UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO

PROYECTO TÉCNICO

“IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE VELOCIDAD PARA TRANSPORTE URBANO Y CONTROL DE PUERTAS CONDICIONADO POR GEOLOCALIZACIÓN”

Autores:

Paul Iván Cevallos Peñaherrera
Alberto Heinz Alvarez Morales

Tutor:

Ing. Pablo Echeverría Msc.

GUAYAQUIL – ECUADOR
2019

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA

Los estudiantes: Paul Iván Cevallos Peñaherrera con cédula de identidad #**0918475898** y Alberto Heinz Alvarez Morales con cédula de identidad **#0926812488**, declaran que este trabajo de titulación "IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE VELOCIDAD PARA TRANSPORTE URBANO Y CONTROL DE PUERTAS CONDICIONADO POR GEOLOCALIZACIÓN." ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Todos los conceptos desarrollados, análisis realizados y conclusiones del presente trabajo, son exclusiva responsabilidad de los autores y la propiedad intelectual es de la Universidad Politécnica Salesiana.

Guayaquil, febrero del 2019

(f) Paul Cevallos Peñaherrera C.I: 0918475898	(f) Alberto Alvarez Morales C.I: 0926812488

DOCUMENTO DERECHOS DE AUTOR

Los estudiantes: **Paul Iván Cevallos Peñaherrera** con cédula de identidad **#0918475898** y **Alberto Heinz Alvarez Morales** con cédula de identidad **#0926812488**, autorizan a la Universidad Politécnica Salesiana a publicar en la biblioteca virtual de la institución el presente trabajo de titulación "IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE VELOCIDAD PARA TRANSPORTE URBANO Y CONTROL DE PUERTAS CONDICIONADO POR GEOLOCALIZACIÓN." cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Guayaquil, febrero del 2019

(f) Paul Cevallos Peñaherrera
C.I: 0918475898

(f) Alberto Alvarez Morales
C.I: 0926812488

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Por medio de la presente se quiere dejar constancia de que el trabajo de titulación "**IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE VELOCIDAD PARA TRANSPORTE URBANO Y CONTROL DE PUERTAS CONDICIONADO POR GEOLOCALIZACIÓN**." que fue realizado por el **Sr. Paul Cevallos Peñaherrera** y el **Sr. Alberto Alvarez Morales**, estudiantes de la carrera de Ingeniería electrónica en telecomunicaciones, ha sido revisado y concluido en su totalidad, tanto el trabajo escrito como práctico.

Particular que informo para los fines pertinentes.

Atentamente

_____

(f) Ing. Pablo Echeverría Msc.

DEDICATORIA

Este trabajo de titulación quiero dedicarlo en primer lugar a Dios por ser el principal motor de mi vida, a mi madre que ha sido mi guía y el pilar principal para mi formación personal y profesional. A mis abuelitos que han estado conmigo desde el inicio de mi vida. A mi esposa por ser mi compañera de vida y apoyarme en todo momento en mi carrera universitaria. A mi compañero de tesis y amigo con quien he compartido toda mi formación académica y a todas las personas que de alguna u otra forma aportaron para mi formación íntegra, social y profesional durante mi carrera universitaria.

Paul Cevallos

DEDICATORIA.

El presente proyecto investigativo quiero dedicarle en primer lugar a Dios por ser la razón principal de tenerme sano y con vida para poder seguir alcanzando cada una de mis metas propuestas, a mis padres Alberto Alvarez Muñoz y Fanny Morales Freire por sus valores y principios inculcados en mí desde muy pequeño que han servido como enseñanza para enfrentar la vida cada día, dándome siempre su apoyo y confianza para nunca desmayar, por dar todo de ellos y ver a su hijo cumplir sus metas, dándome siempre ánimos y ese impulso por seguir adelante, también a mi querida universidad, directivos, docentes y amigos, quienes me han transmitido sus conocimientos a través de estos años, para ser un profesional capaz y brindar excelente aporte a la sociedad para progresar.

Alberto Alvarez

AGRADECIMIENTO

Nuestros más sinceros agradecimientos a nuestro Padre Celestial quien nos orienta y nos da la fortaleza espiritual necesaria para seguir hacia adelante día a día, a nuestros padres quienes siempre estuvieron apoyándonos desde el primer instante de nuestras vidas y nos han dado las pautas necesarias para enfrentar las dificultades de la etapa universitaria. A cada uno de los docentes que nos han impartido sus conocimientos, los cuales fueron necesarios para la realización de esta obra escrita. A nuestros amigos con quienes compartimos muchos momentos que alegraron y enriquecieron nuestra vida universitaria.

Paul Cevallos Peñaherrera
Alberto Alvarez Morales

RESUMEN

El presente proyecto de titulación plantea el desarrollo de un dispositivo capaz de geolocalizar un bus de transporte urbano con el fin de evitar que esta abra sus puertas en cualquier parte del trayecto, ya que solo se deberían abrir sus puertas en las paradas previamente definidas.

Para geolocalizar el bus de transporte urbano se utilizó la técnica geolocalización por satélite GPS y mediante un dispositivo conectado a la red, los datos se suben a un servidor en la nube para que se puede ver la posición del bus desde cualquier parte del mundo.

Se dispone además de una base de datos en la cuál se registrarán las paradas de buses permitidas, es decir datos de latitudes y longitudes según el GPS en las cuales el bus podrá abrir las puertas.

Para el control de apertura y cierre de las puertas se utilizará una electroválvula la cuál interactuará con el sistema de control del bus de transporte urbano.

El estado del bus será visible desde el internet, mediante un servidor alojado en la nube el cual mostrará los registros en una aplicación web.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Descripción del problema

La ciudad de Guayaquil es una de las urbes más importantes del Ecuador, con mayor número poblacional y con más número de vehículos. En el 2016 la ATM matriculó 334 mil vehículos dando un estimado del parque automotor de 334 mil vehículos**.**

Debido al alto número de vehículos y buses que transitan por la urbe muchas zonas y arterias principales son saturadas. Varios editoriales de periódico ya han informado de dicho problema. La ATM afirmó que en 5 calles de Guayaquil existen congestionamiento vehicular las cuales son la Av. Francisco de Orellana, Juan Tanca Marengo, de las Américas, Carlos Julio Arosemena y la Av. Del Bombero. El Diario El Telégrafo señaló también que en las calles del centro de la urbe existe atasco vehicular, “Entre las calles más señaladas como congestionadas están las avenidas Quito, Machala, 10 de agosto, Clemente Ballén, Víctor Manuel Rendón, Panamá y Boyacá” (El Telégrafo , 2017).

Por otra parte, las leyes de tránsito establecen para el transporte urbano un límite de velocidad máxima de 40 Km/h, pero muchos choferes conducen con exceso de velocidad, poniendo la vida de los pasajeros y peatones en riesgo.

El Diario El Universo publicó que “el número de multas por exceso de velocidad, que es el 30% del salario básico ($ 112,50), va en aumento. Pasaron de 15.572 en todo el 2014 a 260.063 durante noviembre de 2017, según cifras de la Agencia Nacional de Tránsito (ANT) –entidad que planifica, regula y controla la gestión del transporte terrestre, tránsito y seguridad vial en el país. Solo la Autoridad de Tránsito Municipal de Guayaquil (ATM) ha emitido 232.913 multas por exceso de velocidad desde agosto de 2015, cuando asumió la competencia hasta el mismo mes de este año (noviembre 2017)”.

Es evidente que el exceso de velocidad es un constante problema en Guayaquil y además el tráfico en ocasiones llega ser caótico debido a que los conductores paran a dejar y recoger pasajeros en cualquier lugar y momento, irrespetando las paradas de buses previamente seleccionadas y consecuentemente poniendo la vida de los transeúntes en riesgo.

Por lo tanto, ante dicha problemática se plantea la implementación de un sistema de monitoreo de velocidad para el tipo de transporte urbano con el sistema de control de puertas, esto con la finalidad de solucionar el congestionamiento vehicular que se forma especialmente en las calles por donde transita la línea 75 A y de esta forma evitar el crecimiento de multas por exceso de velocidad y accidente donde se ponga en peligro la vida de las personas.

1.2. Delimitación

1.2.1. Temporal

El presente proyecto de titulación será efectuado en el periodo de septiembre del 2018 a febrero del 2019.

1.2.2. Académica

Para el presente proyecto de titulación es necesario disponer de un amplio conocimiento en las ciencias de la programación y sistemas embebidos para el desarrollo del dispositivo, además es necesario tener conocimientos en el área de automatización industrial para la instalación de las electroválvulas.

1.3. Antecedentes

En las últimas décadas se ha podido apreciar el incremento explosivo de la demanda de transportes urbanos que se ha generado especialmente en las grandes urbes de cada región. Este crecimiento vehicular y de transito ha provocado mayor congestión vehicular, muchos accidentes y contaminación ambiental.

Ese incremento brota debido al mayor acceso a los automóviles y vehículos pesados, al crecimiento poblacional, la prolongación del poder adquisitivo de las clases de ingresos medios y la enorme cantidad de ofertas y créditos que hoy en día existen más la reventa de carros usados. Aparte la necesidad de poder transportarse de lugares lejanos o apartados y sobre todo a las escasas políticas estructurales en el transporte público urbano.

Este transporte, incide en el congestionamiento de tránsito tanto en los vehículos particulares o automovilistas y a los usuarios de transportes urbanos públicos, provocando una pérdida de eficiencia económica y otras consecuencias negativas al ambiente y la sociedad.

Según el Anuario de Transportes 2015 del Instituto Nacional de Estadísticas y Censos (INEC) la cantidad de vehículos ascendió un 57 % en un periodo de 5 años en todo el Ecuador.

Por otra parte, como fue escrito en el diario El Telégrafo “de acuerdo con datos del Municipio de Quito, a inicios de esta década el incremento del parque automotor superó ampliamente al aumento poblacional. Mientras el número de vehículos se elevó un 11% anual en promedio, el poblacional es de alrededor del 2%. De los 4’565.000 desplazamientos que ocurren a diario por la ciudad, el 84% utiliza transporte público o auto privado”.

1.4. Importancia

En la actualidad es necesaria una regulación y control en el sistema de transporte urbano para que las leyes de tránsito se cumplan, ya que los transportistas en su afán de aumentar sus ingresos se detienen a dejar y recoger pasajeros en cualquier parte de la ciudad, aumentando la congestión vehicular además de que puede producir accidentes ya que los usuarios del transporte urbano se suben y bajan como pueden del bus.

Con el dispositivo desarrollado en el presente proyecto se geolocalizará el bus de transporte urbano y solamente cuando este esté en una de las paradas previamente definidas podrá abrir sus puertas, fuera de estas zonas el sistema neumático cerrará y bloqueará el acceso a las puertas.

Esto beneficiará al tráfico en general ya que habrá menos buses detenidos en cualquier parte y en su lugar existirán paradas definidas para los usuarios, las cuales contarán con todas las seguridades necesarias para los usuarios del servicio de transporte urbano.

1.5. Alcance

El alcance del presente proyecto de titulación es:

- Implementar en un bus de trasporte urbano el sistema de control de paradas.
- Elaborar un software web mediante el cual se pueda crear, editar y eliminar las paradas permitidas en las que le bus podrá abrir sus puertas.
- Tomar una ráfaga de fotos cada vez que las puertas del bus se encuentren abiertas
- El sistema dispondrá de una interfaz web visible en internet mediante la cual se podrá ver remotamente la ubicación del bus.

1.6. Justificación

En la actualidad la congestión vehicular es ocasionada en gran parte por la irresponsabilidad de los conductores del transporte urbano, los cuales se detienen en cualquier lugar para recoger o dejar pasajeros, pese a que en la ciudad existen puntos específicos para dicha actividad (Sepulcre, Mittag, Santi, Hartenstein, & Gozalvez, 2011).

Esto produce un caos vehicular el cual se agrava en las horas pico y en zonas de alta congestión vehicular, por esta razón es necesario que los buses de transporte urbano no se detengan en cualquier lugar, simplemente en las zonas previamente definidas, ya que estas zonas fueron analizadas y diseñadas para interferir lo menos posible en el tráfico.

Desde el 2017 en la ciudad de Guayaquil se viene dando una transformación vial con la implementación del llamado ‘Corredor vial’ que es un tramo de 26 km para uso exclusivo de 70 buses urbanos pertenecientes a 4 líneas de buses. El objetivo del corredor es que los buses paren, recojan o dejen a los pasajeros en los 104 paraderos ya predefinidos por la ATM.

Con el proposito de asignar el uso del sistema de control se dispondrá de una aplicación web desde la cual se podrá crear, editar y eliminar cada uno de los puntos que serán las paradas autorizadas.

1.7. Objetivos

1.7.1. Objetivo general

Proyectar un sistema de control basado en geolocalización que se integre con el sistema de transporte actual para evitar que estos abran y cierren las puertas en zonas no autorizadas disminuyendo la congestión vehicular.

1.7.2. Objetivos específicos

- Diseñar una plataforma web para administrar las paradas autorizadas usando el lenguaje de servidor PHP y que se ejecute en la Raspberry Pi.

- Usar geolocalización para determinar la posición actual del bus de transporte urbano mediante Arduino.

- Controlar la apertura y cierre de las puertas para el ingreso y salida de usuarios del bus de transporte urbano.

- Configurar el módulo GPS y GSM para monitorear la ubicación y comunicación con la unidad.

- Configurar Arduino Nano para controlar el sistema neumático de las puertas.

- Comunicar el Raspberry PI con el Arduino usando protocolo serie para determinar las paradas del bus.

2. MARCO TEÓRICO REFERENCIAL

En el primer capítulo se describió, que el exceso de velocidad y el tráfico ocasionado por los conductores que paran a dejar y recoger pasajeros en cualquier lugar, es un constante problema que se vive en Guayaquil. Ante dicha problemática, este trabajo de titulación tiene como solución la implementación de un sistema de monitoreo de velocidad para transporte urbanos y control de puertas en paradas autorizadas mediante la utilización de un módulo GPS conectado a un Arduino, y este interconectado con una Raspberry PI (Blum, 2013).

Por otra parte, ya se han registrado estudios relacionados con proyectos que también usan esta versátil herramienta llamada Raspberry PI. A continuación, se describen dichos proyectos relacionados.

2.1. Estudios realizados vinculados

Chávez y Yuquilema (2013), en su trabajo de titulación llamado "Aplicaciones con minicomputadores raspberry pi provisto de módulo gps y acelerómetro para control de velocidad y posicionamiento" plantearon como objetivo principal realizar el control de posición y velocidad de un móvil, mediante la adquisición de datos desde el módulo GPS y a través del puerto de propósitos generales GPI del minicomputador Raspberry PI, donde se concluyó que el minicomputador Raspberry PI es una herramienta muy versátil por la adquisición de datos en tiempo real y que facilita la instalación del módulo GPS, del software de GPS y del programa hecho en Python.

Otro estudio de Jiménez (2015) , titulado "*Construcción de un ordenador electrónico vehicular con sistema de seguridad y gps utilizando raspberry pi y hardware libre*" plantea como objetivos el implementar el Ordenador electrónico vehicular con sistema de seguridad basado en comunicaciones inalámbricas y crear una aplicación móvil para regular el ordenador electrónico del vehículo, donde se concluye que el proyecto de investigación denominado Ordenador Electrónico Vehicular, con Sistema de Seguridad y GPS utilizando Raspberry PI y Hardware Libre, supone un cambio importante en la forma de elevar un nivel de seguridad, monitoreo y control de un vehículo de manera personal, el proyecto contempla un diseño acoplado a la estética del vehículo y la tecnología que hoy en día las personas tenemos acceso.

Según Campuzano y Cedeño (2015), en su trabajo de titulación denominado "*Diseño e implementación de un sistema de seguridad de control local y remoto con dispositivos de vigilancia, desarrolado en el software python, centralizandose en una tarjeta raspberry pi*" planteó como objetivo general diseñar e implementar un sistema de seguridad de una vivienda, por medio del uso de mecanismos electrónicos de vigilancia; controlado por una interfaz de fácil interactividad desarrollado con los programas Python y Qt, para ayudar a mejorar la seguridad en la residencia.

2.2. Sistema de control

Se define como un espacio que recibe acciones del exterior, denominadas también, variables de entradas; las mismas que se procesan en el sistema, dando como respuesta las variables de salidas. Se conoce que existen dos clases de acciones externas: las variables de control, las cuales son las variables que se pueden cambiar; y las de perturbación, a las que no se les puede realizar ningún cambio.

Existen varios tipos de sistemas, en este trabajo de titulación se hablará de un sistema de control, que posee una serie de elementos que van incidir en el funcionamiento del sistema y que tiene como objetivo conseguir el dominio de las variables de salidas a través del manejo de las variables de control, alcanzando de esta forma las variables de salida unos valores predefinidos.

Como lo detalla Vélez (2017), por lo tanto, un sistema de control ideal debe certificar la estabilidad y ser fuerte ante las perturbaciones en los modelos; ser lo más efectivo posible, de acuerdo al criterio; ser amigable en su implementación y cómodo de manejar en tiempo real con ayuda de un computador.

Además, un sistema de control está formado básicamente por 3 elementos: (1) sensores, que se conocen como los que dan paso al conocimiento de las magnitudes de las variables; (2) controlador, utilizando los valores establecidos por los censores; (3) actuador; es el mecanismo que realiza la ejecución calculada por el controlador.

2.3. Geolocalización

Es la capacidad de obtener las coordenadas geográficas de un objeto, con el fin de poder ubicarlo en las 3 dimensiones, el procedimiento para representar la posición de un lugar geográfico sobre la superficie del planeta Tierra es a través del uso de las mediciones esféricas de latitud y longitud.

Como lo describe el autor ARCGIS (2014), en el cual indica que la geolocalización no es más que “mediciones de los ángulos (en grados) desde el centro de la Tierra hasta un punto en su superficie. Este tipo de mecanismo de referencia de coordenadas generalmente se designa sistema de coordenadas geográficas. La longitud tasa ángulos en una dirección este-oeste. Las mediciones de longitud se basan en el meridiano de Greenwich, que es una línea imaginaria que realiza un recorrido desde el Polo Norte, a través de Greenwich, Inglaterra, hasta el Polo Sur. Este ángulo es de longitud 0. El oeste del meridiano de Greenwich por lo habitual se registra como longitud negativa y el este, como longitud positiva. Por ejemplo, la ubicación de Los Ángeles, California, tiene una latitud de aproximadamente +33 grados, 56 minutos y una longitud de -118 grados, 24 minutos.” (Anil K. Jain, 2011).

El sistema del GPS funciona mediante el uso de los satélites que existen en órbita, en la actualizada existen varios satélites en diferentes posiciones orbitando el planeta, el GPS necesita tener acceso al menos a tres para que mediante la

diferencia horaria entre los satélites se pueda triangular la posición Figura #1 del dispositivo GPS (Monk, 2012).

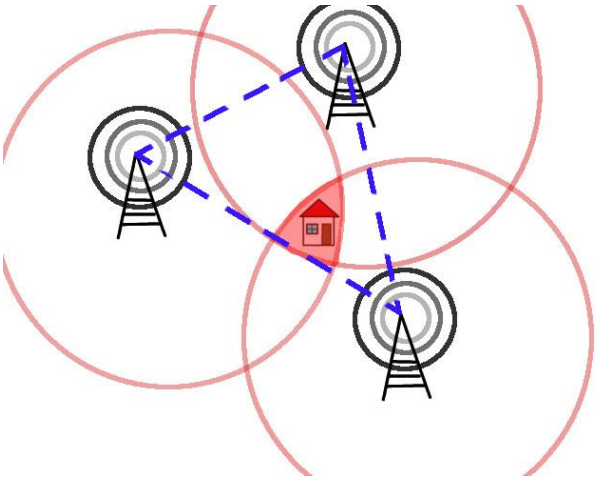

Figura #1 - Funcionamiento del GPS

2.4. Raspberry PI

Es una minicomputadora de placa única, es decir que todos los componentes vienen integrados en una misma placa, el diseño incluye un SOC que contiene la CPU, GPU y memoria RAM

La Raspberry PI tiene la dimensión de una tarjeta de crédito y utiliza un SoC de Broadcom y sus características son:

- Un procesador ARMv6 de 700 MHz
- 256 MB de RAM
- Una GPU 1080 p con salidas HDMI y de video
- Conector de audio de 3,5 mm
- Conector de 26 vías con GPIO, UART, I2C y SPI
- Conectores para JTAG, DSI (display LCD) y CSI (cámara)
- Ranura para tarjeta SD
- USB
- Posee 2 versiones que son:

- Modelo A con un puerto USB

- Modelo B que posee con dos puertos USB y un punto Ethernet

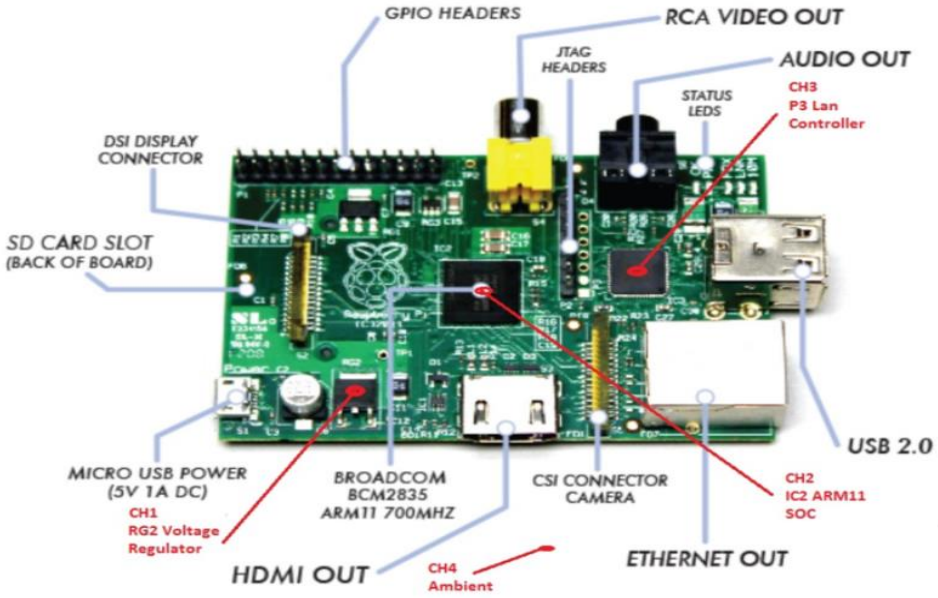

Figura #2 - Elementos de una Raspberry PI modelo B (Franco, 2014)

Los requerimientos que necesita la Raspberry PI para poder trabajar es un mouse y teclado con entrada USB, una fuente de poder que alimente la Raspberry PI, una pantalla, monitor o televisor y una tarjeta SD para poder instalarle el sistema operativo.

Su tamaño tan reducido es lo que facilita para poder llevarlos a varios lugares y además tiene un precio muy reducido.

Como medio de almacenamiento usa una tarjeta SD lo que hace fácil que se pueda instalar y actualizar el sistema a través de otro ordenador, además, las tarjetas se pueden fácilmente cambiar proporcionado de esta forma entornos de software distintos.

Proporciona una gran cantidad de conectores de entradas y salidas lo que fomenta la experimentación con el hardware y si por algún motivo se comete un pequeño error que pondría en riesgo la vida de dicho minicomputador, no resultaría tan costoso el daño en comparación si se usara un computador tradicional ya que estos Raspberry Pi son de precios accesibles.

2.5. SOC

Son las siglas de system-on-a-chip (SOC) lo que quiere decir que dentro de un mismo integrado se incluyen las principales partes de un sistema, es decir es un integrado que contiene las partes principales de un equipo.

Figura #3 - SOC Broadcom de una Raspberry PI (Raspberrypi, 2014)

2.6. CPU

Son las siglas de central-process-unit (CPU), es la parte fundamental de cualquier sistema, ya que es quien ejecuta los principales procesos mediante la acción de operaciones aritméticas básicas.

Estas es la encargada de ejecutar todas las instrucciones secuenciales de la computadora, es ideal para el manejo de operaciones aritméticas.

2.7. GPU

Son las siglas de graphic-process-unit (GPU), esta parte es la encargada de gestionar el procesamiento de gráficos y operaciones de coma flotante, con el fin de aligerar la carga de trabajo al procesador.

Esta es la encargada de ejecutar todas las instrucciones en paralelo de la computadora, es ideal para el procesamiento de imágenes y vídeo ya que puede ejecutar más instrucciones por segundo debido a su paralelismo.

2.8. Linux

Para que los beneficiarios accedan a todos los servicios que facilita el Raspberry, anterior a esto se necesita el montaje de un sistema operativo Linux, para el caso de esta aplicación se ha seleccionado el SO Raspabian.

Se conoce como Linux al kernel creado a partir de UNIX por Linus Trovalds, este es uno de los ejemplos elementales de software libre. Linux es un Unix libre, en otras palabras, Linux es como el sistema operativo Windows, pero con la diferencia que este tiene propietarios. Linux ha evolucionado por miles de usuarios de computadores a través del mundo y su creador fue un estudiante llamado Linux Torvalds que inicialmente fue realizado como un hobby por un estudiante joven, en la universidad de Helsinki de Finlandia.

- Linux posee bastantes ventajas y aplicaciones.

Es uno de los sistemas operativos más fuertes, estables y eficientes. Es de tareas múltiples y multiusuario. A su vez, existen “miles de programas libres para Linux, acomodados a muy diversos objetivos que se encuentran disponibles en internet para usarlos con GNU/Linux. Además, da el acceso a Internet y fusionar máquinas en red de manera natural. Cualquier aplicación Unix puede usarse bajo Linux. Para Linux existe documentación libre en grandes cantidades, a pesar de que no siempre está traducida y las libertades de copia y modificación permiten usar GNU/Linux para facilitar servicios sin depender de terceros.” (Brito, 2015).

2.9. Kernel

Es la parte fundamental de un sistema operativo, se ejecuta en modo privilegiado, es decir realiza instrucciones de bajo nivel.

Su principal trabajo es facilitar y regular el acceso al hardware de la computadora ya que el acceso al hardware suele sr muy complejo, el kernel trae una serie de abstracciones para los desarrolladores.

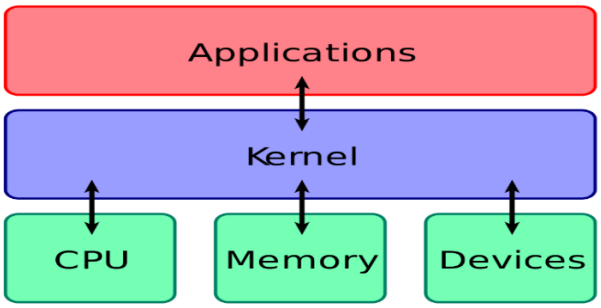

Figura #4 - Ubicación del Kenel en la arquitectura de una computadora

2.10. Raspbian

Es un sistema operativo basado en Debian creada con el objetivo de hacer interactuar el hardware de una Raspberry PI, es netamente puro que además incluye paquetes, programas recompilados, que de esta forma facilita la instalación en el Raspberry PI. (UBUNTU, 2014).

2.11. Debian

Es uno de los suministros más antiguos y estables de Linux que existe, de ella se derivan sientas de otras distribuciones, como por ejemplo Ubuntu, se caracteriza por su estabilidad y practicidad, además que es gratis, no cuenta con licencia para los usuarios.

2.12. Sistema Operativo

Son aquellos sistemas que conservan a uno o más hadware unidos a través de algún medio de información (físico o no) con el propósito elemental de poder compartir los diferentes recursos hardware y software.

Es el programa informático encargado de gestionar los recursos de hardware además de proveer servicios al restante conjunto de software que se ejecuta sobre si, brinda otra capa de abstracción lo que facilita el diseño de aplicaciones de alto nivel.

Figura #5 - Diagrama de bloques de la arquitectura de una PC

2.13. Arduino

El Arduino es una de las plataformas de hardware de código abierto, el cual está basado en una placa de circuito impreso, en la cual se incorpora un microcontrolador de la marca denominada “ATMEL”, donde esta contiene entradas y salidas de forma analógica y digitales, en torno al proceso que está basado el lenguaje de programación processing, este mecanismo permite la conexión al mundo virtual, o al mundo analógico con el digital analógico, caracterizado por sensores, alarmas, sistemas de luces, sistemas de comunicación y actuadores físicos.

Arduino puede medir el entorno a través del ingreso de entradas desde una gran cantidad de sensores y afecta a lo que lo rodea por medio de control de iluminación y otros artefactos. “Se programa el microcontrolador de la placa, usando el Arduino Programming Language (basado en Wiring1) y el Arduino Development Environment (basado en Processing2). Los proyectos de arduino son de dos tipos: autónomos o los que se comunican con software en ejecución a un computador” (Hidalgo, 2015).

Existen variedades de microcontroladores y plataformas disponibles para la computación física en donde las herramientas y funciones resultan ser de índice complicado dentro de la programación de arduino, simplificando así los procesos de trabajar con los microcontroladores, ofreciendo así las ventajas y sus características en cada uno de los sistemas.

Figura #6 - Tarjeta Arduino UNO (Store Arduino)

Ante todo, es un ordenador completo en el cual se integra en un chip, un CPU, la memoria de programas y datos a utilizar, junto a cada uno de los circuitos de controles periféricos, donde el microcontrolador requiere de algunos complementos y auxiliares para tener un correcto funcionamiento entre los cuales se tiene:

- Entrada de alimentación.
- Oscilador de trabajo.

- Circuito de RESET.
- Conexión USB.
- Accesos a las líneas de entrada y salida.

“También consta de un simple, pero completo, entorno de desarrollo, que nos autoriza interactuar con la plataforma de manera muy facil, se puede definir por tanto como un sencillo instrumento de contribución a la creación de prototipos, entornos, u objetos interactivos destinados a proyectos multidisciplinarios y multitecnología.” (Coronel, 2014).

Figura #7 - Placa de arduino con cada una de las especificaciones del Arduino Mega 2560 (UGE-ONE, 2016)

Estas placas son las que ayudan a capacitar cada una de las incorporaciones, en donde se contiene la matriz de cada uno de los terminales, en donde se añade el hardware en consideración a los requerimientos de los prototipos a desarrollarse.

2.14. Servidor web

El servidor web es denominado como el corazón de internet, en donde los principales dispositivos importantes se relacionen como ordenadores, así como los servidores o máquinas dedicadas, para así estar conectada mediante internet, dando disponibilidad a los sitios web, cuando estos son solicitados por usuarios en la red, dichos ordenadores o máquinas son conocidos como servidores web.

“Un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario, este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP, este se puede utilizar para varias tecnologías en el servidor para aumentar así la potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts CGI, seguridad SSL y páginas activas del servidor (ASP).” (Silva, 2014).

Figura #8 - Servidor Web (Barrios, 2011)

Muchos de los equipos son aceptados por medio de peticiones “http” del navegador web del usuario, entregando así las páginas web para poder así ver cuáles son las paginas HTML en donde se enlazará las imágenes y ficheros, en otras palabras, cuando alguien desea entrar a una página web en cuestión pinchado en un enlace, estos ordenadores lanzan peticiones al servidor web donde dicha página está alojada.

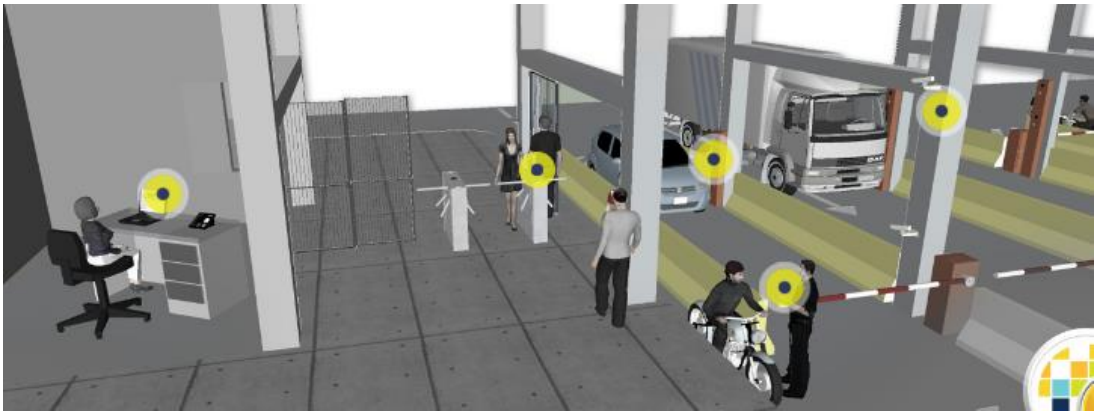

Figura #9 - Sistema de monitoreo y control (Speraccess, 2015)

Gracias al modelo de expansión es compatible con la mayoría de las placas Arduino, no sólo podemos enviar la ubicación del sistema a través de HTTP y guardarla en un servidor web, también permite utilizar Google Maps para observar su ubicación sobre el mapa en todo momento, al igual que todas las placas de este estilo, la antena debe adquirirse por separado, en esta ocasión dependeriamos de una segunda antena específica para GPS. (Rodríguez, 2014)

Muchas de las computadoras de los hogares resultan ser perfectamente servidores web, dado que estos están conectados a internet, sin embargo, estos existen a diferencia de servidores contratados en una compañía y tener un servidor local, por consiguiente, al estar ellos alojados las páginas que se alojan en servidores locales, suelen significar que uno mismo llegue a gestionar entre los niveles técnicos, sobre cada uno de los accesos como las conexiones a las redes, comprobando así que funciona de forma adecuada.

2.15. HTTP

El HTTP de acuerdo a sus siglas representa el Hyper Text Transfer Protocol", en donde para muchos usuarios representan poco, así como la diferencia primordial que existe entre estas, donde día a día se trabaja sobre el internet y el tema de su seguridad es algo que puede pasar de forma desapercibida, dado que se está preocupando por otras cuestiones que en la mayoría de las ocasiones se pasa por alto dichos temas.

La idea principal del HTTP es el poder admitir la transferencia de archivos, que se encuentre en formato HTML, entre un navegador, es decir un servidor web y usuario, que por cuestión se lo denomina al servidos, donde este es ubicado mediante una cadena de caracteres denominada dirección URL (Vicenteño, 2016).

Muchas de las actividades que se ejecutan sobre la plataforma de web se relacionan desde consultas del correo electrónico, la banca en internet, redes sociales, comercios y demás, en donde por medio de estos procesos se intercambia informaciones personales y que deberían de ser confidenciales, dado que la mayoría de estas son transmitidas por medio de mecanismos de naturaleza inseguro, siendo este el punto a tratar.

El servidor http es un programa que el servidor corre, el mismo que escucha lo que solicitan las HTTP, y las realiza. De acuerdo del tipo de solicitud, el servidor web busca una página web o ejecuta un programa del servidor. De todas formas, siempre va a entregar un resultado HTML al cliente o a quien haya navegado solicitando. Este servidor es fundamental en la realización de aplicaciones de lado del servidor, debido a que se va a ejecutar sobre él.

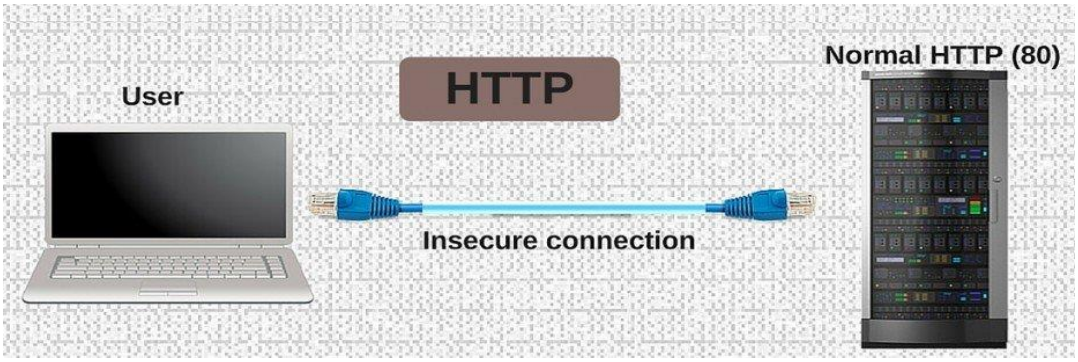

Figura #10 - Conexión HTTP (Ciencia ABC, 2016)

“El funcionamiento esquemático de HTTP es el siguiente; el cliente establece una conexión TCP hacia el servidor, hacia el puerto HTTP (o el indicado en la conexión), envía un comando HTTP de petición de un recurso (junto con algunas cabeceras informativas) y por la misma conexión el servidor responde con los datos solicitados, así como algunas cabeceras informativas” (Castillo, 2012).

El HTTP presenta acciones que cumplen perfectamente la tarea de seguridad y confidencialidad, sin embargo, muchas veces pasa por alto las cuestiones y aspectos de seguridad, es por ello en ocasiones se han formulado interrogantes como ¿los datos que se envian por medio del ciberespacio son seguros? o se ¿alguien posee acceso a la manipulación de la información que manejo?, para lo cual la respuestas a dichas interrogantes suelen ser que al hacer uso del HTTP estos corren los siguientes riesgos, la intercepción y modificaciones por medio de virus informáticos o las suplantaciones de intensidades conocidas como Phishing, entro muchas más.

2.16. HTTPS

El HTTPS es uno de los protocolos de Transferencia de Hiper-Texto, al cual es una versión segura del HTTP, es decir, del Hyper Text Transfer Protocool, del cual se conoce y es utilizado de forma habitual, es por ello que a diferencia del HTTP se puede desarrollar que las actividades de comercio electrónico se permite las realizaciones de transferencias de forma segura.

No es laborioso nombrar algunas razones por las que es recomendable usar HTTPS antes que HTTP. Al cifrar la comunicación entre usuario y servidor, evitamos los ataques man-in-the-middle o eavesdropping, por medio de los cuales un intruso podría conseguir, por ejemplo, el usuario y contraseña de un sitio web porque irían en texto plano (Pérez, 2017).

Es por ello que el este tipo de protocolo de transferencia de hipertexto es seguro dado que la comunicación de internet se protege de acuerdo a la moralidad y la confidencialidad de cada uno de los datos de los usuarios en donde entre sus ordenadores y los sitios web permiten que sus búsquedas y procesos sean manejados de forma privada, dado a que los clientes aspiran que su experiencia online sea adecuada, independientemente de lo que este contenga.

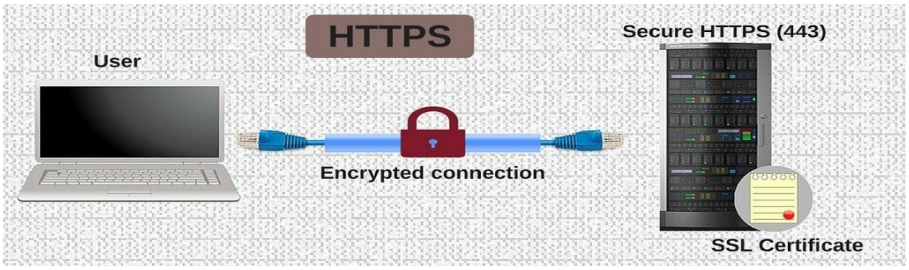

Figura #11 - Conexión HTTPS (Ciencia ABC, 2016)

Hablando rigurosamente, el “HTTPS no es un protocolo separado, lo cual se refiere al uso del HTTP ordinario sobre una capa de conexión segura cifrada: Secure Sockets Layer (SSL) o una conexión con Seguridad de la Capa de Transporte (TLS).” Sánchez, 2014)

Se puede lograr distinguir que la comunicación o las páginas web segura, se debe identificar que el URL debe de comenzar con “https://”, el cual es empleado por el puerto numero 443 por defecto, mientras que el tradicional “http://”, está siendo empleado por 80 puertos por defecto. Es por ello que originalmente el HTTPS solo es utilizable en la encriptación SSL, para poder ser reemplazado por el TLS, para lo cual el HTTPS es adoptado como estándares web por grupos como IETF

2.17. PHP

Es un lenguaje de código abierto que se usa del lado del servidor para generar páginas web dinámicas, este lenguaje es el más usado del lado del servidor y permite fácil comunicación con todo tipo de bases de datos.

Este es un acrónimo recursivo, el cual tiene como significado “Hypertext Pre-processor”, la cual esta publicado bajo la PHP License, considerado, así como un software libre, dado que el PHP es un lenguaje interpretado como propósito general usado ampliamente, en donde se usa y se diseña especialmente para un mejor desarrollo Web, al poder ser embebido dentro del código HTML

Por lo general esto es ejecutable dentro de un servidor Web, tomando como entrada el código en PHP y así crear páginas web como salida, en donde se despliega y en casi todos los sistemas operativos y plataformas sin ningún costo la mayoría de los servidores web.

Los usos principales del PHP son los mencionados a continuación:

“Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base de datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión” (García J., 2014).

Entre las principales ventajas se encuentran las siguientes:

- Lenguaje multiplataforma
- Capacidades de conexión con la mayoría de manejadores que están inscritos en la base de datos utilizando en la actualidad, destacando así su conectividad con el MySQL.
- Es de libre ingreso, por lo cual se expone como una de las alternativas de fácil comunicación para los que deseen utilizarlos.
- Este permite que las técnicas de programación que está orientada en objetos, también permitan que las conexiones respecto a los diferentes tipos de servidores de bases de datos tales como: MySQL, ODBC, DB2, Microsoft SQL Server, Firebird y el SQLite.

2.18. Framework web

Es una estructura definida para el diseño de software, trae patrones de diseño los cuales definen en forma general como se debe desarrollar software y como este estará estructurado, incorpora una serie de librerías y componentes los cuales están predefinidos y realizan tareas comunes al área en la que se aplica el framework.

Las ventas de usar un framework son:

- Evita escribir código repetitivo

- Implementa buenas prácticas de diseño de software
- Permite crear más y mejores cosas en un tiempo menor

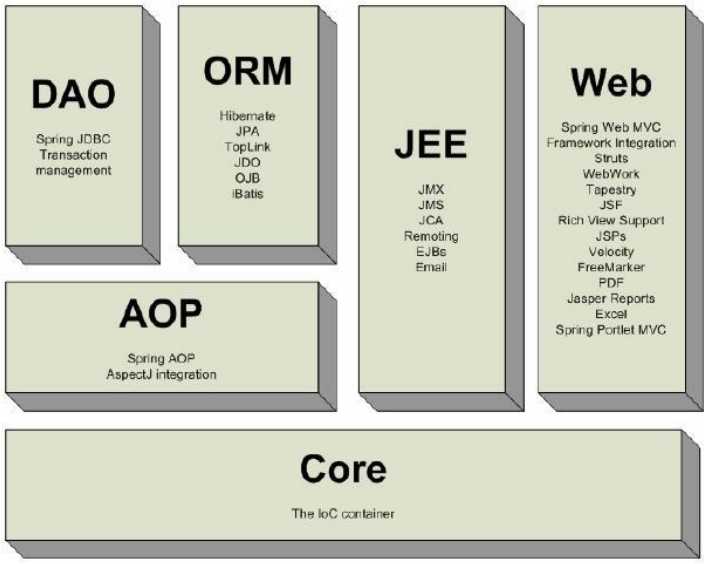

Figura #12 - Estructura del framework spring de JAVA

2.19. Laravel

Laravel es un framework web escrito en PHP el cual destaca por su potencia y simpleza a la hora de desarrollar aplicaciones web.

Laravel busca ofrecer una sintaxis de fácil uso facilitando la reutilización de código, aprovecha todas las funcionalidades que brindan otros frameworks y utiliza correctamente las últimas particularidades de PHP.

Algunas de sus principales particularidades son**:**

- Sistema de ruteo, también RESTful
- Blade, Motor de plantillas
- Peticiones Fluent
- Eloquent ORM
- Basado en Composer
- Soporte para el caché
- Soporte para MVC
- Usa componentes de Symfony
- Adopta las especificaciones PSR-2 y PSR-4

Incorpora un potente asistente de línea de comandos el cual permite automatizar tareas repetitivas en las aplicaciones web.

Algunos de los principales comandos que ofrecen son:

- make:command — Generar un nuevo class
- make:console — Genera un nuevo Artisan command
- make:controller — Genera un nuevo resource controller class
- make:event — Genera un nuevo event class
- make:middleware — Genera un nuevo middleware class
- make:migration — Genera un nuevo migration file
- make:model — Genera un nuevo Eloquent model class
- make:provider — Genera un nuevo service provider class
- make:request — Genera un nuevo form request class
- event:generate — Crea una lista con los controladores y eventos perdidos

Laravel dispone de una serie de paquetes los cuales reducen el trabajo a la hora de desarrollar una aplicación web, entre ellos trae controladores de bases de datos, sistema de eventos y su propio gestor de plantillas llamado Blade.

Figura #13 - Logo de Laravel

2.20. Base de datos

La data es un sistema formado por los conjuntos de datos almacenados en discos en donde estos permitan los accesos directo por medio de conjunto de programas en el cual se manipula los conjuntos de datos, siendo estos como base de los cuales se componente de una o más tablas guardando conjunto con cada uno de los datos.

Las data son lugares donde se guarda información de acuerdo a diferentes formas de organización. Una base de datos está representada por varios aspectos del mundo real, especialmente, aquellos que le interesan al usuario. Los datos son hechos que se conocen, y que se pueden registrar (Gutierrez, 2016).

Dado que la base de datos es una colección de datos los cuales se almacenan y se organizan por medio de programas de ordenadores para que estos puedan seleccionarse de forma rápida y capaz de ser recobrados, insertados, actualizados y capaces de ser borrados, dado que son registrados por medio de archivos electrónicos.

Figura #14 - Proceso de almacenamiento de información en la data
(Engloba Systems, 2014)

El propósito primordial de un sistema de administración de data es asignar una manera de guardar y retomar la información que ha estado guardada en una data y que ésta sea práctica y eficiente.

Los SGBD son diseñados para manipular grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para guardar la información como la provisión de mecanismos para la manipulación de la información. (Gómez F. M., 2013)

Las informaciones no son un dato cualquiera, sino más bien son las colecciones respecto a los hechos significativos y pertinentes, en donde le organismo y las organizaciones se perciben, dado que las definiciones de las informaciones son conformadas por medio de conjunto de datos significativos y pertinentes en donde estos describen cada uno de los sucesos o entidades representativas.

Características de la Data

- Posee autosuficiencia tanto lógica como física de los datos
- Tiene reiteración mínima
- Posee acceso recurrente entre cada una de las partes de múltiples usuarios
- Tiene integridad de datos
- Realiza peticiones dificultusas y optimizadas
- Tiene confianza en cada uno de los accesos de auditoria
- Posee respaldo y recuperación
- Inserta información por medio de lenguajes de programación estándares.

2.21. MySQL

MySQL Figura #15, es un gestor de data relacionales, el permite la creación, edición y eliminación de tablas y registros en las bases de datos.

Figura #15 - Logo de MySQL (MySQL)

Está escrito en C/C++ y permite una facilidad de conexión con los principales lenguajes de programación más importantes de la actualidad como PHP, PEARL o JAVA.

MySQL es OpenSource lo que permite modificarlo y adaptarlo a nuestras necesidades con total libertad, esto no ha hecho más que favorecerlos ya que se han desarrollado muchas herramientas para su desarrollo.

2.22. Python

Es un lenguaje de programación de uso conceptual, orientado a objetos que es principalmente usado para software científico y páginas web dinámicas.

Debido a su simplicidad y sintaxis clara como se puede apreciar en la Figura #16 ha ganado una gran popularidad.

```
1   def foobar(foo=None):
2       """Do something foobar.
3
4       This sets myvar to the value of foo and then returns the value
5       and is generally useless.
6       """
7       # if foo is None, say so.
8       if foo is None:
9           print "Foo is None. Even more useless."
10      myvar = foo
11
12      return myvar
```

Figura #16 - Ejemplo de código Python

Las principales características de Python son:

- Lenguaje de propósito general

- Corre en casi cualquier sistema operativo (Multiplataforma)
- Se compila y ejecuta al momento de ejecución (Interpretado)
- Orientado a objetos
- Sintaxis clara
- Gran cantidad de librerías y software libre para reutilizar

2.23. GPS

El GPS se basa en los cerca de 200 satélites que se encuentran orbitando el planeta, estás perfectamente distribuidos de forma que siempre tengamos al menos 3 satélites visibles en todo momento.

Cada uno de estos satélites conoce la hora exacta y cuando un dispositivo les envía un mensaje en base a la diferencia de tiempo entre el que envia y recibe se puede calcular el espacio, al saber a qué distancia la persona se encuentra de los 3 satélites el dispositivo puede triangular fácilmente su posición como se puede observar en la Figura #17.

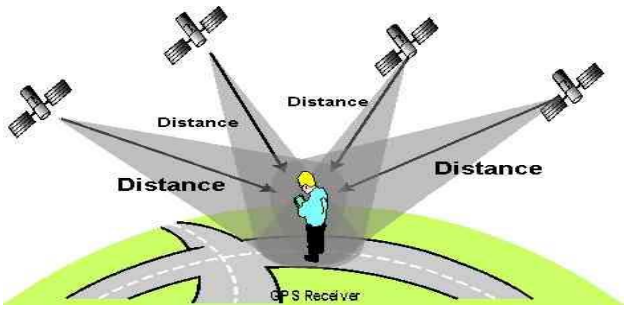

Figura #17 - Funcionamiento del GPS

3. MARCO METODOLÓGICO

3.1. Descripción General

En este capítulo se detallará cada uno de los procedimientos a realizar en la instalación del sistema de monitoreo de velocidad para los transportes urbanos, en donde a su vez se controle cada una de las puertas las cuales serán acondicionadas para trabajar por medio de geolocalización.

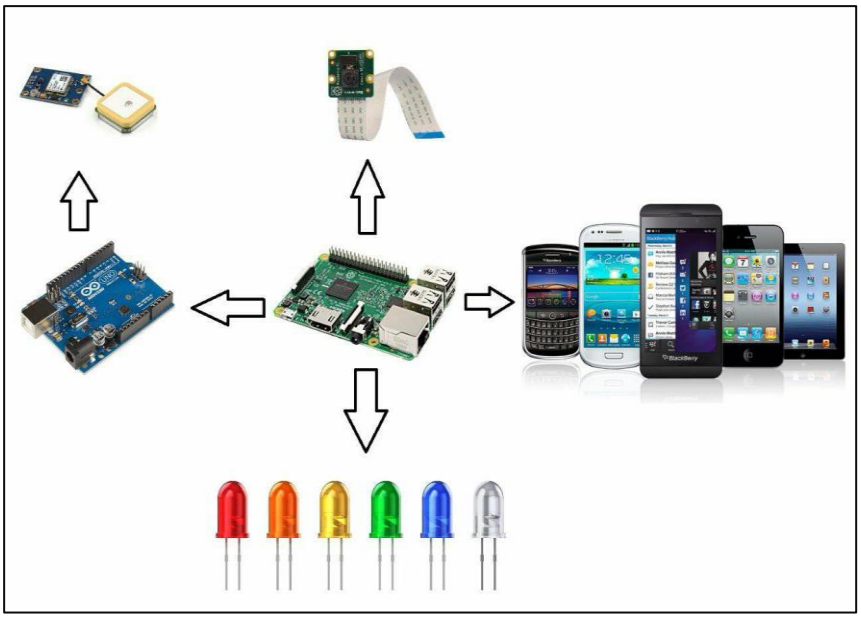

Figura #18 - Esquemático del sistema de monitoreo de velocidad

En la Figura #18 se aprecia el diagrama general del sistema en donde consta como parte principal del prototipo la interconexión entre la Raspberry PI y el Arduino, donde a su vez el arduino lleva conectado un sistema GPS que arrojará la ubicación actual del transporte.

Por otra parte, en la Raspberry PI irá conectado un pequeño circuito de leds indicadores que permitirá tener control de cada elemento conectado a la tarjeta.

Todo este sistema podrá ser accedido por medio de los dispositivos móviles a través de un servidor web.

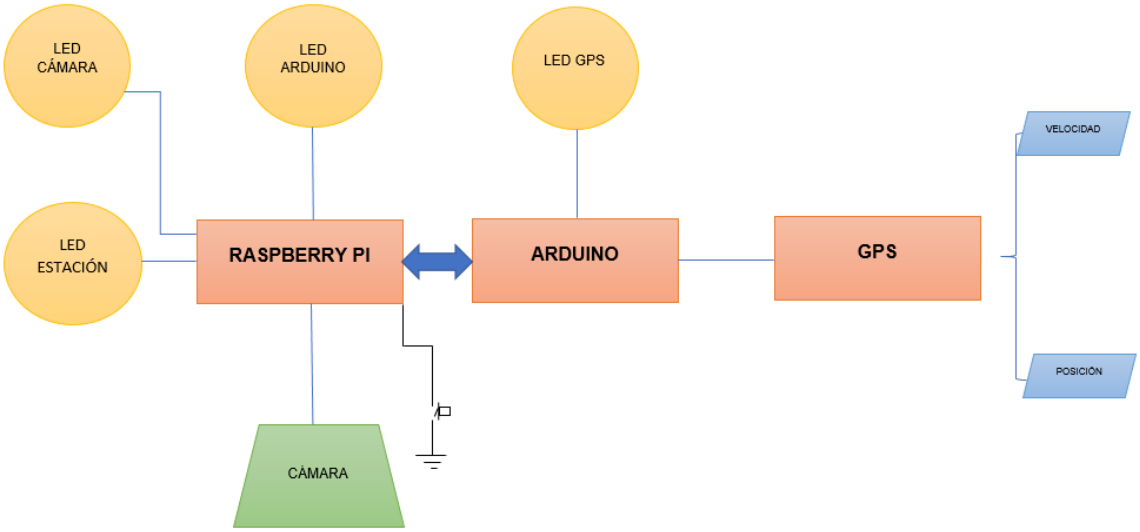

Figura #19 - Diagrama de bloque del sistema

3.2. Módulo GPS

El módulo GPS Figura #20 es un modelo GY-GPS6MV2, el cual viene equipado con una Placa de Circuito Impreso, provista de conectores para la alimentación y transmisión de datos.

Figura #20 - Modulo GPS

El chip EEPROM que permite guardar los datos de configuración cuando se apaga el GPS, indicadores LED de señal y una antena de cerámica.

Este modelo a su vez contiene pines o conectores que pueden ser utilizados para la conexión de algún tipo de microcontrolador por medio de una interfaz serial.

Para que el módulo de GPS funcione correctamente es necesario que se realice varias pruebas en un ambiente despejado o cercano a una ventana, donde esta puede lograr una correcta recepción de señal.

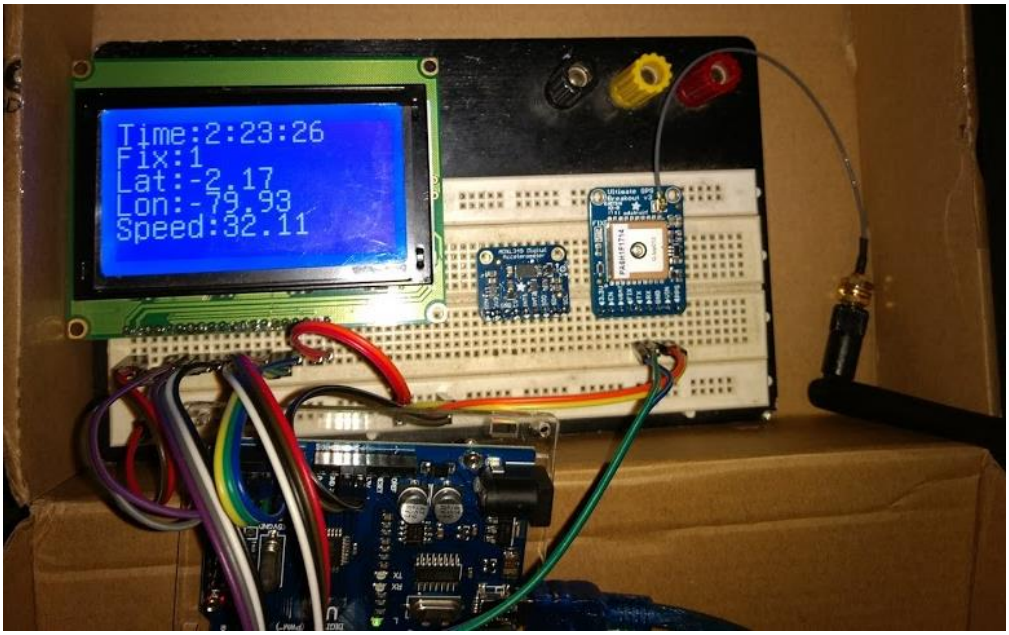

Figura #21 - Pruebas de funcionamiento del GPS en prototipo

Dentro del sistema, mediante el GPS se permitirá obtener parámetros tales como la velocidad y la posición actual del transporte por medio del posicionamiento del satélite, los cuales se irán guardando para un posterior envió, para esto el GPS estará interconectado con el arduino para la lectura de los datos que el GPS estará enviando. A continuación, la Figura #22 detalla la conexión del GPS con el arduino.

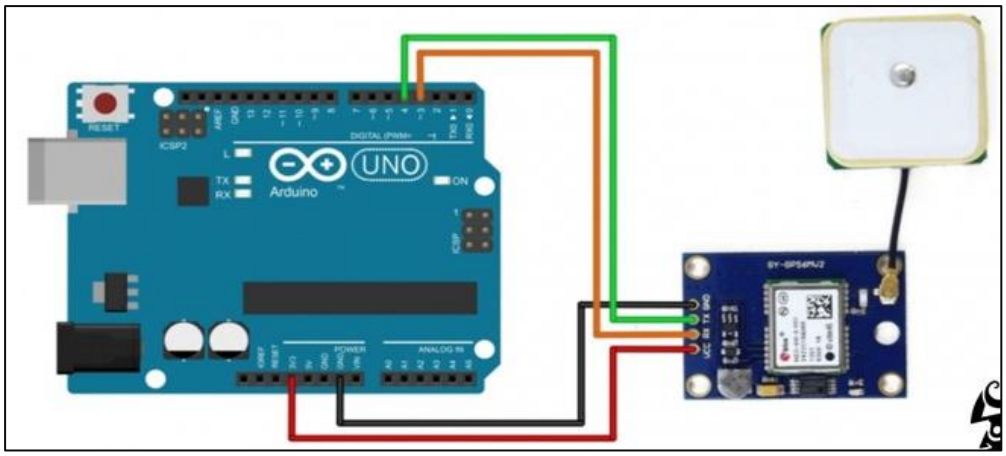

Figura #22 - Conexión del arduino con el GPS (Naylam, 2018)

El módulo GPS estará conectado al arduino mediante un puerto de serie que se emulará por software usando conectores RX y TX, los cuales permitirán comunicarse con la PC y verificar cada uno de los datos recibidos por el GPS. Los pasos para conectar el arduino con el GPS serán los siguientes (Naylamp, 2018):

- El pin 3.3V del Arduino UNO será conectado al pin Vcc del módulo GPS.

- El pin GND del Arduino UNO será conectado al pin GND del módulo GPS.
- El pin digital 4 del Arduino UNO será conectado al pin Tx del módulo GPS.
- EL pin digital 3 del Arduino UNO será conectado al pin Rx del módulo GPS.

3.3. Validación de la parada del bus

Debido a las imprecisiones propias del GPS, ya que su rango de exactitud puede llegar a metros, es necesario que las paradas tengan un rango, es decir no son un punto exacto en el mapa, en su lugar es un área circular de un radio determinado.

Se dio un pequeño rango en el GPS que de manera experimental se estimó que es de 10 metros alrededor del punto deseado como se puede observar en la Figura #23.

Figura #23 - Rango de seguridad del GPS

3.4. Captura de fotos

EL sistema debe capturar de forma automática fotos siempre que la puerta esté abierta, es decir cuando el sensor de las puertas detecte que las puertas estén abiertas independientemente de si el conductor del bus las abrió o no, se tomará una ráfaga de fotos con la cámara de la Raspberry Pi que se puede observar en la Figura #24.

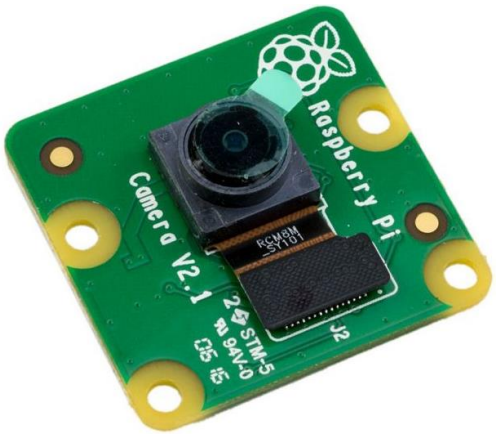

Figura #24 - Cámara de la Raspberry Pi (Amazon)

Para lograr lo anteriormente mencionado se implementó el diagrama de flujo de la Figura #25 en código Python.

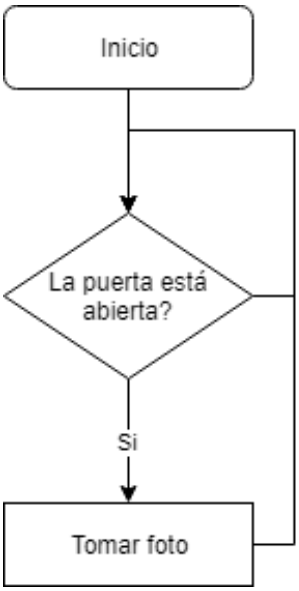

Figura #25 - Diagrama de flujo de captura de fotos

La cámara de la Raspberry Pi dispone de 5Mpx de resolución y soporta zoom, pero no dispone de visión nocturna por lo que es necesario que se disponga de una adecuada iluminación en el interior del bus de transporte urbano.

En la Figura #26 se puede apreciar una foto del interior del bus de transporte urbano, la foto fue tomada con la cámara de la Raspberry Pi.

Figura #26 - Foto del interior del bus de transporte urbano

3.5. Base de datos

Es el instrumento que va a almacenar todos los datos de la aplicación de forma permanente, esta permite simplificar el proceso de almacenamiento y validación de datos ya que dispone de su propio mecanismo interno de validación, con el fin de almacenar los datos de forma organizada se diseñaron 2 tablas principales donde se almacenará la información de las estaciones permitidas, así como de los usuarios con acceso a la aplicación.

3.5.1. Tabla de estaciones permitidas

Para el almacenamiento de los datos de las paradas es necesario disponer de una tabla que tenga los campos descritos en la Tabla 1 - Tabla de estaciones permitidas.

Tabla 1 - Tabla de estaciones permitidas

Nombre	Tipo	Atributos	Nulo	Extra
id	int(10)	Sin Signo	No	Auto incrementable
name	text		No	
longitude	decimal(10,6)		No	
latitude	decimal(10,6)		No	
deleted_at	timestamp		Yes	
created_at	timestamp		Yes	
updated_at	timestamp		Yes	

En la tabla se almacenarán tanto longitud como latitud de las paradas donde el bus podrá abrir sus puertas, cada una de estas paradas tendrá un nombre y junto con ellas se guardarán ciertos campos de auditorías más como las fechas de creación, edición y eliminación de la parada.

3.5.2. Tabla de usuarios registrados

Además de las paradas permitidas se debe tener una tabla para almacenar los usuarios con acceso a la aplicación, el formato de la tabla se describe en la Tabla 2 - Tabla de usuarios.

Tabla 2 - Tabla de usuarios

Nombre	Tipo	Atributos	Nulo	Extra
id	int(10)	Sin Signo	No	Auto Incrementable
name	varchar(191)		No	
email	varchar(191)		No	
password	varchar(191)		No	
remember_token	varchar(100)		Yes	
created_at	timestamp		Yes	
updated_at	timestamp		Yes	
updated_at	timestamp		Yes	

3.6. Interfaz web

Para poder agregar y eliminar las paradas permitidas es necesario que exista una base de datos donde se almacenen las mismas, por este motivo se utilizó MySQL es el gestor del data que almacenará todos los registros.

Pero para que los usuarios finales puedan agregar, editar y eliminar paradas es necesario disponer de alguna interfaz, para este fin se optó por utilizar una interfaz web.

3.6.1. Interfaz de inicio de sesión

Como las únicas ubicaciones donde el bus podrá abrir sus puertas están definidas en la data es importante que no cualquiera pueda modificar dichas paradas, por eso para poder acceder a la interfaz de administración de las paradas es necesario pasar primero por un inicio de sesión como seobserva en la Figura #27 - Interfaz de inicio de sesión.

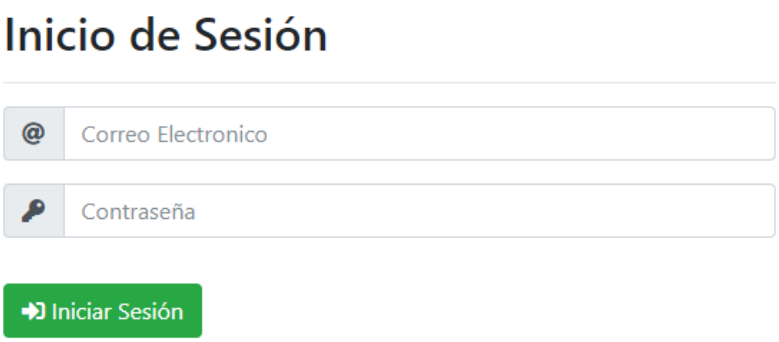

Figura #27 - Interfaz de inicio de sesión

Esta interfaz pedirá los datos al usuario administrador y validará si este tiene acceso o no las paradas.

En caso de que se ingresen las credenciales incorrectas, se desplegará un mensaje de error como se aprecia en la Figura #28 - Mensaje de error de inicio de sesió.

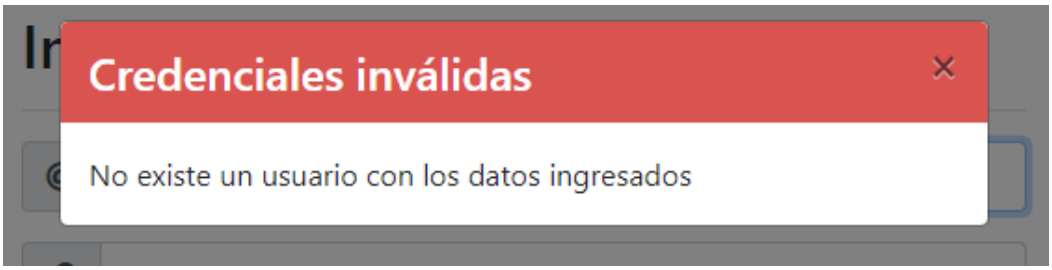

Figura #28 - Mensaje de error de inicio de sesión

3.6.2. Interfaz de administración

La interfaz principal que se puede apreciar en la Figura #29 dará la opción de administrar (crear, editar y eliminar) las paradas existentes, cambiar las credenciales del usuario actual y ver las fotos que se han tomado en el interior del bus.

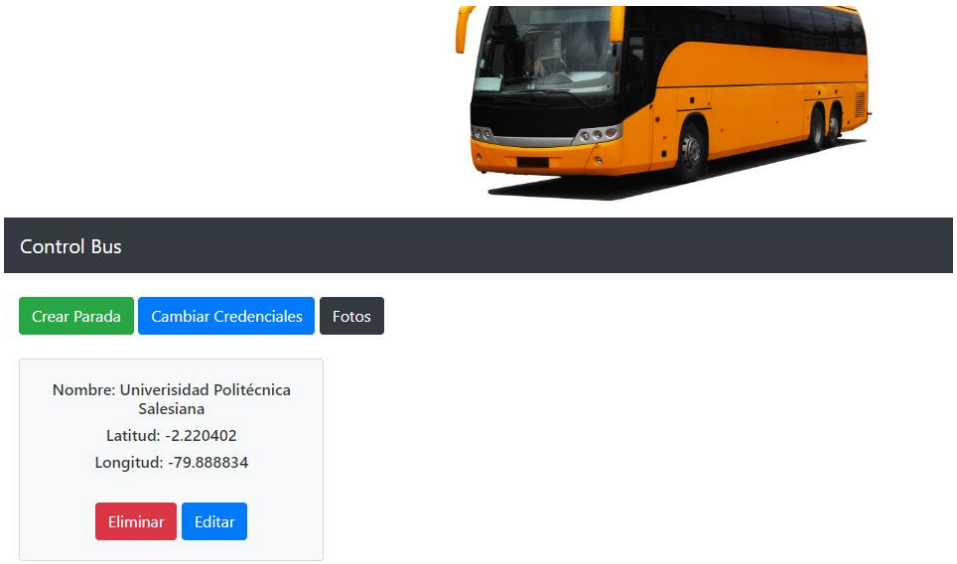

Figura #29 - Interfaz de administración

Esta interfaz es una interfaz web, lo que significa que puede verse desde cualquier dispositivo el cual tenga instalado un navegador web, esto incluye teléfonoo celulares, tablets e incluso cualquier dispositivo inteligente como televisores y hasta relojes, sin que el desarrollador de un soporte especial.

El programa se adapta a cualquier pantalla permitiendo que se pueda hacer un uso cómodo de la aplicación desde cualquier dispositivo, en la Figura #30 se puede apreciar la interfaz del sistema visto desde un dispositivo móvil.

Figura #30 - Interfaz web vista desde una dispositivo móvil

3.6.3. Interfaz de creación de paradas

Una de las principales funcionalidades del sistema es que cualquier usuario que no tenga experiencia en programación ni en bases de datos, sea capaz de gestionar él mismo las paradas permitidas del bus.

Para esto es necesario que dentro de la interfaz de administración exista la funcionalidad de agregar una nueva parada, en la cual el usuario solamente deba brindar la información de latitud, longitud y el nombre de la parada y que el sistema web sea el que procese, valide y registre los datos en la data tal como se define en la Figura #31.

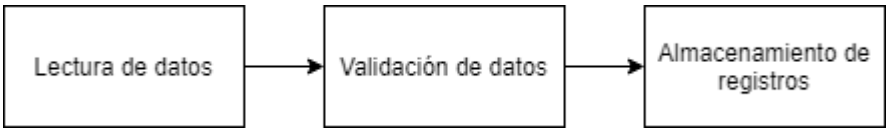

Figura #31 - Pasos para la creación de registros

La lectura de datos consiste en adquirir los datos del usuario mediante la interfaz web, es decir se muestra un formulario el cual llenará el usuario con la información necesaria para crear una nueva parada tal como se ve en la Figura #32, este formulario recogerá los datos y los enviará a la siguiente fase de validación

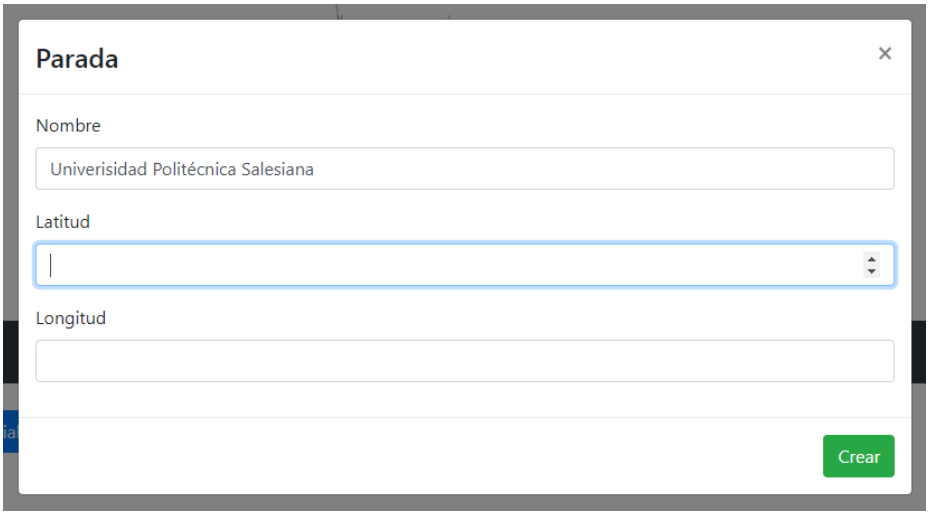

Figura #32 - Formulario de creación de paradas

El proceso de validación de datos consiste en verificar que no existan nombres de paradas repetidas, ya que cuando se almacenan grandes cantidades de paradas es muy común olvidar cuales ya fueron registradas y cuáles no, para eso es ideal que el sistema sea apto de validar la existencia de dichas paradas antes de registrarlas, se validará tanto el nombre como la latitud y longitud de las mismas.

Una vez los datos sean correctamente validados estos serán almacenados en la base de datos previamente definida y se desplegará un mensaje de afirmacion de los datos como se puede observar en la Figura #33.

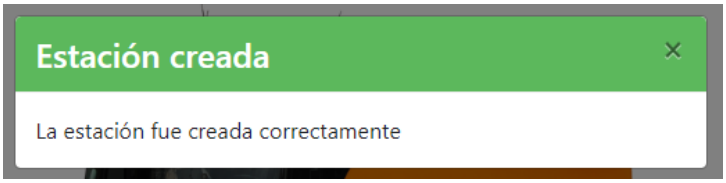

Figura #33 - Mensaje de confirmación

Todas las estaciones creadas se desplegarán en la interfaz principal para poder editarlas o eliminarlas de ser necesario.

3.6.4. Interfaz de eliminación de estaciones

Cuando hay una estación en la que el bus ya no pueda detenerse tanto por cambios de recorrido como por reducción de paradas, estas paradas deben ser eliminadas de la aplicación para que el sistema bloquee la apertura de las puertas.

Para eso cada parada dispone de un botón de eliminar el cual al darle clic mostrará un mensaje de confirmación como se observa en la Figura #34, para evitar que se borren paradas por clics accidentales.

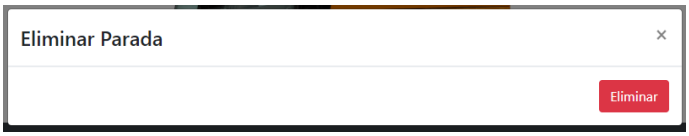

Figura #34 - Mensaje de eliminación

3.6.5. Interfaz de cambio de credenciales

Todo acceso restringido por contraseña es susceptible de ataques de diccionario los cuales buscan descubrir una contraseña mediante la prueba e intento de una serie de contraseñas (diccionario) hasta dar con la contraseña válida.

Existen múltiples alternativas para mitigar un ataque de contraseña, pero la más efectiva y recomendable consiste en el cambio de credenciales cada determinado periodo de tiempo, por eso el sistema que hemos diseñado dispone de una interfaz para el cambio de credenciales.

En la Figura #35 se puede apreciar cómo se solicita la nueva contraseña dos veces con el fin de validar que se la haya escrito correctamente antes de aplicar el cambio, cabe resaltar que esta interfaz para el cambio de contraseña solo está disponible

para usuarios autenticados en la aplicación, es decir se debe haber iniciado sesión previamente para poder cambiar la contraseña de la sesión actual.

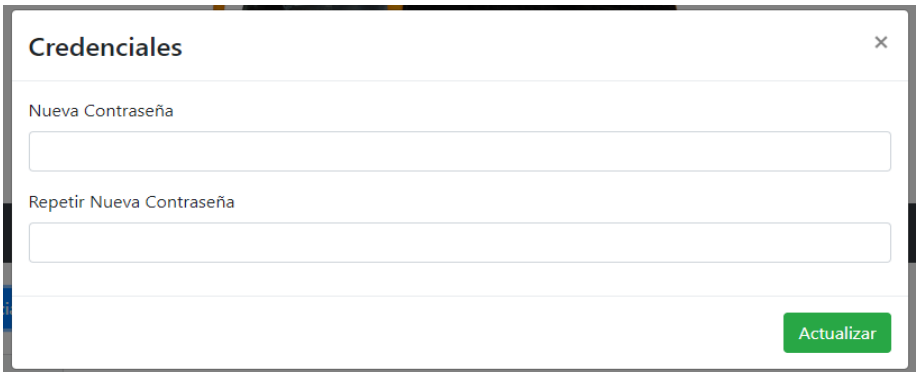

Figura #35 - Interfaz de cambio de credenciales

3.7. Servidor web

El sistema se basa en que la Raspberry Pi funcione como servidor web para que cualquier pueda conectarse a ella, para eso se instaló el software apache, el cual es el servidor web más usado del mundo.

El servidor web solamente brinda acceso a archivos estáticos, es decir a páginas previamente definidas, imágenes, etc. como se observa en la Figura #36.

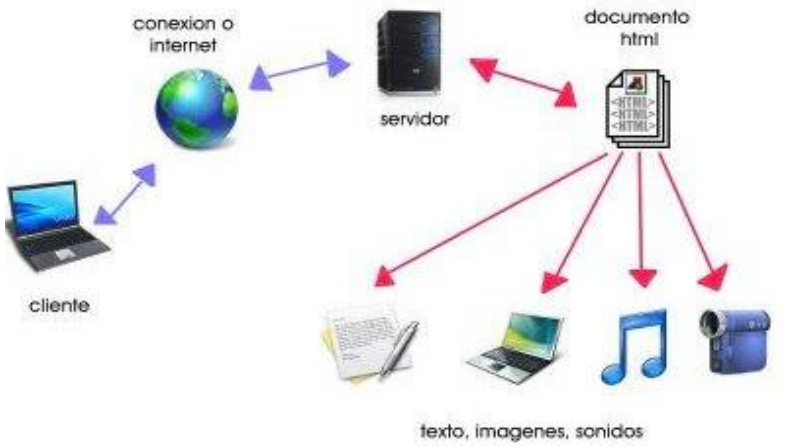

Figura #36 - Servidor web de archivos estáticos

Se desea generar una página diferente en base a los datos que están almacenados en el gestor de bases de datos, para eso es necesario que con la petición web se ejecute un código PHP el cual se conecte a la base de datos y genere una respuesta dinámica como se observa en la Figura #37.

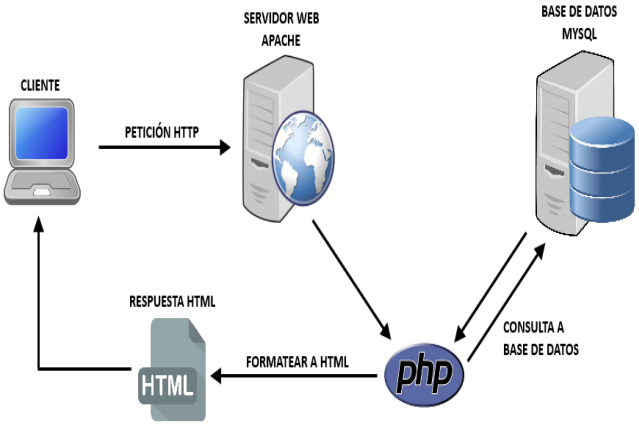

Figura #37 - Generación de contenido dinámico

La petición viaja desde el cliente hasta el servidor web (apache) el cual delega PHP el manejo de la misma, PHP consulta en la data MySQL para determinar que contenido se debe mostrar en la página web y genera una página HTML personalizada para cada consulta.

En la Figura #38 se observa el diagrama de bloques del servidor web que está configurado en la Raspbery Pi.

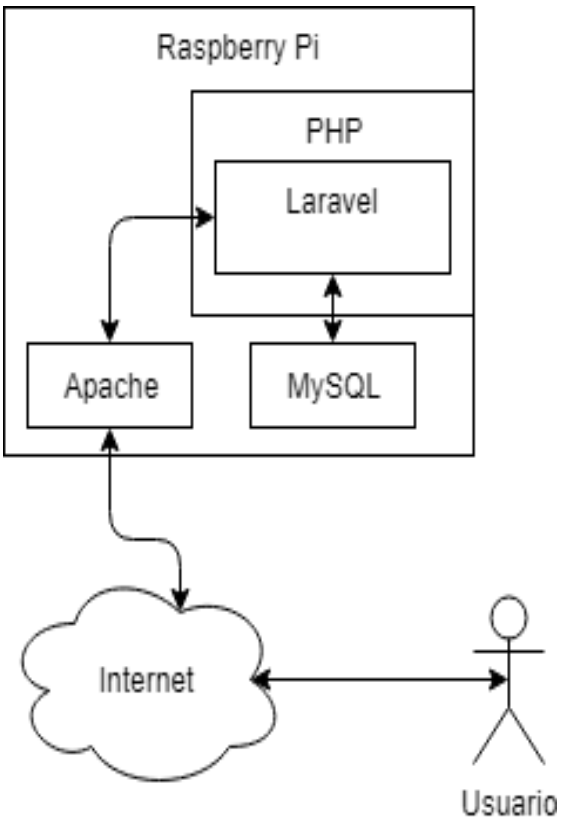

Figura #38 - Esquema del servidor web

3.8. Supervisión remota

Para llevar un mejor control de los buses, se creó un sistema de supervisión remota, que consiste un sitio web el cual muestra la posición del bus dentro del mapa, mediante una interfaz web.

La interfaz web permitirá ver la ubicación en tiempo real del bus, así como su velocidad, esta interfaz solamente mostrará un mapa para poder visualizar los datos, para esto se utilizó lo servicios de Google Maps, como se puede observa en la Figura #39.

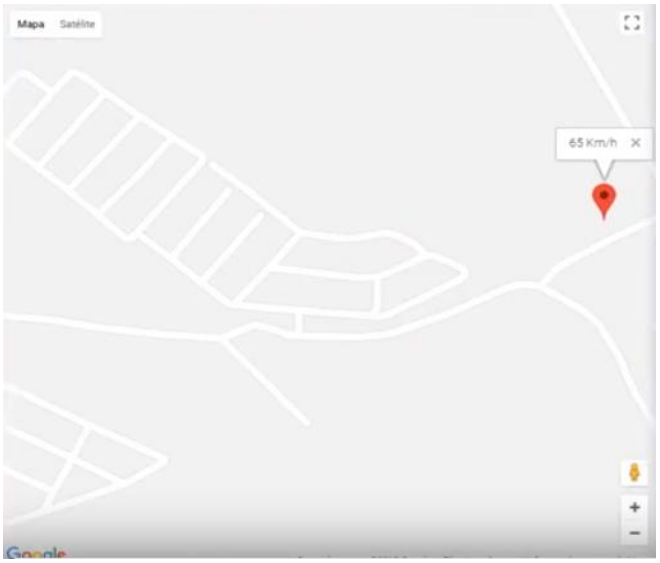

Figura #39 - Interfaz de geolocalización del bus

4. ANÁLISIS DE RESULTADOS

En el presente proyecto de titulación se propuso como objetivo el diseño de un sistema de control basado en geolocalización que se integre con el sistema de transporte actual para evitar que estos abran y cierren las puertas en zonas no autorizadas disminuyendo la congestión vehicular, los resultados obtenidos de la elaboración del proyecto se detallan acontinuación.

4.1. Diagrama de bloques del biométrico

El diagrama de bloque del esquemático es el de la Figura #40 - Diagrama de bloques del proyecto.

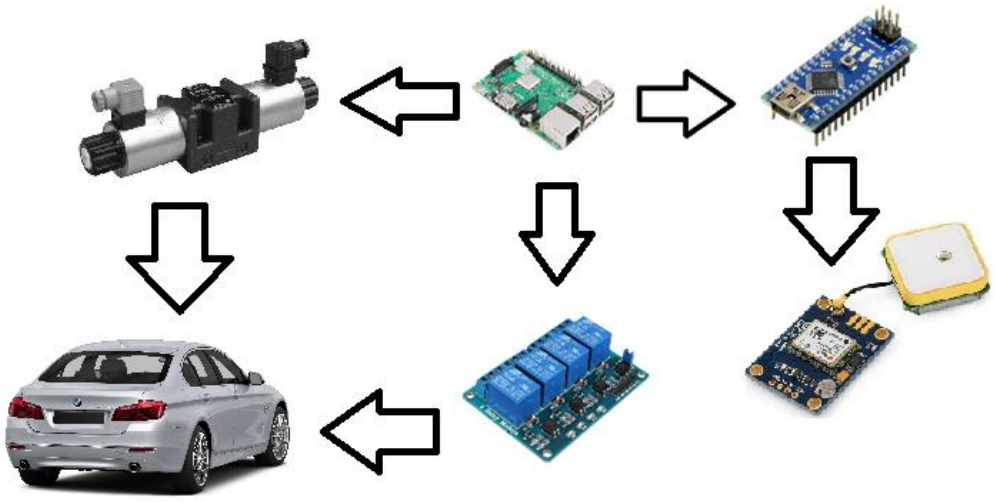

Figura #40 - Diagrama de bloques del proyecto

Para poder determinar la posición del carro en todo momento, es preciso adecuar un dispositivo GPS el cual entregará la información al Arduino Nano el cual procesará y validará realizando varias peticiones de geolocalización para poder descartar las peticiones erróneas reduciendo el margen de error del sistema de geolocalización.

Los datos de geolocalización validados serán envaidos a la Raspberry Pi desde el arduino mediante comunicación serie, la Raspberry consultará su base de datos de paradas permitidas Figura #41.

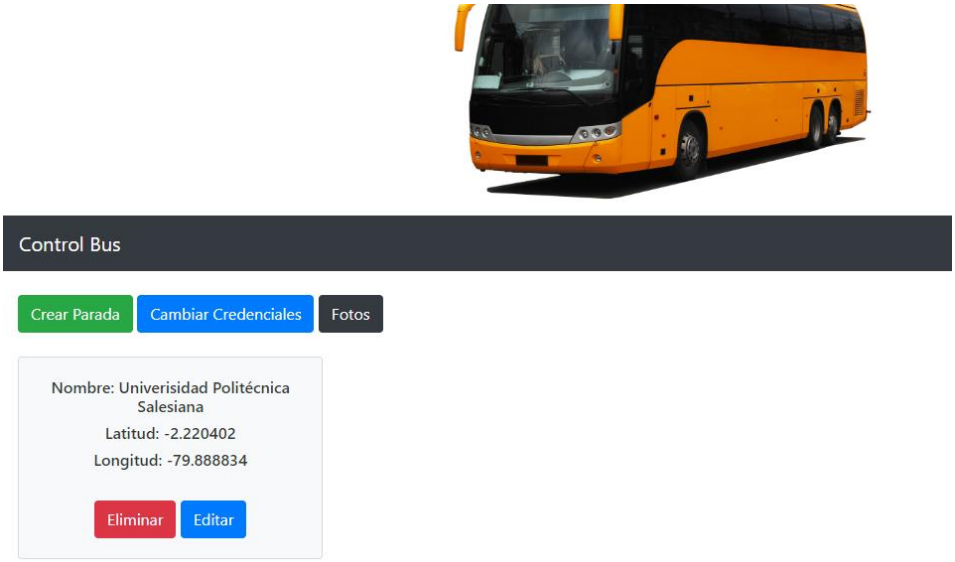

Figura #41 - Interfaz de gestión de paradas

Y si estamos sobre la parada la Raspberry Pi encenderá un relay el controlará una electro-válvula Figura #42 para permitir o no la apertura de la puerta del vehículo.

Figura #42 - Electro-valvula para apertura y cierre

La electro-válvula controlará un cilindro neumático como el que se puede apreciar en la Figura #43, el cuál bloqueará la apertura de la puerta cuando el automóvil este en una zona que no es una parada permitida y permitirá la apertura de la puerta cuando el automóvil se encuentre en una parada permitida.

Figura #43 - Cilindro neumático para control de puertas

Adicional el sistema cuenta con una cámara para la captura de fotos del interior del vehículo cuando las puertas se abra.

4.2. Implementación del proyecto

Se ubicó la Raspberry Pi en el panel frontal del vehículo como se puede apreciar en la Figura #44, con la finalidad de que se puede supervisar en todo momento su estado, mediante los indicadores leds de los que dispone.

Figura #44 - Instalación del sistema de GPS

Es importante que la antena esté lo más expuesta al exterior que sea posible con el fin de mejorar su sensibilidad y ubicarla en el interior del carro podría llegar a suponer problemas de precisión con la lectura del GPS.

El sistema dispone también de una cámara como la que se ve en la Figura #45 mediante la cual se tomará fotos del interior del automóvil cada vez que se abren las puertas, sin importar si el automóvil se encuentra o no en una parada permitida, esto agraga una capa de seguridad al sistema de bloqueo de paradas.

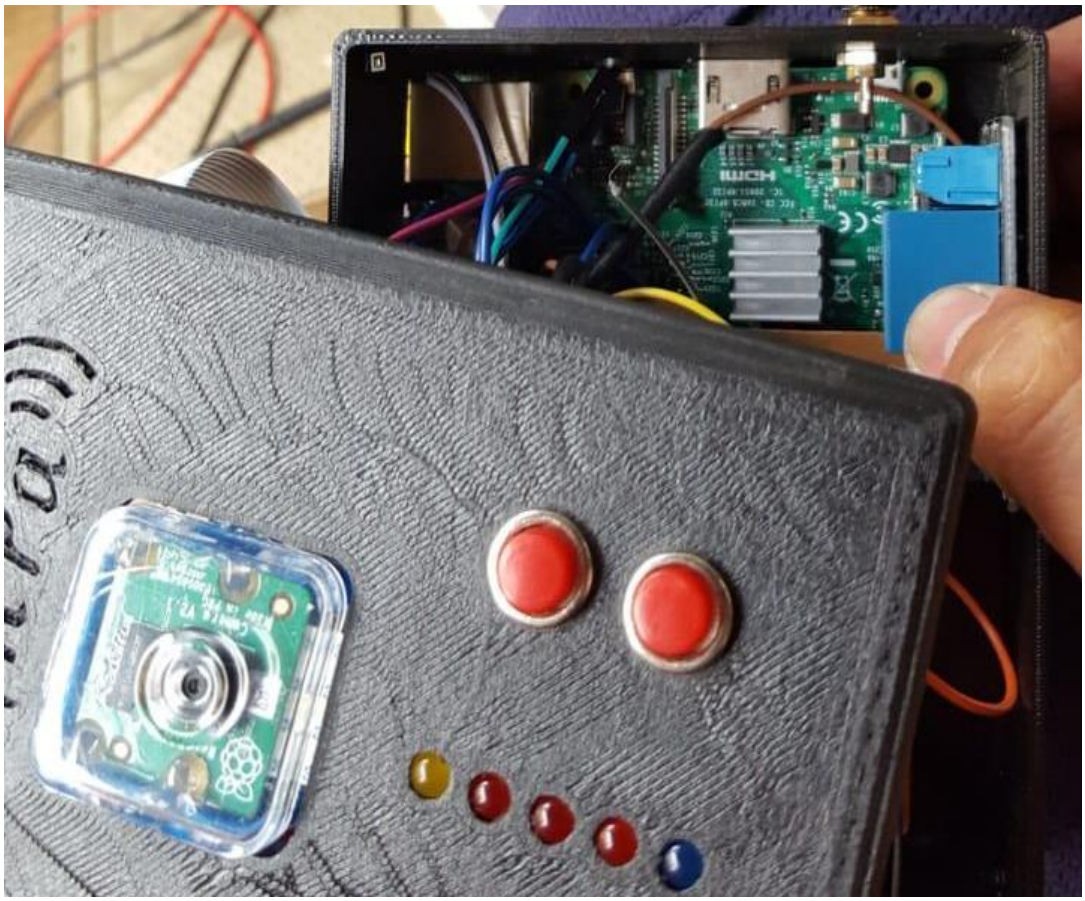

Figura #45 - Cámara del sistema de control

Las fotos tomadas por el sistema son almacenadas en la Raspberry Pi para su posterior recolección y análisis, la Raspberry Pi tiene una tarjeta de memoria de Gb donde puede almacenar las fotos, cuando el disco se llena la Raspberry Pi procede a borrar automáticamente las fotos más antiguas.

Figura #46 - Instalación del sistema

En la Figura #47 como se observa en la electroválvula la cual controla el cilindro neumático el cual controla el bloqueo o desbloqueo de la puerta del automóvil.

Figura #47 - Instalación de la Electro-válvula

Esta electroválvula funciona a 12 voltios DC los cuales son tomados desde la batería del carro la cual se puede observar en la

Figura #48.

Figura #48 - Alimentación del sistema desde la batería del carro

En la Figura #49 se puede apreciar como el cilindro neumático dirige la apertura y cierre de la puerta, la puerta solamente se abrirá cuando el cilindro haya sido activado con la electro-valvula.

Figura #49 - Control de apertura de la puerta con cilindro neumático.

El sistema dispone de dos botones manuales para poder abrir y cerrar las puertas en caso de emergencias, ya que en caso de alguna avería en la Raspberry Pi o de alguno de sus componentes como por ejemplo error en la placa de GPS, el sistema al no ser capaz de detectar correctamente las coordenadas bloqueará las puertas, pero este sistema manual evitará que se produzcan situaciones donde las personas quedan atrapadas en el interior del vehículo.

Figura #50 - Controles manuales para emergencias

4.3. Diagrama de Flujos

El sistema leerá los datos del GPS y si se encuentra en una de las paradas previamente establecidas se desbloquearán las puertas, caso contrario se bloquearán las puertas para que se pueda entrar o salir del automóvil fuera de lasparadas establecidas, tal cual se ve en la Figura #51.

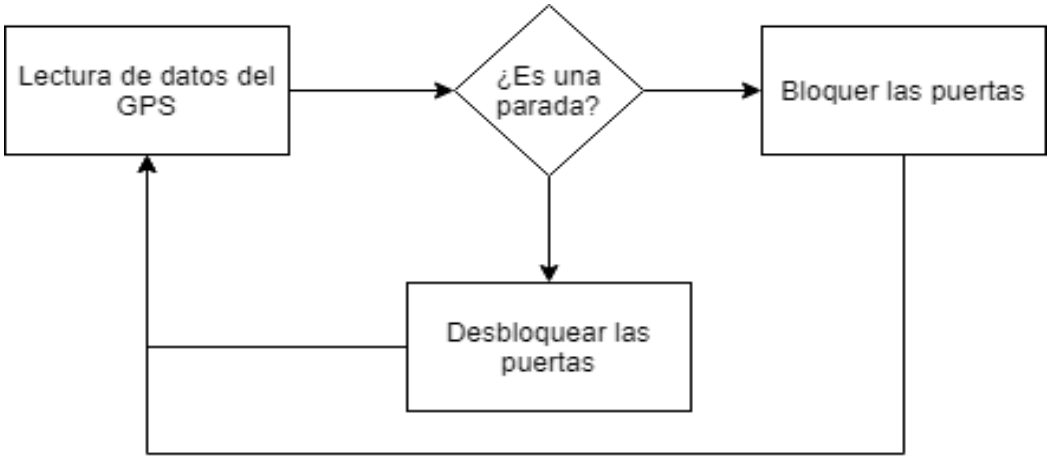

Figura #51 - Diagrama de flujo del control de puertas

El sistema de bloque y desbloque de puertas se realiza mediante un relay el cual convierte la señal de 5 voltios de la Raspberry Pi a los 12 voltios con los que trabaja la electroválvula.

Para identidificar si se encuentra en una parada se debe recorrer todas las paradas existentes y comaprar los valores de latotud y longitud, para esto se diseño el algoritmo de la Figura #52, el cual recorre todas las paradas existentes en la base de datos y una por una compara la latitud y la longitud de las mismas para poder determinar si se encuentra o no en esa parada.

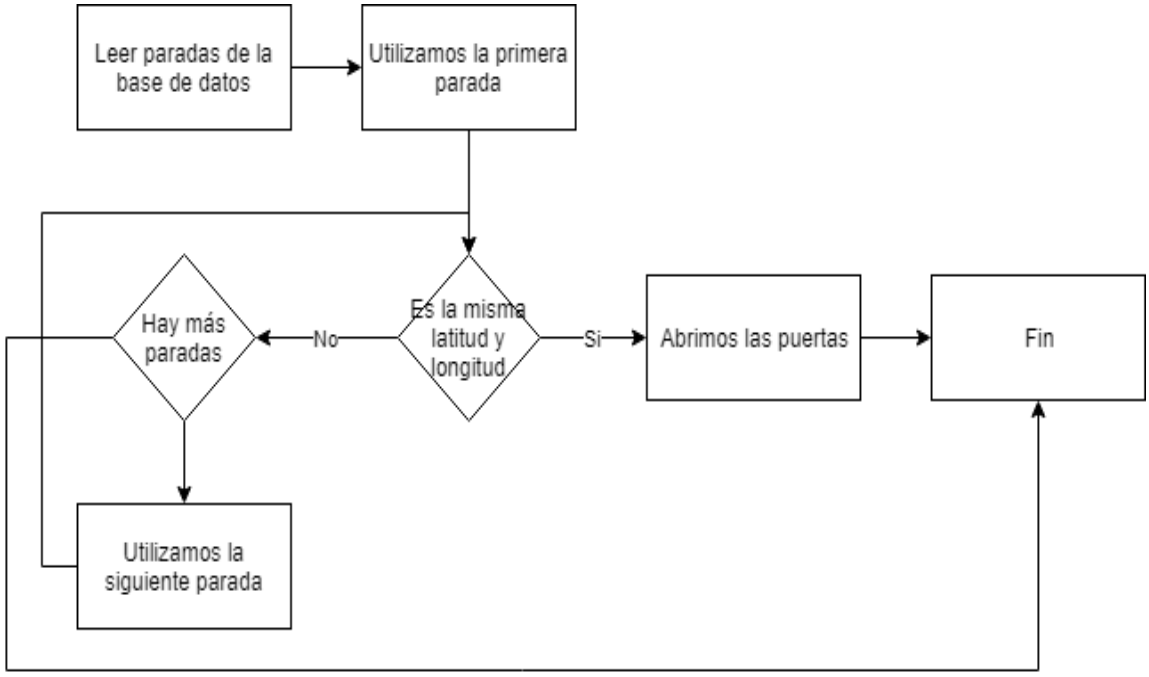

Figura #52 - Algoritmo de busqueda de paradas

Cuando las puertas se encuentran abiertas el sistema toma fotos automáticamente y las almacena en la Raspberry Pi como se peude apreciar en la Figura #53.

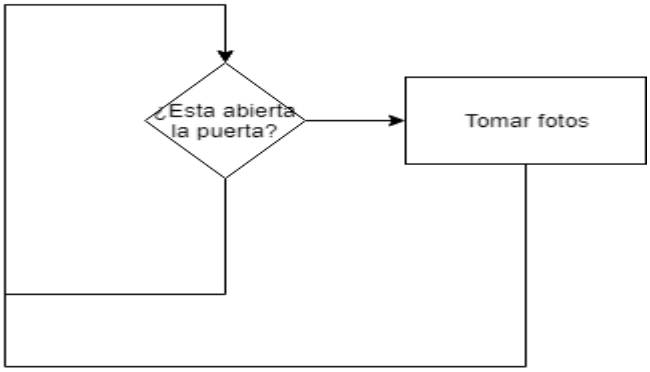

Figura #53 - Diagrama de flujo del sistema de fotos

Caundo se toman fotos estas son almacenadas en la tarjeta de memoria de la Raspberry Pi, pese a que esta tarjeta es de 8GB en algún punto se llenará de fotos y será necesario que se borren automáticamente las más antiguas, por esto es importante que exista un mecanismo de borrado automático de las fotos más antiguas tal como se ve en la Figura #54.

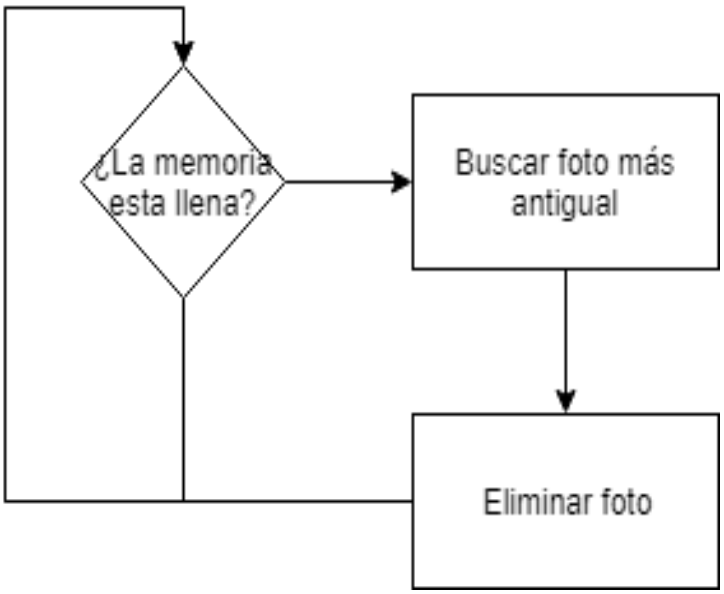

Figura #54 - Algoritmo de gestión de memoria en disco duro

CONCLUSIONES

Se pudo verificar como la Raspberry Pi ofrece las suficientes capacidades para funcionar como servidor web y administrar su propia base de datos.

Se hizo evidente el gran alcance del que dispone el arduino ya que mediante una placa de GPS se puede geolocalizar el bus en cualquier parte del mundo.

Se logró verificar el sistema de control que permite la apertura y cierre de las puertas desde la Raspberry mediante una electroválvula.

Se realizó exitosamente la comunicación entre la Raspberry Pi y el Arduino mediante el uso del protocolo serial.

Se concluye que se puede tomar fotos de alta resolución con la cámara de la Raspberry Pi.

RECOMENDACIONES

Se debe ubicar el dispositivo en un área donde tenga la temperatura adecuada para poder operar.

Se recomienda usar una batería externa para el dispositivo ya que esto asegurará que siempre tengamos la ubicación del bus aun cuando el bus esté apagado o la batería se haya agotado.

Para poder lograr una mayor precisión del GPS es necesario que este esté lo más cerca posible del exterior del vehículo, para evitar retrasos o pérdidas de datos.

No se deben exponer los equipos al agua ni al sol de forma directa ya que estos pueden degradar y dañar de forma permanente los equipos.

El interior del bus debe tener una correcta iluminación para asegurar fotos de buena calidad cada vez que se abren las puertas.

Para el acceso a internet del dispositivo se recomienda usar la conexión vía modem ya que es la que asegura una mejor disponibilidad.

Es altamente recomendado que el dispositivo disponga de algún sistema de ventilación externo para evitar que la temperatura averíe el dispositivo.

BIBLIOGRAFÍA

Andrew W. Senior, J. H. (2004). *Guide to Biometrics.*

Anil K. Jain, A. A. (2011). *Introduction to Biometrics.*

Arduino. (2008). *Arduino Nano.* Obtenido de arduino.cc: https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

B. Navya Rupa, G. K.-h. (2015). Test Report Generation Using JSON. *International Journal of Software Engineering and Its Applications*.

Banzi, M. (2008). *Getting Started with Arduino.*

Batini, C. (s.f.). *Conceptual Database Design, An Entity-relationship Approach.* 1992: Benjamin/Cummings.

Ben Everard, E. U. (2014). *Learning Computer Architecture with Raspberry Pi.*

Blum, C. B. (2013). *Sams Teach Yourself Python Programming for Raspberry Pi in 24 Hours.*

Boulgouris, N. V. (2009). *Biometrics: Theory, Methods, and Applications.*

Boxall, J. (2013). *Arduino Workshop: A Hands-On Introduction with 65 Projects.*

Brock Craft, J. E. (2015). *Raspberry Pi Projects For Dummies.*

C.Calderón, M. C. (2014). Desarrollo de una Aplicación Cliente/Servidor para un Wall View en base a la. *REVISTA EPN*, 7. Obtenido de https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/155/pdf

Campuzano, & Cedeño. (2015). Diseño e implementación de un sistema de seguridad de control local y remoto con dispositivos de vigilancia, desarrolado en el software python, centralizandose en una tarjeta raspberry pi.

Castillo. (2012). HTTP.

Ceballos, E. L. (2013). Diseño web adaptativo o responsivo. *Revista Digital Universitaria*.

Chávez, & Yuquilema. (2013). Aplicaciones con minicomputadores raspberry pi provisto de módulo gps y acelerómetro para control de velocidad y posicionamiento.

El Telégrafo . (2017).

Elio, E. M. (2015). *Microcontrolador Arduino.* Universidad Cristobal Colón.

Engloba Systems. (1 de Agosto de 2014). *Conexión de java ide netbeans a caulquier base de datos (jdbc)*. Obtenido de https://www.youtube.com/watch?v=Nt9gfoM7UJQ

Engloba Systems. (1 de Agosto de 2014). *Conexión de java ide netbeans a cualquier base de datos (jdbc)*. Obtenido de https://www.youtube.com/watch?v=Nt9gfoM7UJQ

Fernando Bizzarro, A. H. (2016). *The V-Dem Party Institutionalization Index: a new global indicator (1900-2015).*

Gómez, F. M. (2013). *Definición de bases de datos y de sistema de administración de bases de datos.* México: UNIVERSIDAD AUTÓNOMA METROPOLITANA .

Guillermo Diez-Andino Sancho, R. M. (2003). Desarrollo de un servidor HTTP para dispositivos móviles en J2ME. *Departamento. Ingeniería Telemática - Universidad Carlos III de Madrid*. Obtenido de http://www.it.uc3m.es/celeste/papers/ServidorHTTP.pdf

Gutierrez, D. A. (2016). *BASES DE DATOS .* Estados Unidos : Atlantic International University. .

Hidalgo, M. (2015). Introducción a la Robótica con Arduino. *Robótica Educativa EDUCABOT, Centro de Profesores de Albacete, España*.

Javier, L. S. (2001). Base de Datos Distribuidas Estudio de Actualización de Réplicas. *Facultad de Informática – UNLP*. Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/3872/Documento_completo__.pdf-PDFA1b.pdf?sequence=1

Jayesh Umre, K. B. (2014). Comparative performance analysis of MySQL and SQLite relational database management systems in Windows 10 environments. *International Journal of Latest Trends in Engineering and Technology*.

Jiménez. (2015). Construcción de un ordenador electrónico vehicular con sistema de seguridad y gps utilizando raspberry pi y hardware libre.

Jimy Alexander Cortés Osorio, F. A. (2010). *Sistemas de seguridad basados en biometría.*

Kilicdagi, A. (2014). *Laravel Design Patterns and Best Practices.*

Margolis, M. (2011). *Arduino Cookbook.*

María F. Maldonado, A. C. (2008). Implementación de un sistema Web para manejo de datos meteorológicos del Laboratorio de Energías Alternativas y Eficiencia Energética de la Escuela Laboratorio de Energías Alternativas y Eficiencia Energética de la Escuela Politécnica Nacional. Obtenido de http://bibdigital.epn.edu.ec/bitstream/15000/4923/1/PAPER%20Implementaci%C3%B3n%20de%20un%20sistema%20Web%20para%20manejo%20de%20datos%20meteorol%C3%B3gicos%20del%20Laboratorio%20de%20Energ%C3%ADas%20Alterna.pdf

Mishra, A. (2014). Critical Comparison Of PHP And ASP.NET For Web Development. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3,*. Obtenido de http://www.ijstr.org/final-print/july2014/Critical-Comparison-Of-Php-And-Aspnet-For-Web-Development.pdf

Molloy, D. (2016). *Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux.* Obtenido de https://www.element14.com/community/community/raspberry-pi?src=raspberrypi

Monk, S. (2012). *Programming the Raspberry Pi: Getting Started with Python.*

Mulloy, B. (2012). *Web API Design, Crafting Interfaces that Developers Love.* apigee.

Oracle. (2011). *PHP Scalability and High Availability Database Resident Connection Pooling and Fast Application Notification.*

Peck, A. (2017). *Jumpstarting the Raspberry Pi Zero W.*

Philbin, C. A. (2013). *Adventures in Raspberry Pi.*

Purdum, J. J. (2012). *Beginning C for Arduino: Learn C Programming for the Arduino.*

Ross, A. A. (2006). *Handbook of Multibiometrics.*

Scott Trent, M. T. (2008). *Performance Comparison of PHP and JSP as Server-Side Scripting Languages.* Obtenido de https://www.researchgate.net/publication/225161349_Performance_Comparison_of_PHP_and_JSP_as_Server-Side_Scripting_Languages

Sepulcre, M., Mittag, J., Santi, P., Hartenstein, H., & Gozalvez, J. (2011). Congestion and awareness control in cooperative vehicular systems. *Proceedings of the IEEE*, 1260-1279.

Spectator, C. (2016). *Comparativa de rendimiento de la red entre distintos proveedores Cloud desde diferentes localizaciones geográficas.*

Vélez, J. (2017). Contribución de la tecnología en la gestión del conocimiento entre los grupos de investigación del área de informática. *Puente*, 21-28.

Vicenteño. (2016). HTTP.

ANEXOS

PRESUPUESTO

Tabla 3 - Presupuesto para implementación

Material	Valor	Cantidad	Subtotal
Raspberry Pi zero	$ 80.00	1	$ 80.00
Arduino nano	$ 28.00	1	$ 28.00
Módulo GSM	$ 80.00	1	$ 80.00
Módulo GPS	$ 50.00	1	$ 50.00
Case para Raspberry	$ 12.00	1	$ 12.00
Tarjeta micrSd de 8Gb	$ 15.00	1	$ 15.00
Cargador microUSB de 2A	$ 15.00	1	$ 15.00
Cable USB tipo B	$ 8.00	1	$ 8.00
Sistema de control neumático	$ 650.00	1	$ 650.00
Materiales varios	$ 45.00	1	$ 45.00
Adaptador USB a LAN	$ 18.00	1	$ 18.00
Adaptador micrUSB a USB	$ 11.00	1	$ 11.00
Adaptador miniHDMI a HDMI	$ 12.00	1	$ 12.00
Total			**$ 1,024.00**

CRONOGRAMA DE ACTIVIDADES

Tabla 4 - Cronograma de actividades a realizar

Actividad	Semanas																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Adquisición de equipos	■	■	■	■																
Diseño de esquemas electrónicos	■	■																		
Diseño de esquemas neumáticos			■	■																
Reconocimiento de los buses de transporte urbano	■	■																		
Creación de la interfaz web					■	■	■													
Configuración de módulo GPS							■	■												
Configuración de módulo GSM							■	■	■											
Configuración del arduino nano										■	■	■								
Comunicación de Raspberry Pi con Arduino										■	■									
Diseño de prototipo de prueba													■	■						
Fase de pruebas de corección de errores															■	■	■	■		
Diseño de prototipo final																			■	■

CÓDIGO FUENTE

LECTURA DEL DATOS DEL GPS CON ARDUINO

El arduino tendrá el control del GPS, será el encargador de solicitar los datos y enviarlos a la Raspberry Pi para su posterior proceso.

```
#include "U8glib.h"
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);

#define SpeedLED 13
#define GPSECHO  false
#define speeds 60

double lonGPS=0;
double latGPS=0;
boolean usingInterrupt = false;
void useInterrupt(boolean);

void setup()
{
  pinMode(SpeedLED, OUTPUT);
  Serial.begin(19200);
  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
  GPS.sendCommand(PGCMD_ANTENNA);
  useInterrupt(true);
  delay(1000);
  mySerial.println(PMTK_Q_RELEASE);
  Serial.println("o");
}

SIGNAL(TIMER0_COMPA_vect) {
  char c = GPS.read();
#ifdef UDR0
  if (GPSECHO)
    if (c) UDR0 = c;
#endif
}

void useInterrupt(boolean v) {
  if (v) {
    OCR0A = 0xAF;
```

```
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}

#define NO_FIX "0x01"
#define FIXED "0x02"
void loop()
{
  gpsLoop();
  if(Serial.available()){
    char cmd=Serial.read();
    if(GPS.fix)
    {
      latGPS=GPS.latitudeDegrees;
      lonGPS=GPS.longitudeDegrees;

Serial.println("{\"state\":\""+String(FIXED)+"\",\"latitude\":\""+String(latGPS,6)+"\",\"
longitude\":\""+String(lonGPS,6)+"\",\"speed\":\""+String(GPS.speed*2,2)+"\"}");
      if(GPS.speed*2>=speeds){
        digitalWrite(SpeedLED,HIGH);
      }else{
        digitalWrite(SpeedLED,LOW);
      }
    }
    else
    {
      Serial.println("{\"state\":\""+String(NO_FIX)+"\"}");
    }
  }
}

uint32_t timer = millis();
void gpsLoop()
{
  if (! usingInterrupt) {
    char c = GPS.read();
    if (GPSECHO)
      if (c) Serial.print(c);
  }

  if (GPS.newNMEAreceived()) {
    if (!GPS.parse(GPS.lastNMEA()))
      return;
```

```
  }

  if (timer > millis())  timer = millis();

  if (millis() - timer > 2000) {
    timer = millis();
  }
}
```

MANIPULACIÓN DE LOS DATOS DEL GPS

En la Raspberry Pi se leerán los datos del puerto serie, entre ellos los datos del GPS, luego consultará las posiciones registradas en la base de datos y validará si la puerta se debe o no abrir.

```
from serial_tools import Serial
from json import loads as json_decoder
from serial import SerialException
import _mysql
import gpiozero
import urllib.request
import urllib.parse

print("entra")

ledBlanco = gpiozero.LED(26)
ledAzul = gpiozero.LED(27)
ledMulti = gpiozero.LED(22)

dist = 0.0003
latArray=[]
lonArray=[]

db=_mysql.connect("localhost","pi","raspberry","control_bus")
db.query("""SELECT longitude, latitude FROM stations WHERE deleted_at
is Null""")
r=db.store_result()
for x in range(0,r.num_rows()):
    query=r.fetch_row()[0]
    latArray.append(float(query[1]))
    lonArray.append(float(query[0]))
db.close()

while True:
    try:
        estadoSerial=False
```

```
                arraySerial=['/dev/ttyUSB0','/dev/ttyUSB2','/dev/ttyUSB1']
                while estadoSerial==False:
                    for serial in arraySerial:
                        try:
                            arduino = Serial(serial,19200)
                            estadoSerial=True
                            break
                        except SerialException as error:
                            print(error)

                arduino.read()
                ledAzul.on()

                while True:
                    estdoLed = False
                    arduino.write('y')
                    serial_data=arduino.read('{','}')
                    json=json_decoder(serial_data)

                    if json['state']=='0x02':

                        print("ok")

                        ledBlanco.on()
                        latGPS = float(json['latitude'])
                        lonGPS = float(json['longitude'])
                        speed = float(json['speed'])


url='http://gpsbus.sugalmedical.com/api/locations/create?latitude={latitude}&longit
ude={longitude}&speed={speed}'.format(latitude=latGPS,longitude=lonGPS,spee
d=speed)
                        response=urllib.request.urlopen(url)

                        for x in range(0,len(latArray)):

                            lat=latArray[x]
                            lon=lonArray[x]

                            if latGPS>=lat-dist and latGPS<=lat+dist and lonGPS>=lon-
dist and lonGPS<=lon+dist:
                                estdoLed = True
                                ledMulti.on()
                                break
                        if estdoLed == False:
                            ledMulti.off()
                    else:
```

```
                    ledBlanco.off()
            except Exception as error:
                ledMulti.off()
                ledBlanco.off()
                ledAzul.off()
                print(error)
```

TOMAR FOTOS

Mientras la puerta esté abierta se deben tomar fotos del interior del bus, para eso necesario leer el estado del sensor de la puerta y si este indica que la puerta está abierta tomar una ráfaga de fotos.

```
        import gpiozero
        import os
        from picamera import PiCamera

        ledAmarillo = gpiozero.LED(17)
        buttonSystem = gpiozero.Button(6)
        buttonPhoto = gpiozero.Button(5)

        frame = 1
        camera = PiCamera()
        while True:
            if buttonSystem.is_pressed==True:
                os.system('sudo shutdown -h now')
            try:
                if buttonPhoto.is_pressed==True:
                    print("entra")
                    ledAmarillo.on()
                    if frame==1:
                        f=os.popen("sudo  TZ=America/Guayaquil  date  +%F_%H-%M-
%S")
                        now=f.read()
                        now=now[0:-1]
                        os.system("sudo mkdir /var/www/html/photo/"+now)
                        os.system("sudo chmod 777 /var/www/html/photo/"+now)
                    camera.resolution=(2592,1944)
                    camera.capture('/var/www/html/photo/'+now+'/%0d.jpg'          %
frame,resize=(1280,720))
                    frame+=1
                    print("sale")
                else:
                    ledAmarillo.off()
                    frame=1
            except Exception as error:
```

```
print(error)
```

GESTION DEL SERVIDOR

El servidor permite administrar paradas permitidas permitidas, para eso es necesario poder conectarse a la base de datos para poder crear, editar y eliminar los recursos.

```
<?php

namespace App\Http\Controllers;

use App\Libraries\Redirector\Redirector;
use App\Libraries\Tools\Paginator;
use App\Station;
use App\User;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Http\Request;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Support\Facades\Auth;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

    function showLogin(){
        return view('login', [
        ]);
    }
    function isLogin(Request $request){
        $loginData=$request->only(['email','password']);
        if(!Auth::attempt($loginData))
        {
            return Redirector::error('Credenciales inválidas',['No existe un
usuario con los datos ingresados']);
        }
        return redirect('/viewStation');
    }
    function isLogout()
    {
        Auth::logout();
        return redirect('/');
    }
    function viewStation(){
        $stations = Station::orderBy('created_at','desc')->paginate(24);
```

```
            return view('stations', [
                'stations' => $stations,
            ]);
        }
        function createStation(Request $request){
                    try{
            $station=new Station();
                            $station->name=$request->name;
                            $station->latitude=$request->latitude;
                            $station->longitude=$request->longitude;
                            $station->save();
                            return Redirector::success('Estación creada',['La estación fue creada correctamente']);
            }catch (QueryException $e){
                return Redirector::warning('No se pudo crear la estación',[$e->getMessage()]);
            }

        }
        function editStation(Request $request)
        {
            try{
                $station=Station::find($request->station_id);
                $station->name=$request->name;
                $station->latitude=$request->latitude;
                $station->longitude=$request->longitude;
                $station->save();
                return Redirector::primary('Estación actualizada',['La estación fue actualizada correctamente']);
            }catch (QueryException $e){
                return Redirector::warning('No se pudo editar la estación',[$e->getMessage()]);
            }
        }
        function deleteStation(Request $request){
            try{
                $station=Station::find($request->station_id);
                $station->delete();
                return Redirector::error('Estación eliminada',['La estación fue eliminada correctamente']);
            }catch (QueryException $e){
                return Redirector::warning('No se pudo eliminar la estación',[$e->getMessage()]);
            }
        }
        function viewVideos(){
            $handleCarpetas = opendir('photo/');
```

```
                $carpetas=array();
                $photos=array();
                $arrayPhotos=array();
                while (false !== ($carpeta = readdir($handleCarpetas))) {
                    if ($carpeta!= "." && $carpeta!= "..") {
                        $carpetas[] = $carpeta;
                    }
                }
                closedir($handleCarpetas);
                rsort($carpetas);
                foreach($carpetas as $carpeta){
                                unset($photos);
                                $photos=array();
                                $handlePhoto = opendir('photo/'.$carpeta.'/');
                                while (false !== ($carpeta = readdir($handlePhoto)))
{
                                        if ($carpeta!= "." && $carpeta!= "..") {
                                                $photos[] = $carpeta;
                                        }
                                }
                                sort($photos);
                                $arrayPhotos[] = $photos;
                                closedir($handlePhoto);
                        }
                        $carpetas=Paginator::paginate($carpetas,12,'/viewVideos');
                    return view('videos', [
                        'carpetas' => $carpetas,
                        'photos'=>$arrayPhotos
                    ]);
                }
                function viewPhoto($carpeta,$foto){
                        $path=public_path().'/../photo/'.$carpeta.'/'.$foto;
                        return \Response::make(file_get_contents($path),200,[
                                'Content-Type'=>'application',
                                'Content-Disposition'=>'inline;filename="'.$path.""
                        ]);
                    }
                    function editUser(Request $request){
                        if(strcmp($request->pass,$request->passRepeat)===0){
                                try{
                                        $id=Auth::id();
                                        $user=User::find($id);
                                        $user->password=bcrypt($request->pass);
                                        $user->save();
                                        return                Redirector::success('Datos
actualizados',['Los datos fueron actualizados correctamente']);
                                }catch (QueryException $e){
```

```
                                    return Redirector::warning('No se pudo
actualizar los datos',[$e->getMessage()]);
                                }
                            }else{
                                return Redirector::warning('No coincidencia',['No son
iguales las contraseñas ingresadas']);
                            }
                        }
                    }
```

RUTAS DE LA APLICACIÓN

Las rutas son los puntos de acceso de todas las aplicaciones web, estas definen como se puede acceder a las funcionalidades que ofrece la aplicación.

```
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::group(['middleware' => [\App\Http\Middleware\IsLogin::class]],
function () {
    Route::get(__('url.login.showLogin'), 'Controller@showLogin')-
>middleware(\App\Http\Middleware\CreateAdmin::class);
    Route::post(__('url.login.isLogin'), 'Controller@isLogin');
});

Route::group(['middleware' => [\App\Http\Middleware\IsNotLogin::class]],
function () {

    Route::get(__('url.login.isLogout'),'Controller@isLogout');
    Route::get(__('url.station.view'),'Controller@viewStation');
    Route::get(__('url.viewVideos'),'Controller@viewVideos');

    Route::post(__('url.station.create'),'Controller@createStation');
    Route::post(__('url.station.edit'),'Controller@editStation');
    Route::post(__('url.station.delete'),'Controller@deleteStation');
    Route::post(__('url.user.edit'),'Controller@editUser');
```

```
Route::get(__('url.viewPhoto',['carpeta'=>'{carpeta}','foto'=>'{foto}']),'Controller@viewPhoto');
        });
```

TABLA DE PARADAS EN LA BASE DATOS

Los datos de las paradas deben ser almacenadas de forma permanente, para luego poder ser consultadas y permitir o no abrir las puertas, por eso es necesario almacenar la latitud y la longitud, así como un nombre para cada una de las paradas permitidas.

```
        <?php

        use Illuminate\Support\Facades\Schema;
        use Illuminate\Database\Schema\Blueprint;
        use Illuminate\Database\Migrations\Migration;

        class CreateStationsTable extends Migration
        {
            /**
             * Run the migrations.
             *
             * @return void
             */
            public function up()
            {
                Schema::create('stations', function (Blueprint $table) {
                    $table->increments('id');
                    $table->text('name');
                    $table->decimal('longitude',10,6);
                    $table->decimal('latitude',10,6);
                    $table->softDeletes();
                    $table->timestamps();
                });
            }

            /**
             * Reverse the migrations.
             *
             * @return void
             */
            public function down()
            {
                Schema::dropIfExists('stations');
```

```
        }
    }
```

VISTAS DE LA APLICACIÓN

Para que el usuario final pueda acceder de forma sencilla a la aplicación esta debe tener una interfaz de usaurio, para el caso actual hemos creado una interfaz web.

INTERFAZ DE INICIO DE SESIÓN

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Login</div>

                <div class="panel-body">
                    <form class="form-horizontal" method="POST" action="{{ route('login') }}">
                        {{ csrf_field() }}

                        <div class="form-group{{ $errors->has('email') ? ' has-error' : '' }}">
                            <label for="email" class="col-md-4 control-label">E-Mail Address</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control" name="email" value="{{ old('email') }}" required autofocus>

                                @if ($errors->has('email'))
                                    <span class="help-block">
                                        <strong>{{ $errors->first('email') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group{{ $errors->has('password') ? ' has-error' : '' }}">
                            <label for="password" class="col-md-4 control-label">Password</label>

                            <div class="col-md-6">
```

```
                                <input id="password" type="password" class="form-
control" name="password" required>

                                @if ($errors->has('password'))
                                    <span class="help-block">
                                        <strong>{{ $errors->first('password') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <div class="checkbox">
                                    <label>
                                        <input type="checkbox" name="remember" {{
old('remember') ? 'checked' : '' }}> Remember Me
                                    </label>
                                </div>
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-8 col-md-offset-4">
                                <button type="submit" class="btn btn-primary">
                                    Login
                                </button>

                                <a class="btn btn-link" href="{{
route('password.request') }}">
                                    Forgot Your Password?
                                </a>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

INTERFAZ DE REGISTRO DE USUARIOS

```
@extends('layouts.app')
```

```
@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Register</div>

                <div class="panel-body">
                    <form class="form-horizontal" method="POST" action="{{ route('register') }}">
                        {{ csrf_field() }}

                        <div class="form-group{{ $errors->has('name') ? ' has-error' : '' }}">
                            <label for="name" class="col-md-4 control-label">Name</label>

                            <div class="col-md-6">
                                <input id="name" type="text" class="form-control" name="name" value="{{ old('name') }}" required autofocus>

                                @if ($errors->has('name'))
                                    <span class="help-block">
                                        <strong>{{ $errors->first('name') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group{{ $errors->has('email') ? ' has-error' : '' }}">
                            <label for="email" class="col-md-4 control-label">E-Mail Address</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control" name="email" value="{{ old('email') }}" required>

                                @if ($errors->has('email'))
                                    <span class="help-block">
                                        <strong>{{ $errors->first('email') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>
```

```
                        <div class="form-group{{ $errors->has('password') ? ' has-
error' : '' }}">
                            <label for="password" class="col-md-4 control-
label">Password</label>

                            <div class="col-md-6">
                                <input id="password" type="password" class="form-
control" name="password" required>

                                @if ($errors->has('password'))
                                    <span class="help-block">
                                        <strong>{{ $errors->first('password') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group">
                            <label for="password-confirm" class="col-md-4 control-
label">Confirm Password</label>

                            <div class="col-md-6">
                                <input id="password-confirm" type="password"
class="form-control" name="password_confirmation" required>
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <button type="submit" class="btn btn-primary">
                                    Register
                                </button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

PLANTILLA PRINCIPAL DE LA APLICACIÓN WEB

```
<!DOCTYPE html>
<html lang="{{ app()->getLocale() }}">
```

```
        <head>
            <meta charset="utf-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-scale=1">

            <!-- CSRF Token -->
            <meta name="csrf-token" content="{{ csrf_token() }}">

            <title>{{ config('app.name', 'Laravel') }}</title>

            <!-- Styles -->
            <link href="{{ asset('css/app.css') }}" rel="stylesheet">
        </head>
        <body>
            <div id="app">
                <nav class="navbar navbar-default navbar-static-top">
                    <div class="container">
                        <div class="navbar-header">

                            <!-- Collapsed Hamburger -->
                            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#app-navbar-collapse">
                                <span class="sr-only">Toggle Navigation</span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                            </button>

                            <!-- Branding Image -->
                            <a class="navbar-brand" href="{{ url('/') }}">
                                {{ config('app.name', 'Laravel') }}
                            </a>
                        </div>

                        <div class="collapse navbar-collapse" id="app-navbar-collapse">
                            <!-- Left Side Of Navbar -->
                            <ul class="nav navbar-nav">
                                 
                            </ul>

                            <!-- Right Side Of Navbar -->
                            <ul class="nav navbar-nav navbar-right">
                                <!-- Authentication Links -->
                                @if (Auth::guest())
                                    <li><a href="{{ route('login') }}">Login</a></li>
                                    <li><a href="{{ route('register') }}">Register</a></li>
                                @else
```

```
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-
toggle="dropdown" role="button" aria-expanded="false">
                                {{ Auth::user()->name }} <span
class="caret"></span>
                            </a>

                            <ul class="dropdown-menu" role="menu">
                                <li>
                                    <a href="{{ route('logout') }}"
                                        onclick="event.preventDefault();
                                                document.getElementById('logout-
form').submit();">
                                        Logout
                                    </a>

                                    <form id="logout-form" action="{{ route('logout')
}}" method="POST" style="display: none;">
                                        {{ csrf_field() }}
                                    </form>
                                </li>
                            </ul>
                        </li>
                    @endif
                </ul>
            </div>
        </div>
    </nav>

    @yield('content')
</div>

<!-- Scripts -->
<script src="{{ asset('js/app.js') }}"></script>
</body>
</html>
```

VISTA DE LAS ESTACIONES PERMITIDAS

```
@component('templates.basic')

    <div class="container-fluid text-center">
        <img class="img-fluid" src="{{asset('img/bus.png')}}" style="max-
height:300px">
    </div>
```

```
            <nav class="navbar navbar-expand-sm bg-dark navbar-dark sticky-top">
               <a class="navbar-brand" href="#">Control Bus</a>

               <ul class="nav navbar-nav ml-auto">
                  <li class="nav-item">
                     <a class="nav-link" href="{{__('url.login.isLogout')}}"><i
class="fas fa-sign-out-alt"></i></a>
                  </li>
               </ul>
            </nav><br>

            <div class="container-fluid">
               <button type="button" class="btn btn-success" data-toggle="modal"
data-target="#myModal">Crear Parada</button>
                <button type="button" class="btn btn-primary" data-toggle="modal"
data-target="#myModalChange">Cambiar Credenciales</button>
               <a class="btn btn-dark" href="{{__('url.viewVideos')}}">Fotos</a>
            </div>

            <!-- The Modal -->
            <div class="modal fade" id="myModal">
               <div class="modal-dialog modal-lg">
                  <div class="modal-content">

                     <!-- Modal Header -->
                     <div class="modal-header">
                        <h4 class="modal-title">Parada</h4>
                        <button type="button" class="close" data-
dismiss="modal">&times;</button>
                     </div>

                     <form action="{{__('url.station.create')}}" method="post">

                        <!-- Modal body -->
                        <div class="modal-body">

                              {{csrf_field()}}
                              <div class="form-group">
                                 <label for="name">Nombre</label>
                                 <input type="text" class="form-control" id="name"
name="name" required>
                              </div>
                              <div class="form-group">
                                 <label for="latitude">Latitud</label>
                                 <input type="number" class="form-control"
id="latitude" name="latitude" step="any" max=9999 min=-9999 required>
                              </div>
```

```
                                <div class="form-group">
                                    <label for="longitude">Longitud</label>
                                    <input type="number" class="form-control"
id="longitude" name="longitude" step="any" max=9999 min=-9999 required>
                                </div>
                            </div>

                            <!-- Modal footer -->
                            <div class="modal-footer">
                                <button type="submit" class="btn btn-
success">Crear</button>
                                <!--<button type="button" class="btn btn-secondary" data-
dismiss="modal">Cerrar</button>-->
                            </div>

                        </form>

                    </div>
                </div>
            </div>

            <!-- The Modal Credenciales-->
            <div class="modal fade" id="myModalChange">
                <div class="modal-dialog modal-lg">
                    <div class="modal-content">

                        <!-- Modal Header -->
                        <div class="modal-header">
                            <h4 class="modal-title">Credenciales</h4>
                            <button type="button" class="close" data-
dismiss="modal">&times;</button>
                        </div>

                        <form action="{{__('url.user.edit')}}" method="post">

                            <!-- Modal body -->
                            <div class="modal-body">

                                {{csrf_field()}}
                                <div class="form-group">
                                    <label for="pass">Nueva Contraseña</label>
                                    <input type="password" class="form-control" id="pass"
name="pass" required>
                                </div>
                                <div class="form-group">
                                    <label for="passRepeat">Repetir Nueva
Contraseña</label>
```

```
                                <input type="password" class="form-control" id="passRepeat" name="passRepeat" required>
                            </div>
                        </div>

                        <!-- Modal footer -->
                        <div class="modal-footer">
                            <button type="submit" class="btn btn-success">Actualizar</button>
                            <!--<button type="button" class="btn btn-secondary" data-dismiss="modal">Cerrar</button>-->
                        </div>

                    </form>

                </div>
            </div>
        </div>


        <div class="container-fluid"><br>
            <div class="row">
                @foreach ($stations as $station)

                    <div class="col-sm-6 col-md-6 col-lg-4 col-xl-3">
                        <div class="card bg-light text-dark">
                            <!--<img class="card-img-top" src="img_avatar1.png" alt="Card image">-->
                            <div class="card-body text-center">
                                <h6>Nombre: {{$station->name}}</h6>
                                <h6>Latitud: {{$station->latitude}}</h6>
                                <h6>Longitud: {{$station->longitude}}</h6>
                                <br>
                                <button type="button" class="btn btn-danger" data-toggle="modal" data-target="#myModalDelete{{$station->id}}">
                                    Eliminar
                                </button>
                                <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModalEdit{{$station->id}}">
                                    Editar
                                </button>
                            </div>
                        </div><br>
                    </div>

                    <!-- The Modal -->
                    <div class="modal fade" id="myModalDelete{{$station->id}}">
```

```
                        <div class="modal-dialog modal-lg">
                            <div class="modal-content">
                                <!-- Modal Header -->
                                <div class="modal-header">
                                    <h4 class="modal-title">Eliminar Parada</h4>
                                    <button type="button" class="close" data-
dismiss="modal">&times;</button>
                                </div>
                                <form action="{{__('url.station.delete')}}" method="post">
                                    {{csrf_field()}}
                                    <input type="hidden" class="form-control"
id="station_id" name="station_id" value="{{$station->id}}" required>
                                    <!-- Modal footer -->
                                    <div class="modal-footer">
                                        <button type="submit" class="btn btn-
danger">Eliminar</button>
                                    </div>
                                </form>
                            </div>
                        </div>
                    </div>

                    <!-- The Modal -->
                    <div class="modal fade" id="myModalEdit{{$station->id}}">
                        <div class="modal-dialog modal-lg">
                            <div class="modal-content">
                                <!-- Modal Header -->
                                <div class="modal-header">
                                    <h4 class="modal-title">Editar Parada</h4>
                                    <button type="button" class="close" data-
dismiss="modal">&times;</button>
                                </div>
                                <form action="{{__('url.station.edit')}}" method="post">
                                    <!-- Modal body -->
                                    <div class="modal-body">
                                        {{csrf_field()}}
                                        <input type="hidden" class="form-control"
id="station_id" name="station_id" value="{{$station->id}}" required>
                                        <div class="form-group">
                                            <label for="name">Nombre</label>
                                            <input type="text" class="form-control" id="name"
name="name" value="{{$station->name}}" required>
                                        </div>
                                        <div class="form-group">
                                            <label for="latitude">Latitud</label>
```

```
                                    <input type="number" class="form-control" id="latitude" name="latitude" value="{{$station->latitude}}" step="any" max=9999 min=-9999 required>
                                </div>
                                <div class="form-group">
                                    <label for="longitude">Longitud</label>
                                    <input type="number" class="form-control" id="longitude" name="longitude" value="{{$station->longitude}}" step="any" max=9999 min=-9999 required>
                                </div>
                            </div>
                            <!-- Modal footer -->
                            <div class="modal-footer">
                                <button type="submit" class="btn btn-primary">Editar</button>
                                <!--<button type="button" class="btn btn-secondary" data-dismiss="modal">Cerrar</button>-->
                            </div>
                        </form>
                    </div>
                </div>
            </div>

            @endforeach
        </div>
    </div><br>

    <div class="row">
        <div class="mx-auto">
            {{$stations->links()}}
        </div>
    </div><br>

@endcomponent
```

APLICACIÓN DEL SERVIDOR

Toda la información se centraliza en un servidor web, el cual dispondrá de su propia base de datos para llevar los registros.

RUTAS DE ACCESO DE LA APLICACIÓN

```
<?php

/*
|--------------------------------------------------------------------------
```

```
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/','SpaController@index');
Route::get('/api/locations/create','Api\LocationsController@create');
Route::get('/api/locations/last','Api\LocationsController@last');
```

COMPONENTE DE GOOGLE MAPS

```
<template lang="pug">
    .full-page(v-bind:id='id')
</template>

<script>
    import Format from '../tools/Format'
    export default {
        props: {
            id: {
                required: true,
                type: String
            },
            latitude: {
                required: true,
                type: Number
            },
            longitude: {
                required: true,
                type: Number
            },
            speed: {
                required: true,
                type: Number
            }
        },
        data(){
            return{
                position:null,
                map:null,
                marker:null
            }
```

```
            },
            watch:{
                latitude(){
                    this.drawMarker()
                },
                longitude(){
                    this.drawMarker()
                }
            },
            mounted(){
                this.drawMap()
            },
            methods:{
                drawMap(){
                    const canvas=document.getElementById(this.id)

                    this.position=new
google.maps.LatLng(this.latitude,this.longitude)
                    const mapOptions={
                        center:this.position,
                        zoom:18
                    }
                    this.map=new google.maps.Map(canvas,mapOptions)
                },
                drawMarker(){
                    this.position=new
google.maps.LatLng(this.latitude,this.longitude)

                    if(this.marker!=null){
                        this.marker.setMap(null)
                    }
                    const markerOptions={
                        position:this.position
                    }
                    this.marker=new google.maps.Marker(markerOptions)
                    this.marker.setMap(this.map)

                    const infowindow=new google.maps.InfoWindow({
                        content: Format.twoDecimals(this.speed)+' Km/h'
                    })
                    infowindow.open(map,this.marker)
                }
            }
        }
    </script>
```

VISTA PRINCIPAL DE LA APLICACIÓN

```
<!doctype html>
<html lang="{{app()->getLocale()}}">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="csrf-token" content="{{csrf_token()}}">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <title>
            GPS Bus
        </title>
        <link type="text/css" rel="stylesheet" href="{{mix('css/app.css')}}">
        <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDuH6NYHe4qRhhxdV
0iwfoHPAjF_zxtLv8"></script>
    </head>
    <body>
        <div id="app">
            <app/>
        </div>
        <script src="{{mix('js/app.js')}}"></script>
    </body>
</html>
```

ESTRUCTURA DE LA BASE DE DATOS

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateLocationsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('locations', function (Blueprint $table) {
            $table->increments('id');
            $table->decimal('latitude',10,7);
```

```
            $table->float('longitude',10,7);
            $table->float('speed');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('locations');
    }
}
```

API DEL SERVIDOR PARA ACCESO A LAS UBICACIONES

```
<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Location;

class LocationsController extends Controller
{
    function create(Request $request)
    {
        Location::create([
            'latitude'=>$request->latitude,
            'longitude'=>$request->longitude,
            'speed'=>$request->speed,
        ]);
    }
    function last()
    {
        $location=Location::orderBy('id','desc')->first();
        return response()->json($location);
    }
}
```