeMarkers.clear();
            for(Circle marker:realTimeMarkers){
                marker.remove();
            }
            realTimeMarkers.clear();
            if(opc=="vacio") {
                for(DataSnapshot snapshot: dataSnapshot.getChildren()){
                    MapsPojo mp = snapshot.getValue(MapsPojo.class);//aqui se
 obtiene la long y lat de datasnapshot, se guarda en el objeto mp
                    String nodo = snapshot.getKey();
                    double Latitud = Double.parseDouble(mp.getLatitud());
                    double Longitud = Double.parseDouble(mp.getLongitud());
                    System.out.println("aqui");
                    Marker marker2 = mMap.addMarker(new MarkerOptions()
```

```
                                .position(new LatLng(Latitud, Longitud))
                                .title(nodo)
                                .snippet("Cargando...")
                                .icon(BitmapDescriptorFactory.fromBitmap(
    iconFactory.makeIcon(nodo))));
                        iconFactory.setBackground(getResources().getDrawable(R.
    drawable.amu_bubble_shadow));

                        Circle circle = mMap.addCircle(new CircleOptions()
                                .center(new LatLng(Latitud, Longitud))
                                .radius(2)
                                .strokeColor(Color.RED)
                                .fillColor(Color.WHITE));
                        tmpRealTimeMarkers.add(marker2);
                        realTimeMarkers.add(circle);
                    }
                } else {
                    //FragmentLayout myLayout = findViewById(R.id.map);
                    //System.out.println(myLayout.getHeight());
                    //System.out.println(opc);
                    DataSnapshot snapshot1 = dataSnapshot.child(opc);
```

- **Código de la interfaz**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="10"
    android:id="@+id/mapa">

    <LinearLayout
        android:id="@+id/fila1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#3F51B5"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/Calidad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="PM2.5 :"
            android:textColor="#fff"/>

        <TextView
            android:id="@+id/Calidad3"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="PM10 :"
            android:textColor="#fff"/>

        <TextView
            android:id="@+id/Sonido"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="Sonido :"
            android:textColor="#fff" />

        <TextView
            android:id="@+id/Calidad2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="Calidad2 :"
            android:textColor="#fff"/>
    </LinearLayout>

    <fragment xmlns:map="http://schemas.android.com/apk/res-auto"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="8"
        tools:context=".MapsActivity" />

    <LinearLayout
        android:background="#3F51B5"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:id="@+id/fila2"
        android:textColor="#fff">

        <TextView
            android:id="@+id/Sep_d"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="Sep_d :"
            android:textColor="#fff"/>
        <TextView
```

```
            android:id="@+id/Acelerometro"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="Acelerometro :"
            android:textColor="#fff"/>

        <TextView
            android:id="@+id/Sep_i"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical|center_horizontal"
            android:layout_weight="1"
            android:gravity="center_vertical|center_horizontal"
            android:text="Sep_i :"
            android:textColor="#fff"/>
    </LinearLayout>

</LinearLayout>
```

E.5. Ventana de análisis de las rutas

En esta ventana se integrara la lógica que permita al usuario seleccionar la ruta que se desea analizar, donde se integra todos los parámetros capturados por lo nodos y son visualizados al recorrer la ruta usando un *slider*.

- **Lógica de ventana**

```
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PatternItem;
import com.google.android.gms.maps.model.Polygon;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.android.gms.maps.model.RoundCap;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

public class BuscarRutas extends FragmentActivity implements OnMapReadyCallback {

    private String idUsuario;
```

```
// integracion del API de backend
private GoogleMap mMap;
private DatabaseReference db;
// array para eliminar hitoria de marcadores
private ArrayList<Marker> tmpRealTimeMarkers = new ArrayList<>();
private ArrayList<Marker> RealTimeMarkers = new ArrayList<>();
//Variables de visualizacion
String getLat;
String getLng;
String getPm10;
String getPm25;
String getcal;
String getson;
String getsepd;
String getsepi;
String getacel;
//actuadores
private EditText IDRuta;
private SeekBar seekBar;
TextView cal,cal1,son,sepi,sepd,ace,cal3;
Button consultar;
Polyline polyline1 = null;
String Ruta;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_buscar_rutas);
    // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
    Bundle miBundle=this.getIntent().getExtras();

    //elazamiento de variebles del archivo xml
    idUsuario=miBundle.getString("Usuario");
    db = FirebaseDatabase.getInstance().getReference();
    consultar = (Button) findViewById(R.id.btnconsultar);
    IDRuta = (EditText) findViewById(R.id.IDRuta);
    seekBar=findViewById(R.id.seekBar);
    cal = (TextView) findViewById(R.id.Calidad);
    cal1 = (TextView) findViewById(R.id.Calidad2);
    son = (TextView) findViewById(R.id.Sonido);
    sepd = (TextView) findViewById(R.id.Sep_d);
    sepi = (TextView) findViewById(R.id.Sep_i);
    ace = (TextView) findViewById(R.id.Acelerometro);
    cal3 = (TextView) findViewById(R.id.Calidad3);

}
@Override
public void onMapReady(final GoogleMap googleMap ) {
    //inicializacion de arreglos
    final List<LatLng> points = new ArrayList<LatLng>();
```

```
        final List<Double>listaPm10=new ArrayList<>();
        final List<Double>listaPm25=new ArrayList<>();
        final List<Double>listaCal=new ArrayList<>();
        final List<Double>listaSon=new ArrayList<>();
        final List<Double>listaSepd=new ArrayList<>();
```

- **Código de la interfaz**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="10"
    android:id="@+id/mapa">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#3F51B5"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="0.5"
            android:background="#3F51B5"
            android:orientation="horizontal">
            <EditText
                android:id="@+id/IDRuta"
                android:layout_width="150dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="60dp"
                android:hint="Ruta#"
                android:paddingLeft="55dp"
                android:paddingRight="10dp"
                android:textSize="16dp"
                android:background="@drawable/btn_rdn_sin_color"
                android:textColorHint="#ccc"
                android:textColor="#FFF" />

            <Button
                android:id="@+id/btnconsultar"
                android:layout_width="match_parent"
                android:layout_marginRight="60dp"
                android:layout_height="wrap_content"
                android:background="@drawable/btn_redondeado"
                android:text="Consultar"
                android:textColor="#403f3f"
                android:textSize="16dp" />
        </LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#3F51B5"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/Calidad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_weight="1"
        android:gravity="center_vertical|center_horizontal"
        android:text="PM2.5 :"
        android:textColor="#fff"/>

    <TextView
        android:id="@+id/Calidad3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_weight="1"
        android:gravity="center_vertical|center_horizontal"
        android:text="PM10 :"
        android:textColor="#fff"/>

    <TextView
        android:id="@+id/Sonido"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_weight="1"
        android:gravity="center_vertical|center_horizontal"
        android:text="Sonido :"
        android:textColor="#fff" />

    <TextView
        android:id="@+id/Calidad2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_weight="1"
        android:gravity="center_vertical|center_horizontal"
        android:text="Calidad2 :"
        android:textColor="#fff"/>
</LinearLayout>
</LinearLayout>
<LinearLayout
    android:id="@+id/fila4"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="8"
    android:background="#3F51B5"
    android:orientation="horizontal">
```

```
    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MapsActivity" />
    </LinearLayout>>
```

E.6. Archivos complementarios

Finalmente, en esta sección se muestra todos los archivos complementarios que permitirán el correcto funcionamiento de cada una de las ventanas que integran la aplicación móvil.

- **DatosRutas.java**

```
package com.example.myapplication;

public class DatosRutas {

    private String latitud;
    private String longitud;

    public DatosRutas() {}

    public String getLatitud() {
        return latitud;
    }
    public String getLongitud() { return longitud; }




}
```

- **DatosUsuario.java**

```
package com.example.myapplication;

public class DatosUsuario {
    String id, name, lastname, ci, email, phone;

    public DatosUsuario(String id, String name, String lastname, String ci,
    String email, String phone) {
        this.id = id;
        this.name = name;
        this.lastname = lastname;
        this.ci = ci;
        this.email = email;
        this.phone = phone;
    }

    public String getId() {
```

```
        return id;
    }

    public String getNombre() {
        return name;
    }

    public String getApellido() {
        return lastname;
    }

    public String getCedula() {
        return ci;
    }

    public String getCorreo() {
        return email;
    }

    public String getTelefono() {
        return phone;
    }
}
```

- **MapsPojo.java**

```
package com.example.myapplication;

public class MapsPojo {
    private String Latitud;
    private String Longitud;
    private String Calidad;
    private String Sonido;
    private String Sep_i;
    private String Sep_d;
    private String Acelerometro;
    private String pm10;
    private String pm25;
    private String Velocidad;

    private MapsPojo() {}
    public MapsPojo(String Latitud, String Longitud, String Calidad
    ,String Sonido,String Sep_i,String Sep_d,String Acelerometro,String pm10,
    String pm25,String Velocidad) {
        //System.out.println(Latitud);
        //System.out.println(Double.parseDouble(Latitud));
        this.Latitud = (Latitud);
        this.Longitud = (Longitud);
        this.Calidad = (Calidad);
        this.Sonido = Sonido;
        this.Sep_d = (Sep_d);
        this.Sep_i = (Sep_i);
        this.Acelerometro = (Acelerometro);
        this.pm10 = (pm10);
        this.pm25 = (pm25);
```

```
        this.Velocidad = Velocidad;

    }
    public String getLatitud() {
        return Latitud;
    }
    public String getLongitud() {
        return Longitud;
    }
    public String getCalidad() {return Calidad; }
    public String getSonido() {return Sonido;}
    public String getSep_i() {return Sep_i;}
    public String getSep_d() {return Sep_d;}
    public String getAcelerometro() {return Acelerometro;}
    public String getpm10() {return pm10;}
    public String getpm25() {return pm25;}
    public String getVelocidad() {return Velocidad;}

}
```

Anexo F

Desarrollo de la aplicación Web

F.1. Página Principal

El código que se muestra en esta sección corresponde a la página de inicio de la aplicación web, donde se integra las funcionalidades con diferentes archivos de estilo *CSS* y archivos *JavaScript* para la interacción con los usurarios, en el código se puede identificar que se importa la biblioteca de *Boostrap*, la *API Maps* de *Google* y la comunicación con la plataforma *Firebase* que son los elementos fundamentales para solicitar los datos y visualizar la ubicación exacta del usuario dentro de la ciudad.

```
<html>

<head>
    <!-- bootstrap 4 -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/
    css/bootstrap.min.css"
        integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/
    Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <!-- Font awesome-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/
    4.7.0/css/font-awesome.min.css">
    <!--css-->
    <link rel="stylesheet" href="./css/main.css">
    <title>Tiempo Real</title>
    <!-- dimenciones del mapa en la pagina -->
    <style>
        #map {
            height: 90%;
        }

        html,
        body {
            height: 95%;
            margin: 0;
            padding: 0;
        }
```

```
    </style>
</head>

<body>
    <!--Encabezado-->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="index.html"></a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
    target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
    navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="/">
                        <i class="fa fa-home"></i>
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="DatosU.html">Rutas</a>
                </li>
            </ul>
        </div>
    </nav>

    <!-- Firebase-->
    <script src="https://www.gstatic.com/firebasejs/7.15.0/firebase-app.js"></script>
    <script src="https://www.gstatic.com/firebasejs/7.15.0/firebase-database.js"></
    script>
    <script>
        var firebaseConfig = {
            apiKey: "AIzaSyD4hf0doODmvPidtgYyWUB5CX4Rhi7yN6Y",
            authDomain: "bikenode-bc16b.firebaseapp.com",
            databaseURL: "https://bikenode-bc16b.firebaseio.com",
            projectId: "bikenode-bc16b",
            storageBucket: "bikenode-bc16b.appspot.com",
            messagingSenderId: "440815565465",
            appId: "1:440815565465:web:29178414e9fe35fb89d608",
            measurementId: "G-Z1K16TY2T3"
        };
        // Inicializacion de Firebase
        firebase.initializeApp(firebaseConfig);
    </script>

    <!--key para la intergacion del Mapa-->

    <div id="map"></div>
    <script async defer
        src="https://maps.googleapis.com/maps/api/js?key=
    AIzaSyAnO0fGL1Xvuk5bB_hCAvW7F3nOZpAYxEs&callback=initMap">
        </script>

    <!--Tabla de registro de valores en tiempo real-->
```

```
    <div class="container">
        <table class="table table-dark">
            <thead>
                <tr>
                    <th scope="col">Usuarios</th>
                    <th scope="col">Caidad del aire (%)</th>
                    <th scope="col">PM2.5 (ug/m^3)</th>
                    <th scope="col">PM10 (ug/m^3) </th>
                    <th scope="col">Nivel de ruido (dB)</th>
                    <th scope="col">Separacion izquierda (cm)</th>
                    <th scope="col">Separacion derecha (cm)</th>
                    <th scope="col">Estado de la vía</th>
                    <th scope="col">Velociad (km/h)</th>
                </tr>
            </thead>
            <tbody id='tabla'>
                <tr>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>
                    <td>0</td>

                </tr>
            </tbody>
        </table>
    </div>
    <!-- Integracion de archivos js (Interaccion con el usuario)-->
    <script src="./js/Realtime.js"></script>
</body>

</html>
```

F.1.1. Archivo js asociado

Dentro de este código se muestra los comandos requeridos para hacer una consulta de todos los usuarios activos en tiempo real y sus respectivas lecturas asociadas, del mismo modo, se muestra como integrar los datos consultados en la tabla de la página principal y el ícono que representa a un usuario activo.

```
// inicializacion de firebase
var db = firebase.database();
var ref = db.ref('RealTime');
var tabla=document.getElementById('tabla');
markers=[];
// inicializacion del mapa
function initMap() {
    var inicioderuta = { lat: -2.9025, lng: -79.0006484 };
    var map = new google.maps.Map(document.getElementById('map'), {
```

```
        zoom: 14.5,
        center: inicioderuta,
        gestureHandling: 'cooperative'
    });
    //consulta de datos en tiempo real
    ref.orderByValue().on("value", function(snapshot) {
        tabla.innerHTML='';
        deleteMarkers();
        //indexacion de los valores a la tabla de la pagina principal
        snapshot.forEach(function(data) {
            tabla.innerHTML+=`
            <td>${data.key}</td>
            <td>${parseFloat(data.val().Calidad)}</td>
            <td>${parseFloat(data.val().pm25)}</td>
            <td>${parseFloat(data.val().pm10)}</td>
            <td>${parseFloat(data.val().Sonido)}</td>
            <td>${parseFloat(data.val().Sep_i)}</td>
            <td>${parseFloat(data.val().Sep_d)}</td>
            <td>${parseFloat(data.val().Acelerometro)}</td>
            <td>${parseFloat(data.val().Velocidad)}</td>
            `
            //icono que representa al ciclista
            var imagen='https://maps.google.com/mapfiles/ms/icons/cycling.png';
            //ubica el usuario en el mapa
            var marker = new google.maps.Marker({
                position: {lat: parseFloat(data.val().Latitud), lng: parseFloat(data.
    val().Longitud)},
                map: map,
                title: (data.key).toString()+"\n"+"Latitud:"+data.val().Latitud+"\n"+"
    Longitud:"+data.val().Longitud,
                icon:imagen
            });
            markers.push(marker)
        });
    });
}

function deleteMarkers() {
    for (var i = 0; i < markers.length; i++) {
        markers[i].setMap(null);
      }
    markers = [];
  }
```

F.1.2. Archivo de estilos asociado

Este código será utilizado, tanto por la página principal como por la página secundaria, dado que el mismo integra los estilos correspondientes a partes del diseño de la aplicación web.

```
* {
    padding: 0;
    margin: 0;
}
```

```
main {
    width: 100vw;
    height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

form {
    width: 400px;
    border: solid 1px #EBEBEB;
    padding: 5rem;
}

button {
    margin-top: 2rem;
}
```

F.2. página de búsqueda de rutas

En esta sección, se presenta el código correspondiente a la página secundaria, donde se integra las mismas aplicaciones de la página principal dado que se requiere disponer de funcionalidades similares, como la consulta de datos y la monitorización de la ruta seguida, dentro de esta página también se integra la librería *morris.js* como se muestra en el código presentado, la misma que permitirá la creación de los gráficos de línea y de área de los datos consultados, se debe mencionar que el *tracking* de las rutas se desarrollara mediante el uso de polilíneas e igualmente se mostrará el punto de inicio y fin de la ruta con diferentes íconos. Para buscar el recorrido se integran dos entradas de texto y un botón, así como un **slider** el mismo que se usará para desplazar el ícono por cada punto donde se haya registrado lecturas.

```
<html>

<head>
    <!-- bootstrap 4 -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/
    css/bootstrap.min.css"
        integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/
    Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <!-- Font awesome-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/
    4.7.0/css/font-awesome.min.css">
    <!--css-->
    <link rel="stylesheet" href="./css/main.css">
    <title>Rutas Almacenadas</title>
        <!-- dimenciones del mapa en la pagina -->
    <style>
        #map {
            height: 90%;
        }
        html,
```

```
        body {
            height: 90%;
            margin: 0;
            padding: 0;
        }
    </style>
</head>

<body>
    <!—Encabezado—>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="index.html"></a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
    target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
    navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="/">
                        <i class="fa fa-home"></i>
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="DatosU.html">Rutas</a>
                </li>
            </ul>
        </div>
    </nav>

    <div class="containner">
        <table class="table table-dark">
            <tbody id='mensaje'>
                <tr>
                    <td>0</td>
                </tr>
            </tbody>
        </table>
    </div>
    <!—Entradas de texto y boton para la busqueda de usarios y rutas—>
    <div class="container">
        <input type="text" id="IDuser" placeholder="# de Usuario">
        <input type="text" id="IDruta" placeholder="# de Ruta">

        <button class="btn btn-success" id="mostrar" onclick="mostrarg()">Guardar</
    button>
    </div>

    <!— Morris Graficas—>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js"></
    script>
    <link rel="stylesheet" href="/libs/morris.css">
    <script src="/libs/morris.min.js"></script>
```

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/raphael/2.1.0/raphael-min.js">
</script>

<!-- Firebase-->
<script src="https://www.gstatic.com/firebasejs/7.15.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.15.0/firebase-database.js"></
script>
<script>
    var firebaseConfig = {
        apiKey: "AIzaSyD4hf0doODmvPidtgYyWUB5CX4Rhi7yN6Y",
        authDomain: "bikenode-bc16b.firebaseapp.com",
        databaseURL: "https://bikenode-bc16b.firebaseio.com",
        projectId: "bikenode-bc16b",
        storageBucket: "bikenode-bc16b.appspot.com",
        messagingSenderId: "440815565465",
        appId: "1:440815565465:web:29178414e9fe35fb89d608",
        measurementId: "G-Z1K16TY2T3"
    };
    // Inisializacion Firebase
    firebase.initializeApp(firebaseConfig);
</script>

<!--Mapa-->
<div id="map"></div>
<script src="./js/Mapa.js"> </script>
<script async defer
    src="https://maps.googleapis.com/maps/api/js?key=
AIzaSyAnO0fGL1Xvuk5bB_hCAvW7F3nOZpAYxEs&callback=initMap">
    </script>

<div class="slider">
    <input type="range" min="0" max="100" value="0" id="recorrer">
    <div id="selector">
        <div class="SelectBtn"></div>
        <div id="selectvalue"></div>
    </div>
</div>

<!--Tablas para contencion de los graficos-->
<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
                <th scope="row">Calidad del aire (Porcentaje respecto al aire puro
)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id='Cambiental'></div>

<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
```

```
                <th scope="row">Niveles de ruido (dB)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id="soni"></div>

<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
                <th scope="row">Particulas inferiores a 10 micras (ug/m^3)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id="pm10g"></div>

<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
                <th scope="row">Particulas inferiores a 2.5 micras (ug/m^3)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id="pm25g"></div>

<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
                <th scope="row">Separacion Izquierda y Derecha (cm)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id="sepid"></div>

<div class="container">
    <table class="table table-dark">
        <thead>
            <tr>
                <th scope="row">Estado de la vía (0 Muy bueno, 1 Bueno, 2 Regular,
 3 Malo, 4 Muy malo)</th>
            </tr>
        </thead>
    </table>
</div>
<div class="container" id="acele"></div>

<!— Integracion de archivos js (Interaccion con el usuario)—>
<script src="./js/Rutas.js"></script>
<script src="./js/ConsultaU.js"></script>
```

```
</body>

</html>
```

F.2.1. Archivos js asociados

Integración del mapa de inicio: Como su nombre de lo indica en esta sección de código se usa únicamente para visualizar el mapa al cambiar de ventana, mientras que en el resto de códigos se tendrá toda la lógica de interacción con el mismo.

```
function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 14.5,
        center: {lat: -2.9025, lng: -79.0006484},
        gestureHandling: 'cooperative'
    });

}
```

Consulta para mostrar el número de usuario que existen registrados: Por otro lado en este código se consultara el número de usuarios registrados, los mismos que se mostrarán en el mensaje informativo apenas ingresar a la página de rutas.

```
var db = firebase.database();
var ref = db.ref("DatosPosicion");
var tabla=document.getElementById('mensaje');
ref.orderByValue().limitToLast(1).on("value", function(snapshot) {
    tabla.innerHTML='';
    snapshot.forEach(function(data) {
        tabla.innerHTML+=`
        <td> Seleeciones una ruta de uno de los ${data.key.substring(7)} usuarios
    registrados</td>`
    });
});
```

Consulta de las rutas solicitadas: Este punto tendrá gran parte de la lógica de interacción de la página para la búsqueda de rutas, dado que la misma integra las consultas que se realizan a *Firebase* de todas las variables capturadas durante el desarrollo de una ruta, valores que luego son usados para crear en el mapa la polilínea que indica la trayectoria seguida por el usuario y generar las respectivas gráficas del comportamiento de las variables, así como también posicionar el ícono durante cualquier punto donde se tomó la muestra.

```
//inicializacion de arreglos
cal=[];
sep=[];
son=[];
dpm10=[];
dpm25=[];
latlng=[];
vel=[];
acel=[];
var map;
// Funcion para iniciar la busqueda del ususario y la ruta solicitada
```

```
function mostrarg() {
    cal=[null];
    sep=[null];
    son=[null];
    dpm10=[null];
    dpm25=[null];
    latlng=[null];
    vel=[null];
    acel=[null];
    //Captura los valores de las entradas de texto
    var IDuser = document.getElementById('IDuser').value;
    var IDruta = document.getElementById('IDruta').value;
    //inicializacion de firbase
    var db = firebase.database();
    var ref = db.ref("DatosPosicion");
    //consultas
    ref.orderByValue().limitToLast(1).on("value", function(snapshot) {
        snapshot.forEach(function(data) {
            var nuser=data.key.substring(7)
            if(IDuser<=nuser){
                var ref = db.ref("DatosPosicion/Usuario"+IDuser);
                ref.orderByValue().limitToLast(1).on("child_added", function(snapshot)
    {
                        var nrut=snapshot.key.substring(4)
                        if(parseFloat(IDruta)<=nrut){
                            Cambiental.innerHTML='';
                            sepid.innerHTML='';
                            soni.innerHTML='';
                            pm10g.innerHTML='';
                            pm25g.innerHTML='';
                            acele.innerHTML='';
                            firebase.database().ref('DatosPosicion/Usuario' + IDuser +
    "/Ruta" + IDruta).once('value').then(function (snapshot) {
                                //almacenamiento de los valores consultados
                                var velocidad=snapshot.val().Velocidad;
                                var longitud = snapshot.val().Longitud;
                                var latitud = snapshot.val().Latitud;
                                var calidad = snapshot.val().Calidad;
                                var sep_d = snapshot.val().Sep_d;
                                var sep_i = snapshot.val().Sep_i;
                                var sonido = snapshot.val().Sonido;
                                var pm10 = snapshot.val().pm10;
                                var pm25 = snapshot.val().pm25;
                                var hora=snapshot.val().tiempo;
                                var fecha=snapshot.val().Fecha;
                                var aceleracion=snapshot.val().Acelerometro;
                                var j = 0;
                                //separacion de los valore en arreglos
                                for (var i = 0; i < calidad.length; i = i + 10) {
                                    latlng[j]={lat: parseFloat(latitud.substring(i, i
    + 9)),lng: parseFloat(longitud.substring(i, i + 9))}
                                    cal[j] = {
                                        x:fecha+" "+hora.substring(i+1,i+9), value:
    calidad.substring(i, i + 9)
                                    };
```

```
                                sep[j] = {
                                    x: fecha+" "+hora.substring(i+1,i+9), value:
sep_d.substring(i, i + 9), value2: sep_i.substring(i, i + 9)
                                };
                                son[j] = {
                                    x: fecha+" "+hora.substring(i+1,i+9), value:
sonido.substring(i, i + 9)
                                };
                                dpm10[j] = {
                                    x: fecha+" "+hora.substring(i+1,i+9), value:
pm10.substring(i, i + 9)
                                };
                                dpm25[j] = {
                                    x: fecha+" "+hora.substring(i+1,i+9), value:
pm25.substring(i, i + 9)
                                };

                                acel[j] = {
                                    x: fecha+" "+hora.substring(i+1,i+9), value:
aceleracion.substring(i, i + 9)
                                };

                                vel[j]={
                                    value:velocidad.substring(i, i + 9)
                                };
                                j = j + 1;
                            }
                            //indexacion de los valores a las graficas de morris.
js
                            morris1.setData(cal);
                            morris2.setData(sep);
                            morris3.setData(acel);
                            morris4.setData(son);
                            morris5.setData(dpm10);
                            morris6.setData(dpm25);
                            //intergacion del mapa
                            map = new google.maps.Map(document.getElementById('map
'), {
                                zoom: 17,
                                center: latlng[0],
                                gestureHandling: 'cooperative'
                            });
                            //creacion de las polilineas para la ruta
                            var flightPath = new google.maps.Polyline({
                                path: latlng,
                                geodesic: true,
                                strokeColor: '#000080',
                                strokeOpacity: 1.0,
                                strokeWeight: 4
                            });
                            flightPath.setMap(map);
                            //marcado de inicio de ruta
                            var imagen='http://maps.google.com/mapfiles/ms/icons/
cycling.png';
                            var marker = new google.maps.Marker({
```

```
                                position: latlng[0],
                                title: "Inicio del recorrido",
                                icon:imagen
                            });
                            //marcador del fin de la ruta
                            marker.setMap(map);
                            markers.push(marker);
                            var imagen2='http://maps.google.com/mapfiles/ms/icons/
cycling.shadow.png';
                            var marker2 = new google.maps.Marker({
                                position: latlng[latlng.length-1],
                                title: "Final del recorrido",
                                icon:imagen2
                            });
                            marker2.setMap(map);
                            markers.push(marker2);
                        });
                    }
                    else{
                        alert("El ususario "+IDuser+" unicamente a generado "+nrut
+" rutas");
                    }
                });
        }
        else{
            alert("El usuario no existe");
        }
    });
});

//creacion de las graficas por parametro
var morris1 = new Morris.Area({
    element: 'Cambiental',
    xkey: 'x',
    ykeys: ['value'],
    labels: ['Calidad del aire'],
    resize: true,
    hideHover: true,
    pointSize: 1
});
var morris2 = new Morris.Line({
    element: 'sepid',
    xkey: 'x',
    ykeys: ['value','value2'],
    labels: ['Separacion Derecha','Separacion Izquierda'],
    resize: true,
    hideHover: true,
    pointSize: 1
});

var morris3 = new Morris.Area({
    element: 'acele',
    xkey: 'x',
    ykeys: ['value'],
    labels: ['Estado de la via'],
```

```
        resize: true,
        hideHover: true,
        pointSize: 1
    });

    var morris4 = new Morris.Area({
        element: 'soni',
        xkey: 'x',
        ykeys: ['value'],
        labels: ['Nivel de ruido (dB):'],
        resize: true,
        hideHover: true,
        pointSize: 1
    });

    var morris5 = new Morris.Area({
        element: 'pm10g',
        xkey: 'x',
        ykeys: ['value'],
        labels: ['Particulas menores a 10ppm'],
        resize: true,
        hideHover: true,
        pointSize: 1
    });
    var morris6 = new Morris.Area({
        element: 'pm25g',
        xkey: 'x',
        ykeys: ['value'],
        labels: ['Particulas menore a 25ppm'],
        resize: true,
        hideHover: true,
        pointSize: 1
    });
    //funcion para recorrer la ruta
    document.getElementById('recorrer').addEventListener('input',recorrerruta);

}
markers=[];
function recorrerruta() {
    //intergacion del tiempo en el slider
    let ruta=document.getElementById("recorrer").value;
    var slider=document.getElementById("recorrer");
    var selector=document.getElementById("selector");
    var selectvalue=document.getElementById("selectvalue");
    selectvalue.innerHTML=slider.value;
    slider.oninput=function(){
    selectvalue.innerHTML=(dpm25[ruta].x).substring(11);
    selector.style.left=Math.floor(this.value*100/latlng.length)+"%";
    }
    //salto entre los puntos de la ruta
    deleteMarkers();
    document.getElementById("recorrer").max = latlng.length-1;
    var imagen='https://maps.google.com/mapfiles/ms/icons/cycling.png';
    var marker = new google.maps.Marker({
        position: latlng[ruta],
```

```
        title: "Velocidad (Km/h): "+parseFloat(vel[ruta].value)+"\n" +
        "Calidad aire (%): "+parseFloat(cal[ruta].value)+"\n"+
        "Pm10 (ug/m^3): "+parseFloat(dpm10[ruta].value)+"\n"+
        "Pm25 (ug/m^3)): "+parseFloat(dpm25[ruta].value)+"\n"+
        "Ruido (dB): "+parseFloat(son[ruta].value)+"\n"+
        "Sep_d (cm): "+parseFloat(sep[ruta].value)+"\n"+
        "Sep_i (cm): "+parseFloat(sep[ruta].value2)+"\n"+
        "Estado de la vía: "+parseFloat(acel[ruta].value)+"\n",
        icon:imagen
    });
    marker.setMap(map)
    markers.push(marker)
    var imagen2='https://maps.google.com/mapfiles/ms/icons/cycling.shadow.png';
    var marker2 = new google.maps.Marker({
        position: latlng[latlng.length-1],
        title: "final",
        icon:imagen2
    });
    marker2.setMap(map)
    markers.push(marker2);
}


function deleteMarkers() {
    for (var i = 0; i < markers.length; i++) {
        markers[i].setMap(null);
      }
    markers = [];
  }
```

F.2.2. Archivo de estilos asociado

En este caso el archivo se usa para agregar los estilos asociados al *slider*, el mismo que permitirá recorrer la ruta y posicionar al usuario sobre cualquier punto donde exista una muestra.

```
body{
    background: #0F2027;  /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027);  /*
    Chrome 10-25, Safari 5.1-6 */
    background: linear-gradient(to right, #2C5364, #203A43, #0F2027); /* W3C, IE 10+/
    Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
            font-family: 'Spicy Rice', cursive;
}

.slider{
    position: relative;
    margin: 3%;
    cursor: pointer;
    width:94%;
}

#recorrer{
    -webkit-appearance: none;
    width: 100%;
```

```
    height: 10px;
    outline: none;
    border-radius: 10px;

}
```

]

Anexo G

Costo de la implementación de la estación móvil

Elemento	Costo ($)
Sensor de Partículas	26
Módulo 4G y GPS	80
Sensores Utrasónicos	3
Sensor de Calidad del aire	3.5
Cámara	15
Sensor de sonido	9.1
Raspberry pi	40
ADC	8
Batería	22
Elementos adicionales	27
Acelerómetro	3.8
Sensor de corriente	12
Costo de importación	40
Estructura	60
Total	349.4

Bibliografía

[1] S. Cheruvu, A. Kumar, N. Smith, y D. M. Wheeler, *Demystifying Internet of Things Security*, 2020.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, y E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, num. 4, pp. 393–422, 2002.

[3] W. Dargie y C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[4] H.-j. Jia y Z.-h. Guo, "Research on the technology of rs485 over ethernet," in *2010 International Conference on E-Product E-Service and E-Entertainment*. IEEE, 2010, pp. 1–3.

[5] K. Kazemzadeh, A. Laureshyn, L. W. Hiselius, y E. Ronchi, "Expanding the Scope of the Bicycle Level-of-Service Concept : A Review of the Literature," 2020.

[6] H. Electronics, "Mq-135 gas sensor," vol. 1, pp. 3–4, 2014.

[7] P. Huynh y M. Yoo, "Vlc-based positioning system for an indoor environment using an image sensor and an accelerometer sensor," *Sensors*, vol. 16, num. 6, p. 783, 2016.

[8] H. A. T. U. Manual, T. Sim, L. T. E. Cat, R. Pi, R. Pi, R. P. Zero, Z. Wh, S. Gps, O. Usb, A. T. Commands, O. Cp, B. Uart, G. P. P. Ts, S. Sim, S. A. T. Class, U. Comes, y R. Pi, "Sim7600E-H 4G Hat," pp. 1–27, 2018.

[9] S. M. Velásquez, D. E. Monsalve Sossa, M. E. Zapata, M. E. Gómez Adasme, y J. P. Ríos, "Pruebas a aplicaciones móviles: avances y retos," *Lámpsakos*, vol. 21, num. 21, pp. 39–50, 2019.

[10] R. Espinoza Molina, Claudio; Narvaez Parra, "Informe de calidad del Aire Cuenca-2015," p. 124, 2015. [En línea]. Disponible: https://www.emov.gob.ec/sites/default/files/CalidaddelAirefinal2015{_}0.pdf

[11] "Laser PM2.5 Sensor specification Product model: SDS011 Version: V1.3," Tech. Rep.

[12] S. Zhu y F. Zhu, "Cycling comfort evaluation with instrumented probe bicycle," *Transportation Research Part A: Policy and Practice*, vol. 129, pp. 217–231, 2019.

[13] F. Telefónica, *Smart Cities: un primer paso hacia la Internet de las Cosas*. Fundación Telefónica, 2011, vol. 16.

[14] Mauricio Bouskela, M. Casseb, S. Bassi, C. D. Luca, y M. Facchina, "La ruta hacia las smart cities: Migrando de una gestión tradicional a plusvalías: el caso de la recuperación del frente costero del río la ciudad inteligente," *Bid*, pp. 1–148, 2016.

[15] F. Firouzi, K. Chakrabarty, S. Nassif, y F. Device, *Intelligent Internet of Things*, 2020.

[16] Ilustre Municipalidad de Cuenca, "Plan de movilidad de Cuenca 2015-2025," *Ilustre Municipalidad de Cuenca*, 2015.

[17] G. de Cuenca, "Ordenanza Para La Promocion Y Fortalecimiento De La Movilidad Activa En El Canton Cuenca." [En línea]. Disponible: http://cimpp.ibague.gov.co/movilidad-sostenible/

[18] L. Calderoni, A. Magnani, y D. Maio, "IoT Manager: An open-source IoT framework for smart cities," *Journal of Systems Architecture*, vol. 98, num. December 2018, pp. 413–423, 2019.

[19] O. Uviase y G. Kotonya, "IoT architectural framework: Connection and integration framework for IoT systems," *Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 264, pp. 1–17, 2018.

[20] S. E. C. Bastidas, H. E. C. Meza, A. J. C. Bastidas, y A. A. Cabrera, "Capítulo 1: Las redes de sensores inalámbricas, arquitectura y aplicaciones," *Libros Universidad Nacional Abierta ya Distancia*, 2018.

[21] M. Kocakulak y I. Butun, "An overview of wireless sensor networks towards internet of things," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–6.

[22] F. Karray, M. W. Jmal, A. Garcia-Ortiz, M. Abid, y A. M. Obeid, "A comprehensive survey on wireless sensor node hardware platforms," *Computer Networks*, vol. 144, pp. 89–110, 2018.

[23] H. Modares, R. Salleh, y A. Moravejosharieh, "Overview of security issues in wireless sensor networks," in *2011 Third International Conference on Computational Intelligence, Modelling & Simulation*. IEEE, 2011, pp. 308–311.

[24] U. Nanda y S. K. Pattnaik, "Universal asynchronous receiver and transmitter (uart)," in *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2016, pp. 1–5.

[25] E. Haro, T. Guarda, A. O. Z. Peñaherrera, y G. N. Quiña, "Desarrollo backend para aplicaciones web, servicios web restful: Node. js vs spring boot," *Revista Ibérica de Sistemas e Tecnologias de Informação*, num. E17, pp. 309–321, 2019.

[26] R. Chaturvedi, "Evaluation and refinement of web application architecture framework," 2018.

[27] L. Delía y N. Galdamez, "Un análisis experimental tipo de aplicaciones para dispositivos móviles," *XVIII Congreso Argentino de Ciencias de la Computación*, pp. 766–776, 2013. [En línea]. Disponible: http://sedici.unlp.edu.ar/handle/10915/32397

[28] I. Gryech, Y. Ben-Aboud, B. Guermah, N. Sbihi, M. Ghogho, y A. Kobbane, "Moreair: A low-cost urban air pollution monitoring system," *Sensors (Switzerland)*, vol. 20, num. 4, pp. 1–24, 2020.

[29] D. Wang y J. G. B, *with Bearing Meta-information*, 2019. [En línea]. Disponible: http://dx.doi.org/10.1007/978-3-030-32388-2{_}58

[30] X. Liu, C. Xiang, B. Li, y A. Jiang, "Collaborative bicycle sensing for air pollution on roadway," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. IEEE, 2015, pp. 316–319.

[31] Y. Cohen, M. Constantinides, y P. Marshall, *Places for News : A Situated Study*, 2019, vol. 2. [En línea]. Disponible: http://dx.doi.org/10.1007/978-3-030-29384-0{_}5

[32] E. Bales, N. Nikzad, N. Quick, C. Ziftci, K. Patrick, y W. G. Griswold, "Personal pollution monitoring: mobile real-time air quality in daily life," *Personal and Ubiquitous Computing*, 2019.

[33] A. Fiduccia, F. Pgliaro, L. Gugliermetti, y L. Filesi, "City Dashboards: The Case of Trieste," *Computational Science and Its Applications*, vol. 10407, num. Part IV, pp. 600–612, 2017. [En línea]. Disponible: http://link.springer.com/10.1007/978-3-319-62401-3

[34] F. Corno, T. Montanaro, C. Migliore, y P. Castrogiovanni, "SmartBike: An IoT crowd sensing platform for monitoring city air pollution," *International Journal of Electrical and Computer Engineering*, vol. 7, num. 6, pp. 3602–3612, 2017.

[35] C. Alberoni, E. Leon, M. Wister, y J. Hernandez.

[36] S. Sweeney, R. Ordonez-Hurtado, F. Pilla, G. Russo, D. Timoney, y R. Shorten, "A Context-Aware E-Bike System to Reduce Pollution Inhalation while Cycling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, num. 2, pp. 704–715, 2019.

[37] A. R. Wiratno y K. Hastuti, "Implementation of firebase realtime database to track brt trans semarang," *Scientific Journal of Informatics*, vol. 4, num. 2, pp. 95–103, 2017.

[38] S. Sarkar, S. Rahman, y M. Biswas, "Secureit using firebase, google map and node. js," 2020.

[39] E. Susanti, J. Triyono, y R. Pi, "Pengembangan sistem pemantau dan pengendali kendaraan menggunakan raspberry pi dan firebase," *Jurnal Informatika*, vol. 1, pp. 144–153, 2016.

[40] N. Cavill, S. Kahlmeier, H. Rutter, F. Racioppi, y P. Oja, "Economic analyses of transport infrastructure and policies including health effects related to cycling and walking: A systematic review," *Transport Policy*, vol. 15, num. 5, pp. 291 – 304, 2008. [En línea]. Disponible: http://www.sciencedirect.com/science/article/pii/S0967070X08000450

[41] K. Sælensminde, "Cost–benefit analyses of walking and cycling track networks taking into account insecurity, health effects and external costs of motorized traffic," *Transportation Research Part A: Policy and Practice*, vol. 38, num. 8, pp. 593 – 606, 2004. [En línea]. Disponible: http://www.sciencedirect.com/science/article/pii/S0965856404000539

[42] L. A. Issn, "Diagnóstico y control de material particulado: partículas suspendidas totales y fracción respirable pm 10 * césar augusto arciniégas suárez 1," num. 34, pp. 195–213, 2012.

[43] World Health Organization (WHO), "Guías de calidad del aire de la OMS relativas al material particulado, el ozono, el dióxido de nitrógeno y el dióxido de azufre." *Oms*, p. 9, 2005.

[44] G. Parmar, S. Lakhani, y M. K. Chattopadhyay, "An IoT based low cost air pollution monitoring system," *International Conference on Recent Innovations in Signal Processing and Embedded Systems, RISE 2017*, vol. 2018-January, pp. 524–528, 2018.

[45] V. Martínez Fuentes, "Introducción a la plataforma arduino y al sensor ultrasónico hc-sr04: experimentado en una aplicación para medición de distancias," B.S. thesis, 2016.