ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"SISTEMA INTEGRAL PARA LA RECONFIGURACIÓN DINÁMICA DE LOS TIEMPOS ENTRE PARADAS DE UNA RUTA PREDEFINIDA DE TRANSPORTE PÚBLICO"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO/A EN TELEMÁTICA

CRISTÓBAL JOSHUE DOMÍNGUEZ VELIZ

MARÍA FERNANDA MORA GARCÍA

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTOS

Gracias a Dios por las bendiciones que me ha dado desde el inicio hasta la culminación de mi carrera, a mis padres por el apoyo incondicional, motivación y enseñanzas que día a día me brindaron, a mi hermana por su cariño y comprensión, a mi compañera de proyecto por el buen desempeño y dedicación empleada para el desarrollo del proyecto, al Ing. Washington Velásquez por su ímpetu, paciencia y apoyo que mostró a lo largo de este semestre, y a todos mis amigos por su constante apoyo.

CRISTÓBAL JOSHUE DOMÍNGUEZ VÉLIZ

Agradezco primeramente a Dios por darme la salud y la sabiduría para seguir adelante, a mis padres y hermano quienes pese a la distancia han sido mi inspiración y mis pilares fundamentales en este largo camino, a mi compañero de proyecto que con esfuerzo logramos culminar con éxito este semestre, a mi profesor Ing. Washington Velásquez por el apoyo mostrado en todo momento durante esta etapa y a todos mis amigos quienes estuvieron prestos a brindarme su ayuda siempre.

MARÍA FERNANDA MORA GARCÍA

DEDICATORIA

La concepción de este proyecto está dedicada en especial a mis padres, Cristóbal y Rosa ya que con su ayuda y esfuerzo permitieron que hoy en día yo esté culminando una etapa más de mi vida estudiantil, son el mejor regalo que Dios me ha podido dar, mi ejemplo a seguir. También forman parte de este trabajo, así como de cada logro que he obtenido y de los que obtendré si Dios lo permite.

A mi hermana que ha estado conmigo en todo momento brindándome su apoyo y confianza.

A toda mi familia que siempre ha estado pendiente de mí, aconsejándome y motivándome a seguir adelante.

CRISTÓBAL JOSHUE DOMÍNGUEZ VÉLIZ

El presente proyecto lo dedico a mis padres Ramiro y Nuri quienes pese a la distancia confiaron en mí en todo momento y son la razón que me impulsan a seguir adelante y crecer profesionalmente. Porque con su esfuerzo, humildad y amor me han enseñado que cada día es una nueva oportunidad para superarme y ser mejor persona, y que los obstáculos que se han presentado a lo largo de mi vida los puedo vencer pues su apoyo ha sido y será incondicional.

A mi hermano Bryan quien a más de ser un hermano ha sido mi mejor amigo brindándome su alegría, humor, ayuda y apoyo.

A mi tía Carmen quien pese a la distancia ha estado siempre pendiente de mí, con consejos, dándome fortaleza y sobre todo su gran afecto.

A todos, muchas gracias, los quiero infinitamente, son lo mejor de mi vida y sin ustedes no lo hubiera logrado.

MARÍA FERNANDA MORA GARCÍA

TRIBUNAL DE EVALUACIÓN

Ing. Washington Velásquez, MSc.

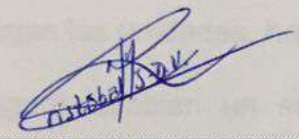
PROFESOR EVALUADOR

Ing. Wendy Yánez, MSc.

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Cristóbal Joshue Domínguez Véliz

María Fernanda Mora García

RESUMEN

El desconocimiento del tiempo exacto de llegada del siguiente vehículo a las estaciones intermedias causa malestar e inconformidad entre los habitantes que hacen uso del transporte público en la ciudad de Guayaquil. Se maneja un tiempo estimado de salida para las unidades el cual se extiende mucho más de lo esperado, debido al tráfico vehicular existente en las principales calles de la ciudad.

En este trabajo se desarrolló un prototipo que se encontrará ubicado en cada uno de los vehículos, el cual consta de una antena GPS y los módulos Arduino UNO y SIM808, los mismos que cada 15 segundos envían las coordenadas y velocidad con la que viajan las unidades, haciendo uso de la tecnología GSM.

Se configuró también un servidor web y de base de datos, para procesar la información obtenida por los módulos, con el objetivo de mostrar en cada una de las estaciones intermedias el tiempo que tarda en llegar el vehículo más cercano mediante una pantalla LCD.

Además, se elaboró una aplicación web con varias funcionalidades, una de las cuales se basa en el uso de Google Maps para conocer la ubicación de los buses o unidades que se encuentran circulando en las distintas rutas de transporte, así mismo se podrá interactuar con esta aplicación para localizar la parada más cercana a un punto origen y a un punto destino ingresado por el usuario.

Este proyecto busca ayudar a los usuarios a optimizar su tiempo al momento de hacer uso de un sistema de transporte público, y así mismo puedan conocer todas las rutas que este sistema brinda.

ÍNDICE GENERAL

INDICE DE FIGURAS

CAPÍTULO 1

1. GENERALIDADES

En este capítulo se detalla la problemática del uso de transporte público existente en la sociedad, así como la descripción de los mecanismos de solución y los alcances que se esperan obtener.

1.1 Antecedentes

Los medios de transporte son un extenso sistema socio-técnico, donde los servicios son provistos empleando una variedad de tecnologías y soluciones [1]. Analizando los diferentes avances tecnológicos que se han venido desarrollando a través del tiempo, se puede observar fácilmente las mejoras en los medios de transporte, y se espera que este progreso continúe en los siguientes años. Por una parte, se puede considerar al transporte como algo que va a ser utilizado por el usuario al igual que cualquier otro servicio, pero desde otro punto de vista, este cambio surgió debido a la necesidad dentro de una sociedad para transportarse de un lugar a otro; es decir, dejó de ser un simple servicio a la comunidad para convertirse en un punto primordial y de gran importancia [2].

En la actualidad se observa sobrepoblación en ciudades como Guayaquil, Quito, Cuenca, Loja y Manta, lo cual dificulta ofrecer servicios ágiles al alcance de todos. Uno de los servicios más afectados es el de transporte público, especialmente en horas pico, debido al gran número de vehículos circulando en las principales calles de la ciudad [3]. El malestar se muestra evidente en todos los usuarios de transporte público, siendo un inconveniente que requiere de planificación, discusión y proposición de soluciones por parte de los municipios locales. La municipalidad de Quito ha tomado una iniciativa con respecto a este problema implementando un sistema denominado “Pico y Placa”, el cual permite que se determine la circulación de vehículos en días y horarios específicos obteniendo una mejora en ese sentido, pero no lo suficiente, considerando que aún se constata el descontento entre las personas que hacen uso del transporte público [4].

En las grandes ciudades del mundo se desarrollan diferentes soluciones a este problema, como por ejemplo Nueva York, existe un sistema de transporte masivo denominado “MTA” (Metropolitan Transportation Authority). En donde el 85% se encuentran satisfechos con el uso de este medio de transporte dadas las ventajas que se ofrecen a lo largo de sus recorridos tanto en seguridad, confort, información y disponibilidad. [5]. Si se llegan a satisfacer estos aspectos en el transporte público de la ciudad de Guayaquil, los usuarios optarían por usar en mayor proporción este servicio, con lo cual el tráfico vehicular puede mejorar reduciendo la cantidad de vehículos que circulan por las calles, permitiendo que el sistema de Trasporte Público sea más eficiente a la hora de brindar sus servicios.

Se ha tomado como referencia el sistema de transporte público “Metrovía”, el cual inició como un proyecto elaborado por la Fundación Metrovía para ser utilizado en la ciudad de Guayaquil, donde el 83% de sus habitantes utiliza el servicio de transporte regular. En la actualidad, este sistema cuenta con 3 troncales que abarcan las principales rutas por donde existe mayor afluencia de personas. El nombre oficial es Sistema Integrado Transporte Masivo Urbano de Guayaquil, gestionado por la “Fundación Municipal Transporte Masivo Urbano de Guayaquil” [6], y está regulada por la Muy Ilustre Municipalidad de Guayaquil.

La puesta en marcha de este sistema de transporte se basó en el sistema de bus rápido “Transmilenio”, ubicado en la ciudad de Bogotá – Colombia [7]. El principal motivo para implementar este proyecto fue reducir el caos vehicular generado en la ciudad de Guayaquil y disminuir el robo en los buses urbanos [8].

Actualmente existen avenidas completas que están dedicadas para el uso exclusivo de los vehículos del sistema de transporte Metrovía [9], así como también carriles propios en las diferentes calles por las cuales circulan los demás automotores, de esta manera se tiene mayor eficiencia y optimización de tiempo con respecto al servicio de transporte urbano que se venía utilizando normalmente.

En el presente, el servicio de transporte público Metrovía ha reducido en gran manera el tráfico de las principales calles de Guayaquil, sin embargo, la gran

demanda de uso de este transporte ha traído consigo robos e inconformidad de los usuarios [10], debido a que la cantidad de personas transportadas supera la capacidad permitida en el vehículo. Este hecho trajo como consecuencia que en el año 2015 existiera una reducción de este servicio en comparación con el año 2014 [11].

Inicialmente para usar este medio de transporte se cancelaba el valor en efectivo, en la actualidad el sistema de pago se ha modernizado adaptándose a las nuevas tecnologías, como el dinero electrónico. Ahora cada usuario debe adquirir una tarjeta magnética, la cual se puede recargar en las principales troncales, así como también en las diferentes paradas de este servicio [12].

Adicionalmente, en la actualidad los usuarios cuentan con una aplicación móvil llamada “MetroGuía” que nos indica los diferentes recorridos que realiza la Metrovía dentro de la ciudad. Esta aplicación se encuentra disponible para teléfonos móviles con sistema operativo Android y fue creada por los ecuatorianos Eduardo Guzmán y Diego Pacheco el 11 de enero del año 2015 [13].

1.2 Descripción del problema

La ciudadanía debe esperar por un tiempo indeterminado las unidades de transporte público, sin poder realizar otras labores por el temor a que dicho vehículo pase en ese momento. Asimismo, las personas tienden a llegar tarde a su lugar de destino, debido al desconocimiento del intervalo en el que el siguiente articulado llega a la estación intermedia, tiempo en el cual pudieron optar por otros medios de transporte, como por ejemplo el servicio de taxi.

Este tipo de problemas ocurren también en todas las estaciones intermedias de la Metrovía, ya que las personas que esperan este medio de transporte desconocen el tiempo exacto de llegada del siguiente vehículo. Existe un tiempo estimado de salida de cada vehículo desde las principales troncales pero debido al denso tráfico existente en la ciudad de Guayaquil, accidentes de tránsito u otras circunstancias inesperadas que se suscitan en el diario vivir, este tiempo cambia y habitualmente se extiende mucho más de lo esperado, lo cual genera malestar e inconformidad entre los usuarios.

1.3 Justificación

En consideración al uso ineficiente del tiempo de los usuarios por la espera de los vehículos de las diferentes rutas de transporte público, se pretende diseñar y generar un prototipo que permita mostrar de manera dinámica los lapsos de llegada de cada medio de transporte en las paradas establecidas, tomando en cuenta las situaciones antes mencionadas, especialmente conociendo la problemática de tráfico vehicular existente en horas pico dentro de las diferentes ciudades.

Dado que el servicio de transporte urbano es el más afectado, provoca que dicho tráfico convierta a las principales ciudades en un caos, motivo por el cual la ciudadanía opta por utilizar los servicios brindados por los Sistemas de Transporte Masivo, ya que estos poseen un carril exclusivo dentro de cada ciudad, sin embargo, los problemas ocasionados por la gran congestión vehicular continúan.

Es por esta razón que este sistema tiene como principal finalidad mostrar a los usuarios en tiempo real y de manera dinámica los lapsos de llegada de cada unidad de transporte a través de cada una de las pantallas ubicadas en las paradas establecidas, de esta manera se pretende ayudar a la ciudadanía a llegar a tiempo a su destino, ya que al conocer los tiempos en el que este medio de transporte llegue a la parada esperada, los usuarios pueden optar por otras medidas dependiendo de sus necesidades.

1.4 Objetivos

A continuación, se detallan los objetivos planteados tanto el general como los específicos correspondientes a este sistema.

1.4.1 Objetivo General

Implementar un sistema de control dinámico de tiempo de paradas entre sub-estaciones utilizando módulos de comunicación de hardware y software libre que ayuden a la ciudadanía a la información del tiempo de espera en terminales de un sistema de transporte público.

1.4.2 Objetivos Específicos

- Identificar las rutas y paradas establecidas de un sistema de transporte público recopilando la información de la ciudad que generen un esquema sobre los puntos de espera de cada usuario.
- Estimar la distancia entre subestaciones con la ayuda de un sistema de geolocalización aplicando algoritmos necesarios para la obtención del tiempo que le toma a cada unidad llegar de una estación a otra.
- Construir un sistema de reconfiguración de tiempo programando un módulo de comunicación de hardware y software libre que permita calcular la velocidad del vehículo en cada trayecto de su ruta.
- Clasificar la información obtenida del geo localizador, tal como velocidad del bus y distancia entre sub-estaciones implementando una base de datos local que permita el almacenamiento de dichos datos, los cuales serán controlados por un administrador.
- Diseñar una aplicación web utilizando un lenguaje de programación que le permita al administrador observar las rutas y la ubicación del vehículo en tiempo real.
- Mostrar en pantalla el tiempo que tarda un articulado en llegar a la siguiente parada permitiéndole al usuario la optimización de su tiempo.

1.5 Alcance y Limitaciones

Se tiene como punto de acción el recorrido que realiza cada unidad del Sistema Integrado de Transporte Urbano Masivo “Metrovía” en la ciudad de Guayaquil, lo que comprende tanto norte, centro y sur para una población de aproximadamente 2’500.000 habitantes. Se considera abarcar una porción del sistema de Transporte Metrovía, especialmente la ruta alimentadora Alborada junto a todas sus paradas actualmente establecidas, un vehículo que representa a un articulado y los equipos incorporados al sistema de transporte.

Debido a que cada unidad de transporte debe tener conectividad durante todo su recorrido, se vuelve imperativo que exista una conexión ininterrumpida con el

servidor. Motivo por el cual se establece un diseño en el que se incluye el uso de la red GPRS, es decir que, en cada uno de los articulados exista un chip con plan de datos activo para hacer efectivo él envió de coordenadas y velocidad actual, y de esta manera llevar un registro de los datos y poder mostrar los tiempos que le toma a cada unidad en llegar a las paradas establecidas dentro de la ruta.

CAPÍTULO 2

2. MARCO TEÓRICO

En el contexto de este capítulo se menciona en detalle las diferentes tecnologías empleadas para la elaboración del sistema. Se describe toda la parte física que concierne al hardware, las herramientas o programas utilizados para la programación, esta sección se la conoce como software y se revisa la parte del desarrollo. Además, se definen las características esenciales y funciones que desempeñan los módulos utilizados, tal es el caso del módulo GPS y el módulo GSM integrados en una sola placa. Al tener conocimiento de cada sección, se busca dando lugar a una mejor comprensión de las distintas etapas que posee el sistema de reconfiguración dinámica de tiempos.

2.1 Tecnologías de Hardware

En esta sección se presentan los diferentes componentes físicos que se emplean para la elaboración del sistema.

2.1.1 Arduino UNO

Consiste en una plataforma electrónica abierta de software libre, flexible, de grandes recursos y muy fácil de usar. Toda la información que recolecta este dispositivo mostrado en la figura 2.1, lo hace a través de los pines de entrada en los cuales pueden conectarse una extensa variedad de sensores, llegando a tener control de la mayoría de cosas que nos rodean. El microcontrolador es programable, por lo que se hace uso de un lenguaje de programación (basado en Wiring [14]) y un entorno de desarrollo en arduino (basado en Processing [15]).

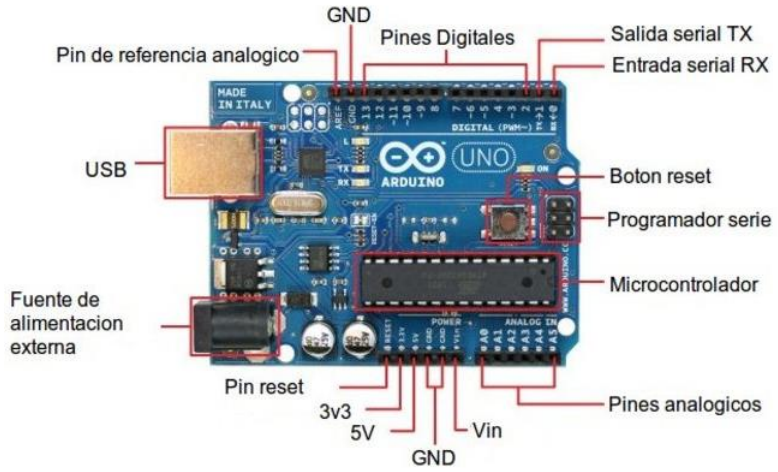


Figura 2.1 Placa electrónica Arduino UNO, Empleada para la interconexión de los dispositivos [16]

2.1.2 Módulo SIM808

Es un módulo completo de Quad-Band GSM/GPRS, el cuál combina la tecnología de navegación por GPS, el cual se puede apreciar en la figura 2.2. El diseño compacto que integra GPRS y GPS en un paquete SMT ahorra significativamente los costos y el tiempo para el desarrollo de aplicaciones. Las principales características son [17]:

- Quad-Band 850/900/1800/1900 MHz
- GPRS class 12: max 85.6 Kbps (carga/descarga)
- Sensibilidad GPS, Tracking: -165dBm; Cold starts: -147dBm
- Temperatura de operación: -40°C ~ 85°C
- Rango de tensión de alimentación: 3.4 ~ 4.4V

Figura 2.2 Modulo SIM808 utilizada para la obtención de coordenadas y envío de datos al servidor [18]

2.1.3 Pantalla LCD

También llamada pantalla de cristal líquido. Permite visualizar contenidos o información de forma gráfica mediante caracteres, símbolos o pequeños dibujos [19]. Este dispositivo es empleado para la visualización del tiempo en que llegará el siguiente bus a una parada establecida, el cual es mostrado en la figura 2.3.

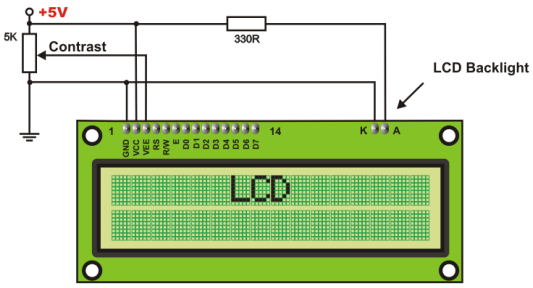


Figura 2.3 Pantalla LCD 2x16 donde se muestra el tiempo de llegada del bus a la parada [20]

2.1.4 Tarjeta SIM

Proviene de las siglas en inglés Subscriber Identify Module (Módulo de Identificación del Suscriptor). Es una tarjeta inteligente que se mostrada en la figura 2.4, con característica de ser desmontable, usada

exclusivamente en una red GSM. Almacena de forma segura la clave de servicio del suscriptor usada para identificarse ante la red. Empleada en teléfonos móviles, módems HSPA y módulos de la serie SIM. Una tarjeta SIM Claro se inserta en el módulo SIM808 y de esta manera se hace uso de la red móvil.

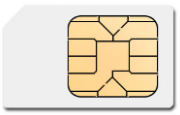


Figura 2.4 Tarjeta SIM que se introduce en el módulo SIM808 para utilizar la red GPRS local [21]

2.2 Tecnologías de Comunicación

En el siguiente apartado se detallan los tipos de comunicación utilizados en el sistema teniendo en cuenta la disponibilidad y compatibilidad de las redes existentes en el país.

2.2.1 GSM

Global System for Mobile Communications (GSM) es un sistema de telecomunicaciones digitales celulares, regularizado por el Instituto para la Normalización en Telecomunicaciones (ETSI). La estructura de su red proporciona enlaces de comunicación para mantener conectados a los usuarios [22]. Su arquitectura se muestra en la figura 2.5.

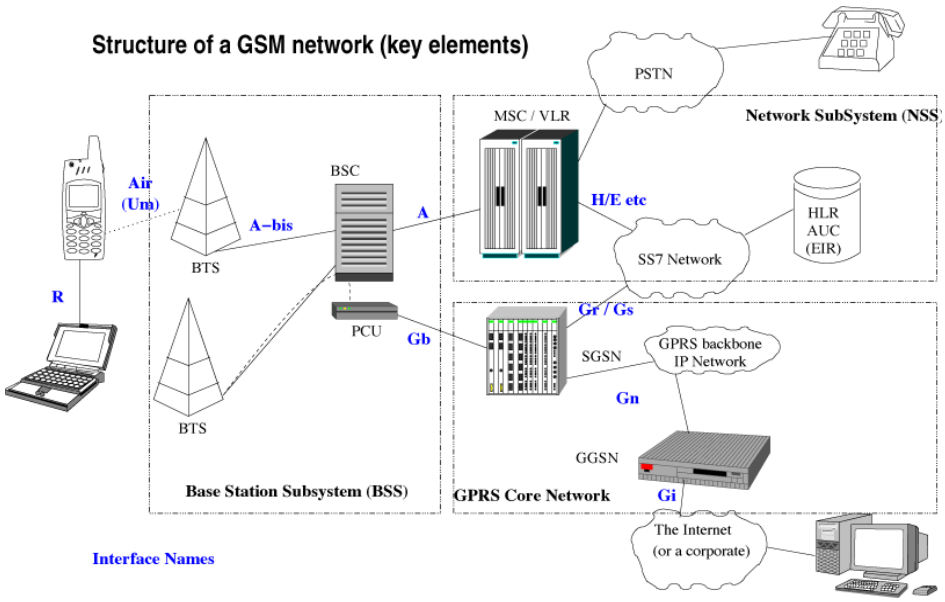


Figura 2.5 Arquitectura de una red GSM [23]

2.2.2 GPS

El sistema global de posicionamiento está compuesto por 24 satélites ubicados alrededor del planeta Tierra, a través de una configuración en triángulo son los encargados de enviar la respectiva información de la ubicación actual desde donde se realiza la consulta. Esta red permite a varias aplicaciones utilizar sus servicios siempre y cuando se necesite precisión en geolocalización, tal es el caso de ubicación de vehículos, corrientes marinas, vigilancia de placas tectónicas o exploración geofísica [24]. Existen varios datos que se obtienen al utilizar esta tecnología, entre los más comunes se tienen las coordenadas expresadas en latitud y longitud. En la figura 2.6 se muestra un bosquejo de esta red.

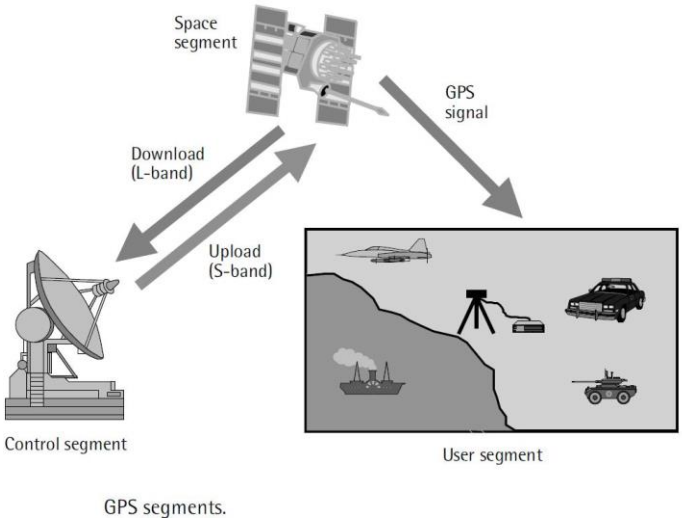


Figura 2.6 Red GPS utilizada para la obtención de las coordenadas y velocidad del móvil [25]

2.3 Tecnologías de Desarrollo

2.3.1 HyperText Markup Language (HTML)

Lenguaje de marcado con mayor predominio para la elaboración de páginas web. Mediante este lenguaje se describe la estructura y el contenido en forma de texto, y permite además complementar el texto con objetos [28].

2.3.2 Cascading Style Sheets (CSS)

Lenguaje que trabaja en conjunto con HTML, este nos permite tener el control del diseño y la estética a nuestro sitio web [29]. Es utilizado para darle un estilo personalizado a los componentes que forman parte de cada uno de los templates o modelos de pantalla.

2.3.3 Javascript

Es un lenguaje de desarrollo de aplicaciones cliente/servidor a través de Internet. Está insertado dentro del lenguaje HTML, proporcionando al usuario la interactividad con las páginas web de forma dinámica [30].

2.3.4 Bootstrap

Framework que aporta al desarrollo de interfaces web en conjunto con CSS y Javascript. Su principal función es la de acoplar la interfaz de la aplicación web al tamaño de diferentes pantallas, sean estas, celulares, tablets o pcs. Esta técnica también se la conoce como diseño adaptativo [31].

2.3.5 Highcharts

Es una librería de gráficos escrita en Javascript, que se utiliza para añadir imágenes interactivas a aplicaciones web. La cual tiene como propiedad cargar los datos a utilizarse desde un servidor externo. Para usarla basta tener el archivo highcharts.js o el archivo jquery.js [32].

2.3.6 API Google

Muchas aplicaciones y millones de sitios web utilizan la API de Google Maps para ofrecer a sus usuarios diferentes servicios relacionados con localización, tales como mapas, imágenes e indicaciones 100% personalizados [33].

2.3.7 MySQL

Es un medio que permite almacenar datos, los cuales se organizan en tablas. Básicamente es un sistema utilizado para gestionar bases de datos relacionales [34]. A través de este medio se crea la base de datos del sistema, el cual contiene toda la información relacionada a las diferentes paradas, vehículos y ubicación de los mismos.

2.3.8 Python

Es un lenguaje de scripting multiplataforma y orientado a objetos, durante los últimos años ha sido muy utilizado gracias a la cantidad de librerías que posee, así como tipos de datos y funciones incorporadas. Éste lenguaje es utilizado para realizar cualquier tipo de programa desde pequeñas aplicaciones a páginas web [35].

2.3.9 Django

Es un framework de programación escrito en Python de alto nivel que permite trabajar en el lado del servidor a la hora de crear páginas web. Se programa más rápido y de una manera más organizada, es muy flexible gracias a que no todo está entrelazado [36]. Es aquí en donde se ejecuta la aplicación web, definiendo las distintas URL's (de sus siglas en inglés Uniform Resource Locator, Localizador de Recursos Uniforme) correspondientes a cada una de las pantallas que se muestran en el navegador.

CAPÍTULO 3

3. IMPLEMENTACIÓN Y DESARROLLO

En este capítulo se describe una solución viable para la implementación del dispositivo encargado de obtener las coordenadas y velocidad de un vehículo en tiempo real. Además, se desarrolla una aplicación web que está conformada por dos interfaces, la primera dedicada al servicio de los usuarios y la segunda corresponde al administrador. Así también, se detallan las líneas de código de los procesos más importantes que han sido implementados en la plataforma.

3.1 Diagrama general

El origen del funcionamiento del sistema radica en un módulo ubicado en cada uno de los vehículos el cual tiene como principal objetivo obtener su velocidad y ubicación actual. Información que es enviada al servidor haciendo uso de la red GPRS y procesada por medio de una codificación realizada en *Python* para ser almacenada en una base de datos. Estos datos obtenidos son utilizados inicialmente para visualizar el vehículo en un mapa específico, el cual se muestra en la página web *Metrotime* y a su vez para realizar varios procesos que determinan el tiempo que tarda un bus en llegar a cada una de las paradas establecidas dentro de su ruta. Cabe destacar que para poder visualizar un vehículo en el mapa primero debe ser agregado por el administrador y su estado debe ser activo (los detalles de cada tabla son descritos posteriormente). El tiempo de llegada obtenido es enviado a un módulo ubicado en cada parada y mostrado al usuario a través de una pantalla LCD. Información que es presentada en la figura 3.1.

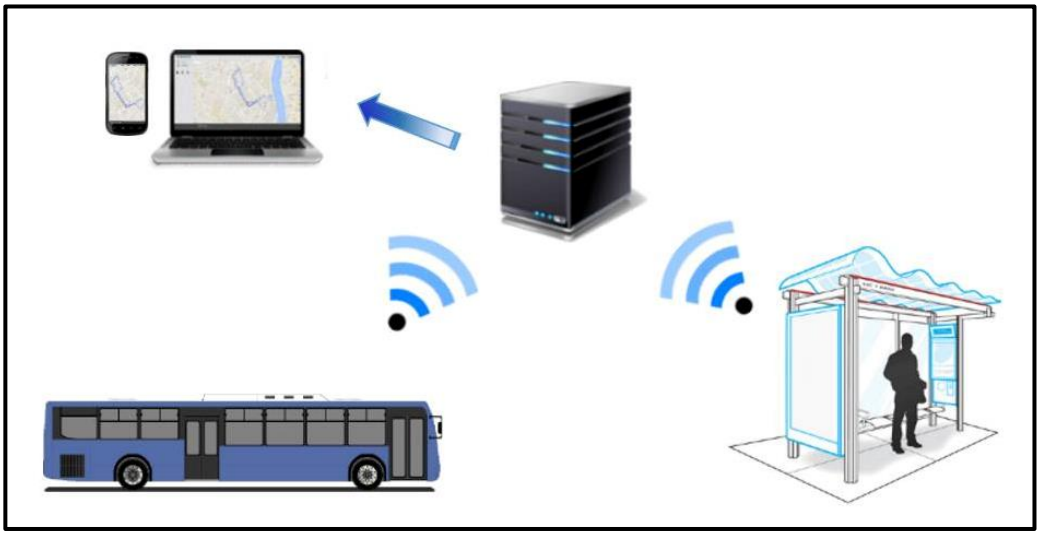


Figura 3.1 Diagrama general del sistema

3.2 Diagrama del módulo en el vehículo

El dispositivo consta de un microcontrolador *Arduino*, un módulo *SIM808* y una fuente de voltaje de 9V, en la Figura 3.2 se muestra cómo se debe realizar la conexión entre estos aparatos electrónicos. El microcontrolador es el núcleo de este dispositivo, mediante su programación es quien le dice a la SIM808 las acciones que se deben realizar. Para el óptimo funcionamiento en conjunto de ambos circuitos deben ser energizados con una fuente de 9V, caso contrario se producen errores durante el envío de información.

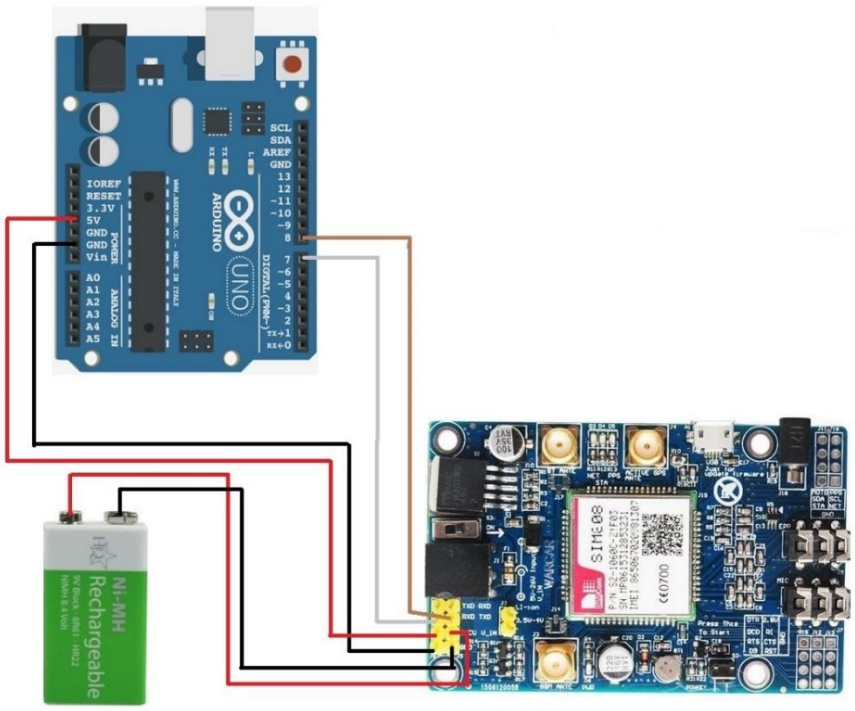


Figura 3.2 Diagrama de conexión del módulo en vehículo

3.3 Trama del módulo GPS

Esta es la trama que se recibe del módulo GPS cada vez que se ejecuta el comando *AT+CGNSINF,* donde muestra una completa información sobre el punto que se encuentra ubicado en ese instante, por ejemplo: hora, latitud, longitud, velocidad con respecto a la tierra, entre otros. El detalle de la trama se muestra en la figura 3.3.

Index	Parameter	Unit	Range	Length
1	GPS run status	--	0-1	1
2	Fix status	--	0-1	1
3	UTC date & Time	yyyyMMddhh mmss.sss	yyyy: [1980,2039] MM : [1,12] dd: [1,31] hh: [0,23] mm: [0,59] ss.sss:[0.000,60.999]	18
4	Latitude	±dd.dddddd	[-90.000000,90.000000]	10
5	Longitude	±ddd.dddddd	[-180.000000,180.000000]	11
6	MSL Altitude	meters		8
7	Speed Over Ground	Km/hour	[0,999.99]	6

8	Course Over Ground	degrees	[0,360.00]	6
9	Fix Mode	--	0,1,2[1]	1
10	Reserved1			0
11	HDOP	--	[0,99.9]	4
12	PDOP	--	[0,99.9]	4
13	VDOP	--	[0,99.9]	4
14	Reserved2			0
15	GPS Satellites in View	--	[0,99]	2
16	GNSS Satellites Used	--	[0,99]	2
17	GLONASS Satellites in View	--	[0,99]	2
18	Reserved3			0
19	C/N0 max	dBHz	[0,55]	2
20	HPA[2]	meters	[0,9999.9]	6
21	VPA[2]	meters	[0,9999.9]	6
			Total: (94) chars	

Figura 3.3 Lista de los parámetros que posee una trama del módulo GPS

De aquella trama se toma los valores de Latitude (latitud), Longitude (longitud) y Speed Over Ground (velocidad sobre la tierra) que son los necesarios para ubicar y monitorear cada uno de los vehículos. En la figura 3.4 se muestra la trama completa en el monitor serie.

```
AT+CSQ
+CSQ: 18,0

OK
AT+CGNSPWR=1
OK
AT+CGNSSEQ=RMC
OK
AT+CGNSINF
+CGNSINF: 1,1,20160804040839.000,-2.134673,-79.899395,10.100,0.07,232.3,1,,0.9,1.3,0.9,,8,7,,,38,,
```

Figura 3.4 Trama completa obtenida del módulo GPS

3.4 Adquisición de datos

Para establecer la comunicación entre el módulo y el servidor, se utiliza la función *SubmitHttpRequest( )*. Este método hace uso de la red GPRS para realizar la conexión a Internet y enviar información a la base de datos, haciendo uso del módulo SIM808 junto con un chip insertado.

El procedimiento que se utiliza para la obtención de los datos se rige a los siguientes pasos:

1. Cuando se enciende el módulo colocado en el dispositivo es necesario activar el GPS, para tal efecto se ejecuta internamente el comando *AT+CGNSPWR=1.* Desde ese momento resulta imprescindible esperar un lapso de 45 segundos mientras se cargan las funciones del GPS.
2. Se ejecuta el comando *AT+CGNSINF* el cual tiene como función retornar toda la información obtenida con el GPS. En aquella información está incluida la latitud, longitud y velocidad con respecto a la tierra.
3. La información obtenida es almacenada en tres distintas variables contenidas en el microprocesador Arduino, para luego poder ser utilizadas.
4. Se inicializa la conexión a la red 3G para hacer uso de los datos móviles y conectarse a Internet. Este proceso se lo realiza con el comando *AT+SAPBR=3,1,\"APN\",\"internet.claro.com.ec*\ en donde se debe indicar la dirección APN del operador correspondiente al chip que se tenga insertado en el módulo SIM808. En este caso se hace uso de la red *Claro*.
5. Se inicializa una solicitud HTTP con el comando *AT+HTTPINIT*.
6. Se ingresan los valores de los parámetros a enviar en la solicitud HTTP con la ayuda del siguiente comando *"AT+HTTPPARA=\"URL\",\http://mfmora.pythonanywhere.com/transporte/conection/?id_vehiculo=GOB0048&latitud=%s&longitud=%s&velocidad=%s*". De donde, %s son las variables de los datos requeridos.
7. Para comprobar que los pasos anteriores se han ejecutado satisfactoriamente utilizamos el comando *AT+HTTPREAD*. Esta acción retorna la respuesta de la dirección URL ingresada en el paso anterior, que en este caso debe retornar “True”.

8. Se repite automáticamente la ejecución de los pasos a partir del paso 2.

Estos pasos se ejecutan automáticamente cada 15 segundos mientras el vehículo se encuentra en circulación.

A continuación se detalla el código de la función GPS( ), siendo el primero en ejecutarse debido a que en esta parte se obtiene información sobre la posición y velocidad. Se tiene un lazo while para repetir la acción hasta que se apague el módulo manualmente.

El inicio del bucle consiste en recorrer la respuesta de la lectura GPS, dada por el comando `AT+CGNSINF` que retorna los parámetros de la trama citada previamente; a cada uno de ellos le corresponde una ubicación fija en el arreglo recibido, por tal motivo se extraen los datos requeridos a través de la posición en la que se encuentran. En el caso de la latitud, se encuentra entre las posiciones 45 y 55, entonces cuando el puntero se ubique entre esos índices va a agregar cada caracter en un *String* llamado *latitu*. La longitud se encuentra entre las posiciones 55 y 66, y para la velocidad le corresponde del 73 al 78.

Los valores deben ser almacenados en un arreglo de carecteres para poder agregar estas variables a la *URL* de conexión con el servidor, razón por la que se utiliza la conversión de *String* a *Char Array* con cada valor obtenido `latitu.toCharArray(latitud,10);`. A partir de la tercera iteración se hace uso del método *SubmitHttpRequest( )* para realizar el envío de los datos al servidor, método que se muestra en la figura 3.5.

```
void GPS(){
   mySerial.println("AT+CGNSPWR=1");
   ShowSerialData();
   delay(2000);
   mySerial.println("AT+CGNSSEQ=RMC");
   ShowSerialData();
   delay(45000);
  while(1){
    mySerial.println("AT+CGNSINF");
    ShowSerialData();
    String response="";
    long int time=millis();
    int leer=0;
    while((time+1000)>millis()){
      while(mySerial.available()){
        char c= mySerial.read();
        response+=c;
        if(leer>45 && leer<55){
          latitu+=c;}
        if(leer>55 && leer<66){
          longitu+=c;}
        if(leer>73 && leer<78){
          velocida+=c;}
        leer++;
      }
    }
    Serial.println(response);
    if(velocida[3]==','){
      velocida[3]=' ';}
    obtenerLatitud();
    obtenerLongitud();
    obtenerVelocidad();
    latitu="";
    longitu="";
    velocida="";
    cont++;
    if(cont>3){
      SubmitHttpRequest();}
    else{
      delay(15000);}
  }
}
```

Figura 3.5 Codificación de la función GPS utilizada para obtener latitud, longitud y velocidad.

Se inicia con la comprobación de la potencia en la señal recibida (señal de operadora móvil local), para lo cual se utiliza el comando `"AT+CSQ"`. Dado que las rutas se encuentran dentro de una ciudad que cuenta con buena cobertura, el envío de datos se efectúa satisfactoriamente. A excepción de los túneles en los cuales no se recibe señal de la operadora, pero al salir de ellos la conexión se reestablece inmediatamente. Acto seguido se comprueba si se tiene disponible un plan de datos `"AT+CGATT?"`, caso contrario no se puede realizar el envío de información.

Luego se indica la información APN (Punto de acceso de la red) del operador móvil local que se está haciendo uso `"AT+SAPBR=3,1,\"APN\",\"internet.claro.com.ec\""`. Para saber el APN que se debe ingresar se lo solicita a la operadora móvil o se revisa en las configuraciones de internet de un teléfono móvil que utilice dicha operadora.

Se inicia el servicio HTTP con el comando `"AT+HTTPINIT"`, servicio que posteriormente será utilizado para realizar solicitudes GET. A continuación, se añade la URL junto con los datos obtenidos de latitud, longitud y velocidad. Este proceso hace que los valores sean almacenados en la base de datos. Codificación mostrada en la figura 3.6.

```
void SubmitHttpRequest()
{
  mySerial.println("AT+CSQ");
  delay(100);
  ShowSerialData();
  mySerial.println("AT+CGATT?");
  delay(100);
  ShowSerialData();
  mySerial.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
  delay(1000);
  ShowSerialData();
  mySerial.println("AT+SAPBR=3,1,\"APN\",\"internet.claro.com.ec\"");
  delay(2000);
  ShowSerialData();
  mySerial.println("AT+SAPBR=1,1");
  delay(1000);
  ShowSerialData();
  mySerial.println("AT+HTTPINIT");
  delay(1000);
  ShowSerialData();
  mySerial.print("");
  sprintf(frame,"AT+HTTPPARA=\"URL\",\"http://mfmora.pythonanywhere.com/transporte/conection/
        ?id_vehiculo=GOB0048&latitud=%s&longitud=%s&velocidad=%s",latitud, longitud,velocidad);
  mySerial.print(frame);
  mySerial.println("\"");
  delay(1000);
  ShowSerialData();
  mySerial.println("AT+HTTPACTION=0");
  delay(5000);
  ShowSerialData();
  mySerial.println("AT+HTTPREAD");
  delay(300);
  ShowSerialData();
  mySerial.println("");
  delay(100);
}
```

Figura 3.6 Codificación de la función SubmitHttpRequest

A través de un monitor serie se puede observar el paso a paso de las acciones que realiza el dispositivo para establecer la comunicación con el servidor. A continuación, en la figura 3.7 se muestran los comandos incluidos en la programación.

```
AT+CSQ
+CSQ: 19,0

OK
AT+CGATT?
+CGATT: 1

OK
AT+SAPBR=3,1,"CONTYPE","GPRS"
OK
AT+SAPBR=3,1,"APN","internet.claro.com.ec"
OK
AT+SAPBR=1,1
ERROR
AT+HTTPINIT
ERROR
AT+HTTPPARA="URL","http://mfmora.pythonanywhere.com/transporte/AT+HTTPACTION=0
OK

+HTTPACTION: 0,200,4
AT+HTTPREAD
+HTTPREAD: 4
True
```

Figura 3.7 Resultados del envío de datos mostrados en un monitor serie

Para reconocer que el método ha logrado enviar la información al servidor, al final de la operación se muestra “True”, valor que retorna la página web cuando se ha terminado con el proceso de almacenar datos. En la figura 3.8 se muestra el producto final a colocarse en el vehículo.

Figura 3.8 Módulo situado en cada uno de los vehículos

3.5 Modelo relacional de Base de datos

El gestor de base de datos utilizado para este sistema es MySQL. En la figura 3.9 se muestra el esquema relacional de las tablas creadas en la base de datos, las cuales son utilizadas para el almacenamiento distribuido y ordenado de la información necesaria para el correcto funcionamiento del sistema.

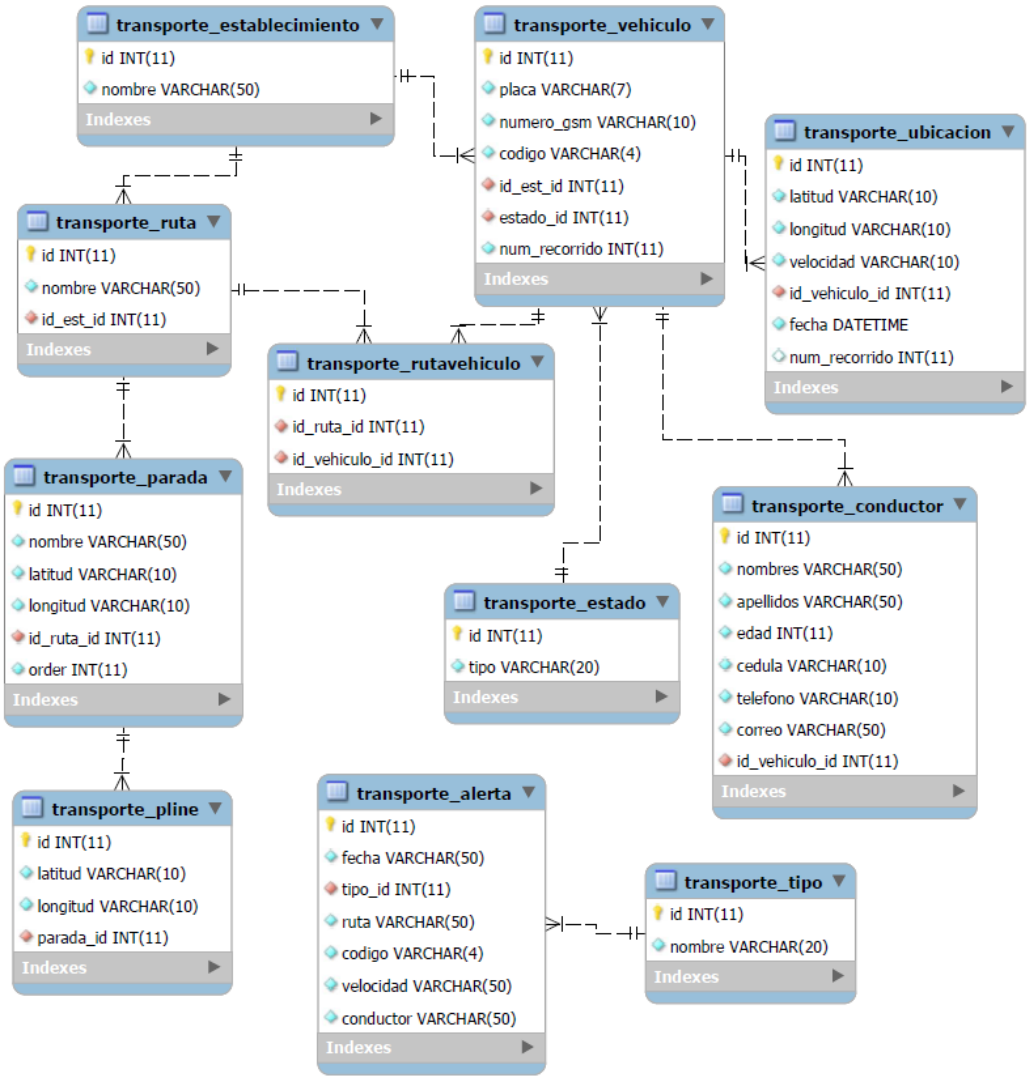


Figura 3.9 Modelo Relacional de la base de datos utilizada

A continuación, se describen cada una de las tablas que conforman la base de datos:

La tabla *transporte_establecimiento* contiene la entidad que solicita este servicio.

La tabla *transporte_vehiculo* posee los vehículos registrados junto con su respectiva información y la asignación a un establecimiento específico.

La tabla *transporte_estado* contiene los posibles estados que se le puede asignar a un vehículo, tales como activo (vehículo en circulación), inactivo (vehículo fuera de servicio) o mantenimiento (vehículo imposibilitado de operar debido a fallas mecánicas).

La tabla *transporte_ubicacion* posee la información de ubicación referente a un vehículo existente.

La tabla *transporte_conductor* contiene la respectiva información de cada conductor, así como el vehículo que se le tiene asignado.

La tabla *transporte_ruta* posee información básica con respecto a una ruta perteneciente a un establecimiento previamente definido.

La tabla *transporte_rutavehiculo* es de transición con el fin de asignar una ruta a un vehículo.

La tabla *transporte_parada* contiene información como la ubicación y nombre de cada una de las paradas que se encuentran dentro de una ruta dada.

La tabla *transporte_pline* contiene cada uno de los puntos de la trayectoria entre las paradas de una ruta.

La tabla *transporte_alerta* posee la información de las alertas producidas por las variaciones de las velocidades de los vehículos.

La tabla *transporte_tipo* posee los posibles tipos de alertas que pueden existir, tales como exceso de velocidad y vehículo detenido.

3.6 Diagrama de Aplicación web

Para una adecuada organización y optimización de recursos los datos a utilizarse se encuentran almacenados en diferentes listas conocidas como diccionarios, los cuales se crean como una vista y son llamados a las funciones a través de URLs,

mismas que varían dependiendo de una clave. Para este sistema las claves vienen dadas por el número de ruta o vehículo específico.

Este sistema consta de dos interfaces, la primera es la interfaz de administración en donde el administrador tiene acceso a todas las rutas con sus vehículos asignados, además de la información propia de cada uno de ellos, su ubicación actualizada y sus respectivos conductores. La segunda parte es la que se encuentra disponible para los usuarios en general, en donde las personas podrán interactuar con los beneficios que brinda la aplicación, desde diferentes dispositivos que posean conexión a internet, pues gracias al uso de Bootstrap esta interfaz se adapta a cualquier tamaño de pantalla.

3.6.1 Interfaz de Administración

Para que el sistema sea exitoso, el rol que desempeña el administrador es vital, pues es quien habilita y deshabilita la información de los vehículos mostrados en el mapa, así como los tiempos que tarda en llegar a cada una de las paradas establecidas sean los correctos. Dentro de las acciones que puede realizar el administrador, se encuentran las de agregar vehículo con su respectivo conductor. Es importante destacar que un bus o articulado puede tener asignado más de un conductor.

La página web no permite datos erróneos en sus formularios. Existen métodos que validan los campos principales, tales como placa, número GSM, código cuando se trata del ingreso de un nuevo vehículo. Así mismo el número de cédula en el caso de un conductor, ejemplo que se muestra en la figura 3.10.

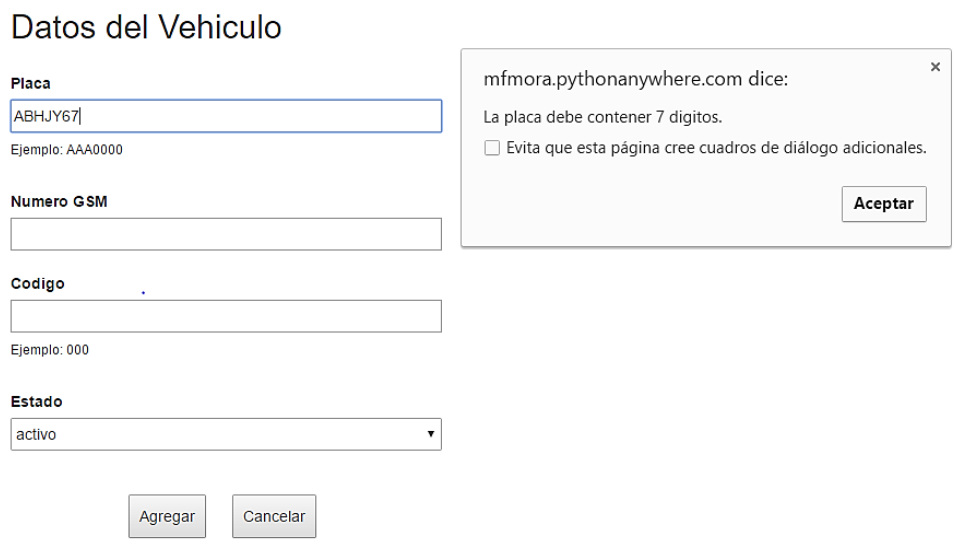


Figura 3.10 Alerta mostrada al momento de ingresar un dato erróneo en un formulario

Para el ejemplo anterior se hace uso de la función validar placa, la cual verifica que exista un total de 3 letras seguidas de 4 dígitos, valor que es declarado en una variable, `var exprr=/^[A-Z]{3}\d{4}$/;` Una vez obtenido el valor ingresado en el bloque de texto `valor_placa=document.getElementById("placa").value;` la función compara el dato obtenido, mismo que no debe estar vacío o tener más de siete dígitos y ser diferente a la variable definida, en este caso, la variable exprr, todo esto con la ayuda de un condicional: `if (valor_placa== null || valor_placa.length != 7 || valor_placa==" " || !exprr.test(valor_placa))`, si este if se cumple, se indica el error mediante una alerta `alert('La placa debe contener 7 dígitos.');` se borra el dato ingresado y se devuelve un valor booleano `false`, informando que no es el correcto, caso contrario la función devuelve un valor `true`, indicando que el dato es correcto. Procesos similares a este, se realizan en la validación de otros campos.

Para que el usuario pueda visualizar el vehículo en el mapa correspondiente y observar los tiempos que demora en llegar a cada una de las paradas, su estado debe cambiarse a activo una vez que inicie su recorrido y el número de recorrido debe aumentar en una unidad, resultando necesaria la intervención del administrador en la tabla mostrada en la figura 3.11. Así también, cuando el vehículo finalice su recorrido su estado debe ser cambiado a inactivo.

Codigo	Placa	Numero GSM	Estado	Recorridos	Accion
001	PCC4364	0991480648	activo	1	Editar
100	GOB0048	0994681761	activo	2	Editar
101	MNG3479	0933586214	inactivo	1	Editar
102	ABC5599	0988411226	activo	1	Editar

Figura 3.11 Tabla de información del estado de los vehículos vista por el administrador

Para tener un mayor control de los vehículos, el administrador puede observar la relación Velocidad vs Hora de un vehículo específico con estado activo. Esto con la ayuda de HightCharts, a la cual se le establece como parámetro “Y” la velocidad del vehículo seleccionado y como parámetro “X” el tiempo actual. Esta gráfica es dinámica y se la puede observar en la figura 3.12, los parámetros se actualizan cada 8 segundos y con la ayuda de Ajax se obtiene el valor de la velocidad del vehículo guardado en el diccionario *listavel* el cual lo obtenemos haciendo un POST a la URL `url:"/transporte/listavel/"+selected+"/"`, enviándole como clave el número de vehículo seleccionado.

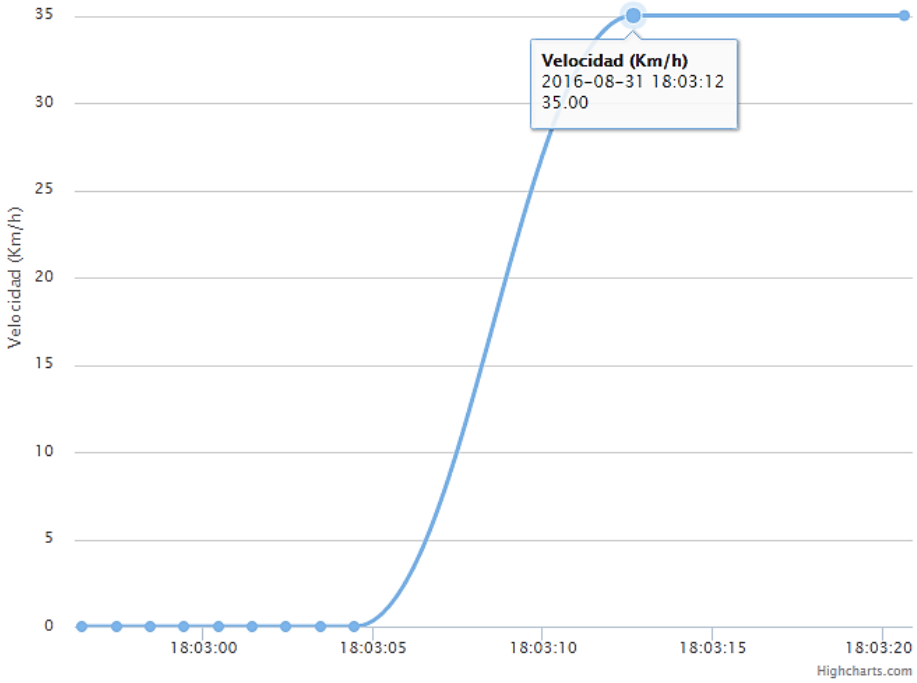


Figura 3.12 Gráfica Velocidad vs Hora de un vehículo seleccionado

Dentro de este entorno, el administrador podrá observar en la parte inferior derecha de la pantalla las notificaciones que se generan cuando se crea una alerta. Cada que un carro tenga una velocidad menor a 0.5km/h se considera como detenido, si es superior a 50km/h exceso de velocidad respectivamente. Dicha notificación muestra además el número de vehículo al cual pertenece la alerta lo cual se puede observar en la figura 3.13.

Figura 3.13 Notificaciones mostradas al administrador

3.6.2 Interfaz de Usuario

El usuario puede hacer uso de varias actividades dentro de la aplicación web, tales como consultar información de rutas, interactuar con los vehículos y verificar las sugerencias propuestas a la hora de hacer uso de este medio de transporte.

- **Información de Rutas**

  El usuario puede informarse con respecto al recorrido de cada una de las rutas establecidas. Observando las calles por donde se realizan los diferentes recorridos.

- **Selección de Rutas**

  Cuando un usuario selecciona una de las rutas establecidas en la página web se muestra un mapa con la ruta trazada y todas sus paradas establecidas. Después de 15 segundos (tiempo en que demora el dispositivo en enviar las coordenadas) el usuario podrá visualizar los carros que se encuentran circulando dentro de esa ruta siempre y cuando haya sido activado por el administrador, este se muestra en el mapa y se va actualizando su posición cada 15 segundos simulando su movimiento en tiempo real. Este algoritmo se muestra en la figura 3.14.

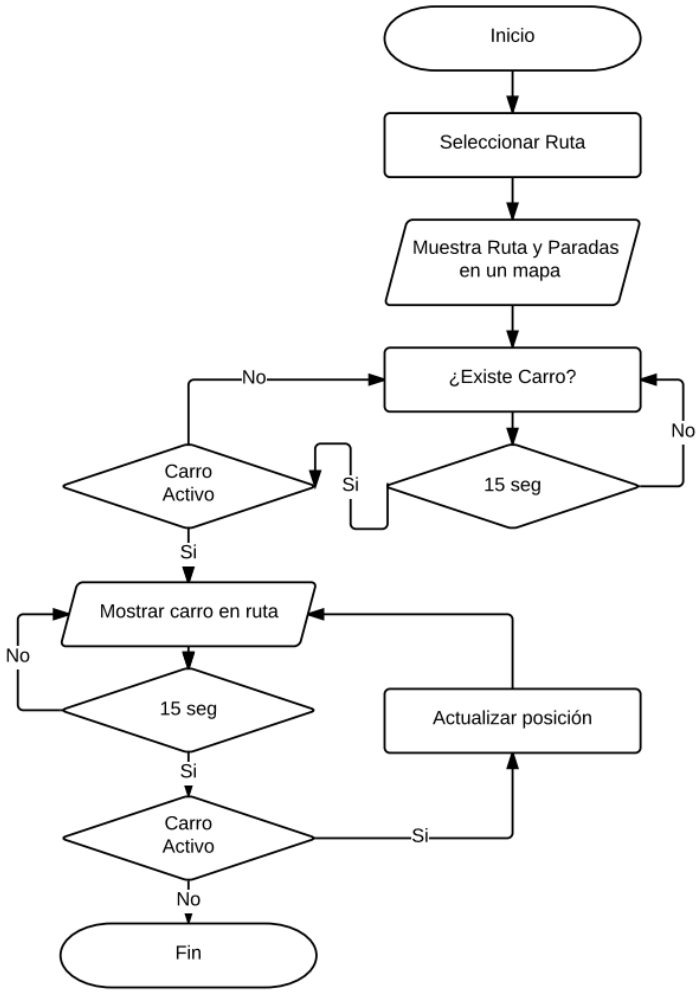


Figura 3.14 Algoritmo para visualizar el contenido de una ruta seleccionada

Este proceso se lo realiza en un javascript que recibe dos diferentes diccionarios para realizar todas las peticiones. Cada ruta contiene un número de identificación o id, el cual al momento de ser seleccionado por el usuario se envía internamente como parámetro principal en conjunto de otros datos necesarios. El mapa mostrado al usuario es proveniente de Google maps, el cual se centra de acuerdo a los otros parámetros enviados.

Inicialmente el script lee el primer diccionario mediante la url: `url: "/transporte/listapar/"+a+"/"`, donde la variable `a` corresponde al id de la ruta seleccionada. Este primero diccionario contiene 2

listas, en las cuales una de ellas muestra información referente a cada parada y la otra muestra las coordenadas de los puntos en donde se debe trazar la ruta entre paradas. La función datos, es la encargada de separar a estas listas para poder leer su contenido.

Una vez obtenida la lista de paradas, en base a las coordenadas leídas se crea un marcador de google por cada valor obtenido, identificado por un ícono personalizado que se muestran en el mapa creado anteriormente, función que se muestra en la figura 3.15.

```
marker = new google.maps.Marker({
    position: new google.maps.LatLng(paradas[j]['latitud'], paradas[j]['longitud']),
    map: map,
    icon: src = "/static/images/parada1.png",
    zIndex:0
});
```

Figura 3.15 Función que crea un marcador por cada coordenada de parada recibida

Todos los marcadores tienen la propiedad de mostrar el número de parada e información de las calles, mediante un evento que se ejecuta cada que el usuario los presiona, como lo indica la figura 3.16.

Figura 3.16 Marcador representativo a una parada que muestra su información y parte de línea trazada correspondiente a la ruta

Para trazar la ruta se hace uso de una funcionalidad del api de google denominado Polyline, el cual recibe como parámetros dos puntos y traza una recta entre ellos, la cual es mostrada en el mismo mapa inicial. Por lo que para la obtención de los puntos se

recorre la lista de tal manera que se toman los datos de un punto y del punto siguiente, codificación que se muestra en la figura 3.17.

```
for (var i = 0; i < puntos.length - 1; i++) {
    var l1 = new google.maps.LatLng(puntos[i]['r_list']['rlat'], puntos[i]['r_list']['rlong']);
    var l2 = new google.maps.LatLng(puntos[i + 1]['r_list']['rlat'], puntos[i + 1]['r_list']['rlong']);
    var miRuta = [l1, l2];
    var trazo = new google.maps.Polyline({
        path: miRuta,
        strokeColor: "#0000FF",
        strokeOpacity: 0.8,
        strokeWeight: 3
    });
    trazo.setMap(map);
}
```

Figura 3.17 Código que realiza el trazo entre dos posiciones dentro de un mapa

El segundo diccionario muestra las últimas posiciones de todos los vehículos activos dentro de la ruta para lo cual, se hace uso de la url: `url:"/transporte/lcar/"+global+"/"`, donde la variable global es la variable a utilizada en la función anterior. En este caso la función que hace uso de este diccionario no puede tener variables enviadas como parámetro dentro de la misma función, ya que cada 15 segundos se llama nuevamente a la función con la finalidad de obtener las posiciones actualizadas haciendo uso de Ajax al diccionario. Esto se realiza con la función: `setInterval(posicion_car,15000);` la cual recibe como parámetros el nombre de la función a actualizarse junto con su respectivo intervalo de tiempo.

La función *posición_car* recorre la lista correspondiente a los vehículos, creando marcadores en base a las coordenadas obtenidas, donde el título corresponde al número de vehículo, guardándolos en una lista llamada *markers.* Al igual que las paradas, estos marcadores tienen un ícono personalizado que los diferencia.

Una vez finalizado el recorrido de la lista, se hace uso de la función `deleteMarkers();` la cual elimina los marcadores de los vehículos mostrados en el mapa, para luego utilizar el método `showMarkers(GlobalMap);`, el cual recibe como parámetro al mapa

principal, con la finalidad de mostrar los nuevos marcadores creados.

Este proceso de eliminar y crear marcadores se lo hace con la finalidad de que la página simule el movimiento de los vehículos dentro de la ruta y si uno de ellos cambia su estado a inactivo ya sea porque terminó su recorrido o por algún daño inesperado, este no deberá mostrarse en el mapa.

Dentro de esta misma interfaz, el usuario tiene la capacidad de poder interactuar con los vehículos en movimiento, pues, cada que se presione cualquiera de ellos, se mostrarán de otro color las paradas que faltan por recorrer, lo cual se muestra en la figura 3.18. Además, el usuario podrá presionar cada una de ellas para conocer el tiempo en el que ese vehículo tardará en llegar a la parada seleccionada. Este algoritmo se lo muestra en la figura 3.19.

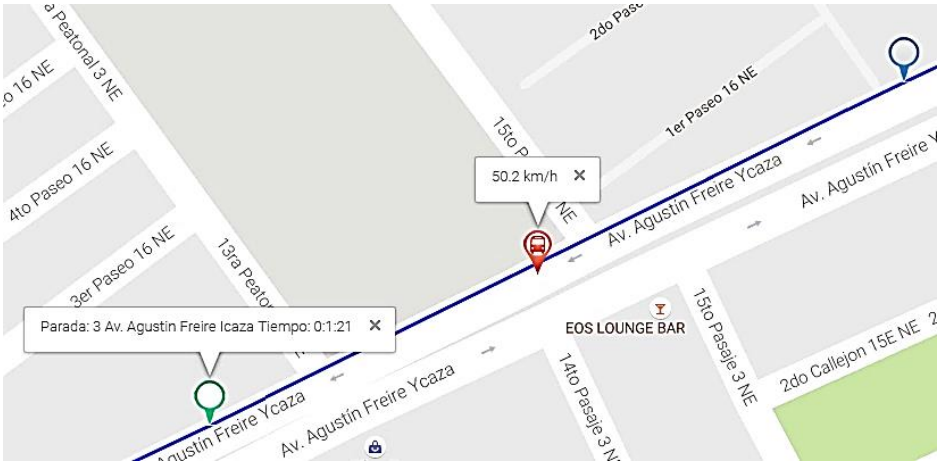


Figura 3.18 Marcadores correspondientes a Parada por recorrer, recorrida y vehículo con sus respectivas informaciones

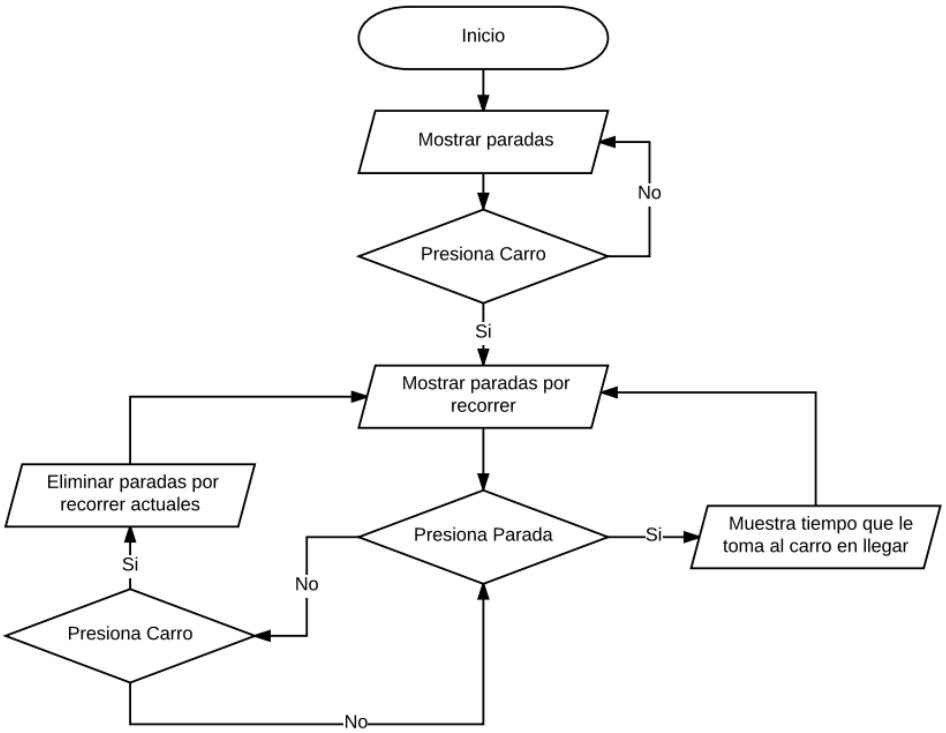


Figura 3.19 Algoritmo de interacción entre el usuario y la interfaz web para conocer las paradas que faltan por recorrer a un vehículo seleccionado

Este proceso se desarrolla en el mismo javascript mencionado anteriormente, en la función *showMarkers* se añade un evento que al momento de hacer clic en el vehículo muestre la velocidad del mismo e invoque a la función *posCarro,* la cual recibe como parámetro el mapa y el marcador seleccionado. Cabe recalcar que dentro de la función *showMarkers* también se va almacenando el número de carro, latitud, longitud y velocidad actual dentro de una lista llamada *posFinal* para usarlo posteriormente.

La función *posCarro* vuelve a hacer uso del diccionario que contiene la lista de los carros activos dentro de la ruta, pero con la finalidad de encontrar el número de recorrido de la última posición del vehículo. Una vez determinado este valor, dentro de la misma función se obtienen todas las posiciones pertenecientes al recorrido actual, las cuales son almacenadas en una nueva lista.

Luego de haber almacenado todos los datos, se llama a la función `distRecorridoCarro(carro)`, la cual recibe como parámetro el número de carro y devuelve el valor de la distancia que ha recorrido el vehículo hasta el momento. Una vez obtenido este valor, se invoca a `determinarTiempos(total, velocidad, map);` la cual recibe como parámetros el mapa inicial, la velocidad y el valor de la distancia recorrida por el vehículo.

Esta última función recorre la lista perteneciente a paradas y va comparando las distancias de cada parada con la distancia recorrida por el carro, cuya finalidad es la de determinar entre qué paradas se encuentra actualmente el vehículo. Pero antes de realizar este proceso, la función verifica que la lista que se crea a continuación esté vacía, de no ser así, elimina todos los marcadores contenidos en la lista, esto se realiza cuando la persona selecciona diferentes carros, y las paradas pertenecientes al anterior deben ser eliminadas para evitar confusiones. Ejemplo que se muestra en la figura 3.20.

Figura 3.20 Paradas por recorrer pertenecientes diferentes carros

Una vez encontradas todas las paradas con distancia mayor a la del vehículo, estas se van guardando en una lista, que aparte de recibir las coordenadas y el mapa también recibe como coordenada el tiempo que le toma al carro en llegar a dicha parada, este tiempo se lo determina con la fórmula 2.1,

$$t = \frac{\mathrm{d}}{v} \qquad (2.1)$$

donde el valor de *d* se lo obtiene de la resta de distancias entre la parada y la recorrida por el vehículo.

Dado que la velocidad se encuentra expresada en $^{km}/_{h}$, el tiempo obtenido se expresa en horas, por lo que, es necesaria la intervención de un convertidor que devuelva este valor en el formato hh:mm:ss, el cual lo realiza la función *determinarTiempo* mostrada en la figura 3.21, que recibe como parámetro el tiempo en horas y lo devuelve en el formato deseado.

```
function determinarTiempo(t, map){
    hora= t - t%1;
    h = t - hora;
    minuto =(( h * 60 )%60)-(( h * 60 )%60)%1;
    m = t - h- minuto;
    segundo = ((((  h * 60 )%60) * 60)%60 ) - ((((  h * 60 )%60) * 60 )%60) %1 ;
    tiempo=hora+":"+minuto+":"+segundo;
    return tiempo;
}
```

Figura 3.21 Función que dado un tiempo en segundos devuelve el tiempo en formato hh:mm:ss

Finalmente, todos los marcadores almacenados en la lista se colocan en el mapa con un icono personalizado del mismo tamaño al de paradas, pero de diferente color. Para resaltar cuales son las paradas que el vehículo debe recorrer, así mismo estas paradas tienen un evento que al momento de presionarlas, muestran el tiempo que tardará el carro en llegar a dicha parada.

- **Mi Ruta**

  Otra de las actividades que puede realizar el usuario es la de Mi Ruta, la cual dada una posición origen y destino, se muestra un mapa que contiene todos los recorridos disponibles y se traza una ruta indicando cómo llegar desde el origen indicado hasta la parada más cercana de este sistema de transporte, y a su vez otra ruta desde otra parada hasta la dirección destino. Este algoritmo se lo visualiza en la figura 3.22.

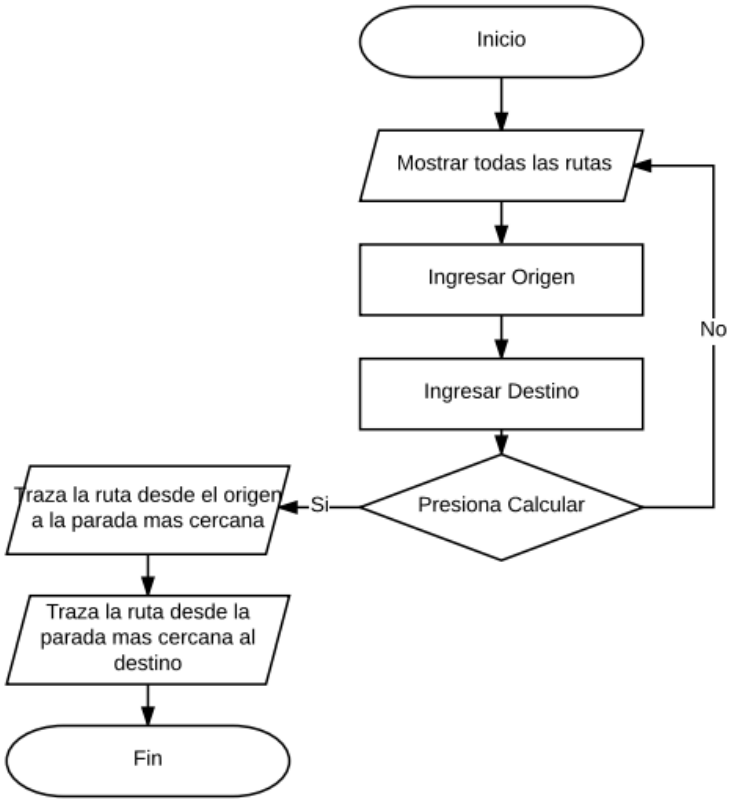


Figura 3.22 Algoritmo que representa la interacción entre el usuario y la opción de calcular ruta

Debido a que se conoce el número total de rutas existentes, para este algoritmo se hace uso de un nuevo javascript, el cual grafica todas estas rutas dentro de un mismo mapa, de la misma forma como se grafican las rutas individuales. Adicional a esto el usuario tiene una sección para poder ingresar el origen y destino desde donde desea que se trace la ruta. Para el uso de estas secciones se utiliza el api de Google, el cual es conocido como *SearchBox*, cuya función es sugerirle al usuario una dirección con respecto al texto que está ingresando y como adicional crea un marcador dentro del mapa mostrado, el cual tiene la opción de desplazarse si el usuario así lo desea.

Una vez obtenidas las posiciones de origen y destino insertadas por el usuario, al presionar el botón de calcular, internamente la

función hace uso de *geometry*, otra de las funcionalidades del api de google, la cual determina la distancia entre dos puntos, para lo cual se tomará la más corta entre las posiciones y las paradas graficadas en el mapa. Una vez que encuentra dicha parada, la función muestra la ruta de cómo llegar desde cada punto a las paradas seleccionadas.

3.7 Diagrama del módulo en la parada

El dispositivo consta de un microcontrolador *Arduino*, un módulo *Ethernet*, una pantalla *LCD 16X2* y una batería de 9V, en la Figura 3.23 se muestra cómo se debe realizar la conexión entre estos aparatos electrónicos.

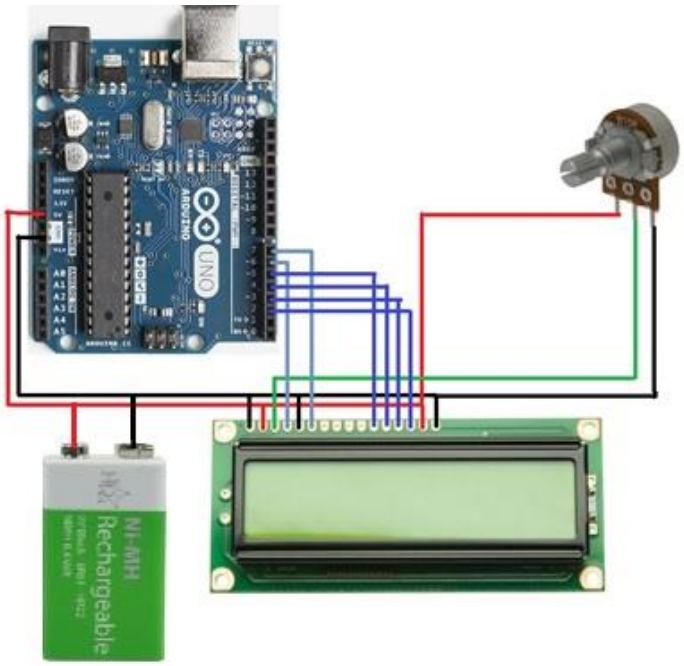


Figura 3.23 Diagrama de conexión del módulo en parada

De donde se tiene la siguiente configuración de pines:

- +5V Arduino -> Pines: 2 y 15 LCD. Pin 3 Potenciómetro
- Gnd Arduino -> Pines: 1, 5 y 16 LCD. Pin 1 Potenciómetro
- Pin 2 Arduino -> Pin 14 LCD
- Pin 3 Arduino -> Pin 13 LCD
- Pin 4 Arduino -> Pin 12 LCD

- Pin 5 Arduino -> Pin 11 LCD
- Pin 6 Arduino -> Pin 4 LCD
- Pin 7 Arduino -> Pin 6 LCD
- Pin 3 LCD -> Pin 2 Potenciómetro (Salida)

Para que la parada pueda mostrar el tiempo del vehículo más cercano, como en los casos anteriores se creó un diccionario que indica un único valor, el cual contiene el tiempo que demora el vehículo más cercano en llegar a dicha parada. La clave de este diccionario es el id de la parada registrado en la base de datos el cual está colocado de forma fija en cada dispositivo. Para determinar este tiempo se realizó una función que inicialmente filtra todas las últimas posiciones de los vehículos que se encuentran actualmente en la ruta correspondiente.

Una vez obtenidas estas rutas, se hace uso de *Vicentry*, que es una de las funciones de la librería *Geopy* (propia de python), donde, dada dos posiciones (latitud y longitud) devuelve la distancia entre dichos puntos expresada en kilómetros. Para este caso, se determina la distancia más corta entre todos los buses existentes y la posición de la parada, siendo este el bus más cercano a la parada. Una vez encontrada esta distancia, se procede a determinar el tiempo en base a la velocidad correspondiente al bus y se guarda en el diccionario. El dispositivo situado en la parada hará la petición de este valor cada 15 segundos, actualizándose así el tiempo de llegada.

Con el tiempo asignado en el servidor a cada parada, se procede a realizar la lectura de ese valor en el módulo. A continuación, se detalla la programación en el microprocesador Arduino que permite conectar el módulo al servidor y mostrar en la *pantalla LCD* el tiempo de llegada de un vehículo.

Dado que se utiliza un módulo Ethernet, se le asigna una dirección MAC al equipo `byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };` También se debe agregar la dirección web del servidor dentro de una variable `char server[] = "mfmora.pythonanywhere.com"`. Además de una dirección IP al dispositivo, misma que debe estar dentro del rango de direcciones proporcionado por la red local `IPAddress ip(192, 168, 0, 177);`. Finalmente se indican los pines que están siendo utilizados por la *pantalla LCD* `LiquidCrystal lcd(6, 7, 5, 4, 3, 2).`

Se inicializa la conexión Ethernet para establecer la comunicación a Internet, mediante el siguiente código: `Ethernet.begin(mac, ip)`. Una vez realizado este proceso se inicializa también la *pantalla LCD* para ejecutar acciones con la misma, para esto se usa `lcd.begin(16, 2)`.

La primera ejecución de la *pantalla LCD* es mostrar el nombre de la parada `lcd.print("Parada 2")`. Luego vienen las acciones que se estarán repitiendo constatemente cada 15 segundos. Por lo que se debe indicar la dirección URL establecida para realizar la petición GET `client.println("GET /transporte/tiempoparada/3/?v=1 HTTP/1.1")`. Para el código anterior el número 3 corresponde al número de parada incrementado en 1, es decir que se está refiriendo a la parada número 2. Valor que debe ser reemplazado de acuerdo a la parada en cuestión.

El tiempo que se requiere viene dado dentro del HTTP response que retorna la petición GET, por tal motivo se recorre esta respuesta hasta llegar a la ubicación del tiempo y almacenarlo en una variable. Una vez obtenido el tiempo, se ubica el cursor de la *pantalla LCD* en la segunda fila, para no perder el nombre de la parada `lcd.setCursor(0,1);` y se lo imprime por pantalla `lcd.print(response)`. Finalmente en cada iteración se necesita detener la comunicación con `client.stop();` para luego inicializar una diferente. En la figura 3.24 se muestra el producto final a usarse en cada una de las paradas.

Figura 3.24 Módulo situado en la parada

CAPÍTULO 4

4. RESULTADOS

En este capítulo se analizan todos los resultados obtenidos durante la elaboración del producto final. Se realizaron las pruebas necesarias para constatar que se ha elaborado un producto de calidad y comprobar su correcto funcionamiento según los requerimientos del sistema.

4.1 Respuesta del GPS

Dentro de los valores obtenidos por parte del GPS integrado en el módulo SIM808 se obtuvo un valor de velocidad mínima de 0.00Km/h y una velocidad máxima de 82.30Km/h. Lo cual se muestra en los eventos presentados más adelante. Se evidenció que la conexión entre el servidor y el módulo se realiza después de un minuto de haber obtenido los primeros valores.

4.1.1 Escenario de velocidad baja

Esta prueba tuvo una duración de 3 min, la cual se realizó en un tramo del recorrido de la Ruta Alborada.

- **Resultados**

  Se obtuvo una velocidad máxima de 28.4km/h y una mínima de 0km/h. Por lo que se evidenciaron variaciones extremas al momento de calcular el tiempo cuando la velocidad era menor a 5km/h. Motivo por el cual se modificó la función de calcular el tiempo a las estaciones intermedias de tal forma que, si el valor de velocidad recibida era menor a 5km/h se tome el valor de la velocidad de la posición anterior verificando así mismo que sea superior a la velocidad mínima. Para este escenario no se obtuvo pérdidas durante el envío de datos, los mismos que fueron 12 en total mostrados en la figura 4.1, lo cual se muestra en la salida serial generada por el módulo y la figura 4.2.

```
-2.134420, -79.899140, 0.20
True
OK
-2.134042, -79.899410, 16.1
True
OK
-2.133857, -79.899548, 0.00
True
OK
-2.133815, -79.899550, 8.04
True
OK
-2.133353, -79.898932, 27.5
True
OK
-2.132682, -79.898038, 25.6
True
OK
-2.132265, -79.897600, 0.37
True
OK
-2.132322, -79.897802, 21.5
True
OK
-2.133017, -79.898720, 26.7
True
OK
-2.133657, -79.899595, 28.4
True
OK
-2.133697, -79.900125, 14.0
True
OK
-2.133215, -79.900492, 17.5
True
OK
```

Figura 4.1 Datos recibidos en el primer escenario

Codigo	Numero de Recorrido	Fecha	Placa	Latitud	Longitud	Velocidad
100	5	6 de Septiembre de 2016 a las 08:23	GOB0048	-2.134420	-79.899140	0.20
100	5	6 de Septiembre de 2016 a las 08:23	GOB0048	-2.134042	-79.899410	16.1
100	5	6 de Septiembre de 2016 a las 08:23	GOB0048	-2.133857	-79.899548	0.00
100	5	6 de Septiembre de 2016 a las 08:23	GOB0048	-2.133815	-79.899550	8.04
100	5	6 de Septiembre de 2016 a las 08:24	GOB0048	-2.133353	-79.898932	27.5
100	5	6 de Septiembre de 2016 a las 08:24	GOB0048	-2.132682	-79.898038	25.6
100	5	6 de Septiembre de 2016 a las 08:24	GOB0048	-2.132265	-79.897600	0.37
100	5	6 de Septiembre de 2016 a las 08:24	GOB0048	-2.132322	-79.897802	21.5
100	5	6 de Septiembre de 2016 a las 08:25	GOB0048	-2.133017	-79.898720	26.7
100	5	6 de Septiembre de 2016 a las 08:25	GOB0048	-2.133657	-79.899595	28.4
100	5	6 de Septiembre de 2016 a las 08:25	GOB0048	-2.133697	-79.900125	14.0
100	5	6 de Septiembre de 2016 a las 08:25	GOB0048	-2.133215	-79.900492	17.5

Figura 4.2 Información almacenada en el servidor correspondiente a los valores de ubicación enviados por el módulo del vehículo para el escenario 1

4.1.2 Escenario de velocidad moderada

Esta prueba tuvo una duración de 4 min, la cual se realizó en un tramo de uno de los recorridos. Los resultados se muestran en la salida serial generada por el módulo y la figura 4.4.

- **Resultados**

  Para este escenario no se obtuvo pérdidas durante el envío de datos, ya que se recibieron todas las ubicaciones enviadas por el modulo, mismas que se muestran en la figura 4.3, donde la velocidad máxima del automóvil fue de 36.9km/h y la mínima de 7.76km/h.

```
-2.132663, -79.901097, 7.76
True
OK
-2.132228, -79.901873, 34.8
True
OK
-2.131747, -79.902773, 10.7
True
OK
-2.132222, -79.901893, 32.9
True
OK
-2.132873, -79.900780, 32.5
True
OK
-2.133855, -79.900075, 14.6
True
OK
-2.133118, -79.900583, 36.9
True
OK
-2.132315, -79.901692, 24.1
True
OK
-2.131857, -79.902530, 31.3
True
OK
-2.131797, -79.902720, 18.2
True
OK
-2.132250, -79.901855, 24.5
True
OK
-2.132925, -79.900735, 30.9
True
OK
-2.133728, -79.900162, 15.2
True
OK
-2.134013, -79.900023, 17.6
True
OK
-2.134470, -79.900423, 17.9
True
OK
-2.133868, -79.899627, 21.0
True
OK
```

Figura 4.3 Datos recibidos en el segundo escenario

Codigo	Numero de Recorrido	Fecha	Placa	Latitud	Longitud	Velocidad
100	5	6 de Septiembre de 2016 a las 08:29	GOB0048	-2.132663	-79.901097	7.76
100	5	6 de Septiembre de 2016 a las 08:29	GOB0048	-2.132228	-79.901873	34.8
100	5	6 de Septiembre de 2016 a las 08:30	GOB0048	-2.131747	-79.902773	10.7
100	5	6 de Septiembre de 2016 a las 08:30	GOB0048	-2.132222	-79.901893	32.9
100	5	6 de Septiembre de 2016 a las 08:30	GOB0048	-2.132873	-79.900780	32.5
100	5	6 de Septiembre de 2016 a las 08:30	GOB0048	-2.133855	-79.900075	14.6
100	5	6 de Septiembre de 2016 a las 08:31	GOB0048	-2.133118	-79.900583	36.9
100	5	6 de Septiembre de 2016 a las 08:31	GOB0048	-2.132315	-79.901692	24.1
100	5	6 de Septiembre de 2016 a las 08:31	GOB0048	-2.131857	-79.902530	31.3
100	5	6 de Septiembre de 2016 a las 08:31	GOB0048	-2.131797	-79.902720	18.2
100	5	6 de Septiembre de 2016 a las 08:32	GOB0048	-2.132250	-79.901855	24.5
100	5	6 de Septiembre de 2016 a las 08:32	GOB0048	-2.132925	-79.900735	30.9
100	5	6 de Septiembre de 2016 a las 08:32	GOB0048	-2.133728	-79.900162	15.2
100	5	6 de Septiembre de 2016 a las 08:32	GOB0048	-2.134013	-79.900023	17.6
100	5	6 de Septiembre de 2016 a las 08:33	GOB0048	-2.134470	-79.900423	17.9
100	5	6 de Septiembre de 2016 a las 08:33	GOB0048	-2.133868	-79.899627	21.0

Figura 4.4 Información almacenada en el servidor correspondiente a los valores de ubicación enviados por el módulo del vehículo para el escenario 2

4.1.3 Escenario de velocidad variada

Esta prueba se la realizó 2 veces desde la Terminal Río Daule, siendo el inicio de la ruta Alimentadora Alborada, hasta las calles Benjamín Carrión y R Baquerizo Nazur. Recorrido que se realizó en un intervalo de tiempo de 13 minutos para la primera prueba y 10 minutos para la segunda.

- **Resultados**

  Para la primera prueba se recibieron 44 ubicaciones de las 50 enviadas, representando un 12% de paquetes que no lograron ser enviados al servidor. Esta pérdida de paquetes se evidenció en el monitoreo de los vehículos, ya que se produjeron de forma consecutiva. Sin embargo, no afectó a la determinación de los tiempos hacia las paradas.

  Para la segunda prueba se recibieron 34 ubicaciones de las 38 enviadas, representando un 10% de pérdida de paquetes. Estos

resultados se muestran en la figura 4.5 y la siguiente salida de código por parte del módulo del vehículo.

Codigo	Numero de Recorrido	Fecha	Placa	Latitud	Longitud	Velocidad
100	5	5 de Septiembre de 2016 a las 23:27	GOB0048	-2.141180	-79.879500	13.4
100	5	5 de Septiembre de 2016 a las 23:27	GOB0048	-2.140992	-79.880352	32.0
100	5	5 de Septiembre de 2016 a las 23:27	GOB0048	-2.140543	-79.881725	38.1
100	5	5 de Septiembre de 2016 a las 23:28	GOB0048	-2.139748	-79.882868	28.9
100	5	5 de Septiembre de 2016 a las 23:28	GOB0048	-2.139372	-79.883183	0.00
100	5	5 de Septiembre de 2016 a las 23:28	GOB0048	-2.139013	-79.883605	21.5
100	5	5 de Septiembre de 2016 a las 23:28	GOB0048	-2.139402	-79.884675	0.83
100	5	5 de Septiembre de 2016 a las 23:29	GOB0048	-2.139417	-79.884687	.02
100	5	5 de Septiembre de 2016 a las 23:29	GOB0048	-2.139935	-79.885592	26.2
100	5	5 de Septiembre de 2016 a las 23:29	GOB0048	-2.140108	-79.886032	0.00
100	5	5 de Septiembre de 2016 a las 23:29	GOB0048	-2.140315	-79.886503	9.32
100	5	5 de Septiembre de 2016 a las 23:30	GOB0048	-2.140395	-79.886642	7.46
100	5	5 de Septiembre de 2016 a las 23:30	GOB0048	-2.140852	-79.887597	33.4
100	5	5 de Septiembre de 2016 a las 23:30	GOB0048	-2.141060	-79.888048	0.00
100	5	5 de Septiembre de 2016 a las 23:30	GOB0048	-2.141283	-79.888540	30.7
100	5	5 de Septiembre de 2016 a las 23:31	GOB0048	-2.141795	-79.889523	35.3
100	5	5 de Septiembre de 2016 a las 23:31	GOB0048	-2.142067	-79.890163	0.06
100	5	5 de Septiembre de 2016 a las 23:31	GOB0048	-2.142280	-79.890545	29.9
100	5	5 de Septiembre de 2016 a las 23:31	GOB0048	-2.142827	-79.891663	33.8
100	5	5 de Septiembre de 2016 a las 23:32	GOB0048	-2.143193	-79.892470	0.00
100	5	5 de Septiembre de 2016 a las 23:32	GOB0048	-2.143388	-79.892857	28.3
100	5	5 de Septiembre de 2016 a las 23:32	GOB0048	-2.143570	-79.893227	37.7
100	5	5 de Septiembre de 2016 a las 23:33	GOB0048	-2.144107	-79.894387	1.61
100	5	5 de Septiembre de 2016 a las 23:34	GOB0048	-2.141523	-79.896997	35.5
100	5	5 de Septiembre de 2016 a las 23:34	GOB0048	-2.140408	-79.897683	9.35
100	5	5 de Septiembre de 2016 a las 23:35	GOB0048	-2.139865	-79.898047	31.3
100	5	5 de Septiembre de 2016 a las 23:35	GOB0048	-2.139565	-79.898268	0.00
100	5	5 de Septiembre de 2016 a las 23:35	GOB0048	-2.139565	-79.898268	0.00
100	5	5 de Septiembre de 2016 a las 23:35	GOB0048	-2.139232	-79.898512	34.0
100	5	5 de Septiembre de 2016 a las 23:36	GOB0048	-2.137967	-79.899408	43.1
100	5	5 de Septiembre de 2016 a las 23:36	GOB0048	-2.136575	-79.900452	52.4
100	5	5 de Septiembre de 2016 a las 23:36	GOB0048	-2.135403	-79.901288	7.78
100	5	5 de Septiembre de 2016 a las 23:36	GOB0048	-2.135383	-79.901297	0.00
100	5	5 de Septiembre de 2016 a las 23:37	GOB0048	-2.135378	-79.901298	4.98

Figura 4.5 Información almacenada en el servidor correspondiente a los valores de ubicación enviados por el módulo del vehículo para el escenario 3

4.2 Alimentación de Módulos

Para la implementación de los módulos situados en el vehículo y en la parada se utilizaron baterías alcalinas de 9Vsobre las cuales se realizó una prueba de envío de datos consecutivos para comprobar su duración. Los resultados de esta prueba revelan que el tiempo máximo de duración de las baterías para el módulo

utilizado en el vehículo es de 5 horas. En el caso del módulo utilizado en una estación intermedia, el consumo de energía es menor, teniendo un tiempo de 10 horas de vida útil de la batería.

Con los resultados obtenidos se genera un costo elevado de mantenimiento, dado que se consideraría realizar cambios frecuentes de las baterías. Una opción para reducir el consumo de estos elementos, sería obtener energía proveniente de la batería del vehículo. Así mismo para el caso de las paradas se podría obtener alimentación de la energía pública. De esta manera se reducirían costos operativos, y la eficiencia del sistema aumentaría considerablemente.

Otra opción sería el uso de baterías recargables, en el caso de que los vehículos hicieran pocos recorridos durante el día, ya que estas tienen una duración aproximada de 8 horas.

4.3 Consumo de datos

Durante las pruebas realizadas tanto para la respuesta del GPS como para la duración de las baterías se establecieron 5 diferentes escenarios en los cuales se determinó la eficiencia de la transmisión y el consumo de envío de los datos. Esta información se la muestra en la Tabla 1.

Escenario	Tiempo (min)	Paquetes enviados	Paquetes recibidos	Eficiencia (%)	Consumo de datos
1	3	12	12	100	0.015MB
2	4	15	15	100	0.021MB
3	13	50	44	88	0.085MB
4	10	38	34	89.47	0.053MB
5	300	769	692	89.98	0.948MB

Tabla 1 Datos obtenidos en los diferentes escenarios planteados

Teniendo en cuenta el consumo de datos por cada escenario, se puede calcular el consumo por cada envío. Los valores se detallan en la Figura 4.6.

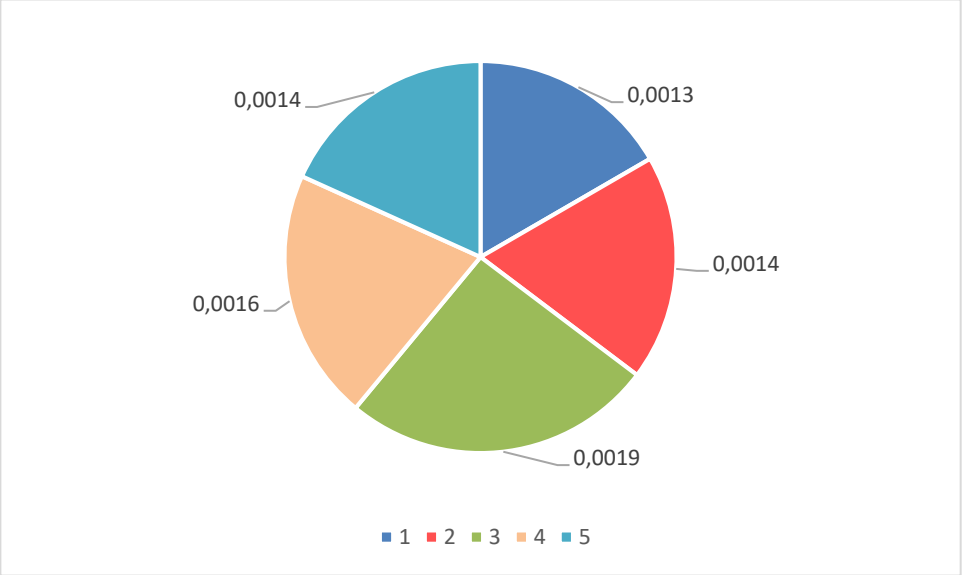


Figura 4.6 Representación del consumo de envío de cada dato en los diferentes escenarios

En donde se puede observar que el consumo de envío por cada dato es mínimo 0,0015MB, los cuales generan un consumo promedio de 35MB mensuales.

4.4 Administración de Aplicación Web

El administrador puede obtener diferentes reportes que son útiles para llevar el control y monitoreo de los vehículos, así como del sistema en general.

Al tratarse de un sistema masivo se requerirá la contratación de varios administradores, ya que como se menciona en los capítulos anteriores, este sistema depende 100% del administrador para su exitoso funcionamiento.

Para evitar esta dependencia se puede implementar un sistema de lectura de tarjetas RFID. Donde el aumento del número de recorrido del vehículo, el cambio de estado de activo a inactivo y viceversa realizado por el administrador, se efectuaría de manera automática en el momento que el vehículo salga o ingrese de la estación.

4.5 Costos de implementación

Para la implementación del sistema se utilizaron diferentes dispositivos los cuales se detallan sus costos en la tabla 2.

Artículo	*Cantidad*	*Costo por unidad*	*Valor Total*
Arduino UNO	2	$ 30,00	$ 60,00
Módulo SIM808	1	$ 85,00	$ 85,00
Pantalla LCD 16x2	1	$ 8,00	$ 8,00
Módulo Ethernet	1	$ 25,00	$ 25,00
Impresión 3D caja del módulo de vehículo	1	$ 35,00	$ 35,00
Impresión 3D caja del módulo de parada	1	$ 30,00	$ 30,00
Plan de datos de telemetría	1	$5,50	$5,50
Baterías 9V	4	$4	$16
		Total	$ 264,50

Tabla 2 Costos de materiales utilizados

Este es el detalle de costos para la implementación de un vehículo y una parada. De donde se puede obtener que el costo individual del módulo en el vehículo es de $163,50 y del módulo en la parada es de $101. Para este sistema se ha considerado usar un servidor en la nube cuyo precio es de $12 mensuales.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

El módulo SIM808 utilizado en el sistema del vehículo integra la funcionalidad de 2 módulos en 1: GPS y GPRS. lo que hace que el prototipo sea más compacto. Debido a su circuitería, posee un tiempo de respuesta menor al de los módulos funcionando por separado.

De acuerdo a los resultados obtenidos, este sistema tiene una eficiencia del 94% en la entrega de paquetes, los cuales no afectan en gran proporción a la determinación de los tiempos de llegada hacia las paradas. Sin embargo, para reducir esta pérdida de paquetes se puede aumentar el tiempo de envío de datos en el módulo.

Al tratarse de un sistema en el que se involucran más de un vehículo, los precios por unidad se reducirían, ya que dependiendo la cantidad de vehículos se haría una compra al por mayor, tanto en equipos como planes datos y servicios en la nube.

Se ha desarrollado una aplicación web para los usuarios donde encontrarán información sobre los servicios de un sistema de transporte mediante Internet.

Con la implementación de este sistema el usuario podrá conocer en mayor detalle las rutas de transporte público y lo que sucede en cada una de ellas. Igualmente ayudará a la empresa pública a tener un control más exhaustivo de los tiempos que le toma a cada unidad realizar su recorrido, con el objetivo de brindar un mejor servicio al público en general.

Al introducir este sistema en la sociedad, se estaría aportando al crecimiento tecnológico, evolucionando los medios de transporte regulares a inteligentes y aportando al desarrollo de una Smart City.

RECOMENDACIONES

Tener en cuenta el uso de baterías recargables en el módulo del vehículo y en el módulo de la parada para el ahorro de gastos en materiales y así evitar el aumento de desechos no renovables.

La pantalla LCD del prototipo realizado es de cortas dimensiones, pero para la implementación en campo, se debe considerar la utilización de un monitor o el uso de vayas leds, que pueda estar a la vista del usuario en cada estación intermedia.

La forma actual de cambiar el estado de un vehículo y su número de recorrido se la realiza manualmente. Sin embargo, esta acción puede ser automática, al utilizar un sistema de lectura de tarjetas RFID.

Actualmente se está haciendo uso de un servidor gratuito por tiempo limitado, por lo que la implementación de este sistema debe considerar la contratación de un servidor de pago para subir la base de datos y página web.

Considerar la creación de una aplicación móvil del sistema implementado, de esta manera el usuario puede hacer uso con mayor comodidad del servicio brindado.

Considerar la contratación de un plan de telemetría ofrecido por las operadoras locales, los cuales cuentan con paquetes que van desde los 5MB hasta los 100MB.

BIBLIOGRAFÍA

[1] H. Auvinen y A. Touminen, «Future Transport Systems: Long-term Visions and Socio-technical Transitions,» de *European Transport Research*, London, Springer, 2014, pp. 343-354.

[2] T. Vanoutrive y A. Verhetel, *Smart Transport Networks. Market Structure, Sustainability and Decision Making,* Worcester: Edward Elgar Publishing, 2013.

[3] G. Chiriboga, «Congestión vehicular en 9 ciudades del Ecuador,» El Comercio, 10 07 2013. [En línea]. Available: http://www.elcomercio.com/actualidad/ecuador/congestion-vehicular-ciudades-del-ecuador.html.

[4] Agencia Metropolitana de Tránsito, «Controles y Fiscalización/Pico y Placa,» 28 05 2010. [En línea]. Available: http://www.amt.gob.ec/index.php/servicios/pico-y-placa/controles.html.

[5] MTA-Transit, «2014 MTA Customer Satisfaction Research Results,» 2015. [En línea]. Available: http://web.mta.info/mta/news/books/docs/2014-Local-Bus-CSS-Board-Presentation.pdf.

[6] Metrovia, «Fundación Metrovia,» 17 Abril 2015. [En línea]. Available: http://www.metrovia-gye.com.ec/fundacionmetrovia.

[7] AM-SUR, «Guayaquil: Metrovía con metrobús y Terminal Terrestre,» 11 Enero 2009. [En línea]. Available: http://www.am-sur.com/am-sur/ecuador/r-L-Guay-2008-08-teil2-Guay/05-Guayaquil-03-metrovia-u-term-terrestre-ESP.html.

[8] S. Neumane, «Tránsito,» 19 Noviembre 2012. [En línea]. Available: http://especiales.eluniverso.com/otroguayaquil/transito/.

[9] Guayaquil es mi destino, «Como transportarse,» 22 Septiembre 2015. [En línea]. Available: http://turismo.guayaquil.gob.ec/es/descubre-guayaquil/como-transportarse.

[10] S. Chang, «Metrovia de Guayaquil,» 05 Julio 2015. [En línea]. Available: http://foroeconomiaecuador.com/fee/metrovia-de-guayaquil/.

[11] El Universo, «18,7 millones de usuarios dejaron de utilizar la Metrovía en 2015,» 12 Febrero 2016. [En línea]. Available: http://www.eluniverso.com/noticias/2016/02/12/nota/5398546/187-millones-usuarios-dejaron-utilizar-metrovia-2015.

[12] Metrovia, «Reglas de uso,» 10 04 2014. [En línea]. Available: http://www.metrovia-gye.com.ec/reglasdeuso.

[13] El Universo, «Recorridos de la Metrovía ahora con aplicación móvil,» 28 Enero 2015. [En línea]. Available: http://www.eluniverso.com/noticias/2015/01/28/nota/4489911/recorridos-metrovia-ahora-aplicacion-movil.

[14] H. Barrragán, «What will you do with the W?,» 2012. [En línea]. Available: http://wiring.org.co/.

[15] B. Fry, «Processing Foundation,» 2012. [En línea]. Available: https://processingfoundation.org/. [Último acceso: 22 05 2016].

[16] I. Perez, «Análisis comparativo de las placas Arduino (oficiales y compatibles),» [En línea]. Available: http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/.

[17] Simcom, «GSM/GPRS+GPS Module,» 23 10 2015. [En línea]. Available: http://simcom.ee/documents/SIM808/SIM808%20SPEC_V1507.pdf.

[18] A. Express, «SIM808 Module GSM GPRS GPS,» [En línea]. Available: http://www.aliexpress.com/item/SIM908-Module-GSM-GPRS-GPS-Development-Board-IPX-SMA-with-GPS-Antenna-for-Raspberry-Pi/32267847079.html.

[19] Vishay, «16 x 2 Character LCD,» [En línea]. Available: http://www.engineersgarage.com/sites/default/files/LCD%2016x2.pdf.

[20] D. Barragán, «USO DEL LCD USANDO PIC-BASIC-PRO,» [En línea]. Available: http://www.matpic.com/esp/microchip/lcd.html.

[21] J. Oliva, «Tarjeta SIM GPS Europa,» [En línea]. Available: http://www.espiamos.com/tarjeta-de-datos-sim-para-gps-europa.html.

[22] A. Panchón, «Evolución de los sistemas móviles celulares GSM,» 29 10 2004. [En línea]. Available: http://www.icesi.edu.co/contenido/pdfs/apachon_gsm.pdf.

[23] J. Benitez, «Redes Móviles 3G 4G,» [En línea]. Available: http://es.slideshare.net/c09271/uni-fiee-scm-sesion-12-redes-moviles-3-g4g.

[24] P. Correia, «Introducción,» de *Guía práctica del GPS*, París, Editions Eyrolles, 2000, pp. 1-3.

[25] R. Al, «SISTEMAS DE SATÉLITE,» [En línea]. Available: http://ssatelitales.blogspot.com/2010/04/la-red-estadounidense-gps.html.

[26] Desarrollo-Web, «HTML a fondo,» [En línea]. Available: http://www.desarrolloweb.com/html/.

[27] J. Eguiluz, «Introducción a CSS,» [En línea]. Available: http://www.librosweb.es/libro/css/.

[28] M. Á. S. Maza, «Qué es Javascript,» de *JavaScript*, Málaga, Innovación y cualificación, S.L, 2001, pp. 9-11.

[29] J. Solis, «¿QUÉ ES BOOTSTRAP Y CÓMO FUNCIONA EN EL DISEÑO WEB?,» arweb, 26 09 2014. [En línea]. Available: http://www.arweb.com/chucherias/editorial/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web.htm.

[30] highcharts, «WHAT IS HIGHCHARTS?,» Highcharts, 2016. [En línea]. Available: http://www.highcharts.com/products/highcharts.

[31] Google developers, «API de Google Maps para web,» [En línea]. Available: https://developers.google.com/maps/documentation/javascript/?hl=es.

[32] Anónimo, «DBA Support,» 21 06 2014. [En línea]. Available: http://www.dbasupport.com.mx/index.php/2-uncategorised/132-caracteristicas-de-mysql. [Último acceso: 23 05 2016].

[33] M. A. Alvarez, «Qué es Python,» 09 11 2003. [En línea]. Available: http://www.desarrolloweb.com/articulos/1325.php.

[34] Django, «Django Documentation,» [En línea]. Available: https://docs.djangoproject.com/en/1.9/.

ANEXOS

Anexo A. Trabajos Futuros

Inicialmente se tiene desarrollado este sistema enfocado en el servicio de movilización masiva “Metrovía” dentro de la ciudad de Guayaquil. Sin embargo, el prototipo es adaptable a cualquier medio de transporte sea este público o privado indistintamente de la ubicación geográfica.

Por ejemplo, se podría implementar este sistema para los servicios de expreso escolar; con la diferencia de que este será privado, ya que cada padre de familia podrá tener su propio usuario y contraseña para ingresar al sistema. De esta forma ellos podrán conocer el momento en el que el expreso está cerca de sus hogares ya sea a la hora de partida y retorno, la velocidad a la que viaja el vehículo en el que van sus hijos y el momento en el que llegan o salen de la institución educativa. Para este caso bastaría tener un solo administrador dependiendo de la cantidad de vehículos que la empresa posea. Así mismo se eliminarían los módulos en cada parada, reduciendo costos de implementación.

Otro enfoque que se puede adaptar a este sistema es el de vigilancia para los buses interprovinciales. Si bien es cierto existen varios asaltos o accidentes dentro de las principales vías de nuestro país. Al colocar estos módulos en cada uno de los vehículos, se podría conocer si el vehículo tuvo algún desvío inesperado, en caso de así serlo se podrá llamar inmediatamente al conductor para determinar las posibles causas, mismas que pueden ser daños en las vías, accidente que ocasiona tráfico, o podría tratarse de un asalto que generalmente ocurre mucho en las vías no pobladas. Así mismo al conocer su velocidad se podrá determinar si el vehículo se encuentra averiado o si pueda causar un accidente o infringir la ley por el exceso de velocidad.

Cabe recalcar que este sistema solo podrá ser monitoreado y gestionado por los responsables de las cooperativas e incluso por la agencia nacional de tránsito, para lo cual se necesita tener conectividad entre las rutas establecidas. Pese a que nuestro país no posee cobertura GSM en toda su zona geográfica, el Ecuador está en un proceso de avance tecnológico, para lo cual se espera que en los próximos años exista conectividad en todo el país.

Anexo B. Descripción de las interfaces web mostradas al administrador

En este primer anexo se describen cada una de las plantillas HTML para el administrador que forman parte de la página web; detallando la funcionalidad y uso de cada una de ellas.

La aplicación web posee un inicio de sesión disponible para el administrador, se debe autenticar para poder realizar modificaciones de los datos y monitorear la circulación de los vehículos. Cabe indicar que tanto el usuario y la contraseña son confidenciales y no pueden ser compartidos a terceras personas.

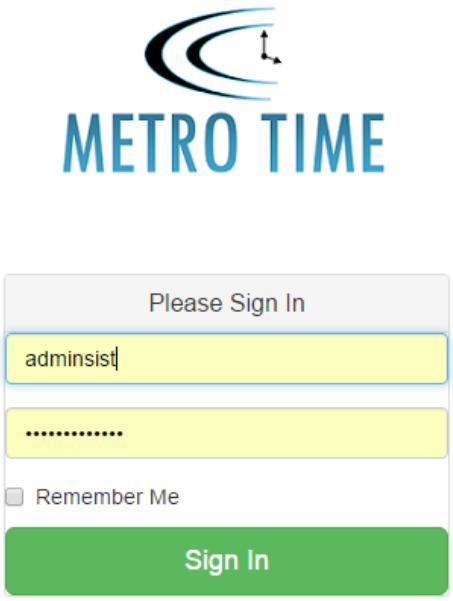


Figura B 1 Pantalla de inicio de sesión para el administrador de la página

Al iniciar sesión se muestra la pantalla de inicio donde el administrador tiene a su alcance las opciones de crear, consultar y modificar los datos del sistema. Así también, se tiene un mapa que permite visualizar todas las rutas establecidas junto con la ubicación de los vehículos activos en ese momento.

Se cuenta con un menú ubicado en la parte superior izquierda, mismo que cuenta con 5 secciones: Inicio, Vehículo, Conductores, Ruta-Vehículo, Alertas. La pantalla de inicio se la muestra en la Figura B 2.

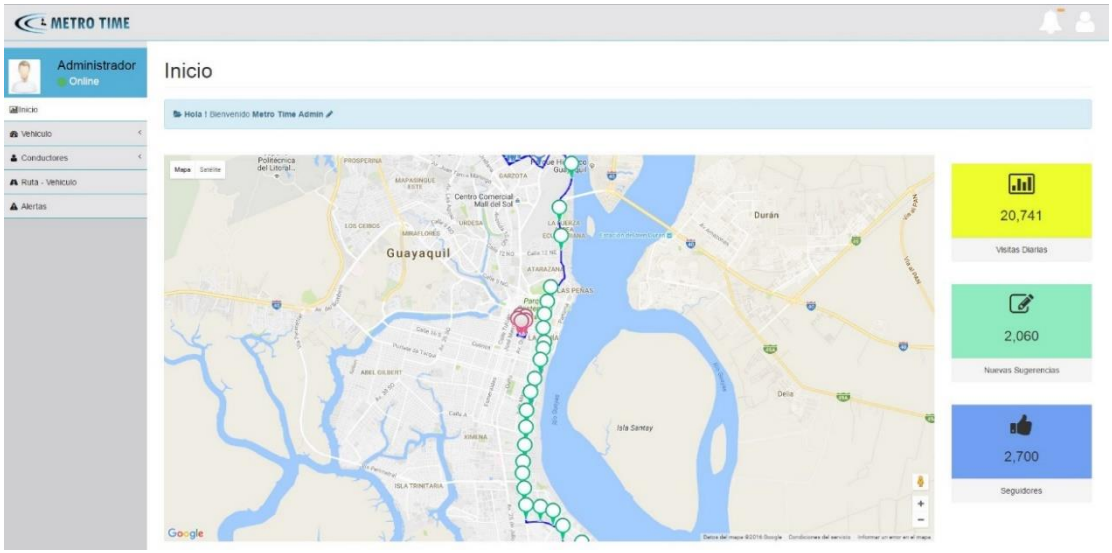


Figura B 2 Pantalla de inicio del administrador

La sección *"Vehículo"* se encarga de mostrar y almacenar información concerniente a las unidades de transporte de la Organización. Se disponen de 4 opciones: Agregar, Información, Velocímetro y Ubicación.

La pantalla *"Agregar nuevo vehículo"* permite añadir un vehículo al sistema con los datos necesarios. Para añadir un vehículo se solicita la siguiente información: placa, número GSM, código y estado. El número GSM es el número correspondiente a la línea telefónica que utiliza el chip del vehículo. Existen 3 tipos de estados: activo, inactivo y mantenimiento. La información solicitada se la muestra en la Figura B 3.

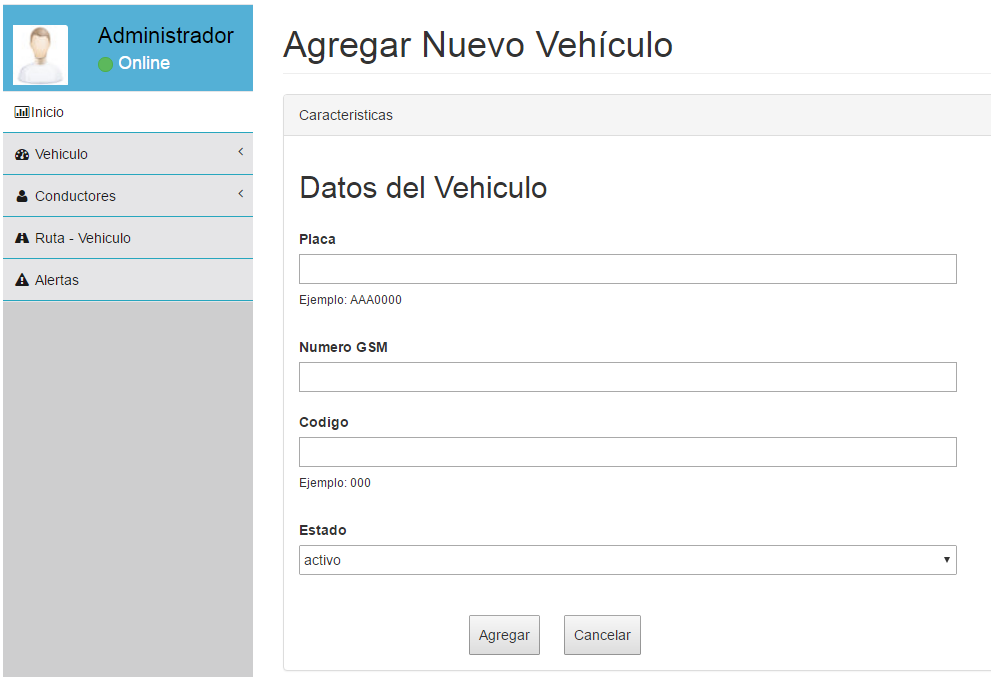


Figura B 3 Pantalla Agregar un nuevo vehículo

La pantalla “Información” permite ver un registro de los vehículos organizándolos en una tabla, los cuales fueron añadidos con la pantalla anterior. Esta pantalla se la muestra en la Figura B 4.

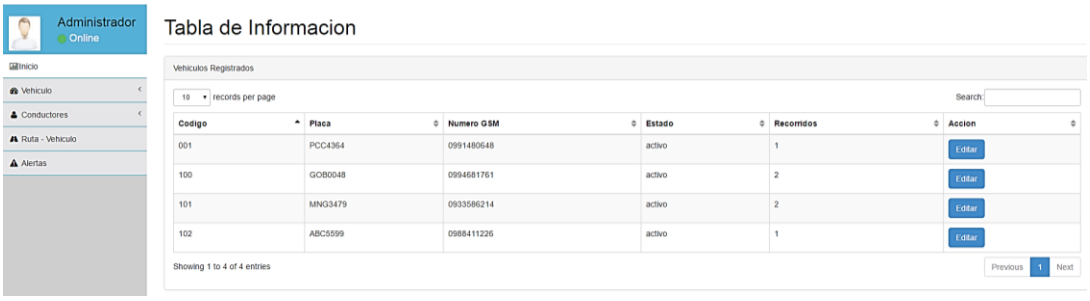


Figura B 4 Pantalla informativa de los vehículos ingresados al sistema

La pantalla “Velocímetro” muestra una gráfica que representa la relación velocidad vs hora. Del vehículo seleccionado, el mismo que se encuentra activo y en circulación. Esta pantalla se la muestra en la Figura B 5.

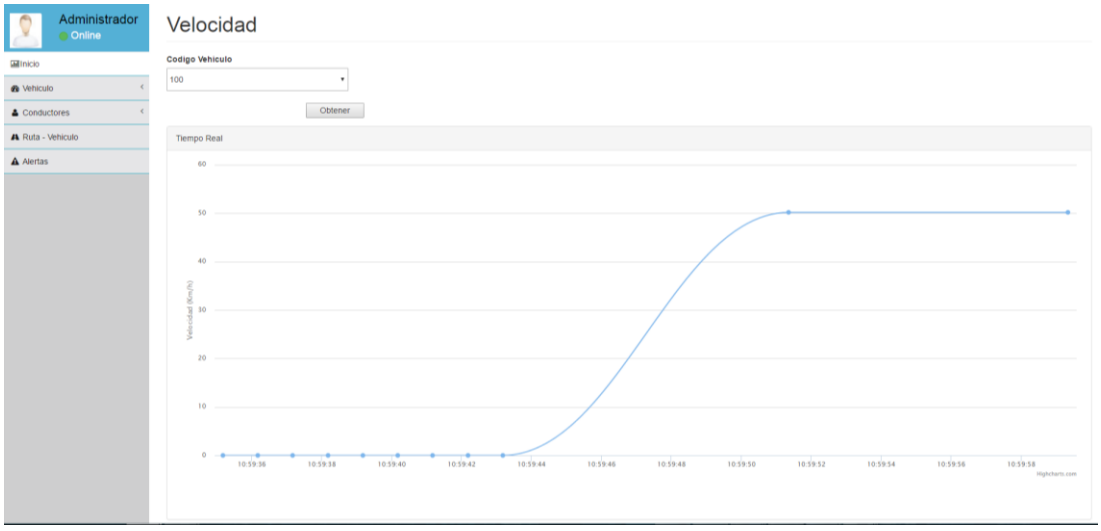


Figura B 5 Pantalla que muestra gráfico de velocidad en tiempo real por cada vehículo

La pantalla perteneciente al submenú de "*Ubicación*" muestras los datos enviados por los módulos, la cual posee la función de filtrar esta información por número de vehículo. Lo cual se aprecia en la figura B 6.

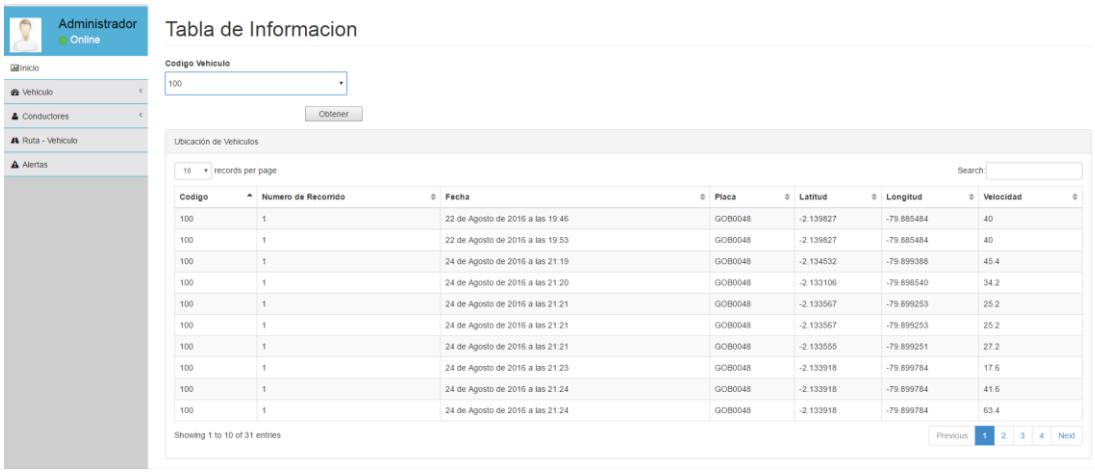


Figura B 6 Pantalla Historial de ubicaciones

La sección *"Conductores"* se encarga de mostrar y almacenar información correspondiente a los conductores que laboran en el establecimiento de transporte urbano. Se disponen de 2 opciones: Agregar e Información.

La pantalla *"Agregar nuevo conductor"* permite añadir un conductor al sistema junto con los datos necesarios. Para añadir un conductor se solicita la siguiente

información: nombres, apellidos, edad, cédula, teléfono, correo electrónico y código de vehículo. El código de vehículo corresponde a la unidad de transporte que tendrá asignado un chofer para conducir. La información solicitada se la muestra en la Figura B 7.

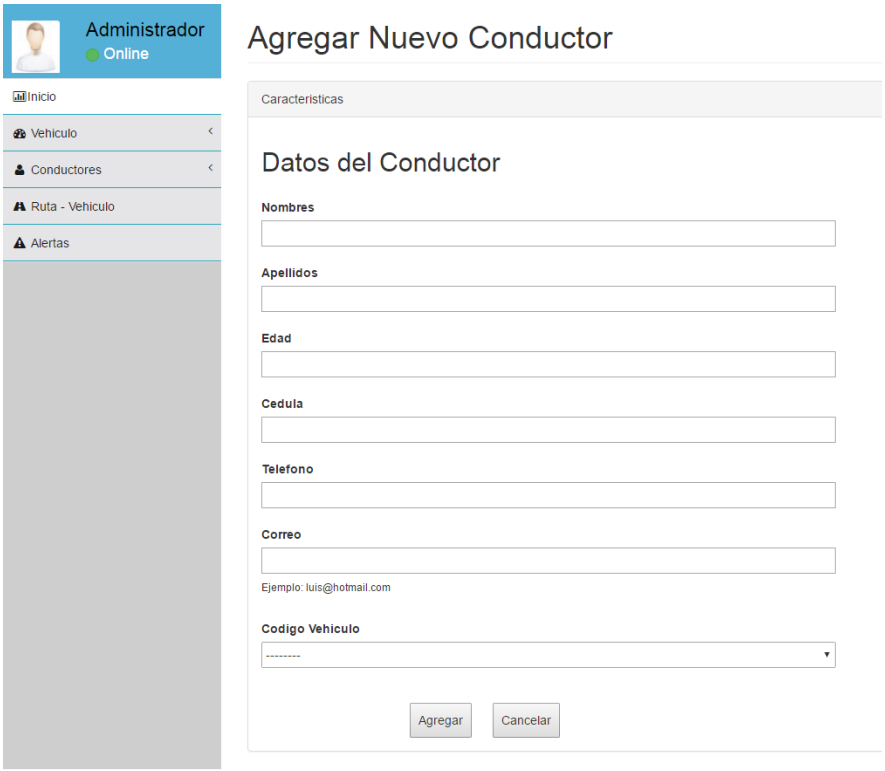


Figura B 7 Pantalla Agregar nuevo conductor

La pantalla "*Información*" muestra los datos de cada uno de los conductores con su respectivo bus asociado. Detalle que se muestra en la figura B 8.

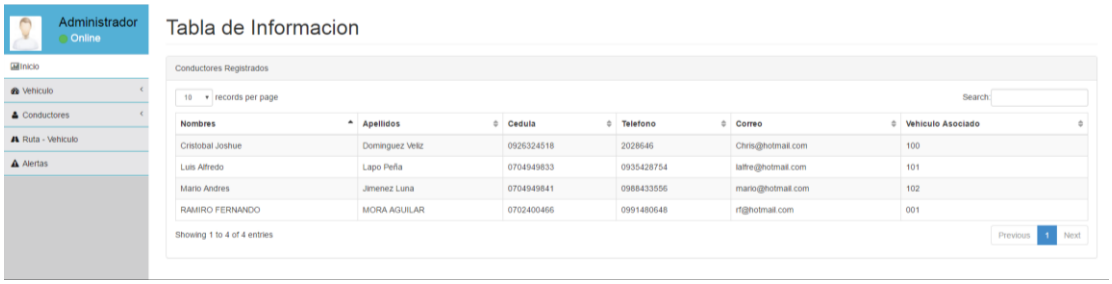


Figura B 8 Pantalla informativa de los conductores ingresados al sistema

La opción de “*Ruta-Vehículo”* permite al administrador asignar vehículos a una ruta específica, en donde un vehículo no puede pertenecer a más de una ruta y esta asociación puede ser eliminada sin problema alguno como lo muestra la figura B 9.

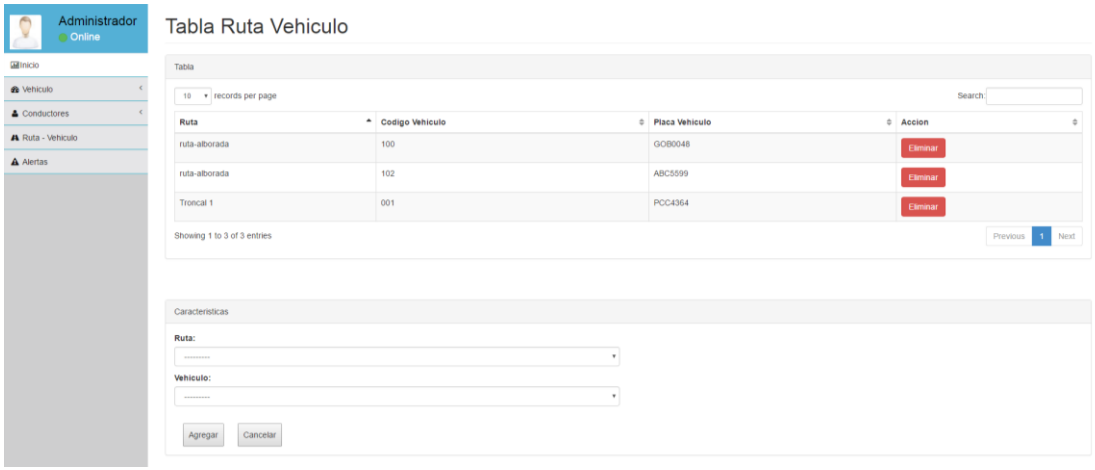


Figura B 9 Pantalla Ruta-Vehículo

Además de mostrarse las notificaciones de los sucesos importantes en cuanto a velocidad de vehículos, al administrador también podrá observar en detalle todas las alertas. Información que se aprecia en la figura B 10.

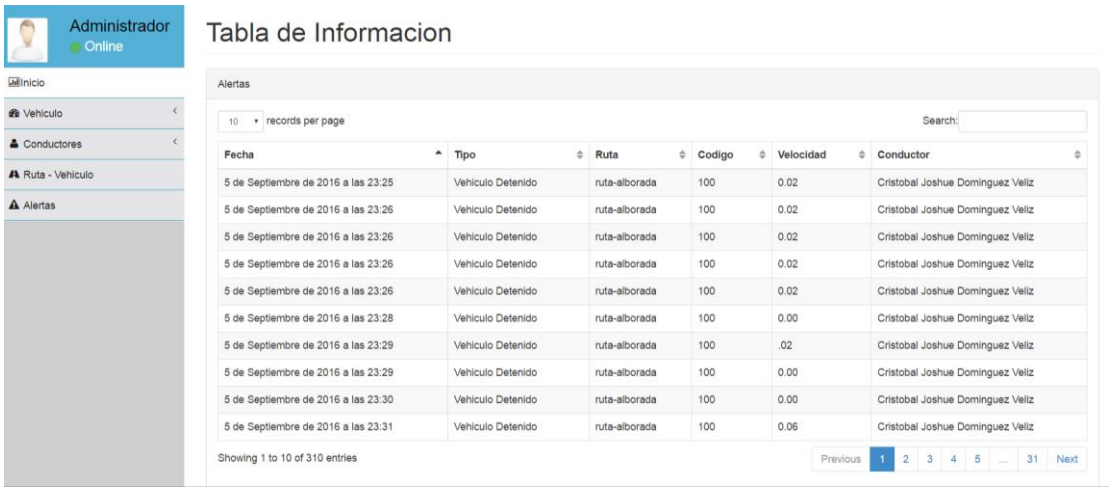


Figura B 10 Pantalla informativa de las alertas generadas

Anexo C. Código para el módulo ubicado en el vehículo

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "gps.h"
#include <String.h>
#define DEBUG true
GPSGSM gps;
SoftwareSerial mySerial(7,8);
char numero_cell[]="+593994681761";
char frame[160];
char lon[11];
char lat[11];
char alt[11];
char tim[20];
char vel[11];
char velo[11];
char latitud[]="-2.091631";
char longitud[]="-79.903490";
char velocidad[]="45.0";
String latitu ="";
String longitu="";
String velocida="";
int tamlat;
int tamlon;
int tamvel;
int cont=0;

void setup()
{
  mySerial.begin(2400);
  Serial.begin(19200);
  delay(500);
}

void loop()
{
  delay(3000);
  GetSignalQuality();
  delay (2000);
  GPS();
}
void GetSignalQuality()
{
  mySerial.println("AT+CSQ");
  delay(100);
  int k=0;
  while(mySerial.available()!=0)
  {
    Serial.write(mySerial.read());
    k+=1;
  }
}
void GPS(){
   mySerial.println("AT+CGNSPWR=1");
   ShowSerialData();
   delay(2000);
   mySerial.println("AT+CGNSSEQ=RMC");
   ShowSerialData();
```

```
   delay(45000);
  while(1){
    mySerial.println("AT+CGNSINF");
    ShowSerialData();
    String response="";
    long int time=millis();
    int leer=0;
    while((time+1000)>millis()){
      while(mySerial.available()){
        char c= mySerial.read();
        response+=c;
        if(leer>45 && leer<55){
          latitu+=c;}
        if(leer>55 && leer<66){
          longitu+=c;}
        if(leer>73 && leer<78){
          velocida+=c;}
        leer++;
      }
    }
    Serial.println(response);
    if(velocida[3]==','){
      velocida[3]=' ';}
    obtenerLatitud();
    obtenerLongitud();
    obtenerVelocidad();
    latitu="";
    longitu="";
    velocida="";
    cont++;
    if(cont>3){
      SubmitHttpRequest();}
    else{
      delay(15000);}
  }
}
void obtenerLatitud(){
  latitu.toCharArray(latitud,10);
  Serial.println(latitud);
}
void obtenerLongitud(){
  longitu.toCharArray(longitud,11);
  Serial.println(longitud);
}
void obtenerVelocidad(){
  velocida.toCharArray(velocidad,11);
  Serial.println(velocidad);
}
void SubmitHttpRequest()
{
  mySerial.println("AT+CSQ");
  delay(100);
  ShowSerialData();
  mySerial.println("AT+CGATT?");
  delay(100);
  ShowSerialData();
  mySerial.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
  delay(1000);
  ShowSerialData();
mySerial.println("AT+SAPBR=3,1,\"APN\",\"internet.claro.com.ec\"");
```

```
  delay(2000);
  ShowSerialData();
  mySerial.println("AT+SAPBR=1,1");
  delay(1000);
  ShowSerialData();
  mySerial.println("AT+HTTPINIT");
  delay(1000);
  ShowSerialData();
  mySerial.print("");

sprintf(frame,"AT+HTTPPARA=\"URL\",\"http://mfmora.pythonanywhere.com/transp
orte/conection/?id_vehiculo=GOB0048&latitud=%s&longitud=%s&velocidad=%s",lat
itud, longitud,velocidad);
  mySerial.print(frame);
  mySerial.println("\"");
  delay(1000);
  ShowSerialData();
  mySerial.println("AT+HTTPACTION=0");
  delay(5000);
  ShowSerialData();
  mySerial.println("AT+HTTPREAD");
  delay(300);
  ShowSerialData();
  mySerial.println("");
  delay(100);
}
void ShowSerialData()
{
  while(mySerial.available()!=0)
  Serial.write(mySerial.read());
}
```

Anexo D. Código para el módulo ubicado en la parada

```
#include <Ethernet.h>
#include <LiquidCrystal.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
char server[] = "mfmora.pythonanywhere.com";
IPAddress ip(192, 168, 0, 177);
LiquidCrystal lcd(6, 7, 5, 4, 3, 2);
EthernetClient client;
String response="";
int i=0;
char m[8];
void setup() {
  lcd.begin(16, 2);
  lcd.print("Parada 2");
  if (Ethernet.begin(mac) == 0) {
    Ethernet.begin(mac, ip);
  }
  delay(1000);
}
void loop() {
  if (client.connect(server, 80)) {
    client.println("GET /transporte/tiempoparada/3/?v=1 HTTP/1.1");
    client.println("Host: mfmora.pythonanywhere.com");
    client.println();
  }
  while(i<280){
  if (client.available()) {
    char c = client.read();
    if(i>270&&i<279){
      response+=c;
    }
    i++;
  }
  }
  client.println("Connection: close");
  i=0;
  lcd.setCursor(0,1);
  if(response!=""){
    lcd.print(response);
  }
  response="";
  client.stop();
  client.flush();
  delay(5000);
}
```