UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Sistema de localización automática aplicado a la flota de vehículos de recolección y transporte de Residuos Sólidos Municipales del GADML

TRABAJO DE TITULACIÓN

AUTOR: Calderón Sanmartín, Nilder Agustín

DIRECTOR: Calderón Córdova, Carlos Alberto, Ing.

LOJA - ECUADOR

2016

2016

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero.
Carlos Alberto Calderón Córdova.
DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: “Sistema de localización automática aplicado a la flota de vehículos de recolección y transporte de Residuos Sólidos Municipales del GADML" realizado por “Calderón Sanmartín Nilder Agustín”; ha sido revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, septiembre de 2016

f) . . . . . .

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo, Calderón Sanmartín Nilder Agustín declaro ser autor del presente trabajo de titulación: Sistema de localización automática aplicado a la flota de vehículos de recolección y transporte de Residuos Sólidos Municipales del GADML, de la Titulación de Ingeniero en Electrónica y Telecomunicaciones, siendo Calderón Córdova Carlos Alberto director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f) .....
Calderón Sanmartín Nilder Agustín
1104460835

DEDICATORIA

Dedico esta tesis a mis padres, quienes han sido un apoyo incondicional en los buenos y malos momentos, por saber que cuento con ellos en cualquier circunstancia.

A mi esposa por su apoyo, dedicación y paciencia, por ser un pilar fundamental en mi vida.

A mi hijo Agustín Alejandro y futura hija, por todos los momentos llenos de felicidad desde que llego a mi vida.

A mis hermanos, Jhon, Henrry, Carlos y Ronar.

A todos mis familiares y amigos en general, por su motivación y apoyo para alcanzar esta meta muy importante.

AGRADECIMIENTO

Durante el transcurso de la vida son muchas las personas que llegaron a formar parte importante de mi formación tanto personal como académica. Los más importantes sin duda alguna, mis padres por inculcarme valores y principios que me han servido hasta hoy para cumplir mis metas. Gracias a ustedes por su constante apoyo para lograr esta meta tan importante.

A mis amigos, con los que he sobrellevado buenos y malos momentos. A ustedes por siempre estar incondicionalmente a mi lado, mi apoyo para no desmayar y salir adelante. Gracias por todo.

Así mismo quiero extender mi agradecimiento a cada uno de los docentes de la UTPL que han sido parte de mi formación académica, principalmente al Ing. Carlos Alberto Calderón Córdova por entregarme su tiempo, empeño, infinita paciencia y dedicación para culminar este trabajo de investigación.

Y sobre todo a ti Dios, por guiarme y ayudarme a sobrellevar obstáculos que se presentaron durante el transcurso de la vida, pero principalmente, por haber puesto en mi camino a personas tan maravillosas que gracias a su compañía y consejos estoy cumpliendo una meta.

¡Gracias!

ÍNDICE DE CONTENIDOS

CARATULA *I*

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN *II*

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS *III*

DEDICATORIA *IV*

AGRADECIMIENTO *V*

ÍNDICE DE CONTENIDOS *VI*

LISTA DE FIGURAS *IX*

LISTA DE TABLAS *X*

RESUMEN EJECUTIVO *1*

ABSTRACT *2*

INTRODUCCIÓN *3*

CAPÍTULO I *5*

ANÁLISIS DE LA SITUACIÓN ACTUAL DE LA RECOLECCIÓN DE RSM Y LOS REQUERIMIENTOS DEL SISTEMA DE MONITOREO **5**

1.1. Gestión de desechos sólidos. 6

1.1.1. Clasificación de los residuos sólidos municipales y peligrosos. 6

1.1.1.1. Residuos sólidos municipales (RSM). 7

1.1.1.2. Residuos sólidos especiales (RSE). 7

1.1.1.3. Residuos peligrosos (RP). 8

1.1.2 Cadena de Gestión de RSM. 8

1.2. Ventajas de la implementación de un sistema de monitoreo. 10

1.3. Diagrama funcional del Sistema de monitoreo del RT-RSM. 11

1.3.1. Propuesta del sistema de monitoreo. 12

1.3.1.1. Unidad de monitoreo. 13

1.3.1.2. Transmisión de datos. 13

1.3.1.3. Plataforma software de monitoreo. 14

1.4. Análisis de costos de sistemas comerciales de rastreo. 15

1.5. Modelo de inversión para el diseño y desarrollo del sistema. 17

1.5.1. Inversión a realizar por la UTPL. 17

1.5.2. Inversión a realizar por el GAD Municipal de Loja. 18

CAPÍTULO II *19*

ANÁLISIS DE LA TECNOLOGÍA UTILIZADA **19**

2.1. Generalidades. 20

2.2. Módulo de procesamiento. 20

2.2.1. Arduino Uno. 20

2.2.2. Raspberry Pi. 22

2.2.3. Comparación de las características técnicas de las tecnologías Arduino y Raspberry Pi. 23

2.3. Módulo GPS. 24

2.4. Módulos GSM/GPRS. 26

2.5. Selección del Shield GPS GPRS y sus componentes. 26
2.5.1. Shield GPRS GPS quaband para Arduino. 27
2.5.2. Chip GPS GPRS SIM908. 28
2.5.3. Antena GSM Conector UFL. 29
2.5.4. Antena GPS Conector UFL. 30
2.6. Selección de batería. 30
2.7. Plataforma de monitoreo. 32
2.7.1. Servidor web. 32
2.7.2. Servidor de Aplicaciones. 33
2.7.3. Servidores de base de datos. 35
2.7.4. Pack de software stacks. 35

CAPÍTULO III ***36***

FABRICACIÓN DEL SISTEMA DE MONITOREO DE VEHÍCULOS RECOLECTORES DE RSM DE BAJO COSTO **36**
3.1. Introducción. 37
3.3. Arquitectura del sistema. 37
3.3.1. Módulo de procesamiento. 38
3.3.2. Módulo de adquisición y envío de datos. 40
3.3.3. Módulo servidor. 42
3.3.3.1 Unidad conexión. 42
3.3.3.2 Base de datos. 44
3.3.3.3 Aplicación web. 45
3.4. Fabricación del sistema de monitoreo para vehículos recolectores de RSM de bajo costo. 47

CAPÍTULO IV ***49***

PRUEBAS Y ANÁLISIS DE RESULTADOS **49**
4.1. Introducción. 50
4.2. Pruebas de conexión nodo remoto y servidor. 50
4.3. Pruebas de campo del sistema. 51
4.3.1. Instalación del prototipo en el vehículo recolector. 51
4.3.2. Pruebas del sistema de monitoreo. 52
4.4. Análisis de resultados. 59
4.4.1. Análisis de la ruta de prueba. 59
4.4.2. Análisis de la ruta 2 diurna. 61

CONCLUSIONES ***63***

RECOMENDACIONES ***64***

BIBLIOGRAFÍA ***65***

ANEXOS ***68***

ANEXO A **69**
Rutas de recolección de RSM en la ciudad de Loja 69

ANEXO B **73**
Desarrollo del script para el funcionamiento de la unidad central de procesamiento del prototipo. 73

ANEXO C **77**
Scripts para el funcionamiento del módulo conexión 77
ANEXO D **85**
Script de la base de datos en MySQL 85
ANEXO E **87**
Scripts para el funcionamiento del servidor web 87

LISTA DE FIGURAS

Figura 1.1. Cadena de suministro de residuos sólidos. 8
Figura 1.2. Rutas de recolección de RSM en la ciudad de Loja. 10
Figura 1.3. Diagrama funcional del sistema de monitoreo de los vehículos de RT-RSM. 11
Figura 2.1. Placa Arduino Uno R3. 21
Figura 2.2. Raspberry Pi modelo B. 23
Figura 2.3. Estructura del protocolo. 25
Figura 2.4. Shield de GSM GPRS (SIM908-SIM900). 27
Figura 2.5. Módulo GPS/GSM. 28
Figura 2.6. Antena GSM. 29
Figura 2.7. Antena GPS. 30
Figura 2.8. Batería 32
Figura 2.9. Arquitectura de funcionamiento de un servidor web. 33
Figura 2.10. Arquitectura de funcionamiento de un servidor de aplicaciones. 34
Figura 3.1. Arquitectura del sistema de monitoreo de la flota de vehículos de RT-RSM. 37
Figura 3.3. Conexión serial vista lateral y frontal. 38
Figura 3.2. Diagrama de flujo del algoritmo principal. 39
Figura 3.4. Jumper para el uso del GPS. 40
Figura 3.5. Configuración del módulo GPRS 41
Figura 3.6. Prototipo de monitoreo 41
Figura 3.7. Diagrama de flujo de la programación de la aplicación. 42
Figura 3.8. Configuración de la interfaz del módulo conexión. 43
Figura 3.9. Interfaz gráfica de la aplicación. 43
Figura 3.10. Datos que recibe el módulo conexión. 44
Figura 3.11. Características de la base de datos. 45
Figura 3.12. Aplicación web. 46
Figura 3.13. Recorrido del vehículo recolector. 46
Figura 3.14. Integración del hardware del prototipo. 47
Figura 3.15. Integración del software para el sistema de monitoreo. 48
Figura 3.16. Diagrama especifico del sistema de monitoreo para los vehículos recolectores de RSM. 48
Figura 4.1. IP Pública y puertos habilitados. 50
Figura 4.2. Interfaz gráfica. 51
Figura 4.3. Instalación del prototipo en el vehículo recolector de RSM. 52
Figura 4.4. Conexión del prototipo con el servidor. 52
Figura 4.5. Información registrada en la base de datos MySQL. 53
Figura 4.6. Distancia error entre el dispositivo de monitoreo y el GPS Garmin Monterra. 54
Figura 4.7. Trazo de la ruta del GPS Monterra. 56
Figura 4.8. Trazo de la ruta del prototipo en la aplicación web. 57
Figura 4.9. Ruta 2 diurna trazada por el GPS Monterra. 58
Figura 4.10. Ruta 2 diurna trazada por el prototipo en la aplicación web. 58
Figura 4.11. Zonas obscuras de la ruta 2 diurna trazada en la aplicación web. 59
Figura 4.12. Latitud con respecto al tiempo de la ruta de prueba. 60
Figura 4.13. Longitud con respecto al tiempo de la ruta de prueba. 60
Figura 4.14. Error absoluto con respecto al tiempo de la ruta de prueba. 60
Figura 4.15. Latitud con respecto al tiempo de la ruta 2 diurna. 61
Figura 4.16. Longitud con respecto al tiempo de la ruta 2 diurna. 61
Figura 4.17. Error absoluto con respecto al tiempo de la ruta 2 diurna. 62

LISTA DE TABLAS

Tabla 1.1. Comparativa de los sistemas comerciales de rastreo aplicables a los vehículos de recolección de RSM. 16
Tabla 1.2. Inversión a realizar por la UTPL para el desarrollo del proyecto. 17
Tabla 1.3. Inversión a realizar por el GADML, para la puesta en marcha del proyecto. 18
Tabla 1.4. Costos asumidos por el GADML, para el manteamiento mensual del proyecto. 18
Tabla 2.1. Características técnicas del Arduino Uno. 21
Tabla 2.2. Características técnicas del Raspberry Pi modelo B. 22
Tabla 2.3. Comparación de Arduino Uno y Raspberry Pi. 23
Tabla 2.4. Características electrónicas del Shield. 28
Tabla 2.5. Características electrónicas del Chip GPS/GSM. 28
Tabla 2.6. Características de Antena GSM (conector UFL). 29
Tabla 2.7. Características de Antena GPS. 30
Tabla 2.8. Características de amperaje, voltaje, y potencia requerida del almacenamiento mínimo de dispositivos a alimentar. 30
Tabla 2.9. Características de la Batería. 32
Tabla 3.1. Pines de conexión entre el Arduino Uno y el shield. 38
Tabla 3.2. Gastos generados para el desarrollo del prototipo. 47
Tabla 4.1. Datos tomados de los dispositivos GPS. 54

RESUMEN

En base a problemas comunes como incumplimiento de horarios y de cobertura de rutas se evidencia la necesidad de que exista un monitoreo de los vehículos recolectores de residuos sólidos municipales (RSM) del Gobierno Autónomo Descentralizado Municipal de Loja (GADML), en la UTPL surge la iniciativa del presente trabajo de investigación e innovación cuyo objetivo es desarrollar un sistema de monitoreo de bajo costo que permita inspeccionar el recorrido del vehículo recolector en base a la información de localización geográfica en función del tiempo.

El sistema diseñado e implementado está formado por el módulo de adquisición de datos, módulo de procesamiento y módulo servidor. La aplicación del presente sistema de localización automática de vehículos (AVL) es monitorear los parámetros en un vehículo recolector del GADML durante el recorrido de una determinada ruta de recolección de RSM, los parámetros del recorrido son: ubicación (latitud, longitud), hora, fecha y velocidad, datos almacenados en tiempo real en una base de datos MySQL, lo cual permite generar reportes bajo demanda, de esta manera el supervisor podrá evaluar y verificar los recorridos.

Palabras clave. - Localización vehicular automatizada (AVL), Sistema de posicionamiento global (GPS), red GPRS, Residuos Sólidos Municipales (RSM).

ABSTRACT

The municipality of Loja (GADML) is always having problems with the trucks used to pick up the municipal solid waste (RSM) because this entity doesn't have a way for monitoring the truck schedule or its routes coverage, that's why at the UTPL emerge the idea of this research, which discusses the development of a low cost monitoring system that is able to show the route, coverage of the truck and some other specific information.

The designed and implemented system is conformed for: acquisition data module, processing module and server module. This system of automatic truck location has an application that can monitoring the parameters of the truck during the route, such as: location (latitude, longitude), time, date and speed. This data is stored in a MySQL database and allow to generate reports through a graphical interface where the personal of the GADML entity can see all information that has been monitoring and let to evaluate and verify the truck route.

Keywords. - Monitoring system, RSM, route, truck, database, application.

INTRODUCCIÓN

La gestión de los residuos sólidos urbanos constituye uno de los principales problemas de los Gobiernos autónomos descentralizados (GADs). La mala gestión causa la presencia de residuos abandonados, lo cual origina una serie de problemas directos en las ciudades, por ejemplo: producen olores molestos, favorecen la presencia de roedores e insectos los cuales son vectores de enfermedades, y, producen sensación de abandono y suciedad del paisaje urbano [1].

En el Ecuador, solo unas pocas municipalidades han elaborado una hoja de ruta de la gestión integral de residuos sólidos, sin embargo, es aún menor el número de ciudades que han implementado algún tipo de sistema o proceso exitoso encaminado a la gestión de residuos sólidos, estas son: Quito, Cuenca, Guayaquil, Manta, Cayambe, Ibarra, Loja, Otavalo, entre otras [1].

En la década de los 90, la municipalidad de Loja creó el programa de gestión de residuos sólidos, tomando como referencia el Plan de Acción Loja Siglo XXI. En la ciudad de Loja, el modelo de gestión de residuos es el Manejo Municipal directo el cual es un modelo utilizado especialmente en núcleos poblacionales medianos y pequeños. Entre todas las aristas que comprende el proyecto, está el componente de recolección de residuos que a su vez está conformada por: clasificación domiciliaria de los residuos, planificación de las rutas de recolección, y, recolección domiciliaria y de los residuos al relleno sanitario [2].

El componente de recolección de residuos sólidos urbanos o municipales (RSM) es un factor importante en todo el proceso y además conlleva costos que representan entre el 70% y 85% del costo total del manejo de RSM. El principal objetivo del sistema de recolección es la preservación de la salud, por lo cual es necesario que el sistema cubra a toda la población de manera ordenada, higiénica y en forma adecuada [3], [4].

Mejorar un sistema de recolección de RSM requiere de datos tales como: cantidad de residuos sólidos generados, población que debe servirse, distancias recorridas en las rutas de recolección, análisis de tiempos de recolección y transporte, características de las unidades recolectoras, personal de recolección, estructura vial de la ciudad, entre otros. Las consecuencias de un deficiente sistema de recolección son: operación ineficiente de la infraestructura, ineficiencia del personal, reducción de las coberturas del servicio de recolección y proliferación de tiraderos clandestinos [3], [4], [5].

El análisis anterior acerca de la importancia del sistema de recolección y transporte de RSM sumado a los problemas actuales identificados por los habitantes de la ciudad de Loja, tales como el incumplimiento de horarios y el incumplimiento de cobertura de rutas de

recolección, ponen en evidencia la necesidad de mejorar el sistema de recolección de RSM que posee el Gobierno Autónomo Descentralizado Municipal de Loja (GADML). La primera etapa para mejorar el sistema en mención es el monitoreo y cuantificación de los parámetros correspondientes a recorridos y tiempos de los vehículos recolectores con el objetivo de evaluar el comportamiento actual y proponer mejoras al sistema.

Este monitoreo de parámetros se lo puede efectuar con la ayuda de dispositivos basados en geo-posicionamiento sin embargo el sistema propuesto debe cumplir características de bajo costo y abierto, estas características cooperan a la disminución del impacto en las partidas presupuestarias del GADML y permiten la integración de los datos recolectados a otras plataformas de análisis y procesamiento. En base a lo anterior el objetivo principal del presente proyecto es desarrollar un prototipo de localización automática de vehículos (AVL-Automatic vehicle location) de bajo costo aplicado al monitoreo de vehículos de recolección y transporte de RSM de la ciudad de Loja. El desarrollo del proyecto se ha desglosado en los siguientes capítulos:

En el primer capítulo se realiza un análisis del estado actual en la que son tratados los RSM en la ciudad de Loja y los requerimientos del sistema propuesto.

En el segundo capítulo se analiza las diferentes opciones tecnológicas aplicadas al desarrollo e implementación del sistema propuesto

En el tercer capítulo se detalla la implementación e integración del sistema propuesto.

En el cuarto capítulo se documenta la evaluación de desempeño y los resultados obtenidos luego de integrar el sistema en un vehículo recolector de RSM de la ciudad de Loja.

CAPÍTULO I

ANÁLISIS DE LA SITUACIÓN ACTUAL DE LA RECOLECCIÓN DE RSM Y LOS REQUERIMIENTOS DEL SISTEMA DE MONITOREO

1.1. Gestión de desechos sólidos.

El medio ambiente tiene tres funciones económicas fundamentales: como proveedor de factores productivos en forma de materiales o de energía, como fuente de servicios de ocio y bienestar (mejorando la calidad de vida, permitiendo el disfrute de parajes naturales, agua y aire limpios, etc.) y como sumidero de residuos generados por la actividad económica [2]. Debido al crecimiento de la población y en consecuencia al aumento de la contaminación en sus diversas formas, se ha excedido la capacidad de auto-purificación de la naturaleza; esto evidencia, entre otras medidas, la necesidad de la intervención humana para llevar a cabo procesos de gestión de desechos sólidos.

Un desecho o residuo es algo que carece de valor de uso y por lo tanto de valor de cambio. La población está dispuesta a pagar para liberarse de los residuos, concluyendo que tienen un valor negativo para el ambiente y para la población [2].

La gestión de residuos se define como el conjunto de operaciones encaminadas a dar a los residuos producidos en una zona determinada el destino más adecuado desde el punto de vista económico y ambiental, según sus características, volumen, procedencia, posibilidades de recuperación y comercialización, coste de tratamiento y normativa legal [2].

1.1.1. Clasificación de los residuos sólidos municipales y peligrosos.

Administrativamente en América Latina y en el Ecuador los municipios son las instituciones responsables de la gestión de residuos sólidos.

Los residuos sólidos pueden clasificarse de acuerdo a su origen (domiciliar, industrial, comercial, institucional, público); a su composición (materia orgánica, vidrio, metal, papel, textiles, plásticos, inerte y otros); o de acuerdo a su peligrosidad (tóxicos, reactivos, corrosivos, radioactivos, inflamables, infecciosos) [6].

Los principales tipos de residuos sólidos urbanos son:

- Residuos sólidos municipales (RSM).
- Residuos sólidos especiales (RSE).
- Residuos peligrosos (RP).

1.1.1.1. Residuos sólidos municipales (RSM).

Los residuos sólidos municipales son aquellos provenientes de la generación residencial, comercial, institucional, industrial (pequeña industria y artesanía) y los residuos sólidos resultantes del barrido de calles de un conglomerado urbano y cuya gestión está a cargo de las autoridades municipales.

El componente residencial o domiciliario está constituido por desperdicios de cocina, papeles, plásticos, depósitos de vidrio y metálicos, cartones, textiles, desechos de jardín, tierra, etc. En América Latina y el Caribe esto representa entre 50 % a 75 % del total de RSM [7].

El componente comercial procedente de almacenes comerciales, oficinas, mercados, restaurantes, hoteles y otros constituye entre 10 % a 20 % de los RSM [7].

El componente institucional proviene de oficinas públicas, escuelas, universidades, servicios públicos, y representa entre 5 % a 15 % de los RSM [7].

Los residuos industriales provienen de la pequeña industria (baterías, confecciones de ropa, zapaterías, etc.) y talleres artesanales (sastrerías, carpinterías, de textiles, etc.). Este componente varía mucho de acuerdo a las características de las ciudades y podría representar entre 5 % a 30 % del total de RSM [7]. Usualmente las industrias y servicios mayores manejan sus residuos por cuenta propia o utilizan contratistas privados; existen algunas municipalidades que prestan estos servicios a la industria, pero en su mayoría lo realizan de forma poco eficiente.

El componente que proviene del barrido de calles y áreas públicas está constituido por residuos sólidos que arrojan los peatones, tierra, poda de árboles, etc. y representa entre 10 a 20% del total de RSM [7].

1.1.1.2. Residuos sólidos especiales (RSE).

Algunos de los residuos especiales por su cantidad o manejo pueden presentar un riesgo a la salud, tales como los residuos sólidos provenientes de establecimientos de salud; los productos químicos y fármacos caducos; los alimentos con plazos de consumo expirados; los desechos de establecimientos como, por ejemplo; baterías, escombros; y los residuos voluminosos que con autorización o por costumbre son manejados por las autoridades

municipales. Otros no peligrosos incluye a los animales muertos, autos abandonados, desperdicios de demolición y construcciones, residuos de parques y jardines, de festivales públicos y otros [6].

1.1.1.3. Residuos peligrosos (RP).

Los residuos peligrosos son aquellos sólidos o semisólidos que por sus características tóxicas, reactivas, corrosivas, radiactivas, inflamables o infecciosas plantean un riesgo sustancial real o potencial a la salud humana o al medio ambiente, cuando su manejo indebido dentro del área urbana se hace autorizada o ilícitamente, en forma conjunta con los residuos sólidos municipales [6].

El presente proyecto de investigación se enfoca sobre la gestión de RSM aplicados a la ciudad de Loja.

1.1.2 Cadena de Gestión de RSM.

De acuerdo al GADML la cadena de gestión de RSM implica: pre-clasificación, recolección, transporte, almacenamiento, clasificación, tratamiento y disposición final de residuos, refiérase a la Figura 1.1.

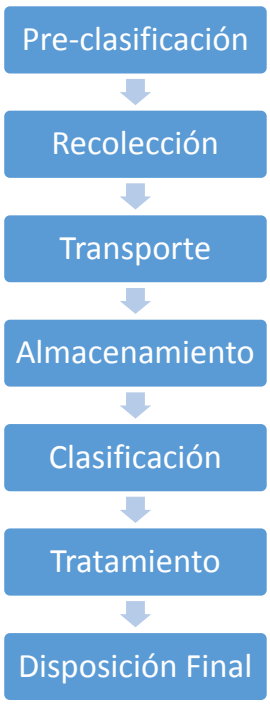

Figura 1.1. Cadena de suministro de residuos sólidos.
Elaboración: Por el autor.

Según el GADML el proceso de pre-clasificación se realiza en cada uno de los hogares, en el cantón Loja se diferencia dos tipos de desechos sólidos: orgánicos (desechos de comida) y no orgánicos (papel, plástico y cartón), el proceso de recolección inicia en el área residencial o comercial, donde el camión con pequeña capacidad y gran facilidad de acceso por cada uno de los callejones y callejuelas de las casas, recoge los desechos sólidos. Existen algunos métodos de recolección: en la acera, en la puerta de salida, y recolección en patio trasero. El método de recolección utilizado en la ciudad de Loja es en la acera; los usuarios depositan los residuos en un contenedor grande plástico y el trabajador recoge los residuos en el lado de la calle. Se tienen diferentes días de recolección tanto para los desechos orgánicos (lunes, miércoles y viernes) como no orgánicos (martes, jueves y sábado).

El transporte de desechos sólidos, según el GADML, cumple diversas rutas que conectan los diversos barrios y ciudadelas con zonas de almacenamiento. Los residuos pueden ser transportados directamente a los puntos de tratamiento o a plantas de transferencia donde se compactan y se cargan en camiones más grandes y adecuados para el transporte hasta su destino definitivo. La ubicación de la estación de transferencia en gran medida influye en la ruta de recolección. Por lo tanto, la distancia de la estación de transferencia debe ser óptima tanto para la ciudad como para el relleno sanitario.

Posteriormente se realiza el proceso de clasificación, tratamiento y disposición final. Los sistemas legales actualmente más utilizados son: el vertido controlado, la incineración, el reciclado y el compostaje.

Es importante impulsar programas para promover el conocimiento sobre la gestión de residuos basado en la jerarquía que da prioridad a la minimización de residuos a través de 3R- reducir, reusar y reciclar.

Con respecto a la recolección y transporte llevado a cabo por el GADML, en la Figura 1.2 y en el Anexo A se presenta un resumen de las rutas de recolección de desechos sólidos en la ciudad de Loja. En la semana se programa un total de 11 rutas, se distribuye un total de 55 trabajadores en horarios de recolección desde las 07h00 hasta las 21h30.

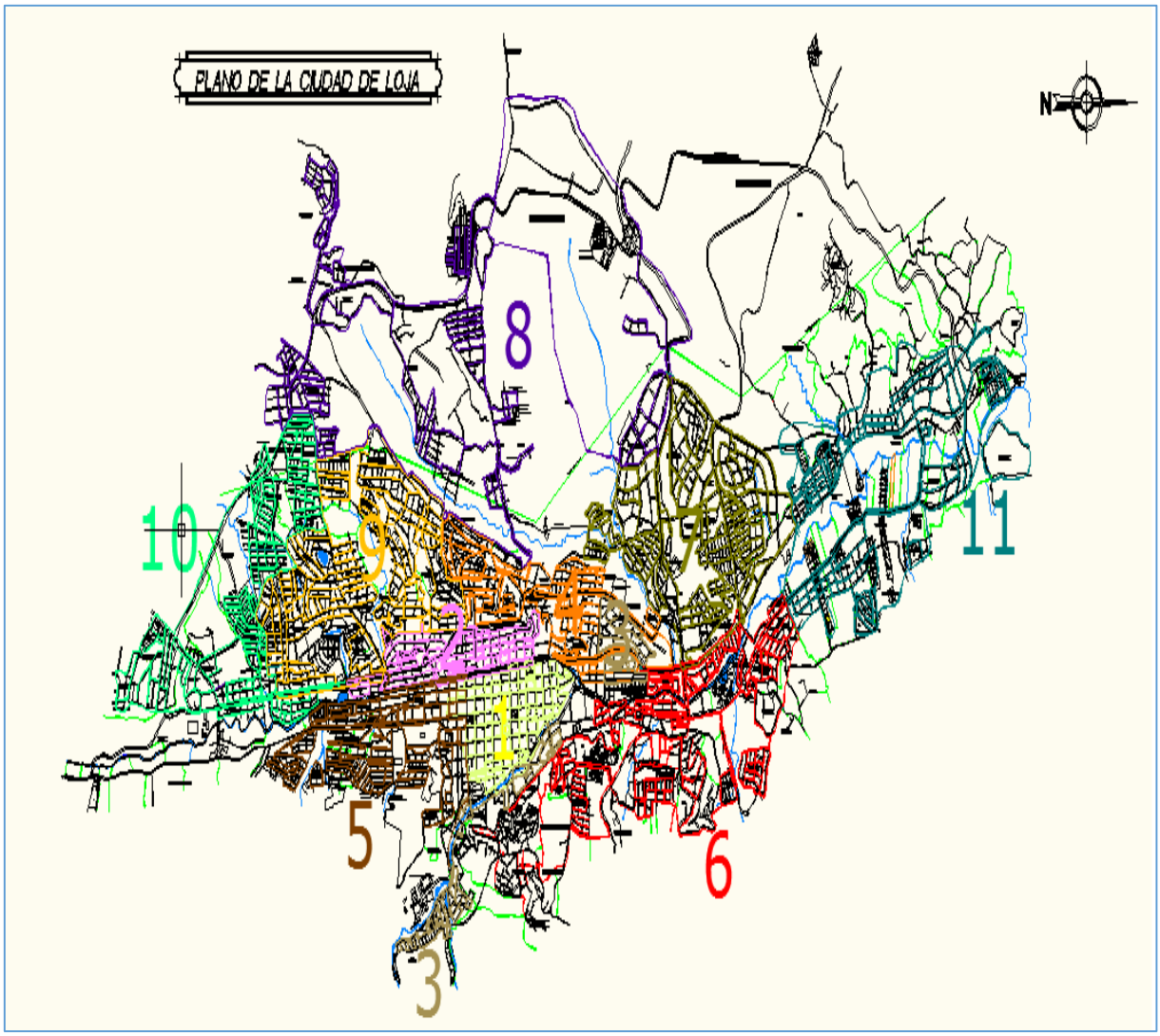

Figura 1.2. Rutas de recolección de RSM en la ciudad de Loja.
Fuente: [GADML].

1.2. Ventajas de la implementación de un sistema de monitoreo.

La recolección de RSM equivale un 70 % a 85 % del costo total de la gestión de residuos sólidos [2], [6]. Las consecuencias de tener ineficientes rutas de recorrido y carecer de una buena programación, son: altos costos de operación y altos niveles de contaminación del aire y ríos. Por lo tanto, es necesario controlar las variables que intervienen en el desempeño del sistema de recolección y transporte de RSM (RT-RSM), pero antes de poder controlar primero se debe medir dichos parámetros.

La implementación de un sistema de monitoreo permite conocer los parámetros: ubicación y velocidad de las unidades de recolección de RSM, distancia y duración del recorrido de las rutas.

A continuación, se resumen algunas ventajas de la implementación de un sistema de monitoreo:

- Verificar el cumplimiento de rutas y horarios por parte de las unidades de recolección de RSM.
- Crear una base de datos, con la información de entrada para la ingeniería de rutas y horarios del sistema de recolección de RSM.
- Verificar las operaciones de la flota de vehículos, mediante parámetros como: tiempos de parada, distancias recorridas, velocidades, entre otros.
- Verificar el estado y ubicación de los contenedores distribuidos en la ciudad.

1.3. Diagrama funcional del Sistema de monitoreo del RT-RSM.

La metodología aplicada para el diseño e implementación del sistema de monitoreo de RT-RSM, se describe a continuación:

- Identificación de los requisitos y/o requerimientos del sistema.
- Investigación y justificación de una arquitectura aplicable al sistema requerido.
- Analizar los parámetros de decisión en cada uno de los módulos de la arquitectura planteada.
- Realizar un análisis comparativo de la solución, prestaciones de los equipos ofrecidos por cada empresa y la inversión requerida.

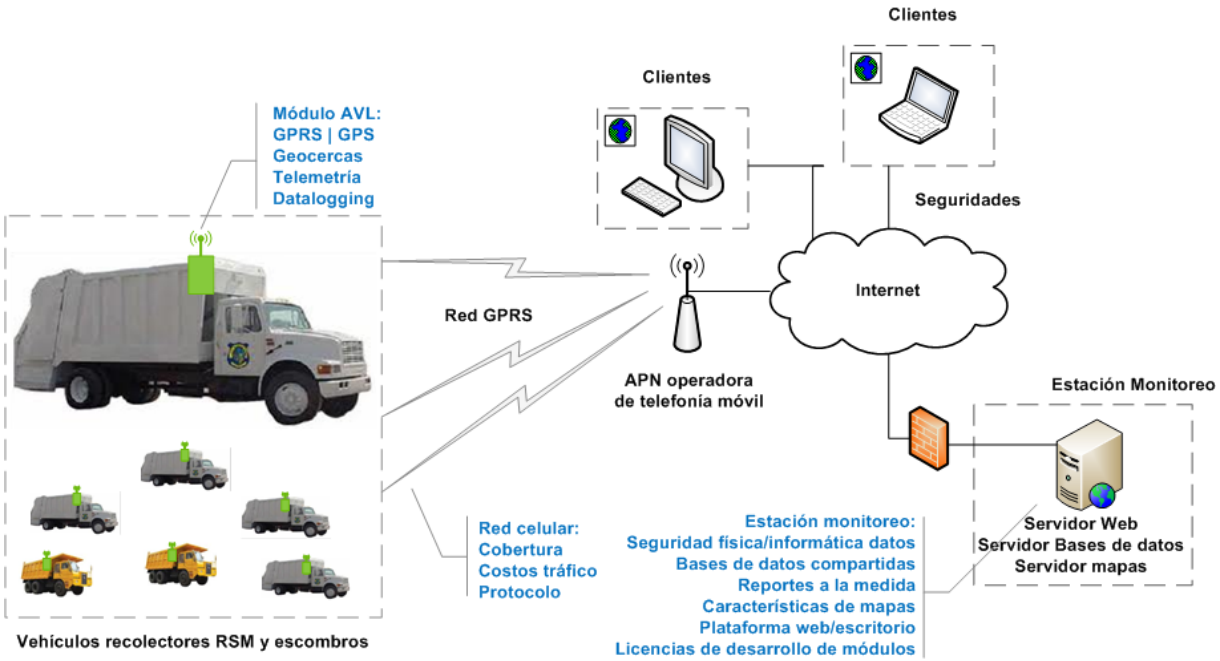

Figura 1.3. Diagrama funcional del sistema de monitoreo de los vehículos de RT-RSM.
Elaboración: Por el autor.

1.3.1. Propuesta del sistema de monitoreo.

Los sistemas AVL (AVL-Automatic vehicle location) permiten a las organizaciones rastrear y coordinar los movimientos de sus flotas vehiculares. La tecnología de los sistemas de rastreo fue posible por la integración de tres nuevas tecnologías: tecnologías de navegación como el sistema de posicionamiento global (GPS- global positioning system), tecnologías de bases de datos como el Sistema de información geográfica (GIS- geographic information system) y las tecnologías de comunicación como las redes GPRS (general packet radio service) [8], [9].

En el entorno académico se han propuesto diferentes sistemas hardware y software aplicados al rastreo vehicular. En [8] se desarrolla un sistema software que visualiza y registra en tiempo real la posición, la velocidad de avance y el nivel de combustible de un vehículo determinado, estos datos permiten la supervisión del estado de la flota vehicular y la gestión de los recorridos. En [10] se presenta un sistema que proporciona servicio de rastreo mediante una aplicación móvil desarrollada en Java y una aplicación web; sin embargo, para la implementación de los sistemas mencionados se utiliza un dispositivo AVL de mercado.

En [11] se integra un sistema AVL basado en los microcontroladores Intel y Atmel, un módem GSM y un dispositivo GPS. En [12] se desarrolla un sistema de rastreo vehicular en tiempo real basado en el dispositivo Arduino Uno, el módulo GPRS SM5100B y el módulo GPS EM406. Estos sistemas fueron desarrollados bajo el concepto de bajo costo, fueron evaluados experimentalmente sin embargo no se cuantificaron los errores y su sensibilidad ante bajos niveles de cobertura de la red GPRS. Sistemas más complejos como los descritos en [13], [14], [15] y [16], presentan el diseño de los componentes hardware y software de un sistema de rastreo vehicular, el servidor central de rastreo almacena en una base de datos la información de posición que puede ser accedida mediante un navegador web, empero no se realiza ni analiza el comportamiento del sistema.

La arquitectura del sistema propuesto debe cumplir las características de bajo costo y abierto, estas características cooperan a la disminución del impacto en las partidas presupuestarias del GADML y permiten la integración de los datos recolectados a otras plataformas de análisis y procesamiento, en base a ello se propone la arquitectura de la Figura 1.3. Los módulos que forman parte de la arquitectura del sistema, son los siguientes:

- Módulo de Unidad de monitoreo.
- Módulo de Transmisión de datos.

- Módulo de Plataforma software de monitoreo.

1.3.1.1. Unidad de monitoreo.

Los parámetros y características que se toma en cuenta en el diseño o selección del módulo de monitoreo son los siguientes:

- **Unidad AVL. -** Esta arquitectura se la conoce como Localizador Automático de Vehículos, debe ser un dispositivo con funcionalidad comprobada, que integre las unidades GPS y GPRS, orientado a instalación en vehículos.
- **Robustez. -** El problema más común en los dispositivos GPRS es que se inhiben al momento de adquirir una señal de comunicación inestable o al momento de pérdida de señal. Existen productos en el mercado cuyos tiempos de inhibición están entre los 20 y 60 minutos.
- **Soporte geo-cercas y puntos de control. -** No todas las unidades AVL soportan las cercas/límites geográficos y puntos de control de rutas.
- **Batería de respaldo. -** Es necesario que los dispositivos de monitoreo posean una batería de respaldo para garantizar el rastreo y comunicación en condiciones de falta de energía en la batería principal del vehículo.
- **Módulo de Entradas/Salidas. -** Es un componente extra de los AVL que permite interactuar remotamente con el vehículo, por ejemplo; aviso de alarmas, bloqueo de funciones del vehículo, entre otros.

1.3.1.2. Transmisión de datos.

Los parámetros y características que se toma en cuenta en el diseño o selección del módulo de transmisión de datos son los siguientes:

- **Cobertura. -** En cualquiera de los sistemas de telefonía celular que se elija, se debe indagar la cobertura de la red hacia todas las rutas de recolección. Por ejemplo, puede ocurrir que, para rutas especiales las dos operadoras de telefonía móvil no posean altos índices de calidad de servicio.
- **Costos GPRS/Datos. -** Al tratarse de sistemas digitales de comunicación de datos la tarifa mensual/anual dependerá de la cantidad de Megabytes de tráfico generado. Esta tarifa es un criterio de decisión del sistema a usar.

- **Protocolo de transmisión de datos TCP/UDP. -** Depende de la empresa que preste el servicio el cliente escoge el protocolo de transmisión. En ello intervienen variables como: velocidad de la transmisión de datos, rendimiento de la entrega de la información, cantidad de tráfico, costos e integridad de la información. Es de entender que existen “compromisos” entre estas variables.

1.3.1.3. Plataforma software de monitoreo.

Los parámetros y características que se toma en cuenta en el diseño o selección del módulo de la plataforma de monitoreo son los siguientes:

- **Manejo de mapas dinámicos. -** Se analiza, si se utiliza planimetría estática o actualizable a lo largo del tiempo, esta última es recomendable en ciudades de continua expansión como Loja.
- **Resolución a nivel de calles. -** No todas las soluciones de mercado poseen la información de planimetría apta para efectuar un control de rutas.
- **Base de datos**. - Se debe indagar si en la base de datos utilizada por el sistema existe la opción de tener la información cifrada o el riesgo de que la información no esté disponible para consultas posteriores con otra aplicación o para una descarga simple.
- **Servidor de datos y seguridad física e informática. -** Es muy importante que el servidor de monitoreo de los datos esté situado en un data center, donde el servidor e información tengan seguridad física y seguridad informática.
- **Plataforma web de escritorio**. - Este parámetro solo afecta para el caso de requerir sistema de monitoreo multi-clientes (varias personas puedan observar la información).
- **Licencias de software de desarrollo**. - Este parámetro afecta en la medida que el cliente desee o requiera desarrollar módulos complementarios. Se debe analizar si este desarrollo adicional involucra la compra de licencias o si se puede hacer uso de licencias de desarrollo gratuitas.
- **Ingeniería de rutas**. - El software de monitoreo debe poseer rutinas que controle y adquiera variables y parámetros de entrada para la futura realización de ingeniería de rutas, las variables recomendadas son las siguientes:

  - Puntos de control de las rutas de recolección.
  - Puntos de control de los contenedores.

- Control de llenado de contenedores.
- Nodos de recolección masiva (tolvas).
- Capacidades de los vehículos recolectores.
- Número de unidades de recolección.
- Topología de rutas y sectorización.
- Asignaciones de rutas por día, por cada vehículo.
- Kilómetros recorridos por ruta.
- Tiempos de recorrido.
- Toneladas recolectadas diariamente.

1.4. Análisis de costos de sistemas comerciales de rastreo.

Desafortunadamente, la mayoría de los Gobiernos Autónomos descentralizados no tienen los recursos necesarios para la gestión de RSM, en la mayoría de los casos, sólo se considera el costo económico que involucra un proceso básico de gestión de desechos sólidos [17].

La Universidad Técnica Particular de Loja (UTPL) comprometida con el desarrollo social-económico del cantón Loja, ha decidido financiar los costos de investigación y desarrollo del trabajo de investigación: Sistema de localización automática aplicado a la flota de vehículos de recolección y transporte de Residuos Sólidos Municipales. El GADML asumirá los gastos correspondientes a la fabricación, puesta en marcha y operación del sistema, luego de la transferencia del proyecto por parte de la UTPL.

Con el objetivo de realizar un análisis de costos de la integración del sistema basado en equipos de mercado versus el desarrollo del sistema en la UTPL, se investigó los costos de sistemas similares ofrecidos por empresas. Como resultado, se obtuvo la Tabla 1.1 la cual presenta una comparación de los sistemas comerciales de rastreo aplicables a los vehículos de recolección y transporte de RSM, todas las soluciones con enfoque y cobertura en la ciudad de Loja.

En base a los costos de los módulos de mercado y debido a que son plataformas cerradas se decidió diseñar y desarrollar el dispositivo de rastreo.

Tabla 1.1. Comparativa de los sistemas comerciales de rastreo aplicables a los vehículos de recolección de RSM.

Empresa	Costo por unidad	Monto mensual/ anual por servicio	Monto software	Observaciones
HUNTER	$ 400	$ 450 anual	$ 3500	Monitoreo mediante web. Servicio de recuperación. Servicio a nivel nacional. Empresas clientes de la provincia de Loja.
CARLINK	$ 670	$ 320 anual	$ 1500	Monitoreo mediante web. Servicio de recuperación. Servicio a nivel nacional. Empresas clientes de la provincia de Loja.
MOTOROLA	$ 800	$ 13 mensual	$ 3000	Monitoreo mediante web. Mapas de AutoCAD. Respuesta lenta. Software cerrado, datos encriptados. Empresas clientes de la provincia de Loja.
PCSERVICIOS	$ 600	$ 10 mensual	$ 4000	Software de escritorio. Protocolo UDP, entre 10% y 15% datos perdidos. Control de rutas. Servicio a nivel nacional. Empresas clientes de la provincia de Loja.
KRADAC	$ 300	$ 350 anual	$ 1200	Monitoreo mediante web. Número ilimitado de consultas. Control de rutas. Servicio a nivel nacional. Empresas clientes de la provincia de Loja.
SORAVI	$ 740	$ 410 anual	$ 4350	Monitoreo mediante web. Servicio a nivel nacional.
SERVIFAST	$ 650	$ 300 anual	$ 2700	Monitoreo mediante web. Servicio a nivel nacional.

POWER CONTROL	$ 800	$ 300 anual	$ 1500	Software de escritorio. Servicio a nivel nacional.
ROADTRACK	$ 1000	$ 310 anual	$ 1500	Software de escritorio. Servicio a nivel nacional.

Elaboración: Por el autor.

1.5. Modelo de inversión para el diseño y desarrollo del sistema.

1.5.1. Inversión a realizar por la UTPL.

A continuación, se detalla los rubros de investigación y desarrollo a realizar por la UTPL (ver Tabla 1.2).

Tabla 1.2. Inversión a realizar por la UTPL para el desarrollo del proyecto.

Componente	Sub-componentes	Costo Unitario	Cant.	Periodo (Meses)	Total
Recurso humano	Gerente de Proyecto	$ 400	1	8	$ 3200
	Analista de sistemas	$ 800	1	2	$ 1600
	Desarrollador ingeniero ciencias computación	$ 800	2	8	$ 12800
	Integrador ingeniero electrónico	$ 800	1	4	$ 3200
Infraestructura: estaciones de trabajo	Equipos de escritorio	$ 900	2	-	$ 1800
	Laptops	$ 1000	1	-	$ 1000
Prototipos para pruebas	Tarjetas electrónicas: unidades GPS, unidades de transmisión GPRS, unidades de procesamiento. Chips con activación de datos, baterías	$ 2000	1	-	$ 2000
Costo licencias para desarrollo	Desarrollo de plataformas web. Desarrollo de rutinas	$ -	-	-	$ -

	de comunicación con red GPRS.				
	TOTAL				**$ 25600**

Elaboración: Por el autor.

1.5.2. Inversión a realizar por el GAD Municipal de Loja.

A continuación, los rubros correspondientes a la puesta en marcha del sistema en el cual se suponen 15 unidades iniciales (ver Tabla 1.3).

Tabla 1.3. Inversión a realizar por el GADML, para la puesta en marcha del proyecto.

Componente	Sub-componentes	Costo Unitario	Cant.	Total
Módulo de rastreo	Módulo electrónico por unidad.	$ 200	15	$ 3000
	Instalación en vehículos.	-	-	-
Software de monitoreo	Servidor para centralizar los datos	$ 1000	1	$ 1000
	Complementos HW y SW	$ 1000	1	$ 1000
TOTAL INVERSIÓN INICIAL				**$ 5000**

Elaboración: Por el autor

Cabe recalcar que el GADML también tendrá que asumir el costo mensual/anual por transmisión de datos (ver Tabla 1.4).

Tabla 1.4. Costos asumidos por el GADML, para el manteamiento mensual del proyecto.

Componente	Sub-componentes	Costo Unitario	Cant.	Periodo (12 meses)
Transmisión de datos	Costos de transmisión de datos a pagar a la operadora de telefonía celular.	$ 6	15	$ 1080
TOTAL COSTO ANUAL				**$ 1080**

Elaboración: Por el autor.

El presente proyecto asume las actividades referentes al diseño y desarrollo electrónico del dispositivo de rastreo y monitoreo de variables correspondientes al sistema de RT-RSM, así también con el objetivo de evaluar su compatibilidad con plataformas software aplicadas a visualización y registro de variables, se contempla el desarrollo de rutinas software que permiten la conectividad, almacenamiento y visualización de los datos enviados por el dispositivo de monitoreo.

CAPÍTULO II

ANÁLISIS DE LA TECNOLOGÍA UTILIZADA

2.1. Generalidades.

Existen algunas tecnologías que se pueden utilizar para el seguimiento de rutas, presentando diferentes opciones que se adecuan a las necesidades y presupuesto de cada empresa, estas tecnologías pueden variar de acuerdo al fabricante, modelo o marca. Cualquiera de estas tecnologías debe cumplir con la disponibilidad y funcionalidad que se requiere en un sistema de monitoreo vehicular, tal como se lo mencionó en el capítulo anterior.

Las principales tecnologías que se pueden utilizar para el procesamiento de datos son: Arduino y Raspberry Pi. Para el monitoreo vehicular se utiliza un módulo GPS y para la transmisión de datos se utiliza un módulo GPRS, existen varios modelos y marcas de estos dos últimos, por lo tanto, se realiza un análisis para determinar cuál se adecua mejor a nuestros requerimientos.

Para el almacenamiento de los datos se requiere de un servidor que presente seguridad física y lógica. Adicional a esto se debe tomar en cuenta el software utilizado para la gestión y administración de los datos. El análisis, comparación y selección de estas tecnologías se la describe a continuación.

2.2. Módulo de procesamiento.

En la actualidad existen tecnologías que integran a los microcontroladores en placa de prototipado, para tener una plataforma más robusta, con mayores funcionalidades y más fácil de programar, lo que hace muy adecuado y relevante para desarrollos experimentales. Dentro de estas tecnologías tenemos Arduino y Raspberry Pi, físicamente estos dispositivos son muy similares, ambos son pequeñas placas de circuito con pines disponibles, pero en realidad son muy diferentes.

2.2.1. Arduino Uno.

Es una plataforma de electrónica abierta utilizada para desarrollar prototipos basados en software y hardware flexibles, enfocada a diseñadores y/o aficionados interesados en crear entornos interactivos. El hardware de la plataforma Arduino ha ido evolucionado conforme ha ido surgiendo la necesidad, actualmente las placas más utilizadas son: Arduino Uno, Arduino mega, Arduino yun y Arduino nano [18]. A continuación, se describe la placa Arduino Uno que más se adapta a este proyecto.

La placa Arduino Uno se basa en el microcontrolador ATmega328, este cuenta con catorce pines de E/S digitales de los cuales seis pueden ser usados como salida PWM, cuenta con seis entradas análogas, posee un conector Jack para la alimentación, o puede ser alimentado mediante el cable USB. Las características principales del Arduino Uno se describe en la Tabla 2.1.

Tabla 2.1. Características técnicas del Arduino Uno.

Parámetros	Características
Microcontrolador	ATmega328
Voltaje de funcionamiento Voltaje de entrada (recomendado) Voltaje de entrada (límites) Pines E/S digitales	5 V 7-12 V 6-20 V 14 (6 proporcionan PWM)
Pines de entrada analógica Corriente DC E/S Corriente CC Pin 3.3V	6 40 mA. 50 mA.
Memoria Flash	32 KB (0.5 KB gestor de arranque)
SRAM	2 KB
EEPROM Tamaño del vector de interrupción	1 KB 2 instrucciones palabra /vector
Dimensiones	68.6 mm × 53.4 mm

Elaboración: Por el autor.
Fuente: [18].

En Arduino Uno existen tres versiones la R1, R2, y R3. La diferencia entre estas es que en lugar de utilizar el chip controlador USB a serial FTDI, utiliza el microcontrolador ATmega16U2 (R3), y ATmega8U2 (R2), programado como convertidor USB a serial [18]. En la Figura 2.1 se observa la placa del Arduino Uno R3. Esta placa es la más utilizada para proyectos donde no se requiere gran cantidad de E/S digitales y porque permite crear programas con una complejidad aceptable.

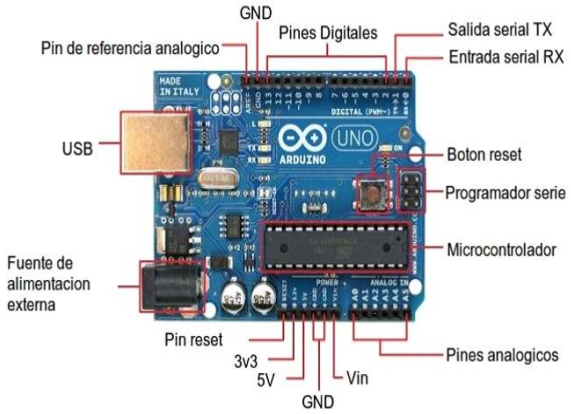

Figura 2.1. Placa Arduino Uno R3.
Fuente: [18].

2.2.2. Raspberry Pi.

El Raspberry Pi es un miniordenador del tamaño de una tarjeta de crédito, con la capacidad de ejecutar varios programas. Este miniordenador puede ser utilizado en proyectos de electrónica y en muchos de los casos se lo utiliza para realizar tareas de una PC de escritorio, como hojas de cálculo, procesadores de texto y juego, también reproduce video de alta definición. Raspberry Pi es un procesador de aplicaciones multimedia BCM2835 (del fabricante Broadcom) y de arquitectura ARM (Advanced RISC Machine), esta arquitectura está diseñada y optimizada para la eficiencia de energía (modelo B). Soporta sistemas operativos que son distribuciones de Linux para sistemas embebidos, como; Raspbian basado en Debian, Pidora basado en Fedora, Arch Linux ARM, basado en Arch Linux, cabe destacar que, el Raspberry Pi no incluye una unidad de estado sólido para el arranque del sistema operativo, para ello utiliza una tarjeta SD externa [19]. En la Tabla 2.2 se resume las principales características del sistema embebido Raspberry Pi.

Tabla 2.2. Características técnicas del Raspberry Pi modelo B.

Parámetros	Características
SOC (System on Chip)	Broadcom BCM2835
CPU	ARM 1176JZF-S a 700 MHz -- 1000MHz
GPU	Broadcom VideoCore IV
RAM	512MB
Puerto USB 2.0	2
Salidas Multimedia	HDMI, RCA, AUDIO 3.5 mm
Conectividad	Fast Ethernet, USB Wi-Fi
Tipo de Almacenamiento	Tarjeta SD externa
Periféricos de bajo nivel:	8 x GPIO, SPI, I²C, UART
RTC	No (Puede sincronizarse con un servidor NTP)
Alimentación	5V Micro USB – 700 mA
Dimensiones	85.6 mm × 53.98mm

Elaborada por el autor.
Fuente: [19].

Raspberry Pi debe utilizar una fuente de poder de 5V con una salida micro-USB que proporciona una corriente mayor a 700 mA (recomendable 1 A), este consumo de energía se debe a que a medida que se conectan los dispositivos como: ratón, teclado, HDMI, RCA, etc. el consumo de corriente aumenta [19]. En la Figura 2.2 se muestra el Raspberry Pi modelo B.

Figura 2.2. Raspberry Pi modelo B.
Fuente: [19].

2.2.3. Comparación de las características técnicas de las tecnologías Arduino y Raspberry Pi.

En el presente trabajo se consideró la aplicación o utilización de un determinado dispositivo, teniendo siempre el criterio de no desperdiciar recursos, ya sea económicos o de funcionalidades, y de esta manera se eligió la alternativa más adecuada para los requerimientos del proyecto. En la Tabla 2.3 se realiza una comparación de Arduino Uno y el Raspberry Pi.

Tabla 2.3. Comparación de Arduino Uno y Raspberry Pi.

Placa	Procesador	Velocidad	SRAM	FLASH	Costo
Arduino Uno	ATmega328	16MHz	2 KB	32KB	$ 30
Raspberry Pi	BCM2835/ARM11	700 MHz	512MB	Tarjeta SD externa	$ 130

Elaboración: Por el autor.
Fuente: [18], [19].

Antes de seleccionar la tarjeta electrónica es necesario describir una serie de datos que se tomó en consideración:

- Se definió la cantidad de pines analógicos y digitales que se va a necesitar, para que con este escrutinio poder descartar algunas placas.
- Se determinó el tamaño de código que se va a generar, se eligió una placa adecuada con la respectiva cantidad de memoria flash.

Por lo mencionado anteriormente para este trabajo de investigación se escogió la tarjeta electrónica de control Arduino Uno, ya que simplifica el proceso de trabajo con microcontroladores y además ofrece algunas ventajas:

- **Bajo costo**: La placa Arduino Uno es relativamente barata comparada con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino pre-ensamblados cuestan menos de 50 dólares.
- **Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux.
- **Entorno de programación simple**: Flexible para que usuarios avanzados puedan aprovecharlo.
- **Código abierto y software extensible:** El software Arduino está publicado como herramientas de código abierto disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, se puede añadir código AVR-C directamente en programas Arduino si se requiere.
- **Código abierto y hardware extensible:** El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo.

2.3. Módulo GPS.

El GPS está basado en el posicionamiento por satélites, fue diseñado para apoyar los requerimientos de navegación y posicionamiento precisos. Hoy en día es una herramienta importante para aplicaciones de posicionamientos de puntos en tierra, mar y aire. Está integrado por tres segmentos o componentes de un sistema: el segmento espacial, el segmento de control y el segmento de usuario [20].

“Segmento espacial es una constelación de satélites que orbitan la Tierra a una altitud aproximada de 20000 km. Los satélites del GPS transmiten dos señales de radio de baja potencia, llamadas "L1" y "L2". La señal GPS contiene tres componentes de información: un código pseudoaleatorio, los datos de efemérides de satélite y datos de almanaque. El código pseudoaleatorio identifica al satélite que transmite su señal. Los datos de efemérides de satélite proporcionan información sobre la ubicación del satélite en cualquier momento. El almanaque contiene información sobre el estado del satélite y la

fecha y hora actuales. En los satélites el tiempo es controlado por relojes atómicos que son cruciales para conocer su posición exacta" [20].

Las posiciones se obtienen mediante la determinación de las distancias a los satélites visibles. El GPS puede dar posiciones muy precisas, pero aún hay fuertes de error, estos incluyen los errores del reloj, los retrasos atmosféricos, las señales que se refleja de los objetos en la superficie de la Tierra, e incluso la degradación intencionada de la señal del satélite [20].

Segmento de control es una serie de estaciones de rastreo distribuidas en la superficie terrestre que continuamente monitorea a cada satélite analizando las señales emitidas por estos y a su vez, actualiza los datos de los elementos y mensajes de navegación, así como las correcciones de reloj de los satélites. Las estaciones se ubican estratégicamente cercanas al plano ecuatorial [20].

"Segmento de usuario lo integran los receptores GPS que registran la señal emitida por los satélites para el cálculo de su posición tomando como base la velocidad de la luz y el tiempo de viaje de la señal, así se obtienen las pseudodistancias entre cada satélite y el receptor en un tiempo determinado, por lo menos cuatro satélites se observan en tiempo común; el receptor calcula las coordenadas X, Y, Z y el tiempo" [20].

Dentro del segmento de usuario existen algunos módulos GPS, estos se basan en un chip, los más conocidos son: el chip SiRFstarIII, y el chip MTK MT3318. El costo de estos módulos oscila entre 65 a 75 dólares. Los datos que se reciben en el módulo GPS siguen el protocolo NMEA siglas de National Marine Electronics Asociation [20]. Para entender mejor este protocolo se presenta la Figura 2.3, que tiene esta estructura:

```
$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020916,,,A*44
```

Figura 2.3. Estructura del protocolo.
Fuente: [20].

Inicia por "$" y termina en un "A*" seguido de dos números, éste es el checksum. Los datos se separan por comas y el primero es el tipo de transmisión, en este caso el GPRMC (o RMC) [20].

- **044235.000.-** es la hora GMT (04:42:35).
- **A.-** es la indicación de que el dato de posición está fijado y es correcto. V seria no válido.

- **4322.0289.-** es la longitud (43º 22.0289´).
- **N.-** Norte.
- **00824.5210.-** es la latitud (8º 24.5210´).
- **W**.- Oeste.
- **0.39**.- velocidad en nudos.
- **65.46.-** orientación en grados.
- **020916.-** fecha (2 de septiembre del 2016).

2.4. Módulos GSM/GPRS.

GPRS es una comunicación basada en paquetes de datos. En GSM, los intervalos de tiempo son asignados mediante una conexión conmutada, en tanto que en GPRS son asignados mediante un sistema basado en la necesidad a la conexión de paquetes. Es decir, que si no se envía ningún dato por el usuario las frecuencias quedan libres para ser utilizadas por otros usuarios [21].

Existen varios módulos GSM/GPRS, dentro de estos se tiene el: SM5100B, el SIM908, MG323. Las características más importantes son: la banda de frecuencia en que operan, la sensibilidad de recepción, la velocidad de transmisión y la potencia de transmisión. La frecuencia en la que operan estos módulos es estándar, soportan las frecuencias 850MHz, 900MHz, 1800MHZ y 1900MHz, en cuanto a la sensibilidad de recepción el módulo SIM908 presenta una mejor sensibilidad -108 dBm, mientas que el módulo SM5100B presenta una sensibilidad de -102dBm, en lo que tiene que ver con la velocidad el módulo con mayor capacidad es el SIM908 [22]. Estos módulos pueden ser integrados dentro de una sola placa con un conjunto chips y elementos electrónicos, que permiten ampliar las funcionalidades de una placa Arduino de manera cómoda y sencilla, a estas placas integradas se las denomina shield (escudo), que no es más que un módulo de expansión en forma de placa impresa que se puede conectar a la parte superior de la placa Arduino para ampliar sus capacidades.

2.5. Selección del Shield GPS GPRS y sus componentes.

En este trabajo de investigación se busca un shield que permita utilizar los sistemas de comunicación móvil para poder interactuar a distancia con la plataforma. En el mercado existe una gran variedad de shields que han sido diseñados específicamente para ofrecer servicios a través de los sistemas GSM, GPRS o una combinación de los mismos. Dentro de las opciones disponibles, se selecciona el shield GPRS GPS quadband.

2.5.1. Shield GPRS GPS quaband para Arduino.

Cuenta con un chip SIM908 integrado en la propia placa, ofrece la posibilidad de utilizar la tecnología GPS para posicionamiento en tiempo real, resultando muy útil para aquellas aplicaciones en las que se necesita conocer la ubicación del dispositivo [23]. En la Figura 2.4 se muestra una imagen del shield.

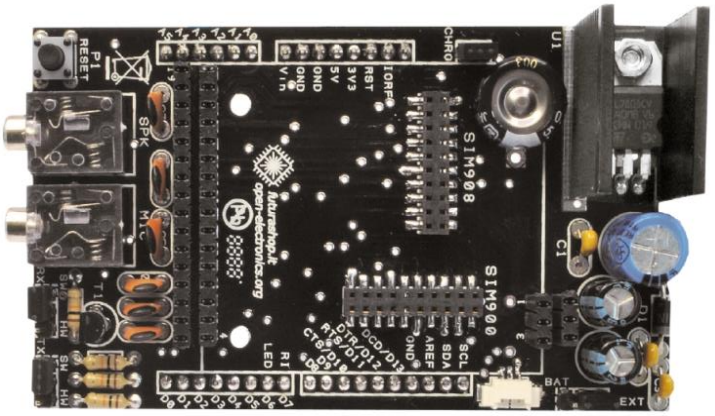

Figura 2.4. Shield de GSM GPRS (SIM908-SIM900).
Fuente: [23].

Este módulo de expansión es compatible con la mayoría de las placas Arduino, no sólo se puede transmitir la ubicación del sistema a través de HTTP y almacenarla en un servidor web, sino que también se puede utilizar Google Maps para ver su localización sobre el mapa en todo momento. Para el correcto funcionamiento se debe adquirir las respectivas antenas, el costo de la placa rodea los 135 dólares [23].

Analizando todas las características y prestaciones del módulo se concluye que se utiliza el shield **GPRS/GSM GPS (SIM908) de DFRobot**, debido a que posee las siguientes ventajas:

- Cuenta con un motor cuatribanda GSM / GPRS que trabaja en las frecuencias EGSM 900MHz/DCS 1800MHz y GSM850 MHz / PCS 1900MHz.
- Se controla a través de comandos AT, el diseño de este shield permite manejar la función GPS y GSM directamente con el ordenador y la tarjeta Arduino Uno. Utiliza un chip integrado SIM908 de SIMCOM. Consta de una interfaz estándar en la industria y la función GPS, la combinación de ambas tecnologías permite realizar un seguimiento en cualquier lugar, vehículos y personas solo con la cobertura de la señal.

A continuación, se detalla las características electrónicas del shield (ver Tabla 2.4).

Tabla 2.4. Características electrónicas del Shield.

Parámetros	Características
Chip Integrado	SIM908
Consumo de energía modo GSM	100 mA.
Consumo de energía reposo	77 mA.
Voltaje de entrada (recomendado)	6-12 V
Interruptor de control	USB / Arduino
Antena	UFL para GPS y GSM
RTC	Admite SuperCap
Temperatura de funcionamiento	-40 °C ~ +85 °C
Protocolo de Comunicación	UART
RoSH	Si
Potencia fuente	9 -20 V
Entrada de voltaje VH	4.5 ~ 5.5 V
Entrada de voltaje VL	-0.3 ~ 0.5 V
Velocidad de transmisión	9600 bps
Entradas y Salidas	Micrófono y altavoces envía señales DTMF, 12 GPIO, 2 PWMs y ADC (lógica 2.8 V)
Dimensiones	71.4 mm × 66 mm × 16 mm

Elaboración: Por el autor.
Fuente: [23].

2.5.2. Chip GPS GPRS SIM908.

Chip completo GSM / GPRS cuatribanda, el diseño compacto de GPRS y GPS en un paquete SMT ahorra tiempo y los costos para los desarrolladores de aplicaciones con GPS [24] (ver Figura 2.5). En la Tabla 2.5 se detalla las características.

Figura 2.5. Módulo GPS/GSM.
Fuente: [24].

Tabla 2.5. Características electrónicas del Chip GPS/GSM.

Características GPS/GSM
Cuatribanda 850/900/1800/1900MHz Clase de estación móvil de GPRS B
Clase 4 (2 W @ 850/900 MHz)
Clase 1 (1 W @ 1800/1900MHz)

Control vía comandos AT (GSM 07.07, 07.05 y SIMCom mejorados los comandos AT) Herramientas de aplicación SIM
Rango de funcionamiento: GPRS: 3.2 ~ 4.8 V GPS: 3.0 ~ 4.5 V Bajo consumo de energía Certificaciones: CE,RoHS
Compatibilidad: Interfaz de comandos AT celular
Tipo de Receptor.42 canales, código GPS L1 C/A, motor de alto rendimiento STE
Sensibilidad: Seguimiento: -160 dBm, arranques en frío: -143 dBm Precisión: Posición-horizontal: <2.5m CEP
Dimensión: 30 mm × 30 mm × 0.32 mm

Elaboración: Por el autor.
Fuente: [24].

2.5.3. Antena GSM Conector UFL.

Se conecta mediante conector UFL al chip SIM 908 con el objetivo de mejorar la recepción de la señal (ver Figura 2.6). En la Tabla 2.6 se detalla las características.

Figura 2.6. Antena GSM.
Fuente: [25].

Tabla 2.6. Características de Antena GSM (conector UFL).

Parámetros	Características
Frecuencia VSWR ROE Ancho de banda	870-960, 1710-1900 MHz ≤ 1.6 ~ 5 MHz
Impedancia Ganancia típica Polarización	50 Ohm 3.5dBi Vertical
Cable y Longitud Temperatura de trabajo	RG174 1 m -40 °C ~ 85 °C
Dimensiones	28 mm × 120 mm
Base magnética/adhesivo	

Elaboración: Por el autor.
Fuente: [25].

2.5.4. Antena GPS Conector UFL.

Proporciona alta ganancia, cuenta con base magnética. Se puede trabajar con cualquier receptor GPS (ver Figura 2.7). En la Tabla 2.7 se detalla las características.

Figura 2.7. Antena GPS.
Fuente: [26].

Tabla 2.7. Características de Antena GPS.

Parámetros	Características
Frecuencia	1575,42 MHz ± 3 MHz
VSWR ROE Ancho de banda	1.5:1 ~ 5 MHz
Impedancia	50 Ohm
Ganancia pico	3dBic Basado en 70 mm × 70 mm plano de tierra
Ganancia	-4dBic a -90 ° <0 <90 ° LNA / filtro
Cifra de ruido Potencia	1.5 dB 2.2 a 5 V
Dimensiones	43 mm × 33 mm × 14 mm
Cable y Longitud Temperatura	RG174 1 m -40 ° C ~ 85 ° C

Elaboración: Por el autor.
Fuente: [26].

2.6. Selección de batería.

Para dimensionar la carga de alimentación, se consideró dos dispositivos: Arduino Uno y Shield con chip. En la Tabla 2.8 se resume los valores típicos de voltaje y amperaje de los dispositivos seleccionados.

Tabla 2.8. Características de amperaje, voltaje, y potencia requerida del almacenamiento mínimo de dispositivos a alimentar.

Dispositivo	I típica, mA.	V típico, V	Consumo típico, W
Arduino Uno	50	7	0.35
Shield con Chip	100	7	0.70
TOTAL			**1.05**

Elaboración: Por el autor.
Fuente: [18], [23]

Para calcular el consumo diario de energía del equipo se lo realiza con ecuación 1 [27]:

$$Ed = (P * 24)f \qquad (1)$$

En dónde:

Ed, es el consumo diario de energía del equipo, Wh/día

f, es factor de reserva (25%)

P, es la potencia, W

$$Ed = 31.5\ Wh/día$$

La capacidad de almacenamiento de la batería, se calcula con ayuda de la ecuación (2) [27]:

$$C\ alm.bat = \frac{Ed}{Vs} \qquad (2)$$

En dónde:

C alm.bat, es la capacidad de almacenamiento de la batería, Ah/día

Ed, es el consumo diario de energía del equipo, Wh/día

Vs, es el voltaje de operación del sistema, V

$$C\ alm.bat = \frac{31.5\ Wh/día}{7V}$$

$$C\ alm.bat = 4.5\ Ah/día$$

El consumo diario de energía del prototipo es de 31.5 Wh/día y necesitamos que la capacidad de almacenamiento de la batería sea de 4.5 Ah/día para tener autonomía energética de un día, de acuerdo con las baterías comerciales se utiliza una de 12V y 7Ah/día como se observa en la Figura 2.8, y con ello obtuvimos una autonomía de 37 horas, las principales características de la batería seleccionada se detallan en la Tabla 2.9.

Figura 2.8. Batería
Fuente: [28].

Tabla 2.9. Características de la Batería.

Parámetros		Características
Voltaje Nominal		12 V
Capacidad (25 °C)	20HR (10.5 V)	7Ah
	10HR (10.5 V)	6.6Ah
	1HR (9.60 V)	4.58Ah
Dimensión	Largo	1511.5mm
	Ancho	651mm
	Altura	941mm
	Total Altura	1001mm
Peso aproximado		2.22kg (4.90lbs)5%
Tipo de terminal		T1/T2
Resistencia interna (Full carga, 25°C)		Aproximadamente 28mΩ
Auto-descargue (25°C)	3 meses	Capacidad restante 91%
	6 meses	Capacidad restante 82%
	12 meses	Capacidad restante 65%
Temperatura nominal de operación		25°C3°C(77°F5°F)

Elaboración: Por el autor.
Fuente: [28].

2.7. Plataforma de monitoreo.

Para la plataforma de monitoreo se requiere de un servidor que permita la lectura de los datos transmitidos por parte del dispositivo de rastreo, así mismo, se los debe almacenar a estos en una base de datos, para posteriormente mediante una página web visualizar el mapa de rastreo, y realizar los respectivos reportes. Para poder entender de una mejor manera de lo que se requiere debemos conocer los términos: servidor web, servidor de aplicaciones y base de datos.

2.7.1. Servidor web.

Consiste en ejecutar el servidor web en un ordenador manteniéndose a la espera de peticiones por parte de un cliente y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla. El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el

servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma [29] (ver Figura 2.9).

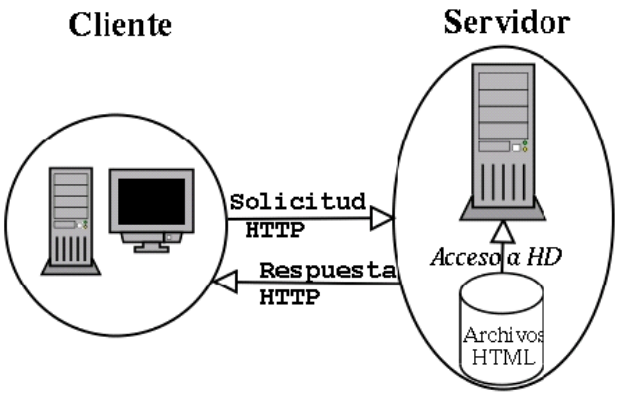

Figura 2.9. Arquitectura de funcionamiento de un servidor web.
Fuente: [29].

En la actualidad existen una multitud de productos que funcionan como servidores web, dentro de estos tenemos los siguientes: Apache, iPlanet web Server, Jigsaw, Microsoft IIS, Caudium, NCSA HTTPd, LiteSpeed web Server, Nginx, Oracle HTTP Server.

Sin duda una de las limitantes que tiene el servidor de Microsoft es que es que tiene licencia de propietario, además solo puede ejecutarse en el sistema operativo Windows, de igual manera luego de haber investigado las características de los demás servidores web mencionados, se verificó que presentan las mismas y/o otras limitantes, o que son utilizados para otros fines como es el caso de LiteSpeed web Server que fue diseñado específicamente para servidores de alto tráfico. El software Apache es ideal para este trabajo, está disponible para una amplia variedad de sistemas operativos como: Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows, OS/2, TPF, OpenVMS y eComStation, este es publicado bajo la licencia Apache, siendo este de código abierto.

2.7.2. Servidor de Aplicaciones.

El servidor de aplicaciones contiene el código que se va a encargar de ejecutar en nombre de los clientes que realicen la petición. Cuando un servidor de aplicaciones recibe una solicitud HTTP, éste también analiza la petición para determinar qué recurso se le ha solicitado. Generalmente, la petición concierne código ejecutable alojado en el servidor. Contrariamente a lo que haría un servidor web en la misma situación, no transfiere al cliente el código, sino que lo ejecuta y es el resultado de la ejecución de este código lo que se

reenvía al cliente [29]. La arquitectura de un servidor de aplicaciones se apoya en un modelo cliente-servidor de tres capas: (ver Figura 2.10).

- **Presentación:** una interfaz, generalmente gráfica que reside en los clientes. El ejemplo típico es un navegador.
- **Lógica de negocio:** donde reside el servidor de aplicaciones y el conjunto de programas a los que da soporte.
- **Almacenamiento:** generalmente una base de datos.

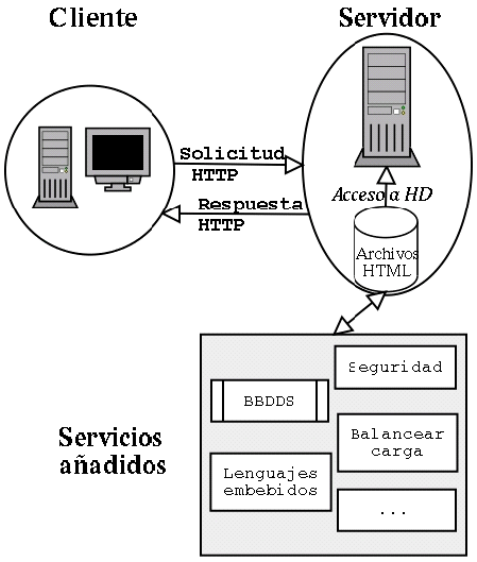

Figura 2.10. Arquitectura de funcionamiento de un servidor de aplicaciones.
Fuente: [29].

Existen algunos productos que funcionan como servidores web, dentro de estos tenemos los siguientes: BEA Weblogic Server, Borland AppServer, Allaire ColdFusion, Netscape application server, Oracle application server.

Para la programación de la interfaz, almacenamiento de datos y servidor web se utiliza el software NetBeans IDE para este caso la versión 8.0.2. NetBeans IDE permite fácil y rápidamente desarrollar escritorios Java, móviles y aplicaciones web, así como aplicaciones HTML5 con HTML, JavaScript y CSS. El IDE también proporciona un gran conjunto de herramientas para desarrolladores de PHP y C/C++. Es gratuito y de código abierto y tiene una gran comunidad de usuarios y desarrolladores de todo el mundo. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones [30].

2.7.3. Servidores de base de datos.

Un servidor de bases de datos se utiliza para almacenar, recuperar y administrar los datos. El servidor gestiona las actualizaciones de datos, permite el acceso simultáneo de muchos servidores o usuarios web y garantiza la seguridad y la integridad de los datos.

Existen algunos softwares de servidores de base de datos, los más conocidos son: Oracle, SQL Server, DB2, Sybase, MySQL.

MySQL es la base de datos más usada de alojamiento web. Se trata de una opción poderosa y de código abierto, diseñada para funcionar con el famoso lenguaje de programación PHP. SQL hace referencia a un lenguaje de consulta estructurado (Structured Query Language). Es la base de datos de código abierto número uno del mundo y es una excelente base de datos embebida. [31].

2.7.4. Pack de software stacks.

Para el presente trabajo de investigación se necesita un pack de software, el cual sea capaz de administrar o acceder a las operaciones más comunes (iniciar o apagar servicios, configuración, administración, gestión de logs, etc.) para los servidores antes mencionados.

Existen stacks que contienen todo lo necesario para hacer funcionar una aplicación web. Tradicionalmente, se suelen denominar WAMP (Windows + Apache + MySQL + PHP) o LAMP (Linux + Apache + MySQL + PHP), así mismo existe el XAMPP (Windows + Linux + OS + X + Apache + MySQL + PHP/PERL). Como se puede distinguir el nombre de cada uno de estos stacks hace referencia a: el servidor web (Apache), al servidor de base de datos (MySQL) y al lenguaje de programación (PHP), por otro lado, están los NMP Server (Nginx + MySQL + PHP). Pero como se describió anteriormente los servidores que se adecuan a nuestras necesidades son el Apache y MySQL, cuanto a los lenguajes de programación los más utilizados son: PHP, Perl y Python, por lo tanto, el stack que más se ajusta al presen te proyecto es el XAMPP [32].

XAMPP crea una distribución fácil de instalar para desarrolladores que se están iniciando en el mundo de Apache, viene configurado por defecto con todas las opciones activadas, es gratuito tanto para usos comerciales como no comerciales [32].

CAPÍTULO III

FABRICACIÓN DEL SISTEMA DE MONITOREO DE VEHÍCULOS RECOLECTORES DE RSM DE BAJO COSTO

3.1. Introducción.

El presente trabajo permite examinar la ubicación del vehículo recolector de RSM, indicar en mapas digitales su posición, velocidad y dirección registrada; además, permite visualizar datos importantes como la hora la fecha en la que se está realizando el monitoreo, analizar los caminos y rutas recorridas en forma detallada.

El sistema debe tener como elemento principal un GPS, tener la facilidad y compatibilidad de acceso a internet a través de un enlace GPRS/GSM, adicional a ello se necesita una tarjeta la cual se encargará del control de los dispositivos, envió, recepción y transmisión de los datos hacia un servidor en el cual se almacena la información, y contará con abastecimiento de energía autónomo.

En este capítulo se describe los parámetros de diseño, la fabricación en detalle y el proceso de implementación del prototipo.

3.3. Arquitectura del sistema.

El Sistema de monitoreo de la flota de vehículos de recolección y transporte de RSM está formado por tres módulos relacionados entre sí: 1) Módulo de procesamiento, 2) Módulo de adquisición y envió de datos, 3) Módulo servidor. La arquitectura del sistema y la interconexión de los bloques se muestran en la Figura 3.1.

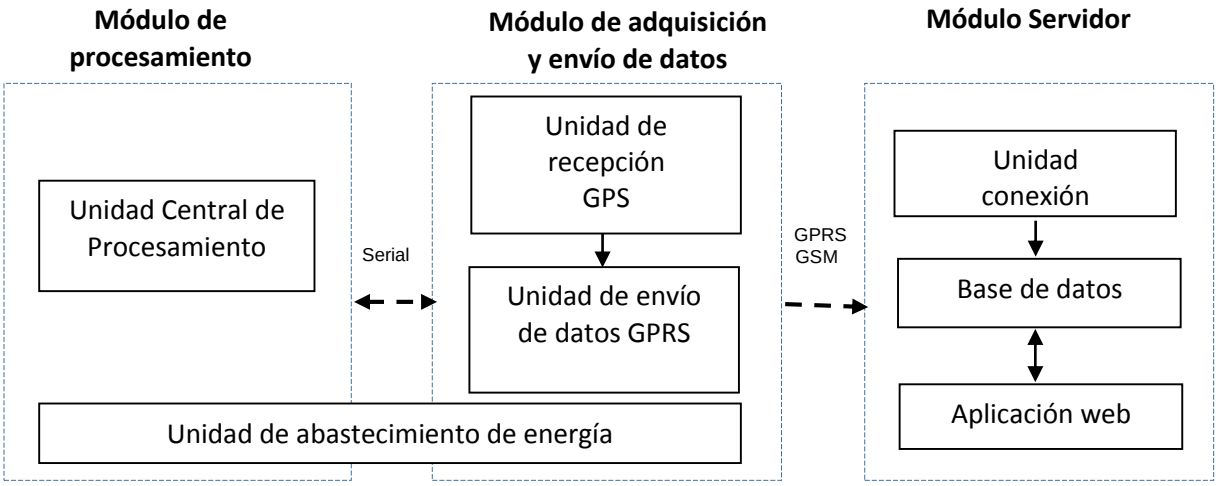

Figura 3.1. Arquitectura del sistema de monitoreo de la flota de vehículos de RT-RSM.
Elaboración: Por el autor.

3.3.1. Módulo de procesamiento.

El módulo de procesamiento está conformado por la unidad central de procesamiento y la unidad de abastecimiento de energía. La unidad central de procesamiento trabaja con Arduino Uno, encargado del control y administración. El algoritmo principal del microcontrolador empieza con la inicialización de librerías, declaración de variables y parámetros de conexión. La programación de las diferentes funciones se las realiza a través de subrutinas específicas, entre las subrutinas que forman parte del programa se encuentran: rutina de conexión serial, rutina de configuración del modem celular y rutina de conexión GPS. En la Figura 3.2 se muestra el diagrama de flujo del algoritmo principal de todo el procesamiento de los datos.

En la rutina de conexión serial de datos se realiza a través de USART del Arduino Uno y el shield a una velocidad de 9600 bps, como se muestra en la Figura 3.3. Este tipo de conexión serial se controla por software, para ello se puentea en el shield que la transmisión y recepción permitan este control. Los pines de conexión entre el Arduino y el shield se muestran en la Tabla 3.1.

Tabla 3.1. Pines de conexión entre el Arduino Uno y el shield.

Arduino Uno		**Shield SIM 908**
Pines Digitales	PB1	D9
	PB2	D10
	PB3	D11
	PB4	D12
	PB5	D13

Elaboración: Por el autor.

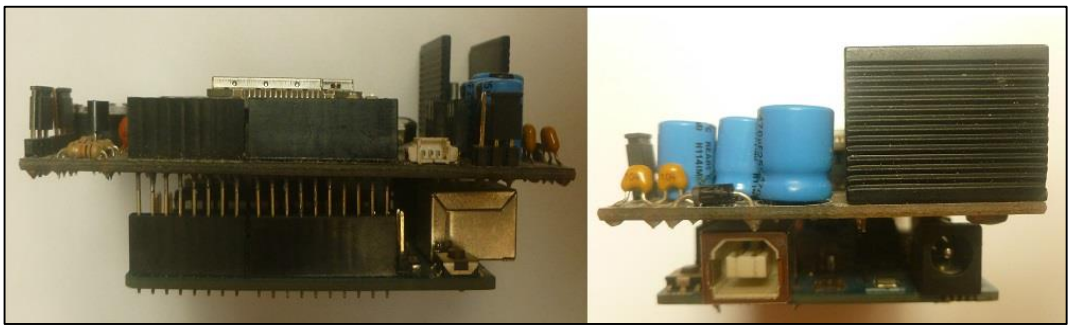

Figura 3.3. Conexión serial vista lateral y frontal.
Fotografía del autor.

La rutina de configuración del modem celular se encarga de enviar la trama hacia la red GPRS, para que pueda ser almacenada en la base de datos. La rutina de conexión GPS sirve para obtener las variables de longitud, latitud, altitud, tiempo y velocidad.

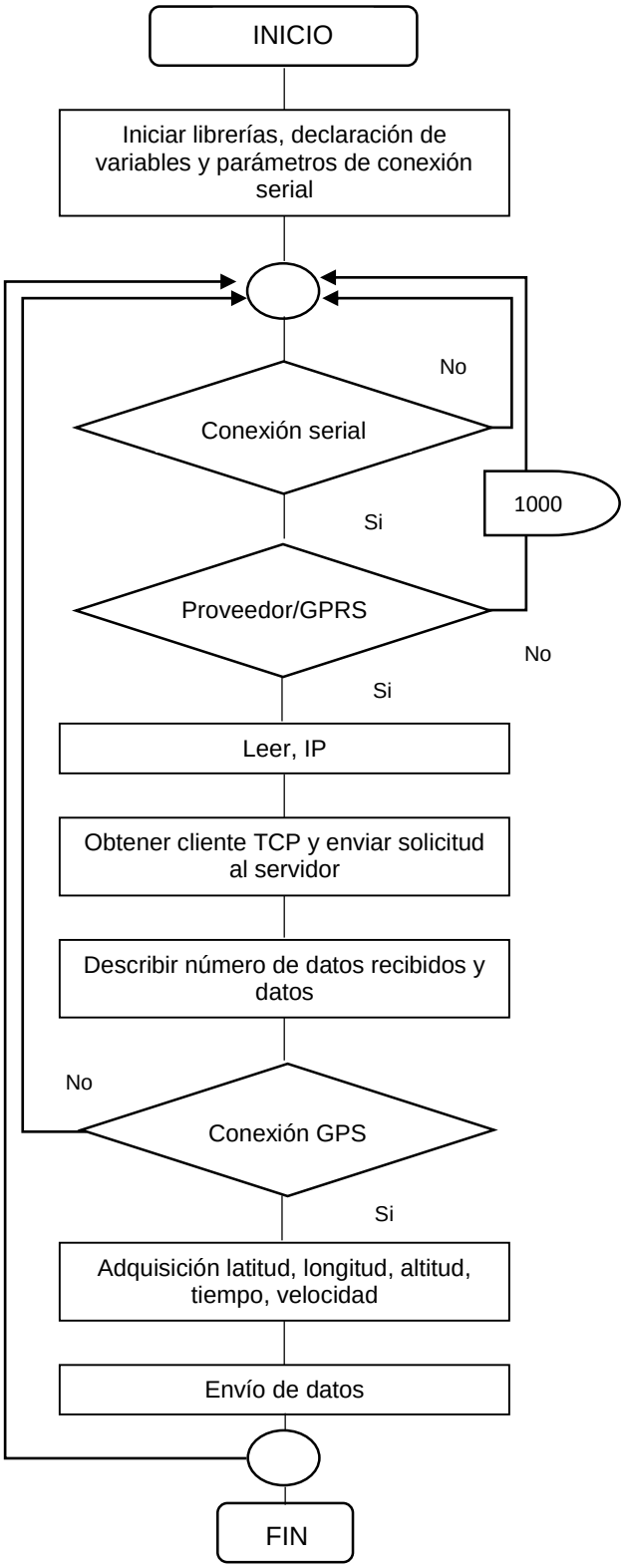

Figura 3.2. Diagrama de flujo del algoritmo principal.
Elaboración: Por el autor.

La configuración del sistema de localización incluye la realización de un script y la modificación de algunos parámetros de las librerías de los dispositivos compatibles con Arduino. En el Anexo B se presenta el script que se desarrolló para dar cumplimiento a los requisitos del sistema.

Entre los cambios a realizar en la librería *inetGSM.cpp,* se resalta los siguientes: selección de variables a enviar y especificación de la tarjeta Arduino Uno mediante la librería GSM.h. Los detalles de las modificaciones realizadas en la librería se muestran en el Anexo B.

3.3.2. Módulo de adquisición y envío de datos.

El módulo de adquisición y envío de datos consta de la unidad de recepción GPS, la unidad de envío de datos GPRS y la unidad de abastecimiento que energía. La unidad de recepción GPS es el encargado de detección de las variables (longitud, latitud, altitud, tiempo y velocidad) mediante la antena GPS, antes de usar el GPS es necesario puntear el J1 del shield SIM908 como se muestra en la Figura 3.4.

Figura 3.4. Jumper para el uso del GPS.
Fotografía del autor.

Y, la unidad de envío de datos GPRS transmite la información por conexión GPRS proporcionado por el proveedor de internet (ver Figura 3.5, fila 34). Este módulo se encuentra conectado mediante conexión serial al Arduino Uno en el cual se configura el protocolo de dirección IP, número de puerto del servidor, en este caso específico se utiliza el protocolo TCP/IP para transmisión de los datos en la red GPRS apuntando a nuestro servidor de aplicaciones cuya dirección IP pública es 200.0.29.28 y el número de puerto el 5080 como se observa en la Figura 3.5, fila 43.

```
//Configuración de la red GPRS
if(started) {
    if (inet.attachGPRS("internet.claro.com.ec", "", "")) //Establecer mi proveedor GPRS
        Serial.println("status=ATTACHED");
    else Serial.println("status=ERROR");
    delay(1000);
    //Lee la dirección IP
    gsm.SimpleWriteln("AT+CIFSR");
    delay(5000);
    gsm.WhileSimpleRead(); //Lee incluso cuando el serial buffer este vacio
    //Obtener el cliente TCP, envia una solicitud al servidor y guarda respuesta
    numdata=inet.httpGET("200.0.29.28", 5080, "/", msg, 50,lon,lat,alt,time,vel);
    //Imprimir resultados
    Serial.println("\nNumber of data received:");
    Serial.println(numdata);
    Serial.println("\nData received:");
    Serial.println(msg);
```

Figura 3.5. Configuración del módulo GPRS
Captura del autor.

A partir de los módulos descritos en la sección anterior se procede a conectar el hardware y cargar el script del Anexo B en el Arduino Uno (ver Figura 3.6).

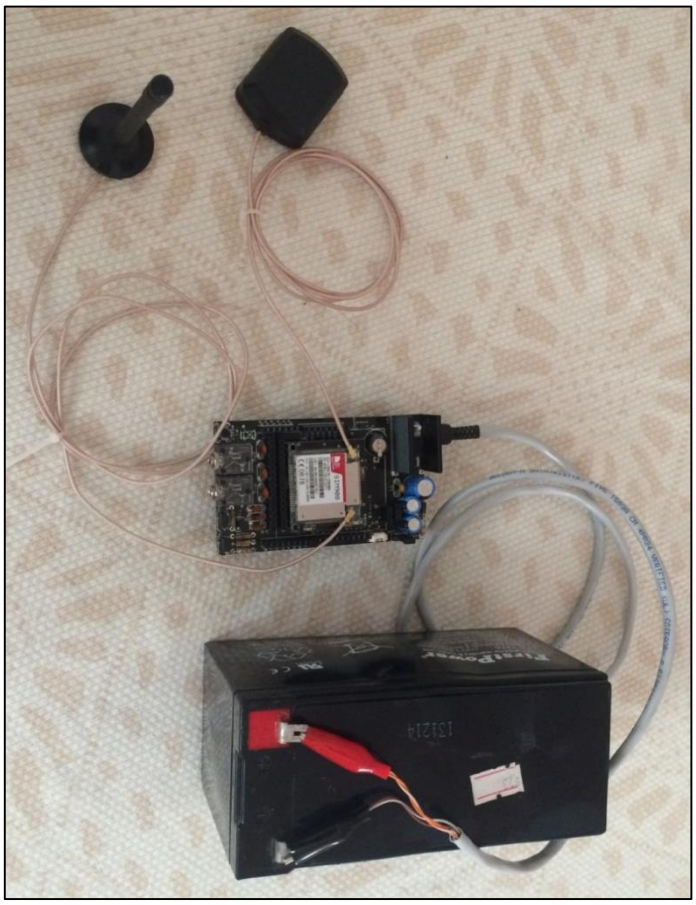

Figura 3.6. Prototipo de monitoreo
Fotografía del autor.

3.3.3. Módulo servidor.

El módulo servidor es la conexión a un ordenador para interpretar los datos, almacenarlos en la base de datos (BD) y finalmente presentar los resultados gracias a un servidor web, consta de: unidad conexión, base de datos y aplicación web.

3.3.3.1 Unidad conexión.

En la Figura 3.7 se muestra el diagrama de flujo de programación de la aplicación dedicada a la unidad conexión, el cual es encargado de: la conexión de la BD, arrancar y escuchar el puerto 5080, puerto por el cual el módulo de adquisición y envió de datos va a transmitir, realiza una compensación de la hora del dispositivo en este caso es -5 UTC debido a la ubicación del país, la conversión de tramas NMEA a grados, minutos y segundos y finalmente procesar las tramas adquiridas para el almacenamiento en la BD.

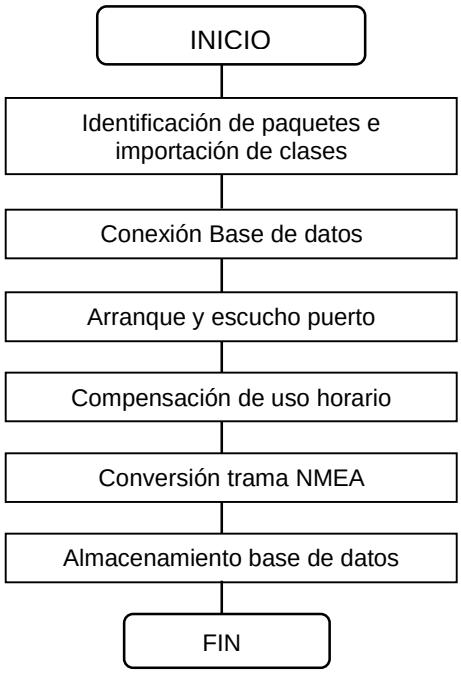

Figura 3.7. Diagrama de flujo de la programación de la aplicación.
Elaboración: Por el autor.

La unidad conexión es una aplicación desarrollada en NetBeans IDE en el cual se crea un archivo .jar que permite ejecutar aplicaciones escritas en lenguaje Java, se denomina ServerArduino. En la Figura 3.8 se muestra el desarrollo de la interfaz del módulo conexión.

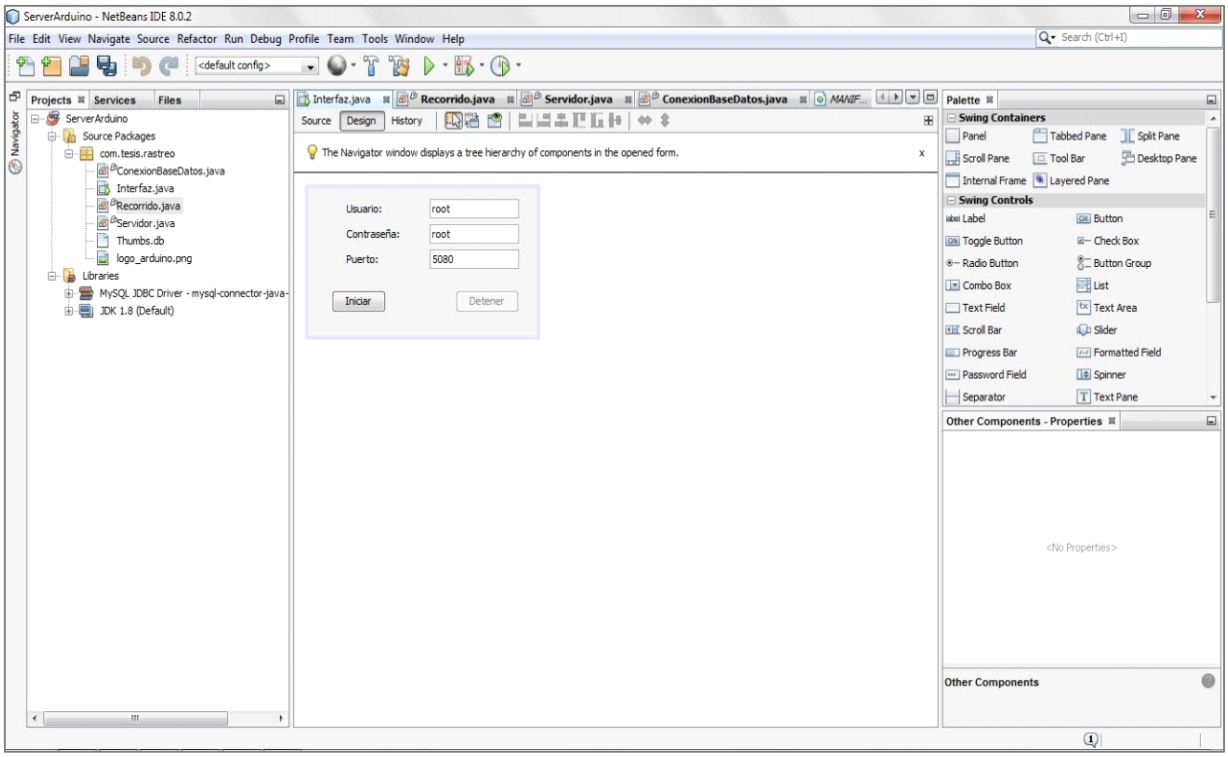

Figura 3.8. Configuración de la interfaz del módulo conexión.
Fuente: Captura del autor.

En la Figura 3.9 se muestra la interfaz gráfica de usuario del módulo conexión, en la cual se tiene que ingresar el usuario, la contraseña y el puerto que se utiliza para la recepción de datos.

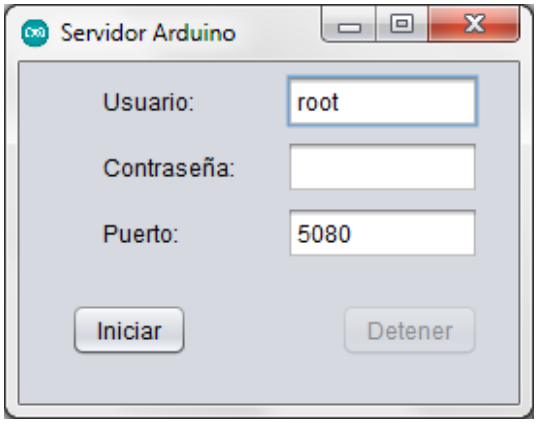

Figura 3.9. Interfaz gráfica de la aplicación.
Captura del autor.

En el Anexo C se detalla los scripts para el funcionamiento del módulo conexión, como son:

- **Script Servidor. -** Conecta el script recorrido y conexión base de datos, activa el puerto, transforma las tramas NMEA a grados, minutos y segundos.
- **Script Recorrido. -** Da formato a las tramas NMEA y almacena los datos recibidos del módulo en la base de datos.

- **Script Conexión base de datos. -** Establece la conexión con la base de datos.
- **Script Interfaz de conexión. –** Muestra el código de la interfaz gráfica de la aplicación.

La trama de los datos que envía el módulo de adquisición y envío de datos, lo recepta el módulo de conexión y contiene datos como: mensaje, longitud, latitud, fecha, hora y velocidad (ver Figura 3.10).

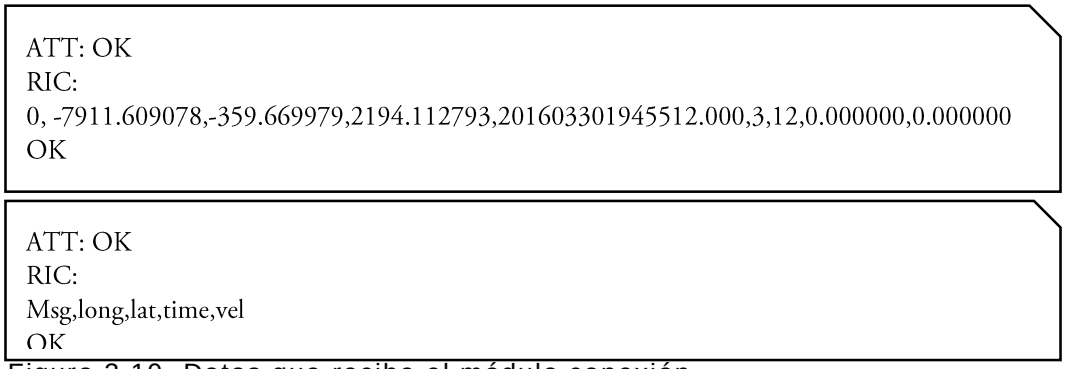

Figura 3.10. Datos que recibe el módulo conexión.
Elaboración: Por el autor.

3.3.3.2 Base de datos.

La base de datos se crea en un entorno ***MySQL Workbench*** con las características que se muestran en la Figura 3.11 con el nombre ***barduinodb*** el cual cuenta con las siguientes columnas:

- **id_recorrido. -** Almacena un ID diferente para cada dato ingresado*.*
- **fecha_hora_equipo. -** Fecha y hora en la cual se obtiene el dato.
- **latitud. -** Latitud del dato.
- **longitud. -** Longitud del dato.
- **velocidad.** - Velocidad del dato.

En el Anexo D se detalla la configuración de la base de datos.

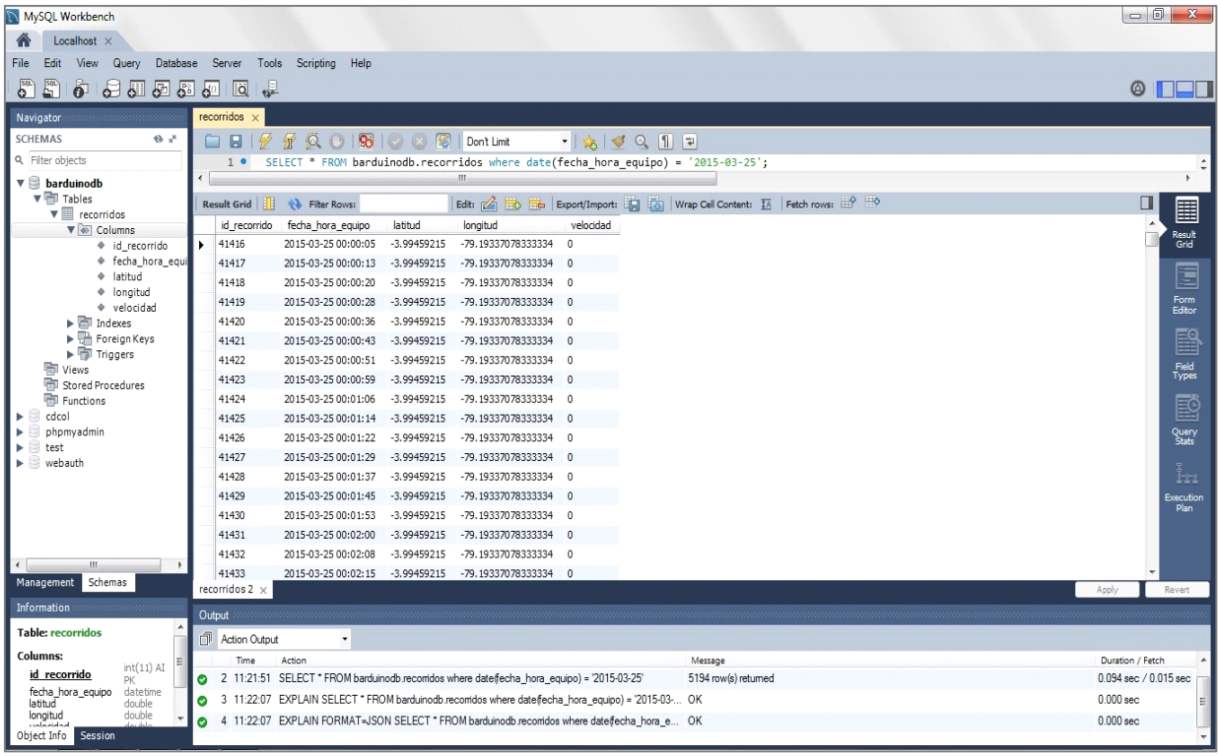

Figura 3.11. Características de la base de datos.
Elaboración: Por el autor.

3.3.3.3 Aplicación web.

La Aplicación web se desarrolla en lenguaje de programación PHP con el objetivo de mostrar la ubicación en tiempo real del vehículo recolector y trazar el recorrido. Para el desarrollo de la aplicación se necesita una IP pública con permisos en los puertos 8080 y 5080, esta es asignada por la UTPL y es la 200.0.29.28, sirve para realizar consultas a través de internet y para la conexión entre el módulo servidor y el prototipo. En el Anexo E se muestra los scripts para el funcionamiento del servidor web como son:

- **Índex. -** Conecta el script posición, script recorrido, y la base de datos, además se configura presentación y características del mapa.
- **Posición. -** Crea el identificador de la posición actual.
- **Recorrido. -** Permite setear un intervalo de tiempo para mostrar el recorrido.

Para poder ingresar a la aplicación web se debe colocar en el navegador el url: 200.0.29.28:8080/barduino (ver Figura 3.12).

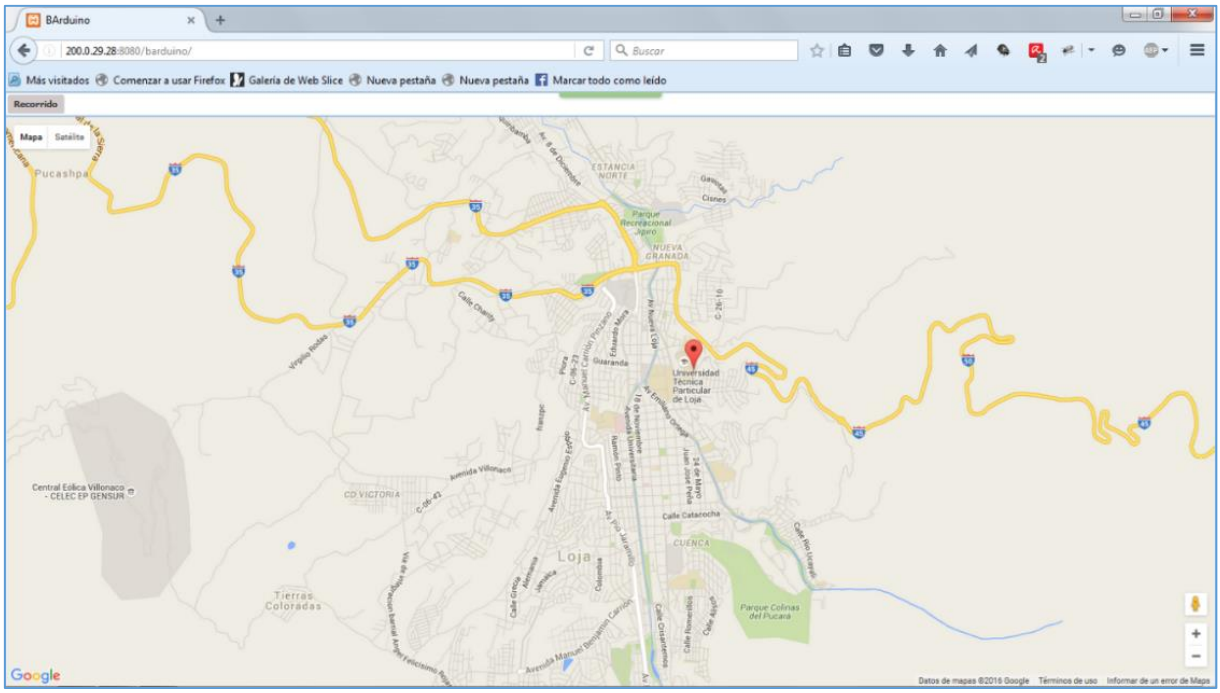

Figura 3.12. Aplicación web.
Captura del autor.

Por otro lado, el recorrido del vehículo recolector se lo realiza en un intervalo de tiempo determinado (ver Figura 3.13).

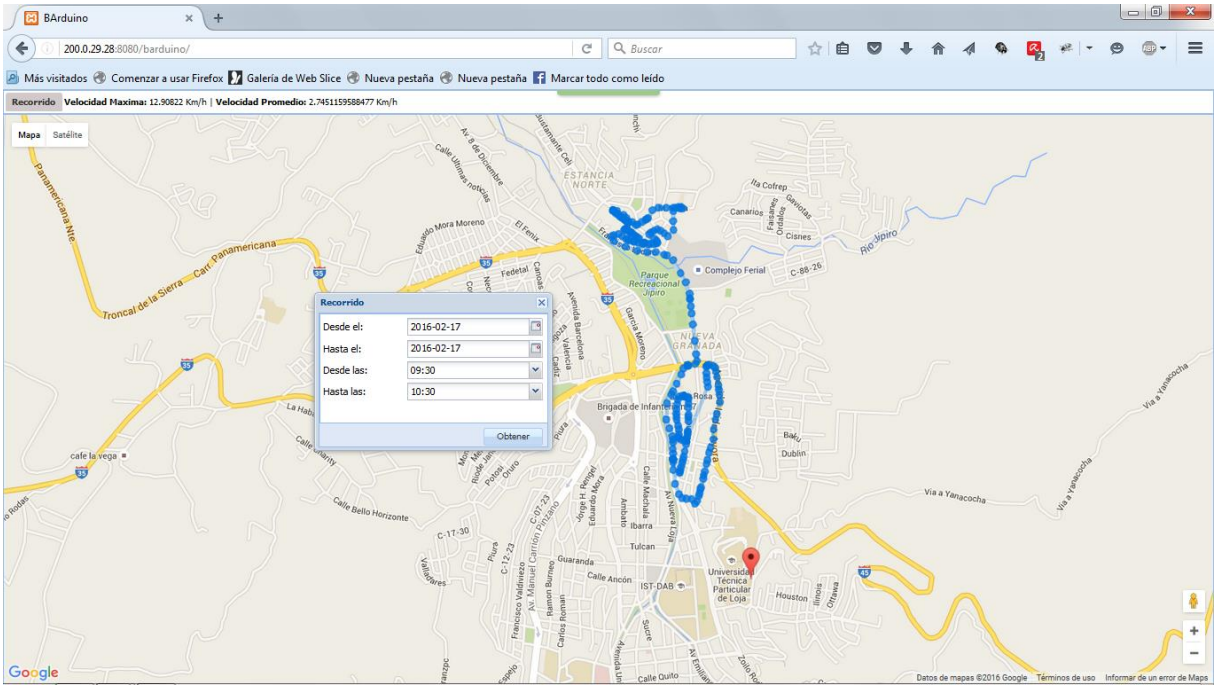

Figura 3.13. Recorrido del vehículo recolector.
Captura del autor.

3.4. Fabricación del sistema de monitoreo para vehículos recolectores de RSM de bajo costo.

En la Figura 3.14 se presenta la interconexión de todos los elementos de hardware para conformar el prototipo final de monitoreo, consta de: 1) Unidad central de procesamiento, unidad recepción GPS y unidad envío de datos GPRS, 2) Unidad de abastecimiento de energía, 3) Antena GPS y GPRS.

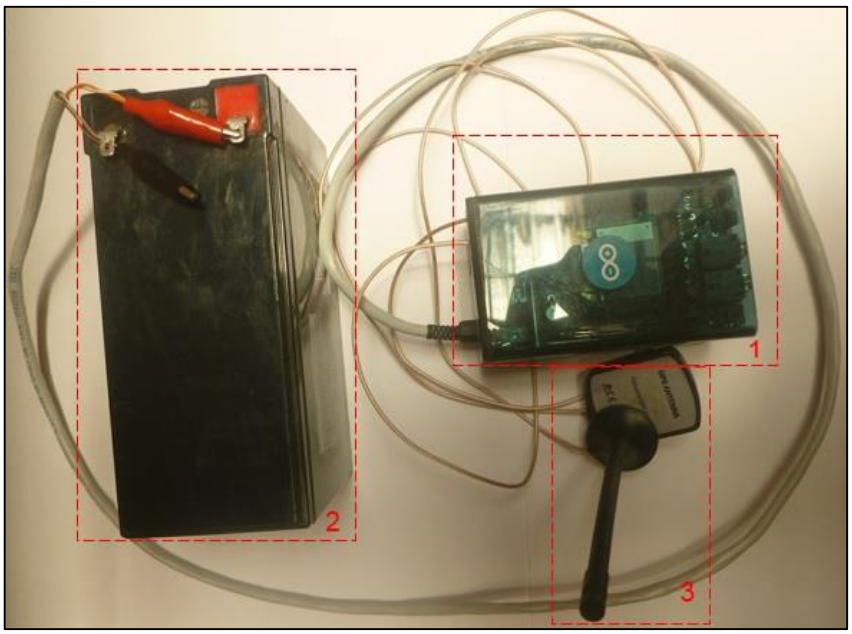

Figura 3.14. Integración del hardware del prototipo.
Elaboración: Por el autor.

Los gastos generados para el desarrollo del prototipo se detallan en la Tabla 3.2, como se observa el presupuesto empleado en la fabricación del prototipo es sumamente bajo comparado con las ventajas que ofrece y con los fabricantes mencionados en el Capítulo I, Tabla 1.2.

Tabla 3.2. Gastos generados para el desarrollo del prototipo.

Componente	Precio $
Arduino Uno	26
Shield	10
Chip SIM908	100
Antena GPS	12
Antena GPRS	7
Batería BLESSPOWER	30
Carcasa	15
Total	**200**

Elaboración: Por el autor.

El prototipo garantiza el principio de costo vs beneficio lo cual lo hace accesible a cualquier persona o institución. El prototipo fácilmente puede ser adaptado a otro vehículo.

En la Figura 3.15 se muestra la parte del software desarrollado, el cual consta de: 1) unidad conexión, 2) base de datos, y, 3) aplicación web, cabe recalcar que este costo es únicamente intelectual.

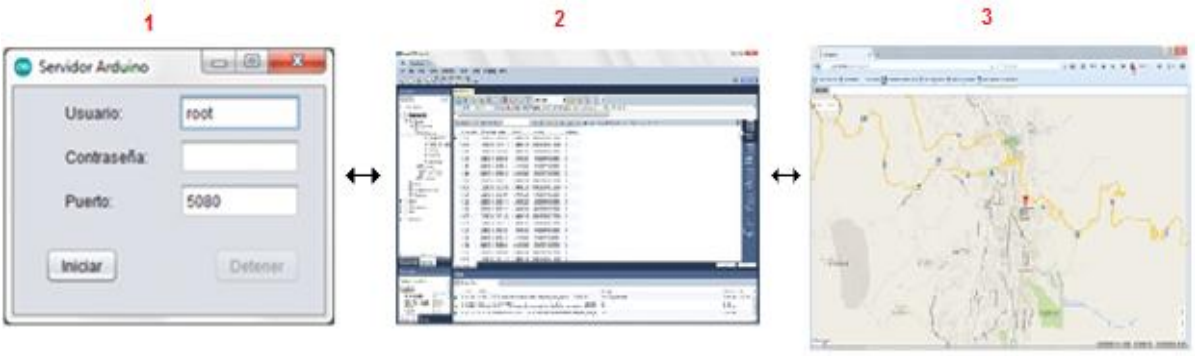

Figura 3.15. Integración del software para el sistema de monitoreo.
Elaboración: Por el autor.

En la Figura 3.16 se presenta el diagrama especifico del sistema de monitoreo para los vehículos recolectores de RSM.

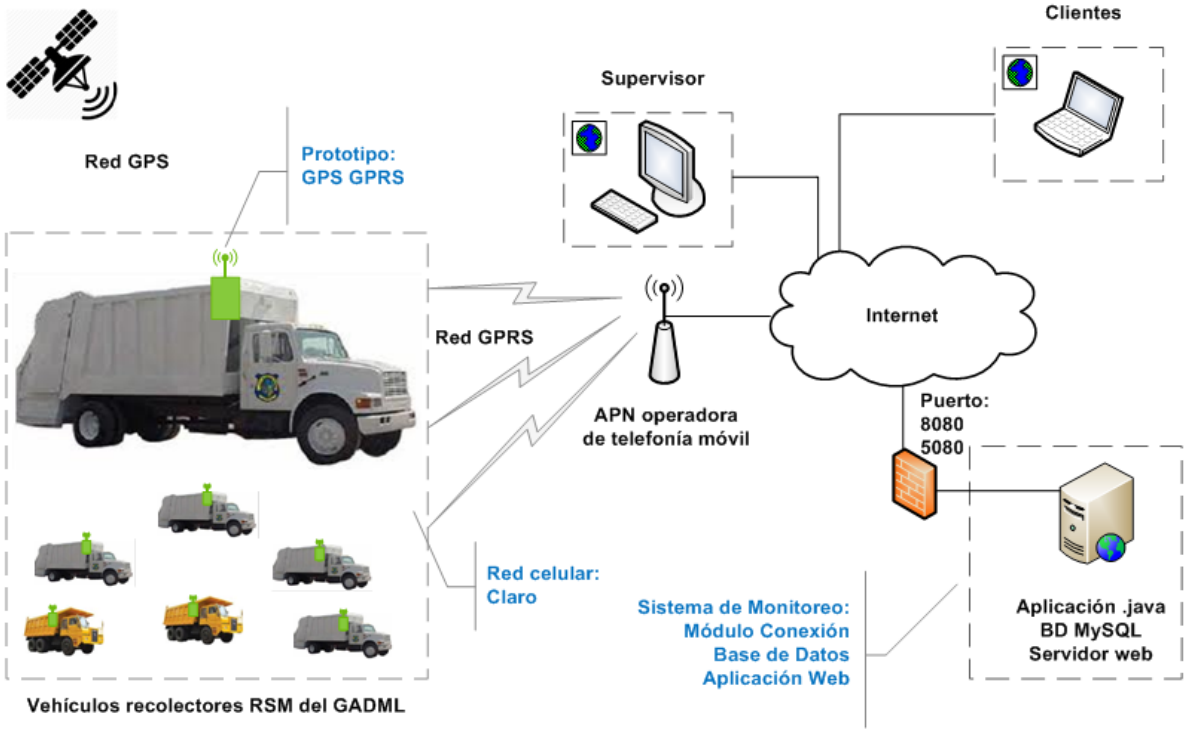

Figura 3.16. Diagrama especifico del sistema de monitoreo para los vehículos recolectores de RSM.
Elaboración: Por el autor.

CAPÍTULO IV

PRUEBAS Y ANÁLISIS DE RESULTADOS

4.1. Introducción.

En el presente capítulo se detalla las pruebas realizadas por el dispositivo de monitoreo de la flota de vehículos de recolección y transporte de RSM de la ciudad de Loja, así también se documenta el análisis de los resultados. Las pruebas realizadas fueron de dos tipos: de laboratorio y de campo. En las pruebas de laboratorio se comprobó la conexión entre los elementos: nodo remoto, cliente y servidor. En las pruebas de campo se realizó dos ensayos: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM, la cual cubre algunos barrios periféricos como: Las Palmas, Atamer, San Cayetano, entre otros.

4.2. Pruebas de conexión nodo remoto y servidor.

En esta etapa se realizó pruebas de la conexión nodo remoto-servidor y cliente-servidor, el nodo remoto es el dispositivo de monitoreo del vehículo de RT-RSM, el cliente es el usuario que accede al servidor por medio de una interfaz web, y finalmente el servidor es el equipo central de adquisición y registro de información, para la puesta en marcha del servidor se utilizó una dirección IP pública la cual está previamente parametrizada en el dispositivo de monitoreo, esta dirección tiene habilitados los puertos 5080 y 8080 para la recepción de datos y para la interfaz web, respectivamente (ver Figura 4.1).

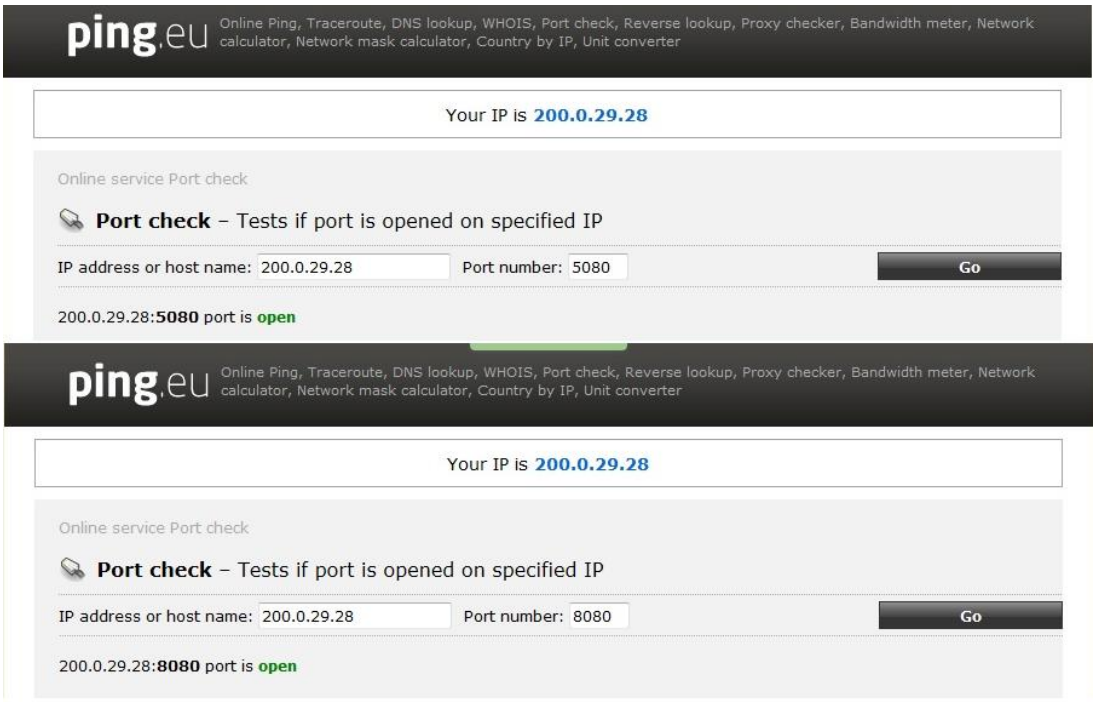

Figura 4.1. IP Pública y puertos habilitados.
Captura del autor.

Con el objetivo de comprobar la conexión entre nodo remoto y servidor, se realizaron pruebas iniciales de conectividad, en la Figura 4.2 se visualizan en la interfaz gráfica del servidor, los datos adquiridos por el dispositivo de monitoreo, lo cual evidencia la correcta comunicación entre los elementos del sistema.

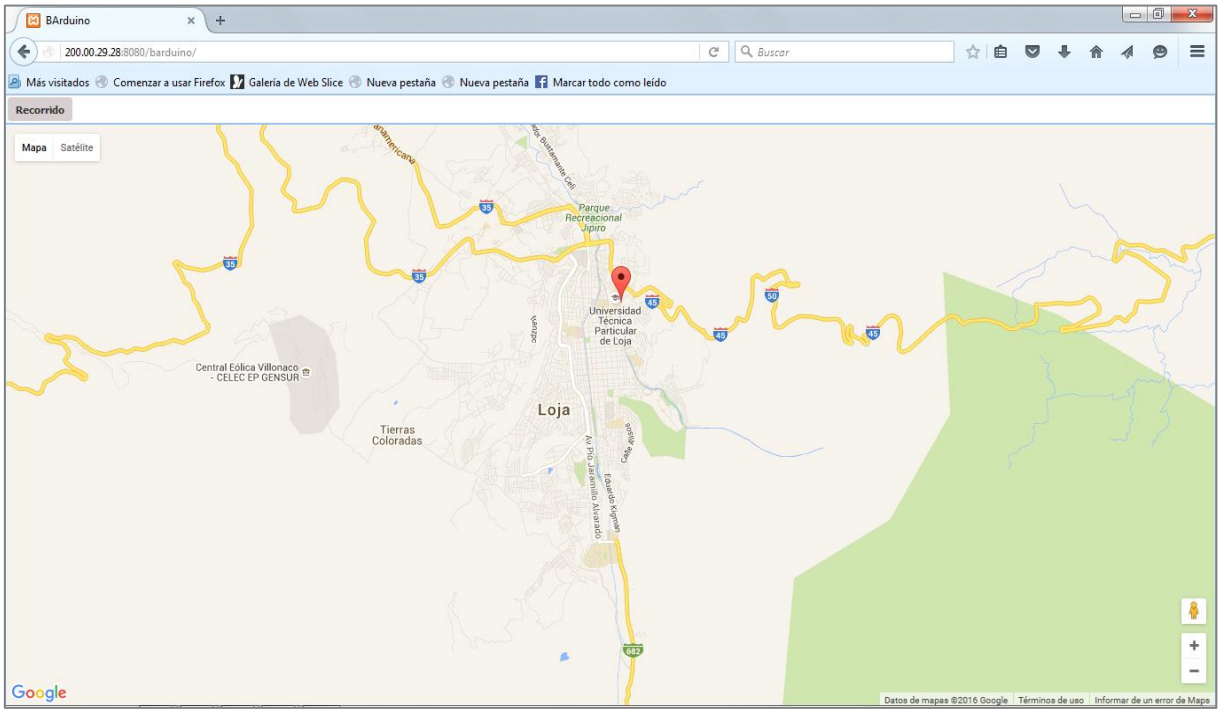

Figura 4.2. Interfaz gráfica.
Captura del autor.

4.3. Pruebas de campo del sistema.

En las pruebas de campo del sistema se realizó dos ensayos: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja realizado el día 17 de febrero de 2016 desde las 12h20 hasta las 12h55, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM realizado el día 15 de junio de 2016 desde las 9h18 hasta las 11h19.

4.3.1. Instalación del prototipo en el vehículo recolector.

La instalación del prototipo se lo realizó el día 17 de febrero de 2016 a las 07:00 horas en el inicio del recorrido de la Ruta 2 diurna (ver Figura 4.3).

Figura 4.3. Instalación del prototipo en el vehículo recolector de RSM.

4.3.2. Pruebas del sistema de monitoreo.

Para realizar las pruebas del sistema de monitoreo se debe conectar el prototipo con el servidor, para esto se ejecuta la aplicación realiza en el módulo conexión, como se muestra en la Figura 4.4.

Figura 4.4. Conexión del prototipo con el servidor.
Captura del autor.

El dispositivo de monitoreo envía las coordenadas de ubicación del vehículo recolector de RSM cada 8 segundos, este intervalo de tiempo es el mínimo que soporta el algoritmo del módulo de procesamiento debido a que ejecuta múltiples tareas de comunicación y procesamiento de datos. Una vez realizada la conexión entre los elementos del sistema, se comprueba la recepción de los datos en la base de datos MySQL residente en el servidor (ver Figura 4.5).

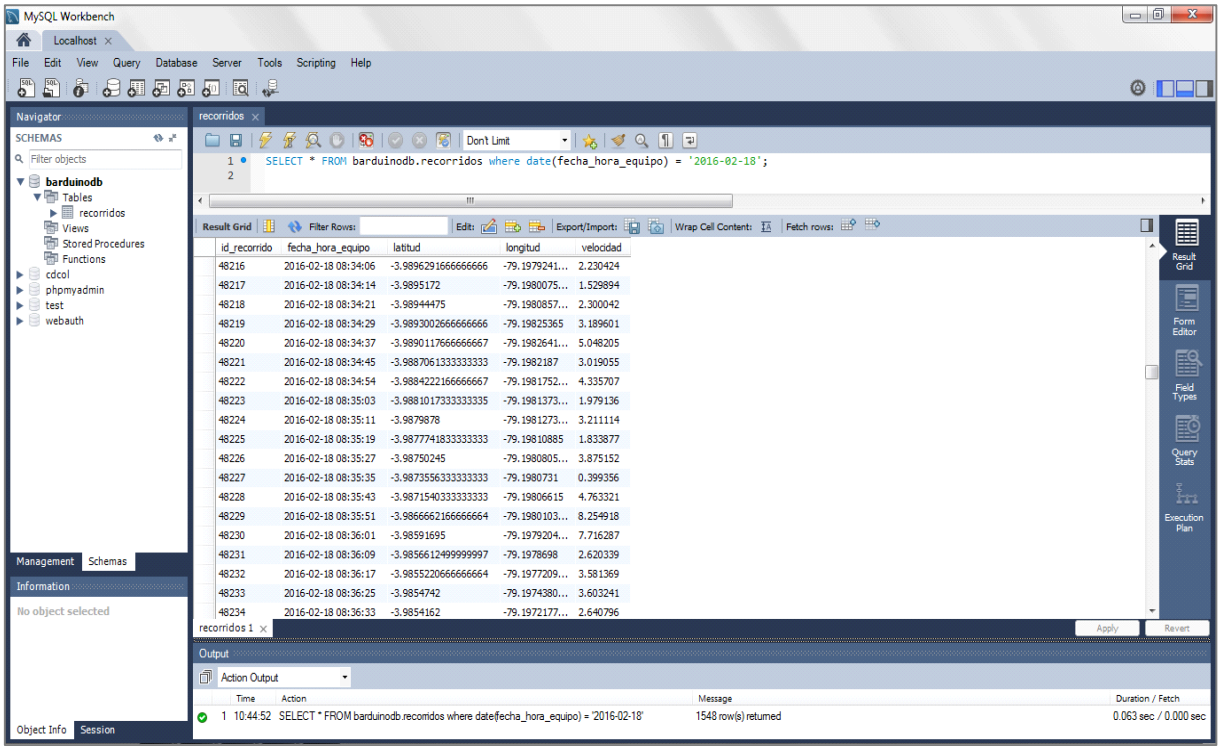

Figura 4.5. Información registrada en la base de datos MySQL.
Elaboración: Por el autor.

Con el objetivo de verificar los errores de magnitud correspondientes a las variables latitud y Longitud, se realizó una comparación entre el dispositivo de monitoreo y el GPS comercial Garmin Monterra (precisión menor a 3 metros en campo abierto y detenido, y menor a 15 metros en movimiento con el WAAS and EGNOS habilitados) [33], primeramente se comparó un punto específico y luego se compararon las dos rutas mencionadas.

Para verificar el error existente entre el dispositivo de monitoreo y el GPS Garmin Monterra, se registró un punto específico perteneciente a la ruta del ensayo 1, la Tabla 4.1 muestra los datos adquiridos por ambos dispositivos.

Tabla 4.1. Datos tomados de los dispositivos GPS.

GPS	LATITUD	LONGITUD	FECHA	HORA
Garmin	-4.0295100	-79.1990570	17/02/2016	12:32:20
Módulo	-4.02949095	-79.199105	17/02/2016	12:32:20

Elaboración: Por el autor.

Para comprobar el error que existe entre los puntos, se procede a graficar utilizando la aplicación de Google Earth Pro y medir la distancia de error entre ellos, el error determinado por la aplicación es de 5.74 m (ver Figura 4.6).

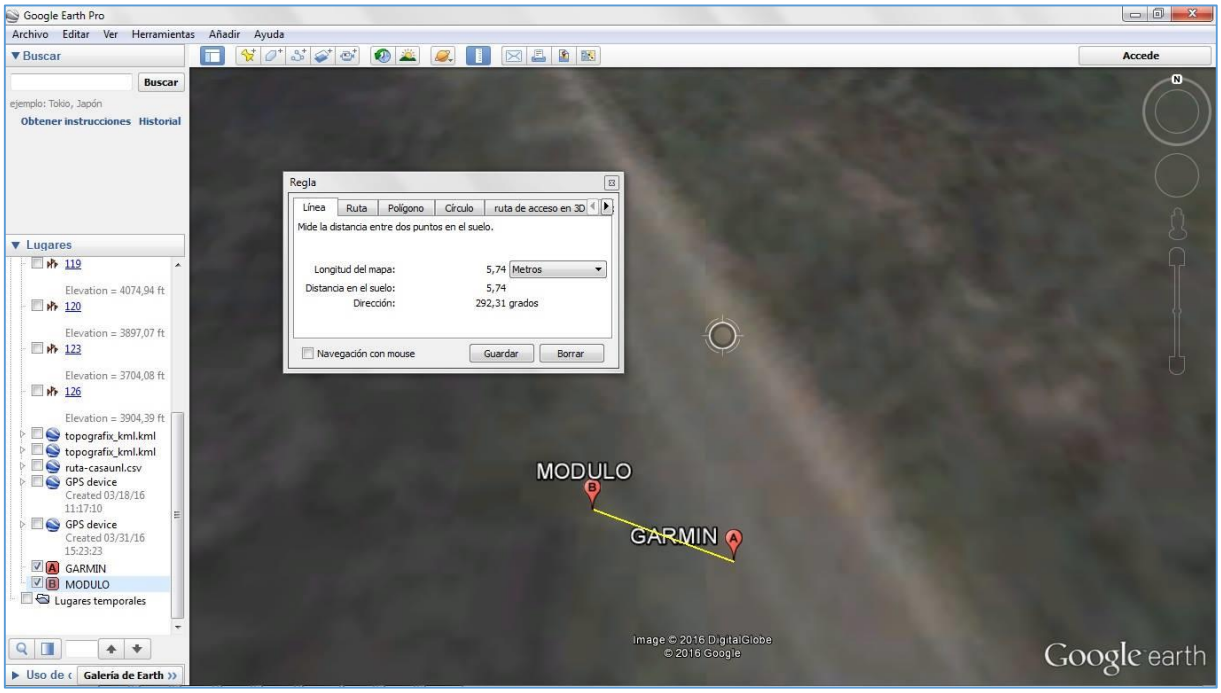

Figura 4.6. Distancia error entre el dispositivo de monitoreo y el GPS Garmin Monterra.
Elaboración: Por el autor.

Desde el punto de vista analítico el error existente entre dos coordenadas geográficas se calcula mediante la ecuación haversine (3) [34].

$$haversine\left(\frac{d}{R}\right) = haversine(\emptyset_1 - \emptyset_2) + cos(\emptyset_1)cos(\emptyset_2).haversine(\Delta\lambda) \quad (3)$$

Dónde:

$$haversine(\theta) = sin^2\left(\frac{\theta}{2}\right) \quad (4)$$

d = distancia entre dos puntos

R = radio de la esfera (radio ecuatorial 6378 km)

$\emptyset$ = latitud

λ = longitud

$\Delta\lambda$ = diferencia de longitud

$\Delta\phi$ =diferencia de latitud

Se Calcula la diferencia de latitud (5) y la de longitud (6) respectivamente, luego al despejar la distancia de la fórmula de haversine (4) se obtiene una nueva ecuación, que presenta algunos problemas para ciertos hemisferios [34], para solucionar estos se propone a fórmula directa de la distancia de haversine (9):

$$\Delta\emptyset = \emptyset_1 - \emptyset_2 \quad (5)$$

$$\Delta\emptyset = -4.0295100 - (-4.02949095)$$

$$\Delta\emptyset = -0.00001905$$

$$\Delta\lambda = \lambda_1 - \lambda_2 \quad (6)$$

$$\Delta\lambda = -79.199105 - (-79.1990570)$$

$$\Delta\lambda = -0.000048$$

$$A = sin^2\left(\frac{\Delta\emptyset}{2}\right) + cos(\emptyset_1).cos(\emptyset_2).sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (7)$$

$$A = sin^2\left(\frac{-0.00001905}{2}\right) + cos(-4.0295100).cos(-4.02949095).sin^2\left(\frac{-0.000048}{2}\right)$$

$$A = 2.08095e - 20$$

$$C = 2.atan2(\sqrt{A}, \sqrt{1-A}) \quad (8)$$

$$C = 2.atan2(\sqrt{2.08095e-20}, \sqrt{1-2.08095e-20})$$

$$C = 0.0000008984$$

$$d = R.C \quad (9)$$

$$d = (6378000m).(0.0000008966)$$

$$\boxed{d = 5.73\ m}$$

Como se observa en el cálculo de la distancia error por medio de las dos formas antes descritas, se concluye que la distancia error se encuentra alrededor de los 6 metros.

Para la prueba del prototipo instalado en el vehículo recolector se recorrió dos rutas: ruta prueba y ruta recorrido 2 diurna. La ruta de prueba inició en la urbanización Atamer y el recorrido cubre la Av. Zoilo Rodríguez, calle Vicente Rocafuerte, Av. Emiliano Ortega, calle Lourdes, calle Bolívar, Av. Eduardo Kigman, redondel UNL, Av. Pío Jaramillo Alvarado, calle Chile, calle Sucre, calle Saraguro, calle Sozoranga, calle Andrés Bello, calle Juan José Peña, calle Leopoldo Palacios, calle Macara, calle Lourdes, Av. Orilla de Zamora, calle 10 de agosto y finaliza en la Urb. Atamer donde inició. Para el trazo de la ruta se tomó los datos del prototipo y del GPS Monterra, se lo realizó en un tiempo de 35 minutos, a una velocidad promedio de 22 km/h y un recorrido de 12.7 km. El prototipo registró 198 posiciones y el GPS Monterra registró 416 posiciones. En la Figura 4.7 se muestra la ruta trazada por el GPS Monterra y en la Figura 4.8 se muestran el trazo de la ruta tomado por el prototipo en la aplicación web del sistema de monitoreo.

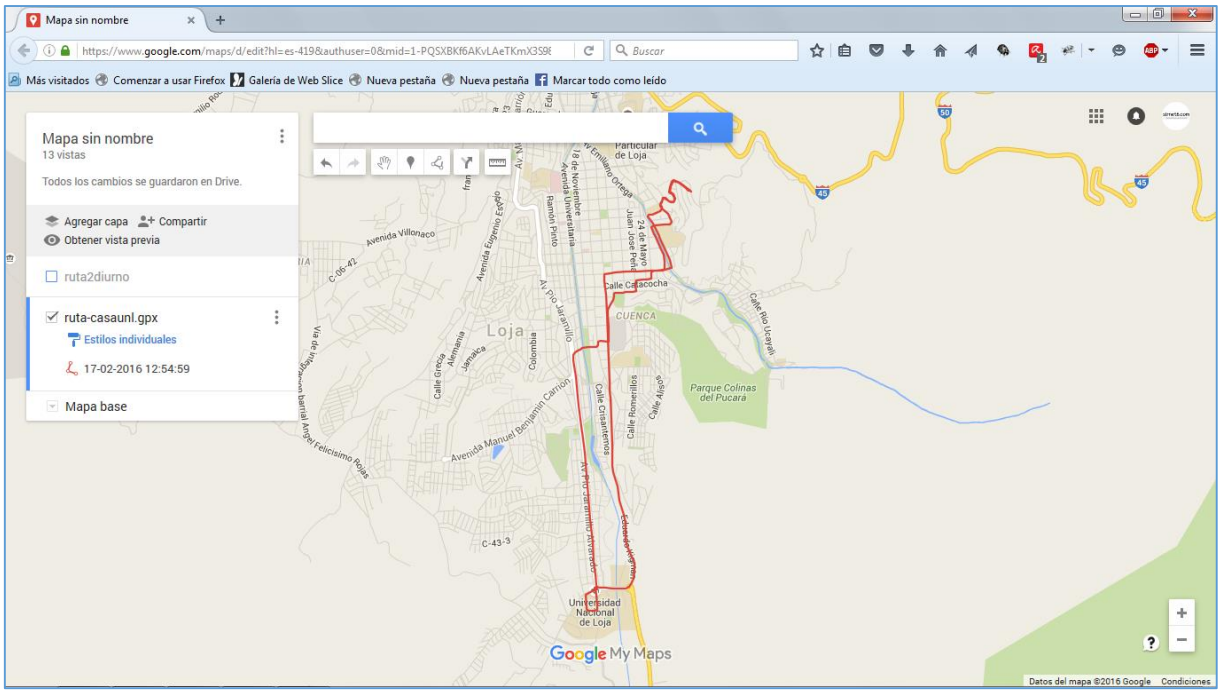

Figura 4.7. Trazo de la ruta del GPS Monterra.
Elaboración: Por el autor.

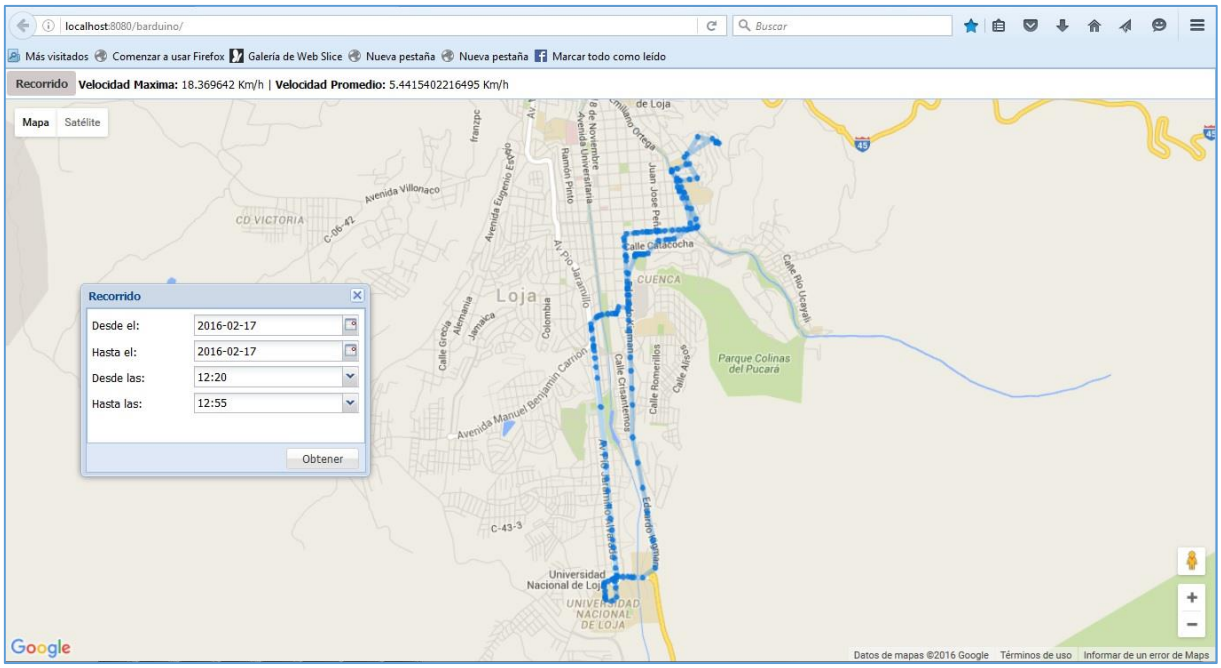

Figura 4.8. Trazo de la ruta del prototipo en la aplicación web.
Elaboración: Por el autor.

Por otro lado, la Ruta 2 diurna inicia en el Barrio el Churo, Barrio Santa Rosa, Barrio San Cayetano Alto, Barrio Las palmas Alto, Barrio Las Palmas, Barrio Isaac Ordoñez, Barrio San Cayetano Bajo, vía Zamora, Barrio Yanacocha, Barrio el Valle, Barrio el Recreo del Valle, Barrio la Samana, Barrio la Estancia Norte, Barrio Nueva Granada. Para el trazo de la ruta 2 diurna se tomó los datos del prototipo y del GPS Monterra, se lo realizó en un tiempo de 2 horas, a una velocidad promedio de 11.6 km/h y un recorrido de 15.6 km. El prototipo registró 490 posiciones y del GPS Monterra registró 796 posiciones. En la Figura 4.9 se muestra la ruta trazada por el GPS Monterra y en la Figura 4.10 se muestran el trazo de la ruta tomado por el prototipo en la aplicación web del sistema de monitoreo.

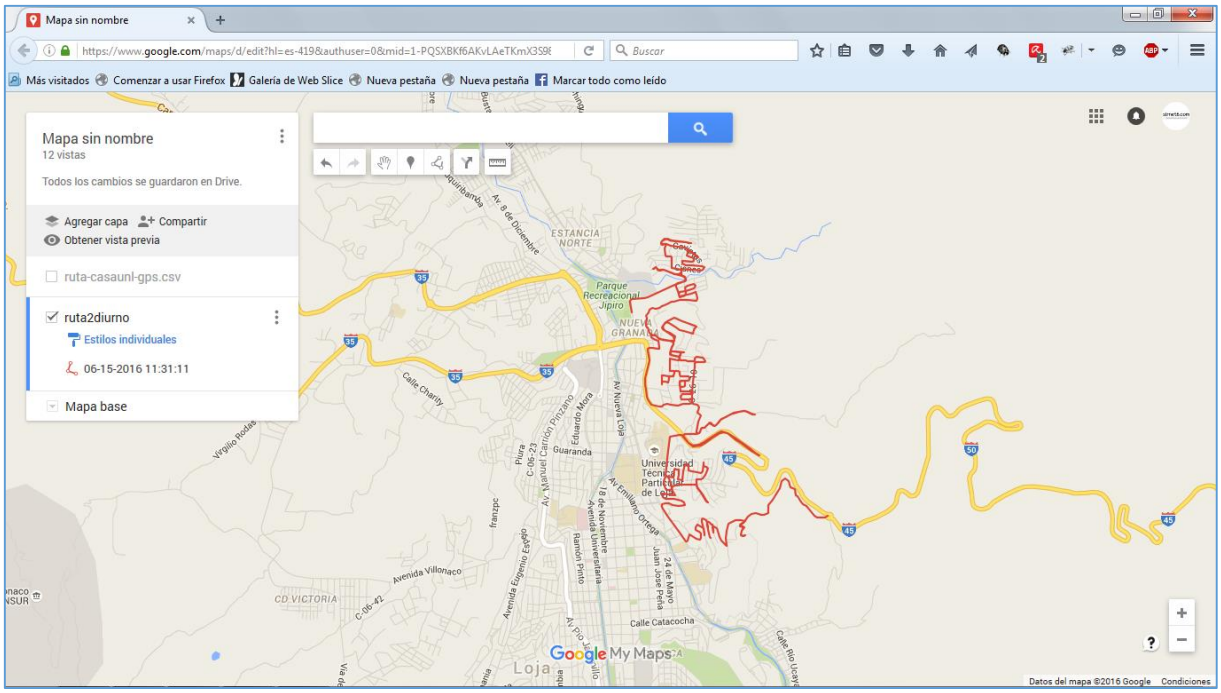

Figura 4.9. Ruta 2 diurna trazada por el GPS Monterra.
Elaboración: Por el autor.

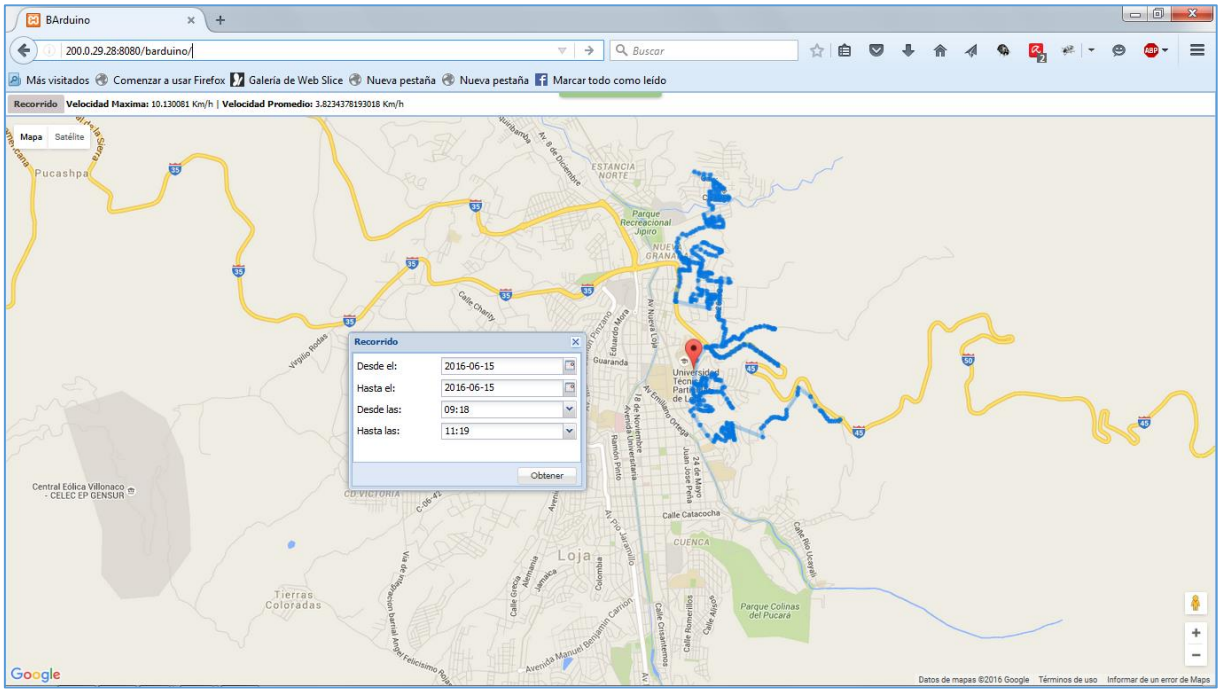

Figura 4.10. Ruta 2 diurna trazada por el prototipo en la aplicación web.
Elaboración: Por el autor.

Como se puede ver en la Figura 4.9 y la Figura 4.10 a pesar de ser registrada la información con diferentes equipos se concluye que su trazo es similar y las variaciones de las gráficas son menores, sin embargo, hay datos faltantes en el prototipo debido al nivel bajo de la señal de cobertura del proveedor de internet en parte del recorrido de prueba.

El principal inconveniente que existió durante las pruebas fue la falta de cobertura móvil en lugares de acceso limitado, por lo que no permitió obtener los datos en partes del recorrido que realizó el vehículo recolector. En la Figura 4.11 se observa las zonas obscuras en las cuales se pierde señal y recepción de datos, esto no depende del sistema de monitoreo sino de la operadora móvil proveedora del servicio de telefonía móvil.

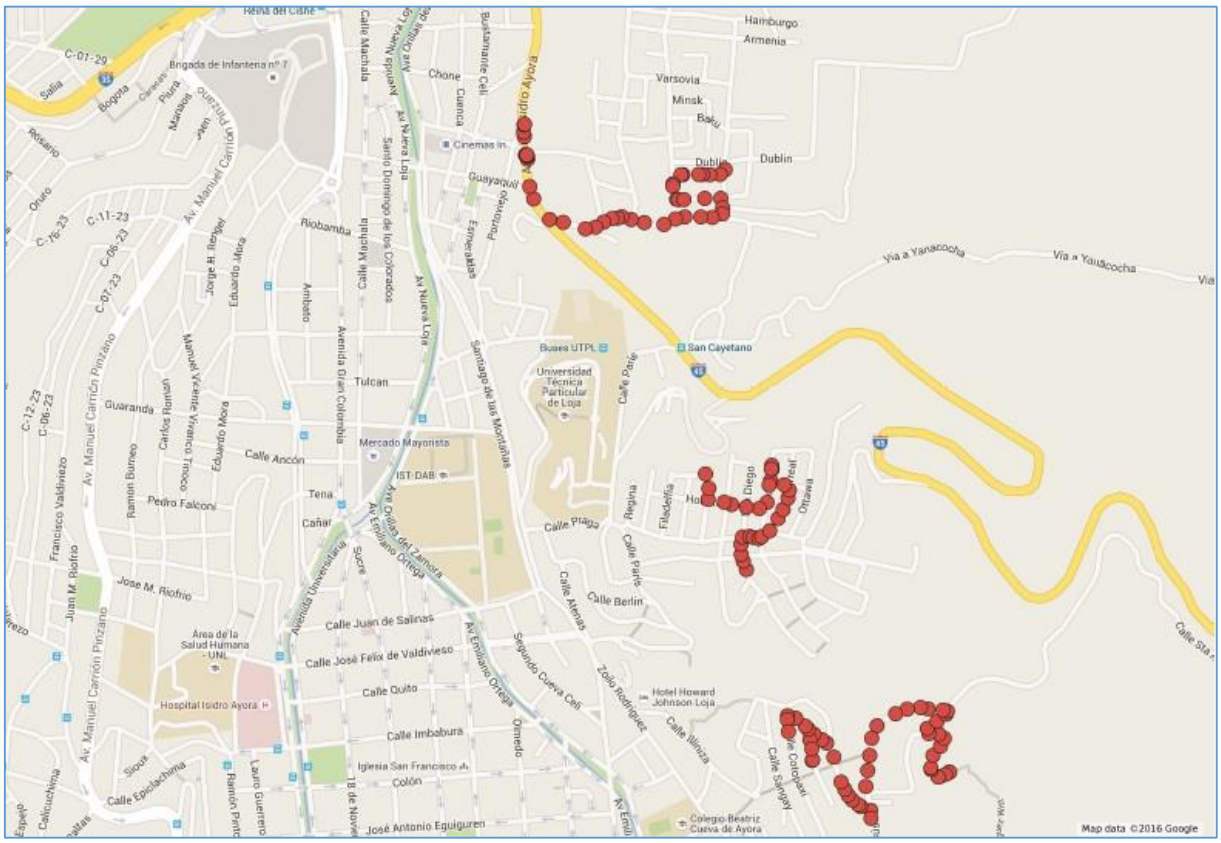

Figura 4.11. Zonas obscuras de la ruta 2 diurna trazada en la aplicación web.
Elaboración: Por el autor.

4.4. Análisis de resultados.

Al terminar las pruebas de funcionamiento del sistema se obtuvo diversos resultados que fueron comparados con el GPS Monterra y analizados por un programa desarrollado en LabVIEW. Para tener el margen de error del sistema de monitoreo se graficaron las variables latitud y longitud con respecto al tiempo de los dispositivos antes mencionados, en las dos rutas.

4.4.1. Análisis de la ruta de prueba.

Se realiza la comparación de la variable latitud respecto al tiempo de los dos GPS, obteniendo que el error máximo es 7.51 m y el error promedio 1.96 m (ver Figura 4.12).

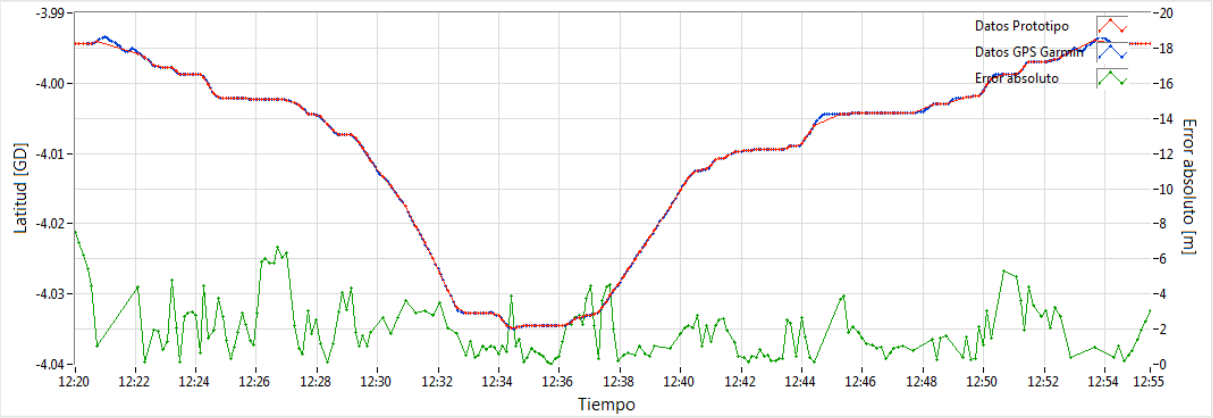

Figura 4.12. Latitud con respecto al tiempo de la ruta de prueba.
Elaboración: Por el autor.

Se realiza la comparación de la variable longitud respecto al tiempo de los dos GPS, obteniendo que el error máximo es 8.20 m y el error promedio 2.24 m (ver Figura 4.13).

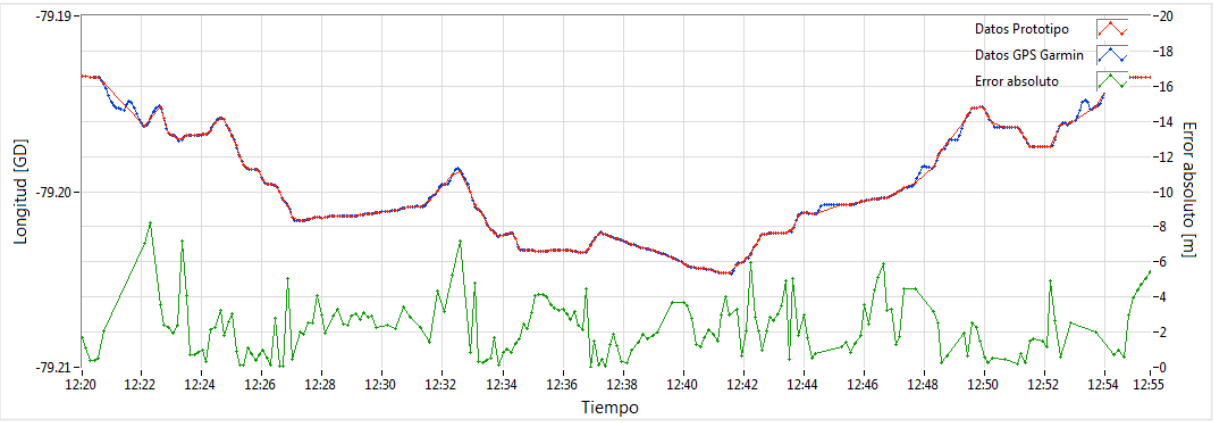

Figura 4.13. Longitud con respecto al tiempo de la ruta de prueba.
Elaboración: Por el autor.

Se grafica el error de las variables latitud y longitud y el error total del prototipo obteniendo que el error máximo es 8.3 m y el error promedio 3.35 m (ver Figura 4.14).

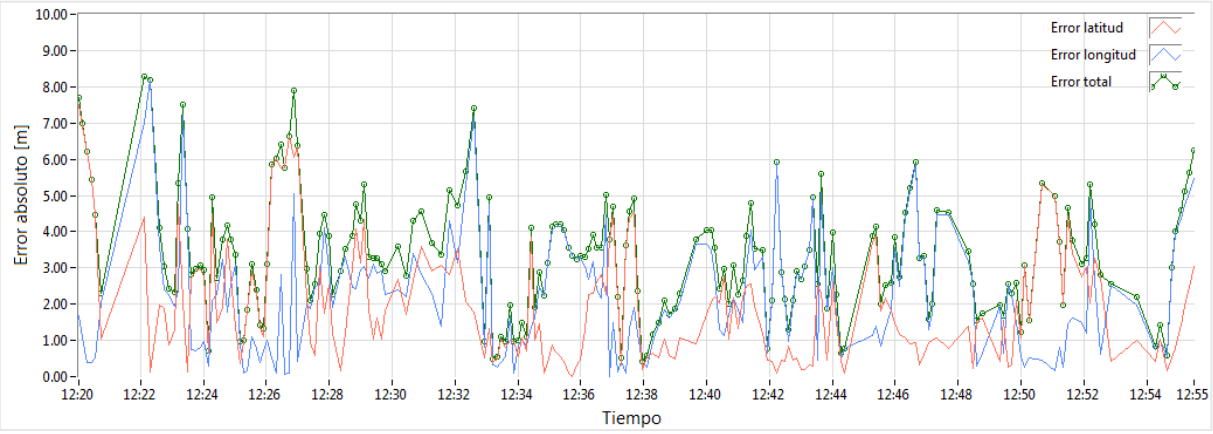

Figura 4.14. Error absoluto con respecto al tiempo de la ruta de prueba.
Elaboración: Por el autor.

Luego de analizar la ruta de prueba se puede concluir que el error absoluto entre dispositivo es 3.35 m, siendo este un valor aceptable para el prototipo.

4.4.2. Análisis de la ruta 2 diurna.

Se realiza la comparación de la variable latitud respecto al tiempo de los dos GPS, obteniendo que el error máximo es 17.43 m y el error promedio 2.25 m (ver Figura 4.15).

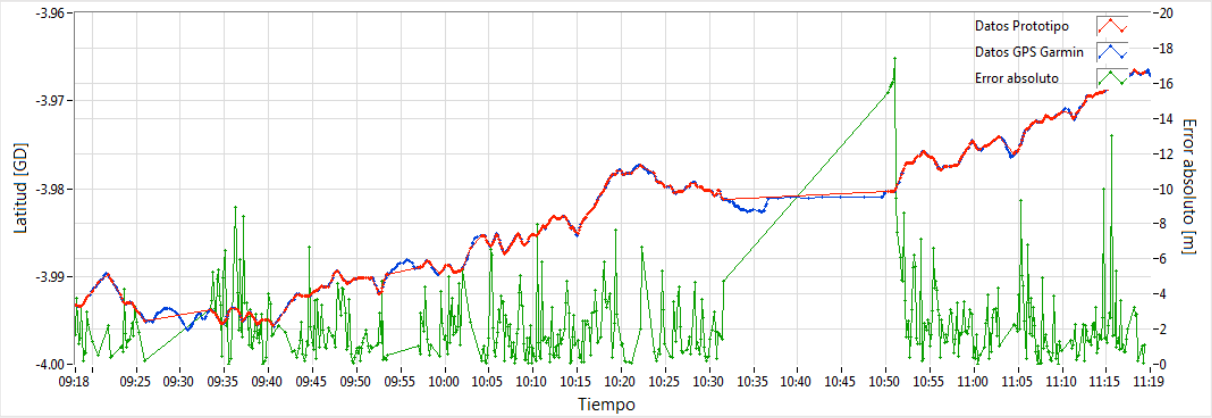

Figura 4.15. Latitud con respecto al tiempo de la ruta 2 diurna.
Elaboración: Por el autor.

Se realiza la comparación de variable longitud respecto al tiempo de los dos GPS, obteniendo que el error máximo es 15.40 m y el error promedio 2.44 m (ver Figura 4.16).

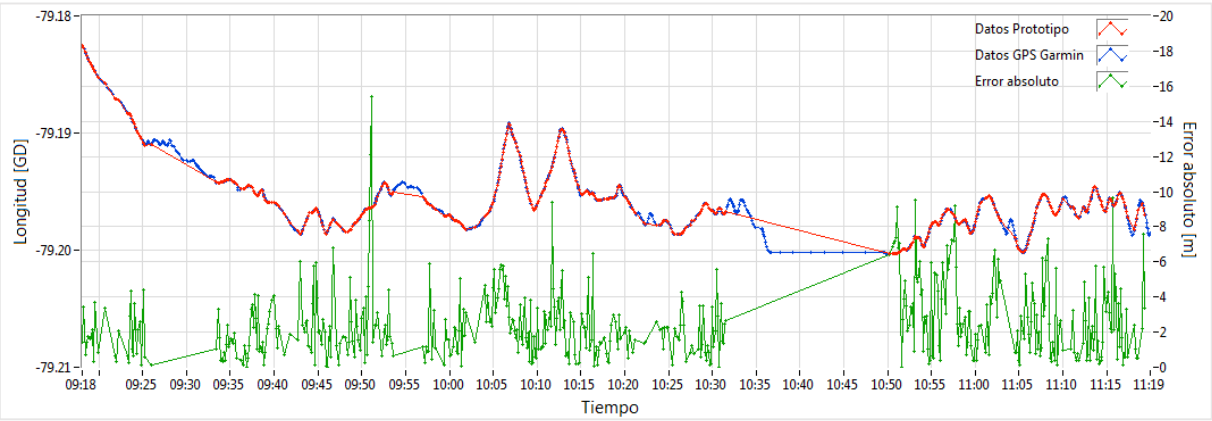

Figura 4.16. Longitud con respecto al tiempo de la ruta 2 diurna.
Elaboración: Por el autor.

Se grafica el error de las variables latitud, longitud y el error total del prototipo obteniendo que el error máximo es 19.68 m y el error promedio 3.67 m (ver Figura 4.17).

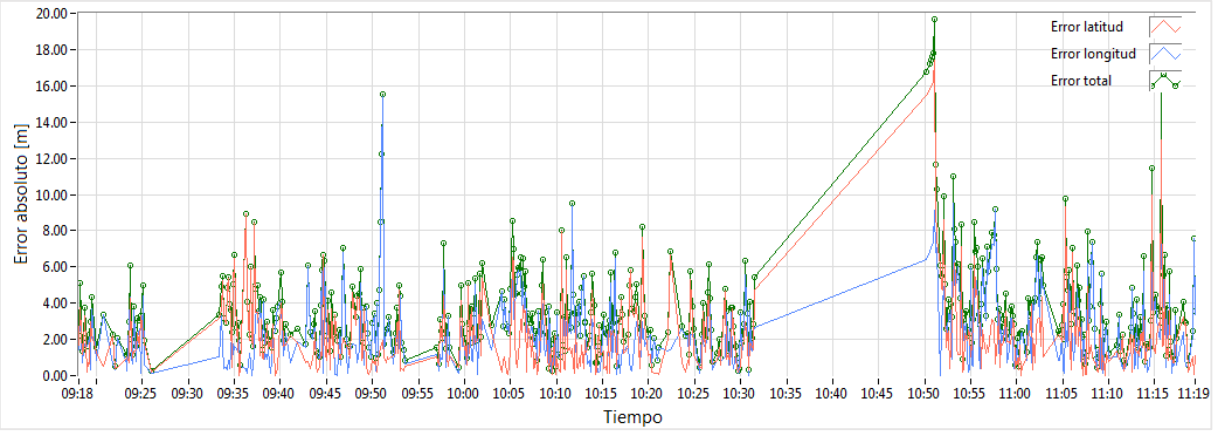

Figura 4.17. Error absoluto con respecto al tiempo de la ruta 2 diurna.
Elaboración: Por el autor.

Luego de analizar la ruta 2 diurna se concluye que el error absoluto promedio y máximo es de 3.67 m y 19.68 m, respectivamente. Este nivel de error no es significativo para la futura aplicación del dispositivo de monitoreo en el seguimiento del sistema de RT-RSM, existen lugares donde es baja la señal de cobertura y debido a ello el prototipo no envía los datos adquiridos. La solución planteada para eliminar este evento es reemplazar la antena de recepción GPRS (ganancia 3.5dBi) por una antena con mayor ganancia.

CONCLUSIONES

- Se desarrolló un sistema de monitoreo conformado por tres módulos principales: 1) módulo de procesamiento que contiene la unidad de procesamiento, 2) módulo de adquisición y envío de datos que consta de unidad de recepción GPS y unidad de envío de datos GPRS, estos dos módulos (prototipo AVL) están conectados a la unidad de abastecimiento de energía y de manera serial entre ellos, y 3) módulo servidor que consta de la unidad conexión, base de datos y aplicación web, este módulo se conecta mediante conexión GSM/ GPRS al prototipo AVL.

- Se realizó dos pruebas de campo: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja realizado el día 17 de febrero de 2016 a una velocidad promedio de 22 km/h y un recorrido de 12.7 km, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM realizado el día 15 de junio de 2016 a una velocidad promedio de 11.6 km/h y un recorrido de 15.6 km. Como resultado se obtuvo un error promedio de posición de 3.67 m y un error máximo de 19.68 m. Este nivel de error no es significativo para la futura aplicación del dispositivo de monitoreo en el seguimiento del sistema de RT-RSM.

RECOMENDACIONES

- El prototipo AVL debe contar con una carcasa de protección con el objetivo de evitar daños e interferencia en la obtención de datos, además debe contar con un mecanismo de sujeción al vehículo para mantenerlo estático.

- Para trabajos futuros es recomendable que la siguiente fase del software incluya ingeniería de rutas, controle, adquiera variables y evaluar parámetros como: puntos de control de las rutas de recolección, puntos de control de los contenedores, control de llenado de contenedores, nodos de recolección masiva (tolvas), capacidades de los vehículos recolectores, topología de rutas y sectorización, asignaciones de rutas por día por cada vehículo, kilómetros recorridos por ruta, tiempos de recorrido y toneladas recolectadas diariamente.

- Existen lugares donde la señal de cobertura es baja y debido a ello el prototipo no envía los datos adquiridos. La solución planteada para eliminar este evento es reemplazar la antena de recepción GPRS (ganancia 3.5dBi) por una antena con mayor ganancia.

BIBLIOGRAFÍA

[1] J. Ojeda, “Análisis situacional de la gestión de residuos sólidos urbanos casos: Catalunya (España) y Loja (Ecuador)”, Tesis de Maestría Universidad Autónoma de Madrid, 2009.

[2] Proyecto de Gestión Integral de Residuos Sólidos (Loja, Ecuador). [Online]. Disponible en: http://habitat.aq.upm.es/bpal/onu02/bp014.html.

[3] J. Medina y I. Jiménez, Guía para la gestión integral de los residuos sólidos municipales. Secretaría de Medio Ambiente y Recursos Naturales (México), 2001.

[4] L. Ecamirosa, C. Del Carpio, G. Castañeda y C. Quintal. Manejo de los residuos sólidos domiciliarios: Tuxtla Gutiérrez, Chiapas. Plaza y Valdez SA (México), enero 2001.

[5] E. Betanzo, M. Torres, J. Romero y S. Obregón, “Evaluación de rutas de recolección de residuos sólidos urbanos con apoyo de dispositivos de rastreo satelital: análisis e implicaciones” *Revista Internacional de Contaminación Ambiental*, 2016, vol. 32, no 3, p. 323-337.

[6] Estudio de factibilidad. [Online]. Disponible en: www.bvsde.paho.org/bvsacd/cd61/guaviare/factibilidad.pdf.

[7] Diagnóstico de la situación del manejo de residuos sólidos municipales en América Latina y el Caribe. [Online]. Disponible en: www2.congreso.gob.pe/sicr/.../con4.../DiagnósticoSituaciónManejoResiduosSólidos.pdf.

[8] O. Aloquili, A. Elbanna, and A. Al-Azizi. “Automatic vehicle location tracking system based on GIS environment.” *IET software*, 2009, vol. 3, no 4, p. 255-263.

[9] B. Ghribi, and L. Logrippo. “Understanding GPRS: the GSM packet radio service.” *Computer Networks*, 2000, vol. 34, no 5, p. 763-779.

[10] I. M. Almomani, N. Y. Alkhalil, E. M. Ahmad and R. M. Jodeh, “Ubiquitous GPS vehicle tracking and management system.” *Applied Electrical Engineering and Computing Technologies (AEECT), 2011 IEEE Jordan Conference on*, Amman, 2011, pp. 1-6.

[11] B. Kodavati, V. K. Raju, S. Srinivasa, A. V. Prabu, T. Appa and Y. V. Narayana, “GSM and GPS based vehicle location and tracking system.” *International Journal of Engineering Research and Applications (IJERA)*, 2011, pp. 616-625.

[12] S. Lee, G. Tewolde and J. Kwon, “Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application.” *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, Seoul, 2014, pp. 353-358.

[13] M. Kamel, “Real-Time GPS/GPRS Based Vehicle Tracking System.” *International Journal Of Engineering And Computer Science*, 2015, vol. 4, no 8, p. 648-652.

[14] S. Murugananham and P. R. Mukesh, “Real time web based vehicle tracking using GPS.” *J. World Acad. Sci. Eng. Technol*, 2010, vol. 61, p. 91-99.

[15] A. S. Dinkar and S. A. Shaikh, “Design and implementation of Vehicle tracking system using GPS.” *Journal of Information Engineering and Applications*, 2011, vol. 1, no 3, p. 1-7.

[16] P. K. Harshadbhai, “Design of GPS and GSM based vehicle location and tracking system.” *International Journal of Science and Research (IJSR)*, India Online ISSN, 2013, p. 2319-7064.

[17] Vulnerabilidad a Nivel Municipal del Cantón Loja. [Online]. Disponible en: http://dspace.cedia.org.ec/bitstream/123456789/851/1/Perfil%20territorial%20LOJA.pdf

[18] Arduino Uno. [Online]. Disponible en: http://arduino.cc/en/Main/arduinoBoardUno.

[19] Waveshare Electronics. Raspberry PI. [Online]. Disponible en: http://www.wvshare.com/product/RPi-B-CN.htm.

[20] Sistema de Posicionamiento Global (GPS). [Online]. Disponible en: http://www.inegi.org.mx/geo/contenidos/geodesia/gps.aspx?dv=c1.

[21] Sistemas de Comunicaciones. [Online]. Disponible en: http://es.slideshare.net/edu395090/sistema-de-comunicaciones-moviles.

[22] Geolocation Tracker (GPRS + GPS) with SIM908 over Arduino and Raspberry Pi. [Online]. Disponible en: http://www.cookinghacks.com/documentation/tutorials/geolocation-tracker-gprs-gps-geoposition-sim908-arduino-raspberry-pi.

[23] Shield para Arduino. [Online]. Disponible en: http://www.open-electronics.org/gsm-gps-shield-for-arduino/.

[24] Módulo SIM900 y SIM908. [Online]. Disponible en: http://www.open-electronics.org/gsmgprs-gps-modem-with-sim900sim908-module/.

[25] Todo Electrónica. [Online]. Disponible en: https://www.todoelectronica.com/antena-gsm-conector-ufl-p-14828.html.

[26] Todo Electrónica. [Online]. Disponible en: https://www.todoelectronica.com/antena-gps-conector-ufl-p-14828.html.

[27] Cavadevices. [Online]. Disponible en: http://www.cavadevices.com/archivos/FOLLETOS/calculo%20de%20bateria.pdf.

[28] Bless Power. [Online]. Disponible en: www.blesspower.com/.

[29] F. LÓPEZ (2001), “Integración de tecnologías a través de servidores Web” [Online]. Disponible en: http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node20.html.

[30] NetBeans IDE. [Online]. Disponible en: https://netbeans.org .

[31] Las 10 razones principales para usar MySQL como base de datos integrada. [Online]. Disponible en: http://www.mysql.com/why-mysql/white-papers/las-10-razones-principales-para-usar-mysql-como-base-de-datos-integrada/.

[32] Apache Friends. [Online]. Disponible en: https://www.apachefriends.org.

[33] Garmin International, Inc., “Monterra™ Manual del usuario”, pp 1-20, octubre 2013. [Online]. Disponible en: http://static.garmincdn.com/pumac/Monterra_OM_ES.pdf.

[34] Genbetadev. [Online]. Disponible en: http://www.genbetadev.com/cnet/como-calcular-la-distancia-entre-dos-puntos-geograficos-en-c-formula-de-haversine.

ANEXOS

ANEXO A

Rutas de recolección de RSM en la ciudad de Loja

Personal que labora en Recolectores									
	Personal	**Actividad**	**Sectores de la Ciudad**						**Horario**
1	**Kléver Chamba. – (03-21).**	Chofer	**Sector 1**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**	
1	Máximo Sánchez.	Trabajadores							
1	José Díaz.	Trabajadores	X	x	x	x	x	x	**16H30 a**
1	Marcelo Vivanco.	Trabajadores	X	x	x	x	x	x	**21H30**
1	Manuel Díaz.	Trabajadores	X	x	x	x	x	x	**horas**
1	**Aníbal Castro. - (03-15).**	Chofer	**Sector 2ª**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		
1	Edgar Reyes.	Trabajador							
1	Ángel Jiménez.	Trabajador	X	x	x	x	x		**07H00 a**
1	Branio Chamba	Trabajador	X	x	x	x	x		**14H00**
1	Manuel Cumbicus.	Trabajador	X	x	x	x	x		**Horas**
1	**Pablo Alvarado. – (03-16).**	Chofer	**Sector 3**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07H00 a**
1	Daniel Cabezas.	Trabajador							**14H00**
1	Carlos Paredes.	Trabajador	X	x	x	x	x		**Horas**
1	Ángel Mátalo Macas.	Trabajador	X	x	x	x	x		
1	Jorge Montoya.	Trabajador	X	x	x	x	x		
1	**Hermel Quevedo.- (03-14).**	Chofer	**Sector 4**						

			Lunes	Martes	Miércoles	Jueves	Viernes		
1	Mauro San Martin.	Trabajador							**07H00 a 14H00 Horas**
1	Darwin Campos.	Trabajador	X	x	x	x	x		
1	Manuel Otuna.	Trabajador	X	x	x	x	X		
1	Kervin Herrera	Trabajador	X	x	x	x	X		
1	**Pablo Alvarado. - (03-16).**	Chofer	**Sector 5**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**16H30 a 21H30 horas**
1	Juan Sánchez.	Trabajador							
1	Eloy Vallejo.	Trabajador	X	x	x	x	X		
1	Alexander Tocto	Trabajador	X	x	x	x	X		
1	Eddy Romero.	Trabajador	X	x	x	x	X		
1	**Klever Chamba. - (03-21).**	Chofer	**Sector 6**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07H00 a 14H00 Horas**
1	José Paredes.	Trabajador							
1	Mario Torres.	Trabajador	X	x	x	x	X		
1	John Bustamante.	Trabajador	X	x	x	x	X		
1	Jorge Rugel.	Trabajador	X	x	x	x	X		
1	**Wilson Poma (alterno03-08)**	Chofer	**Sector 7**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**16H30 a 21H30 horas**
1	Stalin Tapia.	Trabajador							
1	Fabián Puchaicela.	Trabajador	X	x	x	x	X		
1	José Obaco.	Trabajador	X	x	x	x	X		
1	Wilson Romero.	Trabajador	X	x	x	x	X		
1	**Benito Sisalima. - (03-12).**	Chofer	**Sector 2B (8)**						

			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07H00 a 14H00 Horas**
1	Edgar Muicela.	Trabajador							
1	Luis León.	Trabajador	X	x	x	x	X		
1	Luis Puchaicela.	Trabajador	X	x	x	x	X		
1	Ángel Matailo Medina.	Trabajador	X	x	x	x	X		
1	**Wilson Poma. - (03-08).**	Chofer.	**Sector 9**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07 H00 a 14 Horas**
1	Leonardo Gualán.	Trabajador							
1	Fernando Sanmartín.	Trabajador	X	x	x	x	x		
1	Eduardo López.	Trabajador	X	x	x	x	x		
1	Franklin Proaño.	Trabajador	X	x	x	x	x		
1	**Wilder Sánchez. - (03-15).**	Chofer.	**Sector 10**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07 H00 a 14 H00 Horas**
1	Vinicio Chamba.	Trabajador							
1	Marco Cango.	Trabajador	X	x	x	x	x		
1	Guido Morocho.	Trabajador	X	x	x	x	x		
1	Ángel Sarmiento	Trabajador	X	x	x	x	x		
1	**Geovanny Enríquez (03-07).**	Chofer.	**Sector 11**						
			Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**		**07 a 14 Horas**
1	Euclides Sanmartín.	Trabajador							
1	Lorgio Morocho.	Trabajador	X	x	x	x	x		

1	Segundo Valencia.	Trabajador	X	x	x	x	x		
1	Diógenes Toledo.	Trabajador	X	x	x	x	x		
44	**Subtotal trabajadores recolectores.**								
55	**TOTAL incluidos los choferes.**								

Fuente: Municipio de Loja

ANEXO B

Desarrollo del script para el funcionamiento de la unidad central de procesamiento del prototipo.

arduino.ino

```
//Librerías para el desarrollo del sistema
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "inetGSM.h"
#include "gps.h"
//Inicialización de la librería para él envió de comandos AT
InetGSM inet;
GPSGSM gps;
//Declaración de variables
char lon[15];
char lat[15];
char alt[15];
char time[20];
char vel[15];
char msg1[5];
char msg2[5];
char stat;
char msg[50];
int numdata;
char inSerial[50];
int i=0;
boolean started=false;
void setup()
{
    //Conexión serial.
    Serial.begin(9600); //Configuración de la velocidad de transmisión del escudo
    Serial.println("GSM Shield testing.");
    if (gsm.begin(2400)) {
        Serial.println("\nstatus=READY");
        started=true;
    } else Serial.println("\nstatus=IDLE");
    //Configuración de la red GPRS
    if(started) {
            if (inet.attachGPRS("internet.claro.com.ec", "", "")) //Establecer mi proveedor GPRS
           Serial.println("status=ATTACHED");
        else Serial.println("status=ERROR");
        delay(1000);

//Lee la dirección IP
        gsm.SimpleWriteln("AT+CIFSR");
        delay(5000);
        gsm.WhileSimpleRead(); //Lee incluso cuando el serial buffer este vacío
        //Obtener el cliente TCP, envía una solicitud al servidor y guarda respuesta
        numdata=inet.httpGET("200.0.29.28", 5080, "/", msg, 50,lon,lat,alt,time,vel);
        //Imprimir resultados
        Serial.println("\nNumber of data received:");
        Serial.println(numdata);
        Serial.println("\nData received:");
        Serial.println(msg);
    }

    //Configuración de GPS
```

```
    if(started) {
        //Adjuntar GPS
        if (gps.attachGPS())
            Serial.println("status=GPSREADY");
        else Serial.println("status=ERROR");
        delay(20000); //Tiempo para arreglar
        stat=gps.getStat();
        if(stat==1)
            Serial.println("NOT FIXED");
        else if(stat==0)
            Serial.println("GPS OFF");
        else if(stat==2)
            Serial.println("2D FIXED");
        else if(stat==3)
            Serial.println("3D FIXED");
        delay(5000);
        //Obtener datos para el GPS
        gps.getPar(lon,lat,alt,time,vel);
        Serial.println(lon);   //Longitud
        Serial.println(lat);   //Latitud
        Serial.println(alt);   //Altitud
        Serial.println(time);  //Tiempo
        Serial.println(vel);   //Velocidad
    }
};
void loop()
{
   //Envío de datos al servidor
   gps.getPar(lon,lat,alt,time,vel);
   if (inet.attachGPRS("internet.claro.com.ec", "", "")) //Uso red GPRS ya establecida
            Serial.println("status=ATTACHED");
        else Serial.println("status=ERROR");
        delay(1000);
        gsm.WhileSimpleRead();
        numdata=inet.httpGET("200.0.29.28", 5080, "/", msg, 50,lon,lat,alt,time,vel);
        delay (1000);
};
void serialhwread()
{
    //Conexión arduino con escudo GPRS
    i=0;
    if (Serial.available() > 0) {
        while (Serial.available() > 0) {
            inSerial[i]=(Serial.read());
            delay(10);
            i++;
        }
        inSerial[i]='\0';
        if(!strcmp(inSerial,"/END")) {
            Serial.println("_");
            inSerial[0]=0x1a;
            inSerial[1]='\0';
            gsm.SimpleWriteln(inSerial);
        }
        //Envío y guardado de comandos AT mediante el puerto serial

        if(!strcmp(inSerial,"TEST")) {
            Serial.println("SIGNAL QUALITY");
            gsm.SimpleWriteln("AT+CSQ");
        }
```

```
        //Lee el último mensaje guardado

        if(!strcmp(inSerial,"MSG")) {
            Serial.println(msg);
        } else {
            Serial.println(inSerial);
            gsm.SimpleWriteln(inSerial);
        }
        inSerial[0]='\0';
    }
}
void serialswread()
{
     serialhwread();
}
void serialhwreadGPS()
{

    //Conexión arduino con escudo GPS

    i=0;
    if (Serial.available() > 0) {
        while (Serial.available() > 0) {
            inSerial[i]=(Serial.read());
            delay(10);
            i++;
        }
        inSerial[i]='\0';
        if(!strcmp(inSerial,"/END")) {
            Serial.println("_");
            inSerial[0]=0x1a;
            inSerial[1]='\0';
            gsm.SimpleWriteln(inSerial);
        }

        //Envío y guardado de comandos AT mediante el puerto serial

        if(!strcmp(inSerial,"TEST")) {
            stat=gps.getStat();
            if(stat==1)
                Serial.println("NOT FIXED");
            else if(stat==0)
                Serial.println("GPS OFF");
            else if(stat==2)
                Serial.println("2D FIXED");
            else if(stat==3)
                Serial.println("3D FIXED");
        }

        //Lee el último mensaje guardado

        if(!strcmp(inSerial,"MSG")) {
            Serial.println(msg1);
        } else {
            Serial.println(inSerial);
            gsm.SimpleWriteln(inSerial);
        }
        inSerial[0]='\0';
```

```
    }
}
void serialswreadGPS()
{
    gsm.SimpleRead();
}
```

ANEXO C

Scripts para el funcionamiento del módulo conexión

Servidor.java

```
//Adquisición de datos del modulo

package com.tesis.rastreo; //Nombre del Paquete
//Importar clases
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import jdk.nashorn.internal.runtime.regexp.joni.constants.AsmConstants;

public class Servidor extends Thread {//Permiso a la clase y herencia

    ServerSocket serversocket = null;
    Socket socket = null;
    ConexionBaseDatos cbd;
            SimpleDateFormat formatoHora = new SimpleDateFormat("yyyyMMddHHmmss");

    @Override
    public void run() {//Inicio de conexión con Base de Datos
        cbd = new ConexionBaseDatos();
        cbd.openConexion();
        startPort();
    }

    public void startPort() {//Arranque del puerto por el que se va a recibir los datos
        //String trama = "-7911.903477,-359.285823,2144.450684,20141024193544.000,0.000000";
        while (true) {
            try {
                serversocket = new ServerSocket(Integer.parseInt(Interfaz.txtPuerto.getText()));
                socket = serversocket.accept();

                InputStream is = socket.getInputStream();
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);
                String string = null;
                //Dar formato a los datos que ingresan
                while ((string = br.readLine()) != null) {
                    System.out.println(string);
                    if (!string.equals(",,,,")) {
                        String[] partes = string.split(",");
```

```
                    Calendar c = new GregorianCalendar();
                    String fecha = "";

                    if (partes[3].contains(".")) {
                        fecha = partes[3].replace(".", ",");
                    }
                    //Compensación de uso horario -5
                    c.setTime(formatoHora.parse(fecha.split(",")[0]));
                    c.add(Calendar.HOUR, -5);
                    //Almacenamiento de datos
                    cbd.insertRecorrido(new Recorrido(c.getTime(),
                            convertLatLon(partes[1]),
                            convertLatLon(partes[0]),
                            Double.parseDouble(partes[4])));
                }
            }
            socket.close();
            serversocket.close();
        //Advertencias en recepción de datos
        } catch (ParseException e) {
            System.out.println("Problemas al convertir fecha [" + e.getMessage() + "]");
        } catch (IOException ex) {
            System.out.println("Problemas de Puerto [" + ex.getMessage() + "]");
        } catch (Exception ex) {
            System.out.println("Excepcion Desconocida [" + ex.getMessage() + "]");
        }
    }
}

    //Convertir trama NMEA a grados, minutos y segundos
    public double convertLatLon(String latLon) {
        Double primero = Double.parseDouble(latLon.substring(0, latLon.lastIndexOf(".") - 2));
        Double segundo = Double.parseDouble(latLon.substring(latLon.lastIndexOf(".") - 2,
latLon.length())) / 60;
        return primero - segundo;
    }
}
```

Recorrido.java

```
//Almacenamiento de datos recibidos del modulo
package com.tesis.rastreo;//Nombre del Paquete
//Importar clases
import java.util.Date;

public class Recorrido {
    private Date fechaHoraEquipo;
    private double latitud;
    private double longitud;
    private double velocidad;

    public Recorrido(Date fechaHoraEquipo, double latitud, double longitud, double velocidad) {
        this.fechaHoraEquipo = fechaHoraEquipo;
        this.latitud = latitud;
        this.longitud = longitud;
        this.velocidad = velocidad;
    }

    //Regreso de hora y fecha del módulo
```

```
    public Date getFechaHoraEquipo() {
        return fechaHoraEquipo;
    }

    //Establecer hora y fecha del módulo
    public void setFechaHoraEquipo(Date fechaHoraEquipo) {
        this.fechaHoraEquipo = fechaHoraEquipo;
    }

    //Regreso de latitud
    public double getLatitud() {
        return latitud;
    }

    //Establecer latitud
    public void setLatitud(double latitud) {
        this.latitud = latitud;
    }

    //Regreso de longitud
    public double getLongitud() {
        return longitud;
    }

    //Establecer longitud
    public void setLongitud(double longitud) {
        this.longitud = longitud;
    }

    //Regreso de velocidad
    public double getVelocidad() {
        return velocidad;
    }

    //Establecer velocidad
    public void setVelocidad(double velocidad) {
        this.velocidad = velocidad;
    }
}
```

ConexionBaseDatos.java

```
//Conexión con la Base de Datos
package com.tesis.rastreo; //Nombre del Paquete
//Importar clases
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import javax.swing.JOptionPane;
//Conexión Base de Datos
public class ConexionBaseDatos {
    //Declaración de variables privadas
    private Connection conexion;
```

```
    private String driver, url, ip, usr, pass, server;
    private String bd;
    private Statement s;
    private ResultSet rs;
    //Datos de Acceso a la Base de Datos
    public ConexionBaseDatos() {
        this.ip = "localhost";
        this.bd = "barduinodb";
        this.usr = Interfaz.txtUsuario.getText();
        this.pass = Interfaz.txtClave.getText();
        this.driver = "com.mysql.jdbc.Driver";
        this.url = "jdbc:mysql://" + ip + "/" + bd;
    }
    //Abrir y cerrar conexión a Base de Datos
    public void openConexion() {
        try {
            Class.forName(driver);
            conexion = DriverManager.getConnection(url, usr, pass);
            s = conexion.createStatement();
        } catch (ClassNotFoundException | SQLException ex) {
            JOptionPane.showMessageDialog(null, "No se Puede Conectar a la Base de Datos:\n[" +
ex.getMessage() + "]");
            System.exit(1);
        }
    }

    public void closeConexion() {
        if (conexion != null) {
            try {
                conexion.close();
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(null, "No se Puede Cerrar la Conexion con la Base de
Datos:\n[" + ex.getMessage() + "]");
            }
        }
    }
    //Almacenamiento de tramas en la Base de Datos
    public void insertRecorrido(Recorrido r) {
        try {
            SimpleDateFormat formatoFechaHora = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            String sql = "INSERT INTO recorridos (fecha_hora_equipo, latitud, longitud, velocidad) "
                    + "VALUES ('" + formatoFechaHora.format(r.getFechaHoraEquipo()) + "'," + r.getLatitud()
+ "," + r.getLongitud() + "," + r.getVelocidad() + ")";
            s.executeUpdate(sql);
        } catch (SQLException ex) {
            System.err.println("Problemas al guardar la informacion [" + ex.getMessage() + "]");
        }
    }
}
```

Interfaz.java

```
package com.tesis.rastreo;

import java.awt.Toolkit;


public class Interfaz extends javax.swing.JFrame {
    Servidor s;
```

```
    /**
     * Creates new form Interfaz
     */
    public Interfaz() {
        initComponents();
        s = new Servidor();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        btnIniciar = new javax.swing.JButton();
        btnDetener = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        txtUsuario = new javax.swing.JTextField();
        txtClave = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        txtPuerto = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Servidor Arduino");
        setIconImage(new
javax.swing.ImageIcon(getClass().getResource("/com/tesis/rastreo/logo_arduino.png")).getImage());

        btnIniciar.setText("Iniciar");
        btnIniciar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIniciarActionPerformed(evt);
            }
        });

        btnDetener.setText("Detener");
        btnDetener.setEnabled(false);
        btnDetener.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDetenerActionPerformed(evt);
            }
        });

        jLabel1.setText("Usuario:");

        txtUsuario.setText("root");
        txtUsuario.setToolTipText("Ingrese Usuario de la Base de Datos");

        txtClave.setText("root");
        txtClave.setToolTipText("Ingrese Contraseña de la Base de Datos");

        jLabel2.setText("Contraseña:");

        jLabel3.setText("Puerto:");

        txtPuerto.setText("5080");
        txtPuerto.setToolTipText("Ingrese puerto de recepcion de Datos");
```

```
        txtPuerto.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                txtPuertoKeyTyped(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel1)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtUsuario, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(26, 26, 26)
                        .addComponent(btnIniciar)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 78,
Short.MAX_VALUE)
                        .addComponent(btnDetener))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel2)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtClave, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel3)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtPuerto, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(20, 20, 20))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(txtUsuario, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(txtClave, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(txtPuerto, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 22,
Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(btnIniciar)
                .addComponent(btnDetener))
            .addGap(24, 24, 24))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnIniciarActionPerformed
        s.start();
        txtUsuario.setEnabled(false);
        txtClave.setEnabled(false);
        txtPuerto.setEnabled(false);
        btnIniciar.setEnabled(false);
        btnDetener.setEnabled(true);
    }//GEN-LAST:event_btnIniciarActionPerformed

    private void btnDetenerActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnDetenerActionPerformed
        System.exit(0);
    }//GEN-LAST:event_btnDetenerActionPerformed

    private void txtPuertoKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_txtPuertoKeyTyped
        if (!Character.isDigit(evt.getKeyChar()) && !Character.isISOControl(evt.getKeyChar())) {
            Toolkit.getDefaultToolkit().beep();
            evt.consume();
        }
    }//GEN-LAST:event_txtPuertoKeyTyped

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
```

```
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnDetener;
    private javax.swing.JButton btnIniciar;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    public static javax.swing.JTextField txtClave;
    public static javax.swing.JTextField txtPuerto;
    public static javax.swing.JTextField txtUsuario;
    // End of variables declaration//GEN-END:variables
}
```

ANEXO D

Script de la base de datos en MySQL

Barduino.sql

```
CREATE DATABASE  IF NOT EXISTS `barduinodb` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `barduinodb`;
-- MySQL dump 10.13  Distrib 5.6.17, for Win64 (x86_64)
--
-- Host: 127.0.0.1    Database: barduinodb
-- ------------------------------------------------------
-- Server version   5.6.16

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `recorridos`
--

DROP TABLE IF EXISTS `recorridos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `recorridos` (
  `id_recorrido` int(11) NOT NULL AUTO_INCREMENT,
  `fecha_hora_equipo` datetime NOT NULL,
  `latitud` double NOT NULL,
  `longitud` double NOT NULL,
  `velocidad` double NOT NULL,
  PRIMARY KEY (`id_recorrido`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `recorridos`
--

LOCK TABLES `recorridos` WRITE;
/*!40000 ALTER TABLE `recorridos` DISABLE KEYS */;
INSERT INTO `recorridos` VALUES (1,'2014-09-11 18:00:00',-3.9839764353205,-
79.198277147255,43),(2,'2014-09-11 18:30:00',-3.9859764353205,-79.191277147255,30);
/*!40000 ALTER TABLE `recorridos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping events for database 'barduinodb'
--
```

```
--
-- Dumping routines for database 'barduinodb'
--
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2014-10-16 17:37:07
```

ANEXO E

Scripts para el funcionamiento del servidor web

index.php

```
package com.tesis.rastreo;

import java.awt.Toolkit;


public class Interfaz extends javax.swing.JFrame {
    Servidor s;

    /**
     * Creates new form Interfaz
     */
    public Interfaz() {
        initComponents();
        s = new Servidor();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        btnIniciar = new javax.swing.JButton();
        btnDetener = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        txtUsuario = new javax.swing.JTextField();
        txtClave = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        txtPuerto = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Servidor Arduino");
        setIconImage(new
javax.swing.ImageIcon(getClass().getResource("/com/tesis/rastreo/logo_arduino.png")).getImage());

        btnIniciar.setText("Iniciar");
        btnIniciar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIniciarActionPerformed(evt);
            }
        });

        btnDetener.setText("Detener");
        btnDetener.setEnabled(false);
        btnDetener.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDetenerActionPerformed(evt);
            }
        });
```

```
        jLabel1.setText("Usuario:");

        txtUsuario.setText("root");
        txtUsuario.setToolTipText("Ingrese Usuario de la Base de Datos");

        txtClave.setText("root");
        txtClave.setToolTipText("Ingrese Contraseña de la Base de Datos");

        jLabel2.setText("Contraseña:");

        jLabel3.setText("Puerto:");

        txtPuerto.setText("5080");
        txtPuerto.setToolTipText("Ingrese puerto de recepcion de Datos");
        txtPuerto.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                txtPuertoKeyTyped(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel1)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtUsuario, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(26, 26, 26)
                        .addComponent(btnIniciar)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 78,
Short.MAX_VALUE)
                        .addComponent(btnDetener))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel2)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtClave, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(jLabel3)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(txtPuerto, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(20, 20, 20))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(txtUsuario, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(txtClave, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(txtPuerto, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 22,
Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(btnIniciar)
                    .addComponent(btnDetener))
                .addGap(24, 24, 24))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnIniciarActionPerformed
        s.start();
        txtUsuario.setEnabled(false);
        txtClave.setEnabled(false);
        txtPuerto.setEnabled(false);
        btnIniciar.setEnabled(false);
        btnDetener.setEnabled(true);
    }//GEN-LAST:event_btnIniciarActionPerformed

    private void btnDetenerActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnDetenerActionPerformed
        System.exit(0);
    }//GEN-LAST:event_btnDetenerActionPerformed

    private void txtPuertoKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_txtPuertoKeyTyped
        if (!Character.isDigit(evt.getKeyChar()) && !Character.isISOControl(evt.getKeyChar())) {
            Toolkit.getDefaultToolkit().beep();
            evt.consume();
        }
    }//GEN-LAST:event_txtPuertoKeyTyped

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
```

```
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnDetener;
    private javax.swing.JButton btnIniciar;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    public static javax.swing.JTextField txtClave;
    public static javax.swing.JTextField txtPuerto;
    public static javax.swing.JTextField txtUsuario;
    // End of variables declaration//GEN-END:variables
}
```

posición.php

```
<?php
//Dibujar Posición
//Conexión Base de Datos
//Host - Usuario de BD - Contraseña de BD - Nombre de BD
$con = mysqli_connect('localhost', 'root', 'soloyo', 'barduinodb');
if (!$con) {
    die('Could not connect: ' . mysqli_error($con));
}
//Selección de datos actual para la posición
$sql = "SELECT * FROM recorridos WHERE DATE(fecha_hora_equipo) = DATE(NOW()) ORDER BY
fecha_hora_equipo DESC";
$result = mysqli_query($con, $sql);
//Adquisión de datos y trazo de posición
```

```
$json = "data: [";
while ($row = mysqli_fetch_array($result)) {
    $json .= "{fechaHora: '" . $row['fecha_hora_equipo'] . "', "
          . "latitud: " . $row['latitud'] . ", "
          . "longitud: " . $row['longitud'] . ", "
          . 'velocidad: ' . $row['velocidad'] . "},";
}
$json .= "]";
echo "{success: true, $json}";
//Cierre de Base de Datos
mysqli_close($con);
```

recorrido.php

```
<?php

extract($_POST);
//Generar Recorrido
//Conexión Base de Datos
//Host - Usuario de BD - Contraseña de BD - Nombre de BD
$con = mysqli_connect('localhost', 'root', 'soloyo', 'barduinodb');
if (!$con) {
    die('Could not connect: ' . mysqli_error($con));
}
//Selección de inicio y fin del recorrido
$sql = "SELECT * FROM recorridos WHERE fecha_hora_equipo "
       . "BETWEEN CONCAT('$fechaIni', ' ', '$horaIni') AND CONCAT('$fechaFin', ' ', '$horaFin')";
$result = mysqli_query($con, $sql);

$json = "data: [";
if ($result->num_rows > 0) {
    $velMax = 0;
    $velSum = 0;
    //Grafico del Recorrido
    while ($row = mysqli_fetch_array($result)) {
        $json .= "{idRecorrido: " . $row['id_recorrido'] . ", "
              . "fechaHora: '" . $row['fecha_hora_equipo'] . "', "
              . "latitud: " . $row['latitud'] . ", "
              . "longitud: " . $row['longitud'] . ", "
              . 'velocidad: ' . $row['velocidad'] . "},";
        //Promedio de velocidad y velocidad maxima del recorrido
        $velSum += $row['velocidad'];
        if ($row['velocidad'] > $velMax) {
            $velMax = $row['velocidad'];
        }
    }
    $valProm = $velSum / $result->num_rows;
    $json .= "]";
    //Presentación de velocidad maxima y velocidad promedio
    echo "{success: true, $json, velMax: $velMax, velProm: $valProm}";
} else {
    echo "{failure: true}";
}

mysqli_close($con);
```

Sistema de localización automática aplicado a la flota de vehículos de recolección y transporte de Residuos Sólidos Municipales del GADML

Carlos Calderon[1], Nilder Calderón[2]

[1] *Docente Investigador de la SET, Universidad Técnica Particular de Loja*
[2] *Profesional en formación Titulación de IET, Universidad Técnica Particular de Loja*
Loja, Ecuador 2016

[1]cacalderon@utpl.edu.ec, [2]nacalderon@utpl.edu.ec

***Resumen*— En base a problemas comunes como incumplimiento de horarios y de cobertura de rutas se evidencia la necesidad de mejorar el sistema de recolección y transporte de residuos sólidos municipales (RSM) que posee el Gobierno Autónomo Descentralizado Municipal de Loja (GADML), el primer paso para el mejoramiento del sistema es el monitoreo y cuantificación de los parámetros correspondientes a recorridos y tiempos de los vehículos recolectores con la finalidad de evaluar el comportamiento actual y proponer mejoras al sistema. En base a lo anterior surge la iniciativa del presente proyecto cuyo objetivo es desarrollar un sistema de localización de bajo costo y abierto aplicado al monitoreo del recorrido de los vehículos recolectores de RSM. El sistema diseñado e implementado está formado por tres módulos: de adquisición y envío de datos, de procesamiento, y, módulo servidor de aplicaciones. Para la evaluación del prototipo AVL (localización automática de vehículos) implementado, se lo instaló en un vehículo recolector del GADML para monitorear el recorrido de una ruta de recolección de RSM, los parámetros adquiridos por el AVL son: ubicación (latitud, longitud), velocidad, hora y fecha, luego estos datos son transmitidos al módulo servidor para su almacenamiento (DB MySQL) y visualización en tiempo real (Java + Google Maps). Los resultados de las pruebas de campo correspondientes a dos recorridos de 15.6 km y 12.7 km en la zona urbana de la ciudad de Loja, obtuvieron un error promedio de posición de 3.78 m y un error máximo de 20.27 m.**

***Palabras clave*—Localización vehicular automatizada (AVL), Sistema de posicionamiento global (GPS), red GPRS, Residuos Sólidos Municipales (RSM).**

I. INTRODUCCIÓN

La gestión de los residuos sólidos urbanos constituye uno de los principales problemas de los Gobiernos autónomos descentralizados (GADs). La mala gestión causa la presencia de residuos abandonados, lo cual origina una serie de problemas directos en las ciudades, por ejemplo: producen olores molestos, favorecen la presencia de roedores e insectos los cuales son vectores de enfermedades, y, producen sensación de abandono y suciedad del paisaje urbano [1].

En el Ecuador, solo unas pocas municipalidades han elaborado una hoja de ruta de la gestión integral de residuos sólidos, sin embargo, es aún menor el número de ciudades que han implementado algún tipo de sistema o proceso exitoso encaminado a la gestión de residuos sólidos, estas son: Quito, Cuenca, Guayaquil, Manta, Cayambe, Ibarra, Loja, Otavalo, entre otras [1].

En la década de los 90, la municipalidad de Loja creó el programa de gestión de residuos sólidos, tomando como referencia el Plan de Acción Loja Siglo XXI. En la ciudad de Loja, el modelo de gestión de residuos es el Manejo Municipal directo el cual es un modelo utilizado especialmente en núcleos poblacionales medianos y pequeños. Entre todas las aristas que comprende el proyecto, está el componente de recolección de residuos que a su vez está conformada por: clasificación domiciliaria de los residuos, planificación de las rutas de recolección, y, recolección domiciliaria y de los residuos al relleno sanitario [2].

El componente de recolección de residuos sólidos urbanos o municipales (RSM) es un factor importante en todo el proceso y además conlleva costos que representan entre el 70% y 85% del costo total del manejo de RSM. El principal objetivo del sistema de recolección es la preservación de la salud, por lo cual es necesario que el sistema cubra a toda la población de manera ordenada, higiénica y en forma adecuada [3], [4].

Mejorar un sistema de recolección de RSM requiere de datos tales como: cantidad de residuos sólidos generados, población que debe servirse, distancias recorridas en las rutas de recolección, análisis de tiempos de recolección y transporte, características de las unidades recolectoras, personal de recolección, estructura vial de la ciudad, entre otros. Las consecuencias de un deficiente sistema de recolección son: operación ineficiente de la infraestructura, ineficiencia del personal, reducción de las coberturas del servicio de recolección y proliferación de tiraderos clandestinos [3], [4], [5].

El análisis anterior acerca de la importancia del sistema de recolección y transporte de RSM sumado a los problemas actuales identificados por los habitantes de la ciudad de Loja, tales como el incumplimiento de horarios y el incumplimiento de cobertura de rutas de recolección, ponen en evidencia la necesidad de mejorar el sistema de recolección de RSM que posee el Gobierno Autónomo Descentralizado Municipal de Loja (GADML). La primera etapa para mejorar el sistema en mención es el monitoreo y cuantificación de los parámetros correspondientes a recorridos y tiempos de los vehículos recolectores con el objetivo de evaluar el comportamiento actual y proponer mejoras al sistema.

Este monitoreo de parámetros se lo puede efectuar con la ayuda de dispositivos basados en geo-posicionamiento sin embargo el sistema propuesto debe cumplir características de bajo costo y abierto, estas características cooperan a la disminución del impacto en las partidas presupuestarias del GADML y permiten la integración de los datos recolectados a otras plataformas de análisis y procesamiento.

Por otro lado, los denominados sistemas de localización automática de vehículos (AVL-Automatic vehicle location) permiten a las organizaciones rastrear y coordinar los movimientos de su flotas vehiculares. La tecnología de los sistemas de rastreo fue posible por la integración de tres nuevas tecnologías: tecnologías de navegación como el

sistema de posicionamiento global (GPS- global positioning system), tecnologías de bases de datos como el Sistema de información geográfica (GIS- geographic information system) y las tecnologías de comunicación como las redes GPRS (general packet radio service) [6], [7].

En el entorno académico se han propuesto diferentes sistemas hardware y software aplicados al rastreo vehicular. En [6] se desarrolla un sistema software que visualiza y registra en tiempo real la posición, la velocidad de avance y el nivel de combustible de un vehículo determinado, estos datos permiten la supervisión del estado de la flota vehicular y la gestión de los recorridos. En [8] se presenta un sistema que proporciona servicio de rastreo mediante una aplicación móvil desarrollada en Java y una aplicación web; sin embargo para la implementación de los sistemas mencionados se utiliza un dispositivo AVL de mercado.

En [9] se integra un sistema AVL basado en los microcontroladores Intel y Atmel, un módem GSM y un dispositivo GPS. En [10] se desarrolla un sistema de rastreo vehicular en tiempo real basado en el dispositivo Arduino Uno, el módulo GPRS SM5100B y el módulo GPS EM406. Estos sistemas fueron desarrollados bajo el concepto de bajo costo, fueron evaluados experimentalmente sin embargo no se cuantificaron los errores y su sensibilidad ante bajos niveles de cobertura de la red GPRS. Sistemas más complejos como los descritos en [11], [12], [13] y [14], presentan el diseño de los componentes hardware y software de un sistema de rastreo vehicular, el servidor central de rastreo almacena en una base de datos la información de posición que puede ser accedida mediante un navegador web, empero no se realiza ni analiza el comportamiento del sistema.

En base al análisis anterior el objetivo principal del presente proyecto es desarrollar un prototipo AVL de bajo costo y abierto aplicado al monitoreo de vehículos de recolección y transporte de RSM pertenecientes al GADML. Para el desarrollo del objetivo se ha seguido una metodología la cual se desglosa en las secciones del presente documento. En la sección II se realiza un análisis de la gestión actual de los RSM en la ciudad de Loja y de los requerimientos necesarios para desarrollar el sistema de monitoreo. En la sección III se realiza el desarrollo e implementación del prototipo del sistema para los vehículos recolectores de RSM de la ciudad de Loja. En la sección IV se realiza la evaluación del prototipo y su análisis de resultados. Finalmente, las conclusiones principales del presente trabajo son resumidas en la Sección V.

II. GESTIÓN DE RESIDUOS SÓLIDOS

El medio ambiente tiene tres funciones económicas fundamentales: como proveedor de factores productivos en forma de materiales o de energía, como fuente de servicios de ocio y bienestar (mejorando la calidad de vida, permitiendo el disfrute de parajes naturales, agua y aire limpios, etc.), y como sumidero de residuos generados por la actividad económica [2]. Debido al crecimiento de la población y en consecuencia al aumento de la contaminación en sus diversas formas, se ha excedido la capacidad de auto-purificación de la naturaleza; esto evidencia, entre otras medidas, la necesidad de la intervención humana para llevar a cabo procesos de gestión de desechos sólidos.

La gestión de residuos se define como el conjunto de operaciones encaminadas a dar a los residuos producidos en una zona determinada el destino más adecuado desde el punto de vista económico y ambiental, según sus características, volumen, procedencia, posibilidades de recuperación y comercialización, coste de tratamiento y normativa legal [2].

Desde el punto de vista administrativo en América Latina y en el Ecuador los Gobiernos Autónomos Descentralizados Municipales (Municipios) son las instituciones responsables de la gestión de residuos sólidos. Los residuos sólidos pueden clasificarse de acuerdo a su origen (domiciliar, industrial, comercial, institucional, público); a su composición (materia orgánica, vidrio, metal, papel, textiles, plásticos, inerte y otros); o de acuerdo a su peligrosidad (tóxicos, reactivos, corrosivos, radioactivos, inflamables, infecciosos). Los principales tipos de residuos sólidos urbanos son: Residuos sólidos municipales (RSM), Residuos sólidos especiales (RSE) y Residuos peligrosos (RP) [15].

Los residuos sólidos municipales son aquellos provenientes de la generación residencial, comercial, institucional, industrial (pequeña industria y artesanía) y los residuos sólidos resultantes del barrido de calles de un conglomerado urbano y cuya gestión está a cargo de las autoridades municipales.

La cadena de gestión de RSM implica: pre-clasificación, recolección, transporte, almacenamiento, clasificación, tratamiento y disposición final de residuos (ver Fig. 1).

Según la Dirección de Higiene Municipal del GADML, el proceso de recolección inicia en el área residencial o comercial, donde una flota de camiones accede a cada una de las calles para recoger los desechos sólidos. El tipo de recolección utilizado en la ciudad de Loja es en la acera. Se tienen diferentes días de recolección tanto para los desechos orgánicos (lunes, miércoles y viernes) como para los inorgánicos (martes, jueves y sábado). El transporte de desechos sólidos cumple diversas rutas que conectan los diversos barrios y ciudadelas con zonas de almacenamiento. Los residuos pueden ser transportados directamente a los puntos de tratamiento o a plantas de transferencia donde se compactan y se cargan en camiones más grandes y adecuados para el transporte hasta su destino definitivo. La ubicación de la estación de transferencia en gran medida influye en la ruta de recolección. Por lo tanto, la distancia de la estación de transferencia debe ser óptima tanto para la ciudad como para el relleno sanitario. Con respecto a la recolección y transporte llevado a cabo por el GADML en la ciudad de Loja, son 11 rutas semanales para la recolección de RSM, se destina un personal de 55 trabajadores distribuidos en horarios de recolección desde las 07h00 hasta las 21h30.

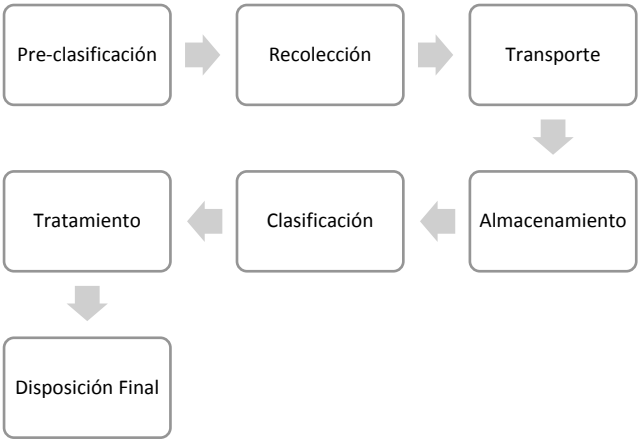

Figura 1. Cadena de suministro de RSM.

Las consecuencias de tener ineficientes rutas de recorrido y carecer de una buena programación, son: altos costos de operación y altos niveles de contaminación del aire y ríos. Por lo tanto, es necesario controlar las variables que intervienen en el desempeño del sistema de recolección y transporte de RSM (RT-RSM), pero antes de poder controlar primero se debe medir dichas variables.

La implementación de un sistema de monitoreo permite conocer las variables: ubicación y velocidad de las unidades de recolección de RSM, distancia y duración del recorrido de las rutas, entre las ventajas de la implementación de este sistema, se citan:

- Verificar el cumplimiento de rutas y horarios por parte de las unidades de recolección de RSM.
- Crear una base de datos, con la información de entrada para la ingeniería de rutas y horarios del sistema de recolección de RSM.
- Verificar las operaciones de la flota de vehículos, mediante parámetros como: tiempos de parada, distancias recorridas, velocidades, entre otros.
- Verificar el estado y ubicación de los contenedores distribuidos en la ciudad.

En base a los costos de los módulos de mercado (valor promedio 600 USD) y debido a que son plataformas cerradas a futuras expansiones se decidió diseñar y desarrollar el dispositivo de rastreo. El presente proyecto asume las actividades referentes al diseño y desarrollo electrónico del dispositivo de rastreo y monitoreo de variables del sistema de RT-RSM, así también con el objetivo de evaluar su compatibilidad con plataformas software aplicadas a visualización y registro de variables, el presente proyecto contempla el desarrollo de rutinas software que permiten la conectividad, almacenamiento y visualización de los datos enviados por el dispositivo de monitoreo.

III. Desarrollo e Implementación del Prototipo del Sistema

En la presente sección se describe los requisitos del sistema, arquitectura del sistema y la implementación del sistema de monitoreo. El sistema debe tener como elemento principal un GPS, tener la facilidad y compatibilidad de acceso a internet a través de un enlace GPRS/GSM, adicional a ello se necesita una tarjeta para el control de los dispositivos, envió y recepción de los datos hacia un servidor en el cual se almacena y presenta la información, con abastecimiento de energía autónomo.

A. Arquitectura del sistema

La metodología aplicada al diseño de la arquitectura del sistema, se describe a continuación:

- Identificación de los requisitos y/o requerimientos del sistema.
- Investigación y justificación de una arquitectura aplicable al sistema requerido.
- Analizar los parámetros de decisión en cada uno de los módulos de la arquitectura planteada.
- Realizar un análisis comparativo de la solución, prestaciones de los equipos ofrecidos por cada empresa y la inversión requerida.

En [16] se sintetiza el principio de operación del sistema y los elementos que lo componen. En base a ello la arquitectura funcional del sistema propuesto para el monitoreo de la flota de vehículos de recolección y transporte de RSM está formado por tres módulos: módulo de procesamiento, módulo de adquisición y envió de datos, y, módulo servidor. La arquitectura del sistema y la interconexión de los bloques se muestra en la Fig. 2.

B. Módulo de procesamiento

El módulo de procesamiento está conformado por la unidad central de procesamiento y la unidad de abastecimiento de energía. La unidad central de procesamiento está basada en el microcontrolador Arduino Uno. El algoritmo principal del microcontrolador realiza la inicialización de librerías, la declaración de variables y la configuración de los parámetros de conexión. La programación de las diferentes funciones se las realiza a través de subrutinas específicas, entre las subrutinas que forman parte del programa se encuentran: rutina de conexión serial, rutina de configuración del modem de comunicación y rutina de conexión al dispositivo GPS. En la Fig. 3 se muestra el diagrama de flujo del algoritmo principal encargado de todo el procesamiento de los datos.

La rutina de conexión serial de datos se realiza a través de USART del Arduino Uno y el Shield SIM908 a una

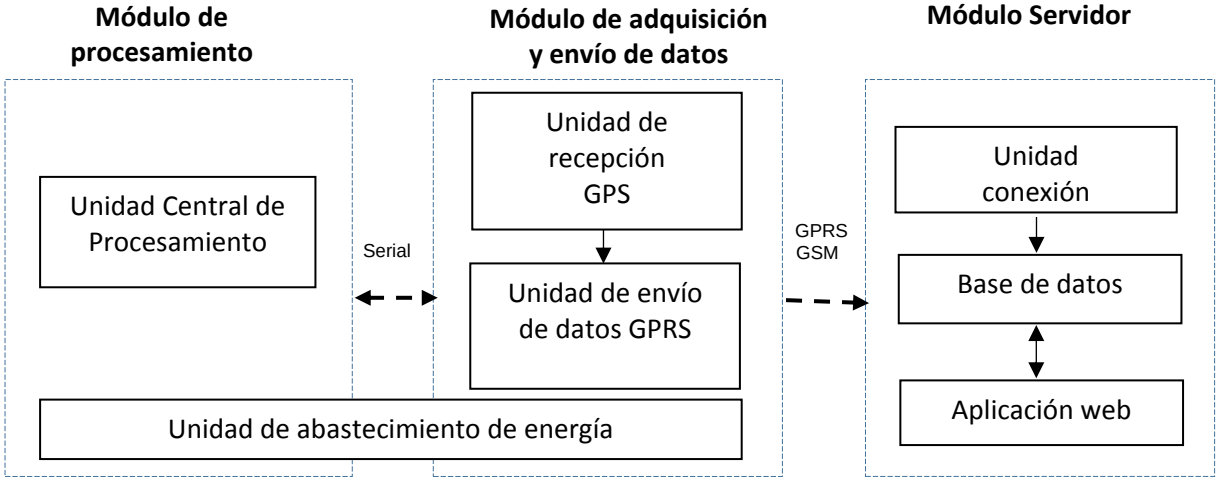

Figura 2. Arquitectura del sistema de monitoreo de la flota de vehículos de RT-RSM.

velocidad de 9600 bps. Los pines de conexión entre el Arduino y el Shield se muestran en la Tabla I.

La rutina de configuración del modem de comunicación se encarga de enviar la trama hacia la red GPRS, en este caso es la red de CLARO (Conecel S. A.), con el objetivo de que pueda ser leída y almacenada en la base de datos por la estación central, mediante comandos AT. La rutina de conexión con el dispositivo GPS adquiere las variables de longitud (GD - Grados decimales), latitud (GD), altitud (msnm), tiempo (s) y velocidad (km/h).

Finalmente, el módulo de abastecimiento de energía consiste en una batería 12V y 7 Ah/día del fabricante Bless Power, con el objetivo de lograr una autonomía energética de 37 horas.

C. Módulo de adquisición y envío de datos

El módulo de adquisición y envío de datos consta del unidad de recepción de datos GPS y unidad de envío de datos a la red GPRS. La unidad de recepción de datos GPS es la encargado de la detección de las variables: longitud, latitud, altitud, tiempo y velocidad, mediante la antena GPS. La unidad de envío de datos transmite la información al servidor por medio de la red GPRS perteneciente a una empresa específica proveedora del servicio. Este módulo se encuentra conectado mediante conexión serial al Arduino Uno en el que se configura el protocolo de dirección IP, el número de puerto del servidor, en este caso específico se utiliza el protocolo TCP/IP para transmisión de los datos, a través de la red GPRS, al servidor de aplicaciones cuya dirección IP pública es 200.0.29.28 y el número de puerto utilizado para la aplicación es 5080. A partir de los módulos descritos se procede a unir el hardware como se muestra en la Fig. 4.

D. Módulo servidor de aplicaciones

Es la conexión hacia un ordenador donde se encuentran montados los servicios para interpretar los datos, almacenarlos en la BD y finalmente presentar los resultados gracias a un servidor web, consta de: unidad conexión, base de datos y aplicación web.

1) Unidad conexión

En la Fig. 5 se muestra el diagrama de flujo de la programación de la aplicación dedicada a la unidad conexión, encargado de: la conexión de la BD, arrancar y escuchar el puerto 5080 por el cual el módulo de adquisición y envió de datos va a transmitir los datos, va a realizar una compensación de la hora del dispositivo que en nuestro caso es -5 UTC, la conversión de tramas NMEA por grados, minutos y segundos y finalmente procesar las tramas adquiridas para el almacenamiento en la BD.

TABLA I. PINES DE CONEXIÓN ENTRE EL ARDUINO UNO Y EL SHIELD.

Arduino Uno		**Shield SIM908**
Pines Digitales	PB1	D9
	PB2	D10
	PB3	D11
	PB4	D12
	PB5	D13

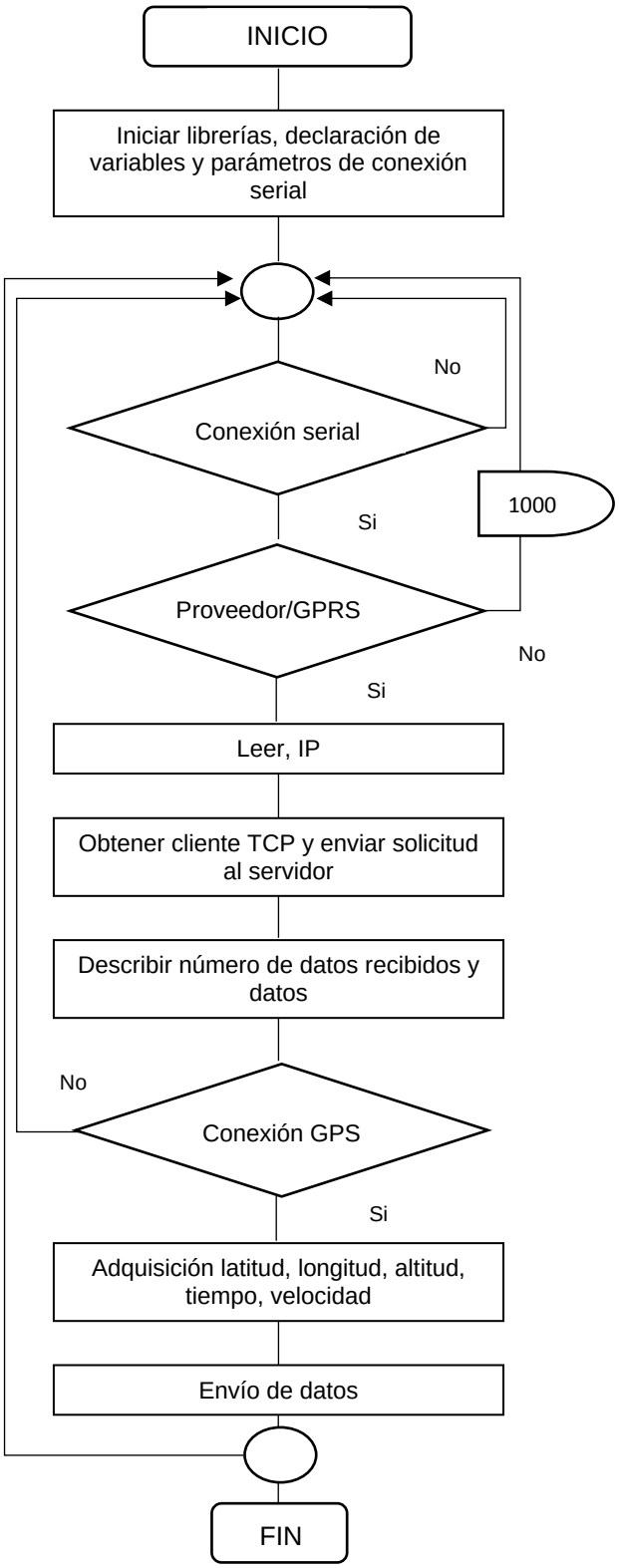

Figura 3. Diagrama de flujo del algoritmo principal.

La unidad conexión es una aplicación desarrollada en NetBeans IDE, en donde se crea un archivo .jar que permite ejecutar aplicaciones escritas en el lenguaje Java, se denomina ServerArduino. En la Fig. 6 se muestra la configuración de la interfaz gráfica de usuario de la aplicación, en la que se debe ingresar: usuario, contraseña y el puerto que se utiliza para la recepción de datos. La principal función de esta aplicación es abrir el puerto 5080 para conectar el módulo de adquisición y envió de datos con el ordenador donde se encuentra montados los servicios, realiza la corrección de uso horario -5 UTC, trasformación y

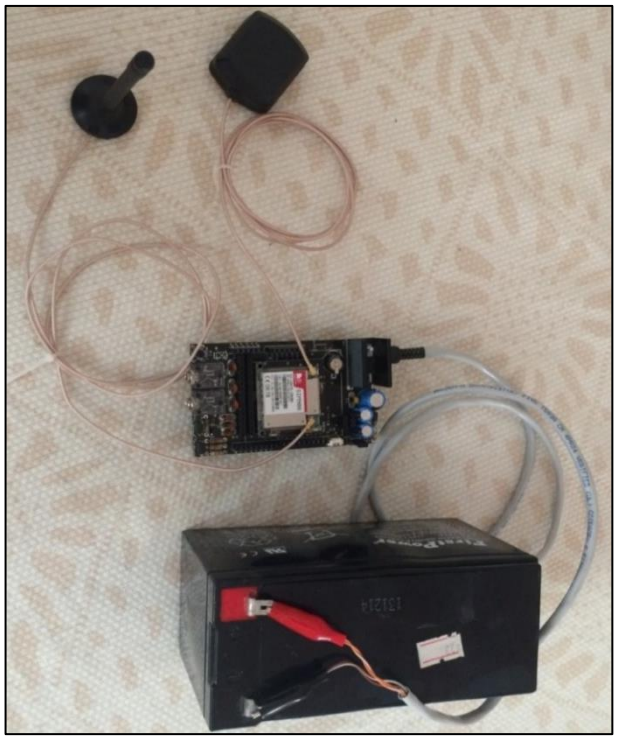

Figura 4. Componentes del prototipo AVL.

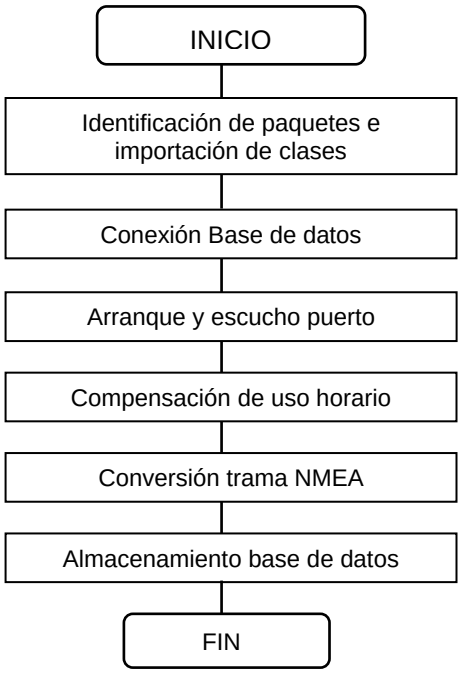

Figura 5. Diagrama de flujo de la programación de la aplicación.

tratamiento de tramas y establece la conexión con la base de datos para realizar el respectivo almacenamiento.

La trama de los datos que envía el módulo de adquisición y envío de datos, lo recepta el módulo de conexión que contiene datos como: mensaje, longitud, latitud, fecha, hora y velocidad (ver Fig. 7).

2) Base de datos

La BD se crea en MySQL Workbench con el nombre barduinodb el cual cuenta con las columnas: id_recorrido, fecha_hora_equipo, latitud, longitud, velocidad (ver Fig. 8).

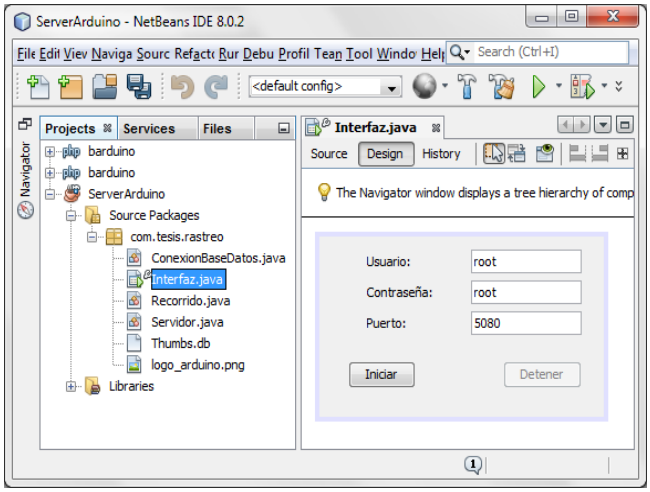

Figura 6. Configuración de la interfaz del módulo conexión.

ATT: OK
RIC:
0, -7911.609078,-359.669979,2194.112793,201603301945512.000, 3,12,0.000000,0.000000
OK

ATT: OK
RIC:
Msg,long,lat,time,vel
OK

Figura 7. Trama de datos de ejemplo (sup.). Estructura de la trama (inf.).

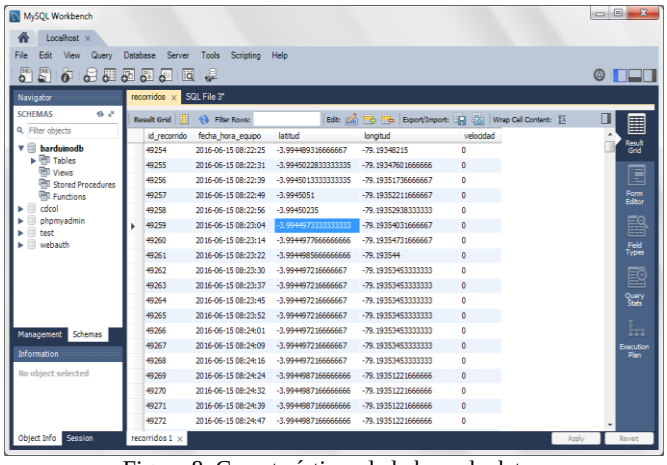

Figura 8. Características de la base de datos.

3) Aplicación web

Se lo desarrolla en lenguaje de programación php con el objetivo de mostrar la ubicación en tiempo real del vehículo recolector y trazar el recorrido (ver Fig. 9a). Para poder ingresar a la aplicación web se debe colocar en el navegador el url: 200.0.29.28:8080/barduino.

Por otro lado, para observar el recorrido del vehículo recolector se lo realiza en un intervalo de tiempo determinado en el cual se debe configurar hora de inicio y finalización de la ruta (ver Fig. 9b).

E. Implementación del sistema

El sistema de monitoreo consta de la parte de hardware y software. El hardware comprende: 1) unidad central de procesamiento, 2) unidad de abastecimiento de energía, y 3) unidad recepción GPS, unidad envió de datos GPRS (ver Fig. 10), y el software desarrollado consta de: 1) unidad conexión, 2) base de datos y 3) aplicación web (ver Fig. 11), cabe recalcar que este costo es únicamente intelectual.

En la Fig. 12 se presenta el diagrama funcional final del Sistema de monitoreo para los vehículos de RT-RSM.

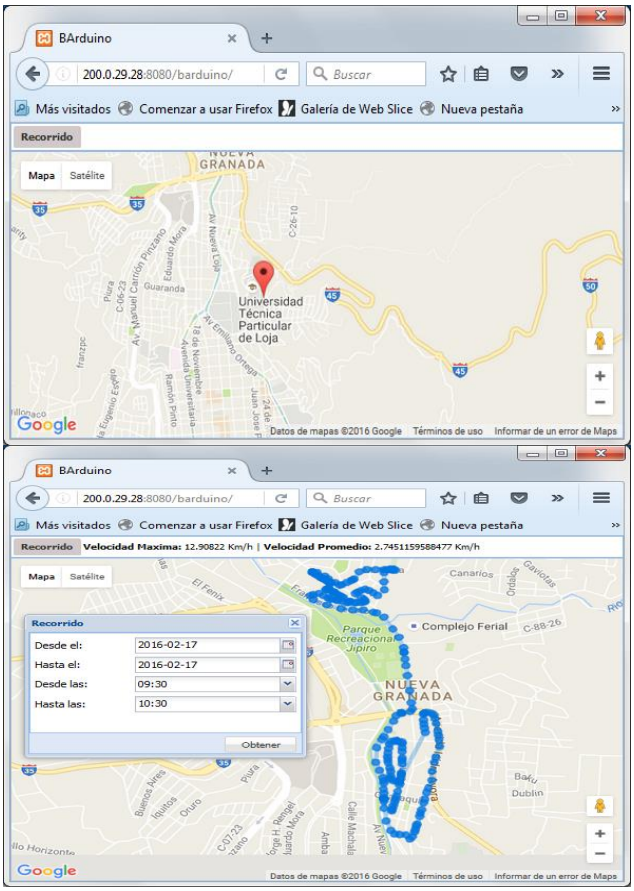

Figura 9. Aplicación web: a) ubicación tiempo real, b) recorrido del vehículo recolector.

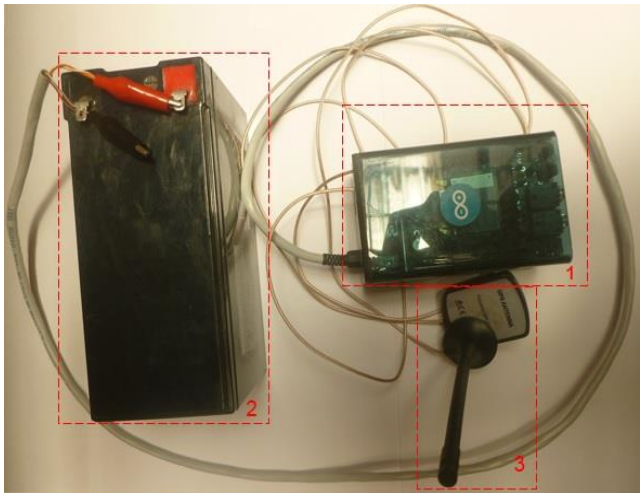

Figura 10. Integración del hardware del prototipo.

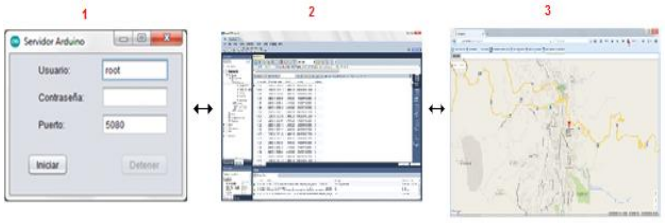

Figura 11. Integración del software para el sistema de monitoreo.

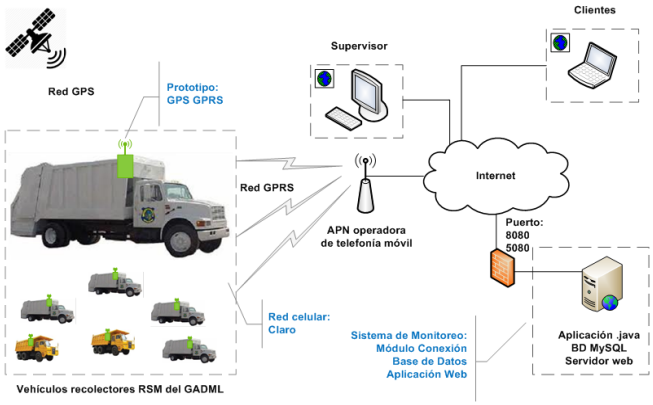

Figura 12. Diagrama funcional del sistema de monitoreo de los vehículos recolectores de RSM.

Los gastos generados para el desarrollo del prototipo se detallan en la Tabla II, El prototipo garantiza el principio de costo vs beneficio lo cual lo hace accesible a cualquier persona o institución. El prototipo fácilmente puede ser adaptado a otro vehículo recolector.

TABLA II. GASTOS GENERADOS PARA EL DESARROLLO DEL PROTOTIPO.

Componente	Precio
Arduino Uno	26.00
Shield	10.00
Chip SIM908	100.00
Antena GPS	12.00
Antena GPRS	7.00
Batería BLESSPOWER	30.00
Carcasa	15.00
Total	**200.00**

IV. EVALUACIÓN DEL PROTOTIPO Y ANÁLISIS DE RESULTADOS

En la presente sección se detalla las pruebas realizadas por el dispositivo de monitoreo de la flota de vehículos de recolección y transporte de RSM de la ciudad de Loja, así también se documenta el análisis de los resultados. Las pruebas realizadas fueron de dos tipos: de laboratorio y de campo. En las pruebas de laboratorio se comprobó la conexión entre los elementos: nodo remoto, cliente y servidor. En las pruebas de campo se realizó dos ensayos: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM, la cual cubre algunos barrios periféricos como: Las Palmas, Atamer, San Cayetano, entre otros.

A. *Pruebas de conexión nodo remoto y servidor*

En esta etapa se realizó pruebas de la conexión nodo remoto-servidor y cliente-servidor, el nodo remoto es el dispositivo de monitoreo del vehículo de RT-RSM, el cliente es el usuario que accede al servidor por medio de una interfaz web, y finalmente el servidor es el equipo central de adquisición y registro de información, para la puesta en marcha del servidor se utilizó una dirección IP pública la cual está previamente parametrizada en el dispositivo de monitoreo, esta dirección tiene habilitados los puertos 5080 y 8080 para la recepción de datos y para la interfaz web, respectivamente.

Con el objetivo de comprobar la conexión entre nodo remoto y servidor, se realizaron pruebas iniciales de conectividad, en la Fig. 13 se visualizan en la interfaz gráfica del servidor, los datos adquiridos por el dispositivo de monitoreo, lo cual evidencia la correcta comunicación entre los elementos del sistema.

B. *Pruebas de campo del sistema*

En las pruebas de campo del sistema se realizó dos ensayos: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja realizado el día 17 de febrero de 2016 desde las 12h20 hasta las 12h55, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM realizado el día 15 de junio de 2016 desde las 9h18 hasta las 11h19.

La instalación del prototipo se lo realizó el día 17 de febrero de 2016 a las 07:00 horas en el inicio del recorrido de la zona céntrica de la ciudad (ver Fig. 14).

Para realizar las pruebas del sistema de monitoreo se debe conectar el prototipo con el servidor, para esto se ejecuta la aplicación realizada en el módulo conexión, como se muestra en la Fig. 15.

El dispositivo de monitoreo envía las coordenadas de ubicación del vehículo recolector de RSM cada 8 segundos, este intervalo de tiempo es el mínimo que soporta el algoritmo del módulo de procesamiento debido a que ejecuta múltiples tareas de comunicación y procesamiento de datos. Una vez realizada la conexión entre los elementos del sistema, se comprueba la recepción de los datos en la base de datos MySQL residente en el servidor (ver Fig. 16).

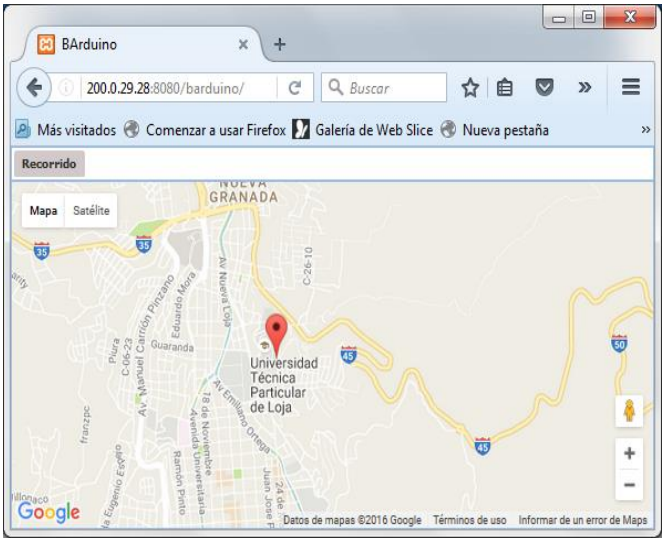

Figura 13. Interfaz gráfica del servidor.

Figura 14. Instalación del prototipo en el vehículo recolector de RSM.

Figura 15. Conexión del prototipo con el servidor.

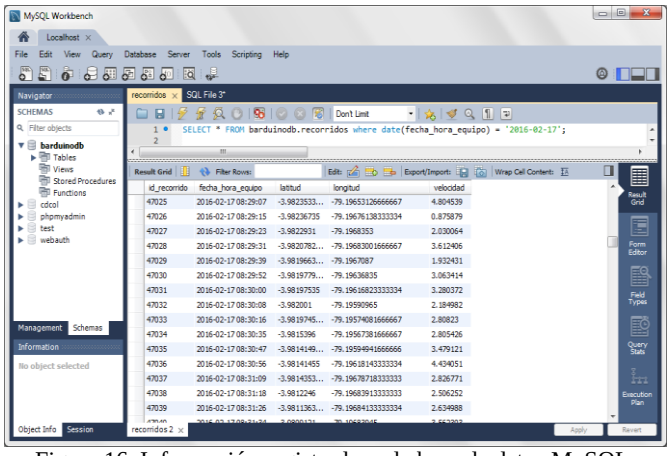

Figura 16. Información registrada en la base de datos MySQL.

Con el objetivo de verificar los errores de magnitud correspondientes a las variables latitud y longitud, se realizó una comparación entre el dispositivo de monitoreo y el GPS comercial Garmin Monterra (precisión de menos 3 metros en campo abierto y detenido, y de menos 15 metros en movimiento con el WAAS and EGNOS habilitados) [17], primeramente se comparó un punto específico y luego se compararon las dos rutas antes mencionadas.

Para verificar el error existente entre el dispositivo de monitoreo y el GPS Garmin Monterra, se registró un punto específico perteneciente a la ruta del ensayo 1, la Tabla III muestra los datos adquiridos por ambos dispositivos.

Para comprobar el error que existe entre los puntos, se procede a graficar utilizando la aplicación de Google Earth

Pro y medir la distancia de error entre ellos, el error determinado por la aplicación es de 5.74 m (ver Fig. 17).

Desde el punto de vista analítico el error existente entre dos coordenadas geográficas se calcula mediante la función haversine (1) [18].

$$a = \sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\,\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right) \quad (1)$$

$$d = 2R\,\mathrm{atan}\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \quad (2)$$

Donde:
d es la distancia entre dos puntos
R es el radio de la esfera (radio ecuatorial 6378 km)
ϕ_1, ϕ_2 es la latitud del punto 1 y del punto 2
λ_1, λ_2 es la longitud del punto 1 y del punto 2

Al realizar el cálculo de la distancia de error por medio de (1), se obtiene un valor de 5.73 metros. Comparando este valor con el obtenido en Google Earth, se constata la validez de la ecuación, por lo tanto se la utilizará para la evaluación del prototipo en los ensayos 1 y 2.

Para la prueba del prototipo instalado en el vehículo recolector se recorrió dos rutas: ruta prueba y ruta recorrido 2 diurna. La ruta de prueba inició en la urbanización Atamer y el recorrido cubre la Av. Zoilo Rodríguez, calle Vicente Rocafuerte, Av. Emiliano Ortega, calle Loudres, calle Bolívar, Av. Eduardo Kigman, redondel UNL, Av. Pío Jaramillo Alvarado, calle Chile, calle Sucre, calle Saraguro, calle Sozoranga, calle Andrés Bello, calle Juan José Peña, calle Leopoldo Palacios, calle Macara, calle Lourdes, Av. Orillas de Zamora, calle 10 de Agosto y finaliza en la Urb. Atamer donde inició. Para el trazo de la ruta se tomó los datos del prototipo y del GPS Monterra, se lo realizó en un tiempo de 35 minutos, a una velocidad promedio de 22 km/h y un recorrido de 12.7 km. El prototipo registró 198 posiciones y el GPS Monterra registró 416 posiciones. En la Fig. 18a se muestra la ruta trazada por el GPS Monterra y en la Fig. 18b se muestran el trazo de la ruta tomado por el prototipo en la aplicación web del sistema de monitoreo.

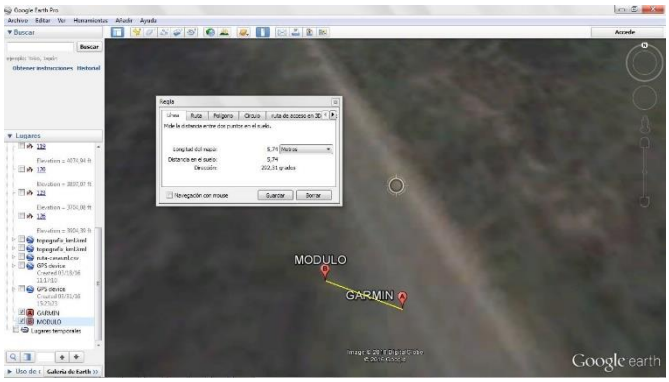

Figura 17. Distancia error entre el dispositivo de monitoreo y el GPS Garmin Monterra.

Por otro lado, la a Ruta 2 diurna inicia en el Barrio el Churo, Barrio Santa Rosa, Barrio San Cayetano Alto, Barrio Las palmas Alto, Barrio Las Palmas, Barrio Isaac Ordoñez, Barrio San Cayetano Bajo, vía Zamora, Barrio Yanacocha, Barrio el Valle, Barrio el Recreo del Valle, Barrio la Samana, Barrio la Granada Estancia de la Norte, Barrio Nueva. Para el trazo ruta 2 diurna se tomó los datos del prototipo y del GPS Monterra, se lo realizó en un tiempo de 2 horas, a una velocidad promedio de 11.6 km/h y un recorrido de 15.6 km. El prototipo registró 490 posiciones y del GPS Monterra registró 796 posiciones. En la Fig. 19a se muestra la ruta trazada por el GPS Monterra y en la Fig. 19b se muestran el trazo de la ruta tomado por el prototipo en la aplicación web del sistema de monitoreo.

Como se puede ver en la Fig. 19a y la Fig. 19b a pesar de ser registrada la información con diferentes equipos se concluye que su trazo es similar y las variaciones de las gráficas son menores, sin embargo hay datos faltantes en el prototipo debido al nivel bajo de la señal de cobertura del proveedor de internet en parte del recorrido de prueba.

El principal inconveniente que existió durante las pruebas fue la falta de cobertura móvil en lugares de acceso limitado, por lo que no permitió obtener los datos en partes del recorrido que realizó el vehículo recolector. En la Fig. 20 se observa las zonas obscuras en las cuales se pierde señal y recepción de datos, esto no depende del sistema de monitoreo sino de la operadora móvil proveedora del servicio de telefonía móvil.

C. Análisis de resultados

Al finalizar las pruebas de funcionamiento del prototipo propuesto se obtuvo un conjunto de datos que fueron comparados con los datos del GPS Monterra, para el análisis de datos se desarrolló una subrutina que incluye: lectura de archivos .csv, interpolaciones, visualización gráfica y cálculo de errores en base a la función haversine.

1) Análisis de la ruta de prueba

Se realiza la comparación de la variable latitud respecto al tiempo de los dos dispositivos, obteniendo que el error máximo es 7.51 m y el error promedio 1.96 m (ver Fig. 21a). Se realiza la comparación de la variable longitud respecto al tiempo de los dos dispositivos, obteniendo que el error máximo es 8.2 m y el error promedio 2.24 m (ver Fig. 21b). Se grafica el error total del prototipo obteniendo que el valor máximo es 8.3 m y el valor promedio es 3.35 m (ver Fig. 21c), siendo este un valor aceptable para el prototipo.

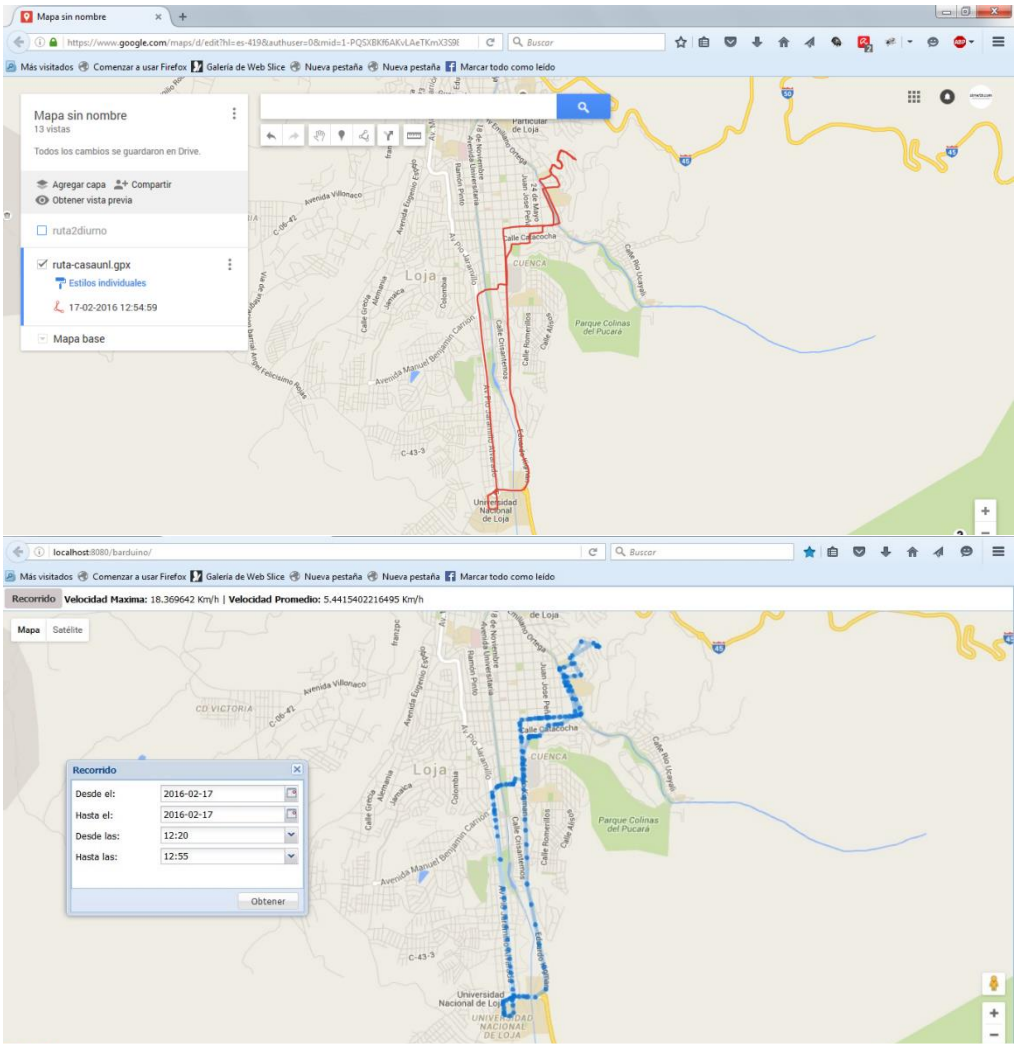

Figura 18. Trazo de las rutas a) GPS Monterra, b) prototipo en la aplicación web.

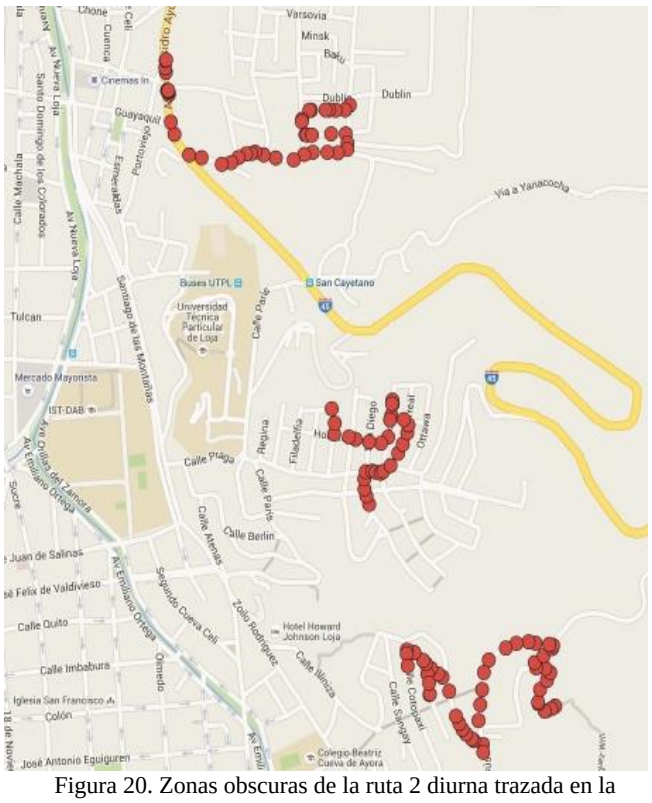

Figura 20. Zonas obscuras de la ruta 2 diurna trazada en la aplicación web.

2) Análisis de la ruta 2 diurna

Se realiza la comparación de la variable latitud respecto al tiempo de los dos dispositivos, obteniendo que el error máximo es 17.43 m y el error promedio 2.25 m (ver Fig. 22a). Se realiza la comparación de la variable longitud respecto al tiempo de los dos dispositivos, obteniendo que el error máximo es 15.40 m y el error promedio 2.44 m (ver Fig. 22b). Se grafica el error total del prototipo obteniendo que el valor máximo es 19.68 m y el valor promedio es 3.67 m (ver Fig. 22c), siendo este un valor aceptable para el prototipo.

Estos niveles de error no son significativos para la aplicación del dispositivo de monitoreo de la posición de los vehículos de recolección y transporte de RSM. Así también, del análisis de los datos se concluye que existen lugares donde es baja la señal de cobertura de la red celular y debido a ello el prototipo no envía los datos adquiridos. La solución planteada para eliminar este evento es reemplazar la antena de recepción GPRS (ganancia 3.5 dBi) por una antena con mayor ganancia.

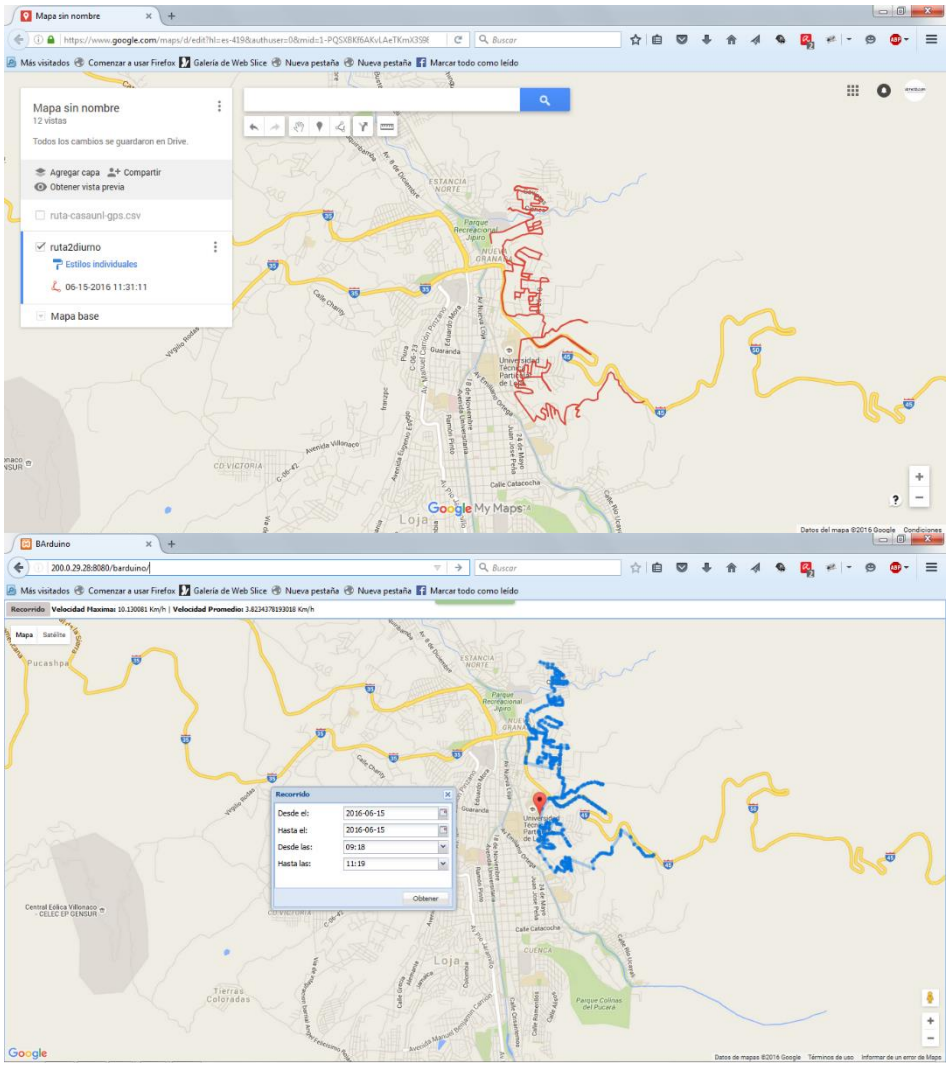

Figura 19. Trazo de las rutas a) Ruta 2 diurna trazada por el GPS Monterra, b) Ruta 2 diurna trazada por el prototipo en la aplicación web.

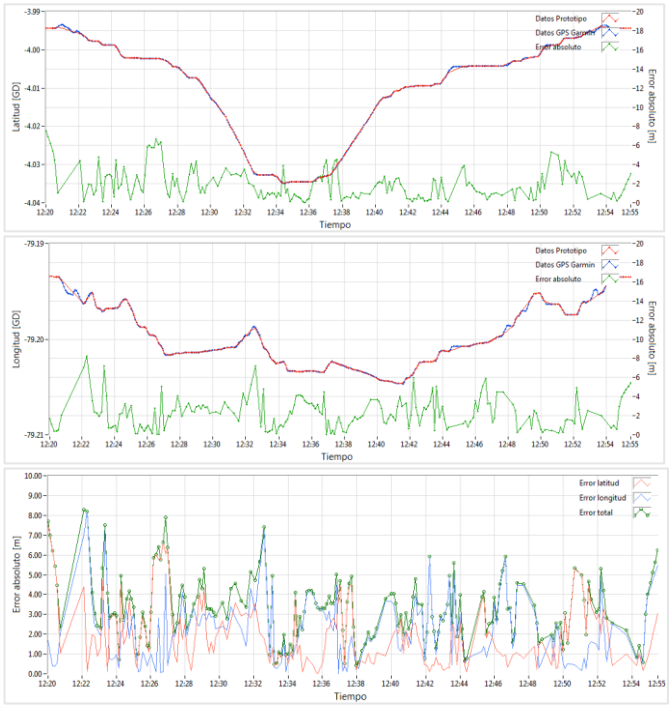

Figura 21. Ruta de prueba a) Latitud con respecto al tiempo, b) Longitud con respecto al tiempo, c) Error absoluto con respecto al tiempo.

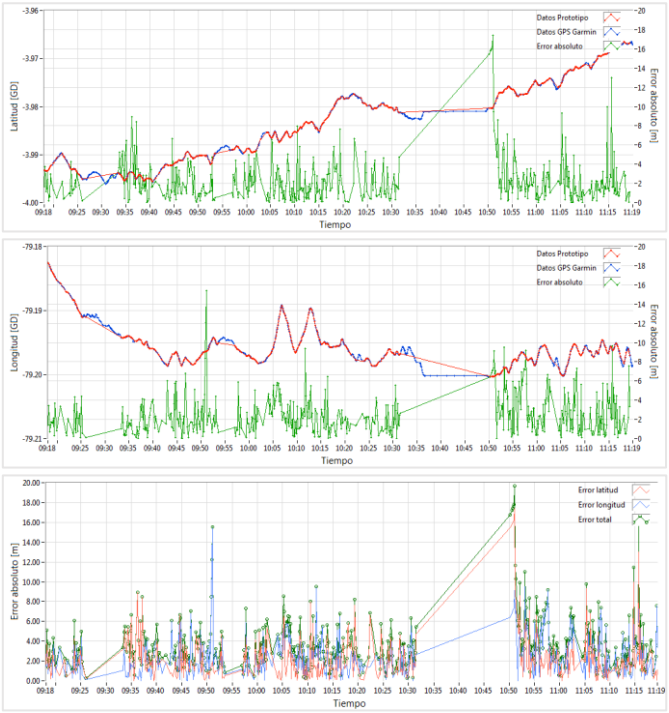

Figura 22. Ruta 2 diurna a) Latitud con respecto al tiempo, b) Longitud con respecto al tiempo, c) Error absoluto con respecto al tiempo.

V. CONCLUSIONES

Se desarrolló un sistema de monitoreo conformado por tres módulos principales: 1) módulo de procesamiento que contiene la unidad de procesamiento, 2) módulo de adquisición y envío de datos que consta de unidad de recepción GPS y unidad de envío de datos GPRS, estos dos módulos (prototipo AVL) están conectados a la unidad de abastecimiento de energía y mediante protocolo serial entre ellos, y 3) módulo servidor que consta de la unidad conexión, base de datos y aplicación web, este módulo se conecta mediante conexión GSM/ GPRS al prototipo AVL.

Se realizó dos pruebas de campo: el ensayo 1 corresponde a un recorrido de prueba por la zona céntrica de la ciudad de Loja realizado el día 17 de febrero de 2016 a una velocidad promedio de 22 km/h y un recorrido de 12.7 km, el ensayo 2 corresponde a la ruta 2 diurna de recolección y transporte de RSM realizado el día 15 de junio de 2016 a una velocidad promedio de 11.6 km/h y un recorrido de 15.6 km. Como resultado se obtuvo un error promedio de posición de 3.67 m y un error máximo de 19.68 m. Este nivel de error no es significativo para la futura aplicación del dispositivo de monitoreo en el seguimiento del sistema de RT-RSM, existen lugares donde es baja la señal de cobertura y debido a ello el prototipo no envía los datos adquiridos. La solución planteada para eliminar este evento es reemplazar la antena de recepción GPRS (ganancia 3.5dBi) por una antena con mayor ganancia.

REFERENCIAS

[1] J. Ojeda, "Análisis situacional de la gestión de residuos sólidos urbanos casos: Catalunya (España) y Loja (Ecuador)" M. S. thesis, Universidad Autónoma de Madrid, Madrid, España, 2009.

[2] G. Vila, "Proyecto de Gestión Integral de Residuos Sólidos Loja, Ecuador". *Biblioteca CF+S* [Online], 2002. Available: http://habitat.aq.upm.es/bpal/onu02/bp014.html. [Accessed: 1-Jul-2016]

[3] J. Medina and I. Jiménez, *Guía para la gestión integral de los residuos sólidos municipales*. México: Secretaría de Medio Ambiente y Recursos Naturales, 2001.

[4] L. Ecamirosa, C. Del Carpio, G. Castañeda and C. Quintal. *Manejo de los residuos sólidos domiciliarios: Tuxtla Gutiérrez, Chiapas*. México: Universidad Autónoma de Chiapas, 2001, ch. 5.

[5] E. Betanzo, M. Torres, J. Romero and S. Obregón, "Evaluación de rutas de recolección de residuos sólidos urbanos con apoyo de dispositivos de rastreo satelital: análisis e implicaciones", *Revista Internacional de Contaminación Ambiental*, 2016, vol. 32, no. 3, pp. 323-337.

[6] O. Aloquili, A. Elbanna and A. Al-Azizi, "Automatic vehicle location tracking system based on GIS environment", *IET software*, 2009, vol. 3, no. 4, pp. 255-263.

[7] B. Ghribi and L. Logrippo. "Understanding GPRS: the GSM packet radio service", *Computer Networks*, 2000, vol. 34, no. 5, pp. 763-779.

[8] I. M. Almomani, N. Y. Alkhalil, E. M. Ahmad and R. M. Jodeh, "Ubiquitous GPS vehicle tracking and management system", *Applied Electrical Engineering and Computing Technologies (AEECT), 2011 IEEE Jordan Conference on*, Amman, 2011, pp. 1-6.

[9] B. Kodavati, V. K. Raju, S. Srinivasa, A. V. Prabu, T. Appa and Y. V. Narayana, "GSM and GPS based vehicle location and tracking system", *International Journal of Engineering Research and Applications (IJERA)*, 2011, pp. 616-625.

[10] S. Lee, G. Tewolde and J. Kwon, "Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application", *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, Seoul, 2014, pp. 353-358.

[11] M. Kamel, "Real-Time GPS/GPRS Based Vehicle Tracking System", *International Journal of Engineering and Computer Science*, 2015, vol. 4, no. 8, pp. 648-652.

[12] S. Murugananham and P. R. Mukesh, "Real time web based vehicle tracking using GPS", *J. World Acad. Sci. Eng. Technol*, 2010, vol. 61, pp. 91-99.

[13] A. S. Dinkar and S. A. Shaikh, "Design and implementation of Vehicle tracking system using GPS", *Journal of Information Engineering and Applications*, 2011, vol. 1, no. 3, pp. 1-7.

[14] P. K. Harshadbhai, "Design of GPS and GSM based vehicle location and tracking system", *International Journal of Science and Research (IJSR)*, 2013, vol. 2, no. 1, pp. 165-168.

[15] Organización Panamericana de la Salud, "Estudio de factibilidad". *Biblioteca virtual de desarrollo sostenible y salud ambiental (BVDSDE)* [Online]. Available: www.bvsde.paho.org/bvsacd/cd61/guaviare/factibilidad.pdf. [Accessed: 1-Jul-2016].

[16] H. Tan, "Design and Implementation of Vehicle Monitoring System Based on GSM/GIS/GPS", *Information Technology and Computer Science (ITCS), 2010 Second International Conference on*, Kiev, 2010, pp. 413-416.

[17] Garmin International, Inc., "Monterra™ Manual del usuario", pp. 1-20, Octubre 2013.

[18] Z. U. A. Lodhi, A. Basit, A. F. Khan, A. Waheed and M. Nasir, "Sensor Fusion Based Data Parser of a GPS Receiver for UAV Systems", *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, Harbin, 2012, pp. 95-99.