UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

TESIS DE GRADO

“INTERNET DE LAS COSAS,
CONTROL Y SEGUIMIENTO DE UN AUTOMÓVIL”

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

AUTOR: IVAN QUISPE CHOQUE
TUTOR METODOLÓGICO: PH.D. YOHONI CUENCA SARZURI
ASESOR: ING.GASTON CESAR BELTRAN VILLALTA
ASESOR ADJUNTO: LIC. REYNALDO JAVIER ZEBALLOS DAZA

LA PAZ – BOLIVIA

2016

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA.

A mis padres Julio y Virginia , porque creyeron en mí y me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ellos, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mí, fue lo que me hizo ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

A mis hermanos Handy y Alison, por haber fomentado el anhelo de triunfo en la vida.

A mi hija Aylin por darme el aliento para seguir adelante.

A mis asesores y docentes quienes nunca desistieron al enseñarme, aun sin importar que muchas veces no ponía atención en clase, a ellos que continuaron depositando su esperanza en mí.

A mis amigos y amigas quienes fueron un gran apoyo emocional durante el tiempo en que escribía esta tesis.

A todos los que me apoyaron para escribir y concluir esta tesis.

Para ellos es esta dedicatoria de tesis, pues es a ellos a quienes se las debo por su apoyo incondicional.

Agradecimiento

Le agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes y experiencias

Agradecer sinceramente a mi Tutor Metodológico PH.D. Yohoni Cuenca Sarzuri por guiarme en la realización de la presente tesis. Al mismo tiempo Agradecer al Ing. Cesar Beltran Villalta y al M.SC. Reynaldo Zeballos Daza por los concejos y el tiempo dedicado a la revisión del presente trabajo.

Agradecer de por vida a mi familia porque siempre estuvieron apoyándome.

A ti Nidya que desde el cielo me brindaste paz, serenidad, firmeza y cada día te sentía a mi lado acompañándome haciéndome sentir que solo necesito confiar en mí para lograr alcanzar mis metas

.

RESUMEN

Desde los años 90 la demanda de automóviles ha ido creciendo notablemente muchas familias adquieren sus automóviles haciendo un gran esfuerzo ahorrando dinero, pero así como el parque automotor creció también creció la inseguridad de los automóviles y por esa razón los sistemas de seguridad para automóviles nacieron. En el anterior milenio se podía observar encadenar el volante de conducción para trancar la dirección, después de algunos años se inventaron bloqueadores de volantes, de palancas de caja de transmisión y en un momento salió las alarmas que daban un sonido fuerte para ahuyentar al delincuente, pero ahora con el avance de la tecnología se puede ir un paso más adelante.

Hoy en día se cuenta con empresas que dan ese servicio de seguridad a los propietarios, dichas empresas venden e instalan sus dispositivos de seguridad y además se debe realizar un pago mensual por el servicio que ofrecen.

En los últimos años se vio una nueva tendencia de ir conectando cualquier tipo de aparato a distancia, y así apareció la Internet de las Cosas, ahora es posibles conectar las luces de casa aparatos electrónicos sin importar la distancia, entonces también es posible conectar el automóvil.

La presente Tesis tiene como objetivo dar más confianza a los propietarios de automóviles que deseen instalarlo, con el diseño e implementación de una aplicación y su controlador que sea capaz de mandar una alerta al propietario sin importar la distancia que los separe vía Smarthphone cuando la misma este activada, también otorgara al propietario la posibilidad de controlar el encendido del automóvil.

El controlador está basado en Arduino y el Shield SIM908 el cual consta de una ranura para chip GSM y un GPS incluido. La aplicación es realizada para dispositivos Android, un Smartphone que tengan un GPS incluido. La conexión entre el controlador y el Smartphone es a través de SMS.

ABSTRACT

Since 90´s the cars requirement has been growing eminently, many families purchase their cars by making a great effort saving money, but the automobile depot has grown, the automobile insecurity has also grown, that is the reason why the automobile security system rose. In the last millennium it could be observed the enchainly of the auto flywheel in order to stride the car direction, besides a few years it was invented an flywheel lock and then an alarm , which could make an loudly sound to banish the felonious, but actually it is able to go an forward step by the developing technology.

Nowadays there are enterprises which offer an security service to the car owners, these companies sell and install their security devices, and their customers must pay for the service monthly.

The last years it appeared a new tendency, which consist in connect any remote device, and thus the things Internet appeared, nowadays it is possible to connect the house lights to electronic devices no matter the distance, so it is possible to use the same system in cars.

This thesis has the as objective to give more security to the car owners who want to install the remote device, which by an application implement and a Smartphone controller allows a message sending to the car proprietary no matter the distance when the Smartphone is activated. And also, the driver will be able to control its turning on.

The controller is based on Arduino and the Shield SIM908, which has a little groove to insert a chip GSM and GPS included. This application is done to Android devices, un Smartphone with GPS included. The connection between the controller and the Smartphone is by SMS.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE PROGRAMAS

1. MARCO REFERENCIAL

1.1. Introducción

Actualmente se puede observar que no existe seguridad al momento de parquear un automóvil, muchas personas según sus posibilidades o la distancia donde viven, pueden optar entre estacionarlos en parqueos privados, que a veces son solo terrenos adaptados, o estacionarlos en la calle más cercana al trabajo, en la cual exista espacio, lo que significa cinco o más cuadras de distancia, pero en ambos casos es muy inseguro, ya que en la calle se deja el automóvil a un cuidador bajo el supuesto que va a cuidarlo, pero cuando el propietario se va el cuidador se aleja y solo vuelve cuando es hora de recoger el automóvil, o en el caso de los parqueos muchos tienen la regla de no hacerse cargo de algún robo en sus instalaciones.

Existen alarmas que pueden ser instaladas en un automóvil, las cuales ya son conocidas por los ladrones, quienes en algunos casos, solo activan la alarma n-veces para que el dueño la desactive, y así se piense que está dañada, en ocasiones activan la sirena para ver si el dueño esta cerca o no del automóvil, ya que como se mencionó en el párrafo superior se deja el automóvil lejos del lugar de estancia del dueño, y de esta forma efectuar el robo.

De las alarmas que existen, la mayoría solo activa una sirena, otras cuentan con un botón puesto en un lugar oculto, que ahora son los más comunes y los ladrones conocen su ubicación, y alguna empresa ofrece el rastreo satelital, pero esto implica un costo adicional elevado.

Existen robos de automóviles cada día, los cuales en ocasiones son realizados por instrucciones de los privados de libertad (desde la cárcel) o ladrones que andan sueltos, los cuales hacen llamadas para que el propietario pueda recuperar el automóvil, pidiendo dinero a cambio de recuperar su automóvil, luego citándolos en lugares alejados para que entreguen el dinero personalmente, lo que implica mas inseguridad, o indicando que depositen el dinero en un determinado lugar. Lo que no da la certeza a la devolución del automóvil en el lugar acordado.

Cuando se deja un automóvil en una calle puede dar lugar al robo completo del mismo, pero en un parqueo ese tipo de robo es mínimo, ya que ahí no se puede sacar el automóvil fácilmente, por consiguiente se puede hacer el hurto de autopartes tanto internas como ser radios, sensores MAF(sensor de entrada de aire al motor), ECU(unidad de control de motor), y externas como llantas, retrovisores y faroles.

En ocasiones se piensa que el automóvil fue sustraído, pero existe alguna situación en la cual familiares hacen uso del mismo sin ningún permiso, lo que implica desconocer quién es la persona que lo maneja, desconociendo si el automóvil fue robado, de algún modo se vería atrayente tener la foto del ladrón o persona que va manejando el automóvil, en caso que fuese un ladrón tener una prueba de que esa persona robó el automóvil y de esta forma tener más elementos de convicción para llevarlo a la justicia.

1.2. Antecedentes

Desde que se creó y se puso en la calles a los automóviles empezaron a aparecer personar que se dedicaban a robar autos o sustraer piezas de los mismos. En la revista Popular Mechanic(1920) se publico un artículo en el que un inventor del estado de Nebraska creo el primer sistema de alarma de auto y el diseño era un interruptor de llave (switch), una caja de acero con un embobinado y una sirena. Este sistema estaba puesto en uno de los ejes y al dar vueltas activa el campo magnético generaba voltaje y sonaba la sirena, claro esto simple y cuando su dueño la activara con el interruptor de llaves. Desde entonces los dueños de los automóviles han optado por recurrir a distintas formas de asegurarlos, algunos con bastones para trabar el volante o la palanca de dirección, swicht manuales de corte de energía, sirenas y otros.

Con el transcurrir del tiempo y el avance de la internet, ya se ingreso a la etapa del Internet de las Cosas, lo cual conlleva a la inquietud de tener más seguridad en los bienes de las personas, en este caso el Control y Seguimiento de un Automóvil. Desde hace unos 30 años que se viene trabajando con la idea de hacer un poco más interactivos todos los objetos de uso cotidiano. Ideas como el hogar inteligente, también conocido como la casa del mañana, han evolucionado de una forma veloz, para entrar al Internet de las cosas (Hipertextual, 2014).

Si los libros, refrigeradores, lámparas, garajes, partes automotrices, etc. estuvieran conectados a Internet y equipados con dispositivos de identificación, no existirían, en teoría, cosas fuera de stock o carencia de medicinas o caducadas, sabríamos exactamente la ubicación, cómo se consumen y se compran productos en todo el mundo; el extravío sería cosa del pasado y sabríamos qué está encendido o apagado en todo momento (Gershenfeld, Krikorian, & Cohen, 2004).

1.2.1. Antecedentes institucionales

En la actualidad existe la necesidad de contar con seguridad para nuestro automóvil, la cual es muy necesaria, pero muy poco aplicada, el único control en el que muchos confían es la instalación de alarmas, sirenas que se activan y desactivan.

Una institución como: CAJA PETROLERA DE SALUD, en el área de transporte de Ejecutivos, tiene un control parecido al que se quiere proponer en este proyecto, pues esta institución lleva un control de sus automóviles, el cual va dirigido a verificar que los conductores no excedan la velocidad, verificar que estén conduciendo en las rutas y horarios permitidos, los choferes no pueden llevar sus autos como uso personal para ir a sus casas, y queda prohibido disponer de los automóviles fuera de horarios de oficina.

Otra institución que maneja un control de automóviles es el SETRAM[1], de la Ciudad de La Paz que cuenta con un control y seguimiento de cada uno de sus buses, sin embargo no se controla el robo de los mismos.

La EMPRESA PRIVADA ASR, la cual ofrece seguridad en casas, negocios, pero también hace conexiones de alarmas a vehículos.

1.2.2. Antecedentes de proyectos similares

En el artículo de Huertas (2014), "Diseño e implementación de un módulo alarma para el monitoreo y control vehicular a través del sistema GSM y GPS", de la Universidad de las Fuerzas Armadas ESPE[2], de Ecuador, se hace un estudio de como implementar un Control y Seguimiento vehicular por medio de redes GSM y GPS.

También en la Tesis, de Esparta (2013), titulada "Taxímetro digital touch con seguridad antirrobo vehicular incorporada", de la Universidad Técnica del Norte, de Ecuador, consiste en un prototipo de taxímetro diseñado específicamente para el servicio de transporte en taxis, con el objetivo de determinar el costo del servicio del transporte en taxi, además está incorporado con varias funciones para proteger al vehículo contra robos.

En la tesis de Castañeta y Carrion (2014), "Sistema de alarma y monitoreo vehicular controlado por un dispositivo móvil utilizando la conexión de redes celulares", de la Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, de Bogotá, Colombia, se presentan los datos teóricos y prácticos, así como esquemas y demás diseños, involucrados en el desarrollo de un dispositivo de un sistema de seguridad de vehículos que puede ser administrado de manera remota, utilizando un dispositivo móvil y realizando la interconexión por medio de las redes de telefonía celular. Con esta propuesta se reducen costos, en comparación con los demás mecanismos de seguridad de vehículos como son los seguros o sistemas de seguridad.

[1] Servicio de Transporte Municipal de la ciudad de La Paz
[2] Antes llamado Escuela politécnica del Ejercito

Marquez (2008), en su proyecto de grado "Diseño de un sistema de seguridad y monitoreo de vehículos", de la Universidad Simón Bolívar, Facultad de Ingeniería Electrónica, de Caracas Venezuela, abarca todos los aspectos del proceso de diseño de un sistema de seguridad y monitoreo de vehículos de calidad comercial que permite, de manera remota, conocer la ubicación geográfica en caso de robo para facilitar su recuperación. Además ofrece, entre otras funciones, el almacenamiento continuo de los recorridos hechos por el vehículo, incluyendo la supervisión y control de velocidad a lo largo del viaje, lo cual resulta útil en vehículos de carga y de transporte público.

Astudillo y Delgado (2012), en el trabajo "Sistema de localización monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS", realizado en la Universidad Politécnica Salesiana, de Argentina, que describe la implementación de un sistema de localización, monitoreo y control vehicular, cuyo objetivo es el almacenamiento, procesamiento y gestión de los datos que son enviados desde los dispositivos vehiculares hacia los servidores alojados en la nube, facilitando las consultas de reportes, historial y monitoreo desde cualquier dispositivo que tenga conexión de Internet. La técnica empleada para el desarrollo de la aplicación, es la utilización de los protocolos GPS/GSM/GPRS en conjunto con los servidores de aplicaciones web y sockets.

Y Valverde (2014), en el proyecto "Monitoreo vehicular mediante módulos GPS y GPRS controlados por Intel galileo", de la Universidad De Costa Rica, Facultad de Ingeniería, en la cual, se describe el desarrollo del prototipo de un sistema de posicionamiento de buses, alrededor de un Intel Galileo. El trabajo hace a partir de la iniciativa de un grupo de trabajo de Componentes Intel Costa Rica por realizar una aplicación móvil de información de rutas y posicionamiento de buses; se requiere de una plataforma incrustada que viaje con los vehículos y que pueda proveer al servidor remoto que alimenta la aplicación móvil de los datos de ubicación de los autobuses. Además, se impulsa el proyecto debido a la necesidad de evaluar la tarjeta de desarrollo Intel Galileo en relación a otros productos similares del mercado. Se realiza un prototipo basado en el Galileo aprovechando las tecnologías GPS y GPRS, que funciona apropiadamente para proveer de datos de posicionamiento al servidor remoto.

1.3. Planteamiento del problema

Cuando se deja un automóvil en la calle puede dar lugar al robo completo del mismo, pero en el parqueo ese tipo de robo es mínimo, ya que ahí no se puede retirar el automóvil fácilmente, pero si se puede hacer el robo de autopartes tanto internas como radios, sensores, ECU. y externas como llantas, retrovisores.

Después de realizar consultas a varios propietarios que sufrieron algún tipo de robo de autopartes o el automóvil completo, se pudo evidenciar la inseguridad y la falta de control que existe en los parqueos y calles de la ciudad.

No hay una forma adecuada de tener dicho control sin que estemos pensando casi todo el tiempo en cómo estará , que está pasando al rededor de él o si alguien ingresó a nuestro automóvil.

La gran demanda de estacionamientos llenan los parqueos, o los mismos son alejados y cuando se enciende la alarma no se está cerca para verificar lo que sucede en el automóvil.

Se prevé que hay seguridad en los parqueos, sin embargo también ocurren robos principalmente de autopartes.

Muchas veces el automóvil es robado, y no sabemos cómo ubicarlo, o simplemente algún familiar saca el auto y se piensa que fue robado, no conocer la ubicación exacta pone más nerviosos al propietario.

Con los problemas descritos anteriormente el problema principal que se plantea en este trabajo es el siguiente:

"Existe inseguridad y falta de control en automóviles en parqueos y calles generando, un alto índice de robos de autopartes, internas y/o externas, incluso el robo total del automóvil, generando incertidumbre, estrés molestias y costos".

1.4. Definición de objetivos

1.4.1. Objetivo general

Diseñar e implementar una aplicación para el seguimiento, control y seguridad de automóviles, que con la ayuda de un controlador, permitirá reducir la inseguridad, robo de autopartes y automóviles, el cual será verificado mediante diferentes pruebas una vez que se tenga el prototipo instalado.

1.4.2. Objetivos específicos

Los objetivos secundarios que deseamos lograr son:

- Desarrollar una alarma que contenga un GPS para localizar el automóvil.
- Desarrollar control a distancia, para cortar la alimentación de gasolina o/y energía eléctrica.

- Desarrollar un modulo de envío de alertas al SmartPhone al momento de que el auto tenga un cambio de estado, ya sea por que se abre una puerta o se produzca un movimiento brusco.
- Desarrollar un modulo para el "Rastreo del Automóvil" y verificar si esta en movimiento.

1.5. Hipótesis

El Diseño de una aplicación para realizar el seguimiento y control de automóviles, que apoyándose en un controlador construido en base a Arduino, ayudará a reducir la inseguridad al momento de dejarlos estacionados en calles o parqueos, evitar el robo de los mismos y de sus autopartes, se realizaran pruebas de control a través del prototipo instalado.

1.5.1. Variable independiente

Diseñó de una aplicación de seguimiento y control de automóviles

1.5.2. Variables dependientes

Reducir la inseguridad de los propietarios en relación a su automóvil.

En la actualidad hay 384.941 automóviles, en la ciudad de La Paz, según datos de él Instituto Nacional de Estadística (INE), y de acuerdo a reportes de DIPROVE se reporta alrededor de un robo por día, de los cuales un 95% son realizados en vía pública.

Evitar el robo de los automóviles y/o de sus autopartes.

En los últimos tres años se reportaron 400 robos de automóviles en 2013, 358 robos en 2014 y 316 robos hasta la fecha, de acuerdo a DIPROVE, también reportaron que en La Paz se reportaron 305 robos de autopartes en 2013 y 273 en 2014, a continuación en la Tabla 1.1, se observa el porcentaje de robo de autopartes en un automóvil.

Tabla 1.1: Porcentaje de Rodo de Autopartes en los ultimos tres años.
Tomado de datos oficiales de DIPROVE(2015)

AUTOPARTE	% DE ROBO
Retrovisores	**56 %**
Radios, ECU's, MAF y flujometros	**41%**
Faroles y Stops	**35%**

1.6. Justificación

1.6.1. Justificación económica

Para poder adquirir un automóvil hoy en día hay que realizar un enorme gasto económico, se debe ahorra por años, dejar un automóvil en un parqueo cuesta en promedio Bs. 5,00 por, hora haciendo una estimación presupuestaria, en una semana una persona gastaría Bs. 200,00 solo en parqueo pero eso no asegura que esté seguro.

Dejarlo estacionado en la calle resulta mucho más económico, pero a la larga puede que roben autopartes o el automóvil completo, y para recuperarlos se debe pagar sumas altas de dinero, ya sea negociando en tiendas legales, o haciendo tratos con los mismo ladrones para que se devuelva las autopartes o el mismo automóvil.

Tener una aplicación que reduzca esos costosos sistemas de rastreo satelital, su implementación seria razonablemente económica y además no se tendría que pagar mensualmente sumas elevadas a empresas de seguridad y control, solo se debería contar con una línea 4G para el Smartphone, de cualquier empresa de telecomunicaciones.

1.6.2. Justificación social

Con el desarrollo de esta Tesis, las personas que deseen implementarlos en sus automóviles, se sentirán más seguras al momento de dejarlo parqueado, los propietarios podrán desempeñar sus funciones con tranquilidad, y se podrá reducir el comercio informal de autopartes que existe en el mercado local.

1.6.3. Justificación tecnológica

El sistema de control y seguimiento de un automóvil, basado en el INTERNET DE LAS COSAS, es algo novedoso, ya que su fin es realizar conexiones inalámbricas a distancia. Se aprovecharía nuestro Satélite TK-SAT1, con el cual ahora ya se puede llegar a cualquier parte de Bolivia, además los nuevos convenios que están realizando para mejorar el internet y las conexiones a las mismas, se está reduciendo los costos de las conexiones y acceso a internet.

Con esta aplicación se logrará la comunicación entre el Automóvil y el Propietario, para ello usaremos una placa Arduino, que hoy es usada por la comunidad y basado mayoritariamente en procesadores RISC[3] de Atmel[4] que permite que cualquier persona con unos conocimientos básicos de

[3] Del ingles Reduced Instruction Set Computer en español Computador con conjunto de Instrucciones Reducidas.

electrónica y programación pueda diseñar e implementar sus ideas. Se creara una aplicación Android para la conexión al Smartphone, se buscará hacer conexiones de sensores para determinar si se abre las puertas, o si hay un cambio brusco en el estado de reposo del automóvil, y hacer corte de energía para bloquear completamente el encendido del mismo.

1.7. Alcances y límites

La presente tesis comprende el diseño y construcción de una aplicación y controlador que permitirá incrementar la seguridad de automóviles de una forma dinámica y efectiva.

1.7.1. Alcances

La ejecución y creación de la aplicación deberá aumentar la confiabilidad para los propietarios de vehículos que decidan instalarlo, al momento de dejar sus automóviles parqueados usando su Smartphone, para comprobar que el mismo este seguro en cualquier momento.

Consideraremos los siguientes módulos:

- Se implementará un modulo para mandar alertas al momento de que se abra una puerta.
- Se implementará un modulo para mandar alertas al momento de producirse un cambio en el estado de reposo del Automóvil.
- Se implementará un modulo para bloquear el auto de tal modo que si el auto sea arrancado podamos bloquearlo, pero teniendo en cuenta que antes recibiremos una notificación que está en movimiento.
- Se implementará un modulo de captura de imagen de la persona que conduce el automóvil.

1.7.2. Límites

Se trabajará solo con automóviles a inyección electrónica, los cuales cuentan con ECU, MAF, Flujómetros, Radios entre las piezas más comunes que son robadas actualmente, automóviles fabricados a partir del año 1998.

[4] Atmel Corp. es una compañía de semiconductores, fundada en 1984, su línea de productos incluye micro controladores.

Se trabajará con placas Arduino, y sensores para ver si hay presencia en el interior del automóvil, otro sensor de dióxido de carbono para verificar si el automóvil esta encendido o no.

La aplicación estará diseñada para Smartphone con Sistema Android.

En una primera fase se aplicara en el área metropolitana de La Paz, se podría ampliar a nuestro país, debido a que el chip GSM funciona a nivel nacional.

1.8. Aportes

1.8.1. Práctico

El presente trabajo de investigación será novedoso por que presentará un modo de interconexión seguro entre automóvil y su Propietario, usando las nuevas tecnologías que van creciendo a grandes pasos.

Asimismo la investigación pretende ser en un futuro una base de referencia para estudios similares donde se quiera usar el internet de las cosas, para poder realizar un control no solo de automóviles, también de muchos bienes que con el tiempo van a necesitar de un control.

1.8.2. Teórico

De igual forma la investigación tratará de anular los intermediarios de seguridad entre automóvil y propietario, que son las empresas que brindan ese servicio de control y seguimiento de automóviles, solo será necesario una instalación, la configuración de claves para su seguridad, y del Chip que se coloque al vehículo por medio del dispositivo Arduino, se mostrará la base de que es posible hacer la conexión para que los propietarios puedan bloquear un automóvil a distancia. Y hacer un rastreo del mismo.

1.9. Metodología

Esta investigación obedece a una investigación aplicada exploratoria debido a que los sistemas a efectuar, planteados en este documento requieren de un estudio tanto teórico como práctico; ya que este tipo de sistemas a desarrollar e implementar ya se encuentran en el mercado, pero con la diferencia de que se realizarán con tecnología relativamente económica y de mayor alcance para todo tipo de público.

En particular se tienen previstos las siguientes actividades para alcanzar cada uno de los objetivos planteados:

- Integrar diferentes dispositivos electrónicos al interior de un vehículo. Búsqueda de información acerca de que es la tecnología GSM/GPS. Realización del listado de materiales a utilizar durante el desarrollo del prototipo. Ejecución de la simulación del prototipo. Montaje de los circuitos simulados en el computador y expectación del comportamiento de los componentes electrónicos a medida que pasa el tiempo.

- Estudiar acerca de la tecnología disponible en el mercado para la realización de la comunicación entre el prototipo a implementar en el vehículo y el usuario. Selección de la tecnología adecuada para la etapa de comunicación. Búsqueda de video tutoriales acerca de los módulos de radio frecuencia de la etapa de comunicación. Activación y puesta a punto de los módulos de radio frecuencia.

- Estudiar e implementar una estrategia de control adecuada para el manejo de dispositivos al interior de un vehículo. Selección del compilador adecuado para la creación, desarrollo y finalización de la estrategia de control. Generación de lluvia de ideas de la posible estrategia de control a implementar. Puesta a punto de la estrategia de control a través de pruebas de ensayo y error. Programación del módulo a utilizar. Montaje y ensamblen las etapas de potencia, regulación, comunicación y finalmente control. Integración de las tarjetas electrónicas de la etapa de potencia, regulación, comunicación y control para el prototipo.

2. MARCO TEÓRICO

2.1. Introducción

El Control y Seguimiento de un automóvil, sirve para que predomine la sensación de confianza de los Propietarios, entendiendo a Control como: Es tener bajo supervisión una variable de un determinado proceso, la cual al final arroja un resultado, dicho resultado llega a un controlador y compara el resultado obtenido al final del proceso con un valor predeterminado y así lograr corregir el margen de error al final del proceso (RAE, 2014). Y se entenderá al Seguimiento para indicar la acción de seguir a algo o bien a alguien, siendo este sentido un sinónimo de uso popular del concepto de persecución. Y también se usa esta misma palabra para indicar la observación y vigilancia pormenorizada y profunda (Definicion ABC, 2007).

De tal modo se entenderá a Control y Seguimiento al poder controlar el automóvil a distancia poder activar y desactivar algunos módulos, hacer el seguimiento del mismo una vez que sea alejado de su posición de reposo, para averiguar su ubicación exacta.

Se definirá Parqueo como acción y efecto de estacionar o estacionarse. Se usa especialmente hablando de los vehículos, al lugar o recinto reservado para estacionar vehículos, lugar donde puede estacionarse un automóvil, también se conoce como estacionamiento al espacio físico donde se deja el vehículo por un tiempo indeterminado cualquiera y, en algunos países hispanohablantes, también al acto de dejar inmovilizado un vehículo (Lexicoon, 2015).

2.2. Internet de las cosas

Para poder conectar los dispositivos a distancia debemos conocer una nueva rama de Internet denominado IdC, El Internet de Las Cosas(o IoT Internet of Things), que representa la próxima evolución de Internet, que será un enorme salto en su capacidad para reunir, analizar y distribuir datos que podemos convertir en información, conocimiento y en última instancia, sabiduría. En este contexto, IdC se vuelve inmensamente importante (Evans, 2011).

El Internet de las cosas no es una idea nueva. A principios de los años noventa, Mark Weiser, director científico del Xerox Palo Alto Research Center, introdujo el concepto de "computación ubicua", que abogaba por un futuro con la integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados, es decir, que formaría parte integral de nuestra vida diaria y resultaría transparente para nosotros (Weiser, 1991). Weiser no acuñó el término «Internet de las Cosas», esta se atribuye al Center del Massachusetts Institute of Technology a finales de los años noventa (Ashton, 2009).

Un mundo en el que todos los aparatos electrónicos estarán interconectados y todo objeto, ya sea físico o electrónico, estará etiquetado electrónicamente desprendiendo información acerca de él mismo. Prevemos el uso de etiquetas físicas(tags) que permitan, a distancia y sin necesidad de contacto entre dichos objetos, el intercambio de información permitiendo así que todos los objetos actúen como nodos en un mundo físico totalmente interconectado (Sarma, Brock, & Ashton, 2000).

Sin embargo, la idea del IdT ha tomado importancia gracias a la rápida evolución de la electrónica durante los últimos años.

En los próximos años se espera un gran aumento en el número de equipos de uso cotidiano interconectados, entre otras cosas, gracias a la inminente llegada de todos estos sensores inteligentes a nuestros hogares. La Figura 2.1 muestra un gráfico con las expectativas de crecimiento en vistas al año 2025 (Kurzweil, 2011).

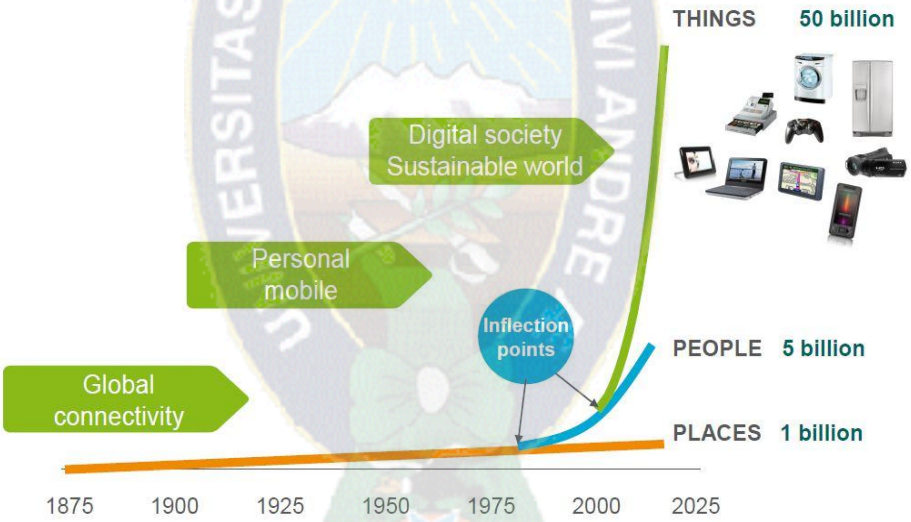


Figura 2.1: El aumento de personas conectadas 2020.
Claramente se notara que con el transcurrir de los años la conexines van aumentando y entre los 70's y hasta la fecha con mas fuerza.
Tomado de Ericsson(2010).

Paul Horn, científico distinguido en Residencia y vicerrector de Investigación senior de NYU[5], ex-vicepresidente senior de IBM y experto del Future Trends Forum[6], afirma que el mundo está siendo instrumentado e interconectado, a la vez que se vuelve más inteligente. Los objetos que forman parte de la vida cotidiana siempre han generado gran cantidad de información, pero esa información estaba

[5] Universidad de New York
[6] El Future Trends Forum es el único tanque de pensamientos multidisciplinar e internacional focalizado en innovación. Formado por más de 400 expertos, el Future Trends Forum anticipa y detecta las tendencias de innovación; analizando su impacto en la sociedad y modelos de negocio.

fuera de alcance. Así cada vez se puede estar más interconectado y las personas y objetos pueden interactuar de manera completamente distinta. Hoy día hay 1.000 millones de usuarios de Internet, 4.000 millones de personas con teléfono móvil y una lista interminable de objetos (coches, electrodomésticos, cámaras, etc.) conectados a Internet de una forma u otra. A su alrededor, se construye entornos inteligentes capaces de analizar, diagnosticar y ejecutar funciones, eliminando posibles errores humanos... para bien y para mal. Por ejemplo, una red eléctrica inteligente es capaz de detectar sobretensiones y de dirigir la electricidad por caminos alternativos para minimizar apagones.

Una de las características más destacables de IdT es su esquema simple de recolección y envío de datos, es usado en todas las aéreas como se pude observar en la Figura 2.2, dónde pequeños sensores están siendo integrados en los objetos del mundo real y son instrumentos que proporcionan información de prácticamente todo lo que es posible medir.

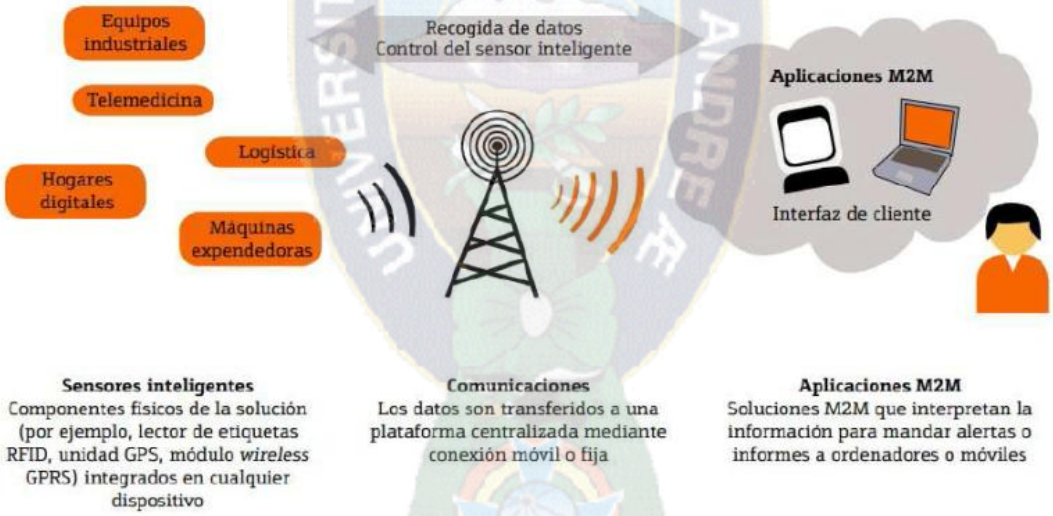


Figura 2.2: Cuando las Cosas se vuelven inteligentes.
Los sensores seran utilizados por distintas areas con el fin de recolectar datos, para trasnmitirlos hacia una aplicación para la toma de decisiones
Tomado de (Fernandez, 2011).

Para poder realizar la conexión con el automóvil debemos conocer la placa Arduino en la cual agregaremos un modulo GSM para lograr el envió de mensajes, además de un Shield GPS para obtener la ubicación del mismo, sensores de movimiento y de gas para verificar si el automóvil está en funcionamiento y un modulo de relee para activar o desactivar energía al automóvil.

2.3. Arduino

Arduino es una plataforma electrónica de hardware libre basada en una placa con un micro controlador. Con software y hardware flexibles y fáciles de utilizar, Arduino ha sido diseñado para adaptarse a las necesidades de todo tipo de público, desde aficionados, hasta expertos en robótica o equipos electrónicos. También consta de un simple, pero completo, entorno de desarrollo, que nos permite interactuar con la plataforma de manera muy sencilla.

Se puede definir por tanto como una sencilla herramienta de contribución a la creación de prototipos, entornos, u objetos interactivos destinados a proyectos multidisciplinares y multitecnología (Arduino, 2015).

A través de los años Arduino ha sido el cerebro de miles de proyectos, a partir de objetos cotidianos a los instrumentos científicos complejos. Una comunidad mundial de los fabricantes, estudiantes, aficionados, artistas, programadores y profesionales, ha reunido en torno a esta plataforma de código abierto, sus contribuciones han añadido hasta una increíble cantidad de conocimiento accesible que puede ser de gran ayuda para los principiantes como para expertos.

El software es de código abierto, y está creciendo a través de las contribuciones de los usuarios en todo el mundo. Los proyectos Arduino pueden ser independientes o se pueden comunicar con el software que se ejecuta en un computador. Los diseños de referencia de hardware están disponibles bajo una licencia de código abierto, es libre de adaptarlos a cada necesidad. La tarjeta de programación Arduino UNO viene lista para usar, basta con conectar la tarjeta al computador por medio del puerto USB o con un adaptador AC/DC, no requiere programador externo; Además es compatible con los diferentes módulos Arduino permitiendo de forma fácil adaptar comunicación inalámbrica.

El micro controlador en la placa Arduino se programa mediante el lenguaje de programación Arduino, basado en Wiring y el entorno de desarrollo Arduino, basado en Processing. Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

Processing es un lenguaje de programación de código abierto, para las personas que quieran crear imágenes, animaciones e interacciones. Inicialmente desarrollado para servir como un software como block de dibujo y para enseñar los fundamentos de la programación de computadora en un contexto visual. Processing también ha estado involucrado como una herramienta para generar trabajos finales profesionales (Fry & Reas, 2015).

Wiring es una plataforma abierta de prototipos electrónicos compuestos de un entorno de programación, una tarjeta de prototipo, documentación, personas con experiencia tanto principiante como nivel intermedio comparten sus ideas, conocimiento y su experiencia colectiva (Barragan, 2015).

Existen muchas versiones de Arduino como se observa en la Figura 2.3 en la cual se agrupan por 5 niveles de uso ya sea desde un principiante a un experto además que se identifican si son módulos, tarjetas, Shield, kits y accesorios.

Figura 2.3: Modelos de Arduino.
Con el trancurrir de los años fueron saliendo nuevos modelos y ahora se pueden agrupar por niveles de uso e identificación de su respectivo tipo.
Tomado de Arduino (2015).

2.3.1. Características Arduino uno

Especificaciones técnicas de Arduino Uno. El Arduino UNO es una placa electrónica basada en el Micro controlador ATmega328. Cuenta con 14 entradas/salidas digitales, 6 entradas analógicas, una de 16 MHz del oscilador de cristal, una conexión USB, un conector de alimentación, una cabecera

de ICSP, que es el acrónimo de la frase en inglés: "In Circuit Serial Programming"[7], y un botón de reinicio, en la Tabla 2.1 se muestra un resumen de las características de la Tarjeta Arduino.

Tabla 2.1: Resumen de características de ARDUINO UNO.
Tomado de Arduino(2015).

Micro controlador	ATMega328
Voltaje de Funcionamiento	5V
Voltaje de Entrada	7-12V
Limite Voltaje de Entrada	6-20V
Digital pines I/O	14(6 proporcionan una salida PWN)
Corriente de I/S de CC Pin	40mA
SRAM	2kb(ATMega328)
EEPROM	1kb(ATMega328)
Velocidad de reloj	16MHz

Alimentación de Energía. Arduino UNO puede ser alimentado a través de la conexión USB o con una fuente externa que se selecciona automáticamente, el poder puede venir de un adaptador de CA a CC (de pared) o la batería. El adaptador se puede conectar a un plug de 2.1mm centro-positivo en el conector de alimentación de la placa, los cables de la batería se pueden insertar en los encabezados de pin GND y Vin del conector de alimentación.

Los límites están entre los 6 y los 12 V. Como única restricción se debe conocer que si la placa se alimenta con menos de 7V, la salida del regulador de tensión a 5V puede dar menos que este voltaje y si sobrepasamos los 12V, probablemente dañaremos la placa.

Pines de alimentación.

VIN. Se puede suministrar tensión a través de este pin cuando el voltaje de entrada a la placa Arduino es diferente a 5 voltios de la conexión USB.

5V. La fuente de alimentación regulada para alimentar el micro controlador y otros componentes en el tablero, se suministra a través del puerto USB.

3V3. Proporciona una tensión de 3,3 V, y una intensidad máxima de 50 mA

[7] Programación serial en Circuito.

Entradas y Salidas del módulo. Cada uno de los 14 pines digitales en Arduino puede ser utilizado como una entrada o salida, usando pin Mode, digital Write y digital Read, operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tienen una resistencia interna pull-up de 20 a 50 kOhm.

Algunos pines tienen funciones especializadas:

Serie: Se utiliza para recibir (RX): 0 y transmisión (TX): 1 datos serie TTL que es la sigla en inglés de transistor-transistor logic, es decir, "lógica transistor a transistor". Es una tecnología de construcción de circuitos electrónicos digitales.

Interrupciones externas: 2 y 3, estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.

PWM: 3, 5, 6, 9, 10 y 11 proporcionan 8-bits de salida con la analogWrite función.

LED: 13. Led conectado al pin digital 13. Cuando el pin es de alto valor, el Led está encendido, cuando el pasador es bajo, es apagado.

Arduino tiene 6 entradas analógicas, cada una proporciona 10 bits de resolución (es decir 1024 valores diferentes).

Comunicación. Arduino tiene una serie de facilidades para comunicarse con un computador, otro Arduino, u otros micro controladores. El micro controlador ATmega328 que se puede observar en la Figura 2.4, ofrece UART TTL, que es el dispositivo que controla los puertos y dispositivos serie. (Beyond, 2010). Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo. de comunicación en serie, que está disponible en los pines digitales (RX) y (TX). El software de Arduino incluye un monitor de serie que permite que se envíe y reciba datos de texto a la placa Arduino a través del chip FTDI y la conexión USB al computador.

Figura 2.4: Chip Atmega.
El micro controlador ATmega328 es el encargado de controlar a Arduino

Características físicas. La longitud máxima y la anchura de Arduino Uno son 2,7 y 2,1 pulgadas respectivamente. Los pines de entrada están separados de los analógicos, y en medio de los mismos está ubicado el micro controlador como se puede observar en la Figura 2.5. Tiene orificios para tornillos que permiten que la tarjeta sea sujeta a una superficie o caja. La distancia entre los pines digitales 7 y 8 es de 160 milésimas de pulgada.

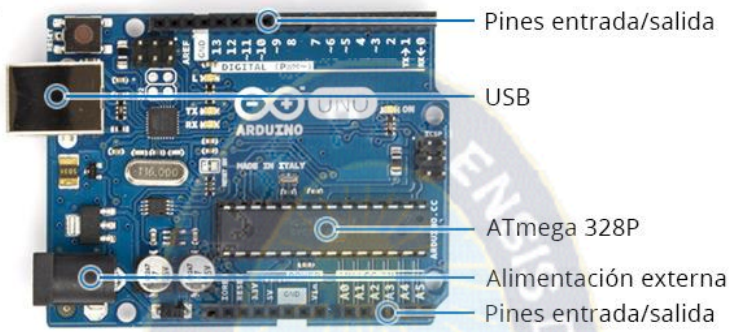


Figura 2.5: Arduino Uno características generales.
Tomado de Arduino(2015).

2.4. Entorno de desarrollo

Para programar la placa es necesario descargar de la página web de Arduino el entorno de desarrollo (IDE). Se dispone de versiones para Windows y para MAC, así como las fuentes para compilarlas en LINUX. La Figura 2.6 muestra el aspecto del entorno de programación. En el caso de disponer de una placa USB es necesario instalar los drivers. Estos drivers vienen incluidos en el paquete de Arduino mencionado anteriormente. Existen en la web versiones para distintos sistemas operativos.

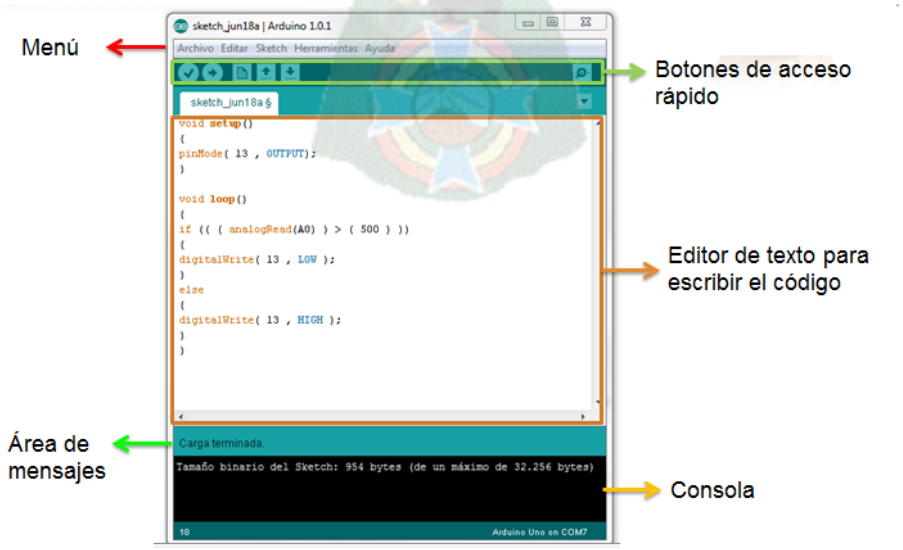


Figura 2.6: Entorno de Desarrollo de Arduino y las características de la ventana

Lo primero que se debe hacer para comenzar a trabajar con el entorno de desarrollo es configurar las comunicaciones entre la placa Arduino y el PC. Para lo cual se debe abrir en el menú "Tools" la opción "Serial Port". En esta opción se selecciona el puerto serie al que está conectada la placa. En Windows, si se desconoce el puerto al que está conectado la placa podemos descubrirlo a través del Administrador de dispositivos , Puertos COM & LPT y USB Serial Port.

Luego se debe comprobar que todo la configuración realizada hasta ahora es correcta y familiarizarse con el interfaz de desarrollo, se debe abrir uno de los ejemplos. Es recomendable abrir el ejemplo "Blink". Para ello debemos acceder a través del menú Archivo: Ejemplos: Basics : Blink.

El ejemplo "Blink" lo único que hace es parpadear un LED que está colocado en el pin número 13 de la placa. Primero comprobamos que el código fuente es el correcto. Para lo cual se pulsa el botón de verificación de código que es el primer botón al lado Izquierdo de la ventana, Figura 2.7 a). Si todo va bien deberá aparecer un mensaje en la parte inferior de la interfaz indicando "COMPILADO". Una vez que el código ha sido verificado se procede a cargarlo en la placa. Para ello se tiene que pulsar el botón de reset de la placa, Figura 2.7 b) e inmediatamente después pulsar el botón que tiene forma de flecha, el cual comienza la carga. Figura 2.7 c).

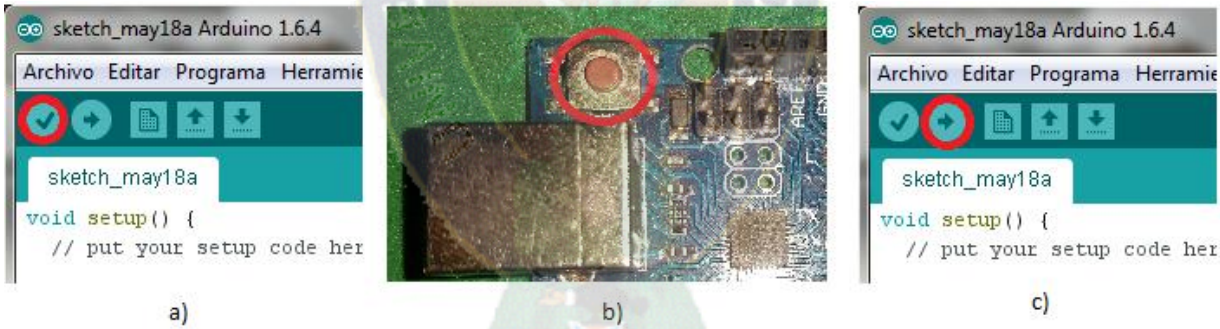


Figura 2.7: Pasos para cargar un programa a Arduino.
Verificamos que no exista errores (a), se debe resetear la placa (b), y por último se carga el código (c).

Durante la carga del programa, en la placa USB, se encenderán los LED que indican que se están enviando y recibiendo información por el puerto serie: TX/RX. Si todo se ha realizado correctamente debe aparecer el mensaje "SUBIDO". Por último se debe esperar unos 8 segundos aproximadamente para comprobar que todo es correcto. Si el Led colocado en el pin 13 de la placa se enciende y se apaga cada segundo entonces todo ha ido bien. Por fin tenemos todo listo para empezar a trabajar con Arduino

2.4.1. Estructura básica de un programa

La estructura básica de programación de Arduino como se ve en el Programa 2.1, es bastante simple y divide la ejecución en dos partes: setup y loop. Setup() constituye la preparación del programa y loop() es la ejecución. En la función Setup() se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar el pinMode (si un determinado pin digital es de entrada o salida) e inicializar la comunicación serie. La función loop() incluye el código a ser ejecutado continuamente (leyendo las entradas de la placa, salidas, etc.).

Programa 2.1: Estructura básica de Programación de Arduino

```
void setup() {
        inMode(pin, OUTPUT);        // Establece 'pin' como salida
}
void loop() {
        digitalWrite(pin, HIGH);    // Activa 'pin'
        delay(1000);                // Pausa un segundo
        digitalWrite(pin, LOW);     // Desactiva 'pin'
        delay(1000);
}
```

Como se observa en este bloque de código cada instrucción acaba con ; y los comentarios se indican con //. Al igual que en C se pueden introducir bloques de comentarios con /* ... */.

2.4.2. Tipos de datos

Arduino permite manejar los siguientes tipos de datos:

- **Byte:** Almacena un valor numérico de 8 bits. Tienen un rango de 0-255.
- **Int:** Almacena un valor entero de 16 bits con un rango de 32,767 a -32,768.
- **Long:** Valor entero almacenado en 32 bits con un rango de 2,147,483,647 a -2,147,483,648.
- **Float:** Tipo coma flotante almacenado en 32 bits con un rango de 3.4028235E+38 a -3.4028235E+38.
- **Arrays:** Se trata de una colección de valores que pueden ser accedidos con un número de índice (el primer valor del índice es 0). Ejemplos de utilización:

Operadores aritméticos

Al emplear variables, valores constantes o componentes de un array se puede realizar operaciones aritméticas y se puede utilizar el operador cast para conversión de tipos. Ej. int a = (int)3.5; Además pueden hacerse las siguientes asignaciones:

x ++. Lo mismo que x = x + 1.

x --. Lo mismo que x = x - 1, o reducir x en -1.

x += y. Lo mismo que x = x + y, o incrementar x en +y.

x -= y. Lo mismo que x = x - y .

x *= y. Lo mismo que x = x * y.

x /= y. Lo mismo que x = x / y.

Para su utilización en sentencias condicionales u otras funciones Arduino permite utilizar los siguientes operadores de comparación:

x == y. x es igual a y.

x != y. x no es igual a y.

x < y, x > y, x <= y, x >= y.

Y los siguientes operadores lógicos:

Y lógico: if (x > 0 && x < 5). Cierto si las dos expresiones lo son.

O lógico: if (x > 0 || y > 0). Cierto si alguna expresión lo es.

NO lógico: if (!x > 0). Cierto si la expresión es falsa.

El lenguaje de Arduino presenta las siguientes constantes predefinidas:

TRUE / FALSE.

HIGH/LOW. Estas constantes definen los niveles de los pines como HIGH o LOW y son empleados cuando se leen o escriben en las entradas o salidas digitales. HIGH se define como el nivel lógico 1 (ON) o 5 V. LOW es el nivel lógico 0, OFF, o 0 V.

INPUT/OUTPUT. Constantes empleadas con la función pinMode() para definir el tipo de un pin digital usado como **entrada** INPUT o salida OUTPUT. Ej. pinMode(13, OUTPUT).

Sentencias condicionales

El lenguaje de Arduino permite realizar sentencias condicionales if, if... else, for, while, do... while. Su utilización es similar a las funciones correspondientes en C.

2.4.3. Entradas y salidas digitales y analógicas

Función pinMode(pin, mode)

Función usada en la function setup() para configurar un pin dado para comportarse como INPUT o OUTPUT. Ej. pinMode(pin, OUTPUT); configura el pin número 'pin' como de salida. Los pines de Arduino funcionan por defecto como entradas, de forma que no necesitan declararse explícitamente como entradas empleando pinMode().

Función digitalRead(pin)

Lee el valor desde un pin digital específico. Devuelve un valor HIGH o LOW. El pin puede ser especificado con una variable o una constante (0-13). Ej. v = digitalRead(Pin);

Funcion digitalWrite(pin, value)

Introduce un nivel alto (HIGH) o bajo (LOW) en el pin digital especificado. De nuevo, el pin puede ser especificado con una variable o una constante 0-13. Ej. digitalWrite(pin, HIGH).

Función analogRead(pin)

Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.

Función analogWrite(pin, value)

Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM. Esta función está activa para los pines 3, 5, 6, 9, 10, 11. Ej analogWrite(pin, v); // escribe 'v' en el 'pin' analógico. Puede especificarse un valor de 0 - 255. Un valor 0 genera 0 V en el pin especificado y 255 genera 5 V.

Funciones de tiempo y matemáticas

- delay(ms). Realiza una pausa en el programa la cantidad de tiempo en milisegundos especificada en el parámetro (máximo 1000, mínimo 1).
- millis(). Devuelve la cantidad de milisegundos que lleva la placa Arduino ejecutando el programa actual como un valor long unsigned. Después de 9 horas el contador vuelve a 0.
- min(x,y). max(x,y). Devuelve el mínimo y el máximo respectivamente de entre sus parámetros.

Puerto serie

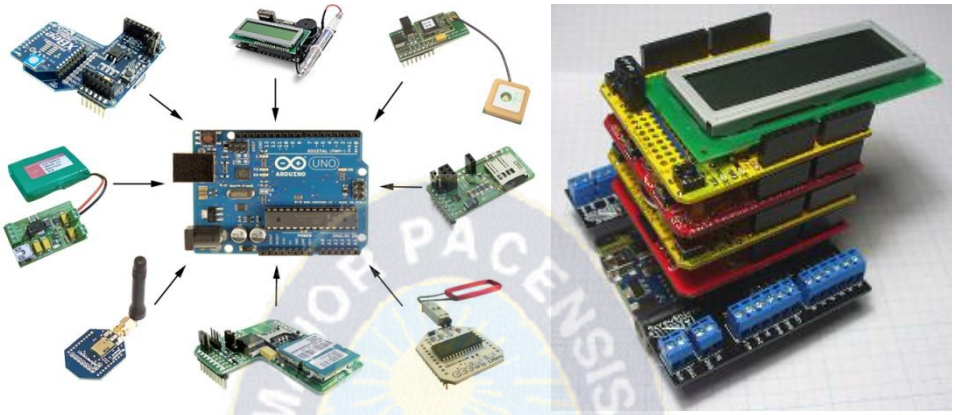
- Serial.begin(rate). Abre un Puerto serie y especifica la velocidad de transmisión. La velocidad típica para comunicación con el ordenador es de 9600 aunque se pueden soportar otras velocidades.
- Serial.read(). Lee o captura un byte (un carácter) desde el puerto serie. Devuelve -1 si no hay ningún carácter en el puerto serie.
- Serial.available(). Devuelve el número de caracteres disponibles para leer desde el puerto serie.
- Serial.println(data). Imprime datos al puerto serie seguido por un retorno de línea automático. Este comando tiene la misma forma que Serial.print() pero este último sin el salto de línea al final. Este comando puede emplearse para realizar la depuración de programas. Para ello puede se mandar mensajes de depuración y valores de variables por el puerto serie. Posteriormente, desde el entorno de programación de Arduino, activando el "Serial Monitor" se puede observar el contenido del puerto serie, y, por lo tanto, los mensajes de depuración. Para observar correctamente el contenido del puerto serie se debe tener en cuenta que el "Serial Monitor" y el puerto serie han de estar configurados a la misma velocidad (Para configurar la velocidad del puerto serie se hará con el comando Serial.begin(rate)).

2.5. Shield para Arduino

Si se desea expandir la funcionalidad que ofrece la plataforma Arduino, es posible adquirir y agregar nuevos Shield que son compatibles con Arduino. De tal manera se podrá dar a la placa más funciones para que pueda realizar tareas especificas.

Un Shield para Arduino es una tarjeta de expansión que provee funcionalidades adicionales a la tarjeta Arduino, se puede conectar a la parte superior de la placa Arduino para ampliar sus capacidades como se ve en la Figura 2.8 a) , permitiendo además ser apiladas unas encima de otras

manteniendo un diseño modular Figura 2.8 b). Hay infinidad de tarjetas en el mercado, con funciones tan variadas como comunicaciones por radio, manejo de Led, control de motores, etc. (GeekFactory, 2015)



a) b)

Figura 2.8: Ejemplo de diseño modular para Arduino.
Hay muhcos Shield diseñados algunos oficiales otros no y se pude conectar shields apilados en Arduino de tal manera de se vea como una torre
Tomado de Cooking-Hacks (2014) y Shield (2015).

Para realizar el prototipo se necesitara un Shield SIM908, el cual puede agregar una línea móvil GSM en Arduino y Shield GPS para poder tener la ubicación inmediata, en tiendas especializadas en Bolivia estos se encuentran solo en módulos separados y no así como Shield, pero se pudo encontrar un Shield que contiene a ambos y es el Shield GSM/GPRS/GPS SIM908 que se muestra en la Figura 2.9 y es perfectamente compatible con Arduino.

Figura 2.9: Módulo GPRS+GPS Quadband para Arduino y Raspberry PI (SIM 908).
Tomado de Cooking-Hacks (2015).

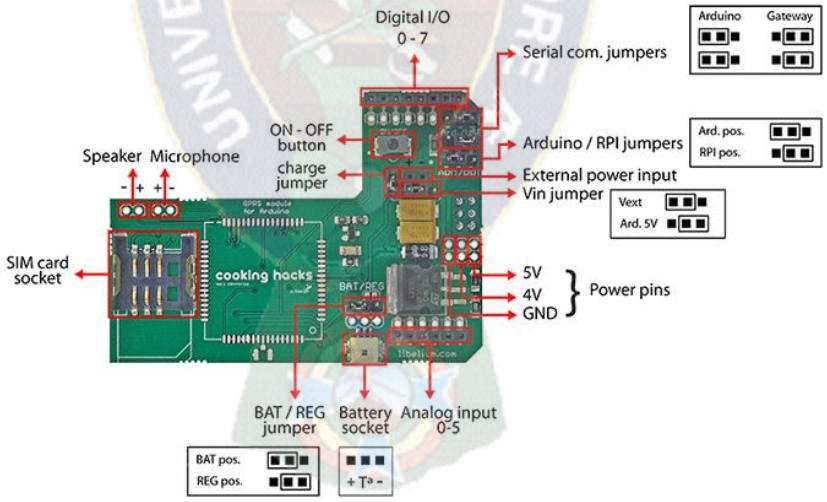
2.5.1. Shield GSM/GPRS/GPS SIM90 para Arduino

SIM908 módulo es un módulo completo de banda cuádruple GSM/GPRS, que combina la tecnología GPS para la navegación por satélite. El diseño compacto que integra GPRS y GPS en un paquete SMT[8] ahorrará significativamente el tiempo y los costes para los clientes para desarrollar aplicaciones basadas en GPS.

Con una función de interfaz y el GPS estándar de la industria, que permite que los activos variables para hacer un seguimiento completo en cualquier lugar y en cualquier momento con cobertura de la señal (simCom, 2016).

2.5.2. Características

Mediante este módulo de expansión compatible con la mayoría de las placas Arduino, no sólo se puede transmitir la ubicación del sistema a través de HTTP y almacenarla en un servidor web, sino que también se puede utilizar Google Maps para ver la localización sobre el mapa en todo momento. A continuación en la Figura 2.10 se puede observar con detalle cada uno de los pines, puertos y distintos elementos que conforman esta placa.



a)

[8] Dispositivo de montaje superficial

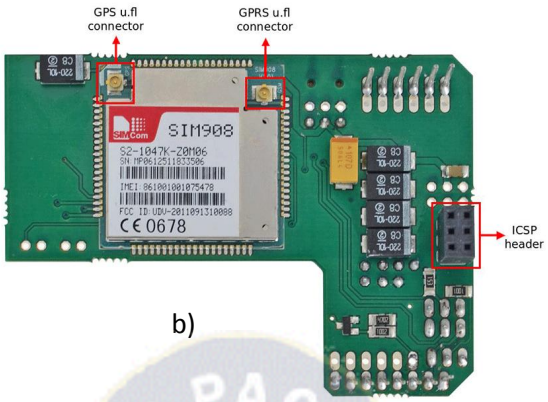


Figura 2.10: Diagrama de conexiones de Shield SIM908.
En la parte a) se observa la ubicación del Chip SIM de los Jumpers y Visto desde la parte inferior se ve el SIM908 y la conexión ICSP
Tomado de Cooking-Hacks(2015).

Características Generales

Quad-Band 850/900/18001900MHz, RS multi-slot clase 10, GPRS estación móvil clase B , Cumple con la norma GSM fase 2/2+ - Clase 4 (2 W @ 850/900 MHz) - Clase 1 (1 W @ 1800/1900MHz), Control vía comandos AT, kit de herramientas de aplicación SIM, rango de tensión de alimentación: -GPRS: 3.2 ~ 4.8 V -GPS: 3.0 ~ 4.5V, Bajo consumo de energía, Dimensiones: 30*30*3.2mm, Peso: - SIM908:5.2g - SIM908-C:11.1g, Temperatura de funcionamiento: -40 °C a 85 °C.

Especificaciones para envió de SMS vía GSM / GPRS

Punto a punto de MO y MT, difusión celular SMS, texto y el modo PDU Especificaciones para audio, Tricodec - Half rate (HR) - Full rate (FR) - Enhanced Full Rate (EFR), Manos libres cancelación de eco.

Especificaciones para transferencia de datos

GPRS clase 8/10: max. 85.6 kbps (downlink), soporte PBCCH, esquemas de codificación CS 1, 2, 3, 4, CSD up to 14.4 kbps, USSD, integrado TCP/IP.

Especificaciones para GPS

Tipo de receptor - 42 canales - GPS L1 código C / A, - motor de STE de alto rendimiento, sensibilidad - Seguimiento: -160 dBm - Los arranques en frío: -143 dBm Los arranques en frío: 30s (típico .) - Hot comienza: 1s (típico).

Interfaces

La interfaz externa de SIM 3V / 1.8V, interfaces de audio analógicas duales, interfaz SPI, copia de seguridad de RTC, interfaz de carga, una interfaz en serie y una interfaz de depuración para GSM / GPRS, interfaz de depuración para la salida de información del GPS NMEA, dos conectores de antena independientes para GSM / GPRS y GPS COMANDOS AT.

2.6. Comandos AT

Los comandos AT son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem.

En un principio, el juego de comandos AT fue desarrollado en 1977 por Dennis Hayes como un interfaz de comunicación con un modem para así poder configurarlo y proporcionarle instrucciones, tales como marcar un número de teléfono. Más adelante, con el avance del baudio, fueron las compañías Microcomm y US Robotics las que siguieron desarrollando y expandiendo el juego de comandos hasta universalizarlo (Lammert, 2016).

Los comandos AT se denominan así por la abreviatura de attention. Aunque la finalidad principal de los comandos AT es la comunicación con módems, la telefonía móvil GSM también ha adoptado como estándar este lenguaje para poder comunicarse con sus terminales. De esta forma, todos los teléfonos móviles GSM poseen un juego de comandos AT específico que sirve de interfaz para configurar y proporcionar instrucciones a los terminales. Este juego de instrucciones puede encontrarse en la documentación técnica de los terminales GSM y permite acciones tales como realizar llamadas de datos o de voz, leer y escribir en la agenda de contactos y enviar mensajes SMS, además de muchas otras opciones de configuración del terminal.

Queda claro que la implementación de los comandos AT corre a cuenta del dispositivo GSM y no depende del canal de comunicación a través del cual estos comandos sean enviados, ya sea cable de serie, canal Infrarrojos, Bluetooth, etc. (BlueHacks, 2016).

Existe muchos comandos AT, a continuación se describirá algunos comandos que son importantes a la hora de realizar la conexión:

- AT – Con este comando se verifica que el módulo está recibiendo nuestras instrucciones. Si todo es correcto, tras enviarlo debe responder con un “OK”. En caso contrario no se obtendrá respuesta alguna.
- AT+CPIN? – Se utilizara esta instrucción para comprobar si la tarjeta SIM está desbloqueada o si por el contrario, requiere introducir el código PIN para proceder con el desbloqueo de la misma. Cuando la SIM esté operativa responderá con un “ready”.
- AT+CPIN=”****” – En el caso de que se necesite introducir el PIN, éste es el comando que se debe enviar, se escribe los 4 dígitos correspondientes al código de desbloqueo en el lugar de los asteriscos, delimitado entre comillas.
- AT+CREG? – Con este comando se pregunta por el estado de conexión a la red. La respuesta recibida seguirá la siguiente notación: +CREG: <n>,<stat>, donde:

  <stat> = 0 : No registrado, no está buscando una conexión de red
  <stat> = 1 : Registrado a la red nacional
  <stat> = 2 : No registrado, pero buscando la red
  <stat> = 3 : Registro denegado

- AT+COPS? – Mediante esta instrucción se recibe la confirmación de la compañía en la que está registrada nuestra tarjeta SIM.
- AT+CMGF=<f> – Selecciona el formato del mensaje SMS, donde:

  <f> = 0 : modo PDU
  <f> = 1 : modo texto

- AT+CSCS=”IRA” – Con este comando se selecciona el set de caracteres que es utilizado para el envío de SMS. En nuestro caso nos interesa configurar el módulo en modo “IRA” (International Reference Alphabet).

- AT+CMGS=”num_movil” – A través de este comando se marca el número de móvil al que se desea hacer llegar el SMS. Irá delimitado por comillas.
- AT+CMGR=<r> – Es el comando utilizado para leer SMS almacenados en la memoria de la tarjeta SIM. En lugar de <r> se pondrá el número correspondiente a la posición del mensaje que se quiere leer. Por ejemplo, si se desea leer el último mensaje recibido, se debe poner un ‘1’, si se desea leer el penúltimo mensaje se debe poner un ‘2’, y así sucesivamente.

- AT+CPMS=”SM” – Con esta instrucción se puede seleccionar el directorio del que se desea leer un mensaje de texto. En función de las siglas que se introduce tras el igual, se puede recopilar la información de distintas memorias. En este caso, con las siglas “SM”, se selecciona como directorio la memoria de la tarjeta SIM.
- AT+CMGD=<n> – Este comando sirve para eliminar SMS de la memoria de la tarjeta SIM. Se cambiara <n> por la posición en la memoria que ocupe el SMS que se desee borrar. Por ejemplo, para borrar el primer mensaje almacenado en la memoria de la SIM la instrucción sería AT+CMGD=1.
- AT+CMGL=”ALL” – Con este comando se puede ver en una lista todos los SMS que tengamos en la SIM, tanto leídos, como no leídos.
- +CMTI: “SM”, <pos> – Esta es la instrucción que se recibe automáticamente por parte del módulo cuando se reciba un SMS, donde <pos> es el número correspondiente a la posición en memoria en la que se ha almacenado dicho mensaje.

Por ejemplo, si se tiene la memoria de la SIM vacía y llega un SMS, la instrucción que nos enviaría el módulo sería +CMTI: “SM”, 1.

Se debe tener en cuenta que para el envío de cualquiera de los comandos AT a través de un Hyperterminal o del monitor serie de Arduino, se tiene que seleccionar siempre los caracteres fin de línea y retorno de carro.

Si uno desea indagar mas sobre estos comandos esta el manual oficial de SIM908 que contiene todos los comandos disponibles (Cooking-Hacks, 2014).

2.7. Redes móviles GSM

La red GSM[9] es el estándar más usado para la comunicación de teléfonos móviles ó portátiles. Se denomina estándar "de segunda generación" (2G) porque, a diferencia de la primera generación de teléfonos portátiles, las comunicaciones se producen de un modo completamente digital. Posteriormente fueron introducidos respectivamente los estándares UMTS[10] y LTE [11], con capacidades adicionales de conexión a internet y mayor velocidad de transferencia de datos (PuntoFlotante, 2015).

[9] Global System for Mobile communications
[10] Universal Mobile Telecommunications Systems, 3G
[11] Long Term Evolution, 4G

Básicamente existen 4 bandas, que son estándares a nivel mundial: 850, 900, 1800 y 1900 Mhz. En la mayoría de los países de Europa, Asia, Australia, Medio Oriente y Africa, se emplean las bandas de 900-1800 Mhz. En los Estados Unidos, Canadá, México y la mayor parte de Centro y Sudamérica se usan las bandas de 850-1900 Mhz. En México se emplea la banda de 1900 Mhz.

Los módems GSM cuatri-banda permiten operar en cualquiera de las 4 bandas mencionadas. Mediante los comandos AT, es posible configurar el módem GSM cuatri-banda para operar en la banda deseada. En la Figura 2.11 de abajo se muestra la asignación de pares de frecuencias GSM a nivel global.

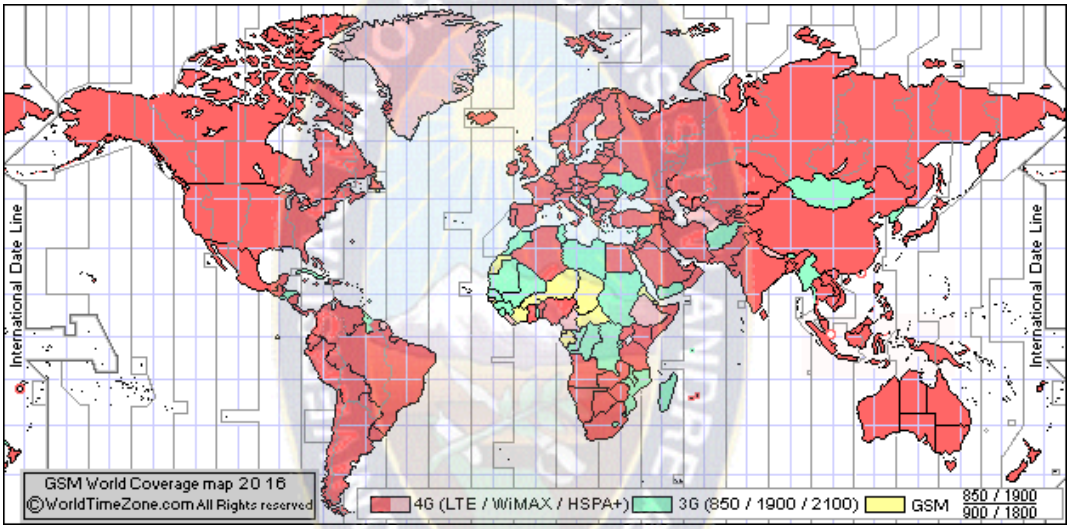


Figura 2.11: Mapa de líneas GSM asignadas a nivel mundial.
La tecnologia 4g en nuestro continente tiene un alcance casi general y a nivel global muy pocos paises solo cuentan con 3g y GSM
Tomado de WTZ (2016).

2.8. Sistema de posicionamiento global GPS

El sistema de Posicionamiento Global (GPS) es un sistema de radionavegación de, basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación, y cronometría gratuita e ininterrumpidamente a usuarios civiles en todo el mundo. A todo el que cuente con un receptor del GPS, el sistema le proporcionará su localización y la hora exacta en cualesquiera condiciones atmosféricas, de día o de noche, en cualquier lugar del mundo y sin límite al número de usuarios simultáneos.

El GPS se compone de tres elementos: los satélites en órbita alrededor de la Tierra, las estaciones terrestres de seguimiento y control, y los receptores del GPS propiedad de los usuarios. Desde el espacio, los satélites del GPS transmiten señales que reciben e identifican los receptores del

GPS; ellos, a su vez, proporcionan por separado sus coordenadas tridimensionales de latitud, longitud y altitud, así como la hora local precisa.

Hoy están al alcance de todos en el mercado los pequeños receptores del GPS portátiles. Con esos receptores, el usuario puede determinar con exactitud su ubicación y desplazarse fácilmente al lugar a donde

desea trasladarse, ya sea andando, conduciendo, volando o navegando. El GPS es indispensable en todos los sistemas de transporte del mundo ya que sirve de apoyo a la navegación aérea, terrestre y marítima. Los servicios de emergencia y socorro en casos de desastre dependen del GPS para la localización y coordinación horaria de misiones para salvar vidas. Actividades cotidianas como operaciones bancarias, de telefonía móvil e incluso de las redes de distribución eléctrica, ganan en eficiencia gracias a de la exactitud cronométrica que proporciona el GPS. Agricultores, topógrafos, geólogos e innumerables usuarios trabajan de forma más eficiente, segura, económica y precisa gracias a las señales accesibles y gratuitas del GPS (GPS, 2016).

El GPS depende en que cada satélite en la constelación transmita su posición exacta y una señal de tiempo extremadamente precisa a los recibidores en la tierra. Dada esta información, los receptores GPS pueden calcular su distancia al satélite, y combinando esta información de cuatro satélites, el recibidor puede calcular su posición exacta usando un proceso llamado trilateración que se podrá observar en la Figura 2.12.

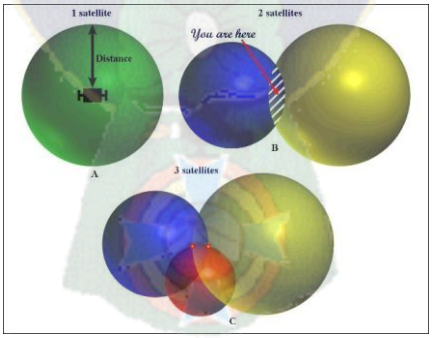


Figura 2.12: Ejemplo de TRILATERACION.
Calculando la posición de un objeto usando tres satélites
Tomado de Newman (2013).

La distancia a los satélites se calcula midiendo el tiempo que toma a la señal a llegar del satélite al recibidor, Distancia = velocidad X tiempo. Debido a que las señales de radio viajan a la velocidad de la luz (186,000 millas por segundo) los tiempos en tránsito de satélites a recibidores son extremadamente pequeños y se necesitan dispositivos de cronometraje extremadamente precisos para medirlos con exactitud, por lo cual surge la necesidad de llevar relojes atómicos en los satélites.

Sin embargo, los receptores no llevan relojes atómicos lo cual introduce errores en ese lado del sistema, y aún errores de cronometraje pequeños pueden resultar en grandes errores de posición. Aquí es donde entra en juego la cuarta medida.

Si las cuatro medidas son exactas, la esfera definida por la cuarta medida debe cruzar las otras tres en un punto que representa la posición actual. Si existen errores, la cuarta esfera no cruzará a todas las otras. Debido a que el error del recibidor es el mismo para las cuatro medidas, un ordenador en el recibidor puede calcular una corrección que haga que las cuatro esferas crucen, y aplicar la corrección a las medidas para obtener la posición correcta.

La cuarta medida también permite al sistema calcular una posición en tres dimensiones que incluye no solo latitud y longitud, sino también altitud. Las medidas de altitud, sin embargo, se reportan con referencia a un modelo matemático de la tierra para poder expresarlas en relación a un dato convencional, normalmente nivel del mar. El modelo es solo una aproximación por lo cual las medidas de altitud son menos precisas que las de latitud y longitud, y los errores son diferentes en diferentes partes del planeta (Rey, 2015).

2.9. Sistema operativo Android

Android es el sistema operativo con el crecimiento más rápido durante los últimos años originalmente creado para dispositivos móviles, tablets, como se ve en la Figura 2.13. También puede ser usado en equipos de cómputo aunque evidentemente fue desarrollado específicamente para dispositivos móviles por lo que muchas funciones no pueden ser utilizadas debido al hardware; más que nada es de demostración. No es un lenguaje de programación; es un entorno de software donde se puede correr ciertas aplicaciones. La principal característica de este sistema operativo es que está basado en Linux (versión 2.6 del Kernel), es libre, gratuito, multiplataforma, ofrece agilidad y portabilidad para aprovechar las numerosas opciones de hardware de los futuros teléfonos equipados con Android (Dimas, 2014).

Figura 2.13: Sistema Operativo Android.
Android esta instalado en una variedad de Smartphones y hay infinidad de aplicaciones para la misma y cada dia van en aumento.
Tomado de Android (2015).

Asimismo otras de las grandes ventajas de este sistema es que, gracias a tener el código libre, a su entorno se ha formado una gran comunidad de desarrolladores, los cuales continuamente están creando programas y utilidades para este sistema operativo con una plataforma actualizable y altamente flexible. Actualmente el número de programas supera los cuatrocientos mil, de los cuales una buena mayoría son gratuitos siendo la razón de su fama.

Por su parte Google liberalizó el código del sistema operativo, lo cual significa que cualquier programador con conocimientos suficientes puede ser capaz de desarrollar nuevo software para Android. Ello supone una diferencia muy grande frente a sistemas operativos como por ejemplo Microsoft Windows, cuyo código es cerrado y limita de forma notable la evolución del propio sistema operativo.

El Sistema Operativo Android se presentó oficialmente el 21 de octubre de 2008 y actualmente es un sistema operativo en permanente desarrollo según Android (2015).

2.9.1. La plataforma de Android

Con la amplitud de las capacidades de Android, sería fácil confundirlo con un sistema operativo de computadora de escritorio. Android es un entorno en capas que usa de base el kernel Linux e incluye vastas funciones. El subsistema de la UI incluye:

- Windows
- Vistas
- Widgets para mostrar los elementos comunes como los recuadros para editar, las listas y las listas desplegables

Android incluye un navegador integrable basado en WebKit[12], el mismo motor de navegador de código abierto que alimenta el navegador Mobile Safari de iPhone.

Android cuenta con una abundante variedad de opciones de conectividad, que incluyen Wi-Fi, Bluetooth y datos inalámbricos sobre una conexión de celular (por ejemplo, GPRS, EDGE y 3G). Una técnica popular en las aplicaciones Android es unirse a Google Maps para mostrar una dirección directamente dentro de una aplicación. El soporte para los servicios basados en la ubicación (como por ejemplo el GPS) y los acelerómetros está disponible también en la pila de software de Android. También existe un soporte para cámara.

[12] Es un motor de navegación web de código libre y además, un framework de Mac OS X que se usó para construir aplicaciones como el mencionado Safari, Dashboard, Mail y otras.

Históricamente, las dos áreas en las que las aplicaciones móviles se han esforzado para mantenerse al día con sus homólogos de escritorio son las de gráficos/soporte físico y métodos de almacenamiento de datos. Android trata el desafío de los gráficos con soporte incorporado para gráficos en 2D y 3D, incluida la biblioteca OpenGL. La carga de almacenamiento de datos se facilita porque la plataforma Android incluye la conocida base de datos de código abierto SQLite (Jirones, 2012).

2.9.2. Arquitectura de Android.

El sistema operativo está formado básicamente de cinco capas lógicas que se interrelacionan para brindar las funcionalidades necesarias, a continuación se describen dichas capas:

- Núcleo del sistema: derivado del Kernel Linux en su versión 2.6.0, el cual controla el acceso a los servicios de seguridad, memoria, gestión de procesos, batería y red; entre otros.
- Librerías: Android provee una biblioteca de librerías (escritas en C/C++), expuestas a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Entre las librerías incluidas, se encuentra la implementación de la librería estándar de C, bibliotecas de gráficos (OpenGL) y multimedia, navegador web (Webkit), base de datos local (SQLite), cifrado de telecomunicaciones (SSL), fuentes de texto (FreeType) y otros.
- Entorno de ejecución: en el mismo nivel de la capa anterior, consiste de bibliotecas de funcionalidades presentes en las librerías base del lenguaje Java. Cada aplicación se ejecuta en su propio proceso por Dalvik, la máquina virtual de Java para Android.
- Marco de aplicaciones base: comprende todas las clases Java y servicios que son usados directamente por los programas para cumplir con sus funciones. Estas aplicaciones son de código libre, por lo que pueden ser modificadas y adaptadas de acuerdo a las necesidades del desarrollador y del dispositivo. Se incluyen los administradores de actividades, ventanas, paquetes, telefonía, recursos, ubicaciones y notificaciones; proveedores de contenidos, vista del sistema y otros.
- Aplicaciones: es la capa más superficial y con la que el usuario comúnmente interactúa; todo Android incluye aplicaciones para correo electrónico, mensajes de texto, calendario, mapas, contactos; escritos tanto en Java (interpretados en Dalvik) como en C/C++ (ejecutados nativamente).

En la Figura 2.14 se presenta un esquema de la arquitectura de las capas del sistema operativo.

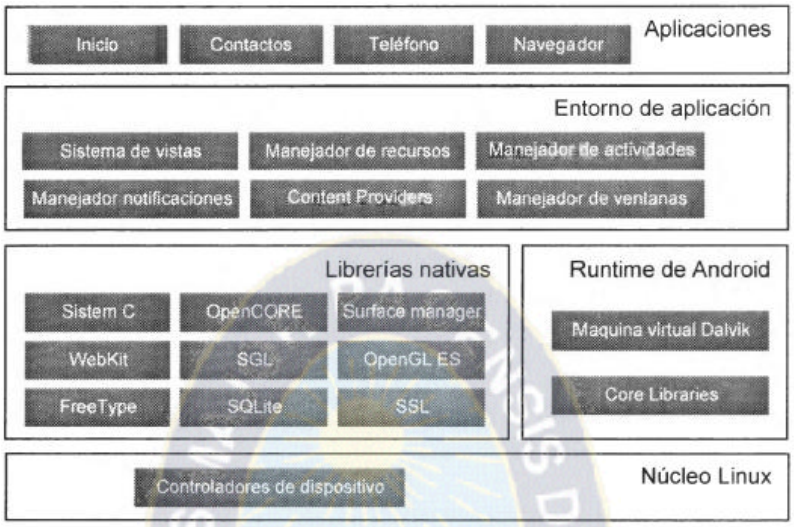


Figura 2.14: Esquema de la Arquitectura de Android.
Donde se puede observar los niveles y jerarquias de de cada capa que compone a Android
Tomado de Jirones (2012).

2.9.3. Patrón de arquitectura

En Android se usa el patrón de arquitectura llamado **Modelo Vista Controlador (MVC)** cuya principal bondad consiste en separar los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes como se ve en la Figura 2.15, los cuales se relacionarán para al final tener como resultado la aplicación.

Figura 2.15: Modelo vista controlador (MVC) de Android.
Los tres niveles de MVC uno que representa a la interfaz gráfica (*Vista*), otro que representa el tratamiento de datos (*Modelo*) y otro que se encarga de toda la lógica que se tiene que llevar a cabo por la aplicación (*Controlador*).
Tomado de Androideity (2016).

De esta forma se puede seccionar de forma más fácil el equipo de trabajo y dedicarse a desarrollar los componentes de tal forma que se construya módulos o librerías con funcionalidades específicas que incluso se podría reutilizar en proyectos posteriores y no simplemente en el proyecto actual.

- **Modelo.** El modelo se refiere a las representaciones que se construirá basadas en la información con la que operará la aplicación. En Java, el modelo viene siendo análogo a los beans que tienen la particularidad de ser reutilizables y ayuda a cumplir con el proverbio de oro **"Don´t Repeat Yourself" (DRY)** haciendo a la aplicaciones escalables. En esta parte del modelo también juega la decisión de qué modelo para almacenar información utilizaré. ¿Base de datos? ¿Web services? El modelo que se elija depende obviamente de las necesidades de información de tu aplicación.
- **Vista.** La vista es la interfaz con la que va a interactuar el usuario. En Android, las interfaces son construidas en XML. Se suele utilizar mucho la analogía de que esta parte es realmente parecida a lo que hacemos en el desarrollo web con los CSS. Se construye el esqueleto en XML que equivale al HTML de un sitio. Posteriormente, con ayuda de estilos, que también se escriben en XML, se puede empezar a darle formato de colores, posiciones, formato, etc. al esqueleto. Esto equivale a los CSS.
- **Controlador**. Finalmente el controlador que son todas esas clases que ayudara a darle vida a esas interfaces que ya fueron construidas y permite desplegar y consumir información de y para el usuario. Estos controladores se programan en lenguaje Java .

  Para empezar a aplicar esta arquitectura debes saber también su flujo:

  1. Todo parte cuando el usuario interactúa con la aplicación, el jugador actual es la vista.
  2. El controlador recibe la notificación de la acción solicitada.
  3. El modelo es llamado para ser modificado.
  4. El controlador nuevamente toma partida para llamar a la vista.
  5. Listo! El usuario ya tiene la nueva interfaz para seguir interactuando con la aplicación y volver a iniciar el ciclo cuando solicite otra acción.

2.9.4. Estructura de una aplicación Android

Tomando como base un ejemplo se explica la estructura de un proyecto (Dimas, 2014) y todos los archivos que tiene, como se ve en la Figura 2.16.

src: Aquí van las clases de nuestra aplicación, es decir los archivos .java.

gen: Son archivos que genera Java y por ninguna razón los debemos tocar. Si se hace, ya no van a servir y puede que ni el proyecto sirva para más adelante. Cada vez que se compila, Java se encarga de actualizarlo y de generarlo de nuevo. Dentro de gen se encuentra 2 archivos: el BuildConfig y R. El archivo R es el archivo que tiene los identificadores de todo lo que tiene la aplicación, por ejemplo imágenes, campos de texto, botones, etc. Java le asigna un identificador y no se tiene que preocuparnos por él, ya que se coloca un nombre común fácil de recordar y Java sabe cómo se llama.

assets: Este directorio contiene recursos de ayuda para la aplicación, audio, videos, bases de datos, la carpeta "assets" y la carpeta "res" sirven ambas para guardar recursos, pero la diferencia es que los que se encuentran en "assets" no generan un identificar en el archivo R que vimos se encuentra en el directorio "gen".

bin: Aquí tenemos archivos generados por el mismo Java, que en realidad no los utilizamos y tampoco debemos manipular, son archivos binarios como bien dice su nombre.

libs: Se encuentran librerías externas que necesita el proyecto.

res: El directorio "res" contiene todos los recursos de la aplicación.

res/drawable: Contiene todas las imágenes y gráficos PNG que se va a incluir en la aplicación. Cada uno representa una densidad.

res/layout: En este directorio se coloca todos los XML que son la parte gráfica de los "activities", es decir, todos los XML que son las pantallas la aplicación.

res/values: Se encuentran archivos con cadenas de texto que se usa en la aplicación, algunos estilos de la aplicación.

AndroidManifest.xml: El archivo Manifest es el más importante para la aplicación, es la columna vertebral del proyecto, en él se declara todas las actividades del proyecto, los permisos, versiones del SDK que usamos y un montón de cosas que vamos a ver más en detalle.

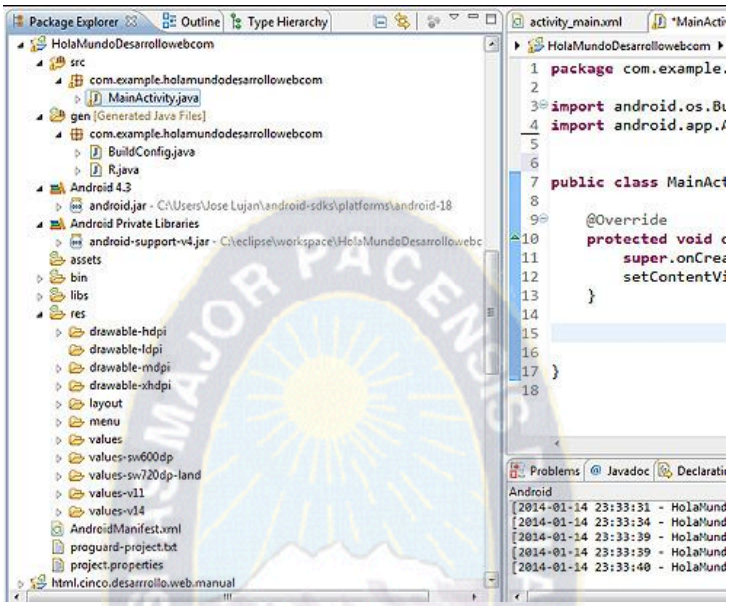


Figura 2.16: Archivos de una aplicación en Android.
Muestra claramente SRC, GEN, LIBRARIES, RES y el manifest de la aplicacion

2.9.5. Componentes en Android.

Se tiene seis componentes que se puede utilizar según Lujan (2014):

Activity:

Las actividades son las pantallas que se ve de un móvil, que está conformada por dos archivos, un archivo XML, parte gráfica y un archivo .Java, la parte lógica.

Services

Los servicios son tareas que se realizan en segundo plano y no contienen una interfaz de vista para el usuario.

Content Provider

El Proveedor de contenido sirve para proporcionar datos la aplicación, esto ayuda a compartir datos sin preocuparnos por el mecanismo de almacenamiento, lo que importa es la información que nos puede proporcionar para una o varias tareas que necesitan los datos en la aplicación.

Intent

Los intents son objetos que sirven para mandar mensajes. Además, se puede pedir ejecutar una acción a otro componente con el mismo intent. Dentro de los intents existen diferentes tipos dependiendo de lo que se necesite.

Widget

Los widgets son “pequeñas” aplicaciones que reciben actualizaciones periódicas, por ejemplo en algunos móviles se tiene widgets del clima en el móvil, sin necesidad de entrar a la aplicación se tiene la opción de colocar el widget en una la ventana principales en Android y lo mismo sucede con algunos reproductores de música, etc. La idea es que el usuario pueda hacer cosas sin entrar directamente en la aplicación.

Processes and Threads

Cuando una aplicación se inicia, lo que sucede de manera muy técnica, es que se crea un proceso. Entonces todo lo que se ejecuta dentro de esa aplicación es parte del mismo proceso de ejecución. Un hilo es otra forma de dividir el trabajo, un proceso puede tener varios hilos de ejecución, el hilo lo podemos tomar como una secuencia de control dentro de cualquier proceso y este ejecuta lo que indiquemos de manera independiente.

2.10. Sensor de movimiento

Un sensor de vibración es un dispositivo que reacciona ante movimientos bruscos, golpes, o vibraciones, pero no a movimientos constantes o progresivos. En el caso de detectar una vibración genera una señal digital, que cesa al finalizar la vibración.

El principio de funcionamiento es muy sencillo. El dispositivo dispone de un cilindro, con dos contactos. Uno de los contactos está unido a una varilla metálica ubicada en el centro del cilindro. A su alrededor, el otro contacto se arrolla a su alrededor en forma de muelle.

En caso de una vibración, el muelle se deforma por efecto de la inercia, estableciendo contacto en varios puntos con el contacto fijo. De esta forma, se establece una conexión eléctrica entra ambos contactos, que puede ser leída con un microprocesador, como si fuera un pulsador, como se vio en entradas digitales de Arduino (Llamas, 2015).

2.9.1. Sensor SW 420 NC

Es un sensor de vibración, normalmente cerrado, tiene tres pines que corresponden a Vcc, GND, Dout, como se muestra en la Figura 2.17, la salida se conecta directamente con el micro controlador para la detección de nivel alto y bajo, con el fin de detectar si en el medio ambiente existe vibración; es utilizado en Arduino, Smart, alarma Terremoto, alarma robo, ruptura de vidrio, apropiados para el diagnóstico de fallas en máquinas o estructuras (Ekstein, 2015).

El SW-420 NC posee las siguientes características:

- Salida del comparador, señal limpia, buena onda, fuerte capacidad de conducción.
- Amperaje > 15 mA
- Voltaje de funcionamiento 3.3V ~ 5V
- Formato de salida: salida de conmutación digital (0 y 1)
- El uso de un comparador de voltaje amplio LM393
- Con agujeros de los tornillos para una fácil instalación
- Tamaño pequeño: 3.2 x 1.4cm

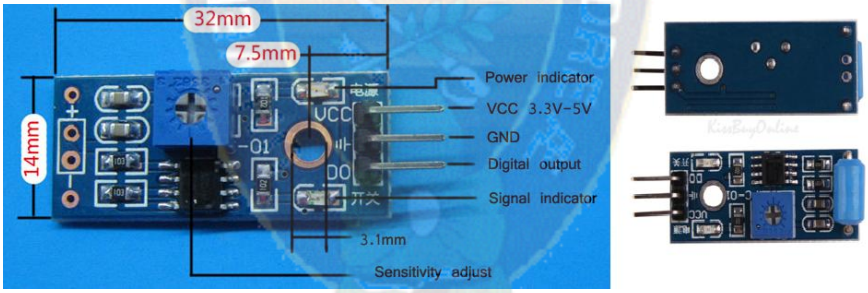


Figura 2.17: Sensor de Vibración SW-420 NC y su respectivos pines.
Tomado de HetPro (2015).

2.11. Sensor de gas mq-7

Es un sensor que es utilizado para detección de Monóxido de Carbono (CO), Figura 2.18, ideal para detectar concentraciones de CO en el aire. El MQ-7 puede detectar concentraciones en el rango de 20 a 2000ppm[13].

Este sensor tiene una alta sensibilidad y un tiempo de respuesta rápido. La salida del sensor es una resistencia análoga. El circuito de interfaz es muy simple, todo lo que se necesita hacer es

[13] Partes por millón, es una unidad de medida con la que se mide la concentración. Se refiere a la cantidad de unidades de una determinada sustancia que hay por cada millón de unidades del conjunto.

alimentarlo con 5V, añadir una resistencia de carga y conectar la salida al conversor análogo – digital (Sparkfun, 2015).

Características

- Fuente de poder 5Volts
- Tipo de interface Analógica
- Pin Out: 1- Salida, 2-GND, 3-VCC
- Respuesta rápida
- Larga vida del sensor
- Resistencia : 33Ω ±5%
- Tamaño: 35x22 mm

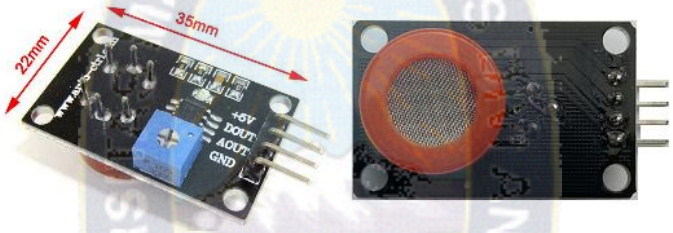


Figura 2.18: Sensor de Gas MQ-7.
Tomado de HetPro (2015).

2.12. Relé

El relé o relevador es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes, según Graf (1984).

2.12.1. Modulo Relé para Arduino

Para poder conectar Arduino al mundo exterior como por ejemplo para encender y apagar luces, controlar el riego del jardín, etc. No hay nada mejor que un interface o Placa de Relés , que aísla el circuito del Arduino con la alimentación de mando de los sistemas exteriores.

En la Figura 2.19 se muestra una tarjeta de relés que incluye 2 canales para ser controlados en forma remota. Ideal para controlar dispositivos en el hogar o en la industria. Cada canal es controlado por una entrada, la cual puede ser fácilmente controlada por un micro controlador o Arduino. Esta placa requiere de una alimentación de 12V (Microtronic, 2015).

Figura 2.19: Modulo Relé de dos canales para Arduino.
Tomado de Microtic (2015).

Características

- 2 Relés (Relays) de 1 polo 2 tiros.
- El voltaje de la bobina del relé es de 5 VDC.
- Led indicador para cada canal (enciende cuando la bobina del relé esta activa).
- Activado mediante corriente: el circuito de control debe proveer una corriente de 15 a 20 mA.
- Terminales de conexión de tornillo.
- Terminales de entrada de señal lógica con cabeceras macho.
- Medidas 8 x 8 x 5 cm.

2.13. Funcionamiento del motor de un automóvil

La parte más importante del motor de un automóvil la forman los cilindros y los pistones como muestra la Figura 2.20. El objetivo del motor es hacer mover el pistón arriba y abajo, para luego transmitir ese movimiento y hacer girar las ruedas. Lo que se hace es quemar gasolina en el interior del cilindro de manera que los gases en su expansión muevan el pistó. Los motores tienen un ciclo de funcionamiento se puede dividir en 4 etapas.

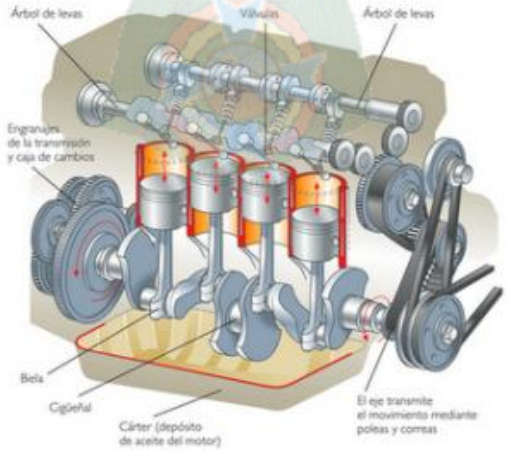


Figura 2.20: Partes importantes de un Motor.
Tomado de Chilton(2000).

La primera es **admisión**, en la que se introduce aire fresco en el pistón, el cilindro posee en la parte superior dos orificios (válvulas), uno para introducir el aire y otro para sacarlo. En realidad los motores tiene más de 2, pero lo simplificaremos para su explicación. En los motores de gasolina lo que se introduce es una mezcla de aire y gasolina. Al comienzo de la primera etapa el pistón está arriba del todo. Durante esta etapa, se abre la válvula de entrada de aire y el pistón baja para llenarse de aire fresco (sin gases de combustión).

En la segunda etapa, la de compresión, el pistón sube desde abajo del todo hasta arriba. Para que no se escape el aire que hemos metido antes, la válvula se cierra aproximadamente al llegar el pistón abajo. Durante esta etapa, la mezcla de aire y gasolina se comprime al subir el pistón y reducir el volumen.

Al final de la anterior etapa, el pistón estaba arriba y el aire estaba en su punto máximo de compresión. En este momento, se prende una chispa con la bujía y la mezcla explota. Fruto de esa explosión, los gases se expanden, llevando el pistón hasta abajo del todo. En estos motores, durante la tercera etapa, inyectan gasolina en gotitas muy pequeñas y a alta presión. Esta etapa se conoce como expansión o explosión.

Finalmente, el pistón estaba abajo del todo y el cilindro contenía gases de combustión que ya no sirven para nada. En esta última etapa, la de escape, se abre la válvula de salida de gases, y el pistón sube hasta arriba, vaciando el cilindro de aires residuales. Una vez arriba del todo, comenzaría otra vez la primera etapa de un nuevo ciclo, llenándose nuevamente el cilindro de aire fresco. El funcionamiento completo se puede observar en la Figura 2.21.

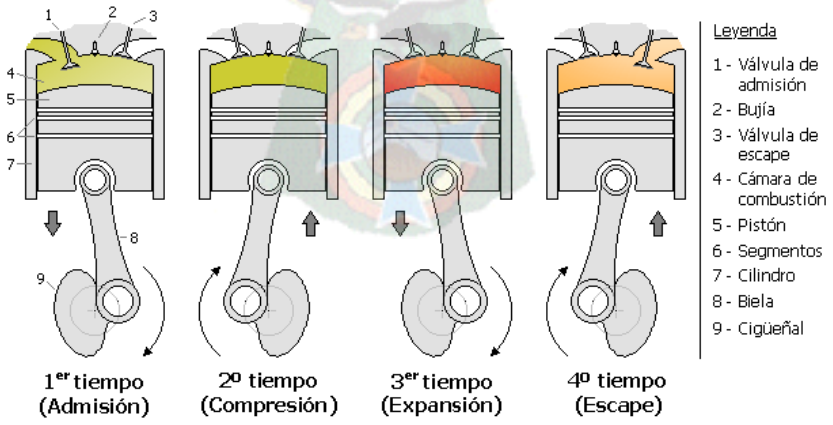


Figura 2.21: Fases del funcionamiento de un Motor. Tomado de Tomado de Chilton(2000).

2.14. Metodologías para el desarrollo de aplicaciones móviles

Una metodología según Avison & Fitzgeral (2006), es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental…). Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

Una definición más comprensible de metodología puede ser el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea.

Se puede destacar que una metodología:

- Optimiza el proceso y el producto software.
- Métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Una metodología define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología se pueden destacar:

Fases: tareas a realizar en cada fase.
Productos: E/S de cada fase, documentos.
Procedimientos y herramientas: apoyo a la realización de cada tarea.
Criterios de evaluación: del proceso y del producto. Saber si se han logrado los objetivos.

Según Vliet (2007), una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo han evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías

disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo.

El marco de trabajo de una metodología de desarrollo de software consiste en:

- Una filosofía de desarrollo de software, con el enfoque o enfoques del proceso de desarrollo de software.
- Múltiples herramientas, modelos y métodos para ayudar en el proceso de desarrollo de software.

Estos marcos de trabajo están con frecuencia vinculados a algunos tipos de organizaciones, que se encargan del desarrollo, soporte de uso y promoción de la metodología. La metodología con frecuencia se documenta de alguna manera formal (Vliet, 2007).

Es debido hacer notar que todas la metodologías existentes tienen pros y contras, por lo que es necesario conocerlas y ver cual brinda más ayuda para el desarrollo de la aplicación móvil.

2.12.1. Metodología RUP

El Proceso Unificado de Rational es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles.

El RUP es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software. Además Rational proporciona herramientas para todos los pasos del desarrollo así como documentación en línea para sus clientes.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. También se conoce por este nombre al software, también desarrollado por Rational, que incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo con las necesidades (Krutchen, 2001).

Además de estas características principales según Gonzales (2001) son las siguientes:

- **Desarrollo basado en componentes:** La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o que se desarrollan y maduran sus componentes.
- **Utilización de un único lenguaje de modelado:** UML es adoptado como único lenguaje de modelado para el desarrollo de todos los modelos.
- **Proceso Integrado:** Se establece una estructura que abarque los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración, el proceso unificado establece una estructura que integra todas estas facetas. Además esta estructura cubre a los vendedores y desarrolladores de herramientas para soportar la automatización del proceso, soportar flujos individuales de trabajo, para construir los diferentes modelos e integrar el trabajo a través del ciclo de vida y a través de todos los modelos.

La estructura estática del proceso unificado según IBM rational (2001), se define en base a cuatro elementos, que son: los roles(antes workers), que responde a la pregunta ¿quién?, las actividades (activities), que responden a la pregunta ¿cómo?, los productos (artifacts), que responden a la pregunta ¿qué?, y los flujos de trabajo (workflows), que responden a la pregunta ¿cuándo?..

2.12.1.1. Ciclo de vida y fases de RUP

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

En la Figura 2.22 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

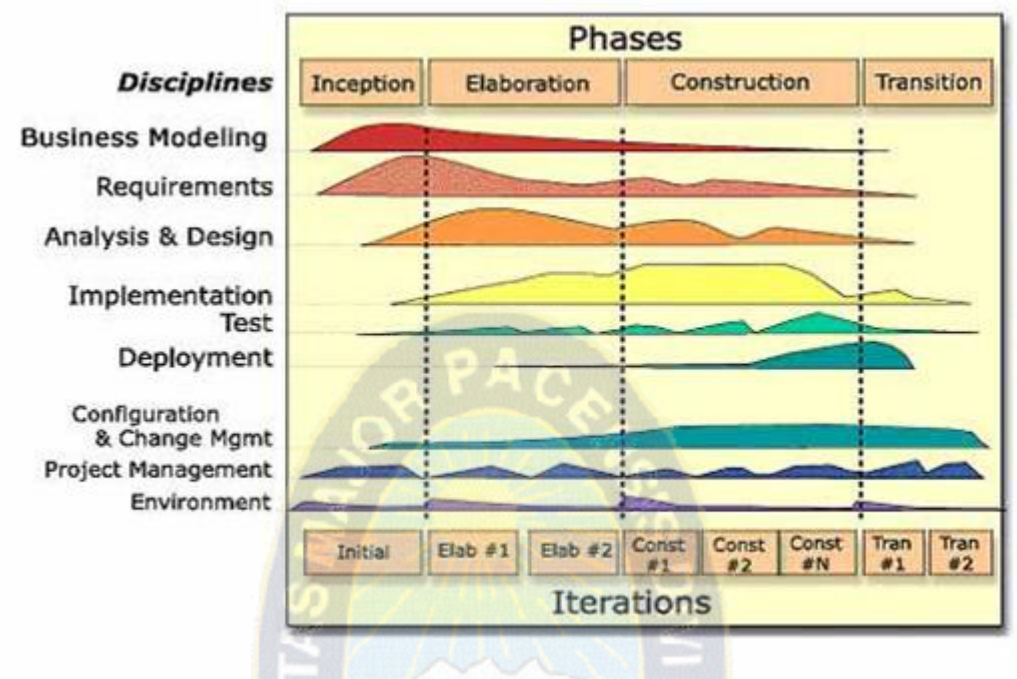


Figura 2.22: Esfuerzo en actividades según fase del proyecto.
Tomado de Krutchen (2001).

RUP se divide en cuatro fases, las cuales pasaremos a ver con más detalle. En la Tabla 2.2 encontramos un resumen de los principales productos de RUP y en qué momento deben iniciarse y terminarse.

Tabla 2.2: Principales productos en RUP. I = inicio, R = refinamiento.
Tomado de Plantillas de productos de RUP (2016)

Flujo	Productos	Inicio	Elaboración	Construcción	Transición
Administración del proyecto	Plan de desarrollo	I	R	R	R
	Caso de negocio	I			
	Lista de riesgos	I	R	R	R
Requisitos	Modelos de casos de uso	I	R		
	Visión	I	R		
	Especificación adicional	I	R		
	Glosario	I	R		
Análisis y Diseño	Modelo de diseño		I	R	
	Documentación de la arquitectura SW		I		
Implementación	Modelo de implementación		I	R	R
Test	Plan Test		I	R	
Despliegue	Plan de despliegue		I		I

- **Inicio**

La fase de inicio trata de responder a estas preguntas : ¿Cuál es el objetivo? ¿Es factible? ¿Lo construimos o lo compramos? ¿Cuánto va a costar?, y a otras más. Sin embargo no se pretende una estimación precisa o la captura de todos los requisitos. Al contrario se trata de explorar el problema lo justo para decidir si va a continuar o no. Generalmente no debe durar mucho más de una semana.

Los objetivos según Krutchen (2001) son :

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los productos de la fase de inicio deben ser:

- Visión del negocio: Describe los objetivos y restricciones a alto nivel.
- Modelo de casos de uso.

Especificación adicional: requisitos no funcionales.

- Glosario: Terminología clave del dominio.
- Lista de riesgos y planes de contingencia.
- El caso de negocio (business case). Para más detalles ver el flujo de modelado del negocio.
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.
- Plan de iteración para la primera iteración de la fase de elaboración.
- Plan de fases.

No todos los productos son obligatorios, ni deben completarse al 100%, hay que tener en cuenta el objetivo de la fase de inicio. Los síntomas de que no se ha entendido la fase de inicio:

- Dura más de unas pocas semanas.
- Se intentan definir todos los requisitos.
- Se espera que las estimaciones o los planes sean muy precisos.

- Definir la arquitectura completamente, en lugar de refinarla en la fase de elaboración.
- No se definen el caso de negocio o la visión.
- Los nombres de la mayoría de los casos de uso o actores no se han definido.
- Todos los casos de uso se escriben con detalle.

Al terminar la fase de inicio se debe comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento los requisitos, evidenciado por la fidelidad de los casos de uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

- **Elaboración**

El propósito de la fase de elaboración según Krutchen (2001), es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. Cuando termina esta fase se llega al punto de no retorno del proyecto: a partir de ese momento se pasa de las relativamente ligeras y de poco riesgo dos primeras fases, a afrontar la fase de construcción, costosa y arriesgada. Es por esto que la fase de elaboración es de gran importancia. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los casos de uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves, bien con este prototipo, bien con otros de usar y tirar.

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.

- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes productos:

- Un modelo de casos de uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Posiblemente un manual de usuario preliminar.

La forma de aproximarse a esta fase debe ser tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todas los productos de la fase de inicio el glosario, el caso de negocio, el ROI (Return Of Invest), etcétera.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

- **Construcción**

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todas los componentes, características y requisitos, que no lo hayan sido hecho hasta ahora, han de ser implementados, integrados y testeados, obteniendo una versión del producto que se pueda poner en manos de los usuarios(una versión beta).

El énfasis en esta fase se pone controlar las operaciones realizadas, administrando los recursos eficientemente, de tal forma que se optimicen los costes, los calendarios y la calidad.

Los objetivos concretos según incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea practico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea practico.

Los productos de la fase de construcción deben ser :

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

- **Transición**

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la

documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y usabilidad del producto.

En concreto se citan algunas de las cosas que puede incluir esta fase:

- Testeo de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios.
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por si mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los productos de la fase de transición según Gonzales (2001) son:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida

Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Las actividades a realizar durante las iteraciones dependerán de su finalidad, si es corregir algún error detectado, normalmente será suficiente con llevar a cabo los flujos de trabajo de implementación y test, sin embargo, si se deben añadir nuevas características, la iteración será similar a la de una iteración de la fase de construcción.

La complejidad de esta fase depende totalmente de la naturaleza del proyecto, de su alcance y de la organización en la que deba implantarse.

Para aplicar el RUP en pequeños equipos y proyectos se deberá mapear los diferentes roles (a los que no hemos dado especial relevancia en este documento) entre los distintos miembros del equipo, pero la diferencia clave con un proyecto de mayor envergadura, es el grado de formalidad a la hora de usar los distintos artefactos, planes del proyecto, requisitos, clases.

El RUP es una metodología completa y extensa que intenta abarcar todo el mundo del desarrollo software, tanto para pequeños proyectos, como proyectos más ambiciosos de varios años de duración.

2.12.2. Metodología XP

Se centra en las mejores prácticas para el desarrollo de software. Según Beck & Zapata (2002) consta de doce prácticas: el juego de planificación, pequeñas emisiones, la metáfora, el diseño sencillo, las pruebas, la refactorización, la programación en parejas, la propiedad colectiva, integración continua, semana 40-h, los clientes en el lugar, y los estándares de codificación. La versión revisada "XP2" se compone de las siguientes prácticas "primarias": sentarse juntos, equipo, espacio de trabajo informativo, el trabajo de energía, programación en parejas, las historias, el ciclo semanal, el ciclo trimestral, flujos de trabajo, construcción de 10 minutos, integración continua, prueba de programación y diseño incremental. Esta versión extendida de XP fue introducida Beck y otros, sin embargo, muy pocas investigaciones se ha centrado en la nueva versión, estos podría ser simplemente debido al hecho de que cualquier método lleva tiempo para ganar fuerza y popularidad, y que sólo puede ser una cuestión de tiempo antes del uso de la versión revisada alcanza los mismos niveles que el uso de la original (Beck & Andres, 2004).

Extreme Programing es descrita por Beck como: una metodología ligera para pequeños y medianos equipos de desarrollo de software en la cara de los requerimientos imprecisos o rápidamente cambiante (Beck & Zapata, 2002), reconoce explícitamente que XP no es un conjunto de técnicas de desarrollo nuevos y revolucionarios. Es un conjunto de principios probados y fiables, bien establecidos como parte de la sabiduría convencional de la ingeniería de software, pero llevado a un extremo nivel de ahí el nombre "programación extrema".

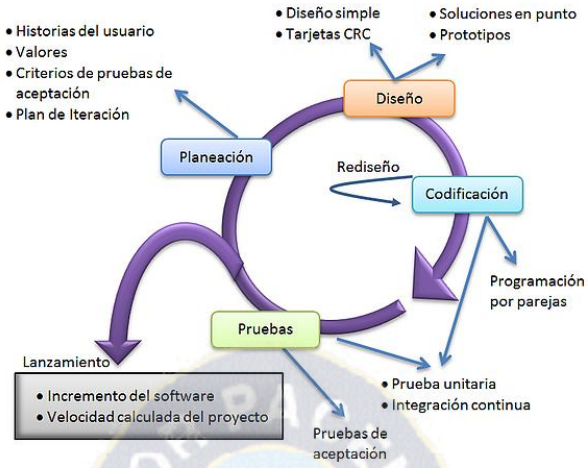


Figura 2.23: Descripción de la metodología XP.
Claramente se observa las tareas que debe realizarse en cada etapa de la metodologia
Tomado de Joskowicz (2008).

2.12.2.1. Fases de la Metodología XP

- **Planificación.**

XP plantea la planificación como un permanente dialogo entre las partes la empresarial (deseable) y la técnica (posible). Las personas del negocio necesitan determinar:

Ámbito: ¿ Qué es lo que el software debe de resolver para que este genere valor ?

Prioridad: ¿Qué debe ser hecho en primer lugar?

Composición de versiones: ¿Cuánto es necesario hacer para saber si el negocio va mejor con software que sin él?. En cuanto el software aporte algo al negocio debemos de tener lista las primeras versiones.

Fechas de versiones: ¿Cuáles son las fechas en la presencia del software o parte del mismo pudiese marcar la diferencia?

El personal del negocio no puede tomar en vació estas decisiones, y el personal técnico tomará las decisiones técnicas que proporcionan la materia prima para las decisiones del negocio.

Estimaciones: ¿Cuánto tiempo lleva implementar una característica?

Consecuencias: Informar sobre las consecuencias de la toma de decisiones por parte del negocio.

Procesos: ¿Cómo se organiza el trabajo y el equipo?

Programación detallada: Dentro de una versión ¿Qué problemas se resolverán primero?

- **Pequeñas versiones.**

Cada versión debe de ser tan pequeña como fuera posible, conteniendo los requisitos de negocios más importantes, las versiones tiene que tener sentido como un todo, no se puede implementar media característica y lanzar la versión.

Es mucho mejor planificar para 1 mes o 2 que para seis meses y un año, las compañías que entregan software muy voluminoso no son capaces de hacerlo con mucha frecuencia.

- **Diseño**

Metáfora.

Una metáfora es una historia que todo el mundo puede contar a cerca de cómo funciona el sistema. Algunas veces podremos encontrar metáforas sencillas “Programa de gestión de compras, ventas, con gestión de cartera y almacén”. Las metáforas ayudan a cualquier persona a entender el objeto del programa.

Diseño sencillo.

El diseño adecuado para el software es aquel que:

- Funciona con todas las pruebas.
- No tiene lógica duplicada.
- Manifiesta cada intención importante para los programadores
- Tiene el menor número de clases y métodos.

- **Desarrollo.**

Re codificación.

Cuando se implementa nuevas características en los programas se debe plantear la manera de hacerlo lo más simple posible, después de implementar cada característica, se debe preguntar cómo hacer el programa más simple sin perder funcionalidad, este proceso se le denomina re codificar o refactorizar (refactoring). Esto a veces puede llevar a hacer más trabajo del necesario, pero a la vez se prepara el sistema para que en un futuro acepte nuevos cambios y pueda albergar nuevas características.

Programación por parejas.

Todo el código de producción lo escriben dos personas frente al ordenador, con un sólo ratón y un sólo teclado. Cada miembro de la pareja juega su papel: uno codifica en el ordenador y piensa la

mejor manera de hacerlo, el otro piensa mas estratégicamente, ¿Va a funcionar?, ¿Puede haber pruebas donde no funcione?, ¿Hay forma de simplificar el sistema global para que el problema desaparezca?.

El emparejamiento es dinámico, puedo estar emparejado por la mañana con una persona y por la tarde con otra, si se tiene un trabajo sobre un área que no se conoce muy bien se puede emparejar con otra persona que si conozca ese área. Cualquier miembro del equipo se puede emparejar con cualquiera.

Propiedad colectiva.

Cualquiera que crea que puede aportar valor al código puede hacerlo, ningún miembro del equipo es propietario del código. Si alguien quiere hacer cambios en el código puede hacerlo. XP propone un propiedad colectiva sobre el código nadie conoce cada parte igual de bien pero todos conoce algo sobre cada parte, esto preparará para la sustitución no traumática de cada miembro del equipo.

Integración continúa.

El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. Una pareja de programadores se encargara de integrar todo el código en una maquina y realizar todas las pruebas hasta que estas funcionen al 100%.

40 Horas semanales.

Esto requiere que trabajemos 40 horas a la semana, mucha gente no puede estar más de 35 horas concentrados a la semana, otros pueden llegar hasta 45 pero ninguno puede llegar a 60 horas durante varias semanas y aun seguir fresco, creativo y confiado. Las horas extras son síntoma de serios problemas en el proyecto, la regla de XP dice nunca 2 semanas seguidas realizando horas extras.

Cliente In-situ.

Un cliente real debe sentarse con el equipo de programadores, estar disponible para responder a sus preguntas, resolver discusiones y fijar las prioridades. Lo difícil es que el cliente ceda una persona que conozca el negocio para que se integre en el equipo normalmente estos elementos son muy valiosos, pero se debe hacer ver que será mejor para su negocio tener un software pronto en funcionamiento, y esto no implica que el cliente no pueda realizar cualquier otro trabajo.

Estándares de codificación.

Si los programadores van a estar tocando partes distintas del sistema, intercambiando compañeros, haciendo refactoring, se debe de establecer un estándar de codificación aceptado e implantado por todo el equipo.

- **Pruebas**

Hacer pruebas.

No debe existir ninguna característica en el programa que no haya sido probada, los programadores escriben pruebas para chequear el correcto funcionamiento del programa, los clientes realizan pruebas funcionales. El resultado un programa más seguro que conforme pasa el tiempo es capaz de aceptar nuevos cambios.

2.12.3. Metodología Mobile-D

Es una metodología de desarrollo nueva, especialmente diseñada para el desarrollo de aplicaciones móviles, es propuesta por Pekka Abrahamsson y su equipo del VTT (Valtion Teknillinen Tutkimuskeskus, en inglés Technical Research Centre of Finland) en Finlandia que lideran una corriente muy importante de desarrollo ágil, está centrada en las plataformas móviles (Abrahamsson, 2007). Las prácticas asociadas a Mobile-D incluyen desarrollo basado en pruebas, la programación en parejas, integración continua y refactorización, así como las tareas de mejora de procesos de software; según Abrahamsson (2007) Mobile-D debe ser utilizado por un equipo de no más de diez desarrolladores, trabajando en conjunto para suministrar un producto listo en un plazo máximo de diez semanas.

Según Nossiere (2012), se trata de método basado en soluciones conocidas y consolidadas:

- Extreme Programming(XP), para las prácticas de desarrollo.
- Crystal Methodologies(Crystal) para escalar los métodos .
- Rational Unified Process (RUP), como base en el diseño del ciclo de vida.

2.12.3.1. Fases de desarrollo de Mobile-D

Mobile-D consta de cinco fases como se puede ver en la Figura 2.24: exploración, iniciación, producción, estabilización y prueba del sistema. Cada una de estas fases tiene un número de etapas, tareas y prácticas asociadas (AGILE, 2012).

En la primera fase, **Explorar**, el equipo de desarrollo debe generar un plan y establecer las características del proyecto. Esto se realiza en tres etapas: establecimiento actores(Stakeholder), definición del alcance y el establecimiento de proyectos. Las tareas asociadas a esta fase incluyen el establecimiento del cliente (los clientes que toman parte activa en el proceso de desarrollo), la planificación inicial del proyecto y los requisitos de recogida, y el establecimiento de procesos.

Figura 2.24: Fases de Desarrollo de Mobile-D.
Al igual que otras metodologias cada fase tiene un rol de tareas que se deben realizar
Tomado de Mobile-D (2016).

En la siguiente fase**, iniciación**, los desarrolladores preparan e identifican todos los recursos necesarios. Se preparan los planes para las siguientes fases y se establece el entorno técnico como los recursos físicos, tecnológicos y de comunicaciones, incluyendo el entrenamiento del equipo de desarrollo.. Esta fase se divide en cuatro etapas: la puesta en marcha del proyecto, la planificación inicial, el día de prueba y día de salida.

En la fase de **producción** se repite la programación de tres días (planificación, trabajo, liberación) se repite iterativamente hasta implementar todas las funcionalidades. Primero se planifica la iteración de trabajo en términos de requisitos y tareas a realizar. Se preparan las pruebas de la iteración de antemano. Las tareas se llevarán a cabo durante el día de trabajo, desarrollando e integrando el código con los repositorios existentes. Durante el último día se lleva a cabo la integración del sistema, en caso de que estuvieran trabajando varios equipos de forma independiente, seguida de las pruebas de aceptación.

En la fase de **estabilización**, se llevan a cabo las últimas acciones de integración para asegurar que el sistema completo funciona correctamente. Esta será la fase más importante en los proyecto multi-equipo con diferentes subsistemas desarrollados por equipos distintos. En esta fase,

los desarrolladores realizan tareas similares a las que debían desplegar en la fase de “producción”, aunque en este caso todo el esfuerzo se dirige a la integración del sistema. Adicionalmente se puede considerar en esta fase la producción de documentación.

La última fase **prueba y reparación del sistema** tiene como meta la disponibilidad de una versión estable y plenamente funcional del sistema. El producto terminado e integrado se prueba con los requisitos de cliente y se eliminan todos los defectos encontrados.

2.15. Descripción de las metodologías.

Es importante aclarar que no existe una metodología general para ser aplicada con éxito a cualquier tipo de proyecto de desarrollo de software de que se quiera realizar. Toda metodología debe ser adaptada a todo el contexto del proyecto como son los recursos, tiempo de desarrollo, tipo de sistema, etc. A continuación en la Tabla 2.3 una comparación de los modelos metodológicos.

Tabla 2.3: Tabla de comparación de Metodologías de desarrollo de Software

	RUP	XP	MOBILE D
BREVE DESCRIPCIÓN	Se caracteriza por ocupar el modelo iterativo e incremental. Esta centrado en la arquitectura	Modelo en el que se define un plan para desarrollar y liberar software. Además poder revisarlo para incorporar nuevas funcionalidades	Modelo ágil de desarrollo rápido, enfocado a grupos pequeños y que Buscan rápidas respuestas
TIPOS DE PROYECTO DE SOFTWARE	Grandes empresas	Aplicaciones móviles	Software para dispositivos móviles
PROGRAMADOR/ RELACIÓN CON EL USUARIO	Certificados en UML	Programación con habilidades blandas y trabajo en equipo	Interactúa con el cliente y tiene buenas relaciones con el grupo
ETAPAS	Inicio Elaboración Construcción Transición	Definir Roles Estimar el esfuerzo Elegir que construir Programar Repetir	Explotación Inicialización Producción Estabilización Testeo
CARACTERÍSTICAS PROPIAS DEL MODELO	Ocupa el modelo incremental y se centra en usar casos de Uso	Pone énfasis en la comunicación	Por cada función se realiza un ciclo de 3 días para planificar y trabajar en el proyecto para pensarlo

3. DISEÑO METODOLÓGICO

En este capítulo se pondrá en ejecución todos los conceptos, métodos, técnicas, que se tomaron en cuenta en el capítulo de Marco Teórico, desarrollando la metodología escogida para el desarrollo de la aplicación, además se presentara el modelo de control y seguimiento de un automóvil, describiendo la arquitectura del mismo.

Teniendo bien definida la arquitectura del controlador, se podrá proceder a describir la conexión de cada uno de los componentes necesarios para el diseño e implementación de del hardware de la aplicación, se debe tomar en cuenta que Arduino, aun no tiene una metodología de diseño, para lo cual en el presente trabajo, se toma el método experimental, se hará pruebas separadas de cada uno de los módulos que sean conectados a Arduino para luego unir todos los módulos. Por otro lado el desarrollo de software como se menciono al final del capítulo anterior tendrá base en la metodología Mobile-D.

3.1. Arquitectura del controlador.

En la aplicación de control y seguimiento de un automóvil, se tiene sensores y actuadores que van actuando para mandar información desde el automóvil, dicha información primero debe ser leída y procesada por la Arduino, luego será enviada al dispositivo móvil Android. Por lo cual la arquitectura que usara es de tipo centralizado como se ve en la Figura 3.1.

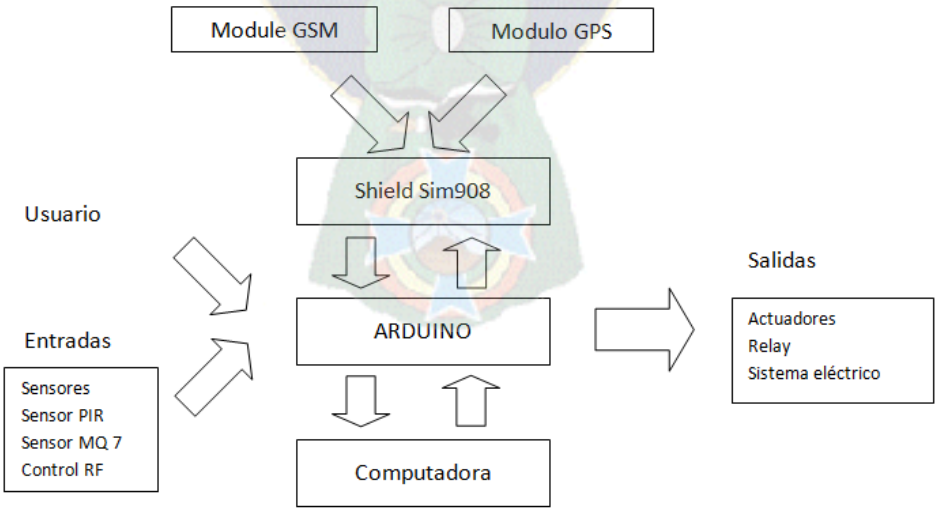


Figura 3.1: Arquitectura del Controlador.
El controlador tendra como base a arduino todos los datos seran procesados por el mismo para obtener salidas y mandar informacion correcta

3.2. Materiales del controlador

Los materiales necesarios para la implementación del prototipo del controlador son los que se detallan en la Tabla 3.1, donde cada ya fue descrito en el capitulo anterior.

Tabla 3.1: Componentes para la elaboración del hardware del prototipo

Nro.	Componente	Cantidad	Descripción
1	Computadora	1	Personal
2	Placa Arduino	1	Uno REV3
3	Shield GSM/GPS	1	Sim908
4	Antena exterior GPS	1	GPS-00464
5	Antena GSM	1	EXP-R24-006
6	Sensor de Presencia	1	PIR HC-SR501
7	Sensor de Gas	1	MQ 7
8	Modulo relé	1	EXP-T04-001
9	Control RF	1	YK 04
10	Protoboard	1	

A continuación se explicara paso a paso cada una de las etapas por las que se debe ir pasando hasta alcanzar el objetivo final del proyecto, incluyendo: pruebas realizadas, problemas que han ido surgiendo en el desarrollo de las mismas, códigos de programación correspondientes a cada una de las funciones que se han ido probando, y por supuesto, la descripción del sistema que finalmente ha sido diseñado, junto con las pruebas a las que ha sido sometido para evaluar su correcto funcionamiento.

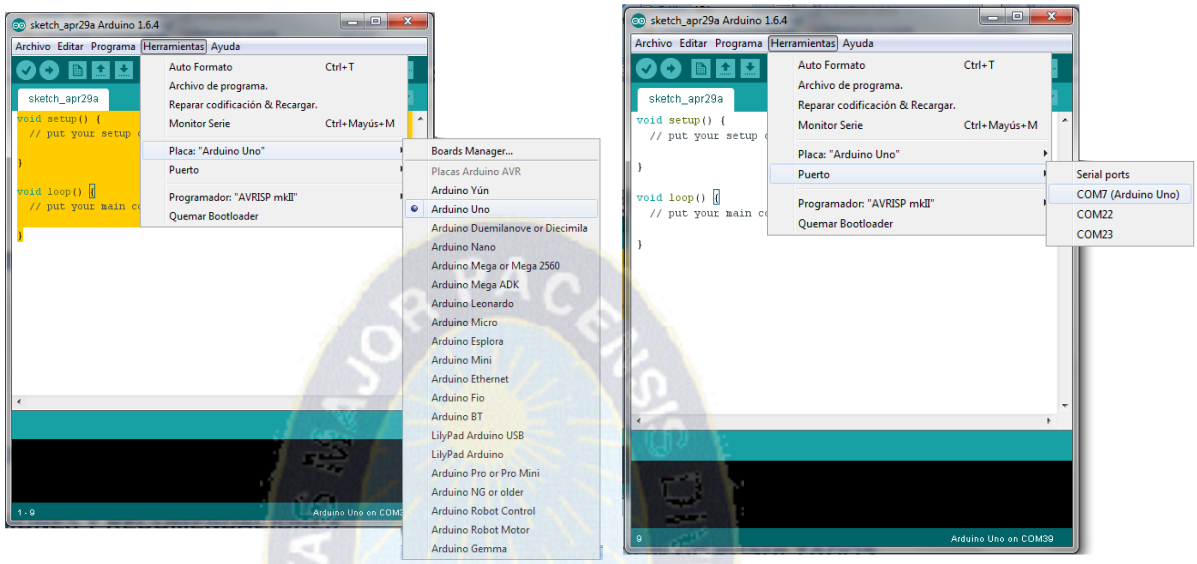
3.3. Instalación de id Arduino

Para poder hacer uso de las distintas funciones que ofrece Arduino, es indispensable haber instalado previamente el entorno de desarrollo. A través del mismo, generaremos los sketch con el código del programa que queramos cargar en nuestro dispositivo.

En primer lugar, se descargara la última versión de software disponible en la propia web de Arduino. (Arduino, 2015), se puede descargar el IDE conforme al sistema operativo que tengas, Windows, Mac OS X o Linux. Una vez descargado, se ejecuta el archivo y se debe seguir los pasos del asistente de instalación. Este paso no debe suponer ningún problema, su instalación es muy sencilla. En Windows, el Arduino es detectado automáticamente.

Una vez completada la instalación llega el turno de seleccionar el modelo concreto de nuestra placa Figura 3.2 a). Para ello se accede al menú "Herramientas", dentro del mismo se pondrá el cursor sobre la pestaña "Board", y a continuación clic sobre la versión correspondiente al modelo, en este

caso, "Arduino Uno". Luego escogemos puerto al que está conectado Arduino 3.2 b), caso contrario no se podrá subir el código del sketch a la placa Arduino.



a) **b)**

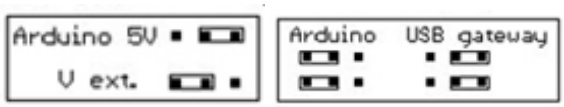
Figura 3.2: Configuración inicial de IDE Arduino.
Primero de sebe realizar la elección del modelo de la placa Arduino a) y luego escoger el puerto correspondiente al que este conectado b)

Para comenzar a trabajar con la placa, sólo tenemos que generar un nuevo sketch y comenzar a escribir el código de programación con las instrucciones que ejecutara Arduino.

3.4. Construcción del controlador

3.4.1. Configuración inicial de Shield GSM/GPS

En la página oficial se hace referencia al diagrama de conexiones del Shield, el cual podemos ver en detalle en la Figura 2.10 del capítulo anterior, se pone énfasis en tener mucho cuidado con los jumper de la placa, uno determina si la alimentación es externa o desde Arduino Figura 3.3 a) y los otros dos seleccionan el modo de funcionamiento de la placa Figura 3.3 b).



a) b)

Figura 3.3: Configuración de Jumper de Alimentacion del Shield.
Para determinar el modo de alimentación a) y funcionamiento b).

Como se ve en la Figura 3.3 a), se trata de un único jumper que admite dos posiciones, "Arduino 5V" y "V ext.". Si se coloca en posición "Arduino 5V", el módulo recibirá la alimentación a través del pin 5V de Arduino. En cambio, si el jumper está en posición "V ext.", esperará recibir la alimentación directamente por el pin V in ext. No se debe relacionar el modo alimentación "V. ext" con tener o no conectada la fuente externa a nuestra placa Arduino. Con esa fuente lo único que se hace es sobrealimentar la plataforma para que sea capaz de suministrar alimentación tanto a la Shield, como a los elementos conectados a ella. El modo "V ext." es para alimentar directamente el módulo a través de una fuente externa, excluyendo de esta función al Arduino. De cualquier otro modo se debe tener el jumper en posición "Arduino 5V", para que reciba alimentación a través del mismo.

Por tanto el jumper de alimentación debe estar en modo Arduino 5V.

Ahora para el modo de funcionamiento Figura 3.3. b), se trata de dos jumper con el que se puede seleccionar dos modos distintos de configuración para el módulo, "Arduino" y "USB gateway".

En el modo USB gateway, Arduino se convierte únicamente en una plataforma de paso de datos, es decir, hace de puente de para la comunicación entre la Shield GPRS/GSM y el puerto USB. Este modo es el que se debe seleccionar para comunicarse con el módulo por medio de comandos AT. De la misma manera, se colocara los jumper en esta posición cada vez que vayamos a cargar un programa en la memoria de Arduino. De no ser así el programa no se cargará. Otra opción es desconectar la Shield de la plataforma, de ese modo también podremos cargar un sketch en nuestro Arduino.

Los jumper estarán en Modo "Arduino" cuando se quiera ejecutar de manera automática un programa cargado en la memoria de Arduino, sin necesidad de tener que introducir manualmente los comandos AT.

Después de tener claro la configuración inicial de jumper de la placa se continuara con el montaje sobre Arduino UNO.

A continuación se empezara a hacer la conexión del Shield GSM/GPS con Arduino y se introduce el chip SIM, como muestra la Figura 3.4 b), también se conectara las respectivas antenas de GSM y GPS como se ve en la Figura 3.4 a).

a) **b)**

Figura 3.4: Conexión del Shield SIM908 y Arduino.
Primero es necesario conectar las antenas en la parte inferior del Shield a) luego poner conectarlo el mismo a Arduino b), teniendo mucho cuidado con los pines y no doblar ninguna conexion.

3.4.2. Pruebas iniciales del Shield GSM/GPS

Ahora si se puede conectar Arduino con la computadora, sin olvidar poner los Jumper del Shield en modo gateway ver Figura 3.3, se abre el IDE de Arduino, se carga el Sketch(por ahora con el código por defecto), y se abre el Motor Serial. En la ventana del motor serial se debe configurar la pestaña de selección de caracteres de fin de línea y se elige la opción "Ambos NL & CR" como se ve en la Figura 3.5.

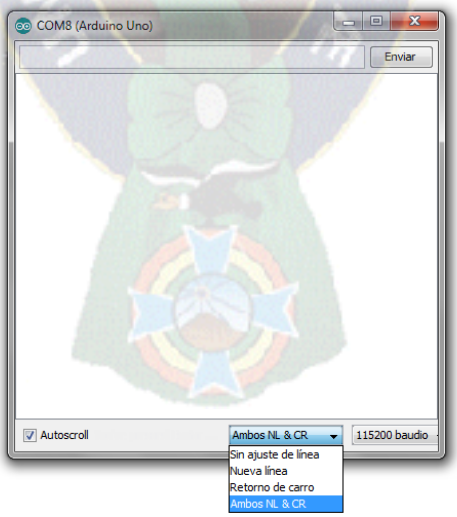


Figura 3.5: Configuración del motor serial de Arduino.
En el Motor serial es donde interactuaremos con el shield GSMGPS

Luego se enciende el Shield pulsando el botón de encendido por 2 segundos. Ahora si se puede realizar el intercambio de comandos AT, antes de empezar se debe tomar en cuenta algunos aspectos al momento de introducir los comando en el monitor de Arduino:

- Utilice mayúsculas para los comandos AT.
- Enviar CR (retorno de carro) y LF (avance de línea) después de que el comando AT.
- Coloque los puentes de comunicación en serie en la posición correcta.
- Utilice una fuente de alimentación externa y colocar los puentes de potencia en la posición correcta. Si el escudo se alimenta desde el Arduino, el puente de alimentación debe estar en la posición Arduino 5V. Es el escudo se alimenta desde la entrada Vin (en el escudo), el puente de alimentación debe estar en la posición Vext.

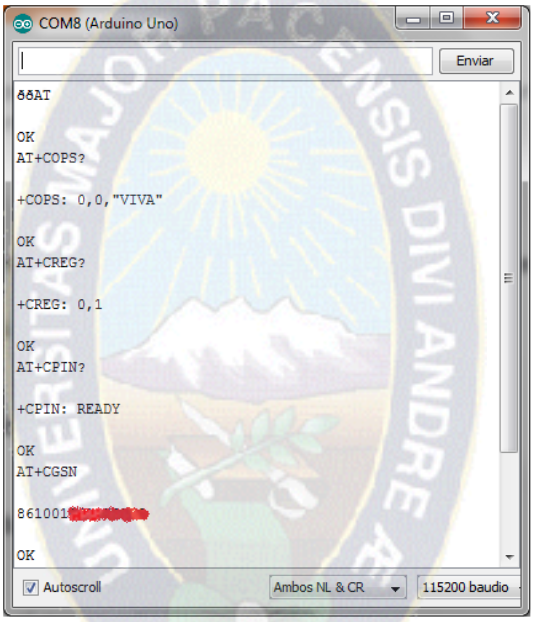


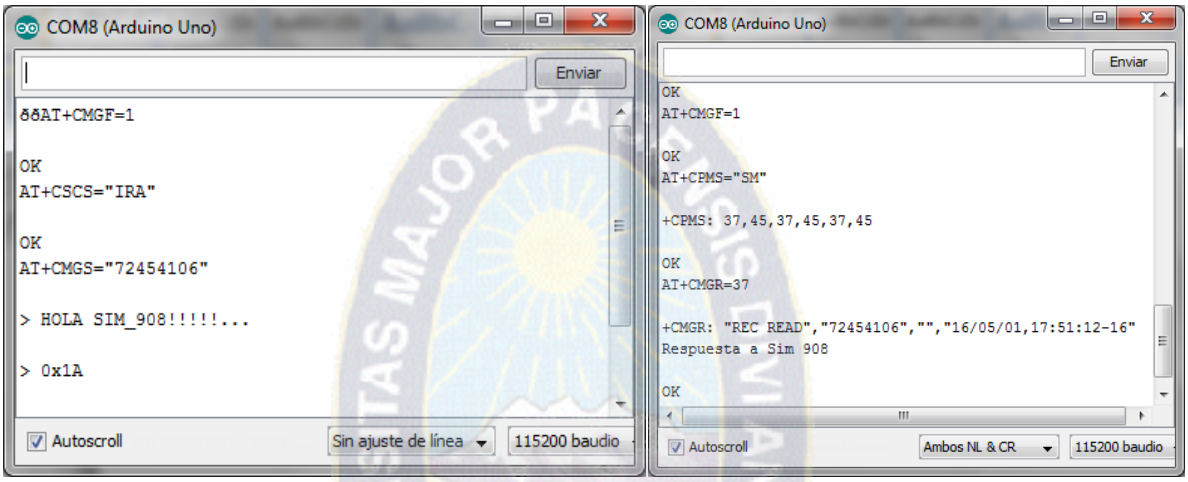
Figura 3.6: Prueba de funcionamiento correcto del Shield en el Monitor Serial de Arduino.

Como se observa en la Figura 3.6, se comienza enviando el comando "AT" para comprobar que el módulo responda correctamente a las instrucciones, si es así, responderá con un "OK". Luego se prueba el comando "AT+COPS?" el cual indicaraica la compañía telefónica a la que pertenece el chip SIM, que en este caso es VIVA, se revisa la información sobre el estado del registro con el comando "AT+CREG?", y por último se verifica el IMEI des Shield con el comando "AT+CGSN".

Para continuar con pruebas en la Figura 3.7 se realiza el envió de SMS. Para lo cual se introduce el comando "AT+CMGF=1" que se utiliza para configurar el Shield en modo SMS y "AT+CSCS=IRA". Después de recibir los OK correspondientes introducir el comando "AT+CMGS="numero móvil", donde el numero móvil es el del destinatario, a continuación se escribe el mensaje a enviar. Se debe tener la precaución de seleccionar la opción “No hay fin de línea” en lugar de “Ambos NL & CR” como se tenía seleccionado hasta este punto. Como se ve en la Figura

3.7 a) para terminar el mensaje se envía el carácter Hexadecimal “0x1A, y al cabo de unos segundo el mensaje será enviado y recibido.

Como todo ha salido según lo esperado y se ha recibido correctamente el SMS en el Smartphone, ahora se pasa a realizar la prueba del envío de un SMS pero esta vez, a través de Arduino, sin necesidad de teclear manualmente los comandos AT.



a) b)

Figura 3.7: Uso del monitor serial de Arduino para envió de SMS a) y lectura de SMS recibidos b)

Ahora para la lectura de de un SMS primero se configura al modo SMS con el comando "AT+CMGF=1", luego el comando "AT+CMPS="SM"", que encuentre el número de mensajes que están almacenados actualmente en el área de almacenamiento de mensajes, en este caso el último mensaje recibido es el 37, para leer dicho mensaje introducir el comando "AT+CMGR=37" tal como se ve en la Figura 3.7 b) , se mostrara el mensaje recibido, con el numero de origen fecha, hora y texto.

Ahora se verificara el funcionamiento del GPS, para lo cual primero se verifica si el GPS esta encendido con el comando "AT+CGPSPWR?", si la respuesta es 0 está apagado y si es 1 esta encendido, en el caso de que este apagado se introduce el comando "AT+CGPSPWR=1", después se puede resetear con el comando "AT+CGPSRST=1", se espera unos minutos y se verifica el estado del GPS con el comando "AT+CGPSSTATUS?", el cual da 3 posibles respuestas, Localidad desconocida, 2D fix que puede dar una posición bidimensional latitud, longitud y 3D fix que da la posición en tres dimensiones latitud, longitud y altura y esta última es la más precisa. Para obtener las coordenadas se introduce "AT+CGPSINF=0", y como se ve en la Figura 3.8 la respuesta son 9 datos(frame, longitud, latitud, altitud, tiempo UTC que está en formato yyyymmddHHMMSS, TTFF, numero de satélites a la vista, velocidad, curso).

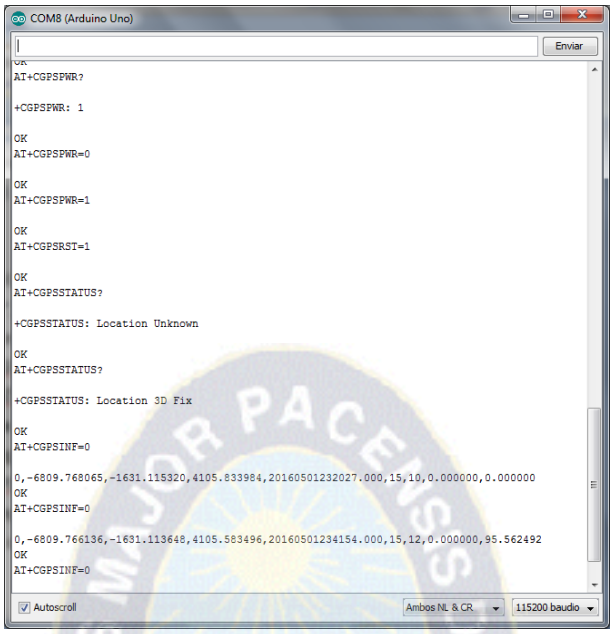


Figura 3.8: Verificación del funcionamiento del GPS y obtencion de coordenadas

Hasta ahora los comandos fueron introducidos manualmente de aquí en adelante se empezara a utilizar el IDE de Arduino para cargar el código necesario para realizar el envío y recepción de SMS así como obtener la ubicación GPS.

Los pasos a seguir para el envío de SMS son:

1) Generar el sketch con los comandos AT adecuados para que el módulo envíe un SMS. el cual está en el Programa 3.1.

2) Verificar el estado de los jumper, es necesario para cargarlo en modo "gateway".

3) Conectar la plataforma compuesta por Arduino y el módulo GSM al puerto USB del ordenador.

4) Cargar el sketch en la memoria Arduino, luego desconectar de la Computadora.

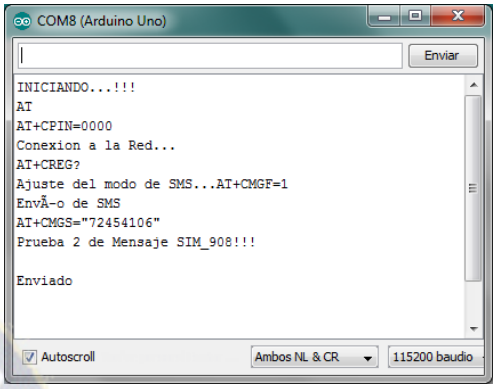
5) Modificar la posición de los jumper a modo "Arduino"

6) Conectar la plataforma a la fuente de alimentación externa, por ahora se solo es necesario conectarlo a la Computadora.

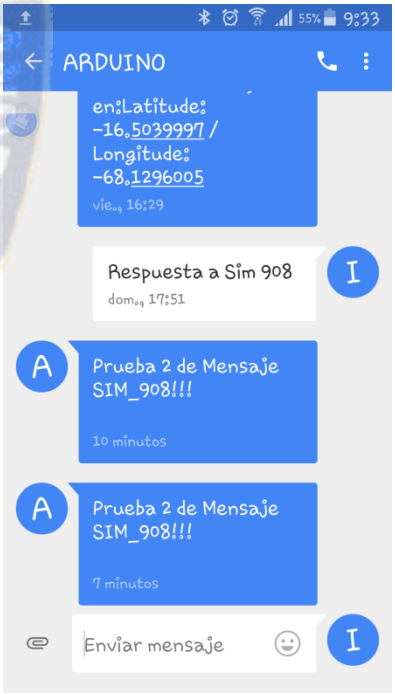
7) El programa comenzará a ejecutarse, como se ve en la Figura 3.9 a),al abrir el motor serial se podrá observar los comandos enviados. Si todo esta correcto, tras unos segundos recibirá el SMS el Smartphone, tal y como se muestra en la Figura 3.9 b).

Programa 3.1: Código para el envío de un SMS

```
int8_t answer;

int onModulePin= 2;

char aux_string[30];

char phone_number[]="72454106";     // número de Celular destino

char pin[] = "0000";                                    //Código PIN

char sms_text[]="Prueba 2 de Mensaje SIM_908!!!";

void setup(){

   pinMode(onModulePin, OUTPUT);

   Serial.begin(115200);

   Serial.println("Starting...");

   power_on();

   delay(3000);

   // Verificacion de numero PIN

   sprintf(aux_string, "AT+CPIN=%s", pin);

   sendATcommand(aux_string, "OK", 2000);

   delay(3000);

   Serial.println("Connecting to the network...");

   while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||

         sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );

   Serial.print("Setting SMS mode...");

   sendATcommand("AT+CMGF=1", "OK", 1000); // Entrando al Modo
SMS

   Serial.println("Sending SMS");

   sprintf(aux_string,"AT+CMGS=\"%s\"", phone_number);
```



a)



b)

Figura 3.9: Monitor Serial con el envío automático de SMS a) y la respectiva verificación en el móvil destino b)

Para la lectura de SMS y la Posición GPS se debe seguir los mismos pasos y los Sketch necesarios estarán disponibles en los Anexos.

A continuación se procede a armar el dispositivo con los sensores que son necesarios y el sketch necesario para el correcto funcionamiento del Dispositivo.

3.4.3. Desarrollo del controlador

Y se pudo observar en puntos anteriores el montaje del Shield SIM908 en la placa Arduino y su correcto funcionamiento, ahora se empezara a adicionar los sensores necesarios para que el dispositivo tengo un correcto funcionamiento.

Primero el sensor de presencia PIR HC-SR50, se conecta su pines VCC y GND a 5 Voltios y Ground, el pin OUT va conectado al pin 6 de Arduino como muestra la Figura 3.10, el cual esta delegado a capturar los datos al existir presencia en el interior del automóvil.

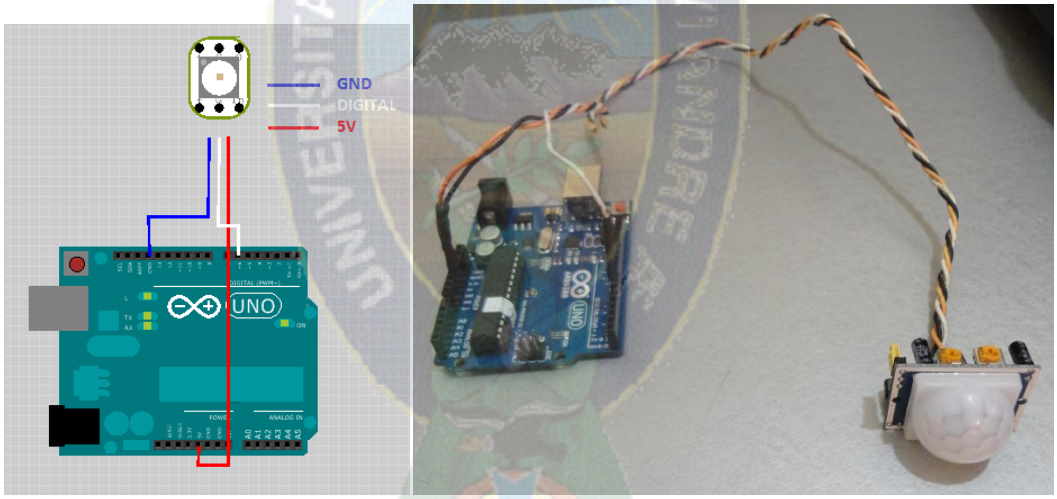


Figura 3.10: Conexión del Sensor de Presencia a la placa Arduino.

Se prosigue con la instalación del sensor de dióxido de carbono(CO2), el cual consta de 4 pines pero, solo se usara 3, VCC a 5 Voltios, GND a Ground, y el pin AO que sirve para la lectura analógica del nivel de CO2 e ira conectado al pin A0 de la placa Arduino, debido a que al inicializar el Sketch se hace un configuración inicial de este sensor obteniendo el nivel de CO2 presente con el automóvil cuando está apagado, para luego hacer la comparación necesaria, la conexión de puede observar en la Figura 3.11.

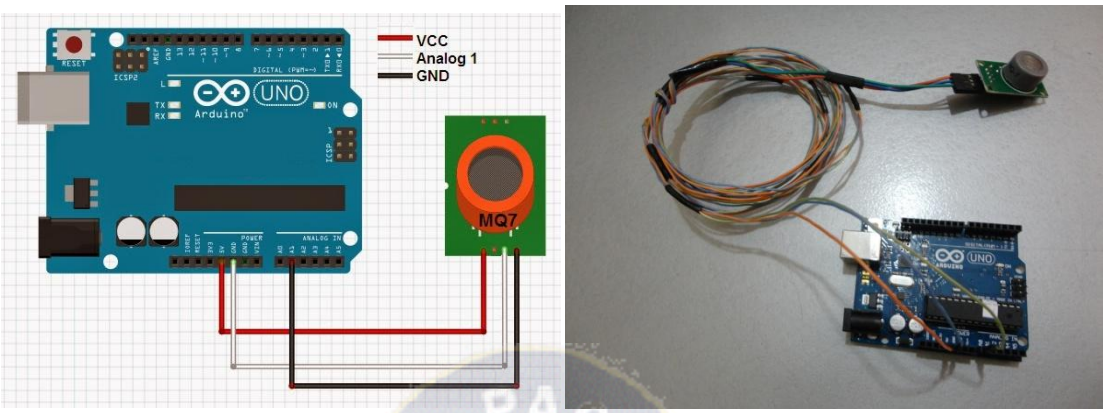


Figura 3.11: Conexión de Sensor MQ 7

Se conecta un control de radiofrecuencia RF, YK04, el pin VT sirve para detectar si hay una transmisión valida, por otra parte, los pines D0, D1, D2, D3 se activan con un HIGH, cuando se pulsa el botón A, B, C, D respectivamente como muestra la Figura 3.12, por lo que se va a conectar a los pines digitales de **Arduino** 8, 9, 10, y 11. Este control tendrá dos propósitos, primero al botón A se le dará la tarea de enviar la posición actual del vehículo a algún familiar del conductor, y el botón B podrá activar o desactivar el funcionamiento de dispositivo.

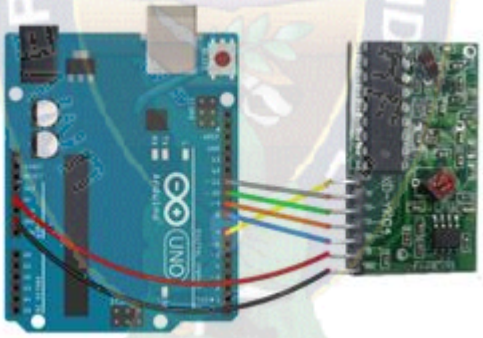


Figura 3.12: Conexión del Control RF

La ubicación de los sensores de CO2 y el de presencia no necesitan realizar conexiones con el sistema eléctrico del automóvil, pero ahora para bloquear el encendido o apagar el automóvil sin importar la distancia que lo separe del propietario es necesario hacer una conexión, la misma que dependerá mucho del modelo de automóvil, puesto que es posible cortar la energía a la bomba de gasolina o cortar la energía a alguna modulo de encendido.

Para realizar el bloqueo se necesitara el modulo Relé, el cual necesita 5 Voltios y Ground, y la señal que recibirá para activarse será por el pin 13 de la placa Arduino. Este Relé también funcionara con el control RF.

Como se menciono en el capítulo 2 de Marco Teórico en la sección de del funcionamiento de una automóvil, queda claro que el funcionamiento de un automóvil es un mundo inmenso el cual es regulado por distintas unidades de procesamiento, ahora para realizar el bloqueo aquí se tiene las opciones de quitar energía a la unidad de control de motor ECU, y otra opción es quitar anergia a la bomba de gasolina. La última opción es mucho más factible debido que es muy peligroso hacer algún cambio de energía directamente a la ECU, por el hecho de que puede ocasionar algún problema en el mismo(quemar alguna pieza eléctrica), además que es más sencillo ubicar los cables de alimentación a la bomba de gasolina.

Un modelo sencillo de cómo va instalada una bomba de gasolina puede ser observado en la Figura 3.13 a) donde se puede notar que hay un relé entre la llave y la bomba de gasolina el cual es activado cuando de mueve la posición de la llave a encendido. Para la opción de agregar un relé en adicional para nuestro controlador pueda bloquear el automóvil, la ubicación más adecuada es antes de la bomba de gasolina y quedaría como muestra la Figura 3.13 b).

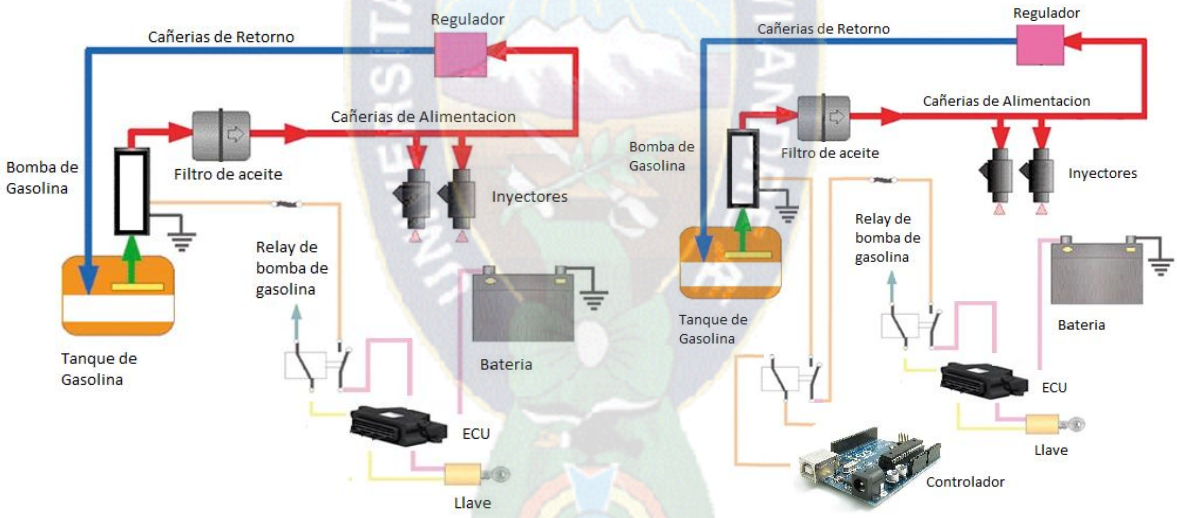


Figura 3.13: Instalación original de una bomba de gasolina a) y conexión del controlador Arduino al sistema de alimentacion de energia a la bomba de gasolina

En las siguientes figuras se observa cómo es el montaje del relé en una auto de forma real, para la Figura 3.14 a) se muestra la conexión realizada con el esquema superior, pero para la Figura 3.14 b) se conecta el relé al cable que lleva la energía a relé original de la bomba de gasolina.

Figura 3.14: Montaje del Relé a un automóvil Ford Escort, conexión directo a la bomba de Gasolina a) y al relé original b)

A antes de montar el controlador al automóvil se debe añadir un protoboard con Led para realizar pruebas necesarias y si algún sensor es activado nos notificara encendiéndose un Led. Y el diseño final de los sensores será como muestra la Figura 3.15.

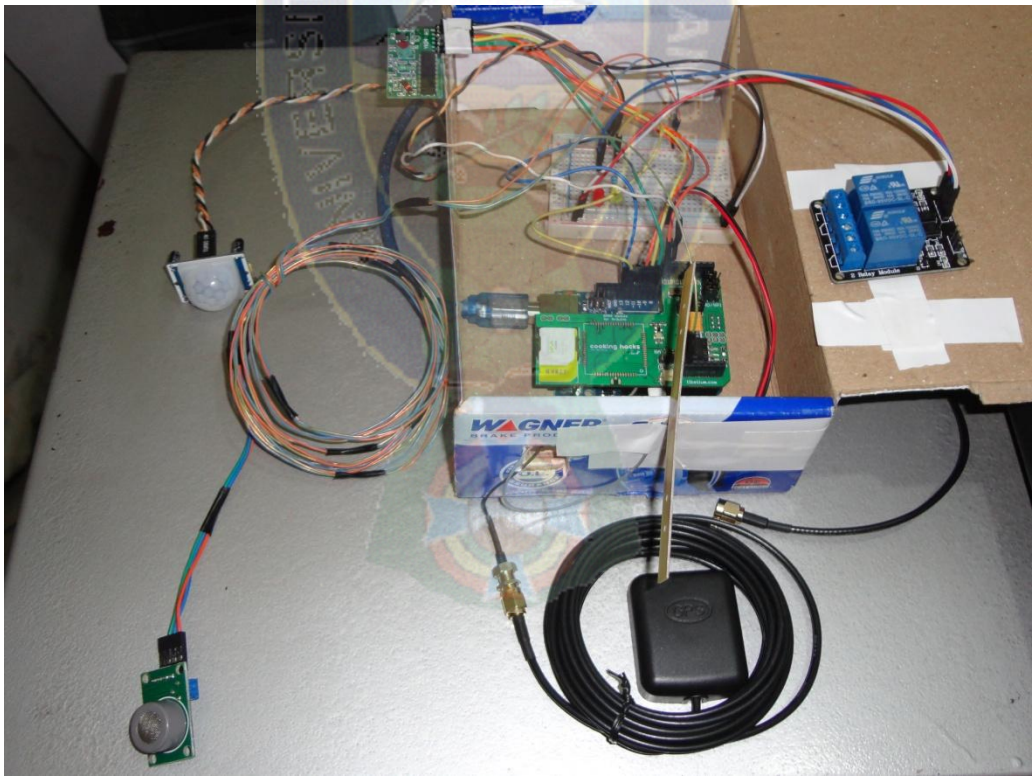


Figura 3.15: Placa Arduino con la integración de los sensores necesarios para el control y seguimiento.

Ahora ya está listo todo los materiales necesarios y conectados de manera correcta como muestra la Figura 3.16, se dará inicio a la creación de el sketch necesarios para que el controlador funcione . Para lo cual se diseñara un diagrama de flujo Figura 3.17, con los principales pasos que deberá seguir. Con el cual se dará inicio a programar el código necesario para que el controlador funcione, tiendo como ayuda varios sketch que individuales de cada uno de los sensores y Shield (valores del sensor de CO2, revisar si hay presencia con el sensor PIR, envió y lectura de SMS, manejo de Relé para encender o apagar, etc.).

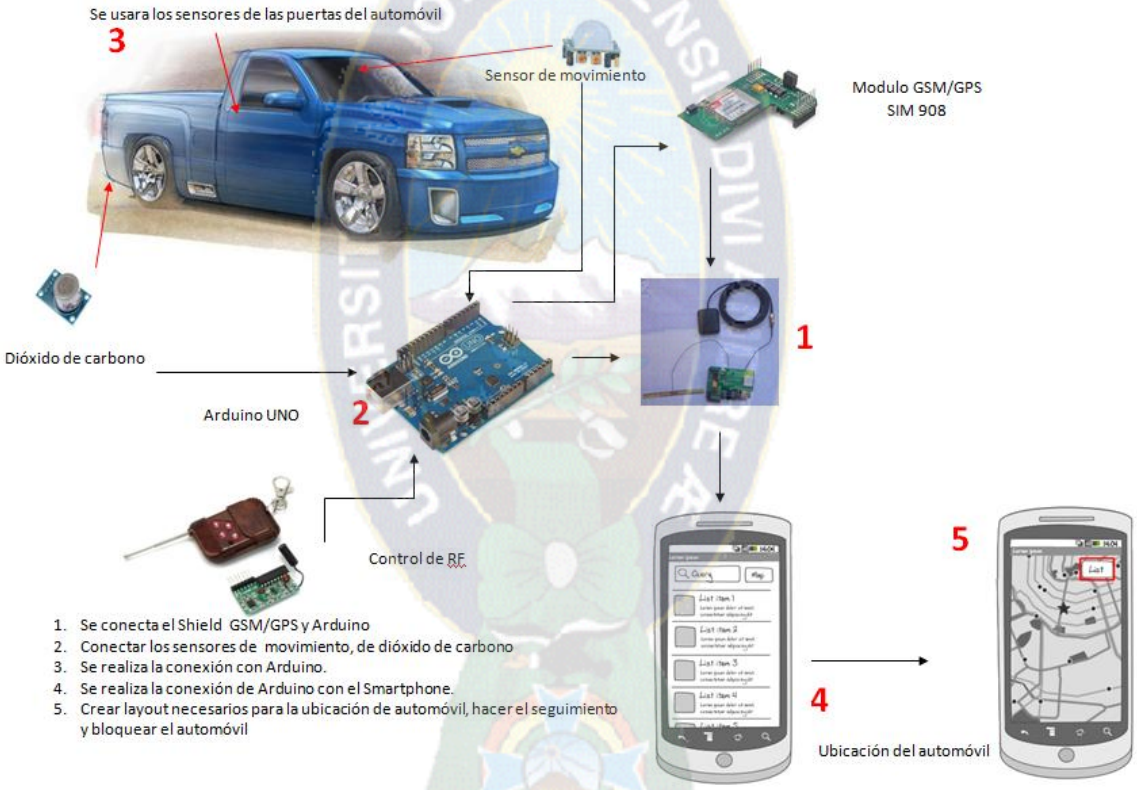


Figura 3.16: Diseño de Ubicación de los sensores y conexión con el Smartphone

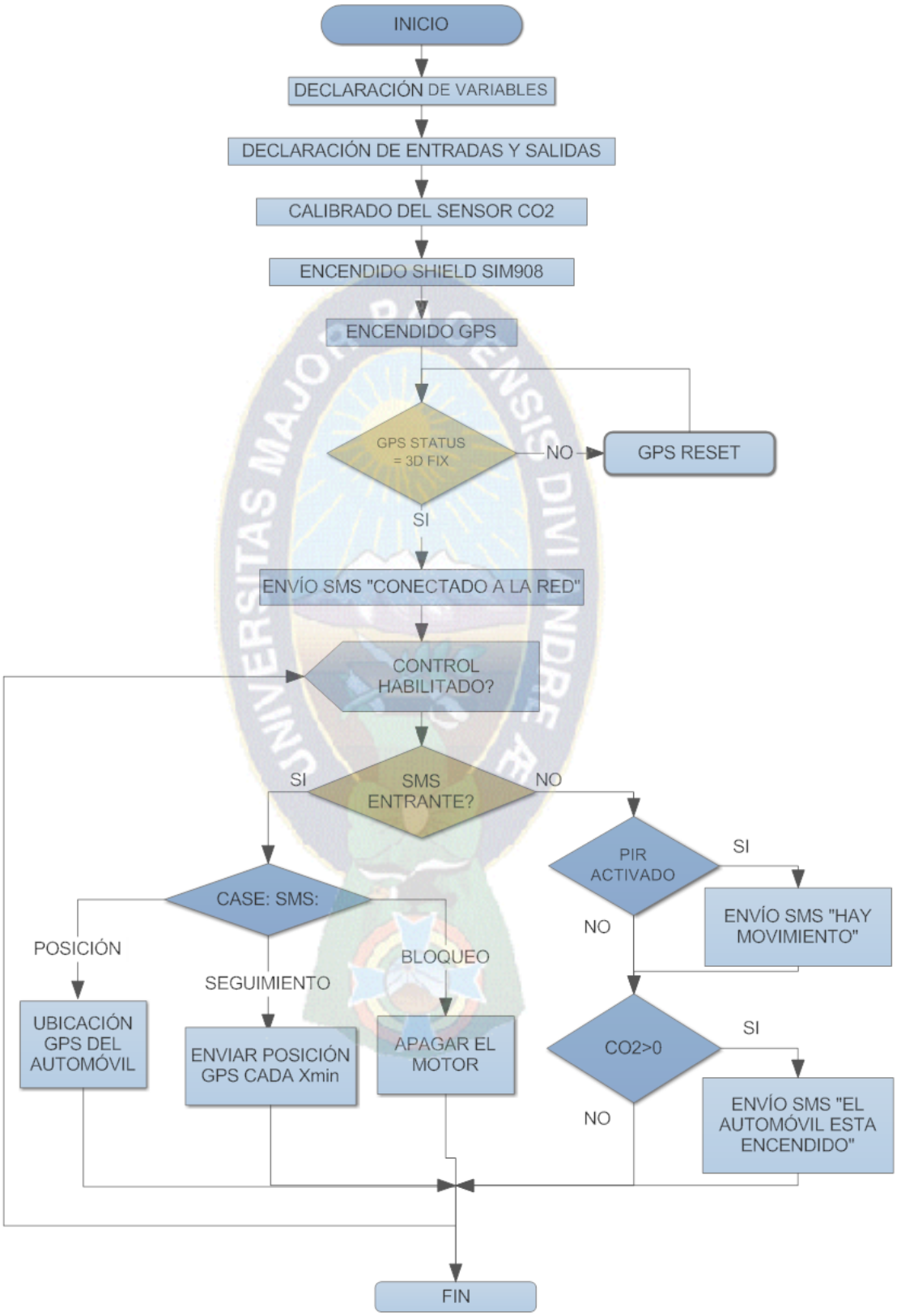


Figura 3.17: Diagrama de flujo correspondiente a el controlador

3.4.4. Codificación del controlador.

La creación del Sketch necesario para el funcionamiento del controlador se basa en conexión la habilitación del GPS ya que si este no logra una conexión 3D fix, el controlador en algún momento podrá mandar una coordenada errónea. A continuación en la Figura 3.18 se puede observar la función que da inicio a GPS, el cual espera una respuesta 3D Fix.

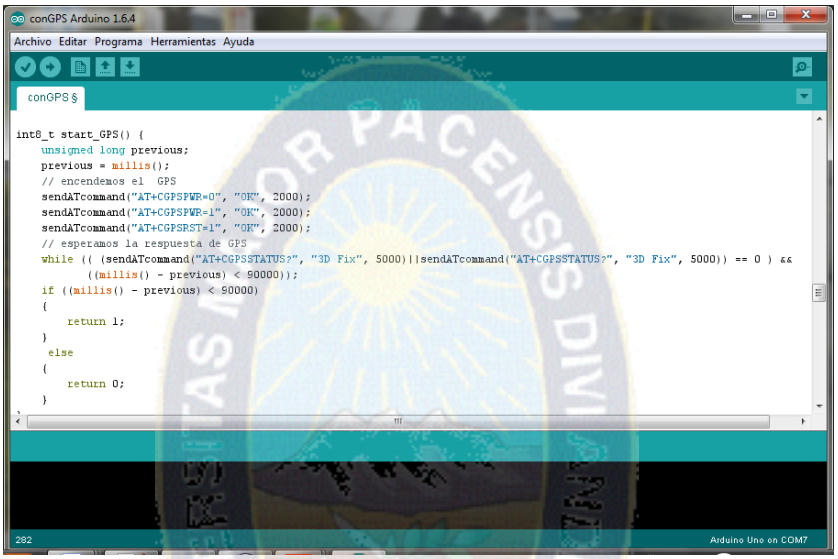


Figura 3.18: Función para el encendido del GPS del controlador

Mandar mensajes SMS de texto también es importante también en la Figura 3.19, se muestra un fragmento del código necesario para realizar esta operación,

Figura 3.19: Fragmento de la función para el envío de SMS

El Sketch completo que se usara para el controlador se lo puede ver en el Anexo A, junto con los funciones necesarias para su correcto funcionamiento

3.5. Desarrollo de la aplicación móvil

La aplicación esta plateada para poder recibir y mostrar la ubicación exacta de una automóvil con la colaboración del controlador basado en la placa Arduino, y la tecnología de envió de SMS, el Smartphone, es capaz de recibir coordenadas de Geolocalización. La aplicación debe funcionar en cualquier área que exista cobertura GSM, también necesita una conexión a Datos de internet o un WiFI. En la Figura 3.20 vemos una descripción general del funcionamiento de la Aplicación.

Para el diseño y desarrollo de la aplicación de el prototipo se sigue le metodología Mobile-D, el cual fue descrito en el anterior capitulo, ya que es el más adecuado para desarrollo ágil de aplicaciones móviles. Recibira datos del controlador, los cuales serán empleados para tomar dar solución a los objetivos planteados.

Figura 3.20: Descripción general del Sistema
Controlador instalado en el Automóvil se conecta por rede GSM al Smartphone y este a travez de google maps obitne la ubicación del automovil

3.5.1. Exploración

El propósito de la exploración es la planificación y el establecimiento del proyecto, dando las bases de la aplicación definiendo el planeamiento y establecimiento del proyecto, para que de esta manera poder implementar el producto en relación al desarrollo de software, y sentar las bases del mismo empezando desde los requerimientos iníciales.

3.5.1.1. Establecimiento de Stakeholders

El propósito de esta tarea es establecer el grupo de interés:

Propietario de Automóvil: Que teniendo la información recibida del controlador al Smartphone este pueda conocer la ubicación del automóvil, si el propietario desea poder hacer un seguimiento del mismo mientras continúe en funcionamiento, podrá verificar el estado del mismo, si hay algún movimiento brusco, si abren alguna puerta o capo, y en caso de robo poder bloquearlo para que deje de funcionar.

3.5.1.2. Actores

Los actores del sistema involucrados con la construcción de la plataforma de Geolocalización de personas son principalmente el usuario, aplicación, controlador y el automóvil.

3.5.1.3. Establecimiento del Proyecto

El escenario que queremos aplicar es el tener toda la información estado y posición GPS del automóvil, en un único dispositivo electrónico que será el controlador. Para lo cual se escogió la placa Arduino junto con el Shield SIM908 que recogerán datos de distintos sensores y serán enviados al Smartphone, el que estará encargado del control y seguimiento.

La comunicación entre el controlador y el Smartphone Android será posible mediante mensajes de texto.

Una vez realizada la comunicación entre el controlador y el Smartphone Android, este último confirmara al usuario luego se hará una llamada para la obtención del mapa con la posición actual del propietario y del Automóvil.

EL Smartphone también mandara datos para que los actuadores del controlador puedan ejecutar alguna acción(bloquear el automóvil, o deshabilitar el controlador).

3.5.1.4. Análisis sobre requisitos del sistema

En este punto del análisis se van a enumerar los requisitos que se deben cumplir para poder hacer un buen uso de esta aplicación.

- Se deberá contar con un dispositivo Smartphone con sistema operativo Android 4.4 o superior.
- Se necesitará un plan de datos o una red inalámbrica WiFi en el Smartphone.
- El Smartphone deberá tener GPS interno.
- La aplicación deberá ser fácil de utilizar para cualquier tipo de usuario.
- Se minimizarán los errores que se pudieran producir a lo largo de la comunicación.

3.5.2. Inicialización

El propósito de esta fase es la de permitir el éxito de las siguientes fases mediante la elaboración y verificación de los requisitos seleccionados por el cliente.

La planificación del proyecto entorno a la parte técnica se hará durante esta fase, definiendo el editor, lenguaje y APIs con el que se va a trabajar la aplicación móvil como tal, se lista las funcionalidades que se hacen necesarias dentro de los módulos a desarrollar.

3.5.2.1. Entorno de trabajo

- JDK java.
- Eclipse para desarrollo de aplicaciones Android.
- ADT (plugins).
- Android SDK(Librerías y paquetes).
- API Google Maps.

3.5.2.2. Lista de funcionalidades

Para llevar con éxito el desarrollo del prototipo, se definió una serie de tareas Tabla 3.2 :

Tabla 3.2: Requerimientos iniciales para el desarrollo de la aplicacion

REQUERIMIENTO	IMPORTANCIA
CONEXIÓN CON EL CONTROLADOR	10
CONFIRMACIÓN DE USUARIOS	9
INFORMACIÓN POSICIÓN EN TIEMPO REAL	9
INFORMACIÓN POSICIÓN PARA SEGUIMIENTO	9
ENVIÓ DE DATOS AL CONTROLADOR	8
TRAZADO DE RUTA	7
PRUEBAS DE ACEPTACIÓN	8

- **Módulo de Conexión**
  a. Recepción de SMS.
  b. Envió de SMS
  c. Establecer conexión.
  d. Verificar de conexión

- **Módulo de Alertas**
  a. Procesamiento datos.
  b. Presentación de datos.

- **Módulo de Ubicación GPS del Smartphone**
  a. Cargar mapas.
  b. Pedir la ubicación actual del Smartphone

- **Módulo de Ubicación del Controlador**
  a. Recepción de datos
  b. Procesamiento de datos
  c. Presentación de datos
  d. Actualización de datos en el mapa

- **Modulo respuestas al controlador**
  a. Envío de SMS hacia el controlador

3.5.3. Iteraciones

Tabla 3.3: Descripción de las iteraciones según la metodologia Mobile D

FASE	ITERACIÓN		DESCRIPCIÓN
Exploración			
Inicialización	**Iteración 0**		Establecimiento del proyecto, Entrenamiento, Análisis de requerimientos iniciales.
PRODUCCIÓN	**Iteración 1**	**Iteración Modulo de Conexión**	Implementación del módulo de conexión. Refinamiento y actualización de historias de usuario. Mejora de interfaces. Generación y ejecución de pruebas de aceptación.
		Iteración Módulo de Alertas	Implementación del modulo de Diálogos de alerta y validación de Usuario. Generación y ejecución de pruebas de aceptación
	Iteración 2	**Iteración Módulo de Ubicación GPS del Smartphone**	Implementación del modulo de Ubicación GPS, habilitar el Ubicación del Smartphone. Generación y ejecución de pruebas de aceptación
		Iteración Módulo Ubicación del controlador y seguimiento	Implementación del modulo de Ubicación de GPS y seguimiento. Generación y ejecución de pruebas de aceptación
	Iteración 3	**Iteración Modulo respuestas al controlador**	Implementación del modulo de respuestas al controlador. Generación y ejecución de pruebas de aceptación
ESTABILIZACIÓN		**Iteración Modulo de Conexión**	Refactorización del módulo de conexión. Refinamiento y actualización de historias de usuario. Refinamiento de interfaces. Generación y ejecución de pruebas de aceptación.
		Iteración Módulo de Alertas	
		Iteración Módulo de Ubicación GPS del Smartphone	
		Iteración Módulo de Ubicación GPS del Controlador	
		Iteración Modulo respuestas al controlador	
PRUEBAS DEL SISTEMA		**Iteración pruebas del sistema**	Se realiza la evaluación de las pruebas y se realiza el análisis de los resultados.

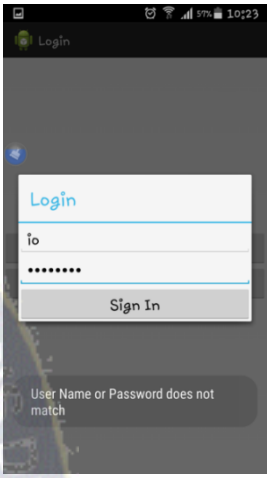
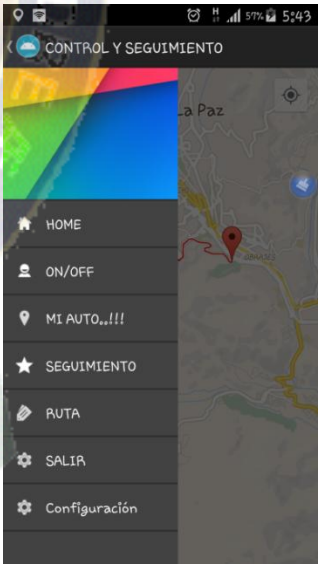
3.5.4. Descripción de la interfaz del usuario

La interfaz es el medio con que se puede comunicar el usuario el Smartphone y así mismo también comunicarse con el controlador, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar y se describirá en la Tabla 3.4.

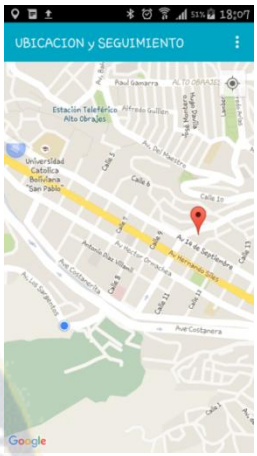
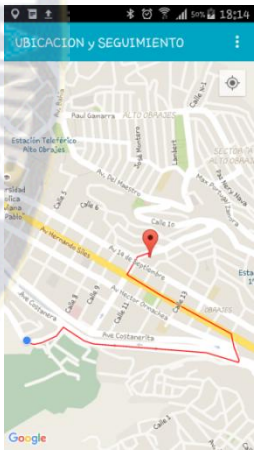
Tabla 3.4: Tabla de descripción de las interfaz de comunicación entre la Aplicación y el Propietario del automóvil

MENÚ DE REGISTRO	
Es la primera pantalla a a la que tendrá acceso después de la instalación de la aplicación se introducirá usuario y clave para que el propietario pueda ingresar	
ALERTA CUANDO HAY MOVIMIENTO EN EL AUTOMÓVIL	
Este cuadro de dialogo aparecerá cuando haya movimiento en el auto, el mensaje del controlador recibido será tratado casi al mismo tiempo que el sistema en general	

.

CLAVE DE INGRESO	
No cualquiera podrá ingresar a la aplicación por lo que seguido al cuadro de dialogo tendrá que ser validado para el ingreso a la aplicación	
MENÚ PRINCIPAL	
Una vez dentro se puede ingresar al menú de opciones: **Home:** Muestra la ubicación del propietario en el mapa. **ON/OFF**: Es para poder habilitar o deshabilitar el automóvil. **MI AUTO..!!!:** Pide la ubicación exacta del automóvil. **SEGUIMIENTO:** Realizar el seguimiento del auto en un determinado tiempo. **RUTA:** Traza la ruta entre el propietario y el automóvil. **Salir:** Sal de la aplicación aunque siempre estar corriendo en segundo plano.	

.

PANTALLA CON LA UBICACIÓN DEL AUTOMÓVIL Y EL PROPIETARIO	
Pantalla en la cual se puede observar a Propietario y el Automóvil, la cual después de validarse aparece directamente.	
TRAZADO DE LA RUTA PARA LLEGAR AL AUTOMÓVIL	
El trazado de la ruta para llegar al automóvil cualquiera sea el caso, perdida, robo o desconocer donde está el automóvil.	

3.5.5. Producción e implementación.

Para esta fase ya se hace el desarrollo de la aplicación en su totalidad, una vez establecidas las listas de funcionalidades se comienza a desarrollar según el método que se recomienda en esta metodología (Planificación, trabajo, liberación) haciendo las iteraciones necesarias para la programación. Identificando la cantidad de actividades, diálogos y formularios que se necesiten para desarrollar los la aplicación.

La intensión de la aplicación primero es poder conectarse con el controladores que estará instalado en el automóvil, de ese modo simplificar el manejo de la información obtenida sobre el estado y el acceso a la posición del automóvil con respecto del propietario para luego ser observada en un mapa.

3.5.5.1. Permisos para recepción y envío de SMS.

Para realizar la conexión con el controlador es necesario poder verificar la recepción de un mensaje SMS recibido desde el controlador, es muy importante ya que toda la comunicación será a través de mensajes SMS. En el Programa 3.2 se ve el procedimiento para recibir SMS.

Programa 3.2: Procedimiento para recibir SMS

```
public void onReceive(Context context, Intent intent) {
	//Log.d("SMStutorial","SMSReceiver.onReceive");
	Bundle bundle = intent.getExtras();
	// obtenemos el array de estructuras pdus
	Object messages[] = (Object[]) bundle.get("pdus");
	SmsMessage smsMessage[] = new SmsMessage[messages.length];
	//Se crea un array donde guardamos el mensaje recibido
	for (int n = 0; n < messages.length; n++) {
	// generamos mensajes sms a partir de las estructuras pdu
	smsMessage[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
		}
}
```

Además para poder responder y poder comunicarse desde el Smartphone al controlador se necesita un procedimiento para el envío de SMS, el cual se puede observa en el Programa 3.3.

Programa 3.3: Procedimiento para envío de SMS.

```
private void sendSMS(String phoneNumber, String message){
		SmsManager sms=SmsManager.getDefault();
		sms.sendTextMessage(phoneNumber,null,message,null,null);
}
```

Estos dos procedimientos son muy importantes a la hora de comunicarse con el controlador Arduino pero antes de ser utilizados se debe aumentar permisos a la aplicación los cuales están ubicados en el Manifest de la aplicación el cual deberá quedar como en las siguientes líneas de código del Programa 3.4.

Programa 3.4: Permisos para recibir o enviar SMS que deben ser aumentados en el Manifest.xml

```
<--permisos para de recibir o enviar SMS -->
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>

<--Aquí se da el permiso para que la aplicación responda al SMS entrante -->
<aplication
......
<receiver android:name=".SMSReceiver"android:enabled="true">
            <intent-filter>
                <action
android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
</receiver>
```

3.5.5.2. Cuadro de dialogo de Alerta .

La aplicación recibirá del controlador un mensaje de conexión con el mismo, momento desde el cual ya puede interactuar ambos. En el momento de recibir un SMS se activara un cuadro de dialogo automático el cual nos indicara si deseamos acceder a la aplicación o ignorarlo. Para realizar este trabajo necesitaremos crear una nueva actividad la clase AlertDialogActivity.java, el cual debe ser creado donde esta nuestra clase principal MainActyvity.java, A continuación en el Programa 3.5 se muestra un fragmento de código de la nueva clase.

Programa 3.5: Fragmento de la Clase AlertDialogActicity

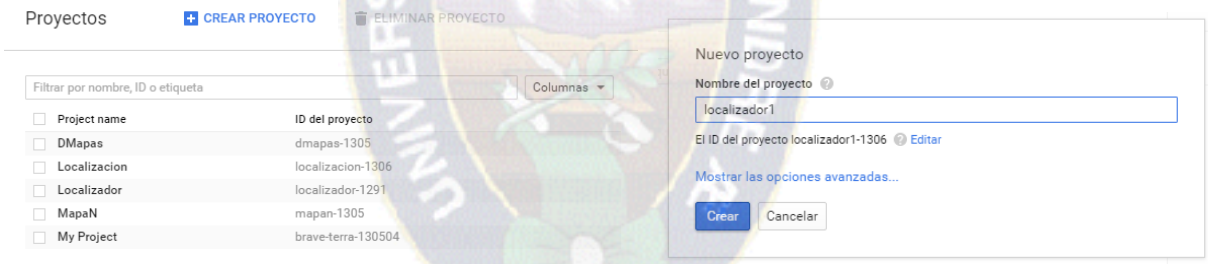
```
public class AlertDialogActivity extends Activity {
      String LatLon;
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    LatLon = getIntent().getStringExtra("LatLong");
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder
        .setTitle("ALERTA")
        .setMessage("Hay movimiento en su auto")
        .setCancelable(false)
        .setPositiveButton("Yes", new DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialog, int id) {
             Intent i = new Intent(AlertDialogActivity.this,
MainActivity.class);
              startActivity(i);
            }
        })
       .setNegativeButton("No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish();
            }
```

3.5.5.3. Uso del GPS API de GoogleMaps.

Como será una aplicación para ubicar un automóvil se necesitara hacer uso de mapas, por ello el prototipo del sistema, trabajará con el sistema GPS que brindan los dispositivos Smartphone y el sistema de mapas de GoogleMaps para visualizar la ubicación del Automóvil.

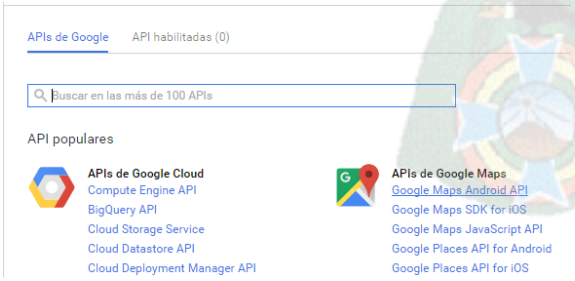
Para realizar este trabajo es necesario obtener un API key, lo cual es un requisito para trabajar con GoogleMaps, y puede ser obtenida en el siguiente enlace: https://developers.google.com/maps/.

Los pasos para obtener una API Key para utilizar el servicio de mapas de Google en una aplicación son: una vez accedido al enlace, se creara un nuevo proyecto mediante el botón “CREAR PROYECTO" Figura 3.21 a), luego aparecerá una ventana que solicitará el nombre del proyecto y generara automáticamente un ID 3.21 b), después en el administrador de APIS se habilita **Google Maps Android API** 3.21 c), después se debe añadir las credenciales para al proyecto 3.21 d) y se obtendrá la API key para trabajar en nuestra aplicación 3.21 e).

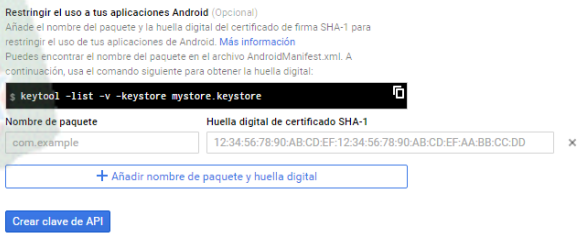


a)

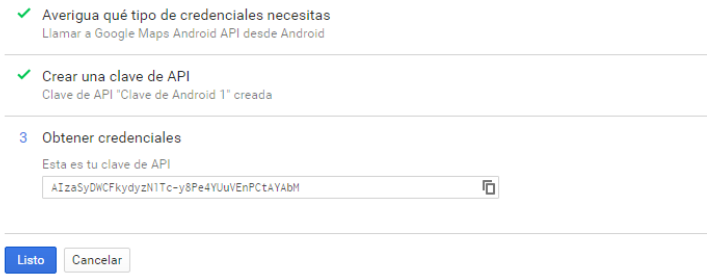
b)



c)



d)



e)

Figura 3.21: Pasos para obtener una API key de Google.
"CREAR PROYECTO" a), generacion automátio de ID b), habilitar Google Maps Android API c), después se debe añadir las credenciales para al proyecto d) y se obtiene la API key e).

Con esto ya está listo los preparativos iniciales necesarios para utilizar el servicio de mapas de Android en nuestras propias aplicaciones.

Para empezar el uso de los mapas se debe añadir al fichero AndroidManifest.xml la API Key que se genero. Para ello se añade, dentro de la etiqueta <application>, un nuevo elemento <meta-data> como muestra el Programa 3.6, además se agregara los permisos necesarios para el uso del mismo como ser: permiso para acceder a internet, conocer el estado de la red (ACCESS_NETWORK_STATE), acceder al almacenamiento externo del dispositivo para la caché de mapas (WRITE_EXTERNAL_STORAGE) y hacer uso de los servicios web de Google (READ_GSERVICES), además de el permiso para poder acceder a la ubicación del Smartphone en nuestra aplicación (ACCESS_LOCATION_EXTRA_COMMANDS).

Programa 3.6: Agregar la API key de Google

```
<aplication
......
<meta-data android:name="com.google.android.maps.v2.API_KEY"
           android:value="Aquí va la clave generada por GOOGLE"/>
<--jsb-->
......
/aplication>

<--Y los permisos corresponidentes -->
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
      <uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
```

El siguiente paso será referenciar el proyecto de librería de los Google Play Services desde proyecto y accediendo a la sección Android de las preferencias. En dicha ventana se puede añadir una nueva librería que en este caso es la de Google APIs Figura 3.22.

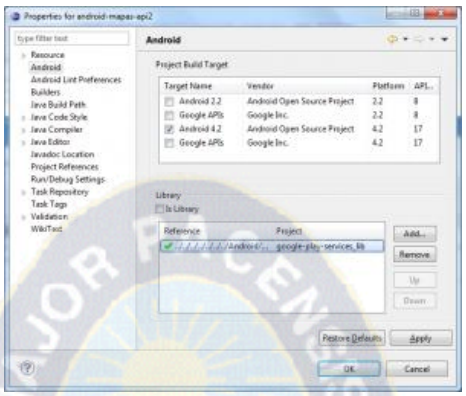


Figura 3.22: Agregar librerias de Google al proyecto

Ahora ya se puede trabajar con un mapa, y lo primero que se debe hacer es el control correspondiente al Layout de nuestra actividad principal. En el caso de la nueva API v2 este "control" se añadirá en forma de fragment , de un tipo de la clase com.google.android.gms.maps.SupportMapFragment, quedando como el Programa 3.7.

Programa 3.7: Fragment donde se muestra el Mapa

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
      class="com.google.android.gms.maps.SupportMapFragment"/>
```

Dado que se usa fragments, la actividad principal también tendrá que extender a FragmentActivity ..

3.5.5.4. Uso del GPS del Smartphone.

Para poder tener acceso a la ubicación del Smartphone en el Programa 3.6, ya fue agregado el permiso ACCESS_LOCATION_EXTRA_COMMANDS, y las líneas en el Programa 3.8 son para acceder a la ubicación usando el GPS del mismo.

Programa 3.8: Acceso a la ubicación del Smartphone

```
mapa.setMyLocationEnabled(true);
LocationManager lm =
(LocationManager)getSystemService(Context.LOCATION_SERVICE)
;
            Location location =
lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            double Olon = location.getLongitude();
            double Olat = location.getLatitude();
```

3.5.5.5. Workshop de post iteración

Mejoras: Se debe mejorar la recepción de datos del controlador para la activación de la alerta

Fortalezas: La aplicación móvil funciona con normalidad. Los datos mostrados están referidos a la ubicación del Automóvil.

Debilidades: Al comienzo de las pruebas con el controlador se recibía datos con un error de casi 300 metros. La obtención de la ubicación del propietario es lenta por el uso del plan de datos, pero con una red WIFI esta se agiliza,

3.5.5.6. Test de Aceptación

Tabla 3.5: Test de Aceptación de Controlador.

HOJA DE PRUEBA DE ACEPTACIÓN	
TEST ID	1
HISTORIA	Funcionamiento del Controlador
FECHA ESCRITA	23/02/2016
FECHA CORRIDA	28/03/2016
PASO/DEFECTO	DEFECTO
DESCRIPCIÓN	1. Acceso a la red de la operadora 2. Habilitar del GPS 3. Habilitar envió y recepción de SMS 4. Ubicar en el mapa el automóvil. 5. Funcionamiento de los sensores de CO2, presencia y de vibración. 6. Activación manual del relé para suministro de energía

RESULTADOS ESPERADO	1. Registro en la red de la operadora 2. GPS en estado 3D Fix 3. Obtención de coordenadas GPS correctos 4. Activación de Led 5. Apagado de bomba de gasolina

Tabla 3.6 Test de Aceptación de la aplicación

HOJA DE PRUEBA DE ACEPTACIÓN	
TEST ID	2
HISTORIA	Conexión Smartphone y Controlador
FECHA ESCRITA	12/03/2016
FECHA CORRIDA	16/03/2016
PASO/DEFECTO	PASO
DESCRIPCIÓN	1. Recibir el mensaje de Conexión con el controlado. 2. Ingresar mediante validación. 3. Obtener la ubicación actual del Propietario. 4. Ubicar en el mapa el automóvil.
RESULTADOS ESPERADO	1. Manejo de la API de Google Maps correcto 2. Conexión constante con el controlador

3.5.6. Estabilización.

En el caso de este trabajo siendo un solo programador para el desarrollo de la aplicación se adapta esta fase para conexión entre la aplicación móvil y la plataforma que se desarrolla simultáneamente. También se realiza la documentación, dado que en anterior fase se van integrando los módulos, aplicando las pruebas respectivas.

3.5.6.1. Workshop de post iteración

En el caso de este trabajo siendo un solo programador para el desarrollo de la aplicación se adapta esta fase para conexión entre la aplicación móvil y la plataforma que se desarrolla simultáneamente. También se realiza la documentación, dado que en anterior fase se van integrando los módulos, aplicando las pruebas respectivas.

Mejoras: Mejora en los estilos de la interfaz gráfica de la aplicación móvil dando una mejor apariencia y presentación al usuario final. Mejora en el controlador para que pueda acceder el GPS a estado 3D Fix.

Fortalezas: La aplicación funciona de manera adecuada contando con una conexión a internet, realiza un rastreo del automóvil en tiempo real y recibe los datos para dar la alerta de movimiento en el automóvil.

Debilidades: La aplicación funciona solamente si se cuenta con una conexión a internet y el controlador y el Smartphone tienen crédito para el envió de SMS

Los resultados de todas las iteraciones y de la estabilización está disponible en el anexo B.

3.5.6.2. Test de Aceptación

Como estamos trabajando con Arduino para el controlador también se aumentara el controlador a la fase de test de aceptación.

Tabla 3.7: Test de Aceptación de Controlador.

HOJA DE PRUEBA DE ACEPTACIÓN	
TEST ID	3
HISTORIA	Correcciones Funcionamiento del Controlador
FECHA ESCRITA	1/04/2016
FECHA CORRIDA	8/04/2016
PASO/DEFECTO	Paso
DESCRIPCIÓN	1. Acceso a la operadora 2. Inicio del modulo GPS y GSM 3. Envió de SMS Automáticos 4. Ubicar en el mapa el automóvil. 5. Calibración de Sensor CO2.
RESULTADOS ESPERADO	1. Inspección Visual 2. Inspección Visual 3. Inspección visual en bandeja de entrada 4. Envió de SMS al activarse un sensor

Tabla 3.8: Test de Aceptación de producción

HOJA DE PRUEBA DE ACEPTACIÓN	
TEST ID	4
HISTORIA	Conexión Smartphone y Controlador
FECHA ESCRITA	9/04/2016
FECHA CORRIDA	21/04/2016
PASO/DEFECTO	DEFECTO

DESCRIPCIÓN	1. Ingresar mediante validación. 2. Trazar ruta entre propietario y automóvil. 3. Realizar seguimiento del automóvil. 4. Envío de SMS para el apagado del Automóvil
RESULTADOS ESPERADO	1. Inspección Visual 2. Inspección Visual 3. Inspección Visual 4. Inspección Visual

3.5.7. Pruebas.

El propósito de esta fase es para ver si el sistema implementa lo que el cliente necesitaba, ya que Mobile-D enfatiza en que se realicen un sin número de pruebas, ya que en esta fase no solo participan los desarrolladores sino el cliente principalmente en las pruebas de aceptación. Dado que durante las anteriores fases se contempla la ejecución de pruebas unitarias y de aceptación, ya la aplicación está lista para pasar por las pruebas de integración siendo la etapa final del desarrollo, dando como resultado el producto final ya probado.

3.5.7.1. Plan de pruebas.

Para la pantalla se prueba lo siguiente:

- Datos válidos
- Valores límite
- Datos inválidos
- El diseño debe ser como esta en la documentación
- Los enlaces entre el dialogo de alerta la pantalla de mapa y la ubicación GPS deben funcionar tal como se las describe en la documentación.

Para las pruebas de tiempo de carga se tomó en cuenta los siguientes criterios:

- Se tomaron en cuenta los sensores instalados en el automóvil.
- Se midió el tiempo de respuesta para la ubicación del GPS del Smartphone, y del controlador.
- Se realizaron las pruebas simulando los datos en la aplicación y controlador repitiendo 217 veces este proceso.

Para las pruebas de acceso se tomó en cuenta los siguientes criterios:

- Se tomó en cuenta el requerimiento funcional más importante para la comparación (ubicación en tiempo real)
- Se considera tiempo de acceso al tiempo desde que un usuario abre una aplicación y recibe totalmente la información deseada.

3.5.7.2. PRUEBAS DE ACEPTACIÓN POR ITERACIÓN

Estas pruebas son realizadas a todos los requerimientos, se ve la facilidad de uso, , se pretende lograr: correcciones especificación completa y clara del problema.

- **RESULTADOS ITERACIÓN 1**

Tabla 3.9: Resultados de la iteración 1

ITERACIÓN 1: CONEXIÓN Y ALERTAS		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	16	80%
Pruebas reprobadas	4	20%
Total	20	100%
Pruebas corregidas	4	100%

- **RESULTADOS ITERACIÓN 2**

Tabla 3.10: Resultados de la iteración 2

ITERACIÓN 2: UBICACIÓN GPS DEL SMARTPHONE Y DEL CONTROLADOR		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	110	91.6%
Pruebas reprobadas	10	8.4%
Total	120	100%
Pruebas corregidas	8	98.3%

- **RESULTADOS ITERACIÓN 3**

Tabla 3.11: Resultados Iteración 3

ITERACIÓN 3 RESPUESTAS AL CONTROLADOR		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	27	90%
Pruebas reprobadas	3	10%
Total	30	100%
Pruebas corregidas	2	97%

- **RESULTADOS ITERACIÓN 4**

Tabla 3.12: Resultados Iteración 4

ITERACIÓN 1: ESTABILIZACIÓN CONEXIÓN Y ALERTAS		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	9	90%
Pruebas reprobadas	1	10%
Total	10	100%
Pruebas corregidas	1	100%

- **RESULTADOS ITERACIÓN 5**

Tabla 3.13: Resultados Iteración 5

ITERACIÓN 2: ESTABILIZACIÓN UBICACIÓN GPS DEL SMARTPHONE Y DEL CONTROLADOR		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	12	100%
Pruebas reprobadas	0	0%
Total	12	12%
Pruebas corregidas	0	

- **RESULTADOS ITERACIÓN 6**

Tabla 3.14 Resultados Iteración 6

ITERACIÓN 3: ESTABILIZACIÓN RESPUESTAS AL CONTROLADOR		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	23	92%
Pruebas reprobadas	2	8%
Total	25	100%
Pruebas corregidas	2	10%

- **RESUMEN DE RESULTADOS DE LAS ITERACIONES**

Tabla 3.15: Resumen de las iteraciones

RESULTADO GENERAL DE ITERACIONES		
	NUMERO DE PRUEBAS	**PORCENTAJE**
Pruebas aceptadas	197	90.8%
Pruebas reprobadas	20	9.2%
Total	217	100%
Pruebas corregidas	3	1.4%

3.5.7.3. Resultados de pruebas externas.

Estas pruebas fueron realizadas a 3 propietarios de automóviles, se realiza una instalación provisional del controlador en sus automóviles, aunque no se habilito el relee para el encendido y apagado del automóvil.

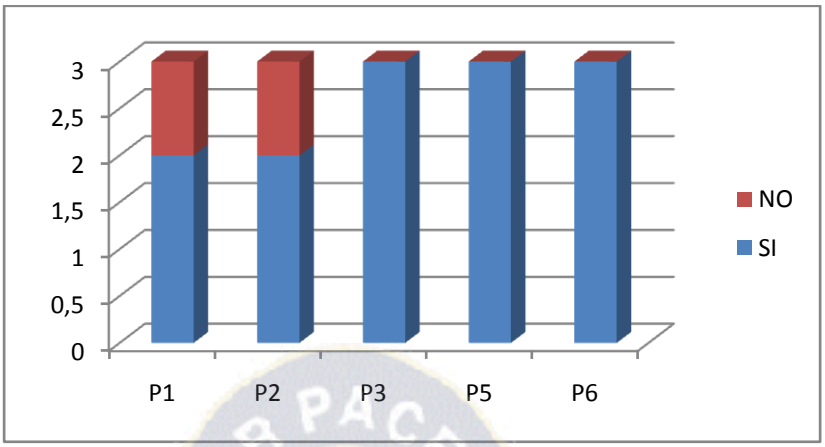
De las cuales se realizo una encuesta la cual se puede observar en el anexo C, que se obtuvo los siguientes resultados los cuales se muestran en la figura 3.23, :

a. La opción de seguimiento funciona correctamente.

b. Solo uno de los propietarios indico tener fallas en la conexión.

c. No hubo alertas falsas.

d. Cuando hay conexión buena entre controlador y aplicación los datos son exactos.



3.23: Resultados de la Encuesta de funcionamiento.
El funcionamiento fue correcto aunque se verifico un error.

Para la pregunta 4 sobre las alarmas falsas se verifico solo dos de ellas aunque no se pudo verificar la situación de las mismas.

A continuación en la tabla 3.16 se observa las líneas telefónicas a las que pertenecen los propietarios, como se aclaro antes solo se pudo realizar la prueba a 3 propietarios, pero adicionando los chips con los que se trabajo para las primeras pruebas se tendría 5 propietarios.

3.16: Tabla de líneas telefónicas.

EMPRESA	PROPIETARIOS
ENTEL	2
VIVA	2
TIGO	1

Se pudo evidenciar que los problemas que se presentaron fue al propietario de que pertenecía a la empresa Entel.

4. PRUEBA DE HIPÓTESIS

A continuación se presenta el desarrollo de las pruebas y el análisis del cumplimiento de los requerimientos, dando lugar a la prueba de la hipótesis planteada y por tanto el cumplimiento de los objetivos planteados.

En resumen una evaluación en los casos de prueba seria:

Tabla 4.1: Tabla de Resultados

	Cantidad	**Porcentaje**
Pruebas Aceptadas	197	90.8%
Pruebas Reprobadas	20	9.8%
Fracasos no arreglados	3	1.4%
Pruebas Totales	217	100%

Este ultimo porcentaje de fracasos no arreglados es a causa que en la fase de producción, por razones que no son comprendidas el GPS daba coordenadas con errores de 100 a 200 metros, pero a medida que se hicieron la pruebas ya no se produjo esa falla además que se cambio de antena de GPS.

Para demostrar la hipótesis se utilizara la distribución Normal o Gaussiana, la cual frecuentemente utilizada en las aplicaciones estadísticas. Su propio nombre indica su extendida utilización, justificada por la frecuencia o normalidad con la que ciertos fenómenos tienden a parecerse en su comportamiento a esta distribución.

Como es notable tenemos un elevado porcentaje de éxito de 90.8%.

La hipótesis planteada es:

"El Diseño de una aplicación para realizar el seguimiento y control de automóviles, que apoyándose en un controlador construido en base a Arduino, ayudará a reducir la inseguridad al momento de dejarlos estacionados en calles o parqueos, y de esta forma evitar el robo de los mismos y de sus autopartes, esta reducción será medida una vez que se tenga el prototipo instalado realizando pruebas de control."

Una Hipótesis es algo que se supone y a lo que se le otorga un cierto grado de posibilidad para extraer de ello un efecto o una consecuencia. Su validez depende del sometimiento a varias pruebas, partiendo de las teorías elaboradas.

Ho: *"El Diseño de una aplicación para realizar el seguimiento y control de automóviles, que apoyándose en un controlador construido en base a Arduino, ayudará a reducir la inseguridad al momento de dejarlos estacionados en calles o parqueos, y de esta forma evitar el robo de los mismos y de sus autopartes, esta reducción será medida una vez que se tenga el prototipo instalado realizando pruebas de control."*

H1: Se rechaza Ho

Ahora para que tenga valides se debe demostrar que el porcentaje de éxitos sea igual o mayor a 90%, tomando un nivel de significancia del 5%, el cual establece el límite de la región de rechazo, por tanto la hipótesis nula en un estudio se rechaza cuando el valor p asociado a la prueba estadística utilizada para contrastar la hipótesis, es inferior al valor alfa establecido por el investigador (valor p < nivel de significancia). De lo que podemos inferir que valores altos de la significancia observada constituyen evidencia a favor de la hipótesis nula, valores "bajos" apoyan la hipótesis alterna.

$H_0 : p_0 \geq 0.9$

$H_1 : p_0 < 0.9$

4.1. Evaluación de resultados

Para determinar si el porcentaje de éxitos obtenido en las pruebas puede ser considerado cercano al 90% de nivel de confianza esperado, se hará uso de una Prueba de Hipótesis para Proporciones.

Las variables usadas en dicha prueba serán las mismas mencionadas en la evaluación de casos de prueba:

$p_0 = 0.9$

X = 197 => número de pruebas aprobadas

n = 217 => número total de pruebas

α = 0.5 => nivel de significancia

4.2. Determinación de la región crítica

La región crítica para la hipótesis planteada, es la siguiente:

Figura 4.1: Región Critica.
Zona de aceptación del prototipo entre -Zo,Zo

Como n se refiere en este caso al número de pruebas, en este caso 197, el punto crítico a usar es $-Z_0$ y se determina mediante:

$$-Z = -Z_{1-\propto} = -Z_{1-0.05}$$

Para obtener el valor de Z se debe recurrir a la Tabla de la función de Distribución Normal, a continuación se elige el valor más cercano a 0.95, el cual está ubicado en la fila 1.6 y columna 0.04 como muestra la Figura 4.2:

t	0.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.4878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

Figura 4.2: Distribución Normal.
Se busca el valor mas cercano y se suman los valores de su fila y columna.

El valor de z se obtiene sumando ambos valores:

$$-Z = -(1{,}6 + 0{,}04) = -1.64$$

4.3. Cálculo estadístico de la prueba

Como se conoce el número total de pruebas el valor estadístico de la prueba se obtiene mediante la fórmula:

$$Z = \frac{X - n * p_0}{\sqrt{n * p_0(1 - p_0)}}$$

Reemplazando los valores y haciendo los cálculos correspondientes obtenemos:

$$Z = \frac{197 - 217 * 0.9}{\sqrt{217 * 0.9(1 - 0.9)}} = 0.38$$

Por lo tanto se tiene:

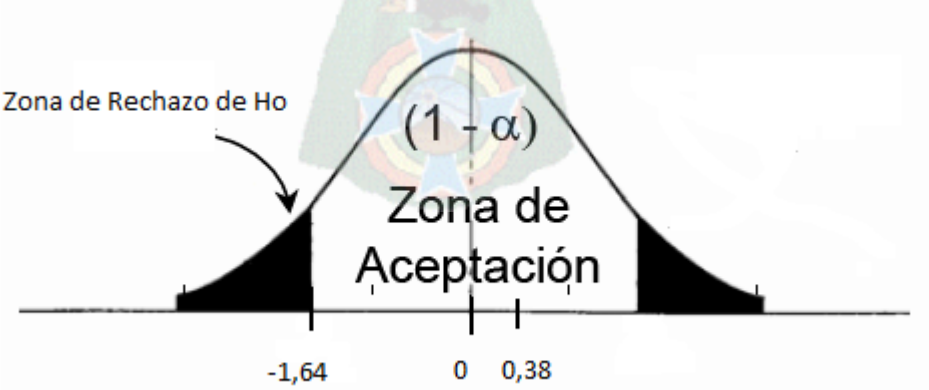


Figura 4.3: Distribución Z para la toma de decisión

El promedio de éxito del prototipo al momento de reconocer las muestras se acerca al 90%.

Por tanto como se acepta H_0 se podría concluir y afinar la hipótesis:

H_0: *"El Diseño de una aplicación para realizar el seguimiento y control de automóviles, que apoyándose en un controlador construido en base a Arduino, ayudará a reducir la inseguridad al momento de dejarlos estacionados en calles o parqueos, y de esta forma evitar el robo de los mismos y de sus autopartes, esta reducción será medida una vez que se tenga el prototipo instalado realizando pruebas de control."*

4.4. Evaluación de resultados externos

La mayoría de los resultados obtenidos fueron exitosos, de los propietarios solo uno presento alguna inconformidad al momento de mandar pedidos desde la aplicación, pero revisando su encuesta se verifico que su línea móvil pertenecía a la empresa telefónica Entel, en la cual los mensajes no son enviados en todo momento de manera instantánea.

Pero dicho error de conexión no era constante, por lo haciendo un intento más sobre la opción escogida la información era correcta.

5. CONCLUSIONES Y RECOMENDACIONES

El objetivo planteado era el siguiente:

"Diseñar una aplicación para realizar el seguimiento y control de automóviles, que con la ayuda de un controlador, ayudará a reducir la inseguridad al momento de dejar dichos automóviles parqueados en calles o parqueos, al igual que evitará el robo de los mismos", planteado en el capítulo 1 se cumplió con la construcción del prototipo y su instalación en un automóvil, que con la ayuda de los distintos sensores y la tecnología GSM y GPS instalados en el controlador Arduino permite obtener un alto grado de seguridad en los automóviles que lo instalen, para evitar robos de los mismos.

Los objetivos específicos fueron cumplidos:

- Se desarrollo una alarma que contenga un GPS para localizar el automóvil.
- Se desarrollo el control a distancia, para cortar la alimentación de gasolina o energía eléctrica.
- Se desarrollo el modulo de envío de alertas al Smartphone para notificar un cambio de estado, ya sea por que se abre una puerta o que haya un movimiento brusco.

Este controlador puede es muy útil a la hora de frustrar un robo dentro del automóvil, pero eso no significa que este robo será evitado por completo por la razón de que el propietario debe estar atento a los mensajes y salir a verificar el estado del mismo cuando llegue una notificación, o por medio de la alerta tomar el recaudo de llamar a radio patrullas para capturar a los delincuentes si la situación lo amerita.

Una parte importante al final de implementar este proyecto de Tesis es evitar la tercerización de la seguridad del Automóvil, se cortara el pago mensual a una empresa y el propietario el momento que desee podrá tener acceso directo a la ubicación de su Automóvil, sin importar hora, día o lugar.

El internet de las cosas va creciendo a grandes pasos por lo que aún hay mucho por conocer para la interconexión no solo de automóviles mediante dispositivos, Arduino es de gran ayuda para realizar esta tarea aunque no es la única opción, pero para armar el controlador, Arduino UNO fue la mejor opción pues soporta muchos sensores mas el Shield GSM/GPS. y no hay mucho desperdicio de pines.

Como en algunos proyectos se ven errores los cuales no pueden ser solucionados, errores que no son a causa de software o hardware, en la tesis se pudo observar que todavía existen algunos puntos en la ciudad de La Paz , que no recibe buena señal de telefonía, esto puede ser debido a la topografía tan accidentada que tiene, pero esto no influiría en el trabajo correcto de la aplicación y el controlador, puesto que son lugares donde no hay parqueos o el flujo de automóviles es sin parada

Armar el prototipo es una parte muy dinámica pero se debe tener cuidado al momento de la elección de los dispositivos a usar, revisar si son los adecuados, debido a que en la presente tesis de grado para el GPS no se verifico el funcionamiento de la correspondiente y hay momento de hacer las pruebas no daba las señales correctas, la antena inicial fue una antena solo para exteriores de baja potencia, y después de mucha lectura en blogs y foros se tuvo que tomar la decisión de buscar y cambiar a una antena de mayor potencia y con la cual se tuvo resultados óptimos.

5.1. El costo del controlador.

Tabla 5.1: Tabla de precios

Nro.	Componente	Cantidad	Precio (bs)
1	Placa Arduino	1	90.00
2	Shield GSM/GPS	1	500.00
3	Antena exterior GPS	1	ITEM 2
4	Antena GSM	1	ITEM 2
5	Sensor de Presencia	1	25.00
6	Sensor de Gas	1	40.00
7	Modulo relé	1	21.00
8	Control RF	1	50.00
9	Sensor de Vibración	1	45.00
10	Batería o PowerBank	1	100.00
		TOTAL	871.00

5.2. Recomendaciones.

Siempre tener cuidado en verificar el crédito del controlador, pues es lógico que sin crédito no se podrá realizar el envió de mensajes de alertas. La aplicación no necesita estar abierta en todo momento corre de en segundo plano.

La batería que alimenta al controlador es independiente y depende mucho del propietario si pone una de mayor capacidad.

La ubicación de los sensores depende mucho del automóvil, en el que se debe tener más cuidado es el relé y hacer un análisis de donde es la mejor ubicación.

5.3. Trabajos futuros.

Para trabajos futuros pueden implementar cámaras de seguridad en el interior, además de aumentar sensores para tratar de igualar a autos de marcas muy importantes que poseen sensores de proximidad para ayuda de estacionamiento.

Como se dijo este trabajo va dirigido a propietarios de automóviles, pero para un trabajo futuro se podría implementar a una empresa para el seguimiento de una flota de automóviles y creando una base de datos más una web service, realizar dicho seguimiento las 24 horas. O también implementar una base de datos para poder consultar el recorrido en una determinada fecha.

Gracias a Arduino se puede realizar muchos propuestas dentro de un auto, y con la salida de nuevos Shield trabajar directamente con la ECU del automóvil, este Shield es el CAN BUS, pero para trabajar con este se debe tener más conocimiento de la redes que existe en un automóvil y el tipo de comunicación que tiene, además que por el momento este Shield tiene un costo muy elevado, pero con el cual se puede anular todos los sensores adicionales , utilizar los datos de la ECU del automóvil y mandar información directo a Arduino o a otro dispositivo para activar o desactivar algún modulo del automóvil, aquí ya se entraría a otra área inmensa que es la Meca trónica.

Bibliografía

Abrahamsson, P. (2007). *Agile software development of mobile information systems», en Proceedings of the 19th international conference on Advanced information systems engineering.*

AGILE. (2012). *Electronics -AGILE - Agile Software Technologies*. Recuperado el 03 de 2016

Alegsa. (2010). Recuperado el 10 de 03 de 2016, de http://www.alegsa.com.ar/Dic/ttl.php

Android. (2015). *Android*. Recuperado el 1 de 2016

Arduino. (2015). Recuperado el 21 de 08 de 2015, de https://www.arduino.cc/en/Main/Products

Arduino. (2015). Recuperado el 21 de 08 de 2015, de https://www.arduino.cc/en/Guide/Introduction

Ashton, K. (2009). *RFID Journal*. Recuperado el 27 de 02 de 2015, de http://www.rfidjournal.com/articles/view?4986

Astudillo, J. P., & Delgado, E. G. (2012). *Joranadas Informaticas Argentinas*. Recuperado el 14 de 11 de 2015, de http://41jaiio.sadio.org.ar/sites/default/files/11_EST_2012.pdf

Avison, D., & Fitzgeral, G. (2006). *Information system development.* Maidenhead: McGraw-Hill Education.

Barragan, H. (2015). *Wiring*. Recuperado el 6 de 12 de 2015

Beck, K., & Andres, C. (2004). *Extreme programming explained : embrace change.* Boston: Addison Wesley.

Beck, K., & Zapata, J. (2002). *Una Explicación de la programación extrema. Aceptar el cambio.* Madrid: Addison Wesly.

Beyond. (2010). *Beyond Logic*. Recuperado el 10 de 03 de 2016, de http://retired.beyondlogic.org/serial/serial.htm#8

BlueHacks. (2016). Recuperado el 2 de 03 de 2016, de http://bluehack.elhacker.net/proyectos/comandosat/comandosat.html

Castañeta, O. J., & Carrion, C. A. (2014). *Sistema de Alarma y monitoreo vehicular controlado por un dispositivo móvil utilizando la conexión de redes celulares.* TESIS, Bogota Colombia.

Cooking-Hacks. (2014). Recuperado el 02 de 03 de 2016, de http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/SIM900_AT_Command_Manual.pdf

Definicion ABC. (2007). Recuperado el 25 de Octubre de 2015, de http://www.definicionabc.com/general/seguimiento.php

Dimas, J. (2014). *DesarrolloWeb.com*. Recuperado el 01 de 2016, de http://www.desarrolloweb.com/articulos/introduccion-android.html

Ekstein. (2015). *Ekstein Komponent*. Recuperado el 21 de 01 de 2016, de http://eckstein-shop.de/XD-48-Vibration-Alarm-Sensor-Modu-lVibrationsschalter-SW-420

Esparza, J. A. (2013). *Repositorio Digital Universidad Tecnica del Norte.* Recuperado el 25 de 10 de 2015, de http://repositorio.utn.edu.ec/handle/123456789/2348?mode=full

Evans, D. (2011). *CISCO IBSG.* Recuperado el 25 de Octubre de 2015, de http://www.cisco.com/web/about/ac79/docs/pov/Planetary_Skin_POV_vFINAL_spw_jc_2.pdf

Fernandez, L. (2011). *Euskadinnova.* Recuperado el 25 de Octubre de 2015, de http://www.euskadinnova.net/es/innovacion-social/noticias/internet-cosas-inteligencia-objetos-cotidianos/7735.aspx

Fry, B., & Reas, C. (2015). *Processing*. Recuperado el 05 de 12 de 2015, de https://www.processing.org/

GeekFactory. (2015). Recuperado el 21 de 01 de 2016, de http://www.geekfactory.mx/categoria-de-producto/shields-arduino/

Gershenfeld, N., Krikorian, R., & Cohen, D. (2004). The Internet Of Things. *Scientific American* .

GPS. (2016). *GPS.gov*. Recuperado el 3 de 3 de 2016, de http://www.gps.gov/spanish.php

Graf, R. (1984). *Diccionario de Electronica.* Piramide.

HetPro. (2015). *HetPro Herramiental tecnologicas profesionales*. Recuperado el 22 de 01 de 2016, de https://hetpro-store.com/modulo-de-vibracion-sw-420-con-comparador/

Hipertextual. (2014). Recuperado el 25 de 10 de 2015, de www.hipertextual.com

Huertas, J. D. (2014). *La Referencia visibilizando la Ciencia.* Recuperado el 25 de Octubre de 2015, de http://repositorio.espe.edu.ec/handle/21000/9256

IBM. (2001). *Rational Unified Process Best Practices for Software Development Teams*. Recuperado el 10 de 04 de 2016, de www.rational.com

Jirones, J. T. (2012). *El gran libro de Android.* MExico: Alfaomega Grupo Editor.

Joskowicz, J. (2008). Reglas y Practicas de eXtreme Programing.

Krutchen, P. (2001). *The Rational Unified Process An Introduction.* Addison Wesley.

Kurzweil. (2011). Recuperado el 25 de 02 de 2016, de http://www.kurzweilai.net/ibm-open-sources-internet-of-things-protocol

Lammert. (2016). *Lammert Bies*. Recuperado el 12 de 03 de 2016, de http://www.lammertbies.nl/comm/info/hayes-at-commands.html

Lexicoon. (2015). Recuperado el 1 de 11 de 2015, de http://lexicoon.org/es/parquear

Llamas, L. (2015). *Luis LLamas. Ingenieria , Informatica Diseño*. Recuperado el 29 de 01 de 2016, de http://www.luisllamas.es/2015/07/medir-vibracion-con-arduino-y-sensor-sw-18020p/

Marquez, A. (2008). *Repositorio Biblioteca Universidad Simon Bolivar*. Recuperado el 14 de 11 de 2015, de http://159.90.80.55/tesis/000138576.pdf

Microtronic. (2015). Recuperado el 02 de 03 de 2016, de http://www.utronic.net/index.php?route=product/product&product_id=124

Mobile-D. (2016). *Mobile D HOME PAGE*. Recuperado el 04 de 2016, de http://agile.vtt.fi/mobiled.html

Newman, J. (2013). *University of Florida IFAS Extensions*. Recuperado el 2016

Nosseir, A., Flood, ,. D., Harrison, R., & Ibrahim, O. (2012). *Mobile Development Process Spiral.*

PuntoFlotante. (2015). *PuntoFlotante s.a.* Recuperado el 31 de 01 de 2016, de http://www.puntoflotante.net/TUTORIAL-MODEM-GSM-GPRS.htm

RAE. (2014). *Real Academia Española*. Recuperado el 25 de Octubre de 2015, de http://lema.rae.es/drae/srv/search?key=control

Rahimian, V., & Ramsin, R. (2008). *Designing an agile methodology for mobile software development: A hybrid method engineering approach* . RCIS.

Ramirez, G. (2001). *docs.google*. Recuperado el 15 de 04 de 2016, de https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxsdWlzb3Rha2V8Z3g6NjVkZWE3ZjMwMTE0NTBmZQ

Rey, J. (2015). *EDIS*. Recuperado el 03 de 03 de 2016, de https://edis.ifas.ufl.edu/in657

Sarma, S., Brock, D., & Ashton, K. (2000). The Networked Physical World. Proposals for Engineering the Next Generation of Computing, Commerce & Automatic-Identification. *WHITE PAPER* , 16.

Shield. (2015). *Arduino Shield List*. Recuperado el 2015

*simCom. (2016). *www.simcom.eu*. Recuperado el 1 de 3 de 2016, de http://www.simcom.eu/index.php?m=termekek&prime=1&sub=41&id=0000000228

Sparkfun. (2015). Recuperado el 15 de 12 de 2015, de www.sparkfun.com/products/9403

Valverde, S. A. (2014). *Escuela de Ingenieria Electronica De Costa Rica*. Recuperado el 14 de 11 de 2015, de http://eie.ucr.ac.cr/uploads/file/proybach/pb0877t.pdf

Vega, J. I., Sanches, R., Salgado, G., & Sanchez, L. A. (s.f.). *Universidad Autonoma Metropolitana de Mexico.* Recuperado el 28 de 10 de 2015, de http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-2.htm

Vliet, H. v. (2007). *Software Engineering: Principles and Practice.* Wiley.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American* .

WTZ. (2016). *WorldTimeZone*. Recuperado el 2016, de http://www.worldtimezone.com/

ANEXOS

ANEXO A

CODIGO DE ARDUINO

```
//Introducir datos
const char pin_number[]="0000";         //Numero PIN de SIM Card
const char phone_number[] = "60677582";//Numero de celular de la aplicación
char sms_text[]="Connected to the network!!";
char sms_MQ7[]="Alerta su Auto fue encendido!!!!!!";
char smsXX[100];
char coma[]=",";
char sms_PIR[]="Alerta hay alguin en su auto!!!!!!";
int SMS=0; //numero de mesaje
int onModulePin = 2;
char aux_string[30];
char aux1_string[50];
unsigned long previous;
long previousMillis = 0;
int count;
int8_t answer;
int flag = 0;
int angle;
long tiempoESPERA = 60000;
int cont=0;
char frame[200];
char latitude[15];
char longitude[15];
char altitude[6];
char date[16];
char time[7];
char satellites[3];
char speedOTG[10];
char course[10];
// SENSOR MQ7
const int DOUTpin=6;
//const int DOUTpin=3;
const int ledPin=4;
int limit;
//END MQ7
int dato1=0;
int i = 0;
int swM=0;
// SENSOR PIR
const int LEDPin= 4;
const int PIRPin= 5;
int swP=0;
// END SENSOR PIR
unsigned long currentMillis;
boolean abierto = false;
int cambios;
int swD=0;
// Rutina de Interrupción
void interrupcion(){
  if (digitalRead(12)==HIGH){
      abierto = true;
      if (cambios == 0){
        cambios ++;
```

```
      }
      else {
        cambios = 0;
        abierto = false;
      }
  }
}
void setup(){
    pinMode(onModulePin, OUTPUT);
    pinMode(LEDPin, OUTPUT);
    pinMode(PIRPin, INPUT);
    Serial.begin (115200);
    pinMode (13, OUTPUT) ;
    pinMode (12, OUTPUT) ;
    for (int i = 7 ; i < 12 ; i++)
          pinMode(i, INPUT) ;
    Serial.println("Iniciando ...");
    power_on();             // Encendido del Shield SIM908
    delay(5000);
    //verificación código PIN
    sprintf(aux_string, "AT+CPIN=%s", pin_number);
    sendATcommand(aux_string, "OK", 2000);
    delay(2000);
    Serial.println("Conectando a la red...!!!");
    //REGISTRO DE RED
    while ( (sendATcommand("AT+CREG?", "+CREG: 0,1", 1000) ||
             sendATcommand("AT+CREG?", "+CREG: 0,5", 1000)) == 0 );
    sendATcommand("AT+CLIP=1", "OK", 1000);
    //encendido y verificacion de GPS
    while ( start_GPS() == 0);
    Serial.println("Conectado a la Red...!!!");
    while (Serial.available() != 0){
        Serial.read();
    }
    //VERIFICACION DE CONEXION
    sendSMS(sms_text);
    sendATcommand("AT+CNMI=2,1,0,0", "OK", 1000);// Modo espera SMS
    Serial.println("Waiting incoming SMS");
    digitalWrite(3, HIGH);
    attachInterrupt(1,interrupcion,HIGH);
}

void loop()
{
   answer = sendATcommand("", "+CMTI: ", 1000);
   //Si hay un mensaje entrante SMS
   if (answer == 1){
    count = 0;
    previous = millis();
    do{
      while (((millis() - previous) < 1000) && (Serial.available() == 0)){
        // Condición para evitar un desbordamiento
        if ( millis() < previous) previous = millis();
      }
      // Condición para evitar un desbordamiento
      if ( millis() < previous) previous = millis();
```

```
    }
    while (((millis() - previous) < 1000) && (Serial.read() != ','));
    previous = millis();
    do {
        while (((millis() - previous) < 1000) && (Serial.available() == 0))
        {
          // Condición para evitar un desbordamiento
          if ( millis() < previous)
                  previous = millis();
        }
        aux1_string[count] = Serial.read();
        count++;
        // Condición para evitar un desbordamiento
        if ( millis() < previous) previous = millis();
    }while((aux1_string[count-1]!='\r')&&((millis()-previous)<1000)&&
(count<30));
    aux1_string[count - 1] = '\0';
    count = atoi(aux1_string);
    Serial.print(F("SMS index: "));
    Serial.println(count, DEC);
    Serial.println(aux1_string);
    if (count != 0){
      //Verifica la conexión con el módulo
      sendATcommand("AT", "OK", 1000);
      //Leemos el SMS recibido
      SMS=count;
      snprintf(aux1_string, sizeof(aux1_string), "AT+CMGR=%d", count);
      answer = sendATcommand(aux1_string, "+CMGR:", 1000);
      if (answer == 1){
        previous = millis();
        do
        {
          while(((millis()-previous)<1000)&&(Serial.available()==0)){
            // Condición para evitar un desbordamiento
            if ( millis() < previous) previous = millis();
          }
          // Condición para evitar un desbordamiento
          if ( millis() < previous)
                  previous = millis();
        } while(((millis()-previous)<1000)&&(Serial.read()!='\n'));
        count = 0;
        previous = millis();
        do {
          while(((millis()-previous)<1000)&&(Serial.available()==0))
          {
            // Condición para evitar un desbordamiento
            if ( millis() < previous)
                  previous = millis();
          }
          aux1_string[count] = Serial.read();
          count++;
          // Condición para evitar un desbordamiento
          if ( millis() < previous) previous = millis();
        } while ((strstr(aux1_string, "OK") == NULL) && ((millis() -
previous) < 1000) && (count < 30));
        aux1_string[count - 1] = '\0';
```

```
        Serial.print("SMS recibido: ");
        Serial.println(aux1_string);
        angle = atoi(aux1_string);
        switch(angle)
        {
           case 50:
                get_GPS();
                delay(500);
                dato1=1;
                sendSMS(dato1);
                break;
           case 100:
                Serial.println("Seguimiento..");
                currentMillis = millis();
                cont=0;
                while((porcentaje_gas(lecturaMQ(MQ1)/Ro,GAS_LP))!=0){
                    currentMillis = millis();
                    if(currentMillis - previousMillis > (tiempoESPERA*2)) {
                          previousMillis = currentMillis;
                           get_GPS();
                           sendSMS(2);
                          cont++;
                    }
                }
                get_GPS();
                sendSMS(3);
                break;
           case 150:
                Serial.print( "APAGAMOS EL MOTOR....!!");
                digitalWrite (13, ! digitalRead(13)) ;
                if(digitalRead(13)==HIGH)
                    sendSMS("AUTO BLOQUEADO");
                else
                    sendSMS("AUTO SIN BLOQUEO");
                break;
            case 200:
                Serial.print( "ALARMA");
                digitalWrite (12, ! digitalRead(12)) ;
                if(digitalRead(12)==HIGH)
                    sendSMS("ALARMA ON");
                else
                    sendSMS("ALARMA OFF");
                break;
        }
      }
    sendATcommand("AT+CMGD=1", "OK", 1000);
    Serial.flush();
    }
}
//end recibir sms
if (digitalRead(12)==HIGH){
  //sensor PIR
  int value= digitalRead(PIRPin);
   if (value == HIGH){
    digitalWrite(LEDPin, HIGH);
    delay(50);
```

```
      digitalWrite(LEDPin, LOW);
      delay(50);
      if(swP==0){
          Serial.println("Mandar SMS");
          sendSMS("Hay alguien en su auto!!!");
      }
      swP=1;
    }
    else{
      digitalWrite(LEDPin, LOW);
      swP=0;
    }
   //fin sensor PIR
   //sensor MQ7
   limit= digitalRead(DOUTpin);//lee el valor limite del sensor de Gas
   if (limit == LOW){
        digitalWrite(ledPin,HIGH);//si se ha alcanzado el límite, se
enciende el LED como indicador de estado
        if(swM==0){
            Serial.println("Mandar SMS_MQ7");
            sendSMS(sms_MQ7);
        }
        swM=1;
    }
    else{
        digitalWrite(ledPin, LOW);//si no se alcanza el umbral, el LED
permanece apagado
        swM=0;
    }
   //fin sensor MQ7
   //magnetico
   if (abierto){
     if(swD==0){
        Serial.println("Abierto");
        sendSMS("AUTO ABIERTO");
     }
     swD=1;
   }
   else
        swD=0;
   //fin magnetico
   //Serial.println( "todo tranquis");
  }
  if (digitalRead(7) ){        //si hay un dato valido
        Serial.println("Valid trans. \t");
        if (digitalRead(8)){
            Serial.println( "Boton A, pulsado");
            digitalWrite (12, ! digitalRead(12)) ;
            if(digitalRead(12)==HIGH)
                 sendSMS("ALARMA ON");
            else
                 sendSMS("ALARMA OFF");
        }
        if (digitalRead(9))
            Serial.println( "Boton B, pulsado");
              /*get_GPS();
```

```
                delay(300);
                dato1=2;
                sendSMS(dato1);*/
          if (digitalRead(10)){
                Serial.println( "Boton C, pulsado");
                get_GPS();
                delay(500);
                get_GPS();
                delay(500);
                dato1=1;
                sendSMS(dato1);
          }
          if (digitalRead(11)){
                Serial.println( "Boton D, pulsado");
                digitalWrite (13, ! digitalRead(13)) ;
                if(digitalRead(13)==HIGH)
                      sendSMS("AUTO BLOQUEADO");
                else
                      sendSMS("AUTO SIN BLOQUEO");
                delay (300) ;
          }
          Serial.println("\t");
        }
        Serial.flush();
}
/**********************************************************************
 ****                Definicion de funciones                       ****
 **********************************************************************/
 void power_on() {
  uint8_t answer = 0;
  // revisamos si el modulo esta encendido
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0){
    //encendido
    digitalWrite(onModulePin, HIGH);
    delay(3000);
    digitalWrite(onModulePin, LOW);
    while (answer == 0){   // Enviar cada dos segundos  el comando AT y
esperar la respuesta
      answer = sendATcommand("AT", "OK", 2000);
    }
  }

}


int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int
timeout) {
    uint8_t x = 0,  answer = 0;
    char response[100];
    unsigned long previous;
    memset(response, '\0', 100);    // Inicializamos el string
    delay(100);
    while ( Serial.available() > 0)
        Serial.read();              // Limpiamos el bufer de entrada
    if (ATcommand[0] != '\0'){
```

```
        Serial.println(ATcommand);    // Envio comandos AT
    }
    x = 0;
    previous = millis();
    // en este loop esperamos la respuesta
    do {
        if (Serial.available()!=0){  //si hay datos en la memoria
intermedia de entrada UART, lo lee y comprueba si la respuesta
          response[x] = Serial.read();
          //Serial.print(response[x]);
          x++;
          if (strstr(response, expected_answer) != NULL)    // comprobar si
la respuesta deseada (OK) es en la respuesta del módulo
          {
              answer = 1;
          }
        }
    } while ((answer == 0) && ((millis() - previous) < timeout));    //
Espera a la respuesta con tiempo de espera
    return answer;
}

int8_t start_GPS() {
    unsigned long previous;
    previous = millis();
    // encendemos el  GPS
    sendATcommand("AT+CGPSPWR=0", "OK", 2000);
    sendATcommand("AT+CGPSPWR=1", "OK", 2000);
    sendATcommand("AT+CGPSRST=1", "OK", 2000);
    // esperamos la respuesta de GPS
    while(((sendATcommand("AT+CGPSSTATUS?","3D Fix",
5000)||sendATcommand("AT+CGPSSTATUS?","3D Fix",5000))==0)&&
          ((millis()-previous)<90000));
    if ((millis() - previous) < 90000){
        return 1;
    }
     else{
        return 0;
    }
}

int8_t get_GPS(){
    int8_t counter, answer;
    long previous;
    // En primer lugar obtener la cadena NMEA
    // Limpiar el buffer de entrada
    while ( Serial.available() > 0)
            Serial.read();     // Solicitamos String basico
    sendATcommand("AT+CGPSINF=0", "AT+CGPSINF=0\r\n\r\n", 2000);
    counter = 0;
    answer = 0;
    memset(frame, '\0', 100);    // Inicializamos el string
    previous = millis();
    // este bucle espera a la cadena
    do {
      if (Serial.available() != 0){
```

```
            frame[counter] = Serial.read();
            counter++;
            // comprobar si la respuesta deseada está en la respuesta del
módulo
            if (strstr(frame, "OK") != NULL){
                answer = 1;
            }
        }
        // Espera la respuesta
    } while ((answer == 0) && ((millis() - previous) < 2000));
    frame[counter - 3] = '\0';
    // Dividimos la cadena
    strtok(frame, ",");
    strcpy(longitude, strtok(NULL, ",")); // Longitud
    strcpy(latitude, strtok(NULL, ",")); // Latitud
    strcpy(altitude, strtok(NULL, ".")); // Altitud
    strtok(NULL, ",");
    strcpy(date, strtok(NULL, ".")); // Fecha
    strtok(NULL, ",");
    strtok(NULL, ",");
    strcpy(satellites, strtok(NULL, ",")); // Obtiene los satelites
    strcpy(speedOTG, strtok(NULL, ",")); // Obtiene velocidad sobre el
suelo. La unidad está nudos.
    strcpy(course, strtok(NULL, "\r")); // Curso
    convert2Degrees(latitude);
    convert2Degrees(longitude);
    return answer;
}

void sendSMS(int x) {
    Serial.print("Setting SMS mode...");
    sendATcommand("AT+CMGF=1","OK",1000);    // entra al modo texto del SMS
    Serial.println("Sending SMS");
    sprintf(aux_string, "AT+CMGS=\"%s\"", phone_number);
    Serial.print("Help me! Hay movimiento. Estoy en:");
    answer = sendATcommand(aux_string, ">", 2000);    // Envia el SMS
    if (answer == 1){
        if(x==1)
            Serial.print("1,");
        if(x==2)
            Serial.print("2,");
        if(x==3)
            Serial.print("3,");
        i = 0;
        while (latitude[i] != 0) {
            Serial.print(latitude[i]);
            i++;
        }
        Serial.print(",");
        i = 0;
        while (longitude[i] != 0) {
            Serial.print(longitude[i]);
            i++;
        }
        Serial.write(0x1A);
        answer = sendATcommand("", "OK", 20000);
```

```
        if (answer == 1){
            Serial.println("Sent ");
        }
        else{
            Serial.print("error ");
        }
    }
    else{
        Serial.print("error ");
        Serial.println(answer, DEC);
    }
}

void sendSMS(char sms[]) {
    Serial.print("Setting SMS mode...");
    sendATcommand("AT+CMGF=1","OK",1000);    // entra al modo texto del SMS
    Serial.println("Sending SMS");

    sprintf(aux_string, "AT+CMGS=\"%s\"", phone_number);
    answer = sendATcommand(aux_string, ">", 2000);    // Envia el SMS
    if (answer == 1){
        Serial.print("4,");
        Serial.println(sms);
        Serial.write(0x1A);
        answer = sendATcommand("", "OK", 20000);
        if (answer == 1){
            Serial.print("Enviado ");
        }
        else{
            Serial.print("error ");
        }
    }
    else{
        Serial.print("error ");
        Serial.println(answer, DEC);
    }

}

int8_t convert2Degrees(char* input) {
    float deg;
    float minutes;
    boolean neg = false;
   //variable auxiliar
    char aux[10];
    char aux2[10];
    if (input[0] == '-'){
        neg = true;
        strcpy(aux, strtok(input + 1, "."));
        strcpy(aux2, strtok(NULL, "\0"));
    }
    else{
        strcpy(aux, strtok(input, "."));
    }
    // convertir un string a int y sumar a la variable de floot definitiva
    deg = atof(aux);
```

```
    strcpy(aux, strtok(NULL, '\0'));
    minutes = atof(aux2);
    minutes /= 1000000;
    if (deg < 100){
        minutes += deg;
        deg = 0;
    }
    else{
        minutes += int(deg) % 100;
        deg = int(deg) / 100;
    }
    // aumento minutos a grados
    deg = deg + minutes / 60;
    if (neg == true){
        deg *= -1.0;
    }
    neg = false;
    if ( deg < 0 ) {
        neg = true;
        deg *= -1;
    }
    float numeroFloat = deg;
    int parteEntera[10];
    int cifra;
    long numero = (long)numeroFloat;
    int size = 0;
    while (1) {
        size = size + 1;
        cifra = numero % 10;
        numero = numero / 10;
        parteEntera[size - 1] = cifra;
        if (numero == 0) {
          break;
        }
    }
    int indice = 0;
    if ( neg ){
        indice++;
        input[0] = '-';
    }
    for (int i = size - 1; i >= 0; i--){
        input[indice] = parteEntera[i] + '0';
        indice++;
    }
    input[indice] = '.';
    indice++;
    numeroFloat = (numeroFloat - (int)numeroFloat);
    for (int i = 1; i <= 7 ; i++){
        numeroFloat = numeroFloat * 10;
        cifra = (long)numeroFloat;
        numeroFloat = numeroFloat - cifra;
        input[indice] = char(cifra) + 48;
        indice++;
    }
    input[indice] = '\0';
}
```

ANEXO B

CODIGO DE ECLIPSE

```
//Clase SMSReciever
public void onReceive(Context context, Intent intent) {
            Log.d("SMS","SMSReceiver.onReceive");
            Bundle bundle = intent.getExtras();
            // obtenemos el array de estructuras pdus
            Object messages[] = (Object[]) bundle.get("pdus");
            SmsMessage smsMessage[] = new SmsMessage[messages.length];
            for (int n = 0; n < messages.length; n++) {
            // generamos mensajes sms a partir de las estructuras pdu
                  smsMessage[n]=SmsMessage.createFromPdu((byte[])
messages[n]);
            }
//dialogo inicio
      if(smsMessage[0].getDisplayOriginatingAddress().equals(Arduino)&&
!smsMessage[0].getMessageBody().equals("4,Red Conectada!!")){
                  try {
                        Intent i=new Intent(context,
AlertDialogActivity.class);
                        String LatLong=smsMessage[0].getMessageBody()
                        i.putExtra("LatLong",LatLong);
                        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        context.startActivity(i);
                  } catch ( ActivityNotFoundException e) {
                        e.printStackTrace();
                  }
            }
            else{}

      }
}

//Main Activity
public class MainActivity extends android.support.v4.app.FragmentActivity{
      private SQLiteDatabase db;
      private SqliteHelper sqlHelper;
      private String[] titulos;
    private DrawerLayout NavDrawerLayout;
    private ListView NavList;
    private ArrayList<Item_objct> NavItms;
    private TypedArray NavIcons;
    NavigationAdapter NavAdapter;
    private CharSequence mDrawerTitle;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mTitle;
    //mapas
    private LatLng latlng;
      String LatLon;
      String LatiLong;
      double latitude;
      double longitude;
      String[] latlong;
      String Arduino="60563148";
```

```
        GoogleMap mapa;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.main);

                    sqlHelper = new SqliteHelper(this);
                 try {
                    sqlHelper.createDataBase();
                 } catch (IOException e) {
                            e.printStackTrace();
                 }
                mapa = ((SupportMapFragment) getSupportFragmentManager()
.findFragmentById(R.id.map)).getMap();
                //Drawer Layout
                NavDrawerLayout=(DrawerLayout) findViewById(R.id.drawer_layout)
                 NavList = (ListView) findViewById(R.id.lista);
                 View header = getLayoutInflater().inflate(R.layout.header, null);
                 NavList.addHeaderView(header);
                 NavIcons=getResources().obtainTypedArray(R.array.navigation_iconos);
                 titulos = getResources().getStringArray(R.array.nav_options);
                 NavItms = new ArrayList<Item_objct>();
                 NavItms.add(new Item_objct("HOME", NavIcons.getResourceId(0, -1)));
                 NavItms.add(new Item_objct("ON/OFF", NavIcons.getResourceId(1, -1)));
                 NavItms.add(new Item_objct("MI AUTO",NavIcons.getResourceId(4,-1)));
                 NavItms.add(new Item_objct("SEGUIMIENTO",NavIcons.getResourceId(2,-
1)));
                  NavItms.add(new Item_objct("RUTA", NavIcons.getResourceId(5, -1)));
                  NavItms.add(new Item_objct("ALARMA", NavIcons.getResourceId(5, -
1)));
                  NavItms.add(new Item_objct("SALIR", NavIcons.getResourceId(6, -1)));
                  //salir
                  NavAdapter= new NavigationAdapter(this,NavItms);
                  NavList.setAdapter(NavAdapter);
                  //Siempre vamos a mostrar el mismo titulo
                  mTitle = mDrawerTitle = getTitle();
                  //Declaramos el mDrawerToggle y las imgs a utilizar
                  mDrawerToggle = new ActionBarDrawerToggle(
                          this,                    //host Activity
                          NavDrawerLayout,         //DrawerLayout object
                          R.drawable.ic_drawer,    //Icono de navegacion
                          R.string.app_name,       //"open drawer" description
                          R.string.hello_world     //"close drawer" description
                          ) {
                 // Called when a drawer has settled in a completely closed state.
                      public void onDrawerClosed(View view) {
                       Log.e("Cerrado completo", "!!");
                      }
                 //Called when a drawer has settled in a completely open state.
                      public void onDrawerOpened(View drawerView) {
                          Log.e("Apertura completa", "!!");
                      }
                  };
```

```
              NavDrawerLayout.setDrawerListener(mDrawerToggle);
              getActionBar().setDisplayHomeAsUpEnabled(true);
              NavList.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int position,
long id) {
             MostrarFragment(position);
            }
              });
             MostrarFragment(1);
             //sms mapas
            mapa = ((SupportMapFragment) getSupportFragmentManager()
                       .findFragmentById(R.id.map)).getMap();
                   mapa.setMyLocationEnabled(true);
                   Bundle bundle = getIntent().getExtras();
                   if(bundle!=null){
                         switch(bundle.getInt("Opcion"))
                       {
                       case 1:
                          String auxLat=bundle.getString("Lat");
                                String auxLon=bundle.getString("Long");
                                latitude=Double.valueOf(auxLat).doubleValue();
                                longitude=Double.valueOf(auxLon).doubleValue();
                                latlng = new LatLng(latitude,longitude);
                                CameraUpdate cameraUpdate =
CameraUpdateFactory.newLatLngZoom(latlng, 17);
                                mapa.animateCamera(cameraUpdate);
                                mapa.addMarker(new MarkerOptions()
                            .position(latlng)
                            .title("Mi Auto..."));

                           break;
                       case 2:
                          db = sqlHelper.getWritableDatabase();
                    String sql = "INSERT INTO Puntos (latitud, longitud, fecha,
opcion, estado) VALUES ('" + bundle.getString("Lat") + "', '" +
bundle.getString("Long") + "', '22/05/2016', 2, 'ACTIVO') ";
                    db.execSQL(sql);
                    MostrarRuta();
                          break;
                       case 3:
                          db = sqlHelper.getWritableDatabase();
                          String sql1 = "INSERT INTO Puntos (latitud, longitud,
fecha, opcion, estado) VALUES ('" + bundle.getString("Lat") + "', '" +
bundle.getString("Long") + "', '22/05/2016', 2, 'ACTIVO') ";
                          db.execSQL(sql1);
                          MostrarRuta();
                    String sql2 = "UPDATE Puntos SET estado = 'INACTIVO'  WHERE
opcion = 2 AND estado = 'ACTIVO'";
                    db.execSQL(sql2);
                          break;
                       case 4:
                          double Olon =-16.504175;
```

```
                                double Olat = -68.131081;
                                latlng = new LatLng(Olon,Olat);
                                CameraUpdate camerU =
CameraUpdateFactory.newLatLngZoom(latlng, 17);
                                mapa.animateCamera(camerU);
                            break;
                        }
                    }
                    else
                    {
                    LocationManager lm=
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
                            Location location =
lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            //Obtenemos la última posición conocida
                            double Olon =location.getLongitude();
                            double Olat =location.getLatitude();
                            latlng = new LatLng(Olon,Olat);
                            CameraUpdate cameraUpdate =
CameraUpdateFactory.newLatLngZoom(latlng, 17);
                            mapa.animateCamera(cameraUpdate);
                    }
      }
      /*Pasando la posicion de la opcion en el menu nos mostrara el Fragment
correspondiente*/
    private void MostrarFragment(int position) {
        Fragment fragment = null;
        switch (position) {
        case 1:
             mapa.clear();
             NavDrawerLayout.closeDrawer(NavList);
            break;
        case 2:
             Toast.makeText(getApplicationContext(),"ON/OFF"+"..!!!!",
Toast.LENGTH_SHORT).show();
             sendSMS(Arduino, "150");
             NavDrawerLayout.closeDrawer(NavList);
            break;
        case 3:
             Toast.makeText(getApplicationContext(),"UBICACION"+"..!!!!",
Toast.LENGTH_SHORT).show();
             sendSMS(Arduino, "50");
             NavDrawerLayout.closeDrawer(NavList);
            break;
        case 4:
             Toast.makeText(getApplicationContext(),"SEGUIMIENTO"+"..!!!!",
Toast.LENGTH_SHORT).show();
             sendSMS(Arduino, "100");
             NavDrawerLayout.closeDrawer(NavList);
            break;
        case 5:
             Toast.makeText(getApplicationContext(),"TRAZANDO RUTA "+"
DISPONIBLE...!!!!", Toast.LENGTH_SHORT).show();
             Truta(latitude,longitude);
```

```
            NavDrawerLayout.closeDrawer(NavList);
            break;
        case 6:
            Toast.makeText(getApplicationContext(),"ON/OFF"+"..!!!!",
Toast.LENGTH_SHORT).show();
            sendSMS(Arduino, "200");
            NavDrawerLayout.closeDrawer(NavList);
            break;
        case 7:
            finish();
            break;
        default:
            //si no esta la opcion mostrara un toast y nos mandara a Home
            Toast.makeText(getApplicationContext(),"Opcion "+titulos[position-
1]+"no disponible!", Toast.LENGTH_SHORT).show();
            NavDrawerLayout.closeDrawer(NavList);
            position=1;
            break;
        }
        //Validamos si el fragment no es nulo
        if (fragment != null) {
            FragmentManager fragmentManager = getFragmentManager();
            fragmentManager.beginTransaction().replace(R.id.content_frame,
fragment).commit();
            // Actualizamos el contenido segun la opcion elegida
            NavList.setItemChecked(position, true);
            NavList.setSelection(position);
            //Cambiamos el titulo en donde decia "
            setTitle(titulos[position-1]);
            //Cerramos el menu deslizable
            NavDrawerLayout.closeDrawer(NavList);
        } else {
            //Si el fragment es nulo mostramos un mensaje de error.
            Log.e("Error  ", "MostrarFragment"+position);
        }
    }
      private void sendSMS(String phoneNumber, String message){
            SmsManager sms=SmsManager.getDefault();
            sms.sendTextMessage(phoneNumber,null, message,null, null);
      }
      private void Truta(double Dlat,double Dlon){
            LocationManager lm =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
            Location location =
lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            double Olon = location.getLongitude();
            double Olat = location.getLatitude();
            LatLng origin =new LatLng(Olat,Olon);
            LatLng dest = new LatLng(Dlat,Dlon);
            // Getting URL to the Google Directions API
            String url = getDirectionsUrl(origin, dest);
            DownloadTask downloadTask = new DownloadTask();
            // Start downloading json data from Google Directions API
            downloadTask.execute(url);
```

```
	}
	private String getDirectionsUrl(LatLng origin,LatLng dest){
		// Origin of route
		String str_origin = "origin="+origin.latitude+","+origin.longitude;
		// Destination of route
		String str_dest = "destination="+dest.latitude+","+dest.longitude;
		// Sensor enabled
		String sensor = "sensor=false";
		// Building the parameters to the web service
		String parameters = str_origin+"&"+str_dest+"&"+sensor;
		// Output format
		String output = "json";
		// Building the url to the web service
		String url =
"https://maps.googleapis.com/maps/api/directions/"+output+"?"+parameters;
	return url;
	}
	/** A method to download json data from url */
    private String downloadUrl(String strUrl) throws IOException{
	  String data = "";
	  InputStream iStream = null;
	  HttpURLConnection urlConnection = null;
	  try{
			URL url = new URL(strUrl);
			// Creating an http connection to communicate with url
			urlConnection = (HttpURLConnection) url.openConnection();
			// Connecting to url
			urlConnection.connect();
			// Reading data from url
			iStream = urlConnection.getInputStream();
			BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));
			StringBuffer sb  = new StringBuffer();
			String line = "";
			while( ( line = br.readLine())  != null){
				sb.append(line);
			}
			data = sb.toString();
			br.close();
	  }catch(Exception e){
			Log.d("Exception while downloading url", e.toString());
	  }finally{
			iStream.close();
			urlConnection.disconnect();
	  }
	  return data;
     }
	// Fetches data from url passed
	private class DownloadTask extends AsyncTask<String, Void, String>{
		@Override
		protected String doInBackground(String... url) {
			String data = "";
			try{
				data = downloadUrl(url[0]);
```

```
                        }catch(Exception e){
                                Log.d("Background Task",e.toString());
                        }
                        return data;
                }
                @Override
                protected void onPostExecute(String result) {
                        super.onPostExecute(result);
                        ParserTask parserTask = new ParserTask();
                        // Invokes the thread for parsing the JSON data
                        parserTask.execute(result);
                }
        }
        /** A class to parse the Google Places in JSON format */
    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String,String>>> >{
        // Parsing the data in non-ui thread
                @Override
                protected List<List<HashMap<String, String>>> doInBackground(String...
jsonData) {
                        JSONObject jObject;
                        List<List<HashMap<String, String>>> routes = null;
               try{
                jObject = new JSONObject(jsonData[0]);
                DirectionsJSONParser parser = new DirectionsJSONParser();
                // Starts parsing data
                routes = parser.parse(jObject);
               }catch(Exception e){
                e.printStackTrace();
               }
               return routes;
                }
                // Executes in UI thread, after the parsing process
                @Override
                protected void onPostExecute(List<List<HashMap<String, String>>>
result) {
                        ArrayList<LatLng> points = null;
                        PolylineOptions lineOptions = null;
                        MarkerOptions markerOptions = new MarkerOptions();
                        // Traversing through all the routes
                        for(int i=0;i<result.size();i++){
                                points = new ArrayList<LatLng>();
                                lineOptions = new PolylineOptions();
                                // Fetching i-th route
                                List<HashMap<String, String>> path = result.get(i);
                                // Fetching all the points in i-th route
                                for(int j=0;j<path.size();j++){
                                        HashMap<String,String> point = path.get(j);
                                        double lat = Double.parseDouble(point.get("lat"));
                                        double lng = Double.parseDouble(point.get("lng"));
                                        LatLng position = new LatLng(lat, lng);
                                        points.add(position);
                                }
                                // Adding all the points in the route to LineOptions
```

```
                        lineOptions.addAll(points);
                        lineOptions.width(4);
                        lineOptions.color(Color.RED);
                    }
                    // Drawing polyline in the Google Map for the i-th route
                    mapa.addPolyline(lineOptions);
                }
    }
public void MostrarRuta(){
             db = sqlHelper.getReadableDatabase();
        String[] args = new String[]{"2", "ACTIVO"};
        Cursor cursorRuta = db.rawQuery("SELECT _id, latitud, longitud FROM Puntos
WHERE opcion =? AND estado =?", args);
        mapa.clear();
        if (cursorRuta.moveToFirst()) {
            //Recorremos el cursor hasta que no haya más registros
            do {
             latitude=Double.valueOf(cursorRuta.getString(1)).doubleValue();

      longitude=Double.valueOf(cursorRuta.getString(2)).doubleValue();
                        latlng = new LatLng(latitude,longitude);
                        mapa.addMarker(new MarkerOptions()
                    .position(latlng)
                    .title("Mi Auto..."));
            } while (cursorRuta.moveToNext());
        }
        CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngZoom(latlng, 17);
             mapa.animateCamera(cameraUpdate);
      }
}


// Clase de alertas
public class AlertDialogActivity extends Activity{
       String LatLon;
       String [] Latilong;
       String [] coordenadas;
       String dato="1";
       @Override
       protected void onCreate(Bundle savedInstanceState){
           super.onCreate(savedInstanceState);
           LatLon = getIntent().getStringExtra("LatLong");
           Latilong =  LatLon.split(",");
           AlertDialog.Builder builder = new AlertDialog.Builder(this);
           AlertDialog alert;
           switch(Integer.parseInt(Latilong[0])){
           case 1:
           case 2:
           case 3:
             Intent i = new Intent(AlertDialogActivity.this, MainActivity.class);
             String latitude = Latilong[1];
                   String longitude = Latilong[2];
                   i.putExtra("Lat",latitude);
                   i.putExtra("Long",longitude);
```

```
                    i.putExtra("Opcion", Integer.parseInt(Latilong[0]));
                    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
              startActivity(i);
              break;
            case 4:
                String alerta = Latilong[1];
                    builder
                        .setTitle("ALERTA")
                        .setMessage(alerta)
                        .setCancelable(false)
                        .setPositiveButton("Yes",new
DialogInterface.OnClickListener(){
                            public void onClick(DialogInterface dialog, int id){
                            Intent i = new Intent(AlertDialogActivity.this,
MainActivity.class);
                             i.putExtra("Opcion", Integer.parseInt(Latilong[0]));
                             String MSGdat= Latilong[1];
                             i.putExtra("msg",MSGdat);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                                        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                              startActivity(i);
                            }
                        })
                        .setNegativeButton("No", new
DialogInterface.OnClickListener(){
                            public void onClick(DialogInterface dialog, int id){
                               finish();
                            }
                        });
                    alert = builder.create();
                    alert.show();
              break;
             }
        }
}
```

ANEXO C

ENCUESTA PARA VERIFICAR FUNCIONAMIENTO

NOMBRE:

		SI	NO
1	La opción de seguimiento le funcionó correctamente.		
obs	……		
2	Presentó alguna falla al momento de pedir la Ubicación del Automóvil		
obs	……		
3	Las alertas le llegaron correctamente		
obs	……		
4	Cuantas alertas falsas le llegaron		
obs	……		
5	Pudo tener información correcta del estado del automóvil		
obs	……		
6	Le pareció útil la aplicación y su controlador		
obs	……		
7	A qué empresa telefónica pertenece su línea telefónica		
obs	……		

UNIVERSITAS MAJOR PACENSIS DIVI ANDREAE