UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN WEB, PARA EL ANÁLISIS Y PREDICCIÓN DEL FLUJO DE PERSONAS, APLICADO AL TRANSPORTE URBANO DE LA CIUDAD DE CAÑAR

Trabajo de titulación previo a la obtención

del título de Ingeniero Electrónico

AUTORES: DANNY ANDRES SALTO SUMBA

JUAN JAVIER VÁZQUEZ VERDUGO

TUTOR: ING. JUAN DIEGO JARA SALTOS, MSc.

Cuenca - Ecuador

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Danny Andres Salto Sumba con documento de identificación N° 0106046634 y Juan Javier Vázquez Verdugo con documento de identificación N° 0302653829; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 10 de agosto del 2022

Atentamente,

Danny Andres Salto Sumba
0106046634

Juan Javier Vázquez Verdugo
0302653829

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Danny Andres Salto Sumba con documento de identificación N° 0106046634 y Juan Javier Vázquez Verdugo con documento de identificación N° 0302653829, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Desarrollo de una aplicación Web, para el análisis y predicción del flujo de personas, aplicado al transporte urbano de la ciudad de Cañar", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 10 de agosto del 2022

Atentamente,

Danny Andres Salto Sumba
0106046634

Juan Javier Vázquez Verdugo
0302653829

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Juan Diego Jara Saltos con documento de identificación N° 0103543658, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UNA APLICACIÓN WEB, PARA EL ANÁLISIS Y PREDICCIÓN DEL FLUJO DE PERSONAS, APLICADO AL TRANSPORTE URBANO DE LA CIUDAD DE CAÑAR, realizado por Danny Andres Salto Sumba con documento de identificación N° 0106046634 y por Juan Javier Vázquez Verdugo con documentode identificación N° 0302653829, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 10 de agosto del 2022

Atentamente,

Juan Diego Jara Saltos
0103543658

Índice

CONTENIDO

Agradecimientos

Nosotros agradecemos a Dios y expresamos nuestra más sincera gratitud a las personas que nos apoyaron durante estos cinco años de formación académica, en especial al Ing. Sebastián Ochoa por ayudarnos a plantear la idea original de este proyecto, también extendemos nuestros agradecimientos al Ing. Juan Diego Jara por su ayuda con el desarrollo y tutoría del proyecto.

Danny Salto y Juan Vázquez.

Dedicatoria

Yo Danny Salto dedico este trabajo a mi padre Luis y mi madre Ana por permitirme el acceso a la educación superior y también a mi hermano Jimmy por apoyarme en mi formación académica

Danny Andres Salto Sumba.

Yo Juan Javier Vázquez dedico este trabajo a mis padres, hermanas y familia que fueron parte fundamental en mi formación académica de esta manera expreso mi gratitud y aprecio por ser parte de mi vida.

Juan Javier Vázquez Verdugo.

Glosario

LSTM: Long Short-Term Memory.

GPS: Sistema de Posicionamiento Global.

PR: Probed Request

MPL: Red Neuronal Multicapa.

RNN: Recurrent Neural Network.

CNN: Convolutional Neural Networks.

RSSI: Received Signal Strenght Indicator.

IOT: Internet Of Things.

CPU: Unidad Central de Procesamiento.

GPU: Unidad de Procesamiento Grafico.

HTTP: Hypertext Transfer Protocol.

MQTT: Message Queue Telemetry Transport.

TIC: Tecnologías de la información y Comunicaciones

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos

Resumen

El garantizar un transporte público monitoreado que mediante diferentes alternativas pueda evitar la congestión en las vías, incremento de contaminación ambiental, retraso en la llegada de los usuarios siendo estos principales factores que afectan a los ciudadanos en su día a día. Este documento presenta soluciones a la problemática anterior enfocado en el conteo de civiles mediante sus dispositivos móviles y realizando una predicción basada en redes neuronales LSTM.

El seguimiento y la obtención de datos se organizó en la ciudad de Cañar en el transporte público urbano, el conteo de los civiles se obtuvo mediante sensores WIFI de los dispositivos móviles de tal manera poder preservar la privacidad del usuario que haga uso del transporte público, el sistema únicamente contabiliza a usuarios que tengan dispositivos móviles habilitados para WIFI. Este proyecto demuestra que el método utilizado para la estimación de civiles es más factible de implementar a comparación de métodos que utilizan visión por computadora por los recursos computacionales y económicos.

Él envió de la información se realizó por medio del protocolo MQTT que es utilizado en aplicaciones IoT ya que no necesita de un ancho de banda excesivo para enviar mensajes cortos.

Se utilizo un modelo de red LSTM para pronosticar la cantidad de usuarios que podrían transportarse en la unidad de transporte público, el modelo genera un rendimiento adecuado con datos de series temporales como es el caso de este proyecto. Finalmente expondremos los datos mediante un servidor WEB que está conectado a una base datos MySQL, la interfaz WEB del servidor expone datos: históricos, en tiempo real y predicciones realizadas en base a datos almacenados.

El análisis de este sistema en general muestra que es una alternativa solida a la hora de realizar pronósticos y monitoreos de multitudes, ofreciendo a los administradores alternativas para que puedan tomar mejores decisiones y mejorar el transporte público en base a los datos expuestos y así garantizar un desarrollo sostenible en el transporte público de la población.

Introducción

La ciudad de Cañar carece de un sistema de monitoreo en el transporte público, recolectar estos datos permite optimizar este recurso. Los transportes públicos ahorran combustible y por ende menos contaminación, siempre que el sistema sea efectivo y eficiente, y este proyecto puede conseguir que el sistema lo sea. Este sistema puede tener otras aplicaciones, por ejemplo: evitar aglomeraciones de personas, evitar que las unidades de transporte tomen rutas en las cuales no encontraran pasajeros, ayudar a un pasajero a planificar su ruta, llegando a su destino cómodamente y a tiempo.

Se propone como solución un sistema que fusiona el concepto de internet de las cosas, Ciudades Inteligentes para desarrollar un prototipo de nodo inteligente que permita censar a los usuarios de transporte público, elaborar una base de datos donde se guardara la información obtenida por los nodos inteligentes, un sistema de predicción y para la visualización de los datos una plataforma web, este análisis ayudara a las autoridades en base a los datos obtenidos proponer y crear soluciones para optimizar el transporte público generando una mejor calidad de vida a la población.

Antecedentes del Problema de Estudio

Actualmente las TICs han facilitado el ingreso a Internet de manera fácil y con costos bajos, gracias a esto muchos dispositivos pueden conectarse a Internet: teléfonos, electrodomésticos, automóviles, etc. En el futuro se espera que todas las cosas estén conectadas a Internet, a este proceso se le ha dado el nombre de Internet de las Cosas (IoT)[1]. Según Cisco se esperaba que para el 2020 haya alrededor de 50 millones de dispositivos conectados a la red [2].

Con el avance de la tecnología ahora podemos tener dispositivos de uso cotidiano conectados a Internet, estos dispositivos pueden recolectar datos y transmitirlos, permitiendo la gestión eficiente de recursos basados en estos datos, mediante técnicas conocidas como el "Big Data"[3].

Los dispositivos de uso cotidiano conectados de manera masiva a la red pueden ayudar a una ciudad a usar sus recursos de manera óptima e inteligente, pudiendo tener control y monitoreo de los servicios básicos que se da a la ciudadanía, así como también los factores que afectan la salud de la población y el medio ambiente [4].

Algunos de los problemas más conocidos de una ciudad son: el consumo de energía, el nivel de contaminación, el tráfico, y, por supuesto, la necesidad de descarbonización, unido a la necesidad de que la ciudad mejore desde todos los puntos de vista del "paraguas" social, etc. En los últimos años se han venido dando soluciones creativas a estos problemas, estas soluciones combinan tecnologías como IOT y Big Data, las ciudades que optaron por estas soluciones se les denomina ciudades inteligentes [3].

De hecho, una de las soluciones a los problemas que presenta una ciudad es el proyecto "Climate Colab" que combina la inteligencia humana y la máquina, facilitando que surja la inteligencia colectiva para dar solución a unos de los problemas globales más complejos: el cambio climático global [5]. Mediante la inteligencia colectiva es posible encontrar mejores soluciones a diferentes problemas, unos de los modelos que surge de la Inteligencia Colectiva es el crowdsourcing[6]. El Crowdsourcing puede definirse como una forma de colaboración en la que casi todo el mundo puede contribuir a la realización de diferentes tareas definidas con anterioridad o surgidas por necesidad [3].

Actualmente ciudades como: Londres, Tokio, Barcelona, etc. Cuentan con plataformas para ser ciudades sostenibles y amigables con el medio ambiente, haciendo uso del IOT y Big Data[4].

Los países latinoamericanos se encuentran en un desarrollo urbanístico comparable al de los países desarrollados, las ciudades concentran el 75% de los habitantes, una característica urbana similar a Estados Unidos y Europa, lo que supone que las ciudades de América Latina tengan tendencia a ser ciudades sostenibles y aplicar soluciones que se utilizan en las Ciudades Inteligentes. En América

Latina, el medio de transporte más utilizado son los autobuses, pero son ineficientes provocando: congestión en las vías, incremento de contaminación ambiental, retraso en la llegada de los usuarios, siendo una de las causas principales que impactan la salud de los habitantes, la contaminación que provocan en el aire estos vehículos [7].

El problema de contaminación del aire y tráfico en Ecuador afecta a la calidad del aire de sus ciudades, según la Organización Panamericana de Salud, se estima que en Ecuador mueren 500 personas por esta causa cada año [8]. La ciudad de Cañar presenta altos índices de contaminación por lo que se realizó un sistema de monitoreo ambiental en tiempo real para analizar factores como la humedad, la temperatura y niveles de CO2, para que las autoridades tomen las prevenciones adecuadas[4].

Justificación (Importancia y Alcances)

Los problemas de desarrollo sostenible a los que se enfrentan las ciudades requieren de servicios eficientes, que apunten a los nuevos objetivos de sostenibilidad, integración de soluciones tecnológicas que apoyen a generar soluciones buscando resolver problemas complejos y que luego estas soluciones, sean extrapoladas a un ámbito global.

Los sucesivos avances tecnológicos en las "Ciudades Inteligentes", genera resultados positivos como:

- Desarrollo sustentable.
- Aumento de la calidad de vida de los habitantes.
- Optimización de los recursos disponibles.

El desarrollo tecnológico en el transporte, permite que los habitantes se trasladen más rápido, con mayor constancia y a distancias largas; estos avances han provocado desplazamientos masivos generando cambios en el comportamiento de los habitantes y en su forma de vida, para ello, la idea central es censar el flujo de multitudes geolocalizadas en el transporte público, esto puede servir para la creación de un sistema que permita solucionar problemas como: aglomeraciones, falta de usuarios en diferentes estaciones de transporte público, permitiendo así, adoptar algunas alternativas al sector de movilidad que tengan como objetivo garantizar una movilidad sostenible. Se puede utilizar en el sector sanitario, en el transporte, en la reducción de tiempos de desplazamiento de usuarios, etc.

En un entorno de pandemia mundial, el transporte público ha sido uno de los principales focos de propagación debido a su uso diario por gran cantidad de personas, con las consecuentes aglomeraciones. El sistema puede ayudar a limitar las probabilidades de contagio, también pretende mejorar el transporte público, reducir de gastos de operación, controlar el aforo no solo en transportes públicos, si no también entidades públicas o privadas que adopten la tecnología posteriormente. A la culminación de este proyecto que se centra específicamente en la implementación del sistema en el sector del transporte público urbano, se pretende obtener resultados que ayudaran mediante el censado de usuarios en tiempo real, para evitar en lo posible las aglomeraciones en el transporte público. Es decir, el sistema da la posibilidad de tomar una unidad de transporte que este vacío para evitar la aglomeración en una unidad que tiene su aforo al máximo. Otra solución que ayuda es cambiar los itinerarios de los viajes.

La investigación convencional en monitorización de multitudes se centra principalmente en sistemas con visión por ordenador. Factores como la oclusión, el enorme tratamiento de los datos, el coste de la aplicación y los problemas de privacidad inhiben la posibilidad de utilizar tales sistemas para el

conocimiento de la situación en tiempo real.

Estas deficiencias llevaron al desarrollo de estrategias alternativas de monitoreo de multitudes utilizando teléfonos celulares, sistema de posicionamiento global (GPS), Bluetooth y WIFI. Una visión de estas tecnologías revela que los sensores WIFI no solo están libres de oclusión, sino que también son no intrusivos y rentables. Se requiere solamente que las personas tengan un teléfono celular con WIFI por lo que su costo de implementación es bajo, las señales WIFI son de baja potencia por lo que no afectan a las personas, la alta demanda de celulares hace posible que este método tenga una taza de éxito elevada. Ya se han demostrado aplicaciones exitosas de redes de sensores inalámbricos en la detección y el monitoreo para: recuento de peatones en el tráfico urbano, automatización de procesos industriales, monitoreo de deslizamientos de tierra, y monitoreo ambiental distribuido.

Motivado por tales aplicaciones, este proyecto contribuye al conocimiento existente mediante la utilización de sensores WIFI no sólo para estimar sino también para pronosticar el recuento de multitudes en eventos públicos a gran escala.

Los datos recolectados por los sensores pueden llegar a ser un gran volumen de datos que se pueden recopilar todos los días, “Big data” es una definición para este suceso. La cantidad de datos no es lo importante, lo determinante es lo que hacen las organizaciones con los datos. La importancia de la recolección de datos supone como resultado optimizar decisiones y acciones estratégicas.

De hecho, existe una rama de la ciencia conocida como “Data Science” que es la ciencia computacional de extraer información significativa de datos recolectados, para luego procesarlos y analizarlos, para luego dar una conclusión de esa información y generar un valor.

Objetivos

OBJETIVO GENERAL

- Desarrollar una aplicación WEB, para el análisis y predicción de flujo de personas, en el transporte urbano de la Ciudad de Cañar.

OBJETIVOS ESPECÍFICOS

- Investigar e implementar mediante software libre, un sistema para la identificación de civiles con terminales WIFI dentro de una unidad de transporte.
- Implementar una base de datos en un servidor WEB, que permita almacenar y visualizar la información recolectada.
- Diseñar una aplicación WEB, para la visualización y predicción gráfica de los datos.

CAPÍTULO 1

1. Fundamentación Teórica o Estado del Arte

En este capítulo definimos la teoría de las Ciudades Inteligentes, IOT, técnicas de Crowdsourcing, Big Data y Crowdsensing. Además, los distintos componentes para el desarrollo de un prototipo capaz de detectar el flujo de civiles enfocado en el transporte público.

1.1. Sistema de Transporte Publico

Es un sistema de transporte que moviliza a la población en masa, capaz de resolver las exigencias de desplazamiento de los ciudadanos. La finalidad del sistema es la movilización eficiente, ligera, agradable y segura de los habitantes a los sitios donde ejercen sus actividades [7].

1.2. Ciudades Inteligentes

Las ciudades inteligentes se les puede definir como las ciudades que utilizan de mejor manera los servicios e infraestructuras tecnológicas integrándolas en los diferentes servicios de energía, TIC y transporte para solventar todas las necesidades sociales como también las económicas. Muchas de las tecnologías inalámbricas han presentado una solución eficiente para que exista un crecimiento en la cantidad de dispositivos de uso cotidiano que se conectan a la red, como también un incremento en la demanda de servicios que pueden realizar un monitoreo a la ciudad, aquí es donde el IOT está generando soluciones confiables y ganando mucha popularidad al implementar aplicaciones en el transporte, salud, medio ambiente, proporcionando en ellos los mecanismos de comunicación necesarios [9].

1.3. IOT (Internet de las Cosas)

Varios investigadores han definido a IOT como: “Una red libre e integral de objetos inteligentes capaces de compartir información, datos y recursos, detectando y procediendo en circunstancias de cambios en el ambiente.” [10].

El IOT constituye dispositivos de uso cotidiano conectados a la red de manera masiva que conforman un sistema mundial de redes informáticas conectadas entre sí y que se basan y ejecutan los protocolos [TCP/ IP] [11].

1.4. Protocolos usados en el Internet de las Cosas.

1.4.1. Protocolo MQTT

El protocolo MQTT está basado o permite una trasmisión de mensajes en agrupaciones específicas de clientes de forma específica, es decir se trata de un sistema de mensajería asíncrona utilizado en

comunicaciones maquina a máquina (M2M). Este modelo está basado en la transacción de mensajes a través de publicación y suscripción [12]. El protocolo concede el acceso a clientes a una suscripción para acceder a temas (tópicos) de interés a un servidor centralizado conocido como Broker MQTT [13].

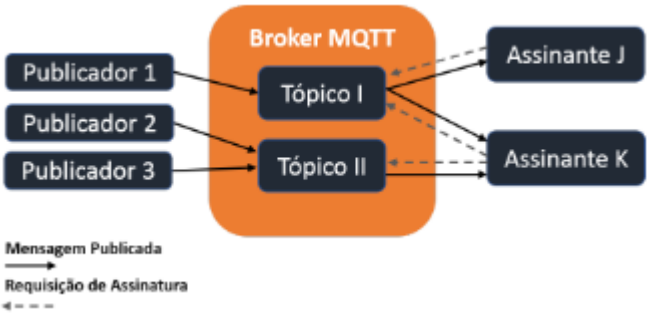


Ilustración 1. Modelo Publisher-Subscriber proporcionado por el protocolo MQTT [13]

En la ilustración 1 se observa un escenario donde tres dispositivos nombrados como publicadores acceden a un broker que tiene dos temas y dos suscriptores, en la representación gráfica observamos el interés del dispositivo del suscriptor J que tiene un interés en el mensaje del Tema 1, el cual tiene la información del Publicador 1 [13].

El suscriptor K recibe los datos por el publicador 1 al suscribirse al Tema 1, a su vez también recibirá datos del publicador 2 y 3 ya que también está suscrito al Tema 2. Los suscriptores y publicadores se los conoce como un cliente MQTT, su ventaja es la facilidad de implementación ya que en cuestiones de Hardware no tiene problemas puesto que un cliente MQTT puede ser implementado desde un computador hasta un microcontrolador. El broker MQTT es el servidor responsable del reparto de mensajes entre el publicador y suscriptor. El broker recibirá los mensajes realiza una inspección del topic al que están suscritos y los dirige hacia el cliente que esta suscritos a ese topic verificado. El broker tiene la capacidad de almacenar los mensajes hasta que el cliente al que va dirigida la información se conecte [12].

MQTT es adecuado en aplicaciones relacionadas con el Internet de las Cosas para propósitos como: monitorización en tiempo real, análisis en tiempo real, mantenimientos, etc. Estas aplicaciones conocidas como: Thingspeak, EMQX, Obzervr y muchas más, son aplicaciones de plataforma abierta que están diseñadas para recoger, almacenar y analizar datos en tiempo real de sensores [14].

1.4.2. Brokers MQTT

Para proyectos de IOT existen muchos brokers disponibles, varios son de código abierto y otros de pago, en la tabla 1 se detalla las ventajas de los mejores brokers mqtt de código abierto.

Tabla 1Ventajas de Brokers MQTT open source.

Broker MQTT	Ventajas
mosquito	Escrito en C, Mosquitto es el bróker más popular para proyectos IOT, es ligero y escalable. Admite MQTT 3.x y MQTT 5.
EMQ X	Escrito en Erlang, EMQ X es el bróker más escalable, soporta mensajería para más de 10 millones de clientes en un solo clúster. Admite MQTT 3.x y MQTT 5.
Cassandana	Cassandana es una solución Java para MQTT y admite todas las características que admite el protocolo.
Ejjaberd	Escrito en Erlang, los servicios de intermediario MQTT 5.0 se ejecutan sobre un servidor XMPP maneja millones de conexiones simultáneas.

Para este proyecto vamos a utilizar el bróker EMQ X por las opciones de escalabilidad disponibles, y además es de fácil implementación como los demás brokers.

1.5. Crowdsourcing

El Crowdsourcing es un grupo de propuestas de tipo colaborativo que se alimenta de procesos como la inteligencia colectiva, algunos investigadores han tratado de definir el crowdsourcing como un desarrollo de soluciones de problemas [15].

El Crowdsourcing es una técnica eficaz que utiliza la inteligencia humana para recopilar datos, las personas actúan como proveedores de servicios. El crowdsourcing móvil (MCS) utiliza mecanismos que combinan las ventajas de las TIC y las comunicaciones móviles [16].

Las principales características de MCS se pueden presentar de la siguiente manera[16]:

- Movilidad: los dispositivos inteligentes reflejan los patrones de movilidad y las interacciones sociales de sus operadores. Por tanto, la información de movilidad de los dispositivos se logra usar para optimizar la utilidad de MCS.
- Colaboración: MCS permite distribuir tareas a los trabajadores elegibles en un grupo de trabajo. De esta manera, los trabajadores colectivos pueden realizar de forma colaborativa un conjunto de tareas divididas para lograr un objetivo global.
- Capacidad humana: las personas o los trabajadores móviles desempeñan el papel de consumidores y productores de datos en MCS, donde sus capacidades de detección, comunicación y procesamiento se utilizan para optimizar la utilidad de los sistemas de MCS.

En este entorno el Crowdsourcing "explicaría una forma de distribución social que surge y se forma a través de las TIC. Las organizaciones de personas surgen, como tales, cuando la tecnología les permite las relaciones sociales básicas para cooperar y ordenar sus funciones de forma colectiva" [17].

1.6. Crowdsensing

Crowdsensing es una red de sensores, donde los sensores son usuarios capaces de detectar el entorno y envían los datos obtenidos utilizando, por ejemplo, sus teléfonos inteligentes, que vienen con un conjunto completo de herramientas de detección (GPS, acelerómetro, cámara, micrófono, entre otros), teniendo estos dispositivos la capacidad para detectar y monitorear su entorno circundante[18]. Además, los propietarios de teléfonos inteligentes pueden detectar distintas propiedades de su entorno y sus descripciones podrían complementar las lecturas de sensores de hardware, lo que conduce a un nuevo nivel de detección, que es la detección inteligente[18].

Confiar en los usuarios para censar el medio ambiente y enviar los datos obtenidos utilizando sus teléfonos inteligentes abre el camino a una nueva generación de red de sensores, comúnmente conocidos como redes de crowdsensing, donde los usuarios se convierten en crowdsensors[18].

Entonces se puede aprovechar el poder de la inteligencia colectiva, junto con las capacidades de detección de estos dispositivos inteligentes para construir redes de sensores más potentes. Aunque las redes de detección colectiva pueden ayudar a superar muchas de las limitaciones de las propuestas existentes en las redes de sensores, su productividad depende de la movilidad de los usuarios y de su voluntad de cooperar[18].

Además, convencer a los usuarios a participar en las tareas de detección de una red es un gran desafío y por eso existen pocas aplicaciones de redes de teledetección. Por ejemplo, el proyecto MetroSense ofrece una arquitectura de red para la detección centrada en las personas que aprovecha la infraestructura urbana existente y la movilidad humana para detectar datos ambientales oportunistas, con aplicaciones como BikeNet o SkiScape. Sin embargo, ninguna de las aplicaciones proporciona una estimación del número de usuarios o de la densidad del sensor necesaria para asegurar su utilidad[18].

1.7. Servidor WEB

El servidor web en un contexto sencillo es un servidor de archivos, donde los clientes se dirigen mediante un protocolo HTTP para adquirir un recurso solicitado. El servidor WEB no realizara ningún tipo de proceso en el recurso antes de ser trasmitido al cliente. El cliente realizara una petición dirigida al servidor, este le otorga una respuesta con el contenido que fue solicitado. El servidor WEB provee algunas funciones para realizar peticiones HTTP como son las siguientes: GET, POST, PUT, DELETE; estos se utilizan dentro del desarrollo de una aplicación para comunicarse con el servidor para adquirir datos de importancia como lo es para este proyecto, en este caso adquiere las coordenadas, latitud y longitud, obtenidas de un recorrido del transporte público [19].

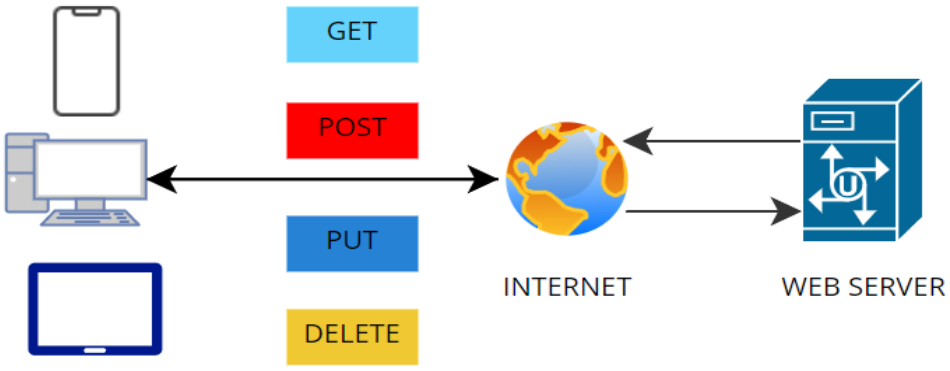


Ilustración 2. Servicio WEB (Fuente: Autor)

1.8. Data Science

En los últimos años, la ciencia de los datos se generó como una disciplina nueva y también importante. Puede verse como una unión de instrucciones clásicas como estadísticas, minería de datos, bases de datos y sistemas distribuidos[20]. Los enfoques existentes deben combinarse para transformar abundantes datos disponibles en valor para las personas, las organizaciones y la sociedad. Las técnicas

de minería de procesos utilizan datos en eventos para descubrir procesos, verificar el cumplimiento, comparar variaciones de procesos y recomendar mejoras. El Data Science puede definirse como un conjunto de principios fundamentales que apoyan y guían la extracción de información y conocimiento los datos.[21]

Se puede utilizar las tecnicas de big data en la ingeniería de datos. Ocasionalmente, las tecnicas de big data se utilizan para implementar técnicas de minería de datos, para el procesamiento en los datos y otras actividades de ciencia de datos. [20]

1.9. Redes Neuronales

La red neuronal se le puede definir como un sistema que consta de una gran cantidad de elementos simples, estos elementos de proceso están interconectados, procesando información a través de un estado dinámico en respuesta a las entradas del exterior [22]. Estas redes tratan de simular el comportamiento del cerebro humano, la simulación se caracteriza por aprender ciertos parámetros a partir de la experiencia y la extracción de conocimiento genérico de un conjunto de datos [22].

Las redes neuronales se consideran como aproximadores no lineales con capacidad para poder mapear funciones no lineales, según la aplicación en la que se vaya implementar, se utiliza varios tipos de arquitecturas neuronales, como perceptrones multicapa (MPL), redes neuronales recurrentes (RNN), memoria a largo plazo (LSTM), red neuronal convolucional (CNN), etc.[23]

Una RNN toma como entradas parámetros con una longitud variable $x = (x_1, \dots \dots \ x_n)$ esta produce estados ocultos $h = (h_1, \dots \dots \ h_n)$, en cada proceso, podemos observar en la ilustración 3, que toma una entrada x_i y h_{i-1} y produce en cada paso un estado oculto h_i generando una salida Y_i [22].

$$h_i = f(Wh_{i-1} + U_{xi})$$

Donde W y U identifican ese proceso de conexión de neuronas que determinan un peso sináptico. Estos pesos guardan la mayor información o conocimiento de la red neuronal sobre el proceso que realiza esta.[24]

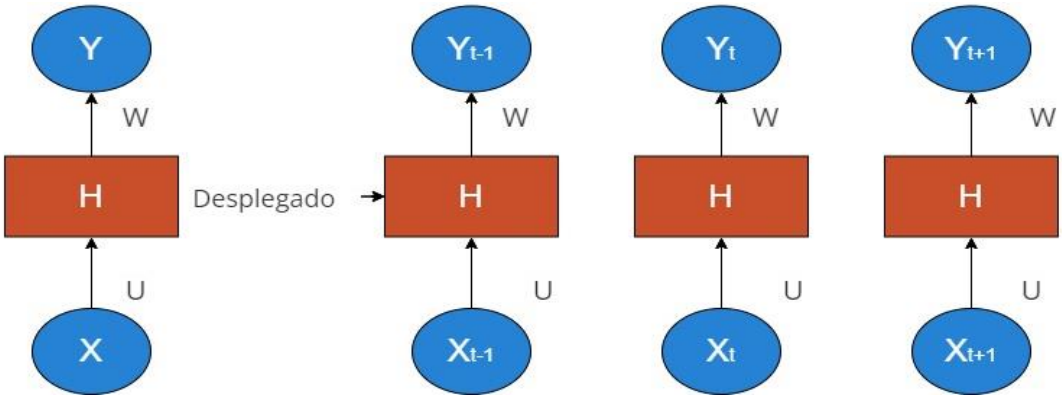


Ilustración 3. Despliegue de una Red Neuronal Recurrente. (Fuente: Autor)

Las RNN hoy en día son soluciones para casos donde las brechas de información y el lugar en donde se necesita su margen de error es pequeña, hay casos donde las brechas son muy grandes ya que su información no se conecta de manera eficiente, para ello surge las redes LSTM pues estas no tienen ese problema.[24]

Con el desarrollo de LSTM al tener la capacidad de almacenar información pasada, el análisis de datos dependientes de factores que puedan cambiar con el paso del tiempo se vuelve más eficientes con respecto al análisis de las diferentes arquitecturas antes mencionadas.[23]

1.9.1. LSTM

Las redes LSTM están diseñadas a partir de las redes neuronales recurrentes (RNN), debido a su estructura de unidad de almacenamiento se ha convertido en uno de los predictores más populares gracias a su capacidad de aproximar de manera no lineal y su aprendizaje no adaptativo.[25]

Una de las claves de una red neuronal LSTM es como almacena su memoria, permitiendo así que la información que ya paso se vaya actualizando a medida que pasa el tiempo y que llega nueva información a la red neuronal.[26]

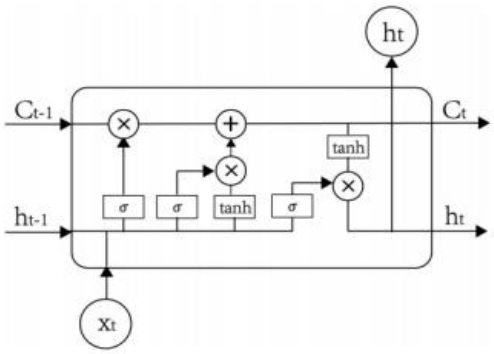


Ilustración 4. Estructura de una red LSTM.[25]

Para analizar el funcionamiento de una red LSTM observamos que la estructura de esta red tiene una entrada C_{t-1} correspondiendo a esta una salida C_t, a estos elementos se les llama como celda de estado actual, a esta red se le podrá añadir y remover datos de la memoria de la red, para modificar los datos de la memoria de la red se utiliza tres compuertas:

- FORGATE GET: Elimina componentes de la memoria.
- UPDATE GATE: Agrega componentes a la memoria.
- OUTPUT GATE: Actualiza los componentes en la memoria.

Estos elementos funcionan como válvulas para permitir y bloquear el paso de información. La variable Xt identifica los datos de entrada, h_{t-1} corresponde a los datos de salida anterior y la variable h_t corresponde a la nueva salida [25].

1.10. WIFI

WIFI es un estándar de comunicación bidireccional inalámbrica, entre un punto de acceso (AP) y una estación de usuario o estación móvil. En 1985 la Comisión Federal de Comunicaciones de EE. UU emitió las bandas de frecuencias científicas, industriales y médicas (ISM) para su uso sin licencia, dando paso para el desarrollo de WIFI. En 1997 el IEEE lanza el primer protocolo para WIFI, el IEEE 802.11, desde entonces ha ido incorporando mejoras para adaptarse a requisitos más exigentes [27].

Ilustración 5. Logo WIFI. (Fuente: www.wi-fi.org)

Los principales componentes de una red WIFI son: puntos de acceso (AP) que son estaciones base habilitadas para WIFI y estaciones cliente que son cualquier dispositivo que tenga una interfaz de red inalámbrica habilitada para WIFI (teléfonos móviles, computadores portátiles, etc.). Entre el AP y las estaciones cliente se establece la comunicación bidireccional inalámbrica [27].

Las estaciones cliente WIFI se conectan a los AP a través de un proceso de descubrimiento de red, este proceso ocurre cuando el dispositivo cliente inicia el descubrimiento de red mediante él envió de una solicitud de sondeo (PR) que es una trama de administración de red que contiene la dirección MAC del dispositivo cliente y el RSSI, esto permite a los AP cercanos saber que un cliente desea establecer una conexión. Cuando se recibe la solicitud de sondeo el AP devuelve una trama de respuesta de sondeo que incluye los parámetros de red necesarios para establecer la conexión, si existe más de un AP se conectara a la red con mejor recepción de señal, para obtener la recepción de la potencia se utiliza el RSSI que también es enviado en la trama de solicitud de sondeo. El cliente no tiene control sobre la trama de administración de red que envían sus dispositivos aparte de apagar el WIFI [27].

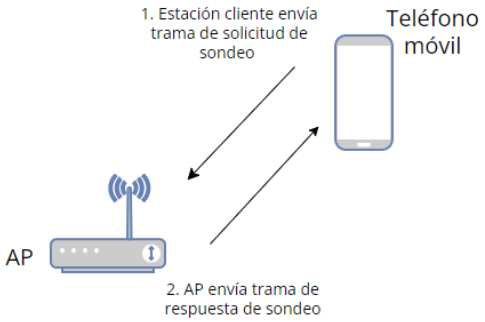


Ilustración 6. Solicitud de sondeo y respuesta de sondeo entre un AP y estación cliente. (Fuente: Autor)

1.11. Sistemas de Detección de Civiles

Existen dos grandes sistemas para la detección de civiles: método de detección por visión de computador y métodos no visibles.

El método de visión por computador emplea redes neuronales que se entrenan para la detección de objetos, en este caso civiles, requiere grandes recursos computacionales. Los métodos no visibles se enfocan en recolección de señales que emiten los objetos que portan los civiles (teléfonos celulares, relojes inteligentes, etc.), este método emplea menos recursos computacionales y menor costo de implementación [28].

Ilustración 7. Reconocimiento de civiles que emplea visión por computador. (Fuente: https://arbusta.net/blog-es/computer-vision-data-annotation-2021/)

En el análisis de multitudes de civiles, hay cinco áreas principales que incluyen ubicación, seguimiento, estimación de densidad, reconocimiento de comportamiento, detección de emociones y anomalías [29].

1.11.1.Estimación de Ocupación Usando solo Mediciones de Potencia WIFI

Este sistema está enfocado en contar el número total de personas que caminan en un área, utiliza solo dos antenas estacionarias (transmisora y receptora), las mediciones de potencia de señal recibida WIFI (RSSI) entre el par de antenas se ve afectada por las personas que dejan su huella en la señal transmitida. El modelo caracteriza el impacto de la multitud en: el bloqueo de la línea de vista (LOS) y el desvanecimiento por trayectos múltiples. El sistema puede estimar el número total de personas con buena precisión y utiliza solo un par de tarjetas WIFI y sus medidas RSSI correspondientes [30].

Ilustración 8. Un transmisor y un receptor fijos son responsables de determinar el número de personas mediante mediciones de potencia. [30]

1.11.2.Estimación de Densidad de Multitudes basada en bluetooh

La base de esta técnica se apoya en la observación de que las personas tienen el Bluetooth de su teléfono en modo detectable. Este método estima la densidad de la multitud en un escaneo de dispositivos detectables y establece que el número de dispositivos devueltos es el número de personas en el área definida por el rango Bluetooth (alrededor de 10m) [31].

1.11.3.Detección de Civiles de Multitudes basado en WIFI

Este método para estimar el conteo de multitudes utiliza solicitudes de sondeo (PR). Las PR ayudan a un smartphone (o cualquier dispositivo habilitado para WIFI) en la búsqueda de puntos de acceso cercanos (AP). Cada PR, contiene la dirección de origen del dispositivo que envía, que es básicamente su dirección MAC, por lo tanto, contar las PR indirectamente supone el número de personas presentes en un evento [28].

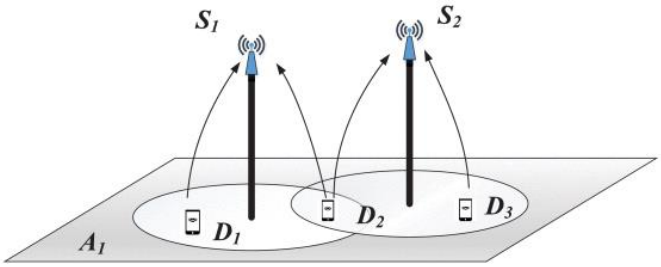


Ilustración 9. Despliegue y detección. Sensores S1 y S2 detectan las PR que envían los dispositivos D1, D2 y D3. [28]

1.11.4.Sistema de Detección de Civiles Seleccionado

Los sistemas de detección de civiles enfocados especialmente en prototipos de visión por computadora son difíciles de implementar para obtener información en tiempo real por elementos como: oclusión, transmisión de grandes cantidades de datos, costos de implementación, etc. Las dificultades por los elementos mencionados han llevado al desarrollo de nuevos métodos para el monitoreo de civiles utilizando teléfonos celulares, sistema de posicionamiento global (GPS), Bluetooth y WIFI [28].

La alta disponibilidad de teléfonos inteligentes equipados con sensores como acelerómetros, GPS, giroscopios, WIFI, etc., ha hecho posible que se utilicen métodos no visibles para la detección, conteo/estimación de civiles [29].

Para este proyecto nos hemos enfocado en el área de conteo/estimación utilizando métodos no visibles, por costos de implementación y porque requiere menos recursos computacionales. Elegimos el

método basado en WIFI, este sistema es una forma más robusta que los otros métodos para la estimación de civiles.

1.12. Base de Datos

Nuestra vida cotidiana actual nos exige que constantemente estemos relacionándonos con base de datos. Por ejemplo, cuando un estudiante revisa sus calificaciones de la universidad accede al sitio web, la información es entregada al estudiante mediante un mecanismo de información. La evolución rápida de las tecnologías de internet y las telecomunicaciones han hecho de las bases de datos un área de estudio elemental [32].

La base de datos representa una recolección de información relacionados en un almacenamiento único para toda una organización, tiene que ser diseñada para administrar la información (los conjuntos de programas para administrar la base de datos se llaman SGBD), encargándose de satisfacer las peticiones del usuario en los niveles: operativo, táctico y estratégico. Los programas de aplicaciones permiten que el usuario interactúe con el SGBD [32].

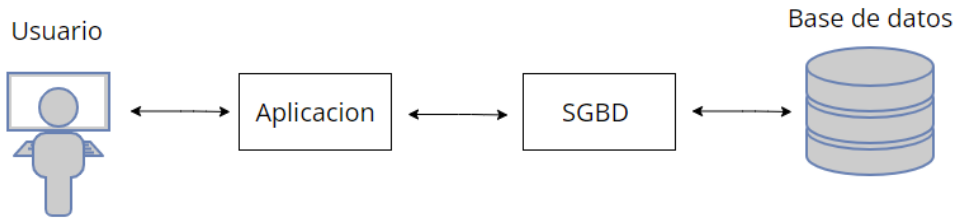


Ilustración 10. Relación usuario, aplicación de bases de datos, SGBD y base de datos. (Fuente: Autor)

1.12.1. Sistema de Gestión de Base de Datos Seleccionada

Tiene como objetivo que los datos sean organizados y manipulados mediante una interfaz estándar para que otros programas puedan ingresar a la base de datos, además, tiene que evitar que un usuario de base de datos manipule directamente la base de datos [33].

En el mercado existen muchos SGBD los más comunes son:

- Oracle
- SQL Server
- MySQL
- PostgreSQL

Tabla 2 Comparación SGBD más comunes.

SGBD	VENTAJAS	DESVENTAJAS
Oracle	Base de datos relacional más usada a nivel mundial Soporte multiplataforma Soporte de transacciones Conectividad segura Innovación	Precio de licencia muy alto, difícil de conseguir para empresas pequeñas o usuarios individuales Una mala configuración lo hace muy lento
SQL Server	Fácil configuración e instalación Permite administrar permisos a todo Conectividad segura Compatible con los paquetes y programas de Microsoft	Licencia privada, precio elevado para algunas organizaciones Utiliza mucha memoria RAM, pudiendo llegar a necesitar maquinas muy potentes Relación Calidad/Precio por debajo de Oracle
MySQL	Licencia gratuita No necesita mucha memoria RAM Fácil configuración e instalación	No existe soporte para la versión gratuita
PostgreSQL	Puede soportar terabytes de información Ideal para organizaciones con presupuesto limitado	Configuración tediosa y complicada No es veloz cuando se realizan consultas grandes

Para este proyecto utilizaremos MySQL porque es de código abierto, tiene funciones para principiantes, y también porque está más valorado en la parte del desarrollo web.

Ilustración 11. Logo MySQL. (Fuente: https://www.mysql.com/)

1.13. Raspberry PI 4

Raspberry Pi 4 es un ordenador , que tiene como objetivo principal realizar aplicaciones relacionadas en el mundo de la programación, en esta placa se puede desarrollar scripts o programas utilizando su lenguaje principal Python, este es el lenguaje del sistema operativo [34].

Raspberry Pi es apto para ejecutar el sistema operativo Kali Linux el cual ha sido elegido para desarrollar este proyecto. Esta tarjeta, de placa reducida no dispone de un disco duro, pues necesita una tarjeta SD para almacenar e instalar el sistema operativo así, como también puede almacenar varias aplicaciones según la necesidad [35].

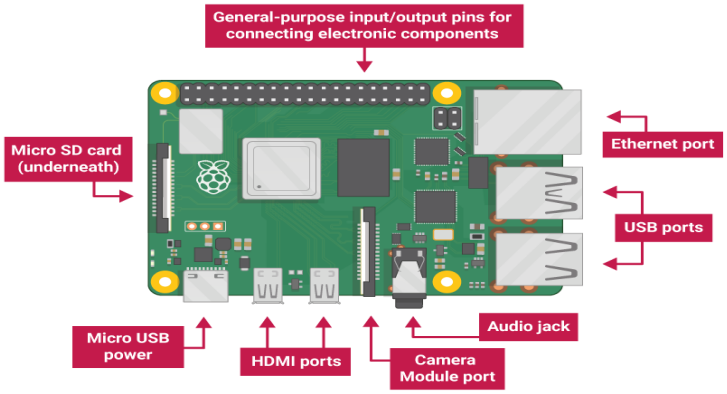


Ilustración 12. Anatomía Raspberry PI. (Fuente: https://canaltic.com)

Tabla 3. Ficha técnica Raspberry PI4. (Fuente: https://canaltic.com)

Raspberry PI 4	
COMPONENTES	CARACTERISTICAS
CPU	Cuatro núcleos de procesamiento a 1.5GHz
GPU	Video: Core VI
Memoria	1/2/4GB LPDDR4 RAM
Conectividad	802.11ac, un-WI-FI/ Bluetooth 5.0, Gigabit Ethernet

Alimentación	5V/3ª vía - USB-C, 5V vía cabezal GPIO
Video y Sonido	2 x puertos micro – HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salidas estéreo de: 4 polos y puerto de video compuesto.
Puertos	2 x USB 3.0, 2 X USB 2.0
Expansión	Cabezal GPIO de 40 pines

La elección de este ordenador de placa simple para nuestro proyecto este vasado en muchos beneficios, uno de los grandes beneficios es el uso de sistemas operativos Linux en los que están basados estos dispositivos, la Raspberry Pi tiene la posibilidad de convertirla en un servidor para garantizar la comunicación servidor-cliente sea segura y sin fallos, además que la gran mayoría de servidores están montados sobre distribuciones Linux [34].

La tabla 2 muestra una breve comparación de diferentes placas electrónicas que existen en el mercado.

Tabla 4 Comparación de Placas Electrónicas. (Fuente: https://predictabledesigns.com/)

Raspberry Pi	Arduino	STM-32
• Computador de placa reducida y única. • Sistema operativo principal Linux • Incluye propios puerto USB, salidas de audio, controlador grafico HDMI • Conexión a dispositivos con Bluetooth y Internet • Su procesador es más rápido en multitarea	• Es un microcontrolador • No puede ejecutar varios programas al mismo tiempo • Facilita la conexión con sensores analógicos, motores, etc. • No se necesita de configuraciones, tampoco de instalaciones de sistemas operativos • Es muy útil para proyectos en los que se necesita	• Es u • n microcontrolador • Permite programar en lenguajes C y C++ • Compatible con IDE de Arduino • El desarrollo de código no es tan simple y directo como en Arduino • STM32 son versátiles y muy configurable.

	obtener rápidos los datos de un sensor de una sola actividad	

En la tabla 3, observamos algunas de las ventajas que se tomó en cuenta a la hora de elegir Raspberry Pi para nuestro proyecto.

Tabla 5. Ventajas Raspberry Pi4.

Raspberry Pi	
Ventajas	Aborda varios lenguajes de desarrollo, aunque Python y JavaScript es el más familiar para esta marca.
	Desplegar aplicaciones de diferentes tipos, de cualquier lenguaje de programación Python, Java, JavaScript, JSON, etc.
	Podemos montar back end y front end, haciendo que varias Raspberry se comuniquen, utilizando como servidores en la nube pudiendo desplegar aplicaciones en un mundo profesional.
	Se puede montar stack (arquitectura de aplicaciones) ya sea desde un front end o back end.

1.14. Geo-Posicionamiento

Para entender el concepto de Geo Posicionamiento debemos traducir su postura semántica que conforma esta palabra “Geo” y “Posicionamiento”. Por Posicionamiento podemos entender como la determinación de espacio de distintos objetos móviles o estacionarios, estos pueden estar determinados por un sistema de coordenadas o también en relación a un punto, tomando a este como origen de un sistema de coordenadas. Según la academia real española “Geo” significa “tierra”.[36]

Partiendo de estos dos conceptos podemos traducir a Geo-Posicionamiento, como un método para identificar o localizar un objeto en espacio y tiempo en la tierra. Para este proyecto utilizaremos el Sistema de Posicionamiento Global (GPS), este nos ayuda a determinar la ubicación en la tierra el objeto que utilizaremos para rastrear la trayectoria, en nuestro caso, el transporte público. El GPS este

compuesto por una red de 24 satélites que orbitan con trayectorias sincronizadas alrededor del planeta Tierra. El funcionamiento del GPS se basa en determinar la posición de un receptor en la superficie terrestre. Sabemos que este receptor está ubicado en algún lugar de la superficie terrestre, en el medio del satélite ya una distancia de radio del receptor. Al obtener información sobre este proceso en dos satélites, se puede determinar la circunferencia obtenida cuando las dos esferas se cruzan en la ubicación del receptor. Al obtener las coordenadas de un tercer satélite el cual borra la falta de sincronización de los relojes de los receptores GPS, como los relojes de satélites, podemos determinar una posición exacta en longitud, latitud, altitud del objeto que estamos rastreando.[37]

El módulo receptor que elegimos para el hardware de nuestro sistema es el "GPS L80-M39", esta gama de GPS es líder mundial en el suministro de tecnologías de posicionamiento y semiconductores inalámbricos, tienen una buena confiabilidad a un coste asequible.[36]

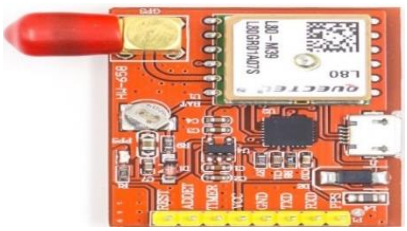


Ilustración 13. Modulo convertidor a USB – GPS L80-M3. (Fuente: https://www.mouser.ec/)

Tabla 6. Especificaciones técnicas GPS L80-M39.

GPS L80-M39	
Especificaciones	
Brand	Quectel
Números de Canales	26
Sensibilidad de seguimiento	-165dBm
Temperatura máxima de funcionamiento	+85ºC
Interfaces de bus	UART
Sensibilidad de adquisición	-148 dBm

Temperatura de funcionamiento mínimo	-40ºC
Dimensiones	16.6*26*18.6 mm.

CAPÍTULO 2

2. Marco Metodológico

Realizar nuestro proyecto requiere: estimar los pasajeros dentro de las unidades de transporte, determinar la ubicación de las unidades en tiempo real, enviar la información a nuestro servidor web, visualizar la información mediante una interfaz web, entrenar los datos recolectados para implementar un algoritmo de predicción de pasajeros.

Para cumplir con lo que requiere el proyecto necesitamos de: una raspberry, servidor web, interfaz web y algoritmo de predicción.

2.1. Arquitectura

Proponemos la arquitectura como se muestra en la ilustración 14, observamos que el proyecto se divide en 4 etapas:

- Sistema de adquisición: recolecta la información para enviar al servidor web
- Back End: Realiza toda la parte lógica y se encarga del funcionamiento de la aplicación
- Front End: se encarga de la interfaz y el diseño de una página web, también permite la comunicación del usuario con el Back-end
- Sistema de predicción: A partir de los datos recolectados entrena la red neuronal LSTM para la predicción de pasajeros.

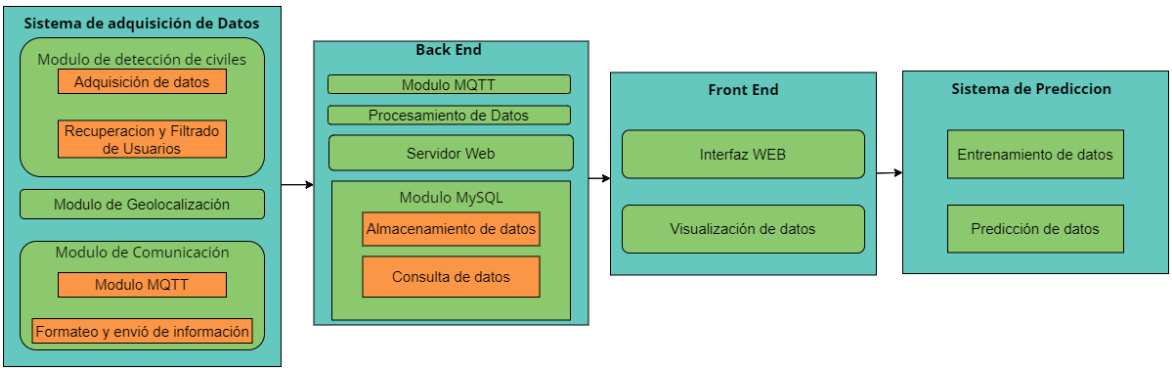


Ilustración 14. Arquitectura General del Proyecto. (Fuente: Autor)

En la arquitectura de la Ilustración 15 observamos una visión general del flujo de la información que seguirá este proyecto, detallamos cada uno de los protocolos, tecnologías y dispositivos que intervienen en la implementación del mismo. El flujo de información se divide en las siguientes etapas:

- Adquisición de datos: Escanea las PR y envía la estimación de usuarios y ubicación de las unidades de transporte por medio del protocolo MQTT.
- Procesamiento de datos: Recupera los datos enviados en formato JSON, luego convierte de formato JSON en formato que pueda ser entendido por el usuario.
- Almacenamiento de datos: Almacena la información procesada en una base datos.

- Servidor WEB: Permite interactuar al usuario con el back end por medio del protocolo http.
- Visualización de datos: Es la interfaz web de la aplicación, muestra: grafica de la información recolectada, grafica en tiempo real, grafica de la predicción del flujo de civiles.

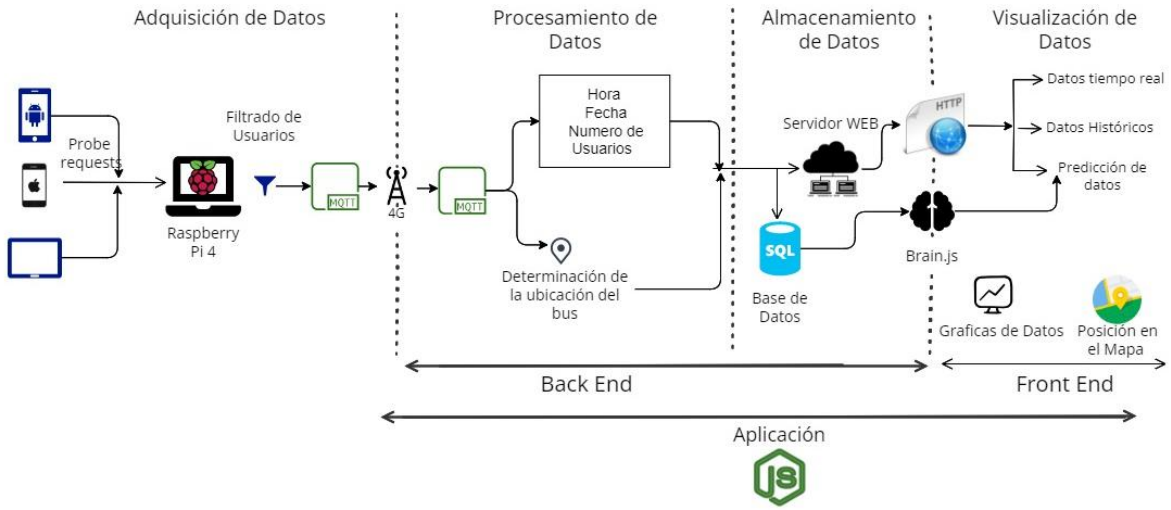


Ilustración 15. Flujo de información del proyecto. (Fuente: Autor)

2.2. Sistema de Adquisición de Datos

El proyecto desarrollara un dispositivo de adquisición de datos capaz de detectar civiles, adquirir coordenadas y enviar estos datos al servidor en tiempo real.

2.2.1. Modulo Detección de Civiles

Este módulo se encarga de detectar a los civiles dentro de las unidades de transporte urbano, realiza el escaneo y filtrado, obteniendo una estimación de usuarios en tiempo real.

2.2.1.1 Construcción del Modulo

El dispositivo consta de un par de baterías de litio, cuya duración dependerá de la cantidad de recursos computacionales desde la raspberry, pero en el caso de este proyecto se estima que la duración media será de tres horas. En la raspberry tenemos tres periféricos conectados: modem 4G, modulo GPS, adaptador WIFI-USB, como se observa en la ilustración 16.

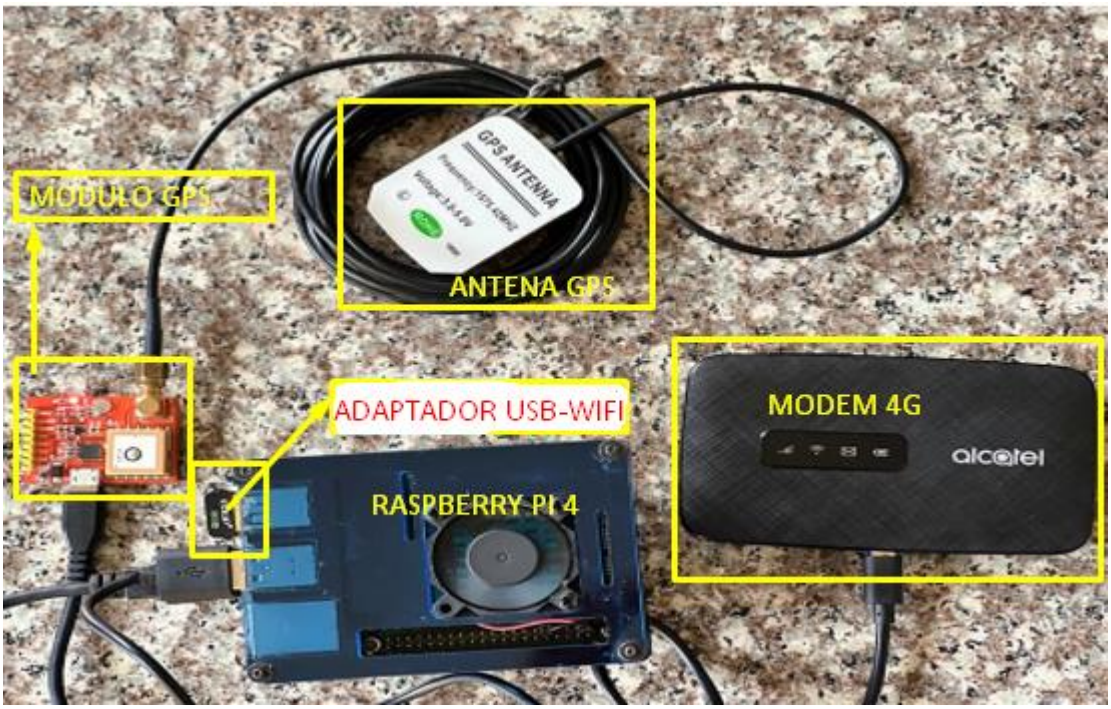


Ilustración 16. Raspberry Pi4, Modem 4G, GPS, Adaptador USB – WIFI. (Fuente: Autor).

El módulo GPS proporciona la ubicación, latitud y longitud, del dispositivo en todo momento para su futura localización y visualización.

El adaptador WIFI se establece en modo monitor para que realice el escaneo de las PR, obteniendo la dirección MAC y el RSSI de los dispositivos móviles. El RSSI posteriormente nos ayudara a filtrar a los usuarios en función de la distancia.

El modem 4G sirve para que la raspberry tenga conexión a internet mediante la red 4G para transmitir los datos recolectados en ese instante, para mayor información de las especificaciones técnicas visitar: https://www.alcatelmobile.com/wp-content/uploads/2019/07/MW40V_UM_ES-RED-V01-171019.pdf

Todos los periféricos junto con la raspberry han sido colocados dentro de una caja diseñada para que se pueda sujetar en cualquier superficie fácilmente, como se muestra en la ilustración 17.

Ilustración 17. Raspberry junto a los diferentes periféricos dentro de la caja. (Fuente: Autor)

2.2.1.2 Adquisición de Datos (Recepción de solicitudes de sondeo PR)

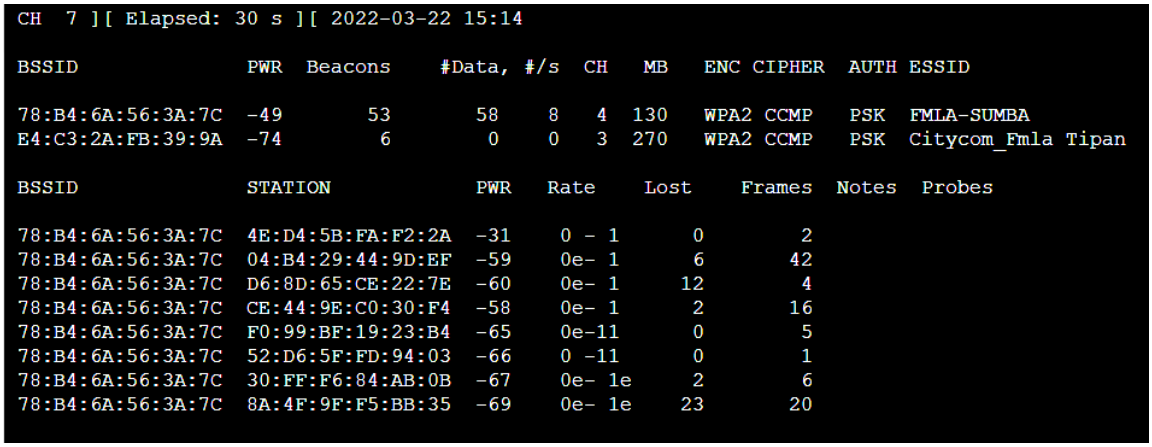
En la adquisición de datos la raspberry recibe las solicitudes de sondeo de los celulares, hemos instalado en la raspberry el sistema operativo Kali Linux. Esta distribución ha sido diseñada para temas de seguridad como; análisis de redes, ataques inalámbricos, etc. Proporciona las herramientas necesarias para realizar un análisis de las PR de usuarios con teléfonos móviles que tengan el WIFI activado. En el apéndice A se detalla la instalación del sistema operativo Kali Linux.

La herramienta de Kali que se utilizo es Aircrack-ng, esta a su vez tiene un conjunto completo de herramientas para diagnosticar la seguridad de la red WIFI. En este caso se usa para obtener las direcciones MAC de los usuarios mediante las PR.

Ilustración 18. Aircrack-ng. (Fuente: https://www.aircrack-ng.org)

Colocamos el adaptador USB WIFI en modo monitor para que pueda realizar el escaneo, después iniciamos Aircrack-ng y empezamos a recibir las PR de dispositivos móviles. En el apéndice A se muestra los comandos necesarios para colocar el adaptador USB WIFI en modo monitor.

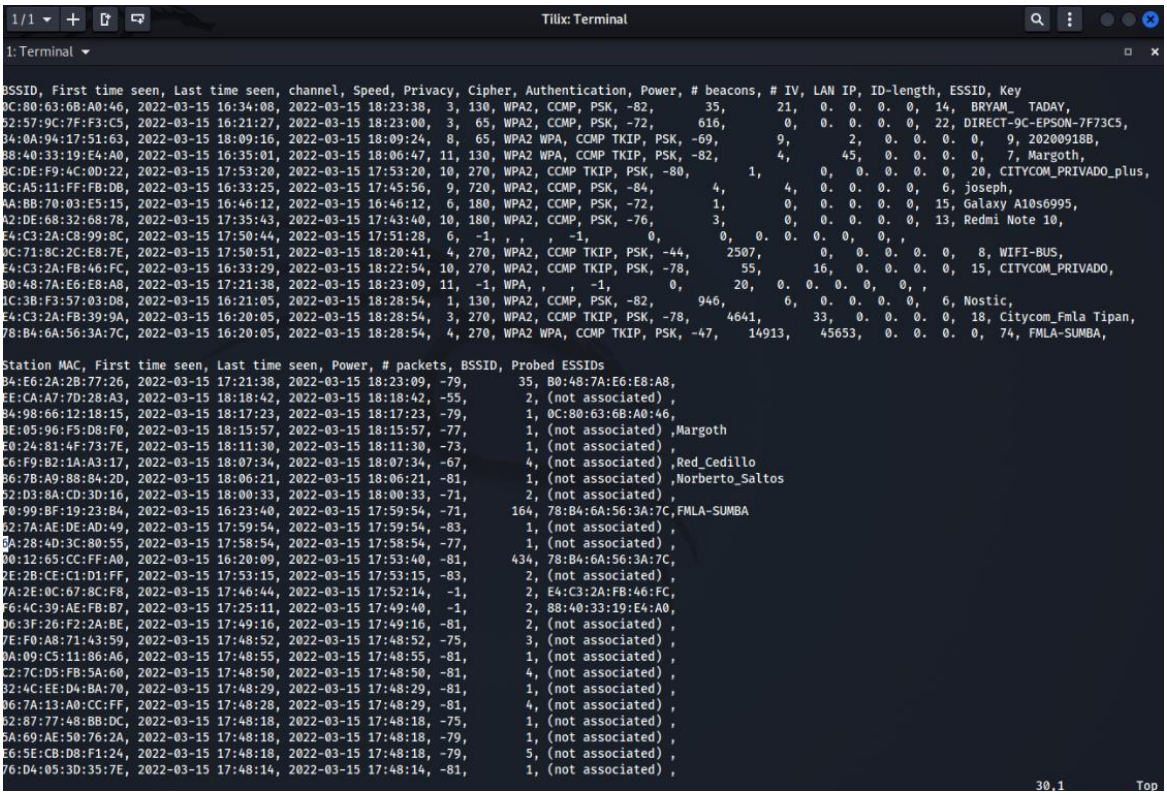
En la ilustración 19 en la parte superior se muestra la lista de AP detectados y en la parte inferior una lista de PR detectados, escaneamos los canales 1 a 14 que corresponden a la banda 2.4GHz.



```
CH  7 ][ Elapsed: 30 s ][ 2022-03-22 15:14

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

78:B4:6A:56:3A:7C  -49       53       58    8   4  130   WPA2 CCMP   PSK  FMLA-SUMBA
E4:C3:2A:FB:39:9A  -74        6        0    0   3  270   WPA2 CCMP   PSK  Citycom_Fmla Tipan

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

78:B4:6A:56:3A:7C  4E:D4:5B:FA:F2:2A  -31    0 - 1      0        2
78:B4:6A:56:3A:7C  04:B4:29:44:9D:EF  -59    0e- 1      6       42
78:B4:6A:56:3A:7C  D6:8D:65:CE:22:7E  -60    0e- 1     12        4
78:B4:6A:56:3A:7C  CE:44:9E:C0:30:F4  -58    0e- 1      2       16
78:B4:6A:56:3A:7C  F0:99:BF:19:23:B4  -65    0e-11      0        5
78:B4:6A:56:3A:7C  52:D6:5F:FD:94:03  -66    0 -11      0        1
78:B4:6A:56:3A:7C  30:FF:F6:84:AB:0B  -67    0e- 1e     2        6
78:B4:6A:56:3A:7C  8A:4F:9F:F5:BB:35  -69    0e- 1e    23       20
```

Ilustración 19. AP y PR escaneados por la raspberry. (Fuente: Autor)

Estos datos son guardados en un documento de formato csv, esto para posteriormente filtrar de las PR las direcciones MAC y el RSSI correspondiente. En la ilustración 20 se observa el documento csv, nuevamente se puede observar en la parte superior información de los AP y en la parte inferior información que contiene las PR. La raspberry guarda todas las PR desde que empieza a escanear por lo que es necesario borrar el documento y volver a crear otro para evitar contabilizar usuarios que ya no se encuentren físicamente en el lugar del escaneo.



```
Tilix: Terminal
1: Terminal

BSSID, First time seen, Last time seen, channel, Speed, Privacy, Cipher, Authentication, Power, # beacons, # IV, LAN IP, ID-length, ESSID, Key
0C:80:63:6B:A0:46, 2022-03-15 16:34:08, 2022-03-15 18:23:38,  3, 130, WPA2, CCMP, PSK, -82,      35,      21,  0.  0.  0.  0,  14,  BRYAM_  TADAY,
52:57:9C:7F:F3:C5, 2022-03-15 16:21:27, 2022-03-15 18:23:00,  3,  65, WPA2, CCMP, PSK, -72,     616,       0,  0.  0.  0.  0,  22, DIRECT-9C-EPSON-7F73C5,
34:0A:94:17:51:63, 2022-03-15 18:09:16, 2022-03-15 18:09:24,  8,  65, WPA2 WPA, CCMP TKIP, PSK, -69,       9,       2,  0.  0.  0.  0,   9, 20200918B,
88:40:33:19:E4:A0, 2022-03-15 16:35:01, 2022-03-15 18:06:47, 11, 130, WPA2 WPA, CCMP TKIP, PSK, -82,       4,      45,  0.  0.  0.  0,   7, Margoth,
8C:DE:F9:4C:0D:22, 2022-03-15 17:53:20, 2022-03-15 17:53:20, 10, 270, WPA2, CCMP TKIP, PSK, -80,       1,       0,  0.  0.  0.  0,  20, CITYCOM_PRIVADO_plus,
BC:A5:11:FF:FB:DB, 2022-03-15 16:33:25, 2022-03-15 17:45:56,  9, 720, WPA2, CCMP, PSK, -84,       4,       4,  0.  0.  0.  0,   6, joseph,
AA:BB:70:03:E5:15, 2022-03-15 16:46:12, 2022-03-15 16:46:12,  6, 180, WPA2, CCMP, PSK, -72,       1,       0,  0.  0.  0.  0,  15, Galaxy A10s6995,
A2:DE:68:32:68:78, 2022-03-15 17:35:43, 2022-03-15 17:43:40, 10, 180, WPA2, CCMP, PSK, -76,       3,       0,  0.  0.  0.  0,  13, Redmi Note 10,
E4:C3:2A:C8:99:8C, 2022-03-15 17:50:44, 2022-03-15 17:51:28,  6,  -1, , ,  , -1,       0,       0,  0.  0.  0.  0,   0, ,
0C:71:8C:2C:E8:7E, 2022-03-15 17:50:51, 2022-03-15 18:20:41,  4, 270, WPA2, CCMP TKIP, PSK, -44,    2507,       0,  0.  0.  0.  0,   8, WIFI-BUS,
E4:C3:2A:FB:46:FC, 2022-03-15 16:33:29, 2022-03-15 18:22:54, 10, 270, WPA2, CCMP TKIP, PSK, -78,      55,      16,  0.  0.  0.  0,  15, CITYCOM_PRIVADO,
B0:48:7A:E6:E8:A8, 2022-03-15 17:21:38, 2022-03-15 18:23:09, 11,  -1, WPA, ,  , -1,       0,      20,  0.  0.  0.  0,   0, ,
1C:3B:F3:57:03:D8, 2022-03-15 16:21:05, 2022-03-15 18:28:54,  1, 130, WPA2, CCMP, PSK, -82,     946,       6,  0.  0.  0.  0,   6, Nostic,
E4:C3:2A:FB:39:9A, 2022-03-15 16:20:05, 2022-03-15 18:28:54,  3, 270, WPA2, CCMP TKIP, PSK, -78,    4641,      33,  0.  0.  0.  0,  18, Citycom_Fmla Tipan,
78:B4:6A:56:3A:7C, 2022-03-15 16:20:05, 2022-03-15 18:28:54,  4, 270, WPA2 WPA, CCMP TKIP, PSK, -47,   14913,   45653,  0.  0.  0.  0,  74, FMLA-SUMBA,

Station MAC, First time seen, Last time seen, Power, # packets, BSSID, Probed ESSIDs
B4:E6:2A:2B:77:26, 2022-03-15 17:21:38, 2022-03-15 18:23:09, -79,      35, B0:48:7A:E6:E8:A8,
EE:CA:A7:7D:28:A3, 2022-03-15 18:18:42, 2022-03-15 18:18:42, -55,       2, (not associated) ,
34:98:66:12:18:15, 2022-03-15 18:17:23, 2022-03-15 18:17:23, -79,       1, 0C:80:63:6B:A0:46,
BE:05:96:F5:D8:F0, 2022-03-15 18:15:57, 2022-03-15 18:15:57, -77,       1, (not associated) ,Margoth
E0:24:81:4F:73:7E, 2022-03-15 18:11:30, 2022-03-15 18:11:30, -73,       1, (not associated) ,
C6:F9:B2:1A:A3:17, 2022-03-15 18:07:34, 2022-03-15 18:07:34, -67,       4, (not associated) ,Red_Cedillo
B6:7B:A9:88:84:2D, 2022-03-15 18:06:21, 2022-03-15 18:06:21, -81,       1, (not associated) ,Norberto_Saltos
52:D3:8A:CD:3D:16, 2022-03-15 18:00:33, 2022-03-15 18:00:33, -71,       2, (not associated) ,
F0:99:BF:19:23:B4, 2022-03-15 16:23:40, 2022-03-15 17:59:54, -71,     164, 78:B4:6A:56:3A:7C,FMLA-SUMBA
52:7A:AE:DE:AD:49, 2022-03-15 17:59:54, 2022-03-15 17:59:54, -83,       1, (not associated) ,
6A:28:4D:3C:80:55, 2022-03-15 17:58:54, 2022-03-15 17:58:54, -77,       1, (not associated) ,
00:12:65:CC:FF:A0, 2022-03-15 16:20:09, 2022-03-15 17:53:40, -81,     434, 78:B4:6A:56:3A:7C,
2E:2B:CE:C1:D1:FF, 2022-03-15 17:53:15, 2022-03-15 17:53:15, -83,       2, (not associated) ,
7A:2E:0C:67:8C:F8, 2022-03-15 17:46:44, 2022-03-15 17:52:14,  -1,       2, E4:C3:2A:FB:46:FC,
F6:4C:39:AE:FB:B7, 2022-03-15 17:25:11, 2022-03-15 17:49:40,  -1,       2, 88:40:33:19:E4:A0,
D6:3F:26:F2:2A:BE, 2022-03-15 17:49:16, 2022-03-15 17:49:16, -81,       2, (not associated) ,
7E:F0:A8:71:43:59, 2022-03-15 17:48:52, 2022-03-15 17:48:52, -75,       3, (not associated) ,
0A:09:C5:11:86:A6, 2022-03-15 17:48:55, 2022-03-15 17:48:55, -81,       1, (not associated) ,
C2:7C:D5:FB:5A:60, 2022-03-15 17:48:50, 2022-03-15 17:48:50, -81,       4, (not associated) ,
32:4C:EE:D4:BA:70, 2022-03-15 17:48:29, 2022-03-15 17:48:29, -81,       1, (not associated) ,
D6:7A:13:A0:CC:FF, 2022-03-15 17:48:28, 2022-03-15 17:48:29, -81,       4, (not associated) ,
62:87:77:48:BB:DC, 2022-03-15 17:48:18, 2022-03-15 17:48:18, -75,       1, (not associated) ,
6A:69:AE:50:76:2A, 2022-03-15 17:48:18, 2022-03-15 17:48:18, -79,       1, (not associated) ,
E6:5E:CB:D8:F1:24, 2022-03-15 17:48:18, 2022-03-15 17:48:18, -79,       5, (not associated) ,
76:D4:05:3D:35:7E, 2022-03-15 17:48:14, 2022-03-15 17:48:14, -81,       1, (not associated) ,
                                                                                    30,1          Top
```

Ilustración 20. Documento con información de las PR en formato csv. (Fuente: Autor)

En el diagrama de flujo de la ilustración 21 se detalla el proceso de adquisición de datos, se elimina y se vuelve a crear el archivo csv.

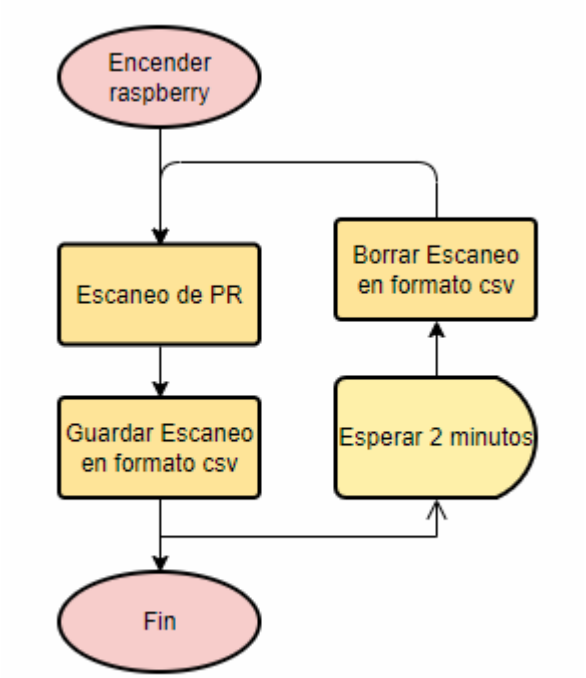


Ilustración 21. Diagrama de flujo de recepción de solicitudes de sondeo (PR). (Fuente: Autor)

2.2.1.3 Recuperación y Filtrado de Usuarios

Después de generar el documento csv eliminamos información redundante y obtenemos solo las PR de dispositivos móviles, luego filtramos para obtener únicamente a los usuarios dentro de un rango de 10 metros. Esta etapa requiere de 5 procesos como se observa en la ilustración 22.

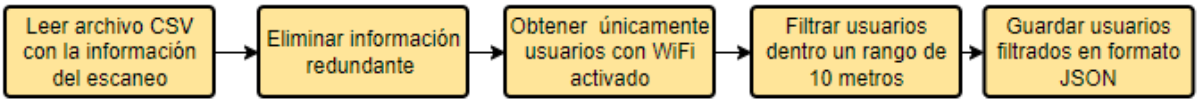


Ilustración 22. Diagrama de bloques de la recuperación y filtrado de usuarios. (Fuente: Autor)

En la ilustración 23 observamos la información que contiene las PR, detallamos la información de estas columnas a continuación:

- Station MAC: contiene las direcciones MAC de las estaciones móviles
- First time seen: contiene fecha y hora de la primera vez que se escanea el dispositivo móvil
- Last time seen: contiene fecha y hora de la última vez que se escanea el dispositivo móvil
- Paquetes: contiene el número de paquetes enviados entre el AP y el dispositivo móvil
- Power: contiene el RSSI

En el apéndice A se detalla la programación para la recuperación de los usuarios con WIFI activado.

Out[6]:

	Station MAC	First time seen	Last time seen	# beacons	# IV	paquetes	Power
17	C4:AD:34:9B:D5:DC	2021-08-31 16:17:33	2021-08-31 16:37:56	NaN	NaN	335.0	-1.0
18	4C:5E:0C:2F:EC:CF	2021-08-31 16:17:00	2021-08-31 16:37:53	NaN	NaN	589.0	-1.0
19	64:D1:54:7D:F1:B7	2021-08-31 16:17:03	2021-08-31 16:37:56	NaN	NaN	185.0	-55.0
20	C4:98:5C:FC:F0:93	2021-08-31 16:30:37	2021-08-31 16:37:50	NaN	NaN	3.0	-55.0
21	DA:A1:19:D0:9D:5E	2021-08-31 16:36:36	2021-08-31 16:36:36	NaN	NaN	3.0	-79.0
22	60:AB:14:0F:4D:86	2021-08-31 16:35:21	2021-08-31 16:35:21	NaN	NaN	1.0	-70.0
23	0E:B0:03:92:DB:CA	2021-08-31 16:31:24	2021-08-31 16:31:24	NaN	NaN	1.0	-55.0
24	08:78:08:35:83:E4	2021-08-31 16:18:12	2021-08-31 16:18:12	NaN	NaN	1.0	-55.0
25	F0:76:6F:72:B3:7F	2021-08-31 16:18:12	2021-08-31 16:18:12	NaN	NaN	2.0	-78.0
26	F0:A3:B2:C5:26:B1	2021-08-31 16:28:09	2021-08-31 16:31:53	NaN	NaN	3.0	-76.0
27	04:B4:29:44:9D:EF	2021-08-31 16:22:10	2021-08-31 16:33:06	NaN	NaN	51.0	-1.0
28	24:18:1D:70:1D:37	2021-08-31 16:17:57	2021-08-31 16:33:53	NaN	NaN	16.0	-72.0
29	16:93:9C:99:86:0C	2021-08-31 16:34:07	2021-08-31 16:34:07	NaN	NaN	1.0	-72.0
30	64:D1:54:0C:47:AF	2021-08-31 16:19:17	2021-08-31 16:36:07	NaN	NaN	48.0	-88.0
31	AA:4F:9F:F5:BB:35	2021-08-31 16:23:24	2021-08-31 16:36:05	NaN	NaN	80.0	-59.0
32	30:FF:F6:84:AB:0B	2021-08-31 16:16:57	2021-08-31 16:37:34	NaN	NaN	25.0	-62.0
33	50:50:A4:7E:4B:B2	2021-08-31 16:16:57	2021-08-31 16:37:34	NaN	NaN	1407.0	-37.0
34	A4:C3:F0:57:EE:4B	2021-08-31 16:17:54	2021-08-31 16:39:55	NaN	NaN	26.0	-81.0

Ilustración 23. Usuarios con WIFI activado. (Fuente: Autor)

Utilizaremos el método llamado Receiver Signal Strenght Indicator (RSSI), el cual nos indica la perdida de energía en el proceso de trasmisión de señales. Este método nos ayudara a determinar la atenuación de la señal, por tanto, el rango de trasmisión que cubriremos con la Raspberry hacia los nodos o dispositivos móviles. Este rango o distancia está asociada directamente a la longitud que mide la unidad de transporte en donde será incorporado este proyecto. La unidad seleccionados para el experimento será un transporte urbano, que de acuerdo con la imagen que se muestra a continuación posee una distancia de entre 10 a 11 metros de longitud.

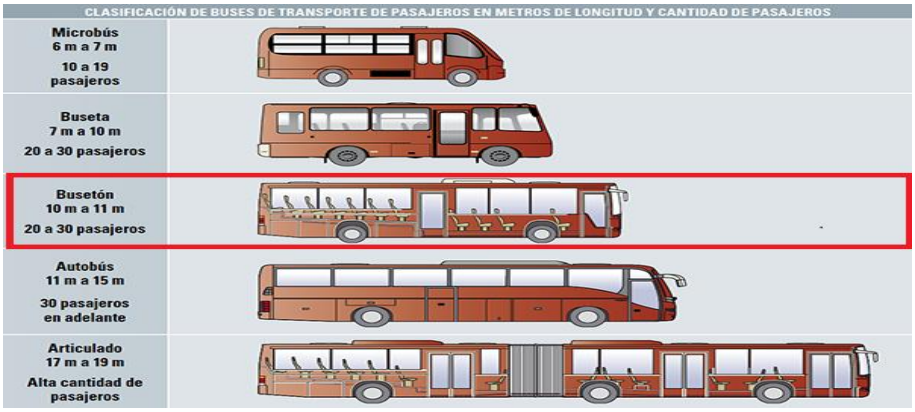


Ilustración 24. Clasificación de transporte urbanos con sus diferentes longitudes. (Fuente: https://www.revistaautocrash.com/)

El método RSSI determina que la perdida de la propagación de la señal recibida por un dispositivo en una distancia d_i hacia el nodo trasmisor es igual a:

$$P_L(d_i)[dB] = P_L(d_o)[dB] + 10 * n * log_{10} * \frac{d_i}{d_o} \quad (1)$$

Donde:

$P_L(d_o)$ hace referencia a la perdida de en una distancia conocida que generalmente es de 1 metro.

n es una constante que identifica los obstáculos que existe entre los nodos dependiendo también del medio, generalmente en el espacio libre es de $n = 2$.

La señal en un entorno se ve afectado por diferentes fenómenos como: reflexión, dispersión o difracción, para ello determinaremos el valor de n empiricamente con un método experimental RSSI.

De la ecuación (1) podemos obtener lo siguiente.

$$n = \frac{P_L(d_i) - P_L(d_o)}{10 * log_{10} * \frac{d_i}{d_o}} \quad (2)$$

Esta ecuación nos permite experimentar el valor de la constante n a partir de la medición de potencia entre la distancia que los separa a los dispositivos.

Antes de realizar este método experimental, la potencia de la señal recibida está definida por la siguiente ecuación:

$$RSSI[dBm] = -10 * n * log_{10} * d + A \quad (3)$$

Donde: d es la distancia entre los dispositivos trasmisor y el receptor, A es el valor de RSSI entre el dispositivo receptor a 1 metro de distancia con el transmisor.

Despejando obtendríamos:

$$d = 10^{-\frac{RSSI-A}{10n}} \quad (4)$$

Este proceso para la medición, hecho referencia en la ilustración 25, consiste en fijar un dispositivo móvil a una distancia preestablecida de la Raspberry, luego, usar esta para medir la intensidad de señal recibida en dBm para su posterior almacenamiento y análisis.

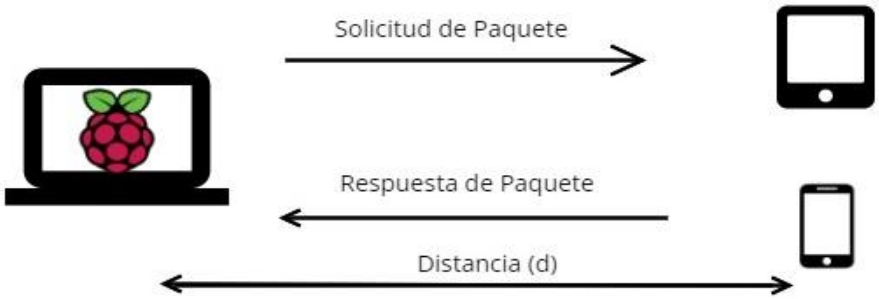


Ilustración 25. Diagrama de conexiones utilizados. (Fuente: Autor)

Se registraron un total de 6 valores de RSSI, cada 5 metros de distancia, en un rango de 10 metros, durante intervalos de 1 minuto para la toma/adquisición de datos.

1. Para el inicio de este método, se mide el valor de RSSI a una distancia de 1 metro aplicando la formula (2).

Tabla 5. Medición para el parámetro A (dBm). .

Distancia (Metros)	RSSI (dBm)								
1	-40	-40	-40	-40	-40	-40	-40	-40	-40

El método nos arroja un valor de $A = -40\ dBm.$

Tabla 6. Valores de RSSI medidos para diferentes distancias de separación entre los dispositivos.

Distancia (Metros)	2	3	4	5	6	7	8	9	10
RSSI (dbm)	-45	-55	-57	-57	-59	-67	-67	-69	-67
	-45	-55	-57	-57	-59	-67	-67	-69	-67
	-45	-55	-57	-57	-59	-67	-67	-69	-67
	-45	-52	-57	-57	-59	-67	-67	-69	-67
	-45	-52	-57	-57	-59	-69	-69	-69	-67
	-45	-52	-55	-57	-59	-69	-69	-69	-67

2. Para las siguientes mediciones de RSSI se obtiene los siguientes valores.

Obteniendo el valor promedio del RSSI con las diferentes distancias, aplicando la ecuación (2) obtenemos los siguientes datos calculados para n.

Tabla 7. Datos calculados para n con las diferentes distancias.

Distancia (Metros)	2	3	4	5	6	7	8	9	10
n	1.66	2.82	2.76	2.43	2.44	2.24	3.06	3.03	2.7

A partir de la Tabla 7, realizamos un promedio entre los datos calculados para obtener un valor empírico de n con este método experimental**.** En promedio resulta $n = 2{,}5759$.

Conociendo los datos de las constantes es posible calcular los valores de las distancias en función de los datos RSSI utilizando la formula (3), también podemos utilizar la formula (4) para obtener una distancia teórica en función de los valores de RSSI obtenidos a partir de la formula (3).

Tabla 8. RSSI medido promedio, distancia teórica, errores calculados a partir de las mediciones.

Distancia real	RSSI medido promedio	Distancia Teórica	Error Absoluto	Error Relativo
2	-45	1,563522941	0,436477059	0,218238529
3	-53,5	3,342577584	0,342577584	0,114192528
4	-56,6666667	4,43623143	0,43623143	0,109057858
5	-57	4,570402799	0,429597201	0,08591944
6	-59	5,465075677	0,534924323	0,089154054
7	-59	5,465075677	1,534924323	0,219274903
8	-67,6666667	11,85887693	3,858876926	0,482359616
9	-69	13,35994579	4,359945793	0,484438421
10	-67	11,17282491	1,172824907	0,117282491

- En la ilustración 26 después de filtrar los usuarios redundantes observamos que nos quedamos con 4 usuarios que se encuentran dentro de la distancia establecida. Revisar el apéndice A donde se detalla la programación para el filtrado de usuarios.

Out[26]:

	Station_MAC	Date_Ft	Time_Ft	Date_Lt	Time_Ft	Power	usuarios
103	DA:58:20:1E:00:5A	2022-03-15	18:13:59	2022-03-15	18:13:59	-53.0	4.0
118	00:12:36:B7:E8:49	2022-03-15	16:43:01	2022-03-15	18:27:55	-54.0	4.0
120	4E:D4:5B:FA:F2:2A	2022-03-15	16:20:20	2022-03-15	18:28:54	-41.0	4.0
121	78:0C:B8:A6:E9:51	2022-03-15	16:20:17	2022-03-15	18:28:40	-37.0	4.0

Ilustración 26. Usuarios Filtrados. (Fuente: Autor)

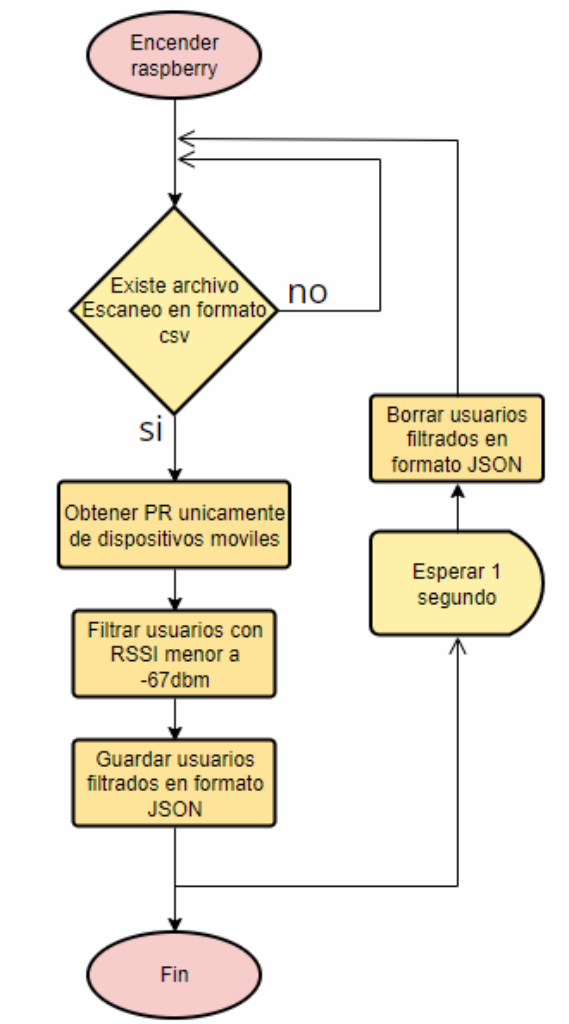


Ilustración 27. Diagrama de flujo para la recuperación y filtrado de usuarios. (Fuente: Autor)

2.2.2. Módulo de Geolocalización

Para obtener la geolocalización del transporte público en tiempo real necesitamos, la latitud y longitud con el módulo GPS conectado a la raspberry obtenemos las coordenadas, para esto realizamos un script de Python que nos devuelve las coordenadas como se observa en la ilustración 28. En el apéndice A se detalla la programación para obtener la ubicación.

```
obteniendo datos ...
obteniendo datos ...
    latitud   longitud hora_ubicacion
0 -2.561637 -78.941357       18:03:49
```

Ilustración 28. Datos obtenidos de la Ubicación. (Fuente: Autor)

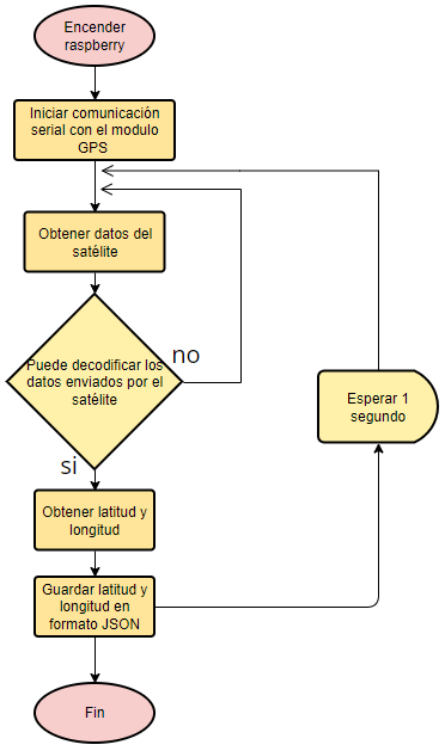


Ilustración 29. Diagrama de flujo del módulo de geolocalización. (Fuente: Autor)

2.2.3. Módulo de Comunicación

Utilizamos el protocolo MQTT para él envió de la información, para tener acceso a internet usamos la red 4G mediante el modem que está conectado a nuestra raspberry.

2.2.3.1 Modulo MQTT

Para este proyecto vamos a utilizar la librería “paho-mqtt” que utiliza lenguaje de programación Python permitiendo que las aplicaciones se conecten a un broker MQTT para publicar mensajes y subscribirse a temas para recibir mensajes publicados, usamos el bróker gratuito ‘broker.emqx.io’. La raspberry va a ser la encargada de publicar mientras que el servidor será el subscriptor.

2.2.3.2 Formateo y envió de información

La información recolectada de los civiles y las coordenadas han sido vinculadas y guardadas en formato JSON, el diagrama de flujo de la ilustración 32 detalla el proceso de vinculación y publicación de la información. En la ilustración 30 observamos toda la información vinculada: latitud, longitud, hora, número de usuarios, weekday (hace referencia al día de la semana; 0 lunes, 4 viernes), potencia (RSSI), la información está lista para ser publicada por el protocolo MQTT. En el apéndice A se detalla la programación de la vinculación de la información.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3.9 informacion.py
   Unnamed: 0   latitud   longitud hora_ubicacion  Unnamed: 0      hora  Unnamed: 0  usuarios  weekday  potencia       fecha
0           0 -2.567315 -78.937745       12:59:26           0  16:50:31           0         1        1     -57.0  2022-06-28
```

Ilustración 30. Vinculación de datos. (Fuente: Autor)

El tópico con el que se publicará la información se llama “python/bus_canar/usuarios” en la ilustración 31 se muestra cómo ha sido publicada la información en el tópico correspondiente. Revisar el apéndice A donde se detalla la programación para la publicación de la información.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3.9 pub_usuarios.py
{'"{\\"Unnamed: 0\\":{\\"0\\":0},\\"Unnamed: 0.1\\":{\\"0\\":0},\\"latitud\\":{\\"0\\":-2.1166833333},\\"
longitud\\":{\\"0\\":-85.1079716667},\\"hora_ubicacion\\":{\\"0\\":\\"17:09:32\\"},\\"Unnamed: 0.2\\":{\\
"0\\":0},\\"hora\\":{\\"0\\":\\"17:10:50\\"},\\"Unnamed: 0.3\\":{\\"0\\":0},\\"usuarios\\":{\\"0\\":2},\\
"weekday\\":{\\"0\\":1},\\"potencia\\":{\\"0\\":-55.0},\\"fecha\\":{\\"0\\":\\"2022-06-28\\"},\\"hora_usu
arios\\":{\\"0\\":\\"17:10:47\\"}}"'}  to topic {'python/bus_canar/usuarios1'}
```

Ilustración 31. Publicación de la información en formato JSON. (Fuente: Autor)

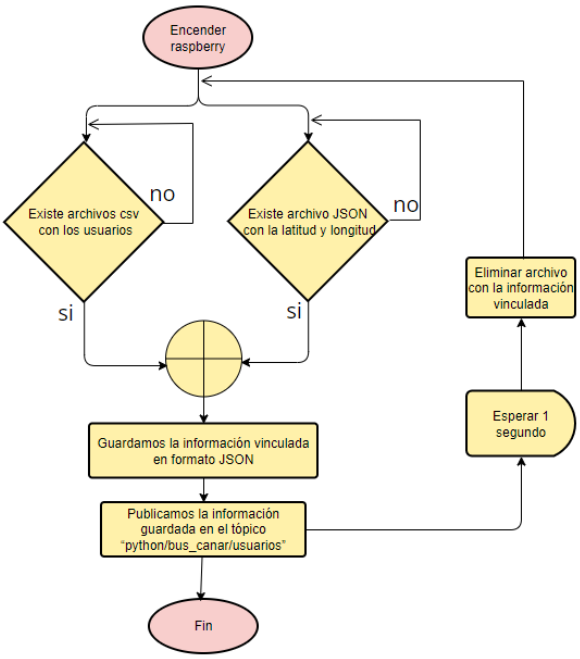


Ilustración 32. Diagrama de flujo de la vinculación y envió de la información. (Fuente: Autor)

2.3. Back END

La aplicación de este proyecto ha sido desarrollada con Node.js un entorno en ejecución de JavaScript, es decir, Node.js permite ejecutar JavaScript para crear aplicaciones de servidor.

Ilustración 33. Node.js. (Fuente: https://nodejs.org/es/)

El Back End es responsable de la lógica y el funcionamiento de nuestra Aplicación. Las operaciones lógicas que realizamos son las siguientes: consulta a la base de datos, implementación del cliente MQTT, implementación del servidor Web.

Tanto el Back end como el Front end requieren de módulos para su funcionamiento. En cada proceso de nuestro proyecto se indicará los módulos necesarios para su desarrollo. Para la instalación de los módulos utilizamos la herramienta "npm" que es un gestor de paquetes de Node.js, cuando iniciamos npm se crea automáticamente un archivo "package.json" que obtiene los datos de los paquetes requeridos, incluyendo la descripción del mismo, versión, autor y dependencias.

Aparte de Node.js necesitamos una manera de escribir el código, para este proyecto usamos el editor de código Visual Studio Code.

2.3.1. Modulo MQTT

Para recibir la información en nuestro servidor necesitamos instalar el módulo "mqtt", esta información nos llega en formato JSON que luego va a ser procesada.

- mqtt: módulo de Node.js para el protocolo MQTT, permite publicar y subscribirse a información.

En nuestra aplicación con el módulo "mqtt" instalado creamos un cliente MQTT para suscribirnos al tema ¨python/bus_canar/usuarios¨, así estamos recibiendo la información publicada por la raspberry, la ilustración 34 muestra la llegada de los mensajes publicados que contiene la siguiente información: latitud, longitud, hora, usuarios, weekday, potencia y fecha. Estos datos serán procesados para que el cliente pueda entender los mismos. Para mayor detalle de la programación para la implementación revisar el anexo B.


```
Conectado al broker
Usuarios:  {"Unnamed: 0":{"0":0},"Unnamed: 0.1":{"0":0},"latitud":{"0":-2.55},"longitud":{"0":-75.1079716667},"hora_ubicacion":{"0":"17:04:58"},"Unnam
ed: 0.2":{"0":0},"hora":{"0":"17:05:22"},"Unnamed: 0.3":{"0":0},"usuarios":{"0":2},"weekday":{"0":1},"potencia":{"0":-53.0},"fecha":{"0":"2022-06-28"}
,"hora_usuarios":{"0":"17:04:10"}}
```


Ilustración 34. Recepción de la información en la aplicación. (Fuente: Autor)

2.3.2. Procesamiento de Datos

Los datos que nos llegan en formato JSON por el protocolo MQTT, necesitan ser procesados para posteriormente ser guardados en la base de datos y también que desde el lado del cliente se pueda consumir y visualizar los datos de manera amigable para el usuario. Revisar el apéndice B donde se encuentra la programación del procesamiento de datos.

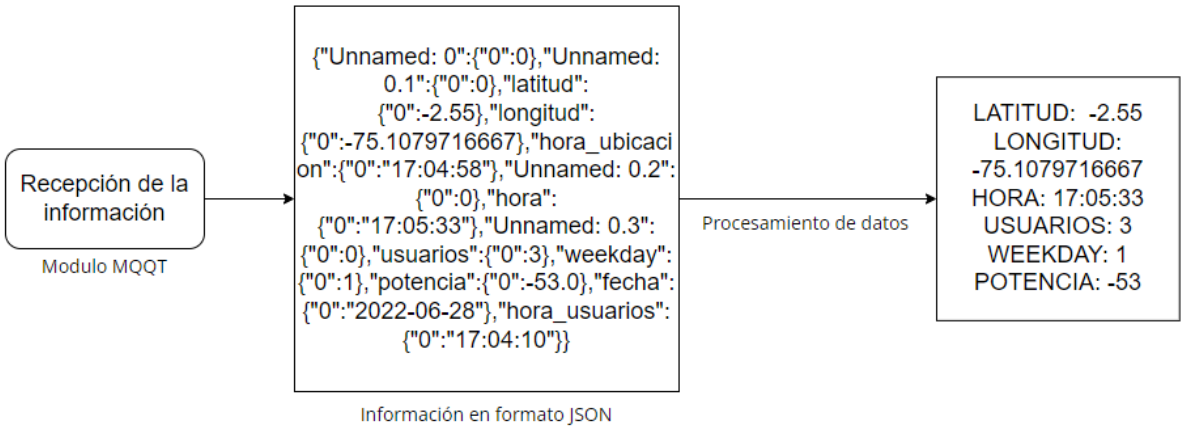


Ilustración 35. Diagrama de bloques del procesamiento de datos. (Fuente: Autor)

2.3.3. Servidor WEB

El contenido cuando recibe peticiones por parte de los usuarios. Permite que los usuarios puedan ver una página web en su navegador. El Front End es el contenido servido, el back-end se comunica con el front end a través de llamadas HTTP hechas desde el front end, levantamos el servidor del lado del back end, para esto instalamos el siguiente modulo:

- Express: permite crear aplicaciones WEB del lado del servidor. Express es un framework de terceros, escrito en JavaScript que podemos usar para poder crear nuestro servidor.

En la ilustración 36 podemos ver el servidor creado con express.

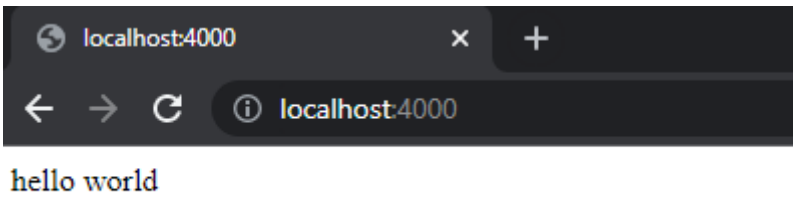


Ilustración 36. Servidor WEB corriendo en el puerto 400. (Fuente: Autor)

2.3.3.1 Peticiones HTTP (rutas)

En nuestra aplicación WEB creamos rutas mediante las peticiones HTTP (get, post, put, etc.). Con las rutas creadas el navegador puede pedir determinados recursos al servidor y con estos métodos especificar qué es lo que se quiere hacer con esos recursos. En la tabla se muestran todas las rutas creadas en esta aplicación.

Tabla 8. Rutas de petición HTTP.

Método	Ruta	Descripción
GET	http://localhost:4000/	Ruta que renderiza el inicio de nuestra página web.

GET	http://localhost:4000/info	Ruta que renderiza información acerca de nuestro proyecto, incluimos información acerca de ¨Quienes somos¨ y ¨Funcionamiento del proyecto¨
GET	http://localhost:4000/consultas	Ruta que muestra las tres opciones propuestas por esta aplicación: Datos tiempo real, datos históricos y predicción de datos. Permite al usuario escoger que datos quiere ver.
POST	http://localhost:4000/consultas	Ruta para visualizar los datos históricos, el usuario envía la fecha en la que desea ver los datos. Permite ver la gráfica del flujo de civiles en la fecha ingresada.
GET	http://localhost:4000/consultas/tiempo_real	Ruta que permite visualizar los datos en tiempo real, renderiza un mapa que muestra la posición del transporte público en tiempo real con el número aproximado de civiles en ese momento.
POST	http://localhost:4000/consultas/prediccion	Ruta para visualizar datos futuros, el usuario envía la fecha futura en la que quiere ver el flujo de civiles. Muestra la gráfica de predicción del flujo de civiles.

Revisar el apéndice B donde se detalla la implementación del servidor WEB y la programación de las rutas.

2.3.4. Modulo MySQL

El módulo que se necesita para este proceso es el siguiente:

- mysql: Modulo de Node.js, permite conectar nuestra aplicación con la base de datos instalada en nuestra máquina de escritorio y realizar consultas SQL a nuestra base de datos desde nuestra aplicación.

Revisar el apéndice B donde se detalla la programación para conectar a la base de datos con la aplicación.

2.3.4.1 Almacenamiento

Creamos nuestra base de datos en ¨MySQL Workbench¨, los datos se almacenarán en el esquema creado de nombre ¨transporte_canar¨ y dentro de este esquema se encuentra la tabla con el nombre ¨sensor2¨, la tabla tiene las siguientes columnas:

- Hora: Almacena la hora en la que los datos han sido enviados
- Fecha: Almacena la fecha en la que se envió de los datos

- Weekday: Almacena el día en el que se envió los datos en un intervalo de 0-6, siendo 0 el lunes y 6 el domingo
- Usuarios: Almacena el número de usuarios contabilizados por la raspberry
- Latitud: Almacena la latitud de la unidad de transporte
- Longitud: Almacena la longitud de la unidad de transporte

Para conectar nuestra base de datos con nuestra aplicación tenemos que realizar la respectiva conexión desde nuestra aplicación, para esto ingresamos las claves requeridas por la base de datos para tener permisos desde nuestra aplicación y realizar consultas a nuestro esquema ya creado. Las claves son: host:'localhost', port:'3306', user: 'danny', password: '*****', database: 'transporte_cañar'.

Estas claves son ingresadas con la ayuda del módulo mysql, ahora ya podemos realizar consultas a nuestra tabla ¨sensor2¨ de la base de datos ¨transporte_canar¨.

El almacenamiento de datos lo realizamos con la consulta SQL ¨INSERT INTO¨, los datos se almacenarán una vez la raspberry empiece a publicar los datos recolectados. Revisar el apéndice B donde se detalla la programación para enlazar con la base de datos y el almacenamiento en la base de datos.

usuarios	latitud	longitud	hora	fecha	weekday
3	-2.56731	-78.9377	11:32:03	2022-06-14	1
3	-2.56731	-78.9377	11:32:04	2022-06-14	1
3	-2.56731	-78.9377	11:32:06	2022-06-14	1
3	-2.56731	-78.9377	11:32:06	2022-06-14	1
3	-2.56731	-78.9377	11:32:06	2022-06-14	1
3	-2.56731	-78.9377	11:32:06	2022-06-14	1
1	-2.56731	-78.9377	11:32:16	2022-06-14	1
1	-2.56731	-78.9377	11:32:17	2022-06-14	1
1	-2.56731	-78.9377	11:32:18	2022-06-14	1
1	-2.56731	-78.9377	11:32:21	2022-06-14	1
1	-2.56731	-78.9377	11:32:21	2022-06-14	1
1	-2.56731	-78.9377	11:32:26	2022-06-14	1
1	-2.56731	-78.9377	11:32:27	2022-06-14	1
1	-2.56731	-78.9377	11:32:31	2022-06-14	1
1	-2.56731	-78.9377	11:32:31	2022-06-14	1
1	-2.56731	-78.9377	11:32:35	2022-06-14	1

Ilustración 37. Tabla ¨ sensor2¨ de la base de datos ¨transporte_canar¨ (Fuente: Autor)

2.3.4.2 Consulta de Datos

Para la consulta de datos el cliente ingresa la fecha en la que desea ver el flujo de pasajeros, la información de la fecha ingresada por el cliente llega a la aplicación por medio de la petición POST (http://localhost:4000/consultas), la aplicación mostrara por medio de la interfaz web los datos con información como se observa en la ilustración 34. La programación de la consulta de datos se detalla en el apéndice B.

2.4. Front END

El front-end de la aplicación es la interfaz de nuestro servidor web, permitiendo al usuario interactuar con el back-end, para esta aplicación instalamos los módulos necesarios para la visualización de los

datos y para que la aplicación tenga un aspecto agradable y sea intuitiva para el usuario. Para el desarrollo del front-end de la aplicación utilizamos los lenguajes HTML, CSS y JavaScript.

Los módulos requeridos para esta sección son los siguientes:

- Leafleat: Es un módulo de JavaScript es un código abierto para mapas interactivos compaginable con sistemas móviles. Con este módulo es posible desplegar un mapa y agregar la posición de la unidad de transporte en tiempo real.
- Canvas: Modulo de gráficos para aplicaciones web. Los gráficos son interactivos, dinámicos, y permite actualizaciones en tiempo real. Permite agregar cientos de miles de puntos de datos sin causar retrasos en el rendimiento.

2.4.1. Interfaz WEB

La estructura básica de la Interfaz WEB se observa en el maquetado de la ilustración 38.

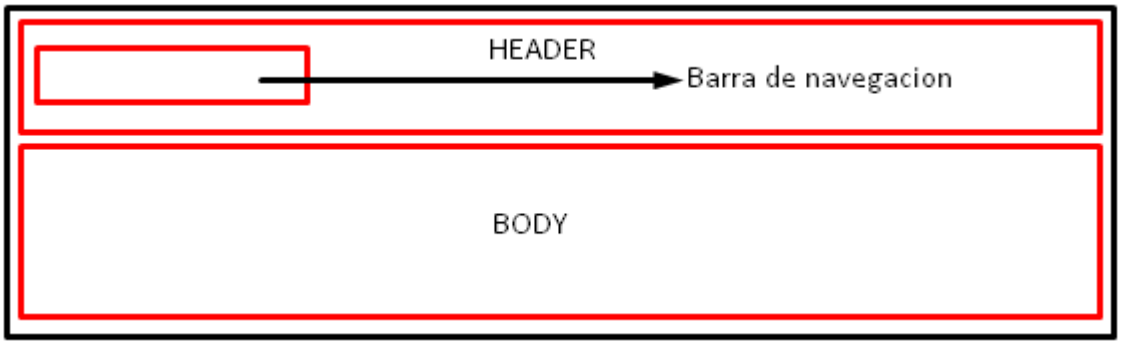


Ilustración 38. Estructura general de la interfaz WEB. (Fuente: Autor)

La estructura HTML de la aplicación tiene las siguientes partes: header, body como se muestra en la ilustración 39. El header es común para todas las rutas, mientras que el body cambia dependiendo del contenido de cada ruta. Utilizamos CSS para dar estilo a nuestro HTML; cargamos imágenes, cambiamos el tamaño del mapa y de los gráficos que muestran el flujo de civiles.

En el header tenemos una barra de navegación que consta de:

- Inicio: Nos lleva a la ruta http://localhost:4000/
- Acerca de Nosotros: Nos lleva a la ruta http://localhost:4000/info

Ilustración 39. Inicio de la aplicación con sus componentes HTML. (Fuente: Autor)

En el inicio de nuestra aplicación el body muestra la funcionalidad de la página, aquí tenemos el botón ¨Empecemos¨ que nos lleva a la ruta: http://localhost:4000/consultas.

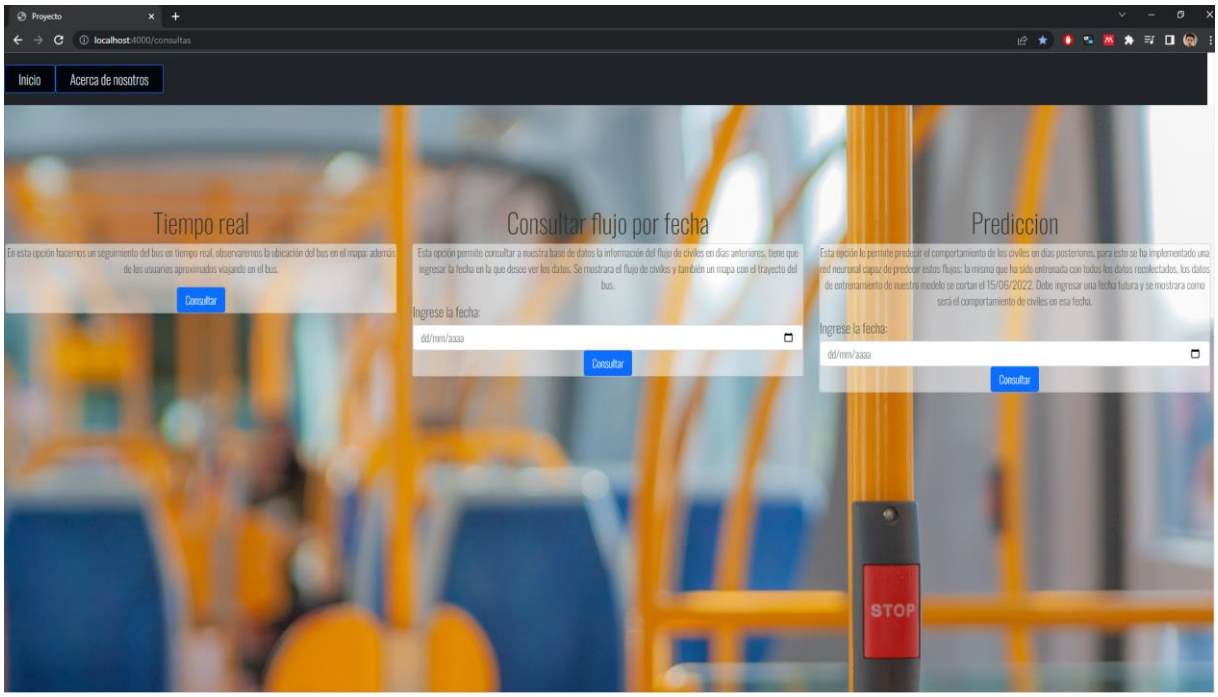


Ilustración 40. Ruta: ¨ http://localhost:4000/consultas¨ renderizada. (Fuente: Autor)

2.4.2. Visualización de Datos

Para la visualización de datos hemos realizado las tres opciones:

- Datos en tiempo real: Para visualizar la ubicación de la unidad de transporte y el número de civiles viajando el transporte público se observa en la ilustración 41.
- Datos históricos: Para consultar los datos almacenados en la fecha que indique el usuario, muestra la gráfica del flujo de civiles y el mapa con el trayecto de la unidad de transporte como se observa en la ilustración 42.
- Predicción de datos: Para visualizar la predicción realizada por el algoritmo implementado en la fecha que indique el usuario como se observa en la ilustración 43.

Para más detalles de la programación de las tres opciones revisar el apéndice C.

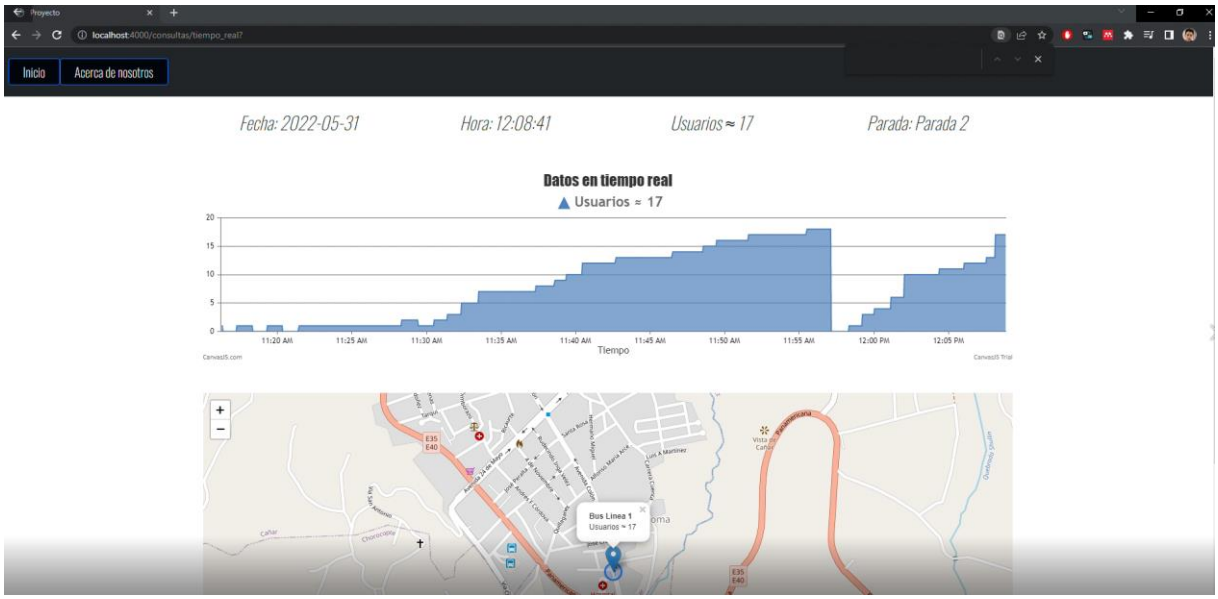


Ilustración 41. Datos en tiempo real, ruta: ¨ http://localhost:4000/consultas/tiempo_real ¨. (Fuente Autor)

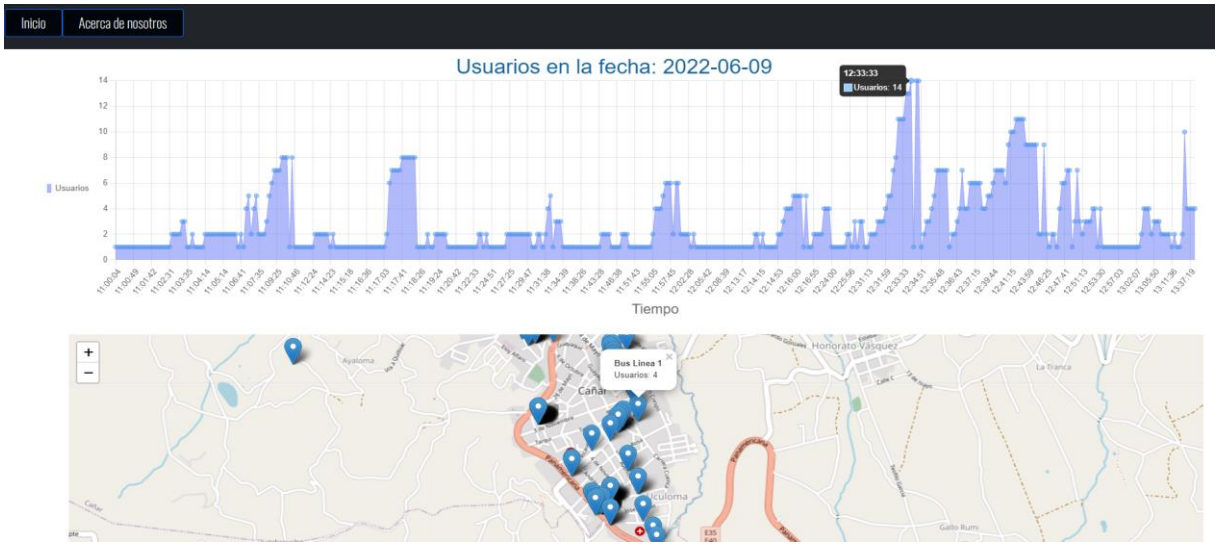


Ilustración 42. Datos históricos, ruta: ¨ http://localhost:4000/consultas¨. (Fuente: Autor)

Ilustración 43 Predicción de datos, ruta "http://localhost:4000/consultas/predicción" (Fuente: Autor)

2.5. Sistema de Predicción

2.5.1. Tratamiento de Datos

Este consiste en crear un dataset que contenga la información de modo que se pueda entrenar con la red LSTM, el proceso se detalla en la ilustración 45. Para más detalle en la programación revisar el apéndice D.

En la ilustración 55 del capítulo 3 en el cuadro rojo observamos un error la adquisición de datos, vemos que la gráfica está llegando aproximadamente hasta los 150 usuarios y en otros días llega a tener 0 usuarios, por lo que decidimos eliminar los datos desde el día 5 hasta el día 9 para el entrenamiento de la red LSTM.

En la ilustración 44 vemos el grafico del dataset con el que se va a entrenar la red, aquí no se observa irregularidades en los datos.

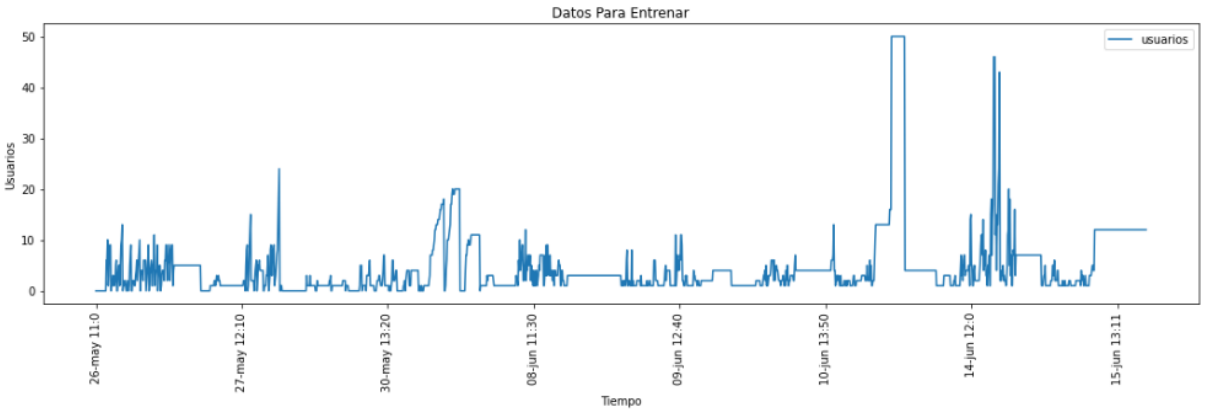


Ilustración 44. Gráfico del dataset final del flujo de usuarios (Fuente: Autor)

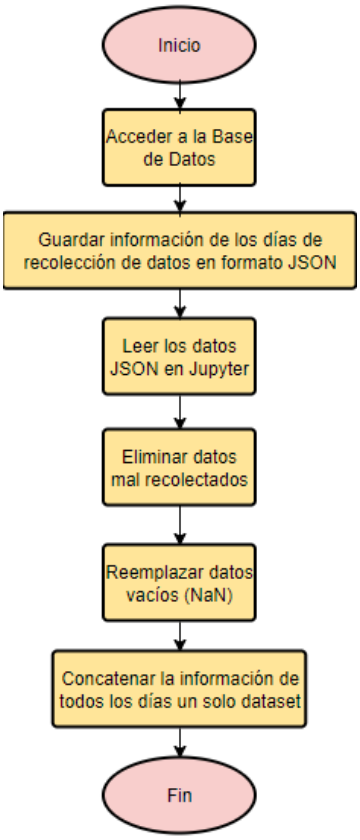


Ilustración 45 Diagrama de flujo del tratamiento de datos. (Fuente: Autor)

2.5.2. Entrenamiento y Predicción de Datos

Con el dataset preparado se ha divido en dos partes: entrenamiento y test. Se ha asignado un 70% del dataset para el entrenamiento y 30% para el test.

El entrenamiento lo vamos a realizar desde nuestra aplicación, para esto necesitamos de un módulo llamado “brain.js”.

- brain.js: es un módulo de redes neuronales acelerada por GPU escrita en JavaScript para Node.js, tiene muchas implementaciones de redes neuronales.

Ilustración 46 Brain.js. (Fuente: https://brain.js.org/#/)

El proceso del entrenamiento de la red LSTM se detalla a partir del diagrama de la ilustración 47, el entrenamiento requiere muchos recursos computacionales por lo que los resultados del entrenamiento han sido guardados, esto para que la aplicación no se detenga hasta que termine de entrenar la red. Los resultados de la red LSTM se detalla en el capítulo 3. Las redes neuronales requieren de grandes cantidades de datos para tener buenos resultados, nuestro proyecto recolecto datos de tres semanas lo que significa una cantidad pequeña de datos, por esta razón el proceso de entrenamiento se realizó para cada día de la semana, teniendo un total de cinco entrenamientos, uno para cada día.

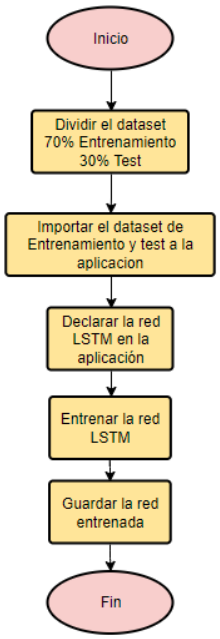


Ilustración 47 Diagrama de flujo del entrenamiento de la red LSTM. (Fuente: Autor)

Teniendo la red LSTM previamente entrenada, procedemos a realizar la predicción, el cliente desde la interfaz web escogerá la fecha en la que desea predecir el flujo de pasajeros, la aplicación identifica el día de semana que el cliente quiere predecir y muestra al cliente una gráfica con el flujo de pasajeros devuelta por la red LSTM como se muestra en la ilustración 43. En el anexo D se detalla la programación del entrenamiento con el módulo brain.js.

CAPÍTULO 3

3. Implementación y Análisis de Resultados

Aquí describimos la implementación en un transporte público de la ciudad de Cañar, también detallamos los costos de todos los componentes necesarios para el despliegue del proyecto. Analizaremos los resultados devueltos por la predicción, como también calculamos el error en la adquisición de datos.

3.1. Implementación

Nuestro dispositivo fue implementado en una unidad del sistema de transporte de la ciudad de Cañar, específicamente en la línea 1. Los días de recolección de datos fueron 15, se instaló el dispositivo de 11:00 a 13:00 de acuerdo con la estimación de la duración de la batería.

El dispositivo fue incorporado en la cabina de pasajeros, sujetado a un asiento como se observa en las ilustraciones 48 y 49.

Ilustración 48. Incorporación del dispositivo en el transporte publico. (Fuente: Autor)

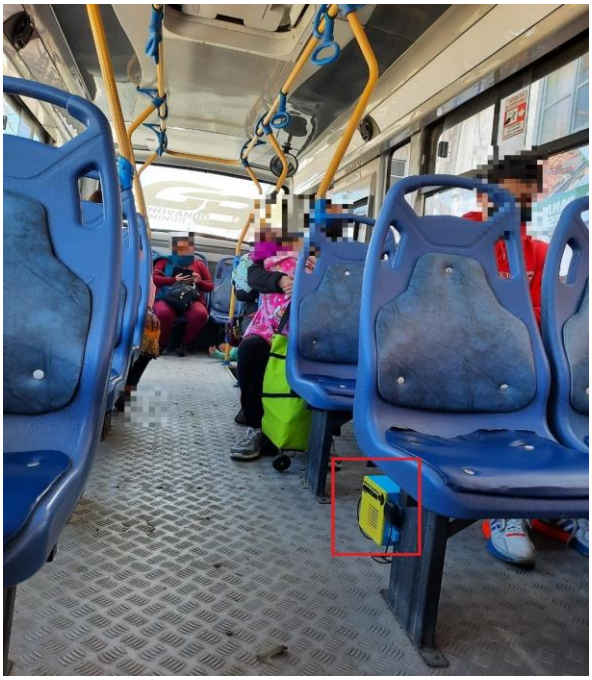


Ilustración 49. Incorporación del sistema en el transporte público. (Fuente: Autor)

Nuestra página WEB resulta ser intuitiva para el usuario, cumple con los requisitos planteados al iniciar proyecto.

En la ilustración 50, se muestra cómo funciona la página WEB, en la opción de tiempo real observamos en la parte de información los datos de recolección: Fecha de recolección de datos, Hora en tiempo real de la recolección de datos, Usuarios aproximados en el transporte público, Parada en la que se encuentra la unidad de transporte. Con respecto a la ilustración 50, en la gráfica Dinámica se muestra el flujo de usuarios que varían en tiempo real. En Ubicación se muestra la posición de la unidad de transporte en tiempo real. (Para observar el funcionamiento del servidor WEB en tiempo real dejamos como evidencia el siguiente video: https://youtu.be/1O33Zh1H5zI)

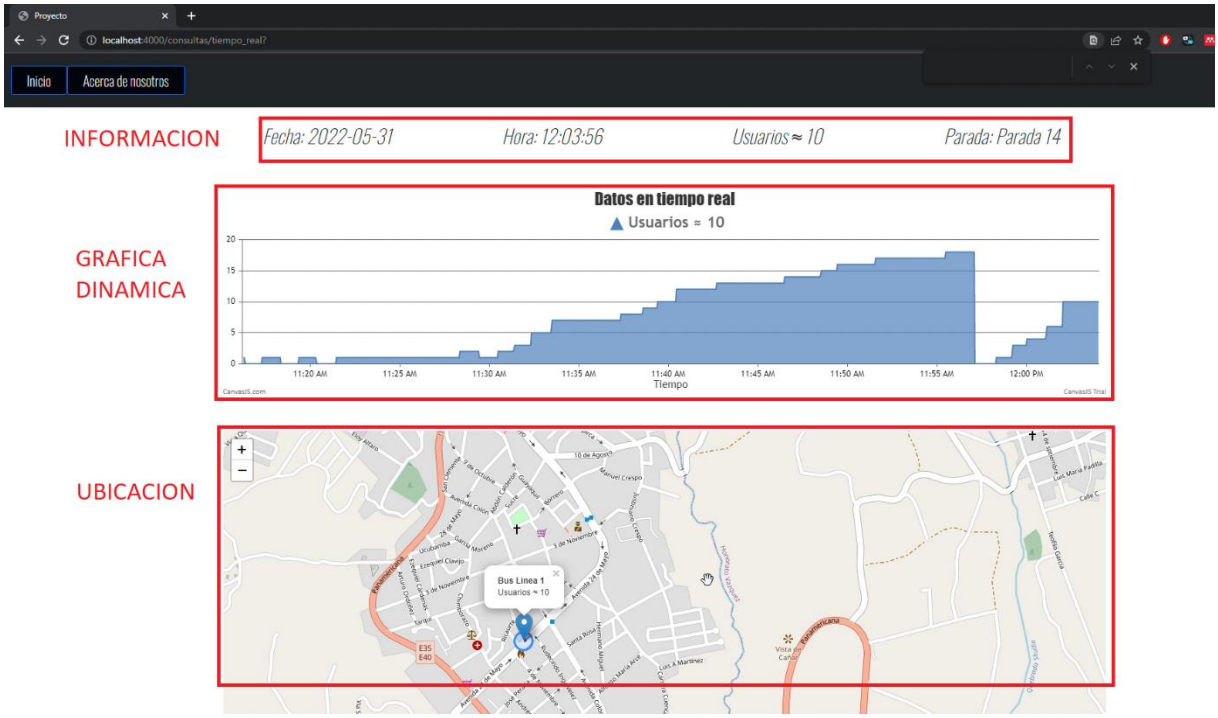


Ilustración 50. Funcionamiento página WEB opción: tiempo real. (Fuente: Autor)

En la ilustración 51 se muestra cómo funciona la página WEB en el apartado de consultar flujo por fecha, el usuario puede ver datos según la fecha ingresada, en la parte de grafica muestra el flujo de usuarios vs el tiempo, en la parte de ubicación se muestra la ruta de la unidad de transporte.

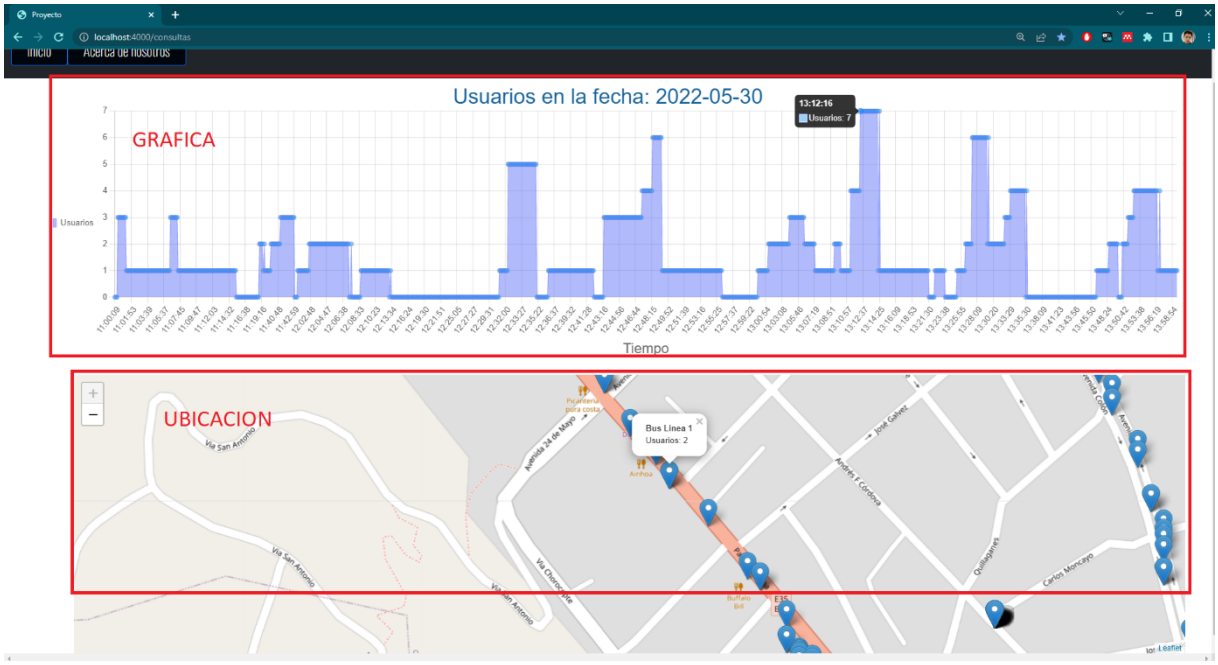


Ilustración 51. Funcionamiento página WEB opción: Consultar flujo por fecha. (Fuente: Autor)

En la ilustración 52 se puede observar el funcionamiento de la página WEB en la opción de predicción, el usuario puede ver la gráfica de predicción obtenida por la Red LSTM según la fecha ingresada para predecir.

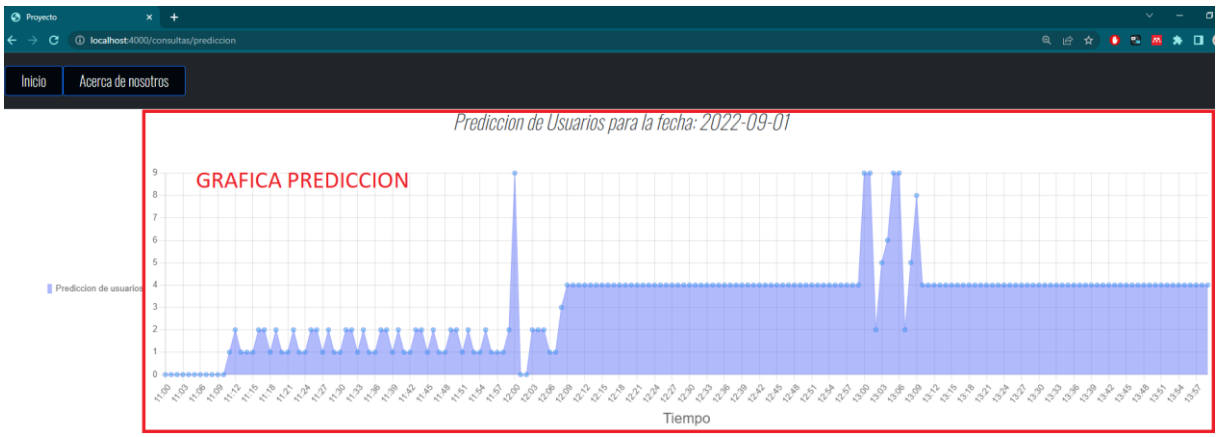


Ilustración 52. Funcionamiento página WEB opción: Predicción. (Fuente: Autor)

3.2. Costo de Implementación

La siguiente tabla detalla el precio de implementación para una unidad de trasporte público:

Tabla 7 Costos de implementación.

Cantidad	Descripción	Costo
1	Raspberry Pi	$140
1	Modem 4G	$60
1	Plan de datos Movistar	$28
1	Modulo GPS	$60
1	Antena GPS	$10
1	Impresión 3D de la caja	$60
1	Módulo de baterías Raspberry Pi	$20
1	Par de baterías de litio	$30
	Mano de Obra	$700
TOTAL		$1108

Si el proyecto se desea implementar en todas las unidades de transporte público con el fin de analizar, gestionar y proponer soluciones para ofrecer una mejor viabilidad en la parte urbana de la ciudad de Cañar el costo será:

Tabla 8 Costo de implementación Total.

Número de Unidades	Costo por Unidad	Total
10	$1108	$11080

3.3. Error en la Adquisición de Datos

Para el cálculo del error se realizó un viaje de 3 horas el día 3, tomando datos reales de pasajeros en el transporte público y fueron comparados con los adquiridos por el módulo detector de civiles, en la ilustración 53 se observa los datos reales y los datos escaneados, a partir de estos datos calculamos: error absoluto y el error relativo. El error relativo del módulo detector de civiles es de 29.55%.

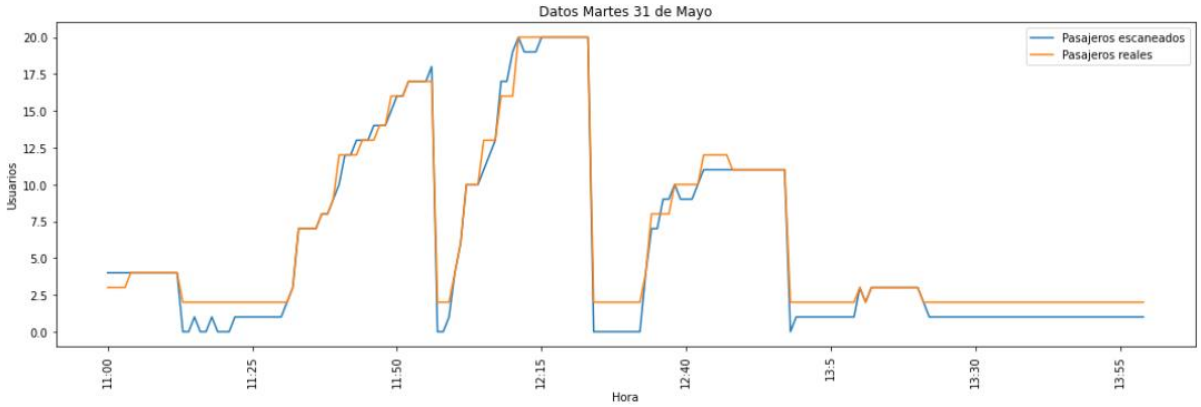


Ilustración 53 Datos reales vs datos escaneados. (Fuente: Autor)

3.4. Análisis de Resultados

Los datos obtenidos por el módulo detector de civiles en los días de recolección de datos se muestran en la ilustración 54, observamos el dataset con las fechas correspondientes a los 15 días de recolección de datos, las columnas contienen el número de pasajeros de acuerdo con la hora escaneada. Los datos vacíos (NaN) son producidos por un error de comunicación, la cobertura de la red 4G en algunos sectores de la ciudad era limitada por lo que el detector de civiles dejaba de transmitir en cortos periodos de tiempo.

	tiempo	jue-26-may	vie-27-may	lun-30-may	mar-31-may	mie-01-jun	jue-02-jun	vie-03-jun	lun-06-jun	mar-07-jun	mie-08-jun	jue-09-jun	vie-10-jun	lun-13-jun	mar-14-jun	mie-15-jun
40	11:40	5.0	NaN	2.0	10.0	NaN	NaN	NaN	NaN	7.0	4.0	1.0	NaN	1.0	1.0	2.0
41	11:41	1.0	NaN	2.0	12.0	1.0	NaN	NaN	NaN	4.0	7.0	NaN	NaN	1.0	4.0	1.0
42	11:42	5.0	NaN	3.0	12.0	0.0	NaN	NaN	NaN	6.0	7.0	1.0	1.0	1.0	7.0	1.0
43	11:43	9.0	NaN	0.0	13.0	0.0	NaN	NaN	NaN	9.0	7.0	1.0	1.0	2.0	3.0	1.0
44	11:44	10.0	NaN	0.0	13.0	1.0	NaN	NaN	NaN	11.0	7.0	2.0	NaN	2.0	5.0	1.0
45	11:45	13.0	NaN	1.0	13.0	NaN	NaN	NaN	NaN	11.0	7.0	1.0	NaN	2.0	2.0	1.0
46	11:46	0.0	NaN	NaN	14.0	0.0	3.0	NaN	NaN	11.0	7.0	1.0	NaN	2.0	6.0	1.0
47	11:47	1.0	NaN	NaN	14.0	0.0	NaN	NaN	NaN	11.0	1.0	1.0	NaN	2.0	7.0	2.0
48	11:48	1.0	NaN	NaN	14.0	NaN	NaN	NaN	NaN	0.0	1.0	2.0	NaN	2.0	7.0	2.0
49	11:49	2.0	NaN	NaN	15.0	NaN	NaN	NaN	NaN	0.0	3.0	NaN	NaN	2.0	3.0	2.0

Ilustración 54 Dataset con el número de usuarios recolectados. (Fuente: Autor)

Los datos del día 5 hasta el día 10, presentan irregularidades en la recolección de datos como se puede apreciar en la ilustración 55. Este problema fue causado por el calor acumulado dentro de la caja que transportaba los dispositivos, esto causaba un error en el escaneo acumulando demasiados usuarios o dejando de escanear, para solucionar este error se incorporó un sistema de enfriamiento. Los días mencionados fueron excluidos de la etapa de entrenamiento y predicción de datos.

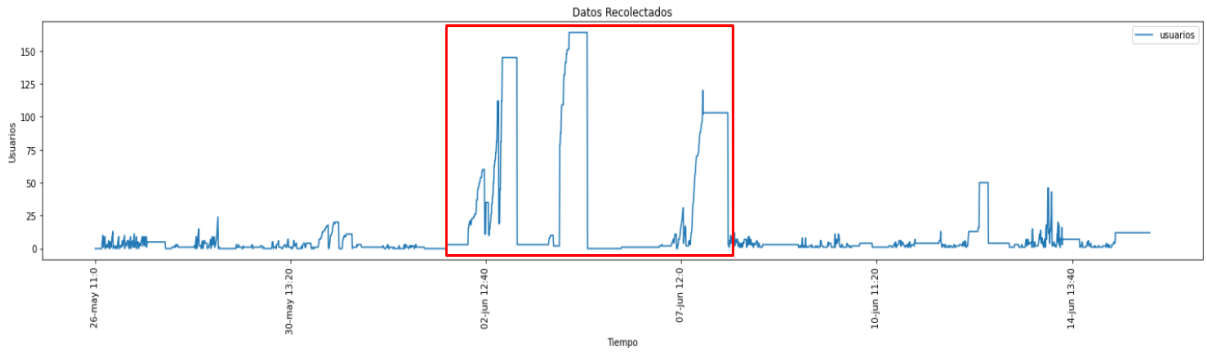


Ilustración 55 Grafico del flujo de usuarios. (Fuente: Autor)

Para realizar un análisis se eliminaron los datos mal recolectados teniendo un dataset final como se observa en la siguiente ilustración 56.

	tiempo	jue-26-may	vie-27-may	lun-30-may	mar-31-may	mie-08-jun	jue-09-jun	vie-10-jun	lun-13-jun	mar-14-jun	mie-15-jun
0	11:0	0	0	0	4	3	1	4	6	1	1
1	11:1	0	0	3	4	2	1	4	6	1	1
2	11:2	0	0	1	4	2	1	4	6	1	1
3	11:3	0	0	1	4	1	2	4	7	1	2
4	11:4	0	0	1	4	6	1	4	13	1	3
...	...	...	...	...	...	...	...	...	...	...	...
175	13:55	5	0	4	1	3	4	4	4	7	12
176	13:56	5	0	4	1	3	4	4	4	7	12
177	13:57	5	0	1	1	3	4	4	4	7	12
178	13:58	5	0	1	1	3	4	4	4	7	12
179	13:59	5	0	1	1	3	4	4	4	7	12

180 rows × 11 columns

Ilustración 56 Dataset final (Fuente: Autor)

El resultado en la recolección de datos fue exitoso en un 70%, en las ilustraciones del apéndice E observamos que el aforo en la unidad de transporte comienza a aumentar alrededor de las 11:30 am en adelante, también podemos observar los errores de recolección del día 5 hasta el día 9.

La ilustración 57 que corresponde al día 1 obtenemos un pico de 13 usuarios que utilizaron la unidad de transporte urbano, como se muestra en la imagen tenemos una media de 6 a 10 usuarios que se transportaron aquel día.

Ilustración 57. Datos tomados del día 1. (Fuente: Autor)

3.5. Predicciones

Los resultados de las predicciones tienen un error de entrenamiento del 29.78% como se observa en la ilustración 58, el error es calculado por el módulo “brain.js”.

```
iterations: 6140, training error: 29.782404299577077
iterations: 6150, training error: 29.779534959130817
iterations: 6160, training error: 29.78340469267633
```

Ilustración 58 Error de entrenamiento. (Fuente: Autor)

El error se debe a la poca cantidad de datos recolectados, aun así, la red LSTM puede ser implementada pues con la poca cantidad de datos llega a tener resultados aproximados a los datos reales, esto se puede observar en las ilustraciones del apéndice E, donde se compara la predicción y los datos reales, en algunos minutos la predicción consigue tener bastante similitud con los datos reales. En la ilustración 59 se observa la similitud entre la predicción y los datos reales.

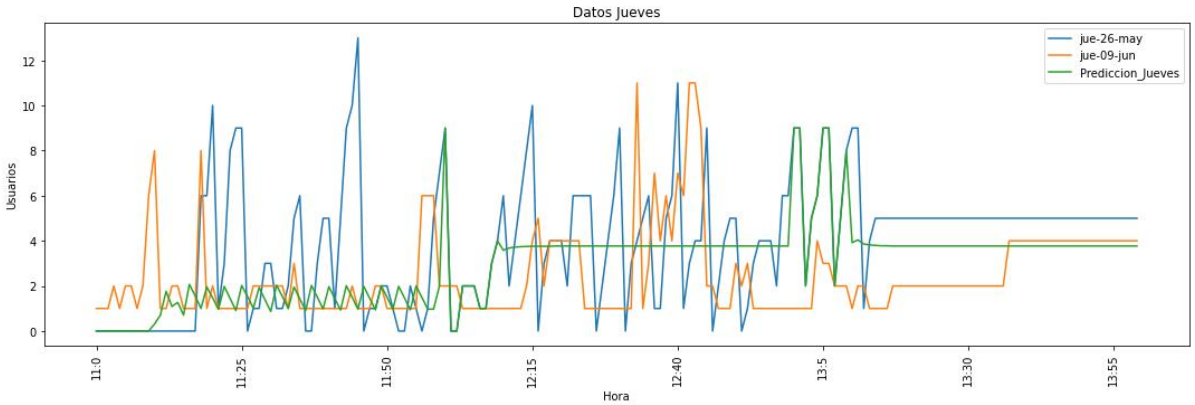


Ilustración 59. Graficas de predicción y datos reales. (Fuente: Autor)

CAPÍTULO 4

4. Conclusiones y Recomendaciones

4.1. Conclusiones

Al implementar el nodo de detección de civiles, se comprueba que los métodos no visibles pueden llegar a ser prácticos y eficientes, gracias a sus bajos costos de implementación y recursos computacionales para su funcionamiento, además es un método que proteger la privacidad del usuario. En este proyecto obtuvimos una precisión del 70% debido a que en la ciudad de Cañar la red 4G tiene una cobertura limitada generando interrupciones en él envió de información. Se comprobó que el protocolo MQTT es eficiente en la trasmisión de datos, llegando a tener un retardo de la información en algunos casos de menos de un segundo.

El servidor WEB muestra graficas intuitivas y con la precisión de la estimación de civiles antes mencionada, constatamos que los administradores de transporte publico pueden hacer uso de la interfaz WEB para tomar decisiones que conduzcan a la mejora el sistema de transporte, por ejemplo, al realizar el análisis de datos recolectados, se pudo verificar que en las horas pico (11:45am hasta 12:30pm) aumenta el número de civiles dentro de la unidad de transporte, los administradores del transporte público en base a las gráficas y a la ubicación de las unidades de transporte que muestra la interfaz WEB pueden tomar acciones para mejor el sistema de transporte en ese horario.

La red LSTM tienen un error de entrenamiento de 29.79%, un error bajo considerando que los datos de entrenamiento fueron limitados, esto demuestra que la red LSTM es aplicable para datos de series temporales, como lo es este proyecto, pues las predicciones realizadas muestran unos resultados similares a los reales. Las predicciones podrían generar soluciones a problemas de movilidad, por ejemplo, según la frecuencia de usuarios de acuerdo con los resultados obtenidos por la predicción se podría disminuir o aumentar la frecuencia de unidades de transporte, estas medidas ayudarían a solucionar problemas como: aglomeraciones en las unidades de transporte, usuarios sin poder hacer uso del transporte público debido a su aforo máximo, unidades que circulan sin pasajeros.

4.2. Recomendaciones y Trabajos Futuros

Las entidades encargadas del Sistema de Transporte deberían considerar la implementación del sistema, puesto que carece de un monitoreo en las unidades de transporte, tener la visualización de datos por parte de los administradores conlleva a la toma de mejores resoluciones para mejorar el sector urbano. Este proyecto comprueba que el monitoreo es beneficioso para los administradores y para los usuarios.

En la ciudad de Cañar el sistema de transporte publico cuenta con 10 unidades, si se implementa el detector de civiles en cada unidad de transporte, un posible trabajo a futuro seria crear en el servidor WEB un sistema de administración, pudiendo el cliente registrarse e iniciar sesión, dándole privilegios para agregar o quitar unidades que cuenten con el detector de civiles, así también los usuarios podrían hacer seguimiento de todos las unidades o elegir solo una, ayudando a planificar mejor sus rutas para llegar al destino deseado.

Al momento de colocar el módulo detector de civiles dentro de la unidad, este tiene que estar en el centro de la cabina de los pasajeros, disminuyendo el error en el escaneo evitando escanear usuarios que se encuentren fuera del transporte público.

En el caso del armado del módulo detector de civiles, se debe tomar en cuenta la ventilación, pues si la raspberry se sobrecalienta tiende a fallar, se recomienda adaptar un ventilador adicional al que ya tiene el módulo. Además, en el diseño de la caja si se aumenta las salidas de aire, podría ayudar a disipar el calor de manera más eficiente.

Debido a que las raspberry pi 4 es costosa un posible trabajo a futuro seria implementar el módulo detector de civiles con una raspberry pi zero, se tendría que tomar en cuenta si soporta los requisitos necesarios para el escaneo de las PR.

Si el proyecto se desea implementar de manera permanente en el sistema de trasporte, se recomienda adaptar el módulo detector de civiles a las baterías de las unidades de transporte para aumentar el tiempo de recolección de datos, esto como una recomendación para futuros investigadores interesados en este proyecto.

Para mejorar los resultados de la predicción es necesario extender los días de recolección de datos, esto ayudara a la red LSTM a disminuir el error de entrenamiento.

Para disminuir el error en los datos recolectados, se sugiere que el módulo detector de civiles guarde los datos en la raspberry, teniendo un respaldo para poder recuperar los datos perdidos en zonas donde la red 4G no tenga cobertura.

Podemos acceder a datos del clima que existen en la nube, si vinculamos estos datos y los relacionamos con el flujo de civiles, se puede crear resultados de predicciones más precisas, pues el clima influye en el uso del transporte público, por lo tanto, un posible trabajo a futuro seria vincular estos datos y relacionarlos con el flujo de civiles en las unidades.

REFERENCIAS BIBLIOGRÁFICAS

[1] M. Alcaraz, "Internet de las Cosas." Accessed: May 26, 2021. [Online]. Available: http://www.uca.edu.py.

[2] J. Ochoa-Zambrano and J. Garbajosa, "Social Internet of Things: Architectural Approaches and Challenges 1."

[3] D. Schuurman, B. Baccarne, L. De Marez, and P. Mechant, "Smart ideas for smart cities: Investigating crowdsourcing for generating and selecting ideas for ICT innovation in a city context," *J. Theor. Appl. Electron. Commer. Res.*, vol. 7, no. 3, pp. 49–62, Dec. 2012, doi: 10.4067/S0718-18762012000300006.

[4] L. Chuqui, "Desarrollo de una plataforma abierta para el almacenamiento y gestión de información de nodos móviles para internet de las cosas de la célula inteligente de la Universidad Politécnica Salesiana," 2016.

[5] "Climate CoLab | MIT Center for Collective Intelligence." .

[6] Center for Collective Intelligence., "Center for Collective Intelligence.," *Handbook of Collective Intelligence.*, 2012. .

[7] A. S. López Borja, "Análisis del comportamiento del transporte público a nivel mundial," *Univ. Part. Int. SEK*, vol. 39, no. April, pp. 1–98, 2018.

[8] V. D. Suárez, "Informe Calidad Del Aire," *Secr. Ambient.*, vol. 7, no. July, pp. 1–25, 2018.

[9] L. Garcia, J. M. Jiménez, M. Taha, and J. Lloret, "Wireless Technologies for IoT in Smart Cities," *Netw. Protoc. Algorithms*, vol. 10, no. 1, p. 23, 2018, doi: 10.5296/npa.v10i1.12798.

[10] A. Cama, E. De la Hoz, and D. Cama, "Las redes de sensores inalámbricos y el Internet de las cosas Palabras clave."

[11] X. Kong, X. Liu, B. Jedari, M. Li, L. Wan, and F. Xia, "Mobile Crowdsourcing in Smart Cities: Technologies, Applications, and Future Challenges," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8095–8113, Oct. 2019, doi: 10.1109/JIOT.2019.2921879.

[12] F. M. Biot, "IMÁGENES MEDIANTE EL PROTOCOLO MQTT," 2020.

[13] S. Quincozes, T. Emilio, and J. Kazienko, "MQTT protocol: Fundamentals, tools and future directions," *IEEE Lat. Am. Trans.*, vol. 17, no. 9, pp. 1439–1448, 2019, doi: 10.1109/TLA.2019.8931137.

[14] Á. Herranz, "Desarrollo de aplicaciones para IoT con el módulo ESP32," *Univ. Alcalá*, p. 37,38, 2019.

[15] "Vista de Clasificación de iniciativas de crowdsourcing basada en tareas." https://revista.profesionaldelainformacion.com/index.php/EPI/article/view/epi.2012.may.09/17923 (accessed May 12, 2021).

[16] "Mobile Crowdsourcing in Smart Cities: Technologies, Applications, and Future Challenges |

IEEE Journals & Magazine | IEEE Xplore." https://ieeexplore.ieee.org/abstract/document/8733838 (accessed May 26, 2021).

[17] M. Isabel alonso de Magdaleno and J. garcía garcía, "Crowdsourcing: la descentralización del conocimiento y su impacto en los modelos productivos y de negocio Crowdsourcing: knowledge decentralization and its impact on production and business models," vol. 14, no. 2, pp. 33–50, 2014, doi: 10.5295/cdg.120351ma.

[18] C. Tanas and J. Herrera-Joancomartó, "Crowdsensing simulation using ns-3," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8313, pp. 47–58, doi: 10.1007/978-3-319-04178-0_5.

[19] I. Contreras and G. Hernández, "Sistema de localización en tiempo real mediante un servidor web y aplicaciones móviles," *Pist. Educ.*, vol. 39, no. 127, pp. 171–186, 2017, [Online]. Available: http://itcelaya.edu.mx/ojs/index.php/pistas%0Ahttp://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/1073.

[20] W. van der Aalst, "Data Science in Action," *Process Min.*, pp. 3–23, 2016, doi: 10.1007/978-3-662-49851-4_1.

[21] V. Dhar, "Data science and prediction," *Commun. ACM*, vol. 56, no. 12, pp. 64–73, Dec. 2013, doi: 10.1145/2500499.

[22] M. L. Forcada and S. Inform, "Tesis doctoral MODELOS PREDICTIVOS BASADOS DE TIEMPO DISCRETO Juan Antonio P ´ Julio de 2002," 2002.

[23] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 1643–1647, 2017, doi: 10.1109/ICACCI.2017.8126078.

[24] J. Escalante Calcina, "TWO - PASS END-TO-END: RNN-T-LAS VS. LSTM-LAS," vol. 1, p. 55, 2021, [Online]. Available: http://hdl.handle.net/20.500.12773/14085.

[25] D. Fabian and H. Cofre, "Universidad Jorge Tadeo Lozano ´ n para el mercado de Predicci o acciones con Redes Neuronales LSTM Universidad Jorge Tadeo Lozano ´ n para el mercado de Predicci o acciones con Redes Neuronales," p. 34, 2020, [Online]. Available: https://expeditiorepositorio.utadeo.edu.co/handle/20.500.12010/13673.

[26] J. J. De Lucio Fernández, "Estimación adelantada del crecimiento regional mediante redes neuronales LSTM," *Investig. Reg. = J. Reg. Res. ISSN 1695-7253, ISSN-e 2340-2717, Nº. 49, 2021, págs. 45-64*, vol. 49, no. 49, pp. 45–64, 2021, [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=7867482&info=resumen&idioma=ENG%0Ahttps://dialnet.unirioja.es/servlet/articulo?codigo=7867482&info=resumen&idioma=SPA%0Ahttps://dialnet.unirioja.es/servlet/articulo?codigo=7867482.

[27] R. C. Braggaar, "Wi-Fi Network-Based Indoor Localisation," 2018.

[28] U. Singh, J. F. Determe, F. Horlin, and P. De Doncker, "Crowd Forecasting Based on WiFi Sensors and LSTM Neural Networks," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 9, pp. 6121–6131, 2020, doi: 10.1109/TIM.2020.2969588.

[29] M. Irfan, L. Marcenaro, and L. Tokarchuk, “Crowd analysis using visual and non-visual sensors, a survey,” in *2016 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2016 - Proceedings*, Apr. 2017, pp. 1249–1254, doi: 10.1109/GlobalSIP.2016.7906041.

[30] S. Depatla, A. Muralidharan, and Y. Mostofi, “Occupancy Estimation Using Only WiFi Power Measurements,” *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1381–1393, 2015, doi: 10.1109/JSAC.2015.2430272.

[31] J. Weppner and P. Lukowicz, “Bluetooth based collaborative crowd density estimation with mobile phones,” *2013 IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2013*, no. March, pp. 193–200, 2013, doi: 10.1109/PerCom.2013.6526732.

[32] O. Merchán, *Diseño de base de datos*. 2018.

[33] “Curso de Introducción a la Administración de Bases de Datos - Miguel Ángel Benítez, Ángel Arias - Google Libros.” .

[34] C. Wai Zhao, J. Jegatheesan, and S. Chee Loon, “Exploring IOT Application Using Raspberry Pi,” *Int. J. Comput. Networks Appl.*, vol. 2, no. 1, pp. 27–34, 2015, [Online]. Available: http://www.digi.com.

[35] Araujo Evelyn, “Implementación de un sistema de video vigilancia para los exteriores de la UPS, mediante mini computadores y cámaras Raspberry PI.,” *Univ. Politec. Sales. Ecuador*, p. 108, 2015, [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/10379%0Ahttps://dspace.ups.edu.ec/bitstream/123456789/10379/1/UPS-GT001404.pdf.

[36] A. D. E. Medios *et al.*, “UNA NUEVA METODOLOGIA PARA LA GESTION DEL CONOCIMIENTO BASADO EN GEOPOSICIONAMIENTO,” 2015.

[37] Engel, “Sistema de Seguridad para el Encendido, Apagado y Bloqueo de motor, utilizando Biometria y Geoposicionamiento,” *Pap. Knowl. . Towar. a Media Hist. Doc.*, 2014.

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

APÉNDICES

APÉNDICE A: SISTEMA DE ADQUISICIÓN DE DATOS

A.1: Instalación kali linux

Desde la página oficial de Kali Linux se puede descargar la última versión del sistema operativo. Seleccionamos de acuerdo a las especificaciones de la raspberry.

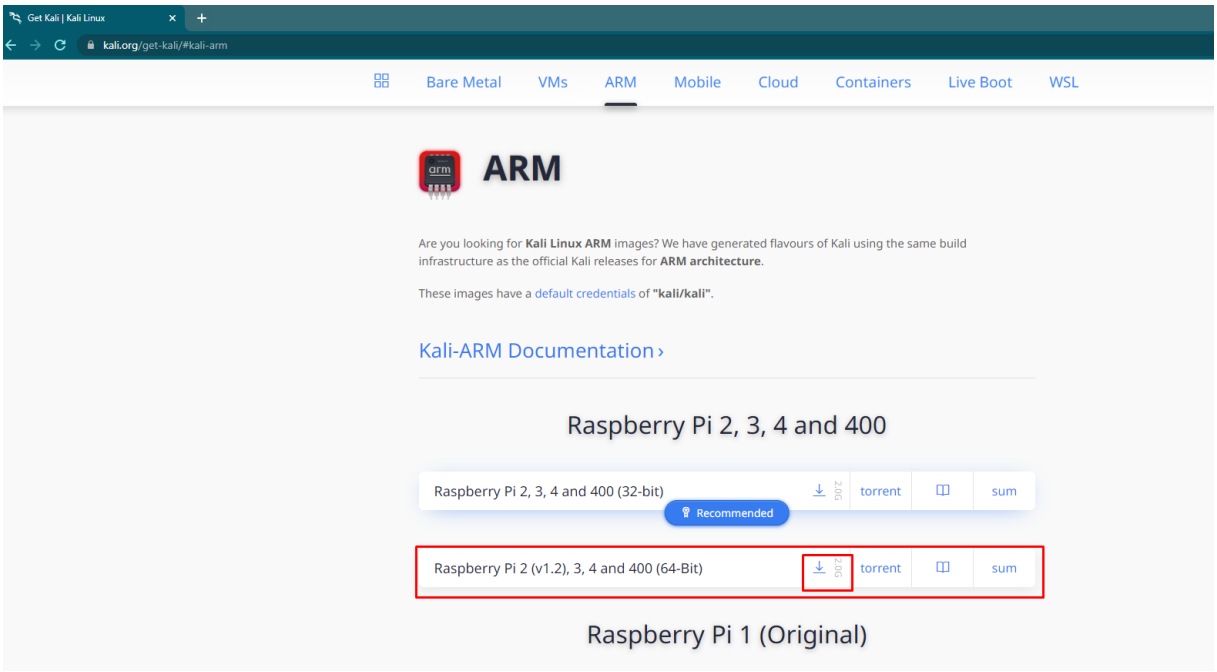


Ilustración A.1 1 Pagina WEB oficial de Kali Linux

Descargamos balenaEtcher es un software que permite grabar imágenes de sistemas operativos en tarjetas SD, en este caso vamos a grabar el sistema operativo anteriormente descargado.

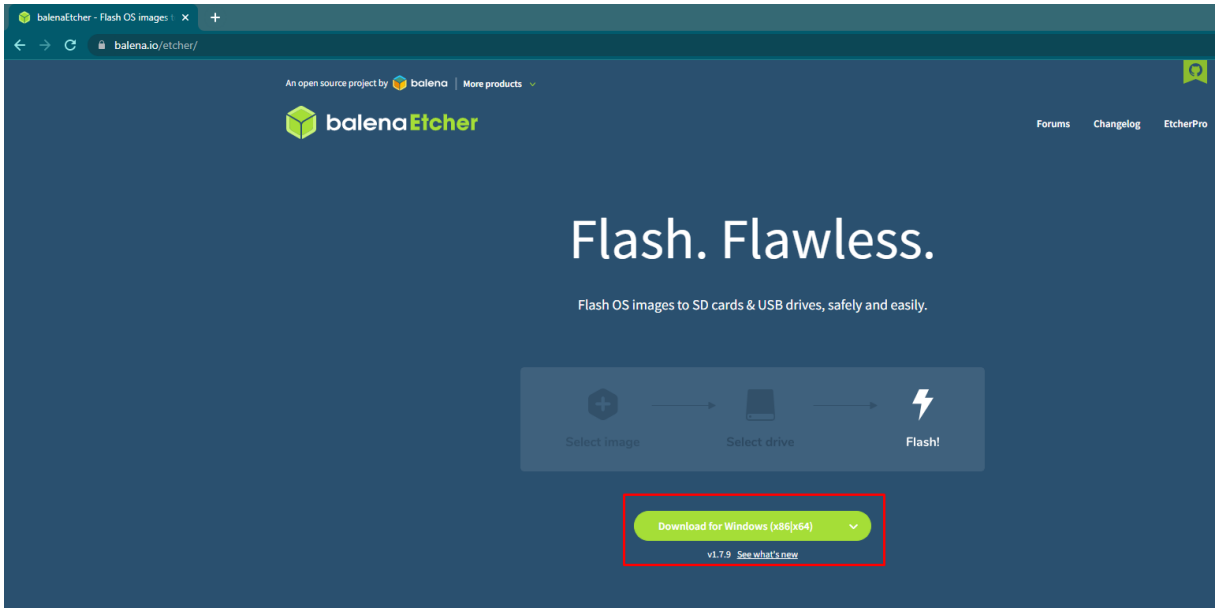


Ilustración A.1 2 Pagina WEB balenaEtcher

Después de descargar e instalar balenaEtcher, abrimos el software seleccionamos el sistema operativo así también como la tarjeta SD en la que se va a grabar la imagen del sistema operativo, se recomienda una tarjeta de 64GB, para este proyecto se usó una tarjeta SD de 128GB. Damos clic en Flash y se graba el sistema operativo.

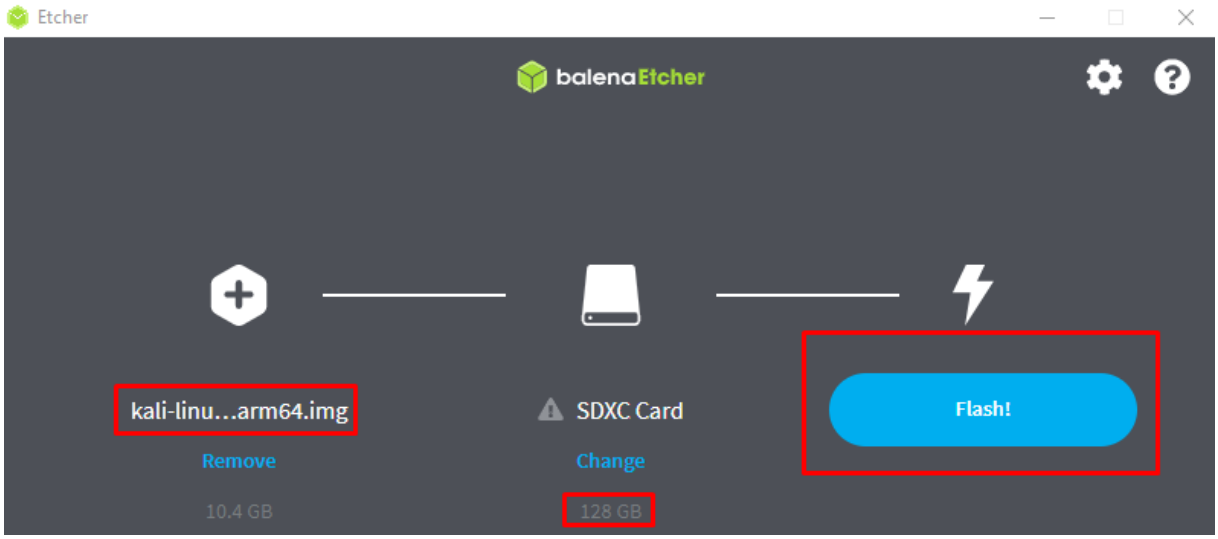


Ilustración A.1 3 Ventana de balenaEtcher

Colocamos la tarjeta SD en la raspberry y encendemos la misma.

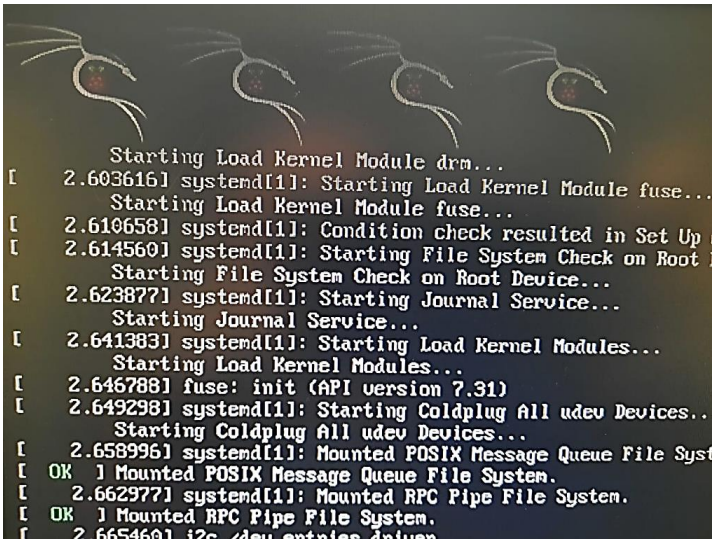


Ilustración A.1 4 Kali Linux iniciando en la raspberry

A.2 Colocar el adaptador USB WIFI en modo monitor

Con la ayuda de Putty nos conectamos con nuestra raspberry, primero ingresamos el comando "airmon-ng".

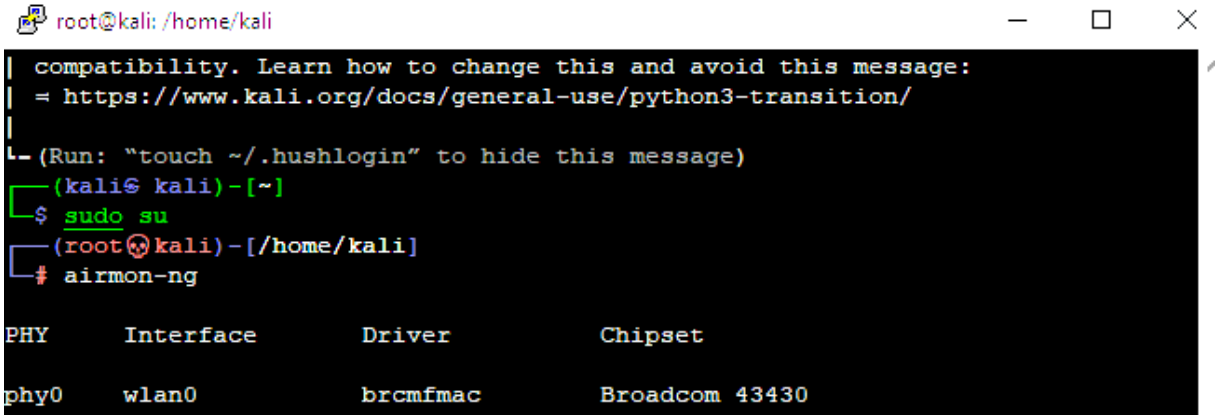


Ilustración A.2 1 Comando "airmon-ng"

Segundo ejecutamos el comando "airmon-ng check kill" para matar todos los procesos que puedan interrumpir en el escaneo.

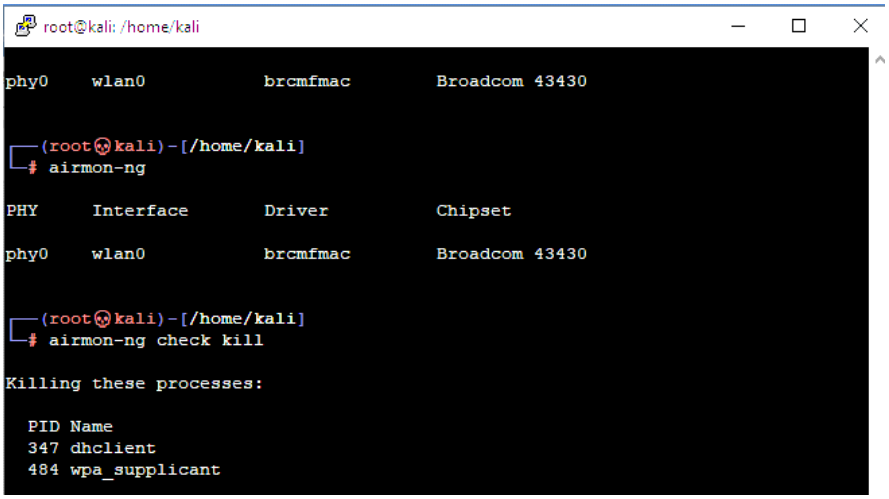


Ilustración A.2 2 Comando "airmon-ng check kill"

Tercero establecemos el adaptador USB-WIFI en modo monitor con el comando "airmon-ng start wlan0".

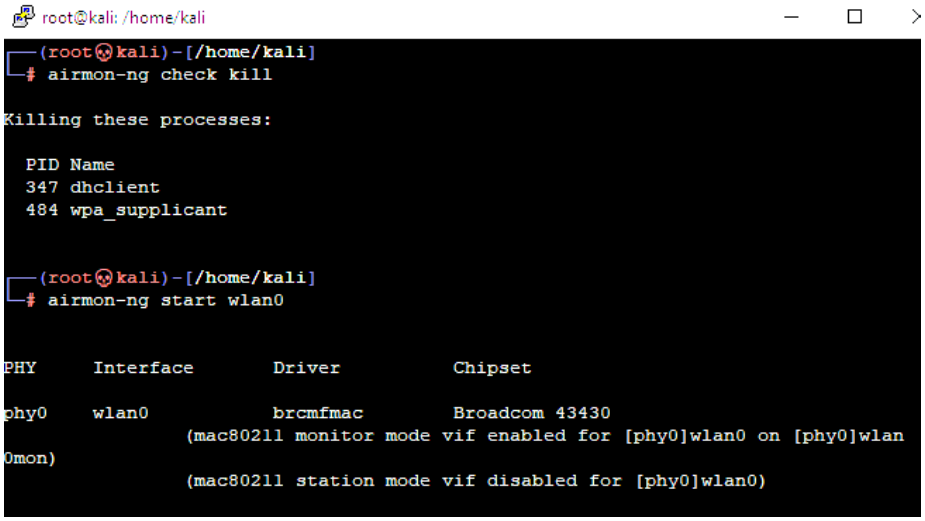


Ilustración A.2 3 comando "airmon-ng start wlan0"

Ahora el adaptador USB-WIFI se encuentra en modo monitor y podemos realizar el escaneo de las PR con el siguiente comando "airodump-ng wlan0mon"

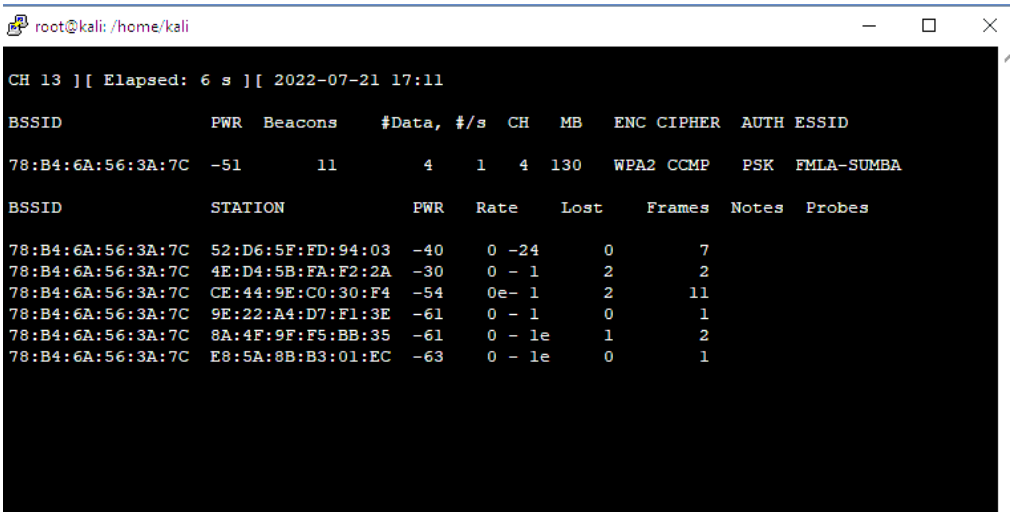


Ilustración A.2 4 Comando "airodump-ng wlan0mon"

Para guardar los datos del escaneo se utiliza el comando "sudo airodump-ng --channel 1-14, --write dump --output-format csv wlan0mon"

A.3 Recuperación y filtrado de usuarios

La recuperación y el filtrado de usuarios se realiza con Python, las siguientes líneas de código realizan el proceso de recuperación de los usuarios con WIFI activado y el filtrado de usuarios.

```
import pandas as pd
import os
import sys
import subprocess
import time
from datetime import date
from datetime import datetime
while True:
    time.sleep(1)
    try:
        data=pd.read_csv("/home/kali/Desktop/dump-01.csv") #leer datos del escaneo
        columnas = ["Station_MAC", "First time seen", "Last time seen", "Power","paquetes","BSSID","Probed ESSIDs", "Key", " ESSID", " Authentication", " Power", " # beacons", " # IV", " LAN IP", " ID-length"]
        data1 = pd.read_csv("/home/kali/Desktop/dump-01.csv", header = None , names = columnas )
```

```
        data2=data1.drop(["Key", " ESSID", " Authentication", " Power", " LAN IP", " ID-length"], axis = 1)
        indice=data.index[data['BSSID'] == 'Station MAC']
        indice_macs=(indice+2).tolist()
        data3 = data2[indice_macs[0]:]
        subset = data3[["Station_MAC", " # beacons", " # IV"]]
        subset1= data3["First time seen"].str.split(expand=True)
        subset2= data3["Last time seen"].str.split(expand=True)
        subset1.columns = ["Date_Ft", "Time_Ft"]
        subset2.columns = ["Date_Lt", "Time_Lt"]
        potencia=data3["Power"].astype(float)
        df_temp=data3["paquetes"].astype(float)
        filter_users = pd.concat([subset, potencia], axis = 1)
        filtrados = filter_users[(filter_users["Power"]>(-50)) & (filter_users["Power"]!=(-1))]
        a=filtrados.shape
        num_macs=a[0]
        dic={'usuarios':[ num_macs]}
        dic_users=pd.DataFrame(dic) #diccionario con el numero de usuarios


        now = datetime.now()
        today = date.today()
        today_number= date.today().weekday() #Devuelve el día de la semana como un número entero, donde el lunes es 0 y el domingo es 6.
        hora = now.strftime('%H:%M:%S')
        dic1={'hora':[hora], 'fecha':[today], 'weekday': [today_number]}
        dic_hora=pd.DataFrame(dic1) # diccionario con la hora
        dic_final= pd.concat([dic_users, dic_hora], axis = 1)
        dic_final.to_csv('usuarios_filtrados.json')
        print(dic_final)
    except:
        valor1=subprocess.call(["echo", "escaneando..."])
```

A.4 Obtener ubicación

El siguiente script de Python realiza la comunicación serial entre el modulo GPS y la raspberry obteniendo los datos de latitud y longitud transmitidas por el GPS.

```
import time
import serial
import pandas as pd
import subprocess
from datetime import date
from datetime import datetime
while True:
  try:
    def readString():
      while 1:
        while ser.read().decode("utf-8") != '$':  # Wait for the begging of the string
          pass  # Do nothing
        line = ser.readline().decode("utf-8")  # Read the entire string
        return line


    def getTime(string, format, returnFormat):
      return time.strftime(returnFormat,
                           time.strptime(string, format))  # Convert date and time to a nice printable format
    def getLatLng(latString, lngString):
      lat = latString[:2].lstrip('0') + "." + "%.12s" % str(float(latString[2:]) * 1.0 / 60.0).lstrip("0.")
      lng = lngString[:3].lstrip('0') + "." + "%.12s" % str(float(lngString[3:]) * 1.0 / 60.0).lstrip("0.")
      return lat, lng
    def printGGA(lines):
      latlng = getLatLng(lines[2], lines[4])
      a= float(latlng[0])
      b= float(latlng[1])
```

```
        latitud = -a
        longitud = -b
        now = datetime.now()
        hora = now.strftime('%H:%M:%S')
        dic={'latitud':[latitud], 'longitud':[longitud],'hora':[hora]}
        dic_gps=pd.DataFrame(dic)
        dic_gps.to_csv('gps.csv')
        print(dic_gps)
        return
    if __name__ == '__main__':
        ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  # Open Serial port
        try:
            while True:
                line = readString()
                lines = line.split(",")
                if lines[0] == "GPGGA":
                        printGGA(lines)
        except KeyboardInterrupt:
                print('Exiting Script')
except:
    print('obteniendo datos...')
```

A.5 Vinculación de la información

El script vincula la información de los datos y los guarda en formato JSON:

```
import time
import pandas as pd
from datetime import date
from datetime import datetime
while True:
    try:
        time.sleep(1)
```

```
        dataset_ubicacion=pd.read_csv("/home/kali/Desktop/gps.csv")
        dataset_hora=pd.read_csv("/home/kali/Desktop/hora_actual.json")
        dataset_users=pd.read_csv("/home/kali/Desktop/usuarios_filtrados.json")
        dic_ubicacion = pd.DataFrame(dataset_ubicacion)
        dic_hora = pd.DataFrame(dataset_hora)
        dic_users = pd.DataFrame(dataset_users)
        dic_final=pd.concat([dic_ubicacion, dic_hora, dic_users], axis=1)
        dic_final.to_csv('informacion.json')
        print(dic_final)
    except:
        print('obteniendo datos...')
```

A.6 Publicación de la información

El siguiente script de Python se conecta el bróker “emqx” con la ayuda de la librería paho-mqtt y publica la información en el tema “python/bus_canar/usuarios”.

```
import random
import time
import pandas as pd
import json
from paho.mqtt import client as mqtt_client
import subprocess
broker = 'broker.emqx.io'
port = 1883
topic = "python/bus_canar/usuarios"
client_id = "Cliente_Datos"
username = 'kali_datos'
password = 'kali_datos'


try:
    def connect_mqtt():
        def on_connect(client, userdata, flags, rc):
```

```
            if rc == 0:
                print("Conectado al broker MQTT")
            else:
                print("Fallo la conexion")
                    #print("Failed to connect, return code %d\n", rc)
        client = mqtt_client.Client(client_id)
        client.username_pw_set(username, password)
        client.on_connect = on_connect
        client.connect(broker, port)
        return client
    def publish(client):
        msg_count = 0
        data=pd.read_csv("/home/kali/Desktop/usuarios_filtrados.json")
        data2=data.to_json()
        data_out=json.dumps(data2)
        msg = data_out
        result = client.publish(topic, msg, qos=0, retain=False)
        print({msg}," to topic" ,{topic})



    def run():
        client = connect_mqtt()
        client.loop_start()
        publish(client)
    if __name__ == '__main__':
        run()
except:
    print ('fallo la conexion')
```

APÉNDICE B: BACK END

B.1 Implementación mqtt en el servidor y procesamiento de datos

Las siguientes líneas de código están escritas en JavaScript, implementan el modulo MQTT y se subscriben al tema “python/bus_canar/usuarios” para recibir los mensajes publicados por la raspberry. Procesa los datos pasando de formato JSON a objeto JavaScript.

```
const mqtt = require('mqtt');
let auxRealTime=null;
//parametros para la implementacion del cliente mqtt
const options = {
  // Clean session
  clean: true,
  //keepalive: 0,
  connectTimeout: 1000,
  // Auth
  clientId: 'Cliente_Users1',
  username: 'kali_datos1.1',
  password: 'kali_datos1.1',
  qos: 2,
}
//declaramos el cliente mqtt
const client  = mqtt.connect('mqtt://broker.emqx.io:1883', options, qos=2)
//nos subscribimos al tema mqtt
client.on('connect', function () {
  console.log('Conectado al broker')
  client.subscribe('python/bus_canar/usuarios1',{ qos: 2}, function (err) {
  })
})
//recibimos el mensaje en formato JSON
client.on('message', async function(topic, message) {
  const realTime = JSON.parse(message);
```

```
  let realTime =  client.getRealTime();
  const aux = JSON.parse(realTime);
  const {usuarios, fecha, weekday, latitud, longitud, hora} = aux; //procesamiento de datos
  auxRealTime = realTime;
});
//funcion para exportar la informacion
client.getRealTime = ()=>{
  try{
    //console.log('Informacion: ',auxRealTime);
    if(auxRealTime==null){
        auxRealTime=0;
    }
    return auxRealTime;
  }
  catch(e){
     console.log(e);
  }
};
module.exports = client;
```

B.2 Implementación servidor web y creación de rutas

El siguiente código implementa el servidor web y crea las rutas del proyecto:

```
const express = require('express');
const path = require('path');
//inicializar
const app = express();
require('./lib/passport');
//settings
app.set('port', '4000');
app.set('json spaces', 2)
app.set('views', path.join(__dirname,'views'));
```

```
//middlewares >> se ejecutan cuando un usuario envia una peticion al servidor
app.use(session({
    secret: 'danny',
    resave: 'false',
    saveUninitialized: false,
    store: new MySQLStore(database)
}));
//Routes >> URLs de nuestro servidor, que es lo que van a hacer cuando un usuario visite esa URL
const index = require('./routes/index');
const auth = require('./routes/autenticar');
const consultas = require('./routes/consultas');
const api_realtime = require('./routes/api/consulta_tiemporeal')
const train_data = require('./lib/pruebaLSTM')
app.use('/', index);
app.use('/auth', auth);
app.use('/consultas', consultas);
app.use('/api/tiempo_real', api_realtime);
app.use('/api/train_data', train_data)
//app.use('/api/pos_real', api_posrealtime);
//Public
app.use(express.static(path.join(__dirname,'public')));
//Starting server
app.listen(app.get('port'), ()=>{
    console.log("server en el puerto: ",app.get('port'))
})
```

B.3 Conexión a la base de datos

El siguiente código conecta el servidor con la base de datos, muestra también el código para el almacenamiento de datos.

```
const mysql = require('mysql');
const {promisify}=require('util');
```

```
const {database} = require('./keys');
const pool = mysql.createPool(database);
pool.getConnection((err, connection)=>{
   if(err){
      if(err.code==='PROTOCOL_CONNECTION_LOST'){
         console.error('DATABASE CONNECTION WAS CLOSED');
      }
      if(err.code==='ER_CON_COUNT_ERROR'){
         console.error('DATABASE HAS TO MANY CONNECTIONS');
      }
      if(err.code==='ENCONNREFUSED'){
         console.error('DATABASE CONNECTION WAS REFUSED');
      }
   }
   if(connection)connection.release();
   console.log('Conectado a la base de datos');
   return;
});
//callback a promesas
pool.query=promisify(pool.query);
module.exports = pool;
```

B.4 Almacenamiento en la base de datos

Código para almacenar datos:

```
const { on } = require('../database');
const pool = require('../database');
const save = {}
const { getRealTime } = require('C:/Users/Usuario/Desktop/Proyecto_web/source/lib/mqtt');
const client = require('C:/Users/Usuario/Desktop/Proyecto_web/source/lib/mqtt');
const data_save = async()=>{
   let realTime = client.getRealTime();
```

```
        const results = JSON.parse(realTime);
        if (results != 0) {
            var usuarios = Object.values(results.usuarios);
            var fecha = Object.values(results.fecha);
            var weekday = Object.values(results.weekday); //0-6, 0 es lunes, 6 es domingo
            var latitud = Object.values(results.latitud);
            var longitud = Object.values(results.longitud);
            var hora = Object.values(results.hora); //hora de los usuarios
             await pool.query(`INSERT INTO sensor2 (usuarios, latitud, longitud, hora, fecha, weekday) VALUES ("${usuarios}","${latitud}","${longitud}","${hora}","${fecha}", "${weekday}")`)
            console.log('datos guardados en db');
        }
        setTimeout(()=>{
            data_save();
        }, 2000);
    }
    setTimeout(()=>{
        data_save();
    }, 1000);
    module.exports = save;
```

B.5 Consulta de datos

El siguiente código JavaScript consulta a la base de datos, según la información enviada por el cliente en el método POST.

```
    router.post('/', async(req,res)=>{
        const {fecha} = req.body;
        const newConsulta ={fecha};
        const {hora} = req.body;
        const consulta_hora = {hora};
       const datos = await pool.query("SELECT DISTINCTROW  usuarios, latitud, longitud, hora, fecha FROM transporte_cañar.sensor2 WHERE (?)"+"ORDER BY hora ASC",[newConsulta]);
```

```
        res.render("C:/Users/Usuario/Desktop/Proyecto_web/source/views/consultas/consultar_fecha.hbs", {filter1});
        auxData = filter1;
    })
```

APÉNDICE C: FRONT END

C.1 Modulo leaflet

El siguiente código JavaScript programa el módulo leaflet.

```
var map = L.map('map-template').setView([-2.5667505488441154, -78.9372192229309], 13);
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);
        setTimeout(()=>{
            getPosition()
        }, 1000);
        var marker, circle;
        function getPosition(){
            // console.log(position)
            if(marker) {
                map.removeLayer(marker)
            }
            if(circle) {
                map.removeLayer(circle)
            }
                                marker = L.marker([temp_lat, temp_long]).bindPopup('Usuarios: '+temp_user).addTo(map)
            circle = L.circle([temp_lat, temp_long], {radius: 30}).addTo(map)
            marker.bindPopup("<b>Bus Linea 1</b><br>Usuarios ≈ "+temp_user).openPopup();
            console.log("Ubucacion actual Lat:"+ temp_lat +" Long: "+ temp_long)
            setTimeout(()=>{
                getPosition()
            }, 1000);
        }
```

C.2 Modulo canvas

El siguiente código JavaScript programa el módulo leaflet canvas.

```
//GRAFICA PLOTS
window.onload = function () {
   var dataPoints1 = [];
   var chart = new CanvasJS.Chart("chartContainer", {
      zoomEnabled: true,
      title: {
         text: "Datos en tiempo real"
      },
      axisX: {
         title: "Tiempo"
      },
      axisY:{
         prefix: ""
      },
      toolTip: {
         shared: true
      },
      legend: {
         cursor:"pointer",
         verticalAlign: "top",
         fontSize: 22,
         fontColor: "dimGrey",
         itemclick : toggleDataSeries
      },
      data: [{
         type: "area",
         xValueType: "dateTime",
         xValueFormatString: "hh:mm:ss TT",
```

```
        showInLegend: true,
        name: "Usuarios",
        dataPoints: dataPoints1
        },
        ]
});
function toggleDataSeries(e) {
    if (typeof(e.dataSeries.visible) === "undefined" || e.dataSeries.visible) {
        e.dataSeries.visible = false;
    }
    else {
        e.dataSeries.visible = true;
    }
    chart.render();
}
var updateInterval = 3000;
var time = new Date;
var horas = time.getHours()
var minutos = time.getMinutes()
var segundos = time.getSeconds()
time.setHours(horas);
time.setMinutes(minutos);
time.setSeconds(segundos);
time.setMilliseconds(00);
setTimeout(()=>{
    updateChart();
}, 3000);
function updateChart(){
    //console.log('obtener datos');
    time.setTime(time.getTime()+updateInterval);
```

```
    dataPoints1.push({
        x: time.getTime(),
        y: temp_user
    });
    // updating legend text with  updated with y Value
    if(temp_user==null){
        chart.options.data[0].legendText = " Calculando... ";
    }
    chart.options.data[0].legendText = " Usuarios ≈ " + temp_user;
    chart.render();
}
// generates first set of dataPoints
updateChart(0);
setInterval(function(){updateChart()}, updateInterval);
}
```

C.3 Obtener la parada de la unidad de transporte

Las siguientes líneas de código se usaron para determinar las paradas de la unidad de transporte.

```
//Distancia entre la parada y las coordenadas transmitidas
        function getEstacion(lat1, lon1, temp_lat, temp_long) {
            var R = 6371; // Radius of the earth in km
            var dLat = deg2rad(temp_lat-lat1);  // deg2rad below
            var dLon = deg2rad(temp_long-lon1);
            var a =
              Math.sin(dLat/2) * Math.sin(dLat/2) +
              Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(temp_lat)) *
              Math.sin(dLon/2) * Math.sin(dLon/2)
              ;
            var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
            var d = R * c * 1000; // Distance in m
```

```
    /*setTimeout(()=>{
        getEstacion()
    }, 1000); */
    return d;
}
function deg2rad(deg) {
    return deg * (Math.PI/180)
}
/*setTimeout(()=>{
    getEstacion()
}, 1000);*/
//PARADAS
var casa = getEstacion(-2.569337, -78.934441,temp_lat, temp_long);
var terminal = getEstacion(-2.567881, -78.937982,temp_lat, temp_long);
var parada1 = getEstacion(-2.569336, -78.934417,temp_lat, temp_long);
var parada2 = getEstacion(-2.567881, -78.934786,temp_lat, temp_long);
var parada3 = getEstacion(-2.566116, -78.935212,temp_lat, temp_long);
var parada4 = getEstacion(-2.559551, -78.935824,temp_lat, temp_long);
var parada5 = getEstacion(-2.557718, -78.937033,temp_lat, temp_long);
var parada6 = getEstacion(-2.554750, -78.939009,temp_lat, temp_long);
var parada7R1 = getEstacion(-2.551255, -78.937670,temp_lat, temp_long);
var parada8R1 = getEstacion(-2.543551, -78.933896, temp_lat, temp_long);
var parada9R1 = getEstacion(-2.539966, -78.932573,temp_lat, temp_long);
var parada10R1 = getEstacion(-2.536033, -78.932777,temp_lat, temp_long);
var parada7R2 = getEstacion(-2.553626,-78.940478, temp_lat, temp_long);
var parada8R2 = getEstacion(-2.552546,-78.940080, temp_lat, temp_long);
var parada8R21 = getEstacion(-2.552056,-78.939977,temp_lat, temp_long);
var parada9R2 = getEstacion(-2.549754,-78.939030,temp_lat, temp_long);
var parada10R2 = getEstacion( -2.546509, -78.938182, temp_lat, temp_long);
var parada7R3 = getEstacion(-2.555773, -78.943670,temp_lat, temp_long);
```

```
var parada8R3 = getEstacion(-2.561323, -78.946693, temp_lat, temp_long);
var parada9R3 = getEstacion(-2.567691, -78.949429, temp_lat, temp_long);
var parada10R3 = getEstacion(-2.556250, -78.940831, temp_lat, temp_long);
var parada11 = getEstacion(-2.557860, -78.937048, temp_lat, temp_long);
var parada12 = getEstacion(-2.559569, -78.935917, temp_lat, temp_long);
var parada13 = getEstacion(-2.561541, -78.936162, temp_lat, temp_long);
var parada14 = getEstacion(-2.563372, -78.937829, temp_lat, temp_long);
var parada15 = getEstacion(-2.564079, -78.938535, temp_lat, temp_long);
console.log(terminal);
if(casa<=1){
   parada.innerText ='Casa Juan Javier';
}
else if(terminal<=100){
   parada.innerText ='Terminal Terrestre';
}
else if(parada1<=30){
   parada.innerText ='Parada 1';
}
else if(parada2<=30){
   parada.innerText ='Parada 2';
}
else if(parada3<=30){
   parada.innerText ='Parada 3';
}
else if(parada4<=15){
   parada.innerText ='Parada 4';
}
else if(parada5<=15){
   parada.innerText ='Parada 5';
}
```

```
else if(parada6<=60){
   parada.innerText ='Parada 6';
}
else if(parada7R1<=30){
   parada.innerText ='Parada 7';
}
else if(parada8R1<=30){
   parada.innerText ='Parada 8';
}
else if(parada9R1<=30){
   parada.innerText ='Parada 9';
}
else if(parada10R1<=30){
   parada.innerText ='Parada 10';
}
else if(parada7R2<=30){
   parada.innerText ='Parada 7';
}
else if(parada8R2<=30){
   parada.innerText ='Parada 8';
}
else if(parada8R21<=30){
   parada.innerText ='Parada 8';
}
else if(parada9R2<=30){
   parada.innerText ='Parada 9';
}
else if(parada10R2<=60){
   parada.innerText ='Parada 10';
}
```

```
else if(parada7R3<=30){
   parada.innerText ='Parada 7';
}
else if(parada8R3<=30){
   parada.innerText ='Parada 8';
}
else if(parada9R3<=30){
   parada.innerText ='Parada 9';
}
else if(parada10R3<=30){
   parada.innerText ='Parada 10';
}
else if(parada11<=15){
   parada.innerText ='Parada 11';
}
else if(parada12<=15){
   parada.innerText ='Parada 12';
}
else if(parada13<=30){
   parada.innerText ='Parada 13';
}
else if(parada14<=30){
   parada.innerText ='Parada 14';
}
else if(parada15<=30){
   parada.innerText ='Parada 15';
}
else{
   parada.innerText ='Viajando . . . ';
}
```

APÉNDICE D: SISTEMA DE PREDICCIÓN

D.1 Tratamiento de datos

Para el tratamiento de datos se utilizó Python con la librería pandas, las siguientes líneas de código realizan el tratamiento de datos:

```
import pandas as pd

import os

import matplotlib.pyplot as plt

import seaborn as sns

jueves26_05=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/jueves26_05.json")

viernes27_05=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/viernes27_05.json")

#SEMANA 1

lunes30_05=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/lunes30_05.json")

martes31_05=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/martes31_05.json")

miercoles01_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/miercoles01_06.json")

jueves02_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/jueves02_06.json")

viernes03_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/viernes03_06.json")

#SEMANA 2

lunes06_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/lunes06_06.json")

martes07_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/martes07_06.json")

miercoles08_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/miercoles08_06.json")

jueves09_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/jueves09_06.json")
```

viernes10_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/viernes10_06.json")

#SEMANA3

lunes13_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/lunes13_06.json")

martes14_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/martes14_06.json")

miercoles15_06=pd.read_json("/Users/Usuario/Desktop/Curso-Python/Jupyter/datasets/pasajeros/miercoles15_06.json")

#extraemos solo los datos de las 11-13, son las horas que dejamos en la unidad de transporte y lo que dura la bateria de la raspberry

jue2605_h11am = jueves26_05[(jueves26_05["hora"]==(11))]

jue2605_h12am = jueves26_05[(jueves26_05["hora"]==(12))]

jue2605_h13am = jueves26_05[(jueves26_05["hora"]==(13))]

vie2705_h11am = viernes27_05[(viernes27_05["hora"]==(11))]

vie2705_h12am = viernes27_05[(viernes27_05["hora"]==(12))]

vie2705_h13am = viernes27_05[(viernes27_05["hora"]==(13))]

lunes30_05_h11am = lunes30_05[(lunes30_05["hora"]==(11))]

lunes30_05_h12am = lunes30_05[(lunes30_05["hora"]==(12))]

lunes30_05_h13am = lunes30_05[(lunes30_05["hora"]==(13))]

martes31_05_h11am = martes31_05[(martes31_05["hora"]==(11))]

martes31_05_h12am = martes31_05[(martes31_05["hora"]==(12))]

martes31_05_h13am = martes31_05[(martes31_05["hora"]==(13))]

miercoles01_06_h11am = miercoles01_06[(miercoles01_06["hora"]==(11))]

miercoles01_06_h12am = miercoles01_06[(miercoles01_06["hora"]==(12))]

miercoles01_06_h13am = miercoles01_06[(miercoles01_06["hora"]==(13))]

jueves02_06_h11am = jueves02_06[(jueves02_06["hora"]==(11))]

jueves02_06_h12am = jueves02_06[(jueves02_06["hora"]==(12))]

jueves02_06_h13am = jueves02_06[(jueves02_06["hora"]==(13))]

viernes03_06_h11am = viernes03_06[(viernes03_06["hora"]==(11))]

viernes03_06_h12am = viernes03_06[(viernes03_06["hora"]==(12))]

```
viernes03_06_h13am = viernes03_06[(viernes03_06["hora"]==(13))]
lunes06_06_h11am = lunes06_06[(lunes06_06["hora"]==(11))]
lunes06_06_h12am = lunes06_06[(lunes06_06["hora"]==(12))]
lunes06_06_h13am = lunes06_06[(lunes06_06["hora"]==(13))]
martes07_06_h11am = martes07_06[(martes07_06["hora"]==(11))]
martes07_06_h12am = martes07_06[(martes07_06["hora"]==(12))]
martes07_06_h13am = martes07_06[(martes07_06["hora"]==(13))]
miercoles08_06_h11am = miercoles08_06[(miercoles08_06["hora"]==(11))]
miercoles08_06_h12am = miercoles08_06[(miercoles08_06["hora"]==(12))]
miercoles08_06_h13am = miercoles08_06[(miercoles08_06["hora"]==(13))]
jueves09_06_h11am = jueves09_06[(jueves09_06["hora"]==(11))]
jueves09_06_h12am = jueves09_06[(jueves09_06["hora"]==(12))]
jueves09_06_h13am = jueves09_06[(jueves09_06["hora"]==(13))]
viernes10_06_h11am = viernes10_06[(viernes10_06["hora"]==(11))]
viernes10_06_h12am = viernes10_06[(viernes10_06["hora"]==(12))]
viernes10_06_h13am = viernes10_06[(viernes10_06["hora"]==(13))]
lunes13_06_h11am = lunes13_06[(lunes13_06["hora"]==(11))]
lunes13_06_h12am = lunes13_06[(lunes13_06["hora"]==(12))]
lunes13_06_h13am = lunes13_06[(lunes13_06["hora"]==(13))]
martes14_06_h11am = martes14_06[(martes14_06["hora"]==(11))]
martes14_06_h12am = martes14_06[(martes14_06["hora"]==(12))]
martes14_06_h13am = martes14_06[(martes14_06["hora"]==(13))]
miercoles15_06_h11am = miercoles15_06[(miercoles15_06["hora"]==(11))]
miercoles15_06_h12am = miercoles15_06[(miercoles15_06["hora"]==(12))]
miercoles15_06_h13am = miercoles15_06[(miercoles15_06["hora"]==(13))]
#eliminamos los segundos y nos quedamos solo con los datos por minuto, para eliminar los segundos en la columna minutos
#eliminamos los duplicados
jue2605_h11am_norep = jue2605_h11am.drop_duplicates(subset="minutos",keep="first")
jue2605_h12am_norep = jue2605_h12am.drop_duplicates(subset="minutos",keep="first")
```

```
jue2605_h13am_norep = jue2605_h13am.drop_duplicates(subset="minutos",keep="first")
vie2705_h11am_norep = vie2705_h11am.drop_duplicates(subset="minutos",keep="first")
vie2705_h12am_norep = vie2705_h12am.drop_duplicates(subset="minutos",keep="first")
vie2705_h13am_norep = vie2705_h13am.drop_duplicates(subset="minutos",keep="first")
lunes30_05_h11am_norep = lunes30_05_h11am.drop_duplicates(subset="minutos",keep="first")
lunes30_05_h12am_norep = lunes30_05_h12am.drop_duplicates(subset="minutos",keep="first")
lunes30_05_h13am_norep = lunes30_05_h13am.drop_duplicates(subset="minutos",keep="first")
martes31_05_h11am_norep = martes31_05_h11am.drop_duplicates(subset="minutos",keep="first")
martes31_05_h12am_norep = martes31_05_h12am.drop_duplicates(subset="minutos",keep="first")
martes31_05_h13am_norep = martes31_05_h13am.drop_duplicates(subset="minutos",keep="first")


miercoles01_06_h11am_norep = miercoles01_06_h11am.drop_duplicates(subset="minutos",keep="first")
miercoles01_06_h12am_norep = miercoles01_06_h12am.drop_duplicates(subset="minutos",keep="first")
miercoles01_06_h13am_norep = miercoles01_06_h13am.drop_duplicates(subset="minutos",keep="first")
jueves02_06_h11am_norep = jueves02_06_h11am.drop_duplicates(subset="minutos",keep="first")
jueves02_06_h12am_norep = jueves02_06_h12am.drop_duplicates(subset="minutos",keep="first")
jueves02_06_h13am_norep = jueves02_06_h13am.drop_duplicates(subset="minutos",keep="first")
viernes03_06_h11am_norep = viernes03_06_h11am.drop_duplicates(subset="minutos",keep="first")
viernes03_06_h12am_norep = viernes03_06_h12am.drop_duplicates(subset="minutos",keep="first")
viernes03_06_h13am_norep = viernes03_06_h13am.drop_duplicates(subset="minutos",keep="first")
lunes06_06_h11am_norep = lunes06_06_h11am.drop_duplicates(subset="minutos",keep="first")
lunes06_06_h12am_norep = lunes06_06_h12am.drop_duplicates(subset="minutos",keep="first")
lunes06_06_h13am_norep = lunes06_06_h13am.drop_duplicates(subset="minutos",keep="first")
```

```
    martes07_06_h11am_norep = martes07_06_h11am.drop_duplicates(subset="minutos",keep="first")
    martes07_06_h12am_norep = martes07_06_h12am.drop_duplicates(subset="minutos",keep="first")
    martes07_06_h13am_norep = martes07_06_h13am.drop_duplicates(subset="minutos",keep="first")
    miercoles08_06_h11am_norep = miercoles08_06_h11am.drop_duplicates(subset="minutos",keep="first")
    miercoles08_06_h12am_norep = miercoles08_06_h12am.drop_duplicates(subset="minutos",keep="first")
    miercoles08_06_h13am_norep = miercoles08_06_h13am.drop_duplicates(subset="minutos",keep="first")
    jueves09_06_h11am_norep = jueves09_06_h11am.drop_duplicates(subset="minutos",keep="first")
    jueves09_06_h12am_norep = jueves09_06_h12am.drop_duplicates(subset="minutos",keep="first")
    jueves09_06_h13am_norep = jueves09_06_h13am.drop_duplicates(subset="minutos",keep="first")
    viernes10_06_h11am_norep = viernes10_06_h11am.drop_duplicates(subset="minutos",keep="first")
    viernes10_06_h12am_norep = viernes10_06_h12am.drop_duplicates(subset="minutos",keep="first")
    viernes10_06_h13am_norep = viernes10_06_h13am.drop_duplicates(subset="minutos",keep="first")
    lunes13_06_h11am_norep = lunes13_06_h11am.drop_duplicates(subset="minutos",keep="first")
    lunes13_06_h12am_norep = lunes13_06_h12am.drop_duplicates(subset="minutos",keep="first")
    lunes13_06_h13am_norep = lunes13_06_h13am.drop_duplicates(subset="minutos",keep="first")
    martes14_06_h11am_norep = martes14_06_h11am.drop_duplicates(subset="minutos",keep="first")
    martes14_06_h12am_norep = martes14_06_h12am.drop_duplicates(subset="minutos",keep="first")
    martes14_06_h13am_norep = martes14_06_h13am.drop_duplicates(subset="minutos",keep="first")
    miercoles15_06_h11am_norep = miercoles15_06_h11am.drop_duplicates(subset="minutos",keep="first")
    miercoles15_06_h12am_norep = miercoles15_06_h12am.drop_duplicates(subset="minutos",keep="first")
```

```
    miercoles15_06_h13am_norep = miercoles15_06_h13am.drop_duplicates(subset="minutos",keep="first")

    #Despues de eliminar los segundos, y dejar los datos medidos por minuto

    #agrupamos nuevamente las horas

    datos_organizados_jue2605 = pd.concat([jue2605_h11am_norep, jue2605_h12am_norep, jue2605_h13am_norep], axis = 0)

    datos_organizados_vie2705 = pd.concat([vie2705_h11am_norep, vie2705_h12am_norep, vie2705_h13am_norep], axis = 0)

    datos_organizados_lunes30_05 = pd.concat([lunes30_05_h11am_norep, lunes30_05_h12am_norep, lunes30_05_h13am_norep], axis = 0)

    datos_organizados_martes31_05 = pd.concat([martes31_05_h11am_norep, martes31_05_h12am_norep, martes31_05_h13am_norep], axis = 0)

    datos_organizados_miercoles01_06 = pd.concat([miercoles01_06_h11am_norep, miercoles01_06_h12am_norep, miercoles01_06_h13am_norep], axis = 0)

    datos_organizados_jueves02_06 = pd.concat([jueves02_06_h11am_norep, jueves02_06_h12am_norep, jueves02_06_h13am_norep], axis = 0)

    datos_organizados_viernes03_06 = pd.concat([viernes03_06_h11am_norep, viernes03_06_h12am_norep, viernes03_06_h13am_norep], axis = 0)

    datos_organizados_lunes06_06 = pd.concat([lunes06_06_h11am_norep, lunes06_06_h12am_norep, lunes06_06_h13am_norep], axis = 0)

    datos_organizados_martes07_06 = pd.concat([martes07_06_h11am_norep, martes07_06_h12am_norep, martes07_06_h13am_norep], axis = 0)

    datos_organizados_miercoles08_06 = pd.concat([miercoles08_06_h11am_norep, miercoles08_06_h12am_norep, miercoles08_06_h13am_norep], axis = 0)

    datos_organizados_jueves09_06 = pd.concat([jueves09_06_h11am_norep, jueves09_06_h12am_norep, jueves09_06_h13am_norep], axis = 0)

    datos_organizados_viernes10_06 = pd.concat([viernes10_06_h11am_norep, viernes10_06_h12am_norep, viernes10_06_h13am_norep], axis = 0)

    datos_organizados_lunes13_06 = pd.concat([lunes13_06_h11am_norep, lunes13_06_h12am_norep, lunes13_06_h13am_norep], axis = 0)

    datos_organizados_martes14_06 = pd.concat([martes14_06_h11am_norep, martes14_06_h12am_norep, martes14_06_h13am_norep], axis = 0)

    datos_organizados_miercoles15_06 = pd.concat([miercoles15_06_h11am_norep, miercoles15_06_h12am_norep, miercoles15_06_h13am_norep], axis = 0)
```

#despues de tener todos los minutos de las horas 11-13, procedemos a completar los usuarios faltantes en los minutos

```
datatotrain_jue2605["weekday"] = datatotrain_jue2605["weekday"].fillna(method='ffill')
datatotrain_jue2605["usuarios"] = datatotrain_jue2605["usuarios"].fillna(method='ffill')
datatotrain_jue2605["latitud"] = datatotrain_jue2605["latitud"].fillna(method='ffill')
datatotrain_jue2605["longitud"] = datatotrain_jue2605["longitud"].fillna(method='ffill')
datatotrain_vie2705["weekday"] = datatotrain_vie2705["weekday"].fillna(method='ffill')
datatotrain_vie2705["usuarios"] = datatotrain_vie2705["usuarios"].fillna(method='ffill')
datatotrain_vie2705["latitud"] = datatotrain_vie2705["latitud"].fillna(method='ffill')
datatotrain_vie2705["longitud"] = datatotrain_vie2705["longitud"].fillna(method='ffill')
datatotrain_vie2705["weekday"] = datatotrain_vie2705["weekday"].fillna(method='backfill')
datatotrain_vie2705["usuarios"] = datatotrain_vie2705["usuarios"].fillna(method='backfill')
datatotrain_vie2705["latitud"] = datatotrain_vie2705["latitud"].fillna(method='backfill')
datatotrain_vie2705["longitud"] = datatotrain_vie2705["longitud"].fillna(method='backfill')
datatotrain_lunes30_05["weekday"] = datatotrain_lunes30_05["weekday"].fillna(method='ffill')
datatotrain_lunes30_05["usuarios"] = datatotrain_lunes30_05["usuarios"].fillna(method='ffill')
datatotrain_lunes30_05["latitud"] = datatotrain_lunes30_05["latitud"].fillna(method='ffill')
datatotrain_lunes30_05["longitud"] = datatotrain_lunes30_05["longitud"].fillna(method='ffill')
datatotrain_martes31_05["weekday"] = datatotrain_martes31_05["weekday"].fillna(method='ffill')
datatotrain_martes31_05["usuarios"] = datatotrain_martes31_05["usuarios"].fillna(method='ffill')
datatotrain_martes31_05["latitud"] = datatotrain_martes31_05["latitud"].fillna(method='ffill')
datatotrain_martes31_05["longitud"] = datatotrain_martes31_05["longitud"].fillna(method='ffill')
datatotrain_martes31_05["weekday"] = datatotrain_martes31_05["weekday"].fillna(method='backfill')
datatotrain_martes31_05["usuarios"] = datatotrain_martes31_05["usuarios"].fillna(method='backfill')
datatotrain_martes31_05["latitud"] = datatotrain_martes31_05["latitud"].fillna(method='backfill')
datatotrain_martes31_05["longitud"] = datatotrain_martes31_05["longitud"].fillna(method='backfill')
datatotrain_miercoles01_06["weekday"] = datatotrain_miercoles01_06["weekday"].fillna(method='ffill')
```

```
    datatotrain_miercoles01_06["usuarios"] = datatotrain_miercoles01_06["usuarios"].fillna(method='ffill')
    datatotrain_miercoles01_06["latitud"] = datatotrain_miercoles01_06["latitud"].fillna(method='ffill')
    datatotrain_miercoles01_06["longitud"] = datatotrain_miercoles01_06["longitud"].fillna(method='ffill')
    datatotrain_miercoles01_06["weekday"] = datatotrain_miercoles01_06["weekday"].fillna(method='backfill')
    datatotrain_miercoles01_06["usuarios"] = datatotrain_miercoles01_06["usuarios"].fillna(method='backfill')
    datatotrain_miercoles01_06["latitud"] = datatotrain_miercoles01_06["latitud"].fillna(method='backfill')
    datatotrain_miercoles01_06["longitud"] = datatotrain_miercoles01_06["longitud"].fillna(method='backfill')
    datatotrain_jueves02_06["weekday"] = datatotrain_jueves02_06["weekday"].fillna(method='ffill')
    datatotrain_jueves02_06["usuarios"] = datatotrain_jueves02_06["usuarios"].fillna(method='ffill')
    datatotrain_jueves02_06["latitud"] = datatotrain_jueves02_06["latitud"].fillna(method='ffill')
    datatotrain_jueves02_06["longitud"] = datatotrain_jueves02_06["longitud"].fillna(method='ffill')
    datatotrain_jueves02_06["weekday"] = datatotrain_jueves02_06["weekday"].fillna(method='backfill')
    datatotrain_jueves02_06["usuarios"] = datatotrain_jueves02_06["usuarios"].fillna(method='backfill')
    datatotrain_jueves02_06["latitud"] = datatotrain_jueves02_06["latitud"].fillna(method='backfill')
    datatotrain_jueves02_06["longitud"] = datatotrain_jueves02_06["longitud"].fillna(method='backfill')
    datatotrain_viernes03_06["weekday"] = datatotrain_viernes03_06["weekday"].fillna(method='ffill')
    datatotrain_viernes03_06["usuarios"] = datatotrain_viernes03_06["usuarios"].fillna(method='ffill')
    datatotrain_viernes03_06["latitud"] = datatotrain_viernes03_06["latitud"].fillna(method='ffill')
    datatotrain_viernes03_06["longitud"] = datatotrain_viernes03_06["longitud"].fillna(method='ffill')
    datatotrain_viernes03_06["weekday"] = datatotrain_viernes03_06["weekday"].fillna(method='backfill')
    datatotrain_viernes03_06["usuarios"] = datatotrain_viernes03_06["usuarios"].fillna(method='backfill')
    datatotrain_viernes03_06["latitud"] = datatotrain_viernes03_06["latitud"].fillna(method='backfill')
```

```
    datatotrain_viernes03_06["longitud"] = datatotrain_viernes03_06["longitud"].fillna(method='backfill')


    datatotrain_lunes06_06["weekday"] = datatotrain_lunes06_06["weekday"].fillna(method='ffill')
    datatotrain_lunes06_06["usuarios"] = datatotrain_lunes06_06["usuarios"].fillna(method='ffill')
    datatotrain_lunes06_06["latitud"] = datatotrain_lunes06_06["latitud"].fillna(method='ffill')
    datatotrain_lunes06_06["longitud"] = datatotrain_lunes06_06["longitud"].fillna(method='ffill')
    datatotrain_lunes06_06["weekday"] = datatotrain_lunes06_06["weekday"].fillna(method='backfill')
    datatotrain_lunes06_06["usuarios"] = datatotrain_lunes06_06["usuarios"].fillna(method='backfill')
    datatotrain_lunes06_06["latitud"] = datatotrain_lunes06_06["latitud"].fillna(method='backfill')
    datatotrain_lunes06_06["longitud"] = datatotrain_lunes06_06["longitud"].fillna(method='backfill')
    datatotrain_martes07_06["weekday"] = datatotrain_martes07_06["weekday"].fillna(method='ffill')
    datatotrain_martes07_06["usuarios"] = datatotrain_martes07_06["usuarios"].fillna(method='ffill')
    datatotrain_martes07_06["latitud"] = datatotrain_martes07_06["latitud"].fillna(method='ffill')
    datatotrain_martes07_06["longitud"] = datatotrain_martes07_06["longitud"].fillna(method='ffill')
    datatotrain_martes07_06["weekday"] = datatotrain_martes07_06["weekday"].fillna(method='backfill')
    datatotrain_martes07_06["usuarios"] = datatotrain_martes07_06["usuarios"].fillna(method='backfill')
    datatotrain_martes07_06["latitud"] = datatotrain_martes07_06["latitud"].fillna(method='backfill')
    datatotrain_martes07_06["longitud"] = datatotrain_martes07_06["longitud"].fillna(method='backfill')
    datatotrain_miercoles08_06["weekday"] = datatotrain_miercoles08_06["weekday"].fillna(method='ffill')
    datatotrain_miercoles08_06["usuarios"] = datatotrain_miercoles08_06["usuarios"].fillna(method='ffill')
    datatotrain_miercoles08_06["latitud"] = datatotrain_miercoles08_06["latitud"].fillna(method='ffill')
    datatotrain_miercoles08_06["longitud"] = datatotrain_miercoles08_06["longitud"].fillna(method='ffill')
    datatotrain_jueves09_06["weekday"] = datatotrain_jueves09_06["weekday"].fillna(method='ffill')
    datatotrain_jueves09_06["usuarios"] = datatotrain_jueves09_06["usuarios"].fillna(method='ffill')
    datatotrain_jueves09_06["latitud"] = datatotrain_jueves09_06["latitud"].fillna(method='ffill')
```

```
datatotrain_jueves09_06["longitud"] = datatotrain_jueves09_06["longitud"].fillna(method='ffill')
datatotrain_viernes10_06["weekday"] = datatotrain_viernes10_06["weekday"].fillna(method='ffill')
datatotrain_viernes10_06["usuarios"] = datatotrain_viernes10_06["usuarios"].fillna(method='ffill')
datatotrain_viernes10_06["latitud"] = datatotrain_viernes10_06["latitud"].fillna(method='ffill')
datatotrain_viernes10_06["longitud"] = datatotrain_viernes10_06["longitud"].fillna(method='ffill')
datatotrain_viernes10_06["weekday"] = datatotrain_viernes10_06["weekday"].fillna(method='backfill')
datatotrain_viernes10_06["usuarios"] = datatotrain_viernes10_06["usuarios"].fillna(method='backfill')
datatotrain_viernes10_06["latitud"] = datatotrain_viernes10_06["latitud"].fillna(method='backfill')
datatotrain_viernes10_06["longitud"] = datatotrain_viernes10_06["longitud"].fillna(method='backfill')
datatotrain_lunes13_06["weekday"] = datatotrain_lunes13_06["weekday"].fillna(method='ffill')
datatotrain_lunes13_06["usuarios"] = datatotrain_lunes13_06["usuarios"].fillna(method='ffill')
datatotrain_lunes13_06["latitud"] = datatotrain_lunes13_06["latitud"].fillna(method='ffill')
datatotrain_lunes13_06["longitud"] = datatotrain_lunes13_06["longitud"].fillna(method='ffill')
datatotrain_lunes13_06["weekday"] = datatotrain_lunes13_06["weekday"].fillna(method='backfill')
datatotrain_lunes13_06["usuarios"] = datatotrain_lunes13_06["usuarios"].fillna(method='backfill')
datatotrain_lunes13_06["latitud"] = datatotrain_lunes13_06["latitud"].fillna(method='backfill')
datatotrain_lunes13_06["longitud"] = datatotrain_lunes13_06["longitud"].fillna(method='backfill')
datatotrain_martes14_06["weekday"] = datatotrain_martes14_06["weekday"].fillna(method='ffill')
datatotrain_martes14_06["usuarios"] = datatotrain_martes14_06["usuarios"].fillna(method='ffill')
datatotrain_martes14_06["latitud"] = datatotrain_martes14_06["latitud"].fillna(method='ffill')
datatotrain_martes14_06["longitud"] = datatotrain_martes14_06["longitud"].fillna(method='ffill')
datatotrain_martes14_06["weekday"] = datatotrain_martes14_06["weekday"].fillna(method='backfill')
datatotrain_martes14_06["usuarios"] = datatotrain_martes14_06["usuarios"].fillna(method='backfill')
datatotrain_martes14_06["latitud"] = datatotrain_martes14_06["latitud"].fillna(method='backfill')
datatotrain_martes14_06["longitud"] = datatotrain_martes14_06["longitud"].fillna(method='backfill')
```

```
    datatotrain_miercoles15_06["weekday"] = datatotrain_miercoles15_06["weekday"].fillna(method='ffill')
    datatotrain_miercoles15_06["usuarios"] = datatotrain_miercoles15_06["usuarios"].fillna(method='ffill')
    datatotrain_miercoles15_06["latitud"] = datatotrain_miercoles15_06["latitud"].fillna(method='ffill')
    datatotrain_miercoles15_06["longitud"] = datatotrain_miercoles15_06["longitud"].fillna(method='ffill')
    datatotrain_miercoles15_06["weekday"] = datatotrain_miercoles15_06["weekday"].fillna(method='backfill')
    datatotrain_miercoles15_06["usuarios"] = datatotrain_miercoles15_06["usuarios"].fillna(method='backfill')
    datatotrain_miercoles15_06["latitud"] = datatotrain_miercoles15_06["latitud"].fillna(method='backfill')
    datatotrain_miercoles15_06["longitud"] = datatotrain_miercoles15_06["longitud"].fillna(method='backfill')
    #datos para entrenar guardados en formatos json
    dic_jue2605.to_json('jue2605_totrain.json', orient = 'records')
    dic_vie2705.to_json('vie2705_totrain.json', orient = 'records')
    dic_lunes30_05.to_json('lunes30_05_totrain.json', orient = 'records')
    dic_martes31_05.to_json('martes31_05_totrain.json', orient = 'records')
    dic_miercoles01_06.to_json('miercoles01_06_totrain.json', orient = 'records')
    dic_jueves02_06.to_json('jueves02_06_totrain.json', orient = 'records')
    dic_viernes03_06.to_json('viernes03_06_totrain.json', orient = 'records')
    dic_lunes06_06.to_json('lunes06_06_totrain.json', orient = 'records')
    dic_martes07_06.to_json('martes07_06_totrain.json', orient = 'records')
    dic_miercoles08_06.to_json('miercoles08_06_totrain.json', orient = 'records')
    dic_jueves09_06.to_json('jueves09_06_totrain.json', orient = 'records')
    dic_viernes10_06.to_json('viernes10_06_totrain.json', orient = 'records')
    dic_lunes13_06.to_json('lunes13_06_totrain.json', orient = 'records')
    dic_martes14_06.to_json('martes14_06_totrain.json', orient = 'records')
    dic_miercoles15_06.to_json('miercoles15_06_totrain.json', orient = 'records')
    #concatenamos todos los datos para luego tener 70% para entrenamiento y 30% para test
```

```
datos_concatedanos = pd.concat([dic_jue2605, dic_vie2705, dic_lunes30_05, dic_martes31_05, dic_miercoles01_06,
                    dic_jueves02_06, dic_viernes03_06, dic_lunes06_06, dic_martes07_06,
                    dic_miercoles08_06, dic_jueves09_06, dic_viernes10_06, dic_lunes13_06,
                    dic_martes14_06, dic_miercoles15_06], axis = 0)
```

D.2 Entrenamiento de datos con BRAIN.JS

Para el entrenamiento de datos usamos código JavaScript con el módulo brain.js

```
const brain = require('brain.js')
var train_lunes = require('../lib/train_lunes.json');
var train_martes = require('../lib/train_martes.json');
var train_miercoles = require('../lib/train_miercoles.json');
var train_jueves = require('../lib/train_jueves.json');
var train_viernes = require('../lib/train_viernes.json');
var net_train = require('../lib/train_data.json');
const express = require('express');
const router = express.Router();
let predicion = null;
let auxData = null;
//datos para entrenar;
let aux = Object.values(train_viernes);
const train = aux.map(item => (
  [item.hora, item.minutos, item.usuarios]
));
//declaramos la red neuronal
const net = new brain.recurrent.LSTMTimeStep({
  inputSize: 3,
  hiddenLayers: [20],
  outputSize: 3,
});
```

```
const trainingData = [
  train
];
//entrenamos la red
/*net.train(trainingData, { log: true, errorThresh: 0.09 });
const networkState = net.toJSON();
auxData=networkState;*/
//cargamos la red previamente entrenada
function parseData(data) {
  if (!data) return {};
  if (typeof data === 'object') return data;
  if (typeof data === 'string') return JSON.parse(data);
  return {};
}
const net_save = parseData(net_train)
net.fromJSON(net_save);*/
router.get('/', async(req,res)=>{
  res.json(auxData)
})
router.get('/predict', async(req,res)=>{
  res.json(predicion)
})


module.exports = router;
```

APÉNDICE E: GRAFICAS DE RESULTADOS

E.1 Graficas de datos recolectados

Aquí podemos observar todo el recopilatorio de gráficas con la detección de usuarios de los días que fue implementado el módulo detector de civiles.

Ilustración E 1.1. Datos tomados del día 1. (Fuente: Autor)

Ilustración E 1.2. Datos tomados del día 2. (Fuente: Autor)

Ilustración E 1.3. Datos tomados del día 3. (Fuente: Autor)

Ilustración E 1.4. Datos tomados del día 4. (Fuente: Autor)

Ilustración E 1.5. Datos tomados del día 5. (Fuente: Autor)

Las siguientes ilustraciones muestran un error de recolección de datos desde el día 2 de junio hasta el día martes 7 de junio, el día en el que se pudo solucionar estos problemas.

Ilustración E 1.6. Datos tomados del día 6. (Fuente: Autor)

Ilustración E 1.7. tos tomados del día 7. (Fuente: Autor)

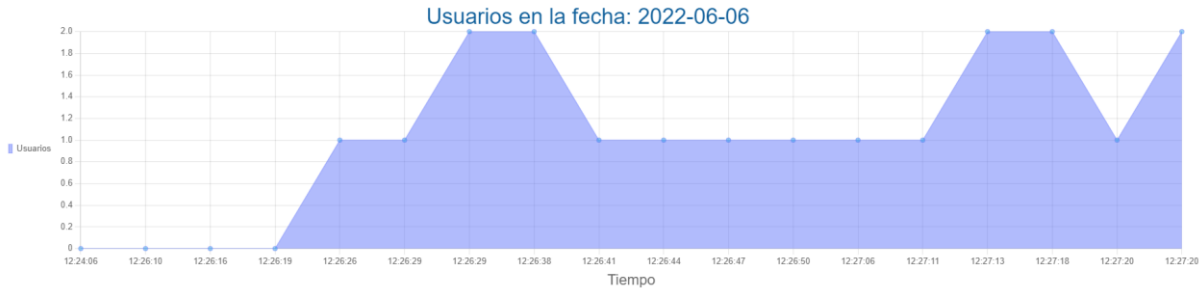


Ilustración E 1.8. Datos tomados del día 8. (Fuente: Autor)

Ilustración E 1.9. Datos tomados del día 9. (Fuente: Autor)

Ilustración E 1.10 Datos tomados del día 10. (Fuente: Autor)

Ilustración E 1.11. Datos tomados del día 11.

Ilustración E 1.12. Datos tomados del día 12. (Fuente: Autor)

Ilustración E 1.13. Datos tomados del día 13. (Fuente: Autor)

Ilustración E 1.14. Datos tomados del día 14. (Fuente: Autor)

Ilustración E 1.15. Datos tomados del día 15. (Fuente: Autor)

E.2 Predicciones

Las siguientes graficas muestran las predicciones obtenidas a partir de la red LSTM.

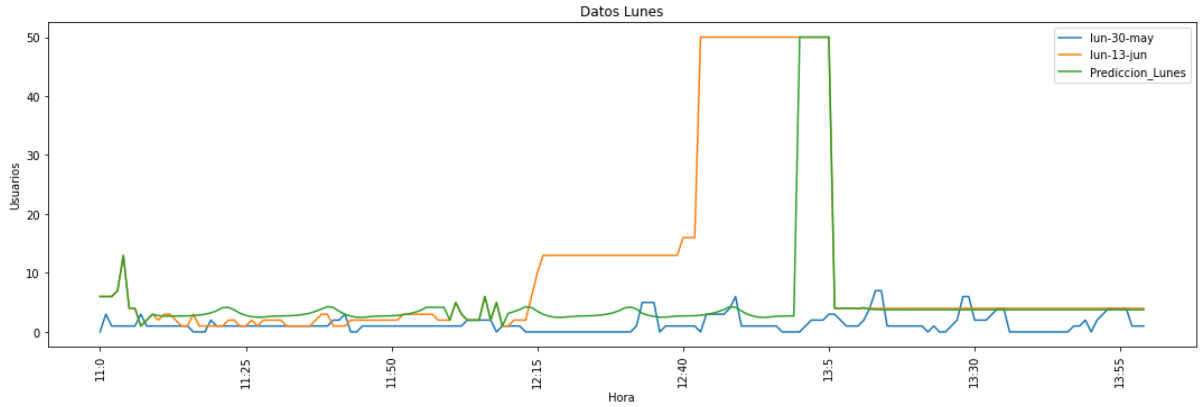


Ilustración E 2.1. Predicción día 1. (Fuente: Autor)

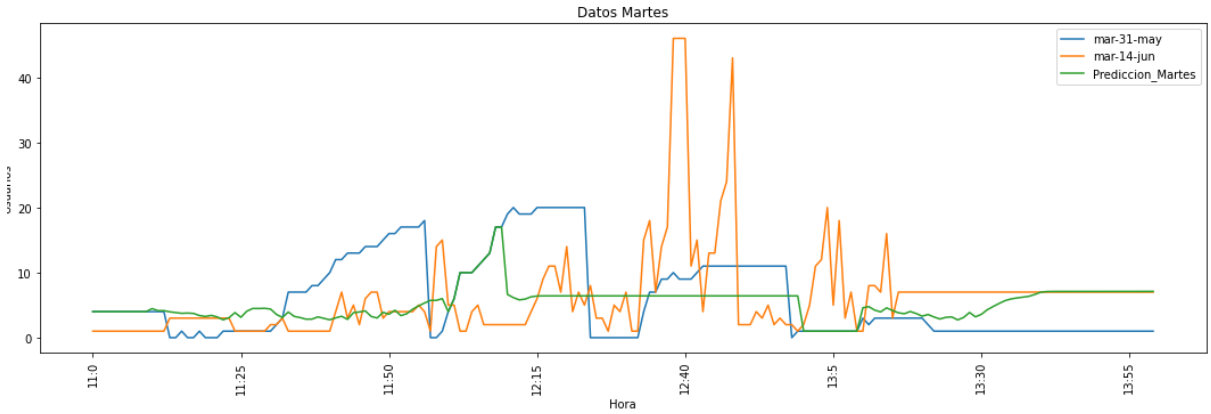


Ilustración E 2.2. Predicción día 2. (Fuente: Autor)

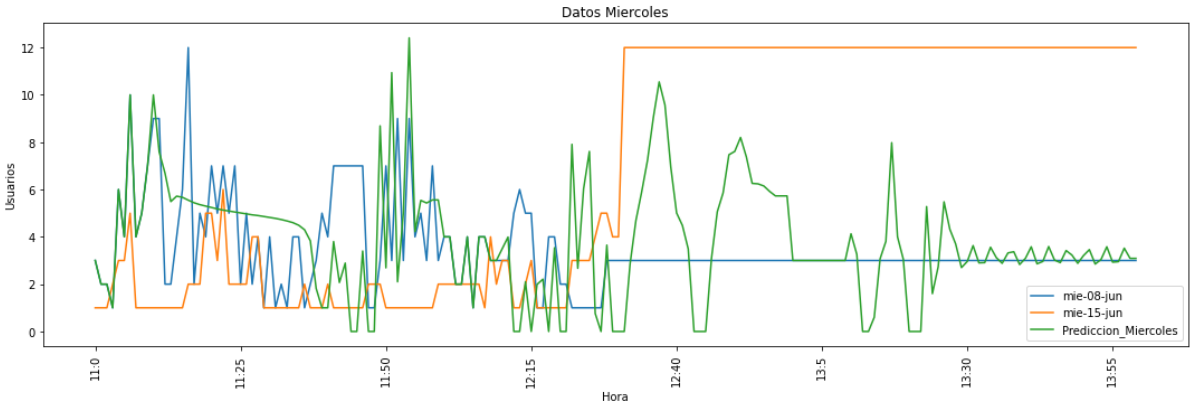


Ilustración E 2.3. Predicción día 3. (Fuente: Autor)

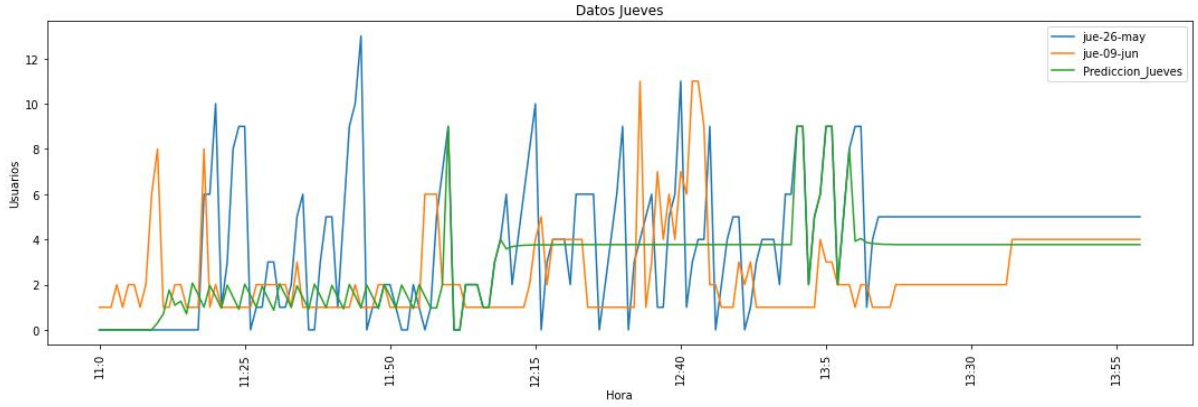


Ilustración E 2.4. Predicción día 4. (Fuente: Autor)

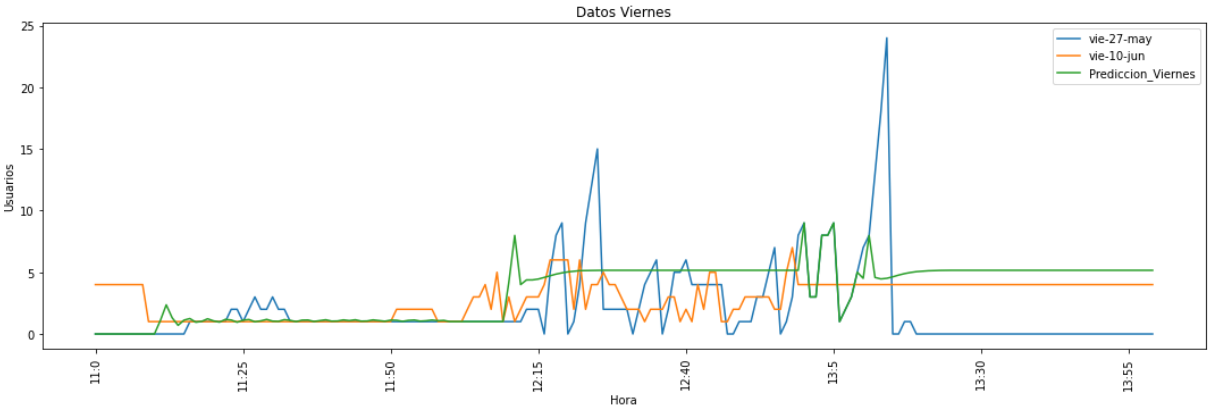


Ilustración E 2.5. Predicción día 5. (Fuente: Autor)