Neiva, 13 de diciembre de 2018

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Diego Armando Gaona Cuenca, con C.C. No. 1082215652,

Jorge Leonardo Cuellar Dussán, con C.C. No. 1075258538,

autor(es) de la tesis y/o trabajo de grado titulado Prototipo de monitoreo remoto y alerta temprana por GPRS a un vehículo presentado y aprobado en el año 2018 como requisito para optar al título de Ingeniero electrónico;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

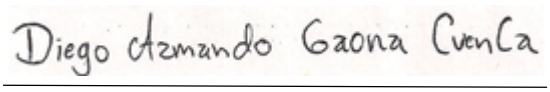
- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

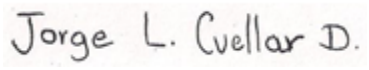
- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE: Diego Armando Gaona Cuenca

Firma: 

EL AUTOR/ESTUDIANTE: Jorge Leonardo Cuellar Dussán

Firma: 

TÍTULO COMPLETO DEL TRABAJO: Prototipo de monitoreo remoto y alerta temprana por GPRS a un vehículo.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Gaona Cuenca	Diego Armando
Cuellar Dussán	Jorge Leonardo

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Quintero Polanco	Jesús David

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico.

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería electrónica.

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2018 **NÚMERO DE PÁGINAS:** 77

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas__X_ Fotografías_X_ Grabaciones en discos___ Ilustraciones en general_X__ Grabados___
Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___
Tablas o Cuadros__X_

SOFTWARE requerido y/o especializado para la lectura del documento: Lector de PDF

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés
1.	Android	Android
2.	Base de datos	DataBase
3.	Cámara	Camera
4.	Enlace remoto	Remote link
5.	Heroku	Heroku
6.	Python	Python
7.	Raspberry pi	Raspberry pi
8.	Sensor	Sensor
9.	Servidor	Server
10.	Ubicación	Location

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Se implementó un prototipo de monitoreo remoto y alerta temprana el cual brindará un apoyo al propietario del vehículo en la seguridad del mismo. Con la ayuda de dispositivos electrónicos como lo son el GPS ublox que se encarga de la ubicación o posición del vehículo, la cámara cuya función es tomar fotografías dentro del interior del vehículo y hacer el streaming para su posterior observación, un sensor de movimiento que detecta el intruso y este a su vez está integrado a un punto de mando como lo es la raspberry Pi, la cual es la central inteligente del sistema. La respectiva programación de la tarjeta Raspberry Pi se realizó con lenguaje de alto nivel, Python.

Mediante el desarrollo de una aplicación móvil se logra conocer el momento en que se hace una apertura no autorizada del vehículo, enviando mensajes de correo electrónico como mensaje de alerta y una foto de quien se encuentre en el volante, también si se

desea desde la aplicación móvil se puede ver el streaming de lo que pasa al interior del vehículo, también se hace seguimiento por GPS de la posición del mismo. A demás la programación de la base de datos se realizó en PostgreSQL. La programación para el dispositivo móvil se realiza en Android SDK. El prototipo propuesto se cree que establece las bases para obtener un producto final competitivo dentro del mercado relacionado con los sistemas de seguridad vehicular.

ABSTRACT: (Máximo 250 palabras)

A prototype of remote monitoring and early warning was implemented which will provide support to the owner of the vehicle in its safety. With the help of electronic devices such as the ublox GPS that is responsible for the location or position of the vehicle, the camera whose function is to take photographs inside the vehicle interior and make the streaming for later observation, a motion sensor that detects the intruder and this in turn is integrated into a command point such as raspberry Pi, which is the intelligent central system. The respective programming of the Raspberry Pi card was made with high level language, Python.

Through the development of a mobile application it is possible to know the moment in which an unauthorized opening of the vehicle is made, sending emails as an alert message and a photo of whoever is on the steering wheel, also if desired from the application mobile you can see the streaming of what happens inside the vehicle, also GPS tracking of the position of the same. In addition, the programming of the database was done in PostgreSQL. The programming for the mobile device is done in Android SDK. The proposed prototype is believed to establish the basis for obtaining a competitive final product in the market related to vehicle safety systems.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Jesús David Quintero Polanco

Firma: _____

Nombre Jurado: Albeiro Cortés Cabezas

Firma: 

Nombre Jurado: Julián Adolfo Ramírez Gutiérrez

Firma: 

PROTOTIPO DE MONITOREO REMOTO Y ALERTA TEMPRANA POR GPRS A UN VEHÍCULO

JORGE LEONARDO CUELLAR DUSSÁN
COD: 2009287939
DIEGO ARMANDO GAONA CUENCA
COD: 2010296654

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA (HUILA)
2018

PROTOTIPO DE MONITOREO REMOTO Y ALERTA TEMPRANA POR GPRS A UN VEHÍCULO

JORGE LEONARDO CUELLAR DUSSÁN
COD: 2009287939
DIEGO ARMANDO GAONA CUENCA
COD: 2010296654

Proyecto de grado presentado como requisito para optar al título de INGENIERO ELECTRÓNICO

Asesor
JESUS DAVID QUINTERO
Docente Programa Ingeniería Electrónica

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA (HUILA)
2018

NOTA DE ACEPTACIÓN

_____
_____
_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Neiva, Mayo del 2018

DEDICATORIA

A DIOS por haberme permitido lograr con satisfacción mis objetivos propuestos, además de su infinita bondad y amor.
A mis padres María del Carmen y Alfredo por su constante e incondicional apoyo y motivación.
A mis cuatros hermanos por siempre confiar en mí y siempre alentarme para seguir adelante; y demás personas y amigos que pusieron su granito de arena para culminar con éxitos este proyecto.

DIEGO ARMANDO GAONA CUENCA

A mis padres y hermana, por su constante e incondicional apoyo durante todo el tiempo en el que estuve luchando para alcanzar mis metas;
A mis familiares y amigos por apoyarme y darme ánimo para levantarme y seguir luchando por mis sueños;
y demás personas que de alguna u otra forma ayudaron para la culminación de este proyecto.

JORGE LEONARDO CUELLAR DUSSÁN

AGRADECIMIENTOS

Los autores de este proyecto expresan su agradecimiento a:

A la Universidad Surcolombiana, la cual está conformada por unos excelentes docentes y excepcional personal administrativo, que nos brindaron sus servicios y experiencia, no solo para la realización de nuestros estudios de pregrados, también para la vida diaria y la profesional que nos espera.

Ing. Jesús David Quintero Polanco, director de trabajo de grado, por brindarnos su tiempo, apoyo, asesoría, además de las ideas que nos aportó y que nos facilitaron la elaboración de este proyecto.

Nuestros jurados encargados: Ing. Albeiro Cortés Cabezas, Ing. Julián Adolfo Ramírez Gutiérrez; Les agradecemos su digna labor y le damos nuestro reconocimiento por fomentar la educación con tanta pasión, inculcando los mejores valores, sembrando el conocimiento y formando a buenos profesionales.

Nuestros familiares y amigos por la confianza y fe depositada, entusiasmo que nos brindaron además de su colaboración y acompañamiento durante este estimulante proceso para poder lograr la meta de realizar este proyecto de grado.

A todas las persona que directa o indirectamente colaboraron en la realización de este proyecto de grado.

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE CUADROS

LISTA DE ANEXOS

Pág.

GLOSARIO

API: Interfaz de programación de aplicaciones (Application Programming interface). Conjunto de funciones y procedimientos que poseen muchas funcionen con el objetivo de ser utilizadas por otro software.

ARM: Es una arquitectura RISC (Reduced Instruction Set Computer) de originalmente de 32 hoy en día también disponible en 64 bits desarrollada por la compañía ARM Holdings. Se usan en computadores personales y dispositivos móviles que maneja un sistema de instrucciones realmente simple lo que le permite ejecutar tareas con un mínimo consumo de energía.

BUS I2C: Estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos. Permite el intercambio de información a una velocidad aceptable.

BUS SPI: El bus de interfaz de periféricos serie es un estándar que sirve para controlar cualquier dispositivo digital que acepte un flujo de bits serie que es regulado por un reloj.

GPIO: (General Purpose Input/Output, Entrada/Salida de Propósito General). Es un puerto que sirve a la Raspberry Pi para poder establecer comunicación con dispositivos externos.

GPRS: (General Packet Radio Service). Es una extensión de la tecnología de comunicaciones móviles GSM. En ella la información es dividida en pequeños bloques, los que posteriormente se reagrupan al llegar a destino. Este tipo de transmisión permite una mayor capacidad y velocidad.

GPU: Unidad de procesamiento de gráficos. Se encarga de procesar todos los gráficos que utiliza el sistema en el cual esta implementado.

SoC: (System on a chip). Es una tendencia la cual busca la integración de todos los módulos de un sistema en un solo chip o circuito, lo cual reduce en grandes proporciones el tamaño del dispositivo que lo incluye.

UART: (Universal Asynchronous Receiver-Transmitter). Dispositivo de ciertos sistemas digitales cuya misión principal es convertir los datos recibidos en forma paralela a forma serial, con el objetivo de comunicarse con otro sistema externo. También realiza el proceso contrario.

RESUMEN

Se implementó un prototipo de monitoreo remoto y alerta temprana el cual brindará un apoyo al propietario del vehículo en la seguridad del mismo. Con la ayuda de dispositivos electrónicos como lo son el GPS ublox, la cámara, un sensor de movimiento integrado a un punto de mando como lo es la raspberry PI, se logra conocer a través del IOT el momento en que se hace una apertura no autorizada del vehículo, enviando mensajes y una foto de quien se encuentre en el volante y hacer seguimiento por GPRS de la posición del mismo. Toda la programación se realizó en Python y la base de datos en PostgreSQL. La programación para el dispositivo móvil se realiza en Android SDK.

ABSTRACT

A prototype of remote monitoring and early warning is implemented that has support for the owner of the vehicle in its safety. With the help of electronic devices such as the ublox GPS, the ip camera, a motion sensor integrated to a command point such as the raspberry PI, it was possible to know through IOT the moment in which an opening is made. Authorized vehicle, sending messages and a photo of who is in the steering wheel and track by GPRS the position of the same. All the programming was done in Python and the database in PostgreSQL. The programming for the mobile device is done in Android SDK.

INTRODUCCIÓN

El desarrollo de éste proyecto, se hace a partir de la necesidad de tener un sistema o mecanismo adecuado de procesamiento y monitoreo de datos obtenidos por los dispositivos (sensor de movimiento, GPS, cámara) instalados en un vehículo, para proporcionar a los usuarios una interfaz (app) que posea un entorno agradable, confiable y seguro.

En la actualidad, el robo de vehículos esta disparado ya que las bandas delincuenciales se dedican a vender las partes de los automóviles robados, ya sea dentro o fuera del país, [1] esto se debe a que hay una gran demanda de repuestos de automóviles, además no hay un control eficaz a la hora de manejar la procedencia de dichos repuestos que muchas veces son de automóviles robados.

El robo de vehículos en Colombia se ha convertido en una poderosa industria, que para el año 2016 dejó una pérdida de alrededor de 680 millones de dólares y para el año 2017 creció a 685 millones de dólares. A su alrededor se han montado sofisticadas organizaciones de atracadores y asaltantes que, según las aseguradoras, configuran la tercera fuerza delictiva, después del narcotráfico y la guerrilla. Las cifras son alarmantes. En el año 2017 se robaron 34.800 vehículos en todo el país y para el 2018 se estima un crecimiento hasta de un 50%.[2] Este delito se está generalizando y nadie está exento de ser víctima de las bandas de jaladores de carros que cada vez actúan con mayor agresividad y descaro. Se calcula que a diario, en el país son hurtados 5 vehículos cada hora, siendo la capital del país una de las más afectadas, ya que acapara el mayor número de casos. Además, los conductores dejan muy descuidados los vehículos en la vía pública, de modo que todo esto incide para que se roben más vehículos.[3]

Existen algunos fabricantes que ofrecen el servicio de vigilancia vehicular, que muestran la ubicación del carro por medio de un GPS, sin saber quién o quiénes son las personas que hurta el vehículo. En este proyecto se diseñó e implementó una aplicación móvil en la cual se visualiza la imagen de quien realice una apertura no autorizada del vehículo, posición en tiempo real del mismo de forma ordenada y concisa, para prevenir posibles casos de hurto o recuperación del vehículo. El objetivo principal de este proyecto es crear perfiles de las personas que se dedican a estos actos criminales, para que las autoridades competentes se pueden encargar de capturarlos y poco a poco desmantelar estas peligrosas organizaciones.

Por otra parte, es necesario resaltar que este proyecto tendrá un óptimo funcionamiento con una buena cobertura de internet, ya que de esta dependerá

la rapidez con la que cual se transmitirá la información desde el vehículo al usuario; se debe aclarar que en zonas donde no haya cobertura el vehículo estará expuesto, debido a que el sistema estará inoperativo y no habrá manera de enviar la información.

1.https://www.semana.com/nacion/articulo/robo-y-venta-de-vehiculos-en-colombia-y-latinoamerica/568305

2.https://noticias.canalrcn.com/invitados/esta-incrementando-el-robo-carros-colombia

3.https://noticias.caracoltv.com/bogota/ladrones-ahora-secuestran-carros-y-exigen-rescate-cambio-de-no-desguazarlos-ie26636

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Implementar un prototipo de monitoreo remoto y alerta temprana el cual brindará un apoyo al propietario del vehículo en la seguridad del mismo.

1.2 OBJETIVOS ESPECÍFICOS

- Implementar una alerta de advertencia temprana para la puerta del conductor (opcional para el resto de puertas) para cada vez que se abra envíe una alarma al móvil del propietario.

- Acondicionar el funcionamiento de la(s) cámara(s) IP mediante un control on/off, que se activará manualmente o por medio de la aplicación.

- Diseñar e implementar una interfaz (aplicación web y/o móvil) la cual permita al usuario un sencillo y seguro monitoreo de su vehículo, además almacenar la información adquirida en una base de datos mediante dicha aplicación.

- Implementar un sistema de transmisión que permita un buen envío y recepción de la información adquirida.

2. MARCO TEÓRICO

2.1 DIAGRAMA GENERAL DEL SISTEMA

Además de la alarma audible que posee el vehículo la cual permite informar al propietario de que su auto posiblemente está siendo accesado, también posee un sistema de monitoreo sofisticado el cual permite obtener la ubicación del automóvil en tiempo real. El sistema posee una cámara la cual permitirá obtener un registro visual de los hechos que podrá ser visto en tiempo real en teléfono celular con plataforma Android mediante una aplicación “Apps”.

Los componentes del sistema de monitoreo son: la placa RASPBERRY PI 3 a la cual va conectada el sensor de movimiento, el módulo GPS Ublox NEO-6M y la cámara IP. El sistema también posee su propia alimentación, en este caso la provee la batería del automóvil mediante una adecuación del puerto de mechero.

En la figura 1 se observa el diagrama general del sistema de monitoreo instalado en el automóvil con todos sus respectivos componentes.

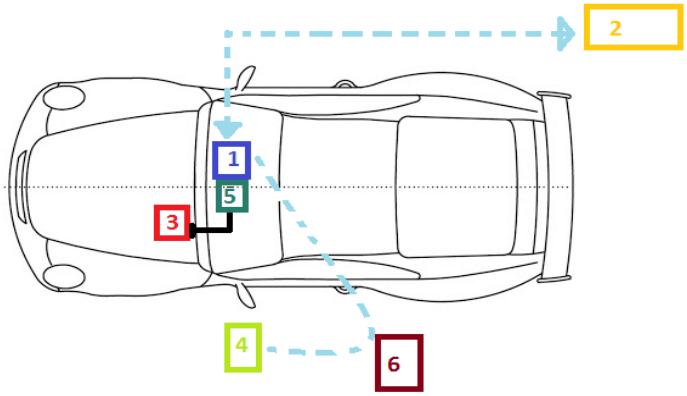


Figura 1. Diagrama general del sistema

Fuente: Autores

1. Raspberry Pi 3 + GPS Ublox NEO-6M
2. Satélite GPS
3. Mecanismo de alimentación dual 5V/12V
4. Dispositivo móvil Android
5. Sensor de movimiento/Cámara(s)
6. Red

2.1.1 Mecanismo de alimentación dual 5V/12V. Se implementa a partir de la batería del automóvil, que es la fuente de alimentación de donde se suministrará energía eléctrica al sistema de seguridad a través de la adecuación del puerto de mechero.

2.1.2 Raspberry Pi 3 + GPS Ublox NEO-6M. El módulo GPS Ublox NEO-6M y el sensor de movimiento están conectados a la tarjeta inteligente, raspberry Pi 3, la cual es la central inteligente del sistema, se encarga de sincronizar el funcionamiento del sistema de monitoreo con el dispositivo móvil. Esta tarjeta posee su propia antena de Wifi lo cual nos permite el envío y/o recepción de datos a la nube.

2.1.3 Sensor de movimiento (HC-SR501). Es un sensor piroeléctrico (pasivo) infrarrojo, posee un lente de Fresnel, (ver figura 2) cuya forma es un encapsulado semiesférico hecho de polietileno de alta densidad cuyo objetivo es permitir el paso de la radiación infrarroja en el rango de los 8 y 14 micrones. El lente detecta radiación en un ángulo con apertura de 110° y, adicionalmente, la energía se concentra en la superficie de detección del sensor PIR, permitiendo una mayor sensibilidad del dispositivo. El sensor PIR consta en realidad de 2 elementos detectores separados, siendo la señal diferencial entre ambos la que permite activar la alarma de movimiento. En el caso del HC-SR501, la señal generada por el sensor ingresa al circuito integrado BISS0001, el cual contiene amplificadores operacionales e interfaces electrónicas adicionales[4].

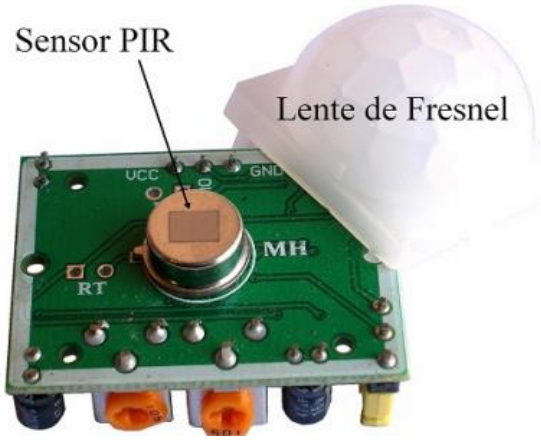


Figura 2. HC-SR501 PIR sensor infrarrojo de movimiento

Fuente: https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf

En los sensores de movimiento, el sensor PIR posee 2 elementos detectores separados, siendo la señal diferencial entre ambos la que permite activar la alarma de movimiento. En el caso del HC-SR501, la señal generada por el sensor ingresa al circuito integrado BISS0001, el cual contiene amplificadores operacionales e interfaces electrónicas adicionales.

El módulo PIR modelo HC-SR501 es de bajo costo, pequeño, e incorpora la tecnología más reciente en sensores de movimiento. El sensor utiliza 2

potenciómetros y un jumper que permiten modificar sus parámetros y adaptarlo a las necesidades de la aplicación: sensibilidad de detección, tiempo de activación, y respuesta ante detecciones repetitivas. Sus especificaciones técnicas son:

- Voltaje de alimentación: de 5 a 12 VDC
- Usa el PIR LHI778 y el controlador BISS0001
- Consumo promedio: <1 miliamperio
- Rango de distancia de 3 a 7 metros ajustable (ver figura 3).

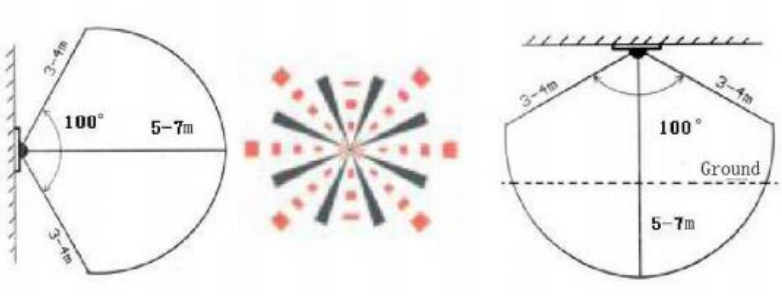


Figura 3. Rango variable del sensor PIR

Fuente: http://www.electronicoscaldas.com/datasheet/HC-SR501.pdf

- Ángulo de detección: Cono de 110°.

- Ajustes: Dos potenciómetros para ajuste de rango de detección y tiempo de alarma activa.

- Jumper para configurar la salida de alarma en modo mono-disparo o disparo repetitivo (‘rettrigerable’).

- Salida de alarma de movimiento con ajuste de tiempo entre 3 segundos a 5 minutos.

- Salida de alarma activa Vo con nivel alto de 3.3 voltios y 5 mA source, lista para conexión de un led o un transistor y relevador.

- Tiempo de inicialización: Después de alimentar el módulo HC-SR05, deben transcurrir 1 minuto antes de que inicie su operación normal. Durante ese tiempo, es posible que el módulo active 2 o 3 veces su salida.
- Tiempo de salida inactiva: Cada vez que la salida pase de activa a inactiva, permanecerá en ese estado los siguientes 3 segundos. Cualquier evento que ocurra durante ese lapso es ignorado.

- Temperatura de operación: -15° a +70° C.

- Dimensiones: 3.2 x 2.4 x 1.8 cm.

En la figura 4 se observan los parámetros de dicho sensor y posteriormente en el cuadro 1 se encuentran las características del mismo.

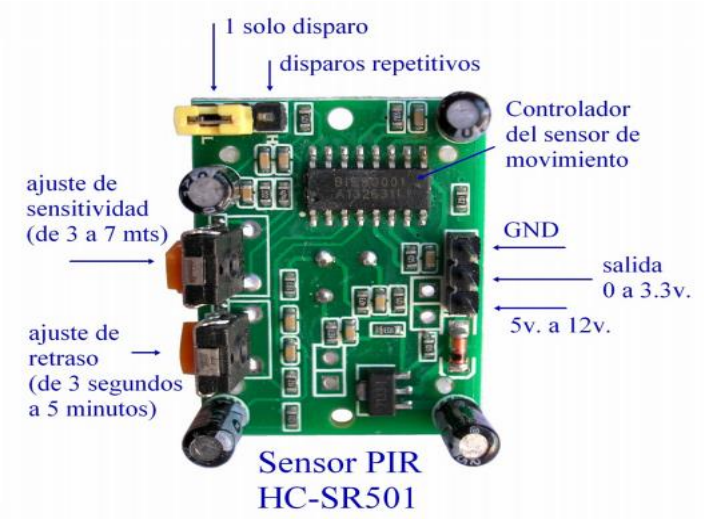


Figura 4. HC-SR501 PIR Modificación de parámetros

Fuente: https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf

Cuadro 1. Características sensor PIR

Tipo de producto	Body Sensor Module
Rango de voltaje de operación	DC 4.5-20V
Corriente de inactividad	<50uA
Nivel de salida	High 3.3 V /Low 0V
Trigger	L can not be repeated trigger/H can be repeated trigger(Default repeated trigger)
Tiempo de retardo	5-200 S(adjustable) the range is (0.xx second to tens of second)
Tiempo de bloqueo	2.5 S (default)Can be made a range(0.xx to tens of seconds)
Dimensiones de la placa	32mm*24mm
Angulo de detección	<100 ° cone angle
Temperatura de operación	-15-+70 degrees
Tamaño del lente del sensor	Diameter: 23mm(Default)

Fuente: http://www.electronicoscaldas.com/datasheet/HC-SR501.pdf

4. Referenciado por:https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf

2.1.4 Cámara(s). El sistema de monitoreo posee cámaras las cuales están conectadas a la Raspberry, y permiten tener acceso audiovisual del entorno del automóvil y ver qué pasa en el mismo. La cámara para la raspberry pi 3 (ver figura 5), está diseñado especialmente para Raspberry Pi. Se conecta en uno de los dos pequeños sockets que tiene la tarjeta Raspberry en su cara superior. Esta conexión, usa la interfaz dedicada CSI, la cual es óptima para conectar una cámara gracias a su alta capacidad de transmisión de datos.

El sensor óptico, tiene una resolución de 5 millones de pixeles (Mpx), y lleva un lente de foco fijo. La cámara es capaz de capturar imágenes de hasta 2592×1944 pixeles y soporta video en 1080p a 30 cuadros por segundo (fps).[5]

También se hace uso de una cámara IP, la cual no va conectada al sistema de monitoreo, esta es independiente y sirve como respaldo ante cualquier situación. Esta cámara posee una aplicación móvil mediante la cual se puede manipular la cámara remotamente.

Figura 5. Modulo PI, Cámara

Fuente: https://images-na.ssl-images-amazon.com/images/I/61QwH%2BZS5pL._SL1000_.jpg

2.1.5 GPS Ublox NEO-6M. Módulo GPS para raspberry y microcontroladores, basado en el receptor de la marca Ublox modelo NEO 6M (ver figura 6), incluye su antena cerámica la cual se pone directamente sobre el PCB, por lo que ya viene listo para operar sin requerir más accesorios.
El GPS para Raspberry puede funcionar con un voltaje de alimentación en el rango de 3.0 a 5.0 voltios, las señales que entran y salen son de 3.3 voltios, por lo que se requiere un convertidor de niveles si la raspberry o microcontrolador de 5 voltios va a comunicarse (transmitir) hacia el módulo GPS. Si solamente se desean recibir los datos NMEA basta con conectar el pin TX con el RX de la Raspberry y recibir los datos que envía el módulo, en este caso, no hace falta conversión de niveles por que la raspberry reconoce los 3.3 voltios como nivel alto[6].

Características:

- Comunicación serial
- Voltaje de alimentación: (3.5 – 5)VDC
- Antena cerámica activa incluida
- LED indicador de señal
- Tamaño de antena 22x22mm
- Tamaño de módulo 23x30mm
- BAUDRATE: 9600
- EEPROM para guardar configuración de parámetros
- Sistema de coordenadas: WGS-84
- Sensibilidad de captura -148 dBm
- Sensibilidad de rastreo: -161 dBm
- Máxima altura medible: 18000
- Máxima velocidad 515 m/s
- Exactitud: 1micro segundo
- Frecuencia receptora: L1 (1575.42 MHz)
- Código C/A 1.023 MHz
- Tiempo de inicio primera vez: 38s en promedio
- Tiempo de inicio : 35s en promedio

Figura 6. GPS Ublox NEO-6M

Fuente: Autores

2.1.6 Dispositivo móvil Android. Es un teléfono celular sofisticado con sistema operativo Android de software libre, además permite conectividad 3G, 3.5G y 4G. Son equipos que poseen alta velocidad de procesamiento y almacenamiento de información. Este equipo permite la conexión entre el usuario y el sistema de monitoreo del automóvil. Esto es posible ya que se diseñó una aplicación que en sí, es la interfaz que permite tener un control remoto del sistema de monitoreo.

5. https://www.raspberrypi.org/documentation/hardware/camera/
6. Referenciado por: https://electronilab.co/tienda/modulo-gps-ublox-neo-6m-v2-con-memoria-eeprom/

2.2 INTRODUCCION A RASPBERRY

Esta sección se encontrará información detallada de la tarjeta inteligente raspberry. Se describen los principales elementos que la componen, además del entorno en la cual puede ser programada, es decir, el software que pueden actuar sobre ella.[7]

2.2.1 ¿Qué es raspberry? Raspberry PI es una placa computadora (SBC) de bajo costo, se podría decir que es un ordenador de tamaño reducido. El concepto es el de un ordenador que no posee sus periféricos y aun así, no afecta el funcionamiento básico del mismo. Está formada por una placa que soporta varios componentes necesarios en un ordenador común y es capaz de comportarse como tal

.

2.2.2 Hardware. En la actualidad existen 2 modelos diferentes de raspberry Pi. El modelo A y modelo B. En nuestro caso es de sumo interés hablar solo del modelo B ya que está se utilizó para llevar a cabo este prototipo. El uso de esta tarjeta es muy amplio, permite crear elementos interactivos a partir de una serie de datos que son recolectados a través de sensores, interruptores y demás dispositivos que se pueden conectar a la tarjeta mediante los pines de entrada que proporciona la misma y dar una respuesta ante el evento sucedido mediante los pines de salida.

2.2.3 Raspberry Pi 3 modelo B. Es una de las últimas versiones que están disponibles en el mercado. Esta tarjeta electrónica posee un procesador multimedia Broadcom BCM2837. Esto quiere decir que la mayor parte de los componentes del sistema, incluidos la CPU y la GPU junto con el audio y el hardware de comunicaciones, se encuentran integrados dentro de aquel único componente oculto ubicado justo debajo del chip de la memoria de 1GB en el centro de la placa. No es sólo el diseño del SoC lo que hace al BCM2837 diferente del procesador de un PC o portátil. Lo que lo hace también diferente es que utiliza una arquitectura de conjunto de instrucciones distinta, conocida como ARM. Posee un conector de expansión de 40 pines incluyendo 27 GPIOs con UART, Bus I2C y Bus SPI. Dispone de 4 puertos USB 2.0, un conector de alimentación por puerto USB micro B o conector de pines GPIO.

Figura 7. Tarjeta Raspberry Pi 3 modelo B
Fuente: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

2.2.4 Características raspberry Pi 3 Modelo B. En el siguiente cuadro se resume las principales características de esta tarjeta inteligente:

Cuadro 2. Características Raspberry Pi 3 Modelo B

Procesador	Broadcom BCM2837
Arquitectura del núcleo	ARMv8-A
CPU	ARM Cortex-A53 a 1.2 GHz
GPU	Broadcom VideoCore IV® @ 400 MHz
Memoria	1GB SDRAM LPDDR2-900
Fuente de alimentación	Micro USB socket 5V, 2.5A
Ethernet	10/100 BaseT Ethernet socket
Bluetooth	4.1
Wifi	802.11 b/g/n
Salida de video	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)
Salida de audio	3.5mm jack, HDMI
USB 2.0	4 conectores USB
Conector GPIO	Conector de expansión de 40 pines de 2.54mm (100 mil): 2x20 pines. Incluyendo 27 GPIOs, I²C, SPI y UART, así como alimentación a 3.3 V, 5V y GND.
Conector de la cámara	5 pines MIPI Camera Serial Interface (CSI-2) (SKU: RPI-005, RPI-006)
Conector de display	Display Serial Interface (DSI) cable plano de 15 hilos con dos de datos y uno de reloj.
Ranura de tarjeta SD	Micro SD
Dimensiones	85 x 56 x 17mm

Fuente: https://fabricadigital.org/productos/raspberry-pi-3-modelo-b/

En la figura 8 se muestra las partes principales que conforman la tarjeta inteligente, Raspberry Pi 3 modelo B.

7. http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/

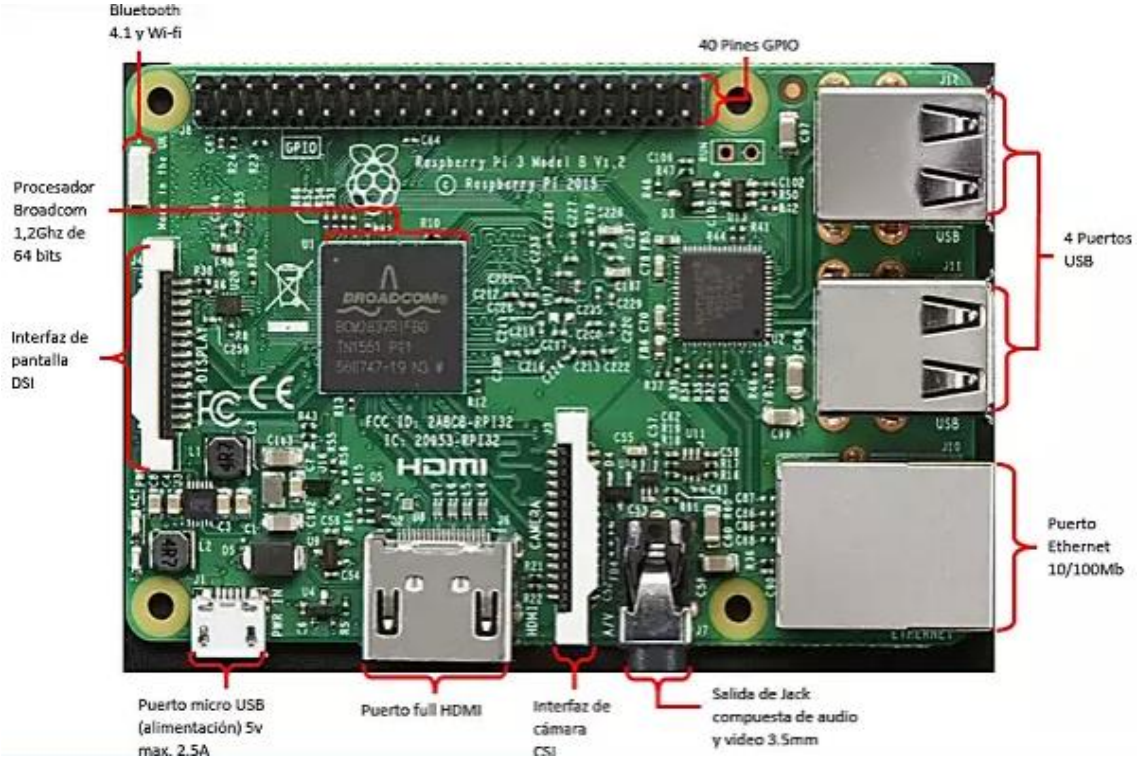


Figura 8. Partes principales de la Raspberry Pi 3 modelo B

Fuente: http://learn.teslabem.com/raspberry-pi-3-pantalla-touch-hdmi-5/3/

A continuación se dará una breve descripción de la funcionalidad de algunos de los pines GPIO.[8]

- Los pines 8 y 10, se pueden configurar como interfaz UART para un puerto serie convencional.

- Los pines 3 y 5 pueden configurarse como interfaz I2C, esto permite que pueda interactuar con periféricos que siguen este protocolo.

- El pin 12 puede configurarse como PWM (modulación por ancho de pulso).

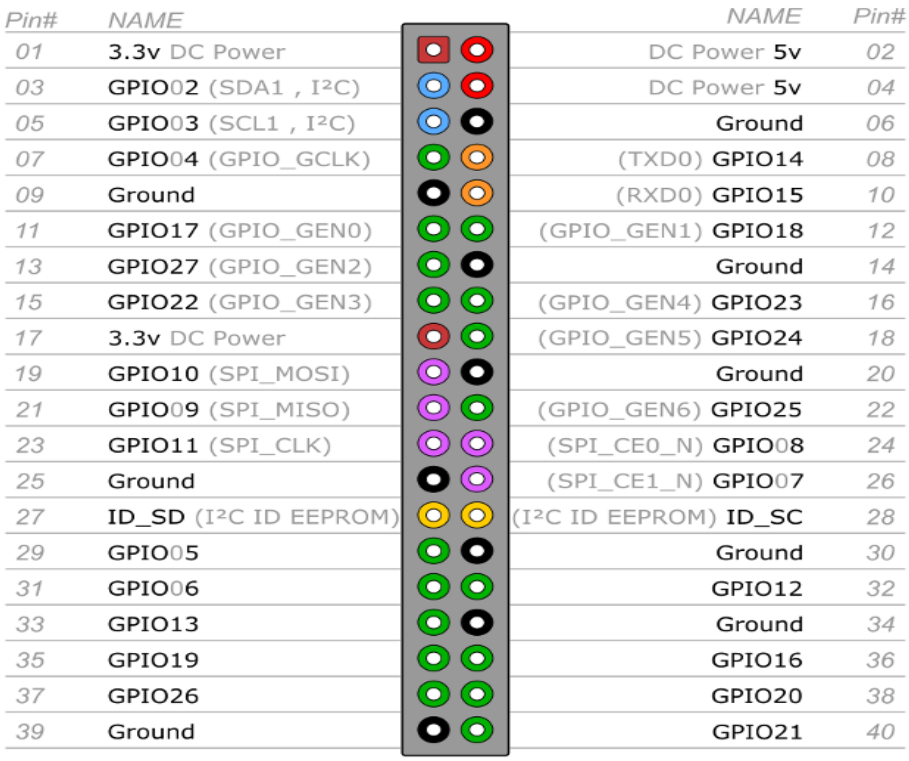
- Los pines 19, 21, 23, 24 y 26 se pueden configurar como la primera interfaz SPI (SPI0) para interactuar con periféricos que siguen este protocolo.

- Los pines 27 y 28 no están disponibles. Estos son los únicos pines que en el arranque se configuran como salidas, el resto inicialmente como entradas para evitar problemas.

- Los pines 29, 31, 32, 33, 35, 36, 37, 38 y 40 proporcionan acceso a nuevas pines de GPIO que no estaban disponibles en los modelos originales. Estos pines pueden tener otros usos adicionales. Por ejemplo los pines 32, 33 y 35 pueden utilizarse para salidas PWM (solo dos

canales disponibles). Además estos pines completan los pines necesarios para configurar otra interfaz SPI (SPI1).

En la figura 9 se puede observar de manera detalla la distribución de los pines de la Raspberry Pi 3 modelo B.



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

Figura 9. Distribución de los pines de la Raspberry Pi 3 modelo B

Fuente: https://cenaptec.com/2017/05/18/raspberry-pi-gpio-circuitos-led/

2.2.5 Software (entorno de desarrollo). La Raspberry pi 3 modelo B puede soportar diferentes lenguajes de programación como Python, Java, C++, C, PHP, etc. Cada uno de estos lenguajes posee determinadas características, lo cual hace que el uso de cada uno depende del tipo de aplicación que se pretenda desarrollar.

Estos lenguajes soportan la programación orientada a objetos, orientada a eventos y orientada a aspectos. El lenguaje de programación que se ajusta a las necesidades, factibilidad y fiabilidad es Python, por ende se hace una descripción a continuación: [9]

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de este un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

8. Referenciado por: https://franciscomoya.gitbooks.io/taller-de-raspberry-pi/content/es/elems/gpio.html

El Python nos permite separar el programa en módulos; este lenguaje tiene una gran variedad de módulos estándar que se pueden utilizar para programar, o incluso como una base para aprender a programar en Python.[10]

2.3 BASE DE DATOS

Cuando el sistema de monitoreo se activa, toda la información que proporciona este (coordenadas del GPS y fotos) debe ser procesada transmitida y almacenada en la nube, aquí es donde es necesario crear una base de datos capaz de almacenar toda esta información y que nos proporcione seguridad de la misma, cabe notar que la base de datos será alojada en un servidor gratuito.

El concepto de base de datos se refiere a un conjunto de datos o información que pertenecen a un mismo contexto que se almacenan sistemáticamente para su posterior uso. Hay diferentes modelos de bases de datos que permiten ordenar y organizar la información para poder acceder de manera sencilla a ella. A continuación, se mencionan los respectivos modelos.[11]

- **Bases de datos jerárquicas:** La estructura de los datos son muy estables cuando se gestiona gran cantidad de datos muy interrelacionados.

- **Bases de datos en red:** Nacen de las jerárquicas, su característica principal es que mejoran la gestión de datos reiterativos manteniendo su rendimiento en consulta de datos.

- **Bases de datos transaccionales:** Se diseñan con un único fin, el cual es el envío y recepción de información a altas velocidades de forma constante.

- **Bases de datos relacionales:** La información se almacena siempre haciendo referencia a otra por lo que se facilita la gestión y su uso por personal no especialista. En este modelo el lugar y la forma donde se guarde la información es secundario.

- **Base de datos orientada a objetos:** Está orientada a los sistemas de programación orientada a objetos.

- **Base de datos documentales:** Almacenan textos completos, esto permite el tratamiento informatizado de grandes cadenas de caracteres.

9. http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf
10. Referenciado por: http://www.larevistainformatica.com/Python.htm
11. Referenciado por: http://www.ymant.com/tipos-base-datos/

2.3.1 Sistemas gestores de base de datos. Un sistema gestor de bases de datos (SGBD, en ingles DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación[12].

Un SGBD relacional es muy efectivo porque permite a varios usuarios acceder a los datos al mismo tiempo, brindando facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos (la base de datos que se creó para este proyecto es de este tipo de gestor de base de datos).

A continuación, se mencionan SGBD más usados en la actualidad:

- **MySQL:** Es un sistema gestor de base de datos de acceso libre, es decir, de código abierto con licencia comercial disponible. Gestiona datos relacionales de manera rápida, sólida y flexible. Se recomienda para la creación de base de datos con acceso desde páginas web dinámicas. También se utiliza para la creación de cualquier otra solución que implique el almacenamiento de datos, posibilitando realizar múltiples y rápidas consultas.

- **Microsoft SQL Server:** Es un sistema gestor de base de datos con licencia de pago. Gestiona datos relacionales, proporciona integridad de datos, además posee optimización de consultas, control de concurrencia, backup y recuperación.

- **PostgreSQL:** Es un sistema gestor de base de datos relacionales orientada a objetos, es de código abierto, es decir, de acceso libre. Proporciona un control de concurrencia multiversión que permite trabajar con grandes cantidad de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación.

  Posee una integridad referencial e interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows.

  La base de datos del proyecto fue desarrollada con el SGBD PostgreSQL mediante la interfaz gráfica pgAdmin III; es una herramienta de código abierto, es la más utilizada comúnmente para administrar las instancias de PostgreSQL, es tanto que es considerada como la interfaz gráfica oficial de este SGBD. La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor.[13]

12. Referenciado por: https://www.ecured.cu/Sistema_Gestor_de_Base_de_Datos
13. Referenciado por: https://www.ecured.cu/PgAdmin3

En la figura 10 se puede ver la base de datos ya elaborada. En la parte izquierda inferior de la figura se puede ver las respectivas tablas.

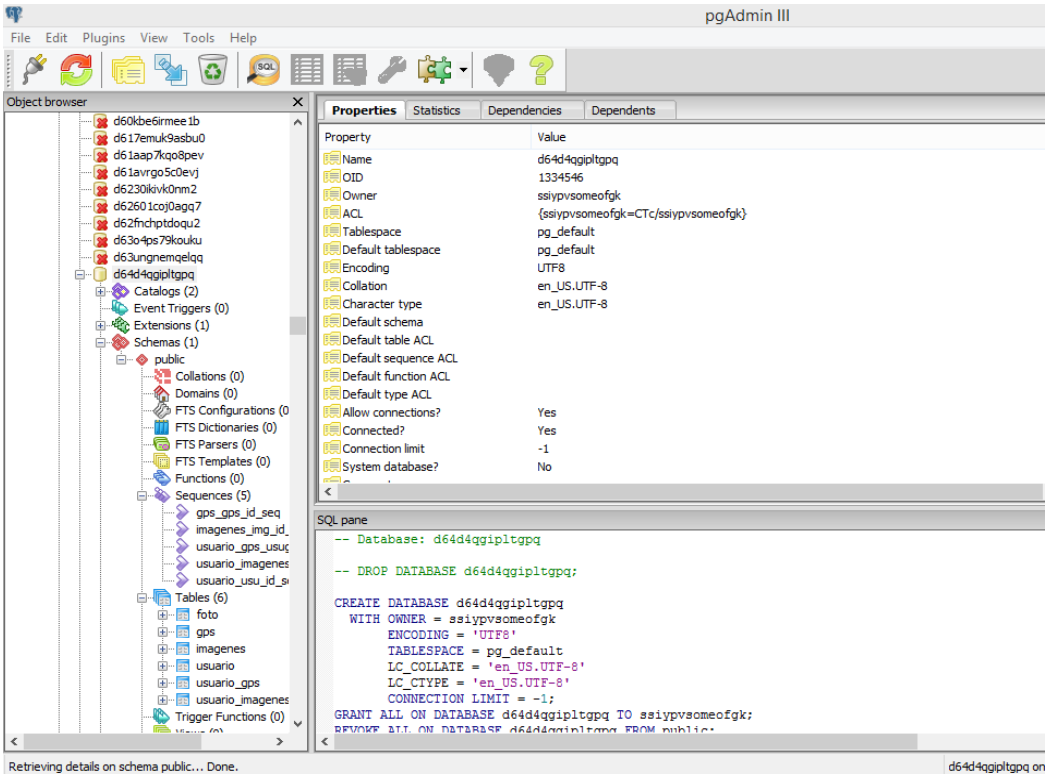


Figura 10. Base de datos ya elaborada en pgAdmin III

Fuente: Autores

2.3.2 Plataforma Heroku. Es uno de los PaaS (Platform as Service) más utilizados en la actualidad. Es una plataforma en la nube que permite crear, distribuir, monitorear y escalar aplicaciones en entornos empresariales. Soporta lenguajes de programación como Ruby, Java, Node.js, PHP, Python, Go, Scala y Clojure.

Se sabe que cualquier aplicación importante maneja gran cantidad de datos, ya sean datos de clientes o datos sobre servicio que brinda. El amplio y nutrido ecosistema de servicios de Heroku incluye Heroku Postgres, un servicio de base de datos que viene con experiencia operacional, integrado. Los desarrolladores poseen acceso inmediato a una base de datos escalable y de alta disponibilidad con reversión, una que admite sus aplicaciones y su estilo de desarrollo.

Heroku Postgres permite maximizar los datos en lugar de perder tiempo en la configuración y el mantenimiento de la base de datos. Puede probar nuevas migraciones de esquema, gestionar los niveles de acceso a la base de datos y proteger las consultas, escalar horizontalmente y permitir que su equipo acceda rápidamente a los datos.
Ya sea que se trate de funciones como Continuous Protection o la aplicación perfecta del parche de seguridad más reciente. [14]

Como ya se había mencionado anteriormente, la base de datos fue diseñada mediante la interfaz gráfica pgAgmind III, la cual es posible conectar a la nube mediante Heroku Postgres.

2.4 ANDROID

Android es un sistema operativo de desarrollo libre basada en Linux y de código abierto. Unas de sus grandes ventajas es que se puede usar "customizar" el sistema sin pagar. Las aplicaciones finales son desarrolladas en Java, lo que asegura que podrán ser ejecutadas en una gran variedad de dispositivos, esto se consigue gracias al concepto de máquina virtual.

Android ofrece una gran cantidad de servicios incorporados, por ejemplo, localización basada tanto en GPS como en redes, base de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia, etc. En pocas palabras, Android nos ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para dispositivos móviles de pantalla táctil, en este caso los smartphones y tablets. [15]

Las plataformas que se pueden usar para desarrollar las aplicaciones en entorno Android son la plataforma Eclipse, que permite desarrollo de código abierto basado en Java, y la plataforma Android Studio que proporciona un entorno de desarrollo integrado (IDE), es gratuita y se puede descargar desde su página oficial. El desarrollo de la aplicación de este proyecto se elaboró mediante la plataforma Android Studio.

2.4.1 Arquitectura de Android. Está formada por cuatro capas, cada una está basada en software libre. En la figura 11 se puede observar la arquitectura Android.

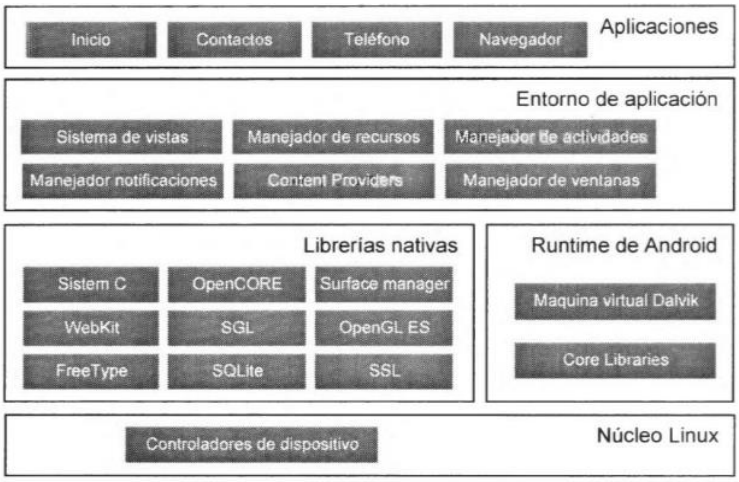


Figura 11. Arquitectura de Android

Fuente: El gran libro de Android. Jesús Tomás Gironés.

14. Referenciado por: https://www.heroku.com/postgres
15. GIRONÉS, Jesús Tomás. El gran libro de Android.

- Núcleo Linux: El núcleo de Android está formado por el sistema operativo Linux. Esta capa proporciona servicios como manejo de la seguridad, el manejo de la memoria el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. También actúa como capa de abstracción entre el hardware y el resto de la pila.

- Librerías nativas y el Runtime de Android: Esta capa contiene un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en el código nativo del procesador. El concepto de Runtime se basa en el concepto de máquina virtual utilizado en Java. Google creo la máquina virtual Dalvik la cual optimiza los recursos.

- Entorno de aplicación: Esta capa se diseñó con el fin de simplificar la reutilización de componentes. Una de las fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación de Java. Además proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones.

- Aplicaciones: Esta última capa la conforman el conjunto de aplicaciones instaladas en la máquina virtual Dalvik, esto garantiza la seguridad del sistema. Por lo general las aplicaciones Android están desarrolladas en Java, lo cual podemos utilizar el Android SDK.

2.4.2 Versiones de Android y niveles de API: Cuando se desea iniciar un proyecto en Android, primero se debe elegir la versión del sistema para la que se desea realizar la aplicación. Es conveniente detallar que hay clases y métodos que están disponibles a partir de una versión. En grado tal que se desee usar se debe conocer la versión mínima necesaria.

Al lanzar una nueva versión de la plataforma esta siempre será compatible con las versiones anteriores. Es decir, solo añaden nuevas funcionalidades y, en el caso de que se modifique alguna funcionalidad, esta no se elimina, solo se etiquetan como obsoletas pero su pueden seguir usando.

Las plataformas lanzadas hasta la fecha se identifican de tres formas alternativas: versión, nivel de API y nombre. El nivel de API corresponde a números enteros comenzando desde 1. Para los nombres se han elegido postres en orden alfabético Cupcake (v1.5), Donut (v1.6), Éclair (v2.1), Froyo (2.2), Gingerbread (v2.3), Honeycomb (v3.0), Ice cream (v2.4), etc.

2.4.3 Elección de la plataforma de desarrollo: El primer paso a seguir es consultar si se necesita alguna característica en especial que solo esté disponible a partir de una versión. Todos los usuarios con versiones inferiores a la seleccionada no podrán instalar la aplicación. Por ende, se recomienda seleccionar la menor versión posible que la aplicación pueda soportar.

2.4.4 Componentes de una aplicación: Para desarrollar aplicaciones en Android existen una serie de elementos claves que son imprescindibles para este proceso. A continuación se describirán brevemente los elementos más importantes:

- Vista (view): Son elementos que componen la interfaz de usuario de una aplicación. Son por ejemplo, un botón o una entrada de texto. Todas las vistas van a ser objetos descendientes de la clase View y, por tanto, pueden ser definidos utilizando código Java. Sin embargo, lo habitual va a ser definir las vistas utilizando un fichero XML y dejar que el sistema cree los objetos por a partir de este fichero.

- Layout: Un Layout es un conjunto de vistas agrupadas de una determinada forma. Hay diferentes tipos de Layouts para organizar las vistas como son de forma lineal, en cuadrícula o indicando la posición absoluta de cada vista. Los Layouts también son objetos descendientes de la clase View. Igual que las vistas los Layouts pueden ser definidos en código, aunque la forma habitual de definirlos es utilizando código XML.

- Actividad (Activity): Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, coloquialmente conocidos como pantallas de la aplicación. En Android cada uno de estos elementos, o pantallas, se conoce como actividad. Su función principal es la creación del interfaz de usuario. Una aplicación suele necesitar varias actividades para crear el interfaz de usuario. Toda actividad ha de pertenecer a una clase descendiente de Activity.

- Servicio (Service): Un servicio es un proceso que se ejecuta "detrás", sin necesidad de una interacción con el usuario. Es algo parecido a un demonio en Unix o a un servicio en Windows. En Android se dispone de dos tipos de servicios: servicios locales, que pueden ser utilizados por aplicaciones del mismo terminal y servicios remotos, que pueden ser utilizados desde otros terminales.

- Intención (Intent): Representa la voluntad de llevar a cabo una acción, como por ejemplo, realizar una llamada de teléfono, visualizar una página web. En si se utiliza cada vez que se quiera lanzar una actividad, un servicio, un anuncio de tipo broadcast y comunicarnos con un servicio. Los componentes pueden ser internos o externos a la aplicación. Las intenciones también serán usadas para el intercambio de información entre estos componentes. En muchas ocasiones una intención no será inicializada por la aplicación, si no por el sistema, por ejemplo, cuando se pide visualizar una página web. En otras ocasiones es necesario que la aplicación inicialice su propia intención. Para ello se crea un objeto de la clase Intent.

- Receptor de anuncios (Broadcast receiver): Recibe y reacciona ante anuncios de tipo broadcast. Existen muchos originados por el sistema; como por ejemplo Batería baja, llamada entrante. Aunque, las

aplicaciones también pueden lanzar un anuncio broadcast, no tienen interfaz de usuario, aunque pueden iniciar una actividad para atender a un anuncio.

- Proveedores de contenido (Content Provider): Compartir información entre teléfonos móviles es un tema de gran importancia. Android define el mecanismo estándar para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros. De esta forma se podrá acceder a datos de otras aplicaciones, por ejemplo, la lista de contactos, o proporcionar datos a otras aplicaciones. [16]

2.5 ESTADO DEL ARTE DEL PROTOTIPO

Un prototipo es un sistema que interactúa con varios elementos entre sí para llevar a cabo una función para la cual ha sido diseñado. Un prototipo es el primer sistema o dispositivo que se elabora y del cual se adquieren las ideas más relevantes para la construcción de otros diseños, y representan algunas de las ideas en cuanto a diseño, soporte y tecnología que puedan implementar los diseñadores.

Actualmente, el hurto de automóviles se ha ido incrementando de una manera considerable, por lo cual se han desarrollado prototipos de seguridad con el fin de reducir ésta problemática. A continuación se mencionan algunos proyectos que se han elaborado en esta área. Cabe resaltar que estos proyectos se utilizaron como precedentes para llevar a cabo el proyecto que se ha planteado en este documento.

Bedoya, Salazar y muñoz (2013), llevaron a cabo la “Implementación, control y monitoreo de un sistema de seguridad vehicular por redes GSM/GPRS”. Este proyecto plantea implementar un prototipo de telemetría, control y monitoreo de un sistema de seguridad para vehículos, usando las redes móviles como medio de comunicación. Además de la seguridad del automóvil, este sistema se encarga también del estado del mismo, es decir, proporciona información de variables físicas como la temperatura, niveles de gasolina o aceite. Esta información se envía por mensajes de texto al teléfono móvil del usuario. La estructura del proyecto mencionado es similar al que se expone en este documento, solo que en éste se hace uso de la internet para envío y recepción de la información (al usuario le llega la notificación mediante un correo electrónico) en vez de un módulo GSM (mensaje de texto y llamada). Cabe notar la gran ventaja que se obtiene al usar la internet ya que se puede acceder al sistema implementado desde cualquier parte del mundo.

La información proporcionada por este proyecto fue de gran utilidad a pesar de manejar diferentes objetivos; la esencia de estos dos proyectos es la seguridad del automóvil, independientemente de cómo se implementaron dichos sistemas de seguridad.

16. GIRONÉS, Jesús Tomás. El gran libro de Android

Se analizó el trabajo elaborado por Restrepo Lacerna (2015), cuyo nombre es: “Sistema de operación y monitoreo para un vehículo mediante Raspberry pi” el cual plantea realizar un prototipo funcional de un vehículo operado y monitoreado remotamente mediante internet.

Estará equipado con una cámara la cual transmitirá streaming a una interfaz gráfica programada en HTML (página web), la que permite la operación del mismo desde cualquier PC o Smartphone conectado a internet en cualquier parte del mundo, mientras que el prototipo expuesto en este escrito, el streaming se podrá observar directamente en la aplicación móvil desarrollada con la cual se podrá observar los acontecimientos desde cualquier parte, siempre y cuando cuente con cobertura a internet. Los lenguajes de programación en común de los dos proyectos son Python, php y Java, ya que estos lenguajes son muy prácticos para realizar este tipo de proyectos gracias a su sencillez. Aparte, el aporte de este proyecto fue enfocado en la parte de la programación, ya que brindo una importante guía de estructuración, la cual facilito la ejecución de algunas etapas de este proyecto.

Ríos y Salazar (2014), llevaron a cabo el “Diseño e implementación de un sistema de seguridad con localización GPS para automóviles en un entorno controlado y monitoreado por una aplicación de dispositivo móvil”. Este proyecto integra un sistema de alarmas de fabricación GENIUS el cual es adaptado con el sistema Arduino UNO y el módulo 3G/GPS. Su funcionalidad es similar a la del sistema que se habla en este escrito, ambos permiten la geolocalización del vehículo pero con la diferencia de que uno envía solo las coordenadas al usuario mediante un mensaje de texto y en el sistema expuesto en este escrito es posible ir viendo la ubicación en tiempo real mediante la aplicación móvil desarrollada e instala en un teléfono móvil. La red que usan en estos sistemas para la transmisión de datos son la GSM y GPRS respectivamente, cada una aportando sus mejores funcionalidades. El aporte de este proyecto fue muy significativo, su estructura sirvió de base para la implementación de nuestro proyecto, se mejoraron ciertas cosas como por ejemplo, el mensaje de alerta se da mediante un correo electrónico en vez de un mensaje de texto por ende se podrá acceder al sistema implementado desde cualquier parte del mundo mediante la internet.

No cabe duda que los proyectos mencionados contribuyeron de manera significativa al desarrollo del proyecto planteado en este documento, ya que de estos se soporta la base teórica para diseñar de forma correcta lo planteado inicialmente en este escrito.

3. DISEÑO

3.1 COMPONENTES DE HARDWARE

El hardware que se utilizó para desarrollar y poner en funcionamiento el prototipo de seguridad para el vehículo está compuesto por los siguientes elementos: mecanismo de alimentación, la tarjeta (placa) raspberry pi 3 modelo B V1.2, modulo GPS Ublox neo-6m, sensor de movimiento infrarrojo HC-SR501, cámaras (una cámara conectada a la tarjeta y otra independiente, esto como método de protección en caso de algún fallo), estos elementos están ubicados de la mejor manera para no incomodar a los ocupantes del vehículo.

3.1.1. Implementación del mecanismo de alimentación. Se uso la batería del automóvil que proporciona un voltaje con el cual no es posible alimentar la tarjeta raspberry, puesto que no cumple con la especificaciones de la tarjeta, para ello se adaptó en el toma de mechero, un dispositivo con una salida de 5 Voltios el cual hace posible la alimentación para esta tarjeta. Con relación a la cámara IP, el voltaje que proporciona la batería para él toma de mechero es el indicado para su óptimo funcionamiento.

3.1.1.1 Puerto de mechero USB. Se optó por tomar la alimentación de este sistema a partir de la batería del auto, esto por medio del puerto de mechero que se encuentra en la parte interna del vehículo (situada entre el conductor y el asiento del copiloto), esta adaptación se realizó por medio de un adaptador de mechero USB, el que brinda dos salidas de 5 voltios con una corriente máxima de 2.1 Amperios, además se tomará directamente de la conexión del toma de mechero su voltaje de 12V para poder alimentar la cámara IP.

3.1.1.2 Sistema de alarma y datos. Se instaló un sistema de alerta temprana silencioso en el vehículo familiar (este sistema se puede adaptar a cualquier tipo de vehículo), este sistema de alerta cuenta con un módulo principal (tarjeta raspberry) el cual recibe todas las señales que captan los elementos instalados en el auto como el sensor de movimiento, el modulo GPS y modulo PI camera. La tarjeta se configuró para cuando los datos son captados por los módulos se suban a la base de datos y el sensor envíe una alarma al propietario del vehículo. En la figura 12 se puede apreciar cómo están conectados los módulos y el sensor, el cuadro naranja señala el modulo del GPS, el cuadro rojo señala el sensor de movimiento, el cuadro verde señala el módulo de la cámara PI de la raspberry y el cuadro azul es la alimentación para la tarjeta.

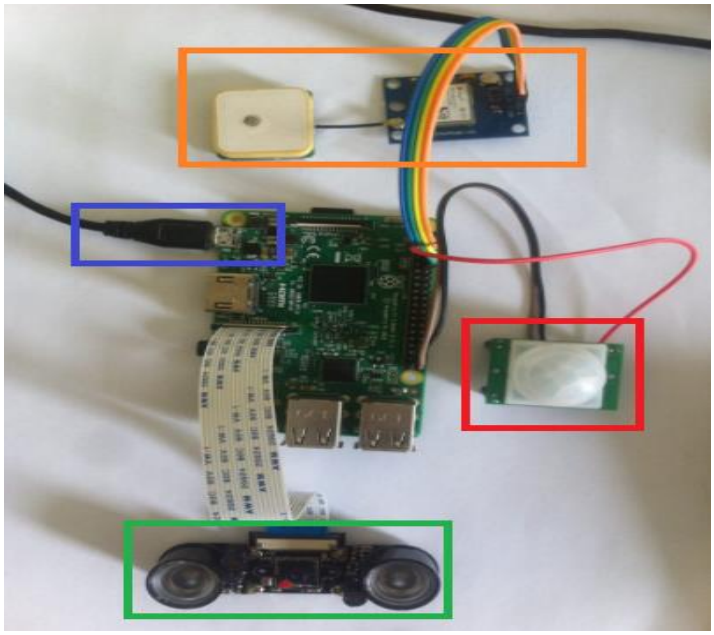


Figura 12. Módulos conectados a la tarjeta

Fuente: Autores

El modulo GPS Ublox neo-6m toma constantemente la posición del vehículo en el formato de Latitud y Longitud, esta información es leída y se envía a un archivo csv (esto se logró por medio de código, ver anexo C), al mismo tiempo se toma el último dato del GPS el cual es subido a la base de datos (ver figura 13).

Por otro lado el sensor de movimiento está unido con el modulo Pi Camera (esto se logró mediante el código implementado, ver anexo C), este fue programado para que el sensor detecte un intruso y automáticamente tome una foto la cual se sube a la base de datos (ver figura 14) diseñada previamente para almacenar este tipo de dato, posteriormente se envía un mensaje de alerta (ver figura 15) al usuario por medio del correo (el correo utilizado fue gmail), este mensaje se enviará cada determinado tiempo para no ocasionar una falsa alarma, entrará en pausa el envió del mensaje por medio de la aplicación móvil y luego se reactivará por el mismo medio. El tiempo estimado en viajar la información del sensor al usuario (correo electrónico) por medio de la tarjeta raspberry está entre un rango de 10 a 20 segundos, esto dependerá del servicio de internet que posea el sistema. Este mensaje se dejará de recibir cuando el usuario por medio de la aplicación móvil desactive el sensor, como se mencionó anteriormente para evitar el spam de los mensajes de alarma (es importante que el usuario reactive el sensor luego de verificar que no se presente ninguna situación anormal).

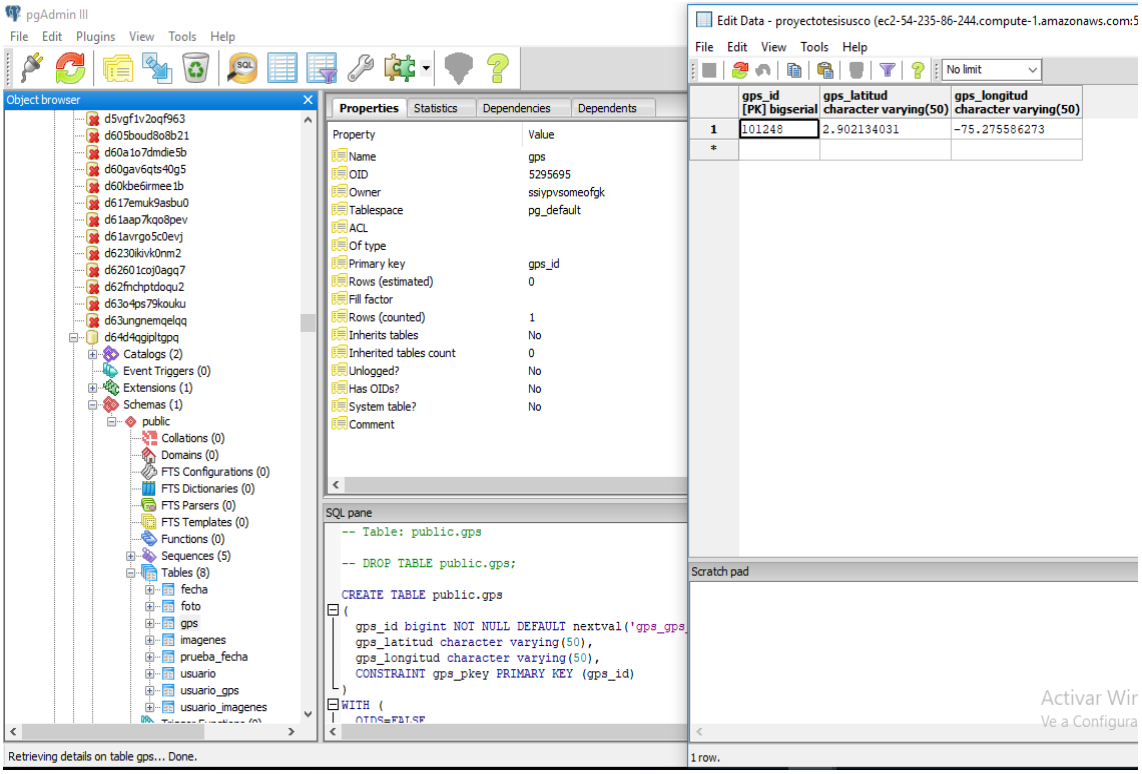


Figura 13. Base de datos GPS

Fuente: Autores

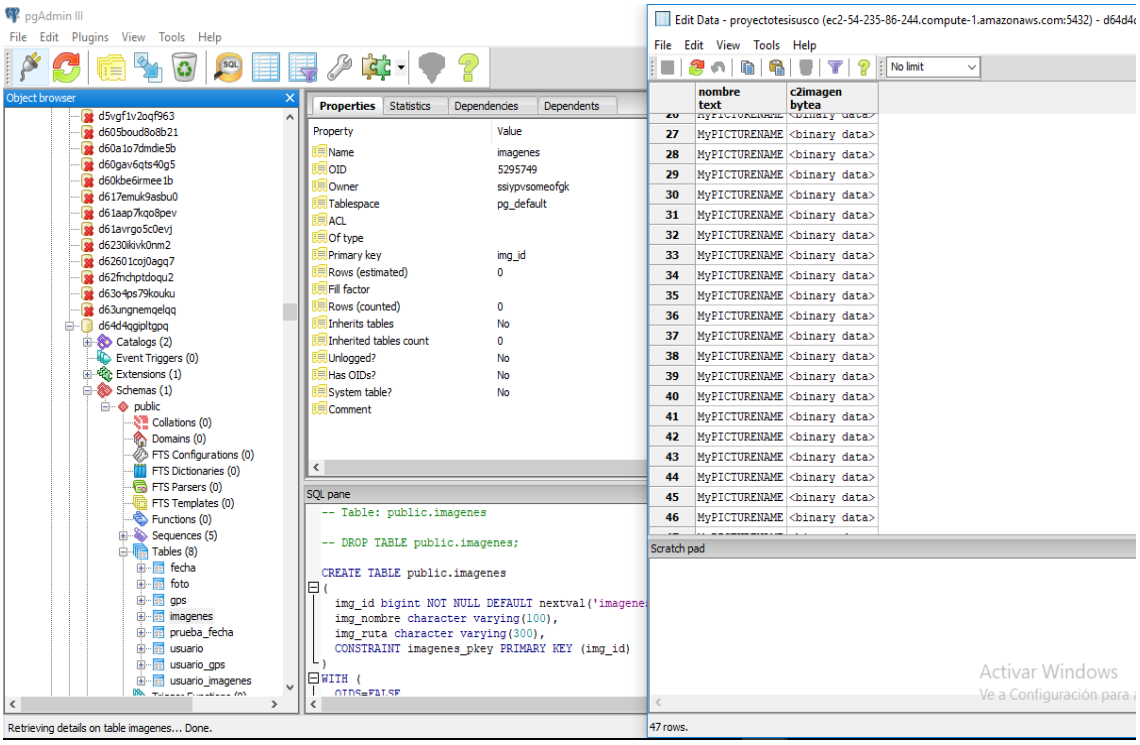
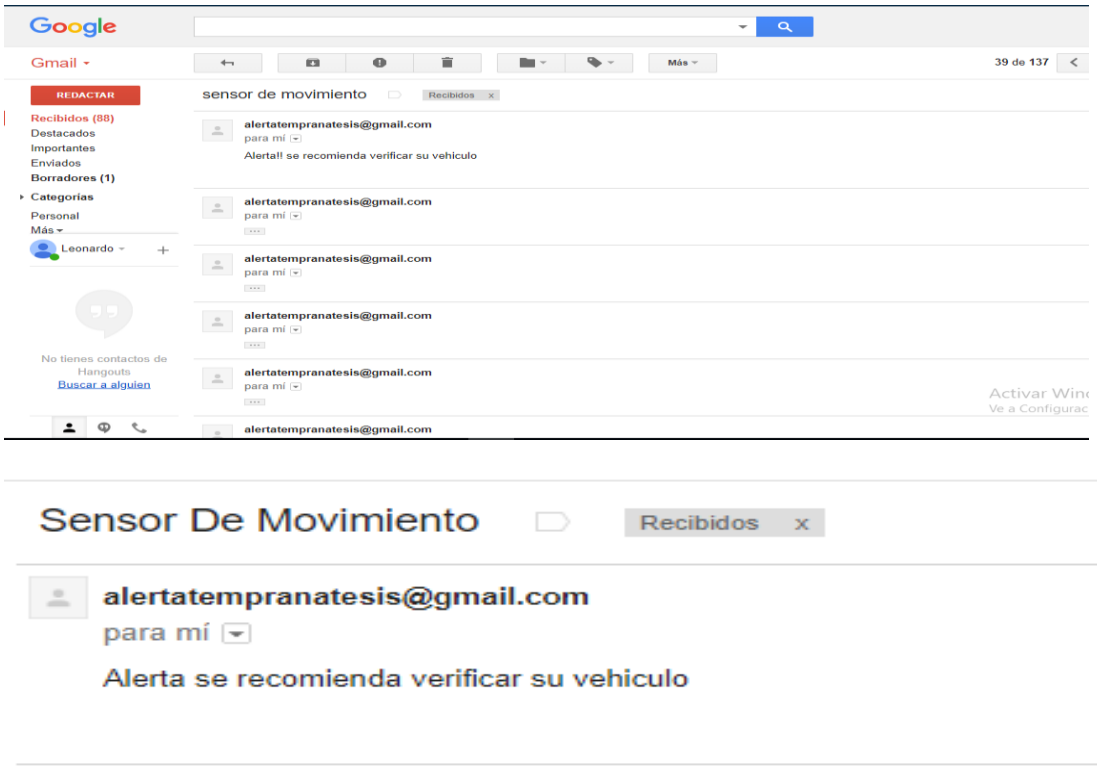


Figura 14. Base de datos fotos

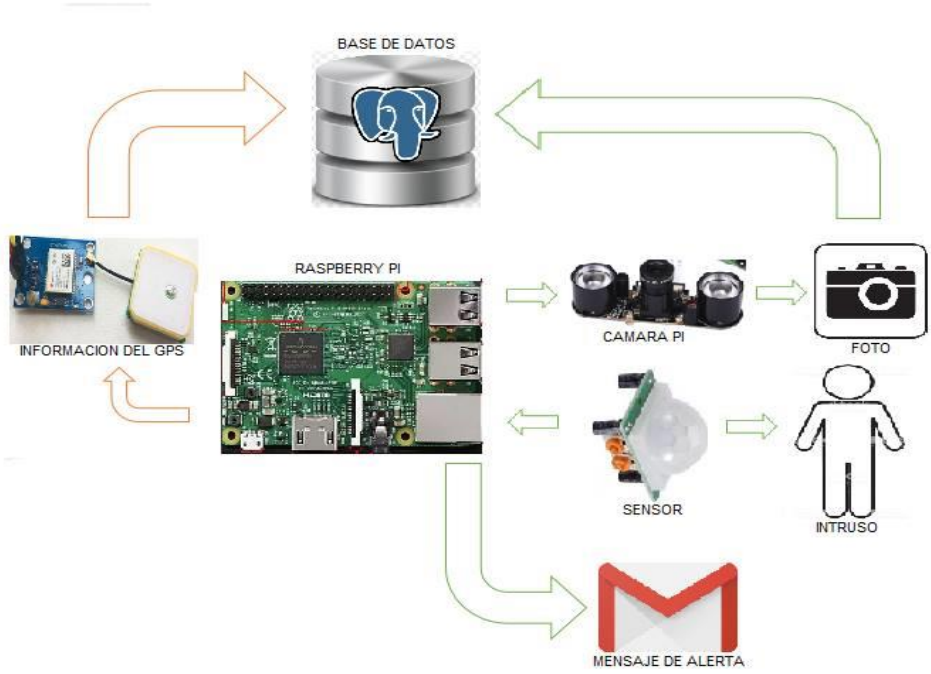
Fuente: Autores



Figuro 15. Mensaje de alerta que se envía al correo

Fuente: Autores

En la figura 16 se puede ver el diagrama el cual específica a grandes rasgos el funcionamiento del prototipo respecto a la parte Hardware.



Figuro 16. Diagrama de funcionamiento hardware

Fuente: Autores

3.2 COMPONENTE DE SOFTWARE

El diseño y desarrollo de la aplicación para monitorear remotamente se ha implementado bajo el sistema operativo Android (Android studio). La principal ventaja de adoptar Android es que ofrece un enfoque unificado para el desarrollo de aplicaciones. Además los desarrolladores solamente necesitan utilizar ProGuard para poder optimizar y reducir el código del proyecto al exportarlo a APK para dispositivos de gama con limitaciones, esto para que la aplicación pueda ejecutarse en numerosos dispositivos, siempre y cuando estos utilicen este sistema operativo.

Para el desarrollo de la aplicación fue fundamental emplear el Android SDK, el cual incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono, documentación, ejemplos de código y tutoriales.

3.2.1 Interfaz de usuario. Las diferentes interfaces de la aplicación se determinan a partir de ficheros XML, los cuales contienen los diferentes Layout que conforman las pantallas de la APP. Se utilizó este método, puesto que nos permiten integrar diferentes elementos de interacción con el usuario, como pueden ser textos, botones, imágenes, entre otros, esto haciendo más amena la aplicación con el usuario.

A continuación se muestra el conjunto de ficheros XML ubicados en el directorio Layout que constituyen las interfaces gráficas, se pondrá como ejemplo el Layout activity_login.xml. (Ver figura 17, 18, 19).

El conjunto de figuras 16, 17 y 18 muestran el resultado de programar el fichero inicial activity_login.xml, el cual representa la interfaz de identificación al lanzar la aplicación.

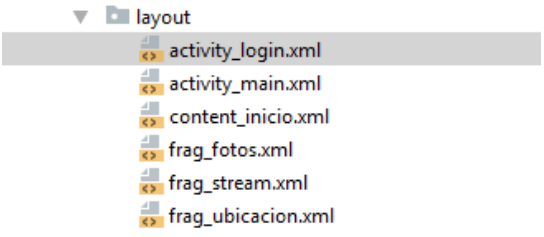


Figura 17. Ficheros XML

Fuente: Autores

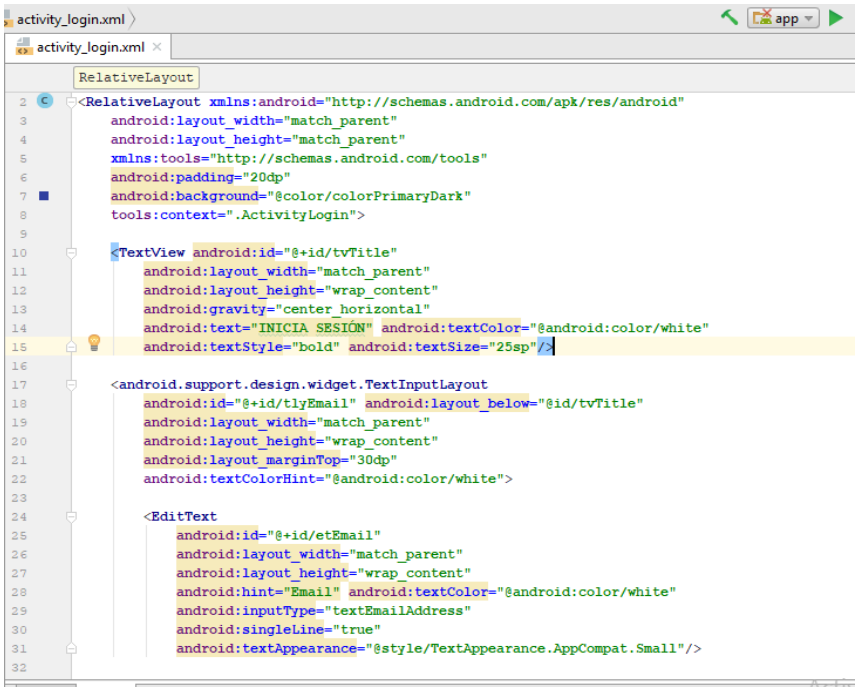


Figura 18. Activity_login.xml

Fuente: Autores

Figura 19. Página de inicio de la aplicación

Fuente: Autores

Por otro lado, para poder ingresar a las funciones que posee la aplicación móvil que se ha diseñado para este prototipo de monitoreo remoto de un vehículo, se debe tener el usuario que en este caso usaremos un email y una contraseña la cual puede contener letras, números y caracteres esto para aumentar el número de combinaciones de esta y dificultarla. Además se optó por no colocar un botón de cerrar sesión una vez se ingresa a las funciones, ya que se ha evaluado una posibilidad de riesgo que sería el hurto o pérdida del dispositivo

móvil del usuario, con lo cual el vehículo podría quedar vulnerable a posibles hurtos, lo cual se quiere evitar con este trabajo.

Como se ha mencionado anteriormente, los ficheros XML contienen la estructura visual de la aplicación, por lo que carecen de lógica o algoritmos de cálculo. Estos se ubican en los ficheros .java (ver figura 20), los cuales se vinculan con los elementos gráficos a través de sus identificadores. Los ficheros .java (ver figura 21) son los responsables de inflar el Layout que corresponda en cada situación y por tanto, controlar la ventana de interacción entre el usuario y el dispositivo.

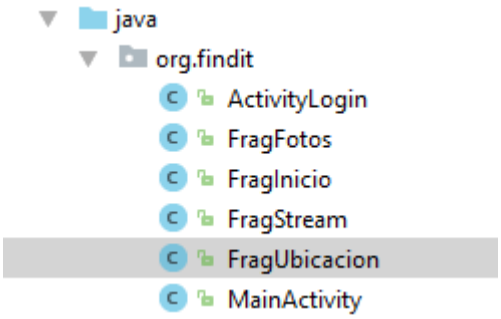


Figura 20. Ficheros .java

Fuente: Autores

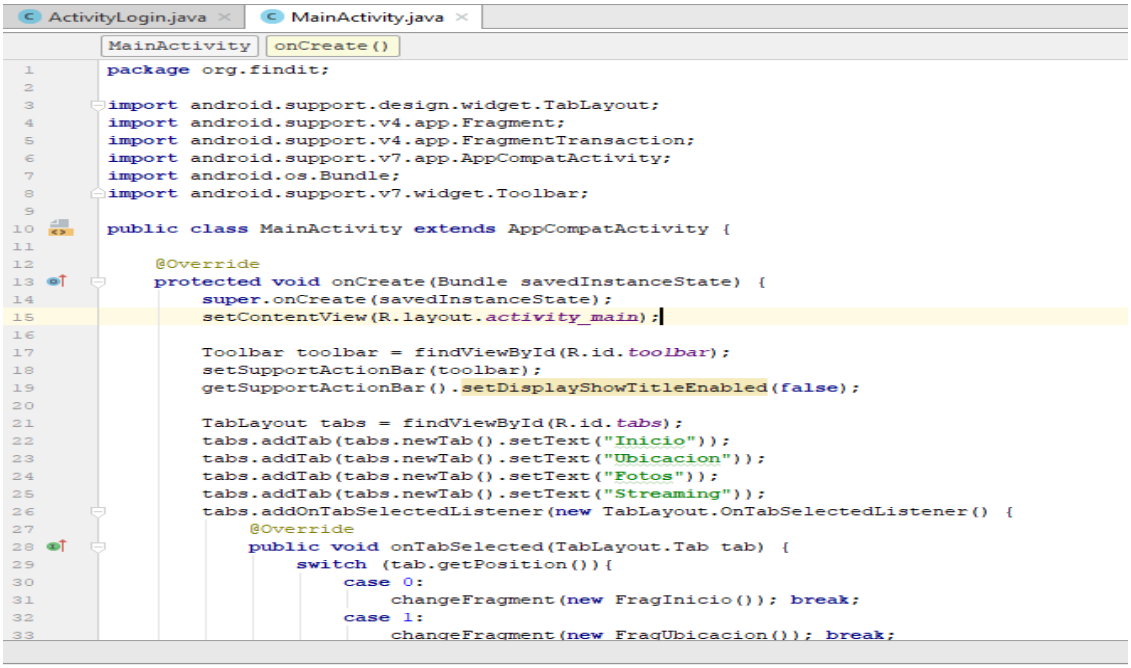


Figura 21. Fichero .java

Fuente: Autores

A continuación se mostrará un esquema del árbol de salto (ver figura 22) entre los diferentes Layout de la aplicación para poder dar una mejor idea en general de cómo se diseñó esta aplicación en la herramienta de Android, y como se puede mover el usuario a través de ella.

Figura 22. Jerarquía de layout

Fuente: Autores

Como se puede apreciar en la figura 22, una vez se abre la aplicación móvil, se encontrarán con una ventana de login, la cual pedirá un usuario y una contraseña para poder ingresar al resto de las funciones que esta posee; una vez ingresado se direccionará a la pestaña de inicio en la cual encontrará una información, y de esta pestaña se podrá mover el usuario a cualquier otra de las pestañas que contiene esta aplicación, se optó por este diseño para que el usuario tenga una dinámica óptima con esta aplicación y tenga un acceso limpio y seguro a esta.

3.2.2 Activities y Fragments. Como se sabe la palabra Activity del inglés significa actividad en español. Por lo tanto, una actividad es cada una de las pantallas que forman nuestra aplicación móvil. En la figura 23 se puede observar cómo se logran sacar las actividades que se usaron para el desarrollo de esta aplicación, teniendo en cuenta un fácil uso de la misma.

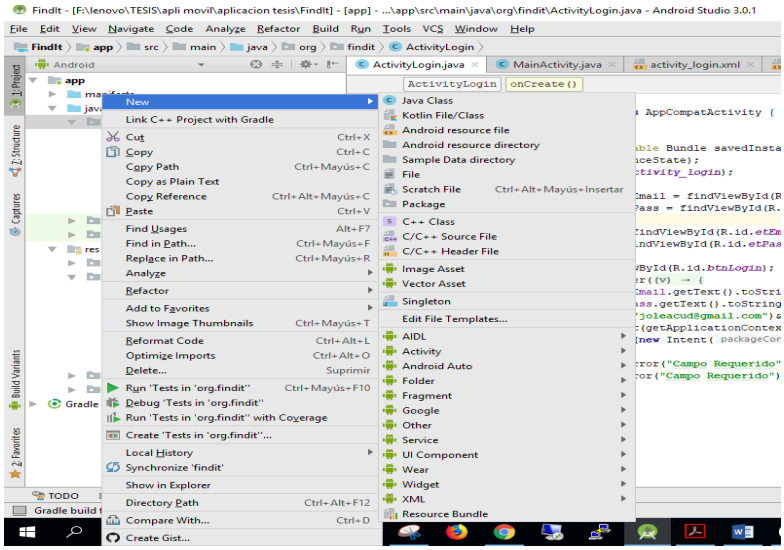


Figura 23. Activities

Fuente: Autores

Los Fragments o en español fragmentos son aquellas partes o comportamientos de la interfaz de usuario en una actividad. En esta aplicación, se crearon múltiples fragmentos en una sola actividad para poder mostrar cuando se ingrese a las funciones de la aplicación en una sola pantalla una barra de menú en la que se encuentran todos los accesos a la vigilancia del vehículo (inicio, fotos, GPS, etc.), estas partes de la interfaz son comúnmente llamadas fragmentos, y quienes contienen a dichos fragmentos son las actividades que fueron creadas para poder tener una aplicación móvil visualmente sencilla de entender. De la figura 23 también se puede sacar estos fragmentos, ya que se han programado en .java.

En pocas palabras lo que se ha querido decir, es para no hacer la aplicación tan densa o complicada con muchas actividades, puesto que podría dañar la experiencia del usuario, se usó la opción de los fragmentos ya que estos son pequeñas partes de la pantalla que se pueden modificar fácilmente (ver figura 24), puesto que se toma como referencia una aplicación móvil ya hecha y de gran uso como es instagram que está hecha por fragmentos y no tanto por actividades.

Figura 24. Fragmentos en una actividad (barra de menú)

Fuente: Autores

3.2.3 Login de identificación. La primera tarea que lleva a cabo la Activity al ser ejecutada es montar el Layout correspondiente, en este caso el fichero de login. Esta interfaz permite al usuario interactuar con la Activity, con el objetivo de loguearse en el sistema y tener acceso a la aplicación (ver figura 19).

Cabe destacar, que para este prototipo solo se hizo un usuario y una contraseña para poder ingresar a la aplicación, si este prototipo se pone en marcha, se creó una tabla en la base de datos (ver figura 25) en la cual se ingresaría los usuarios, las contraseñas y una id que identifique a cada tarjeta (raspberry), para que cada usuario acceda a su información y no a la de otro por error.

Por último, como se puede observar en la figura 25 se creó una tabla en la cual se tienen unas columnas para poder realizar lo mencionado en el párrafo anterior, se crea un usu_id el cual sería el usuario para poder ingresar a la aplicación, usu_nombre y usu_apellido son los nombres y apellidos del usuario, usu_identificacion sería la contraseña para poder ingresar a la aplicación y usu_idraspberry sería la identificación de Raspberry de cada usuario, esto con el fin ya mencionado de acceder a la información correspondiente a cada usuario propietario de este sistema de vigilancia.

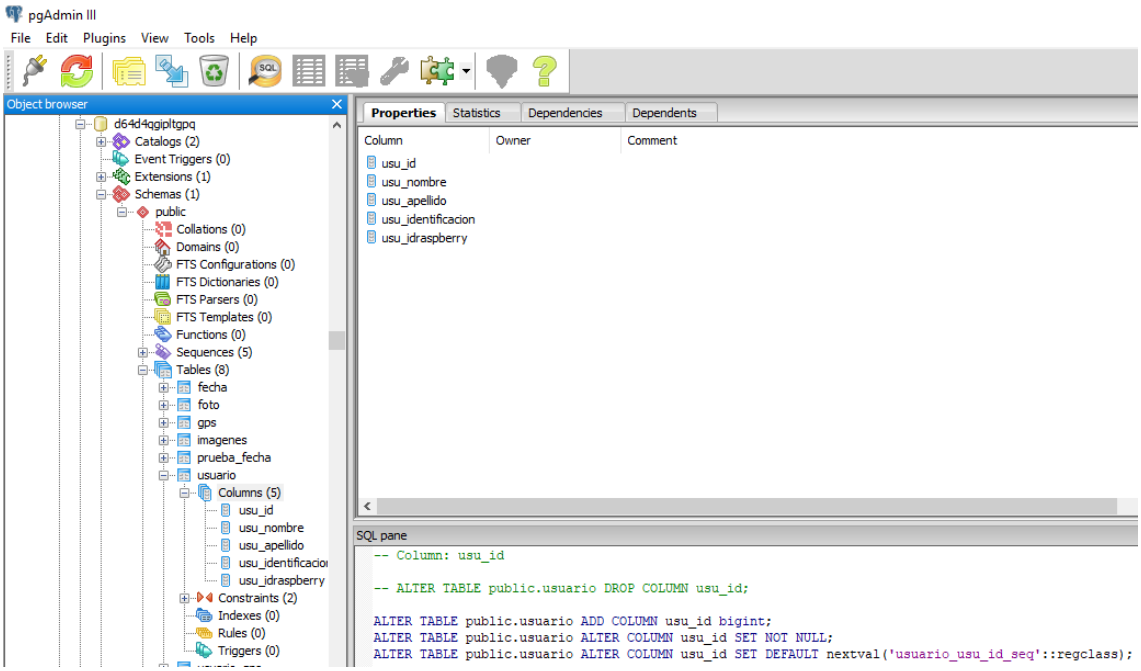


Figura 25. Identificación de cada usuario

Fuente: Autores

3.2.4 Menú de usuario. La clase menú es la encargada de montar la interfaz de acceso a las herramientas de la aplicación. Una vez con acceso al menú, el usuario puede seleccionar libremente a cuál de estas opciones quiere ingresar, puesto que la actividad donde se encuentra el menú contiene los fragmentos de las funciones en los botones de su interfaz, como puede ser el botón del GPS (ubicación) el cual ejecutará la función para mostrar la ubicación del vehículo.

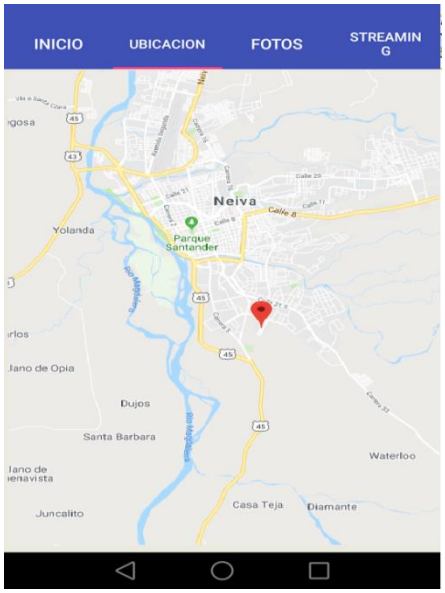


Figura 26. Acceso al menú (ubicación)

Fuente: Autores

3.2.4.1 Inicio. En esta pestaña de la aplicación se encontrara una información en relación al motivo por el cual se está realizando este proyecto, sugerencias personales y de investigaciones en relación al hurto de vehículos o a sus partes, información relevante del hardware y del software que estamos presentado en este informe y el contacto de los estudiantes encargados de realizar esta investigación y proyecto los cuales serán los correos institucionales (ver figura 27).

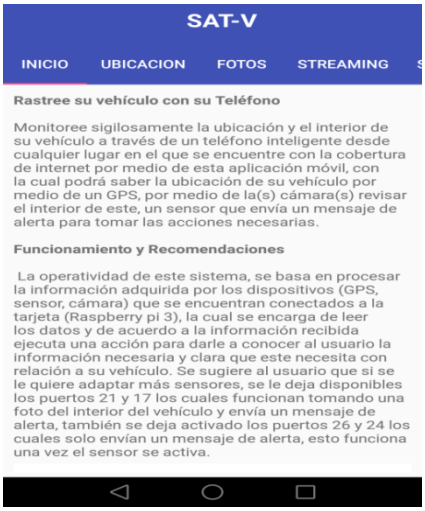


Figura 27. Inicio de la App

Fuente: Autores

3.2.4.2 Ubicación. Esta pestaña es fundamental ya que en esta se encuentra el enlace que se hace por medio de programación a la base de datos de la cual se extrae la información que nos arroja el GPS (ver figura 13) que está conectado a la tarjeta. A continuación se muestra el código que se implementó para poder acceder a la ubicación del vehículo mediante la aplicación móvil. (Ver figura 28 y 29)

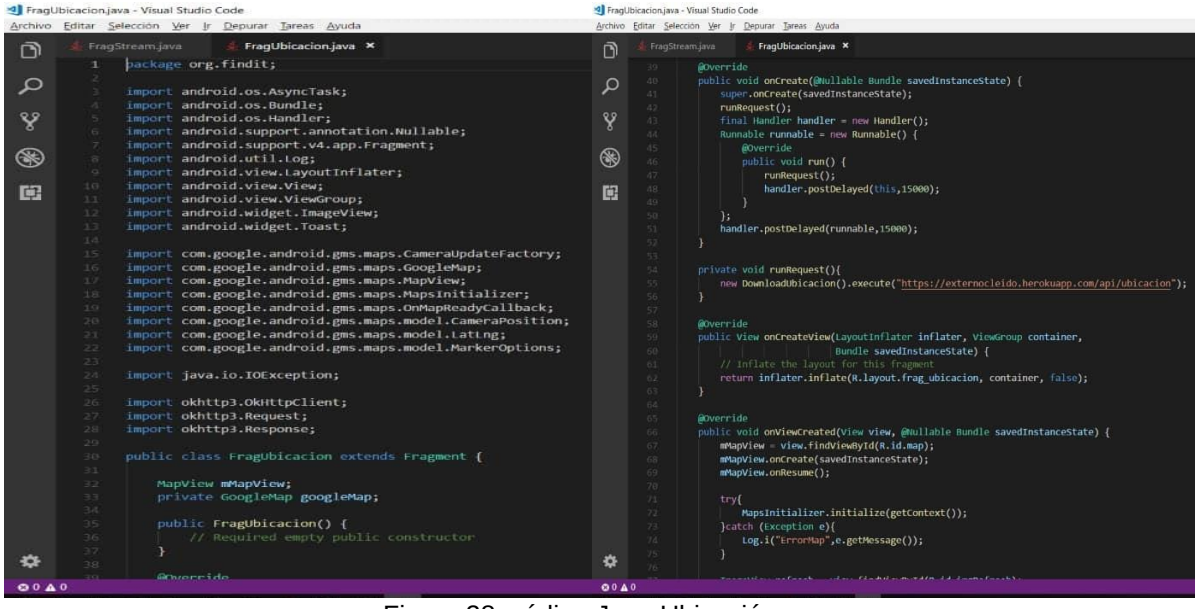


Figura 28: código Java Ubicación

Fuente: Autores

En la figura 28 (parte izquierda) se empieza llamando a todas las librerías necesarias para poder realizar el código correctamente. Una vez hecho esto se procede a comunicar la aplicación con la base de datos que está alojada en la nube, y posteriormente con Google Maps (parte derecha).

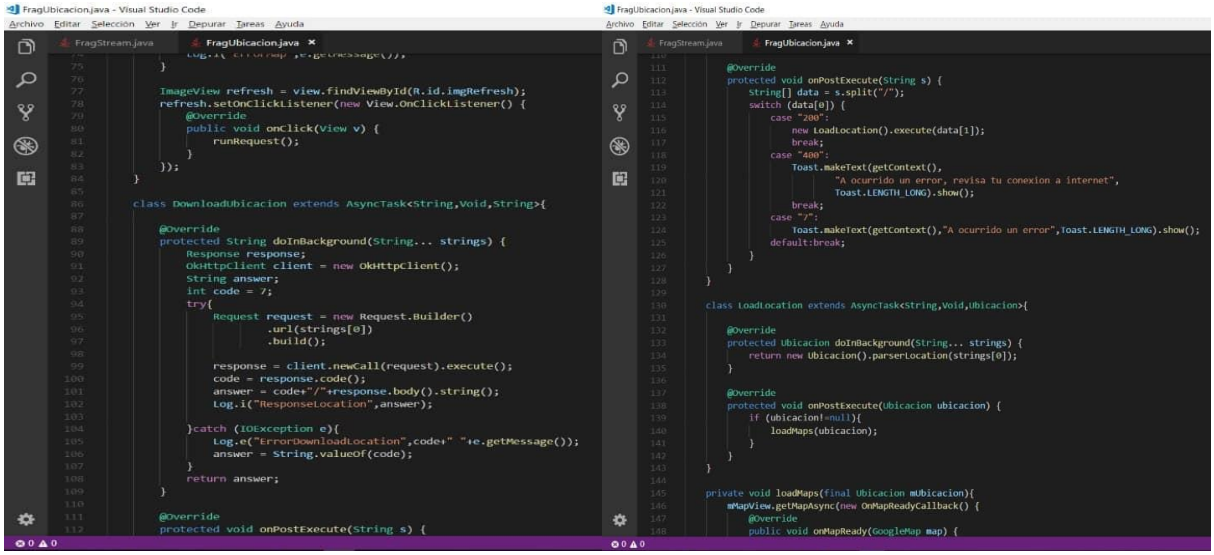


Figura 29: Código Java Ubicación (de eso se trata el código)

Fuente: Autores

En la figura 29 se define el tipo de respuesta que debe dar la aplicación respecto a la petición que haga el usuario, ya sea que muestre la ubicación o algún tipo de error (no es común que aparezca).

Una vez obtenida la información de la base de datos esta se enlaza con google Maps que está instalado en la aplicación (ver figura 30) y se coteja esta información y se logra tener la ubicación del vehículo (este GPS maneja un rango de exactitud de ± 15 metros), se debe aclarar que la rapidez con la que se logra enviar y recibir la información va a estar sujeta a la velocidad del internet que posea el prototipo y el dispositivo móvil.

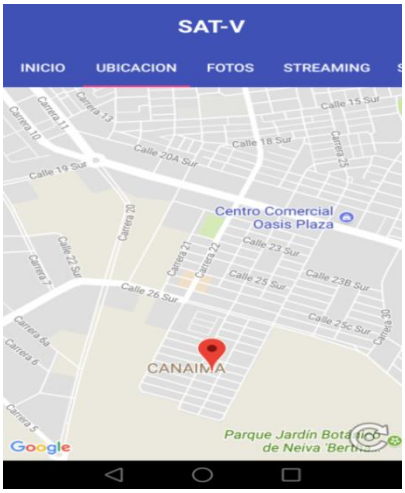


Figura 30. Conexión entre la App “GPS” y la BD

Fuente: Autores

3.2.4.3 Fotos. La opción de fotos se encarga de mostrarle al usuario por medio de unas imágenes que son tomadas por medio de la cámara de la raspberry lo que sucede en el interior del vehículo, esto se logra a través de la conexión que se hace desde la aplicación a la base de datos donde se encuentran alojadas estas imágenes. En la figura 31 se muestra el código implementado para poder ver las fotos mediante la aplicación móvil.

Se debe aclarar que las imágenes se guardan en un formato especial en esta base de datos el cual es bytea (ver figura 14), una vez consultada la carpeta de las fotos en la base de datos, se extrae esta información y es convierte en un formato adecuado para poder visualizar las imágenes que se presentan en la aplicación (ver figura 32). Solo se podrá ver las últimas 5 fotos guardadas en la base de datos con su respectiva hora y fecha en la cual fue tomada.

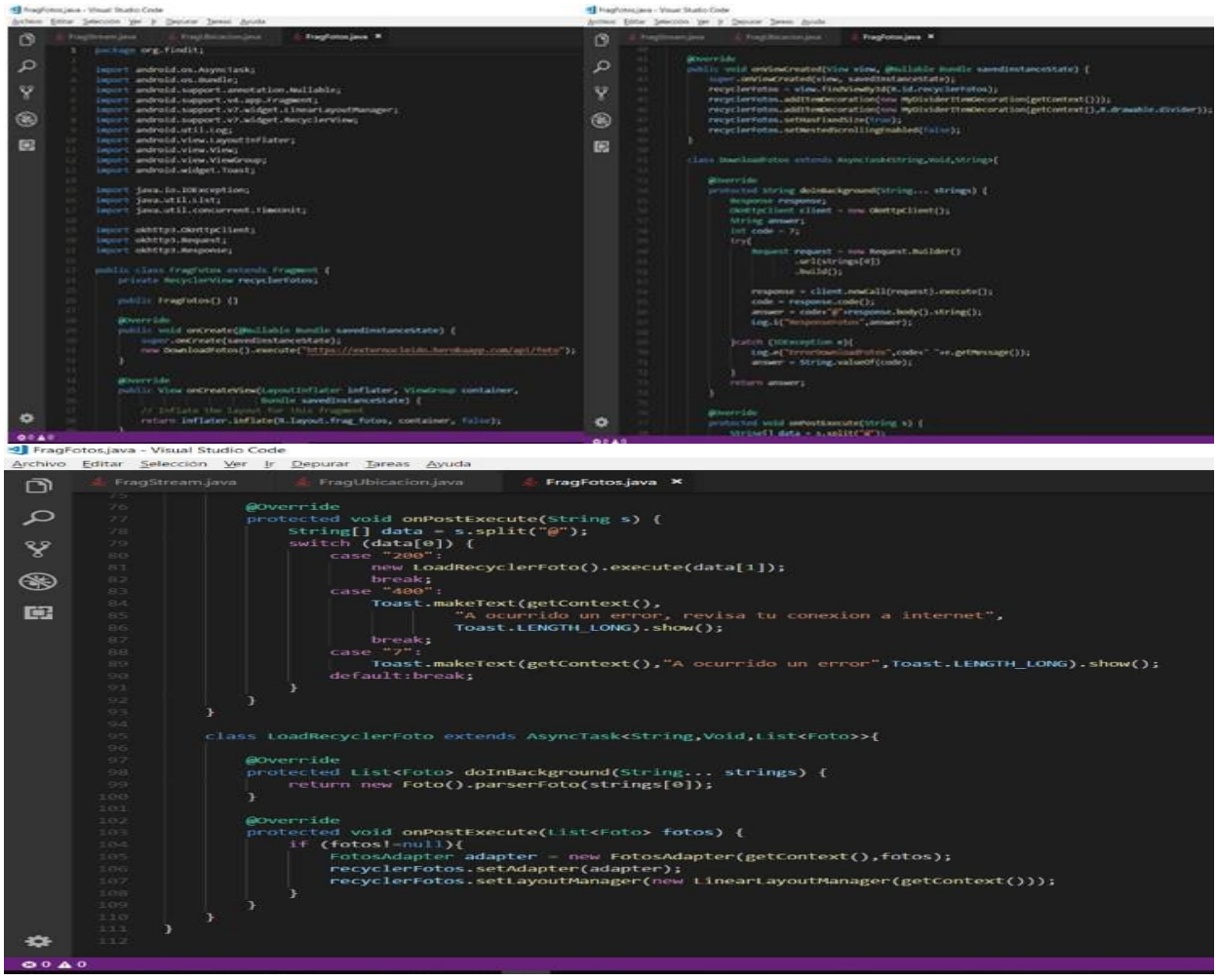


Figura 31: Código Java para fotos

Fuente: Autores

Figura 32. Conexión entre la App “FOTOS” y la BD

Fuente: Autores

3.2.4.4 Streaming. Este fragmento de la aplicación muestra lo que ocurre dentro del vehículo, pero esta vez no por imágenes si no por medio de trasmisión de video, igual como se mencionó anteriormente esto está sujeto a la calidad del internet con el cual se esté trabajando el prototipo, ya que la perdida de frame por segundo puede causar que el video no sea en tiempo real, puesto que esto puede causar un retraso lo suficientemente grande (unos segundos) que no se tomaría como trasmisión en vivo. En la figura 33 se observa el código desarrollado para poder ver el streaming mediante la aplicación móvil.

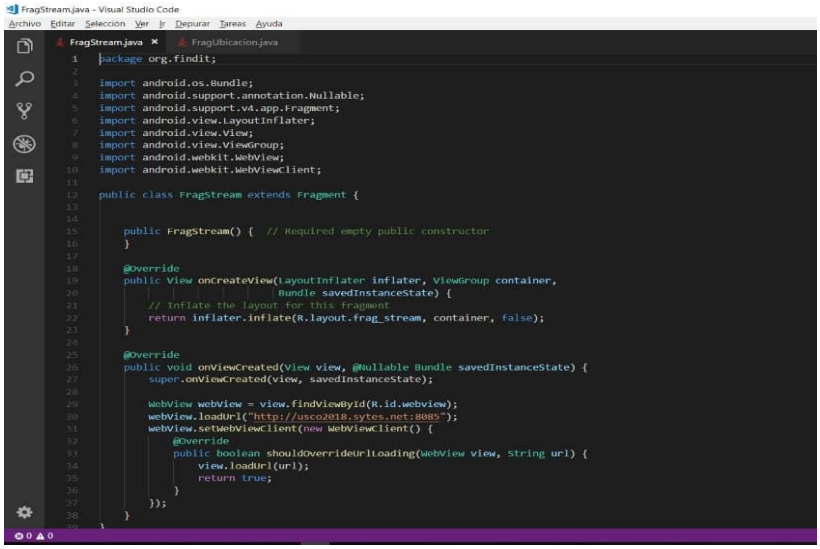


Figura 33: Código Java para streaming

Fuente: Autores

Esta transmisión se hizo por medio de un host creado con anterioridad (ver anexo A), y este host se coloca en el código de este fragmento del streaming para poder visualizar lo que está captando la cámara (ver figura 34).

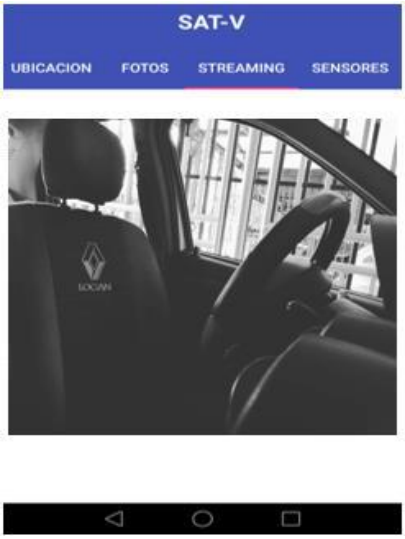


Figura 34. Streaming

Fuente: Autores

3.2.4.5 Registro de eventos. Es una parte fundamental de la aplicación puesto que está creando un registro de los eventos ocurridos en el vehículo, estos registros se crean a partir del sensor de movimiento. Cada vez que se detecte algún movimiento se activa la cámara pi, la cual toma una foto, la que posteriormente se observará en la aplicación móvil con su respectiva fecha y hora a la cual fue tomada, esto permite al usuario revisar en qué momento se ha presentado algún inconveniente con su vehículo (Ver figura 35).

Figura 35. Registro de eventos

Fuente: Autores

3.2.4.6 Botones de Activar/Desactivar. La función de estos botones es simple, es desactivar el sensor de movimiento para evitar el spam de mensajes de alerta que recibe el usuario y reactivar el sensor una vez el usuario se haya cerciorado de que todo está correctamente con su vehículo (ver figura 36). En esta opción de activar y desactivar el sensor se presenta la idea de activar y desactivar el stream con el cual se visualiza el interior del vehículo, con el propósito de disminuir el consumo de datos, el método implementado funcionara de tal manera que el usuario al desactivar el sensor de movimiento activara el stream y al activar el sensor de movimiento automáticamente se desactivara el stream, ya que el usuario accederá al stream solamente cuando se le notifique por medio de mensajes que sucede algo anormal en el vehículo y deba desactivar el sensor para evitar que lleguen más mensajes en forma de spam.

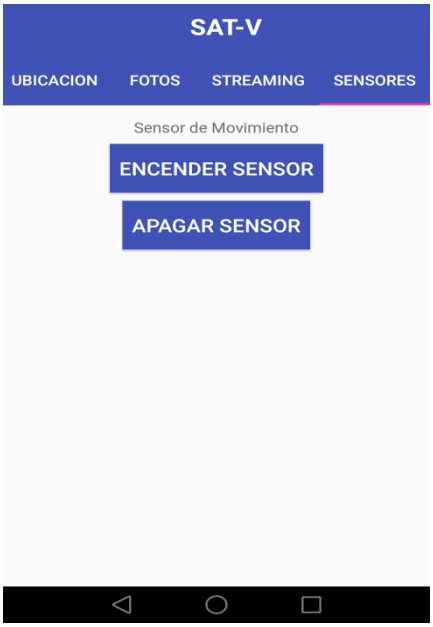
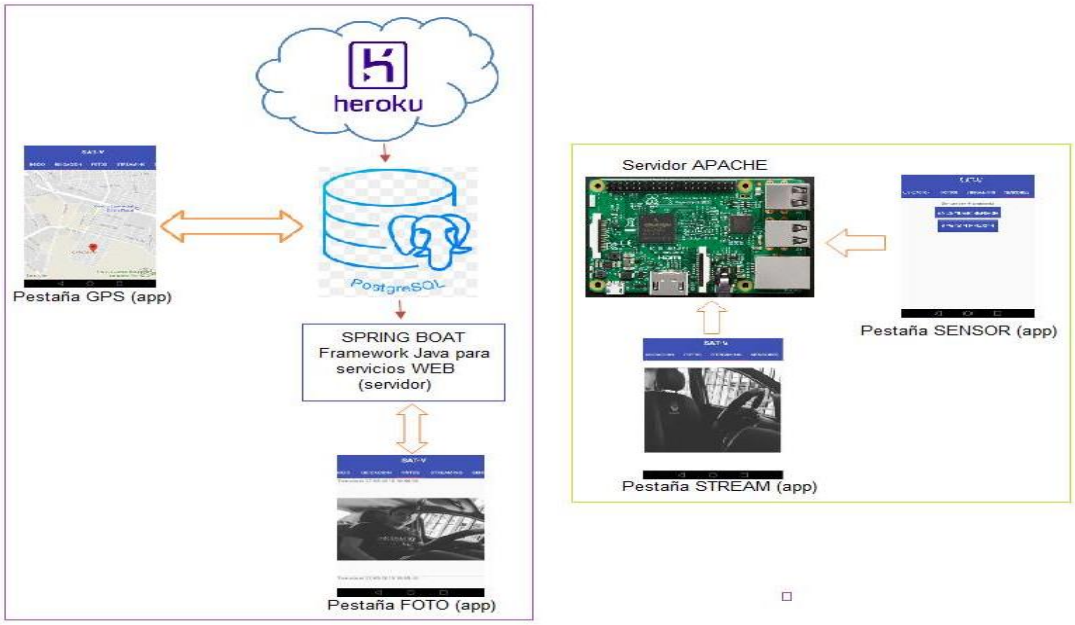


Figura 36. Botones de Activar/Desactivar

Fuente: Autores

Finalmente, se planteó en el transcurso del desarrollo de este prototipo, hacer dos diseños del monitoreo remoto, uno más económico y accesible, y este diseño económico, tendría unas variaciones en cuestión al deshabilitado del sensor de movimiento, este se modificaría por código para que se desactive después de enviar “X” alertas y se reactive en “Y” tiempo luego de haber enviado las alertas y no tendría acceso al streaming, esto debido a los problemas obtenidos con la empresa prestadora de servicio de internet.

En la figura 48 se puede ver el diagrama el cual específica a grandes rasgos el funcionamiento del prototipo respecto a la parte software.



Figuro 37. Diagrama de funcionamiento software
Fuente: Autores

3.3 ACOPLAMIENTO DE LOS COMPONENTES

En este espacio se hablará de cómo se desarrolló el prototipo para la seguridad vehicular en base a cada uno de sus componentes, para darle al lector un resumen detallado de lo realizado, esto con el fin de despejar dudas e inquietudes que le surjan al lector a medida que se lee el documento presente.

3.3.1 Enlace remoto (PuTTY). Para lograr la conexión mediante este programa debe tener la IP que maneja la tarjeta y tener activado la conexión mediante SSH en la tarjeta, estos datos se obtuvieron cuando se hizo la conexión a un televisor mediante el puerto HDMI, una vez se establece esta conexión y conectando unos periféricos (teclado y mouse) a la tarjeta tenemos un acceso completo a ella. Luego se accede al menú de aplicaciones (ver figura 38) y se busca PREFERENCIAS y allí se busca CONFIGURACION DE LA RASPBERRY PI (ver figura 39). Se abrirá una ventana y se busca en el menú INTERFACES y allí se activa la opción SSH (ver figura 40). Además en la ventana del escritorio selecciona el icono (parte superior izquierda de la pantalla se encuentra el icono de red) y se busca la red a la cual se va a conectar (La raspberry pi 3 ya cuenta con el módulo de WiFi integrado) y esta arroja automáticamente la IP (ver figura 41). Una vez obtenido estos datos se abre el programa PuTTY y se coloca esta información para luego seleccionar el modo SSH y así lograr la conexión remota entre la tarjeta y el PC, sin necesidad de cableado (ver figura 42).

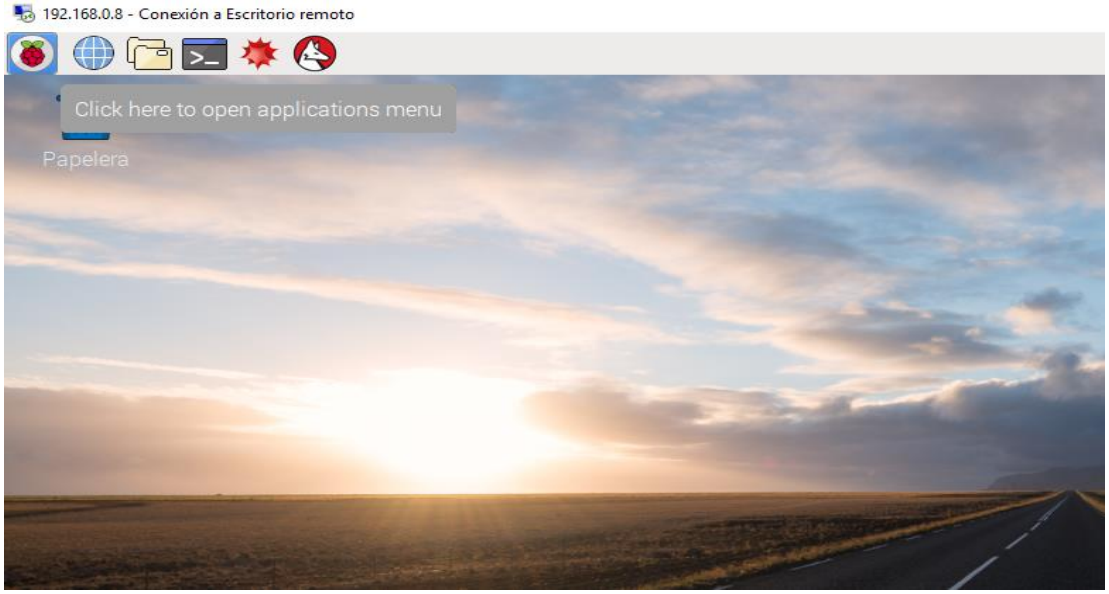


Figura 38. Menú de aplicaciones de la Raspberry Pi 3

Fuente: Autores

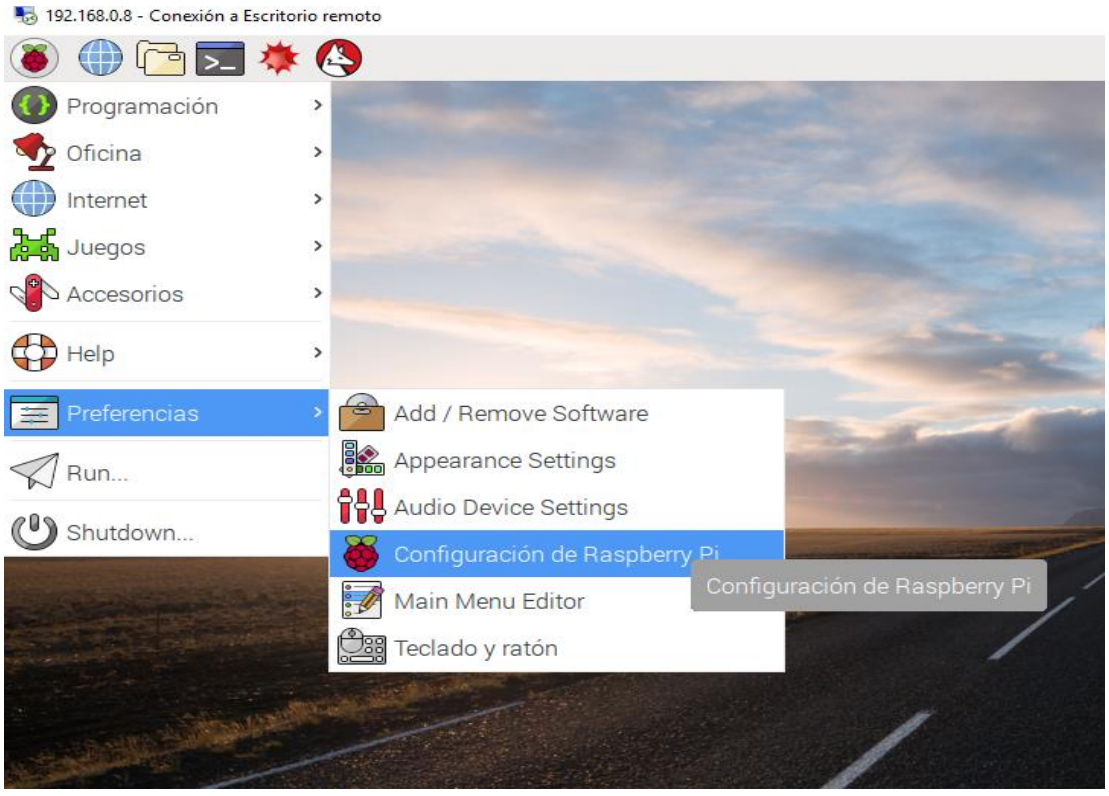


Figura 39. Configuración de la Raspberry Pi 3

Fuente: Autores

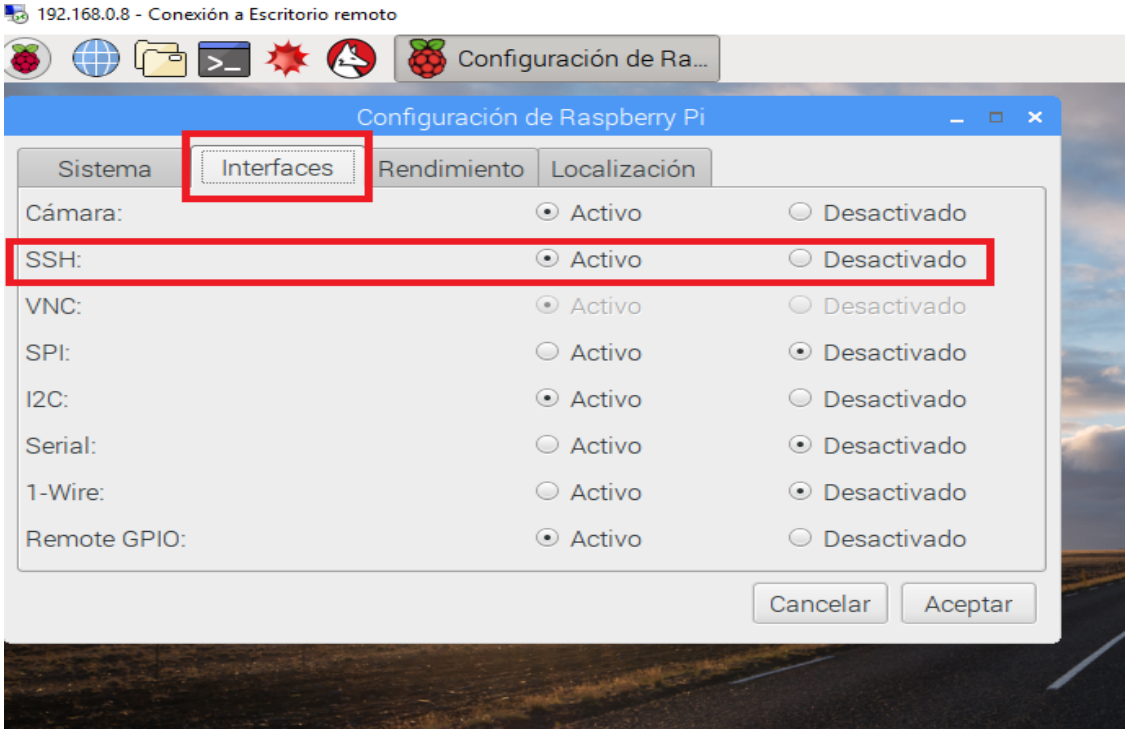


Figura 40. Selección de interfaces

Fuente: Autores

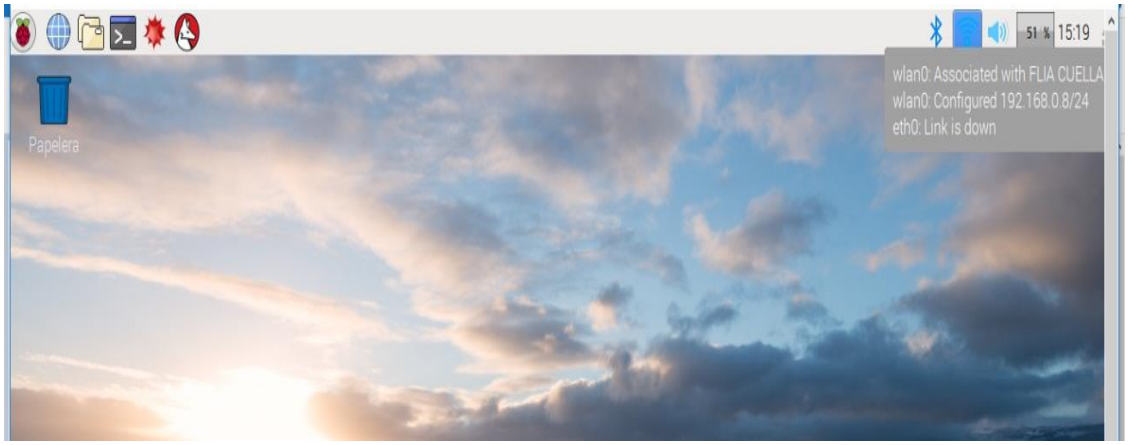


Figura 41. Datos de la red (IP)

Fuente: Autores

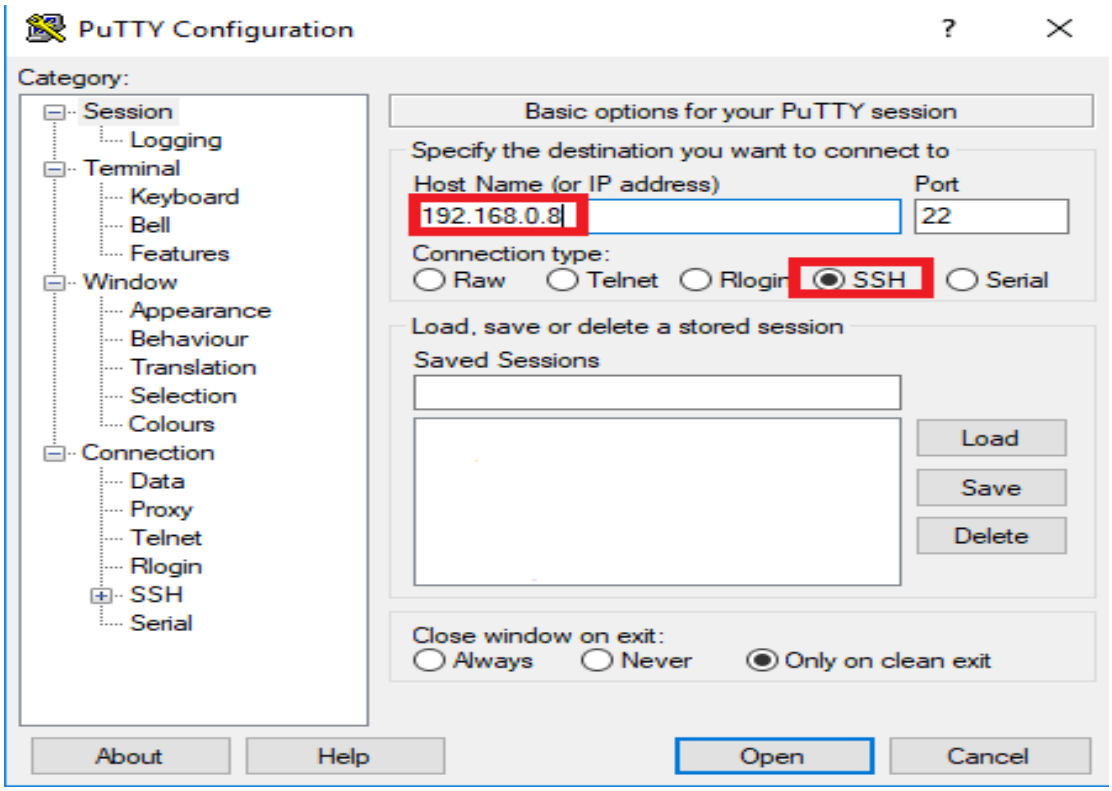


Figura 42. PuTTY (enlace remoto)

Fuente: Autores

3.3.2 Configuración del GPS. Para lograr un óptimo funcionamiento del instrumento, se necesitó instalar un paquete (apt–get install gpsd gpsd–clients) y aparte se modificaron unos archivos para poder leer los datos que arrojaba este (/boot/cmdline.txt, /boot/config.txt, /etc/modules, /etc/default/gpsd) (ver figura 43). Aparte poder una vez leído los datos, almacenarlos en la tarjeta en un archivo csv mediante un script, este hecho en Python. Además, se hizo un script (Python) para almacenar el último dato captado por el GPS en la base de datos creada previamente (este código se puede modificar para guardar no solo el último dato, sino un registro de todos los datos leídos).

3.3.3 Configuración sensor de movimiento. Se realizó un sencillo script en el cual se importaban las librerías que se utilizarían, se nombraban el pin a usar y se configuraba como entrada. Se crea un bucle infinito, en el cual se pone la condición que cada vez que el sensor detecte algún movimiento se envíe un mensaje de alerta al correo del usuario, con el mensaje que se haya programado, este bucle se romperá una vez el usuario lo deseé.

De la misma manera, para poder enviar el mensaje de alerta al usuario se necesitó instalar un paquete (apt-get install mutt) para lograr el enlace entre la tarjeta y el correo. Una vez con el paquete instalado se configura para enviar desde el root (nano /root/.muttrc) (ver figura 42). Para ello se crea un fichero y se rellena una información fundamental tal como es el correo que envía el mensaje la contraseña de dicho correo.

Figura 43. Ventana de comandos

Fuente: Autores

3.3.4 Configuración cámara. Esta configuración se activa el módulo de la cámara de la raspberry pi 3 (ver figura 44, se activar la conexión SSH por medio del programa PuTTY.

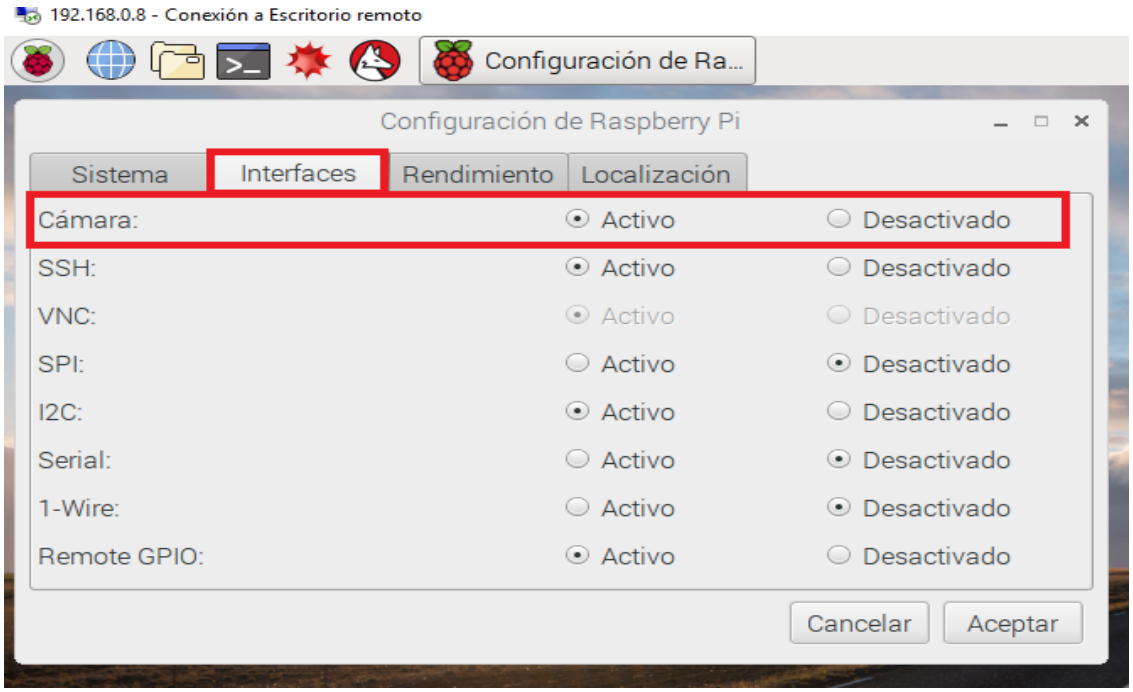


Figura 44. Activar módulo de la cámara
Fuente: Autores

3.3.5 Arranque automático. Este proceso se realizó mediante la modificación del archivo autostart (ver figura 45), cuya función es ejecutar los scripts que se encuentran alojados en este archivo (ver figura 46), este arranque automático se ejecuta en un segundo plano una vez arranque el sistema operativo de la tarjeta en este caso la raspberry pi 3, esto se realizó de esta manera para evitar que el usuario hiciera esta labor ya que se puede presentar el caso que no inicie estos procesos fundamentales y dejaría obsoleto al prototipo planteado en este libro.

```
sudo leafpad ~/.config/lxsession/LXDE-pi/autostart
```

Figura 45. Comando de arranque automático

Fuente: Autores

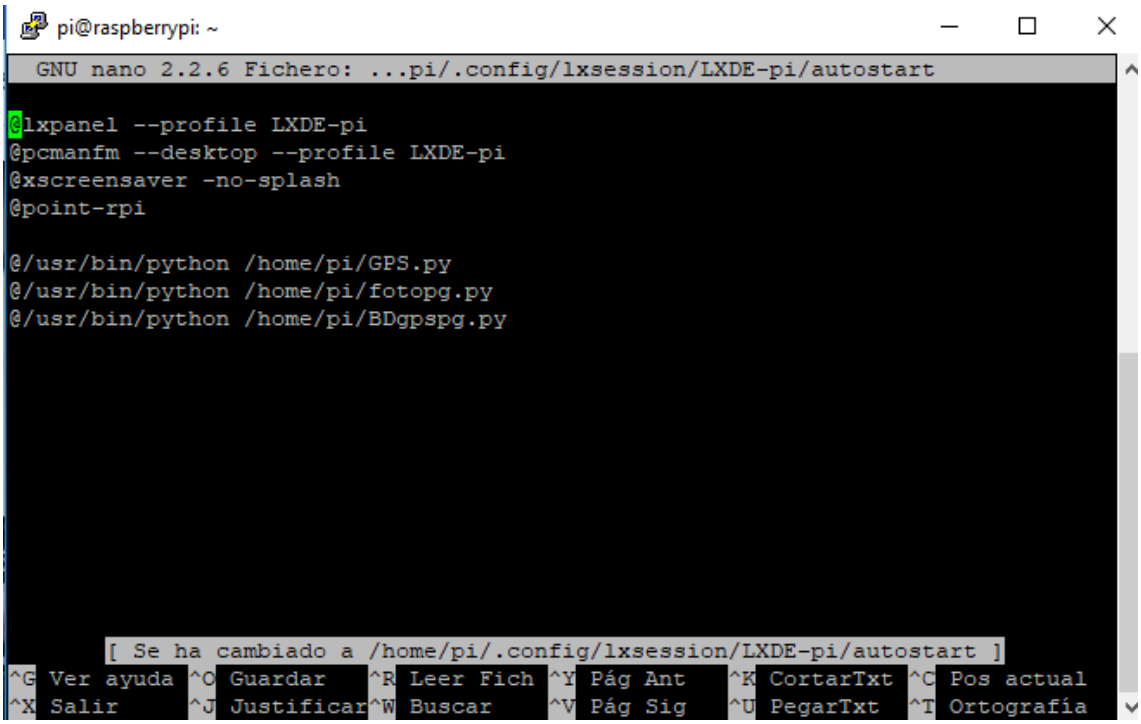


Figura 46. Lista de scripts

Fuente: Autores

3.3.6 Configuración HEROKU. Se registra en la página oficial de HEROKU (https://www.heroku.com) para poder crear y administrar la base de datos Heroku Postgres mediante los datos que nos arroja esta plataforma (ver figura 47).

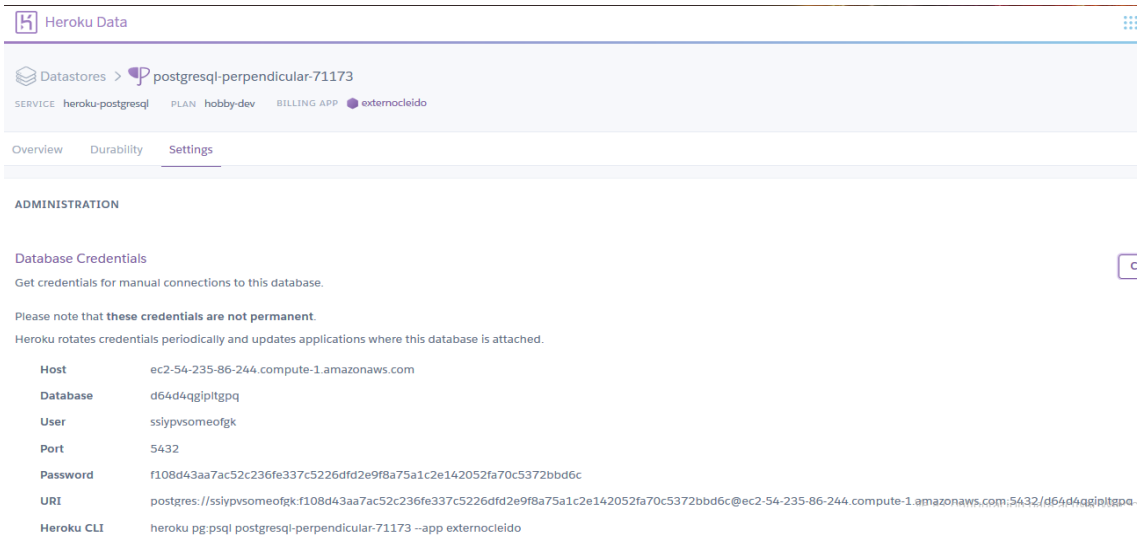


Figura 47. Datos de Heroku

Fuente: Autores

3.3.7 Configuración Base de Datos (Postgres). Para crear la base de datos, se inserta los datos que brinda la plataforma de HEROKU para hacer la conexión (ver figura 48), luego se busca la base de datos en la lista que nos arroja al conectarnos.

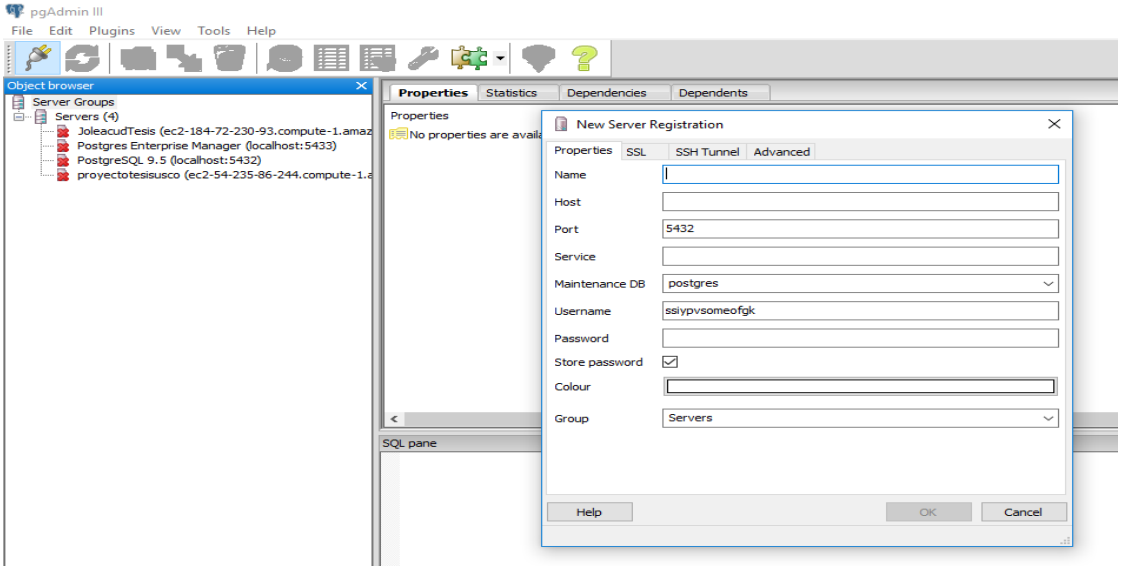


Figura 48. Conexión entre Heroku y Postgres
Fuente: Autores

Finalmente, se crearon las tablas de acuerdo a las necesidades del proyecto, para este caso se almaceno variables de tipo bytea (foto) y character varying (GPS, Latitud y Longitud).

3.4 EVALUACIÓN DEL SISTEMA

Consumo: el consumo de energía de este sistema es moderado; respecto al sensor se usó la configuración personalizada del rango del infrarrojo de movimiento PIR HC-SR501, la cual se estableció en 3 metros para la detección y de un tiempo de respuesta aproximadamente de 3 segundos. (Ver sección 2.1.3), la alimentación de este dispositivo se toma del pin 2 de la tarjeta raspberry pi 3 (ver figura 9) el cual es de 5 V, el consumo de corriente es de 16.2 mA en estado activo, por ende el consumo estimado de potencia de este dispositivo es de 81 mW y en estado de reposo el consumo de corriente es de 50 uA lo que permite calcular un consumo de potencia de 0.25mW. [17]

Respecto a las cámaras se utilizó la cámara IP 3D VR Camera, como método de respaldo para el sistema, ya que ésta posee su propia aplicación móvil la cual permite monitorear el interior del vehículo, en caso de presentar el sistema alguna falla. La alimentación de este dispositivo es de 12 V a 1 A, con un consumo máximo de 3 W. Se especifica que esta cámara no está enlazada con la tarjeta ni con ninguno de sus periféricos, por ende la alimentación se toma directamente de la batería del vehículo (Datos tomados del manual de instrucciones de la cámara).

Por otro lado, el uso del GPS Ublox NEO-6M es constante; todo el tiempo que el sistema esté en funcionamiento, este estará censando la posición del vehículo. La alimentación del GPS puede ser entre 3.3 V a 5V (sección 2.1.5), para este diseño se conectó al pin 1 de la tarjeta Raspberry que suministra una tensión de 3.3 Voltios con una corriente promedio de operación de 45 mA. Se estima que el consumo de energía es de 149 mW. El GPS, aproximadamente toma dos (2) datos por segundo los cuales son procesados por la tarjeta raspberry y transmitidos a la base de datos (el tiempo estimado para dicha trasmisión es de aproximadamente 2 segundos, aunque este valor está sujeto a la calidad del servicio de internet del sistema) que al mismo tiempo es consultada por la aplicación móvil para poder ver la ubicación del vehículo. Ese lapso de tiempo oscila entre 10 y 15 segundos, dependiendo de la velocidad del internet que posea el dispositivo móvil.[18]

Costos: el costo del sistema realmente no es elevado, el precio de los periféricos (sensor, GPS y cámaras), y de la tarjeta raspberry Pi 3 son de un precio asequible. El precio del sensor de movimiento es de $4500, el módulo GPS ublox Neo 6M costó $40000, la cámara pi tiene un precio de $65000, la tarjeta raspberry Pi 3 costó $150000 y la cámara IP valió $150000. En total el costo del sistema estará alrededor de $409500. Se debe tener en cuenta también el costo del servicio de internet que se le proporciona al sistema, eso puede variar dependiendo de la empresa prestadora del servicio. Cabe notar que los precios de estos dispositivos dependen de la empresa que los distribuye y de la calidad que poseen estos mismos.

Confiablidad: a partir de la calidad de los componentes, de la buena implementación del sistema y posteriormente la puesta en marcha del mismo se puede estimar que su confiablidad es alta (se considera del 90%), ya que realmente cumple con los objetivos para el cual fue diseñado. Se debe aclarar que para este prototipo se enfocó solamente en la implementación y no en la seguridad de la información, esta depende del sistema de seguridad de terceros (ejemplo: HEROKU y Postgres).

17.https://www.mpja.com/download/31227sc.pdf
18.https://www.terraelectronica.ru/pdf/show?pdf_file=%2Fz%2FDatasheet%2FU%2FUART+GPS+NEO-6M+User+Manual.pdf

4. LIMITACIONES

- El sistema de seguridad no estará disponible si la placa (tarjeta raspberry) no cuenta con una cobertura de red que le permita enviar la información captada por los ya mencionados elementos conectados a ella. Por ejemplo en zonas rurales de difícil acceso y/o geométricamente no cubiertas por las células. Si se cuenta con un acceso lento e inestable a Internet, la capacidad del sistema disminuye puesto que la información suministrada por la tarjeta puede que llegue de mala forma a la base de datos y la aplicación móvil no pueda leer correctamente los datos que allí se encuentran.

- El usuario propietario tendrá que instalarle un acceso a internet al vehículo para que el prototipo tenga todo el tiempo acceso a internet.

- La alimentación del prototipo se adaptó a la salida de 12 V que nos proporcionó el vehículo (toma de mechero), lo que implica que este sólo funcionará cuando el auto esté encendido, debido a que este prototipo es portable para poder realizar varias pruebas en diferentes tipos de vehículos.

- El video que se graba del interior del vehículo, solo se puede realizar, mediante la aplicación que maneja la cámara independiente, y esta queda grabada en el dispositivo, y sólo puede grabar cuando el usuario da esta opción.

- Se debe tener una compañía prestadora de servicio de internet, la cual facilite la apertura de los puertos, la asignación de host, ya que ciertas compañías por políticas de estas no dejan que el usuario pueda acceder al router para realizar las modificaciones mencionadas

- Con el constante avance tecnológico, se debe tener en claro que estos proyectos, deben ser mejorados constantemente, ya que con las nuevas tecnologías que van surgiendo se puede disminuir costos, mejorar la calidad del usuario y disminuir la contaminación que ocasionan los elementos implementados.

5. CONCLUSIONES

- Se desarrolla e implementa un sistema de alerta temprana para un vehículo, el cual usa una tarjeta Raspberry pi 3, la cual se encarga del manejo de la información captada y posterior envío a la base de datos. Además vincula una aplicación móvil (para Android) amena con el usuario, la cual accede a los datos guardados en dicha base de datos, la cual está alojada en un servidor gratuito, esto con una facilidad y eficiencia notable.

- El prototipo propuesto se cree que establece las bases para obtener un producto final competitivo dentro del mercado relacionado con los sistemas de seguridad vehicular.

- Este es un sistema de seguridad adaptable, ya que se puede implementar en muchos casos, dependiendo de la necesidad del cliente, se puede instalar en casos como: rastreo y monitoreo de cargas, en todo tipo de trasporte público terrestre y con algunas variaciones se puede implementar en otro tipo se seguridad como la del hogar, etc. Además proporcionar evidencia contundente en caso de robo, secuestro, etc. siendo este un sistema óptimo, con un costo razonable para el consumidor.

- Este sistema cuenta con un respaldo de información que queda almacenada en la tarjeta raspberry y otra en la base de datos, esto en caso que se requiera mayor información, y el usuario no la posea en su dispositivo.

- Este prototipo de seguridad de alerta temprana puede ayudar a desarticular bandas delincuenciales, dependiendo del uso que se le dé, se pueden poner vehículos carnada para que la delincuencia los hurte e ir identificando a las personas que se dedican a este crimen.

- El desarrollo de la aplicación Android ha ofrecido un cuadro de mando versátil y de fácil uso con el usuario. Se han conseguido desarrollar tanto las tareas de petición como de visión propuestas en este trabajo, además, gracias a la estructura basada en Activities y Fragments, es posible aumentar el número de controles de la aplicación de una forma fácil y rápida. A parte sumado a la gran compatibilidad de la aplicación entre todas la versiones del SO.

- Se desarrolla todo el software empleando código opensource, lo que otorga una gran ventaja desde el punto de vista económico y desde la posibilidad de encontrar fácilmente soporte a través de la web, con lo que se contribuye al desarrollo de proyectos de código abierto, los cuales, cualquier usuario puede probar, modificar, mejorar y publicar, esto para ir avanzando junto con las nuevas propuestas que se van desarrollando con el tiempo, lo cual genera un impacto tanto global,

social y económico, puesto que cualquiera puede acceder a esta información y manipularla.

- Al trabajar en la plataforma de Android Studio, se facilita la implementación de la aplicación móvil propuesta, puesto que esta plataforma cuenta con una gran cantidad de librerías las cuales permiten desarrollar la app expuesta con poca complejidad.

- Luego de realizar las evaluaciones correspondientes al sistema y la aplicación móvil, se puede dar unas posibles mejoras en el ámbito mecánico del vehículo, ya que los desarrolladores no poseen conocimiento en esta rama de la cual pueden surgir mejoras al sistema planteado en este escrito.

- La llegada de próximas tecnologías pueden llegar a ofrecer una mejor transmisión y recepción de la información, y nuevas placas como la utilizada en este proyecto pero cada vez más potentes, harán que proyectos de este tipo mejoren su rendimiento en un gran porcentaje.

- Con el sistema presentado se desea mejorar la calidad de vida de los usuarios, puesto que van a poseer una herramienta que les librará de presión y estrés, porque siempre sabrán donde está su vehículo, incluso en caso de hurto.

- La coordinación entre sectores (sociedad, entidades de control y vigilancia) son las características determinantes de los sistemas de vigilancia vehicular; por lo tanto, el trabajo articulado en todas las etapas del proceso debe ser eficaz para lograr el objetivo principal del proyecto que es la disminución de la delincuencia en el país, en este caso particular es disminuir el creciente caso de hurto a vehículos.

6. RECOMENDACIONES

- Uso de otros métodos de almacenamiento de información (como en la propia memoria del dispositivo para tener siempre a la mano la información), ya que los servidores gratuitos y algunos pagos presentan restricciones que afectan el rendimiento del sistema, puesto que no se puede guardar toda la información deseada.

- Usar otras herramientas para recibir las alertas y la información que suministra los sensores y los módulos, ya sea por medio de llamadas, o mensajería por medio de la aplicación de WhatsApp.

- Ampliar el esquema de seguridad para el vehículo, como el baúl, llantas, bloqueo de motor, bloqueo de puertas. También volver el sistema invasivo con elementos que no consuman mucha energía para no afectar negativamente al auto.

- Adquirir servicios adecuados con los operadores para el óptimo funcionamiento del sistema y el dispositivo móvil.

- Uso de hardware de menos costo y que tenga más flexibilidad.

- Actualizar y revisar periódicamente el sistema de la raspberry, la base de datos, sensores y módulos para su óptimo funcionamiento.

- En caso de ser implementado el sistema se debe evitar mover el sensor, GPS, cámara ya que esto puede ocasionar que no se tomen los datos correctamente, y esto ocasionaría que el sistema genere falsas alertas o no envié información.

BIBLIOGRAFÍA

[1]. BEDOYA Yeferson, SALAZAR Cristian, MUÑOZ Jhon. Tesis; implementación, control y monitoreo de un sistema de seguridad vehicular por redes GSM/GPRS. (Fecha de consulta, 4 de Mayo del 2017)

[2]. CAUSER Mike, 2014. Raspberry- Pi-ITead-Studio-GPS-NEO-6M. DISPONIBLE EN: https://github.com/mcauser/Raspberry-Pi-ITead-Studio-GPS-NEO-6M. (Fecha de consulta, 17 de Abril del 2018)

[3]. CULQUICHICÓN, Juan Carlos. Tesis; Domolab: Sistema de monitoreo y control remoto de viviendas. (Fecha de consulta, 10 de Junio del 2017)

[4]. GIRONÉS, Jesús Tomás. El gran libro de Android. Marcombo S.A. Ediciones técnicas, Barcelona. (Fecha de consulta, 2 de Abril del 2018)

[5]. GONZÁLEZ DUQUE, Raúl. Python para todos. Este libro se distribuye bajo una licencia Creative Commons Reconocimiento 2.5 España. (Fecha de consulta, 20 de Febrero del 2017)

[6]. GONZÁLEZ Lucas, 2017. Manual del usuario sensor de movimiento Pir Hc Sr501. DISPONIBLE EN: https://es.scribd.com/document/357270269/Manual-Del-Usuario-Sensor-de-Movimiento-Pir-Hc-Sr501. (Fecha de consulta, 8 de Abril del 2018)

[7]. RAMÍREZ, Juan Carlos. Tesis; sistema contra robo de vehículos. (Fecha de consulta, 15 de Marzo del 2017)

[8]. Raspberry Pi Foundation, 2016. Documentation Python. Página Web. DISPONIBLE EN: https://www.raspberrypi.org/documentation/usage/python/README.md. (Fecha de consulta, 15 de Enero del 2017)

[9]. Raspberry Pi Foundation, 2016. Página Web. DISPONIBLE EN: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera. (Fecha de consulta, 8 de Abril del 2018)

[10]. Raspberry Pi Tutorials. Connect and control Raspberry Pi motion detector Pi. DISPONIBLE EN: https://tutorials-raspberrypi.com/connect-and-control-raspberry-pi-motion-detector-pir/. (Fecha de consulta, 29 de Marzo del 2017)

[11]. RESTREPO LACERNA, Javier. Tesis; sistema de operación y monitoreo para un vehículo mediante raspberry pi. (Fecha de consulta, 16 de Mayo del 2017)

[12]. RÍOS Jorge, Salazar Leonardo. Tesis; diseño e implementación de un sistema de seguridad con localización GPS para automóviles en un entorno controlado y monitoreado por una aplicación de dispositivo móvil. (Fecha de consulta, 18 de Febrero del 2017)

[13]. VACA, Cesar Daniel. Tesis; Sistema remoto de monitoreo y control en casa de habitación. (Fecha de consulta, 25 de Agosto del 2017)

ANEXOS

ANEXO A.

NO-IP: FREE DYNAMIC DNS

El siguiente manual es una pequeña guía de cómo crear una cuenta en la plataforma de NO-IP, para poder acceder a sus beneficios, ya que estos son muy buenos y son de uso gratuito, aunque también está la versión paga, pero en este caso nos enfocaremos en la versión gratuita.

Lo primero que se debe hacer es acceder a la página oficial de NO-IP para poder registrarnos (https://www.noip.com/), una vez se ingresa en la página, busca “Sign up”, el cual está en la parte superior derecha (ver figura 49), le da click y esta nos redirige a la página donde se ingresarán los datos personales con los cuales la página nos identificará, estos simplemente un correo y una contraseña, es opcional crear de inmediato el host, esto lo puede hacer luego simplemente seleccionando la opción “Create my hostname later” (ver figura 50).

Figura 49. Sign up

Fuente: Autores

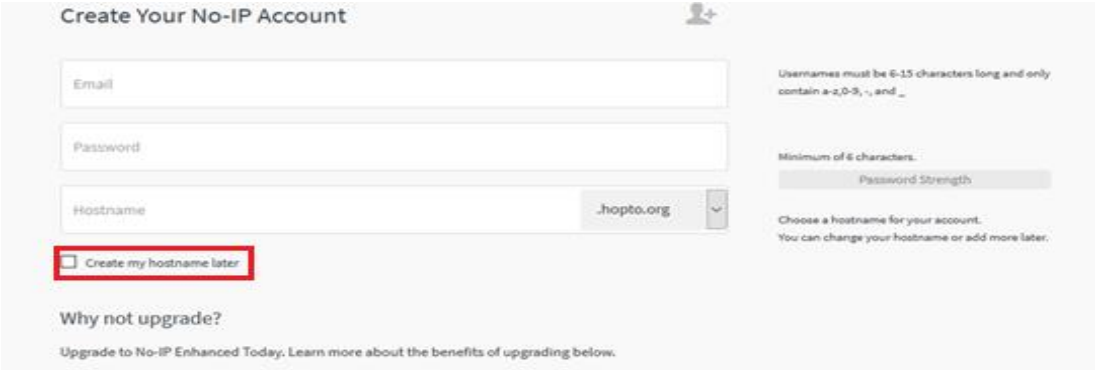


Figura 50. Resgistro

Fuente: Autores

Finalmente, nos envían un mensaje automáticamente al correo que se ingresó al momento del registro, esto con el motivo de validar la información que se ingresó y activar nuestra cuenta (ver figura 51). Una vez realizado estos pasos se podrá disfrutar de todas las opciones que nos brinda esta plataforma de forma gratuita.

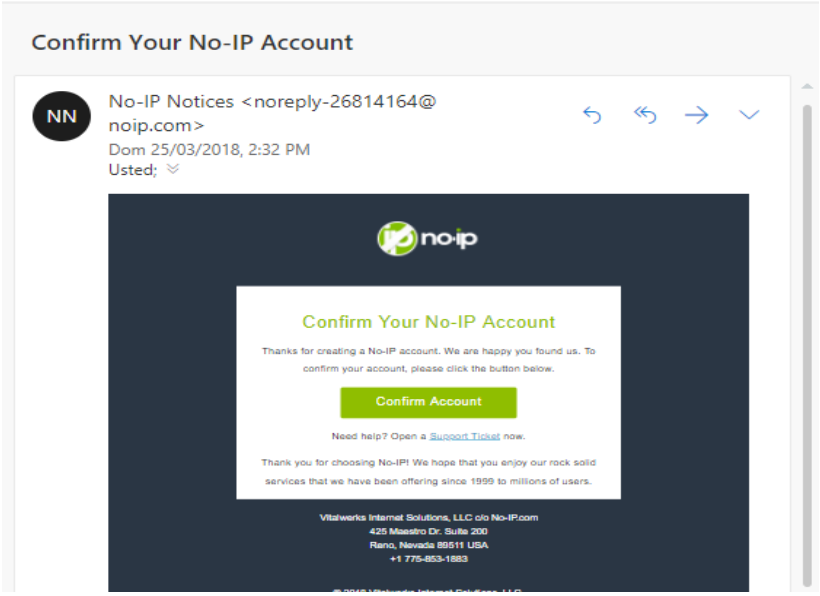


Figura 51. Verificación de correo

Fuente: Autores

Así mismo, una vez realizado estos pasos ya se podrá crear el nombre de nuestro host, en el caso que no se halla creado al momento del registro. Esto se logra ingresando con el correo y la contraseña con la cual se hizo el registro y se valida como se observó anteriormente en este anexo. Una vez logeados en la página de NO-IP se busca en la parte izquierda de la pantalla una columna y selecciona “Dynamic DNS Free” (1) la cual mostrará unas opciones, entre las cuales está “Hostnames” (2) y se selecciona. Se procede a crear nuestro host (3) con el nombre que se deseé y con el dominio gratuito que quiera utilizar del listado disponible (ver figura 52).

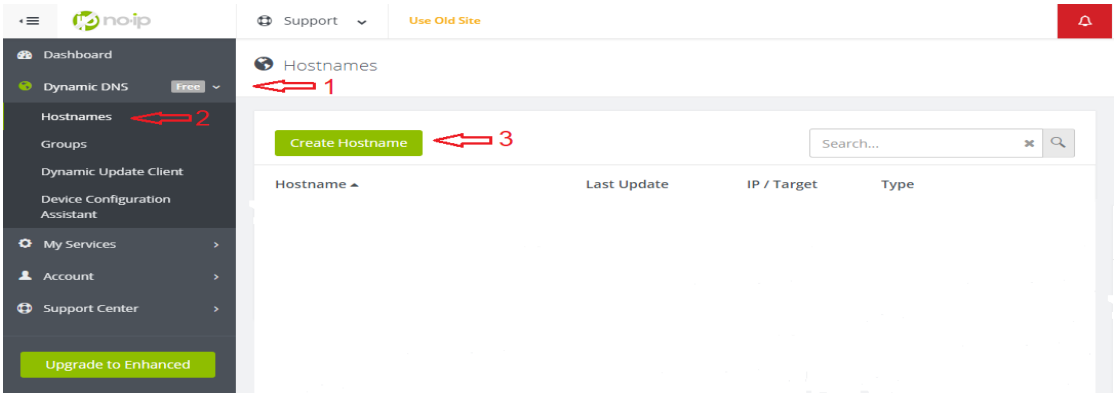


Figura 52. Pasos para crear el host

Fuente: Autores

Una vez hecho los 3 pasos mencionado en el párrafo anterior, se nos abrirá una ventana para ingresar los datos de nuestro host (ver figura 53), en la cual se observar que se puede poner el nombre que se desee a nuestro host (siempre y cuando ese nombre no esté en uso), también se puede poner el dominio gratis que deseé. Además se observa el IPv4 Address y Record Type estas opciones no se modifican se dejan con la configuración de predeterminada. Y listo ya se ha creado el host con el dominio que se haya seleccionado.

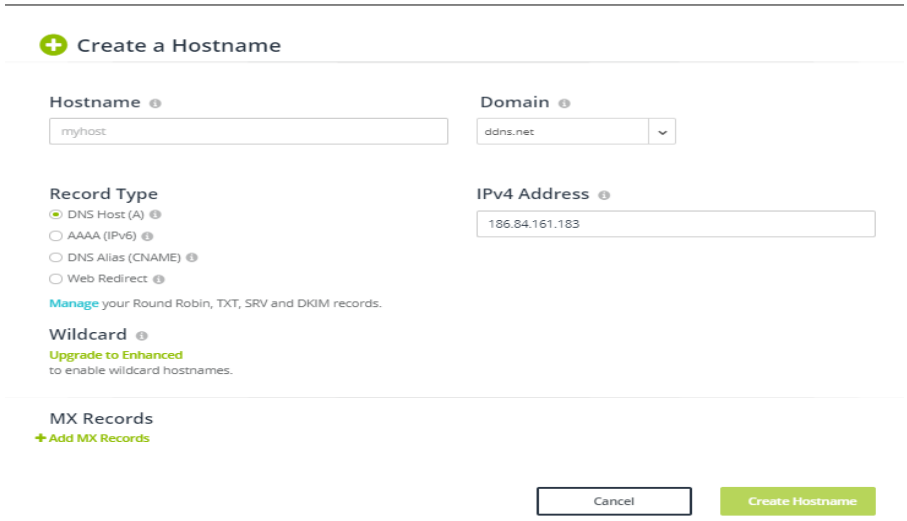


Figura 53. Crear Host

Fuente: Autores

Ahora sólo falta configurar este host de NO-IP en el router. Se ingresa al router con el usuario y contraseña dadas por el proveedor de internet, se busca la opción DDNS, esta ubicación varía de acuerdo al router que se esté manejando, una vez ubicado se selecciona el servicio, en este caso NO-IP, se ingresa el usuario y contraseña con el cual se hizo el registro en la página de NO-IP y se coloca el nombre del host con el dominio que se ha creado y ya quedaría listo para poder acceder remotamente.

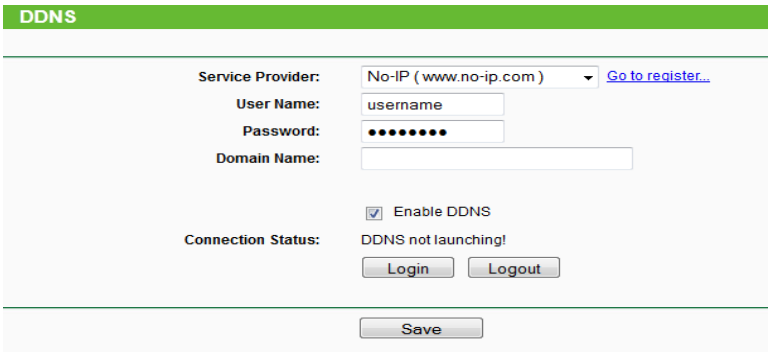


Figura 54. Configuración router

Fuente: Autores

ANEXO B.

MANUAL DE USUARIO

El siguiente manual es una pequeña guía de cómo cambiar o modificar los componentes que se usaron para el desarrollo de este prototipo y también la guía de la aplicación móvil. Procurando aclarar cualquier posible duda, ya sea con relación a la instalación y al mantenimiento adecuado para cada una de las piezas utilizadas o con su aplicación.

MANUAL DE LA APP

Con relación a la aplicación móvil, su uso es muy fácil, ya que consta de buscar el icono de la App del monitoreo remoto y acceder a ella, al ingresar se le pedirá al usuario que ingrese con sus datos, una vez hecho esto, accederá a las herramientas de la aplicación ya que en la página de inicio estarán todas las herramientas disponibles desplegadas en una barra de menú (ver figura 55), a las cuales accederá simplemente pulsando sobre el nombre de la herramienta a la cual quiere ingresar entre las cuales están inicio, GPS, fotos, streaming y registro de actividades. Además para cerrar sesión el usuario simplemente tiene que cerrar la aplicación y con esto ya se habrá salido automáticamente de la aplicación, y para ingresar y acceder a las herramientas tendrá que seguir los pasos mencionados anteriormente.

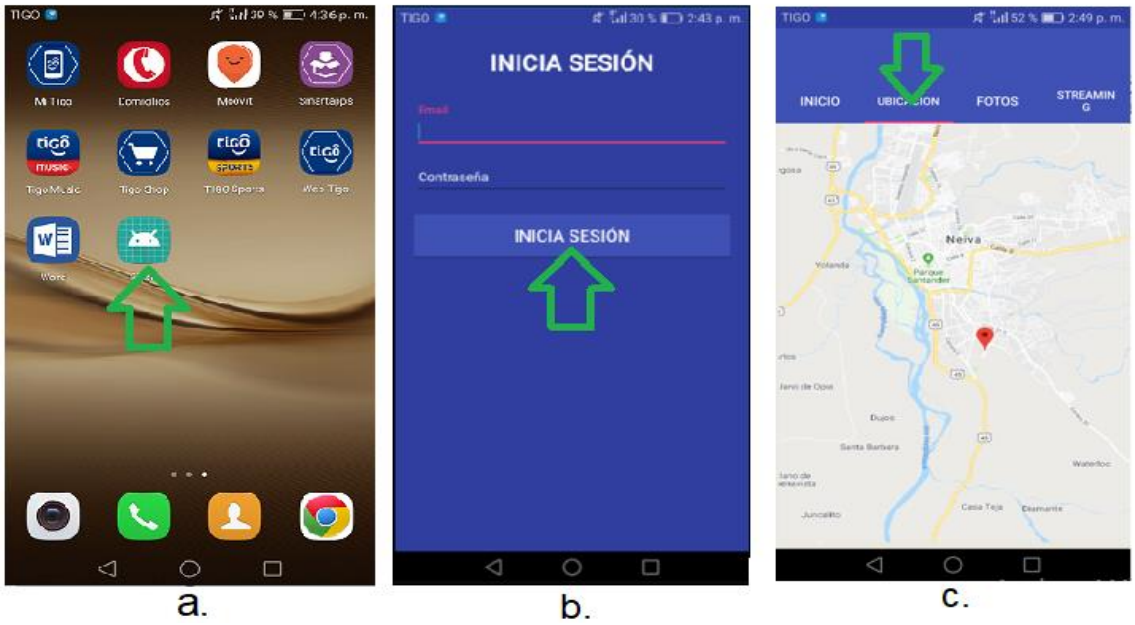


Figura 55. a) Lanzador App, b) Login, c) Barra del menú

Fuente: Autores

MANUAL DE LA TARJETA Y LAS PIEZAS CONECTADAS A ELLA

Este pequeño manual es una guía de instalación de los recursos necesarios para la puesta en marcha del sistema de vigilancia. La tarjeta posee el sistema operativo Raspbian, pero se le pueden instalar otros sistemas operativos (SO), eso depende del gusto de cada persona, pero se le recomienda instalar este sistema operativo. Este SO es muy completo y eficiente, una vez instalado el sistema operativo se le instalaron unos paquetes y librerías, además se le modificaron unos archivos para poder que funcionara correctamente cada uno de los dispositivos conectados a ella, por este motivo en este manual no se explicara cómo se hizo, puesto que al tener algunas modificaciones ciertos archivos, si el usuario los altera, el prototipo de monitoreo remoto podría dejar de funcionar correctamente y dejar vulnerable el vehículo; por esta razón en la aplicación diseñada para funcionar con este sistema se deja el contacto de los creadores, para cualquier inquietud, modificación o daño con relación a lo mencionado, contactarnos e ir a realizar una visita técnica para solucionar los inconvenientes, ya sean físicos o de programación. Sin embargo, decidimos dejar unos pines previamente programados para anexar en este caso sensores de movimiento, o en caso de dañarse el pin donde está conectado el sensor simplemente el usuario tendría que cambiar el sensor de pin.

A continuación se puede apreciar más detalladamente como hacer esto, como se puede observar en la figura 9 de este libro encontraremos la nomenclatura de los pines de la tarjeta. Están disponibles para hacer la conexión directa los pines GPIO 17 y GPIO 21 (ver figura 56-a) para conectar el sensor de movimiento que esta enlazado con la cámara de la raspberry y la envío del mensaje de alerta al usuario (funcionamiento: el sensor se activa, la cámara toma la foto, se envía el mensaje de alerta), también se dejan habilitados los pines GPIO 24 y GPIO 26 (ver figura 56-b), pero en esta ocasión el sensor de movimiento se activará, y una vez hecho esto, solo enviará el mensaje de alerta al usuario, esta programación de pines está basada en GPIO.setmode (GPIO.BCM) para ambos casos.

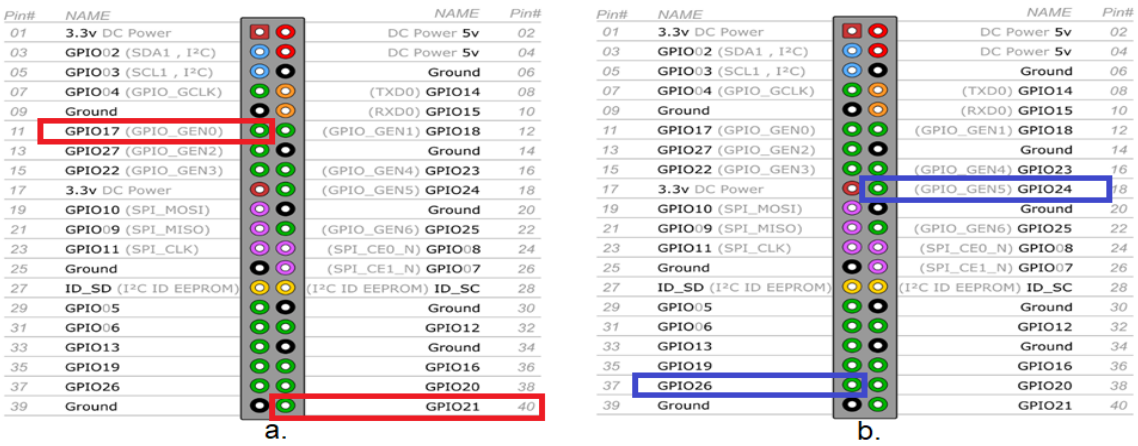


Figura 56. Pines habilitados

Fuente: Autores

Para el GPS, es recomendable que se use el Ublox Neo-m6, ya que este es muy económico y tiene un margen de error que consideramos aceptable que es de ± 15 m; los pines del GPS vienen nombrados como VCC para la alimentación, GND como polo a tierra, TX como trasmisión de datos y RX como la recepción de datos.

Una vez que ya está todo preparado, toca hacer las conexiones de hardware, se recomienda tener apagada la tarjeta (raspberry pi 3) y se hace de la siguiente manera (ver figura 57):

GPS a Tarjeta Color del Cable

- VCC a Pin 1 (3.3 V)----------Rojo
- GND a Pin 6 (GND)----------Negro
- RX a Pin 8 (GPIO 14 “TX”)--------Amarillo
- TX a Pin 10 (GPIO 15 “RX”)------Azul

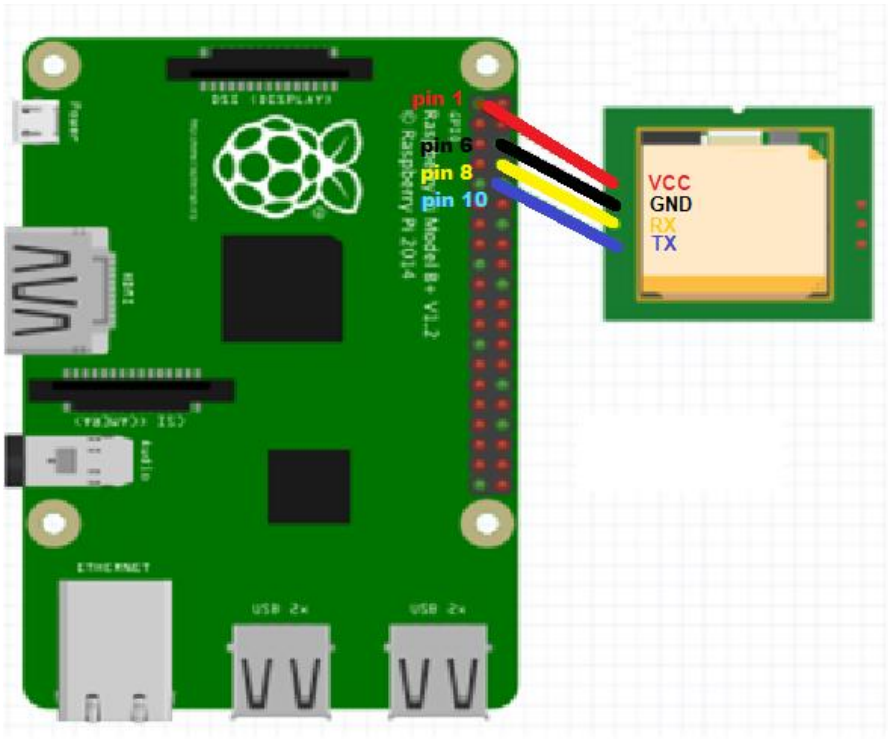


Figura 57. Conexión GPS

Fuente: Autores

La conexión con la cámara de la raspberry es muy sencillo, ya que la tarjeta trae un puerto para esta camara y lo unico que se debe hacer es conectarla a este puerto y ya estará funcionando, para la camara Ip se necesita agregar el host que se creó en la configuracion interna de la cámara para poder ver la transmision de la camara en el host que se creo con NO_IP en este caso, pero puede ser cualquier host, que tenga los puertos requeridos abiertos para poder esa transmisión de video.

ANEXO C.

CODIGOS UTILIZADOS

```python
import os
import gps
import psycopg2

session = gps.gps("localhost", "2947")
session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)

conn = psycopg2.connect(database="d64d4qgipltgpq", user="ssiypvsomeofgk",
 password="f108d43aa7ac52c236fe337c5226dfd2e9f8a75a1c2e142052fa70c5372bbd6c",
 host="ec2-54-235-86-244.compute-1.amazonaws.com",
 port="5432")
cur = conn.cursor()

try:
    while True:
       report = session.next()
       if report['class'] == 'TPV':
          if hasattr(report, 'lat'):
           #print 'Latitud: ' + str(report.lat)
           'Latitud: ' + str(report.lat)
          if hasattr(report, 'lon'):
           #print 'Longitud: ' + str(report.lon)
           'Longitud: ' + str(report.lon)
          cur.execute("TRUNCATE gps CASCADE;")
          La = str(report.lat)
          Lo = str(report.lon)
          cur.execute("INSERT INTO gps(gps_latitud, gps_longitud) VALUES (%s, %s)",(La, Lo))
          conn.commit()
       #cur.close()

except KeyboardInterrupt:
   print 'Datos GPS'
   conn.commit()
   cur.close()
```

Figura 58. Código GPS + conexión a base de datos en Python

Fuente: Autores

```python
import os
import RPi.GPIO as GPIO
import time
import psycopg2
import picamera

GPIO.setmode(GPIO.BCM)
PIR_PIN = 21
GPIO.setup(PIR_PIN, GPIO.IN)

conn_string = "host='ec2-54-235-86-244.compute-1.amazonaws.com' dbname='d64d4qgipltgpq' user='ssiypvsomeofgk' password='f108d43aa7ac52c236fe337c5226dfd2e9f8a75a1c2e142052fa70c5372bbd6c' port='5432'"
conn = psycopg2.connect(conn_string)

try:
    while True:
            if GPIO.input(PIR_PIN):
                with picamera.PiCamera() as picam:
                    picam.start_preview()
                    picam.capture('image.jpg')
                    picam.close()
                print ("MOVIMIENTO DETECTADO")
                os.system('echo "Alerta!! se recomienda verificar su vehiculo" | mutt -s "sensor de movimiento" joleacud@gmail.com')
                #time.sleep(10)
                mypic=open('image.jpg','rb').read() #abre el archivo y lo lee.
                cur = conn.cursor()
                cur.execute("INSERT INTO foto(nombre,c2imagen) VALUES (%s,%s);", (time.strftime("%d-%m-%Y %H:%M:%S"), psycopg2.Binary(mypic)))
                conn.commit()
                cur.close()

except KeyboardInterrupt:
        print " SENSOR DESACTIVADO"
        GPIO.cleanup()
```

Figura 59. Código sensor de movimiento, mensaje de alerta y foto + conexión con base de datos en Python

Fuente: Autores

```python
import os
import RPi.GPIO as GPIO
import time
from time import gmtime, strftime

GPIO.setmode(GPIO.BCM)
PIR_PIN = 24
GPIO.setup(PIR_PIN, GPIO.IN)

try:
    while True:
    if GPIO.input(PIR_PIN):
    timex = strftime("%d-%m-%Y %H:%M:%S", gmtime()) # arreglar la hora
    print timex + " MOVIMIENTO DETECTADO"
    os.system('echo "Alerta!! se recomienda verificar su vehiculo" | mutt -s "sensor de movimiento" joleacud@gmail.com')
    time.sleep(10)

except KeyboardInterrupt:
        print " SENSOR DESACTIVADO"
        GPIO.cleanup()
```

Figura 60. Código sensor de movimiento y mensaje de alerta en Python

Fuente: Autores