ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"GEOLOCALIZACIÓN Y SEGURIDAD PERIMETRAL PARA EL CUIDADO DE BICICLETAS DENTRO DE LA ZONA UNIVERSITARIA"

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del Título de:

INGENIERO EN TELEMÁTICA

ALDO FRANCISCO ALVARADO BARCO

BYRON EDUARDO BAJAÑA BARAHONA

GUAYAQUIL – ECUADOR

AÑO: 2016

TRIBUNAL DE EVALUACIÓN

Ing. Washington Velásquez, M. Sc.

PROFESOR EVALUADOR

Ing. Miguel Molina, Mg

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Aldo Francisco Alvarado Barco

Byron Eduardo Bajaña Barahona

RESUMEN

El problema surge de la inseguridad que se tiene al momento de prestar una bicicleta a algún ciclista, ya que en la actualidad nuestro país afronta un nivel muy alto de delincuencia, con lo cual se busca encontrar una solución viable y accesible.

La solución propuesta es un sistema de geolocalización y seguridad perimetral para el control de préstamos de bicicletas dentro de la zona universitaria, teniendo como fuente de energía una batería recargable, la cual será alimentada por el pedaleo del ciclista. Con esto se ayuda en el monitoreo, cuidado y optimización del consumo eléctrico.

Para la elaboración del módulo se necesitó implementar una caja de acrílico que vaya en la bicicleta, la cual contenga la circuitería para cargar la batería recargable y la tarjeta de desarrollo BeagleBone Black junto con el Sistema de Posicionamiento Global Adafruit Ultimate. Se implementó un circuito que aproveche la energía mecánica que se origina al pedalear la bicicleta para convertirla en energía eléctrica, esto se logra a través de un dínamo o alternador que va colocado en la llanta.

Por medio de la tarjeta de desarrollo BeagleBone Black, se realiza el manejo de los datos recibidos por el sensor Adafruit Ultimate, los cuales son almacenados en la base de datos, y luego esta información será observada a través de una aplicación web adaptativa, así el administrador podrá monitorear la ubicación de la bicicleta prestada. En la aplicación web se tendrá el mapa con el perímetro permitido y un marcador que representa a cada módulo, todo esto será en tiempo real con lo cual se tendrá un monitoreo continuo conociendo su ubicación exacta. Este sistema está en capacidad de notificar al administrador vía mensaje de texto y correo electrónico, cuando alguna bicicleta se encuentre fuera del perímetro permitido.

Por tanto, este documento presenta el análisis e implementación de un sistema de geolocalización y seguridad perimetral para el cuidado de bicicletas dentro de la zona universitaria, conteniendo la descripción de los dispositivos que conforman el módulo, y las aplicaciones y recomendaciones utilizadas para el desarrollo del mismo, conjuntamente con la integración de diferentes conocimientos tanto electrónicos como informáticos.

ÍNDICE GENERAL

CAPÍTULO 1

1. DESCRIPCIÓN DEL PROBLEMA.

El procedimiento que se utiliza para el préstamo de bicicletas normalmente no tiene un sistema que permita hacer un seguimiento a las rutas que toman, incluso el registro que se tiene es manual. Uno de los principales motivos porque este proceso no es automatizado en la ESPOL es que las alternativas de geolocalización existentes son costosas y por consiguiente poco accesibles para una institución pública.

Este proyecto surge de la necesidad de diseñar un prototipo de sistema de geolocalización y seguridad perimetral para el control de préstamos de bicicletas dentro de la zona universitaria, además, por medio de una interfaz web adaptativa el administrador podrá saber exactamente la ubicación donde se encuentra cada bicicleta prestada, y este a su vez recibirá alertas mediante correo electrónico o mensajes de texto en un celular en caso de que se encuentre fuera del perímetro permitido.

Otro aspecto importante del prototipo es que el módulo tendrá una batería recargable que será alimentada por el pedaleo. El objetivo es diseñar un circuito que aproveche la energía mecánica que se origina al pedalear la bicicleta para convertirla en energía eléctrica capaz de cargar la batería del módulo.

1.1 Definición del problema.

1.1.1 Planteamiento del Problema

La inexistencia de un sistema de control en la ESPOL en cuanto a los préstamos de bicicletas realizados a estudiantes implica implementar y crear nuevas soluciones a este problema utilizando la tecnología de telecontrol que tenemos disponible.

Actualmente los sistemas de geolocalización para los vehículos de transporte personal como las bicicletas son muy costosos, por lo que la universidad no cuenta con el presupuesto suficiente para invertir en estos dispositivos. Con el uso de las redes de comunicaciones móviles a través

de la tecnología GSM/GPRS, se espera hacer uso eficiente de un sistema de rastreo y alertas para el control de las bicicletas que son proporcionadas en la universidad para estudiantes que deseen practicar deporte en las instalaciones de la institución.

1.1.2 Justificación

El sistema de geolocalización y seguridad perimetral para el control de préstamos de bicicletas, es una idea innovadora que le ofrece al administrador la posibilidad de conocer exactamente la posición de cada bicicleta dentro de la zona universitaria con la recepción de alarmas a través de un mensaje de texto SMS o vía email en caso de que una bicicleta se encuentre fuera del perímetro permitido.

GSM/GPRS nos proporciona algunas ventajas tales como: tiene un menor consumo energético, un tiempo de respuesta instantáneo, permite la comunicación inalámbrica omnidireccional fiable y de dos vías, agilidad de canales para una mejor compatibilidad con otras tecnologías inalámbricas de 2,4 GHz que permiten una instalación y configuración sencilla.

También da un aporte al medio ambiente ya que este módulo tendrá adaptada una batería recargable que será alimentada por el pedaleo de los usuarios. Uno de los motivos más importante que nos lleva a la implementación de este módulo es la sensación de inseguridad que se vive en nuestro país; como también evitar el uso indebido de las bicicletas.

1.1.3 Objetivos

1.1.3.1 Objetivo General

Implementar un sistema que permita controlar el perímetro de ESPOL mediante un módulo de geolocalización para el cuidado de las bicicletas dentro de la zona universitaria.

1.1.3.2 Objetivos Específicos

- Seleccionar los protocolos adecuados que permitan establecer un canal bidireccional de información para la comunicación de los diferentes dispositivos.
- Estudiar e implementar una estrategia de control apropiada para el manejo del módulo de geolocalización.
- Utilizar una batería recargable que sea alimentada por el pedaleo de los usuarios para respaldar el funcionamiento del módulo.
- Implementar una interfaz web adaptativa que controle el perímetro para el monitoreo de las bicicletas.
- Enviar alertas a los administradores mediante dos vías: email y GSM, para tomar las medidas correspondientes.

1.1.4 Limitaciones y Alcance

El dispositivo con el cual se trabajará en este proyecto receptará los datos de posicionamiento global para determinar la ubicación del mismo. Estos datos podrían ser almacenados en una base de datos, los cuales serán utilizados mediante una interfaz web adaptativa para visualizar su localización.

Para determinar la posición precisa donde se encuentra el dispositivo, éste tiene que localizar mínimo 3 satélites que funcionen como puntos de referencia, por lo general siempre hay 8 satélites dentro del campo visual.

Una limitación de estos dispositivos, es su no funcionamiento bajo techo o árboles, por lo cual es preferible colocarlos en lugares abiertos y con amplio campo de visión hacia los satélites.

CAPÍTULO 2

2. MARCO TEÓRICO.

Para el desarrollo del proyecto se ha optado por la utilización de software libre porque se puede lograr grandes proyectos sin necesidad de hacer gastos innecesarios en compra de licencias.

Se desea tener un nivel de autenticación para que cualquier persona no pueda ingresar al sistema, es decir; se necesita acceder por medio de un usuario y contraseña con los permisos necesarios.

En esta parte del capítulo, se detallan los elementos y dispositivos que fueron utilizados para el desarrollo del proyecto.

2.1 BeagleBone Black.

BeagleBone Black es una tarjeta de bajo costo desarrollada por la organización Beagleboard.org, la cual está dirigida a incitar el uso de software y hardware open-source, así como el conocimiento y el intercambio de ideas. [1]

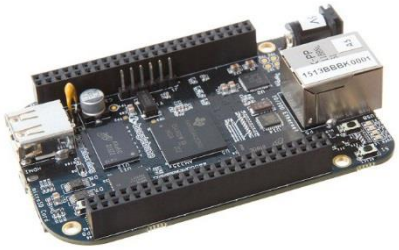

Figura 2.1: BeagleBone Black. [1]

En la figura 2.1 se puede observar una BeagleBone Black, que es una plataforma que corre bajo un sistema operativo Linux con distribución Debian; contiene entradas y salidas de propósito general que cuentan con diversas funciones, entre las cuales se encuentran (I/O Digitales, Entradas Analógicas, Salidas con modulación por ancho de pulso, soporte para I2 &

SPI). Además, cuenta con un puerto Ethernet y un puerto USB 2.0 para la comunicación con otros dispositivos. [2]

Este microcontrolador cuenta con la opción de poder trabajar independientemente por medio de un monitor, es decir; al encender la BeagleBone Black veremos que entramos directamente al escritorio del sistema operativo. Si se requiere trabajar con la tarjeta BeagleBone Black por SSH, podemos conectarla mediante USB a una computadora. [2]

2.2 Sistema de Posicionamiento Global Adafruit Ultimate

Potente y compacto módulo de posicionamiento global, tiene una alta sensibilidad (rastreo de -165 dBm), funciona con 5V y puede ser pinchado directamente sobre un protoboard sin mayor complicación. Puede alcanzar una velocidad de 10Hz (programable) y soporta hasta 66 canales con un máximo de 22 satélites simultáneamente. Está construido a partir del chip MTK3339, un GPS de alta calidad y grandes prestaciones con antena integrada y que consume tan solo 20mA. [3]

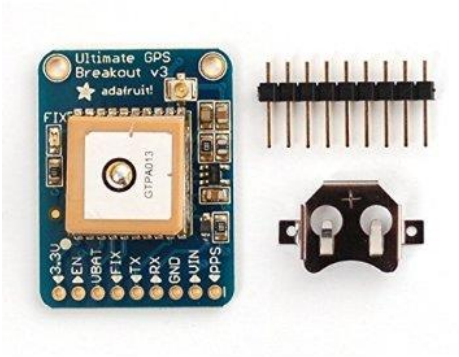

Figura 2.2: Adafruit Ultimate GPS. [3]

En la figura 2.2 se muestra el dispositivo GPS con sus respectivos pines y una vincha que se coloca en la parte posterior en caso de ser ubicada en una posición específica.

Tiene unas características muy interesantes, como por ejemplo un regulador interno de 3.3V muy eficiente que permite alimentarlo de 3.3 a 5Vdc sin problema. Tiene un LED indicador que parpadea cuando el dispositivo se encuentra buscando satélites y una memoria interna que permite guardar 16 horas de información. [3]

2.3 Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. [4] La elegante sintaxis de Python junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. El intérprete de Python y la extensa biblioteca estándar están a libre disposición en el sitio web de Python, y puede distribuirse libremente. Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables.

Python es fácil de usar, ofreciendo mucha más estructura y soporte para programas grandes. Por otro lado, Python ofrece mucho más chequeo de error que C, y siendo un lenguaje de muy alto nivel, tiene tipos de datos incorporados como arreglos de tamaño flexible y diccionarios. Python te permite separar tu programa en módulos que pueden reusarse en otros programas y viene con una gran colección de módulos estándar que pueden usarse como base.

Python es un lenguaje interpretado, lo cual puede ahorrarte mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. El intérprete puede usarse interactivamente, lo que facilita experimentar con características del lenguaje, escribir programas descartables, o probar funciones cuando se hace desarrollo de programas de abajo hacia arriba. Python permite escribir programas compactos y legibles. [5]

Python es extensible: si ya sabes programar en C es fácil agregar una nueva función o módulo al intérprete. Una vez que estés realmente entusiasmado, puedes enlazar el intérprete Python en una aplicación hecha en C y usarlo como lenguaje de extensión o de comando para esa aplicación. La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos, como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos. [6]

2.4 Django

Django proporciona una serie de características que facilitan el desarrollo rápido de páginas web orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores programen y creen vistas para las áreas de administración de la página, Django provee una aplicación incorporada para gestionar los contenidos, que puede incluirse como parte de cualquiera y que puede administrar varias vistas hechas con Django a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno.

Entre las características de Django tenemos un sistema extensible de plantillas basado en etiquetas, incluyendo traducciones incorporadas de la interfaz de administración, documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones). Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. [7]

2.5 Adafruit PowerBoost 500 C

Es la fuente de energía perfecta para un proyecto portátil. La salida está ajustada para que sea 5.2V en vez de 5.0V para aquellos casos en los cuales los cables son largos, alto empate y la adición de un diodo en la salida si lo desea. Los 5.2V son seguros para todos esos controladores que son accionados con 5V como Arduino, Raspberry Pi, o BeagleBone [8]; tal como se muestra en la figura 2.3 a continuación:

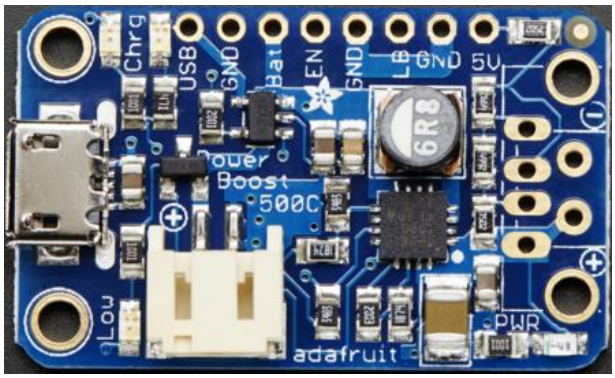

Figura 2.3: Adafruit PowerBoost 500 C. [8]

El PowerBoost 500 C tiene como principal elemento un convertidor TPS61090. Este chip convertidor tiene algunas características interesantes como la detección de batería baja, interruptor interno 2A, la conversión síncrona, excelente rendimiento y el funcionamiento de alta frecuencia de 700KHz. Funcionamiento síncrono significa que se puede desconectar por completo la salida conectando el pin Enable a tierra. Interruptor interno 2A significa que puede obtener 500mA desde un mínimo de 1.8 V, 750 mA de 2 baterías de NiMH o alcalinas, y al menos 1000 mA de un LiPoly / batería de iones de litio de 3.7 V o 3 NiMH / alcalinas. El indicador de batería baja es un LED que se ilumina en rojo cuando la caída de tensión esté por debajo de 3.2 V, optimizados para el uso más común de Li-Po / Li-Ion de la batería. [8]

2.6 MySQL

El sistema de base de datos operacional MySQL es hoy en día uno de los más importantes en lo que hace al diseño y programación de base de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas por usuarios del medio. MySQL se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo. Una de las características más interesantes es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a diferentes necesidades y requerimientos. Por otro lado, MySQL es conocida por desarrollar alta velocidad en la búsqueda de datos e información, a diferencia de sistemas anteriores. [9]

MySQL está disponible para múltiples plataformas, sin embargo, las diferencias con cualquier otra plataforma son prácticamente nulas, ya que la herramienta utilizada en este caso es el cliente mysql-client, que permite interactuar con un servidor MySQL (local o remoto) en modo texto. De este modo es posible realizar todos los ejercicios sobre un servidor instalado localmente o, a través de Internet, sobre un servidor remoto. [10]

Algunas características importantes que se puede mencionar son su velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento; su bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema; la facilidad de configuración e instalación; soporta gran variedad de Sistemas Operativos, baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor. [10]

2.7 Mini Ralink RT5370

Este módulo Wi-Fi alcanza tasas de transmisión inalámbrica de hasta 150 Mbps. Soporta varios tipos de encriptación como WEP, WPA/WPA2 y

WPA-PSK/WPA2-PSK y algunos sistemas operativos como XP, VISTA, WIN7, MAC, LINUX. Cuenta con una antena desmontable y su protocolo operativo de red es CSMA/CA con ACK. Su tamaño reducido es muy conveniente e ideal para poder ser transportado sin problema. [11]

Figura 2.4: Mini Ralink RT5370 Wi-Fi Adapter. [11]

En la figura 2.4 se muestra el módulo Wi-Fi Ralink RT5370 en un diseño muy compacto y con la ventaja de tener una antena desmontable.

2.8 Proteus

Para realizar el simulado del circuito se escogió utilizar el programa Proteus porque es uno de los mejores ya que tiene la mayoría de elementos o componentes que permiten obtener una buena prueba del circuito a implementar.

Proteus es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción. Con Proteus las fases de prueba no suponen la necesidad de volver a construir nuevos prototipos con el ahorro de costos y tiempo que ello supone. Cualquier modificación que se realice en nuestro esquema, podrá ser reenviado desde ISIS a ARES donde aparecerán resaltados los cambios que se hayan producido. De esta forma la modificación y rediseño de nuestra placa se realizará de forma mucho más simple y segura. [12]

CAPÍTULO 3

3. IMPLEMENTACIÓN Y DESARROLLO

En este capítulo se analiza y diseña la solución para que el módulo de la bicicleta tenga una batería recargable y sea alimentada por el pedaleo de los usuarios, ahorrando tiempo y dinero en la fuente de energía que se va a utilizar para el mismo.

Además, se desarrolla una interfaz web adaptativa que controle el perímetro de las bicicletas y envíe alertas a los administradores, mediantes dos vías, email y GSM, en caso de que alguna sobrepase los límites establecidos. En la interfaz se muestra un mapa del perímetro de ESPOL junto con la ubicación de la bicicleta dentro del mapa.

3.1 Diagrama General

La *BeagleBone Black* al recibir los datos de la ubicación por medio del *Adafruit Ultimate GPS*, los procesa por medio de *Python* y son subidos o almacenados a la base de datos, estos son usados por la aplicación para mostrar la ubicación de la bicicleta en la interfaz web.

Figura 3.1: Diagrama General.

En la figura 3.1 se puede observar el diagrama general del proyecto, el cual se indica el módulo colocado en la bicicleta que envía la información necesaria a la base de datos; para luego ser presentada al administrador por medio de la aplicación web.

3.2 Diagrama del Módulo en la Bicicleta

En este módulo se usa una dínamo de bicicleta, el cual en realidad es un alternador. Al momento de pedalear, los alternadores generan corriente alterna y llegan a entregar hasta 10 Voltios. Se necesita corriente continua y bajar ese voltaje a 5 Voltios para alimentar el Adafruit PowerBoost, por lo cual la dínamo va conectado a un circuito que realice este proceso. En la figura 3.2 se muestra el diagrama del módulo en la bicicleta, el cual será detallado a continuación:

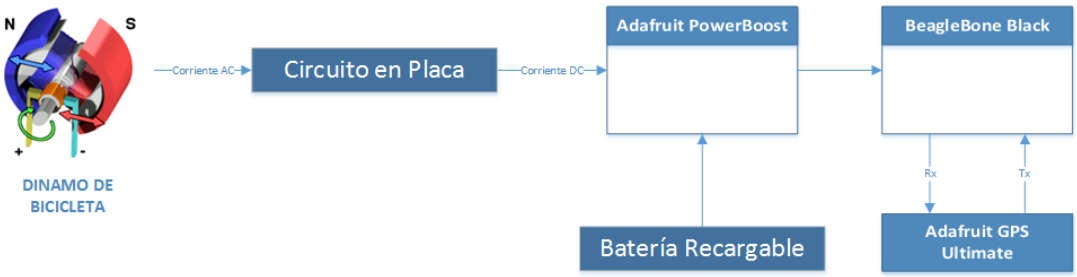

Figura 3.2: Diagrama del Módulo en la Bicicleta.

Los polos del dínamo que entregan corriente alterna van acoplados al circuito en placa obteniendo corriente continua a la salida y se conecta al *PowerBoost 500C*, el cual es un circuito cargador de batería incorporado capaz de mantener el módulo funcionando incluso mientras se recarga la batería. El *PowerBoost 500C* es la fuente de energía perfecta para esta clase de proyectos.

El *PowerBoost 500C* alimenta la *BeagleBone Black*, la cual está conectada con el *Adafruit Ultimate GPS*, estos dos permanecen comunicándose e intercambiando información a cada instante.

3.3 Diseño del Circuito en Proteus

El circuito simulado con sus respectivos elementos conectados y los valores de voltaje y corriente entregados se muestran en la figura 3.3 a continuación:

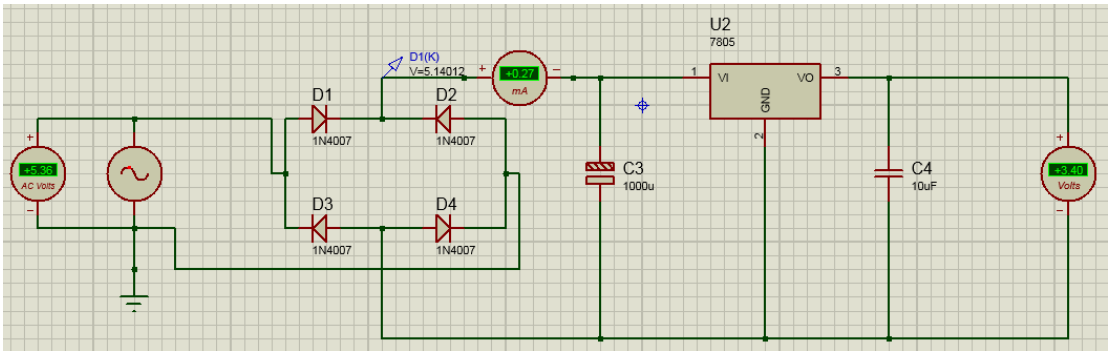

Figura 3.3: Circuito Simulado.

En la primera parte del circuito, los polos del dínamo pasan por un puente rectificador, el cual son 4 diodos ordenados de una forma en particular como se observa en la figura 3.4, esto nos da corriente siempre positiva y no alternada pero con constantes caídas de voltaje.

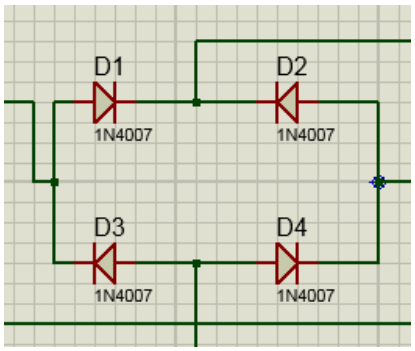

Figura 3.4: Puente Rectificador.

Para solucionar este problema, se colocó un condensador, como se muestra en la figura 3.5, el cual se carga y descarga muy rápido, pero en un tiempo necesario con ello compensando la variación de voltaje y logrando una corriente mucho más estable.

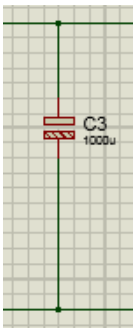

Figura 3.5: Condensador de 1000uF.

Ahora solo se tiene que bajar el voltaje, este trabajo lo hace el regulador, el cual baja el voltaje de entrada variable a uno de 5 Voltios. En la figura 3.6 se tiene el regulador de voltaje.

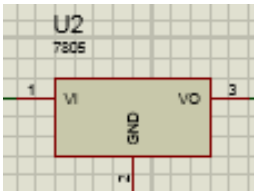

Figura 3.6: Regulador de Voltaje LM7805.

Finalmente se agrega un condensador a la salida para filtrar los ruidos, como se aprecia en la figura 3.7.

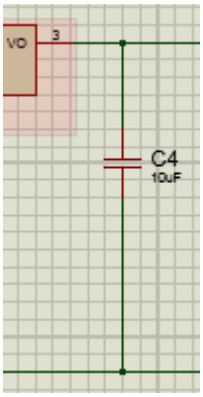

Figura 3.7: Condensador de 10uF.

3.4 Diseño del Circuito Impreso

En el programa Proteus, se tiene la ventaja de simular el circuito y a la vez diseñar la placa del mismo, con lo cual se ahorra tiempo y trabajo.

En la figura 3.8 se muestra el diseño del circuito en PCB creado en Proteus:

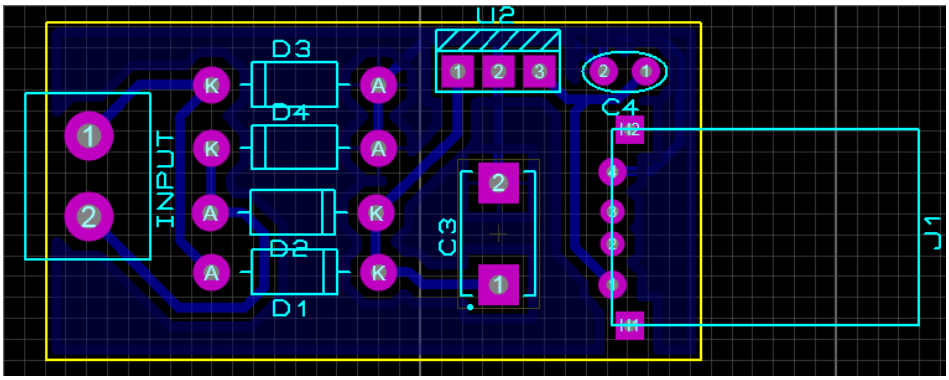

Figura 3.8: Circuito en PCB.

En este diseño se busca colocar los componentes de manera que se reduzca el espacio de uso para que no sea muy grande la placa, así mismo tomar en cuenta algunos parámetros o requerimientos como el grosor de los caminos.

3.5 Adquisición de Datos

La captura de datos del sensor *Adafruit Ultimate GPS* son trabajados por la tarjeta de desarrollo *BeagleBone Black* y enviados a la base de datos por medio de un script en Python, el cual es un algoritmo que consiste en receptar todas las tramas que son enviadas por el sensor *GPS* que contiene los datos específicos de latitud y longitud para la localización de la bicicleta, estos datos son guardados en la base de datos previo a realizar la conexión. Los datos que se insertan en la tabla son: Latitud, Longitud, Módulo y Fecha.

Para obtener los resultados antes descritos se utiliza el siguiente código:

```
if NMEA1_array[0]=='$GPRMC':
        self.timeUTC=NMEA1_array[1][:-8]+':'+NMEA1_array[1][-8:-
6]+':'+NMEA1_array[1][-6:-4]
        self.latDeg=NMEA1_array[3][:-7]
        self.latMin=NMEA1_array[3][-7:]
        self.latHem=NMEA1_array[4]
        self.lonDeg=NMEA1_array[5][:-7]
        self.lonMin=NMEA1_array[5][-7:]
        self.lonHem=NMEA1_array[6]
        self.knots=NMEA1_array[7]
```

3.6 Detalle de la Trama

La trama NMEA es receptada por el sensor *Adafruit Ultimate GPS*, la cual es tomada desde el satélite y es necesaria para conocer el punto exacto donde se encuentra el dispositivo. La trama NMEA recibida es detallada en la figura 3.9 a continuación.

NMEA 2.0

Name	Garmin	Magellan	Lowrance	SiRF	Notes:
GPAPB	N	Y	Y	N	Auto Pilot B
GPBOD	Y	N	N	N	bearing, origin to destination - earlier G-12's do not transmit this
GPGGA	Y	Y	Y	Y	fix data
GPGLL	Y	Y	Y	Y	Lat/Lon data - earlier G-12's do not transmit this
GPGSA	Y	Y	Y	Y	overall satellite reception data, missing on some Garmin models
GPGSV	Y	Y	Y	Y	detailed satellite data, missing on some Garmin models
GPRMB	Y	Y	Y	N	minimum recommended data when following a route
GPRMC	Y	Y	Y	Y	minimum recommended data
GPRTE	Y	U	U	N	route data, only when there is an active route. (this is sometimes bidirectional)
GPWPL	Y	Y	U	N	waypoint data, only when there is an active route (this is sometimes bidirectional)

Figura 3.9: Trama NMEA.

De todas estas tramas recibidas la que utilizamos son las GPRMC, ya que estas nos proporcionan la información mínima necesaria para la ubicación del dispositivo.

Esta trama tiene 10 campos separados por comas y está descompuesta de la siguiente manera:

Primer Campo: Indica el tipo de trama precedido por las siglas GP, en este caso es la trama RMC.

Segundo Campo: Este campo contiene la hora exacta en la que fue capturada la posición según GMT.

Tercer Campo: Este campo nos muestra el estado del satélite, estos pueden ser A (active) o V (void), el estado A nos indica que SI se están recibiendo datos, mientras que el estado V nos dice que NO se ha encontrado ninguno del cual se pueda recibir información.

Cuarto, Quinto, Sexto y Séptimo Campo: Estos campos son los datos de geolocalización, el cuarto campo es la latitud, el quinto la longitud; ambos dados en grados, minutos y orientación (N, S, E, W). El sexto campo representa la velocidad respecto al suelo en nudos mientras que el séptimo indica el ángulo de seguimiento en grados, esto nos da aproximadamente hacia qué dirección nos estamos dirigiendo.

Octavo Campo: Representa la fecha actual en la que se captura la posición.

Noveno Campo: Representa el Checksum o Suma de Comprobación de Transferencia de Datos, está dado en formato Hexadecimal y siempre comienza con un *.

A continuación podemos observar un ejemplo de la trama GPRMC.

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
Where:   RMC          Recommended Minimum sentence C
         123519       Fix taken at 12:35:19 GMT
         A            Status A=active or V=Void.
         4807.038,N   Latitude 48 deg 07.038' N
         01131.000,E  Longitude 11 deg 31.000' E
         022.4        Speed over the ground in knots
         084.4        Track angle in degrees True
         230394       Date - 23rd of March 1994
         003.1,W      Magnetic Variation
         *6A          The checksum data, always begins with *
```

3.7 Modelo Relacional de Base de Datos

En la figura 3.10 se presenta el esquema de base de datos que se utilizó según los requerimientos del proyecto.

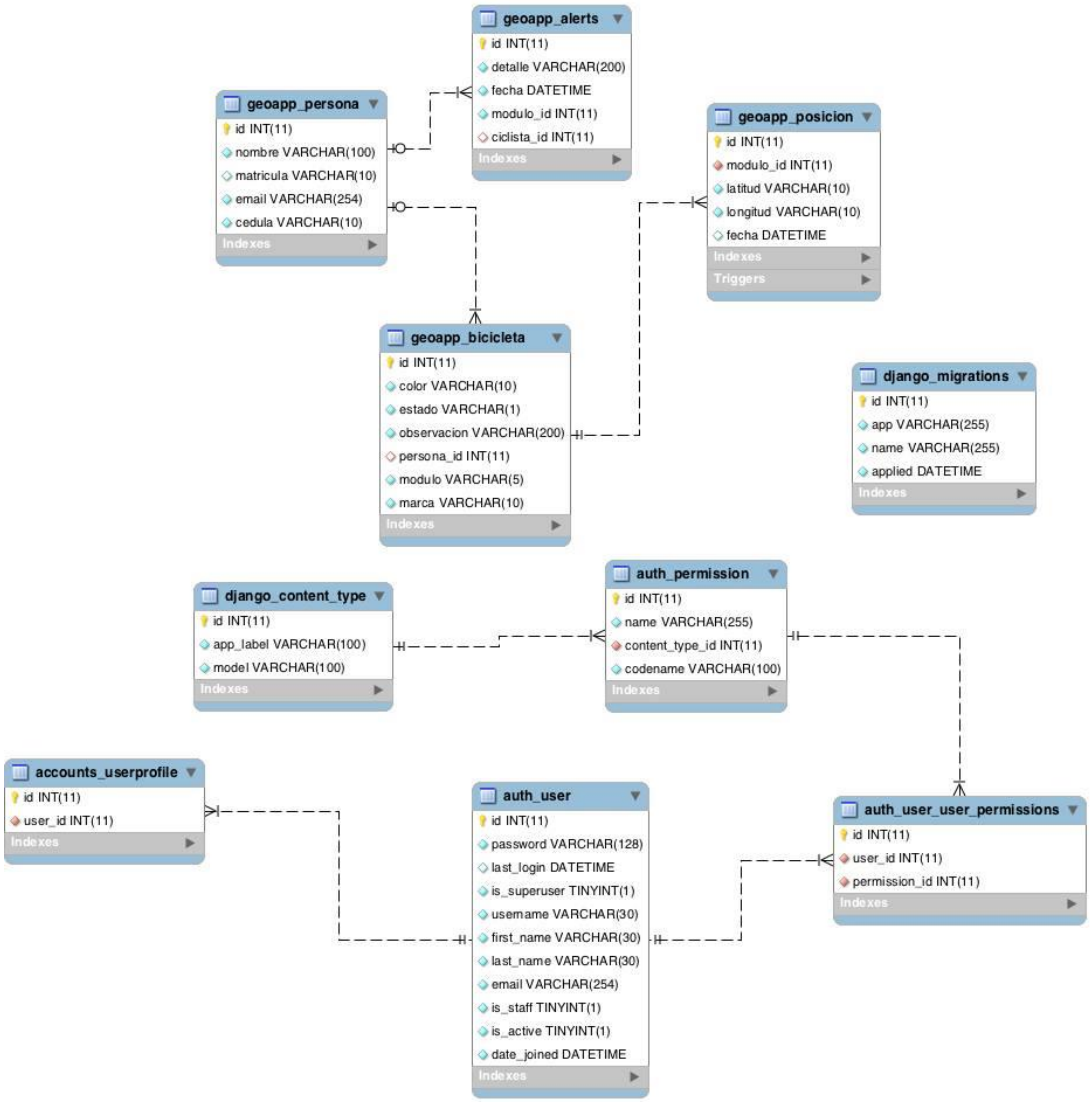

Figura 3.10: Modelo Relacional de Bases de Datos.

Se crearon tablas para representar cada uno de los diferentes objetos que se iban a necesitar, y se los relacionó con sus tablas correspondientes para asociar los datos respectivos.

La tabla *accounts_userprofile* se asocia a *auth_user* para el inicio de sesión de los usuarios creados por la aplicación. En la tabla 1 y 2 podemos ver los campos que contienen respectivamente.

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
user_id	INT(11)	Identificador del usuario al que se asocia en la tabla auth_user.

Tabla 1: Tabla accounts_userprofile

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
password	VARCHAR(128)	Contraseña cifrada.
last_login	DATETIME	Fecha de último ingreso de usuario.
is_superuser	TINYINT(1)	Contiene 1 si el usuario tiene privilegios de súper usuario y 0 en caso contrario.
username	VARCHAR(30)	Nombre del usuario con el que ingresa.
first_name	VARCHAR(30)	Nombres de la persona dueña del user.
last_name	VARCHAR(30)	Apellidos de la persona dueña del user.
email	VARCHAR(254)	Email del usuario.
is_staff	TINYINT(1)	Contiene 1 si el usuario tiene privilegios de administrador y 0 en caso contrario.
is_active	TINYINT(1)	Contiene 1 si es un usuario activo 0 en caso contrario.
date_joined	DATETIME	Fecha en la que se registró el usuario.

Tabla 2: Tabla auth_user

La tabla *geoapp_persona* contiene la información de los ciclistas registrados. Sus campos son mostrados en la tabla 3 a continuación.

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
nombre	VARCHAR(100)	Nombre de la persona.
matricula	VARCHAR(10)	Matrícula en caso de que sea estudiante.
email	VARCHAR(254)	Email del ciclista.
cedula	VARCHAR(10)	Número de cédula del ciclista.

Tabla 3: Tabla geoapp_persona

La tabla *geoapp_bicicleta* tiene los datos de las bicicletas registradas como se aprecia en la tabla 4.

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
color	VARCHAR(10)	Color de la bicicleta.
estado	VARCHAR(1)	L=libre, P=prestado, F=fuera del perímetro, E=bicicleta con inconveniente.
observacion	VARCHAR(200)	Observaciones que tenga la bicicleta.
persona_id	INT(11)	Identificador del ciclista.
modulo	VARCHAR(5)	Módulo de geolocalización asociado.
marca	VARCHAR(10)	Apellidos de la persona dueña del user.

Tabla 4: Tabla geoapp_bicicleta

La tabla *geoapp_posicion* contiene los datos de posición de los módulos. En la tabla 5 tenemos sus respectivos campos.

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
modulo_id	INT(11)	Identificador de la bicicleta.
latitud	VARCHAR(10)	Posición latitud.
longitud	VARCHAR(10)	Posición Longitud.
fecha	DATETIME	Fecha de captura de posición.

Tabla 5: Tabla geoapp_posicion

La tabla *geoapp_alerts* tiene las alertas que se han generado cuando un ciclista sale del perímetro permitido. Los campos que se llenan son los mostrados en la tabla 6 a continuación.

CAMPOS	TIPO DE DATO	DESCRIPCION
Id	INT(11)	Identificador de la tabla.
detalle	VARCHAR(200)	Detalle de la alerta.
fecha	DATETIME	Fecha en la que se registró la alerta.
modulo_id	INT(11)	Identificador de la bicicleta.
ciclista_id	INT(11)	Identificador de la persona.

Tabla 6: Tabla geoapp_alerts

3.8 Posicionamiento y Perímetro

Para poder visualizar la posición de cada bicicleta en un mapa se utilizó *Google Maps JavaScript API* que nos brinda este servicio de forma gratuita.

Se creó un mapa en la página principal con la clase Map, estableciéndose como centro el corazón de ingenierías en el campus Gustavo Galindo de la ESPOL. Una vez creado el mapa, se dibujó un polígono definiendo el perímetro permitido para el tránsito de las bicicletas en el campus; éste polígono será nuestra referencia para monitorear si una bicicleta se encuentra dentro de los límites establecidos y así poder enviar alertas a los administradores, en caso de que sea necesario, éste fue dibujado con una de las clases dadas por la API de GOOGLEMAPS, llamada *"google.maps.LatLngBounds"*, que representa un rectángulo en coordenadas geográficas que recibe como parámetro dos posiciones; la primera es el punto máximo al suroeste de nuestro perímetro, mientras que el segundo es la posición máxima al noreste del mismo.

Esto se logra con las líneas de código que se encuentran a continuación:

```
var bounds = new google.maps.LatLngBounds(
        new google.maps.LatLng(-2.151133, -79.968832),
        new google.maps.LatLng(-2.142960, -79.958458)
      );
var mapProp = {
center:bounds.getCenter(),
zoom: 13,
mapTypeId: google.maps.MapTypeId.ROADMAP,
disableDefaultUI:true
};
map = new google.maps.Map(document.getElementById("map"),mapProp);
map.fitBounds(bounds);
```

Al tener creado el mapa y el perímetro de tránsito se procede a crear los marcadores con la clase *"google.maps.Marker"* por medio de una función que será invocada cada segundo. Estos marcadores nos indican la posición de las bicicletas que se encuentran prestadas, estas posiciones son obtenidas de la

base de datos y son actualizadas cada segundo, dando la sensación al usuario de que el marcador está en movimiento. En la función se realiza la validación de que la posición recibida se encuentre dentro del rango permitido, en caso de que esto suceda se visualizará una alerta en la página y serán enviadas notificaciones vía email y mensajes de texto a los administradores para que tomen las medidas correspondientes.

Para colocar el marcador en el mapa se utiliza la siguiente función mostrada en el siguiente código:

```
function placeMarker() {
  var location =new google.maps.LatLng({{ posic.latitud }},{{
posic.longitud }})
  var marker = new google.maps.Marker({
    map: map,
    position: location,
    draggable: true });
  if (bounds.contains(location) == false) {
    alert("Marcador fuera de rango permitido");
  }
}
```

En la figura 3.11 se puede apreciar la página principal con las respectivas opciones y el mapa con el marcador respectivo mostrando la ubicación de una bicicleta dentro del perímetro establecido.

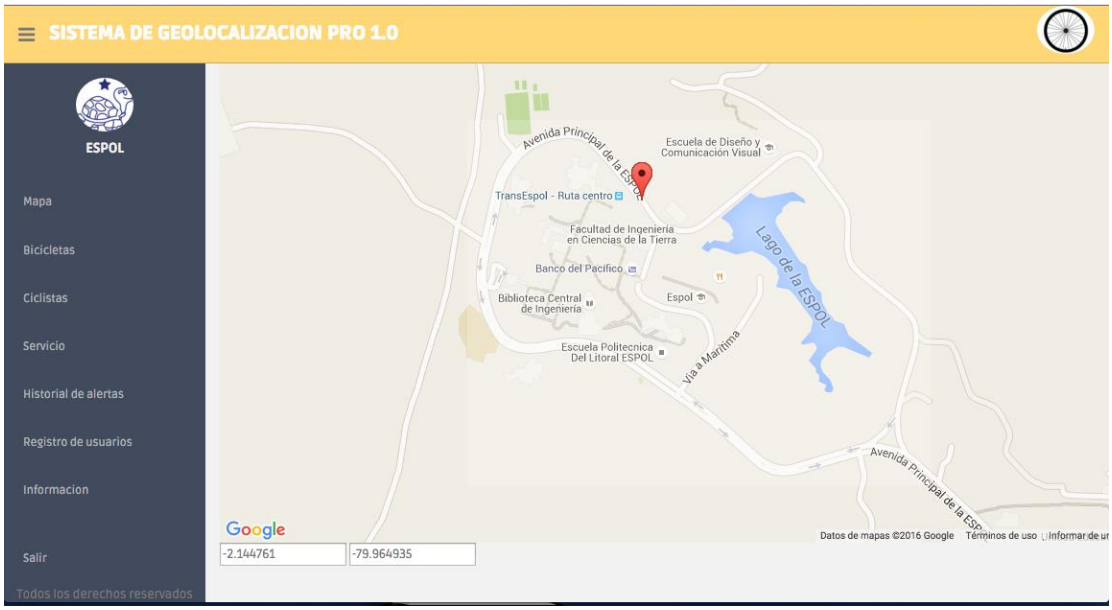

Figura 3.11: Página Principal.

3.9 Registro de Estudiantes vía WebService Espol

Para poder realizar el registro de los estudiantes usando el WebService de la ESPOL, se realiza una petición http mediante el método POST, enviando el número de matrícula del estudiante que desea el registro.

La aplicación envía al servidor esta información por medio de una función y se obtienen los datos requeridos en un formato XML en el cual podemos recorrer y encontrar la información que nos interesa como es: nombre del estudiante, su número de cédula y su email. En caso de no ser estudiante, se lo podrá registrar al ciclista tomando sus datos teniendo como principal campo su número de cédula.

A continuación se muestra el código de la función mencionada:

```
def webservice(matricula):
   server_addr = "ws.espol.edu.ec"
   service_action = "/saac/wsandroid.asmx/wsInfoEstudiante"
   body = "codigoEstudiante="+matricula
```

```
    request = httplib.HTTPConnection(server_addr)
    request.putrequest("POST", service_action)
    request.putheader("Content-Type", "application/x-www-form-urlencoded")
    request.putheader("Cache-Control", "no-cache")
    request.putheader("Pragma", "no-cache")
    request.putheader("SOAPAction", "http://" + server_addr +
service_action)
    request.putheader("Content-Length", str(len(body)))
    request.endheaders()
    request.send(body)
    response = request.getresponse().read()
    dom = parseString(response)
    return dom
```

En la figura 3.12 tenemos la pantalla donde se realiza la búsqueda del estudiante con su número de matrícula por medio del WebService.

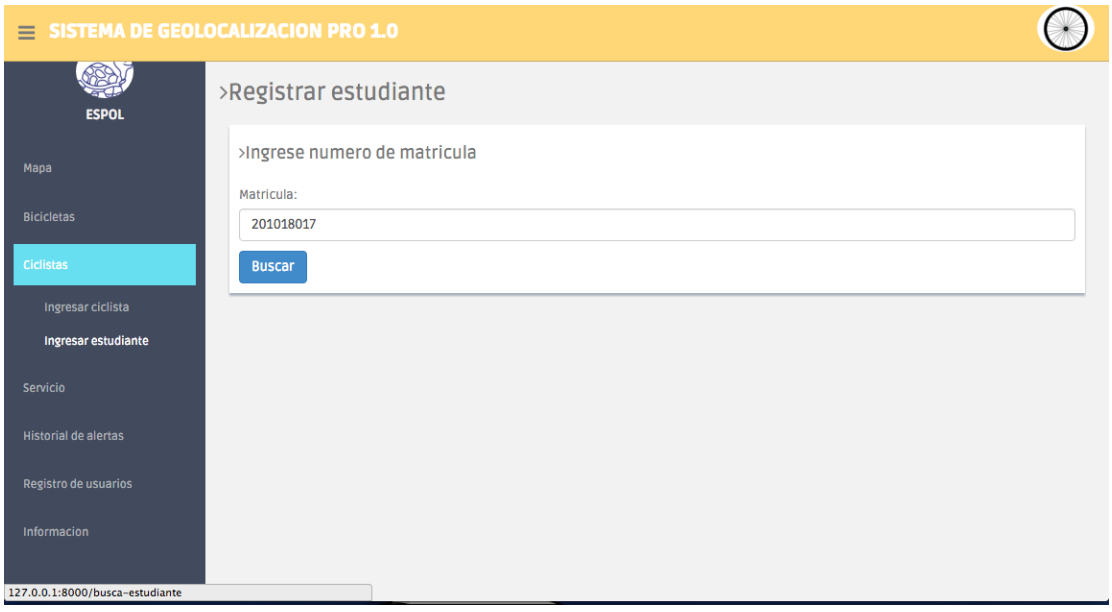

Figura 3.12: Página del Ingreso de Estudiantes.

3.10 Registro de Bicicletas

Para poder realizar el registro de una bicicleta, se llena un formulario con los datos ingresados por el usuario y son guardados directamente en la base de datos con un estado ‘L’ (Libre).

El código para dicho registro lo tenemos a continuación:

```
def registro_bicicleta_view(request):
if request.method == 'POST':
       # Si el method es post, obtenemos los datos del formulario
        form = RegistroBicicleta(request.POST, request.FILES)
        if not request.user.is_authenticated():
              return   redirect('%s?next=%s'   %   (settings.LOGIN_URL,
       request.path))

       if request.user.is_authenticated():
              if form.is_valid():
              cleaned_data = form.cleaned_data
              marca = cleaned_data.get('marca')
              color = cleaned_data.get('color')
              modulo = cleaned_data.get('modulo')
              bicicleta = Bicicleta()
              bicicleta.marca=marca
              bicicleta.color=color
              bicicleta.modulo=modulo
              bicicleta.save()
              return render(request, 'accounts/gracias.html')
else:
 form = RegistroBicicleta()
context = {'form': form}
return render(request, 'formulario-bici.html', context)
```

En la figura 3.13 observamos la pantalla de ingreso de una bicicleta.

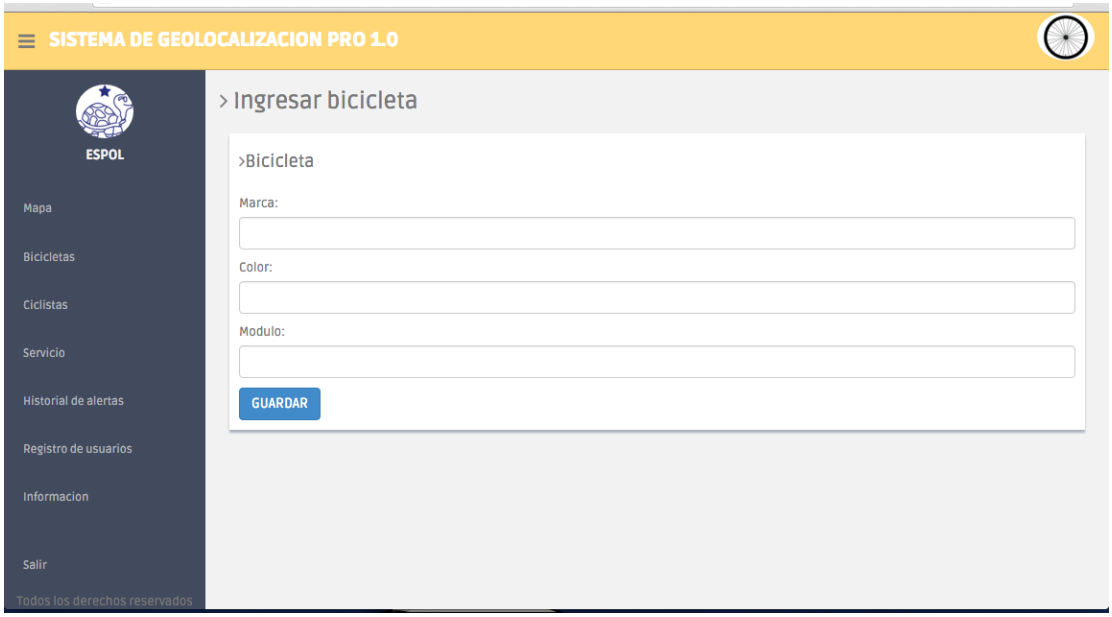

Figura 3.13: Página del Registro de Bicicletas.

3.11 Préstamo de Bicicletas

Para el préstamo de bicicletas, se realiza una búsqueda en la base de datos por medio del número de matrícula en caso de ser estudiante, o por número de cédula en caso de no pertenecer a la universidad; con lo cual se traen los datos de la persona que desea realizar el préstamo. Luego de obtener los datos pertinentes, se procede a seleccionar una bicicleta libre y se presiona el botón *Prestar.*

La implementación de esta opción consiste en 2 pasos, la primera es el botón *Buscar* que nos presentará el formulario con la información de la persona que realiza el préstamo utilizando el método POST, mientras que el botón *Prestar* cumple con la función de almacenar los datos del ciclista junto con el módulo prestado.

En la figura 3.14 tenemos la pantalla donde se realiza el préstamo de una bicicleta a un estudiante.

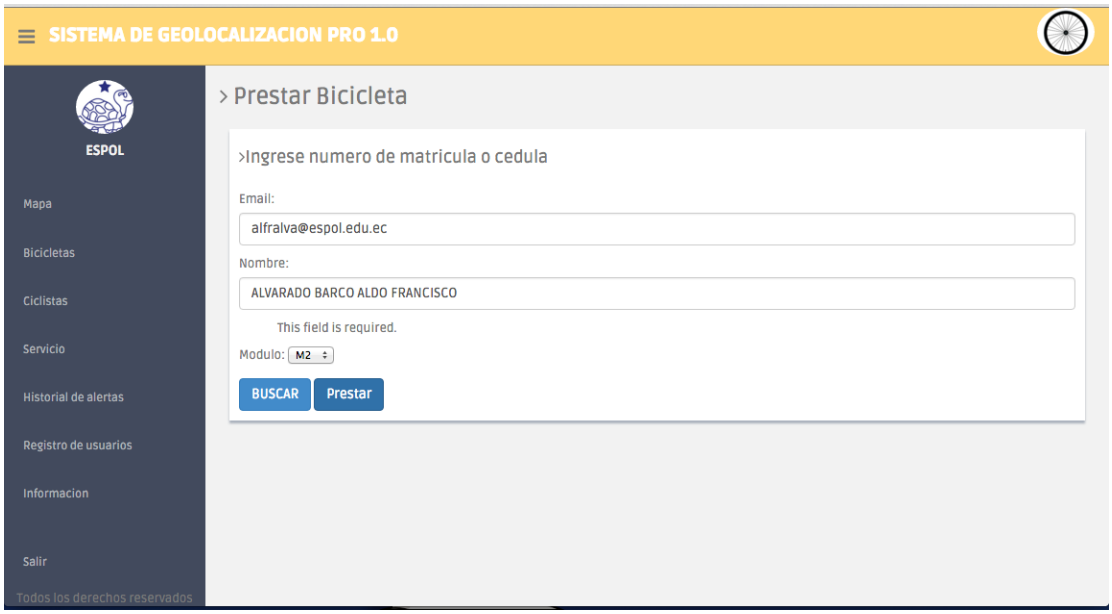

Figura 3.14: Página del Préstamo de Bicicletas.

3.12 Envío de Alertas

Las alertas de bicicletas fuera del perímetro permitido son enviadas a través de una función, que recibe como parámetro el identificador de la bicicleta y el nombre del ciclista; para así poder concatenar los datos con el mensaje predefinido.

Para lograr enviar las alertas vía SMS o mensaje de texto, se creó una cuenta en *twilio.com* que es una página que brinda los servicios de envíos de SMS por medio de la web, para esto se deben utilizar sus propias librerías y tener una cuenta activa. Por otro lado, para el envío de emails, se creó una cuenta en Gmail para que a través de ella se envíen las notificaciones a las personas correspondientes.

El algoritmo de la función consiste en buscar en la base de datos los números celulares y los emails a los que se les deben de enviar las notificaciones, luego

de obtenerlos, se realizan las conexiones a los diferentes servidores enviando un mensaje con el formato: *“El ciclista Bajaña Barahona Byron Eduardo salió del perímetro permitido en la bicicleta M1”*

Para poder enviar los SMS y mails de alerta, se utilizó el siguiente código:

```
…
server = smtplib.SMTP('smtp.gmail.com',587)
server.starttls()
server.login(fromaddr, password)
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
…
accountSid = "SID"
authToken = "Token"
twilioClient = TwilioRestClient(accountSid, authToken)
myTwilioNumber = "TWnumber"
destCellPhone = "AdminNumber"
myMessage = twilioClient.messages.create(body = Mensaje a enviar,
from_=myTwilioNumber, to=destCellPhone)
```

3.13 Historial de Alertas

En el momento que una bicicleta salga del perímetro, se guarda en la base de datos un registro con: el nombre del ciclista, el identificador del módulo, la fecha de la alerta y el detalle de la misma. Es muy importante que el administrador pueda ver el historial completo de las alertas que se han generado para poder identificar a los usuarios reincidentes, por esto se creó una opción que muestra una tabla dinámica con todos los registros de las alertas que se encuentren en la base de datos, esta tabla es ordenable, ya sea por nombre del ciclista, número de módulo o fecha, también existe un cuadro de búsqueda en el cual se podrá escribir una palabra referente a la alerta que está buscando.

Para la implementación de este cuadro, se crea una aplicación que permita definir objetos pertenecientes a una tabla, como son: columnas y filas, incluso se necesitó asociar widgets para poder realizar los filtros antes mencionados.

Existen algunas propiedades que ya están definidas en el API *Bootstrap-DataTables* que es utilizada en esta opción. Se utilizaron librerías como *DjangoJSONEncoder* y *BaseListView* dentro de las vistas de la aplicación que generan la tabla para poder darle la dinámica que se necesita.

Para obtener estos resultados se utilizó el código mostrado a continuación:

```
class FeedDataView(JSONResponseMixin, BaseListView):
    """
    The view to feed ajax data of table.
    """

    context_object_name = 'object_list'

    def get(self, request, *args, **kwargs):
        if not hasattr(self, 'token'):
            self.token = kwargs["token"]
        self.columns = TableDataMap.get_columns(self.token)

        query_form = QueryDataForm(request.GET)
        if query_form.is_valid():
            self.query_data = query_form.cleaned_data
        else:
            return self.render_to_response({"error": "Query form is
invalid."})
        return BaseListView.get(self, request, *args, **kwargs)
```

3.14 Algoritmo de Posicionamiento de Bicicletas en Mapa

Lo primero es inicializar el mapa en la primera llamada, es decir; en el momento que se abre la página se crea el recuadro que contiene el perímetro permitido para el tránsito de las bicicletas dentro de la zona universitaria, luego se buscan aquellas en la base de datos que se encuentran en estado P de prestada o F de fuera del perímetro para que puedan ser creados sus respectivos marcadores y así poder ser visualizados en el mapa. Estos marcadores son guardados en un arreglo por medio de *JavaScript*.

Una vez inicializado el mapa, se realiza un proceso de búsqueda y posicionamiento de marcadores cada segundo para así poder observar en tiempo real el movimiento de cada bicicleta prestada. Las bicicletas con estado L de libre no serán mostradas en el mapa. Lo siguiente es tomar la última posición de cada bicicleta seleccionada previamente y verificar que no se encuentre fuera del perímetro definido.

Si la posición de la bicicleta se encuentra dentro del perímetro permitido, solamente se actualiza la posición del marcador en el mapa. En caso de que se encuentre fuera del perímetro, existen dos casos; dependerá del estado de la bicicleta.

Si la bicicleta se encuentra en estado F, esto indica que la notificación de alerta ya fue enviada a los administradores y por lo tanto no será enviada nuevamente, sin embargo el marcador de dicha bicicleta seguirá visualizando y actualizándose su posición en el mapa conforme se vaya moviendo.

Por otro lado, si la bicicleta se encuentra en un estado P, mediante *Ajax* se cambia el estado de la bicicleta a ‘F’ y se registra la alerta en la base de datos con un estado ‘I’, de ingresado, para que a su vez sea llamado el script de notificaciones, el cual busca en la tabla de alertas los registros con estado ‘I’ y proceda con el envío de los correos electrónicos y mensajes de texto a los administradores.

Al instante de ser enviadas las notificaciones se cambia el estado de las alertas procesadas a ‘E’, de enviada, y de la misma manera el marcador se mantendrá visible y activo.

La figura 3.15 muestra el algoritmo de cómo funciona el posicionamiento de las bicicletas en el mapa con la implementación de las notificaciones de alertas del sistema de monitoreo.

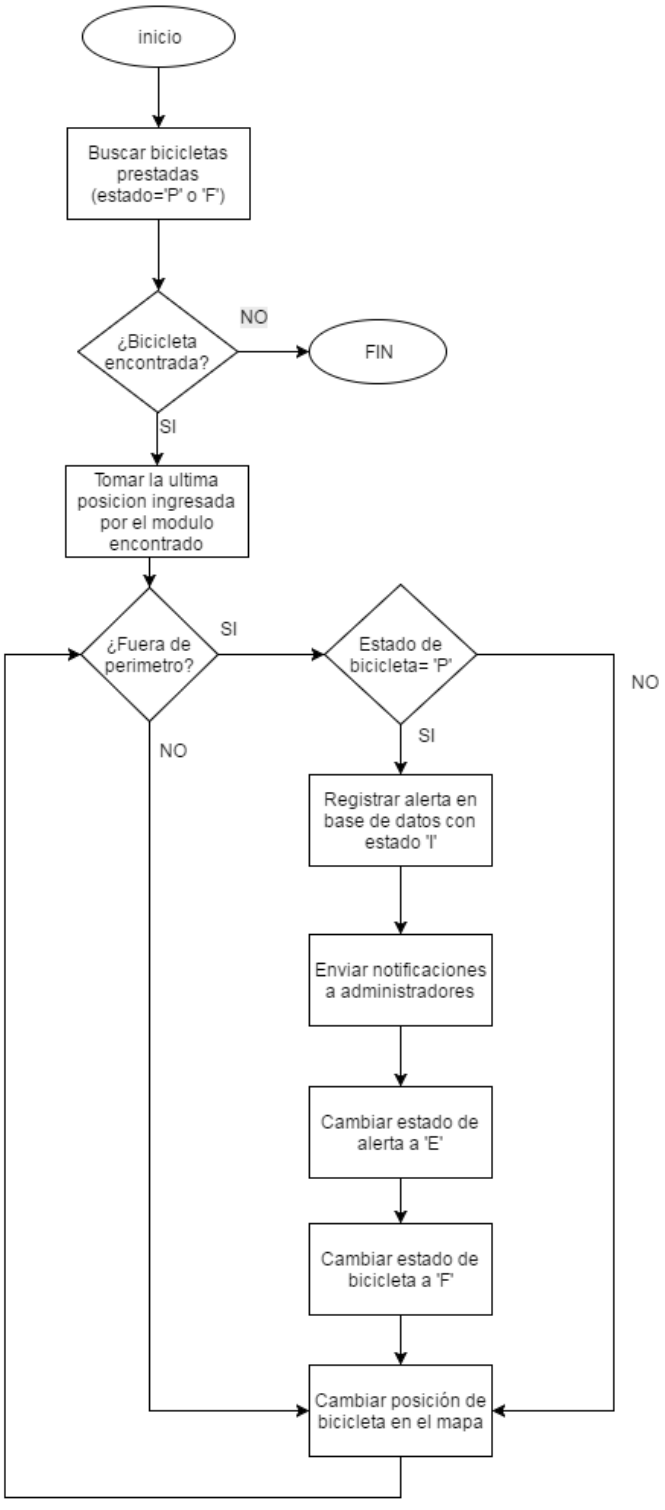

Figura 3.15: Algoritmo de Posicionamiento de Bicicletas en Mapa.

3.15 Algoritmo del Sistema de Posicionamiento Global

La figura 3.16 muestra el algoritmo utilizado para capturar la geolocalización.

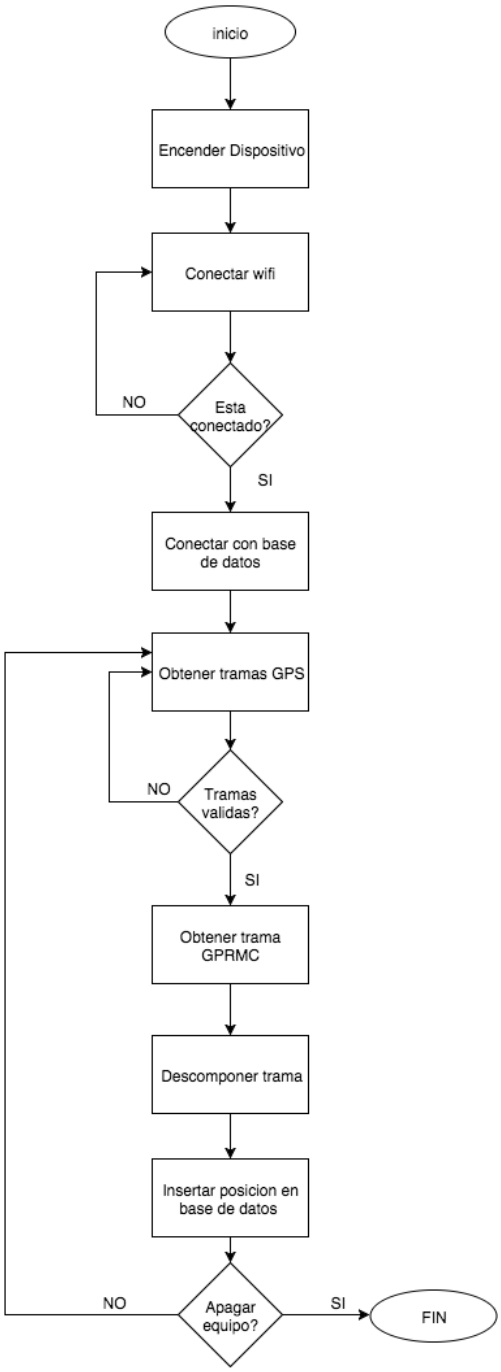

Figura 3.16: Algoritmo GPS.

Al encender la *BeagleBone Black* se ejecuta automáticamente y de manera ininterrumpida un script que verifica que el dispositivo se encuentre conectado a cierta red inalámbrica, este proceso se realiza todo el tiempo mientras esté encendido el módulo. En caso de no realizar la conexión, seguirá intentándolo hasta tener éxito. Luego de conectarse a la red, se procederá a la conexión con la base de datos. Una vez realizada dicha conexión, se obtendrán los datos adquiridos por el sensor *Adafruit Ultimate GPS*.

En caso de que las tramas recibidas por el sensor no sean válidas, se seguirá realizando la petición de tramas hasta obtener un posicionamiento válido. Cuando ya se esté recibiendo la posición sin inconvenientes, se procederá a registrar la información en la tabla adecuada en la base de datos. Este proceso se repetirá hasta que el dispositivo sea apagado.

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS.

En el presente capítulo se presentan 3 escenarios con el módulo propuesto para verificar los resultados y comprobar el funcionamiento correcto de la solución.

4.1 Escenario 1

En este primer escenario se realiza el procedimiento para cargar la batería recargable. El pedaleo del ciclista hace trabajar al dínamo colocado en la llanta de la bicicleta generando energía, la cual pasa a través del circuito regulándola, para así poder cargar la batería por medio del *Adafruit PowerBoost* 500C.

4.1.1 Resultados

A continuación en la figura 4.1 se muestra el resultado del primer escenario.

Figura 4.1: Resultados del Escenario 1.

Como se puede apreciar al momento de pedalear y pasar por el circuito, llegamos al Adafruit *PowerBoost 500C*, donde al encenderse el led naranja nos indica que la batería está cargando. Una vez que la carga esté completa, se encenderá el led verde.

4.2 Escenario 2

En este segundo escenario se va a realizar el proceso de monitoreo al momento de prestar una bicicleta a algún ciclista. El procedimiento consiste en hacer el seguimiento del módulo circulando dentro del perímetro permitido, para luego observar los resultados y comprobar el correcto funcionamiento.

4.2.1 Resultados

A continuación en las figuras 4.2, 4.3 y 4.4 se muestran los resultados del segundo escenario.

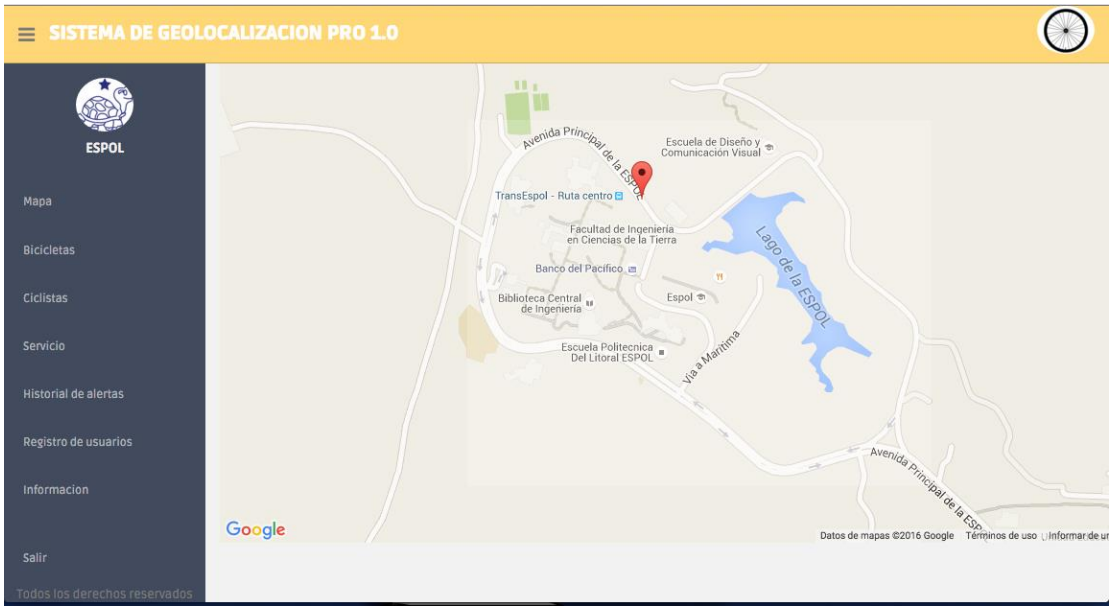

Figura 4.2: Resultados del Escenario 2.

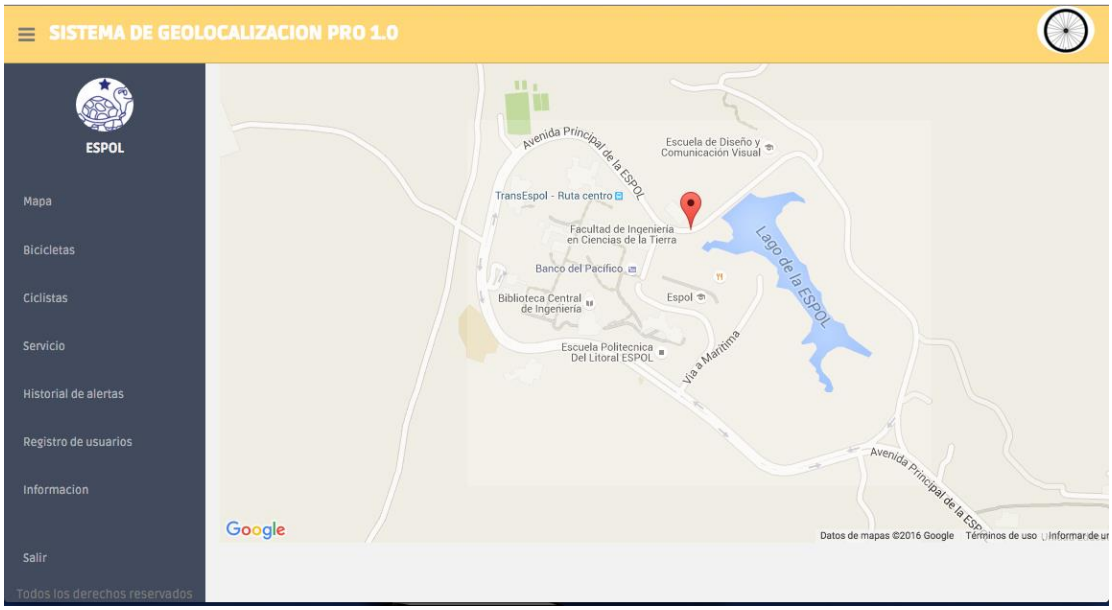

Figura 4.3: Resultados del Escenario 2.

```
[03/Feb/2016 07:37:09] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:10] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:12] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:13] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:15] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:17] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:18] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:20] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:21] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:22] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:23] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:24] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:25] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:27] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:29] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:31] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:33] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:35] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:37] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:39] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:41] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:43] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
[03/Feb/2016 07:37:45] "GET /busqueda_ajax/?id=1 HTTP/1.1" 200 108
```

Figura 4.4: Resultados del Escenario 2.

Se visualiza el movimiento del marcador en el mapa comprobándose que se está realizando la conexión a la base de datos constantemente para obtener la última posición enviada por el sensor *Adafruit Ultimate GPS* en el módulo de la bicicleta. También podemos observar las peticiones periódicas de datos para posicionamiento en el mapa.

4.3 Escenario 3

En este tercer escenario se comprueba el sistema de seguridad mientras se realiza el monitoreo de las bicicletas y alguna se encuentre circulando fuera del perímetro permitido. Se busca comprobar que la notificación de alerta sea enviada a los administradores.

4.3.1 Resultados

A continuación en las figuras 4.5, 4.6, 4.7 y 4.8 se observan los resultados del tercer escenario.

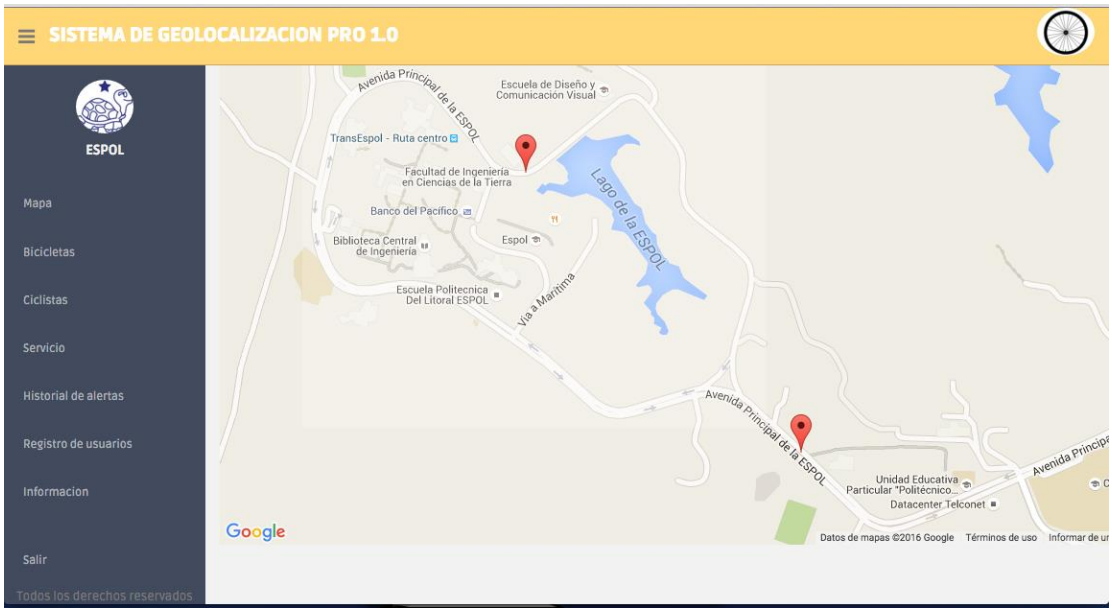

Figura 4.5: Resultados del Escenario 3.

Figura 4.6: Resultados del Escenario 3.

Figura 4.7: Resultados del Escenario 3.

Figura 4.8: Resultados del Escenario 3.

Se puede notar que la bicicleta se encuentre fuera del perímetro permitido, por lo cual el sistema procede a enviar las notificaciones de alerta a los administradores. Otra observación importante es que la alerta generada fue registrada en el sistema con éxito.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Se ha utilizado y comprobado la gran velocidad de procesamiento de datos de la tarjeta de desarrollo *BeagleBone Black*, y se confirmó su efectividad con lo cual da un mejor desarrollo de la aplicación.
2. Se evidenció que fue de gran ayuda utilizar el lenguaje de programación *Python* para el manejo y presentación de los datos, por su fácil relación entre la *BeagleBone Black* y la base de datos.
3. *Django* se adaptó fácilmente al trabajo con *Python* y *MySQL*, con lo cual se consiguió compatibilidad al momento de juntar todos los elementos necesarios de trabajo.
4. Se implementó el módulo de geolocalización con éxito, obteniendo la ubicación de la bicicleta en tiempo real por medio de la aplicación web y actualizando la posición cada segundo dando resultados positivos.
5. Se logró alimentar la batería recargable por medio del pedaleo de los usuarios y se observó que no es necesario andar a gran velocidad para iniciar la carga. Además se apreció que el proceso de carga no interfiere con la alimentación de la tarjeta de desarrollo *BeagleBone Black*.
6. El sistema de seguridad responde efectivamente cuando una bicicleta sale del perímetro permitido, enviando las notificaciones de alerta a los administradores en un periodo de tiempo muy corto.

Recomendaciones

1. Se recomienda apagar la *BeagleBone Black* cuando la bicicleta sea devuelta, ya que mantenerla encendida desgasta su vida útil como cualquier tarjeta de desarrollo, así mismo se aconseja desconectar la batería recargable.
2. Para lograr un monitoreo eficaz se aconseja trabajar con una red inalámbrica de gran alcance y velocidad.

3. Se debe tomar en cuenta que para obtener datos válidos del sensor *Adafruit Ultimate GPS*, la antena tiene que apuntar hacia arriba con una vista clara del cielo.

GLOSARIO

GPS: Es un sistema que permite determinar en todo el mundo la posición de un objeto con una precisión de hasta centímetros.

GPRMC: Es una trama que contiene información acerca de la ubicación del dispositivo GPS y ciertas variables calculadas por los satélites GPS.

NMEA: Es una organización de comercio electrónico estadounidense que establece estándares de comunicación entre electrónica marina.

SMS: Es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos, conocidos como mensajes de texto, entre teléfonos móviles.

API: Es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

BIBLIOGRAFÍA

[1] G. Coley, "eLinux.org," [En línea]. Disponible en: http://elinux.org/Beagleboard:BeagleBoneBlack. [Último acceso: 22 Diciembre 2015].

[2] mauricioesa, "Mecatrónica UASLP," 628 Febrero 2014. [En línea]. Disponible en: https://mecatronicauaslp.wordpress.com/2014/02/28/introduccion-a-beaglebone/. [Último acceso: 22 Diciembre 2015].

[3] L. Ada, "Adafruit Industries," [En línea]. Disponible en: https://learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf. [Último acceso: 26 Diciembre 2015].

[4] "Python Software Foundation," [En línea]. Disponible en: http://docs.python.org.ar/tutorial/2/. [Último acceso: 10 Enero 2016].

[5] "Pyhton Software Foundation," [En línea]. Disponible en: http://docs.python.org.ar/tutorial/2/appetite.html. [Último acceso: 10 Enero 2016].

[6] M. Á. Álvarez, "Desarrollo Web," [En línea]. Disponible en: http://www.desarrolloweb.com/articulos/1325.php. [Último acceso: 11 Enero 2016].

[7] "Wikipedia," [En línea]. Disponible en: https://es.wikipedia.org/wiki/Django_(framework). [Último acceso: 27 Diciembre 2015].

[8] L. Ada, "Adafruit Industries," [En línea]. Disponible en: https://learn.adafruit.com/adafruit-powerboost-500-plus-charger/overview. [Último acceso: 28 Diciembre 2015].

[9] "Definicion ABC," [En línea]. Disponible en: http://www.definicionabc.com/tecnologia/mysql.php. [Último acceso: 10 Diciembre 2015].

[10] Á. Castillo, "Club Ensayos," [En línea]. Disponible en: https://www.clubensayos.com/Temas-Variados/Electronica/1433977.html. [Último acceso: 11 Enero 2016].

[11] "Amazon," [En línea]. Disponible en: http://www.amazon.com/niceEshop-Wireless-802-11-Adapter-Antenna/dp/B008IZQCGK/ref=sr_1_4?s=hi&ie=UTF8&qid=1453261875&sr=8-4&keywords=rt5370. [Último acceso: 15 Enero 2016].

[12] "Hubor," [En línea]. Disponible en: http://hubor-proteus.com/proteus-pcb/proteus-pcb/2-proteus.html. [Último acceso: 17 Enero 2016].