Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2

Pablo Luis Pérez del Aguila

Asesorado por el Ing. Guillermo Antonio Puente Romero

Guatemala, septiembre de 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

PABLO LUIS PÉREZ DEL AGUILA
ASESORADO POR EL ING. GUILLERMO ANTONIO PUENTE ROMERO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, SEPTIEMBRE DE 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Ángel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Jurgen Andoni Ramírez Ramírez
VOCAL V	Br. Óscar Humberto Galicia Nuñez
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Angel Roberto Sic García a.i.
EXAMINADOR	Ing. Armando Alonzo Rivera Carrillo
EXAMINADORA	Inga. Ingrid Salomé Rodríguez de Loukota
EXAMINADOR	Ing. José Aníbal Silva de los Ángeles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 17 de noviembre de 2016.

Pablo Luis Pérez del Aguila

Guatemala, 19 de julio de 2017.

Ing. Julio César Solares Peñate
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Ingeniero Solares:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **"DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2"**, desarrollado por el estudiante Pablo Luis Pérez del Aguila carné No. 200925291, ya que considero que cumple con los requisitos establecidos, por lo que el autor y mi persona somos responsables del contenido y conclusiones del mismo.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

Ing. Guillermo Antonio Puente Romero
ASESOR
Colegiado 5898

Guillermo A. Puente R.
INGENIERO ELECTRONICO
COL. # 5898

FACULTAD DE INGENIERIA
Escuelas de Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, Técnica y Regional de Post-grado de Ingeniería Sanitaria.
Ciudad Universitaria, zona 12
Guatemala, Centroamérica

Guatemala, 9 de agosto de 2017

Señor Director
Ing. Otto Fernando Andrino González
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **"DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHICULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MODULO GPS6MV2"**, desarrollado por el estudiante **Pablo Luis Pérez del Águila**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

NIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 35 2017.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; PABLO LUIS PÉREZ DEL AGUILA titulado: DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2, procede a la autorización del mismo.

Ing. Otto Fernando Andrino González

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 28 DE AGOSTO 2,017.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.D.436.2017

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica al trabajo de graduación titulado: **DISEÑO DE UN SISTEMA DE RASTREO EMBEBIDO EN VEHÍCULOS TERRESTRES, ENLAZADO A SERVICIOS EN LA NUBE, UTILIZANDO EL MÓDULO GPS6MV2,** presentado por el estudiante universitario: **Pablo Luis Pérez del Aguila,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, septiembre, de 2017

/cc

ACTO QUE DEDICO A:

Mi madre	Alicia del Aguila Estrada, a quien debo la vida y todo lo que soy.
Mi padre	Pablo Pérez Urízar, por todo el apoyo brindado.
Mi familia	Por todo el apoyo incondicional.
Mis amigos	Quienes han estado siempre en los buenos y malos momentos.
Mis profesores y mentores	Por haberme formado con sus conocimientos y experiencia.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Mi alma mater, grande entre las del mundo. Forjadora de grandes y exitosos profesionales.
Facultad de Ingeniería	Por todo el conocimiento y formación brindada.
Mis amigos de la Rama Estudiantil IEEE-USAC	Por todas las experiencias vividas.
Mis amigos	John Rojas, Juan Pablo Zapeta, Óscar Ramírez Milián y Víctor Carranza, por el apoyo brindado en el transcurso de la carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperios
α	Aplanamiento
φ	Latitud
λ	Longitud
m	Metros
s	Segundos
V	Voltaje

GLOSARIO

Acelerómetro Dispositivo que mide la aceleración. Es útil para detectar las vibraciones en los sistemas o para aplicaciones de orientación.

Amorfo Que no tiene forma o estructura interna definida.

Applet Componente de una aplicación que se ejecuta en el contexto de otro programa.

Arduino Placa de circuito impreso que cuenta con un microcontrolador y otros elementos pasivos. Puede ser programada mediante un entorno de desarrollo**.**

Automoción Sector industrial concerniente a mecanismos gobernados por la acción de motores.

Baliza Objeto señalizador, utilizado para indicar un lugar geográfico o una situación de peligro potencial.

Baudio Unidad de medida de la velocidad de transmisión de señales que se expresa en símbolos por segundo.

Bit Dígito del sistema de numeración binario. Puede tener valores de ceros y unos y es la unidad más pequeña de almacenamiento en una memoria.

Chroot	En los sistemas operativos derivados de Unix, es una operación que invoca un proceso. Cambia para este y sus hijos el directorio raíz del sistema.
Código abierto	Es el *software* desarrollado y distribuido libremente, caracterizado por tener libre acceso al código fuente.
Datum	Término utilizado cuando se hace una relación hacia alguna geometría de referencia importante, sea esta una línea, un plano o una superficie plana o curva.
Embebido	Sistema compuesto de varios componentes reunidos en un mismo empaquetado, diseñado para realizar una o algunas pocas funciones específicas.
Genuino	En electrónica, es una placa de Arduino desarrollada en Estados Unidos.
Hosting	Servicio que provee el espacio en Internet para los sitios *Web*.
Html	Lenguaje de marcado para la elaboración de páginas *web*.
MIPS	Millón de instrucciones por segundo
Multiparadigma	Que soporta más de un paradigma de programación.

Multiplataforma	Que es capaz de funcionar en distintos tipos de sistemas operativos.
Navegador	*Software* o aplicación que permite el acceso a Internet.
Plugin	En informática, es una aplicación que funciona como complemento y se relaciona con otra para agregarle una función nueva y generalmente muy específica.
Protocolo	En comunicaciones, se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores.
Proxy	Servidor que hace de intermediario en las peticiones de recursos que realiza un cliente a otro servidor.
Raster	Denominación que se le da a una matriz de celdas o píxeles organizadas en filas en la que cada celda contiene un valor que representa información.
Robusto	Capacidad y proceso de reacción apropiada ante condiciones que se encuentren fuera del alcance del *software*.
Router	Dispositivo que proporciona conectividad a nivel de red, entre varios dispositivos.

Screen	Pantalla de inicio en una aplicación desarrollada en *Android*.
Script	Programa, usualmente simple, de texto plano, interpretado, que permite realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.
Sincronía	Coincidencia en el tiempo de dos más dispositivos, por lo general tomando en cuenta a uno como referencia.
Smartphone	Teléfono móvil construido sobre una plataforma informática, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional.
SSH	Protocolo y programa que lo implementa, y sirve para acceder servidores privados a través de una puerta trasera, o remota.
TELNET	Protocolo de red que accede a otra máquina para manejarla remotamente.
Teselación	Hace referencia a una regularidad o patrón de figuras que recubren o dividen completamente una superficie plana.

Trilateración	Método matemático para determinar las posiciones relativas de objetos usando la geometría de triángulos de forma análoga a la triangulación.
Vectorial	Relacionado a vectores.
Web	Concepto que se utiliza en el ámbito tecnológico para nombrar a una red informática y, en general, a Internet.
Wireless	Término usado para describir las telecomunicaciones en las cuales las ondas electromagnéticas, en vez de cables, llevan la señal sobre parte o toda la trayectoria de la comunicación.

RESUMEN

El desarrollo del presente trabajo se divide en cuatro capítulos. En el primero se describen las características del sistema GPS, se muestra las partes fundamentales y los principios en los que se basan los dispositivos y sistemas de rastreo.

En el segundo capítulo se exponen los conceptos de los servicios *web,* que se utilizan en el desarrollo del sistema. Se detallan las características de un servidor y un cliente y se presentan los servicios más utilizados en la actualidad, con sus respectivas características.

El tercer capítulo describe cada uno de los componentes y dispositivos que se utilizan en el diseño del sistema. Se incluye una explicación de los lenguajes de programación utilizados.

En el cuarto y último capítulo se explica detalladamente del proceso de elaboración del sistema con la configuración de *hardware* y *software*. Se incluye una sección de pruebas de campo, con parámetros tomados con el dispositivo embebido finalizado.

OBJETIVOS

General

Diseñar un sistema de rastreo embebido en vehículos terrestres, enlazado a servicios en la nube, utilizando el módulo GPS6MV2.

Específicos

1. Presentar un sistema de posicionamiento global.

2. Desarrollar un aplicación *web* para el rastreo satelital de vehículos terrestres.

3. Describir los dispositivos utilizados en el desarrollo del sistema de detección satelital.

4. Proponer el diseño de un sistema de rastreo satelital de bajo costo y alta eficiencia.

INTRODUCCIÓN

En la actualidad, en distintos ámbitos de la industria es fundamental el traslado de un punto a otro, ya sea por transporte terrestre, aéreo o marítimo. En el caso del transporte por tierra, las empresas utilizan flotillas de vehículos para movilizar desde mercancías hasta pasajeros. La administración en el manejo de las flotillas engloba diferentes factores, por ejemplo, las distancias a recorrer, los costos de combustible dependiendo de la distancia requerida y, sobre todo, las trayectorias para realizar dichos recorridos.

A lo anterior se puede agregar que, dependiendo de la mercancía, es importante mantener un control estricto en lo que a normas de seguridad se refiere; es decir, un monitoreo en caso de robo o accidente de cualquier elemento de la flotilla.

Para esto y más, es bastante atractivo para las empresas contar con un sistema que les garantice el monitoreo satelital vía GPS. Incluso si desean un sistema de seguridad y un monitoreo en tiempo real de su vehículo, les resulta ampliamente rentable.

Contar con un sistema práctico y de bajo costo que utilice esta tecnología, permitiría conocer y controlar en tiempo real las diferentes variables presentes en una trayectoria, seguir el trayecto de un vehículo que ha sido robado o controlar si algún empleado de una flota tomó una ruta que no estaba estipulada.

La eficiencia de las empresas puede aumentar considerablemente con esta tecnología, en términos de seguridad y control de vehículos terrestres.

1. SISTEMA DE POSICIONAMIENTO GLOBAL GPS

El sistema de posicionamiento satelital GPS (del inglés *Global Positioning System*) surge como un proyecto militar del gobierno de los Estados Unidos de América. Actualmente su uso se ha extendido a diversos ámbitos no militares.

Se trata de un conjunto de veinticuatro satélites que orbitan el globo terrestre, que permiten determinar la posición de un objeto en la Tierra con una precisión de unos cuantos metros. De la misma manera que los antiguos sistemas de navegación, el GPS utiliza el principio matemático de la triangulación, de tal forma que para determinar la posición de un punto es necesario que el receptor determine con exactitud la distancia que lo separa de los satélites.

Existen diversos sistemas que funcionan utilizando GPS, desde los más sencillos hasta los más complejos, los cuales están preparados para determinar con un margen mínimo de error la longitud, latitud e incluso la altitud desde cualquier punto de la Tierra. También existen los que muestran puntos en los que se ha estado con anterioridad y son capaces de trazar una trayectoria recorrida en forma de mapa.

1.1. Sistemas de Información Geográfica

Un sistema de información geográfica es un conjunto de datos que proporciona información acerca de un determinado espacio físico. Desde tiempos atrás, ha sido necesario el empleo de mapas para representar de alguna manera, la descripción gráfica de diferentes delimitaciones, que van

desde sistemas de carreteras, curvas de nivel o características de un terreno, hasta densidad poblacional en un área determinada.

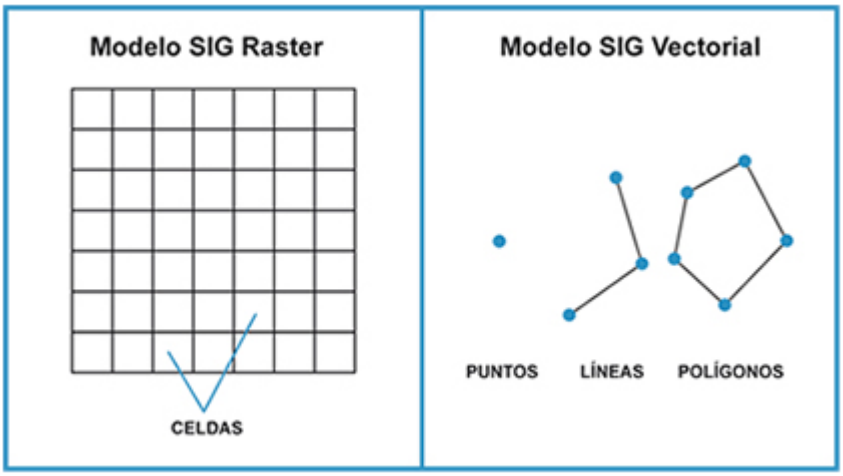
Los mapas eran representados en papel, pero el surgimiento de las computadoras provocó una completa revolución en la manera de plasmar la información geográfica. Sumado a lo anterior, la utilización de los SIG creció ampliamente cuando el GPS estuvo disponible para usos no militares.

En la actualidad es mucho más sencilla la recolección de datos geográficos, gracias a las distintas herramientas disponibles que permiten, incluso, obtener información de lugares remotos en cualquier parte del mundo, mediante servicios *web*. Programas que antes mostraban solamente imágenes de satélites de baja resolución y en blanco y negro, pasaron a ser robustos sistemas geográficos de navegación. Un ejemplo de SIG son los servicios que prestan *Google Earth* y *Google Maps*; este último es de acceso gratuito e incluso permite desarrollar aplicaciones *web,* utilizando su entorno de programación.

1.1.1. Tipos de SIG

La mayoría de datos recolectados para su utilización geográfica, pueden ser representados mediante modelos geométricos o modelos de celdas de información. Estas representaciones son denominadas vectorial y *raster*, respectivamente.

Figura 1. **Tipos de SIG**



Fuente: *Tipos de SIG*. http://sig.cea.es/tipos_SIG. Consulta: 15 de enero de 2017.

1.1.1.1. Modelo vectorial

El modelo vectorial define el espacio a través de conformaciones geométricas para delimitar las características de una zona determinada. La disposición de las áreas sigue un proceso en el que se utilizan vectores que guardan una relación con la forma de los objetos presentes en la zona.

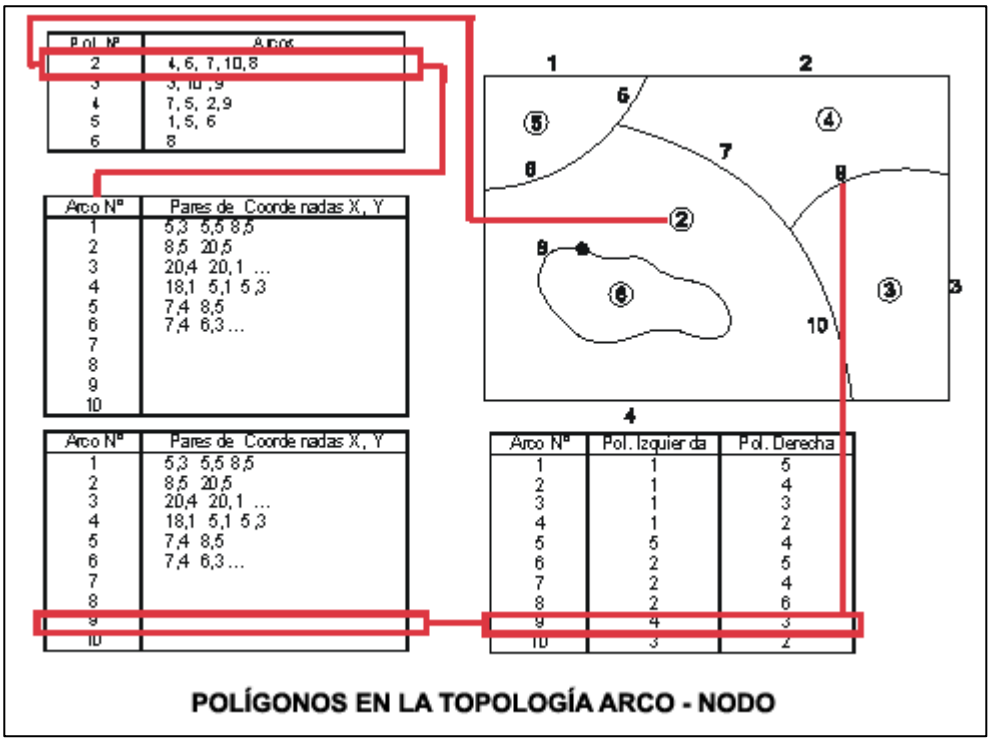
Un punto representa la ubicación de un par de coordenadas; con dos puntos es posible generar una línea, y con un conjunto de líneas es posible delimitar un área formando polígonos.

Al tener una agrupación de puntos, líneas y polígonos, se puede procesar la información en varias bases de datos por medio de tablas, con identificadores para relacionarlas entre sí. Este proceso basa la estructuración de la información geográfica en pares de coordenadas y se denomina “topología

arco-nodo”, al relacionar los arcos (líneas) y los nodos (los vértices que forman estas).

Es adecuado usar este modelo en regiones en las que se tiene claramente delimitados los bordes o contornos del área en cuestión

Figura 2. **Modelo vectorial**



Fuente: *Introducción a los sistemas de información geográfica (SIG)*. http://www.catalonia.org/cartografia/Clase_08/Sistemas_SIG/What_is_GIS_01.html. Consulta: 15 de enero de 2017.

1.1.1.2. Modelo raster

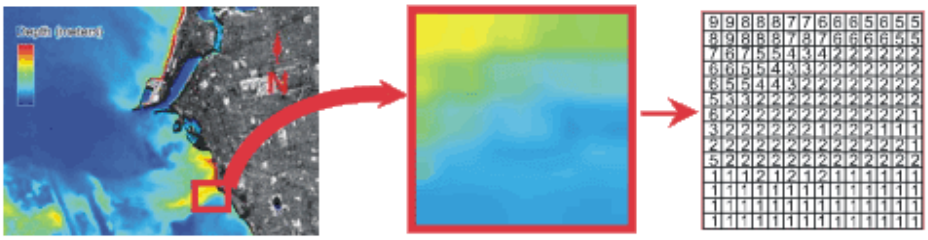
Consiste en una división del espacio o zona de afección en pequeñas celdas, también llamadas pixeles, a las cuales se les asigna un valor. A este proceso se le denomina teselación.

La teselación de una zona permite crear una matriz de información, la cual será almacenada en una base de datos para posteriormente ser procesada.

Para aumentar la precisión de la descripción de la zona geográfica, se necesita una gran cantidad de información almacenada en la base de datos. Esto se consigue al aumentar el número de pixeles, lo que se conoce como alta resolución. Lo anterior no siempre es beneficioso, ya que una alta resolución involucrará un mayor esfuerzo de procesamiento de cómputo.

El modelo raster es especialmente útil en zonas en las que no se tienen contornos claramente definidos, como por ejemplo, propagación de un gas o un líquido en determinado espacio.

Figura 3. **Modelo *raster***



Fuente: *Introducción a los sistemas de información geográfica (SIG)*. http://www.catalonia.org/cartografia/Clase_08/Sistemas_SIG/What_is_GIS_01.html. Consulta: 15 de enero de 2017.

1.1.2. Sistemas de coordenadas geográficas

Para describir el globo terráqueo de forma geográfica se utiliza una serie de puntos y líneas imaginarias. De esta manera, los polos son los puntos que determinan el eje de rotación de la Tierra; los meridianos son los círculos que van de un polo al otro; el Ecuador es la línea que cruza la parte media del globo y las líneas paralelas a este y concéntricas al eje, se conocen como paralelos.

Partiendo de los elementos descritos surgen las coordenadas geográficas, las cuales son un sistema que permite determinar, mediante ángulos, la ubicación de cualquier punto en la Tierra y, a su vez, especificarla mediante números, letras o símbolos. En un plano de dos dimensiones se toman en cuenta dos referencias de posición horizontal; en uno de tres, se agrega una tercera que determine altitud. Las dos referencias de posición horizontal se conocen como latitud y longitud. Su punto de referencia es el centro de la Tierra y comúnmente se expresan en grados sexagesimales.

1.1.2.1. Latitud

La latitud geográfica es una medida de la ubicación de un punto formado por una línea vertical a la Tierra y la intersección con el plano del Ecuador. La vertical en dicho punto se considera la recta que lo une con el centro de la Tierra. El punto de referencia u origen de las latitudes se especifica en el Ecuador, y su valor va desde los 0 a los 90 grados sexagesimales; es decir, desde el Ecuador hasta los polos. Se representa con la letra griega *phi*, φ.

La latitud es positiva cuando el punto en estudio se encuentra en el hemisferio norte y negativa si se encuentra en el sur. El ángulo complementario

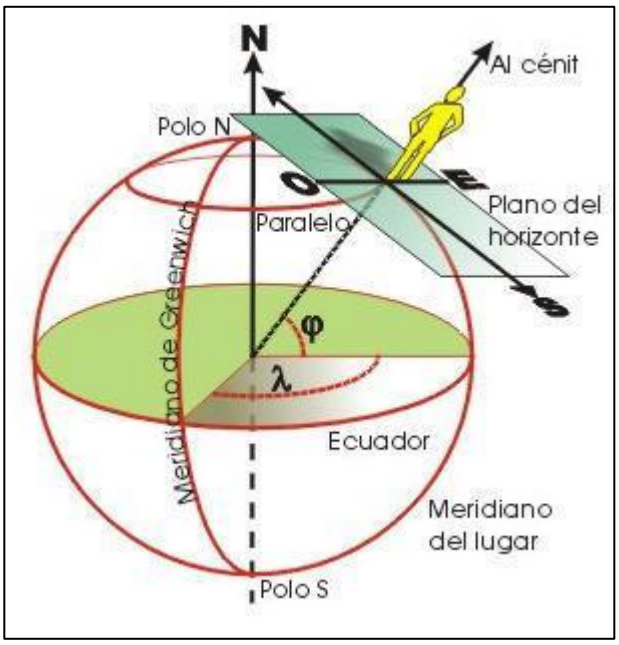
que se forma con el ángulo de la latitud, o sea 90 grados menos de la latitud de un punto, se conoce como colatitud.

1.1.2.2. Longitud

Longitud geográfica se denomina al ángulo formado por el meridiano que pasa por un punto y un meridiano que se toma de referencia. Se representa con la letra griega *lambda*, λ. El meridiano que se toma actualmente como origen es el que pasa por el antiguo observatorio astronómico de Greenwich, Londres, Inglaterra. En la antigüedad existieron muchos puntos de referencia para situar el meridiano 0; el más reciente fue el Meridiano del Hierro en las Islas Canarias, España. Este último se usó por mucho tiempo en Europa, hasta la estandarización del actual. La línea opuesta al Meridiano de Greenwich que completa la circunferencia terrestre es la línea internacional del cambio de fecha, que cruza el Océano Pacífico.

El valor de la longitud se determina a partir de la ubicación del punto respecto al Meridiano de Greenwich. Cuando el punto está hacia la izquierda del meridiano se llama longitud oeste y se dice que es negativa; si está a la derecha, se dice que es longitud este y se le considera positiva.

Figura 4. **Representación de un punto en coordenadas geográficas**

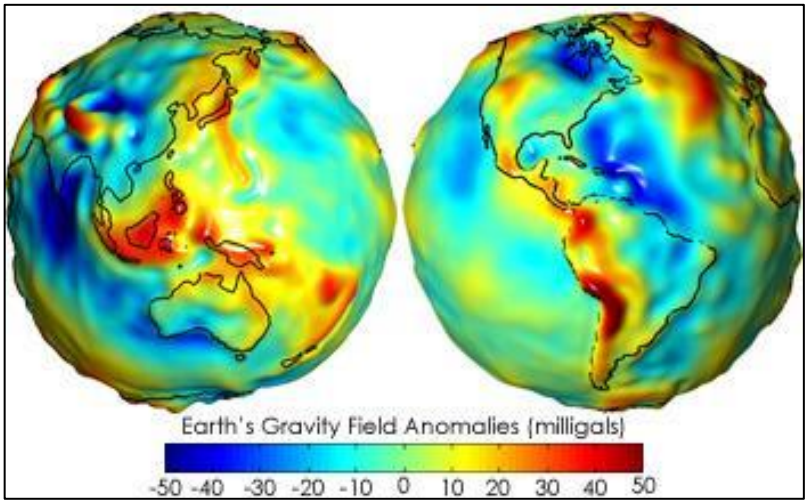


Fuente: Xataka ciencia. *Las coordenadas geográficas*. https://www.xatakaciencia.com/sabias-que/las-coordenadas-geograficas. Consulta: 15 de enero de 2017.

1.1.3. Sistemas geodésicos de referencia

A diferencia de como popularmente se ha representado gráficamente en la mayoría de los casos, la Tierra no tiene una forma esférica; más bien dista bastante de ser un volumen perfecto. Si se toman en cuenta las grandes diferencias que existen en la corteza terrestre, en cuanto a la elevación de los lugares más pronunciados y los valles más profundos, lo que se obtiene es una figura bastante amorfa. Agregado a la anterior, se sabe que debido a la interacción del campo magnético de la Tierra y la rotación de la misma sobre su eje, se produce un ensanchamiento en el Ecuador y un achatamiento en los polos.

Figura 5. **Forma real de la Tierra**



Fuente: NASA. *Mapas de anomalías de gravedad y el geoide*. https://earthobservatory.nasa.gov/Features/GRACE/page3.php. Consulta: 15 de enero de 2017.

Esto provoca muchas discrepancias, particularmente en el caso de querer obtener una ubicación exacta de un punto o si se quiere tomar alguna referencia en cuanto a alturas, por ejemplo. Es acá en dónde toma importancia la geodesia. La geodesia es la parte de la geología que se encarga de desarrollar modelos matemáticos para obtener una figura aproximada de la Tierra, a manera de estandarizar ciertos criterios y facilitar su manejo de una forma geométrica, respetando las características reales de la forma del planeta.

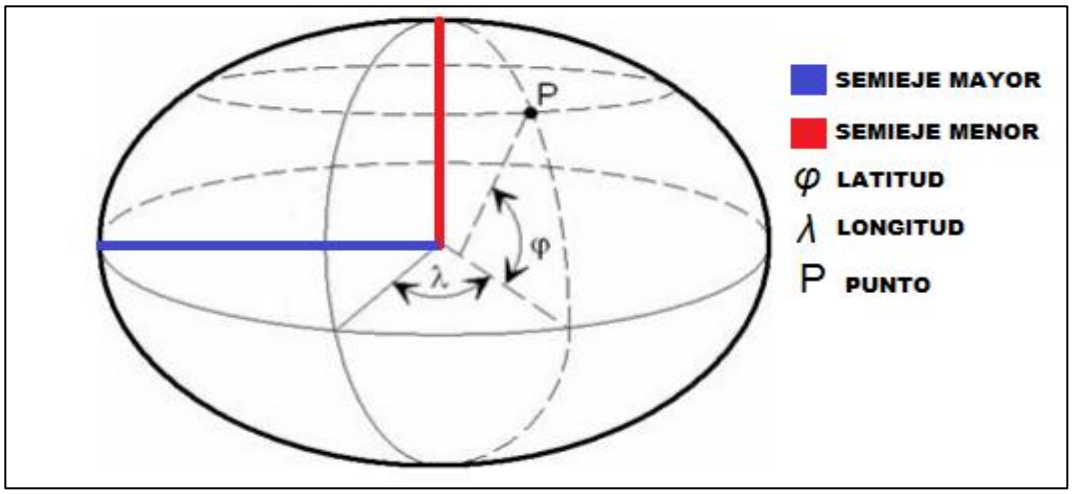
Dado entonces que la figura real del planeta se asemeja más a un elipsoide de revolución, achatado en los polos, se plantea un modelo denominado “geoide”, el cual representa la forma teórica del planeta Tierra.

El geoide toma como referencia de altitud la superficie equipotencial del campo magnético del planeta, que da como resultado el nivel medio del mar, sin

tomar en cuenta la perturbación de las mareas. De este concepto surge el término “sobre el nivel del mar”.

Para la representación del geoide se emplea un elipsoide de revolución que se basa en dos semiejes: el semieje ecuatorial o mayor, el cual toma la longitud del semieje correspondiente al Ecuador, desde el centro de masas de la Tierra hasta la superficie terrestre. El segundo semieje se denomina polar o semieje menor; toma su longitud desde el centro de masas de la Tierra hasta uno de sus polos. A partir de los dos semiejes, se emplean ejes o líneas de referencia en la superficie y es sobre estas que se emplean los sistemas de coordenadas geográficas.

Figura 6. **Representación del geoide mediante un elipsoide de revolución**



Fuente: elaboración propia.

Se utiliza el elipsoide de revolución que mejor se adapte al geoide en determinada zona de la Tierra. Este sistema de encontrar un punto utilizando

los elipsoides de revolución constituye el concepto de Sistema Geodésico de Referencia. A lo largo de la historia diversos elipsoides se han utilizado para definir el Sistema de Referencia de cada país, región o zona, de manera que se define aquel que mejor se ajuste al geoide, según sea el caso. En geodesia existirán dos *datum*, horizontal y el vertical. Por *datum* se entiende la superficie de referencia respecto a la que se definen las altitudes. Lo más normal es que el mismo geoide sirva como referencia de altitud.

1.1.3.1. *European Datum 1950* (ED50)

En 1924 se propuso ante la Asamblea Internacional de Geodesia y Geofísica, un Elipsoide Internacional de Referencia, con semieje mayor, a = 6378388 y aplanamiento, α = 1/297. El elipsoide se nombró *European Datum* y fue utilizado por la mayoría de países europeos. Años después sería perfeccionado por la Unión Astronómica Internacional en Hamburgo, la cual estableció unos nuevos valores de a = 6378160 y α = 1/298,25.

La orientación de este sistema sería estipulada de manera que el eje menor del elipsoide de referencia fuera paralelo a la dirección definida por el Origen Internacional Convencional (OIC) para el movimiento del polo, y el meridiano de referencia, el Meridiano de Greenwich.

1.1.3.2. *World Geodetic System* 1984 (WGS84)

El *World Geodetic System* WGS-84 es un sistema de referencia terrestre único para referenciar las posiciones y vectores. Fue diseñado para ser utilizado como estándar para el sistema de satélites de navegación, NNSS, de forma que se adoptara de manera más real a la forma de la Tierra. Se define como un sistema cartesiano geocéntrico.

Toma en cuenta los centros de masas de la Tierra, incluyendo océanos y la atmósfera.

El eje Z es paralelo a la dirección del polo O.I.C. o polo medio definido por el BIH, con una precisión de 0,005". El eje X es la intersección del meridiano origen, Greenwich, y el plano que pasa por el origen y es perpendicular al eje Z; el meridiano de referencia coincide con el meridiano cero del BIH con una precisión de 0,005".

1.1.4. *International Terrestrial Reference Frame* (ITRF)

Para conseguir una realización práctica de un marco geodésico global de referencia se deben establecer una serie de puntos con un conjunto de coordenadas. Un conjunto de puntos consistentes involucra elementos como la localización de un origen, la orientación del sistema de ejes cartesianos ortogonales y una escala.

En otras palabras, un conjunto de estaciones con coordenadas bien determinadas permite una realización de un Marco de Referencia Terrestre (TRF, *Terrestrial Reference Frame*). De acuerdo a lo anterior, estos efectos temporales definen los sistemas y marcos de referencia terrestres. El *International Terrestrial Reference System* describe procedimientos para crear marcos de referencia adecuados para su uso con mediciones en o cerca de la superficie de la Tierra. Esto se hace de la misma manera que un estándar físico puede ser descrito como un conjunto de procedimientos para crear una realización de ese estándar. Tiene una diferencia de unos pocos centímetros respecto al WGS84.

1.1.4.1. European Terrestrial Reference System (ETRS89)

La Subcomisión de la Asociación Internacional de Geodesia (IAG) para el marco de referencia europeo (EUREF), recomendó que el Sistema de Referencia Terrestre para Europa que debía ser adoptado, fuera el denominado *European Terrestrial Reference System* 1989 (ETRS89). Este es un sistema de referencia geodésico determinado a partir de la placa continental europea. Su datum geodésico espacial está diseñado para soportar modernos sistemas de navegación por satélite, como GPS, GLONASS y el europeo GALILEO.

1.2. Rastreo mediante el uso de satélites

Un sistema de posicionamiento global, tal y como se ha explicado anteriormente, basa su funcionamiento en un conjunto de satélites, los cuales siguen una serie de pasos para ubicar un punto sobre la Tierra.

A continuación se describe cada uno de los pasos que realiza el GPS para su funcionamiento.

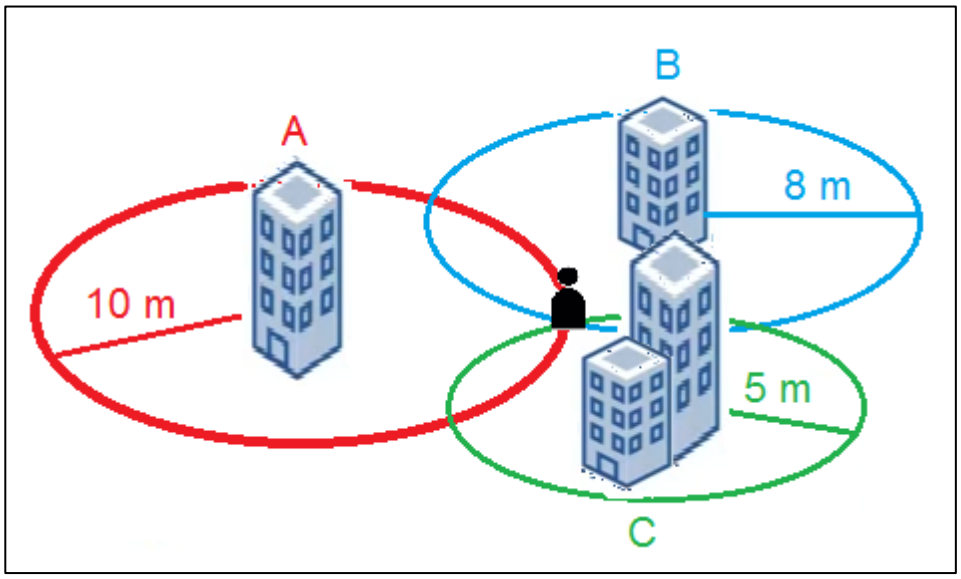
1.2.1. Método de la triangulación

El proceso de triangulación de puntos a partir de distancias obtenidas respecto a ciertas referencias es ampliamente usado en cartografía. Aunque en un modo más estricto el término más apropiado sería trilateración, ya que no se manejan ángulos sino distancias, se conoce con este nombre al método en el cual se basa la ubicación de puntos en el GPS.

En un plano cualquiera, si se desea obtener la ubicación de un punto, se necesitan al menos tres referencias, respecto de las cuales se aproximará una distancia hacia el punto en cuestión. Por ejemplo, en una ciudad, si alguien se encuentra a 10 metros del edificio A, se entiende que la persona se encuentra en un radio de 10 metros en torno al edificio, esto da muchas probabilidades de ubicación alrededor de una circunferencia.

Si además de esta información, se dice que la persona se encuentra a 8 metros del edificio B, la intersección de estas circunferencias limita el universo posible de puntos en donde se podría ubicar. Si además se dice que la persona se encuentra a 5 metros del edificio C, se puede saber con más exactitud en dónde se encuentra, ya que se encontrará en la intersección de las tres, limitando el número de puntos aleatorios.

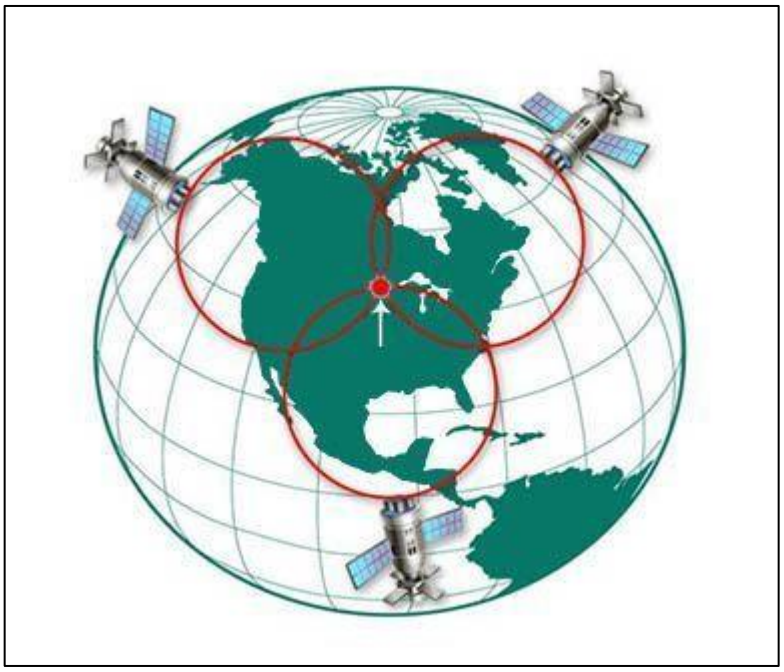
Figura 7. **Ubicación por método de triangulación**



Fuente: elaboración propia.

En el caso del GPS, los puntos de referencia son los 24 satélites que el sistema posee orbitando alrededor del planeta. El receptor de GPS envía señales electromagnéticas que llegan a los satélites, y mediante las distancias entre estos últimos y el receptor, se puede realizar una triangulación entre los más cercanos y, con ello, definir la ubicación del receptor.

Figura 8. **Satélites en el método de la triangulación**



Fuente: *Cómo funciona el GPS del coche*. http://www.lavidacotidiana.es/ wp-content/uploads/2015/09/triangulacion-gps.jpg. Consulta: 18 de enero de 2017.

1.2.2. Cálculo de distancias

Parte del proceso de detección satelital involucra el cálculo de las distancias entre los satélites del sistema GPS y el punto que se desea encontrar. El conjunto de 24 de satélites del sistema es constantemente

monitoreado por quienes lo administran, por lo que ellos mismos tienen la información sobre la posición sobre la órbita en la cual se encuentran. Resulta especialmente beneficioso, ya que esta información debe ser tomada en cuenta por el receptor del sistema.

Para calcular la distancia se realiza una operación sencilla. Basta con encontrar la distancia mediante una ecuación de física básica:

$$Distancia = velocidad * tiempo$$

Primero, el receptor de GPS envía una señal de radio (onda electromagnética propagándose en un medio no conductor), la cual llega al satélite y retorna. En este proceso se realiza una sincronización de tiempo y, por ende, un protocolo de comunicación. Se calcula el tiempo en el que la señal emitida va y vuelve del satélite y como se sabe que una onda electromagnética propagándose en el vacío (y en el aire, en este caso al hacer contacto con la atmósfera) viaja a la velocidad de la luz, ya se tiene el valor de $3X10^8\ m/s.$

Si se supone que el tiempo que recorre la señal de radio, desde el receptor hacia un satélite y regresa es de $0{,}06\ s$, se obtiene la distancia al multiplicar estos dos datos, lo que da como resultado:

$$Distancia = 3X10^8 \frac{m}{s} * 0{,}06\ s$$

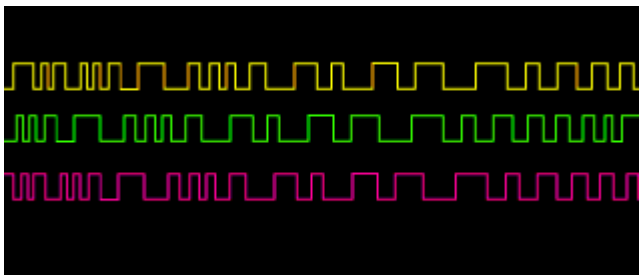
$$Distancia = 18\,000\ km$$

Por lo tanto, el receptor estará a 18 000 km del satélite; el proceso es igual con los demás hasta completar la triangulación. Los receptores de GPS

actuales son capaces de establecer comunicación con varios satélites en simultáneo, en intervalos cortos de tiempo.

El protocolo de comunicación que se establece entre receptor (que en realdad es un emisor y receptor) y cada satélite, consiste en una señal emitida llamada Código Pseudo Aleatorio, PRC por sus siglas en inglés o *Pseudo Random Code.* Tal como su nombre lo indica, es un código digital generado al azar. La sucesión de bits es lo suficientemente compleja para que sea prácticamente irrepetible y no captar por accidente otra señal.

Figura 9. **Código Pseudo Aleatorio**



Fuente: *Como funciona el GPS*. http://gutovnik.com/como_func_sist_gps.htm. Consulta: 18 de enero de 2017.

El término “pseudo” hace referencia a que debido a la complejidad del código, se asemeja bastante a un ruido eléctrico generado al azar, lo que permite que sea altamente improbable que una señal pueda tener la misma secuencia.

Los satélites tienen su propio y único PRC, lo que brinda la posibilidad de que todos trasmitan en la misma frecuencia sin interferirse entre ellos. Lo anterior también evita que alguien intente interferir el sistema induciendo ruido

y, a la vez, permite al Departamento de Defensa de EEUU administrar el sistema de GPS.

El código se basa en la "teoría de la información" para amplificar las señales de GPS. Debido a esto, las señales demasiado débiles emitidas por los satélites pueden llegar a los receptores de GPS sin necesidad de utilizar grandes antenas.

1.2.3. El problema del tiempo y la corrección de errores

Al hablar del cálculo de las distancias entre el receptor y los satélites con los cuales se realiza el método de la triangulación, se mencionó el tema del tiempo, el que es necesario para saber cuánto toma para la señal de radio salir del receptor, establecer comunicación con el satélite y volver. Debe ser una medición bastante precisa. Los satélites cuentan con costosos relojes atómicos que son ajustados cada cierto tiempo. La cuestión acá es si los receptores tuvieran que utilizar relojes de estas características, su precio se elevaría de forma considerable.

La sincronía se resuelve no incrementando las capacidades de los relojes convencionales de los receptores, sino al efectuar la llamada “cuarta medición”. Una vez realizado el método de la triangulación, se sabe en dónde está el punto rastreado; pero si tres mediciones perfectas pueden posicionar un punto en un espacio tridimensional, cuatro mediciones imperfectas pueden obtener el mismo resultado.

Una medición adicional remedia el desfase de sincronización. Con relojes imperfectos, una cuarta medición, efectuada a modo de lazo cerrado, no se intersectará con los tres primeros puntos obtenidos en la triangulación. De

esta manera, el receptor detectará la diferencia y lo atribuirá a una sincronización imperfecta con la hora universal.

Como cualquier variación con la hora universal afectará a las cuatro mediciones, el receptor buscará un factor de corrección único aplicándolo a sus mediciones de tiempo y hará que los rangos coincidan en un solo punto. Dicha corrección permitirá al reloj del receptor ajustarse nuevamente a la hora universal y, de esa manera, se obtendrá un resultado bastante similar al de un reloj atómico. Cuando el receptor aplica dicha corrección al resto de sus mediciones, se obtiene un posicionamiento bastante preciso.

Otro factor con el que se debe lidiar es con las pérdidas por interferencias en el medio. Como se mencionó anteriormente, en realidad la onda electromagnética no viaja en el espacio vacío, más bien en un conjunto de medios, desde su interacción con la atmósfera terrestre, con las complejidades climáticas que conlleva, hasta llegar al espacio exterior y hacer contacto con el satélite. Para tratar de contrarrestar este error, se realiza una medición de la velocidad promedio; es decir, la que se toma en un día normal sin complicaciones meteorológicas. A esto se lo llama modelación y tiene la complejidad de que las condiciones atmosféricas raramente se ajustan a un promedio previsto.

Otra manera de reducir errores inducidos por la atmósfera es comparar la velocidad relativa de dos señales diferentes. Estas mediciones son de doble frecuencia y utilizan tecnología de avanzada, disponible solo en receptores GPS de un costo elevado.

Existen otros tipos de errores que son solventados según las características de los receptores GPS, que van desde interferencias por

obstaculización hasta redundancia de satélites y errores intencionales. El primero de ellos se refiere a que, en lugares encerrados, la señal emitida se atenuará más fácilmente. Estas zonas de obstaculización se denominan “Zonas de Fresnel”. Lo segundo sucede cuando el receptor encuentra más satélites de los que necesita para realizar la triangulación. Mientras más inteligente sea la tecnología que utilice, utilizará aquellos satélites en los cuales las intersecciones de sus circunferencias brinden una mejor localización del punto. Por último, se debe mencionar los errores intencionales, los cuales son ocasionados adrede por el Departamento de Defensa de los Estados Unidos, para garantizar que el sistema no sea usado con fines terroristas. Inducen errores de precisión en determinadas zonas a su conveniencia.

1.3. Sistemas embebidos utilizados en el rastreo satelital de vehículos terrestres

Gracias a que en los últimos años se han reducido los costos para la fabricación de dispositivos electrónicos, los sistemas GPS lo han hecho de la misma manera. Sumado a esto, la tecnología ha tendido a la miniaturización de los componentes de un sistema. Este fenómeno fue descrito por Gordon Moore, fundador de Intel. Moore formuló dos leyes en las que establecía que el tamaño de un transistor se reducía a la mitad cada año y medio; así mismo, la segunda dice que la inversión para construir una fábrica de circuitos integrados se duplica cada 3 años.

Esta tendencia ha marcado el rumbo de los componentes electrónicos, lo cual permite que sistemas más complejos quepan en espacios más reducidos. Esto también rige los sistemas de detección satelital; de esta manera, los componentes necesarios para el proceso de GPS pueden encapsularse en un componente denominado “embebido”.

Tal y como se describirá a continuación, existen otros sistemas de navegación satelital, los cuales tienen grandes similitudes en su forma de operación. En general, todos utilizan un sistema de satélites que permite detectar un punto en la superficie terrestre.

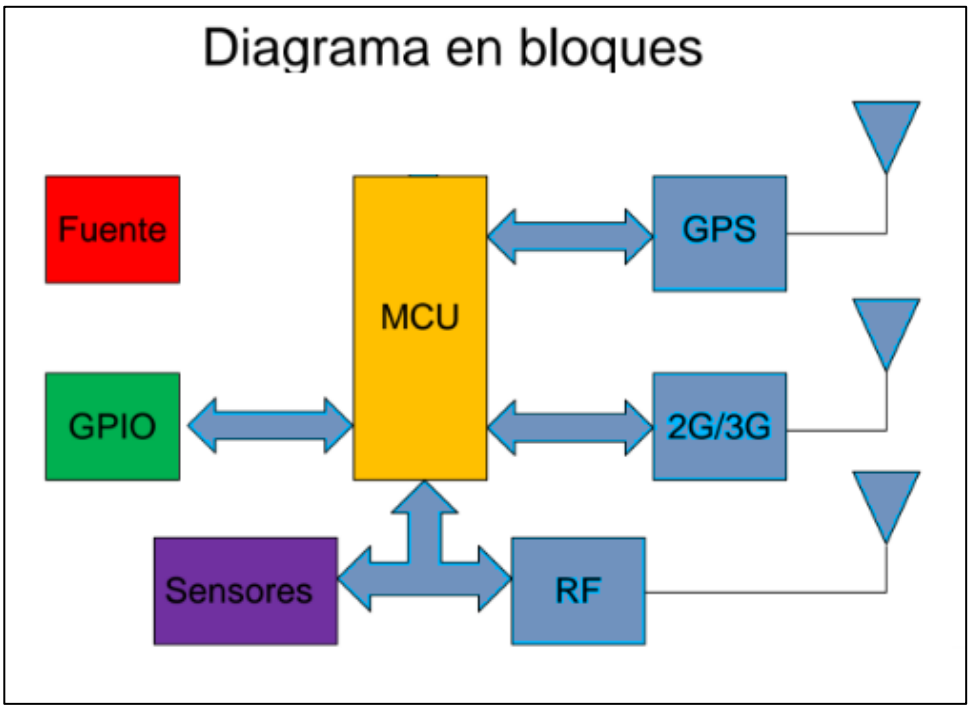
Debido a la producción en masa de estos sistemas en componentes embebidos, se ha masificado su uso en el rastreo de vehículos terrestres. Existe una gran variedad de dispositivos diseñados para que funcionen y utilizan cualquiera de los sistemas satelitales existentes.

1.3.1. Sistema embebido

El término embebido procede del vocablo inglés "*embedded*", que se traduce como incrustado, encapsulado o empaquetado. Se refiere a un sistema electrónico que cuenta con procesamiento de datos, pero a diferencia de una computadora, están diseñados para realizar una función en específico. Ejemplos de un sistema embebido pueden ser, un reloj digital, un teléfono celular o un *router*.

Un sistema embebido es comandado por un cerebro, que por lo general puede ser un microcontrolador, microprocesador, un DSP o un FPGA. El fin de su fabricación no solamente es la reducción del espacio del dispositivo final, sino también su costo de producción y posterior comercialización, además de optimizar su desempeño. Mejora algunas características como precisión y exactitud y también asegura un bajo consumo de energía, ya que su fin primordial es su utilización en entornos que requieren que sean alimentándolos con batería.

Figura 10. **Sistema embebido**



Fuente: elaboración propia.

La fabricación de sistemas embebidos ha significado un gran impulso en el desarrollo tecnológico, no solo en sistemas GPS sino en sistemas de navegación satelital en general.

1.3.2. Los módulos GNSS

Hasta el momento se ha descrito un sistema en específico, el GPS, sin embargo en la actualidad existen otros tres sistemas mundialmente utilizados, según la región que los haya desarrollado.

El GPS, como se ha mencionado, fue desarrollado en los Estados Unidos de América, y es administrado por su Departamento de Defensa. En Europa se desarrolló el sistema Galileo, en Rusia el GLONASS y en China el BeiDou Compass.

Todos estos sistemas se engloban en el concepto de *Global Navigation Satellite System (GNSS)*, los dispositivos embebidos se diseñan para que funcionen bajo un sistema GNSS en específico.

1.3.2.1. Sistema Galileo

Fue desarrollado entre de la Unión Europea y la Agencia Espacial Europea. La alianza enfatizó en que su uso, a diferencia del GPS, sería completamente civil.

El sistema espera estar completo en 2020, aunque en fase de prueba comenzó operaciones en 2016. Actualmente cuenta con 18 satélites en órbita y se espera que sean un total de 30. Los dispositivos embebidos actualmente diseñados para operar bajo este sistema, son compatibles con el GPS, pero tienen una precisión de solo un metro. El sistema brinda además del servicio de posicionamiento abierto global, el de posicionamiento cifrado y protegido, para servicios de seguridad, utilizado en flotillas de empresas y que además cuenta con una precisión de solo centímetros, y el de retransmisión de señales de balizas de emergencia, pensado para su utilización en cuerpos de socorro.

Uno de los primeros dispositivos diseñados para ser compatibles con Galileo, es el *smarthphone* BQ Aquaris X5.

1.3.2.2. GLONASS

El sistema GLONASS, del ruso *Global'naya Navigatsionnaya Sputnikovaya Sistema,* es un sistema de navegación satelital, desarrollado por la antigua Unión Soviética y actualmente administrado por la Federación Rusa.

Está formado por una constelación de 31 satélites, de los cuales 24 están en servicio activo, 3 son de repuesto y 2 son utilizados para mantenimiento de todos los demás, y los otros dos para servicio y pruebas. La constelación está distribuida en grupos de 8 satélites en tres órbitas diferentes.

Los satélites del GLONASS orbitan alrededor del planeta a una altitud de 19 100 km, unos kilómetros más abajo del GPS que orbita a 20 200 km. La constelación toma 11 horas y 15 minutos en completar una órbita alrededor de la Tierra.

Entre los dispositivos embebidos para funcionar con GLONASS están los circuitos integrados SiRFstar V, los ST Teseo II y los Mediatek MT3333. Además de integrar posicionamiento desde GNSS, brindan soluciones de radio terrenal como Wi-Fi y celular, y sensores como acelerómetros, giroscopios y brújulas.

1.3.2.3. BeiDou Compass

El GNSS *BeiDou* es un proyecto desarrollado por la República Popular de China. La palabra "*Beidou*" es un vocablo chino para referirse a la constelación de la Osa Mayor. Tuvo sus inicios con *BeiDou*-1 en el 2000 y se trata de un sistema local que tiene cobertura en China y países cercanos.

Posterior a este, se empezó a trabajar en la segunda parte del sistema que en un inicio se denominó *BeiDou*-2. Cuando esté completamente finalizado, pasará a llamarse *Compass*.

Inicialmente China estaba trabajando en conjunto con la Unión Europea en el desarrollo del Sistema Galileo, pero optó después por desarrollar un sistema propio. La primera fase del *BeiDou* cuenta con una constelación de 14 satélites. A diferencia de los sistemas GPS, GLONASS, y Galileo, que utilizan satélites posicionados en órbitas más bajas y geoestacionarias, el sistema no necesita de un gran número de satélites. Esto también reduce su cobertura a China y alrededores.

Una de las diferencias más marcadas respecto a otros sistemas es que calcula las coordenadas únicamente con dos satélites y una estación en la Tierra. No realiza el método de la triangulación, sino que envía una señal desde un dispositivo remoto para confirmar posición.

Cuando se complete la fase *BeiDou Compass*, ofrecerá dos tipos de servicios, un abierto y uno de pago. La diferencia será el grado de precisión: el primero contará con una de hasta 10 metros por 0,2 del segundo.

Debido a que su funcionamiento es bastante similar al GPS, los dispositivos embebidos que funcionan con este sistema también son compatibles con *BeiDou*, aunque no se sabe si en *Compass* se restringirá esta compatibilidad.

Actualmente *SiRFstar* IV y V, ST Teseo II y los *Mediatek* MT3333 funcionan con este sistema.

1.3.3. Uso de la telefonía celular para el rastreo satelital

En sus inicios era bastante complicado hacer uso de sistemas de navegación satelital, ya que era necesaria la adquisición de un equipo especial y caro para realizar el rastreo. Con el paso del tiempo, el sistema llegó, junto con otros tantos servicios, al campo de la telefonía móvil. Hoy es bastante común el uso de GPS en teléfonos celulares.

En el mercado existe una gran cantidad de sistemas embebidos utilizados en modelos de teléfonos, mediante los cuales es posible el uso del sistema de posicionamiento global. Este funciona mediante la red GSM para comunicarse con un receptor GPS.

Los teléfonos establecen comunicación con los receptores mediante tecnología inalámbrica, en un inicio *Bluetooth*, y más actualmente con sistemas más robustos, para realizar esta función específica. Para esto, el sistema es diseñado con un dispositivo de navegación que interpreta los datos obtenidos de los satélites.

La mayoría de teléfonos celulares cuentan con un sistema operativo, bien de software libre o privativo como *Symbian OS* y *Microsoft Windows Mobile*. Lo anterior permite que prácticamente cualquier *smartphone* sea capaz de incorporar la función GPS y localizar cualquier punto siempre que sea usado. De hecho, existen muchas aplicaciones que utilizan esta función. Entre las más recientes y útiles se puede mencionar a Waze, un sistema de tráfico y navegación que mantiene informados a los usuarios sobre las rutas más óptimas para llegar a determinado lugar, o simplemente brindar información relevante respecto al tránsito vehicular.

La integración de la tecnología GPS en la telefonía celular ha revolucionado la interactividad del usuario. Significa un avance tan grande que hoy forma parte de la vida cotidiana. Llega incluso a utilizarse en juegos de realidad aumentada, como el revolucionario Pokémon Go.

2. APLICACIÓN *WEB* PARA EL RASTREO SATELITAL DE VEHÍCULOS TERRESTRES

La elaboración del sistema se puede abordar al dividir su estructura en una parte de *software* y otra de *hardware*. La primera engloba el código necesario para que el sistema como tal funcione; en esta parte se describe cada uno de los lenguajes de programación utilizados. Se cuenta, además, con una aplicación *web*; es decir, el sitio o página en el cual el usuario puede acceder al sistema través de Internet. Descrito de otra forma, la parte en la cual se codifica en un lenguaje soportado por los navegadores *web*.

Se aborda la relación que existe entre servidor y cliente, así como la utilización final de este mediante una aplicación desarrollada para que sea ejecutada, ya sea desde una computadora o un teléfono celular.

2.1. Servidores *web*

Un servidor *web* es un programa informático que permite que una computadora cumpla funciones específicas, de manera que brinde servicios a otras máquinas o a usuarios finales, denominados clientes. El término puede referirse al programa como tal o a la computadora que lo posee. En informática se le denomina a estos programas “el lado del servidor” y a la interfaz con la que interactúa con el usuario, “el lado del cliente”.

Del lado del servidor, se establecen conexiones, ya sea unidireccionales o bidireccionales, síncronas o asíncronas con el cliente, mediante un código de

programación en un lenguaje o aplicación. Del lado del cliente el código es compilado y ejecutado por un navegador *web*.

Al establecer comunicación entre ambos, se establece un protocolo, que es una serie de pasos entre ambos para interactuar. El protocolo que regularmente se utiliza es el HTTP, que pertenece a la capa de aplicación en el Modelo OSI.

2.1.1. Funcionamiento de un servidor *web*

La funcionalidad de un servidor *web* es de gran importancia en el mundo de la informática. El servidor es el encargado de integrar la información a la cual se accede desde diferentes dispositivos o plataformas, sean de *software* o *hardware*. El lenguaje más utilizado para integrar información es XML.

El principio de funcionamiento se denomina "Modelo cliente-servidor". Se refiere a que un servidor se encarga de prestar el servicio, y un cliente a utilizar tal servicio. De lo anterior se puede decir que un cliente *web* es un programa mediante el cual el usuario solicita a un servidor, compartir información. La información se transfiere mediante una interfaz de comunicación llamada Protocolo HTTP. La información compartida entre ambos, es un conjunto de código en lenguaje HTML.

El cliente *web* realiza una función de intérprete y muestra la información al usuario en el formato correspondiente. Ejemplos de clientes *web* son los navegadores de Internet, como Google Chrome, Microsoft Internet Explorer o Mozilla Firefox.

2.1.1.1. La relación servidor-cliente

Diversas aplicaciones se ejecutan en un entorno cliente-servidor. Esto significa que los equipos clientes (equipos que forman parte de una red) contactan a un servidor; un equipo generalmente muy potente en materia de capacidad de entrada y salida, que proporciona servicios a los equipos clientes. Estos servicios son programas que proporcionan datos como la hora, archivos, o una conexión.

Los servicios son utilizados por programas llamados “programas clientes” que se ejecutan en “equipos clientes”. Por eso se utiliza el término "cliente" para referirse a un programa que se ha diseñado para ejecutarse en un equipo cliente, capaz de procesar los datos recibidos de un servidor.

A manera de describir la forma de trabajo entre los clientes y los servidores, se define como cliente a la computadora que pide información a otra mediante la aplicación de un programa llamado cliente. Este contacta con el servidor y da formato a la petición de la información y a la respuesta. Un servidor es una computadora que ofrece la información mediante la aplicación de un programa llamado servidor, que recibe la información y la procesa, y da respuesta a la petición del cliente.

El servidor *web* se ejecuta en una computadora y se mantiene a la espera de peticiones por parte de un cliente (un navegador *web*), que responde a estas peticiones mediante una página *web* que se mostrará en el navegador.

2.1.1.2. Aplicaciones del lado del servidor

En el servidor *web* se ejecuta la aplicación, la que a su vez genera código HTML; el servidor envía al cliente este código recién creado por medio del protocolo HTTP.

Las aplicaciones en el lado del servidor suelen ser la mejor opción para desarrollar aplicaciones *web*. La razón es que, al ejecutarse en el servidor y no en la máquina del cliente, este no necesita ninguna capacidad añadida para ejecutar la aplicación, como sí ocurre en el caso de ejecutar aplicaciones que contengan *scripts* con JavaScript o Java.

Una aplicación en el lado del servidor es cualquier programa o conjunto de instrucciones diseñadas con la finalidad de que un servidor *web* las procese para realizar alguna acción. Las aplicaciones del lado del servidor están escritas mediante un lenguaje de programación. Entre los que más se utilizan están PHP, ASP, Perl, Python o Rubi.

2.1.1.3. Aplicaciones del lado del cliente

El cliente *web* es el encargado de ejecutar las aplicaciones en en la máquina del usuario. Son las aplicaciones tipo Java "*applets*" o Javascript. El servidor proporciona el código de las aplicaciones al cliente y éste las ejecuta mediante el navegador *web*. Por tanto, es necesario que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones también llamadas *scripts*.

Generalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje Javascript y Java, aunque pueden añadirse más lenguajes mediante el uso de extensiones, llamadas *plugins*.

2.1.2. Servidores *web* más usados

En el mundo informático el servidor dominante ha sido Apache. Actualmente le compite de cerca Microsoft. Existe un significativo porcentaje de sitios activos al día de hoy que no usan ninguno de los descritos a continuación.

Se describe algunos de los más utilizados.

2.1.2.1. Apache

Está diseñado para ser un servidor *web* potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Su integración multiplataforma hace que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular.

Este diseño permite a los administradores de sitios *web* elegir qué características van a ser incluidas en el servidor al seleccionar los módulos que se van a cargar, ya sea al compilar o al ejecutar el servidor. Este es el más común y más utilizado en todo el mundo.

Además, es gratuito y de código abierto, así que se podría decir que corre casi sobre cualquier plataforma. Apache muestra una filosofía, al igual que Linux, de toda una comunidad de cooperación dentro de Internet, capaz de producir aplicaciones de gran robustez y funcionalidad alta.

2.1.2.2. Microsoft IIS

Es el servidor *web* de Microsoft, el IIS (*Internet Informatión Server*). Es el motor que ofrece esta compañía a modo profesional. Con él es posible programar en ASP (*Active Server Pages*) las cuales funcionan de manera parecida al lenguaje PHP. Este servidor posee componentes programables desde ASP y se accede a cada uno de sus módulos para una función específica.

Este tipo de servidor lo llevan solo los sistemas Windows NT y sucesores como Windows 2000 Professional, Windows 2000 Advanced Server, Windows XP Professional, los cuales traen las versiones 4.0, 5.0 y 5.1.

A partir de los sistemas Windows XP el IIS no viene instalado por defecto, por lo que se debe realizar una instalación de forma manual.

2.1.2.3. Cherokee

Cherokee es, al igual que Apache, un servidor multiplataforma. Entre sus características más destacadas están la rapidez y la funcionalidad, sin dejar de ser liviano comparado con otros servidores *web*. Está desarrollado en lenguaje de programación C. Puede usarse como un sistema embebido y soporta complementos para aumentar sus funcionalidades. Es software libre, disponible bajo la Licencia Pública General de GNU.

Soporta la configuración de servidores virtuales; además, permite redirecciones y dispone de un panel de autenticación, *plain htpasswd* y *htdigest.*

2.1.2.4. Nginx

Nginx es un servidor de alto rendimiento y un proxy para protocolos de correo electrónico. Es de *software* libre y de código abierto, licenciado bajo la licencia BSD simplificada, aunque existe una versión comercial distribuida bajo el nombre de *nginx plus*. Es multiplataforma, lo que le permite correr en sistemas tipo Unix, como GNU/Linux, BSD, Solaris o Mac OS X, incluso en Windows.

El sistema es usado por una larga lista de sitios *web* conocidos, entre ellos, WordPress, Netflix, Hulu, GitHub, Ohloh, SourceForge, TorrentReactor y algunas partes de Facebook como el servidor de descarga de archivos zip pesados.

2.1.2.5. Lighttpd

Es un servidor *web* de código abierto optimizado para entornos de velocidad crítica, mientras que sigue siendo compatible con los estándares, seguridad y flexibilidad.

Lighttpd es *software* libre y se distribuye bajo la licencia BSD. Corre los sistemas operativos, GNU/Linux y UNIX de forma oficial. Para Microsoft Windows actualmente hay una distribución conocida como Lighttpd For Windows mantenida por Kevin Worthington.

Cuenta con virtual hosting, que le permite alojar varios dominios en la misma IP, además de CGI, SCGI y FastCGI y soporte para PHP, Ruby, Python y otros. Posee un entorno chroot y cifrado SSL.

2.2. Interfaz de rogramación de aplicaciones (API)

Una interfaz de programación de aplicaciones, API, del inglés *Application Programming Interface*, es un grupo de normas que especifican el protocolo que deben seguir las aplicaciones para comunicarse entre ellas. Su fin primordial es utilizar pequeños bloques de código, que pueden haber sido elaborados por un tercero, para realizar tareas específicas. Estos bloques de código pueden ser subrutinas, funciones y procedimientos; en el caso de la programación orientada a objetos, pueden ser métodos.

Las API, además de permitir la comunicación entre aplicaciones, permiten comunicarse con el sistema operativo o con bases de datos. Su importancia radica en que facilitan el uso de funciones ya existentes en otro software, para no reinventar soluciones, posiblemente ya encontradas, reutilizar código u optimizarlo para que funcione correctamente según las necesidades del programador.

Muchas de las API son de código abierto y libre; existen también las que no necesariamente por ser de uso libre son de código abierto, lo que implica que el desarrollador puede reservarse la información de cómo está construida internamente.

En los últimos años se ha incrementado y popularizado su uso gracias a las múltiples redes sociales como Twitter, Facebook, Youtube y otras plataformas online que prestan servicios de gran utilidad como Google Maps, Google Earth o WordPres.

2.2.1. El uso de las *API* en la *web*

Un programador que esté trabajando en un proyecto que requiera comunicación a Internet, podría hacer que en su sistema, cada vez que requiera de conexión, un programa realice la acción de comunicarse con la tarjeta de red. Esto sería reinventar el agua azucarada, ya que lo más fácil es utilizar una API que ya lo haga, con lo que aparte de reducir líneas de código en su programa, lo optimizaría, ya que no perderá tiempo en hacer una acción repetitiva.

De esta manera, los desarrolladores *web* hacen uso todo el tiempo de estas “herramientas de programación”. En una página *web* es muy común ver barras de búsqueda que enlazan directamente con el buscador de Google o botones que direccionan a Facebook, Twitter o Instagram, desde las funciones más sencillas hasta las más complejas, existen muchos ejemplos de API, algunos se describen a continuación.

2.2.1.1. CORBA

Esta API permite que programas desarrollados en distintos lenguajes, puedan trabajar en un mismo entorno, haciendo una convivencia heterogénea de software en un solo bloque.

Los trozos de programa son denominados objetos, y cada uno posee una interfaz única, además de una identidad. Cada objeto se podrá implementar en lenguajes distintos en cualquier sistema operativo.

Para hacer uso de ellos, un servidor *web* creará enlaces que harán referencia a cada uno, esperando a que el cliente los mande a llamar. Una vez

llamados los objetos, invocaran a las subrutinas que lo conforman. Dichas subrutinas son llamadas métodos.

El cliente manda a llamar los objetos y utiliza un lenguaje de descripción de interfaz o IDL, e invoca un acceso remoto hacia ellos.

2.2.1.2. Drupal

Drupal es un sistema de gestión de contenidos de código libre y abierto, ya que su estructura es modular y multipropósito. Es bastante confiable y eficiente para manejar la publicación en páginas *web*, desde archivos e imágenes, hasta realizar encuestas, sistemas de votación o preguntas y respuestas a modo de foro.

Tiene la característica de ser un sistema dinámico, por lo que en lugar de almacenar sus contenidos en sistemas de ficheros en el servidor, el texto de las páginas y otras configuraciones se almacenan en una base de datos y se editan utilizando un entorno *web*.

Cuenta con licencia de GNU/GPL, y su código está escrito en PHP, y trabaja con MySQL. Al ser de código libre y abierto, cuenta con una gran comunidad activa de usuarios, quienes brindan soporte todo el tiempo.

Su código es bastante robusto y gracias a ello, tiene una buena usabilidad y consistencia de todo el sistema. Su diseño es bastante útil para gestionar comunidades en foros de Internet, ofrece flexibilidad y adaptabilidad, lo que lo hace perfecto para desarrollar diferentes modelos de sitos *web*.

2.2.1.3. Glibc

La GNU C *Library*, más conocida como *Glibc*, se trata de la biblioteca estándar de lenguaje C de GNU. Cuenta con licencia GNU LGPL y al ser la librería de uno de los lenguajes de programación más famosos, C, es utilizada en casi la mayoría de programas.

Permite definir las llamadas la API y a otras funciones básicas de la misma, es bastante usada sobre todo en sistemas GNU y en general en distribuciones de Linux.

2.2.1.4. API de Windows

La interfaz de programación de aplicaciones de Windows, cuyo nombre en inglés es Windows API, se caracteriza porque sus funciones residen en bibliotecas dinámicas llamadas DLL, las cuales permiten que una aplicación corra en un sistema operativo determinado.

Utiliza nomenclaturas para hacer uso de la API de acuerdo a la versión de sistema operativo, de tal manera que indican si por ejemplo está diseñada para Windows XP o 7 y hace mención del conjunto de bibliotecas para cada caso, como Win32, por ejemplo.

Las funciones de la API pueden ser de depuración o manejo de errores o de configuración de entradas y salidas de dispositivos. También pueden administrar el manejo de la memoria o del consumo de energía del sistema. Otras controlan las unidades de almacenamiento o la información del sistema, incluso la interfaz gráfica del mismo.

2.2.2. API de *Google Maps*

Descritas las más famosas, mención aparte merece la que es utilizada en el presente sistema, la *Google Maps* API.

Google Maps es un servidor *web* que proporciona mapas en los cuales es posible realizar desplazamientos en tiempo real, cálculo de distancias, trazo de trayectorias, entre otros. Su entorno permite visualizar imágenes de mapas a modo de bosquejos de calles y rutas o como fotografías captadas por satélite prácticamente de todo el mundo.

Posee características importantes, como la capacidad de hacer acercamientos o alejamientos en un mapa, así como desplazamientos e incluso la posibilidad de ingresar una dirección, el nombre de una calle, un lugar famoso o un área completa.

Google Maps se basa en el sistema geométrico de referencia WGS842 para mostrar sus coordenadas, por lo que muestra la latitud y la longitud, positiva para Norte y Este, negativa para Sur y Oeste.

La API de *Google Maps* es muy utilizada en diversidad de servidores, desde páginas web hasta servicios de rastreo y vigilancia, los cuales funcionan con algún GNSS. Su utilización es de uso libre y por su robustez, es bastante precisa, salvo en países en los que por motivos militares está restringida la búsqueda de ciertos lugares.

Para utilizarla, ya sea en alguna aplicación móvil o en un sitio web, se debe mandar a llamar mediante código de programación en lenguaje

JavaScript. Al hacer esto también es posible modificar formatos de estilo, como alto y ancho del entorno, dependiendo de las necesidades de la aplicación.

En la siguiente imagen se aprecia un sitio *web* en el que se muestra un mapa de la Ciudad de Guatemala. En rojo se ve un trozo de código JavaScript en el que se manda a llamar a la API de *Google Maps*.

Figura 11. **Uso de API de *Google Maps***

Fuente: *Mapa de Guatemala con sus calles y avenidas gracias a Google*! http://desdeguate.com/blog/2007/09/14/mapa-de-guatemala-con-sus-calles-y-avenidas-gracias-a-google/. Consulta: 18 de enero de 2017.

Como la API cuenta con un motor de búsqueda de coordenadas, permite realizar localizaciones en un tiempo considerablemente corto. A través de lecturas de coordenadas almacenadas en una base de datos, se obtiene un sistema de rastreo bastante eficiente.

2.3. Diseño de aplicaciones *Android* en *APP Inventor*

Android es un sistema operativo desarrollado con base en Linux, inicialmente pensado para teléfonos móviles. De la misma manera que iOS, Symbian y *Blackberry* OS, con la llegada de los teléfonos inteligentes se popularizó su uso, ya que es un *software* gratuito y multiplataforma.

Es un sistema robusto que permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema provee todas las herramientas necesarias para desarrollar aplicaciones que permiten acceder a las funciones de un teléfono móvil; es decir, desde realizar llamadas hasta la utilización de las herramientas como *Bluetooth* o GPS. La creación de aplicaciones resulta bastante amena, ya que utiliza el lenguaje de programación Java.

Como se ha mencionado, una de las razones por las que se masificó su utilización es porque su uso es completamente libre, por lo que para programar alguna aplicación y usarla posteriormente en un teléfono no se debe pagar nada. Lo anterior es propicio para que exista una extensa comunidad de desarrolladores y soporte, a su vez, intensifican la velocidad con la que las aplicaciones ya existentes se actualicen constantemente.

Una aplicación en *Android*, no es más que un programa diseñado, creado y montado para un teléfono celular. Las aplicaciones pueden abarcar un sinfín de necesidades móviles, desde programas para hacer llamadas, enviar mensajes, hasta aquellos que permiten activar alguna función específica, como el GPS, la cámara, entre otras.

Figura 12. **Ejemplo de aplicación programada en *Android***

Fuente: *Aplicaciones Android de ejemplo*. https://code.tutsplus.com/es/tutorials/android-sample-apps--mobile-12528. Consulta: 25 de febrero de 2017.

3. DESCRIPCIÓN DE COMPONENTES UTILIZADOS EN LA ELABORACIÓN DEL SISTEMA DE DETECCIÓN SATELITAL

Puntualizados los elementos importantes en el concepto rastreo y geolocalización, y tras abordar las características y tipos de servidores *web,* se procede a detallar los elementos utilizados en el prototipo diseñado.

3.1. Introducción al módulo GPS6MV2

El módulo GPS6MV2 es una placa de circuito impreso. Contiene, además de los elementos necesarios para transmitir datos de forma serial, un circuito integrado de la serie NEO-6, una memora tipo EEPROM, una batería para mantener los datos en la memoria EEPROM sin que se borren y una antena de cerámica, para establecer comunicación satelital.

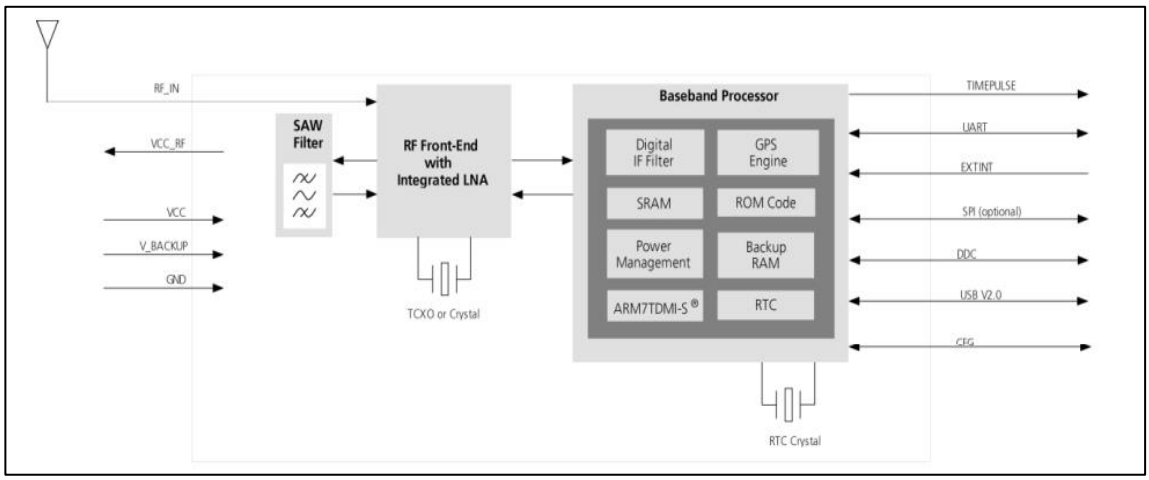
Es desarrollado por la compañía suiza U-Blox, quienes además se dedican a la fabricación de distintos tipos de semiconductores inalámbricos y módulos para los mercados de consumo, automoción e industriales (100).

Figura 13. **Módulo GPS6MV2**

Fuente: Thingnovation. *Productos destacados*- http://thingnovation.com. Consulta: 25 de febrero de 2017.

Los integrados NEO-6 son una familia de receptores GPS independientes que utilizan un motor de posicionamiento de alto rendimiento. Estos receptores de señales satelitales se caracterizan por ser flexibles y rentables, ofreciendo una buena conectividad en un empaquetado miniatura de 16 x 12,2 x 2,4 mm, que facilita su uso en sistemas embebidos de pequeñas dimensiones y por su arquitectura. Son ideales para dispositivos móviles alimentados por baterías.

Figura 14. **Diagrama de bloques del módulo GPS6MV2**



Fuente: *NEO-6 u-blox 6 GPS Modules*. https://www.ublox.com/sites/default/files/products/documents/NEO-6_DataSheet. Consulta: 25 de febrero de 2017.

El motor de posicionamiento U-BLOX 6, utiliza 50 canales con *TTFF* de menos de 1 segundo. Esto permite que sea capaz de realizar búsquedas en espacio paralelo en tiempo y frecuencia, pudiendo encontrar satélites de forma instantánea.

3.2. Microcontrolador ATmega328

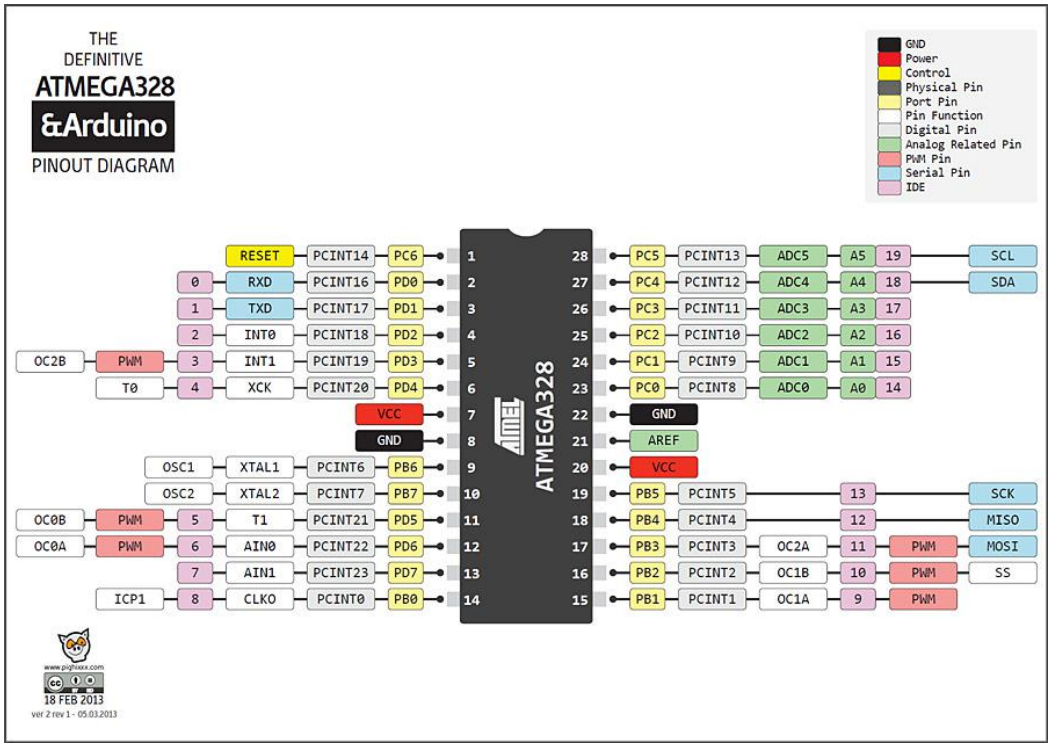
Es un microcontrolador con arquitectura AVR, una familia RISC de Atmel. Es un circuito integrado de alto rendimiento combinando 32 KB ISP flash con la capacidad de lectura y escritura en simultáneo.

Posee además 1 KB de memoria EEPROM, 2 KB de SRAM, 23 líneas de entradas y salidas y de propósito general, 32 registros de proceso general, tres temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interface serial

orientada a byte de dos cables, SPI en puerto serial, 6 canales, conversor analógico/digital de 10 bits, un temporizador con función de “perro guardián” que puede ser programable con oscilador interno, además de cinco modos de ahorro de energía seleccionables mediante *software*.

El integrado funciona con voltajes entre 1.8 y 5.5 voltios. Gracias a su robustez, alanza una respuesta de 1 MIPS, permitiendo ejecutar un gran número de instrucciones en un solo ciclo de reloj, sumándole a su velocidad de proceso, un bajo consumo de energía.

Figura 15. **Descripción de pines ATmega328**



Fuente: Patagoniatec. *IC Atmega328*. http://saber.patagoniatec.com/atmega-328-micro-microcontrolador-chip-uno-nano-pro-mini-tutorial-arduino-arduino-argentina-ptec/. Consulta. 25 de febrero de 2017.

3.3. Raspberry Pi, la computadora de bajo costo

La Raspberry Pi, también conocida coloquialmente como “Raspi”, es un dispositivo que comprende una computadora completa en un circuito reducido del tipo SBC. Las siglas SBC provienen del término en inglés *Single Board Computer*, el cual se refiere a una computadora de placa de pequeñas dimensiones que integra su funcionalidad en un microprocesador con memoria RAM, un módulo de entradas/salidas y las características necesarias de una computadora convencional, en una placa base. Su fabricación surge a partir de la tendencia, luego de la aparición de las computadoras portátiles, a desarrollar sistemas que incluían funcionalidades como sonido, red e incluso gráficos en una sola placa.

Una ventaja importante de la Raspberry Pi es que ofrece la posibilidad de acceso remoto si se dispone de una red. Este se puede realizar de varias formas: a través de un terminal Telnet o SSH, mediante una Computadora Virtual de Red (VNC, *Virtual Network Computer*) o utilizando el Protocolo de Escritorio Remoto (RDP - *Remote Desktop Protocol*).

El método más rápido y directo para ingresar de manera remota a la Raspberry Pi es utilizar el protocolo Telnet. En caso de que se quiera disponer de una conexión segura utilizar el protocolo SSH (*Secure Shell*), Raspbian ya viene con lo necesario para poder utilizar ambos protocolos. Solo se tiene que ejecutar un terminal TELNET o SSH en la computadora remota, indicar la dirección IP y la contraseña de usuario de la Raspberry Pi para tener el acceso a ella.

3.4. Lenguajes de programación utilizados en el diseño del sistema de detección satelital

Una vez referida la relación Cliente-Servidor y las características de ambas partes, se puede abordar específicamente de qué se encargará cada una en el presente sistema.

Desde una versión de lenguaje C++ para operar el ATmega328, hasta una llamada a la base de datos para extraer parámetros, pasando por un noble lenguaje Python para ejecutar pequeños programas, a continuación se describe de manera breve, los diferentes lenguajes que forman parte del sistema de rastreo.

3.4.1. Programación en C++

El microcontrolador ATmega se puede programar desde varios lenguajes. Su peculiar popularidad se debe a su implementación en placas de desarrollo Arduino o Genuino en muchas de sus versiones.

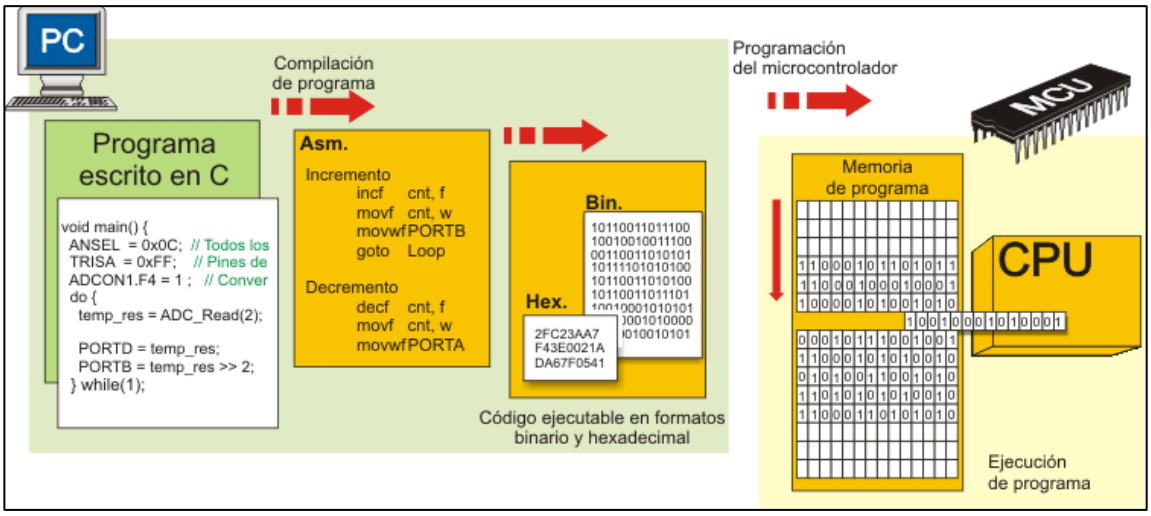
En el caso de Arduino, utiliza un IDE (*Integrated Development Environment*). Es una versión que simplifica el modo de programación, utiliza librerías prefabricadas, aunque no es una constante, ya que se pueden implementar funciones a código puro. Esta IDE es también utilizada en controladores de Texas Instruments, bajo el nombre “Energia”.

Como tal, el lenguaje de programación C++ fue diseñado a mediados de los años 80. Fue desarrollado como una manera continuar con el éxito del lenguaje C, incluyendo mecanismos que permitiesen la manipulación de

objetos. Se dice que C++ es un lenguaje híbrido o multiparadigma ya que se le añadieron funciones para la programación estructurada y la orientada a objetos.

Existe una versión de este lenguaje, propietario de Microsoft, llamada C#, que mezcla las características básicas de C++ con Java. Este lenguaje forma parte de la plataforma de desarrollo ".NET".

Figura 16. **Elementos básicos en la programación de C++**



Fuente: *Lenguaje de programación C++*. https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-c/. Consulta: 26 de marzo de 2017.

3.4.2. Programación en Python

Python es un lenguaje de programación de los llamados "interpretados", lo que significa que es capaz de analizar y ejecutar otros programas. Es un lenguaje de fácil aprendizaje que cuenta con estructuras de datos de alto nivel y una visión simplificada de la programación orientada a objetos. La estructura de su sintaxis y su escritura dinámica, junto con sus características de lenguaje

interpretado, permiten que sea ampliamente utilizado en el desarrollo rápido de aplicaciones en la mayoría de plataformas.

Python y su biblioteca son de libre distribución y están disponibles en su sitio *web.* El intérprete de Python se puede extender agregando nuevas funcionalidades y tipos de datos implementados en otros lenguajes como C o C++. Python también puede usarse como un lenguaje de extensiones para distintas aplicaciones.

Figura 17. **Logo de Python**

Fuente: Python. *Funciones definidas*. http://www.python.org/. Consulta: 26 de marzo de 2017.

En el mundo de los usuarios de Python existe la denominada Filosofía Python, la cual es una analogía a la filosofía de Linux o Unix. Se dice que el código que sigue los principios de Python es “pythonico". Entre los usuarios de código abierto hay una marcada preferencia por la utilización de este lenguaje; varias distribuciones de Linux lo traen instalado por defecto. Es esencialmente útil en el desarrollo de aplicaciones con Raspberry Pi.

4. PROPUESTA DE DISEÑO

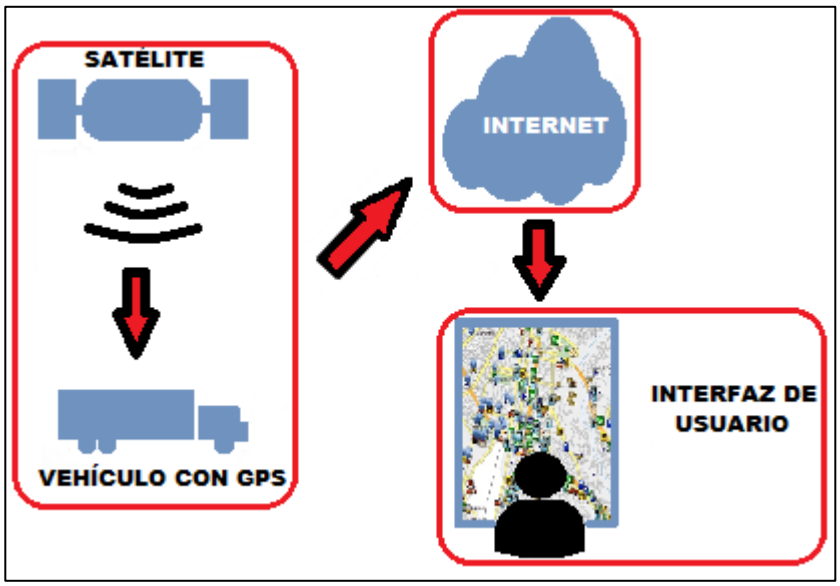
En informática se le da el nombre de “La nube” al procesamiento y almacenamiento masivo de datos en servidores web, de manera que estos alojen información de determinado usuario, para su posterior utilización. Considerando este paradigma, se diseña un sistema de rastreo satelital, con todos sus componentes integrados dentro de un sistema embebido, que sea capaz de procesar información para poder almacenarla en “La nube” y que sea consultada por cualquier usuario que tenga acceso a esta.

Por medio del módulo GPS6MV2 se establece el protocolo de comunicación con los satélites del sistema GPS, se envía la información al microcontrolador ATmega328, quien la transfiere a la Raspberry Pi. Una vez la Raspi obtiene los datos, los envía al servidor *web*.

El monitoreo y visualización de trayectorias y ubicaciones se realiza a través de una aplicación *Android*, utilizando una API de *Google Map*. Se logra una solución práctica para el manejo y monitoreo de flotillas de vehículos, para el control de distintos parámetros como la distancia recorrida, el uso adecuado del combustible y las rutas óptimas que se deben tomar. A la vez, la detección brinda la posición del vehículo en todo momento, para facilitar su rastreo en caso de percances, desperfectos mecánicos o incluso robo.

El esquema para el diseño del sistema está dividido en los siguientes bloques:

Figura 18. **Diagrama de bloques del sistema**



Fuente: elaboración propia.

Cada bloque del diagrama representa una etapa que, a su vez, está dividida en procesos específicos. Brevemente cada una de las etapas se describe de la siguiente manera:

La primera de ellas engloba la transmisión de ondas electromagnéticas entre el satélite y el módulo GPS montado en el vehículo automotor. La segunda es la comunicación entre el procesador de datos y el servidor *web.* Este último se comunica a través de la cuarta etapa, la interfaz del usuario final. En este caso el usuario puede monitorear y hacer consultas de forma remota.

La primera etapa está formada por el sistema sensorial constituido por el módulo GPS6MV2 y el microcontrolador ATmega328. El módulo recibe señales de satélites y determina la longitud y latitud en la que está posicionado. Luego se envían los datos de forma serial al microcontrolador.

La segunda comprende la comunicación entre el microcontrolador y el procesador Raspberry Pi. Los datos son recibidos mediante el puerto USB del procesador. Este dispositivo se utiliza por su pequeño tamaño y gran capacidad de procesamiento de datos. Raspberry Pi almacenará los datos recibidos del bloque anterior y los procesará para su posterior envío a través de Internet.

Como se requiere una conexión a Internet, es necesario instalar una serie de paquetes para montar los servidores en la Raspberry Pi. Con los servidores montados se puede realizar la conexión a Internet. Para este caso se utiliza un módulo wi-fi USB para la conexión inalámbrica.

Un *script* programado en Python ejecuta las instrucciones para que la información sea cargada a la base de datos montada en el servidor *web*. El usar un servidor montado en Internet agiliza los procesos y optimiza las consultas remotas a la base de datos.

La última etapa está formada por la interfaz final de usuario. Se utiliza un API de *Google Maps* que permite tener acceso a la base de datos montada en el servidor *web*. De esta manera es posible utilizar aplicaciones hechas en *Android*, con la ayuda de APP Inventor, para su uso en dispositivos móviles o de escritorio. Con lo anterior, el usuario podrá monitorear en tiempo real el desplazamiento o trayectoria de un vehículo en el que se haya montado el sistema GPS.

4.1. Etapa I. Detección satelital y obtención de datos

Una de las partes fundamentales en el desarrollo del presente sistema es, sin lugar a dudas, la obtención de datos de ubicación mediante los satélites; es

decir, realizar el proceso de comunicación con el GPS y manipular esa información.

En este caso se utiliza el módulo GPS6MV2 por su practicidad uso y su precisión de unos cuantos metros, sumado a que tiene un precio bastante bajo; sin embargo, cualquier otro dispositivo con características similares sería útil para realizar esta función, la recolección de datos.

4.1.1. Elementos de un rastreador satelital

Lo primero es establecer comunicación con la constelación de satélites sobre el sistema que se va a utilizar. En la actualidad, algunos dispositivos receptores son compatibles con GPS, GLONASS y Galileo. Para este caso, será suficiente uno que trabaje con GPS.

La comunicación con GPS se puede dividir en una estructura compuesta por una parte espacial, una parte de control y una de recepción.

4.1.1.1. Parte espacial

Tal y como está descrito en el primer capítulo, un conjunto de satélites forman el sistema, tanto de navegación como de comunicación, y permiten interactuar con las diferentes señales que envían y reciben cada uno de los receptores.

La constelación está diseñada para garantizar una cobertura global en cualquier parte del planeta y proporciona una cobertura de cuatro a ocho satélites por encima del horizonte. Estos emiten señales en varias frecuencias.

Todos cuentan con relojes atómicos para garantizar una sincronización con los dispositivos en Tierra, algunos son de cesio, rubidio o cristal de cuarzo.

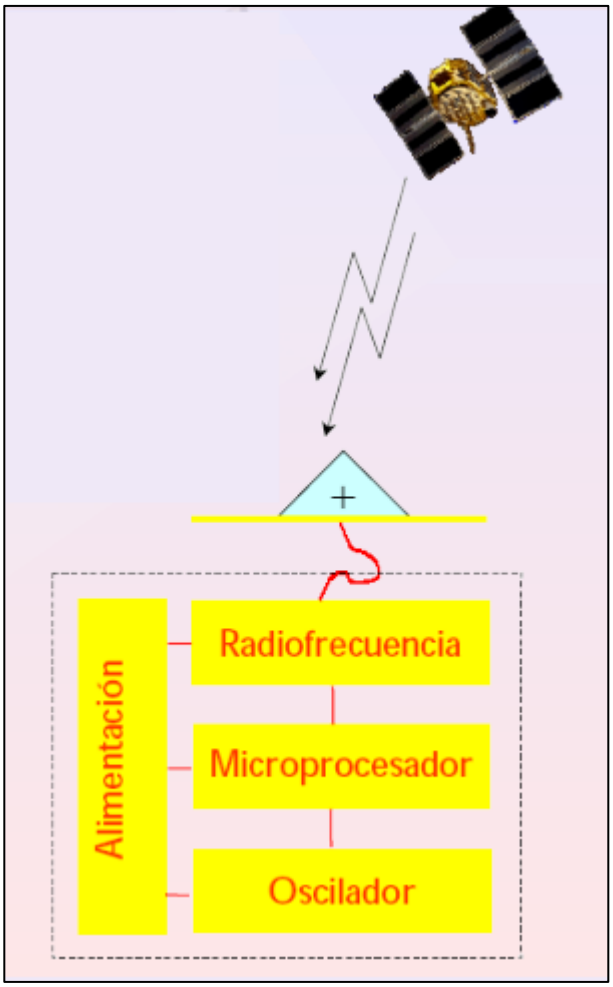
4.1.1.2. Parte de control

El dispositivo GPS cuenta con un circuito integrado que se mantiene emitiendo ondas electromagnéticas. Esto permite que realice su función de emisor-receptor.

En esta parte, los elementos del receptor, las señales emitidas (ondas de radio frecuencia), el microprocesador interno, el oscilador y la fuente de alimentación del dispositivo realizan su operación de manera secuencial y repetitiva.

Mediante la antena receptora se logra una cobertura hemisférica y omnidireccional. Al conectar con por lo menos tres satélites, pone en marcha su receptor de tipo heterodino, el cual, a través de una mezcla de frecuencias, permite adecuar la señal recibida a una antena de baja frecuencia para ser manejada de forma electrónica.

Figura 19. **Elementos del receptor GPS**



Fuente: Sistemas GNSS. Funcionamiento, posicionamiento y precisión. http://lagc.uca.es/web_lagc/docs/curso_rap/Presentacion_II.pdf. Consulta: 26 de marzo de 2017.

4.1.1.3. Parte de recepción

Es aquí donde el dispositivo recibe información procedente del segmento espacial y calcula su posición. Mediante un cálculo matemático, y con el ya mencionado método de la triangulación y el cálculo de la distancia, se determinan las coordenadas, latitud y longitud.

Posterior a realizar el cálculo de coordenadas, las señales son convertidas en pulsos digitales, las cuales formarán la trama que se enviará al microcontrolador de forma serial.

4.1.2. Comunicación entre módulo GPS6MV2 y ATmega328

Una vez el dispositivo receptor haya obtenido los datos necesarios para realizar la detección de un punto, es momento de enviar la trama de pulsos digitales al microcontrolador. La comunicación entre ambos se realiza de forma serial, por lo que se debe establecer un protocolo de transmisor y receptor.

Utilizado el IDE de Arduino, se utiliza la librería *TinyGPS,* la cual permite sincronizar la comunicación entre el GPS6MV2 y el microcontrolador ATmega328. La librería cuenta con códigos de ejemplo entre los cuales se puede realizar desde una conexión sencilla hasta una en la cual se pueden obtener tanto las coordenadas, como la hora de detección y la fecha. Para este diseño, la funcionalidad se ha simplificado a obtener únicamente la latitud y longitud.

Mediante las siguientes líneas de programación se manda a llamar a la librería en mención y a la que permite la comunicación de forma serial.

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
```

A continución es necesario crear la instancia de un objeto de la siguiente manera:

```
TinyGPS gps;
```

Como es necesario determinar qué pines del microcontrolador serán asignados como transmisor y receptor del módulo serial, se definen así:

```
SoftwareSerial ss (4, 3);
```

Donde el objeto es ss y 4 y 3 son los pines asignados como transmisor y receptor, respectivamente.

El *baudrate* o velocidad de transmisión en baudios es un punto clave siempre que se trabaja con el módulo serial, tanto en la comunicación entre el módulo GPS y el ATmega328, como en la de este último y la *Raspi.* La sincronización dependerá de este factor. En el módulo ss es configurado a 9600 baudios y el puerto, a 115200. El desarrollador de la librería recomienda hacerlo de esta manera, aunque para otros dispositivos GPS, puede variar.

```
Serial.begin(115200);
 ss.begin(9600);
```

La librería carga un caracter a la vez utilizando el método “encode ()”. Cuando este devuelve *true*, una sentencia válida acaba de cambiar el estado interno del objeto TinyGPS, y realiza la lectura del módulo.

```
bool nuevoDato = false;

  for (unsigned long inicio = millis(); millis() - inicio < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      if (gps.encode(c))
        nuevoDato = true;
    }
```

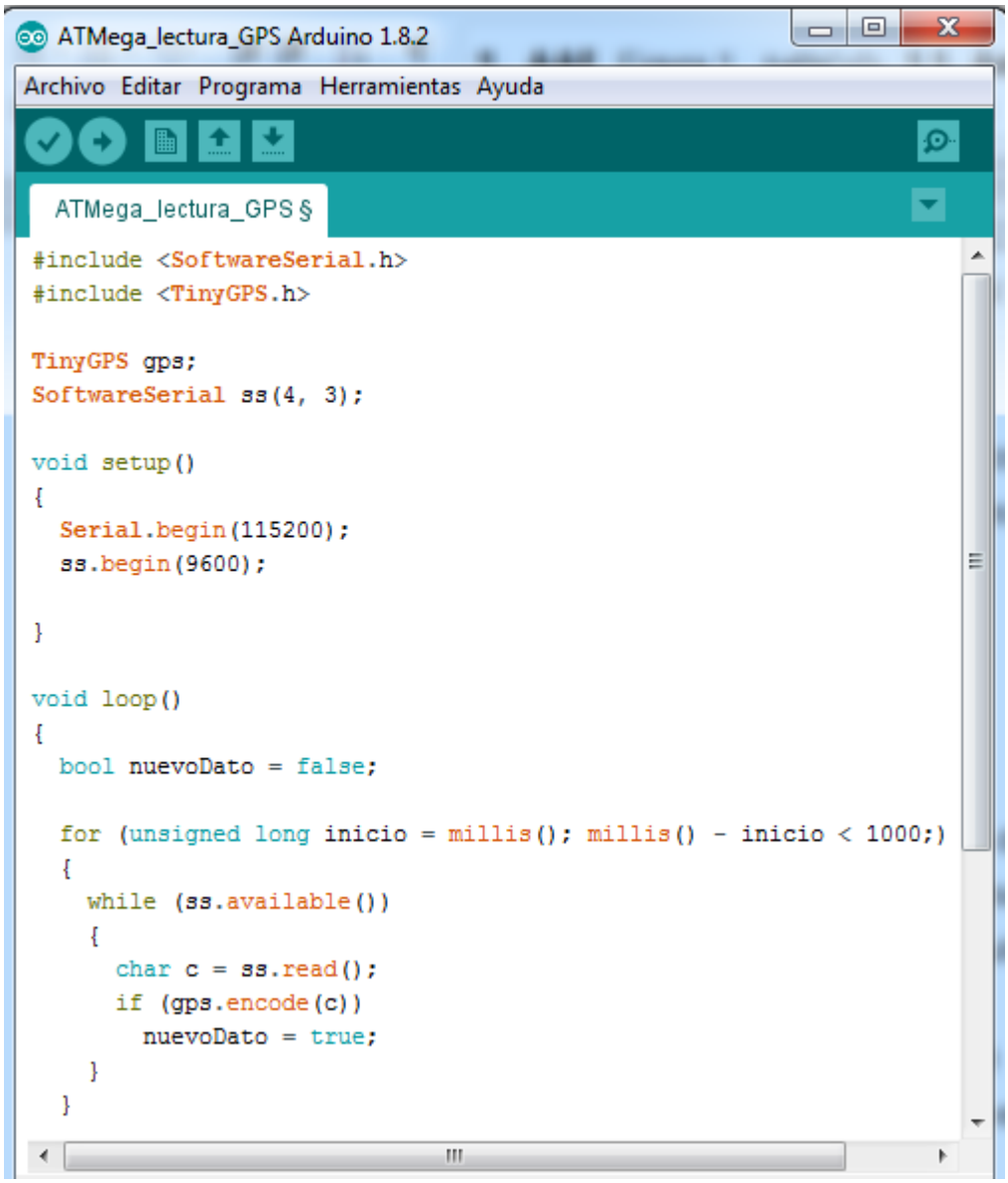
Se definen las variables de punto flotante *flat* y *flon*. Mediante la función “f_get_position” se asignan los caracteres a flat y flong.

```
float flat, flon;
gps.f_get_position(&flat, &flon);
```

Se imprimen los valores obtenidos, los cuales serán enviados a la Raspberry Pi, de igual manera vía serial. Al finalizar se agrega un retardo de mil milisegundos que servirá de sincronización con la comunicación hacia la *Raspi.*

```
Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
Serial.print(";");
Serial.println(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
delay (1000);
```

Figura 20. **Programación del ATmega328**



Fuente: Obtenida del IDE de Arduino.

4.2. Etapa II: entorno *web*

La Raspberry Pi se encarga de procesar la información recibida del microcontrolador. Se crea un programa en Python que ejecute una lectura constante en el puerto serial y que, a su vez, envíe lo recibido a una base de datos.

El servidor *web* es una parte importante del sistema, ya que almacena las coordenadas obtenidas para su posterior utilización en el trazo del mapa.

4.2.1. Montaje de un servidor *web* en Raspberry Pi

Para la elaboración del servidor *web* se procede a configurar la Raspberry Pi como tal. En este sistema se utilizará, a modo de demostración, la misma Raspi como procesador de datos y como servidor. No necesariamente debe hacerse de esta manera; inclusive se podría utilizar algún alojamiento virtual en la red, de los muchos que existen.

Se tendrá acceso vía Internet a los datos almacenados en el servidor. Lo primero es configurar la Raspberry Pi con las características necesarias para que funcione como servidor *web*.

Para montar el servidor, es necesario instalar un conjunto de componentes que prestan sus servicios para las funciones requeridas. A este conjunto se le conoce como “LAMP”, acrónimo de Linux, Apache, MySQL y PHP.

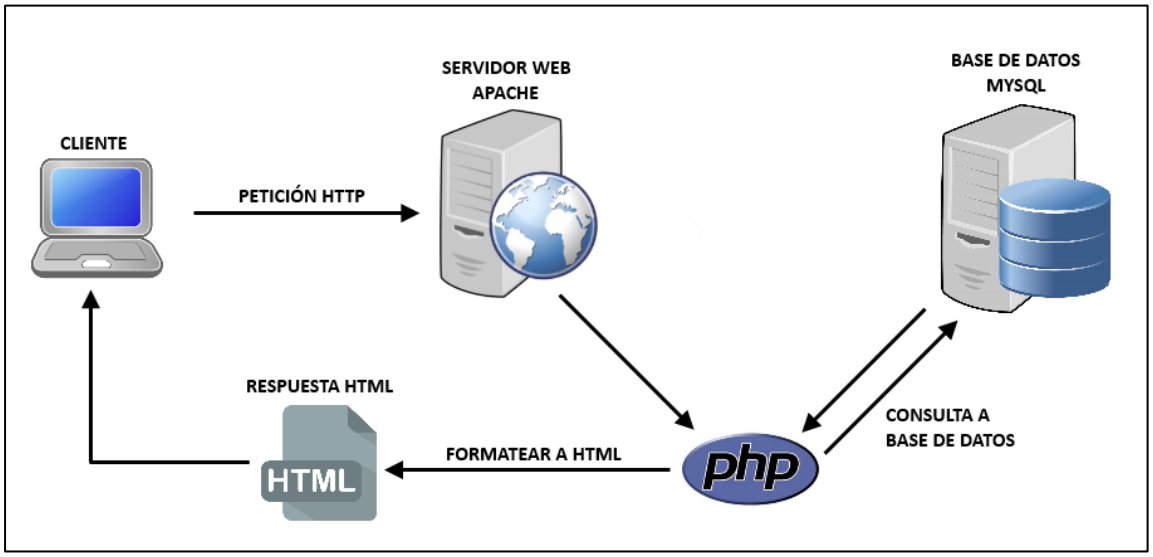
- Apache: el servidor *web* HTTP de código abierto.

- MySQL: lugar en donde se creará la base de datos.

- PHP (*Personal Home Pages*): lenguaje de programación que permite crear páginas dinámicas en el servidor *web*.

- PHPmyAdmin: permite administrar la base de datos a través de interfaz *web*.

Figura 21. **Funcionamiento de un servidor *web***



Fuente: Makers. *Raspberry PI como servidor web*. http://diymakers.es/raspberry-pi-como-servidor-web/. Consulta: 26 de marzo de 2016.

En la figura 21 se puede observar el procedimiento de un servidor web con la interacción de cada uno de sus componentes.

Antes de proceder a instalar LAMP, se debe reiniciar la Raspi. Toda manipulación de la Raspberry Pi en este sistema, se hace desde la terminal. Los comandos a ejecutar son los siguientes:

```
pi@raspberrypi:~ $ sudo apt-get upgrade
pi@raspberrypi:~ $ sudo apt-get update
```

4.2.1.1. Instalación de Apache

Para instalar el servidor *web* Apache, primero es necesario crear un grupo de usuarios para el servidor, de la siguiente manera:

```
pi@raspberrypi:~ $ sudo groupadd www-data
pi@raspberrypi:~ $ sudo usermod-a -G www-data www-data
```

Una vez realizado el anterior procedimiento, se instala Apache, con el siguiente comando:

```
pi@raspberrypi:~ $ sudo apt-get install apache2
```

4.2.1.2. Instalación de PHP

El lenguaje de programación PHP permite la creación de páginas *web* dinámicas. Se entiende por una página dinámica, aquella en las que la información presentada se genera a partir de una petición del usuario de la página. A diferencia de lo que ocurre con las páginas estáticas, en las que su contenido se encuentra predeterminado, en las páginas dinámicas la información aparece inmediatamente después de una solicitud realizada por el usuario.

Se procede a instalar el lenguaje PHP en la Raspberry Pi:

```
pi@raspberrypi:~ $ sudo apt-get install php5
```

Para completar la instalación, es necesario instalar una serie de paquetes y librerías que permiten su interacción con el servidor Apache y con MySQL.

```
pi@raspberrypi:~ $ sudo apt-get install libapache2-mod-php5 libapache2-mod-perl2
 php5 php5-cli php5-common php5-curl php5-dev php5-gd php5-imap php5-ldap php5-m
hash php5-mysql php5-odbc
```

4.2.1.3. Instalación de MySQL

MySQL es una base de datos de código abierto que sirve para almacenar datos para su posterior manipulación. Se realiza su instalación de la siguiente manera:

```
pi@raspberrypi:~ $ sudo apt-get install mysql-server mysql-client php5-mysql
```

Como todo gestor de datos, por seguridad es necesario establecer una contraseña. Al terminar la instalación, se debe iniciar el MySQL:

```
pi@raspberrypi:~ $ sudo service mysql start
```

Para ingresar a la base de datos, se debe escribir el siguiente comando:

```
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
```

Será solicitada la contraseña que se ha establecido en el momento de la instalación. Una vez ingresada, si el acceso es correcto, se muestra que se está en la línea de comandos de MySQL:

```
mysql>
```

Para salir de la línea de comandos de MySQL, se debe presionar Ctrl-C.

4.2.1.4. Instalación de PHPmyAdmin

PHPmyAdmin es un software en lenguaje PHP que permite administrar la base de datos MySQL a través de una interfaz *web*. Se instala de la siguiente manera:

```
pi@raspberrypi:~ $ sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin
```

En este punto, se debe seleccionar el servidor *web* que se ha instalado. Se configura Apache, la base de datos y luego la contraseña que se estableció en MySQL y una nueva para PHPmyAdmin.

Es necesario configurar PHPmyAdmin modificando el archivo:

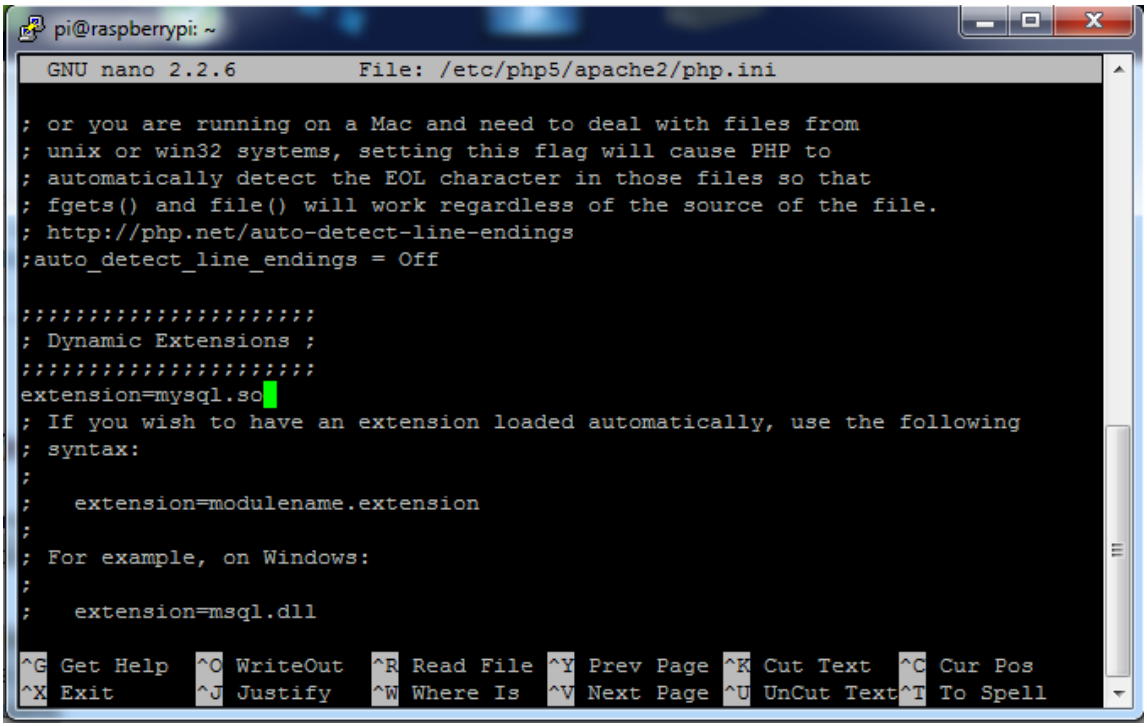
```
pi@raspberrypi:~ $ sudo nano /etc/php5/apache2/php.ini
```

Se debe escribir el siguiente comando en una nueva línea, en la sección "*Dynamic Extensions*":

```
extension=mysql.so
```

En la siguiente figura se muestra la edición del archivo:

Figura 22. **Configuración de PHPmyAdmin**



Fuente: generada en la terminal de Raspbian.

Para que los cambios en la Raspberry Pi se guarden correctamente, es necesario reiniciarla de la siguiente manera:

```
pi@raspberrypi:~ $ sudo reboot
```

Para finalizar la configuración de PHPmyAdmin, se debe ejecutar dos comandos más:

```
pi@raspberrypi:~ $ sudo ln-s /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf
pi@raspberrypi:~ $ sudo /etc/init.d/apache2reload
```

4.2.2. Creación de la base de datos

Comúnmente se utiliza el entorno de PHPmyAdmin para la administración de tablas y bases de datos. Su uso es bastante práctico e intuitivo, pero este trabajo se realiza en línea de comandos. Esto no significa que sea innecesario instalar PHPmyAdmin; lo es en el sentido de configurar el servidor *web* como tal.

En primer lugar se accede a MySQL con el comando que se mostró anteriormente:

```
pi@raspberrypi:~ $ mysql -u root -p
```

Una vez escrita la contraseña establecida, se tendrá acceso al entorno de la línea de comandos de MySQL.

```
mysql>
```

4.2.2.1. Una nueva base de datos

El proceso de crear una nueva base de datos desde la línea de comandos es bastante sencillo. Se crea la base de datos “gps”. En esta se crea una tabla llamada “coordenadas”, que contiene la información de latitud y longitud.

Se crea la base de datos “gps” con el siguiente comando:

```
mysql> mysql> CREATE DATABASE gps;
```

Para que MySQL sepa qué base de datos se manipulará, se debe indicar mediante el comando siguiente:

```
mysql> USE gps;
```

Al indicar sobre qué base se trabaja, se procede a crear la tabla en la cual se alojará la información.

4.2.2.2. Creación de una tabla

Ya configurada la base de datos e indicado sobre cuál se trabajará, se crea una tabla para almacenar las coordenadas que envía el microcontrolador hacia la Raspbery Pi.

Los datos se manipulan por medio de cadenas de caracteres, por lo que la variable a utilizar es de modo *string*. En MySQL los *strings* se utilizan con un tipo de variable *varchar.* Se debe especificar el tamaño del *string;* para el caso de la latitud y longitud, se manejan cadenas de 10 caracteres.

Se crea la tabla con el siguiente comando:

```
mysql> mysql> CREATE TABLE coordenadas (lat VARCHAR(10), lng VARCHAR(10));
```

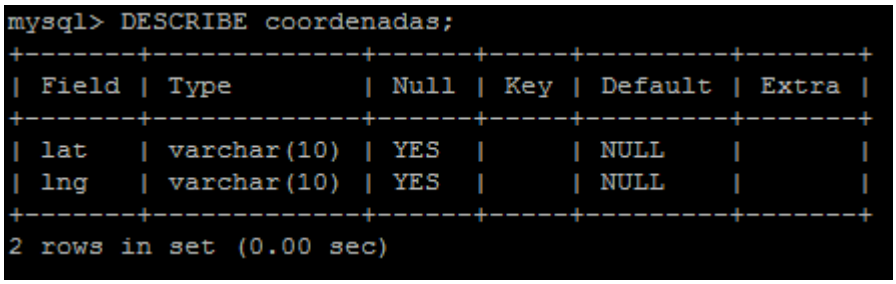
Se muestra la tabla llamada “coordenadas” y el tamaño que tiene cada cadena, lat y lng, para latitud y longitud, respectivamente.

La tabla creada se puede visualizar al ingresar el siguiente comando:

```
mysql> DESCRIBE coordenadas;
```

Se muestra la tabla en la siguiente figura:

Figura 23. **Descripción de tabla en la base de datos**



```
mysql> DESCRIBE coordenadas;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| lat   | varchar(10) | YES  |     | NULL    |       |
| lng   | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Fuente: Obtenida de la terminal de Raspbian.

4.2.3. Programa en Python para la recepción de datos

Un programa en Python permite recibir los datos del microcontrolador ATmega328 y, al mismo tiempo, se encarga de enviar dichos datos a la base de datos configurada en el servidor *web*.

Se utiliza la librería serial de Python para comunicarse con el puerto, además de "*sys*" que permite escribir en consola, al igual que *print.* También es necesario MySQLdb, puesto que se debe enviar la información recibida a la base de datos.

```
import serial
import sys
import MySQLdb
```

Se abre el puerto serial y se configura el puerto en el cual está conectado el ATmega328. En este caso, queda de esta forma:

```
ATmega = serial.Serial('/dev/ttyACM0',115200)
ATmega.readline()
```

La conexión con la base de datos:

```
db = MySQLdb.connect("localhost","root", "sistemagps", "gps")
cursor = db.cursor()
latitud = []
longitud = []
```

La separación del *string*:

```
while 1:
    for i in range (0,100):
        datoString = ATmega.readline()
        #sys.stdout.write(datoString)
        datos = str(datoString).split(";")
        latitud.append (datos[0])
        longitud.append (datos[1][:-2])
        #print (latitud[i]+ " "+ longitud[i])
```

Insertar los valores a la tabla:

```
cursor.execute("INSERT INTO coordenadas(lat,lng) VALUES (%s,%s)"%(latitud[i],longitud[i]))
        db.commit()
```

4.3. Etapa II: interfaz de usuario final

Se desarrolla la creación de un mapa a través de código html, el cual puede correr desde una página *web* o bien enlazado desde una aplicación de Android, construida en APP Inventor.

Se realizan pruebas de campo que marquen los puntos que forman una trayectoria en la geolocalización. Se muestra el dispositivo final en su sistema embebido, listo para ser instalado en un vehículo.

4.3.1. Enlace de API de *Google Maps* con la base de datos

Para tener acceso a las APIs de *Google Maps*, es necesario obtener una llave de desarrollador o key. Esta se obtiene a través de la plataforma de *Google Developers* y es gratuita para las aplicaciones básicas.

La plataforma muestra distintas opciones de mapa, según los requerimientos del entorno en el que se vaya a utilizar. En este sistema se utiliza un mapa sencillo para un entorno *web*.

Una vez obtenida la llave, será posible utilizar cualquiera de los ejemplos de código en JavaScript, con los que cuenta la plataforma.

4.3.1.1. Programa en PHP

Se precisa de un programa elaborado en lenguaje PHP, el cual conecta con la base de datos para extraer los parámetros de latitud y longitud. Dentro del mismo, se escribe una rutina en JavaScript, la cual no es más que el código de la API de *Google Maps*.

La primera parte consiste en declarar la base de datos sobre la cual se va a trabajar:

```
$server = "localhost";
  $user = "root";
  $pass = "sistemagps";
   $dbc = "gps";
   $con = mysqli_connect($server,$user,$pass,$dbc);

if (!$con) {
    die("Connection failed: " . mysqli_connect_error());
```

```
}
```

El usuario definido por defecto en la base es *localhost*, y el nombre de la misma es gps; la contraseña se fijó como sistemagps. El comando $con = mysqli_conect establece la comunicación con la base. Se define un *if,* en caso de que la conexión sea fallida.

Posteriormente, se manda a llamar a la tabla definida previamente como tabla. Se utiliza la función en MySQL, “SELECT * FROM tabla” mediante el comando “echo” se obtienen los valores de latitud y longitud. Se crea un ciclo *while* para que realice esta acción de forma repetitiva.

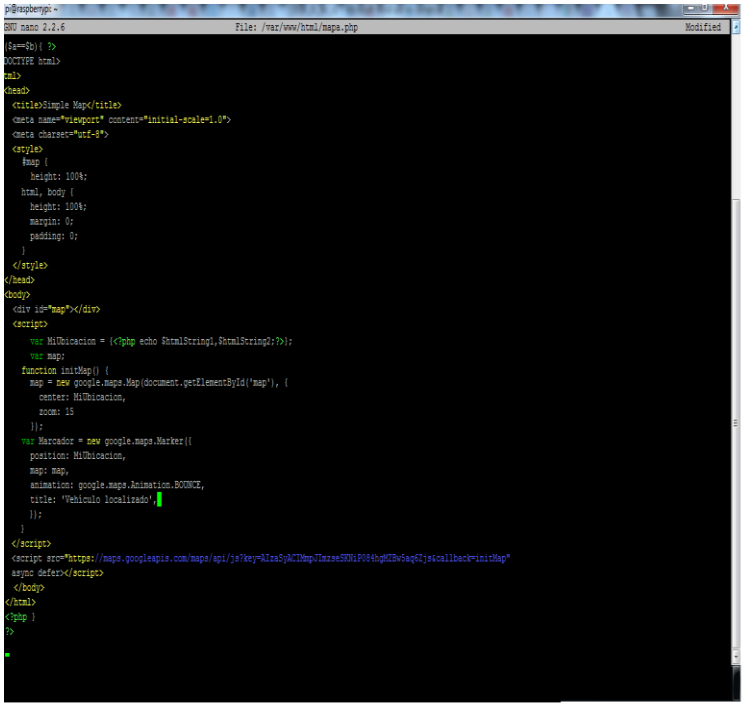
```
$sql = "SELECT * FROM tabla ";

$result = mysqli_query($con, $sql);

$result_count = mysqli_num_rows($result);

if ($result_count > 0) {
  // output data of each row
 while($row = mysqli_fetch_assoc($result)) {

  echo "<tr>";
//        echo "<td>". $row["id"]. '</td>';

          //echo "<td>".$row["lat"]. '</td>' ;

          //echo "<td>".$row["lng"]. '</td>' ;
$htmlString1= "lat: ".$row["lat"];
$htmlString2= ", lng: ".$row["lng"];
 //echo "$htmlString1";
//echo "<br>";

}
}
else {
echo "0 results";
```

}
$con->close();

Los valores obtenidos de latitud y longitud se guardan en *strings,* ya que se deben leer por la API; esta última se encuentra en código de JavaScript, por lo que es la manera más sencilla de hacerlo. Una vez obtenido lo requerido, se cierra la consulta mediante "*$con->close".*

Para insertar código en JavaScript dentro de un bloque en PHP, es necesario definir el inicio y final del mismo. Con el comando "if($a==$b){ ?>" se define que en este punto iniciará el *script* de JavaScript, mientras que con "<?php }" se indica que finaliza.

Figura 24. **Enlace de la API con PHP**



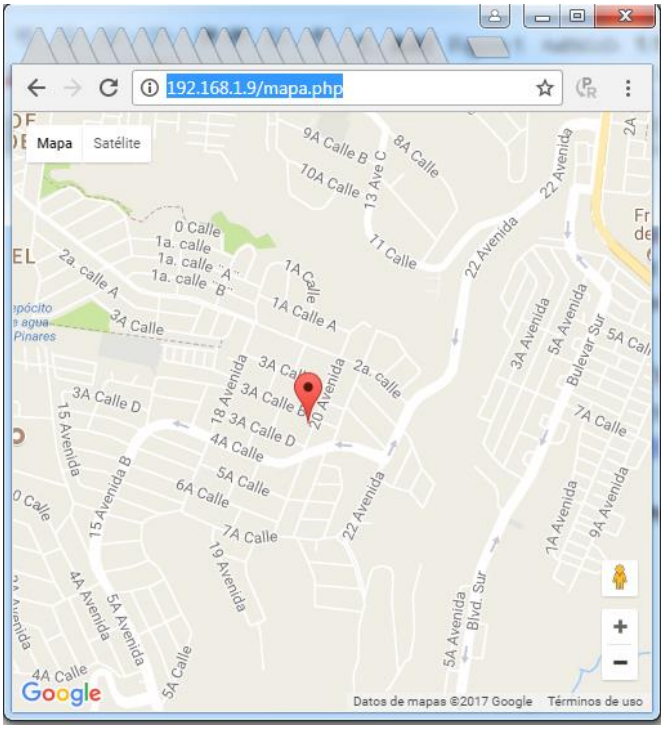
Fuente: Generada en la terminal de Raspbian.

4.3.1.2. Generación del mapa

Una vez incrustada la API de *Google Maps* en el código de PHP, el archivo está listo para ser enrutado desde Internet. En este caso se utiliza una red local, puesto que el servidor está montado sobre una Raspberry Pi.

El archivo generado se nombró como mapa.php. Para enrutarlo, simplemente se escribe la dirección IP de la Raspi y el nombre del archivo PHP, en el navegador. Para este caso sería http://192.168.1.9/mapa.php.

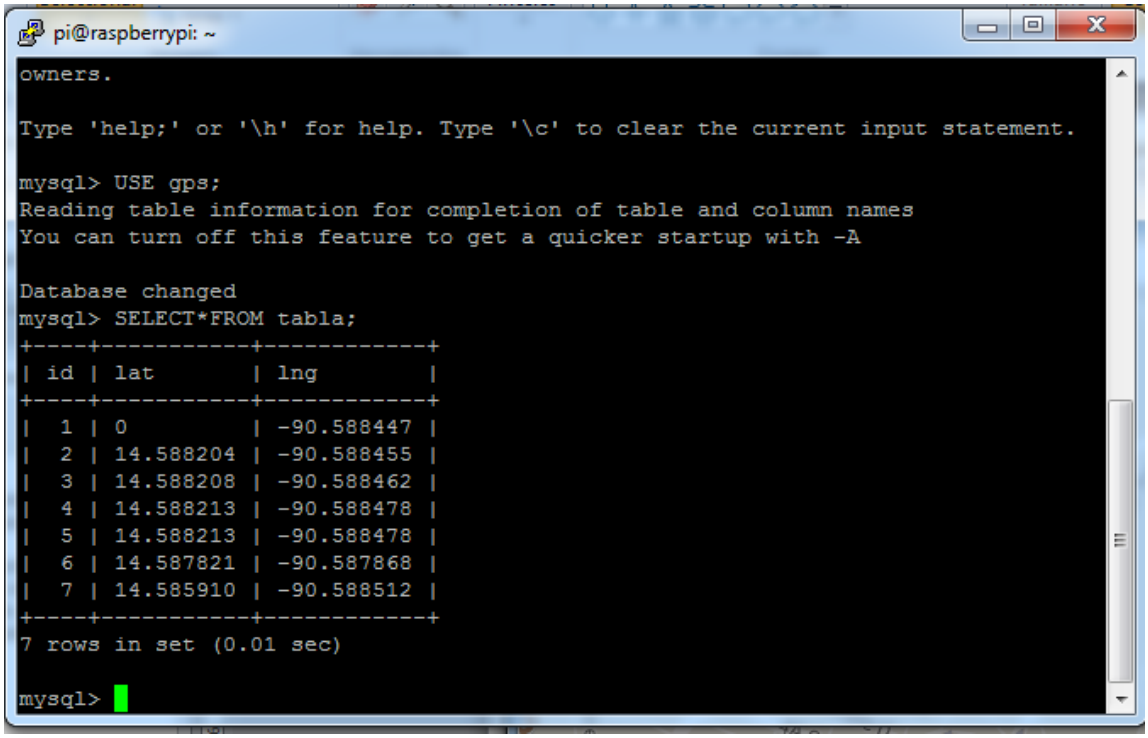
Figura 25. **Mapa en PHP**



Fuente: Tomada del navegador Google Chrome.

La figura 25 muestra el ícono de rastreo en las coordenadas: "*lat*= 14,585910, *lng*= -90,588512". Los datos fueron obtenidos de la tabla en la base de datos establecida. La ubicación muestra el último punto recibido por el receptor GPS.

Figura 26. **Valores en tabla**



Fuente: Generada en la terminal de Raspbian.

En la figura anterior se muestran los últimos puntos guardados en la base de datos, obtenidos por el GPS6MV2.

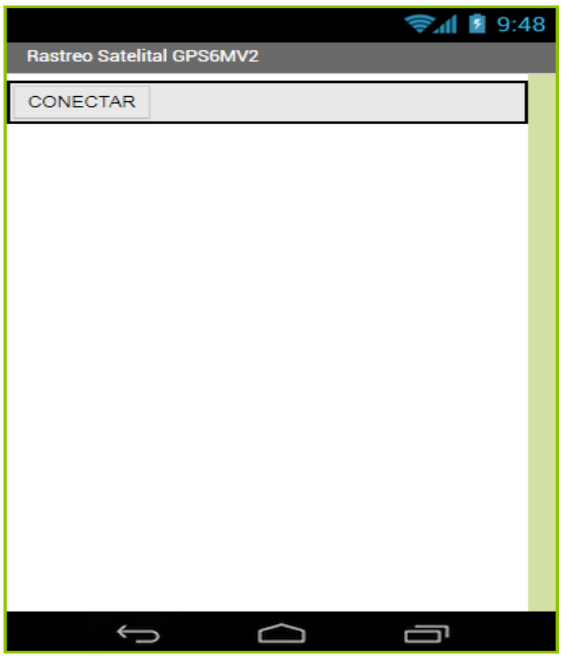
4.3.2. Desarrollo de aplicación Android en App Inventor

Se utiliza el entorno de desarrollo de APP Inventor para construir la aplicación móvil. En este sistema se utiliza el APP Inventor 2 *web* y su emulador.

4.3.2.1. Configuración de componentes

En el *Designer* del APP Inventor se realiza la configuración de los componentes y la conexión mediante un botón de texto mediante la propiedad ActivityStarter.

Figura 27. **Screen de inicio**

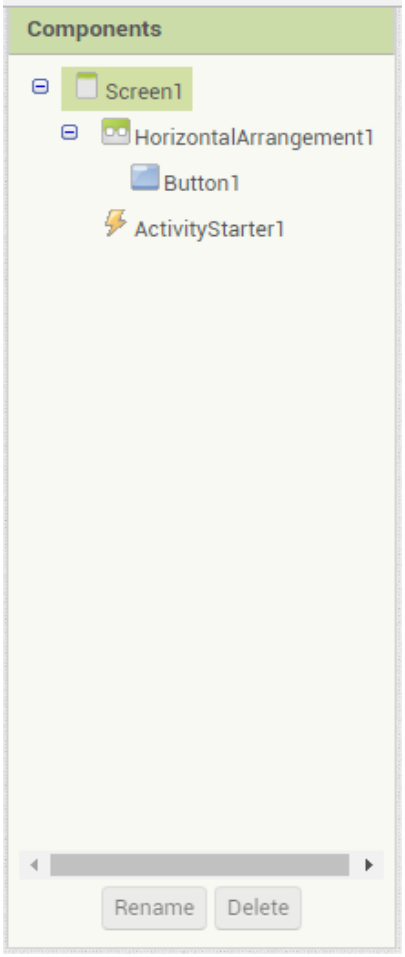


Fuente: Tomada del *Designer* de App Inventor 2.

La figura 24 muestra el *screen* de inicio de la aplicación. Al momento de pulsar en el botón de conectar, se enlaza con la dirección de la página PHP cargada en el servidor. Se obtiene el mapa generado y se puede visualizar el dispositivo rastreado.

Los componentes de la aplicación se describen en un árbol en la sección de “componentes”,

Figura 28. **Componentes de la aplicación**



Fuente: Tomada del *Designer* de App Inventor 2.

Mediante el *HorizontalArrangement1* se dispone el botón de forma horizontal dentro del *screen.* El botón 1 conecta con el servidor *web* y el ActivityStarter permite que el botón 1 realice su acción.

4.3.2.2. Programación de los bloques

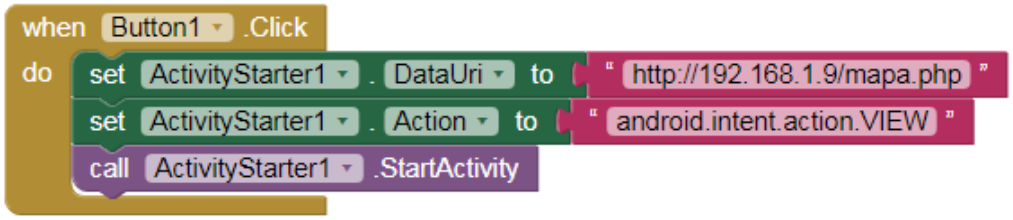
Utilizando las propiedades de los componentes, se realiza la programación mediante la interfaz de *Blocks* del App Inventor. La estructura está definida por el bloque de enlace “While Do”, del *Button1*.

Cada vez que el botón es presionado, se manda a llamar al *ActivityStarter1*, el cual mediante un *set* define la conexión a la dirección que se le ha ingresado mediante un enlace de texto. Para este caso la dirección es http://192.168.1.9/mapa.php.

Una vez atendida la conexión en la dirección IP, se pone en marcha el comando *android.intent.action.VIEW*, el cual permitirá que el sitio enrutado sea desplegado.

Posteriormente, se pone en marcha todo el proceso mediante *StartActivity*, y se cierra el bucle. Cada vez que se realice esta acción se repetirá el mismo proceso. Si el mapa se encuentra cargado, lo que hará será refrescar la última ubicación disponible.

Figura 29. **Programación en bloques**



Fuente: Tomada del *Blocks* de App Inventor 2.

4.3.2.3. Obtención del mapa con el emulador

Para visualizar que la consulta del mapa al servidor *web* sea correcta, se utiliza el *Android Emulator*, el cual se debe descargar a la computadora para enlazar con la aplicación desarrollada.

Se establece comunicación con el App Inventor y se utiliza desde el emulador, como si fuese un teléfono móvil.

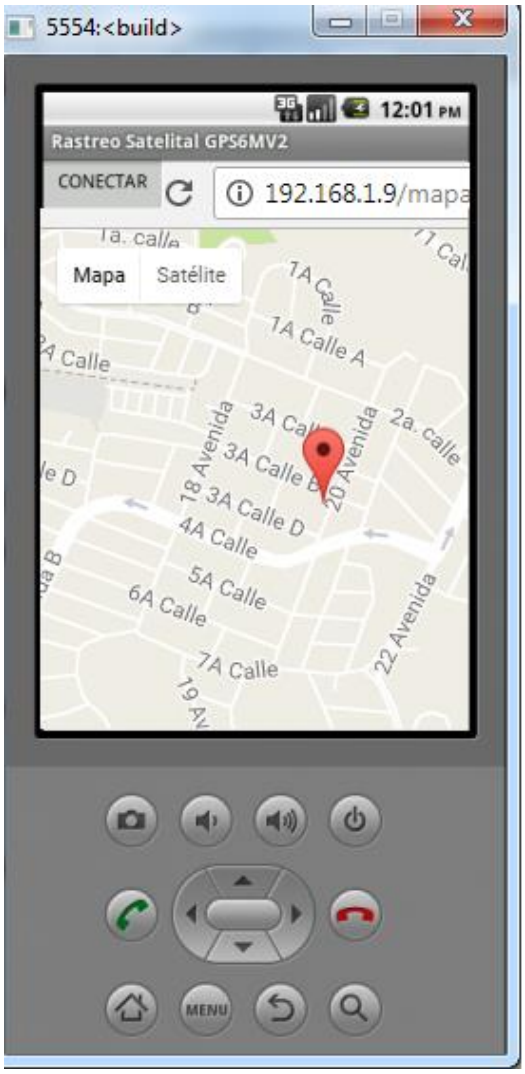
Figura 30. **Android Emulator**

Fuente: Tomada de Android Emulator.

Se realiza la conexión y devuelve el mapa generado con la aplicación de Android.

En la siguiente figura se aprecia el funcionamiento de la aplicación en la que ya es posible realizar el rastreo en tiempo real. Por las propiedades configuradas en la API, es posible arrastrar el mapa si la aplicación es cargada hacia un teléfono móvil.

Figura 31. **Funcionamiento de la aplicación de Android**



Fuente: Tomada de Android Emulator.

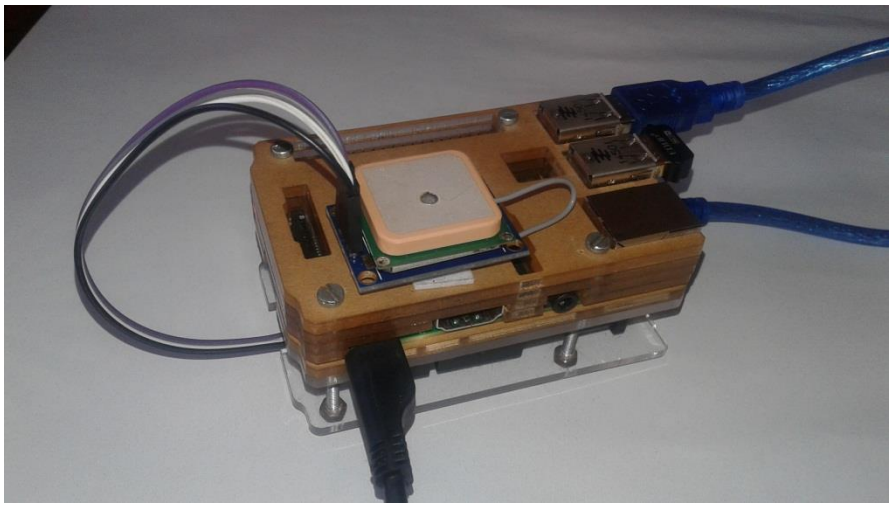
4.3.3. Pruebas de campo

Se realizaron pruebas finales con el dispositivo embebido. Se llevó a cabo un monitoreo en distintos puntos para observar el cambio de visualización del ícono del dispositivo rastreado.

4.3.3.1. Dispositivo final

El sistema se logró integrar en un dispositivo compacto, con el fin de que pueda implementarse en un vehículo terrestre. Naturalmente, al ser un dispositivo de seguridad, debe ser instalado a la parte electrónica del vehículo, recubierta por el tablero. Además se debe instalar con un regulador que suministre 5VDC para su alimentación.

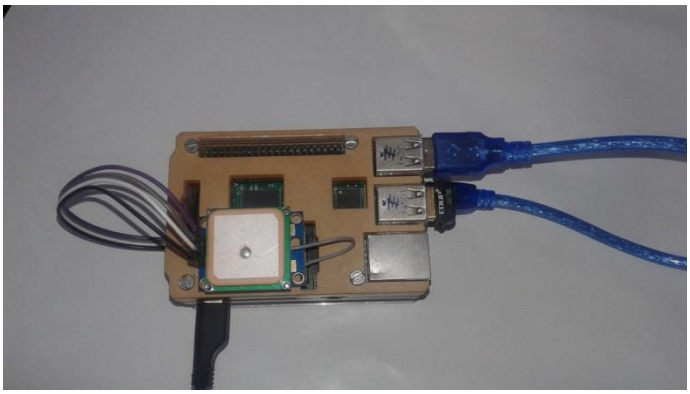
Figura 32. **Dispositivo embebido**



Fuente: elaboración propia.

El dispositivo está ensamblado en una carcasa de acrílico transparente, ya que de esta manera permite una fácil visualización de las conexiones entre los componentes, lo que da además un buen aspecto estético. La conexión entre la Rasberry Pi y el ATmega328 es a través de un cable USB y entre el GPS6MV2 y el microcontrolador, con cables *Dupont* hembra-macho.

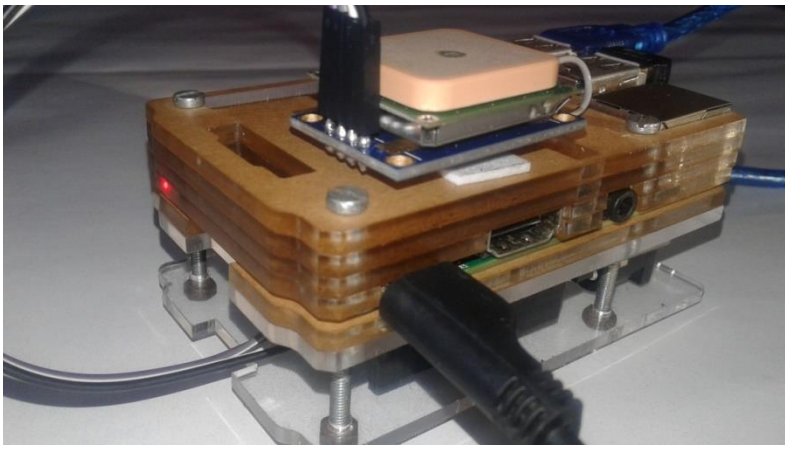
Figura 33. **Conexión de componentes**



Fuente: elaboración propia.

Para su utilización, el dispositivo cuenta con un módulo wi-fi y una conexión a internet inalámbrico, a través de un *acces-point* fijo, vía *wireless*. La conectividad inalámbrica es una parte importante en el diseño, ya que le da la posibilidad de utilizarse como sistema móvil.

Figura 34. **Funcionamiento del dispositivo**



Fuente: elaboración propia.

4.3.3.2. Monitoreo desde la aplicación Android

Se realizaron pruebas en tiempo real y se monitoreó desde la aplicación Android del emulador. Para lo anterior, se realizó un pequeño recorrido y se realizaron cuatro mediciones en un intervalo de tiempo constante entre ellas. Como resultado se tiene una trayectoria marcada de los puntos en mención.

Los datos fueron almacenados en la base de datos del servidor. En la siguiente figura se muestran los marcadores de los rastreos obtenidos en la prueba.

Figura 35. **Prueba de rastreo**

Fuente: Obtenida del Android Emulator.

Las pruebas resultaron ser bastante exitosas. El sistema realizó un rastreo bastante preciso, lo que conlleva a una geolocalización muy confiable.

El dispositivo tiene un tiempo de retardo al establecer comunicación por primera vez con los satélites; debido a esto, al momento del primer encendido toma unos cuantos minutos en empezar a procesar datos. Este tiempo es significativamente menor en exteriores que en interiores.

CONCLUSIONES

1. Se diseñó un sistema embebido en vehículos terrestres, enlazado a servicios en la nube, con un módulo GPS6MV2.

2. Se presentó un sistema de posicionamiento global con una aplicación Android, para la detección satelital.

3. Se desarrolló una aplicación *web* para el rastreo satelital de vehículos terrestres.

4. Se realizó una descripción de los dispositivos utilizados en el desarrollo del sistema de detección satelital.

5. Se propuso el diseño de un sistema de rastreo satelital de bajo costo y alta eficiencia.

RECOMENDACIONES

1. Considerar que en sistemas y aplicaciones móviles, es importante el uso de dispositivos de bajo consumo energético.

2. Para mejorar la precisión del sistema, tomar en cuenta la utilización de receptores GPS con una incerteza no mayor a unos cuantos metros.

3. Actualizar constantemente los paquetes del servidor *web*, para evitar que el sistema se vuelva obsoleto.

4. Para evitar un mal funcionamiento o daño del sistema, es importante que la alimentación de la Raspberry Pi suministre al menos 2 amperios.

5. Si se desea adentrar en un entorno *web* más exigente, se pueden utilizar otras versiones de *APIs*, disponibles en la página de desarrolladores de Google.

BIBLIOGRAFÍA

1. *API*. [en línea]. <https://hipertextualcom/archivo/2014/05/que-es-api/>. [Consulta: 20 de noviembre de 2016].

2. *Base de datos en MySQL*. [en línea]. <https://geekytheory.com/java-php-mysql-ii-creacion-de-una-base-de-datos-en-mysql/>. [Consulta: 20 de marzo de 2017].

3. *Características de U-blox*. [en línea]. <https://en.wikipedia.org/wiki/U-blox>. [Consulta: 22 de noviembre de 2016].

4. *Comparativa raster y vectorial*. [en línea]. <http://www.um.es/geograf/sigmur/temariohtml/node25_mn.html>. [Consulta: 2 de diciembre de 2016].

5. *Configuración módulo Wi-Fi*. [en línea]. <https://geekytheory.com/tutorial-raspberry-pi-configurar-wif/>. [Consulta: 24 de marzo de 2017].

6. *Coordenadas geográficas*. [en línea]. <https://www.xatakaciencia.com/sabias-que/las-coordenadas->. geograficas [Consulta: 24 de marzo de 2017].

7. *Definición de API*. [en línea]. <http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/>. [Consulta: 20 de noviembre de 2016].

8. *Elementos de un rastreador GPS*. [en línea] <http://lagc.uca.es/web_lagc/docs/curso_rap/Presentacion_II.pdf>. [Consulta: 17 de febrero de 2017].

9. *Geodesia*. [en línea]. <https://www.ign.es/layoutln/actividadesGeodesiaStmagd.do>. [Consulta: 2 de diciembre de 2016].

10. *Ingresar información a una base de datos*. [en línea]. <https://geekytheory.com / internet-de-las-cosas-parte-2-subir-los-datos-a-una-ba se-de-datos>. [Consulta: 20 de marzo de 2017].

11. *Instalación de Apache en la Raspberry Pi*. [en línea]. <https://www.geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache -mysql-php/>. [Consulta: 20 de marzo de 2017]

12. *Método de la triangulación*. [en línea]. <http://www.gutovnik.com/como_func_sist_gps.htm>. [Consulta: 17 de enero de 2017].

13. *Protocolos de un servidor web*. [en línea]. <https://www.ecured.cu/Servidor_web>. [Consulta: 4 de febrero de 2017].

14. *Servidor web con Raspberry Pi*. [en línea]. <https://diymakers.es/raspberry-pi-como-servidor-web/>. [Consulta: 20 de marzo de 2017]

15. *Sistemas de Información Geográfica*. [en línea]. <http://sig.cea.es/SIG>. [Consulta: 8 de diciembre de 2016].

16. *Sistemas de referencia geodésicos*. [en línea]. <http://www.ideandalucia.es/portal/iderap-portlet/content / 300e9cf2-5fa1-471a-9885-26f36f6 8b9b7>. [Consulta: 6 de enero de 2017].

17. *Sistemas embebidos GPS*. [en línea]. <http://www.profetolocka.com.ar/2014/09/03/los-gps-en-sistemas-embebidos/>. [Consulta: 3 de febrero de 2017]

18. *Sistemas embebidos utilizados en el rastreo satelital de vehículos terrestres*. [en línea]. <http://emb.cl/gerencia/articulo.mvc?xid=2746>. [Consulta: 28 de noviembre de 2016].

19. *Uso de la telefonía celular en GPS*. [en línea]. <http://www.informatica-hoy.com.ar /soluciones-moviles/El-GPS-en-los-telefonos-celulares.p hp>. [Consulta: 19 de enero de 2017].

APÉNDICES

Apéndice 1. **Programa del ATmega238**

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>

TinyGPS gps;
SoftwareSerial ss(4, 3);

void setup()
{
  Serial.begin(115200);
  ss.begin(9600);

}

void loop()
{
  bool nuevoDato = false;

  for (unsigned long inicio = millis(); millis() - inicio < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      if (gps.encode(c))
        nuevoDato = true;
    }
  }

  if (nuevoDato)
  {
    float flat, flon;
    gps.f_get_position(&flat, &flon);
```

Continuación del apéndice 1.

```
            Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 :
      flat, 6);
            Serial.print(";");
            Serial.println(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 :
      flon, 6);
            delay (1000);
          }


        }
```

Fuente: elaboración propia.

Apéndice 2. Programa en Python para comunicación ATmega328 – Raspberry Pi

```
<?php
$server = "localhost";
  $user = "root";
  $pass = "sistemagps";
   $dbc = "gps";
   $con = mysqli_connect($server,$user,$pass,$dbc);

if (!$con) {
    die("Connection failed: " . mysqli_connect_error());

}

$sql = "SELECT * FROM tabla ";

$result = mysqli_query($con, $sql);
```

Continuación del apéndice 2.

```
$result_count = mysqli_num_rows($result);

if ($result_count > 0) {
  // output data of each row
 while($row = mysqli_fetch_assoc($result)) {

  echo "<tr>";
        echo "<td>". $row["id"]. '</td>';

$htmlString1= "lat: ".$row["lat"];
$htmlString1= ", lng: ".$row["lng"];

}
}
else {
echo "0 results";

}

$con->close();
if($a==$b){ ?>
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
```

Continuación del apéndice 2.

```
    <style>

      #map {
        height: 100%;
      }
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
            var MiUbicacion = {<php echo $htmlString1, $htmlString2;?>};
          var map;
      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: MiUbicacion,
          zoom: 15
        });
       var Marcador = new google.maps.Marker({
          position: MiUbicacion,
          map: map,
          animation: google.maps.Animation.BOUNCE,
      }
    </script>
```

Continuación del apéndice 2.

```
    <script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=
initMap"
    async defer></script>
  </body>
</html>
<?php}
?>
```

Fuente: elaboración propia.

Apéndice 3. **Programa PHP para el mapa**

```
<?php
$server = "localhost";
  $user = "root";
  $pass = "sistemagps";
   $dbc = "gps";
   $con = mysqli_connect($server,$user,$pass,$dbc);

if (!$con) {
    die("Connection failed: " . mysqli_connect_error());

}

$sql = "SELECT * FROM tabla ";

$result = mysqli_query($con, $sql);

$result_count = mysqli_num_rows($result);

if ($result_count > 0) {
  // output data of each row
 while($row = mysqli_fetch_assoc($result)) {
```

Continuación del apéndice 3.

```
  echo "<tr>";
        echo "<td>". $row["id"]. '</td>';

$htmlString1= "lat: ".$row["lat"];
$htmlString1= ", lng: ".$row["lng"];

}
}
else {
echo "0 results";

}

$con->close();
if($a==$b){ ?>
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>

      #map {
        height: 100%;
      }
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
    var MiUbicacion = {<php echo $htmlString1, $htmlString2;?>};
        var map;
      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: MiUbicacion,
          zoom: 15
```

Continuación del apéndice 3.

```
            });
            var Marcador = new google.maps.Marker({
          position: MiUbicacion,
          map: map,
          animation: google.maps.Animation.BOUNCE,
             }
           </script>
           <script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=
initMap"
           async defer></script>
          </body>
        </html>
        <?php}
        ?>
```

Fuente: elaboración propia.

ANEXOS

Anexo 1. **Especificaciones de Raspberry Pi**

Raspberry Pi

Raspberry Pi 2, Model B

Product Name	**Raspberry Pi 2, Model B**
Product Description	The Raspberry Pi 2 delivers 6 times the processing capacity of previous models. This second generation Raspberry Pi has an upgraded Broadcom BCM2836 processor, which is a powerful ARM Cortex-A7 based quad-core processor that runs at 900MHz. The board also features an increase in memory capacity to 1Gbyte.
Specifications	
Chip	Broadcom BCM2836 SoC
Core architecture	Quad-core ARM Cortex-A7
CPU	900 MHz
GPU	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V, 2A
Connectors:	
Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4)
Audio Output	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
JTAG	Not populated
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Micro SDIO

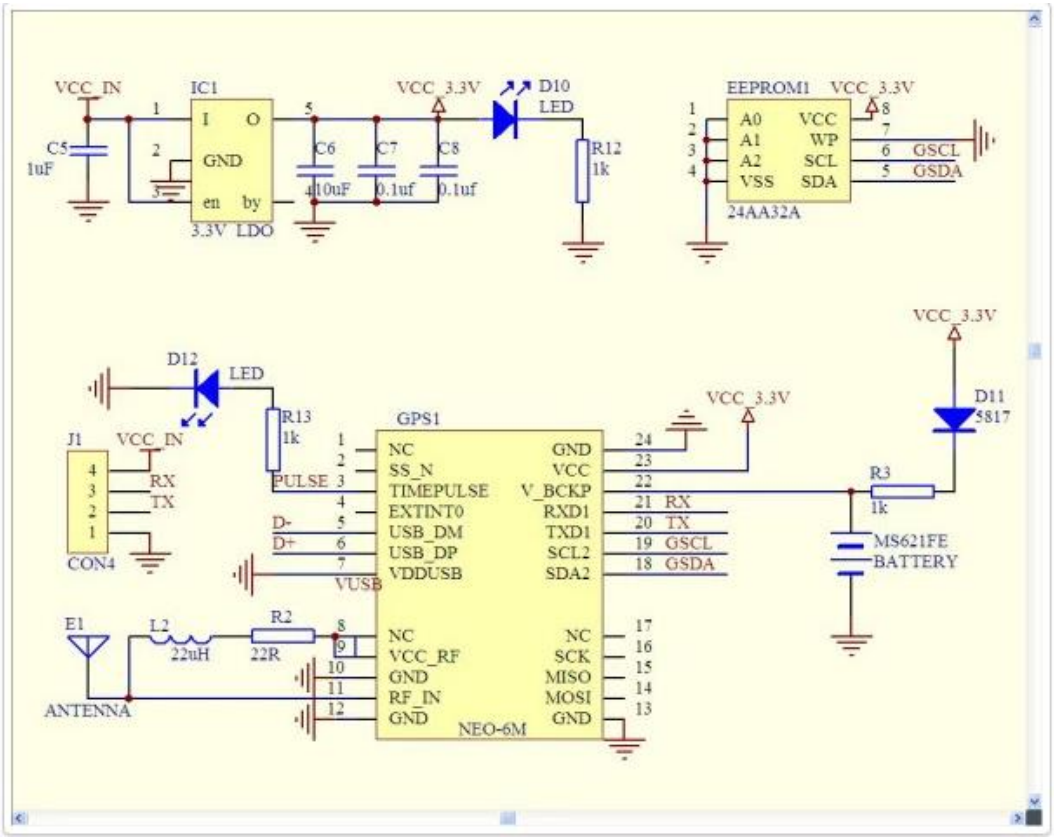
Fuente: Makers. *Raspberry PI como servidor web. http://diymakers.es/raspberry-pi-como-servidor-web/.* Consulta: 26 de marzo de 2016.

Anexo 2. **Especificaciones de ATmega328**

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Fuente: Makers. *Raspberry PI como servidor web. http://diymakers.es/raspberry-pi-como-servidor-web/.* Consulta: 26 de marzo de 2016.

Anexo 3. **Especificaciones del GPS6MV2**



Interface: RS232 TTL
Power: 3-5v
Baudrate default:9600bps

Power supply :3V-5V

Models: GY – GPS6MV2

The LED signal lights

The default baud rate: 9600

Module size 25 mm x35 mm/0.98"x1.37"(inch) (approx)

The antenna size: 25x 25 mm/0.98"x0.98"

Fuente: Makers. *Raspberry PI como servidor web. http://diymakers.es/raspberry-pi-como-servidor-web/.* Consulta: 26 de marzo de 2016.