ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Maestría en Sistemas de Información Gerencial

"DESARROLLO DE UNA APLICACIÓN DE SOFTWARE (APP) ANDROID QUE SIRVA DE COMUNICACIÓN ENTRE UN RASTREADOR SATELITAL Y UN CELULAR INTELIGENTE"

TESIS DE GRADO

Previa a la obtención del título de:

MAGISTER EN SISTEMAS DE INFORMACIÓN

GERENCIAL

Presentado por:

FABRICIO FABIÁN AGUILAR DOMÍNGUEZ

Guayaquil – Ecuador

Año: 2015

AGRADECIMIENTO

El presente trabajo primero le agradezco a Dios, por sus bendiciones recibidas y permitir que pueda culminar esta etapa de mi vida.

También agradezco a mi madre, quien siempre me ha brindado su apoyo incondicional, a mi padre por su ejemplo de constancia en el estudio e inculcarme en seguir preparándome profesionalmente, por sus sabios consejos que me han llevado a tomar buenas decisiones en la vida.

A mis hermanos que son mis amigos incondicionales con los que podré contar siempre.

DEDICATORIA

El presente trabajo se lo dedico a mi familia, a mi esposa por ser incondicional en darme su cariño, su apoyo y comprensión para poder culminar este trabajo de grado.

A mi hijo a quien le dejo el ejemplo del estudio que me inculcó mi padre.

TRIBUNAL DE SUSTENTACIÓN

Mgs. Lenin Freire C
DIRECTOR DE TESIS

Mgs. Karina Astudillo
MIEMBRO PRINCIPAL

Mgs. Juan Carlos García
MIEMBRO SUPLENTE

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL.)

_____

Ing. Fabricio Aguilar Domínguez

RESUMEN

El presente trabajo tiene como objetivo el desarrollo de una aplicación de Android que sirva para comunicar un rastreador satelital y un celular inteligente, de esta manera poder enviar de forma amigable los comandos mediante mensajes de texto (SMS) al rastreador y que este nos envíe un SMS con las coordenadas GPS de los eventos suscitados al celular, procesarlos, almacenarlos y visualizarlos en un mapa sin necesidad de tener internet para ello.

En el capítulo 1 veremos una introducción al sistema operativo Android, el entorno de desarrollo, lo que es la localización móvil y una introducción al rastreador satelital o GPS Tracker.

En el capítulo 2 veremos usos y aplicaciones del rastreador satelital, como se configura, como se instala, los protocolos de comunicación que maneja y los comandos SMS del rastreador.

En el capítulo 3 veremos un análisis y diseño de la aplicación, donde veremos las especificaciones de requerimientos funcionales y no funcionales.

En el capítulo 4 veremos el desarrollo y pruebas de la aplicación, configuración detallada del entorno de desarrollo utilizado y pruebas de interacción entre el vehículo, el rastreador y el celular.

ÍNDICE GENERAL

ABREVIATURAS Y SIMBOLOGÍA

Android	Es un sistema operativo basado en el núcleo Linux.
App	Aplicación de software para Android.
GPRS	Es un protocolo de comunicación de red.
GPS	Sistema de posicionamiento Global.
GPS Tracker	Rastreador satelital.
Smartphone	Teléfono móvil inteligente.
SMS	***S****hort* ***M****essage* ***S****ervice, servicio de* mensajes cortos.
ODB2	"On Board Diagnostics" (Diagnóstico de a bordo)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los teléfonos móviles nunca han sido más populares; los potentes smartphones son ahora una elección habitual para los consumidores; y el mundo Android ha ampliado su gama para incluir dispositivos como tabletas y televisión.

Cientos de estilos y de dispositivos versátiles - características de hardware que incluye GPS, acelerómetros, NFC y pantallas táctiles, combinadas con planes de datos a un precio aún no tanto razonable en nuestro país, proporcionan una plataforma atractiva sobre la cual crear aplicaciones innovadoras para todos los dispositivos Android.

Android ofrece una alternativa abierta para el desarrollo de aplicaciones móviles. Sin barreras, los desarrolladores de Android son libres de escribir aplicaciones que aprovechan al máximo el hardware del móvil para su distribución en un mercado abierto. Como resultado, el interés de los desarrolladores para dispositivos Android ha disparado las ventas de teléfonos móviles que han seguido creciendo. Desde el 2012, hay cientos de

teléfonos móviles y tabletas de diferentes fabricantes, incluyendo HTC, Motorola, LG, Samsung, ASUS, y Sony Ericsson. Más de 300 millones de dispositivos Android han sido activados, y ese número está creciendo a un ritmo aproximado de 850.000 activaciones diarias.

El uso de Google Play para su distribución, los desarrolladores pueden tomar ventaja de una tienda abierta, sin proceso de revisión, para la distribución de aplicaciones gratuitas y de pago para todos los dispositivos compatibles con Android. Construido en un marco de código abierto, y con potentes bibliotecas de SDK, Android ha permitido a más que 450.000 aplicaciones que se publiquen en Google Play.

CAPÍTULO 1

GENERALIDADES.

1.1 DESCRIPCIÓN DEL PROBLEMA.

El índice de delincuencia aumenta día a día, el robo de autos, los secuestros, robo de viviendas, etc. Esto hace necesario que las personas tomen medidas para cuidar de su seguridad y de los suyos.

Actualmente crece la demanda en el mercado de utilizar dispositivos de rastreo satelital, estos equipos nos permite mediante coordenadas GPS (Global Positioning System) localizar un objeto enviando su posición geográfica expresada en latitud y longitud a un computador o celular. Estos equipos utilizan un chip de telefonía celular para

comunicarse por medio de internet usando el protocolo GPRS o por mensajes de texto SMS.

El dispositivo de rastreo maneja sus propios comandos, tanto para su configuración como para el control de las funciones que debe realizar.

Existen dos tipos de rastreadores, los portátiles para personas o animales y los fijos que son para instalar en objetos motorizados, este último es más completo ya que permiten controlar algunas funciones como la apertura de puertas, apagado de motor, botón de pánico en caso de secuestro, consumo de combustible, entre otras.

Estos rastreadores pueden enviar la coordenada del punto donde se encuentra ubicado, mediante un mensaje de texto donde me indica la longitud y latitud. Para poder saber dónde es exactamente dicha dirección, se debe tener a la mano una computadora con google maps o google earth para ingresar las coordenadas y visualizarlas en el mapa. En cambio si se tiene un teléfono inteligente (Smartphone) se podría visualizar directamente pero se necesita de una conexión a internet.

Las compañías establecidas en el mercado tienen un alto costo por dar este servicio, todas sirven de intermediarias entre el dispositivo y el usuario, muchas veces el cliente no ve el servicio hasta que le sustraen el carro.

1.2 SOLUCIÓN PROPUESTA.

Se propone desarrollar una App de Android para un teléfono móvil inteligente, el cual envíe los comandos de una forma amigable al dispositivo de rastreo satelital y este a su vez envíe las coordenadas al celular, el SMS se procesara y mostrara el punto en un mapa offline, sin necesidad de tener internet en el celular.

Este software también podrá interactuar con el dispositivo de rastreo, con las funciones de apagado de motor, apertura de puertas, alertas de movimientos, geo cercas, botón de pánico, entre otras.

Con esta aplicación desde un smartphone con android se podrá localizar un objeto, indicando el punto exacto de su ubicación en un mapa, y en caso de ser un vehículo tener todo el control en la palma de la mano, sin tener que llamar a una operadora para que abra las puertas en caso de que se queden olvidadas las llaves en el interior del auto o que bloquee el motor en caso de un robo, o envíe una señal de alerta a varios números en caso de una emergencia.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Desarrollo de una App de Android que sirva para comunicar un rastreador satelital y un celular inteligente.

1.3.2 OBJETIVOS ESPECÍFICOS

- Conocer acerca del sistema operativo android, características, el por qué es el más utilizado en los teléfonos inteligentes, así también conocer la importancia de los dispositivos de rastreo satelital.

- Conocer el uso y aplicaciones de los dispositivos de rastreo satelital, instalación, configuración, protocolos y comandos que maneja.

- Desarrollar un software en Android, que permita de interface entre un dispositivo de rastreo y un celular, en el que se pueda enviar de forma fácil los comandos a través de mensajes de texto al rastreador, y a su vez

recibir mensajes de texto del dispositivo, procesarlos y presentarlos en un mapa offline en el celular.

1.4 INTRODUCCIÓN A ANDROID.

Android es un sistema operativo, al principio fue pensado solo para teléfonos móviles, está basado en Linux, que es de código abierto, lo que significa que es gratuito, este ha logrado introducirse en los principales mercados del mundo.

El lenguaje de programación utilizado en Android es Java y el entorno de tiempo de ejecución conocido como máquina virtual Dalvik.

“Android es en primer lugar un proyecto de Google, en colaboración con la Open Handset Alliance, que es una agrupación de casi 50 entidades comprometidas con el objetivo de comercializar una telefonía móvil mejor y más abierta”[**1**].

Se ha convertido en unos pocos años en una fuerza de cambio en el mercado de la telefonía móvil ganándose por el camino el respeto de muchos, y la burla de otros tantos.

En la actualidad se puede encontrar en dispositivos, como computadores, tabletas, GPS, reproductores multimedia, tv box, mini computadores, cámaras digitales, etcétera[**2**].

1.4.1 LA ARQUITECTURA ANDROID.

El siguiente gráfico muestra la arquitectura de Android. Lo componen cuatro capas. Las características más significativas es que todas son fundamentadas en software libre.

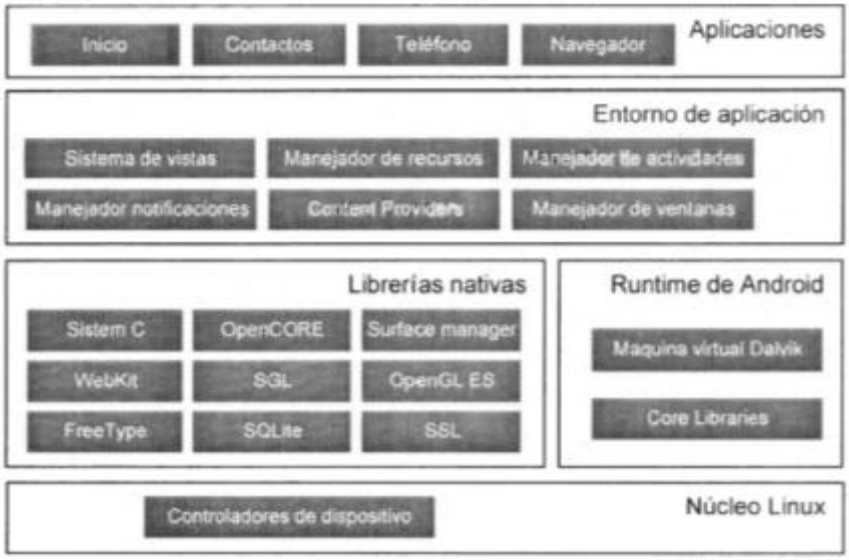

Figura 1.1 Arquitecturade Android

Fuente: Gironés Tomás Jesús, El gran libro de Android 2º Edición

1.4.1.1. EL NÚCLEO LINUX

El núcleo de android es Linux 3.6 con aproximadamente 115 parches. Esto proporciona un nivel de abstracción entre el hardware del dispositivo y contiene todos los controladores de hardware esenciales como la cámara, teclado, pantalla, etc. Además, el kernel maneja todas las cosas que Linux es realmente bueno como las redes y una amplia gama de controladores para los dispositivos.

1.4.1.2. RUNTIME DE ANDROID

En esta sección se proporciona un componente clave llamado Dalvik Virtual Machine que es una especie de máquina virtual de Java especialmente diseñado y optimizado para Android.

El Dalvik VM hace uso de las características principales de Linux como la gestión de la memoria y el multi-threading, que es intrínseca en el lenguaje Java. El Dalvik VM permite correr a todas las aplicaciones de

Android para en su propio proceso, con su propia instancia de la máquina virtual Dalvik.

El tiempo de ejecución de Android también proporciona un conjunto de bibliotecas del núcleo que permiten a los desarrolladores de aplicaciones Android escribir aplicaciones de Android usando el lenguaje de programación Java estándar.

1.4.1.3. LIBRERÍAS NATIVAS

Esta categoría incluye las bibliotecas basadas en Java que son específicos para el desarrollo de Android. Los ejemplos de las bibliotecas en esta categoría incluyen las bibliotecas marco de aplicación, además de las que facilitan la creación de interfaz de usuario, gráficos y el acceso de base de datos. Un resumen de algunas bibliotecas Android fundamentales clave a disposición de los desarrolladores de Android son las siguientes:

Android.app - Proporciona acceso al modelo de solicitud y es la piedra angular de todas las aplicaciones de Android.

Android.content - Facilita el acceso al contenido, la edición y la mensajería entre aplicaciones y componentes de aplicaciones.

Android.database - Se utiliza para acceder a los datos publicados por los proveedores de contenidos e incluye clases de manejo de base de datos SQLite.

Android.opengl - Usa la interfaz de Java para la API de los gráficos OpenGL ES 3D renderizado.

Android.os - Proporciona aplicaciones con acceso a los servicios estándar del sistema operativo, incluyendo mensajes, servicios del sistema y la comunicación entre procesos.

Android.text - Se utiliza para representar y manipular texto en una pantalla del dispositivo.

Android.view –Son los bloques fundamentales de las interfaces de construcción de las aplicaciones del usuario.

android.widget - Una rica colección de componentes de interfaz de usuario pre-construidos, como botones, etiquetas, las vistas de listas, los administradores de diseño, botones de radio, etc.

Android.webkit - Un conjunto de clases destinadas a permitir a las capacidades de navegación web que se construirán en las aplicaciones.

Después de haber cubierto las bibliotecas del núcleo basadas en Java en el tiempo de ejecución de Android, ahora es el momento de centrar nuestra atención en las bibliotecas basadas C / C ++ contenidas en esta capa de la pila de software Android.

1.4.1.4. ENTORNO DE APLICACIÓN.

La capa Application Framework proporciona muchos servicios de nivel superior para las aplicaciones en forma de clases de Java. Los desarrolladores de aplicaciones pueden hacer uso de estos servicios en sus aplicaciones.

El marco Android incluye los siguientes servicios clave:

Activity Manager - Controla todos los aspectos del ciclo de vida de la aplicación y de las actividades.

Content Providers - Permite a las aplicaciones publicar y compartir datos con otras aplicaciones.

Resource Manager - Proporciona acceso a los recursos embebidos tales como strings, ajustes de color y diseños de interfaz de usuario.

Notifications Manager - Permite a las aplicaciones mostrar alertas y notificaciones para el usuario.

View System - Un conjunto extensible de Viewers que se utilizan para crear interfaces de usuario de aplicaciones [**3**].

1.4.1.5. APLICACIONES.

Aquí se encuentra las aplicaciones instaladas en un equipo con Android. Estas apps son ejecutadas desde Dalvik Virtual Machine.

Los dispositivos de android vienen típicamente con una colección de aplicaciones preinstaladas que forman parte de Android Open Source Project (AOSP), como:

- Un cliente para correo electrónico
- Una aplicación para SMS
- Lista de direcciones y de contactos, calendario.
- Un navegador basado en WebKit.
- Reproductor de música y galería de fotos

- Una aplicación para tomar fotos y grabar videos.
- Una calculadora
- Una pantalla de inicio
- Un despertador

La mayoría de los dispositivos Android viene con aplicaciones instaladas de Google como:

- Google Play, es una tienda de Google que te permite descargar aplicaciones.
- Aplicaciones que usan google Maps, que incluye vista de calles, dirección de manejo y navegación, vista de satélites y condiciones de tráfico.
- Un cliente de correo Gmail.
- Mensajes instantáneos de Google Talk.
- Reproductor de video de Youtube [**4**].

1.4.1.6. NOMBRES DE LAS VERSIONES DE ANDROID.

Figura 1.2 Logo de Android.

Fuente: http://www.brandemia.org

Las versiones de Android reciben, en inglés, es el nombre de diferentes dulces. En cada versión el dulce empieza por una letra distinta, en orden alfabético:

A: Apple Pie (v1.0): Pie de manzana.

B: Banana Bread (v1.1): Pan de banana.

C: Cupcake (v1.5)

D: Donut (v1.6): (API 4): sep. 2009.

E: Éclair (v2.0/v2.1): (API 7): oct. 2009.

F: Froyo (v2.2): Yogurt (API 8): mayo 2010.

G: Gingerbread (v2.3): Pan de jengibre (API 10): dic. 2010

H: Honeycomb (v3.0/v3.1/v3.2): Panal de miel (API 11 - 12 - 13):feb. 2011

I: Ice Cream Sandwich (v4.0): Sanduche de helado (API 14 - 15): oct 2011

J: JellyBean (v4.1/v4.2/v4.3): Gominola(API 16): jun. 2012

K: KitKat (v4.4) API 19 (oct 2013)

L: Lollipop (v5.0/v5.1):API 21 (nov.2014)

M: Android M (v6.0): Sin nombre confirmado.

(Se espera en el último trimestre del 2015) [**5**]

1.4.1.7. CARACTERÍSTICAS DE ANDROID.

El verdadero atractivo de Android como un entorno de desarrollo reside en sus APIs.

Como una plataforma de aplicaciones neutral, Android le da la oportunidad de crear aplicaciones que son una parte tan importante del teléfono. En la siguiente lista destaca algunas de las características más destacadas de Android:

- GSM, EDGE, redes 3G, 4G y LTE para telefonía o de transferencia de datos, lo que le permite realizar o recibir llamadas o mensajes SMS, o

para enviar y recibir datos a través de redes móviles.

- API integrales para los servicios basados en la localización como el GPS y la ubicación basada en la triangulación de antenas.
- Soporte completo para las aplicaciones que se integran los controles mapa como parte de sus interfaces de usuario.
- Acceso Wi-Fi a conexiones de hardware y punto a punto.
- Completo control de hardware multimedia, incluyendo la reproducción y la grabación con la cámara y micrófono.
- Bibliotecas de medios para reproducir y grabar una variedad de audio / vídeo o imágenes en diferente formatos.
- APIs para el uso del sensor, incluyendo acelerómetros, brújulas, y barómetros.
- Bibliotecas para el uso de Bluetooth y NFC para la transferencia de datos punto a punto.
- Comunicación entre procesos IPC.

- Almacenamiento de datos compartidos y APIs para los contactos, redes sociales, calendario y multimedia.
- Servicios de background, aplicaciones y procesos.
- Widgets de pantalla Inicio y Live Wallpaper.
- La capacidad de integrar la búsqueda de resultados de aplicaciones en el sistema de búsqueda.
- Un integrador de código abierto un navegador HTML5 basado en WebKit.
- Optimizador para móviles, acelerados gráficos por hardware, incluyendo una biblioteca de gráficos 2D y soporte para gráficos 3D usando OpenGL ES 2.0.
- La localización a través de un marco de recursos dinámicos.
- Un marco de aplicación que fomenta la reutilización de los componentes de la aplicación y el reemplazo de aplicaciones nativas [**4**].

1.5 ENTORNO DE DESARROLLO ANDROID.

Para el desarrollo de las aplicaciones vamos a poder utilizar un potente y moderno entorno de desarrollo. Al igual que Android, todas las herramientas están basadas en software libre. Aunque existen varias alternativas para desarrollar aplicaciones en Android. Para el desarrollo de este sistema se utilizó el software enumerado a continuación:

- Java Runtime Environment 7.0 o superior.
- Eclipse (Eclipse IDE for Java Developers).
- Android SDK (Google).
- Eclipse Plug-in (Android Development Toolkit- ADT).

1.5.1. INSTALACIÓN DE LA MÁQUINA VIRTUAL JAVA.

Este software va a permitir ejecutar código Java en el equipo. A la máquina virtual Java también se la conoce como entorno de ejecución Java, Java Runtime Environment (JRE) o Java Virtual Machine (JVM).

Para instalar la máquina virtual Java accede a *https://www.**java**.com/es/download/* y descarga e instala el fichero correspondiente a tu sistema operativo.

1.5.2. INSTALACIÓN DE ECLIPSE

Eclipse resulta el entorno de desarrollo más recomendable para Android, es libre y además es soportado por Google (ha sido utilizado por los desarrolladores de Google para crear Android). Para el desarrollo de este trabajo de tesis se utilizó la versión Luna.

Para instalar Eclipse Luna, ir a su website www.eclipse.org/downloads/ y baje la versión de Eclipse Luna para Java. Una vez bajado, descomprima el archivo zip en la carpeta de su elección.

1.5.3. INSTALAR ANDROID SDK DE GOOGLE

El siguiente paso va a consistir en instalar Android SDK de Google.

Se lo descarga desde http://developer.android.com/sdk/ tomar en consideración que versión de sistema operativo tenemos, 32 o 64 bits.

Este software no requiere una instalación específica, simplemente descomprimir los ficheros en la carpeta que prefieras [**3**].

Este SDK contiene todas las herramientas necesarias para crear una aplicación Android. Está disponible para Windows, Mac OS y Linux.

1.5.4. INSTALACIÓN DE ADT DE ECLIPSE

Para instalar el plugin de eclipse, nos dirigimos a la pestaña Help –> Install New Software.

Ver en detalle la instalación en el capítulo 4.1 Instalación y configuración del lenguaje de programación y entorno de desarrollo.

1.6 ACCEDER A INFORMACIÓN DEL TELÉFONO.

1.6.1. EL ACCESO A HARDWARE, INCLUYENDO LA CÁMARA, GPS Y SENSORES.

Android incluye bibliotecas API para simplificar el desarrollo que envuelve el hardware del dispositivo.

Esto asegura de que no es necesario crear implementaciones específicas de su software para diferentes dispositivos, por lo que pueden crear aplicaciones Android que funcionen como se espera en cualquier dispositivo con este sistema.

El SDK de Android incluye APIs para el hardware basado en la localización (como el GPS), la cámara, audio, conexiones de

red, Wi-Fi, Bluetooth, sensores (incluyendo acelerómetros), NFC, la pantalla táctil, y administración de energía.

1.6.1.2. LAS TRANSFERENCIAS DE DATOS CON CONEXIÓN WI -FI, BLUETOOTH Y NFC.

Android ofrece un rico soporte para la transferencia de datos entre dispositivos, incluyendo Bluetooth, Wi- Fi Direct y Android Beam. Estas tecnologías ofrecen una rica variedad de técnicas para el intercambio de datos entre los dispositivos vinculados, dependiendo del hardware disponible en el dispositivo subyacente, lo que permite crear aplicaciones de colaboración innovadoras. Además, Android ofrece API´s para gestionar las conexiones de red, las conexiones Bluetooth, y la lectura de etiquetas NFC.

1.6.1.3. MAPAS, GEO CODIFICADOS Y SERVICIOS BASADOS EN LOCALIZACIÓN.

Mapas de apoyo incorporado le permite crear una gama de aplicaciones basadas en mapas que aprovechan la movilidad de los dispositivos Android. Le permite diseñar interfaces de usuario que incluyen mapas

interactivos Google que se puede controlar mediante programación y usar anotaciones usando una rica biblioteca de gráficos de Android.

Servicios basados en la localización como el GPS y redes de Google que usa tecnología de localización para determinar la posición actual del dispositivo. Estos servicios hacen cumplir una abstracción de la tecnología de localización de detección específica y permiten especificar los requisitos mínimos (por ejemplo, exactitud o costo) en lugar de seleccionar una tecnología en particular. Esto también significa que su aplicación basada en localización funcionará sin importar que tecnología sea compatible con el dispositivo.

Para combinar mapas con localización, Android incluye una API para reenviar y geo codificación inversa que te permite encontrar las coordenadas por una dirección y la dirección de la posición de un mapa.

1.6.1.4. SERVICIOS EN SEGUNDO PLANO.

Android es compatible con aplicaciones y servicios diseñados para ejecutarse en segundo plano mientras la aplicación no está siendo usada activamente.

Los móviles modernos y tabletas son por naturaleza dispositivos multifunción; sin embargo, sus tamaños de pantalla y modelos de interacción significan que en general sólo una aplicación interactiva es visible en cualquier momento.

Las plataformas que no admiten la ejecución en segundo plano limitan la viabilidad de las aplicaciones que no necesitan su atención constante.

Los servicios en segundo plano hacen posible la creación de componentes de aplicaciones invisibles que realizan procesamiento automático sin la acción directa del usuario. La ejecución en segundo plano permite a sus aplicaciones actualizar sus eventos regularmente, lo cual es perfecto para monitorear marcadores de juegos de fútbol o precios de mercado, generando alertas basadas en localización, o priorizar y preseleccionar llamadas entrantes y mensajes SMS.

Las notificaciones son el medio estándar por el cual un dispositivo móvil alerta de los eventos que han ocurrido en una aplicación en segundo plano. Utilizando el Administrador de notificaciones, puede desencadenar alertas audibles, vibraciones y destellos LED del dispositivo, así como el control del estado de la barra de notificaciones.

1.6.1.5. BASE DE DATOS SQLITE PARA EL ALMACENAMIENTO Y RECUPERACIÓN DE DATOS.

El almacenamiento y recuperación de datos que sea rápida y eficiente son esenciales para un dispositivo cuya capacidad de almacenamiento es relativamente limitada.

Android ofrece una base de datos relacional ligera para cada aplicación a través de SQLite. Sus aplicaciones puede tomar ventaja de este motor de base de datos relacional logrado almacenar datos de forma segura y eficiente

Por defecto, cada aplicación de base de datos es segura, su contenido sólo está disponible para la aplicación creada, pero los proveedores de contenido dan un mecanismo para la compartición controlada de estas bases de datos, así como proporcionar una abstracción entre la aplicación y el código fuente.

1.6.1.6. DATOS COMPARTIDOS Y LA COMUNICACIÓN ENTRE APLICACIONES.

Android incluye varias técnicas para hacer que la información de las aplicaciones estén disponibles para su uso en otros lugares, principalmente: Intents y Content Providers.

Los Intents proporcionan un mecanismo para pasar mensajes entre aplicaciones. Con el uso de Intents, usted puede difundir una acción deseada (como marcar desde el teléfono o editar un contacto) para todo el sistema y manejar otras aplicaciones. Utilizando el mismo mecanismo, puede registrar su propia aplicación

para recibir estos mensajes o ejecutar las acciones solicitadas.

Se puede utilizar proveedores de contenido para ofrecer, gestionar acceso seguro a las bases de datos privados de sus aplicaciones.

Los almacenes de datos para aplicaciones nativas, como el gestor de contactos, se exponen como proveedores de contenidos por lo que pueden leer o modificar estos datos desde sus propias aplicaciones.

1.6.1.7. EL USO DE WIDGETS Y LIVE WALLPAPER PARA MEJORAR LA PANTALLA DE INICIO.

Widgets y Live Wallpaper permiten crear aplicaciones con componentes dinámicos que proporcionan una ventana en sus aplicaciones, puede ofrecer información útil y oportuna, directamente en la pantalla de inicio.

Ofreciendo una manera para que los usuarios interactúen con su solicitud directamente desde la pantalla de inicio, esto aumenta la participación de los usuarios, dándoles acceso instantáneo a información

interesante sin necesidad de abrir la aplicación, así como la adición de un acceso directo dinámico en su aplicación desde la pantalla de inicio.

1.6.1.8. AMPLIO SOPORTE DE MEDIOS GRÁFICOS 2D Y 3D.

Pantallas más grandes y más brillantes, las pantallas de mayor resolución han ayudado a hacer dispositivos móviles multimedia. Para ayudarle a sacar el máximo provecho del hardware disponible, Android ofrece librerías gráficas para dibujo en 2D y 3D con OpenGL.

Android también ofrece bibliotecas integrales para el manejo de imágenes, vídeo, y archivos de audio, incluyendo soporte para MPEG4, H.264, HTTP Live Streaming, VP8, WEBP, MP3, AAC, AMR, HLS, JPG, PNG y GIF.

1.6.1.9. MENSAJERÍA DE DISPOSITIVOS EN LA NUBE.

La nube de Android para mensajería de dispositivos (C2DM) servicio que proporciona un mecanismo eficiente para los desarrolladores permite crear

aplicaciones orientadas a eventos en base a descargas del lado del servidor.

Usando C2DM puede crear una conexión permanente pero ligera entre la aplicación móvil y su servidor, que le permite enviar pequeñas cantidades de datos directamente a su dispositivo en tiempo real.

El servicio C2DM es usado típicamente para solicitar nuevos datos disponibles de las aplicaciones en el servidor, reduciendo la necesidad de conexiones, disminuyendo la descarga de la batería en actualizaciones de las aplicaciones.

1.6.1.10 OPTIMIZACIÓN DE MEMORIA Y GESTIÓN DE PROCESOS

Al igual que Java y .NET, Android utiliza su propio tiempo de ejecución y máquina virtual para gestionar la memoria de la aplicación. A diferencia con cualquiera de estos otros frameworks, el tiempo de ejecución Android también gestiona los tiempos de vida de

proceso. Android se responsabiliza de las aplicaciones al detener y matar procesos como sea necesario para liberar recursos para las aplicaciones de mayor prioridad.

En este contexto, se da la máxima prioridad a la aplicación con la que el usuario está interactuando. Asegurándose de que sus aplicaciones se preparan para una muerte rápida, pero todavía son capaces de seguir respondiendo para actualizar o reiniciar en segundo plano si es necesario, es una importante consideración en un entorno que no permite a las aplicaciones poder controlar su propio tiempo de vida [**4**].

1.7 LOCALIZACIÓN MÓVIL.

La plataforma Android dispone de un interesante sistema de posicionamiento que combina varias tecnologías:

Sistema de localización global basado en GPS. Este sistema solo funciona si disponemos de visibilidad directa de los satélites.

Sistema de localización basado en la información recibida de las torres de telefonía celular y de puntos de acceso Wi-Fi. Funciona en el interior de los edificios.

Estos servicios se encuentran totalmente integrados en el sistema y son usados por gran variedad de aplicaciones.

El sistema de posicionamiento global, GPS, fue diseñado inicialmente con fines militares pero hoy en día es ampliamente utilizado para uso civil. Gracias al desfase temporal de las señales recibidas por varios de los 28 satélites desplegados, este sistema es capaz de posicionarnos en cualquier parte del planeta con una precisión de 1 metro.

El GPS presenta un inconveniente; solo funciona cuando tenemos visión directa de los satélites. Para solventar este problema, Android combina esta información con la recibida de las torres de telefonía celular y de puntos de acceso Wi-Fi [**3**].

1.8 INTRODUCCIÓN AL GPS TRACKER.

1.8.1. SISTEMA DE POSICIONAMIENTO GLOBAL GPS.

Sistema Global de posicionamiento basado en 28 satélites que orbitan la tierra a una altitud de 12.000 millas que entrega información precisa de posicionamiento y navegación mundial 24 horas al día. También es llamado sistema NAVSTAR.

Se fundamentan en determinar la distancia entre un punto: el usuario, a otros puntos de referencia: los satélites. Conociendo la distancia que separa los tres puntos se puede establecer nuestra ubicación relativa a esos mismos puntos. Es necesario mínimo cuatro satélites para establecer la posición correctamente.

1.8.1.2. INTEGRACIÓN CON TELEFONÍA MÓVIL.

La integración de los celulares con la tecnología GPS viene en dos características. Por un lado el celular puede tener un chip GPS instalado, o puede enchufarse a uno con cables o vía bluetooth. Los celulares con GPS son compatibles con Java, y nos sirve para indicarnos como llegar a un sitio que marquemos previamente. Para usar estas funciones,

necesitas: Un celular inteligente o receptor GPS compatible.

Un programa de mapas offline como Navit o Maps.me entre otros, o google maps pero debes tener un plan de datos.

Los usos más comunes son como guía de orientación o como localizadores de objetos o personas [**6**].

1.8.2. EQUIPOS DE RASTREO SATELITAL (GPS TRACKER).

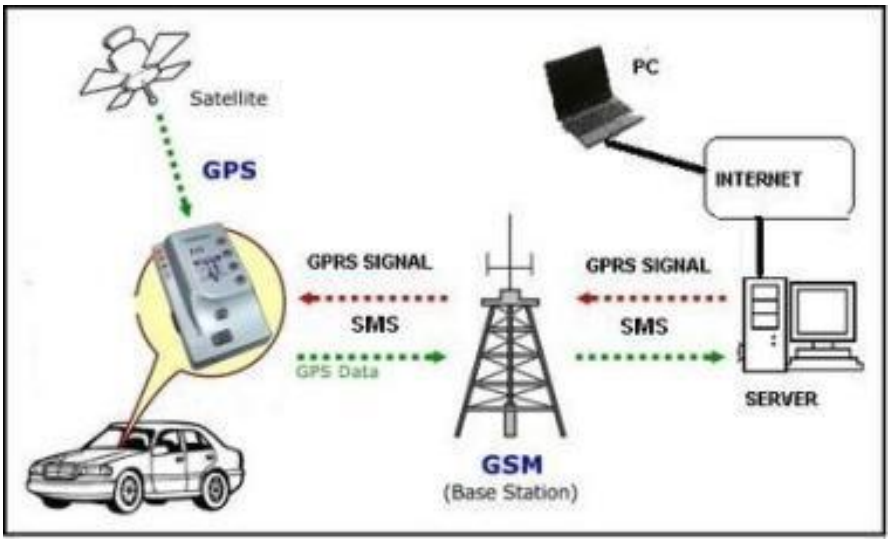

Figura 1.3 Rastreo satelital vehicular.

Fuente: Internet

El Rastreo Satelital es un servicio que permite localizar vehículos, personas u objetos en cualquier parte del mundo por medio de triangulación de señales emitidas por 28 satélites geo estacionarios alrededor del planeta. El servicio de Rastreo Satelital como tal es abierto, aunque para hacer uso de él es necesario tener un dispositivo habilitado con GPS (Global Positioning System), comúnmente un celular, PDA, navegador personal o equipo AVL.

Los equipos que pueden hacer uso del servicio de Rastreo Satelital se dividen en dos tipos: los que trabajan fuera de línea o "pasivos" y los que son en línea o "activos", teniendo como diferencia el poder comunicar la información de posición de manera instantánea o no a una central de monitoreo remota.

Un equipo de Rastreo Satelital "pasivo" generalmente muestra la información en el mismo aparato, otros equipos pasivos guardan esa información en memoria para eventualmente ser descargada y analizada. Celulares con GPS, navegadores personales y loggers de posición entran en esta categoría.

Un equipo de Rastreo Satelital "activo" utiliza un medio alterno de comunicación como una red celular para enviar su información hacia una central remota de monitoreo en tiempo real. En esta categoría están los sistemas AVL (Automatic Vehicle Location) como el usado en este proyecto.

El Rastreo Satelital en tecnología AVL es una herramienta poderosa en la administración de flotillas, asignación de vehículos de emergencia, sistemas de transporte público, recuperación de vehículos robados, especialmente si se integra con sistemas de call center, centrales de monitoreo, planificadores de ruta, sistemas de bodega, sistemas de despacho, etc [**7**].

Actualmente crece la demanda en el mercado de utilizar dispositivos de rastreo satelital, estos equipos nos permite mediante coordenadas GPS localizar un objeto enviando su posición geográfica expresada en latitud y longitud a un computador o celular. Estos equipos utilizan un chip de telefonía celular para comunicarse por medio de internet usando el protocolo GPRS o por mensajes de texto SMS.

El dispositivo de rastreo maneja sus propios comandos, tanto para su configuración como para el control de las funciones que debe realizar.

Existen dos tipos de rastreadores activos, los portátiles para personas o animales y los fijos que son para instalar en objetos motorizados, este último es más completo ya que permiten controlar algunas funciones como la apertura de puertas, apagado de motor, botón de pánico en caso de secuestro, consumo de combustible, entre otras.

Estos rastreadores pueden enviar las coordenadas del punto donde se encuentra ubicado, mediante su protocolo de comunicación donde me indica la longitud y latitud, si se está utilizando una plataforma (comercial) de monitoreo la comunicación entre el rastreador y la central es por medio de GPRS, se necesita contratar un paquete de datos para el envío de la información.

También se puede usar los equipos de manera no comercial por mensajes de texto SMS, para poder saber dónde es exactamente dicha dirección, se debe tener a la mano una

computadora con Google maps o Google earth para ingresar las coordenadas y visualizarlas en el mapa. En cambio si se tiene un teléfono inteligente (Smartphone) se podría visualizar directamente pero se necesita de una conexión a internet.

CAPÍTULO 2

LEVANTAMIENTO DE REQUERIMIENTOS Y CONFIGURACIÓN DE HARDWARE.

2.1 GPS TRACKER, USOS Y APLICACIONES.

Figura 2.1 Usos y aplicaciones del rastreador GPS

Fuente: http://alsimachinery.com

El rastreador satelital tiene diferentes usos de carácter personal y comercial entre los principales tenemos:

Sistema de rastreo de personas y objetos: Esta aplicación es útil para quienes deseen conocer la ubicación de sus familiares, amigos o mascotas. De igual manera si se requiere ubicar y rastrear objetos, este sistema puede ser usado en vehículos y otros objetos de valor para ser localizados rápidamente y con precisión.

Sistema de control de rutas: La localización de un vehículo se realiza partiendo de la instalación de rastreador satelital en el mismo; a continuación el usuario puede ver en su computador en donde se encuentra el vehículo, qué rutas ha realizado, qué velocidad lleva, dónde se ha estacionado, entre otros. Este servicio le permitirá observar e identificar a toda su flota de camiones o vehículos.

Localización de vehículos, robados o extraviados facilitando su recuperación, además cuenta con bloqueo de motor remoto, lo que hace que el vehículo se apague en cualquier lugar que se encuentre.

Evitar un secuestro mandando una señal, botón de pánico escondido en el vehículo, regularmente va instalado en un punto estratégico de la unidad [**8**].

Los equipos más avanzados también pueden ser conectados a video cámaras que envían la señal de video a la central para ser monitoreados.

2.2 INSTALANDO EL GPS TRACKER.

El dispositivo de rastreo satelital utilizado en este proyecto es de la marca Meitrack modelo MVT380.

Figura 2.2 Rastreador satelital Meitrack MVT380

Fuente: Manual Meitrack

2.2.1. FUNCIONES Y ESPECIFICACIONES:

2.2.1.1. FUNCIONES:

- SiRF III GPS and Quad Band GSM

850/900/1800/1900Mhz

- AGPS (with GSM Base Station ID)
- Track by SMS/GPRS (TCP/UDP) (MEITRACK Protocol)
- Track on Demand
- Track by Time Interval
- Track by Distance Interval
- Track on Mobile Phone
- Listen-in (Voice Monitoring) or Two-way Audio (Optional)
- Internal 4Mb Memory for Logging
- Inbuilt Motion Sensor
- 850mAh Internal Backup Battery
- SOS Alarm
- Geo-fence Alarm
- GPS Blind Area Alarm
- Low Battery Alarm
- Speeding Alarm
- Tow Alarm
- GPS Antenna Cut Alarm
- External Power Cut Alarm
- Mileage Report

- Engine Cut (Engine immobilization)
- Inbuilt Super Magnet (optional)
- 5 Digital Inputs (2 positive triggering and 3 negative triggering), 5 Outputs.
- Analog Input Detection

2.2.1.2. ESPECIFICACIONES:

Tabla 1 Especificaciones del MVT380

Items	**Especificaciones**
Dimensiones	105*65*26mm
Peso	190g
Voltaje de entrada	DC 9V~36V/1.5ª
Back-up Battery	850mAh/3.7V
Consumo de energía	65mA standbycurrent
Temperatura de operación	-20℃~55℃
Humedad	5%~95%
Tiempo de batería	43 horas en modo ahorrador de energía y 10 horas en modo normal
LED	2 luces LED para

	mostrar GPS, GSM y otros estados.
Botones	1 SOS and 1 power on/off
Micrófono	Opcional
Memoria	4M Byte
Sensor	Motion sensor
Frecuencia GSM	GSM 850/900/1800/1900MHz
GPS Chip	Latest GPS SIRF-Star III chipset
Sensibilidad GPS	-159dB
Precisión	10 metros, 2D RMS
I/O	5 entradas digitales (2 positivas and 3 negativas) 2 entradas analógicas, 5 salidas de detección, 1 USB port

1.2.2. ACCESORIOS:

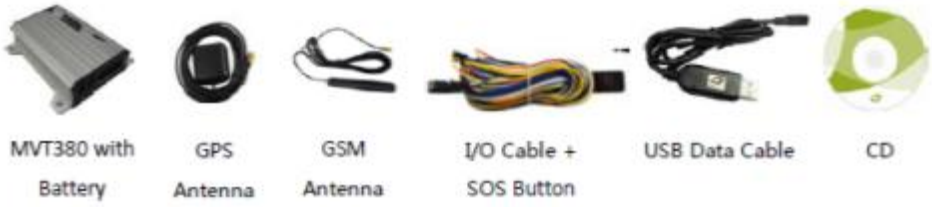

Figura 2.3 Accesorios del MVT380

Fuente: Manual Meitrack

2.2.3. DETALLES:

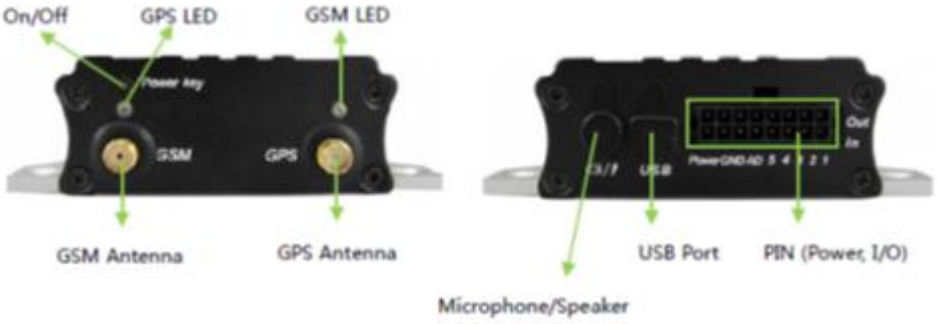

Figura 2.4 Detalles del MVT380

Fuente: Manual Meitrack

2.2.4. PRIMER USO.

2.2.4.1. INSTALANDO UNA TARJETA SIM.

Revisar que la tarjeta SIM tenga saldo para enviar y recibir mensajes de texto, para esto puedes probarla antes en un celular.

Revisa que el código de seguridad de la SIM este desactivado.

Antes de instalar la Sim, asegúrate de que el MVT380 este apagado.

Remueve los tornillos de la cubierta.

Inserta la tarjeta SIM, en la ranura viendo el diseño de la tarjeta luego cerrarla y asegurar deslizándola.

Luego poner la cubierta y asegurarla con los tornillos.

Figura 2.5 Como insertar el chip en el MVT380

2.2.4.2. CARGANDO

Conectar los cables GND (-negro) y el positivo (+red) a una fuente de poder externa de 12v o 24 y asegurarse que la batería se cargue por lo menos 3 horas.

Se sugiere que la primera vez se configure y se pruebe antes de instalarlo.

2.2.4.3. INDICADORES LED

Presionar y mantener el botón de encendido por 3 a 5 segundos para encender o apagar el MVT380.

Figura 2.6 Indicadores Led del MVT380

Fuente: Manual Meitrack

Tabla 2 Indicadores Led del MVT380

GPS LED (azul)	
Encendido	Un botón es presionado o una entrada esta activa.
Intermitente (cada 0.1 seg.)	Iniciando o la batería esta baja
Intermitente (0.1 seg encendido y 2.9 seg apagado)	MVT380 detecta señal GPS
Intermitente (1 seg encendido y 2 seg apagado)	MVT380 no detecta señal GPS
GSM LED (Verde)	
Encendido	Una llamada está entrando o se hace una llamada
Intermitente (	Iniciando

cada 0.1 seg)	
Intermitente (0.1 seg encendido y 2.9 seg apagado)	MVT380 está conectado a la red GSM
Intermitente(1 seg encendido y 2 seg apagado)	MVT380 no está conectado a la redGSM

2.2.4.4. LOCALIZACIÓN POR LLAMADA.

Realiza una llamada al MVT380 y este enviara el siguiente mensaje de texto SMS.

Ejemplo,

Now,110727

02:48,V,16,23Km/h,61%,http://maps.google.com/maps?f=q&hl=en&q=22.540103,114.082329

&ie=UTF8&z=16&iwloc=addr&om=1

Figura 2.7 Mensaje recibido del MVT380 al realizar una llamada

Fuente: Manual Meitrack

Si presionas sobre el link te lleva a Google Maps y te muestra el punto en tu celular.

Si tu celular no puede acceder a sitios HTTP, ingresa la latitud y longitud en Google Maps desde una computadora para que te muestre la posición:

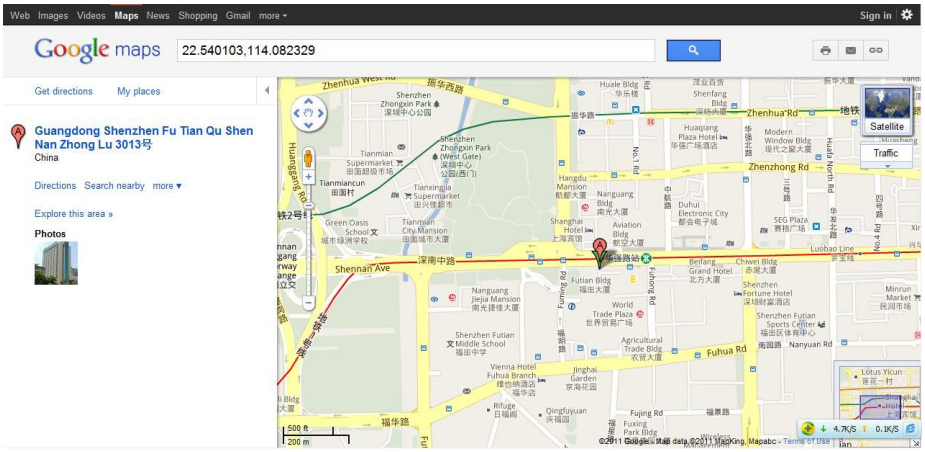

Figura 2.8 Localización de coordenada por Google maps

Fuente: Manual Meitrack

2.2.4.5. INSTALANDO EL RASTREADOR EN UN VEHÍCULO.

2.2.4.5.1. INSTALACIÓN DEL CABLE ENTRADAS Y SALIDAS.

Tabla 3 Entradas y salidas del MVT380

16 Power(+)	15 GND(-)	14 AD input 2	13 Output 5	12 Output 4	11 Output 3	10 Output 2	9 Output 1
8 Power(+)	7 GND(-)	6 AD input 1	5 Input 5	4 Input 4	3 Input 3	2 Input 2	1 Input 1

PIN Number	Color	Description
1 (IN1/SOS)	White	Digital input 1 (negative triggering)
2 (IN2)	White	Digital input 2 (negative triggering), for detecting status of vehicle door.
3 (IN3)	White	Digital input 3 (negative triggering), for detecting status of vehicle door or ACC.
4 (IN4)	White	Digital input 4 (positive triggering), for detecting status of vehicle door.
5 (IN5)	White	Digital input 5 (positive triggering), for detecting status of vehicle door or ACC.
6 (AD1)	Blue	10 Bits Resolution Analog inputs. 0~6V DC Detection. It can be used to connect with temperature/fuel sensor etc.
7 (GND)	Black	Ground
8 (POWER)	Red	DC in (power source). Input voltage: 9V~36V. 12V suggested.
9 (OUT1)	Yellow	Output1. It can be used to connect with relay for engineer immobilization. Low voltage (0V) when effective and open drain (OD) when ineffective. Output open drain sink voltage (ineffective): 45V max.
		Output low voltage sink current (effective): 500mA max.
10 (OUT2)	Yellow	Output2. It can be used to connect with relay for engineer immobilization. Low voltage (0V) when effective and open drain (OD) when ineffective. Output open drain sink voltage (ineffective): 45V max. Output low voltage sink current (effective): 500mA max.
11 (OUT3)	Yellow	Output3. It can be used to connect with relay for engineer immobilization. Low voltage (0V) when effective and open drain (OD) when ineffective. Output open drain sink voltage (ineffective): 45V max. Output low voltage sink current (effective): 500mA max.
12 (OUT4)	Yellow	Output4. It can be used to connect with relay for engineer immobilization. Low voltage (0V) when effective and open drain (OD) when ineffective. Output open drain sink voltage (ineffective): 45V max. Output low voltage sink current (effective): 500mA max.
13 (OUT5)	Yellow	Output5. It can be used to connect with relay for engineer immobilization. Low voltage (0V) when effective and open drain (OD) when ineffective. Output open drain sink voltage (ineffective): 45V max. Output low voltage sink current (effective): 500mA max.
14 (AD2)	Blue	10 Bits Resolution Analog inputs. 0~6V DC Detection. It can be used to connect with temperature/fuel sensor etc.
15 (GND)	Black	Ground. It can be used to connect with temperature/fuel sensor etc.
16 (POWER)	Red	DC in (power source). Input voltage: 9V~36V. 12V suggested. Same as PIN8

2.2.4.5.2. POWER/GND (PIN7/PIN8)

Conectar los cables GND (-negro) y positivo (+red) en la batería del vehículo.

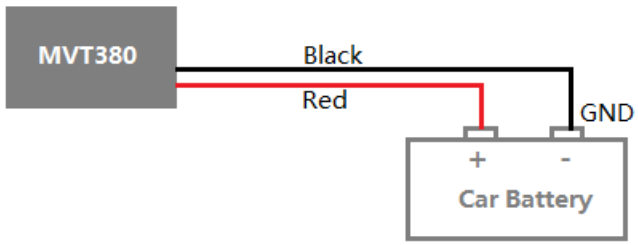

Figura 2.9 Alimentación del MVT380 con una batería.

Fuente: Manual Meitrack

2.2.4.5.3. ENTRADAS DIGITALES (PIN1/PIN2/PIN3 PULSO NEGATIVO)

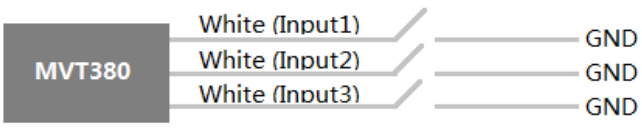

Figura 2.10 Entradas digitales pulso negativo

Fuente: Manual Meitrack

2.2.4.5.4. ENTRADAS DIGITALES (PIN4/PIN5 PULSO POSITIVO)

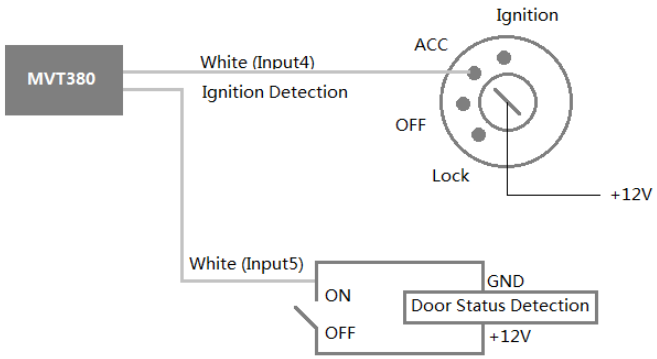

Figura 2.11 Entradas digitales pulso positivo

Fuente: Manual Meitrack

2.2.4.5.5. SALIDA (PIN9/PIN10/PIN11/PIN12/PIN13)

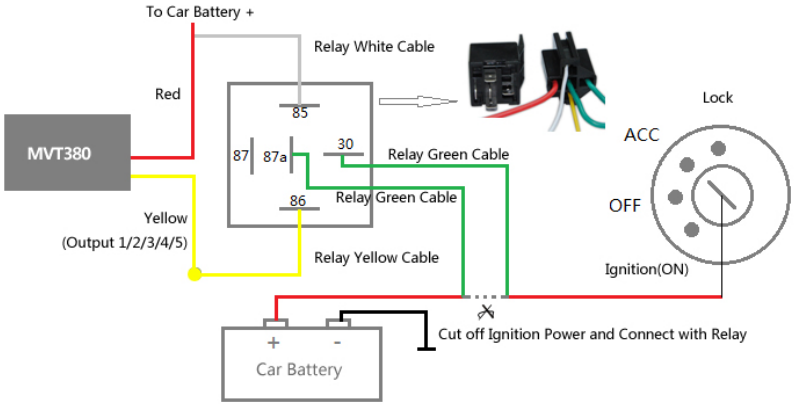

Figura 2.12 Salidas del MVT380.

Fuente: Manual Meitrack

2.2.4.5.6. ENTRADA ANALÓGICA (PIN6/PIN14)

2.2.4.5.6.1. Entrada analógica aplicación 1 – Detección externa de voltaje

Rango de entrada: 0-6v

Fórmula para calcular el voltaje:

Voltaje de entrada=(AD*6)/1024

0x0377=>887(Decimal)=>(887*6)/1024=5.1972V(Voltaje)

0x02FB=>763(Decimal)=>(763*6)/1024=4.4707V(Voltaje)

2.2.4.5.6.2. Entrada analógica aplicación 2 – Detección de combustible (%)

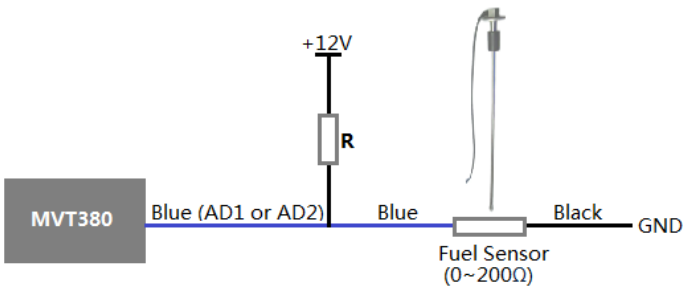

Figura 2.13 Instalando sensor de combustible

Fuente: Manual Meitrack

Nota:

El sensor de nivel de combustible lo provee la compañía y viene con resistencias de 0 - 200Ω (ohm).

El circuito mostrado en la figura de arriba, si el VCC es 12V, la resistencia debería ser de 200Ω y si el VCC es 24V la resistencia debería ser 600Ω para que la entrada AD1 y AD2 sea de 0-6V.

La siguiente formula calcula el porcentaje de combustible para este sensor:

Porcentaje disponible = (valor AD / (1024*2 – valorad)) * 100%

El valor puede ser convertido en decimal, por ejemplo, 0x0267 es 615 en decimal.

2.2.4.5.7. INSTALANDO LAS ANTENAS GPS/GSM

Figura 2.14 Instalando antenas del MVT380.

Fuente: Manual Meitrack

Conecta la antena GSM al conector SMA con la etiqueta “GSM”. La antena GSM es no direccional por lo que puedes esconderla en cualquier lugar del vehículo.

Conecta la antena GPS al conector con etiqueta “GPS”. Una óptima instalación de la antena GPS es en el techo del vehículo, asegurarse que la parte plana este hacia abajo y la curvatura hacia arriba, asegurarla con goma.

Nota: no cubrir la antena con ningún objeto de metal.

2.2.4.5.8. INSTALANDO MICRÓFONO Y PARLANTE (OPCIONAL)

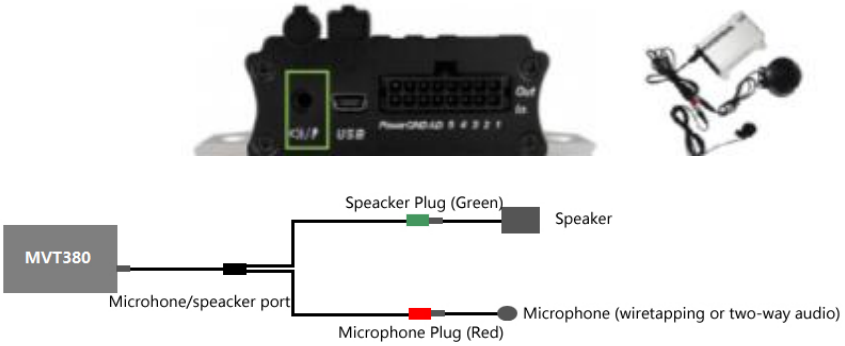

Figura 2.15 Instalando parlantes y microfono.

Fuente: Manual Meitrack

2.3 CONFIGURACIÓN DEL GPS TRACKER.

2.3.1. CONFIGURANDO CON UNA COMPUTADORA.

Aquí se muestra las partes básicas para usar los parámetros de edición MEITRACK.

Nota: No conectar el MVT380 a una batería externa mientras se configura.

Para instalar el driver del cable USB ejecuta 'PL2303_Prolific_DriverInstaller', que está en la carpeta "USB-232 Driver" del Cd.

Conectar el cable USB entre el MVT380 y el PC.

Figura 2.16 Conexión por USB para configurar el MVT380.

Fuente: Manual Meitrack

Abrir el administrador de dispositivos del panel de control. Busca el dispositivo ‘Prolific USB-to-Serial Comm Port’ como en la imagen.

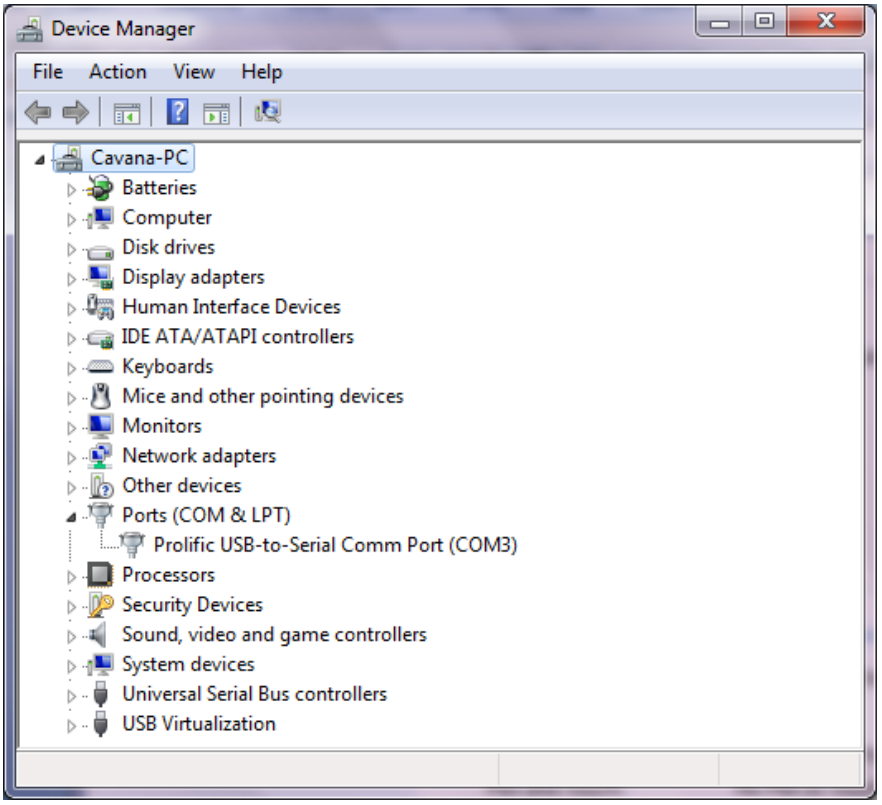

Figura 2.17 Administrador de dispositivos para ver el puerto COM asignado.

Fuente: Manual Meitrack

Recuerda el número del puerto COM, esto lo necesitas para ingresarlo en el editor de parámetros MEITRACK

Ejecuta Editor.exe para abrir:

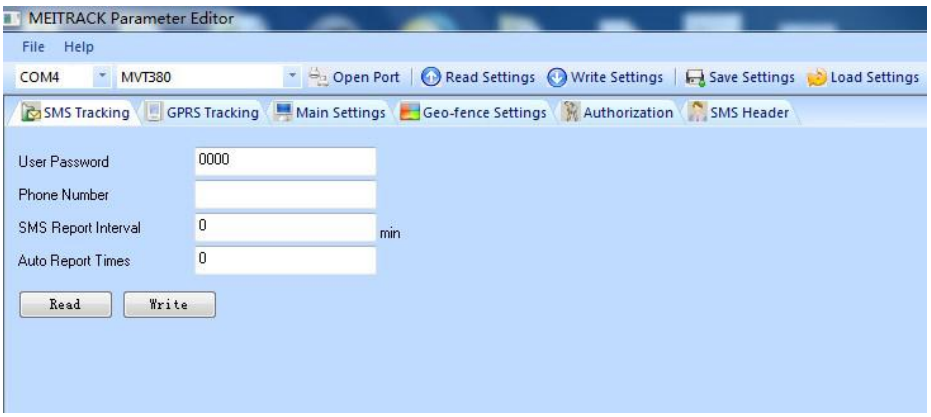

Figura 2.18 Pantalla para configurar el rastreador desde el PC

Fuente: Manual Meitrack

Selecciona el puerto correcto COM de acuerdo al administrador de dispositivos y presionar el botón Open port.

Selecciona el botón “Readsetting” para mostrar la configuración predeterminada o la configuración previa del rastreador.

2.3.1.1 RASTREO SMS

2.3.1.1.1. RASTREO POR INTERVALO

Hacer click en la pestaña: SMS Tracking

Tabla 4 Significado de los campos para configurar el MVT380

Ítem	Descripción
User Password	SMS password para envío de comandos SMS el predeterminado es 0000.

PhoneNumber	Un número donde se recibe los reportsSMS.
SMS ReportInterval	Rastreo por intervalo de tiempo vía SMS. Ingresa un intervalo de tiempo para el recibir reportes = 0, detiene el rastreo (default); = [1,65535], rastreo por intervalos en minutes.
Auto Report Times	= 0, tiempo ilimitado para reportes. = [1,255], se detendrá la presentación de informes al llegar el tiempo establecido.
Read	Leer las opciones establecidas del rastreador.
Write	Escribir los nuevos parámetros al rastreador.

2.3.1.1.2. REPORTE SMS

Click en la pestaña authorization

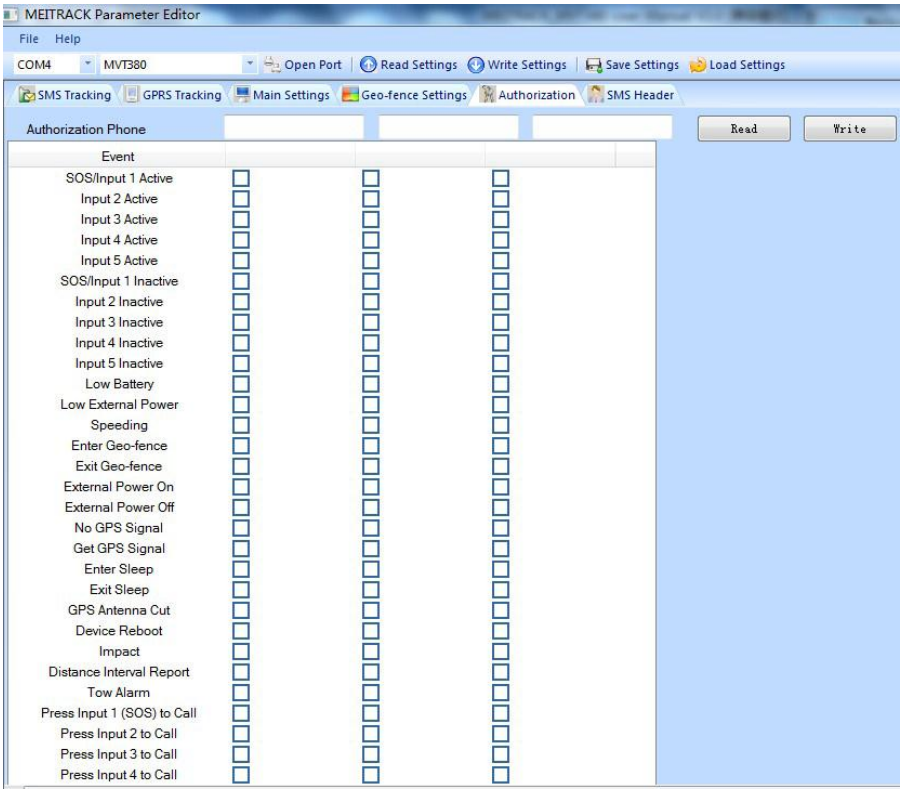

Figura 2.19 Pestaña autorización en el configurador de parámetros del MVT380

Fuente: Manual Meitrack

Tabla 5 Significado de los campos en la pantalla de autorización del MVT380

Item	Description
Authorization Phone	Un número para recibir los eventos de los reportes SMS
Event	Selecciona eventos para enviar a los números autorizados.

Read	Leer las opciones establecidas del rastreador.
Write	Escribir los nuevos parámetros al rastreador.

Descripción del Evento:

Tabla 6 Descripción de cada evento en la pantalla de autorización del MVT380

Evento	Descripción (Solo los números autorizados pueden recibir los eventos por SMS o llamadas)
SOS/Entrada 1 activa	Reporta cuando la entrada 1(SOS) es activada/presionada.
Entrada 2 activa	Reporta cuando la entrada 2 es activada.
Entrada 3 activa	Reporta cuando la entrada 3
Entrada 4 activa	Reporta cuando la entrada 4
Entrada 5 activa	Reporta cuando la entrada 5
SOS/entrada1 inactiva	Reporta cuando la entrada 1 está inactiva

Entrada 2 inactiva	Reporta cuando la entrada 2 está inactiva
Entrada 3 inactiva	Reporta cuando la entrada 3 está inactiva
Entrada 4 inactiva	Reporta cuando la entrada 4 está inactiva
Entrada 5 inactiva	Reporta cuando la entrada 5 está inactiva
Batería baja	Reporta cuando la batería está por debajo de 3.5V.
Poder externo bajo	La batería del carro esta baja del voltaje predefinido, tú lo puedes definir en la configuración principal.
Velocidad	Reporta cuando el MVT380 supera la velocidad definida.
Entrar Geo cercas	Reporta cuando MVT380 entra en la geo cerca.
Salir Geo cercas	Reporta cuando el MVT380 sale de la geo cerca.
Encendido externo	Alarma cuando la energía proporcionada externamente es encendida o recobrada.
Apagado externo	Alarma cuando la energía

	proporcionada externamente es apagada o cortada.
No hay señal GPS	Reporta cuando el MVT
Coger señal GPS	Reporta cuando el MVT380 sale del área blindada y capta señal GPS.
Modo Sleep	Reporta cuando el MVT380 entra en modo sleep
Salir modo sleep	Reporta cuando el MVT380 sale del modo sleep
Desconecta antena GPS	Alarma cuando la antena del GPS es desconectada.
Reinicio de dispositivo	Reporta cuando el MVT380 es reiniciado.
Hearbeat Report	Enciende el reporte de vibración Tu puedes definir el intervalo
Cambio de titulo	Reporte automático cuando el MVT380 cambia de dirección sobre el ángulo definido.
Intervalo de distancia	Rastreo por distancia.
Alarma de movimiento	Envía alarma cuando el rastreador vibra en un periodo

	de tiempo.
Presiona entrada 1 (SOS) para llamar	Hace una llamada al número autorizado cuando presiona la entrada 1 (SOS).
Presiona entrada 2 para llamar	Hace una llamada al número autorizado cuando presiona la entrada 2.
Presiona entrada 3 para llamar	Hace una llamada al número autorizado cuando presiona la entrada 3.
Presiona entrada 4 para llamar	Hace una llamada al número autorizado cuando presiona la entrada 4.
Presiona entrada 5 para llamar	Hace una llamada al número autorizado cuando presiona la entrada 5.
Rechazar llamadas entrantes	Contesta automáticamente las llamadas hechas del número autorizado
Reporte de localización por llamada	Rechaza las llamadas entrantes o reportes SMS cuando no contesta
Contestador automático de	Contesta automáticamente la llamada para conversar

llamadas entrantes	
Escucha (monitoreo de voz)	Contesta automáticamente la llamada para monitorear lo que está pasando en el vehículo.

2.3.1.1.3. RASTREO POR GPRS

Hacer click en la segunda pestaña GPRS tracking.

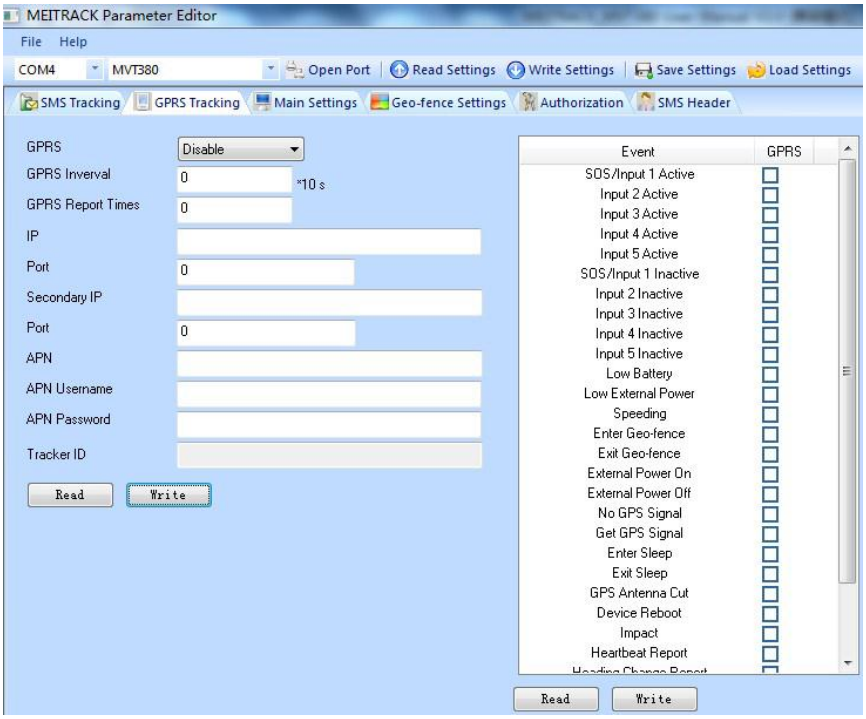

Figura 2.20 Pestaña GPRS Tracking

Fuente: Manual Meitrack

Tabla 7 Descripción de cada campo en la pantalla de GPRS Tracking del MVT380.

Item	Descripción
GPRS	Selecciona TCP/UDP para habilitar la comunicación GPRS.
GPRS Interval	Rastrea por intervalo de tiempo vía GPRS. Ingresar el intervalo de tiempo para rastreo por GPRS. El intervalo es en unidades de 10 segundos. Intervalo = 0, detiene el rastreo. Max interval de tiempo = 65535*10 segundos
GPRS Report Times	= 0, reporte sin límite. = [1,65535], ingresa el reporte del tiempo, el MVT380 puede parar de enviar el reporte cuando ya lo ha enviado muchas veces.
IP & Port	Ingresa la dirección IP y numero de puerto.

Secondary IP and Port (Backup Server)	Ingresa la dirección IP secundaria del servidor y número de puerto en caso de que se caiga el servidor principal.
APN APNusername APN password	APN, APN username, APN password: max 32 bytes. Si no se requiere user y password dejarlos en blanco.
Tracker ID	MVT380's IMEI. Este es el ID en el paquete GPRS.
Read	Lee la configuración actual del rastreador.
Write	Escribe la nueva configuración en el rastreador.

Descripción del Evento:

Tabla 8 Descripción de cada evento en la pantalla de GPRS Tracking del MVT380.

Evento	Descripción (Solo los números

	autorizados pueden recibir los eventos por SMS o llamadas)
SOS/Entrada 1 activa	Reporta cuando la entrada 1(SOS) es activada/presionada.
Entrada 2 activa	Reportacuando la entrada 2 es activada.
Entrada 3 activa	Reporta cuando la entrada 3
Entrada 4 activa	Reporta cuando la entrada 4
Entrada 5 activa	Reporta cuando la entrada 5
SOS/entrada1 inactiva	Reporta cuando la entrada 1 está inactiva
Entrada 2 inactiva	Reporta cuando la entrada 2 está inactiva
Entrada 3 inactiva	Reporta cuando la entrada 3 está inactiva
Entrada 4 inactiva	Reporta cuando la entrada 4 está inactiva
Entrada 5 inactiva	Reporta cuando la entrada 5 está inactiva

Batería baja	Reporta cuando la batería está por debajo de 3.5V.
Poder externo bajo	La batería del carro esta baja del voltaje predefinido, tú lo puedes definir en la configuración principal.
Velocidad	Reporta cuando el MVT380 supera la velocidad definida.
Entrar Geo cercas	Reporta cuando MVT380 entra en la geo cerca.
Salir Geo cercas	Reporta cuando el MVT380 sale de la geo cerca.
Encendido externo	Alarma cuando la energía proporcionada externamente es encendida o recobrada.
Apagado externo	Alarma cuando la energía proporcionada externamente es apagada o cortada.
No hay señal GPS	Reporta cuando el MVT
Coger señal GPS	Reporta cuando el MVT380 sale del área blindada y

	capta señal GPS.
Modo Sleep	Reporta cuando el MVT380 entra en modo sleep
Salir modo sleep	Reporta cuando el MVT380 sale del modo sleep
Desconecta antena GPS	Alarma cuando la antena del GPS es desconectada.
Reinicio de dispositivo	Reporta cuando el MVT380 es reiniciado.
HearbeatReport	Enciende el reporte de vibración Tu puedes definir el intervalo
Cambio de titulo	Reporte automático cuando el MVT380 cambia de dirección sobre el ángulo definido.
Intervalo de distancia	Rastreo por distancia.
Alarma de movimiento	Envía alarma cuando el rastreador vibra en un periodo de tiempo.

[**9**]

2.4 PROTOCOLOS QUE MANEJA EL GPS TRACKER.

El rastreado GPS maneja dos protocolos para comunicarse estos son SMS o mensajes de texto y el protocolo GPRS.

2.4.1. MENSAJES SMS.

SMS de sus siglas en ingles Short Message Service, es el servicio de mensajes cortos, es una prestación en los celulares que admite enviar mensajes cortos o mensajes de texto entre celulares, esto lo creó el finlandés, Matti Makkonen, que trabajó para Nokia, quién también inventó la tecnología GSM en el año 1985. En la actualidad se sigue utilizando incluso en la tecnología 4G.

2.4.2. PROTOCOLO GPRS.

General Packet Radio Service (GPRS) es un servicio de datos móvil orientado a paquetes en el sistema global de sistema de comunicación celular 2G y 3G para las comunicaciones móviles (GSM). GPRS se estandarizó originalmente por European Telecommunications Standards Institute (ETSI) en respuesta temprana a la CDPD (Cellular Digital Packet Data) y i-mode conmutación de paquetes de tecnologías celulares. Ahora es mantenido por el 3rd Generation Partnership Project (3GPP).

El tráfico GPRS está normalmente cargado en base al volumen de datos transferidos, que contrastan con conmutación de circuitos de datos, que normalmente se facturan por minuto de tiempo de conexión.

GPRS es un servicio de mejor esfuerzo, lo que implica rendimiento variable y depende de la latencia que el número de usuarios que comparten el servicio al mismo tiempo, a diferencia de la conmutación de circuitos, donde una cierta calidad de servicio (QoS) está garantizada durante la conexión. En los sistemas 2G, GPRS ofrece velocidades de datos de 56 a 114 kbit /segundo.

2.5 COMANDOS SMS DEL GPS TRACKER.

PROTOCOLO SMS MEITRACK®

Sirve para los rastreadores de modelos MT90 MVT340/MVT380/MVT100/MVT600/T1/T3 TC68/TC68S/MVT800

2.5.1. FORMATOS DE LOS COMANDOS

2.5.1.1. FORMATO SMS

Desde el teléfono móvil (o modem SMS) al rastreador:

Password,<commando>,<datos>

Nota:

Password es de solo 4 dígitos por defecto es 0000

Desde el rastreador al celular (o SMS modem):

(1) SMS Get

IMEI, <commando>,<datos>

(2) Reporte de localización

SMS header,<->yymmddHHMMSS,Z,G,valocidad,

bateria,Googlemap link

SMS Example:

Now,110727,02:48,V,16,23Km/h,61%,http://maps.googl

e.com/maps?f=q&hl=en&q=22.540103,114.082329&ie=

UTF8&z=16&iwloc=addr&om=1

Descripción de datos SMS:

Tabla 9 Descripción de datos SMS

Parámetros	**Descripción**	**Ejemplo**
Cabecera	Tipo de reporte SMS,	Now

SMS	incluyendo reporte normal o reporte de alarma. Favor referir abajo: Descripción de código de evento y cabecera SMS	Quiere decir el reporte de localización real.
yymmddHH MMSS	Formato YYMMDD hh:mm Yy= año Mm = mes Dd = dia HH = hora MM = minuto SS = segundos	150705 16:40 = hora: 16:40, 5 de julio, 2015
Z	Indicador del estatus del GPS: A = valido, V = invalido	A = Valido
G	Señal de GSM. Decimal (0 ~31).	12 = GSM señal:12

	Si el valor >16, GPRS es seguro enviar correctamente.	
Velocidad	Km/h. Decimal.	56
Reserva de batería	Nivel de batería (porcentaje)	97% =nivel de batería: 97%
Link de Google Map	Link de Google Maps con latitud y longitud que puedes hacer click y visitarlo directamente desde tu teléfono. Si tu móvil no puede visitar sitios HTTP, ingresa la latitud y longitud en Google Maps. (maps.google.com)	Now,110727,02:48,V,16,23Km/h,61%,http://maps.google.com/maps?f=q&hl=en&q=22.540103,114.082329&ie=UTF8&z=16&iwloc=addr&om=1 lat=22.513015 long= 114.057235

2.5.1.2. DESCRIPCIÓN DE CÓDIGOS DE EVENTOS Y CABECERA SMS.

Tabla 10 Descripción de códigos de eventos y cabecera sms

Código de evento	Evento	Cabecera SMS (max 16 bytes)	Bandera GPRS	Bandera SMS	Bandera de imagen
1	Entrada 1 activa (Botón de SOS)	SOS	Y	Y (Solo para el primer número autorizado)	Y
2	Entrada 2 activa	Ln2	Y	N	N
3	Entrada 3 activa	Ln3	Y	N	N
4	Entrada 4 activa	Ln4	Y	N	N
5	Entrada 5 activa	Ln5	Y	N	N
9	Entrada 1		N	N	N

	inactiva (SOS libre)				
10	Entrada 2 inactiva		N	N	N
11	Entrada 3 inactiva		N	N	N
12	Entrada 4 inactiva		N	N	N
13	Entrada 5 inactiva		N	N	N
17	Batería baja	LowBatter y	N	N	N/A
18	Bajo poder externo	LowPowe r	N	N	N/A
19	Speeding	Velocidad	Y	Y	N
20	Enter Geo-fence	Entras geo cercas	Y	Y	N
21	Exit Geo-fence	Salir Geo cercas	Y	Y	N
22	Encendido externo	Power ON	N	N	N
23	Apagado externo	Power Off	N	N	N/A
24	No GPS Signal	No Fix	N	N	N/A

25	Get GPS Signal	Fix	N	N	N/A
26	EnterSleep	EnterSleep	N	N	N/A
27	ExitSleep	ExitSleep	N	N	N/A
28	GPS AntennaCut	AntennaCut	N	N	N
29	DeviceReboot	Reboot	N	N	N/A
30	Impact	Impact	Y	N	N
33	DistanceIntervalReport	Distance	Y	N	N/A
34	CurrentLocationReport	Now	A/A	A/A	N/A
35	Time IntervalReport	Interval	A/A	A/A	N/A
36	TowAlarm	Tow	Y	N	N
65	Press Input 1 (SOS) to Call	/	N/A	N	N/A
66	Press Input 2 toCall	/	N/A	N	N/A
67	Press Input 3	/	N/A	N	N/A

	toCall				
68	Press Input 4 toCall	/	N/A	N	N/A
69	Press Input 5 toCall	/	N/A	N	N/A
70	RejectIncomingCall	/	N/A	Y	N/A
71	Report Location after receiving an incoming call	/	N/A	Y	N/A
72	Auto AnswerIncoming Call	/	N/A	Y	N/A
73	Listen-in (voicemonitoring)	/	N/A	Y	N/A

Notas:

1) Sobre las figuras todas son configuraciones de fábrica.
2) Y = yes; N = no; N/A = no aplica o no disponible; A/A = Siempre disponible y no puede ser cambiado.
3) Tú puedes usar comandos SMS para definir las cabeceras, añadir o borrar banderas de cada función.

2.5.2. LISTA DE COMANDOS.

Tabla 11 Lista de comandos SMS

Comandos	Definición	SMS/GPRS	Modelo aplicable
A00	Track bajo demanda	SMS	All
A02	Track por interval de tiempo	SMS	All
A10	Track por longitud y latitud	SMS/GPRS	All
A12	Track por interval de tiempo	GPRS	All
A14	Track por interval de distancia	SMS/GPRS	All
A15	Track parqueo por intervalo de tiempo	GPRS	MVT100/340/380/600/800/T1/T3
A16	Track parqueo por intervalo de tiempo on/off	GPRS	MVT100/340/380/600/800/T1/T3
A21	Set GPRS	SMS/GPRS	All
A22	Set DNS Server IP	SMS/GPRS	All
A23	Set secondary GPRS Server	SMS/GPRS	All

A70	Get all Autorized Phone number	SMS/GPRS	All
A71	Set Multiple Functions Phone Number	SMS/GPRS	All
A72	Set Listen-in Phone Number	Call	All
A73	Set Sleep Mode	SMS/GPRS	All
B00	Get Authorized Phone Number and SMS Event Flag	SMS/GPRS	All
B01	Authorize Phone Number and SMS Event Flag	SMS/GPRS	All
B02	Add SMS Event Flag to Authorized Phone Number	SMS/GPRS	All
B03	Delete SMS Event Flag from Authorized Phone	SMS/GPRS	All
B05	Set Geo-fence Alarm	SMS/GPRS	All
B06	Delete Geo-fence Waypoint	SMS/GPRS	All
B07	Set Speeding Alarm	SMS/GPRS	All
B08	Set Tow Alarm	SMS/GPRS	MVT100/340/380/600/800/T1/T3/MT90/TC68/TC68S
B09	Set Tremble Sensitivity (MVT100/MVT340/MVT380)	SMS/GPRS	MVT100/340/380/T1/T3

B20	Set Tremble Sensitivity (MVT600)	SMS/GPRS	MVT600
B21	Set Anti-theft	SMS/GPRS	MVT100/340/380/600/800/T1/T3
B31	Set Extended Functions	SMS/GPRS	All
B34	Set Log Interval	SMS/GPRS	MT90/ MVT100/380/600/800/T1/T3/TC68/TC68S
B35	Time Zone Setting(For SMS Report)	SMS/GPRS	All
B36	Time Zone Setting(For GPRS Report)	SMS/GPRS	All
B91	Set SMS Header for Event	SMS/GPRS	All
C01	Output Control	SMS/GPRS	MVT100/340/380/600/800/T1/T3
C03	Protocol Control	SMS/GPRS	All
C04	Set Delivery Mode of GPRS Buffer	GPRS	MT90,MVT100/380/600/800/T

			1/T3/TC6 8/TC68S
C11	SMS Message Display	SMS	MVT600/T 1/T3
E91	Get Firmware Version and SN	SMS/GPRS	All
F01	Reboot GSM Module	SMS/GPRS	All
F02	Reboot GPS Module	SMS/GPRS	All
F06	Clear Journey and Running Time	SMS/GPRS	All
F08	Set Mileage and Running Time	SMS/GPRS	All
F09	Clear SMS/GPRS Buffer	SMS/GPRS	MT90,MV T100/380/ 600/800/T 1/T3/TC6 8/TC68S
F11	Initialization	SMS/GPRS	All
F20	Change Password	SMS	All
FAB	Initialize Password	SMS	All

2.5.3. DETALLES DE LOS COMANDOS.

a) Rastreo bajo demanda – A00

SMS Set:	0000, A00
SMS Get:	Now,Date&Time,PositionStatus,GSM Signal Level,Speed,BatteryLife,IPlink
Descripción:	Toma el reporte de localización del rastreador.

Ejemplo

SMS Tx:	0000,A00
SMS Rx:	Now,110721 16:40,V,12,56Km/h,97%,http://maps.meigps.com/?lat=22.513015&lng=114.057235

b) Rastreo por intervalo de tiempo (SMS) _ A02

SMS Set:	0000,A02,interval,tiempo
SMS Get:	IMEI,A02,OK
Descripción:	Intervalo= 0, detiene el rastreo por intervalo de tiempo; Intervalo= [1,65535], rastrea por intervalo en minutos. Tiempo = 0, rastrea por intervalo continuo; Tiempo = [1,255], define cuantas veces

recibe SMS en un intervalo de tiempo

Ejemplo

SMS Tx: 0000,A02,10,3

SMS Rx: 353358017784062,A02,OK

En este ejemplo recibe 3 SMS en un intervalo de tiempo de 10 minutos.

Interval,12080918:45,V,16,0Km/h,67%,http://maps.google.com/maps?f=q&hl=en&q=3.342336,0.000500&ie=UTF8&z=16&iwloc=addr&om=1

c) Rastreo por Longitud y latitud (SMS) _ A10

SMS Set: 0000,A10

SMS Get: Now,,<->Latitude,<->Longitude,Date and Time,Status,Quantity of Satellites ,GSM Signal Level,Velocity,Direction,Positioningaccuracy,Altitude,Mileage,Time,,The Status of Input and Output,,

Descripción: Mira la ubicación actual, toma los datos de longitud y latitud. A10, si el GPRS trabaja y los parámetros son correctos, el rastreador puede enviar los datos de ubicación al servidor.
Este comando es para usuarios de rastreo por SMS.

Ejemplo

SMS Tx: 0000,A10

SMS Rx: 353358017784062,Now,22.535888,114.063034,080310161834,A,9,27,30,179,0,15,8890,1346,,0000,,

d) Rastreo por intervalo de tiempo (SMS) _ A12

SMS Set: 0000,A12,interval,tiempo

SMS Get: IMEI,A12,OK

Descripción: Intervalo es en unidades de 10 segundos.
Intervalo = 0, detiene el rastreo por

intervalo de tiempo.

Intervalo máximo de tiempo = 65535*10 segundos.

Tiempo = 0, rastreo por intervalo continuo;

Tiempo = [1,65535], ingresa cuantas veces (reporte) se rastrea con un intervalo especifico.

Ejemplo

SMS Tx: 0000,A12,6,0

SMS Rx: 353358017784062,A12,OK

e) Rastreo por intervalo de distancia - A14

SMS Set: 0000,A14,distancia

SMS Get: IMEI,A14,OK

Descripción: Distancia = 0, detiene el rastreo por intervalo de distancia;

Distancia = [1, 4294967295], ingresa intervalos de tiempo.

Si el rastreo es por intervalo de

distancia y rastreas por intervalo de tiempo a la misma vez, el reporte de ubicación del GPS cumple con primero en llegar es el primero en reportar, y el intervalo para el próximo reporte es re calculado inmediatamente.

Ejemplo

SMS Tx: 0000,A14,1000

SMS Rx: 353358017784062,A14,OK

En este ejemplo, se recibe un reporte si la distancia cambia en 1000 metros.
353358017784062,Distance,22.547278,114.047723,080310080934,A,7,21,30,88,1,12,8525,563,,0000,,

f) Rastreo de parqueo por intervalo de tiempo (SMS) - A15

SMS Set: 0000,A15,intervalo,tiempo.

SMS Get: IMEI,A15,OK

Descripción: Este comando es usado por

vehículos con rastreador GPS.

Ingresa el intervalo de tiempo, este es mejor porque reduce los tiempos de envíos de datos GPRS, para ahorrar datos.

Después de programar el comando A15, se programa automáticamente el A16. Para más detalles del status del motor (on/off), favor refiérase al comando A16.

El intervalo es en unidades de 10 segundos.

Intervalo = 0, detiene el intervalo de rastreo.

Máximo tiempo = 655356*10 segundos.

Tiempo= 0, Rastrea por intervalo continuo (Este es usando para rastreo en plataformas, configurado en 0)

Tiempo= [1,65535], puedes ingresar

el tiempo para rastrear con intervalo especifico.

Ejemplo

SMS Tx: 0000,A15,6,0

SMS Rx: 353358017784062,A15,OK

g) Rastreo de parqueo por intervalo de tiempo on/off(SMS) - A16

SMS Set: 0000,A16,Status

SMS Get: IMEI,A16,OK

Descripción: Este comando es usado por vehículos con rastreador GPS. La primera entrada positiva (HIGH) del rastreador del vehículo puede conectar en la detección del motor, o no puede trabajar. Lo siguiente es a entrada al primer positivo de los diferentes modelos de rastreadores:

Modelo Primera entrada positiva

MVT100	Entrada 2
MVT340	Entrada 2
MVT380	Entrada 4
MVT600	Entrada 3
T1	Entrada 3
T3	Entrada 3

Estatus = 1, los datos GPRS pueden ser enviados por intervalos de tiempo:

Motor On: Los datos GPRS pueden ser enviados por intervalos de tiempo en A12

Motor Off: Los datos GPRS pueden ser enviados por intervalos de tiempo en A15.

Estatus=0, los datos GPRS pueden ser enviados por intervalos de tiempo:

Motor On: Los datos GPRS pueden ser enviados por intervalos de

tiempo en A12

Motor Off: Los datos GPRS pueden ser enviados por intervalos de tiempo en A12.

Ejemplo

SMS Tx: 0000,A16,0

SMS Rx: 353358017784062,A16,OK

h) GPRS – A21

SMS Set: 0000,A21,X,IP,Port,APN,APNuser name,APN password

SMS Get: IMEI,A21,OK

Descripción: X = 0, Cierra GPRS;

X = 1, Abre TCP;

X = 2, Abre UDP.

IP: Dirección IP o nombre de dominio, max 32 bytes.

Port: max 5 bytes.

	APN / APN username / APN password: max 32 bytes c/u. Si no requiere usuario y clave, dejarlos en blanco.

Ejemplo

SMS Tx:	0000,A21,1,www.meiigps.com,8500,CMNET,,
SMS Rx:	353358017784062,A21,OK

i) Configurar DNS Server IP – A22

SMS Set:	0000,A22,DNS Server IP
SMS Get:	IMEI,A22,OK
Descripción:	Si tu servidor IP no está configurado correctamente con el comando A21 no trabaja. Tu puedes primero usar el comando para configurar el Server IP (Favor revisar con tu proveedor de DNS

por la IP Server) y re enviar el comando A21.

DNS Server IP: msx 16 bytes

Ejemplo

SMS Tx: 0000,A22,202.1.2.30

SMS Rx: 353358017784062,A22,OK

j) Configurar servidor GPRS secundario – A23

SMS Set: 0000,A23,IP,port

SMS Get: IMEI,A23,OK

Descripción: IP : max 32 bytes.

Port: max 5 bytes.

Cuando el primer servidor falla en el envío de datos configurado con el comando A21, podemos configurar un segundo servidor para el envío y que los datos no se pierdan.

Ejemplo

SMS Tx: 0000,A23,112.91.12.222,8500

SMS Rx: 353358017784062,A23,OK

k) Coger todos los teléfonos autorizados – A70

SMS Set: 0000,A70

SMS Get: IMEI,A70,SOS número de teléfono 1, SOS número de telefono2, SOS número de teléfono 3, escuchar en el número 1, escucha en el número 2.

Descripción: Coge todos los números autorizados.

Ejemplo

SMS Tx: 0000,A70

SMS Rx: 353358017784062,A70,13811111111 ,13822222222,13833333333,138444 44444,13855555555

l) Autoriza múltiples funciones para números telefónicos – A71

SMS Set: 0000,A71, número 1, número 2,

número3

SMS Get: IMEI,A71,OK

Descripción: Autoriza un numero para alarma SOS, llamadas para reporte de ubicación, alarmas de geo cercas, alarma de batería baja.

Numero teléfono: Max 16 caracteres.

Si no ingresar un número, el default queda vacío.

Envía comando “0000,A71” para borrar todos los números.

Cuando presionan el botón de SOS, el rastreador comienza a realizar llamadas a los números 1,2 y 3. Este deja de llamar en el momento que un número contesta.

Ejemplo

SMS Tx: 0000,A71,13811111111,13822222222,13833333333

SMS Rx: 353358017784062,A70,OK

m) Configura monitoreo de voz – A72

SMS Set: 0000,A72, número 1, número 2

SMS Get: IMEI,A72,OK

Descripción: Autoriza un número para realizar llamadas silenciosas. El rastreador contesta la llamada automáticamente y puedes escuchar que está pasando alrededor. En el vehículo no se escucha nada cuando se está llamando.

Número de teléfono: Max 2, 16 caracteres.

Si no ingresa un número telefónico este quedara vacío (default).

Envía el comando “0000,A72” para borrar todos los números.

Ejemplo

SMS Tx: 0000,A71,13811111111,13822222222

SMS Rx: 353358017784062,A72,OK

n) Configura modo dormir – A73

SMS Set: 0000,A73,X

SMS Get: IMEI,A73,OK

Descripció Esta configuración es para ahorro de

n: energía.

X = 0, apaga modo dormir (default)

X = 1, modo dormir normal. El módulo GSM trabaja, el modulo GPS trabaja pero entra en modo de intermitencia. El dispositivo puede trabajar 25% menos. Nota: esto no es recomendado para usuarios quienes configuran rastreo por intervalo o por intervalos cortos, porque esto podría afectar los datos del rastreador.

X =2, modo dormir profundo, el rastreador entre en este modo después de estar inactivo o estacionario (No SOS/desencadenamiento de un evento/entradas/llamadas entrantes/mensaje/movimiento) por 5 minutos. El modulo del GPS y GSM para de trabajar cuando entra en este modo. El rastreador se queda en este modo hasta que se active el botón SOS/desencadenamiento de un

evento/entradas/llamadas entrantes/mensaje/movimiento. Después el proceso se repite.

Nota: MT90 puede entra en modo dormir bajo movimiento, y los movimientos no despierta de este modo al MT90.

En cualquier condición, el dispositivo puede salir del modo dormir y regresar a la normalidad por comando SMS o GPRS y desactivar este modo.

Ejemplo

SMS Tx: 0000,A73,2

SMS Rx: 353358017784062,A73,OK

o) Coge los numero telefónicos autorizados y envía una bandera de evento SMS – B00

SMS Set: 0000,B00,P

SMS Get: IMEI,B00,P,Phone No, bandera de evento.

Descripción: P: 1 a 3

Phone No: max 16 caracteres. Si no se ingresa un número, no hay número autorizado.

Bandera de evento: 16+8 bytes, cadena HEX. Ver anexo 2 para más detalles.

Ejemplo

SMS Tx: 0000,B00,1

SMS Rx: 353358017784062,B00,1,13612345678, 0000000000000F0A00000000

p) Autoriza número de teléfono y bandera de evento

SMS – B01

SMS Set: 0000,B01,P,phone No,eventcode

SMS Get: IMEI,B01,OK

Descripción: P: 1 a 3

Phone No: max 16 caracteres.

Código de evento: Si no se estipula ningún código, cuando se aplique la bandera predeterminada para autorizar un número de teléfono.

Ver anexo 2 para más detalle de códigos de

evento y configuraciones de fábrica.

Ejemplo

SMS Tx: 0000,B01,1,13612345678,1

SMS Rx: 353358017784062,B01,OK

q) Añadir bandera de evento SMS para números autorizados – B02

SMS Set: 0000,B02,P,event code

SMS Get: IMEI,B02,OK

Descripción: P: 1 a 3

Código de evento: Ver anexo 2 para más detalle

Ejemplo

SMS Tx: 0000,B02,1,17

SMS Rx: 353358017784062,B02,OK

r) Borrar bandera de evento SMS para números autorizados – B03

SMS Set: 0000,B03,P,event code

SMS Get:	IMEI,B03,OK
Descripción:	P: 1 a 3 Código de evento: Ver anexo 2 para más detalle

Ejemplo

SMS Tx:	0000,B03,1,17
SMS Rx:	353358017784062,B03,OK

s) Configura alarma de Geo cercas – B05

SMS Set:	0000,B05,P,latitude,longitude,radius,in,out
SMS Get:	IMEI,B05,OK
Descripción:	P: 1 a 8. Max 8 puntos de Geo cercas para ingresar. Latitud: Se ingresa en grados decimal en el centro del punto; ingresar 6 decimales, añade ceros si es menor a 6 dígitos, para que no falle el comando. Longitud: Se ingresa en grados decimal en el centro del punto; ingresar 6

decimales, añade ceros si es menor a 6 dígitos, para que no falle el comando.

Radio: [1,4294967295] en metros.

In = 0, Apaga la alarma cuando el rastreador ingresa a la geo cerca.

In = 1, Prende la alarma cuando en rastreador ingresa a la geo cerca.

Out = 0, apaga la alarma cuando el rastreador sale de la geo cerca.

Out = 1, prende la alarma cuando el rastreador sale de geo cerca.

Ejemplo

SMS Tx: 0000,B05,1,
22.913191,114.079882,1000,0,1

SMS Rx: 353358017784062,B05,OK

Si el rastreador esta fuera del círculo, (centro:
22.913191,114.079882 y radio 1000 metros) el siguiente mensaje se recibe.

353358017784062,ExitGEO,22.918186,1
14.089823,080229123816,A,10,22,16,32,

1,21,6667,850,,0000,,

t) Borrar puntos de geo cercas – B06

SMS Set: 0000,B06,P

SMS Get: IMEI,B06,OK

Descripción: P: 1 a 8. Solo puede ser borrado un punto por comando SMS.

Ejemplo

SMS Tx: 0000,B06,P

SMS Rx: 353358017784062,B06,OK

u) Configura alarma de velocidad – B07

SMS Set: 0000,B07,velocidad

SMS Get: IMEI,B07,OK

Descripción: Speed = 0, cancela la alarma de velocidad (default);

Speed = [1,255], configura la velocidad limite en Km/h.

Ejemplo

SMS Tx: 0000,B07,60

SMS Rx: 353358017784062,B07,OK

En este ejemplo, el siguiente mensaje es recibido del rastreador si la velocidad es mayor a 60 Km/h.

353358017784062,Speeding,22.914891,114.087491,080229203703,A,10,22,55,44,1,24,6635,388,,0000,,

v) Configura alarma de movimiento – B08

SMS Set: 0000,B08,tiempo

SMS Get: IMEI,B08,OK

Descripción: Cuando el rastreador se mueve o tiembla sobre el tiempo definido, este puede enviar un reporte a los números autorizados. El modo dormir puede estar presente en nivel 2 a través del comando A73 y el valor de “Tremble Time” con el comando B08 antes de usar la alarma de remolque, de lo contrario la alarma no trabaja.

Tiempo = 0, cancel alarma de remolque (default);

Tiempo = [1,255], configura el tiempo en segundos.

Ejemplo

SMS Tx: 0000,B08,3

SMS Rx: 353358017784062,B08,OK

En este ejemplo, el siguiente mensaje es recibido del rastreador si tiembla por más de 3 segundos.

353358017784062,Tow,22.914891,114.087491,080229203703,A,10,22,55,44,1,24,6635,388,,0000,,

w) Configura sensibilidad de alarma de movimiento (MVT100/MVT340/MVT380/T1/T3) – B09

SMS Set: 0000,B09,sensibilidad

SMS Get: IMEI,B09,OK

Descripción: Sensibilidad = [1,65535], configura el

grado de sensibilidad para el sensor de movimiento.

El sensor de movimiento es más sensible si el grado es menor.

Default es 1.

Ejemplo

SMS Tx: 0000,B09,10

SMS Rx: 353358017784062,B09,OK

x) Configura sensibilidad de alarma de movimiento (MVT600) – B20

SMS Set: 0000,B20,sensibilidad

SMS Get: IMEI,B20,OK

Descripción: Sensibilidad = [1,65535], configura el grado de sensibilidad para el sensor de movimiento.

El sensor de movimiento es más sensible si el grado es menor.

Default es 5.

Ejemplo

SMS Tx: 0000,B20,10

SMS Rx: 353358017784062,B20,OK

y) Configura anti robo– B21

GPRS Set: B21,Status

GPRS Get: B21,OK

Descripción: Estatus = 1, configura anti robo (default); El dispositivo de la serie MVT se alarma cuando la primer entrada negativa y la primera entrada positiva son activadas excepto SOS.

Estatus = 0, cancela el anti robo; El dispositivo de la serie MVT se alarma cuando la primer entrada negativa y la primera entrada positiva son activadas excepto SOS.

Nota: Esta función solo es aplicada en rastreadores de la serie MVT. La relación entre las entradas y las salidas:

Modelo Entrada negativa Entrada positiva

MVT100 - Entrada 2

MVT340 - Entrada 2

MVT380 Entrada 2 Entrada 4

MVT600 Entrada 2 Entrada 3

T1/T3 Entrada 2 Entrada 3

Ejemplo

GPRS Set: 0000,B21,1

GPRS Get: 353358017784062,B21,OK

z) Configura funciones extendidas – B31

SMS Set: 0000,B31,AB

SMS Get: IMEI,B31,OK

Descripción: A = 0, todos las luces led trabaja normalmente (default);

A = 1, todos las luces led están apagadas cuando trabaja bien;

B, reservado.

Ejemplo

SMS Tx: 0000,B31,11

SMS Rx: 353358017784062,B31,OK

aa) Configura Log de intervalos – B34

SMS Set:	0000,B34,intervalo
SMS Get:	IMEI,B34,OK
Descripción:	Configura el intervalo para almacenar los datos del GOS en la memoria flash. Intervalo = 0, apaga el log (default); Intervalo = [1,65535], configura log por intervalos de segundos.
Ejemplo	
SMS Tx:	0000,B33,60
SMS Rx:	353358017784062,B34,OK

bb) Configura Zona horaria (SMS) – B35

SMS Set:	0000,B35,minuto
SMS Get:	B35,OK
Descripción:	El tiempo predeterminado del rastreador es en GMT, tú puedes usar este comando para corregir la

hora y fecha del rastreador.

Minuto = 0, GMT (default);

Minuto = [-32768,32767], ingresa el tiempo en minutos GMT.

La zona horaria para SMS es diferente por GPRS.

Ejemplo

SMS Tx: 0000,B35,480

SMS Rx: 353358017784062,B35,OK

cc) Configura Zona horaria (GPRS) – B36

SMS Set: 0000,B36,GPRS minuto

SMS Get: IMEI,B36,OK

Descripción: Minuto = 0, GMT (default); Nota: MS02 puede reconocer automáticamente la zona horaria de la computadora. Favor no cambies la zona horaria GPRS, este permanece por default en la zona horaria 0. Si cambiaste la hora, puede ocurrir una

discrepancia entre la hora del rastreador y la hora actual.

Minuto = [-32768,32767], ingresa el tiempo en minutos GMT.

Ejemplo

SMS Tx: 0000,B36,480

SMS Rx: 353358017784062,B36,OK

dd) Ingresa la cabecera SMS para eventos – B91

SMS Set: 0000,B91,codigo_evento,cabecera

SMS Get: IMEI,B91,OK

Descripción: Cabecera: max 16 bytes.

Favor ver Anexo 2 para más detalles.

Ejemplo

SMS Tx: 0000,B91,1,SOS

SMS Rx: 353358017784062,B91,OK

ee) Salidas de control – C01

SMS Set: 0000,C01,velocidad,ABCDE

SMS Get:	IMEI,C01,OK
Descripción:	Velocidad = 0, no hay límite; Velocidad = [1,255], en Km/h, ingresa velocidad limitada por la salida de control. Cuando la velocidad es por debajo de la ingresada, la salida es activada. A = 0, cierra salida (OUT1) – open drain; A = 1, abre salida (OUT1) – conecta a GND; A = 2, permanece status previo. B = 0, cierra salida (OUT2) – open drain; B = 1, abre salida (OUT2) – conecta a GND B = 2, permanece status previo. C = 0, cierra salida (OUT3) – open drain; C = 1, abre salida (OUT3) – conecta a GND;

C = 2, permanece status previo.

D = 0, cierra salida (OUT4) – open drain;

D = 1, abre salida (OUT4) – conecta a GND;

D = 2, permanece status previo.

E = 0, cierra salida (OUT5) – open drain;

E = 1, abre salida (OUT5) – conecta a GND;

E = 2, permanece status previo.

Ejemplo

SMS Tx: 0000,C01,20,12221

SMS Rx: 353358017784062,C01,OK

ff) Protocolo de control – C03

SMS Set: 0000,C03,X

SMS Get: IMEI,C03,OK

Descripción: X = 0, Reporte de auto eventos (default);

X = 1, El reporte de eventos necesita confirmación del servidor por comando AFF.

Ejemplo

SMS Tx: 0000,C03,0

SMS Rx: 353358017784062,C03,OK

gg) Configura modo de entrega de GPRS – C04

SMS Set: 0000,C04,X

SMS Get: IMEI,C04,OK

Descripción: X = 0, FIFO primero en entrar primero en salir (default), los datos son almacenados en cola.

X = 1, FILO primero en entrar, último en salir, los datos son almacenados por montón.

Ejemplo

SMS Tx: 0000,C04,1

SMS Rx: 353358017784062,C04,OK

hh) Mensaje SMS en el display – C11

SMS Set:	0000,C11,Txt
SMS Get:	IMEI,C11,OK
Descripción:	Mensaje del teléfono móvil puede mostrarse en la pantalla LCD. Txt: Contenido del mensaje. Puede ser un string ASCII, Max 150 Bytes.
Ejemplo	
SMS Tx:	0000,C11,SMS mensaje
SMS Rx:	353358017784062,C11,OK

ii) Configura versión de firmware y SN – E91

SMS Set:	0000,E91
SMS Get:	IMEI,E91,versión,SN
Descripción:	Toma la versión del firmware actual y detalles S/N del rastreador.
Ejemplo	

SMS Tx: 0000,E91

SMS Rx: 353358017784062,E91,FWV1.00 ,12345678

jj) Reiniciar módulo GSM – F01

SMS Set: 0000,F01

SMS Get: IMEI,F01,OK

Descripción: Reinicia módulo GSM.

Ejemplo

SMS Tx: 0000,F01

SMS Rx: 353358017784062,F01,OK

kk) Reiniciar modulo GPS – F02

SMS Set: 0000,F02

SMS Get: IMEI,F02,OK

Descripción: Reinicia modulo GPS.

Ejemplo

SMS Tx: 0000,F02

SMS Rx: 353358017784062,F02,OK

ll) Borra viaje y tiempo recorrido – F06

SMS Set:	0000,F06,X
SMS Get:	IMEI,F06,OK
Descripción:	Borra viaje y tiempo recorrido. X = 1, borra viaje X = 2, borra tiempo recorrido X = 3, borra viaje y tiempo recorrido.

Ejemplo

SMS Tx:	0000,F06,1
SMS Rx:	353358017784062,F06,OK

mm) Configura kilometraje y tiempo recorrido – F08

SMS Set:	F08, Running time, kilometraje
SMS Get:	F08,OK
Descripción:	Tiempo recorrido: [0, 4294967295], formato decimal, unidad en metros, no se puede establecer nulo.

Ejemplo

SMS Tx: @@D40,353358017784062,F0
8,0,4825000*51\r\n

SMS Rx: $$D28,353358017784062,F08,
OK*FA\r\n

nn) Borra buffer SMS/GPRS – F09

SMS Set: 0000,F09,X

SMS Get: IMEI,F09,OK

Descripción: X = 1, borra SMS buffer

X = 2, borra GPRS buffer

X = 3, borra SMS y GPRS buffer.

Ejemplo

SMS Tx: 0000,F09,1

SMS Rx: 353358017784062,F09,OK

oo) Inicializar – F11

SMS Set: 0000,F11

SMS Get: IMEI,F11,OK

Descripción: Regresan todos los valores de
fábrica excepto el password.

Ejemplo

SMS Tx:	0000,F11
SMS Rx:	353358017784062,F11,OK

pp) Cambiar clave – F20

SMS Set:	0000,F20,clave nueva
SMS Get:	IMEI,F20,OK
Descripción:	Cambia la clave por SMS, Clave de 4 dígitos.

Ejemplo

SMS Tx:	0000,F20,1234
SMS Rx:	353358017784062,F20,OK

qq) Cambiar clave – FAB

SMS Set:	8888,FAB
SMS Get:	IMEI,FAB,OK
Descripción:	Cambia la clave por el valor de fábrica en caso de pérdida de la clave. Solo el número autorizado

puede enviar este comando.

Ejemplo

SMS Tx: 8888,FAB

SMS Rx: 353358017784062,FAB,OK

[**10**]

CAPÍTULO 3

ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1 ANÁLISIS DE LOS REQUERIMIENTOS DE LA APLICACIÓN

Este sistema nos ayuda con la comunicación entre un rastreador satelital de la marca Meitrack instalado en un vehículo y el usuario final por medio de un teléfono celular inteligente con el sistema operativo Android.

Por medio de la App creada y a través de su interfaz gráfica con sencillos botones, se puede enviar al rastreador, comandos por mensajes de texto SMS que por su complejidad no sería fácil recordarlos. De la misma manera el rastreador envía respuestas o informes por mensajes de texto, el sistema es capaz de procesarlos y

mostrarlos en un mapa offline su coordenada GPS para su localización.

Para su correcto funcionamiento se debe descargar e instalar la App gratuita de la tienda Play Store de Google llamada MAPS.ME (MapsWithMe) con el respectivo mapa del país donde nos encontramos, si este aplicativo no se encuentra instalado en el celular, nuestro programa automáticamente nos pedirá descargarlo al momento de tratar mostrar una coordenada en pantalla.

Las coordenadas recibidas por parte del rastreador son almacenadas en una base de datos en el celular para su consulta posterior.

3.1.1 ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

3.1.1.1 REQUERIMIENTOS FUNCIONALES.

El rastreador satelital funciona con un chip de telefonía celular 2G, en cualquier banda o frecuencia, por lo que se le puede instalar el chip de cualquiera de las operadoras celulares del país. Para poder interactuar

con el usuario debe tener acreditado saldo para el envío de mensajes de texto.

Es recomendable ubicar bien las antenas de recepción en el vehículo en especial la del GPS que debe estar direccionada cerca del vidrio y no bloquearla o escondida en el chasis. La antena del celular por ser omnidireccional no tiene inconvenientes donde se la ubique, si por alguna razón en el sitio donde este el vehículo no se encuentre cobertura celular o GPS, y el usuario este haciendo un requerimiento, éste será enviado con la última ubicación registrada en el GPS y cuando vuelva a tener cobertura celular.

El rastreador satelital puede abrir las puertas del vehículo siempre y cuando tenga un sistema automático y centralizado de seguros.

3.1.1.2 REQUERIMIENTOS NO FUNCIONALES.

Este sistema está desarrollado para funcionar en un celular desde la versión 2.2 de Android, para su mejor

funcionamiento necesita de un procesador de doble núcleo, 1Gb en memoria ram y una tarjeta SD de 2Gb.

El rastreador satelital usado es de marca Meitrack modelo MVT380, utiliza una tarjeta SIM prepago de la operadora celular Claro.

La App fue desarrollada con Eclipse Luna sobre Windows 7, no se utilizó Android Studio por resultar muy pesado para el computador donde fue desarrollado, una portátil de procesador Intel Pentium Dual Core de 2 Ghz, con 4Gb de Ram, también se utilizó el JDK de Oracle versión 7.

Es importante tener en cuenta que para tener comunicación entre el rastreador satelital con el celular ambos deben tener saldo para el envío de mensajes de texto.

3.2 DISEÑO DEL MODELADO DE DATOS DE LA APLICACIÓN.

3.2.1. Diagrama de flujo de datos.

El sistema ha sido realizado para procesar solo los mensajes de texto entrantes pertenecientes al número de celular asignado al chip instalado en el rastreador satelital.

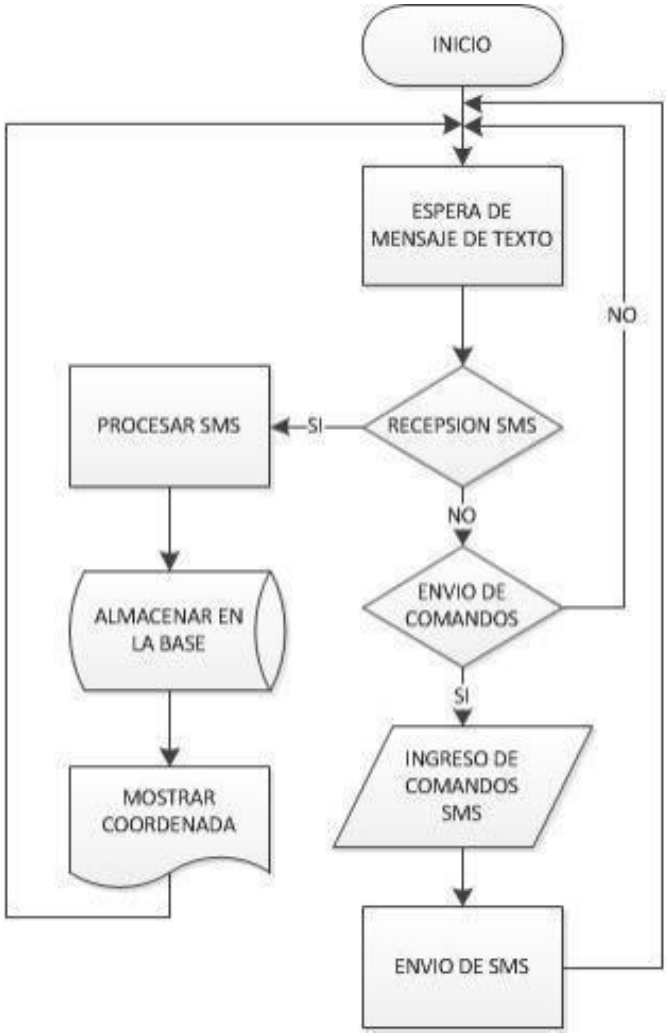

Figura 3.1 Diagrama de flujo de datos

Elaborado por: El autor

3.2.2 Diagrama de casos de usos.

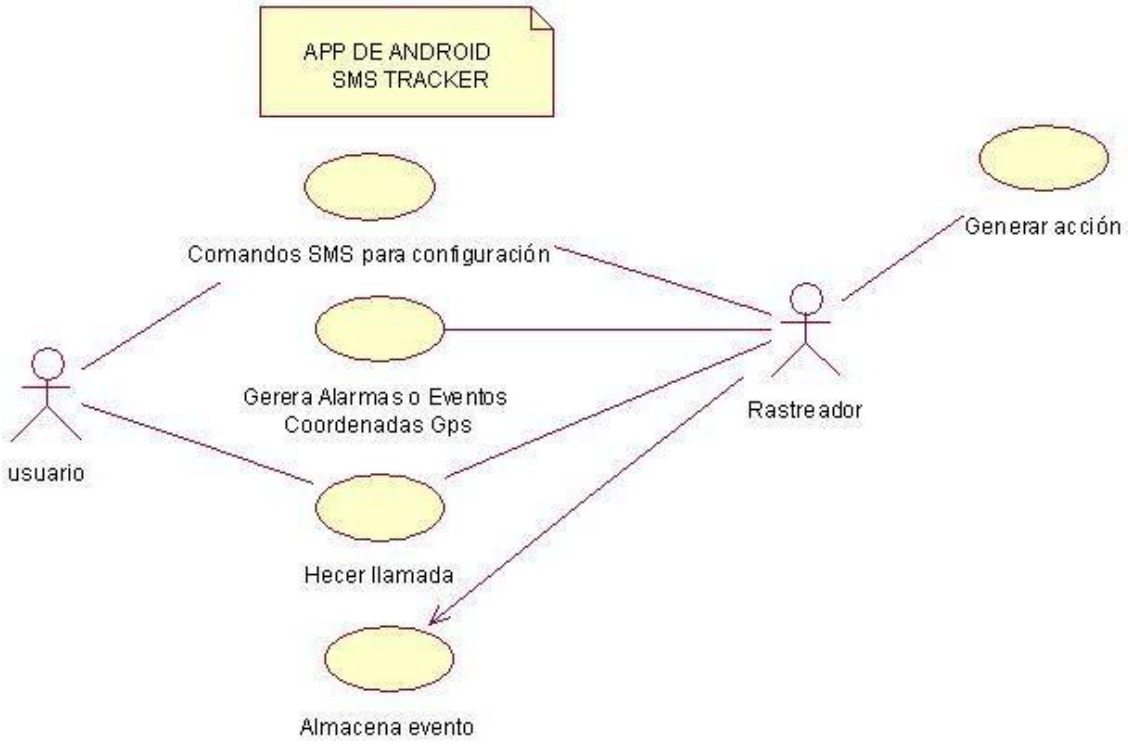

Figura 3.2 Diagrama de casos de uso.

Elaborado por: El autor

3.2.3. Diagrama de Entidad Relación.

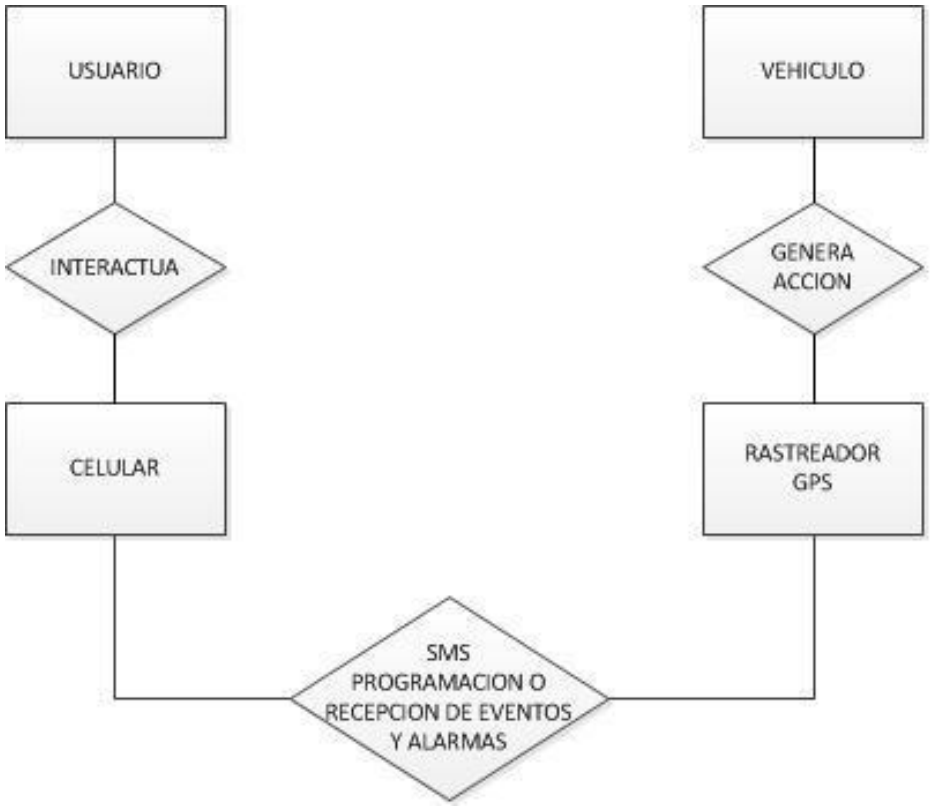

Figura 3.3 Diagrama entidad relación.
Elaborado por: El autor

CAPÍTULO 4

DESARROLLO Y PRUEBAS DE LA APLICACIÓN

4.1 INSTALACIÓN Y CONFIGURACIÓN DEL LENGUAJE DE PROGRAMACIÓN Y ENTORNO DE DESARROLLO.

El entorno de desarrollo para Android que se descargó y configuró fue el siguiente:

- **JDK para 32 bits:**http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

Para instalar el JDK 7 solo necesitamos hacer doble click sobre el archivo descargado jdk-7u79-windows-i586.exe, generalmente ponemos siguiente a todo y dejamos las direcciones predeterminadas que se instale.

- **Eclipse versión Luna para 32 bits** https://www.eclipse.org/luna/

  Para instalar el Eclipse Luna, debemos descargarlo son 158Mb tiene el nombre eclipse-java-luna-SR2-win32.zip, este archivo debemos descomprimirlo en la dirección de nuestra preferencia, eso es todo al descomprimirlo veremos el archivo ejecutable eclipse.exe, al momento de abrirlo por primera vez nos pedirá que configuremos el workspace que es el lugar donde eclipse guarda los proyectos creados dejamos la dirección por defecto y ponemos un visto para que no vuelva a aparecer el mensaje cada vez que abrimos, nos aparecerá una pantalla como la siguiente:

Figura 4.1 Pantalla de inicio de Eclipse.

- **Androit ADT**: http://developer.android.com/sdk/index.html#Other

Instalación de SDK tools de android, tomado del blog de MERINOX [**11**]:

1.- Ejecutamos Eclipse escogemos la pestaña Help – Install New Software.

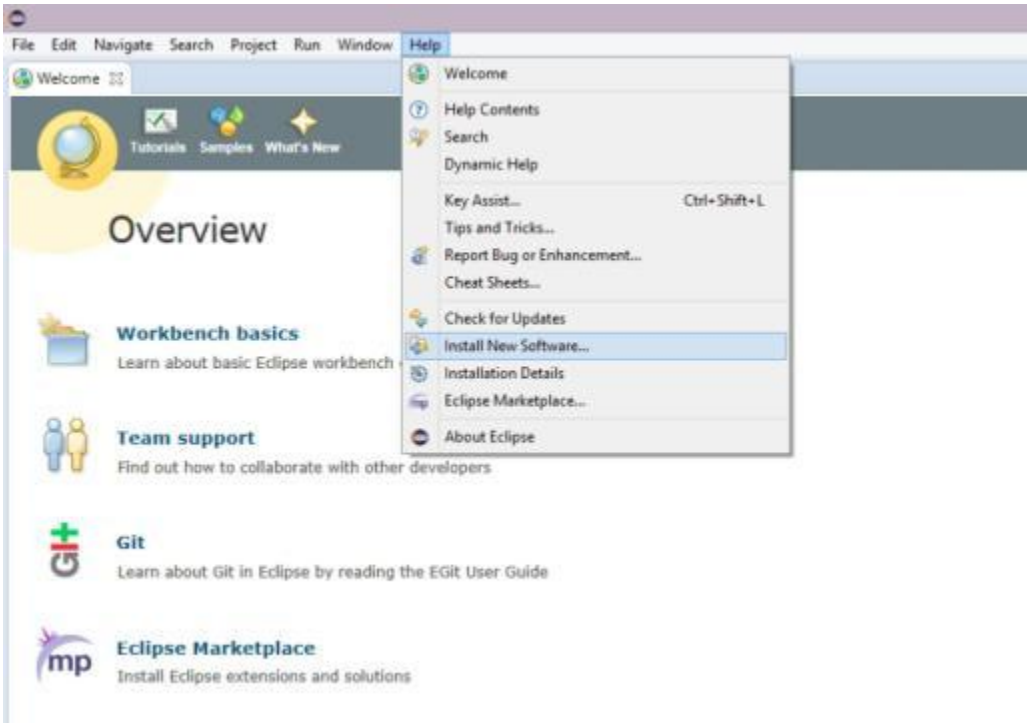

Figura 4.2 Ventana para agregar funcionalidades en Eclipse.

2.- Aparece una ventana donde vamos agregar funcionalidades a eclipse. Escogemos el Botón Add aparece una nueva ventana con el nombre Add Repository, instalamos las herramientas de desarrollo de Android (ADT), estas incluyen lo necesario para desarrollar aplicaciones Android; para esto ponemos en el cuadro *Name* un

nombre para identificar la instalación y en Location el link: https://dl-ssl.google.com/android/eclipse/

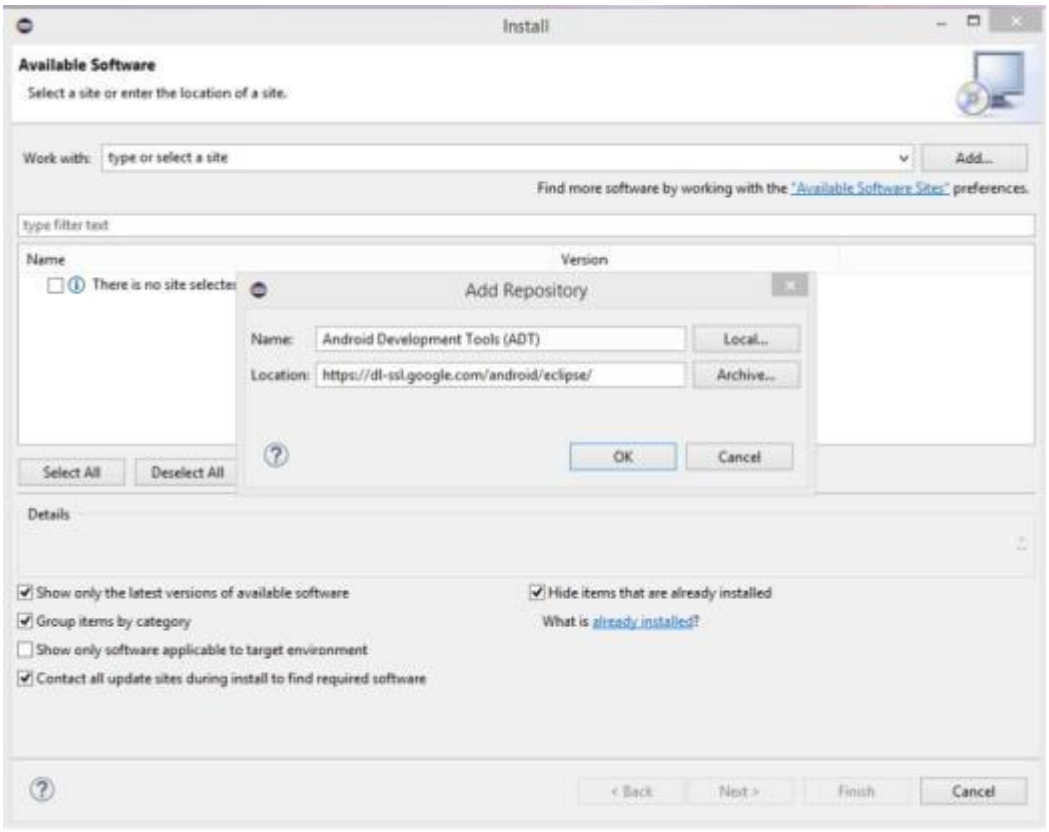

Figura 4.3 Añadir repositorio para instalar las herramientas de desarrollo en Eclipse.

Damos en OK y continuamos.

3.- Nos muestra una ventana con las herramientas de desarrollo disponibles para la descarga; seleccionamos todas y damos en Next

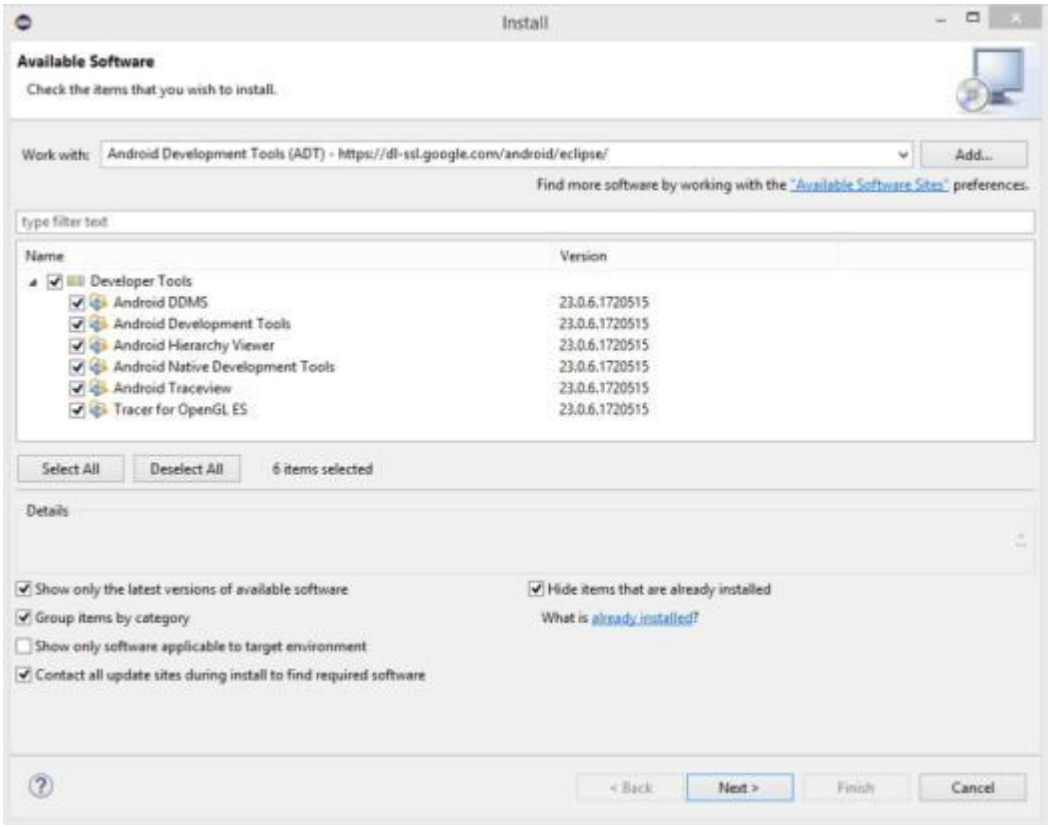

Figura 4.4 Selección de herramientas de desarrollo en Eclipse.

4.- Al hacer click en Next nos muestra los detalles de la instalación; volvemos a dar en Next

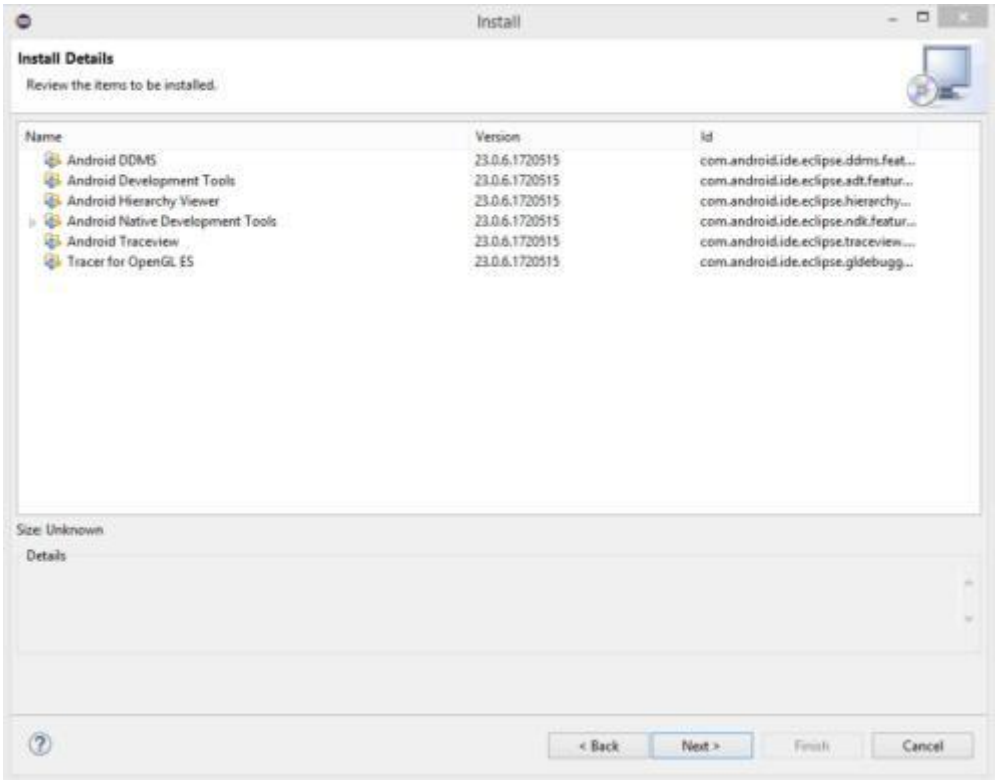

Figura 4.5 Detalles de la instalación de las herramientas de desarrollo en Eclipse.

5.- Nos muestra una ventana con la revisión de las Licencias del ADT de Android; aceptamos los Términos y damos en Finish

.

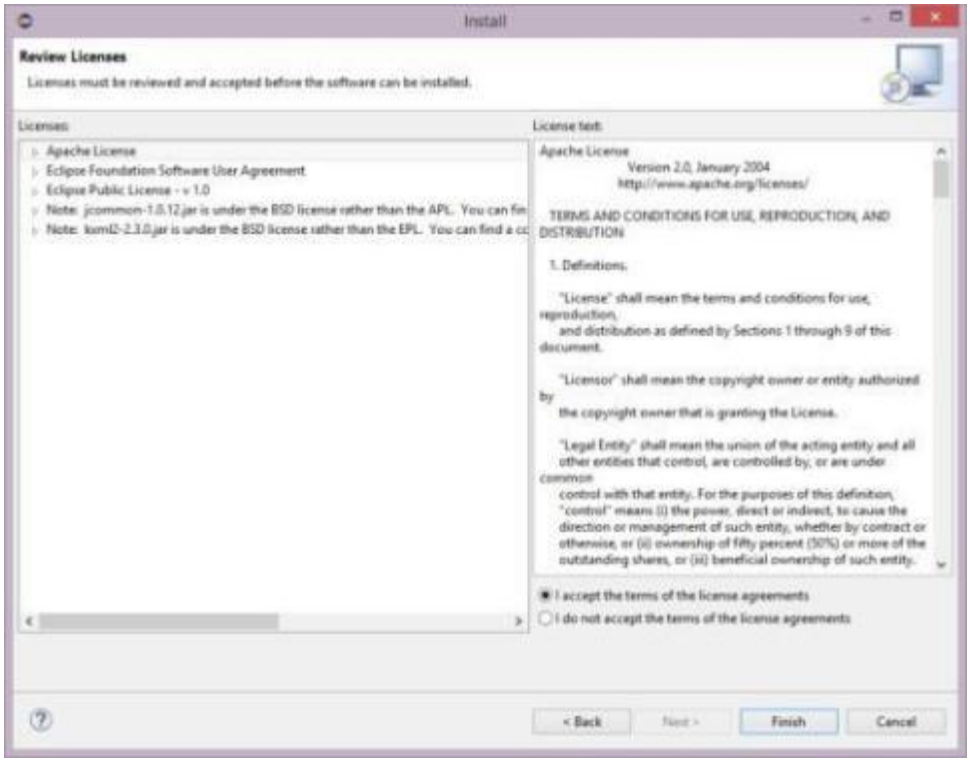

Figura 4.6 Revisión de las licencias del ADT de Android.

6.- Dejamos que termine de descargar e Instalar el ADT de Android.

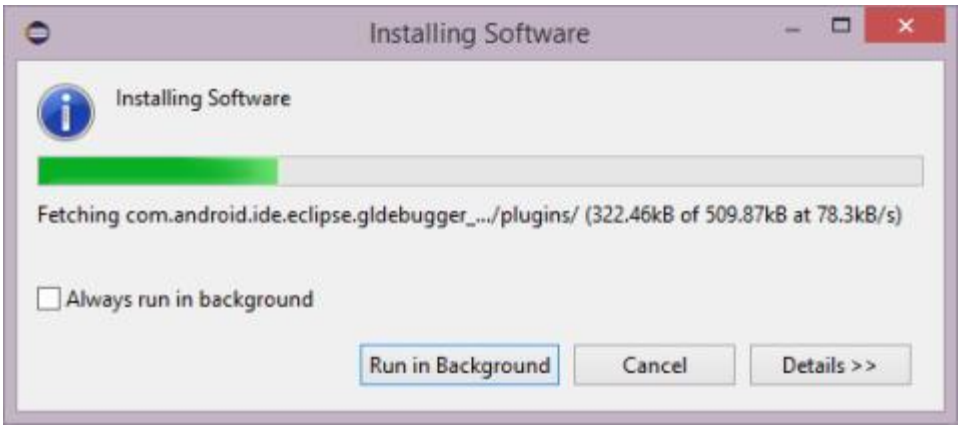

Figura 4.7 Instalando el software adicional.

Nos muestra una advertencia de seguridad damos en ok y continuamos.

Figura 4.8 Aviso de seguridad.

7.- Al terminar la instalación nos pide que se reinicie Eclipse, damos en YES y esperamos a que abra de nuevo.

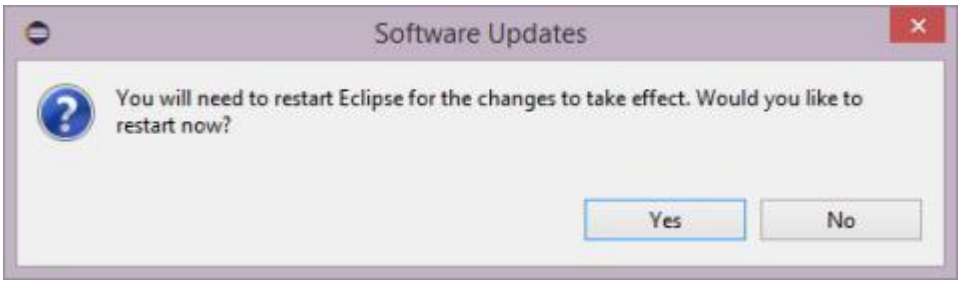

Figura 4.9 Aviso de reinicio de la aplicación de Eclipse.

8.- Al abrir eclipse muestra la ventana de configuración del SDK de Android; Seleccionamos Install New SDK y los dos vistos siguientes, dejamos la ubicación por defecto del *androids-sdks* y damos en Next

Figura 4.10 Pantalla de bienvenida para la instalación del SDK

9.- Google nos pide mandar estadísticas de uso elegimos No y dar en Finish.

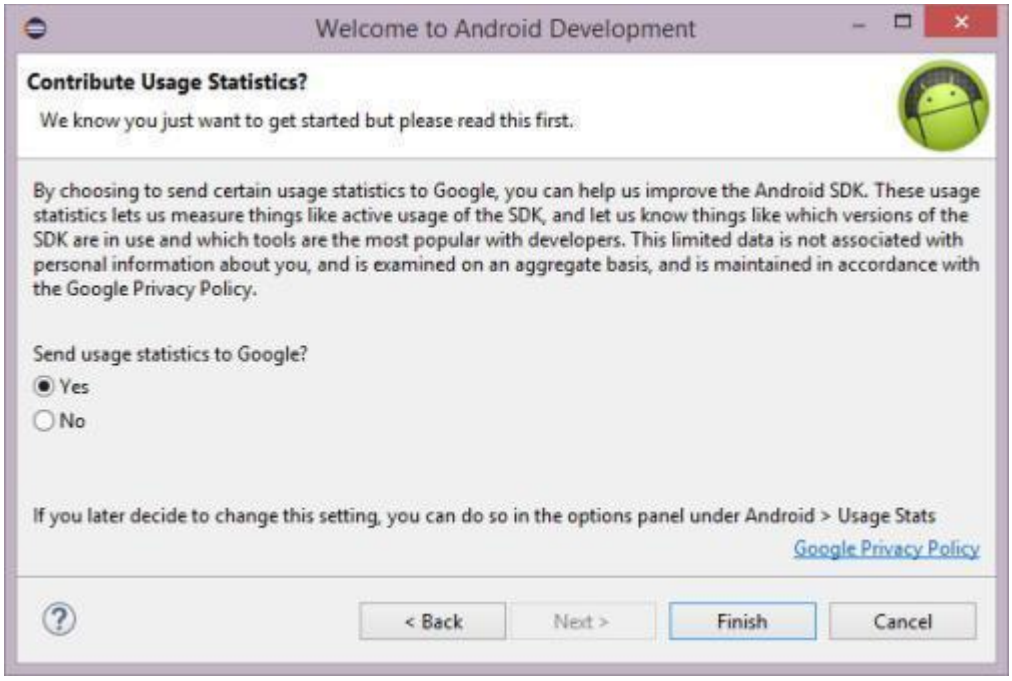

Figura 4.11 Envío de estadísticas de uso a Google

Aguardamos la descarga del SDK

Figura 4.12 Instalación del SDK

10.- Aceptar licencia de los paquetes del SDK y damos en Install

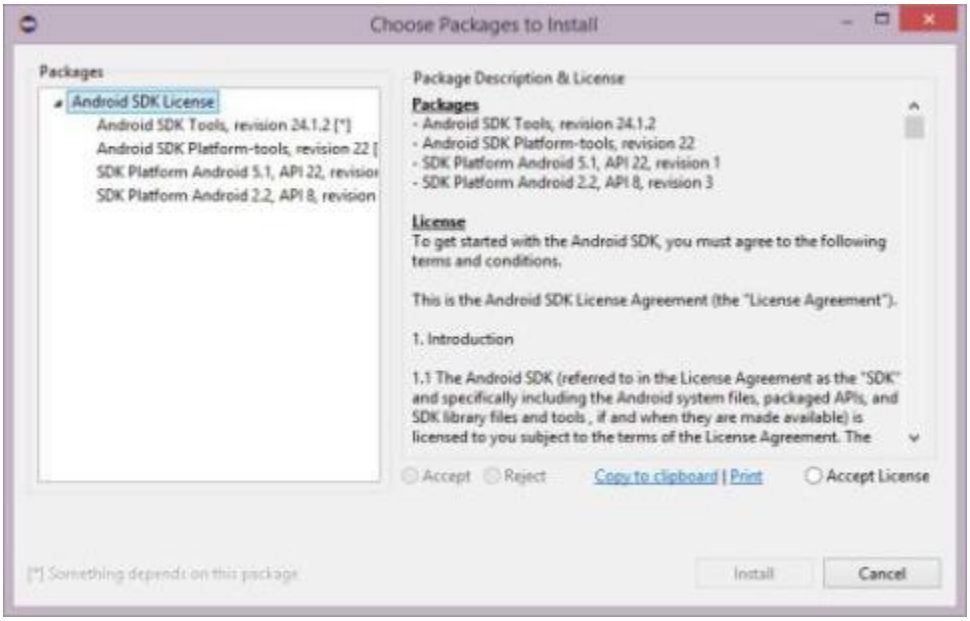

Figura 4.13 Aceptación de la licencia de los paquetes del SDK

11.- El SDK requiere que se descarguen versiones de android la cual queremos que corran nuestra aplicación, descargamos el paquete “Build-Tools” que nos hace falta.

Figura 4.14 Solicitud de instalación de componentes del SDK

Damos en Open SDK Manager y abre otra ventana en la que debemos seleccionar las API’s y el Android Build-tools 22.0.1. En mi caso

seleccione desde la API 8 de Android 2.2 que se puede correr en casi todos los dispositivos móviles con Android. Damos en Install Packages.

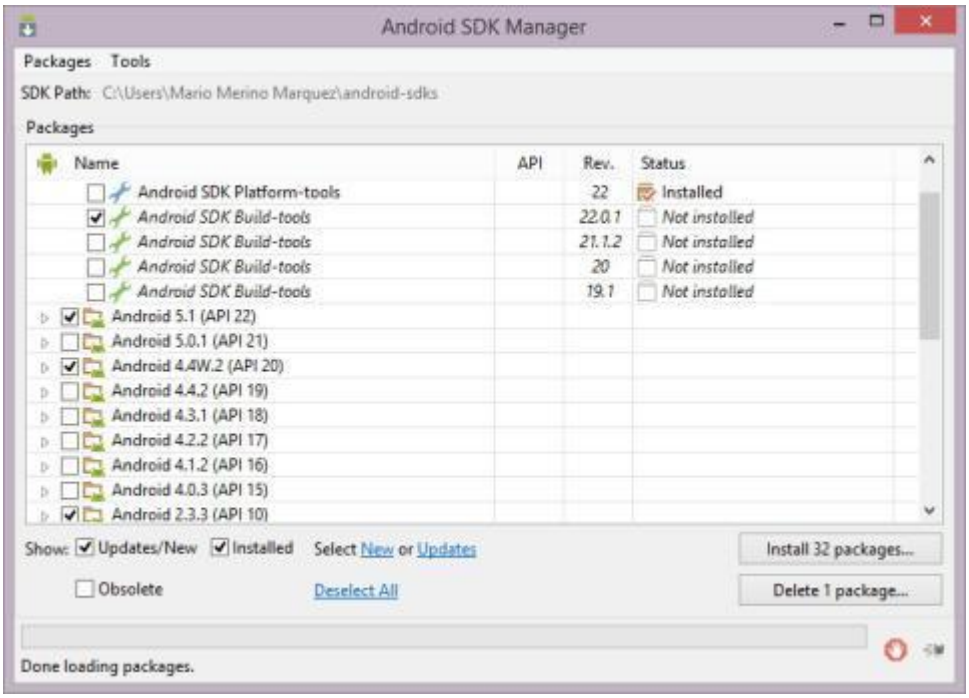

Figura 4.15 Selección de API´s que usaremos en el SDK

Aceptamos las licencias y damos en Install

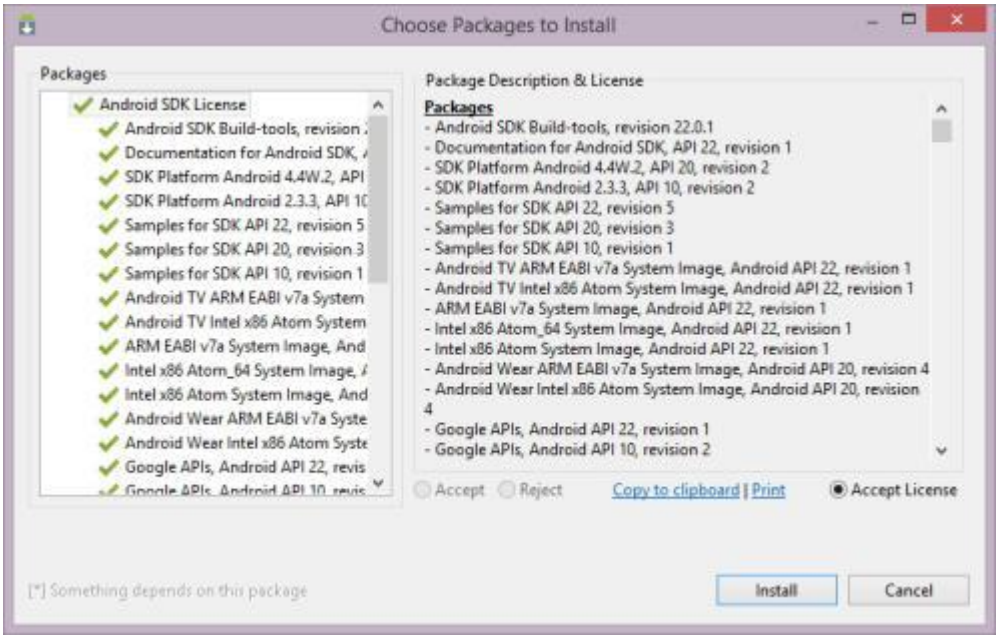

Figura 4.16 Aceptación de licencia para instalar las API´s

Aguardamos que descargue e instale con lo que tendremos listo para empezar a desarrollar aplicaciones en Android.

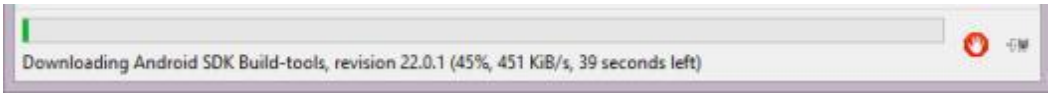

Figura 4.17 Descarga de las API´s que instalaremos

Ahora si cuando abramos el asistente para la creación de un nuevo proyecto nos aparecer la opción de Android.

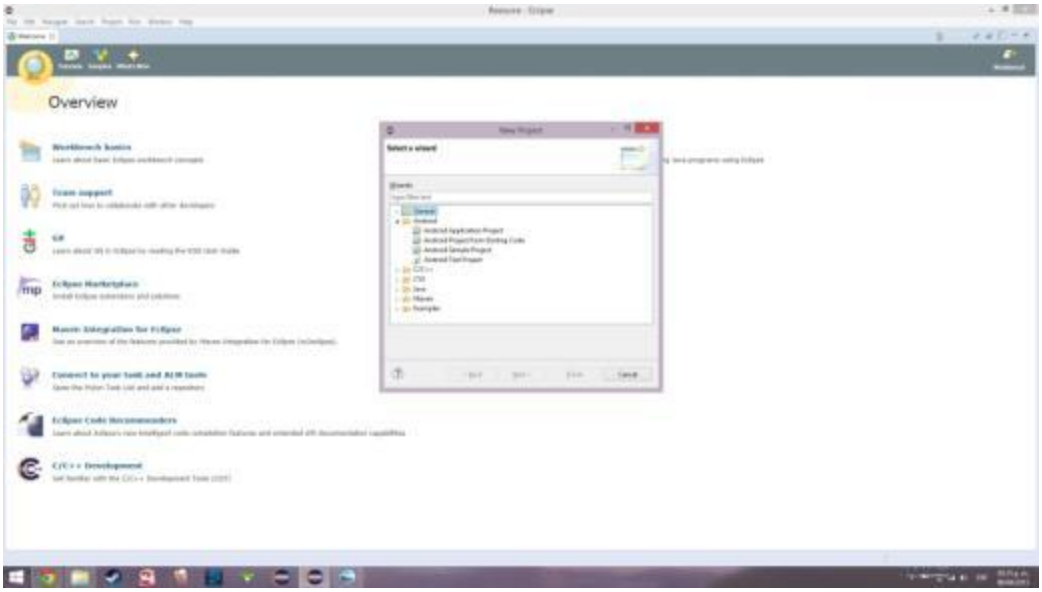

Figura 4.18 En Eclipse al abrir un nuevo proyecto aparece la opción de Android.

Ahora ya podemos desarrollar cualquier proyecto.

4.1.1 PROGRAMACIÓN DE LA APP USANDO UN EMULADOR DE ANDROID PARA PC.

Para poder realizar el sistema debemos tener conocimiento de Java, para abrir un nuevo proyecto ingresamos a Eclipse a File- New- Other – Android – Android Application Project

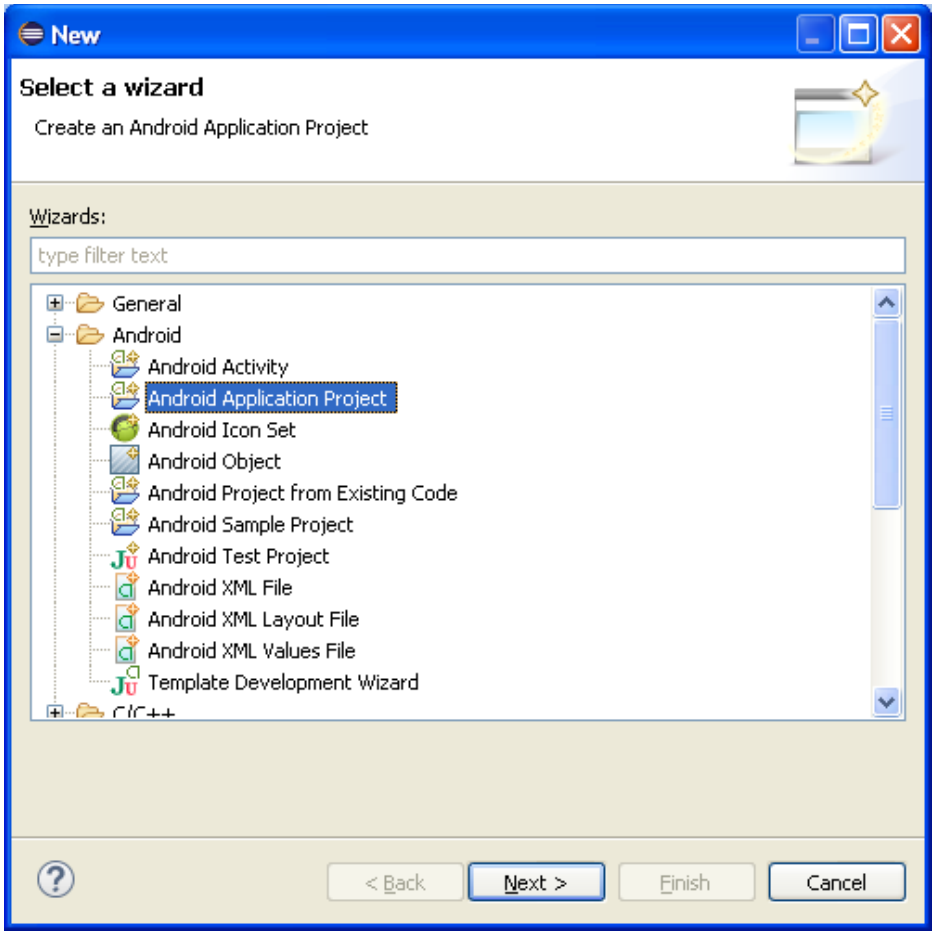

Figura 4.19 Pantalla de nuevo proyecto para Android.

Ponemos Next, luego ingresamos el nombre del proyecto, el nombre del paquete, en mi caso elegí el API 2.2 para que pueda correr en teléfonos Android no tan actualizados.

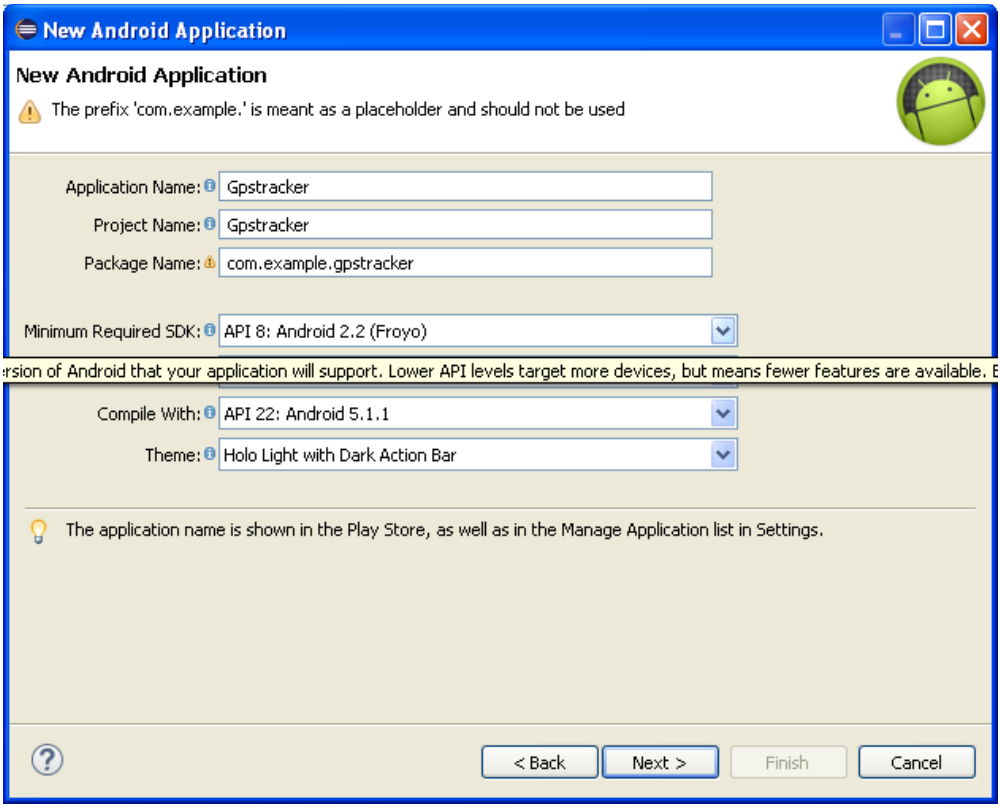

Figura 4.20 Ingreso de datos para el proyecto de Android.

Ponemos Next

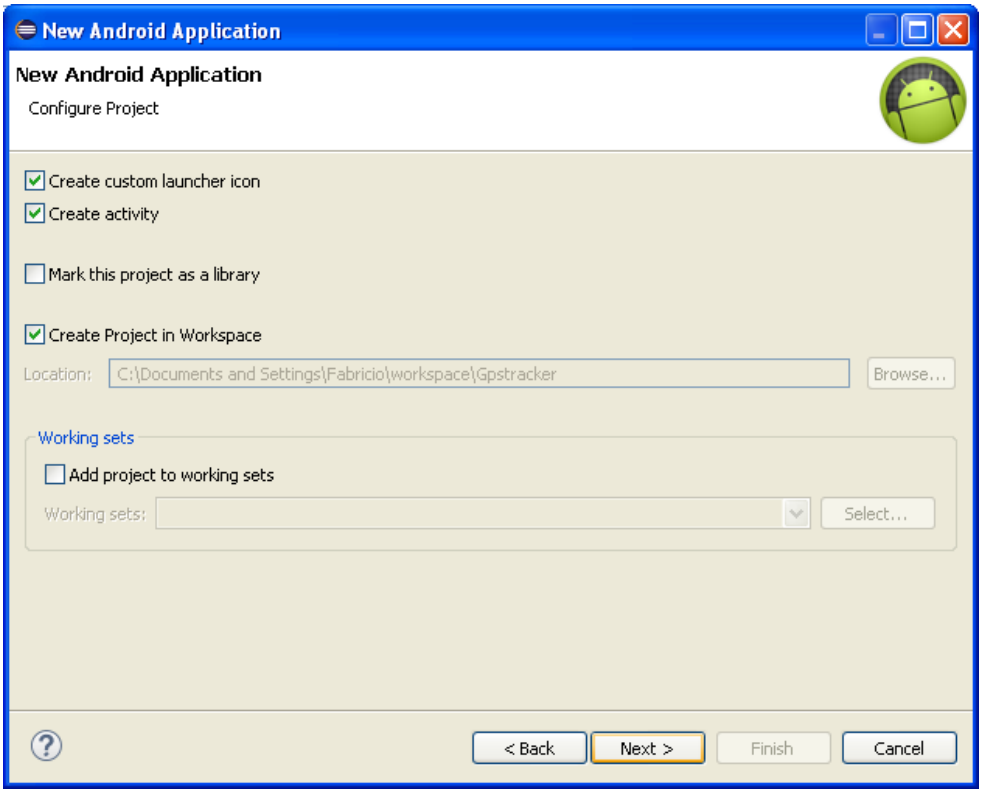

Figura 4.21 Selección de icono y creación de actividad.

Ponemos Next

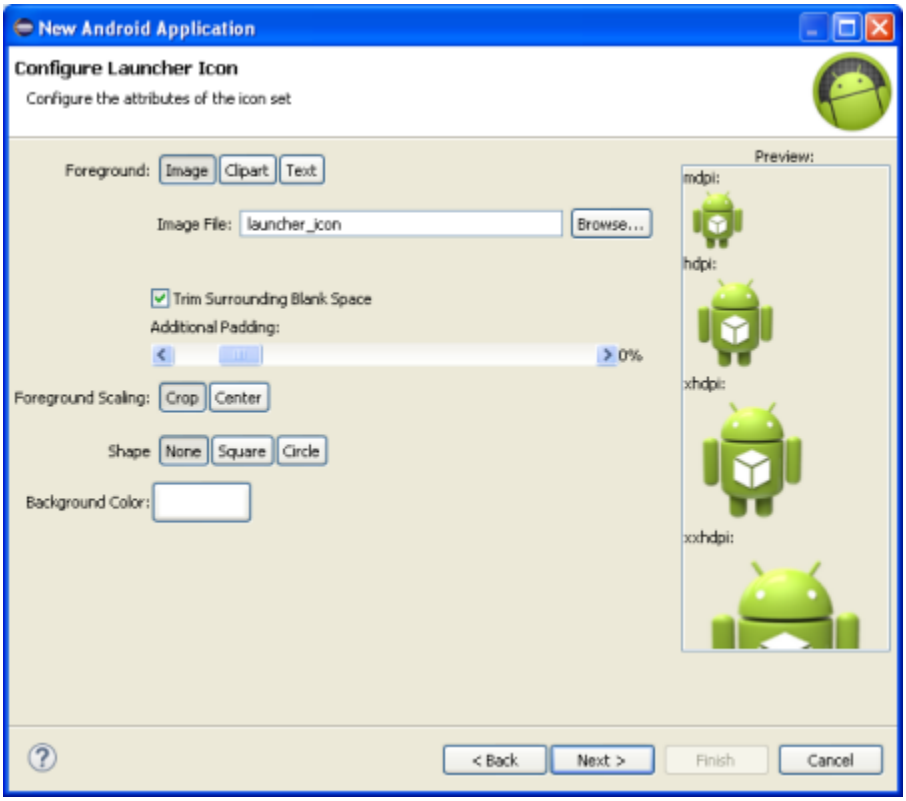

Figura 4.22 Configurando el icono que usaremos en la aplicación.

Ponemos Next

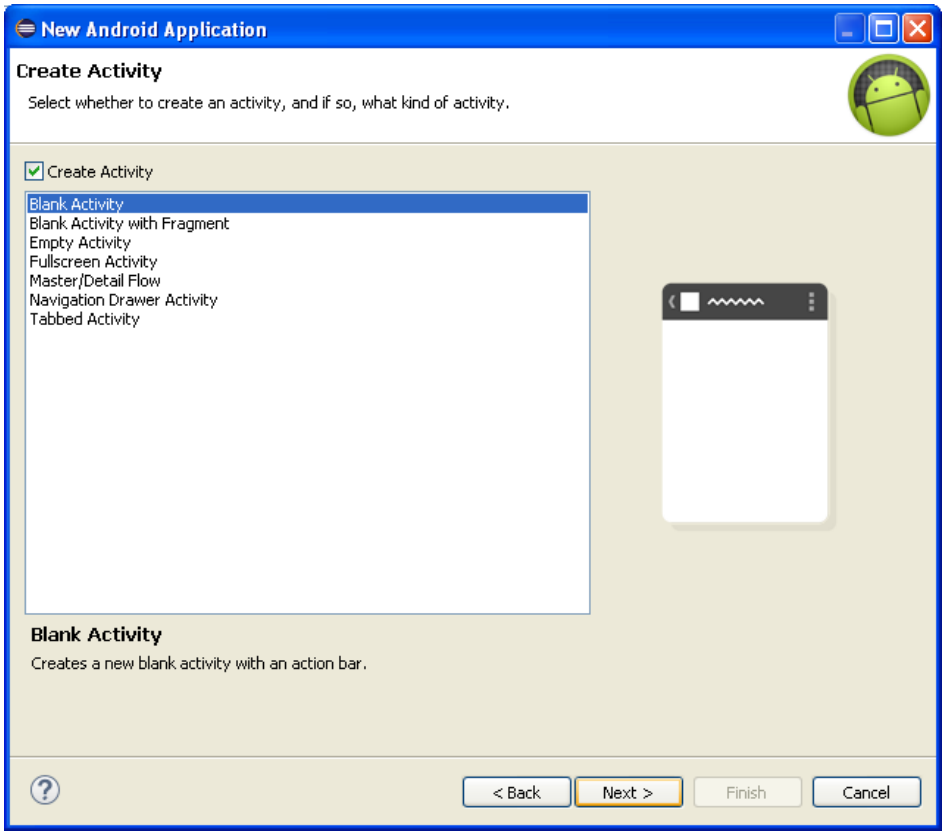

Figura 4.23 Formato para la creación de la actividad.

Ponemos Next, en la siguiente pantalla modificamos si queremos el nombre de la actividad principal.

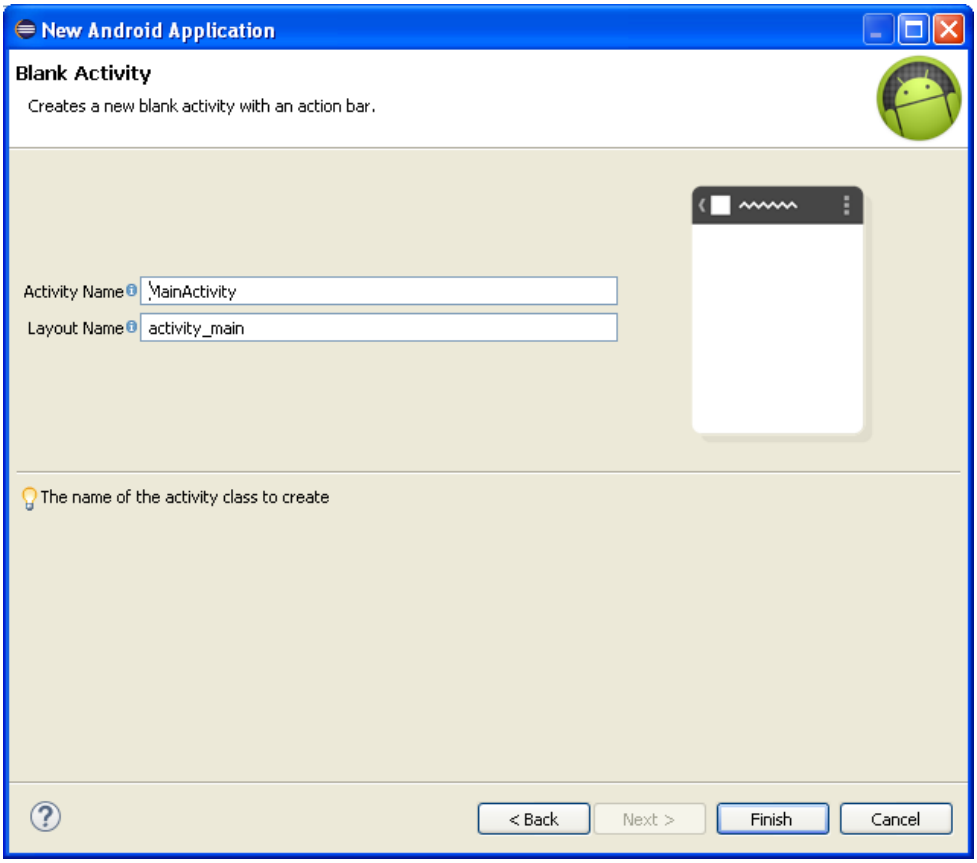

Figura 4.24 Ingreso de nombre de la actividad y layout principal.

Le damos Finish para que eclipse construya la aplicación base.

Debemos añadir una librería (API) de código abierto, esta API es la que nos permite visualizar las coordenadas GPS en el mapa offline en el proyecto, esta App se llama MAPS.ME

Figura 4.25 Icono utilizado para identificar a la aplicación desarrollada.

4.1.1.1. Clases principales implementadas en el proyecto.

- **La actividad SMSNotify.java la cual es la que espera cuando llega un SMS:**

```
package com.gps.tracker;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Vibrator;
import android.telephony.SmsMessage;
//import android.util.Log;
Import android.widget.Toast;
Public class SMSNotify extends BroadcastReceiver {
        Localiza localiza;
privatestaticfinal String LOG_TAG = "SMSReceiver";
publicstaticfinalintNOTIFICATION_ID_RECEIVED = 0x1221;
```

```
staticfinal String ACTION =
"android.provider.Telephony.SMS_RECEIVED";
int mytracker =985976575;
   String phone = "";
Public void onReceive(Context context, Intent intent) {
        localiza = new Localiza(context);
if (intent.getAction().equals(SMSNotify.ACTION)) {

          Bundle bundle = intent.getExtras();
if (bundle != null) {
Object[] pdus = (Object[]) bundle.get("pdus");
             String smsMessageStr ="";
for (inti = 0; i<pdus.length; ++i) {
                SmsMessage smsMessage =
SmsMessage.createFromPdu((byte[]) pdus[i]);
                String smsBody =
smsMessage.getMessageBody().toString();
                String address =
smsMessage.getOriginatingAddress();
if ((address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Now")) ||
            (address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("In")) ||
```

```
            (address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Power")) ||
            (address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Speeding")) ||
            (address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Distance")) ||
            (address !=
null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Reebot"))) {

        abortBroadcast();

        Intent result = newIntent(context,ReceiveActivity.class);
        result.putExtra(ReceiveActivity.MESSAGE, smsBody);
        result.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(result);

                }
            }
        }
    }
}
    }
```

- **La clase Enviacomandos.java es la que permite enviar los comandos via SMS al rastreador.**

```
package com.gps.tracker;
import com.gprtracker.sms.R;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class Enviacomandos extends Activity implements
OnClickListener{
        String msg = "";
        String clave = "0000";
```

```
        Button bloqueamotor, nobloqueamotor, abrirpuerta,
desactivarpuerta;
        Button trackdemanda, trackinterv, ctrackinterv, trackmeters,
ctrackmeter, alarmspeed, calarmspeed;
        Button settimeEcuador;
        String numero = "0985976575";
        TextView carro, telef, clave1;
        @Override
        Protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.comandos);
        carro = (TextView)findViewById(R.id.Etvehiculo);
        telef = (TextView)findViewById(R.id.Etphone);
        clave1 = (TextView)findViewById(R.id.Etclave);
        carro.setText("Terracan");
        telef.setText(numero);
        clave1.setText(clave);
                bloqueamotor
=(Button)findViewById(R.id.Btbloquear);
                nobloqueamotor =
(Button)findViewById(R.id.Btnobloquear);
                abrirpuerta =
(Button)findViewById(R.id.Btabrirpuerta);
                desactivarpuerta =
(Button)findViewById(R.id.Btdesabpuerta);
                trackdemanda = (Button)findViewById(R.id.bttrack);
                trackinterv = (Button)findViewById(R.id.bttres);
```

```
        ctrackinterv = (Button)findViewById(R.id.btcmin);
        trackmeters = (Button)findViewById(R.id.bttmeters);
        ctrackmeter = (Button)findViewById(R.id.btcmeters);
        alarmspeed = (Button)findViewById(R.id.btspeed);
        calarmspeed = (Button)findViewById(R.id.btcspeed);
        settimeEcuador = (Button)findViewById(R.id.bthora);

        bloqueamotor.setOnClickListener(this);
        nobloqueamotor.setOnClickListener(this);
        abrirpuerta.setOnClickListener(this);
        desactivarpuerta.setOnClickListener(this);
        trackdemanda.setOnClickListener(this);
        trackinterv.setOnClickListener(this);
        ctrackinterv.setOnClickListener(this);

}

@Override
publicvoid onClick(View arg0) {
        // TODO Auto-generated method stub
        switch(arg0.getId()){
        case R.id.Btbloquear:
        msg = clave+",C01,00,12222";
        sendSms(numero,msg);
        msg = "";
                break;
        case R.id.Btnobloquear:
```

```
            msg = clave+",C01,00,02222";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.Btabrirpuerta:
            msg = clave+",C01,00,21222";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.Btdesabpuerta:
            msg = clave+",C01,00,20222";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.bttrack:
            msg = clave+",A00";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.bthora:
            msg = clave+",B35,-300";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.bttres:
            msg = clave+",A02,3,1";
            sendSms(numero,msg);
```

```
msg = "";
            break;
case R.id.btcmin:
            msg = clave+",A02,0,0";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.bttmeters:
            msg = clave+",A14,1000";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.btcmeters:
            msg = clave+",A14,0";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.btspeed:
            msg = clave+",B07,100";
            sendSms(numero,msg);
msg = "";
            break;
case R.id.btcspeed:
            msg = clave+",B07,0";
            sendSms(numero,msg);
msg = "";
            break;            }
```

```
}
        Private void sendSms(String phoneNumber, String
message)
   {
       String ENVIA = "SMS_ENVIADO";
       String DELIVERED = "SMS_ENTREGADO";

       PendingIntent enviaPI = PendingIntent.getBroadcast(this, 0,
new Intent(ENVIA), 0);

       PendingIntent entregaPI = PendingIntent.getBroadcast(this, 0,
new Intent(ENTREGA), 0);

//---when the SMS has been sent---
registerReceiver(new BroadcastReceiver(){
@Override
publicvoid onReceive(Context arg0, Intent arg1) {
switch (getResultCode())
              {
case Activity.RESULT_OK:
Toast.makeText(getBaseContext(), "SMS enviado",
                        Toast.LENGTH_SHORT).show();
// Toast.makeText(getApplicationContext(), msg ,
Toast.LENGTH_SHORT).show();
break;
case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
Toast.makeText(getBaseContext(), "Generic failure",
```

```
                        Toast.LENGTH_SHORT).show();
break;
            }
          }
      }, newIntentFilter(SENT));

//---when the SMS has been delivered---
registerReceiver(new BroadcastReceiver(){
@Override
publicvoid onReceive(Context arg0, Intent arg1) {
switch (getResultCode())
            {

Toast.makeText(getBaseContext(), "SMS no entregado",
                        Toast.LENGTH_SHORT).show();
break;
            }
          }
      }, newIntentFilter(DELIVERED));

      SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, sentPI,
deliveredPI);
}

}
```

4.1.1.2. EL ARCHIVO ANDROIDMANIFEST.XML UTILIZADO PARA EL DESARROLLO DEL PROYECTO

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="com.gprtracker.sms"
android:versionCode="1"
android:versionName="1.0">

<uses-sdk
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name="com.gps.tracker.Main"
android:label="@string/app_name">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity
android:name="com.gps.tracker.ReceiveActivity"
```

```
android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Enviacomandos"
android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Base"

android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Acercade"
android:launchMode="singleTop">
</activity>
<activity
android:name="com.mapswithme.Localiza"
android:launchMode="singleTop">
</activity>
<receiverandroid:name="com.gps.tracker.SMSNotify">
<intent-filterandroid:priority="999">
<actionandroid:name="android.provider.Telephony.SMS_RECEIVE
D"/>
</intent-filter>
</receiver>
</application>
</manifest>
```

4.2 INSTALANDO LA APP EN UN SMARTPHONE CON ANDROID.

Una vez probada la aplicación desarrollada, en el emulador de Android AVD (Android Virtual Device), para poder instalarla en un celular debemos copiar el archivo de extensión .apk que lo encontramos en la carpeta de Workspace con el nombre del proyecto y dentro de la carpeta bin.

Copiamos el archivo vía USB o enviándonos por correo desde nuestro computador a nuestro correo de Gmail asociado a nuestro celular.

Una vez copiado el archivo lo abrimos para lo cual debemos tener habilitado en nuestro celular la opción que dice “orígenes desconocidos” de lo contrario el celular no permitirá la instalación. Además es importante tener instalada la App Maps.me, para que nos permita visualizar las coordenadas GPS en el mapa, si acaso nos hemos olvidado de instalar esta aplicación nos aparecerá un link para descargarla e instalarla al momento de querer mostrar un punto en el mapa.

4.3 PRUEBAS DE INTERACCIÓN DE LA APP CON EL GPS TRACKER.

Para realizar las pruebas de interacción de la aplicación con el rastreador, previamente se configuro en un computador el dispositivo, indicando el numero al cual solo es permitido enviar los eventos o

permitir recibir llamadas y lo más importante el o los números a llamar en caso de presionar el botón de pánico.

Se instaló un chip de celular prepago de una compañía proveedora del medio el cual tiene asignado un número celular.

La primera prueba que se realizo es haciendo una llamada al número del rastreador, para que este a su vez envíe la coordenada GPS al programa en nuestro celular, desplegándose la ubicación en el mapa del mismo.

Las otras pruebas consistieron en ir configurando por medio de los botones de la aplicación las alertas programadas e ir haciendo que se cumplan los eventos para recibirlos en nuestro celular.

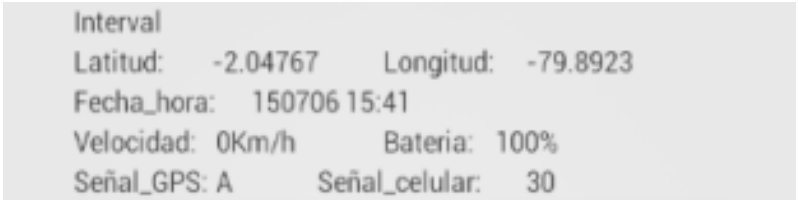

Figura 4.26 Evento recibido programado por intervalo de tiempo.

4.4 PRUEBAS DE INTERACCIÓN DEL GPS TRACKER Y UN VEHÍCULO.

Para la realización de este proyecto se instaló el rastreador satelital MVT380 en un vehículo Hyundai del año 2006 automático.

Figura 4.27 Vehículo que se utilizó para instalar el rastreador.

La carcasa del rastreador que se utilizo es magnética por lo que es muy fácil la instalación pues solo se pega a la carrocería, además nos da la facilidad de desmontarla fácilmente para poder realizar pruebas.

Figura 4.28 Rastreador MVT380 instalado.

Se realizaron las dos pruebas principales de interacción con el vehículo, abrir puertas en caso de que se nos quede las llaves dentro del vehículo y la de apagado de motor en caso de un robo, ambas pruebas se realizaron de manera satisfactoria dándonos los resultados esperados.

Ver video de aperturas de puertas: http://youtu.be/3wv6q-g5pks

Ver video de apagado de motor:https://youtu.be/JDOXNQQhqKo

4.5 MANUAL DEL USUARIO PARA LA APLICACIÓN.

Para ingresar a la aplicación buscamos el icono asociado a ella.

Figura 4.29 Grupo de programas instalados en el celular.

La abrimos y nos aparecerá la primera pestaña que es pantalla principal, esta es la del historial de los eventos recibidos que se encuentran almacenados en una base Sqlite, se van almacenando en forma secuencial.

Pantalla del Historial. Se la identifica porque tiene el siguiente icono**:**

Figura 4.30 Icono de la pantalla que almacena el historial.

Figura 4.31 Historial del sistema.

Hay que indicar que siempre al reiniciar el celular el programa se ejecuta en segundo plano ya que el evento de recepción de mensajes está a la espera de la llegada de un SMS con el número de celular usado en el rastreador.

Borrar evento almacenado. Si queremos borrar uno de estos eventos almacenados lo podremos hacer, manteniendo presionado por 2 segundos dicho evento para lo cual nos aparecerá una pantalla de confirmación si estamos seguros de borrarlo.

Figura 4.32 Confirmación de borrado de registro.

Recepción de SMS del rastreador. Cuando el rastreador satelital tiene algún evento envía un SMS a nuestro celular de inmediato se desplegara el punto en el mapa offline indicándonos la coordenada GPS donde está sucediendo dicho evento.

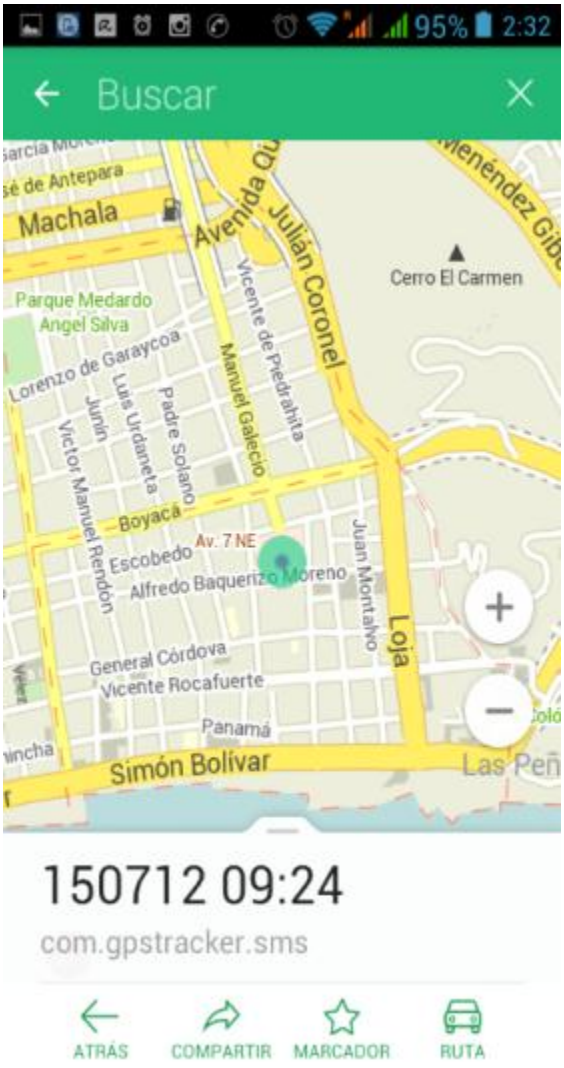

Figura 4.33 Pantalla desplegada al recibir SMS del rastreador.

Si el teléfono celular está bloqueado, recibiremos una alerta de llegada de mensaje SMS con el sonido predeterminado, vibración y el indicador de luz led de color azul, además en la barra de notificaciones nos aparece el icono del programa como indicador de la llegada de un evento nuevo.

Pantalla de comandos. La identificamos por la pestaña con el siguiente icono.

Figura 4.34 Icono de la Pantalla de comandos.

En esta pantalla podremos enviar los comandos al rastreador satelital de una manera fácil tan solo con presionar el botón correspondiente al comando a enviar.

Tenemos los siguientes comandos:

Bloquear motor: Con este botón nosotros le enviamos la instrucción al rastreador que corte la corriente del sensor del carro lo cual hace que se apague de inmediato, esto nos puede ayudar como bloqueo antirrobo del auto o si ya fuimos víctimas de la sustracción del carro

por este medio lo apagamos y podemos empezar a monitorearlo para recuperarlo.

Desactiva bloqueo: Este botón es importante volver a presionarlo pues si ya hemos enviado la instrucción de bloqueo la computadora del carro estará sin corriente y la volvemos a establecer con este botón.

Abrir puerta: Este botón nos permite abrir el seguro de la puerta del vehículo en caso de que se nos quedara la llave dentro, para su funcionamiento el carro debe tener seguros automáticos centralizados.

Desactivar puerta: Este botón es importante presionarlo después de presionar "abrir puerta", no hace la función de cerrar los seguros, pero vuelve el pulso digital a la normalidad, de lo contrario quedaría invertido.

Auto Track: Este botón sirve para pedir la posición del vehículo en ese instante, nos muestra en el mapa la posición GPS.

Ponerlo en Hora: Si por algún motivo se desconectó la batería del rastreador y se cambió la fecha y hora, con este botón volvemos a configurar a GMT -5 la zona horaria de Ecuador.

Track/3 min: Con este botón nos permite recibir cada tres minutos la ubicación GPS del vehículo, mostrándonos en el mapa el sitio automáticamente.

Cancel Track: Como su nombre lo indica con este botón cancelamos la orden anterior.

Track/Km: Presionando este botón configuramos para que el rastreador nos envíe la coordenada cada Km que es desplazado el vehículo.

Cancel Track: Como su nombre lo indica con este botón cancelamos la orden de rastreo cada km desplazado.

Alarma 101 km/h: Este botón nos permite configurar que nos envíen una alarma con la posición cuando sucede en el vehículo el evento de ir a una velocidad mayor a 100Km/h.

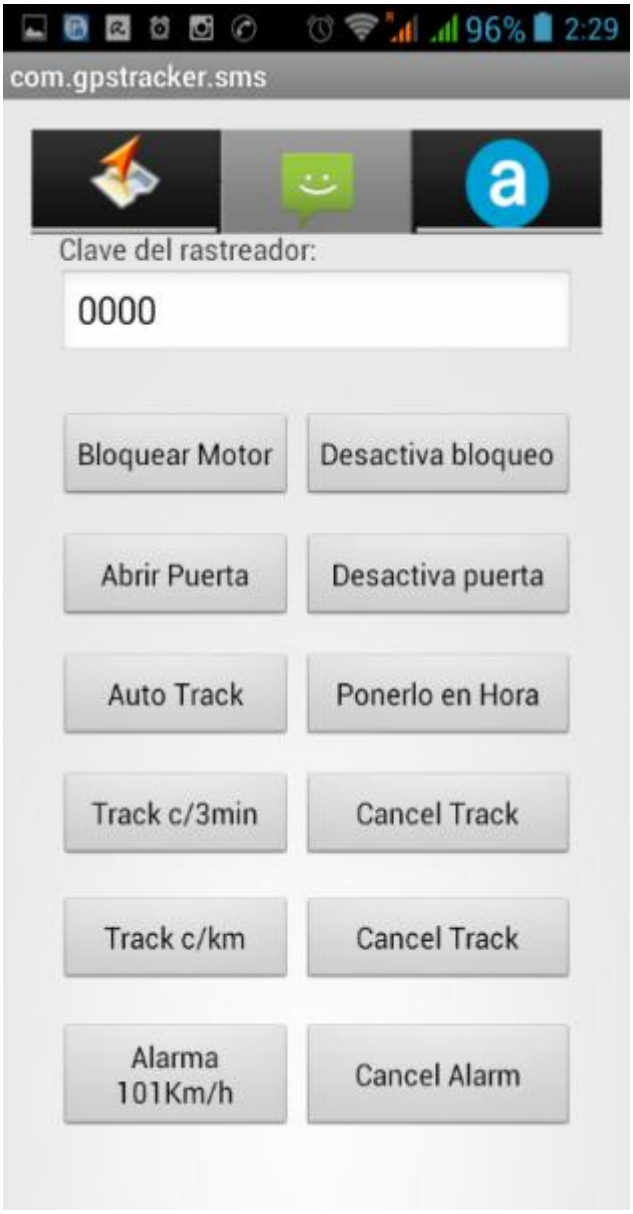

Figura 4.35 Pantalla de envío de comandos.

Pantalla acerca de: La identificamos por la pestaña con el siguiente icono.

Figura 4.36 Icono de la Pantalla acerca de.

Esta pantalla es informativa nos dice desde que versión de Android puede funcionar este programa y qué modelo de rastreador de la marca Meitrack es compatible, además que se debe tener saldo en el celular y el rastreador para el envío de SMS.

Figura 4.37 Pantalla Acerca de.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. La idea de este proyecto nació de la necesidad propia de crear una interface más amigable para interactuar con los rastreadores satelitales que en ese momento me econtraba estudiando su funcionamiento. Estos rastreadores también empezaban a ser comercializados en el país, la problemática es que los comandos SMS usados no son tan fáciles de recordar. Las empresas comercializadoras de estos equipos generalmente ofrecen el servicio de rastreo usando plataformas propias o desarrolladas la cual tiene un costo anual de uso muy elevado.

2. Esta aplicación no está dirigida para la parte comercial que necesita monitorear constantemente su personal en las calles o flota de vehículos, más bien está dirigida para el público particular, el cual no necesita estar monitoreando todo el tiempo el vehículo, sino que hace uso de este en casos de emergencias o por seguridad al dejarlo en algún sitio peligroso.

3. Para la apertura de puertas y apagado de motor no es necesario tener saldo para mensajes de texto en el rastreador, basta con tener saldo en el celular, pero si es importante tener saldo en el rastreador para que pueda enviar las alarmas y eventos a nuestro celular.

4. Para el uso de esta App no se necesita tener internet en el celular pues la App puede mostrar las coordenadas GPS sin necesidad de tener contratado un paquete de datos, solo se debe tener saldo en el celular y en el rastreador para envío de mensajes de texto.

RECOMENDACIONES

1. Para la instalación del rastreador satelital en el vehículo se recomienda que se lo haga con una persona con conocimientos en instalación de alarmas para autos o mucho mejor si la persona ya tiene experiencia en la instalación de equipos de rastreo satelital, para no correr el riesgo de dañar la computadora del vehículo.

2. Esta App fue diseñada para trabajar con teléfonos inteligentes con sistema operativo Android 2.2 o superior.

3. Si estamos utilizando un chip prepago en el rastreador satelital es recomendable no dejar que pase más de 3 meses sin por lo menos ponerle una recarga, recibir un mensaje de texto o llamada telefónica por parte nuestra, ya que la compañía celular si no detecta su uso anulan la línea celular.

4. Para autos modernos del 2010 en adelante no se recomienda instalar el rastreador satelital del modelo MVT380 pues al hacerlo se corre el riesgo de quemar algún sensor o dañar la computadora, para estos modelos es mejor los que se montan en el socket ODB2.

5. Una vez realizado este proyecto podemos acoplar esta App para cualquier dispositivo de rastreo satelital, solo debemos conocer los comandos SMS utilizados.

6. Con el código fuente java y un poco más de conocimiento podemos pasar esta App a otro sistema operativo como IOS para instalarlo en dispositivos Iphone, así poder llegar con esta solución a más personas.

7. En una segunda versión para uso comercial, se puede poner una pantalla inicial de autenticación de usuarios, así como dar la opción de poder registrar más de un rastreador satelital para dar seguimiento y recibir eventos, de igual manera dar la opción de editar los campos de comandos para que puedan ser personalizados al gusto del usuario, entre otras cosas.

BIBLIOGRAFÍA

[1] Ableson Frank, king Chris, and Sen Robi, Android Guía para desarrolladores Segunda Edición, Anaya, 2011.

[2] Fernández Robledo David, Desarrollo de Aplicaciones para Android II, España: Catálogo de publicaciones del Ministerio: www.educacion.gob.es.

[3] Gironés Tomás Jesús, El gran libro de Android 2º Edición, Marcombo, 2012.

[4] Meier Reto, Professional Android 4 Application Development, John Wiley & Sons, Inc., 2012.

[5] Guerrero Ramírez Ricardo. (2015, July) Slideshare. [Online]. http://es.slideshare.net/ricardoV5/sistema-operativo-mvil-android-50239886

[6] Benbourahala Nazim, Android 4 Principios del desarrollo de aplicaciones Java, ENI, 2013.

[7] Wikipedia. Wikipedia Sistema de posicionamiento global. [Online]. https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

[8] Max4. Max4. [Online]. http://www.max4systems.com/rastreo-satelital.html

[9] Aguilar Fabricio, Gavilánez Ruth, and Basantes Víctor. (2005) CIB

ESPOL. [Online]. www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-33823.pdf

[10] Cheung Cavana, Guia de uso del meitrack MVT380, (2013, June).

[11] Wikipedia. Wikipedia. [Online]. https://es.wikipedia.org/wiki/Servicio_de_mensajes_cortos

[12] Walker Jcr. monografias.com. [Online]. http://www.monografias.com/trabajos13/gpts/gpts.shtml

[13] Cheung Cavana, Protocolo SMS Meitrack V1.7, (2013, June).

[14] Blog de MERINOX, https://mariuszmerinox.wordpress.com/2015/04/13/instalacion-sdk-tools-android-eclipse-luna-4-4-2/, (2015, Abril)

[15] MAPS.ME. MAPS.ME. [Online]. http://maps.me/en/home

[16] Alonso Joss. Youtube. [Online]. tutos4u.com

[17] Lee Wei-Meng, *Beginning Android 4 Application Development*., 2012.

[18] codigo2go. codigo2go. [Online]. http://codigo2go.info/blog/?p=81

ANEXO

CÓDIGO DEL PROGRAMA

La actividad principal Main.java:

```
package com.gps.tracker;
import com.gprtracker.sms.R;
import android.app.TabActivity;
import android.content.Intent;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TabHost;
public class Main extends TabActivity implements
TabHost.OnTabChangeListener{
        @Override
        protectedvoid onCreate(Bundle savedInstanceState)    {
                super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
           TabHost tabhost = getTabHost();
           TabHost.TabSpec spec;
           Intent intent;
           Resources res = getResources();
        intent = new Intent().setClass(this,Base.class);
        spec = tabhost.newTabSpec("muestrabase").setIndicator("",
res.getDrawable(R.drawable.historico)).setContent(intent);
        tabhost.addTab(spec);
        intent = new Intent().setClass(this,Enviacomandos.class);
```

```
        spec = tabhost.newTabSpec("comandos").setIndicator("",
res.getDrawable(R.drawable.sms)).setContent(intent);
        tabhost.addTab(spec);
        intent = new Intent().setClass(this,Acercade.class);
        spec = tabhost.newTabSpec("acerca").setIndicator("",
res.getDrawable(R.drawable.acerca)).setContent(intent);
        tabhost.addTab(spec);
        tabhost.setOnTabChangedListener(this);
         }
        @Override
        publicvoid onTabChanged(String tabId) {
                // TODO Auto-generated method stub
        }
}
```

La actividad SMSNotify.java la cual es la que espera cuando llega un SMS:

```
package com.gps.tracker;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
//import android.util.Log;
Public class SMSNotify extends BroadcastReceiver {
        Localiza localiza;
```

```
            (address != null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Power")) ||
            (address != null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Speeding")) ||
            (address != null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Distance")) ||
            (address != null&&address.endsWith(Integer.toString(mytracker))
&&smsBody.startsWith("Reebot"))) {
        abortBroadcast();
        Intent result = newIntent(context,ReceiveActivity.class);
        result.putExtra(ReceiveActivity.MESSAGE, smsBody);
        result.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(result);
              }
            }
     }
        }
   }
  }
```

La actividad ReceiveActivity.java es la que cuando llega un SMS lo procesa, crea el evento de vibración, sonido, el icono de alerta, lo almacena en la base y lo muestra en el mapa:

```
package com.gps.tracker;
```

```
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import com.gprtracker.sms.R;
import com.mapswithme.maps.api.MapsWithMeApi;

public class ReceiveActivity extends Activity{
        public static final String MESSAGE = "message";
        int notificationID = 1;
          @Override
        protected void onCreate(Bundle savedInstanceState)
         {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_received);
           Button regresar = (Button) findViewById(R.id.btback);
```

```
        regresar.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        //Intent intent = new Intent(ReceiveActivity.this,
Main.class);
                        //startActivity(intent);
                        onPause();
                }
        });

           Intent intent = getIntent();
           String message = intent.getStringExtra(MESSAGE);
        if ((message != null && message.startsWith("Now")) || (message !=
null && message.startsWith("In"))

        String[] arreglocadena = message.split(",");
        String Indicador =(arreglocadena[0]);
        String Fecha= (arreglocadena[1]);
        String Senal_Gps = (arreglocadena[2]);
        String Senal_cel = (arreglocadena[3]);
        String Velocidad = (arreglocadena[4]);
        String Bateria = (arreglocadena[5]);
        String latitud_antes = (arreglocadena[6]);
        String longitud_antes = (arreglocadena[7]);

        double latitud =
Double.parseDouble(latitud_antes.substring(40,latitud_antes.length()));
```

```
        double longitud =
Double.parseDouble(longitud_antes.substring(0,longitud_antes.indexOf('&')-
1));
        Localiza localiza = new Localiza(this);
        try {   localiza.abrir();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        localiza.crearEntrada(Indicador, Fecha, Senal_Gps, Senal_cel,
Velocidad, Bateria, latitud, longitud);
        localiza.cerrar();
        MapsWithMeApi.showPointOnMap(this, latitud, longitud, Indicador,
15);
          }
        else {

usuario selecciona esta notificacion
                        PendingIntent pi = PendingIntent.getActivity(this, 0, i,
0);
                        NotificationManager nm = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
                        Uri defaultSound =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

                        //Creamos la notificación, para acceder al icono de la
APP se usa esta declaración
```

```
                //y en android solo con R.drawable y el icono que contenga
                        Notification notificacion = new
Notification(R.drawable.ic_launcher2,
                                        "GPS TRACKER",
                                        System.currentTimeMillis());
                        //Creamos unos mensajes
                        CharSequence from = "Gps Tracker";
                        CharSequence message = "Tienes una nueva
alerta";
                        notificacion.setLatestEventInfo(this, from, message,
pi);
                        notificacion.defaults |=
Notification.DEFAULT_SOUND;
                        notificacion.ledARGB = Color.BLUE;
                    notificacion.ledOnMS = 300;
                notificacion.ledOffMS = 1000;
                    notificacion.flags |= Notification.FLAG_SHOW_LIGHTS;

                        //espera 100 ms, vibra 250 ms, pausa por 100ms y
vuelve a vibrar por 500ms
        protected void onPause(){
        super.onPause();
        finish();       //termina la actividad
        }
}
```

La clase Localiza.java es la que nos crea la base de datos y contiene los diferentes métodos para manejar las consultas en la base sqlite.

```
package com.gps.tracker;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;


publicclass Localiza {

        publicstaticfinalintCOL_FILA = 0;

        publicstaticfinalintCOL_EVENTO = 1;

        publicstaticfinalintCOL_FECHA_HORA = 2;

        publicstaticfinalintCOL_GPS = 3;

        publicstaticfinalintCOL_CEL = 4;

        publicstaticfinalintCOL_VELOCIDAD = 5;

        publicstaticfinalintCOL_BATERIA = 6;

        publicstaticfinalintCOL_LAT = 7;

        publicstaticfinalintCOL_LONG = 8;


        publicstaticfinal String ID_FILA = "_id";

        publicstaticfinal String ID_EVENTO = "ind";

        publicstaticfinal String ID_LAT = "lat";

        publicstaticfinal String ID_LONG = "long";

        publicstaticfinal String ID_VELOCIDAD = "vel";
```

```
        publicstaticfinal String ID_FECHA_HORA = "fecha";
        publicstaticfinal String ID_BATERIA = "bateria";
        publicstaticfinal String ID_GPS = "sengps";
        publicstaticfinal String ID_CEL = "sencelular";
        publicstaticfinal String[] ALL_KEYS = new String[] {ID_FILA,
ID_EVENTO, ID_FECHA_HORA, ID_GPS, ID_CEL, ID_VELOCIDAD,
ID_BATERIA, ID_LAT, ID_LONG};
        String puntosrecibidos;
        privatestaticfinal String N_BD = "gpstracker";
        privatestaticfinal String N_TABLA = "Tabla_de_puntos";
        privatestaticfinalintVERSION_BD = 1;
        private BDHelper nHelper;
        privatefinal Context nContexto;
        private SQLiteDatabase nBD;

        privatestaticclass BDHelper extends SQLiteOpenHelper {

            public BDHelper(Context context) {
                super(context, N_BD, null, VERSION_BD);
            }

            @Override
            publicvoid onCreate(SQLiteDatabase db) {
                // TODO Auto-generated method stub
                db.execSQL("CREATE TABLE " + N_TABLA + "(" +
                ID_FILA + " INTEGER PRIMARY KEY
AUTOINCREMENT, " +
```

```
                    ID_EVENTO + " TEXT NOT NULL, " +
                    ID_FECHA_HORA + " TEXT NOT NULL, " +
                    ID_GPS + " TEXT NOT NULL, " +
                    ID_CEL + " TEXT NOT NULL, " +
                    ID_VELOCIDAD + " TEXT NOT NULL, " +
                    ID_BATERIA + " TEXT NOT NULL, " +
                    ID_LAT + " DOUBLE NOT NULL, " +
                    ID_LONG + " DOUBLE NOT NULL);"
                    );
            }
            @Override
            publicvoid onUpgrade(SQLiteDatabase db, intoldVersion,
intnewVersion) {
                    // TODO Auto-generated method stub
            db.execSQL("DROP TABLE IF EXISTS " + N_TABLA);
            onCreate(db);
            }
    }

    public Localiza (Context c){
            nContexto = c;
    }
    public Localiza abrir()throws Exception{
            nHelper = new BDHelper(nContexto);
            nBD = nHelper.getWritableDatabase();
            returnthis;
    }
```

```
publicvoid cerrar() {
        nHelper.close();
}
publiclong crearEntrada(String ind,String fecha,String sg,String sc
,String vel,String bat,Double lat,Double lon) {
        ContentValues cv = newContentValues();
        cv.put(ID_EVENTO, ind);
        cv.put(ID_FECHA_HORA, fecha);
        cv.put(ID_GPS, sg);
        cv.put(ID_CEL, sc);
        cv.put(ID_VELOCIDAD, vel);
        cv.put(ID_BATERIA, bat);
        cv.put(ID_LAT, lat);
        cv.put(ID_LONG, lon);
        returnnBD.insert(N_TABLA, null, cv);
}
        }
        // Seleccionacolumnapor (rowId)
        public Cursor getRow(longrowId) {
                String where = ID_FILA + "=" + rowId;
                Cursor c =      nBD.query(true, N_TABLA,
ALL_KEYS, where, null, null, null, null, null);
                if (c != null) {
                        c.moveToFirst();
                }
                returnc;
        }
```

}

La clase Base.java es la que se utiliza para mostrar el historial de las coordenadas GPS recibidas en el sistema, permite actualizar y borrar los registros.

```
package com.gps.tracker;
import com.gprtracker.sms.R;
import com.mapswithme.maps.api.MapsWithMeApi;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.AlertDialog.Builder;
import android.content.DialogInterface;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.Toast;
publicclass Base extends Activity {
        TextView tv;
        longbandera=0;
        Localiza info = newLocaliza(this);
```

```
protectedvoid onCreate(Bundle savedInstanceState)
 {
super.onCreate(savedInstanceState);
setContentView(R.layout.muestrabase);


try {
                info.abrir();
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
muestralistadelabase();
hacerclickenlista();
 }
@Override
protectedvoid onDestroy() {
        super.onDestroy();
        info.cerrar();
}
privatevoid muestralistadelabase() {
        // TODO Auto-generated method stub
        Cursor cursor = info.getAllRows();
        startManagingCursor(cursor);
                                        fromFieldNames,
        // DB Column names
                                        toViewIDs
        // View IDs to put information in
```

```
                                                    );

                    // Set the adapter for the list view
                    ListView myList = (ListView) findViewById(R.id.listView1);
                    myList.setAdapter(myCursorAdapter);
            }

            privatevoid hacerclickenlista() {
                            ListView myList = (ListView)
findViewById(R.id.listView1);
                            myList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                                    @Override
                                    publicvoid
onItemClick(AdapterView<?>parent, View viewClicked,
                                                    intposition, longidInDB) {

                                            muestrapuntoporID(idInDB);
                                    }
                            });
                            myList.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
                                    @Override
                                    publicboolean
onItemLongClick(AdapterView<?>parent, View viewClicked,
                                                    intposition, longidInDB) {
```

```
                                        // TODO Auto-generated method
stub
                                        bandera=idInDB;
                                        showDialog(0);
                                        returntrue;
                                    }
                                });
                        }
                @Override
                        protected  Dialog onCreateDialog(intid){

                        DListener listener= newDListener();
                        Dialog dialogo=null;
                        if(id==0){
                                Builder builder=newAlertDialog.Builder(this);
                                builder=builder.setIcon(R.drawable.alerta);
                                builder=builder.setTitle("ALERTA");
                                builder=builder.setMessage("¿Esta seguro de
BORRAR este registro?");
                                builder=builder.setPositiveButton("Si", listener);
                                builder=builder.setNegativeButton("No", listener);
                                dialogo=builder.create();
                        }
                        returndialogo;
                }

                class DListener implements DialogInterface.OnClickListener{
```

```
                    @Override
                    publicvoid onClick(DialogInterface dialog, intwhich)
{
                        if(which ==
DialogInterface.BUTTON_POSITIVE){
                            info.deleteRow(bandera);
                            muestralistadelabase();
                            bandera=0;
                        }
                        if(which ==
DialogInterface.BUTTON_NEGATIVE)
                        return;
                }
                }

            privatevoid muestrapuntoporID(longidInDB) {
                    Cursor cursor = info.getRow(idInDB);
                    if (cursor.moveToFirst()) {
                        longidBD =
cursor.getLong(Localiza.COL_FILA);

                        String lat =
cursor.getString(Localiza.COL_LAT);
                        String lon =
cursor.getString(Localiza.COL_LONG);
```

```
                                    String fecha =
cursor.getString(Localiza.COL_FECHA_HORA);

                                cursor.close();
                        }
                }
```

La clase Enviacomandos.java es la que permite enviar los comandos via SMS al rastreador.

```
package com.gps.tracker;
import com.gprtracker.sms.R;
import android.os.Bundle;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.telephony.SmsManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
public class Enviacomandos extends Activity implements OnClickListener{
```

```
TextView carro, telef, clave1;
@Override
Protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.comandos);
carro = (TextView)findViewById(R.id.Etvehiculo);
telef = (TextView)findViewById(R.id.Etphone);
clave1 = (TextView)findViewById(R.id.Etclave);

carro.setText("Terracan");
telef.setText(numero);
clave1.setText(clave);
        bloqueamotor =(Button)findViewById(R.id.Btbloquear);
        nobloqueamotor = (Button)findViewById(R.id.Btnobloquear);
        abrirpuerta = (Button)findViewById(R.id.Btabrirpuerta);
        desactivarpuerta =
(Button)findViewById(R.id.Btdesabpuerta);
        trackdemanda = (Button)findViewById(R.id.bttrack);
        trackinterv = (Button)findViewById(R.id.bttres);
        bloqueamotor.setOnClickListener(this);
        nobloqueamotor.setOnClickListener(this);
        abrirpuerta.setOnClickListener(this);
        calarmspeed.setOnClickListener(this);
        settimeEcuador.setOnClickListener(this);
}
@Override
```

```
Public void onClick(View arg0) {
        // TODO Auto-generated method stub
        switch(arg0.getId()){
        case R.id.Btbloquear:
        msg = clave+",C01,00,12222";
        sendSms(numero,msg);
        msg = "";
                break;

        case R.id.Btabrirpuerta:
                msg = clave+",C01,00,21222";
                sendSms(numero,msg);
        msg = "";
                break;

        case R.id.Btdesabpuerta:
                msg = clave+",C01,00,20222";
                sendSms(numero,msg);
        msg = "";
                break;

        case R.id.bttrack:
                msg = clave+",A00";
                sendSms(numero,msg);
        msg = "";
                break;
```

```
case R.id.bthora:
        msg = clave+",B35,-300";
        sendSms(numero,msg);
msg = "";
        break;

case R.id.bttres:
        msg = clave+",A02,3,1";
        sendSms(numero,msg);
msg = "";
        break;

case R.id.btcmin:
        msg = clave+",A02,0,0";
        sendSms(numero,msg);
msg = "";
        break;

case R.id.bttmeters:
        msg = clave+",A14,1000";
        sendSms(numero,msg);
msg = "";
        break;

case R.id.btcmeters:
        msg = clave+",A14,0";
        sendSms(numero,msg);
```

```
                    msg = "";
                        break;

                    case R.id.btspeed:
                        msg = clave+",B07,100";
                        sendSms(numero,msg);
                    msg = "";
                        break;

                    case R.id.btcspeed:
                        msg = clave+",B07,0";
                        sendSms(numero,msg);
                    msg = "";
                        break;
                    }
            }
            privatevoid sendSms(String phoneNumber, String message)
{
      String ENVIO = "SMS_ENVIADO";
      String ENTREGADO = "SMS_ENTREGADO";

Public void onReceive(Context arg0, Intent arg1) {
switch (getResultCode())
          {
case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
Toast.makeText(getBaseContext(), "Generic failure",
                  Toast.LENGTH_SHORT).show();
```

```
break;
case SmsManager.RESULT_ERROR_NO_SERVICE:
Toast.makeText(getBaseContext(), "No service",
                        Toast.LENGTH_SHORT).show();
            }
          }
      }, newIntentFilter(ENVIO));
registerReceiver(new BroadcastReceiver(){
@Override
publicvoid onReceive(Context arg0, Intent arg1) {
switch (getResultCode())
                        Toast.LENGTH_SHORT).show();
break;
            }
          }
      }, new IntentFilter(ENTREGADO));
entregadoPI);
   }
}
```

LAYOUT

Main.xml es la pantalla principal asociada con Main.java

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:android1="http://schemas.android.com/apk/res/android"
android1:id="@+id/LinearLayout1"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".Main">
<TabHost
android1:id="@android:id/tabhost"
android1:layout_width="match_parent"
android1:layout_height="match_parent">
<LinearLayout
android1:layout_width="match_parent"
android1:layout_height="match_parent"
android1:orientation="vertical">
<TabWidget
android1:id="@android:id/tabs"
android1:layout_width="match_parent"
android1:layout_height="55dp">
</TabWidget>
<FrameLayout
android1:id="@android:id/tabcontent"
android1:layout_width="match_parent"
android1:layout_height="match_parent">
</FrameLayout>
</LinearLayout>
```

```
</TabHost>
</LinearLayout>
```

Pantalla activity_received.xml asociada a ReceiveActivity.java

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:android1="http://schemas.android.com/apk/res/android"
android1:id="@+id/LinearLayout2"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android1:orientation="vertical"
tools:context=".ReceiveActivity">
<android.widget.Space
android1:id="@+id/space1"
android1:layout_width="match_parent"
android1:layout_height="wrap_content"/>
<TextView
android1:id="@+id/textView1"
android1:layout_width="wrap_content"
android1:layout_height="wrap_content"
```

```
android1:text="Mensaje recibido del rastreador: "
android1:textAppearance="?android:attr/textAppearanceMedium"/>
<android.widget.GridLayout
android1:layout_width="match_parent"
android1:layout_height="wrap_content"/>
<TextView
android1:id="@+id/textView2"
android1:layout_width="wrap_content"
android1:layout_height="wrap_content"
android1:text=""
android1:textAppearance="?android:attr/textAppearanceLarge"/>
<TextView
android1:id="@+id/textMessage"
android1:layout_width="wrap_content"
android1:layout_height="wrap_content"
android1:layout_gravity="clip_vertical"
android1:text="TextView"/>

<TextView
android1:id="@+id/textView3"
android1:layout_width="wrap_content"
android1:layout_height="wrap_content"
android1:text=""
android1:textAppearance="?android:attr/textAppearanceLarge"/>
<Button
android1:id="@+id/btback"
android1:layout_width="wrap_content"
```

```
android1:layout_height="wrap_content"
android1:layout_gravity="center_horizontal"
android1:text="Regresar"/>
</LinearLayout>
```

Pantalla Comandos.xml asociada a Enviarcomandos.java

```
<ScrollViewxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin">
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:text="Comandos SMS"
android:textAppearance="?android:attr/textAppearanceMedium"/>
<TableLayout
```

```
android:id="@+id/TableLayout1"
android:layout_width="match_parent"
android:layout_height="632dp"
android:layout_weight="0.91">
<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:hint="Terracan"
android:text="Vehiculo"
android:textAppearance="?android:attr/textAppearanceSmall"/>
<EditText
android:id="@+id/Etvehiculo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:focusable="false">

<requestFocus/>
</EditText>
<TextView
android:id="@+id/textView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:hint="0985976575"
android:text="Telefono: "
android:textAppearance="?android:attr/textAppearanceSmall"/>
```

```
<EditText
android:id="@+id/Etphone"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:focusable="false"
android:inputType="phone"/>
<TextView
android:id="@+id/textView4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:hint="0000"
android:text="Clave del rastreador:"
android:textAppearance="?android:attr/textAppearanceSmall"/>
<EditText
android:id="@+id/Etclave"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:focusable="false"
android:inputType="number"/>
<TextView
android:id="@+id/textView5"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=""
android:textAppearance="?android:attr/textAppearanceMedium"/>
```

```
<RelativeLayout
android:layout_width="wrap_content"
android:layout_height="352dp"
android:layout_weight="0.04">
<Button
android:id="@+id/Btbloquear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
android:layout_weight="5"
android:text="Bloquear Motor"/>
<Button
android:id="@+id/Btnobloquear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:layout_alignParentTop="true"
android:layout_weight="5"
android:text="Desactiva bloqueo"/>
<Button
android:id="@+id/Btdesabpuerta"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/Btabrirpuerta"
android:layout_alignLeft="@+id/Btnobloquear"
android:layout_alignParentRight="true"
```

```
android:text="Desactiva puerta"/>
<Button
android:id="@+id/bttrack"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignRight="@+id/Btabrirpuerta"
android:layout_below="@+id/Btabrirpuerta"
android:layout_marginTop="15dp"
android:text="Auto Track"/>
<Button
android:id="@+id/Btabrirpuerta"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignRight="@+id/Btbloquear"
android:layout_below="@+id/Btbloquear"
android:layout_marginTop="16dp"
android:text="Abrir Puerta"/>
<Button
android:id="@+id/bttres"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignRight="@+id/bttrack"
android:layout_below="@+id/bttrack"
android:layout_marginTop="15dp"
```

```
android:text="Track c/3min"/>
<Button
android:id="@+id/btcmin"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/bttres"
android:layout_alignBottom="@+id/bttres"
android:layout_alignLeft="@+id/Btdesabpuerta"
android:layout_alignParentRight="true"
android:text="Cancel Track"/>
<Button
android:id="@+id/bthora"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/bttres"
android:layout_alignLeft="@+id/Btdesabpuerta"
android:layout_alignParentRight="true"
android:text="Ponerlo en Hora"/>
<Button
android:id="@+id/bttmeters"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignRight="@+id/bttres"
android:layout_below="@+id/bttres"
android:layout_marginTop="18dp"
android:text="Track c/km"/>
```

```
<Button
android:id="@+id/btcspeed"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/btspeed"
android:layout_alignLeft="@+id/btcmeters"
android:layout_alignParentRight="true"
android:layout_alignTop="@+id/btspeed"
android:text="Cancel Alarm"/>
<Button
android:id="@+id/btcmeters"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/btspeed"
android:layout_alignLeft="@+id/btcmin"
android:layout_alignParentRight="true"
android:text="Cancel Track"/>

<Button
android:id="@+id/btspeed"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignRight="@+id/bttmeters"
android:layout_below="@+id/bttmeters"
android:layout_marginTop="19dp"
android:text="Alarma 101Km/h"/>
```

```
</RelativeLayout>
</TableLayout>
</LinearLayout>
</ScrollView>
```

Pantalla muestrabase.xml asociada a Base.java

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/LinearLayout1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="top"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".Base">
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:text="Historial"
android:textAppearance="?android:attr/textAppearanceMedium"/>
<ListView
```

```
android:id="@+id/listView1"
android:layout_width="match_parent"
android:layout_height="wrap_content">
</ListView>

</LinearLayout>
```

AndroidManifest.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="com.gprtracker.sms"
android:versionCode="1"
android:versionName="1.0">

<uses-sdk
android:minSdkVersion="9"
android:targetSdkVersion="21"/>
<uses-permissionandroid:name="android.permission.RECEIVE_SMS"/>
<uses-permissionandroid:name="android.permission.SEND_SMS"/>
<uses-permissionandroid:name="android.permission.READ_SMS"/>
<uses-permissionandroid:name="android.permission.VIBRATE"/>
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
```

```
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity
android:name="com.gps.tracker.ReceiveActivity"

android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Enviacomandos"
android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Base"
android:launchMode="singleTop">
</activity>
<activity
android:name="com.gps.tracker.Acercade"
android:launchMode="singleTop">
</activity>
<actionandroid:name="android.provider.Telephony.SMS_RECEIVED"/>
</intent-filter>
</receiver>

</application>
</manifest>
```