ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA GEOLOCALIZACIÓN DE BICICLETAS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

Roger Steven Reyes Medina
roger.reyes@epn.edu.ec

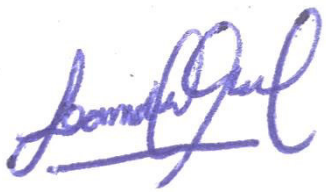
DIRECTOR: ING. LEANDRO ANTONIO PAZMIÑO ORTIZ, MSC.
leandro.pazmino@epn.edu.ec

CODIRECTOR: ING. MONICA DE LOURDES VINUEZA RHOR, MSC.
monica.vinueza@epn.edu.ec

Quito, junio 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por el Sr Reyes Medina Roger Steven como requerimiento parcial a la obtención del título de TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES, bajo nuestra supervisión:



MSc Leandro Pazmiño	**MSc. Monica Vinueza**
DIRECTOR DEL PROYECTO	CODIRECTORA DEL PROYECTO

DECLARACIÓN

Yo Reyes Medina Roger Steven con CI: 1725408718 declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, soy titular de la obra en mención y otorgo una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entrego toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Roger Steven Reyes Medina
estudiante

AGRADECIMIENTO

A mi querida madre Sra. Elizabeth Medina, su apoyo incondicional me dio los ánimos para seguir adelante, gracias por enseñarme a nunca rendirme, a perdonar, a dar siempre lo mejor, a ser agradecido y sobre todo a luchar por mis sueños, gracias por ser un ejemplo de perseverancia, valentía, bondad y humildad. Gracias por todos tus sabios consejos que han sido las palabras que siempre he necesitado para no desistir y seguir adelante. Gracias a ti culminare mi carrera.

A mi querido padre Sr. Rodolfo Reyes, su apoyo y comprensión me brindo las fuerzas para culminar mis metas, gracias por ser mi guía y un padre que formo mi carácter para ser mejor persona, un hombre de bien y un buen profesional.

A mi primo Ing. Cristian Tintin, MSc, a quien considero un hermano gracias a su guía, consejos y apoyo me ha enseñado a ser una mejor persona y un gran profesional, gracias por apoyarme desde mis inicios como universitario y por ayudarme a culminar mi trabajo de titulación, sin tu guía, tolerancia y conocimientos, este proyecto no habría sido posible.

A mis queridas sobrinas Antonela, Lesly y Nila, su alegría, cariño, confianza y honestidad me motivan a convertirme en un mentor, un apoyo incondicional y en un excelente profesional para ustedes.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este informe se explican los factores más relevantes utilizados para el desarrollo del prototipo de geolocalización basado en una investigación bibliográfica.

En la sección uno, se incluye la introducción, se redacta el objetivo general y los objetivos específicos, también se abarcan los fundamentos teóricos importantes que se presentan conforme se avanza por el informe.

En la sección dos, se muestra la metodología empleada para el desarrollo del prototipo, redactando de manera más general como se logran los objetivos específicos propuestos para el proyecto.

La sección tres, resultados y discusión se explican de manera detallada como se logra cada uno de los objetivos específicos, por medio de diagramas, tablas, imágenes y basándose en referencias bibliográficas, además se describen las pruebas realizadas para lograr con el cumplimiento de los objetivos.

En la sección cuatro, se redactan las conclusiones que se obtuvieron en base a los resultados de las pruebas realizadas en la sección anterior, también, se brindan recomendaciones para mejorar el prototipo o evitar ciertos inconvenientes que pueden presentarse en el desarrollo del mismo.

La sección cinco se incluyen las referencias bibliográficas utilizadas para el desarrollo de este proyecto.

PALABRAS CLAVE: Arduino, sensores, rastreador, GPS.

ABSTRACT

This report explains the most relevant factors used for the development of the geolocation prototype based on a literature review.

In section one, the introduction is shown, the general objective and the specific objectives are established. In addition, it includes the important theoretical foundations that will be presented as the report advances.

In the section two, the methodology used for the development of the prototype is shown. It specifies in a more general way how the specific objectives proposed for the project will be achieved.

The section three, results and discussion, explains in detail how each of the specific objectives is achieved, using diagrams, tables, images and based on bibliographic references. It also describes the tests performed to achieve the objectives.

In the section four, conclusions are drawn from the results of the tests carried out in the previous section. Recommendations are given to improve the prototype or to avoid certain inconveniences that may arise during its development.

The section five shows the bibliographic references used for the development of this project.

KEYWORDS: *Arduino, sensors, tracker, GPS.*

1 INTRODUCCIÓN

El uso de bicicletas en Ecuador para años futuros va en aumento según el Instituto Nacional de Estadísticas y Censos INEC, se apreció que en el año 2015, 1'872.729 personas usaban bicicletas y para el año 2016, 2'481.343 personas las usaban. También, el "Expreso" destaca este hecho asegurando que las ventas de los negocios de bicicletas en Guayaquil han subido a 20%, en los primeros meses del año 2017, las compras de las bicicletas en toneladas subieron a 201,7% con relación a enero y febrero del año 2016, mostrando una tendencia creciente del uso de bicicletas [1] [2].

El aumento del uso de bicicletas es debido, a los beneficios de salud, ambientales y económicos que este conlleva, además, por ciertas políticas que se han implementado en el país como: eliminar los impuestos a la importación de bicicletas de competencia, el fomento al ciclo paseo en provincias como Loja, Quito, Cuenca y Ambato, entre otras [3] [4] [5].

La adquisición de una bicicleta de cualquier gama en Quito significa también un riesgo, debido al posible robo de esta. Los índices de delincuencia en Quito han mostrado una tendencia creciente de 6278 robos en el año 2017 a 6656 robos en el año 2018, esto según el Ministerio del Interior; estos robos también involucran bicicletas de diferentes gamas. Además, una vez robada la bicicleta es muy difícil volver a recuperarla ya que son negociadas en el mercado negro [6] [7] [8].

Según el INEC en una encuesta del 2015 y 2016 el 1.9% de las personas a nivel nacional utiliza una bicicleta como un medio de transporte [1], dicha cifra continúa en ascenso [9]. Mientras las personas realizan sus actividades dejan la bicicleta en un estacionamiento de bicicletas, en Quito existen aproximadamente 163 plazas de estacionamiento gratuitas [10]. Cuando se deja la bicicleta en algún estacionamiento no se garantiza la seguridad total de la mismas, además, este es un hecho crítico considerando que la delincuencia en el país va en aumento.

Este proyecto de titulación propone implementar un sistema de geoposicionamiento para bicicletas, con el uso de esta tecnología se ayudará a un ciclista a monitorear en tiempo real la ubicación de la bicicleta y así evitar preocupaciones como la inseguridad al dejar dicho artículo un algún lugar externo fuera de la vista del dueño.

Los dispositivos que permiten la localización de bicicletas no son comúnmente comercializados en Ecuador, y los que existen en el mercado son solo para establecer

rutas en una carrera, más no para permitir al usuario establecer la localización de la bicicleta.

Con este prototipo se podrá conocer la localización de la bicicleta al momento de ser robada, para reclamarla al encontrarla y así evitar la pérdida de grandes cantidades de dinero que el dueño ha invertido en su bicicleta. Los beneficiarios directos del uso del prototipo son los dueños de las bicicletas debido a que pueden monitorear en tiempo real la ubicación de su bicicleta.

1.1 Objetivo general

Implementar un prototipo para la geolocalización de bicicletas

1.2 Objetivos específicos

Objetivo 1: Identificar los requerimientos necesarios para la implementación del prototipo

Objetivo 2: Analizar el *hardware* adecuado según los requerimientos necesarios para el prototipo

Objetivo 3: Construir el prototipo de geolocalización de bicicletas

Objetivo 4: Desarrollar el *software* para que el usuario determine de manera gráfica la ubicación de la bicicleta.

Objetivo 5: Realizar pruebas de funcionamiento del prototipo.

1.3 Fundamentos

Diferencias en los sistemas de navegación

Los sistemas de navegación existentes son: GPS desarrollado por los Estados Unidos, el sistema GLONASS desarrollado por Rusia y Galileo por la unión europea. Cada sistema tiene características que los diferencian [11].

Galileo es una tecnología que sigue en desarrollo, pero ofrece una mayor precisión que los sistemas GPS y GLONASS, además es diseñado para uso civil y no militar a diferencia de los otros dos sistemas. Galileo al ser una nueva tecnología de navegación el consumo comercial está también en desarrollo existe alrededor de 30 compañías que producen chips del sistema Galileo, sin embargo, obtener uno puede ser complicado, además, está destinada para el uso en toda Europa [12] [13].

El sistema de navegación GLONASS puede tener una precisión similar a GPS, sin embargo, cuenta con menor número de satélites que los que cuenta el sistema GPS, exactamente GLONASS cuenta con 24 satélites, esto puede presentar un inconveniente al momento de establecer conexión con los satélites. Otro factor importante a destacar es, en el hemisferio sur el sistema de posicionamiento ruso no es tan preciso como lo es GPS [14].

Sistema de navegación GPS

El Sistema de Posicionamiento Global, es un programa desarrollado por el programa NAVSTAR (*Navigation Satellite Timing And Ranging*), y puesto en marcha desde el año 1973, el encargado del mantenimiento y desarrollo es el Departamento de Defensa de los Estados Unidos, por ello su uso es con fines militares. En la actualidad el uso civil del sistema GPS se ha ampliado masivamente, por ello el sistema es un servicio público [15].

El sistema GPS se empleó para reemplazar otros sistemas de navegación como son Decca, Loran C, Omega, Transit, Tacan, ILS, Radiofaros, etc [15].

GSM

El Sistema Global para Comunicaciones Móviles (GSM), utiliza multiplexación por división de frecuencia, es utilizado en 200 naciones con 600 millones de suscriptores y forma parte del grupo de las redes 2G [16].

Los canales de GSM almacenan pocos usuarios, exactamente 8, la tasa de cada canal es de 270.33 bps. Esta tasa al ser ocupada por 8 usuarios se debe dividir para el número de usuarios dando un total de 33.854 kbps por usuario [17].

GPRS

GPRS o Servicio de Radio de Paquetes Generales, permite el envió de paquetes IP a través de estaciones móviles, por medio de las celdas de sistemas de voz, en sí, es una red sobrepuesta a la red GSM. La tecnología GPRS forma parte de la red 2.5G.

Para enviar paquetes, se hace una petición a la estación base de una o más ranuras de tiempo, si la solicitud llega a la estación, esta asignará una ranura de tiempo y la frecuencia a la que se podrá enviar los paquetes. Cuando los datos llegan a la base estos son enviados a Internet por una red cableada [17].

WAN

WAN viene de *Wide Area Network*, significa redes de área amplia. La cobertura que puede llegar a alcanzar va desde 100 hasta 1 000 (km) [18], es decir, cubre distancias desde países o continentes.

Las WAN están formados por dos principales componentes que son elementos de conmutación y líneas de transmisión. Las líneas de transmisión pueden ser cable de cobre, fibra óptica, radio enlaces, etc. Los elementos de conmutación son computadoras encargadas de conectar dos o más líneas de transmisión.

Interfaz UART

UART es acrónimo de *Universal Asynchronous Receiver-Transmitter* y es una interfaz que permite la comunicación entre dispositivos. Comúnmente es para las comunicaciones de sistemas, utilizado por estándares como RS-323, RS-422 y RS-485. UART junto con el estándar RS-232 son utilizados en los puertos seriales de computadoras, permitiendo así comunicación entre dos computadoras. Sin embargo, no es posible la comunicación con más de dos equipos. Empleando UART es posible la comunicación con cualquier tipo de trama y velocidad en baudios [19] [20].

Librería TinyGPS++

Existen librerías en Arduino que permiten ahorrar el trabajo de separar la trama del formato NMEA 0183 en los valores que se desean, una de estas librerías es TinyGPS++.

Esta librería está diseñada para proporcionar funcionalidades de NMEA GPS, extrae los valores de posición, tiempo, fecha, número de satélites, entre otros. Estos son extraídos de los datos $GPGGA y $GPRMC.

NMEA

NMEA (*National Marine Electronics Association) e*s utilizado en los receptores del sistema de navegación GPS. Este estándar permite establecer varios requerimientos como el protocolo de transmisión de datos, la hora y el formato para la transmisión de datos [21].

AJAX

AJAX (*Asynchronous JavaScript and XML)* es un conjunto de tecnologías como JavaScrip, XML, XSLT, *Document Object Model* y *XMLHttpRequest*. Principalmente utilizaba el formato XML, pero luego se empezó a utilizar el formato JSON. La función de AJAX es realizar peticiones al servidor sin la necesidad de renderizar nuevamente la página Web [22] [23].

JSON

JSON (*JavaScript Object Notation*) es un formato de intercambio de datos del tipo palabra clave, JSON es un formato fácil de escribir y leer, tanto para humanos como para máquinas. Además, tiene características similares a la familia del lenguaje de programación como C, C++, Java, JavaScript y Perl.

JSON puede ser comparado con XML, incluso se ha investigado que JSON ha sido capaz de tomar el lugar de XML como formato de intercambio de datos en los servicios web [24].

PHP

PHP (*Hypertext Preprocessor*) es un lenguaje de programación de lado del servidor, el código puede ser escrito directamente en el código de HTML, y ser ejecutado en el lado del servidor por medio de un intérprete. Es un lenguaje *Open Source*, es decir, tanto el código como el intérprete pueden ser accesibles a todo el público [25].

JavaScript

Es un lenguaje interpretado se destaca de otros lenguajes por sus motores de interpretación esto incrementa la rapidez del procesamiento del código, así logra procesamientos tan rápidos como las aplicaciones de escritorio. Este lenguaje es el más recomendado para el desarrollo de aplicaciones web [26].

HTML

HTML (*HyperText Markup Language*) es un lenguaje de etiquetas para la construcción de páginas web. Las etiquetas de HTML están conformadas por los signos mayor y menor que, además entre ellos almacena palabras y atributos claves. Cada etiqueta tiene una apertura y un cierre [26].

Servidor *ByetHost*

ByetHost cuenta con sus propios servidores, *datacenters*, y utilizan su propio *software* y soluciones. Es un free *Hosting* que se ejecuta en *software* personalizado, alojamiento en una red única o en un grupo de servidores que permite que el sitio web se ejecute a través de cientos de nodos de servidores al mismo tiempo. Todo el *hardware* y *software* han sido creado especialmente para proveer rápida disponibilidad en línea con velocidad, características y confiabilidad. Todos los servicios *hosting* incluyen características como PHP, MySQL, FTP, Email, etc.

Servidor *ThingSpeak*

ThingSpeak se describe en su página oficial como “un servicio de plataforma de análisis de IoT”. Este permite visualizar los datos y analizarlos en vivo a través de la nube [27].

MySQL

Una base de datos en MySQL es implementada como un directorio que contiene archivos que corresponden a tablas en la base de datos. Por medio de MySQL se puede crear, modificar, actualizar, etc., la base de datos, todo por medio de comandos SQL [28].

phpMyAdmin

Es un *software* escrito en PHP, da la posibilidad de manejar la administración de MySQL sobre la web. phpMyAdmin soporta operaciones tanto MySQL como MariaDB [29].

Comandos AT

Los comandos AT son comandos que se utiliza para comunicarse con el módulo A7, estos comandos permiten enviar mensajes de texto sms, realizar o cerrar llamadas telefónicas, obtener datos GPS, conectarse con un servidor externo, entre otras aplicaciones.

En la Tabla 1.1 se muestran los comandos que fueron necesarios para realizar este prototipo.

Tabla 1.1 Comandos AT del módulo A7

Comandos	Sintaxis	Descripción
Llamadas	ATH	Desconecta llamada existente, Cuelga toda llamada conectada, incluyendo activos, esperas y llamadas retenidas.
Detectar llamadas	AT+CLCC	Detecta la llamada, el formato que responde es: CLCC: <id1>, <dir>, <stat>, <mode>, <mpty>, <*number*>, <*type*> **<id1>** Número de identificación de llamadas **<dir>** 0 llamada originada en el móvil 1 llamada con terminación móvil **<stat> (estado de la llamada)** 0 activo 5 espera (llamada MT) 7 lanzamiento (la red libera esta llamada) **<mode>** 0 voz 1 dato 9 desconocido **<mpty>** 0 no es partes de la llamada multipartida 1 la llamada es partes de la llamada multipartida ***<number>*** Número telefónico en formato <*type*> ***<type>***

Continúa →

Comandos	Sintaxis	Descripción
		Tipo de octeto de dirección en formato entero
Servicios de red	AT+CREG?	Utilizado para consultar el registro de la red, devuelve el estado actual de registro. Cuelga toda llamada conectada, incluyendo activos, esperas y llamadas retenidas. El formato que responde es: CREG: <n>, <stat> **<n>** 2: Habilitado la no solicitud del resultado del código y de la información de locación. **<stat>** 1: Registrado, red doméstica 3: Registro denegado 4: Desconocido 5: Registrado, *roaming*
GPRS	AT+CGATT= <state>	Utilizado para adjuntar al MT o para desplegarlo del servicio de dominio. El valor de "*state*" puede ser: **<state>** Indicadores del estado de la unión PS 1: Adjuntado
parámetros PDP	AT+CGDCONT= <Rango de apoyo(cid)>, <Tipo de PDP>, <APN>	Define la configuración del parámetro PDP. PDP es el proceso en Security Gateway responsable de recopilar y compartir identidades. Los parámetros de la sintaxis son: **<cid>** Identificador de contexto de PDP, el rango de valores permitidos valor mínimo 1 y valor máximo 7,

Continúa →

Comandos	Sintaxis	Descripción
		<Tipo de PDP> Especifica el tipo de protocolo de paquete de datos. IP – Protocolo de Internet IPv6 - Protocolo de Internet, versión 6 PPP – Protocolo Punto a punto **<APN>** Nombre del punto de acceso, parámetro *string*, es un nombre lógico que es utilizado para seleccionar el GGSN o la red de paquete de datos externos.
TCP/IP	AT+CIPSTART = <Modelo>, <Nombre de Dominio>, <Puerto>	Inicia la conexión TCP o UDP. **<Modelo>** Parámetro *string* que indica el tipo de conexión TCP → Establecer conexión TCP UDP → Establecer conexión UDP **<Puerto>** Puerto del servidor remoto **<Nombre de Dominio>** Nombre de dominio del servidor remoto
Enviar datos	AT+CIPSEND	Envía datos a través de la conexión TCP o UDP. Responde “>”, posteriormente se envía datos y para finalizar el mensaje se da a CTRL+Z o *char* (26).
Cerrar conexión TCP/UDP	AT+CIPCLOSE	El comando solo cierra el estado de la conexión establecida TCP/UDP

Continúa →

Comandos	Sintaxis	Descripción
Desconectar la conexión inalámbrica	AT+CIPSHUT	Permite desconectar la conexión inalámbrica, excluyendo el estado “IP INITIAL”, se puede aplicar el comando, después de cerrar el estado “IP INITIAL”.

2 METODOLOGÍA

2.1 Descripción de la metodología usada

Objetivo 1

Identificar los requerimientos necesarios para la implementación del prototipo

Para el diseño del prototipo se utilizó una metodología basada en proyectos además de una investigación bibliográfica, este proyecto hace uso del sistema que brinda coordenadas geográficas el cual es GPS.

Además, el dispositivo va integrado en la bicicleta, por lo tanto, los datos de coordenadas tanto de latitud y longitud, se envían de manera inalámbrica, entre estas redes se puede encontrar GPRS, UMTS, GSM, WLAN, Wimax, Bluetooth, entre otros.

Por último, todos los datos que el prototipo almacene se envían al usuario final, es por ello, que se utiliza un interfaz amigable, sencillo de entender y utilizar.

Objetivo 2

Analizar el *hardware* adecuado según los requerimientos necesarios para el prototipo

El *hardware* existente en el mercado es diverso, es por ello que se analizó y determinó el mejor dispositivo en base a una comparación entre servicios, características técnicas, precios, disponibilidad en el país, facilidad de uso, calidad, tamaño, entre las principales características a considerar.

Objetivo 3

Construir el prototipo de geolocalización de bicicletas

La construcción del prototipo se basó en el diseño realizado, utilizando simuladores que permitieron observar el comportamiento del diseño verificando el correcto funcionamiento del diseño electrónico planteado y que permitió editar las veces que

fueron necesarias. Además, se dispuso de la capacidad de crear el circuito impreso (PBC) que se necesitó para crear el circuito en la baquelita. Uno de los programas que posibilitaron estas actividades fue Proteus.

Objetivo 4

Desarrollar el *software* para que el usuario determine de manera gráfica la ubicación de la bicicleta

Primero se trabajó en un sistema operativo el cual permite desarrollar el programa, para este caso el *software* se realizó para teléfonos celulares, únicamente, para el sistema operativo Android. Por consecuencia se seleccionó el tipo de *software* que permitió desarrollar la aplicación móvil en el sistema operativo, además se previó utilizar una interfaz GPS, de uso fácil, con suficientes librerías que facilitó la programación, entre otros.

Objetivo 5

Realizar pruebas de funcionamiento del prototipo

Se realizó pruebas por separado tanto para *hardware* como de *software* para verificar su funcionamiento de manera independiente, posteriormente fueron unificados y se hicieron trabajar de manera conjunta. Cuando se presentaron fallas que no permitieron el cumplimiento de lo deseado se realizaron los cambios respectivos para solucionarlos.

3 RESULTADOS Y DISCUSIÓN

El prototipo de geolocalización para bicicletas permite tener un control de la ubicación del prototipo en la bicicleta; con ayuda de la aplicación móvil, así como de la página web, se muestra en un mapa la ubicación de la bicicleta.

El usuario tiene la capacidad de iniciar o detener el envío de datos de posición del prototipo por medio de la aplicación móvil o de la página web.

3.1 Identificación de los requerimientos para el prototipo

Los requerimientos fueron identificados en base a una investigación de las funciones principales que cuentan los dispositivos de rastreo, así como características técnicas de dispositivos inalámbricos, dentro de las cuales se logran determinar los siguientes requerimientos:

- Capacidad para obtener la ubicación geográfica.

- Transmisión de la información.
- Dispositivos portables.
- Interfaz de comunicación.
- Fuente de alimentación.
- Fuente de procesamiento.
- Memoria de almacenamiento.

Para el desarrollo del prototipo los requerimientos necesarios a emplear para lograr los objetivos planteados, se detallan en esta sección.

Sistema de navegación

Se destacaron los factores principales que limita el uso de los sistemas GALILEO y de GLONASS en el punto Diferencias en los sistemas de navegación de la sección de Fundamentos, todas estas desventajas no las tiene el sistema GPS, es decir, fue lanzado en los años 1978 lleva varios años en desarrollo, cuenta con 32 satélites en seis orbitas y funciona bien en el hemisferio sur, estos factores permite que tenga una mayor precisión que GLONASS, pero menor que GALILEO. Además, GPS cuenta con mucha más variedad de productos en el mercado que GALILEO y GLONASS. Por estos factores se va a implementar GPS en el prototipo de geolocalización [30].

Red inalámbrica

El dispositivo va integrado en la bicicleta, por lo tanto, los datos de coordenadas tanto de latitud y longitud se deben enviar de manera inalámbrica, entre estas redes se puede encontrar GPRS, UMTS, GSM, WLAN, Wimax, Bluetooth, entre otros.

Se descarta el uso de redes PAN y LAN debido a su pequeña cobertura que alcanza desde oficinas a edificios. Las redes inalámbricas MAN tienen una cobertura mucho mayor a PAN y LAN, alcanzando varios metros incluso algunos kilómetros, pero las redes WAN pueden alcanzar cobertura desde 80 kilómetros o más [18].

Por lo tanto, se determinó que se requiere que la red inalámbrica tenga la mayor cobertura para trabajar con el prototipo de geolocalización, por tal razón, se utiliza las redes WAN. Las tecnologías disponibles en las redes WAN son GSM, GPRS, UMTS, LTE, es decir todas las redes celulares inalámbricas existentes [31] [18]. A partir de la generación 2.5G, es posible emplearlas para la transmisión de datos.

Debido a que el prototipo de geolocalización enviará información de latitud y longitud en una cadena de datos, la cantidad de datos que ocupa está en el orden de los kilobytes, por tal razón se puede optar por tecnologías como GPRS, UMTS, etc., para enviar los

datos a la red, pero se seleccionó la tecnología GPRS debido a que es la que cuenta con una mayor cobertura en Ecuador.

Interfaz para usuario

Se ha optado por el desarrollo de una aplicación móvil, debido a que permite monitorear al prototipo de geolocalización, adicionalmente, se ha desarrollado una página web que muestre el posicionamiento de la bicicleta, así el usuario podrá acceder a la página por medio del navegador web del celular o de una computadora. Esto posibilita que el usuario tenga más opciones para el monitoreo del prototipo y así no depender solo de una aplicación.

Servidor *ThinkSpeak*

La razón de utilizar el servidor *ThingSpeak* es debido a que va a actuar como un intermediario para la conexión entre el prototipo de geolocalización y el servidor web, como se observa en la Figura 3.1.

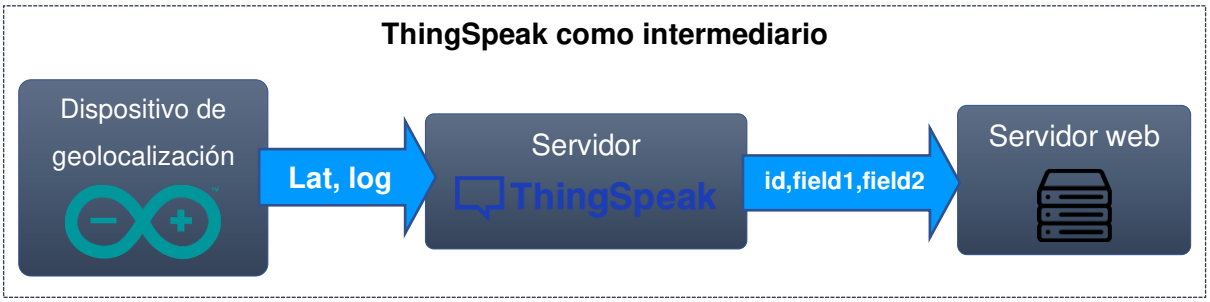


Figura 3.1 *ThingSpeak* como intermediario

El servidor *ThingSpeak*, también es utilizado para receptar y almacenar la información de posicionamiento GPS en una base de datos, dando la posibilidad a otros servidores de acceder a dicha información por medio de una llave API pública.

ThingSpeak habilita llaves API lo cual permite escribir datos a un canal o leer datos desde un canal privado. La llave API es creada una vez se generan los canales.

En la Figura 3.2 se muestran las llaves API que se implementó para escribir o leer datos del canal, por temas de seguridad no se muestran los códigos de las llaves.

La llave para escribir datos en el canal se le asigna al prototipo de geolocalización ya que entrega datos de latitud y longitud al canal, mientras que la llave API para leer datos almacenados en el servidor *ThingSpeak* se le asigna al sitio web en *ByetHost* ya que muestra los datos recogidos por el servidor por medio de un mapa.

Escribir en canal

```
GET https://api.thingspeak.com/update?api_key=2CNOXEGVDQ3X35K7&field1=0
```

Leer en canal

```
GET https://api.thingspeak.com/channels/1073852/feeds.json?results=2
```

Figura 3.2 Api *key* proporcionado por *ThingSpeak*

3.2 Análisis del *hardware* para el prototipo

Módulo A7

Existen varios módulos que implementan tecnologías con redes WAN y con sistemas de navegación, el módulo A7 es uno de ellos, la gran diferencia con respecto a otros, es el precio y el tamaño.

El módulo A7 soporta GSM, GPRS y GPS, es útil para enviar SMS, llamar a un número telefónico, brindar servicios de datos y de GPS. El módulo es controlado por comandos AT vía UART ver Comandos AT en la sección Fundamentos. En la Tabla 3.1 se detallan las características técnicas del módulo.

Tabla 3.1 Características técnicas de módulo A7 [32]

Características	Valores
Fuente de alimentación	5 – 9 V_{DC}.
Corriente	3 A
Sensibilidad	< -105dB
Fuente de alimentación	1.0636
Bandas GSM/GPRS	850, 900, 1800, 1900 MHz.
Tráfico de bajada GPRS 2.5G	85.6 Kbps
Tráfico de subida GPRS 2.5G	42.8 Kbps
GSM red worldwide	2G
2.5G GPRS clase	10
Soporta	Llamadas de voz Envío de SMS, comandos AT y TCP/IP

En la Figura 3.3 se aprecia los componentes del módulo A7.

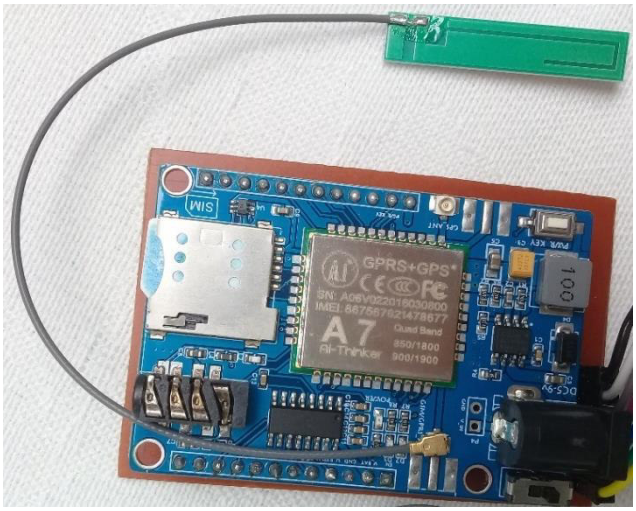


Figura 3.3 Módulo A7 GSM/GPRS/GPS

En la Tabla 3.2 se detallan las funciones de los pines que el módulo A7.

Tabla 3.2 Descripción de pines del módulo A7 [32]

Pines	Función
EN	Pin habilitado para control de potencia del módulo (puede controlar el interruptor de encendido, habilita alto nivel predeterminado)
V_BAT	Pin suministrar potencia con batería de litio de 3.5 – 4.2 V. el módulo tiene dos pines V_BAT.
GND	Pin a tierra, el módulo tiene 5 pines a tierra
U_RXD	Puerto serial para comandos AT, PIN receptor
U_TXD	Puerto serial para comandos AT, PIN transmisor
H_TXD	Pin de actualización del kernel firmware, PIN transmisor
H_RXD	Pin de actualización del kernel firmware, PIN receptor
232_TXD	Nivel de control para conector RS232, PIN transmisor.
232_RXD	Nivel de control para conector RS232, PIN receptor
GPS_TXD	Datos GPS, PIN de transmisión (salidas de datos del GPS)
PWR_KEY	PIN de control de encendido del módulo por *software*, inicio de cortocircuito de V_BAT.
SLEEP	En alto nivel entra en modo de bajo consumo, y para bajo nivel entra en abandono.
U_CTS	Puerto serial para comandos AT, PIN CTS

En la Figura 3.4 se muestra la distribución de los pines del módulo A7.

Figura 3.4 Distribución de pines en módulo A7

Arduino Nano

Existe diversos microcontroladores los más económicos, con mayores características y gran comunidad es Arduino. Para el prototipo se desea que los componentes a utilizar sean del menor tamaño posible. La compañía Arduino cuenta con esta variedad como Arduino Mini, Micro, Nano, entre otros. El microcontrolador utilizado para este prototipo es Arduino Nano, ya que contiene mayor memoria SRAM y *Flash*, por esta razón se ha seleccionado a este microcontrolador.

El Arduino Nano cuenta con 14 pines estos pueden ser utilizados como entrada y salida si se emplea la función *pinMode(), digitalWrite() y digitalRead()*. Cada pin opera a 5 (V) y puede entregar o recibir un máximo de 40 (mA). En la

Tabla **3.3** se describen los pines que tiene el Arduino Nano.

Tabla 3.3 Descripción de los pines del Arduino Nano [33]

Nombre	Pines	Descripción
Serial	0 (RX) y 1 (TX)	Usado para recibir (RX) y transmitir (TX) datos serial TTL. Estos pines están conectados a los pines del FTDI USB-to-TTL chip serial.
Interrupción externa	2 y 3	Estos pines pueden ser configurados para activar una interrupción al enviar un valor de bajo nivel (0), en un desborde ascendente o descendente, o en un cambio de nivel.
PWM	3,5,6,9,10 y 11	Proporciona salidas PWM de 8 bits con la función *analogWrite().*
SPI	10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)	Estos pines soportan la comunicación SPI.
LED	13	Se incorpora un led conectado al pin digital 13. Cuando el pin está en valor "*HIGH*", el led se enciende, mientras el pin esta en valor "*LOW*", está apagado.
I2C	A4 (SDA) y A5 (SCL)	Soporta la comunicación I2C utilizando la librería Wire
AREF	REF	Voltaje de referencia para el entrada analógica, utilizado con la función *analogReference().*
Reset	RST	Para resetear el microcontrolador

Arduino Nano tiene 8 pines analógicos de entrada, cada uno proporciona 10 (bits) de resolución (1024 valores). Los pines analógicos 6 y 7 no pueden ser utilizados como pines digitales. En la Figura 3.5 se aprecia el microcontrolador Arduino Nano [33].

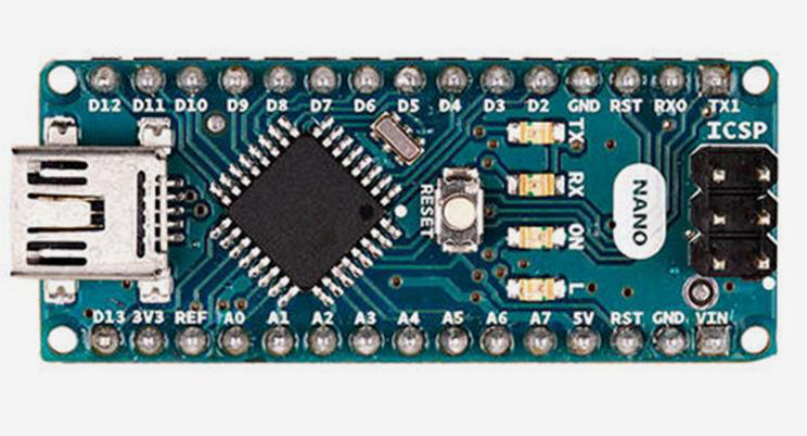


Figura 3.5 Microcontrolador Arduino Nano [33]

Neo6m GPS

El módulo A7 cuenta con sistema de navegación GPS, debido a que la sensibilidad es de -105 (dBm) no resultó ser el más adecuado para realizar el seguimiento del prototipo de geolocalización. Principalmente presentó problema para obtener la señal de posicionamiento inicial y fallas de precisión, por tal motivo, se optó por usar otro modulo GPS.

Las especificaciones del módulo Neo6m se aprecian en la Tabla 3.4**.**

Tabla 3.4 Características técnicas de módulo Neo6m [34]

Parámetro	Especificación
Tipo de receptor	50 canales, Frecuencia GPS L1
Sensibilidad	-161 (dBm)
Velocidad límite de operación	500 (m/s)
Precisión de posición horizontal	2.5 (m)
Fuente	2.7 – 3.6 (V)
Interfaz	UART, USB, SPI
Numero de pines	4
Dimensiones	16 x 12.2 x 2.4 (mm)

El módulo Neo6m se adapta a las características del prototipo debido a que cuenta con un interfaz UART, las dimensiones del módulo son pequeñas y cuenta con mayor precisión que el módulo A7, además solventa los problemas encontrados con el otro módulo. En la Figura 3.6 se aprecia al módulo Neo6m.

Figura 3.6 Módulo Neo6m con componentes [34]

La distribución de los pines se aprecia en la Tabla 3.5.

Tabla 3.5 Descripción de pines del módulo Neo6m [34]

Pines	Función
VCC	Pin suministro de energía desde 2.7 a 3.6 (V)
RX	Puerto serial para recepción de datos UART
TX	Puerto serial para transmisión de datos UART
GND	Pin a tierra

Batería

Para alimentar a todo el sistema se requiere de una batería que proporcione energía a todo el prototipo y que soporte un determinado tiempo de funcionamiento. Se determinó la corriente que cada componente requiere por medio de las fichas técnicas y empleando la Ecuación 3.1.

$$\mathrm{I} = \frac{\mathrm{V}}{\mathrm{R_{eq}}}$$

$$\mathrm{I} = \frac{\mathrm{V_T} - V_R}{\mathrm{R_{eq}}}$$

Ecuación 3.1 Ley de Ohm [35]

Se va a estimar el consumo del Arduino Nano, basado en la ficha técnica, cuenta con un procesador que consume 2.4 (mA) a 5 (V), además, utiliza el chip USB UART, cuyo consumo es de 24 (mA) [33] [36] [37]. El Arduino Nano se encuentra conectado con 6 leds con resistencias de 680 (Ohms), aplicando la Ecuación 3.1.

Donde:

V : 5 (V) voltaje

$\mathrm{V_R}$: 1.6 (V) voltaje de led

R_{eq} : 680 (Ω) resistencia equivalente

I : (A) corriente

Por lo tanto:

$$I = 5\ (mA)$$

Del cálculo anterior se determina que la corriente para un led es de 5 (mA). Por último, cada pin de entrada o salida entrega 40 (mA) [33], para el proyecto se ha configurado tres pines como salida.

Ahora se determina la corriente total que consume el microcontrolador Arduino Nano. Se aplica la ley de corrientes de Kirchhoff, esta ley se muestra en la Ecuación 3.2.

$$\sum_{k=1}^{n} lk = I_1 + I_2 + I_3 + \cdots + I_n$$

Ecuación 3.2 Ley de corrientes de Kirchhoff [35]

Donde:

I_1 : 2.4 (mA) corriente

I_2 : 24 (mA) corriente

I_3 : 6*5 (mA) corriente

I_4 : 3*40 (mA) corriente

I_K : (mA) corriente

Usando la Ecuación 3.2 se obtiene:

$$\mathrm{I_k} = 176.4\ (\mathrm{mA})$$

A la corriente total estimada para el Arduino Nano, se agrega un margen de veinte por ciento si llegara el circuito a demandar más corriente que el valor calculado, por lo tanto, el resultado total será de 194.04 (mA).

Ahora se estima la corriente que consume el módulo Neo6m, según la ficha técnica oficial del módulo el consumo máximo de amperio es de 67 (mA) cuando se alimenta a un voltaje de 3.6 (V), se tomara este valor porque el módulo se conecta al pin del Arduino que abastece los 3.3 (V) [34].

Por último, el módulo A7, basándose en la ficha técnica, tiene bajo consumo de corriente, en modo espera puede alcanzar un consumo de hasta 3 (mA), sin embargo, el máximo consumo de corriente es de 70 (mA), este valor se utiliza para el cálculo del consumo total del prototipo.

Teniendo el consumo de corriente eléctrica de todos los módulos se procede a obtener el consumo de corriente total, aplicando la Ecuación 3.2.

Donde:

$I_{arduino}$: 194.04 (mA) corriente
I_{neo6m} : 67 (mA) corriente
I_{A7} : 70 (mA) corriente
I_T : (mA) corriente

Usando la Ecuación 3.2 se obtiene:

$$\mathrm{I}_T = 331.04\ (\mathrm{mA})$$

La corriente total estimada que el prototipo de geolocalización va a consumir, será de 331.04 (mA).

El tiempo que el dispositivo esté funcionando también es un factor muy importante antes de seleccionar la batería. Este factor es conocido como amperio-hora (Ah) o miliamperio-hora (mAh) [38], este valor determina la cantidad de corriente que una batería es capaz de entregar en un determinado tiempo. Sin embargo, el consumo de corriente de la batería va a depender de la carga que el circuito este utilizando [38]. Se tiene una relación para calcular la corriente necesaria de una batería dependiendo del tiempo estimado a utilizar el dispositivo, según la Ecuación 3.3.

$$\mathrm{t} = \frac{\mathrm{l}_{\mathrm{bateria}}}{\mathrm{I}_{\mathrm{circuito}}}$$

Ecuación 3.3 Tiempo de descarga de batería [38]

Donde:

t : 6 (h) horas
$\mathrm{l}_{\mathrm{batería}}$: (mA) miliamperio
$\mathrm{I}_{\mathrm{circuito}}$: 331.04 (mA) miliamperio

Usando la Ecuación 3.3 se obtiene:

$$\mathrm{I}_{\mathrm{batería}} = 1986.24\ (\mathrm{mA})$$

Se estima que el dispositivo tenga un funcionamiento de seis horas, por lo tanto, se requiere de una batería que suministre más de 1986 (mAh). Tener en cuenta que la corriente de consumo del prototipo es una estimación, por tal razón, el tiempo de funcionamiento del dispositivo durará más o menos de seis horas.

Finalmente, el tipo de baterías debe ser recargable entre este grupo se encuentran las de níquel-cadmio (Ni-Cd), níquel-hidruro metálico (NiMH), ion-litio (Li-ion) y polímero de ion-litio (LiPo). Las baterías del tipo LiPo son las que presentan mayor densidad de carga [38]. Debido a que las baterías LiPo solo brindan voltajes de 3.7 (V), 7.4 (V), 11.8 (V), etc., se seleccionó un voltaje de 7.4 (V) y miliamperio-hora de 1500 (mAh), debido a que se utiliza un conversor reductor DC-DC, realizando las conversiones se obtuvo la corriente necesaria para alimentar al prototipo por seis horas.

Para demostrar lo anterior se emplea la Ecuación 3.4 y Ecuación 3.5, dado a que se utiliza un reductor DC-DC, se va a disminuir el voltaje de 7.4 (V) a 5 (V), lo que implica que la corriente va a aumentar a la salida del conversor DC-DC.

Se utiliza la Ecuación 3.4 para calcular la potencia de entrada como la potencia de salida.

$$\mathrm{P} = \mathrm{V} * \mathrm{I}$$

Ecuación 3.4 Potencia eléctrica [35]

Donde:

$\mathrm{P}_{\mathrm{entrada}}$: (W) watts
$\mathrm{V}_{\mathrm{entrada}}$: 7.4 (V) voltaje
$\mathrm{I}_{entrada}$: 1500 (mA) corriente

Por lo tanto:

$$\mathrm{P}_{entrada} = 11100\ (\mathrm{mW})$$

El valor de potencia de salida va a ser igual a la potencia de entrada, aplicando la Ecuación 3.5, con el fin de determinar cuál será la corriente de salida.

$$\mathrm{P}_{\mathrm{entrada}} = \mathrm{P}_{\mathrm{salida}}$$
$$\mathrm{V}_{\mathrm{entrada}}\mathrm{I}_{\mathrm{entrada}} = \mathrm{V}_{\mathrm{salida}}\mathrm{I}_{\mathrm{salida}}$$

Ecuación 3.5 Igualdad de potencias [39]

Donde:

$P_{entrada}$: 11100(mW) potencia de entrada

V_{salida} : 5 (V) potencia de salida

I_{salida} : (mA) amperios

Por lo tanto:

$$I_{salida} = 2220\ (\mathrm{mW})$$

Como la corriente de salida del conversor DC-DC, es mayor al valor calculado para la corriente de la batería de 1986 (mAh), entonces, por ello se ha seleccionado la batería LiPo con voltaje de 7.4 (V) y miliamperio-hora de 1500 (mAh).

Conversor DC-DC *step-down* LM2596

Una batería de 7.4 (V), con corriente de 1500 (mAh), proporciona la energía necesaria al prototipo por un determinado tiempo, debido a que todos los módulos trabajan a un voltaje menor o igual a 5 (V), por tal razón, se opta por un conversor DC-DC para disminuir el voltaje de 7.4 (V) a 5 (V). En la Tabla 3.6 se aprecia las características técnicas del conversor DC-DC.

Tabla 3.6 Características técnicas del conversor DC-DC *step down* LM2596 [40]

Características	Valores
Voltaje de entrada	4.5 – 40 (V_{DC})
Voltaje de salida	1.23 – 37 (V_{DC})
Suministro máximo de voltaje	45 (V)
Corriente de salida máxima	3 (A)
Eficiencia de conversión	95%
Frecuencia de trabajo	150 (KHz)
Dimensiones	43*21*13 (mm)

Este conversor es un regulador conmutado y es más eficiente que los reguladores lineales como el LM7805, además, el voltaje de salida es ajustable siempre que el voltaje de entrada sea mayor a 1.5 (V_{DC}) [40]. En la Figura 3.7 se aprecia el conversor DC-DC, como se observa cuenta con cuatro pines que son dos entradas y dos salidas de voltaje.

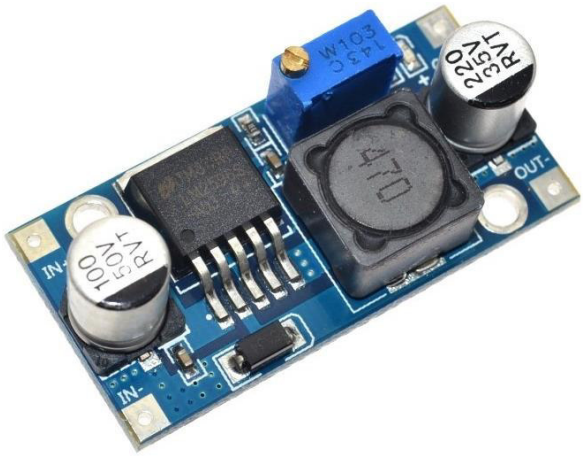


Figura 3.7 Conversor DC-DC *Step-Down* LM2596 [40]

3.3 Construcción del prototipo de geolocalización

Desarrollo de código para microcontrolador

El microcontrolador Arduino Nano es el encargado de controlar los dos módulos para obtener los datos de posición que se envían por la red, para lograr esto se necesita conocer las características de *software* de los módulos tanto los comandos AT, conversión de los datos NMEA, entre otros.

Descripción del código del prototipo de geolocalización

La descripción general del código del prototipo de geolocalización se muestra en Figura 3.8, se detalla como el dispositivo empieza a recolectar la información para enviar al servidor *ThingSpeak*.

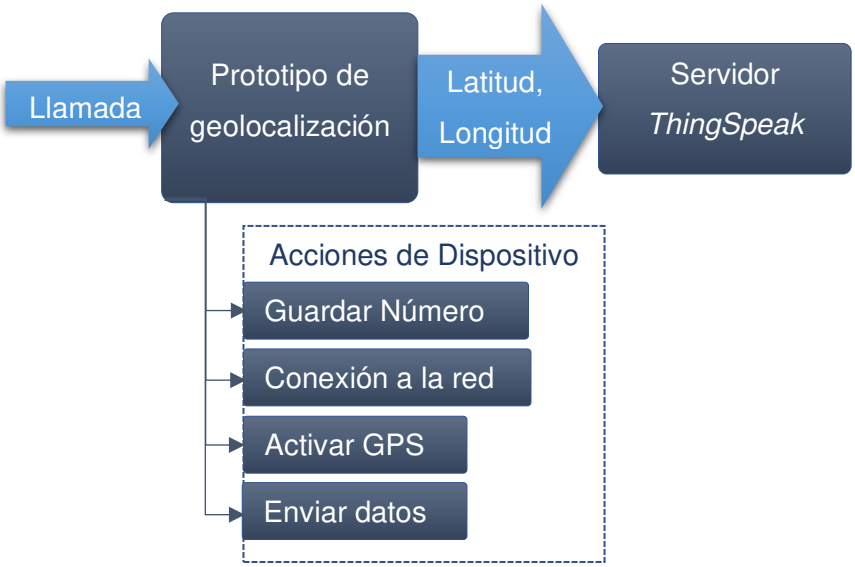


Figura 3.8 Descripción general del código del prototipo de geolocalización

Primero recibe una llamada el prototipo de geolocalización porque cuenta con tecnología GSM, posteriormente el dispositivo realiza una serie de actividades que tienen el siguiente orden:

1. Guarda el número telefónico.
2. Establece la conexión a la red por GPRS.
3. Activa el GPS para la recolección de datos de posición.
4. Establece la conexión con el servidor *ThingSpeak*.
5. Envía datos de posición al servidor *ThingSpeak*.
6. Cierra la conexión con el servidor *ThingSpeak*.

Conexión entre Arduino Nano y los módulos

La lógica UART realiza la función de interfaz, transporta los datos y los transforma de carácter a carácter [19] [20]. El Arduino Nano tiene varias formas de comunicación con otros microcontroladores, o dispositivos. El ATmega 328 (cerebro del Arduino) brinda comunicación serial UART TTL, los cuales están habilitados en el pin digital 0 (RX) y 1 (TX). Estos pines son utilizados para la comunicación con la computadora, es decir, es el puerto serial principal destinado a propósito de depuración del código de Arduino [33]. Por tal razón no se utilizaron los pines 0 (RX) y 1 (TX), para la comunicación serial UART con el módulo A7.

Por lo tanto, se requirió de otro puerto serial, por ello se utiliza la librería *SoftwareSerial*, el cual permite asignar una comunicación serial en otros pines digitales de Arduino. Esta librería utiliza *software* para replicar las funcionalidades de las líneas de RX y TX. En el proyecto se utilizaron los pines 8, 9, 10 y 11 de Arduino para ser usadas como líneas seriales virtuales RX y TX. Los pines virtuales RX (8, 10) se configuraron para escuchar cualquier dato que venga por vía serial y luego son transmitidos por la línea virtual TX (9, 11) [41].

El módulo A7 utiliza dos pines para la comunicación: PIN receptor y PIN trasmisor, en el módulo se nombra como U_RXD y U_TXD respectivamente. Estos pines permiten la comunicación con el Arduino a través de comandos AT, los pines se muestran en la Figura 3.9.

Figura 3.9 Pines de transmisión y recepción del módulo A7

Para establecer la conexión entre Arduino Nano y el módulo A7 GSM/GPRS/GPS, se utilizaron los pines 9 (TX) y 8 (RX) de Arduino que se conectaron a los pines U_RXD y U_RXD del módulo A7, es decir, el pin U_RXD se conectó con 9 (TX) y U_TXD con 8 (RX). Mientras que los pines del módulo Neo6m se conectaron al Arduino Nano en los pines 10 (RX) y 11 (TX) para la transmisión de datos NMEA. Es decir, el pin 10 (RX) se conectó al pin TX y el pin 11 (TX) con el pin RX del módulo Neo6m.

Comunicación serial

Para la comunicación del Arduino Nano y los módulos, se utilizó la librería de Arduino "SoftwareSerial", como se mencionó previamente, esta librería permite configurar los pines del Arduino para la recepción y transmisión de datos de manera serial. En la Figura 3.10 se muestra el diagrama de flujo para la comunicación serial entre el Arduino y los módulos.

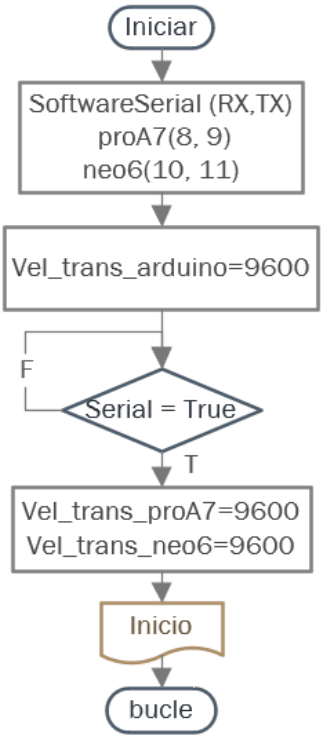


Figura 3.10 Diagrama de flujo de conexión serial con los módulos

Se denominó “proA7” al módulo A7 y se utilizaron los pines 8 y 9 del Arduino, mientras al módulo Neo6m se asignó el nombre de “neo6” y se usaron los pines 10 y el 11 del Arduino. La librería permite asignar la velocidad de transmisión de datos serial (medida en “baudios”) y se estableció la velocidad de 9 600 (baudios) tanto para el Arduino y para los módulos. Los valores vel_trans_arduino, vel_trans_A7 y vel_trans_Neo6 hacen referencia a la velocidad de transmisión que se asignan al Arduino y a los módulos.

Se estableció la forma de receptar todos los datos que recibió el Arduino de los módulos utilizando la función llamada “ComunicaciónSerial” y otra función dentro del mismo denominada “serialA7”, en la Figura 3.11 se muestra el diagrama de flujo que describe esta función.

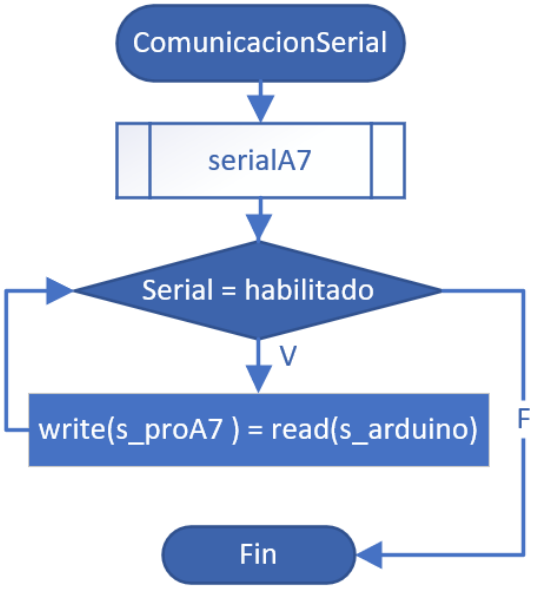


Figura 3.11 Recepción de datos con Conexión Serial

Dentro de la función “ComunicacionSerial” se llama a la función “serialA7” la cual es encargada de receptar los datos y procesarlos, por medio de un bucle se verifica si el serial del Arduino está enviando comandos AT. Si es verdadero el Arduino lee los comandos mandados, para que el módulo A7 lo recepte, procese y realice la acción con respecto al comando AT.

En la Figura 3.12, se describe la función denominada “serialA7” que se encarga de receptar y procesar los datos que llegan del módulo A7 al momento de receptar una llamada telefónica.

Primero se indica al Arduino Nano que empiece a escuchar al módulo A7 con la función llamada “*listen*()” y se verifica si está habilitado y receptando datos. Si es verdadero se

almacenan los valores que entrega el módulo A7 en formato “char” en la variable “valorChar”, sin embargo, el objetivo es recolectar cada carácter *char* que es enviado al Arduino, por tal razón, el contenido que tiene la variable “valorChar” es almacenado en otra variable con nombre “valorStr”, con esto se logra formar una cadena de *char’s* o también conocido como *string.*

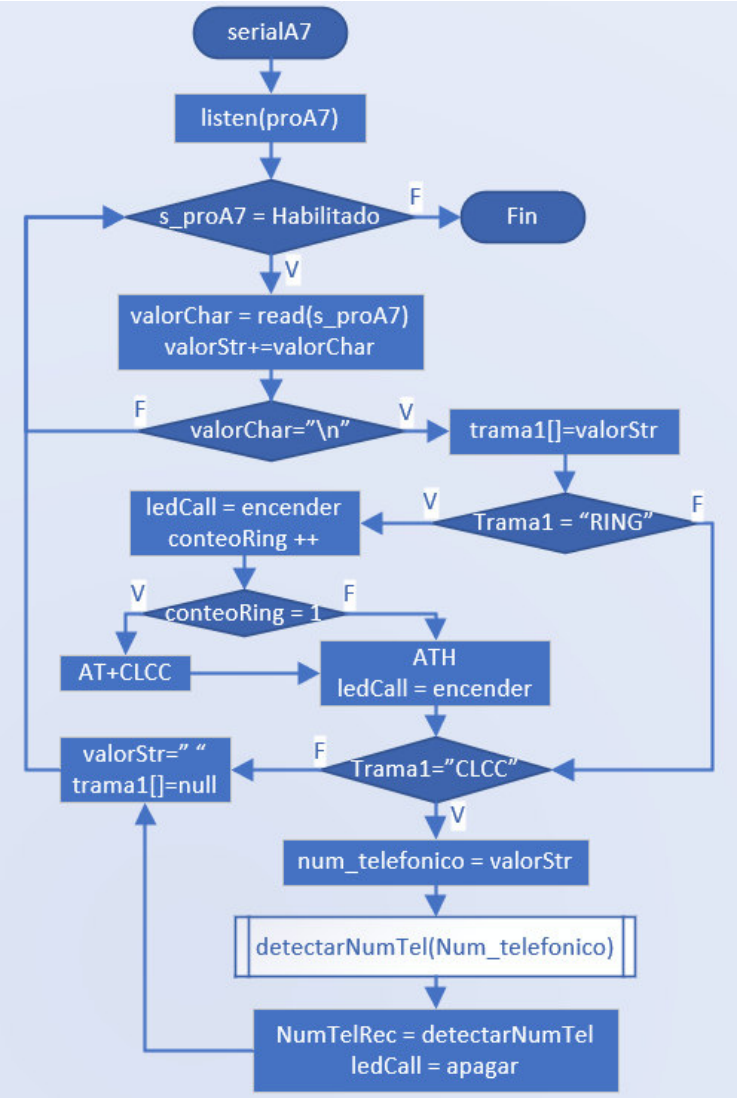


Figura 3.12 Función para procesar llamada al módulo A7

Un valor *char* puede ser un salto de línea (\n), es por ello que al detectar un “\n” la variable “valorStr” se almacena en una “trama1” caso contrario vuelve a verificar si el módulo A7 está enviando datos. La “trama1” se utiliza para verificar si contiene el valor de “*RING*”, esto significa que está recibiendo una llamada por la red GSM del módulo A7, si esto es verdadero se enciende el led denominado “*CALL*”. Además, se ejecutan el comando AT+CLCC usado obtener el número telefónico de quien realiza la llamada y ATH para colgar las llamadas.

Si la “trama1” es igual a “CLCC” significa que se ha ejecutado el comando AT+CLCC, el cual contiene una cadena de valores y entre todos esos valores contiene el número telefónico, por tal razón se extrajo dicho valor por medio de la función denominada “detectarNumTel”.

El código de programación de las funciones “ComunicacionSerial” y “serialA7” desarrollado en Arduino se incluye en el Anexo 2.

Detectar número telefónico

La función “detectarNumTel” mencionada en la sección anterior permite extraer el número telefónico, en el diagrama de flujo de la Figura 3.13, se aprecia como esta función logra dicho objetivo. La función “detectarNumTel” utiliza el comando de “Detectar llamadas” de los comandos AT del módulo A7.

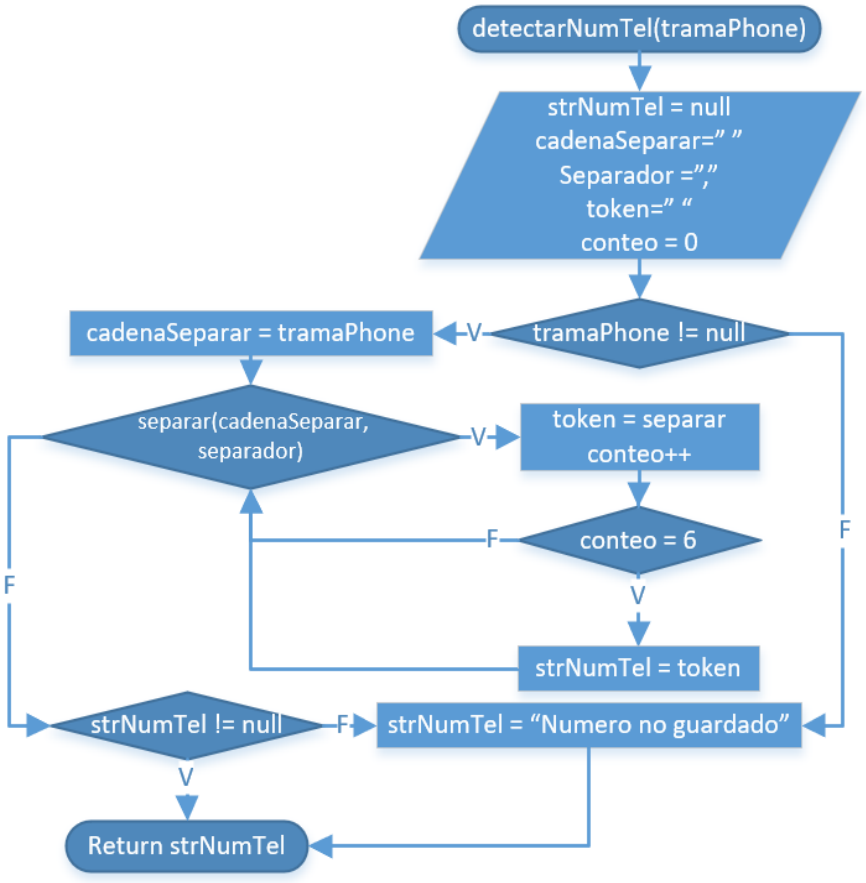


Figura 3.13 Detección de llamadas

La función “detectarNumTel” devuelve un valor del tipo “*string*”, cuenta con un parámetro que va a ser una trama de datos que contiene el número telefónico, con una condición se comprueba que no esté vacío este parámetro, si no es el caso, se almacena en una variable llamada “cadenaSeparar”. Previamente, se declara la variable “Separador” y su

valor es una coma (,) ya que la trama separa sus valores con este signo. Ahora con la acción “separar” se divide la trama y el resultado de cada uno se guarda en la variable “*token*”. El número telefónico estará ubicado en la sexta posición, se utiliza un contador que cuenta las veces que está dividiendo la trama como se aprecia en la Figura 3.14.

Trama de resultado CLCC													
CLCC:	id1	,	dir	,	stat	,	**mode**	,	**mpty**	,	**number**	,	**type**
	1°		**2°**		**3°**		**4°**		**5°**		**6°**		**7°**

Figura 3.14 Contenido de trama CLCC

Si el conteo es igual a seis entonces se almacena la variable “*token*” en la variable “strNumTel” y posteriormente la función “detectarNumTel” entrega la variable “strNumtel”, que contiene el número telefónico.

El código de programación en Arduino se aprecia en Anexo 3.

Datos GPS

El módulo Neo6m transmite datos NMEA los cuales el Arduino Nano va a recibir por el puerto serial. Todos los valores GPS se envían en el formato NMEA 0183. La Figura 3.15 muestra este formato utilizado a nivel mundial en los dispositivos de navegación satelital.

```
$GPGGA,220329.00,0016.68738,S,07832.97892,W,1,10,0.83,2901.6,M,13.7,M,,*6E
$GPGSA,A,3,01,03,30,07,28,06,09,14,46,19,,,1.80,0.83,1.60*0B
$GPGSV,4,1,14,01,27,030,30,02,03,218,,03,29,102,30,06,40,219,34*70
$GPGSV,4,2,14,07,74,163,37,09,23,173,26,11,10,030,30,14,24,352,23*70
$GPGSV,4,3,14,19,25,286,29,22,,,20,28,19,347,28,30,64,315,32*4E
$GPGSV,4,4,14,46,32,270,32,51,56,270,37*7D
$GPGLL,0016.68738,S,07832.97892,W,220329.00,A,A*68
$GPRMC,220330.00,A,0016.68735,S,07832.97893,W,0.012,,010321,,,A*77
$GPVTG,,T,,M,0.012,N,0.022,K,A*20
```

Figura 3.15 Valores en formato NMEA

De todos estos datos que se devuelven en el formato NMEA 0183, los valores que son importantes para el proyecto van a ser el $GPRMC y el $GPGGA. Los datos que $GPRMC, ver Figura 3.16, va a entregar valores muy importantes, estos son:

A. Hora en coordenadas universales hh:mm:ss.
B. Indicador de conexión a GPS si el valor es “A” significa activa o “V” significa sin señal.

C. Envió de trama con valores de latitud y longitud, expresados en grados y minutos, y ubicación de hemisferio. Los valores de hemisferios se muestran negativos si la ubicación se encuentra en el Oeste (W) y al Sur (S), pero si la ubicación es el Norte (N) o el Este (E), entonces no se agrega el signo menos.

Figura 3.16 Datos de $GPRMC

Los valores de latitud y longitud están expresados en grados y minutos estos valores no son útiles si se desea ingresarlos en un interfaz gráfico de mapas, ya que estos solo aceptan en grados, por tal razón se debe de convertir de grados y minutos a solo grados. Dado a que un grado equivale a 60 minutos, se debe dividir los valores de minutos para 60.

Ejemplo de datos $GPRMC

$GPRMC,012320.000,A,0016.68840,S,07832.97737,W,0.00,0.00,060320,,,A*62

Campos

- **Primero:** 012320.000 Hora GMT.
- **Segundo:** A = Activo.
- **Tercero y cuarto:** 0 grados 16.688840 minutos S (-0.27814733).
- **Quinto y sexto:** 78 grados 32.97737 minutos W (-78.5496228).
- **Séptimo:** Velocidad 0.00 (estático no se mueve).
- **Noveno:** 06/03/20 DD/MM/AA.
- **Ultimo campo:** Variación magnética A*62.

Para el caso de datos $GPGGA, ver Figura 3.17, va a estregar los siguientes valores:

A. Hora en coordenadas universales hh:mm:ss
B. Envió de trama con valores de latitud y longitud, expresados en grados y minutos, y la ubicación de hemisferio.
C. Indicador arreglado de ubicación, si el valor es 1 (uno) significa que se corrigió la ubicación, caso contrario entrega un 0 (cero).
D. Indicador de número de satélites para el seguimiento.
E. Indicador de dilución horizontal de la posición.
F. Indicador de altura en metros sobre el nivel del mar, este valor es utilizado en sistemas de telemetría aérea o para analizar datos de vuelo.

G. Indicador de altitud relativa de la ubicación.

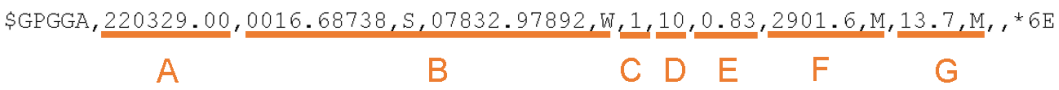


Figura 3.17 Datos de $GPGGA

Ejemplo de datos $GPGGA

$GPGGA,203544.000,0438.9198,N,07404.3962,W,1,10,0.94,2597.0,M,3.4,M,,*69

Campos

- **Primero:** 203544.000 Hora GMT.
- **Segundo y tercero:** 04 grados 38.9198 minutos N (4,648663).
- **Cuarto y quinto:** 74 grados 04.3962 minutos W (-74,07327)
- **Sexto:** 1 ubicación arreglada
- **Séptimo:** 10 satélites en seguimiento
- **Octavo:** 0.94 dilución horizontal
- **Noveno y Décimo**: 2597.0,M altitud en metros sobre nivel del mar
- **Undécimo y duodécimo**: 3.4,M altitud relativa

Obtención de los datos

Para obtener los datos se emplea la librería TinyGPS++, la cual analiza repetidamente los valores que entrega el módulo GPS utilizando el método "*encode()*". El objeto principal de TinyGPS++ contiene varios sub-objetos importantes, entre ellos los que se utilizaron para el prototipo se muestran en la Tabla 3.7.

Tabla 3.7 Sub-objetos de la librería TinyGPS++

Sub-Objetos	Descripción
GPS.encode(serial.read())	Codifica los datos que llegan desde el serial.
gps.location.lat(),6	Obtiene datos de latitud en grados
gps.location.lng(),6	Obtiene datos de longitud en grados

Cada uno de estos sub-objetos se obtienen tanto de los datos $GPRMC y $GPGGA, y se van a utilizar solo los necesarios para obtener la posición del dispositivo de geolocalización.

Recepción de datos GPS por Arduino

Para conseguir los datos de posición para el desarrollo del prototipo de geolocalización, se describen los procesos realizados por medio del diagrama de flujo de la Figura 3.18**.**

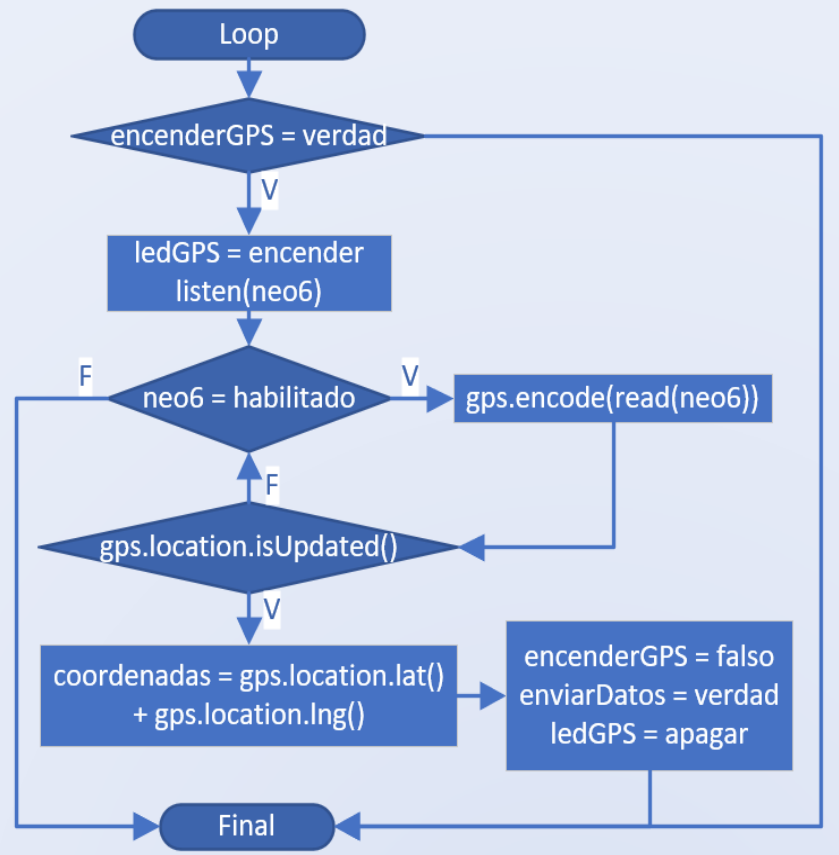


Figura 3.18 Procesos para la recepción de datos GPS

Este código forma parte de la función "*Loop*". Primero se verifica que la variable "encenderGPS" sea verdadera, en el caso de serlo se enciende el ledGPS, posteriormente, para empezar a recolectar los datos que envía el módulo Neo6m, se debe de especificar al Arduino que empiece a escuchar al módulo con la función "*listen()*", después se verifica que el módulo Neo6m este habilitado, y si es así se utiliza el sub-objeto "*gps.encode()*" que va a recoger los datos GPS en formato NMEA y se codificaran los datos en sus respectivas valores de latitud y longitud.

Posteriormente se utiliza el sub-objeto "*gps.location.isUpdated()*" y si es verdadero entonces se recoge los datos de latitud y longitud con "*gps.location.lat()*" y "*gps.location.lng()*" respectivamente, y se guardan en la variable "coordenadas", finalmente se asigna a la variable "encenderGPS" el valor de falso, a la variable "enviarDatos" el valor de verdadero y se apaga el led GPS.

El código de recepción de datos GPS se encuentra en el Anexo 4**.**

Conexión desde prototipo de geolocalización a la red

Se debe de acceder a Internet para enviar los datos de latitud y longitud. El módulo A7 cuenta con las funciones de GPRS y TCP las cuales son necesarios para establecer la conexión a la red y mandar los datos.

Para establecer una conexión a la red por GPRS y TCP se debe utilizar los comandos AT del módulo A7. Los comandos utilizados son los denominados servicios de red, GPRS, parámetros PDP, TCP/IP, en la Figura 3.19, se muestra el diagrama de flujo del código en Arduino que establece la conexión a la red.

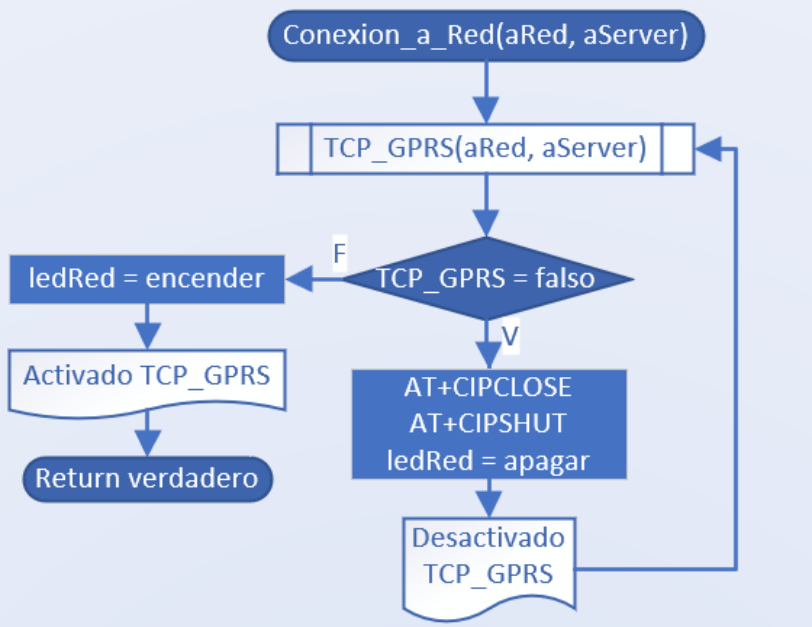


Figura 3.19 Procesos para establecer conexión a la red

La función que controla la conexión a la red se denomina “conexion_a_Red” esta llama a otra denominada “TCP_GPRS”. Estas dos funciones entregan valores booleanos y también son las encargadas de establecer la conexión a la red móvil para acceder a Internet. Las dos funciones cuentan con dos parámetros “aRed” y “aServer”, se tratará sobre estos parámetros en los siguientes párrafos.

Dentro de la función “conexion_a_Red” al llamar a la función “TCP_GPRS” que es la encargada de establecer conexión tanto en la red como con el servidor, si esta devuelve el valor de verdadero entonces está conectado a la red, caso contrario cierra la conexión a la red con los comandos AT+CIPCLOSE y AT+CIPSHUT, volviendo a llamar a la función “TCP_GPRS”.

En la Figura 3.20 se aprecia el diagrama de flujo de la función “TCP_GPRS”, esta función es la más importante debido a que cuenta con dos parámetros que son aRed y

aServer las cuales especifican con quien se desea establecer conexión, si con la red o con el servidor.

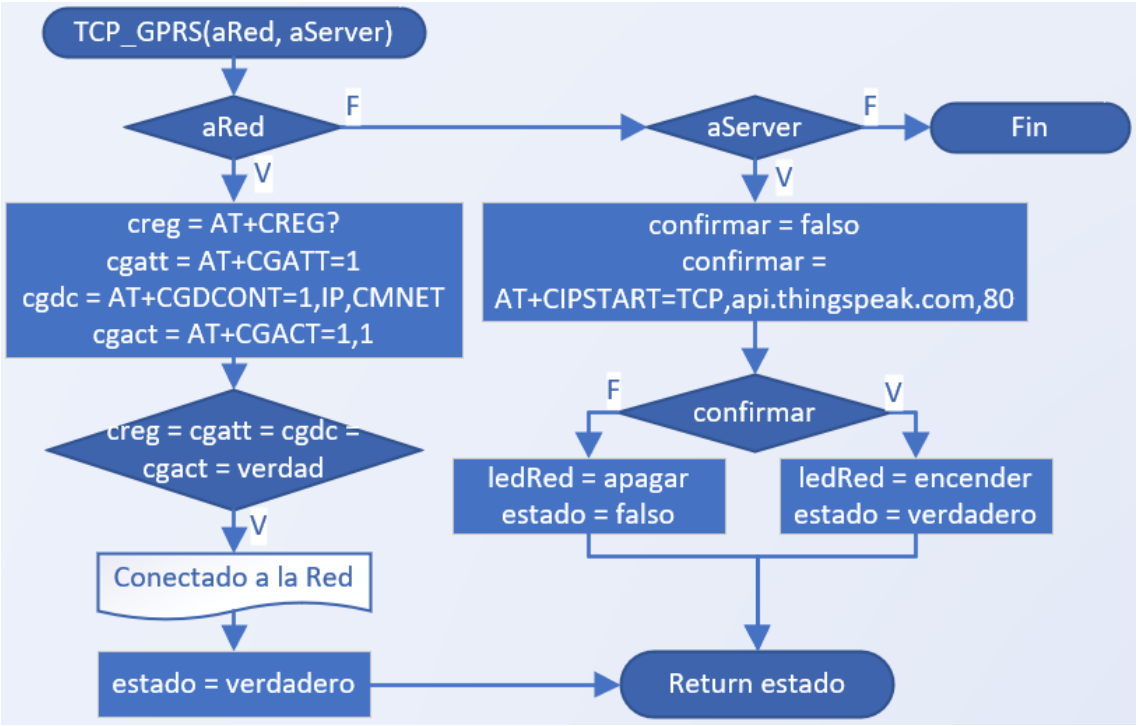


Figura 3.20 Función principal para la conexión a la red

En el diagrama de flujo de la función “TCP_GPRS” se utilizan condicionales que verifican si desea activar la red con el parámetro “aRed”, si este parámetro es verdadero entonces se ejecutan los comandos AT para establecer conexión a la red como AT+CREG?, AT+CGATT, AT+CGDCONT, por último, AT+CGACT y si se ejecutan correctamente los comandos entonces se envía un mensaje de éxito al estar “Conectado a la red”, finalmente a la variable “estado” se le asigna el valor de verdadero.

Si el parámetro “aServer” es verdadero entonces se efectúa el comando AT para establecer conexión con el servidor *ThingSpeak*, como es AT+CIPSTART y si se ejecuta correctamente el comando entonces se enciende el led “RED” y la variable denominada “estado” se asigna un valor de verdadero, caso contrario no se enciende el led “RED” y la variable “estado” toma el valor de falso, por último, retorna el valor de “estado”.

El código de Conexión desde prototipo de geolocalización a la red se encuentra en el Anexo 5.

Proceso del envió de datos GPS al servidor

Para el envío de datos de latitud y longitud al servidor *ThingSpeak*, se debe seguir una serie de pasos que se describe en la Figura 3.21.

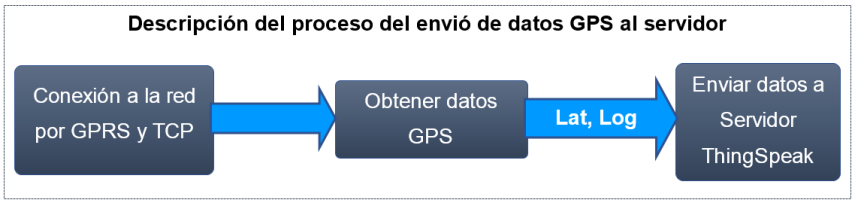


Figura 3.21 Descripción del proceso para enviar datos por la red

Los procesos para enviar los datos de posición al servidor se muestran en la Figura 3.22, en donde la función a utilizar para realizar esta acción se denomina "enviarDatosGPS", la cual cuenta con un parámetro llamado "posicion". Este parámetro contiene las coordenadas de latitud y longitud que se obtienen con el módulo Neo6m.

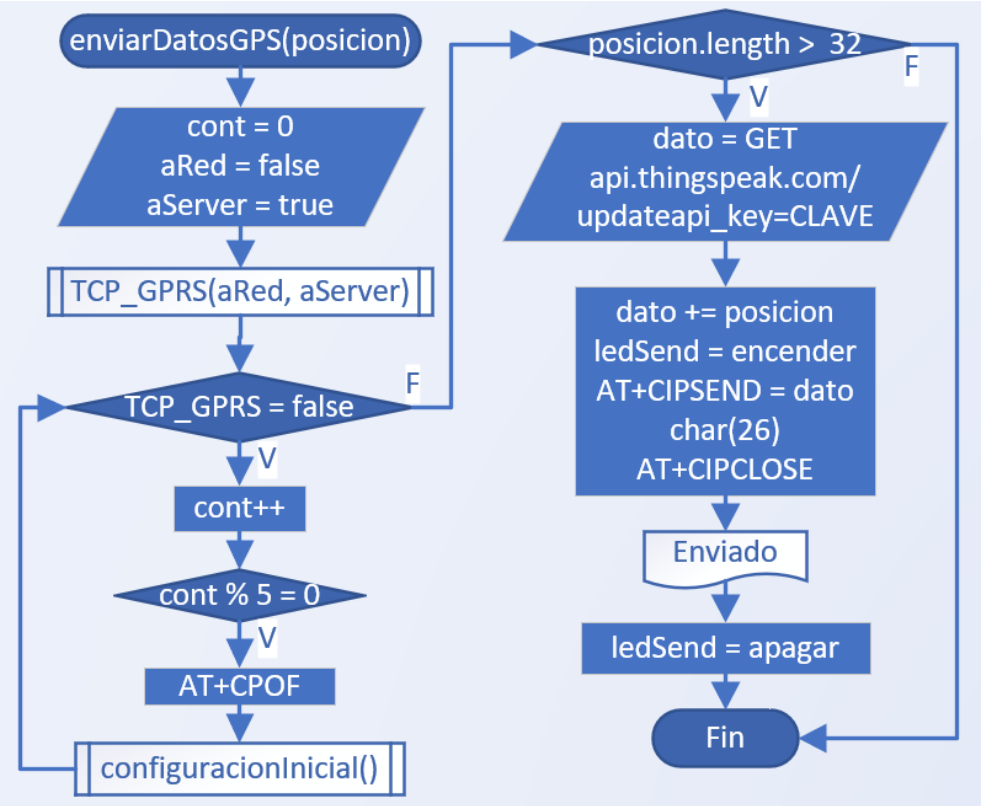


Figura 3.22 Procesos para enviar datos GPS al servidor *ThingSpeak*

A las variables denominadas "aRed" y "aServer" se les asigna los valores de falso y verdadero respectivamente, se asignan estos valores debido a que se desea establecer conexión con el servidor y no con la red. A las variables "aRed" y "aServer" se las envía a la función "TCP_GPRS" cuya función ya se ha detallado en la sección anteriores.

Si el valor que devuelva la función “TCP_GPRS” es falso entonces se vuelve a ejecutar el código de la función TCP_GPRS y la variable “cont” empieza a aumentar de valor tal que si “cont” llega a ser múltiplo o igual a cinco se ejecuta el comando AT +CPOF, el cual apaga los servicios GPRS y GSM del módulo A7, finalmente se ejecuta la función “configuracionInicial” en el Arduino Nano que se encarga de las configuraciones iniciales del módulo A7 y de Neo6m, esto se realiza debido a que se apagaron los servicios.

Si la función “TCP_GPRS” devuelve un valor verdadero entonces significa que se ha establecido la conexión con el servidor, por lo tanto, se pueden enviar los datos de posición. Posteriormente se verifica que el contenido de la variable “posicion” tenga una longitud mayor a 32 esto permite comprobar que la variable no está vacía.

Si la condición es verdadera se le asigna a la variable “dato” la llave API del servidor *ThingSpeak* para escribir datos en el servidor, también se le agrega los valores que contenga la variable “posicion” y por último se enciende el led “*SEND*” indicando que se envían los datos al servidor, cuando se terminan de enviar los datos se muestra un mensaje indicando que se a enviando al servidor y se apaga el led “*SEND*”.

El código de Proceso del envió de datos GPS al servidor se encuentra en Anexo 6.

Proceso central controlado por llamada

Una vez que se han descrito los procesos más importantes como: Detectar número telefónico, Datos GPS, recepción de datos GPS, conexión desde prototipo de geolocalización a la red y proceso de envió de datos GPS al servidor. Ahora por medio de un proceso central denominado “*Loop*” se pueden utilizar todos los procesos descritos anteriormente para enviar los datos de posición al servidor, cuando recibe una llamada telefónica. Para lograr esto se siguió la serie de procesos detallados en la Figura 3.23.

Al encender el prototipo de geolocalización en la función “*Loop*” se verifica si las condiciones son verdaderas, en el caso de que no sean entonces se ejecuta la función “ComunicacionSerial”. Dicha función ya se ha explicado en la sección Comunicación serial, en resumen, detecta cuando el módulo A7 recibe una llamada y si lo detecta recoge el número telefónico y lo almacena en la variable global “NumTelRec”.

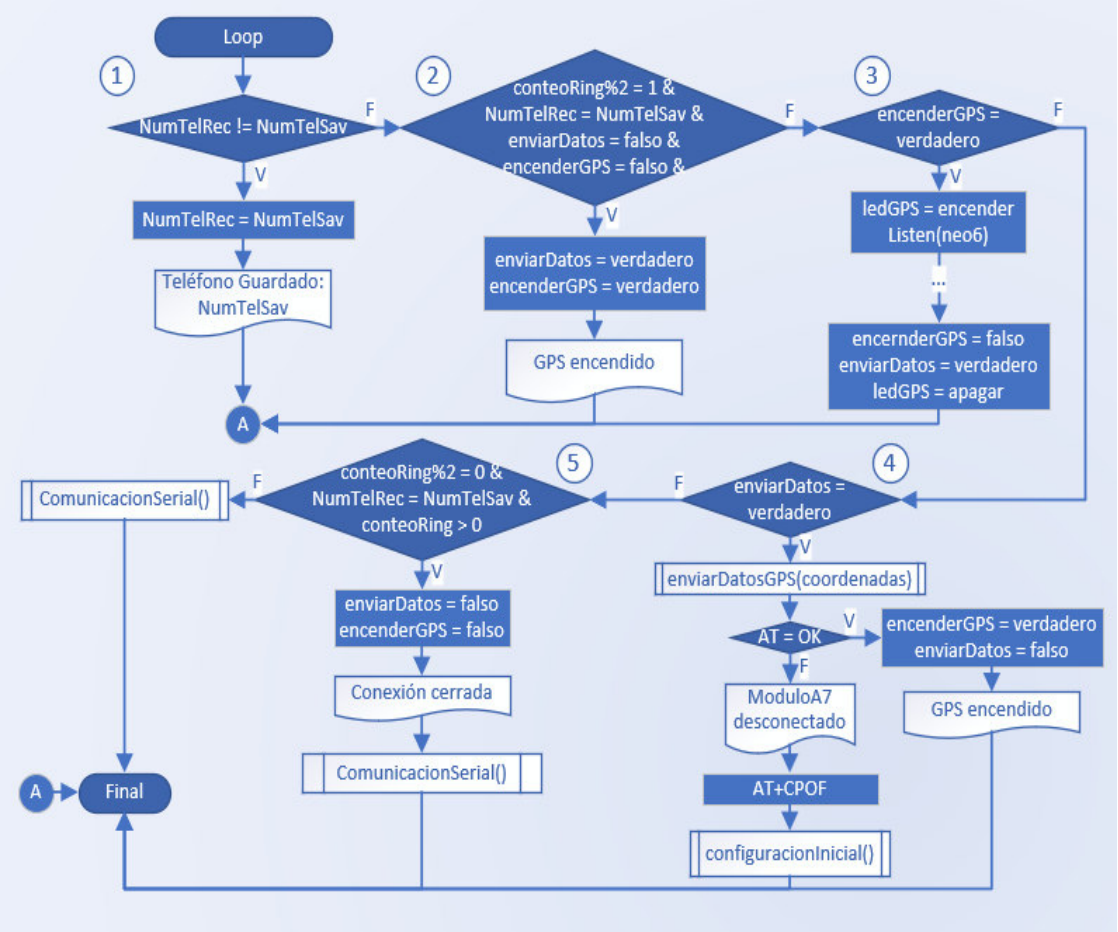


Figura 3.23 Proceso central en bucle *Loop*

En la primera condición se verifica si las dos variables “NumTelRec” y “NumTelSav” son diferentes, estas variables al inicio están vacías por ello son iguales. La variable “NumTelRec” se declara en la función “serialA7” (dentro de la función “ComunicacionSerial”) y cuyo valor es dado por la función “detectarNumTel”. La variable “NumTelRec” recibe un valor cuando el módulo A7 recibe una llamada y recoge el número telefónico. Por lo tanto, las dos variables dejan de ser iguales y se cumple la primera condición para posteriormente asignarle el valor de la variable “NumTelRec” a la variable “NumTelSav”.

En la segunda condición se compara diferentes variables como “conteoRing”, “NumTelRec”, “NumTelSav”, “enviarDatos” y “encenderGPS”. La variable “conteoRing” es un contador de llamadas, se declara en la función “serialA7”, a esta variable se le aplica el módulo para dos, si el resultado es igual a 1 significa que el valor de “conteoRing” es un valor impar, pero si es cero significa que es par.

Para ejecutar la segunda condición, debe cumplir con los tres parámetros que son:

1. Si la variable “conteoRing” es un valor impar.
2. Si las variables “NumTelRec” y “NumTelSav” contienen el número telefónico de la llamada realizada al prototipo de geolocalización.
3. Si la variable “enviarDatos” y “encenderGPS” tienen el valor de falso

Solo si cumplen estas tres condiciones se efectúa la segunda condición y se pueden ejecutar los procesos para empezar a tomar datos de posicionamiento y para enviar los datos al servidor *ThingSpeak* para lograr estas acciones a las variables “enviarDatos” y “encenderGPS” se le asignan el valor de verdadero, estas variables son importantes para que se ejecuten las siguientes condiciones que se encuentran en el “*Loop*”.

En la tercera condición si la variable “encenderGPS” es verdadera se ejecuta el código para recoger los datos de posición que se obtiene del módulo Neo6m y almacenarlo en la variable “coordenadas”, este código ya se ha detallado en la sección Recepción de datos GPS por Arduino.

Posteriormente, luego de obtener los datos de posición y almacenarlo en la variable “coordenadas”, en la cuarta condición se verifica si la variable “enviarDatos” es verdadera, si es el caso llama a la función “enviarDatosGPS” asignado como parámetro el valor de la variable “coordenadas”, el código que conforma la función “enviarDatosGPS” se detalla en la sección Proceso del envió de datos GPS al servidor. Luego de enviar los datos al servidor, se envía un comando AT al módulo A7 para verificar si el módulo está funcionando correctamente o si se ha desconectado. Si la respuesta es “OK” el módulo A7 está funcionando, por lo tanto, a las variables “encenderGPS” y “enviarDatos” se asignan los valores verdadero y falso respectivamente, este proceso se repite para obtener valores de posición y luego enviar los datos al servidor, hasta que se vuelva a llamar al módulo A7.

La quinta condición compara las variables “conteoRing”, “NumTelRec” y “NumTelSav”, si se cumple la condición se efectúan los procesos para detener la recolección de datos de posición enviados por el módulo Neo6m y también detener el envió de los datos al servidor *ThingSpeak*. Para ejecutar la condición se debe cumplir con algunos parámetros:

1. Si la variable “conteoRing” es par.
2. Si “NumTelRec” y “NumTelSav” son iguales.
3. Si “conteoRing” es mayor a cero.

Solo si cumplen estos tres parámetros se ejecuta la quinta condición, la cual consiste en declarar como falso a las variables “enviarDatos” y “encenderGPS”, posteriormente se llama a la función ‘ComunicacionSerial’. Dicha función se ha detallado en la sección Comunicación serial.

El código de proceso central de inicio de actividades por llamada se aprecia en el Anexo 7**.**

Diseño del circuito

Para el diseño del circuito se utilizó el *software* Proteus 8, el cual brinda la capacidad de diseñar y simular el circuito, así como la placa del circuito impreso (PCB). Los componentes eléctricos utilizados para el diseño del circuito del prototipo de geolocalización se enlistan en la Tabla 3.8.

Tabla 3.8 Componentes Eléctricos para prototipo de geolocalización

Componente	Descripción
Microcontrolador	Arduino Nano
Módulo GSM/GPRS	módulo A7
Módulo GPS	Neo6m
Módulo conversor DC-DC	DC-DC reductor LM2596
Batería	Litio 7 V 1500 mA
Transistores	NPN 2N2222
Leds	Verde 1.6 V
Resistencia	10 KΩ
Interruptor	Simple
Pulsador	Simple

El diseño del circuito se muestra en la Figura 3.24, en la imagen se aprecia el diseño completo del circuito implementado en el prototipo de geolocalización. El circuito cuenta con un pulsador de reinicio tanto para el Arduino Nano como para el módulo A7, con esto se puede resetear todo el circuito. Además, cuenta con un interruptor el cual se conecta con la batería, y la función que cumple es proporcionar energía a todo el circuito. El circuito tiene leds que sirven como indicadores del comportamiento del prototipo de geolocalización. Por último, se encuentran los componentes más importantes que son el Arduino Nano para controlar a todo el sistema, el módulo A7 que brinda los servicios de GSM y GPRS, y el Neo6m que obtiene los datos de posición.

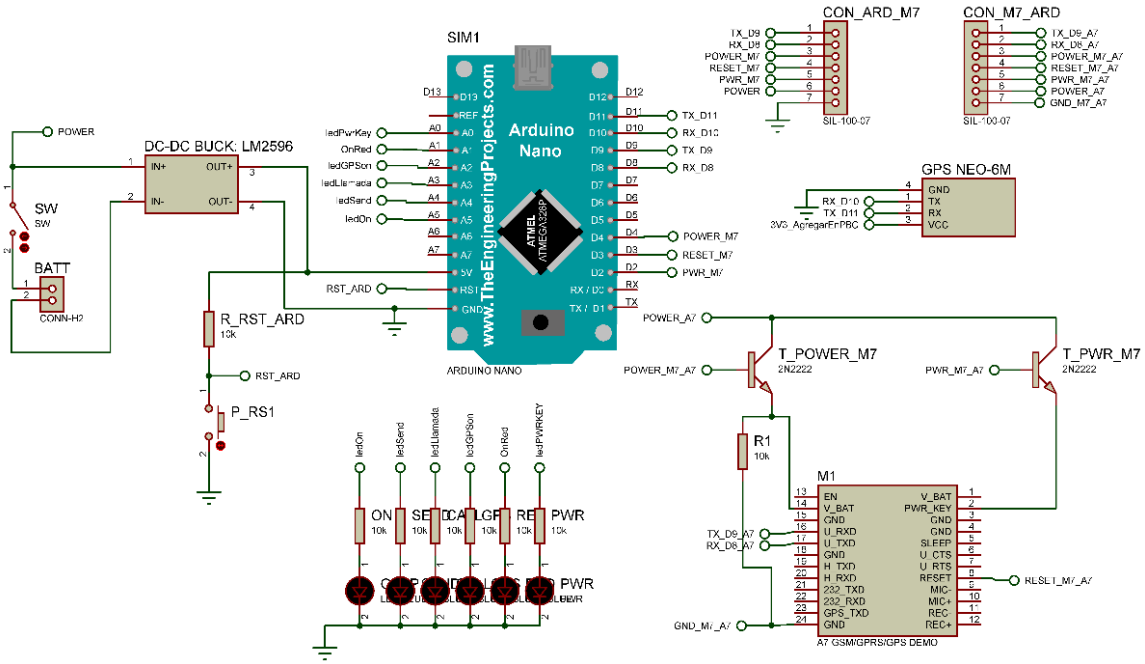


Figura 3.24 Diagrama de circuito implementado

Los leds son indicadores de las acciones que el circuito realiza como el envió de datos, la recepción de llamadas, la activación de GPS, entre otras. En la siguiente lista se detalla que indica cada led.

A. **ON:** El circuito este alimentado a la batería y esta energizado el circuito.
B. ***SEND*:** Se ha enviado el mensaje de datos a través de la red.
C. ***CALL*:** El dispositivo está recibiendo una llamada.
D. **GPS:** Se ha activado el GPS y está recogiendo valores de posición.
E. **RED:** Indicador que está conectado el módulo A7 a la red GSM.
F. **PWR:** Señal de pulso enviado para activar al módulo A7.

Diseño del PCB

Para el diseño del PCB del prototipo de geolocalización, se utilizó el programa Proteus 8. En la Figura 3.25, se muestra el diseño que se implementó para la montura de los componentes electrónicos sobre la placa PCB.

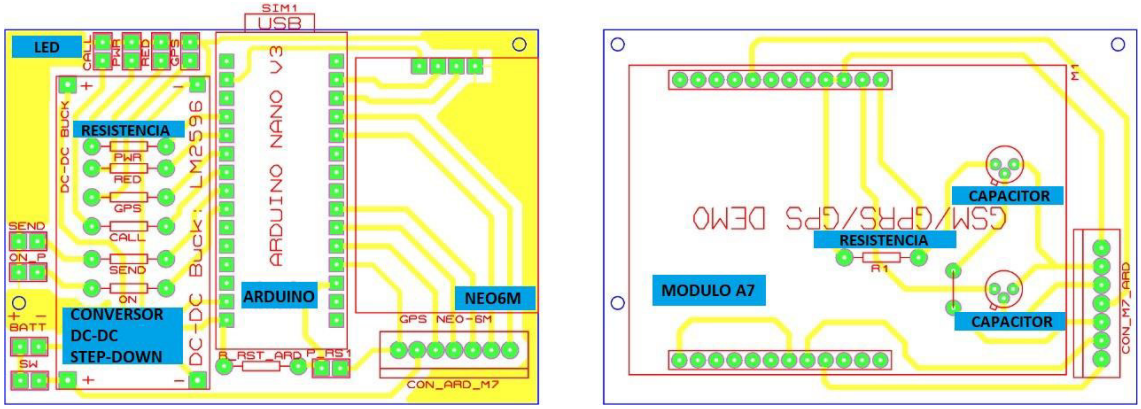
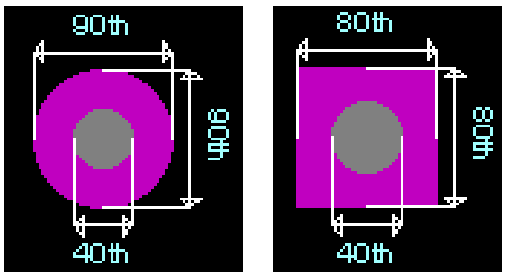


Figura 3.25 Diagrama de la PCB

Para el diseño de la placa PCB, se tiene utiliza *pads* circulares y cuadrados. Para los *pads* circulares de un tamaño de C-90th-40th es decir de 2.25mm y 1mm, mientras que para los *pads* cuadrados es de C-80th-40th o 2mm y 1mm, esto se indica en la Figura 3.26. Las pistas que son las que conectan cada elemento tienen un valor de T25.



C-90-40 **C-80-40**

Figura 3.26 *Pads* circulares y cuadrados

En la Figura 3.27 se muestra el diseño de la placa PCB, en donde se van a montar los componentes electrónicos y módulos necesarios para el desarrollo del prototipo. Y en la Figura 3.28 se muestran todos los componentes electrónicos y módulos montados en la placa.

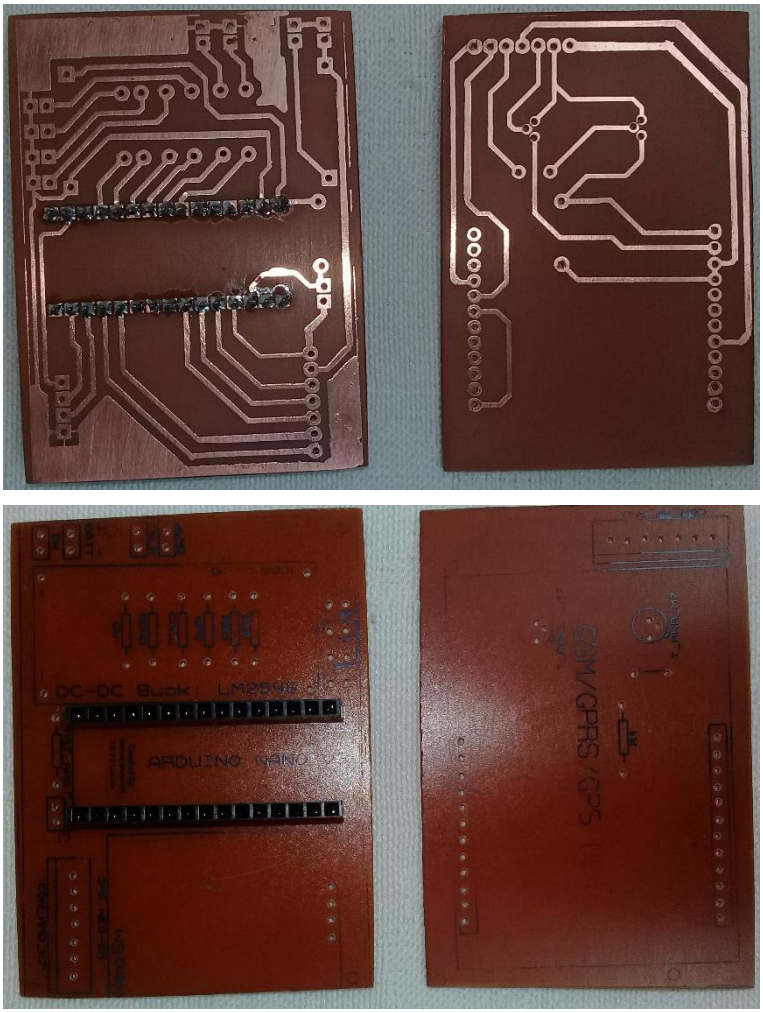


Figura 3.27 Placa PCB del prototipo

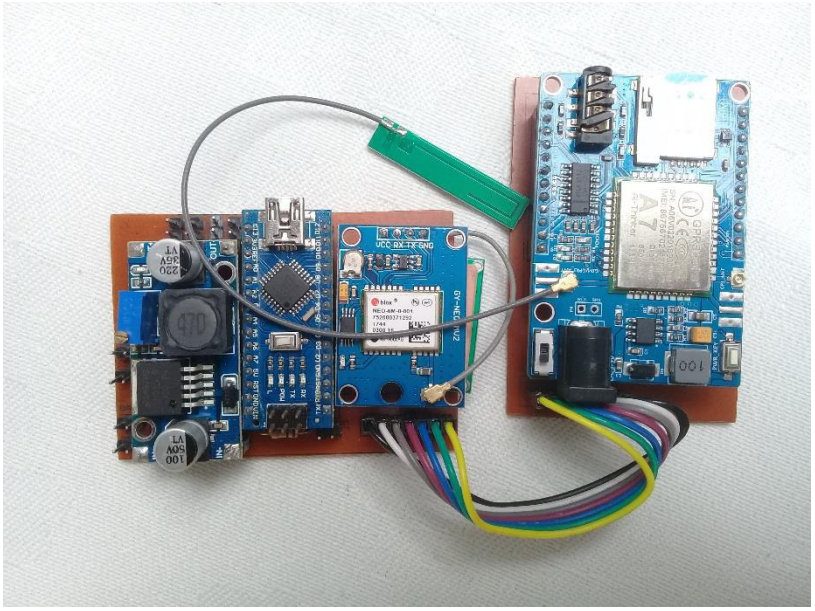


Figura 3.28 Placa PCB con componentes eléctricos montados

Construcción de la carcasa para el prototipo

Para la construcción de la carcasa del prototipo de geolocalización, se debe considerar el tamaño, para evitar robos debe ser lo más pequeña posible. En este caso las dimensiones de la carcasa no son tan reducidas debido a varios factores como el tamaño de batería y la utilización de módulos los cuales no son muy pequeños, sin embargo, se ha logrado tener un tamaño compacto.

En la Figura 3.29 se muestra la carcasa que tiene el prototipo, las dimensiones son altura 7.5 cm, ancho 6.4 cm y largo 9.4 cm. En la parte frontal del dispositivo se cuenta con los seis leds indicadores.

Figura 3.29 Carcasa del prototipo de geolocalizacion

En la Figura 3.30 se muestra en la parte inferior de la carcasa del prototipo un interruptor que es utilizado para encender y apagar el prototipo, y tambien se tiene un pulsador que permite reinicar el dispositivo.

Figura 3.30 Botones del prototipo de geolocalización

3.4 Desarrollo del *software* para el usuario

Recepción de datos entre el servidor *ThingSpeak* y el servidor web

Los datos de posición se encuentran almacenados en el servidor *ThingSpeak* pero se deben recoger dichos valores para mostrarse en la página web. Para esto, se utiliza la función de AJAX de la librería jQuery, AJAX permite ingresar un *array* asociativo, es decir, asignar un valor con una palabra clave. Para la conexión en este proyecto se utilizan las opciones mostradas en la Tabla 3.9.

Tabla 3.9 Opciones de AJAX de jQuery [23]

Opciones	Descripción
Type	Indica el tipo de petición que se va a realizar. Los valores que soporta son *GET* y *POST*.
url	Ingreso del URL del servidor que se va a realizar la petición.

Para la consulta en la opción "*Type*" se va a utilizar el valor "*GET*", mientras que en la opción "url" se ingresa el canal de lectura del API para realizar la consulta del canal, como se aprecia en Figura 3.31.

```
$.ajax({
    type: 'GET',
    url: 'https://api.thingspeak.com/channels/1073852/feeds.json?results=2',
```

Figura 3.31 Código de AJAX con las opciones *type* y url

Los valores a receptar se guardan en un parámetro de una función de AJAX, esta función se llama "*success*". La función "*success*" es llamada si la consulta se ha realizado correctamente [42]. La función recibe tres argumentos:

1. Datos devueltos desde el servidor.
2. Datos formateados de acuerdo a los parámetros *dataType*.
3. Función de devolución de llamada *dataFilter*.

Una cadena que describe el estado. La sintaxis de la función "*success*" es la siguiente:

Function (*Anything* dato, *String textStatus*, jqXHR jqXHR)

Como solo se requiere receptar los datos desde el servidor *ThingSpeak* se utiliza el primer argumento "*Anything* dato".

En el primer argumento se va a recibir información en formato JSON, y los valores a recoger serán "*entry_id", "field1"* y *"field2"*, los cuales corresponde a valores de identidad, de latitud y longitud respectivamente. Estos valores se guardan en variables de JavaScript, estas variables son id_pos, latitud y longitud.

En el diagrama de flujo de la Figura 3.32**,** se describe el código utilizado en el servidor para recoger los datos.

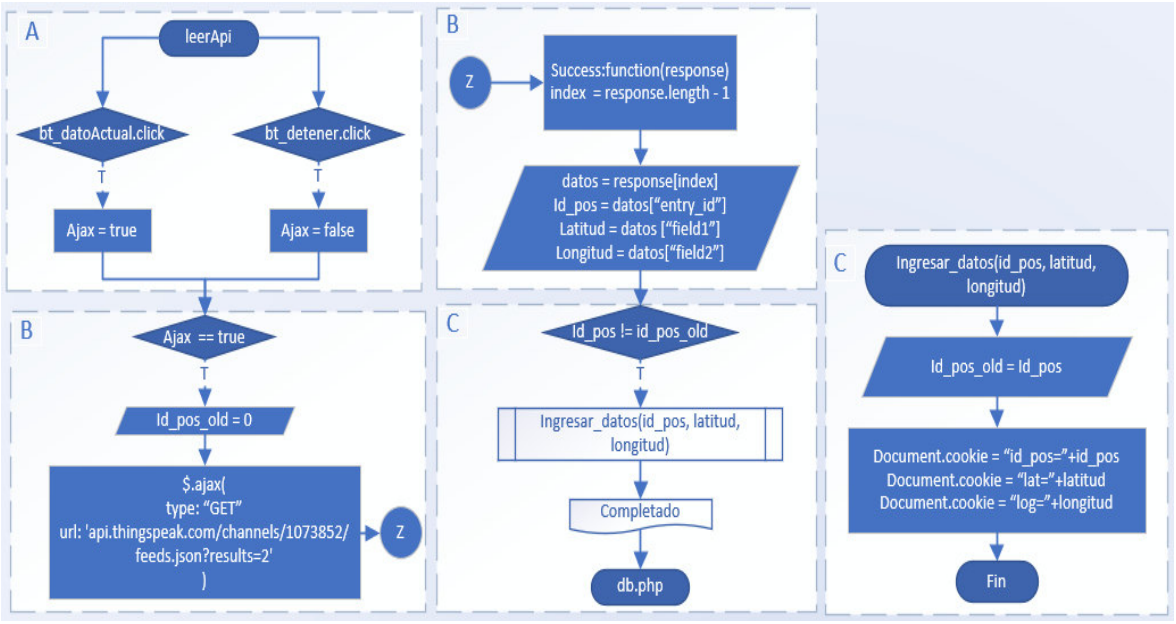


Figura 3.32 Proceso para obtener datos entre el servidor web y *ThingSpeak*

Para entender el diagrama, se va a explicarlo por funcionalidad.

A. Activar o desactivar la obtención de datos del servidor *ThingSpeak*

Se utilizan botones en la página web “Ubicar Bici”, las cuales van a permitir activar la recolección de datos del servidor *ThingSpeak*. La función de “*click*” en JavaScript se utiliza para detectar si ha presionado el botón o no. Al activar el botón “bt_datoActual” la variable “Ajax” es verdadera, pero si activa el botón “bt_detener”, “Ajax” es falso.

B. Recoger datos por método AJAX

Posteriormente se realiza la comparación si la variable “Ajax” es verdadero, en caso de serlo se declara la variable “id_pos_old” con valor 0 (cero), posteriormente se utiliza la función “$.ajax” con las opciones *type* y url, ya explicadas anteriormente.

Luego de utilizar la función “$.ajax”, se utiliza la función “*success*”, la cual va a recibir todos los datos en la variable “*response*”, esta variable contiene datos en formato JSON.

Cada JSON contiene los valores de *entry_id, field1, field2*, etc. Como son varios valores, se debe seleccionar el último valor registrado en el servidor *ThingSpeak*, por tal razón se aplica la propiedad “*length*”, esta permite determinar la longitud de una cadena de valores, en unidades de código UTF-6 [43]. Es decir, va a contar el total de datos que contiene la variable “*response*” y como el último valor es el total se guarda el resultado en la variable “*index*”, esto se muestra en la Figura 3.33.

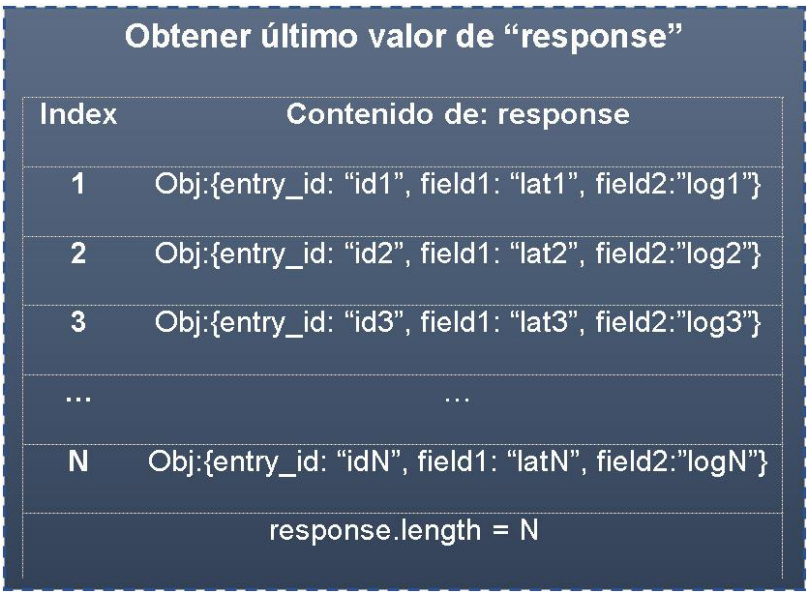


Figura 3.33 Obtención de ultimo valor de response

Cada “*index*” se coloca como *array* en la variable “*response*” y el resultado del último valor registrado en la variable “*response*” se guarda en la variable “datos”. En la variable

"datos" ya se podrá obtener los valores de *"entry_id", "field1"* y *"field2"*, y se guardan en las variables id_pos, Latitud y Longitud respectivamente.

Una vez obtenido "id_pos", se realiza una comparación con "id_pos_old", esta variable tiene la función de evitar que se guarde el mismo valor en la variable "id_pos" de manera repetitiva, esto es importante para evitar repeticiones de valores al almacenar en la base de datos.

C. Guardar datos recogidos por método AJAX

Si "id_pos" y "id_pos_old" son diferentes llamara a la función "ingresar_datos". Esta asigna a la variable "id_pos_old" el valor que contiene "id_pos", posteriormente dentro de la función los datos de id_pos, latitud y longitud se guardan en la *Cookie* con los nombres id_post, lat y log respectivamente, así se puede ingresar a estos valores en diferentes páginas del sitio web. Cuando se termina de ejecutar la función "ingresar_datos" se ejecuta la página db.php, esta tiene la función de almacenar en la base de datos los valores.

En el Anexo 8**,** se puede encontrar el código que permite la recolección de datos en el servidor web.

Desarrollo de página Web

Para el desarrollo de la página web se utiliza el servidor *ByetHost*. Los lenguajes de programación a utilizar son PHP, JavaScript, HTML, CSS, además del uso de base de datos MySQL y la utilización de la librería jQuery.

Características de servidor *ByetHost* para sitio web

Los parámetros que se muestran en la Tabla 3.10, son detalles del servidor que fueron utilizados para realizar el sitio web, entre estos valores se va a encontrar el nombre de dominio, las credenciales para MySQL, versión y características de PHP, etc.

Tabla 3.10 Detalles del servidor [44]

Característica	Valor
DNS (Nombre de Dominio)	proa7gps.*byethost*6.com
DIRECCION IP	185.27.134.3
Puerto MySQL	3306
Servidor API	Apache 2.0 Handler
Versión PHP	7.4.8
Nombre de *host*	sql306.byethost6.com

Estructura de página web

Para el desarrollo de la página web se ha optado por 3 páginas, cada uno va a tener una estructura de diseño similar, pero con funcionalidades y objetivos diferentes, estas páginas se denominan "*Index"*, "Ubicar Bici" y "Como Usar".

Para el diseño de las páginas se utilizaron dos lenguajes HTML y CSS, con HTML se da la estructura es decir se ingresa el encabezado, la barra de menús, el contenido y el pie de página, mientras con CSS se da el diseño o los estilos como el color de fondo y de texto, el formato del texto, la posición de los componentes, etc.

La página *"Index"* es la primera en aparecer al momento que se ingresa el nombre de dominio en la URL del navegador. En esta página se muestra el objetivo y las razones para realizar este proyecto. En la Figura 3.34 se muestra el contenido que tiene la página *"Index"* de la página web.

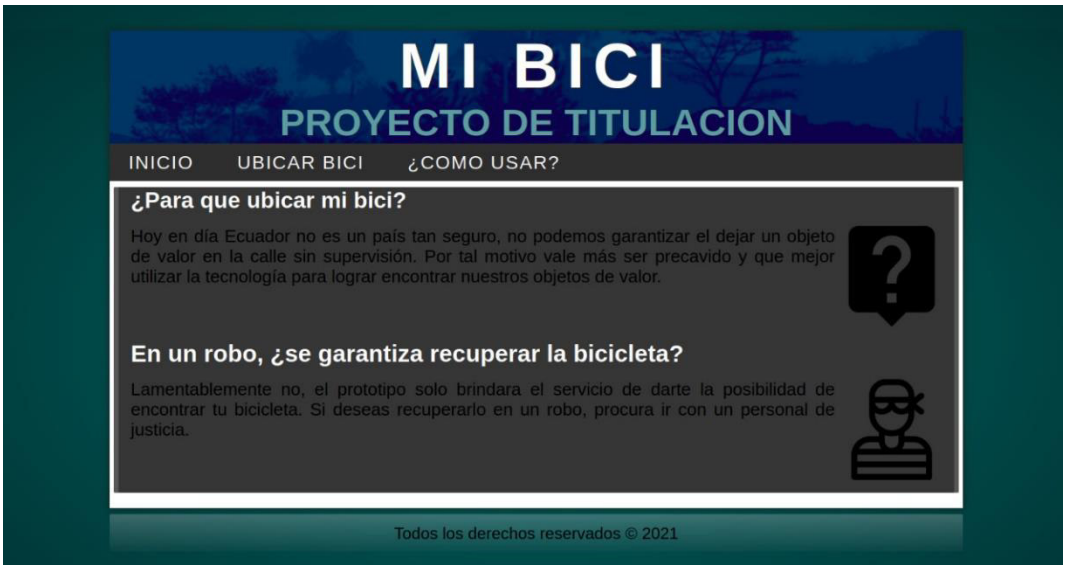


Figura 3.34 Maquetado de la página *"Index"* del sitio web

La página web "Como Usar", mostrada Figura 3.35, está creada para informar al usuario cómo utilizar tanto la página web, el dispositivo de geolocalización y la aplicación móvil, para ello se utilizaron etiquetas con información de cada uno. Para la creación de esta página se utilizaron HTML y CSS, los lenguajes de programación PHP y JavaScript y la librería jQuery.

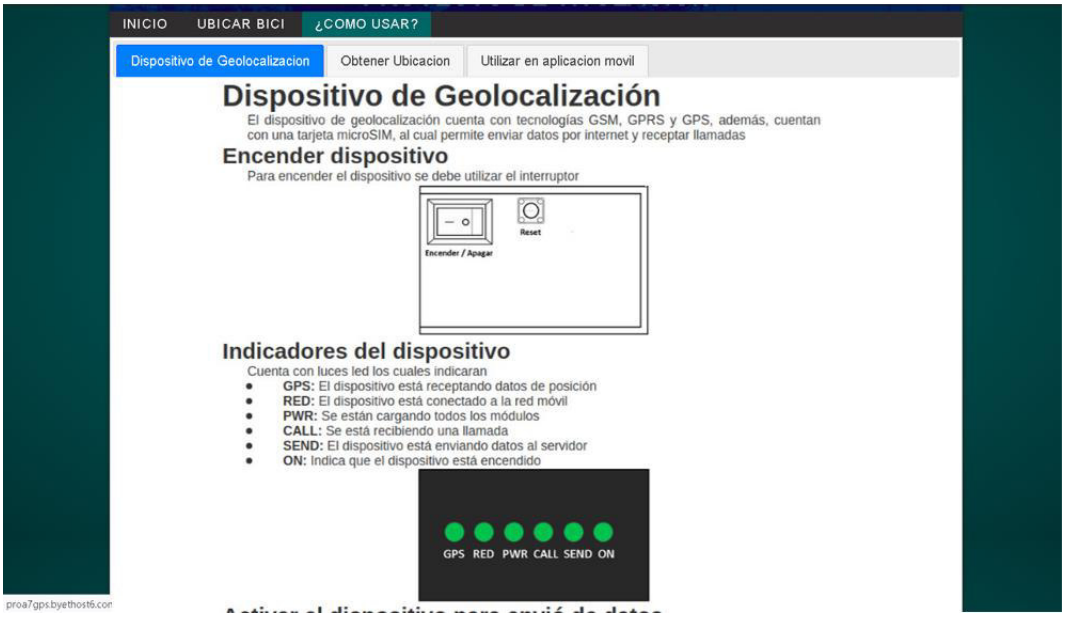


Figura 3.35 Maquetado de la página “Como Usar” de página web

En la página “Ubicar Bici” va a contener con la función de ubicar la bicicleta de manera gráfica, es decir, va a contener un mapa con marcadores de posición, también contiene información y controles que permite aumentar la funcionalidad en la página como son consultas con la base de datos, recepción de datos desde el servidor *ThingSpeak*, mostrar de manera gráfica la posición de bicicletas, etc.

Los lenguajes de programación utilizados son PHP, JavaScript, MySQL y la librería jQuery. En la Figura 3.36 se aprecia la maquetación que tiene la página “Ubicar Bici” de la página web.

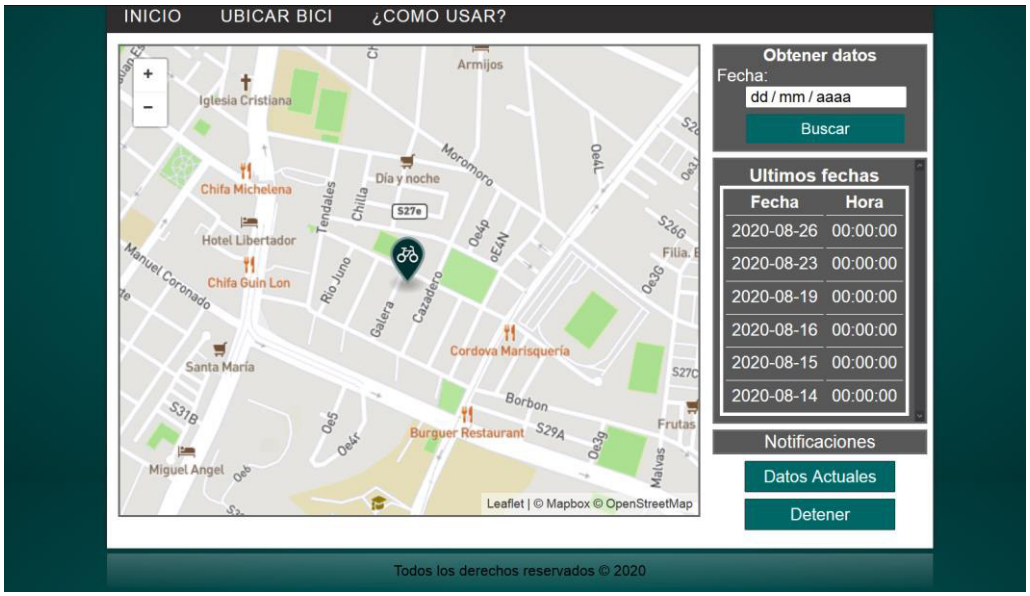


Figura 3.36 Maquetación de la página “Ubicar Bici” de página web

Los procesos de la página “Ubicar Bici” se detallan en la Figura 3.37, ahí se muestran los pasos que se realiza para mostrar los datos obtenidos desde el servidor *ThingSpeak* al servidor web.

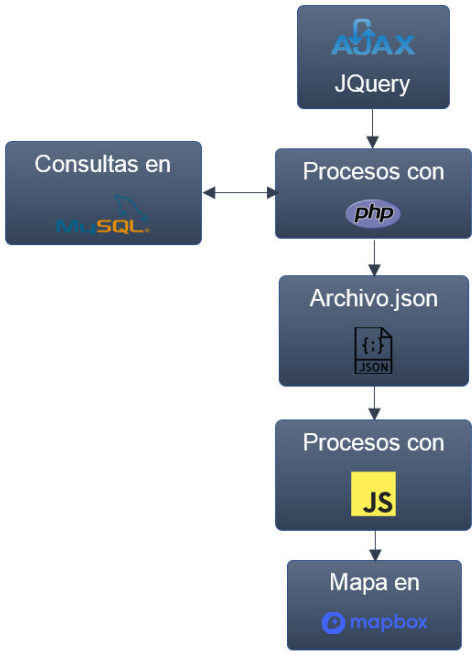


Figura 3.37 Procesos que realiza la página

Antes de explicar de manera detallada cada uno de los procesos en la página, se debe comprender como se llega hasta estos procesos, para ello se utilizan los botones que se ubican en la página “Ubicar Bici”, las acciones que realizan estos botones se indican en las figuras Figura 3.38 y Figura 3.39.

Figura 3.38 Botón “Detener” para no recibir datos del servidor *ThingSpeak*

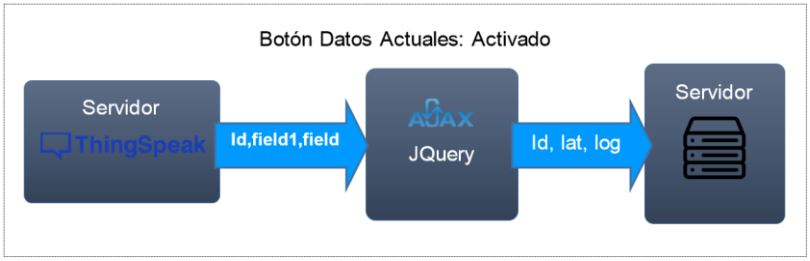


Figura 3.39 Botón de “Datos Actuales” para recibir datos del servidor *ThingSpeak*

El botón de “Datos Actuales” al momento de ser activado recibe los datos que recoge el servidor “*ThingSpeak*” del dispositivo de geolocalización. Estos datos los recepta el servidor web, para procesar los datos en la página web.

Procesos en la página “Ubicar Bici”

El diagrama de flujo mostrado en la Figura 3.37, se observa el proceso realizado para obtener datos de posición, almacenarlo en la base de datos y posteriormente mostrarlo en el mapa.

Para explicar cada proceso del diagrama de flujo, se va a separar por partes para entender la función de cada parte de los procesos.

A. Recoger datos llegados desde AJAX

Luego de obtener los datos que llegan de la función AJAX, ver sección Recepción de datos entre el servidor *ThingSpeak* y el servidor web, se procede a procesarlos.

En la Figura 3.40, se desarrolla el código en PHP, se utiliza la función “db”, con la condición *if* para verificar si los valores que llegan desde la *Cookie*, existen y si no están vacías, si es el caso entonces volverá a la página “UbicarBici” mostrando el mensaje “No existen valores”, pero en el caso de existir estos valores se almacenaran en las variables “id_posicion”, “latitud” y “longitud”, y posteriormente se va a la función “guardar”.

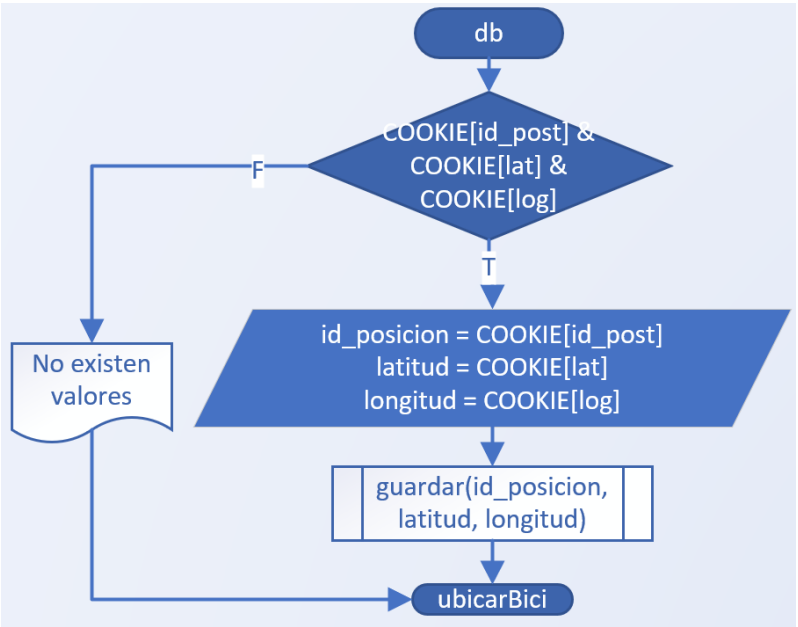


Figura 3.40 Recoger datos desde la página UbicarBici

B. Guardar datos recogidos en la base de datos

En la Figura 3.41, se muestra el diagrama de flujo que explica el funcionamiento de la función “guardar”. Al inicio de la función se borran las *Cookies* que contienen los datos de latitud (lag), longitud (log) y el id de la posición (id_post).

Posteriormente al eliminar las *Cookies*, se establece conexión con la base de datos, para realizar consultas como seleccionar la “id_position” y el resultado se guarda en la variable “res_id_posicion”. En la segunda condición se comprueba que las variables “res_id_posicion” y “id_posicion” sean diferentes, y solo si cumple esta condición se guardan en la base de datos los parámetros enviados en la función que son id_posicion, latitud y longitud.

Si se guardan los valores con éxito se ejecuta la función “ShowInMap”, caso contrario retorna a la página “UbicarBici”.

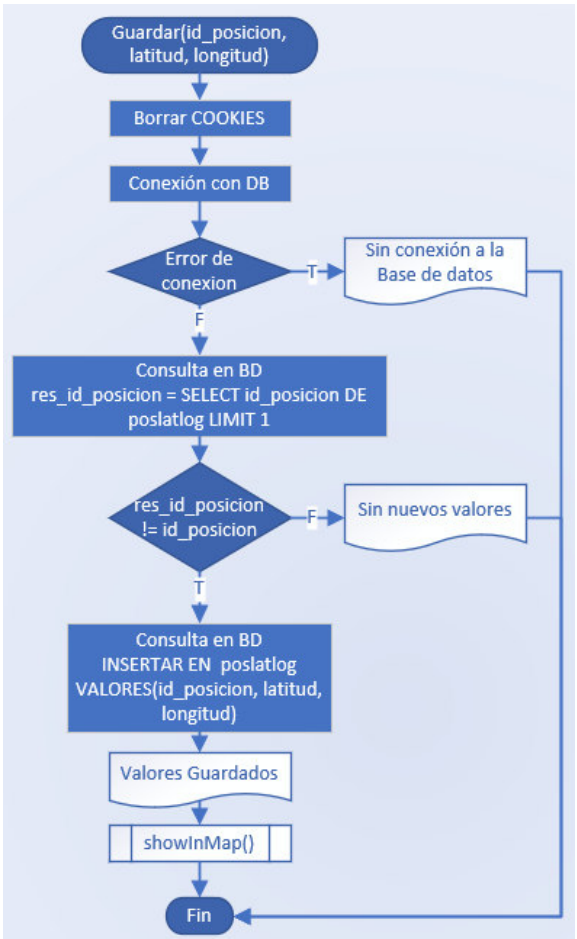


Figura 3.41 Proceso para almacenar datos en la base de datos

En Tabla 3.11 se muestran las consultas a la base de datos realizadas en la página “UbicarBici”.

Tabla 3.11 Consultas a la base de datos en página UbicarBici

Descripción	Consulta en base de datos
Selecciona el ultimo valor del id_posicion	SELECT id_posicion FROM poslatlog ORDER BY id_posicion DESC LIMIT 1
Guarda datos en la base de datos	INSERT INTO poslatlog VALUES (NULL, id_posicion, latitud, longitud, CURDATE(), SYSDATE())

C. Proceso para mostrar valores en mapa

En el diagrama de flujo de la Figura 3.42, la función "showInMap", llama un objeto llamado "mostrarMarkerenMap" el cual realiza el proceso de almacenar todos los datos en el archivo JSON, posteriormente vuelve a la página "UbicarBici".

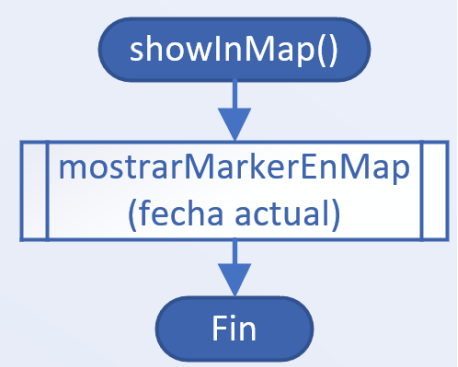


Figura 3.42 Proceso para mostrar mapa

En el Anexo 9, se aprecia el código de los procesos en PHP en la página.

Archivo JSON

En esta sección se va a detallar como los valores almacenados en la base de datos a partir de procesos en PHP, se van a escribir en un archivo JSON. Esta acción se realiza debido a que el archivo JSON funciona como un intermedio entre los procesos de PHP y los procesos en JavaScript.

En el diagrama de flujo de la Figura 3.43, se utiliza el método "mostrarMarkerenMap" el cual contiene un parámetro. El parámetro a pasar es la fecha la cual se utiliza para realizar una consulta en la base de datos, y los valores que devuelve esa consulta son todos los datos de latitud, longitud, fecha y hora, estos valores se van a almacenar en un *array* llamado "posicionJSON". Posteriormente con una condición "*if* en php" se comprueba si existe el directorio "*text*", si no existe, entonces se crea el directorio y

posteriormente se crear el archivo “valores.json”, una vez creada se escribe en este archivo “valores.json” el contenido del *array* “posicionJSON”.

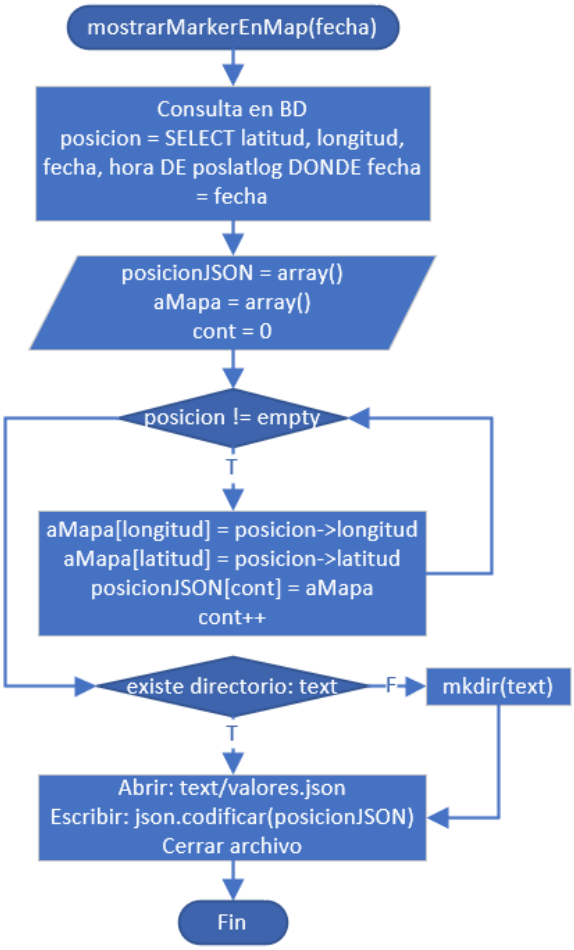


Figura 3.43 Proceso para escribir en el archivo JSON

Los valores de latitud y longitud ya se encuentran guardados en el archivo “valores.json” ahorra se van a procesar con JavaScript para recoger estos datos y mostrarlos en el mapa.

La consulta realizada a la base de datos utilizada en el proceso para escribir en el archivo JSON se muestra en la Tabla 3.12.

Tabla 3.12 Consulta en base de datos para procesos de escribir en archivo JSON

Descripción	Consulta en base de datos
Seleccionar valores por fecha	SELECT latitud, longitud, fecha, hora FROM poslatlog WHERE fecha LIKE= “$fecha”

El código de archivo JSON se encuentra en el Anexo 10.

Procesos con JavaScript

Una vez que los valores de latitud y longitud se encuentran en el archivo JSON de la sección anterior, se va a recoger estos valores por medio del lenguaje de programación JavaScript, esto es debido a que el mapa "*mapbox*" se maneja por este lenguaje, además, de ser más sencillo de manejar las variables con JSON.

Antes de recoger los valores en el archivo JSON se va a realizar una consulta en la base de datos para ello se utiliza el lenguaje de programación PHP, esta consulta es para recoger el ultimo valor de latitud y longitud almacenado, estos datos son para representar la vista central que tiene el mapa.

En la Figura 3.44, el diagrama de flujo muestra los procesos seguidos para obtener los valores que contiene el archivo JSON, primero se realiza la consulta en la base de datos y se los guarda en las variables latitud y longitud, a su vez estos valores se guardan en la variable "vistaCentral" en formato JSON, posteriormente, se procede a obtener los valores en el archivo JSON de la sección anterior, que se almacenan en la variable "*value*". Por medio de un bucle "*while* de JavaScript" se obtienen todos los valores y se almacenan en otro JSON que es destinado a los marcadores en el mapa, almacenados en un *array* llamado "geol". Finalmente, solo si la variable "geol" no está vacía, se llama a la función "initMap" con los parámetros destinados a la vista central y a los marcadores de ubicación, las variables que se mandan como parámetros son "vistaCentral" y "geol".

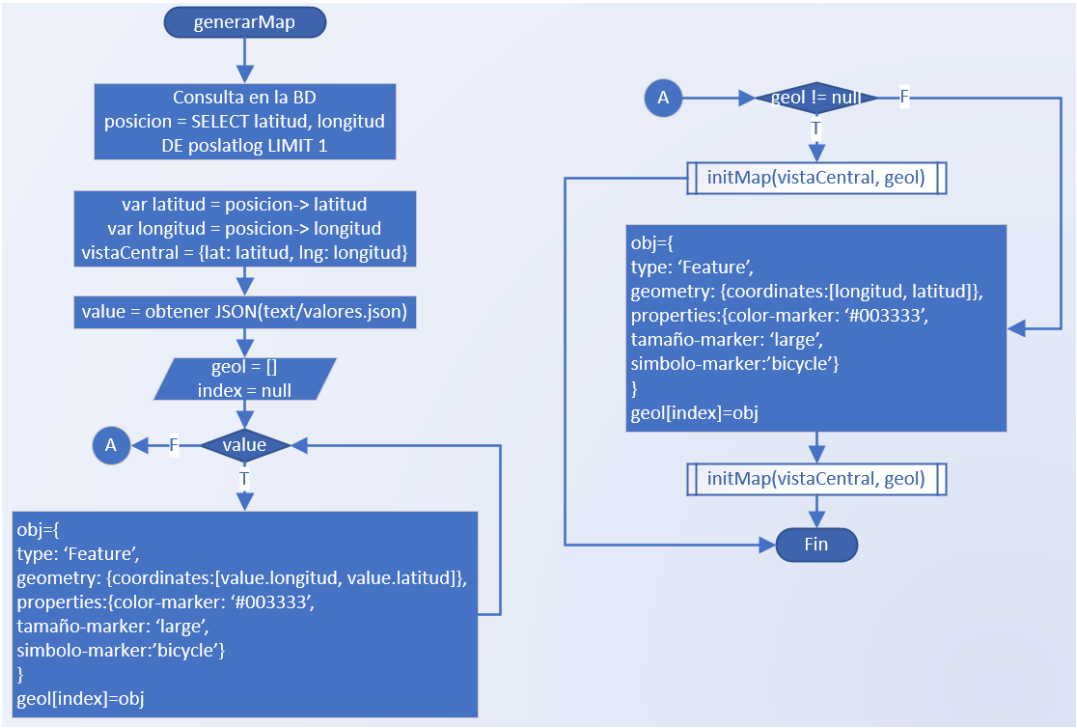


Figura 3.44 Proceso para mostrar datos de posición en el mapa

La consulta realizada a la base de datos se muestra en Tabla 3.13.

Tabla 3.13 Consulta en base de datos para mostrar últimos datos de posición en el mapa

Descripción	Consulta en base de datos
Seleccionar últimos valores de latitud y longitud	SELECT latitud, longitud FROM poslatlog ORDER BY id_posicion DESC LIMIT 1

El código de procesos con JavaScript se encuentra en el Anexo 11.

Mapa con mapbox

El mapa a utilizar es mapbox y se lo emplea porque no requiere de ningún pago inicial como otros mapas, y es de libre uso. En esta sección se describe lo que contiene la función "initMap" que tiene el objetivo de mostrar el mapa con los marcadores.

En el diagrama de flujo de la Figura 3.45, los parámetros vistaCentral y geojson son utilizados para colocar la vista central del mapa y los marcadores respectivamente. Primero se llama al mapa mapbox para poder utilizar sus variables, uno de ellos es "*setView*()" el cual permite colocar la vista central y la otra es "*setGeoJSON*()" que permite agregar marcadores.

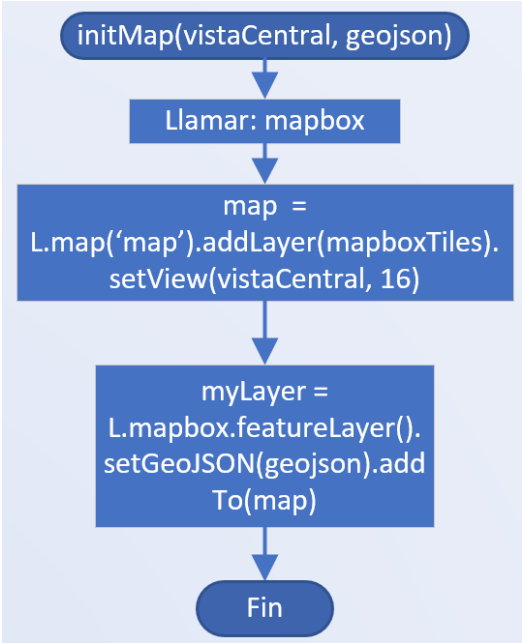


Figura 3.45 Proceso para mostrar mapa con marcadores en mapbox

El código de mapa en mapbox se encuentra en el Anexo 12.

Base de datos

La base de datos se encarga de almacenar todos los datos que llegan del servidor *ThingSpeak*, además, a cada valor que recibe se lo registra con la hora y fecha de llegada de los datos. Esta base de datos es muy importante ya que permite realizar consultas en la página "UbicarBici", como son el historial del uso y consultar los marcadores de la bicicleta por la fecha**.**

Para la creación y administración de la base de datos se utiliza el *software* MySQL y phpMyAdmin.

Creación de base de datos y tabla

En la Tabla 3.14, se muestra la sentencia utilizada para la creación de la base de datos y la creación de la misma con el nombre de "bd_pbici".

Tabla 3.14 Sentencia para crear base de datos [28]

Descripción	Consulta en base de datos
Crear un directorio sobre el directorio de datos MySQL	CREATE DATABASE nombre
Creación de la base de datos para el proyecto	CREATE DATABASE bd_pbici

La creación de las tablas dentro de la base de datos, se debe declarar tanto el nombre de la tabla, los datos y atributos, y el motor de almacenamiento [28]. En la Tabla 3.15 se detallan las sentencias utilizadas para la creación de tablas en la base de datos.

Tabla 3.15 Sentencias para crear tabla [28]

Descripción	Consulta en base de datos
Creará una tabla con el nombre específico de la tabla, sobre la base de datos.	CREATE TABLE tbl_name

En la Tabla 3.16, se muestra los datos y atributos utilizados para la creación de la tabla**.**

Tabla 3.16 Datos y atributos para las tablas [28]

Descripción	Consulta en base de datos
Aplicado solo a los tipos “integer” y “floating-point”.	**AUTO_INCREMENT**
Si no se especifica si es NULL o NOT NULL, la columna se especificará como NULL, este parámetro tiene la función de determinar si una columna puede o no estar vacío [28].	**NOT NULL**
El rango soportado es “1000-01-01” a “9999-12-31”. MySQL muestra los valores de DATE (fecha) en el formato “YYYY-MM-DD” [28].	**DATA**
El rango es “-838:59:59.000000” a “838:59:59.000000”. MySQL muestro los valores de TIME (tiempo) en el formato “hh:mm:ss” [28].	**TIME**
Es del tipo entero (integer), su almacenamiento es de 4 bytes, su valor mínimo es -2147483648 y el valor máximo es 2147483647, mientras que su valor máximo sin signos es 4294967295 [28].	**INT**
Significa que los valores pueden ser almacenados hasta M dígitos, de los cuales D dígitos pueden estar después del punto decimal [28]. Ejemplo FLOAT (7,4) se mostrará - 999.9999.	**FLOAT (M,D)**

Para la tabla del proyecto se va a tomar el nombre de “poslatlog”, además, los valores que contiene la tabla son el id, id_posicion, latitud, longitud, fecha y hora como se muestra en la Tabla 3.17.

Tabla 3.17 Creación de la tabla para el prototipo [28]

Descripción	Consulta en base de datos
Creación de la tabla "poslatlog"	CREATE TABLE poslatlog(id int (255) auto_increment not null, id_posicion int (255) not null, latitud float (100,8), longitud float (100,8), fecha date not null, hora time not null, constraint pk_poslatlog PRIMARY KEY (id),)ENGINE=InnoDB;

En la Figura 3.46 por medio de phpMyAdmin se puede ver todas las configuraciones realizadas anteriormente. Con este *software* se puede ingresar todas las sentencias para crear la base de datos y la tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**id**	int(255)			No	*Ninguna*		AUTO_INCREMENT
2	**id_posicion**	int(255)			No	*Ninguna*		
3	**latitud**	float(100,8)			Sí	*NULL*		
4	**longitud**	float(100,8)			Sí	*NULL*		
5	**fecha**	date			No	*Ninguna*		
6	**hora**	time			No	*Ninguna*		

Figura 3.46 Tabla de MySQL vista en phpMyAdmin

Desarrollo de aplicación Móvil

Para el desarrollo de la aplicación móvil se utiliza la aplicación Android Studio, que va de la mano con el lenguaje de programación java, el cual permite un desarrollo para dispositivos móviles que manejan el sistema Android.

El diseño de la aplicación móvil tiene el objetivo de realizar la llamada para que el dispositivo de geolocalización empiece a entregar valores de posición para posteriormente, la página web los recoja y muestre los valores en el mapa. Finalmente, la aplicación móvil accede a la página web para mostrarlo en la aplicación móvil. Todo este proceso se detalla en la Figura 3.47.

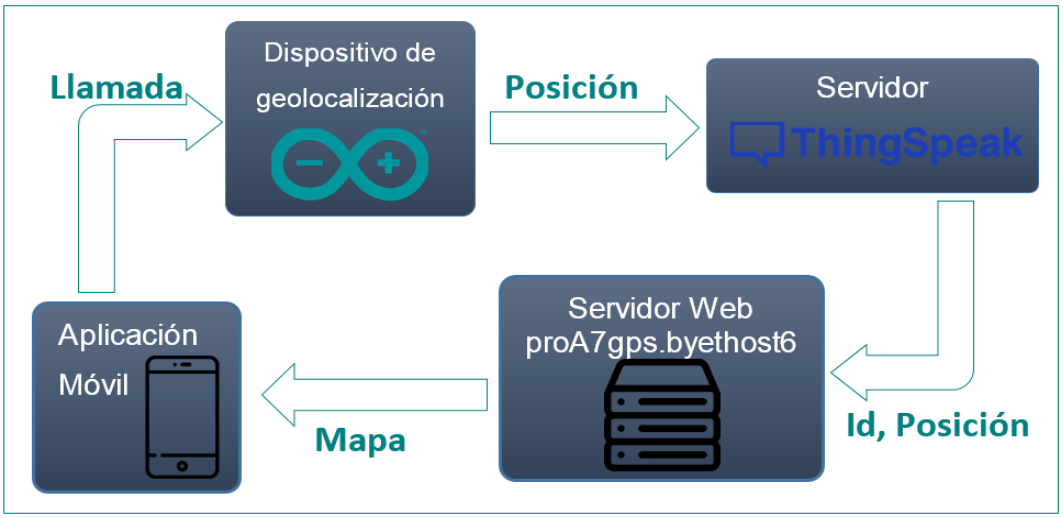


Figura 3.47 Descripción de todo el sistema del prototipo de geolocalización

Al realizar una llamada, los datos de posición se empiezan a mostrarse en el mapa de la página web.

Para el diseño de la aplicación móvil se utilizan las “actividades” las cuales van a tener un objetivo en específico, con el fin de lograr llamar al dispositivo de geolocalización y mostrar los valores de posición en el mapa.

Actividades para la aplicación móvil

Para el entorno grafico se utiliza las siguientes actividades, las cuales se pueden clasificar en “Inicio”, “Configuración”, “Información” y “Mostrar mapa”.

La pantalla de la actividad “Inicio” en la Figura 3.48 es la primera página que se muestra cuando la aplicación se ejecuta por primera vez, en esta actividad se puede encontrar tres botones, estos botones permite cambiar de actividad, para realizar una determinada acción.

Figura 3.48 Actividad de “Inicio” de aplicación móvil

Las funciones de los botones son los siguientes:

A. Botón “BUSCAR BICI”

Este botón permite ir a la actividad de “Mostrar mapa” ver Figura 3.49, en el cual llevara a la dirección web “proA7gps.*byethost*6.com”, específicamente a la página de “Ubicar Bici”. Esto con el objetivo de mostrar el mapa con la posición de la bicicleta.

En esta actividad cuenta con un botón en forma de bicicleta, el cual tiene la función de realizar la llamada al dispositivo de geolocalización, esta función es importante para indicar al dispositivo que debe recoger y entregar los valores de posición al servidor *ThingSpeak*.

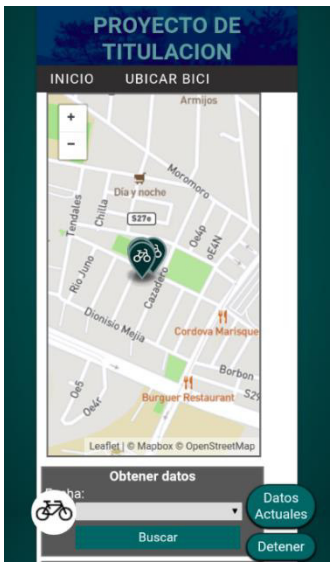


Figura 3.49 Actividad de “Mostrar mapa” de aplicación móvil

B. Botón “INFORMACION”

Este botón al igual que el botón anterior, se redirige a la página “Como Usar” de la página web. Con esta página se puede obtener información del funcionamiento del prototipo de geolocalización, es decir, tanto del uso del dispositivo de geolocalización, el uso de la aplicación móvil y de cómo obtener la ubicación de la bicicleta, Figura 3.50.

Figura 3.50 Actividad de “Información” de aplicación móvil

C. Botón “CAMBIAR NUMERO”

Por último, se tiene el botón de configuración, el cual no redirige a la actividad “Configuración”, ver Figura 3.51, ahí va a permitir cambiar el número telefónico al prototipo a llamar, el numero ingresado en esta actividad, es el número telefónico de la tarjeta micro SIM que tiene el dispositivo de geolocalización.

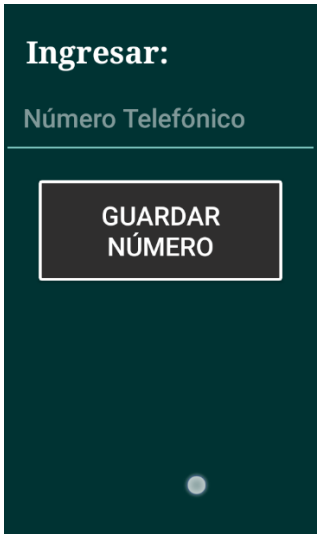


Figura 3.51 Actividad de “Configuración” de aplicación móvil

Descripción general de los procesos de las actividades

En esta sección se va a detallar como las actividades están conectadas entre sí y como se procesan los datos de manera individual para lograr el objetivo de la aplicación. En la Figura 3.52 se especifica cómo se conectan las actividades.

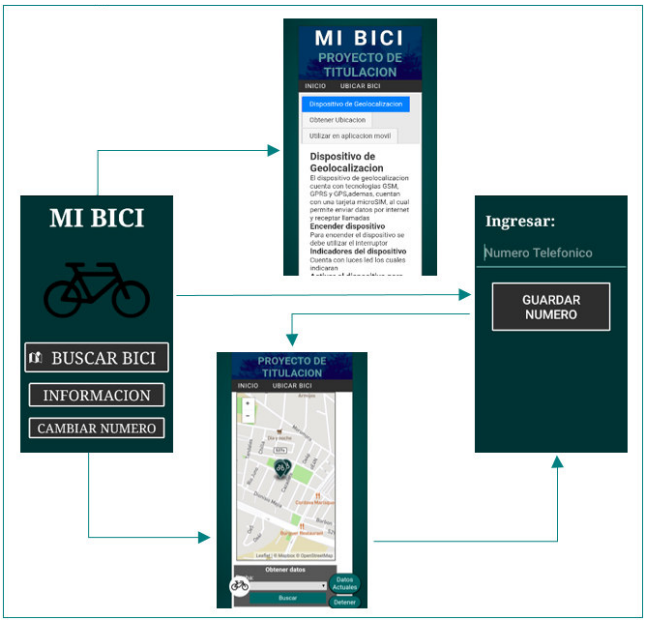


Figura 3.52 Conexión con las actividades

Como se detalló en la sección Actividades para la aplicación móvil, la actividad “Inicio” llama a las otras actividades, pero cuando la actividad de “Mostra mapa” es llamada este a su vez llama a la actividad de “Configuracion”, esto es debido a que el botón de realizar

llamada debe de tener un número telefónico y este valor es almacenado en la actividad "Configuración".

Actividad de "Inicio"

Para los procesos de la pantalla de inicio se cuenta con los botones que permite cambiar de actividad, para lograr esto se utiliza la clase publica *Intent*. Un *Intent* proporciona facilidad para realizar enlaces de ejecución entre el código de diferentes aplicaciones [45]. El uso de *Intent* es el lanzamiento de actividades, esto es utilizado como un enlace entre actividades. Es básicamente una estructura de datos pasivos que contiene una descripción abstracta de una acción a realizar [45].

El proceso que realiza la actividad inicial se muestra en la Figura 3.53.

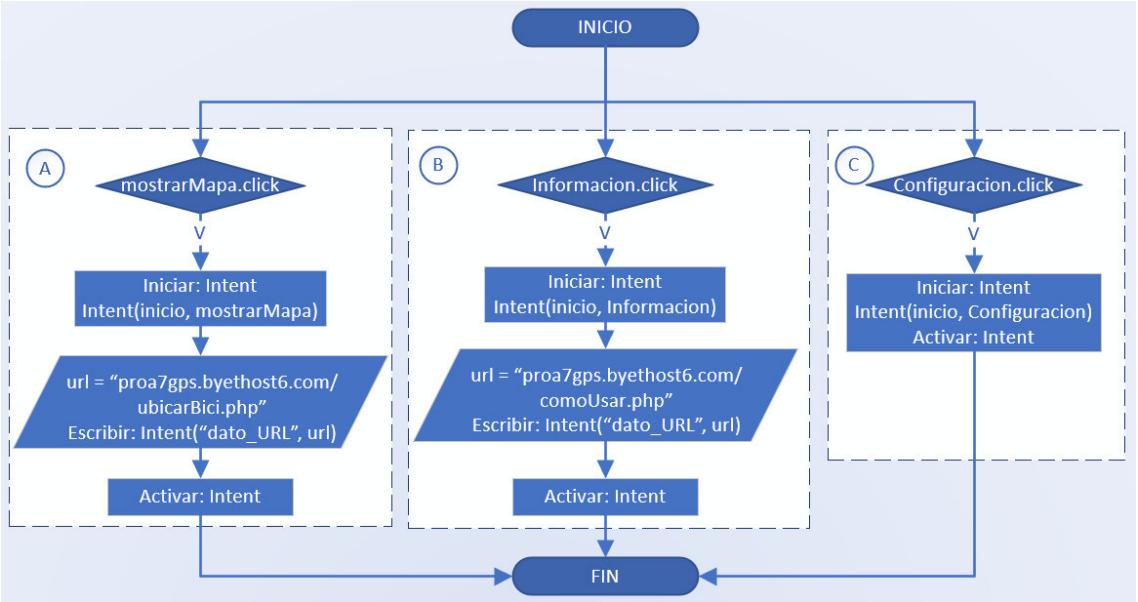


Figura 3.53 Procesos para actividad "Inicio"

Para describir el proceso de la actividad se va a separar por partes:

A. Pasar de actividad "Inicio" a actividad de "Mostrar mapa"

Como se tiene botones para acceder a las diferentes actividades, entonces se va a detectar si han sido presionadas, si se presionó el botón de "BUSCAR BICI" cuya denominación en el código es "mostrarMapa", este va a llamar a un *Intent* el cual describe dos parámetros, el primer parámetro significa la actividad actual y el segundo parámetro significa la actividad a dirigirse, posteriormente se agrega una variable llamada "url" el cual contiene la dirección del sitio web específicamente a la página "Ubicar Bici", esta variables indica a la aplicación móvil que realice una dirección a la página web.

Intent permite ingresar datos para posteriormente ser usado en otra actividad, para ello requiere un nombre como identificador y el valor a enviar, en este caso, el nombre identificador es “dato_URL” y el valor a enviar es la variable “url”.

Finalmente se da a iniciar *Intent*, dicha actividad permite cambiar de la actividad inicial a otra actividad, en este caso se pasará a la actividad de “Mostrar mapa”.

B. Pasar de actividad “Inicio” a actividad de “Información”

Para el caso de presionar el botón de “INFORMACION” cuya denominación en el código es “informacion”, se sigue los mismos procesos detallados anteriormente, la diferencia es en la asignación a la variable llamada “url” que contiene la dirección al sitio web, específicamente a la página “comoUsar” y que va a pasar a la actividad “Información”

C. Pasar de actividad “Inicio” a actividad de “Configuración”

Para el caso de presionar el botón “CAMBIAR NUMERO” cuya denominación en el código es “Configuracion”, utiliza un *Intent* pero a comparación de los otros este no enviara ningún valor, solo se direccionara a la actividad “Configuracion”.

En el Anexo 13, se encuentra el código para la Actividad inicial.

Actividad de “Configuración”

Para la actividad de configuración se utiliza *Intent* para pasar de actividad y también pasar valores a dicha actividad, adicionalmente, en esta actividad se cuenta con un *Text Edit* (Texto Editable) del tipo numérico, este va a permitir recoger el valor que se le ingrese, es decir el número telefónico. Dicho valor en el *Text Edit* no se debe eliminar al momento de cerrarse la aplicación móvil, es por ello que se utiliza la clase *SharedPreferences*.

En el diagrama de flujo de la Figura 3.54, se muestra los procesos que se realizan.

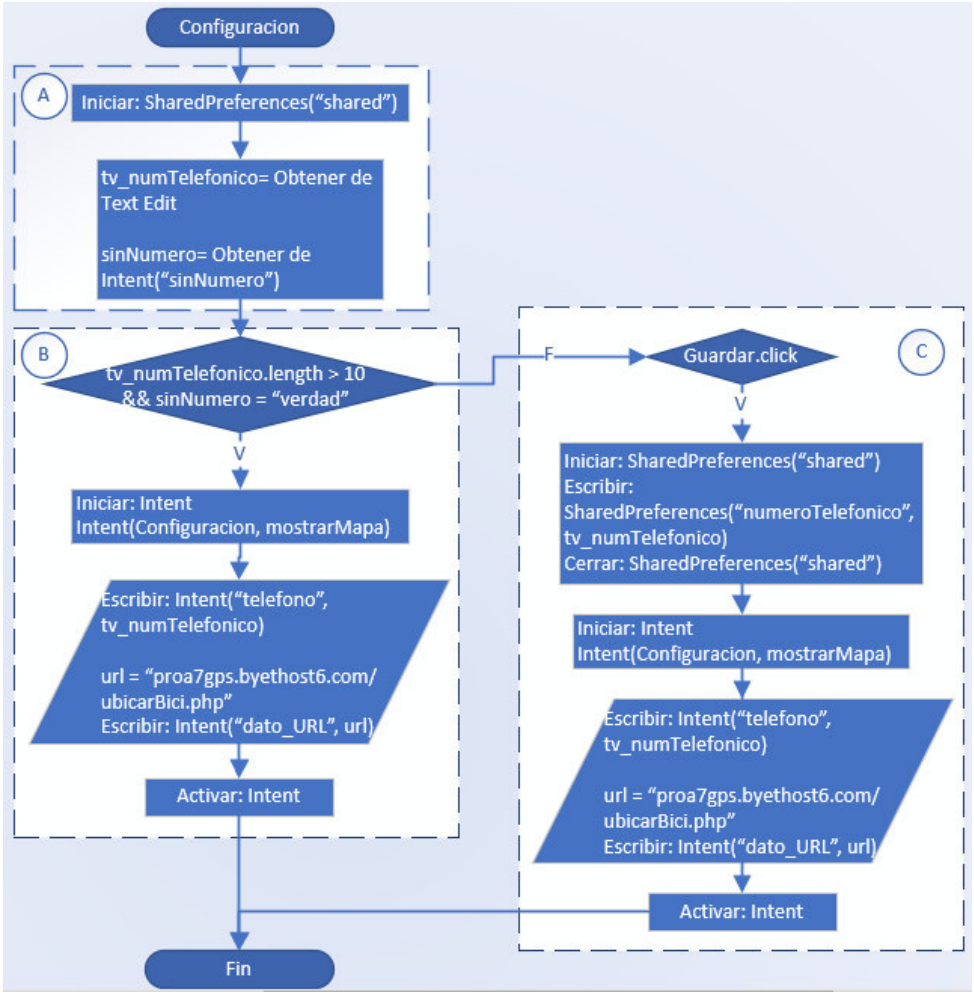


Figura 3.54 Proceso para actividad de "Configuracion"

Para la descripción del diagrama de flujo, se va a separar por partes.

A. Declara variables

Primero se va a iniciar *SharedPrederences*, luego se obtiene los valores que contiene el *Text Edit* y se guarda en la variable llamada "tv_numeroTelefonico", posteriormente, se obtiene valores de *Intent* con el nombre "sinNumero", este valor se asigna en la actividad "Mostrar mapa" y el valor que entrega es "verdad", siempre y cuando en la actividad "Mostrar mapa" no se haya registrado el número telefónico para realizar llamadas, caso contrario el valor que contendrá la variable "SinNumero" será nulo.

B. Primera Comparación

En la primera comparación se compara los variables "tv_numeroTelefonico" si tiene una longitud mayor a diez y si "SinNumero" es igual a "verdadero", esto se realiza para detectar si en la actividad "Mostrar mapa" tiene el número telefónico para realizar la llamada. En la

Tabla **3.18** se muestra la comparación con los posibles resultados.

Tabla 3.18 Comparación en la actividad configurar

tv_numeroTelefonico	SinNumero	Resultado	Descripción
Falso	**Falso**	**Falso**	El usuario se ha dirigido directamente a la actividad "Configuracion" y en el *Text Edit* no hay valores (recordar que la variable tv_numeroTelefonico obtiene su valor de *TextEdit*).
Verdadero	**Falso**	**Falso**	El usuario se ha dirigido directamente a la actividad "Configuracion" y en el *Text Edit* si hay valores.
Falso	**Verdadero**	**Falso**	El usuario se ha dirigido directamente a la actividad "Mostrar mapa" y no ha detectado número telefónico, entonces se redirige a la actividad "Configuracion" para obtener número telefónico, pero en el *Text Edit* no hay valores
Verdadero	**Verdadero**	**Verdadero**	El usuario se ha dirigido directamente a la actividad "Mostrar mapa" y no ha detectado número telefónico, entonces se redirige a la actividad "Configuracion" para obtener número telefónico, y en *Text Edit* si existe valores, por lo que se redirige nuevamente a "Mostrar mapa" y envía el número telefónico por *Intent*

Solo si las dos condiciones son verdaderas se ejecutará la condición, que consiste en iniciar el *Intent* para pasar de la actividad "Configuracion" a "Mostrar mapa" y, además, se agrega un valor en *Intent* llamado "teléfono" cuyo valor es el que contiene "te_numeroTelefonico", es decir, se envía el número telefónico. También se le agrega otro valor en *Intent* llamado "dato_URL" necesario para cargar la actividad "Mostrar mapa" con la página web.

C. Acción del botón Guardar

Si no cumple con la condición y el resultado es falso, entonces la actividad "Configuracion" estará pendiente que el botón Guardar sea presionado.

Cuando es presionado la acción a ejecutar será la misma que la primera condición, es decir, se tomara el número telefónico de la variable “tv_numeroTelefonico” y la URL, se agrega como valores en el *Intent* y se redirige a la actividad “Mostrar mapa”.

En el Anexo 14, se encuentra el código para la Actividad de “Configuración”.

Actividad de “Mostrar mapa”

Para la actividad de mostrar mapa se utiliza la clase “*WebView*” la cual permite mostrar una página web, esto como parte de una aplicación cliente. Esta clase es una extensión de la clase *View* de Android Studio [45].

Adicionalmente como esta actividad va a realizar llamadas por medio del botón, se requiere de los permisos de llamadas, de no contar con ellos no se podrá realizar las llamadas telefónicas.

En la Figura 3.55 se aprecia el diagrama de flujo de la actividad “Mostrar mapa”.

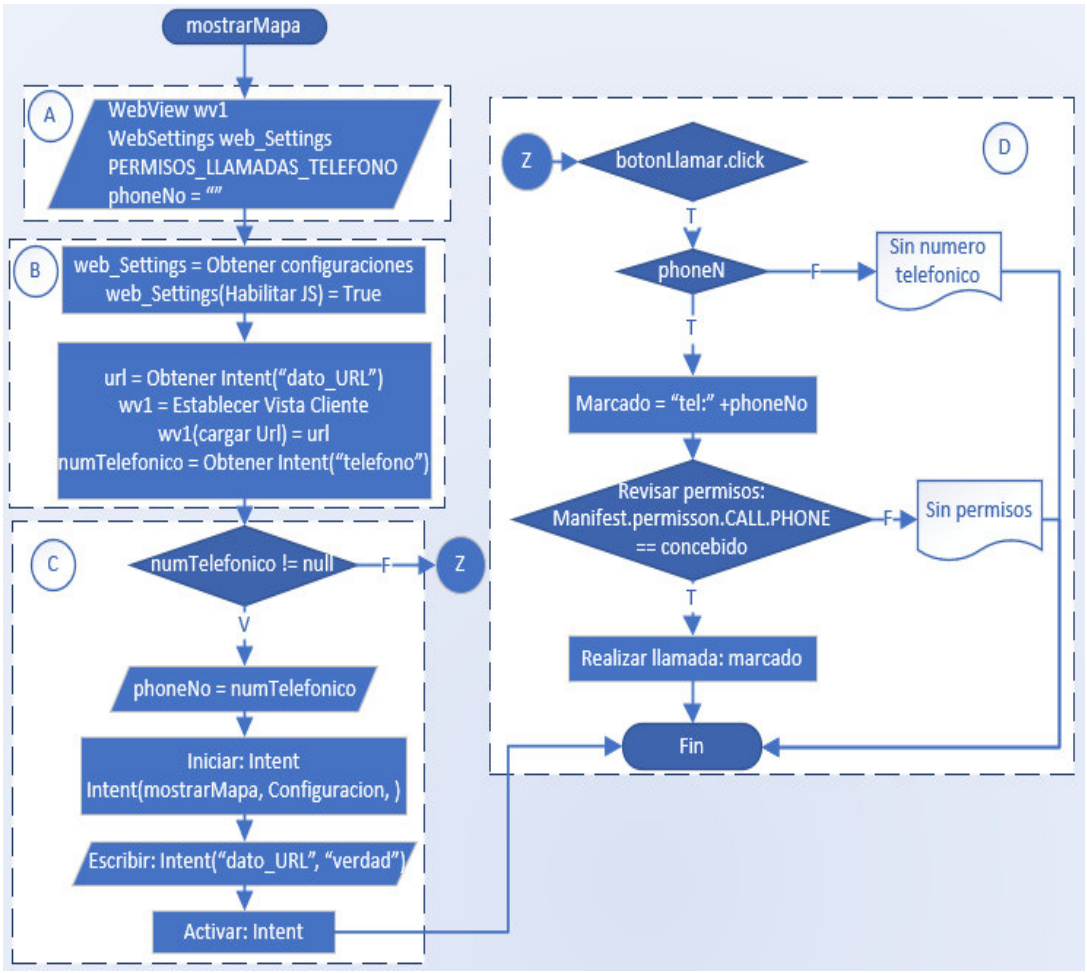


Figura 3.55 Procesos en la actividad “Mostrar mapa”

Los procesos que realiza la actividad se muestran por separado.

A. Declarar variables

Primero asignan dos variables relacionadas a *WebView*, uno es para asignarlo como cliente y el otro para agregar configuraciones.

Posteriormente se asignan los permisos con “PERMISSOS_LLAMADA_TELEFONICA”, otra variable a utilizar es “phoneNo” el cual almacenara el número telefónico de la actividad “Configuracion”.

B. Mostrar página web en la actividad

Primero se muestra la página web de “Ubicar Bici”, para ello se ejecuta las variables *WebView*, a la variable “wv1” se les agrega las configuraciones con “*getSettings*” y el resultado se los guardara en “*web_Settings*” esto se realiza debido a que la página web desarrollada, utiliza JavaScript y para que la página web se pueda ver en la aplicación móvil, se habilita esta función asignando un valor de verdadero.

Posteriormente por medio de la clase *Intent* se va a obtener el valor de la URL, este valor se guarda con el nombre de “dato_URL”. Una vez obteniendo los datos por medio del *Intent*, se va a guardar en la variable “URL”, posteriormente se le asigna a “wv1” como vista cliente, llamando al objeto “*WebViewClient*”, y finalmente cargara la página web con la propiedad “*loadURL*” agregando la variable “URL” que contiene la dirección web de la página.

C. Obtener número Telefónico

Para obtener el número telefónico de la actividad de “Configuracion”, se utiliza la clase *Intent* el cual llama al valor que contiene “teléfono”, es decir, el número telefónico y se lo agrega a la variable “numTelefonico”.

En la comparación si existe el número telefónico en la variable “numTelefonico” entonces se asigna el valor que contiene “numTelefonico” en la variable “phoneNo”. En el caso de no existir el número telefónico, entonces por medio de un *Intent* se cambia de la actividad “Mostrar mapa” a la actividad “Configuracion”, el cual contiene el número telefónico, además, se envía un valor “verdad” en el *Intent* llamado “sinNumero”

D. Activación del botón llamar (forma de bicicleta)

Si el botón de llamar es activado, se crea la variable de “marcado” el cual será el encargado de contener la variable para realizar la llamada, contiene el número telefónico

y una cadena “tel:”. Posteriormente en la comparación se verifica si tiene permisos para realizar la llamada si es verdad entonces utiliza la variable “marcado”, para realizar la llamada.

En el Anexo 15, se encuentra el código de la Actividad de mostrar mapa.

Actividad de “Información”

Igual a la actividad “Mostrar mapa”, utilizara la clase “*WebView*”, el procedimiento es el mismo que en la anterior actividad. Es decir, se crea las variables “wv1” para asignarlo como cliente y la variable “*web_setting*” para agregar configuraciones.

Para obtener la URL se llama al *Intent* y se obtiene la dirección web, esta variable del *Intent* se llama “dato_URL”, y se carga la url en la variable “wv1”.

En el diagrama de flujo de la Figura 3.56, se muestra los procesos que se realiza.

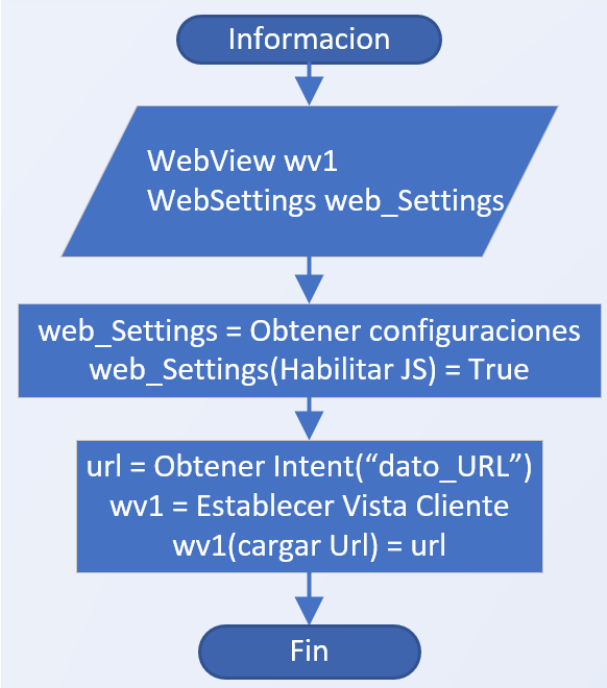


Figura 3.56 Procesos para actividad “Información”

El código para la Actividad de información se encuentra en el Anexo 16.

3.5 Pruebas y Análisis de Resultados

Pruebas de funcionamiento del *hardware*

El *hardware* del prototipo de geolocalización de bicicletas cuenta con luces leds que permite indicar que acción esta realizado el dispositivo. En esta sección se muestra como el dispositivo primero recibe la llamada, luego recoge los valores de GPS y por último los envía al servidor solo mostrando la iluminación de los leds, los pasos son:

1. El prototipo se conecta a la red GSM por ello el led 2 llamado “RED”, debe estar encendido y posteriormente recibe una llamada a la tarjeta micro SIM que tiene incorporado, una vez realizada la llamada, el led 4 llamado “*CALL*” se encenderá, posteriormente el dispositivo cuelga automáticamente la llamada, y se apaga el led 4 y guarda el número telefónico como se aprecia en la Figura 3.57.

Figura 3.57 Prueba del dispositivo encendido del led *Call*

2. Luego de colgar la llamada, activa el GPS para recoger los datos de localización, al mismo tiempo el led 1 que indica ‘GPS’ se enciende, y cuando los datos de latitud y longitud sean almacenados en el dispositivo de geolocalización, desactiva el GPS y apagará el led 1, ver Figura 3.58.

Figura 3.58 Prueba del dispositivo encendido de led GPS

3. Después de obtener los datos de posición, empieza a establecer la conexión con el servidor *ThingSpeak*, del mismo modo el led 5 denominado '*SEND'* se enciende, indicando que se están enviando los datos de localización, que ha registrado el dispositivo, al servidor *ThingSpeak* y se apagará el led cuando se haya terminado de enviar, ver Figura 3.59.

Figura 3.59 Prueba del dispositivo encendido de led *SEND*

Cuando los datos se han enviado al servidor el dispositivo termina la conexión con el servidor *ThingSpeak*, más no la conexión con la red, esto es debido a que se va a seguir enviando datos al servidor.

Todo este se va a repetir hasta recibir nuevamente la llamada del mismo número telefónico que realizo la llamada para activar la geolocalización, o presionando pulsando el botón de reseteo.

Pruebas de funcionamiento de página web

En esta prueba se muestra como las paginas tanto el *"Index"*, "Ubicar Bici" y "Como Usar", se han maquetado y su funcionamiento al momento de visitar cada página.

1. La página *"Index"* se carga cuando se ingresa el nombre de domino, en la URL del navegador, la página contiene la justificación de la creación del proyecto, ver Figura 3.60.

Figura 3.60 Prueba de página *"Index"*

2. En la página "Ubicar Bici", cuando los datos de posición se encuentran almacenados en el servidor *ThingSpeak*, el navegador web va a receptar todos los datos recientes que el servidor reciba, solo si activa el botón llamado "Datos Actuales", ver Figura 3.61.

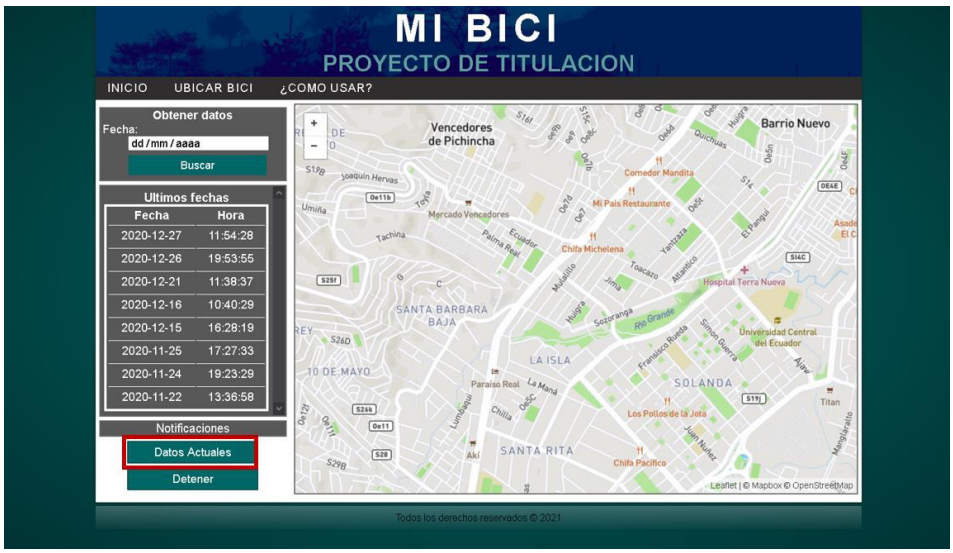


Figura 3.61 Prueba de página "Ubicar Bici" con botón "Datos Actuales"

Cuando se presiona el botón empieza a mostrar en el mapa el marcador con la posición del dispositivo indicando los valores de latitud y longitud de manera gráfica, además, en “Últimas fechas” se mantendrá un registro por fechas y horas de los registros de posición realizados, ver Figura 3.62.

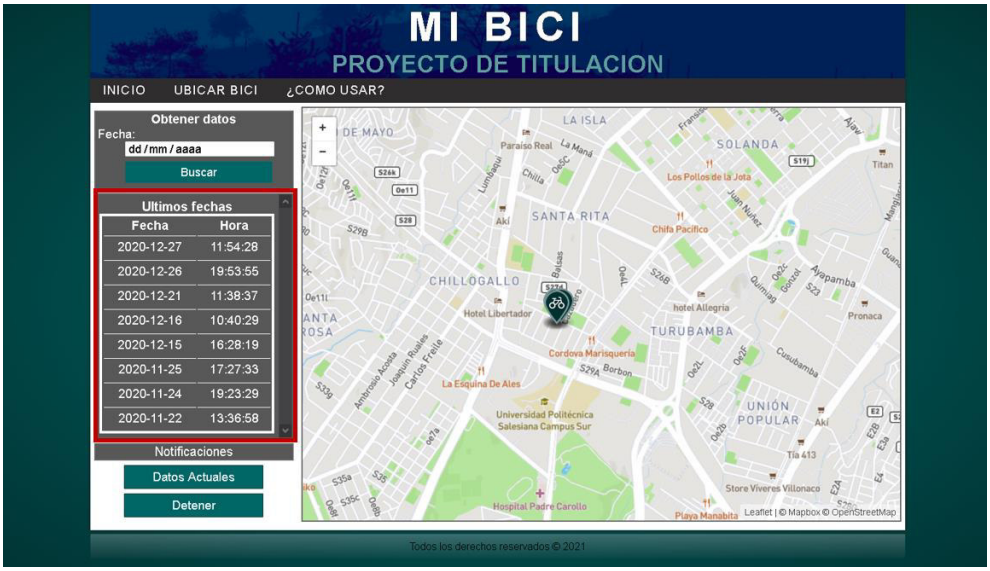


Figura 3.62 Prueba de página “Ubicar Bici” con marcadores de posición

3. En la opción “Obtener datos” de la página “Ubicar Bici”, se puede seleccionar la fecha deseada, en el caso de no tener ningún registro dicha fecha no mostrara ningún marcador de posición.
   Una vez selecciona la fecha se presiona el botón “Buscar”, lo cual muestra en el mapa los marcadores almacenados en dicha fecha, ver Figura 3.63.

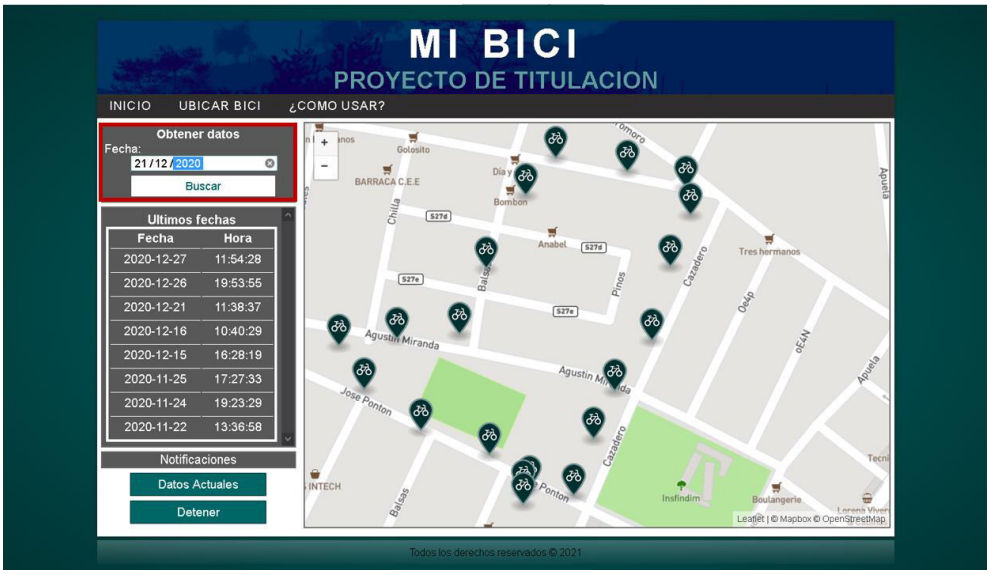


Figura 3.63 Prueba de página “Ubicar Bici” consulta de posición por fecha

4. Finalmente, si desea saber cómo utilizar tanto la página web como el dispositivo de geolocalización, se debe ir a la página “Como Usar”, el cual mostrara información de cómo obtener los datos de geolocalización. Ver Figura 3.64.

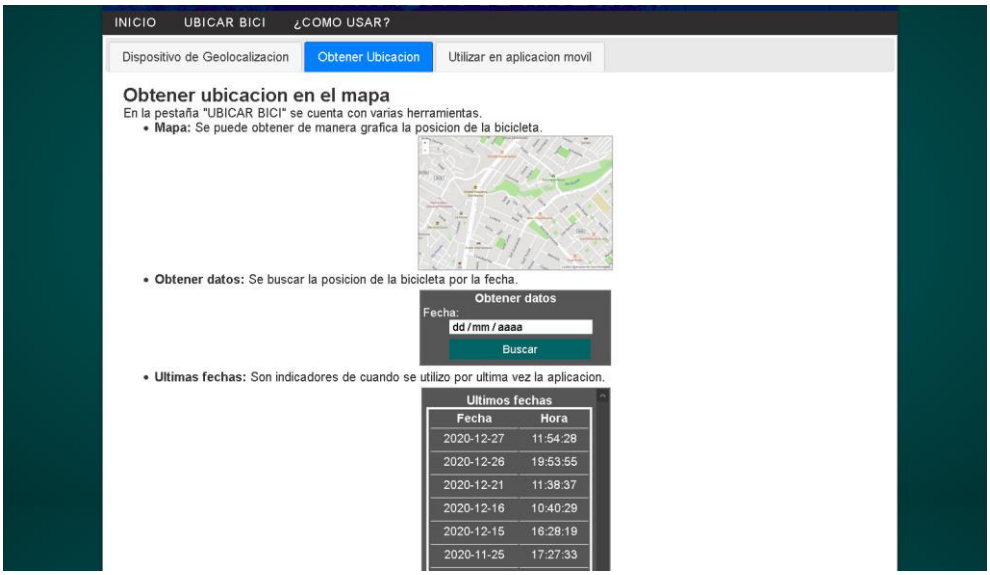


Figura 3.64 Prueba de página “Como Usar”

Pruebas de funcionamiento de aplicación móvil

Para las pruebas del funcionamiento de la aplicación móvil se muestra cómo se utiliza y que acción realiza cada actividad. La aplicación móvil puede indicar al dispositivo de geolocalización cuando debe enviar los datos de posición al realizar una llamada, para posteriormente mostrar los datos por el mapa. Las pruebas realizadas son las siguientes:

1. Cuando se inicia la aplicación móvil se carga la actividad “Inicio”, las cuales cuenta con los botones de “BUSCAR BICI”, “INFORMACION” y “CAMBIAR NUMERO”, ver Figura 3.65.

Figura 3.65 Prueba con actividad “Inicio”

2. Para mostrar la ubicación del dispositivo en la bicicleta se va a la opción “BUSCAR BICI” y en vez de dirigirse a la actividad “Mostrar mapa” se va a dirigir a la actividad “Configuracion” para pedir el número telefónico de la tarjeta micro SIM del dispositivo de geolocalización para realizar la llamada y así iniciar la entrega de datos de posición, ver la Figura 3.66.

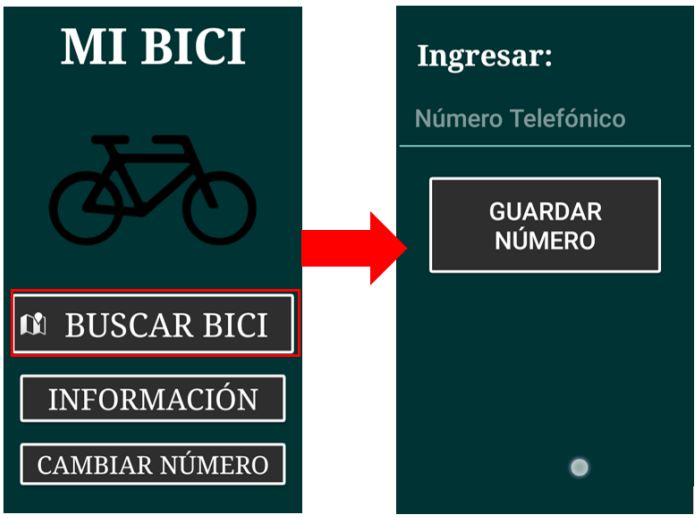


Figura 3.66 Prueba en actividad “Inicio” al precionar botón buscar bici por primera vez

3. Luego de ingresar el número telefónico pasara de la actividad “Configuracion” a la actividad “Mostrar mapa”, esta actividad se direcciona al sitio web, específicamente a la página de “Ubicar Bici” el cual muestra el mapa de la página web “Ubicar bici”, ver Figura 3.67. Para que empiece a mostrar la posición del

dispositivo se debe de realizar la llamada telefónica por medio del botón en forma de bicicleta y posteriormente empezara a llamar al dispositivo Figura 3.68.

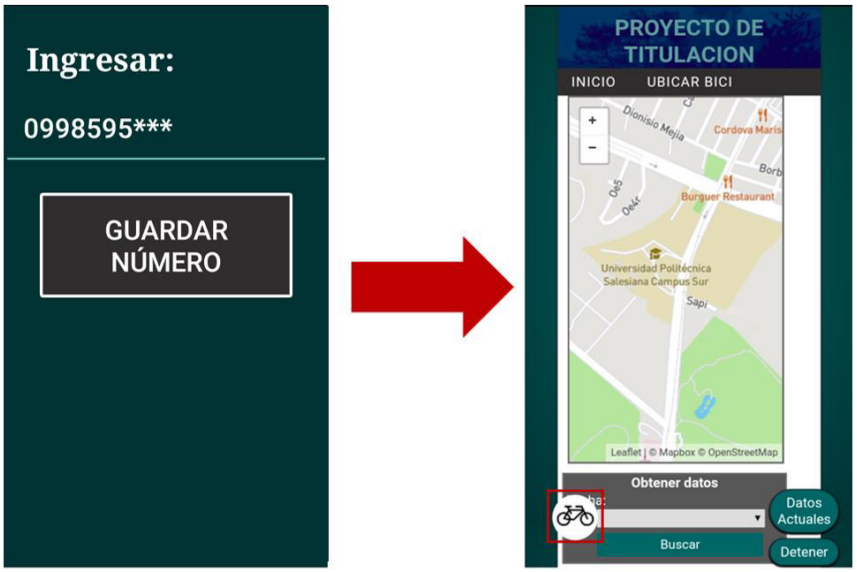


Figura 3.67 Prueba en actividad “Configuración” al presionar botón guardar número

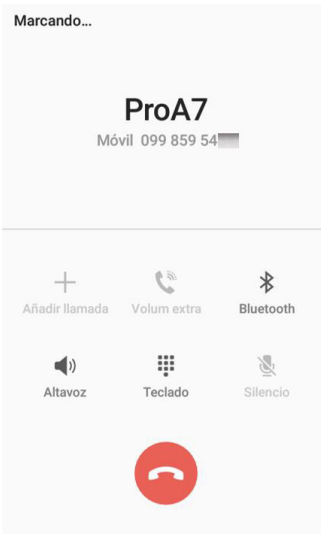


Figura 3.68 Prueba en actividad “Mostrar mapa” al presionar botón de bicicleta

4. De manera automática el dispositivo colgara la llamada indicando que va a empezar a recoger datos de posición y enviarlos, lo cual en la aplicación móvil se debe presionar el botón llamado “Datos Actuales” y así empezará a mostrar la ubicación de la bicicleta en el mapa con uno o más marcadores, ver Figura 3.69.

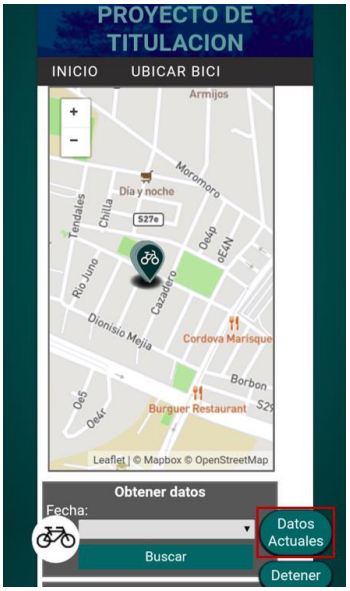


Figura 3.69 Prueba en actividad “Mostrar mapa” al presionar botón “Datos Actuales”

Para finalizar la recepción de datos de posición, se debe de volver a realizar la llamada apretando el botón en forma de bicicleta, esto para indicar al dispositivo que debe detener él envió de datos de posición al servidor *ThingSpeak*.

5. Los datos de posición al ser enviados son almacenados con una fecha y una hora, esto se puede apreciar en la opción “Ultimos fechas” de la actividad “Mostrar mapa”, además se puede acceder a ese registro nuevamente ingresando la fecha en la opción “Obtener datos”, ver Figura 3.70.

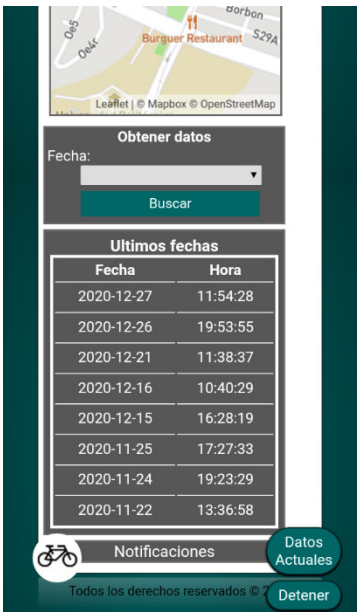


Figura 3.70 Actividad “Mostrar mapa” con opciones últimas fechas y obtener datos

6. Al seleccionar la fecha que contiene los registros de posición tomadas en ese día, se presiona el botón “Buscar” para mostrar en el mapa los marcadores de posición guardadas en esa fecha, ver Figura 3.71.

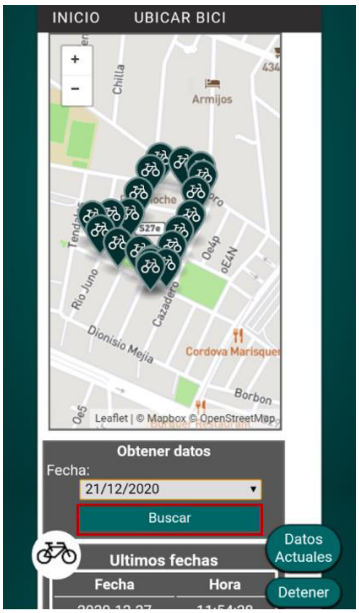


Figura 3.71 Actividad “Mostrar mapa” con fecha en obtener datos

3.6 Manual de Uso y Mantenimiento

Manual de uso

https://epnecuador-my.sharepoint.com/:v:/g/personal/leandro_pazmino_epn_edu_ec/ET44zDr5SU1Bp7SyCv_j-IABOhSPRMpGUanpeRZcFvd77Q?e=ec2JUN

Manual de mantenimiento

https://epnecuador-my.sharepoint.com/:v:/g/personal/leandro_pazmino_epn_edu_ec/ESUz09LSTkNNjkeitCiwiREBBAzBRm98kR1H6eAj6SUP-g?e=AI7nkm

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Las tecnologías identificadas para la implementación del prototipo han sido seleccionadas de manera correcta, ya que han cumplido parámetros como cobertura amplia, incluso cubriendo una cuidad completa sin presentar inconvenientes con respecto a la pérdida de la señal, por tal razón se ha seleccionado la red GSM y para la transmisión de datos en el orden de los kilobytes se eligió la red GPRS. Además, por la precisión, disponibilidad y el número de satélites el sistema de posicionamiento GPS se ajusta a los requerimientos del prototipo.

- El desarrollo del *software* del prototipo permite que el tiempo de reacción sea rápido tanto en el momento de solventar un problema que se presente o simplemente al trabajar normalmente para enviar datos de posición. Además, para introducir una mayor seguridad en el prototipo se requiere el registro del número telefónico y tener la llave API para enviar los datos al servidor.
- El análisis realizado permitió seleccionar el hardware que posea las características justas y necesarias para su correcto funcionamiento con el menor consumo de energía posible, esto posibilitó un mayor tiempo de uso del prototipo y una gran eficiencia, por tales motivos se seleccionó al Arduino Nano, el módulo A7, el Neo6m GPS y el conversor DC-DC step-down LM2596.
- El prototipo permite al usuario acceder a los datos de posición por medio de la aplicación móvil y por la página web, esto ofrece una ventaja al tener más formas de acceder a la información de localización y de esta manera no depende solo de una plataforma tecnológica.
- La aplicación móvil presenta una mayor ventaja que la página web, debido a que cuenta con la facilidad de llamar al dispositivo de manera automática sin necesidad de marcar el número telefónico, sin embargo; la página web presenta mayor rapidez en mostrar los datos de posición, debido a que la petición lo hace directamente al servidor *ThingSpeak*, mientras que la aplicación móvil, hace la petición al servidor web, a pesar de esto, este tiempo extra que se demora en cargar los datos de posición no es notable en la práctica.
- En las pruebas se comprobó que, al iniciar el prototipo de geolocalización, el tiempo requerido para conectarse a la red GSM es aproximadamente uno o dos minutos, posteriormente el prototipo puede recibir la llamada para empezar a

entregar coordenadas de posición, además, si ocurre algún error en el proceso, el prototipo es capaz de auto reiniciarse y volverse a conectar a la red para esperar nuevamente la llamada.

- El consumo de saldo en la tarjeta micro SIM no es muy significativo porque solo envía datos de posición en el orden de los kilobytes.
- El tamaño del prototipo no se encuentra optimizado, éste presenta dimensiones significativas son debido a que se emplean módulos y una batería de tamaños grandes, sin embargo, se ha logrado un tamaño del prototipo adecuado.
- La precisión del prototipo es apropiada para lograr localizarlo, su precisión en función al módulo GPS Neo6m es de 2.5 (m), este valor es mejor que los 3 o 5 (m) estándar de los módulos GPS del mercado. Debido a la gran precisión puede permitir la ubicación del prototipo cuando se presente un robo.
- El emplear servicios de terceros tanto en la red telefónica celular, así como en los servidores web *ByetHost* y *ThingSpeak*, no permite tener un control total del funcionamiento del prototipo, debido a que al fallar alguno de estos servicios, el prototipo de geolocalización no funcionara como se desea.
- Para empezar a tomar datos de posición es necesario que se realice una llamada al prototipo, activar el botón en las aplicaciones para tomar datos de posición y para detener la recepción de datos se debe presionar el botón de “Detener” en las aplicaciones, finalmente volver a llamar al dispositivo para detener el envió de datos del prototipo, todo este proceso representa demoras y es molesto. Una forma de solucionar este problema es aplicando solo la tecnología GPRS, es decir, solo aplicar peticiones entre servidores y el prototipo.
- El prototipo solo funciona para un usuario, por tal razón en la aplicación móvil y página web no requiere autenticación, sin embargo, si se quiere aplicar para uso comercial, es necesario la creación de una base de datos para gestionar a los usuarios, también se requiere de un servidor para receptar los datos de posición de cada usuario, de manera directa y no necesitar un intermediario como lo hace el servidor *ThingSpeak* en este proyecto.

4.2 Recomendaciones

- El tamaño del prototipo es un factor clave para evitar que este sea quitado en el momento de un robo, por tal razón, es preferible emplear módulos más pequeños o diseñar una placa PCB más compacta con todas las tecnologías necesarias para el prototipo de geolocalización.
- El servidor *ThingSpeak* para este proyecto funciona como un intermediario, esto implica que los tiempos para mostrar datos al servidor web sean más largos, para evitar los retardos y tener un control total del sistema, se recomienda en trabajos a futuro, levantar un servidor que realice funciones similares a *ThingSpeak*.
- El prototipo no cuenta con características muy importantes como es indicador de descarga o carga de la batería y del saldo de la tarjeta micro SIM, para futuras mejoras estas características deben ser incorporadas.
- Se debe optimizar los tiempos de respuestas tanto del *hardware* como del *software* del prototipo, aunque los tiempos ya son adecuados para la ubicación, si es posible aumentar el tiempo de ejecución.
- Con la aplicación móvil y la página web es posible ubicar al prototipo dentro del mapa, una posible mejora es permitir mostrar también la posición del usuario en el mapa, este factor puede ser muy importante para agilizar la búsqueda.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] INEC, «A pedaliar 19 de abril dia mundial de la Bicicleta,» 17 abril 2017. [En línea]. Available: https://bit.ly/39vCGCq. [Último acceso: 31 Marzo 2020].

[2] L. Zambrano, «Los negocios de bicicletas pedalean hacia la cima,» Diario Expreso, 22 abril 2017. [En línea]. Available: https://bit.ly/37pLGaP. [Último acceso: 30 Marzo 2020].

[3] «El Comercio,» Grupo EL COMERCIO, 16 febrero 2019. [En línea]. Available: https://bit.ly/2OPiAej. [Último acceso: 31 Marzo 2020].

[4] M. Cardenas, «¿Realmente habia impuesto a la importacion de bicicletas de competencia en Ecuador?,» Metro, 12 junio 2019. [En línea]. Available: https://bit.ly/2UWuneG. [Último acceso: 01 Abril 2020].

[5] F. F. D. A. Nataly Pinto Alvaro, «La situacion de la bicicleta en Ecuador: avances, retos y perspectivas,» 01 Marzo 2015. [En línea]. Available: https://bit.ly/38rtUVY. [Último acceso: 01 Abri 2020].

[6] Universo, «Robos a personas aumentaron en Quito,» El Universo, 19 febrero 2019. [En línea]. Available: https://bit.ly/3bzp2A1. [Último acceso: 13 Noviembre 2019].

[7] ElComercio, «El Comercio,» El Comercio, 24 Junio 2019. [En línea]. Available: https://bit.ly/2SnglRL. [Último acceso: 10 Abril 2020].

[8] ElComercio, «El Comercio,» EL Comercio, 22 julio 2019. [En línea]. Available: https://bit.ly/2HmMfrk. [Último acceso: 13 noviembre 2019].

[9] E. T. D. d. Cuenca, «19 a 30 años, las que mas usan bicicletas en Cuenca,» Medios Publicos EP, 30 septiembre 2018. [En línea]. Available: https://bit.ly/2SoBUkQ. [Último acceso: 12 Abril 2020].

[10] M. Ecuador, «¿Cuantos estacionamientos para bicicletas existen en Quito?,» Metro Ecuador, Jueves 21 2017. [En línea]. Available: https://bit.ly/2ORN4fW. [Último acceso: 21 Abril 2020].

[11] B. Eissfeller, G. Ameres, V. Kropp y D. Sanroma, «Performance of GPS, GLONASS and Galileo,» 2007.

[12] «European Global Navigation Satellite Systems Agency,» EU Agency, 17 Julio 2018. [En línea]. Available: https://bit.ly/3bqYcMS. [Último acceso: 09 Enero 2021].

[13] «European Global Navigation Satellite Systems Agency,» EU Agency, 17 Abril 2020. [En línea]. Available: https://bit.ly/3bqY542. [Último acceso: 09 Enero 2021].

[14] B. Hofmann-Wellenhof, H. Lichtenegger y E. Wasle, GNSS – Global Navigation Satellite Systems, Springer Science & Business Media, 2007.

[15] E. Huerta, A. Mangiaterra y G. Noguera, GPS Posicionamiento satelital, Argentina: UNR, 2005.

[16] C. Leon W, Sistemas de comunicación digitales y analógicos, Mexico: PEARSON, 2008.

[17] A. S. Tanenbaum, Redes de computadoras, Mexico: Pearson Eduacion, 2003.

[18] W. Stalling, Comunicaciones y Redes de computadores, Madrid: PEARSON, 2004.

[19] C.-M. Lu, «COMMUNICATION SYSTEM FOR DEVICES WITH UART INTERFACES». Estados Unidos Patente US 7,650,449 B2, 19 Junio 2010.

[20] P. J. Hays y W. M. Mansfield, «SYSTEM FOR SETTING FRAME AND PROTOCOL FORTRANSMISSION IN A UART DEVICE». Estados Unidos Patente US 6,678,751 B1, 13 Junio 2004.

[21] N. M. E. Association, «NMEA,» National Marine Electronics Association, [En línea]. Available: https://bit.ly/3nt1WQs. [Último acceso: 09 Enero 2021].

[22] J. James Garrett, «adaptivepath,» 18 Febrero 2005. [En línea]. Available: https://bit.ly/3hZXI1F. [Último acceso: 09 Enero 2021].

[23] A. Ayoze Castillo, Curso de Programación Web JavaScript, Ajax y jQuery, ITCampus Academy, 2017.

[24] D. Peng, L. Cao y W. Xu, «Using JSON for Data Exchanging in Web Service Applications,» 2011.

[25] A. Cobo, P. Gómez, D. Pérez y R. Rocha, PHP y MySQL tecnologías para el desarrollo de aplicaciones web, España: Díaz de Santos, 2005.

[26] J. D. Gauchat, El gran libro de HTML5, CSS3 y JavaScript, Marcombo, 2012.

[27] «MathWorks,» The IoT Platform with MATLAB Analytics, [En línea]. Available: https://bit.ly/2LBbgEu. [Último acceso: 09 Enero 2021].

[28] D. Axmark y M. Widenius, «MySQL,» MySQL Documentation Team, [En línea]. Available: https://bit.ly/2K4DBmk. [Último acceso: 09 Enero 2021].

[29] M. Delisle, «phpMyAdmin,» phpMyAdmin contributors, [En línea]. Available: https://bit.ly/3brft8A. [Último acceso: 09 Enero 2021].

[30] L. Pan, X. Zhang, X. Li, X. Li, C. Lu, J. Liu y Q. Wang, «Satellite availability and point positioning accuracy evaluation on a global scale for integration of GPS, GLONASS, BeiDou and Galileo,» *Advances in Space Research,* vol. 63, pp. 2696-2710, 2019.

[31] J. Salazr, Redes Inalámbricas, Republica checa: TechPedia, 2016.

[32] Ai-Thinker, «A6/A7/A6C User Manual,» 2016.

[33] «Arduino,» Arduino - ArduinoBoardNano, [En línea]. Available: https://bit.ly/38zwOut. [Último acceso: 09 Enero 2021].

[34] U-blox, «NEO-6 u-blox 6 GPS Modules,» 2017. [En línea]. Available: https://bit.ly/35t0ZS1. [Último acceso: 09 Enero 2021].

[35] M. N. O. Charles K. Alexander, Fundamentos de circuitos electricos, mexico: McGranwHill, 2006.

[36] F. Van Den Abeele, J. Haxhibeqiri, I. Moerman y J. Hoebeke, «ATmega48A / PA / 88A / PA / 168A / PA / 328 / P megaAVR ® Data Sheet ATmega48A / PA / 88A / PA / 168A / PA / 328 / P,» *IEEE Internet of Things Journal,* vol. 4, pp. 2186-2198, 2017.

[37] T. Ft, «Ft232R Usb Uart Ic,» *Technology,* pp. 1-40, 2008.

[38] Ó. Torrente Artero, ARDUINO Curso práctico de formación, RC Libros, 2013.

[39] G. E. Harper, El Libro Practico De Los Generadores, Transformadores Y Motores Electricos / The Practical Book of Generators, Transformers and Electical Motors, Limusa, 2005.

[40] Datasheet, «Step-Down Switching Regulator,» Europe, Middle East and Africa Technical, 2018.

[41] Arduino, «Arduino,» Software Serial Example, [En línea]. Available: https://bit.ly/3q2grML. [Último acceso: 09 Enero 2021].

[42] j. License, «jQuery,» OpenJS Foundation and jQuery contributors, [En línea]. Available: https://bit.ly/3sb3yBU. [Último acceso: 09 Enero 2021].

[43] M. Contributors, «MDN contributors,» Mozilla and individual contributors, 17 Noviembre 2020. [En línea]. Available: https://mzl.la/3scx1vm. [Último acceso: 09 01 2021].

[44] ByetHost, Byet Internet Services, 2005. [En línea]. Available: https://byet.host/. [Último acceso: 04 Abril 2020].

[45] A. Studio, «developers,» Google Developers, [En línea]. Available: https://developer.android.com/studio. [Último acceso: 9 Enero 2021].

ANEXOS

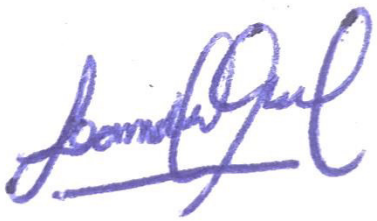
ANEXO 1: CERTIFICADO DE FUNCIONAMIENTO

ESCUELA POLITECNICA NACIONAL
Campus Politécnico "J. Rubén Orellana R

Quito, 11 de junio de 2021

CERTIFICADO DE FUNCIONAMIENTO DE PROYECTO DE TITULACIÓN

Yo, *Leandro Antonio Pazmiño Ortiz,* docente a tiempo completo de la Escuela Politécnica Nacional y como director de este trabajo de titulación, certifico que he constatado el correcto funcionamiento del prototipo diseñado, desarrollado e implementado por Roger Reyes. Por lo tanto, este proyecto cumple con todos objetivos establecidos en el plan de titulación.



DIRECTOR
Ing. Leandro Antonio Pazmiño Ortiz., Msc.

Ladrón de Guevara E11-253, Escuela de Formación de Tecnólogos, Oficina 28. EXT: 2729
email: pablo.proano@epn.edu.ec Quito-Ecuador

ANEXO 2: CODIGO COMUNICACIÓN SERIAL Y SERIALA7

```
/*******FUNC COMUNICACION_SERIAL*************************************/
void ComunicacionSerial() {
  ////////FUNCION SERIALA7////////////////////////////////
  serialA7();
  ///////SERIAL DE ARDUINO//////////
  while (Serial.available()) {
    //Serial arduino lee, modulo A7 escribe comando
    proA7.write(Serial.read());
  } delay(50);
}
```

```
/*********FUNCION SERIALA7*****************************************/
void serialA7() {
  proA7.listen();
  while (proA7.available() > 0) {
    valorChar = proA7.read(); // Recepta valores Char
    valorStr += valorChar; //valorStr recoge los valores char de valorC
har
    if (valorChar == '\n') { // Detecta "\n" (salto de linea)
      Serial.print("c_s:"); Serial.println(valorStr);
      valorStr.toCharArray(trama1, 100); //ValorStr se almacena en arre
gloChar
      ///////DETECTA EL RING/////////
      if ((strstr(trama1, " RING") != NULL)) { // Detecta si trama1 = R
ING
        analogWrite(ledLlamada, encender);  //Indicador de llamada entr
ante
        valorStr = ""; // Limpia valorStr
        memset(trama1, '\0', 100); // Limpia Trama
        conteoRing++; // Aumenta conteoRing
        //Detectar si conteoRing es igual a uno
        if(conteoRing==1){
          //Se ejecuta cada vez que inicializa el dispositivo
          proA7.println("AT+CLCC"); // Identifica que numero llamada
          delay(100);
        }
        proA7.println("ATH"); // Cuelga la llamada
        delay(100);
        analogWrite(ledLlamada, apagar); //Indicador de llamada entrant
e
        delay(100);
      }
```

```
      ///////DETECTA EL +CLCC contiene el numero telefonico/////////
      // Detecta si trama1 = +CLCC:...
      else if ((strstr(trama1, "+CLCC: 1,1,4,0,0,") != NULL) && NumTelS
av.length() <= 11) {
        // num_telefoYmas contiene el numero telefonico y mas valores
        valorStr.toCharArray(num_telefoYmas, 100);
        //Llamar la funcion "detectarNumTel", devolvera solo el numero
telefonico
        NumTelRec = detectarNumTel(num_telefoYmas);
        valorStr = " "; // Limpia valorStr
        memset(trama1, '\0', 100); // Limpia Trama
        analogWrite(ledLlamada, apagar); //Indicador de fin de llamada
      }
      ///////BORRA STRING Y TRAMA1/////////
      else {
        valorStr = " "; // Limpia valorStr
        memset(trama1, '\0', 100); // Limpia Trama
      }
    }
  }
}
```

ANEXO 3: CODIGO DETECTAR NUMERO TELEFONICO

```
/**********DETECTAR NUMERO TELEFONICO*****************************/
String detectarNumTel(char *tramaPhone){ //Devolvera un valor String
  String strNumTel = ""; // StrNumTel almacenara el numero telefonico
  char *cadenaSeparar; // Cadena a separar
  char *token;  // token es SubStrings de un String almacena la separac
ion
  byte conteo = 0; // contador

  if (tramaPhone[0] != NULL ) { // Si tramaPhone no esta vacio
    cadenaSeparar = tramaPhone; // Almacena "tramaPhone" en "cadenaSepa
rar"
    // "," es el delimitador para separar
    while ((token = strtok_r(cadenaSeparar, ",", &cadenaSeparar)) != NU
LL) {
      conteo ++; // Aumenta "conteo"
      if (conteo == 6) { // Si "conteo"=6
        strNumTel = token; // Token se almacena en "strNumTel"
      }
    }
    if (strNumTel != NULL) { // Identifica si strNumTel no esta vacio
      return strNumTel; //Devuelve un string con el numero telefonico
    }
    else{
      return "Numero no guardado"; //Devuelve si strNumTel esta vacio
    }
  }
}
```

ANEXO 4: CODIGO RECEPCION DE DATOS GPS POR ARDUINO

```
//PROCESO DE RECOGER COORDENADAS
  else if (encenderGPS) {
    //Encender led GPS
    analogWrite(ledGPSon, encender);
    //Escuchar al modulo neo6m por puerto serial
    neo6.listen();
    //Neo6m habilitado
    while (neo6.available() > 0) {
      //Codificar los valores NMEA entregada por modulo Neo6m
      gps.encode(neo6.read());
      //Deteccion de nuevas coordenadas
      if (gps.location.isUpdated()) {
        //Trama a enviar con datos de posicion en los campos field1 y f
ield2
        coordenadas = "&field1=" + String(gps.location.lat(), 6) + "&fi
eld2=" + String(gps.location.lng(), 6);
        //Asignacion de variables
        encenderGPS = false;
        enviarDatos = true; //enviarDatos sera verdadera
        //Apagar led GPS
        analogWrite(ledGPSon, apagar);
      }
    }
  }
```

ANEXO 5: CODIGO CONEXIÓN DESDE PROTOTIPO DE GEOLOCALIZACION A LA RED

```
/**********TCP_GPRS*****************************/
bool TCP_GPRS(bool aRed, bool aServer) {
  if (aRed) { //Condicion para conectar a la red
    bool creg = false, cgatt = false, cgdcont = false, cgact = false;
    creg = escribirComando("AT+CREG?", 2000, false); //Consultar regris
tro de red
    delay(100);
    serialA7();//___
    cgatt = escribirComando("AT+CGATT=1", 6000, false);//
    delay(100);
    serialA7();//___
    //Configuracion del paramtro PDP
    cgdcont = escribirComando("AT+CGDCONT=1,\"IP\",\"CMNET\"", 2000, fa
lse); //PDP
    delay(100);
    serialA7();//___
    //Activar PDP, Abrir servicios de internet
    cgact = escribirComando("AT+CGACT=1,1", 10000, false);
 //Activate PDP, open Internet service
    delay(100);
    serialA7();//___
    if (creg && cgatt && cgdcont && cgact && !aServer) {
      Serial.println("Conectado a Red");
      return true;
    }
  }
  if (aServer) { //Condicion para conectar al servidor
    bool confirmar = false;
    serialA7();
    //Establecer conexion con servidor ThingSpeak
    confirmar = escribirComando("AT+CIPSTART=TCP,api.thingspeak.com,80"
, 10000, true);
    if (confirmar) {
      analogWrite(onRed, encender);//Enciende Led "RED"
      return true; //Retorna Verdadero
    }
    analogWrite(onRed, apagar); // Apaga Led "RED"
    return false; //Retorna Falso
  }
}
```

```
//////////FUNCION PARA CONECTAR A LA RED
bool coneccion_a_Red (bool aRed, bool aServer) {
  //Comprobar si activa todo de TCP_GPRS
  while (TCP_GPRS(aRed, aServer) != true) {
    ////////ACTIVA TCP_GPRS
    escribirComando("AT+CIPCLOSE", 4000, true);//Cierra la conexion al
servidor
    delay(100); //1000
    serialA7();//___
    escribirComando("AT+CIPSHUT", 4000, true); //Cierra la conexion a l
a red
    delay(100);
    serialA7();//___
    Serial.println("Desactivado TCP_GPRS");
    analogWrite(onRed, apagar); //Apaga Led "RED"
  }
  analogWrite(onRed, encender); //Enciende Led "RED"
  delay(500);
  Serial.println("Activado TCP_GPRS!!!");
  return true; // Devuelve true por que ya existe conexion
}
```

ANEXO 6: CODIGO PROCESO DEL ENVIO DE DATOS GPS AL SERVIDOR

```
/******ENVIA EL MENSAJE DE POSICION POR DATOS***********/
void enviarDatosGPS(String posicion) {
  //CONECTANDO CON SERVIDOR A LA RED
  byte cont = 0;
  bool aRed = false; // No conectar a la red
  bool aServer = true; // Para conectar al servidor
  //Llamar a la funcion "TCP_GPRS"
  while (TCP_GPRS(aRed, aServer) != true) {
    delay(500);
    cont++;
    if (cont % 5 == 0) {
      //Reintentar conexion con modulo A7
      proA7.println("AT+CPOF"); delay(1000);
      configuracionInicial(); // Configuracion inicial
    }
  }
  serialA7();//___

  //ENVIANDO DATOS AL SERVIDOR
  if (posicion.length() > 32) {
    //Llave API del servidor ThingSpeak
    String dato = "GET https://api.thingspeak.com/update?api_key=2CNOXE
GVDQ3X35K7";
    dato += String(posicion); //Agrego los valores de posicion obtenida
s por el Neo6
    serialA7();//___
    Serial.println(dato);
    serialA7();//___
    delay(200);
    analogWrite(ledSend, encender); //LED "Send" encendido
    proA7.println("AT+CIPSEND"); // Comando AT para enviar
    delay(500);
    serialA7();
    proA7.println(dato);//Especifivar la variable a enviar
    delay(500);
    serialA7();
    proA7.println((char)26);//Indicador de fin de mensaje
    delay(6000);
    serialA7();

    escribirComando("AT+CIPCLOSE", 5000, true); //Cierro conexion con s
ervidor
    serialA7();//___
```

```
    delay(500);
    Serial.println("Enviado");//Indicando que se ha enviado el mensaje
al servidor
    serialA7();//___
    analogWrite(ledSend, apagar); //Apagar LED Send Indicando el envio
del mensaje
    delay(100);
  }
}
```

ANEXO 7: CODIGO PROCESO CENTRAL CONTROLADO POR LLAMADA

```
/*****************FUNC LOOP*****************************/
void loop() {
  ////////MODO NORMAL
  if (NumTelRec != NumTelSav) { // Son diferentes, ninguna similitud NU
LL
    NumTelSav = NumTelRec; //Se almacena el NumTelRec en NumTelSav
    Serial.print("Telefono Guardado: "); Serial.println(NumTelSav);
  }
  //ACTIVAR DATOS
  //Los numeros telefonicos son iguales, ConteoRign es un numero Par y
mayor a cero, enviarDatos es falso
  else if ((conteoRing % 2) == 1 && NumTelSav == NumTelRec && conteoRin
g > 0 && enviarDatos == false && encenderGPS == false) {
    Serial.println("---DATOS ENCENDIDO---");
    delay(500);
    encenderGPS = true;
    enviarDatos = false; //enviarDatos sera true
    Serial.println("GPS on");
  }
  //PROCESO DE RECOGER COORDENADAS
  else if (encenderGPS) {
    //Encender led GPS
    analogWrite(ledGPSon, encender);
    //Escuchar al modulo neo6m por puerto serial
    neo6.listen();
    //Neo6m habilitado
    while (neo6.available() > 0) {
      //Codificar los valores NMEA entregada por modulo Neo6m
      gps.encode(neo6.read());
      //Deteccion de nuevas coordenadas
      if (gps.location.isUpdated()) {
        //Trama a enviar con datos de posicion en los campos field1 y f
ield2
        coordenadas = "&field1=" + String(gps.location.lat(), 6) + "&fi
eld2=" + String(gps.location.lng(), 6);
        //Asignacion de variables
        encenderGPS = false;
        enviarDatos = true; //enviarDatos sera verdadera
        //Apagar led GPS
        analogWrite(ledGPSon, apagar);
      }
    }
  }
```

```
  //POCESO DE ENVIO DE DATOS
  else if (enviarDatos) {
    enviarDatosGPS(coordenadas); //Llamar a la funcion "enviarDatosGPS"
    serialA7();//___
    if (escribirComando("AT", 4000, true) != true) { // Envia AT y espe
ra respuesta OK
      Serial.println("--Desconectando A7--");
      proA7.println("AT+CPOF"); delay(1000);
      configuracionInicial(); // Configuracion inicial
    }
    serialA7();//___
    encenderGPS = true;
    serialA7();//___
    enviarDatos = false; //enviarDatos sera true
    Serial.println("GPS on");
    serialA7();//___
    delay(2000);
    serialA7();//___
    delay(2000);
    serialA7();//___
    delay(2000);
    serialA7();//___
    delay(2000);
    serialA7();//___
    delay(2000);
  }
  //APAGAR DATOS
  //Los numeros telefonicos son iguales, ConteoRign es un numero Impar
y mayor a cero
  else if ((conteoRing % 2) == 0 && NumTelSav == NumTelRec && conteoRin
g > 0) {
    ////////DESACTIVAR enviarDatos
    enviarDatos = false; //enviarDatos sera false
    ////////DESACTIVAR enviarDatos
    encenderGPS = false; //enviarDatos sera false
    Serial.println("--Conexion cerrada--");
    ComunicacionSerial(); //Llamar a funcion "ComunicacionSerial"
  }
  else {
    ComunicacionSerial(); //Llamar a funcion "ComunicacionSerial"
  }
}
```

ANEXO 8: CODIGO RECEPCION DE DATOS ENTRE SERVIDOR *THINKSPEAK* Y SERVIDOR WEB

```
//Activar la recepcion de datos
    $('.bt_datoActual').click(function(){
        //Memoria local, asignando a 'ajax' el valor true
        localStorage.setItem('ajax',true);
        location.reload();
    });
    //Detener la recepcion de datos
    $('.bt_detener').click(function(){
        //Memoria local, asignando a 'ajax' el valor false
        localStorage.setItem('ajax',false);
        location.reload();
    });
    //Obtener valores de la memoria local con el nombre 'ajax'
    var estado = localStorage.getItem('ajax');
    console.log(estado=="true", estado!=null);
    //Comporando si 'estado' es verdadero o si no esta vacio
    if(estado=="true" && estado!=null ){
        var id_pos_old=0;
        //Establecer un intervalo de 5seg y llama a la funcion 'ajax'
        var intervalo = setInterval(ajax,5000);
    }
    //Funcion ajax
    function ajax(){
        //Opcion de AJAX de jQuery
        $.ajax({
            //Recepcion GET de HTML
            type: 'GET',
            //url del servidor ThingSpeak
```

```
            url: 'https://api.thingspeak.com/channels/1073852/feeds.json?
results=2',
            //Antes de Enviar
            beforeSend:function(){
                    console.log('Cargando');
            },
            //Recoger valores de ThingSpeak y se almacena en 'response'
            success:function(response){
                console.log('alimentacion: ',response.feeds);
                //Recoger el index
                var index = (response.feeds.length)-1;
                console.log('id: ',response.feeds[index]);
                //Recoger datos por la index
                var datos = response.feeds[index];
                //Variables almacena posicion y id
                var id_pos = datos['entry_id'];
                var latitud = datos['field1'];
                var longitud = datos['field2'];
                //Verificar si ya existe 'id_pos'
                if(id_pos !== id_pos_old){
                    console.log('idOldAjx: ',id_pos_old);
                    var result = ingresar_datos(id_pos, latitud, longitud
)
                    console.log(result);

                    console.log('Completado!!!!!');
                    document.location.href='http://localhost/master-
php/proyectMap/db.php';
                }
                else{
                    console.log('Ningun dato nuevo');
                }
```

```
            },
            error:function(){
                    console.log('error');
            }
        });
    }
```

ANEXO 9: CODIGO PROCESOS EN PHP

```php
<?php
include_once 'model/conexionDB.php';
include_once 'controller/queryDBcontroller.php';
include_once 'controller/mostrarMarkerController.php';

$api_key= $latitud = $longitud = "";

//COOKIES
if(isset($_COOKIE['id_pos']) && isset($_COOKIE['lat']) && isset($_COOKIE['log']) && isset($_COOKIE['api_key'])){
    //Recoger Cookies
    //Recoger valor de id_pos
    $id_posicion = $_COOKIE['id_pos'];
    //Recoger valor de lat
    $latitud = $_COOKIE['lat'];
    //Recoger valor de log
    $longitud = $_COOKIE['log'];
    //Recoger valor de api_key
    $api_key = $_COOKIE['api_key'];
    //Llamar a la funcion guardar y enviar parametros
    guardar($id_posicion,$latitud, $longitud, $api_key);
}
else{//Si no contiene valor las Cookies
   $_SESSION['error_cookie'] = 'No existe valores en Cookie';
    goto dirigir;
}
//Redireccionar a la pagina 'ubicarBici'
dirigir: header('location: ubicarBici.php');
```

```php
//Funcion guardar para GUARDAR DATOS EN BASE DE DATOS
function guardar($id_posicion,$latitud, $longitud, $api_key){
    //Borramos cookies
        setcookie('id_pos' ,'',time()-100);
        setcookie('lat'    ,'',time()-100);
        setcookie('log'    ,'',time()-100);
        setcookie('api_key','',time()-100);

        $api_key_value = "12345";
    //Comparacion de api_key_value
    if($api_key === $api_key_value) {
        // Si hay algo en sesion
        if(empty($_SESSION)){
            session_start();     //Iniciar Sesion
        }

        //CREO CONEXION CON BASE DE DATOS
        $conexion = new DB();
        $conn= $conexion->conexionDB();

        // Revisar la conexion
        if ($conn->connect_error) {
            //Si existe algun error se envia un mensaje
            $_SESSION['error_bd'] = "Sin conexion con base de datos: </br
>" . $conn->connect_error;
            $conn->close();
            goto dirigir;
        }

        //Se llama al controlador
        $query_id_posicion = new QueryDBcontroller();
```

```
        //Se optiene la id_posicion de la base de datos
        $result_id_posicion = $query_id_posicion-
>seleccionar_id_posicion();
        //Si "$result_id_posicion" existe
        if($result_id_posicion == TRUE ){
            $resultQuery_id_posicion = $result_id_posicion-
>fetch_Object();//Separo objeto
            $resultQuery_id_posicion = $resultQuery_id_posicion-
>id_posicion;


            //COMPROBAR SI EXISTEN DATOS
            if( $resultQuery_id_posicion != $id_posicion){
                //Codigo para almacenar los datos en la base de datos
                $sql = "INSERT INTO poslatlog VALUES ( NULL,$id_posicion,
 $latitud, $longitud, CURDATE(), SYSDATE())";
                //Se almacena en la base de datos
                if ($conn->query($sql) === TRUE) {
                    //Mensaje de datos almacenados correctamente
                    $_SESSION['exito'] = "Nuevos datos guardados!!";


                    //GUARDO LAT Y LOG EN SESION
                    $_SESSION['latitud']=$latitud;
                    $_SESSION['longitud']=$longitud;
                    //Mostrar todos los datos por dia
                    showInMap();
                }
                //Si existe algun erro en almacenar en base de datos
                else {$_SESSION['error']['guardar']="Error en guardar dat
os: <br>" . $conn->error;}
                $conn->close();
            }
            //Si el id_posicion no a cambiado
```

```
            else{$_SESSION['error_id_posicion'] = "Sin nuevos valores";}
        }
        //Si no encuentra un valor de id_posicion
        else{
           $_SESSION['error_consultaId'] = 'Error de comparacion: '.$conn
->error;
            $conn->close();
            goto dirigir;
        }
    }
    //Si la llave API no coincide
    else {
        $_SESSION['error_api'] = "Error de llave API";
         goto dirigir;
    }
    //Redirige a 'ubicarBici'
    dirigir: header('location: ubicarBici.php');
  }

//funcion showInMap
function showInMap(){
    //Se llama al controlador 'MostrarMarkerController'
    $mostrarEnMap = new MostrarMarkerController();
    //Llamo a objeto 'mostrarMarkerEnMap'
    $mostrarEnMap->mostrarMarkerEnMap("CURDATE()");
    //Redirige a 'ubicarBici'
    header('location: ubicarBici.php');
}
?>
```

ANEXO 10: CODIGO DE ARCHIVO JSON

```php
<?php
include_once 'queryDBcontroller.php';
class MostrarMarkerController{
    //Funcion 'mostrarMarkerEnMap' con parametro
    function mostrarMarkerEnMap($fecha){
        //Llamo al objeto 'QueryDBcontroller'
        $queryUltimosDatos = new QueryDBcontroller();
        //Lamar al metodo 'posicionPorFecha'
        $posiciones = $queryUltimosDatos->posicionPorFecha($fecha);
        //Variables para JSON
        $posicionJSON=array();
        $aMapa = array();
        $cont=0;

        while ($posicion = $posiciones->fetch_object()) {
            $aMapa['longitud']=$posicion->longitud;
            $aMapa['latitud']=$posicion->latitud;
            //Almacenar un arrya dentro del otro con un index
            $posicionJSON[$cont]=$aMapa;
            $cont++;
        }
        //Crear Directorio
        if(!is_dir('text')){
            //Asignar permisos
            mkdir('text',0777);
            if(!file_exists('valores.json')){
                //Escribir en el archivo 'valores.json'
                $myfile = fopen("text/valores.json", "w");
                fwrite($myfile, json_encode($posicionJSON));
                fclose($myfile);// Cerrar el archivo
            }
        }else{
            //Crear archivo 'valores.json'
           $myfile = fopen("text/valores.json", "w");
           //Escribir en el archivo 'valores.json'
           fwrite($myfile, json_encode($posicionJSON));
           fclose($myfile); // Cerrar el archivo
        }
    }
}
```

ANEXO 11: CODIGO PROCESOS CON JAVASCRIPT

```
<!--OBTENER DATOS-->
<?php
include_once('model/queryBD.php');
//Obtener de bases de datos
$res = new QueryBD;
//Obtener ultimos datos registrados
$val = $res->ultimosDatosPosicion();
//Se almacenan en las variables 'longitud' y 'latitud'
$row = $val->fetch_assoc();
$longitud = $row["longitud"];
$latitud = $row["latitud"];
?>

<!--FUNCION MAPA-->
<script>
   //Ubicacion ultimos datos
   var longitud = "<?php echo $longitud ?>";
   var latitud = "<?php echo $latitud ?>";
   console.log(longitud + ' - ' + latitud);
   //Variable para 'vistaCentral' en formato JSON
   var vistaCentral = {lat: parseFloat(latitud), lng: parseFloat(longit
ud)};

   //RECOJER JSON
   //Obtener lo almacenado en el formato JSON y se almacena en 'data'
   var contenedor = $.getJSON('text/valores.json',function(data){
       //Definir array
        var geol=[];
        //Recorrer lo que contiene 'data'
        $.each(data, function( index, value ) {
            //Definir valores para mostrar por el mapa
            obj={
                    type: 'Feature',
                    //Posicion
                    geometry: {
                        type: 'Point',
                        coordinates: [value.longitud, value.latitud]
                    },
                    //Estilo del marcador
                    properties: {
                        'marker-color': '#003333',
                        'marker-size': 'large',
                        'marker-symbol': 'bicycle'
                    }
                };
```

```
            //El objeto se almacena en el array
            geol[index]=obj;
        });
        if(geol!=null){ initMap(vistaCentral, geol); }
        else{
            //GEOJSON
            var geojson = '';
             geojson = [{
                            type: 'Feature',
                            geometry: {
                                type: 'Point',
                                coordinates: [longitud, latitud]
                            },
                            properties: {
                                'marker-color': '#003333',
                                'marker-size': 'large',
                                'marker-symbol': 'bicycle'
                            }
                        }];
                initMap(vistaCentral, geojson);
            }
    });
```

ANEXO 12: CODIGO MAPA CON MAPBOX

```
   //GENERAR MAPA
   //Funcion com parametros para vista central y con coordenadas
    function initMap(vistaCentral, geojson) {
        //Llamar funciones de mapbox
       L.mapbox.accessToken = 'pk.eyJ1Ijoicm9nZXJyeWVzIiwiYSI6ImNrZHZhY
XNrNDBjeTIyeG84cjRweTU5YXcifQ.J9OVmAObzSp_--B6DA6lgg';
       //Para mapa
       var mapboxTiles = L.tileLayer('https://api.mapbox.com/styles/v1/
mapbox/streets-
v11/tiles/{z}/{x}/{y}?access_token=' + L.mapbox.accessToken, {
           attribution: '© <a href="https://www.mapbox.com/feedback/">
Mapbox</a> © <a href="http://www.openstreetmap.org/copyright">OpenStree
tMap</a>',
           tileSize: 512,
           zoomOffset: -1
       });

       //VISTA CENTRAL CON ZOOM
       var map = L.map('map') //Seleccion del id en el DOM de HTML
               .addLayer(mapboxTiles)
               .setView(vistaCentral, 16); //[latitud, longitud]

       //AGREGA MARQUER
       //Con coordenadas
       var myLayer = L.mapbox.featureLayer().setGeoJSON(geojson).addTo
(map);
    }
```

ANEXO 13: CODIGO PARA ACTIVIDAD INICIAL

```
//PAGINA INICIO
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //Metodo Mapa para ir a la pagina de mostrarMapa
    public void Mapa(View view){
        //Pasara de pagina Inicio a pagina mostrarMapa
        Intent i=new Intent( this, Main2Activity.class);
        //Asignar la URL de la pagina web
        String url = "http://proa7gps.byethost6.com/ubicarBici.php";
        //Enviamos la url por intent como variable "dato_URL"
        i.putExtra("dato_URL", url);
        //Iniciar el paso de pagina
        startActivity(i);
    }
    //Metodo Informacion para ir a la pagina de Informacion
    public void Informacion(View view){
        //Pasara de pagina Inicio a pagina de Informacion
        Intent i=new Intent( this, Main4Activity.class);
        //Asignar la URL de la pagina web
        String url = "http://proa7gps.byethost6.com/comoUsar.php";
        //Enviamos la url por intent como variable "dato_URL"
        i.putExtra("dato_URL", url);
        //Iniciar el paso de pagina
        startActivity(i);
    }
    //Metodo numTelefonico para ir a la pagina de Configuracion
    public void numTelefonico(View view){
        //Pasara de pagina Inicio a pagina de Configuracion
        Intent i=new Intent( this, Main3Activity.class);
        //Iniciar el paso de pagina
        startActivity(i);
    }
}
```

ANEXO 14: CODIGO PARA ACTIVIDAD DE CONFIGURACION

```
//PAGINA CONFIGURACION
public class Main3Activity extends AppCompatActivity {
    private EditText et_numTelefonico;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);

        //SHAREDPREFERENCES
        //Variable receptora del EditText
        et_numTelefonico=(EditText)findViewById(R.id.txt_numTelefonico)
;
        //Declaracion de SharedPreferences como preferences
        SharedPreferences preferences = getSharedPreferences("share", C
ontext.MODE_PRIVATE);
        //Asignacion la variable "numTelefonico" en preferences
        et_numTelefonico.setText(preferences.getString("numTelefonico",
""));

        //Asignar el contenido de TextView a una variable
        String texto = et_numTelefonico.getText().toString();

        //Sin numero telefonico, obtiene un valor
        String SinnumTelefonico = getIntent().getStringExtra("sinNumero
");

        //Comparacion si tiene o no un numero telefonico
        if(texto.length()>=10 && ("verdad".equals(SinnumTelefonico))){
            //Pasara de pagina Configuracion a pagina de mostrar mapa
            Intent intent  = new Intent(Main3Activity.this,Main2Activit
y.class);
            //Asignar el numero TELEFONO que almacena "numTelefonico" a
 la variable "telefono"
            intent.putExtra("telefono", preferences.getString("numTelef
onico",""));
            //URL
            String url = "http://proa7gps.byethost6.com/ubicarBici.php"
;
            //Enviamos la url por intent como variable "dato_URL"
            intent.putExtra("dato_URL", url);
            //Iniciar el paso de pagina
            startActivity(intent);
        }
    }
```

```
    //Metodo para el boton
    public void Guardar(View view){
        //SharedPreferences
        SharedPreferences preferences = getSharedPreferences("share",Co
ntext.MODE_PRIVATE);//
        //creamos una clase "Editor" con la clase SharedPreferences
        SharedPreferences.Editor obj_editor = preferences.edit();
        //Con el "Editor" se puede asignarle el numero telefonico al "n
umTelefonico"
        obj_editor.putString("numTelefonico", et_numTelefonico.getText(
).toString());
        //guardara los datos dentro del SharedPreferences
        obj_editor.commit();
        //Intent
            //TELEFONO
            //Pasara de pagina Configuracion a pagina de mostrar mapa
            Intent intent  = new Intent(Main3Activity.this,Main2Activit
y.class);
            //Asignar el numero TELEFONO que almacena "numTelefonico" a
 la variable "telefono"
            intent.putExtra("telefono", preferences.getString("numTelef
onico",""));
            //URL
            String url = "http://proa7gps.byethost6.com/ubicarBici.php"
;
            //Iniciar el paso de pagina
            intent.putExtra("dato_URL", url);

        //Toast es un mensaje que se muestra por pantalla
        Toast.makeText(this,"Valor almacenado: "+preferences.getString(
"datos",""), Toast.LENGTH_LONG).show();
        startActivity(intent);
    }
}
```

ANEXO 15: CODIGO PARA ACTIVIDAD DE MOSTRAR MAPA

```
//PAGINA MOSTRAR MAPA
public class Main2Activity extends AppCompatActivity {
    //Variables para receptar URL
    WebView wv1;
    WebSettings web_Settings;

    //PERMISOS NUMERO TELEFONICO
    private static final int MY_PERMISSION_REQUEST_CALL_PHONE = 0;
    //Variable para almacenar el numero telefonico
    String phoneNo = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        //VARIABLES PARA PAGINA WEB
        wv1=(WebView)findViewById(R.id.webView);
        //wv1.setOnTouchListener(this);


        web_Settings = wv1.getSettings();
        //Habilita el uso de JavaScript
        web_Settings.setJavaScriptEnabled(true);

        //REDIRECION A PAGINA WEB
        String URL=getIntent().getStringExtra("dato_URL");
        //wv1.setWebViewClient(new WebViewClient());
        wv1.setWebChromeClient(new WebChromeClient());
        wv1.getSettings().setJavaScriptEnabled(true);
        wv1.loadUrl(URL);

        //leer numTelefonico
        String numTelefonico = getIntent().getStringExtra("telefono");

        //Verificar numero telefonico
        if(numTelefonico != null){ //numTelefonico.length() >= 10
            //Se almacena el numero Telefonico en phoneNo
            phoneNo = numTelefonico;
            Toast.makeText(this,"Numero: "+phoneNo,Toast.LENGTH_LONG).s
how();
        }else{//No hay numero telefonico en numTelefonico
            //Pasara de pagina Mostrar mapa a pagina de Configuracion
            Intent intent  = new Intent(Main2Activity.this,Main3Activit
y.class);
```

```
            //Enviar el string "verdad" en la variable "sinNumero"
            intent.putExtra("sinNumero", "verdad");
            //Iniciar el paso de pagina
            startActivity(intent);
        }
    }

    ///////DECLARA VARIABLES DE BUTTON
    public void Llamar(View view){
        //Comprobar si existe "phoneNo"
        if (!TextUtils.isEmpty(phoneNo)) {
            //Asignacion a marcador "dial" para llamar
            String dial = "tel:" + phoneNo;
            //Comprobar permisos
            if (ActivityCompat.checkSelfPermission(Main2Activity.this,
android.Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GR
ANTED) {
                ActivityCompat.requestPermissions(Main2Activity.this, n
ew String[]{Manifest.permission.CALL_PHONE}, MY_PERMISSION_REQUEST_CALL
_PHONE);
            } else { //Se tiene los permisos
                try {
                    //Realiza la llamada
                    startActivity(new Intent(Intent.ACTION_CALL, Uri.pa
rse(dial)));
                } catch (SecurityException e) {
                    e.printStackTrace();
                }
            }
        } else {//Envia un mensaje
            Toast.makeText(Main2Activity.this, "Ingresar un numero tele
fonico", Toast.LENGTH_LONG).show();
        }

    }
}
```

ANEXO 16: CODIGO PARA ACTIVIDAD DE INFORMACION

```
//PAGINA INFORMACION
public class Main4Activity extends AppCompatActivity {
    //Variables para receptar URL
    WebView wv1;
    WebSettings web_Settings;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);

        //VARIABLES PARA PAGINA WEB
        wv1=(WebView)findViewById(R.id.webView);
        web_Settings = wv1.getSettings();
        //Habilita el uso de JavaScript
        web_Settings.setJavaScriptEnabled(true);
        //REDIRECION A PAGINA WEB
        String URL=getIntent().getStringExtra("dato_URL");
        wv1.setWebViewClient(new WebViewClient());
        wv1.loadUrl(URL);
    }
}
```