

UNIVERSIDAD
NACIONAL
DE COLOMBIA

**Prototipo Sistema de monitoreo basado en
una red inalámbrica de sensores simulada,
como apoyo a la planeación de rutas de
recolección de basuras.**

Universidad Nacional de Colombia
Facultad de Ingeniería
Bogotá, Colombia
2020

Prototipo Sistema de monitoreo basado en una red inalámbrica de sensores simulada, como apoyo a la planeación de rutas de recolección de basuras.

Miguel Ángel Montañez Gómez

Trabajo final de Maestría para optar al título de:
Magister en Ingeniería de Sistemas y Computación

Director (a):
Ph.D. Luis Fernando Niño Vásquez

Línea de Investigación: Computación aplicada

Grupo de Investigación:
Laboratorio de investigación en sistemas inteligentes (LISI)

Universidad Nacional de Colombia
Facultad Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2020

A Dios por tantas bendiciones concedidas y la sabiduría para usarlas correctamente.

A mi esposa, por su apoyo incondicional y sus maravillosos sueños que me motivan a ser mejor cada día.

A mis padres por su convicción en mis capacidades y absoluta confianza

A mis compañeros y profesores por compartir su invaluable conocimiento y ayuda

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.



Miguel Ángel Montañez Gómez

Fecha 15/04/2021

Fecha

Resumen

Prototipo de Sistema de monitoreo basado en una red inalámbrica de sensores simulada, como apoyo a la planeación de rutas de recolección de basuras

La producción en masa y el aumento de la población en los asentamientos urbanos ha generado un incremento proporcional en la generación de residuos urbanos, el tratamiento eficiente de dichos residuos se ha convertido en un desafío para las agencias de limpieza, las cuales siguen realizando dicha operación usando rutas periódicas fijas en las que se recorren exhaustivamente las calles de los barrios en búsqueda de los residuos, en estas operaciones no se separa el material aprovechable del que no lo es, causando un impacto ambiental negativo. La propuesta en este trabajo consiste en un sistema que calcule las rutas de recolección considerando los niveles de llenado reportados por los contenedores de basura, para ello los contenedores son dotados con un dispositivo de medición de nivel, la información se transmite por medio una red IoT que soporta comunicación de larga distancia, los datos reportados por los contenedores son usados para decidir cuáles de los contenedores deben ser incluidos o excluidos de la ruta que se está planificando de acuerdo con el nivel de llenado que reportan. Dado que para la operación de recolección se requieren múltiples camiones, se usó un método llamado *k-means*, con el cual se agrupan los contenedores que están más cercanos geográficamente, las secuencia de recolección se delegó a un servicio Web de uso libre, el resultado se muestra por medio de una aplicación móvil en Android, la aplicación utiliza el servicio de enrutamiento ofrecido por Google Mapas, esto demuestra la viabilidad técnica para la implementación de este tipo de sistemas a operaciones en ambientes reales, se utilizó como región de estudio la localidad de Engativá en la Ciudad de Bogotá.

Palabras clave: IoT, recolección de residuos urbanos, red de sensores inalámbricos, LoRa, sistema de monitoreo, enrutamiento, información geográfica, aplicación móvil.

Abstract

Sensing system Prototype base on a Wireless Sensor Network as support for Solid Waste collection route planning:

Mass production and population growth in urban settlements has produced an increase in generation of municipal solid waste MSW, then efficient treatment of waste has become a challenge for cleaning enterprises, as they continue to perform collecting operation using fixed periodic routes, that exhaustively go across neighbourhood streets in search of waste, further in this operations recyclable material is not separated from disposable one, causing a negative impact to the environment. This work aims to prototype a sensing system that generates routes based on actual fulfilment level reported by dumpsters, to accomplish so dumpsters are equipped with a level measurement device, information gets transmitted using an Long Range IoT network, collected data is used to determine which dumpsters are to be collected and which are not in the route being planned, according to fulfilments reported levels. Since commonly an operation requires many collecting trucks, a k-means method is used to group dumpsters geographically closer, collecting sequence is calculated using an open Web service, results are shown on Android mobile application, the mobile app consumes Google Maps routing service, this demonstrates technical viability for implementing this sorts of systems in real operation environments, a single district called Engativá in Bogota was selected as study Area.

Keywords: IoT, Urban Waste Collection, wireless sensor network, LoRa, sensing system, routing, geographic information, mobile application.

Contenido

Introducción	19
Marco Conceptual	23
Desarrollo Sostenible	23
Reciclaje	23
Sistemas de recolección de residuos.	25
Sistemas geo referenciados	26
Sistema de posicionamiento global GPS	27
Internet de las Cosas	29
Sistemas embebidos	30
Sensores	31
Telecomunicaciones	32
Comunicaciones inalámbricas	32
LoRa	33
loraWAN	35
Tipos de dispositivos	36
Autenticación	37
Bandas ISM	39
Protocolos de comunicación nivel de aplicación.	40
Protocolo avanzado para encolamiento de mensajes (AMQP)	40
Transporte de telemetría por cola de mensajes (MQTT)	41
Protocolo para aplicaciones con restricciones (CoAP)	42
Seguridad en IoT	44
Antecedentes	46
Algoritmos genéticos y de enjambre	49
Soluciones basadas en IoT	51
Marcos de referencia para la implementación	55

Desafíos para la implementación	56
Implementación de Dispositivos de Medición	59
Arduino	63
Conexión de los componentes físicos	67
Programación lógica del micro-controlador	68
Pruebas de emisión de los mensajes	70
Dragino	71
Conexión de los componentes físicos	73
Pruebas de emisión de los mensajes	74
TTGO	75
Conexión de los componentes físicos	77
Pruebas de emisión de datos	78
Comparación de las opciones:	78
Librería Arduino-LMIC de Matthijs Kooijman	79
Consideraciones adicionales de nodo de medición	80
Configuración de RIS y Arquitectura IoT	82
Gateway	82
LG02 de Dragino.	83
Configuración del Gateway	84
Servidor IoT	88
The Things Network TTN	88
Registro de dispositivos	89
Prueba de recepción de datos:	94
Pruebas con TTGO (01nodo):	94
Pruebas con Shield (Dragino):	96
Pruebas con Arduino y SX1276	97
Bróker MQTT	98
Prueba Cliente MQTT	100

Base de Datos	102
Visualización de los datos obtenidos.	104
Alcance de la señal	104
Simulación de Datos	109
Conjunto de datos	110
Depuración de datos	111
Análisis exploratorio	113
Representación geográfica de las rutas	118
Simulación	119
Generación de datos	121
Representación geográfica de los datos generados	122
Técnicas de agrupamiento	124
Generación de Rutas de Recolección	128
Problema del enrutamiento de vehículos	128
Open Route Service ORS	129
Servicio de rutas	129
Implementación.	130
Programación de la Interfaz Visualización de Rutas	132
Flutter:	132
Autenticación JWT	133
Consumo de servicios Rest	135
Integración con Google Maps	137
Configuración del Servicio	137
Implementación del mapa	138
Directions Service	141
Generación, firma del APK.	143
8.5. Consideraciones para Despliegue del Software en Entornos de Producción	144
Conclusiones y recomendaciones	147

Conclusiones	147
Recomendaciones	149
Bibliografía	153

Listas de figuras

	Pág.
Figura 2-1 Proyección de coordenadas geográficas sobre la superficie terrestre	24
Figura 2-2 Trilateración de distancia para determinar posición relativa	25
Figura 2-3 Evolución número de dispositivos conectados a internet	27
Figura 2-4 Arquitectura típica red LoRaWAN	33
Figura 2-5 Activación en el aire OTAA	35
Figura 2-6 Autenticación por personalización ABP	35
Figura 2-7 Secuencia de comunicación AMQP	38
Figura 2-8 Esquema de interacción cliente bróker en MQTT	39
Figura 2-9 Ejemplo respuesta inmediata req/respond CoAP	40
Figura 2-10 Ejemplo respuesta posterior req/respond CoAP	40
Figura 3-1 Solución generada por ablandamiento térmico en la ciudad de Sanandaj	44
Figura 3-2 Diagrama de flujo general del algoritmo genético	47
Figura 3-3 Fundamentos del marco de referencia de sistemas sostenibles e inteligentes de gestión de RSU	52
Figura 4-1 Diagrama de bloques arquitectura del dispositivo de medición.	56
Figura 4-2 Sensor de proximidad laser Gyvl53l0xv2	57
Figura 4-3 Sensor de proximidad infrarrojo FC-51	58
Figura 4-4 Sensor de proximidad por ultrasonido HC-SR04.	58
Figura 4-5 Módulo de transmisión SX1276	61
Figura 4-6 Módulo de localización Neo 6M Ublox	62
Figura 4-7 Esquema de conexión Arduino, SX1276 neo 6M y HC-SR04.	64
Figura 4-8 Ensamblado de los componentes físicos configuración con arduino UNO	65
Figura 4-9 Segmento del programa en el entorno de arduino estudio.	66
Figura 4-10 Configuración de la tarjeta en el entorno de desarrollo.	67
Figura 4-11 Verificación de las operaciones en el micro controlador desde el monitor serial en Arduino.	68
Figura 4-12 Integración de Dragino Shield y Arduino Mega 2560	69
Figura 4-13 Esquema de conexión Arduino Mega, Dragino Shield y HC-SR04	70
Figura 4-14 Verificación de las operaciones en el micro controlador desde el monitor serial en Dragino.	72
Figura 4-15 Tarjeta TTGO compatible con arduino SX1278	73
Figura 4-16 Esquema de conexión TTGO y HC-SR04	74

Figura 4-17 Verificación de las operaciones en el micro controlador desde el monitor serial en TTGO	75
Figura 4-18 Instalación de la Librería desde el gestor de librerías de arduino studio.	76
Figura 4-19 Diagrama de bloques para nodo de bajo consumo	78
Figura 5-1 Diagrama general componentes Gateway LG02	80
Figura 5-2 Interfaz de configuración del LG02.	81
Figura 5-3 Configuración del canal de recepción LG02.	82
Figura 5-4 Configuración del canal de transmisión LG02.	83
Figura 5-5 Configuración de los parámetros de transmisión y recepción en la librería	83
Figura 5-6 Registro de parámetros de operación del LG02.	84
Figura 5-7 Registro de mensajes retransmitidos por LG02.	84
Figura 5-8 Localización de Gateways registrados en Bogotá.	86
Figura 5-9 Registro exitoso del Gateway en la consola de TTN.	87
Figura 5-10 Selección de un manejador de tráfico para la aplicación.	88
Figura 5-11 Información de la aplicación creada.	88
Figura 5-12 Registro del 01nodo en la aplicación TTN.	89
Figura 5-13 Lista de dispositivos asociados a la aplicación	89
Figura 5-14 Configuración de los códigos de autenticación OTAA en los nodos.	90
Figura 5-15 Organización del arreglo con carga útil en los Nodos finales.	90
Figura 5-16 Función para decodificación de la carga útil recibida en el servidor.	91
Figura 5-17 Inspección del registro del 01nodo usando el monitor serial.	91
Figura 5-18 Datos emitidos por el 01nodo en la aplicación TTN	92
Figura 5-19 Salida de la función de decodificación en tiempo de ejecución.	92
Figura 5-20 Inspección del registro del Dragino usando el monitor serial.	93
Figura 5-21 Datos emitidos por el Dragino en la aplicación TTN.	93
Figura 5-22 Inspección del registro del Arduino usando el monitor serial.	94
Figura 5-23 Datos emitidos por el Arduino en la aplicación TTN.	94
Figura 5-24 Datos recibidos tema de dispositivos en la app 01basurasbogota.	95
Figura 5-25 Mensajes recibidos en el servidor para la app 01basurasbogota.	96
Figura 5-26 Datos recibidos por mensajes ascendentes en la app 01basurasbogota.	96
Figura 5-27 Función SDK TTN que se suscribe a los mensajes ascendentes de la aplicación mediante mosquitto.	97
Figura 5-28 Función pos procesamiento de datos.	97
Figura 5-29 Registro de datos de la aplicación TTN.	98
Figura 5-30 Registro de datos del cliente MQTT en java.	98
Figura 5-31 Consulta de los últimos registros recibidos en la tabla de mediciones.	99
Figura 5-32 Modelo relacional del segmento de BD relacionado con contenedores y mediciones, generado usando DBeaver.	100
Figura 5-33 Interfaz web representación geográfica de los contenedores y las mediciones más recientes.	101
Figura 5-34 Coordenadas reportadas por el arduino en pruebas de alcance.	102
Figura 5-35 Distancia aproximada de alcance usando el circuito de arduino.	103

Figura 5-36 Imagen de Dragino emitiendo señal y antena apuntando en dirección al Gateway ubicado en el edificio que se ve en la parte posterior.	104
Figura 5-37 Coordenadas reportadas por Dragino en pruebas de alcance.	104
Figura 5-38 Distancia aproximada de alcance usando el circuito Dragino.	105
Figura 6-1 Estructura del conjunto de datos.	108
Figura 6-2 Codificación de la función para hallar la distancia en una superficie esférica.	109
Figura 6-3 Resumen distancias por ruta.	111
Figura 6-4 Resumen Pesos de la carga por ruta.	111
Figura 6-5 Histograma general de cargas.	112
Figura 6-6 Histograma general de cargas.	113
Figura 6-7 Relación entre las longitudes de las rutas (Km) y el volumen promedio recolectado en la misma (L).	114
Figura 6-8 Evolución en la generación de Residuos en la ciudad de Austin entre 2004 y 2017.	114
Figura 6-9 Mapa con rutas clasificadas según el promedio de carga recolectado.	115
Figura 6-10 Mapas correspondientes a las mallas viales y manzanas	116
Figura 6-11 Estructura del conjunto de datos geográficos catastrales de Bogotá.	117
Figura 6-12 Estructura DB del proyecto luego de Migrar las tablas con la información espacial de catastro distrital.	118
Figura 6-13 Fase 1 simulación de las ubicaciones de los contenedores.	120
Figura 6-14 Fase 2 simulaciones de los niveles de llenado en los contenedores.	120
Figura 6-15 Análisis de convergencia cantidad de grupos basado en el método del codo.	123
Figura 6-16 Calles priorizadas agrupadas usando K = 1.	124
Figura 6-17 Calles priorizadas agrupadas usando K = 9.	124
Figura 7-1 Inclusión del botón para calcular rutas en la aplicación Web.	127
Figura 7-2 Configuración de prueba del servicio de enrutamiento.	128
Figura 7-3 Respuesta del servicio de enrutamiento.	128
Figura 8-1 Ejemplo de información transmitida usando el estándar JWT.	130
Figura 8-2 Implementación de la solicitud de token usando usuario y contraseña.	131
Figura 8-3 Interfaz de autenticación y consulta de rutas, selector de fechas.	132
Figura 8-4 Fragmento de código implementación de HTTP para consulta de rutas.	133
Figura 8-5 Interfaz con la lista de rutas encontradas para una fecha específica.	133
Figura 8-6 Generación de la clave para el consumo de API.	134
Figura 8-7 Configuración para el consentimiento de OAuth.	135
Figura 8-8 Segmento código del Constructor Componente Google Mapas App Móvil	136
Figura 8-9 Tipos de mapas ofrecidos google maps	136
Figura 8-10 Interfaz del mapa suministrado por Google con marcadores para los contenedores de una Ruta.	137
Figura 8-12 Interfaz de direccionamiento calculado por el servicio de Google. (entre la ubicación inicial del transportador y el primer contenedor, progreso de ruta 14,8%)	139
Figura 8-13 Interfaz de direccionamiento entre 2 contenedores de la ruta 131.	139

Figura 8-14 Resultado de la generación del APK	140
Figura 8- 15 Diagrama de componentes de Software	141
Figura 10-1 Comprobación de la instalación y los requisitos usando flutter doctor.	147
Figura 10-2 Emulador de Android elegido para el desarrollo.	148
Figura 10-3 Conexión del emulador usando ADB.	148
Figura 10-4 Infografía funcionamiento de los nuevos contenedores en Bogotá	149

Lista de tablas

	Pág.
Tabla 1-1 Abreviaturas	15
Tabla 2-1 Rangos de frecuencias autorizados para Banda ISM internacionalmente	35
Tabla 3-1 Pruebas de alcance y efectividad de trasmisión con ZigBee, tomada y traducida de (Karthikeyan, Rani, Sridevi & Bhuvaneswari, 2017)	48
Tabla 3-2 Síntesis de los métodos hallados en la revisión sistemática de literatura.	53
Tabla 4-1 Comparación sensores de proximidad.	58
Tabla 4-2 Conexiones de arduino uno a SX1276	63
Tabla 4-3 Conexiones de arduino uno a HC-SR04	64
Tabla 4-4 conexiones de arduino uno a HC-SR04	64
Tabla 4-5 Conexiones de Dragino a HC-SR04	69
Tabla 4-6 Conexiones entre arduino Mega 2560 y Dragino Shield	70
Tabla 4-7 conexiones pines de TTGO y HC-SR04	73
Tabla 4-8 Tabla comparativa de los nodos de medición.	74
Tabla 6-1 Código de color para representación de prioridades.	118
Tabla 6-2 Definición de prioridades de las calles según niveles reportados.	120
Tabla 8 -1 directorio al repositorio y código fuente de las aplicaciones.	143

Abreviaturas

Abreviatura	Término
<i>AMQP</i>	Protocolo avanzado para el encolamiento de mensajes
<i>CIA</i>	Confiabilidad, Integridad y Disponibilidad
<i>CoAP</i>	Protocolo para aplicaciones con restricciones
<i>COP</i>	Pesos Colombianos
<i>dB</i>	Decibeles
<i>GA</i>	Algoritmo Genético
<i>ML</i>	Aprendizaje de Máquina
<i>MPR</i>	Material Potencialmente Reciclable
<i>GA</i>	Algoritmo Genético
<i>GCP</i>	Plataforma en la nube de Google
<i>GIS</i>	Sistema de información geográfico
<i>GPS</i>	Sistema de posición global
<i>GVNS</i>	Búsqueda por vecindarios con variable general
<i>Hz</i>	Hercios
<i>IA</i>	Inteligencia Artificial
<i>IoT</i>	Internet de las cosas
<i>IMT</i>	Telecomunicaciones móviles internacionales
<i>ISM</i>	Bandas industriales científicas y médica
<i>JWT</i>	JSON Web Token
<i>LoRa</i>	Trasmisión de radio de largo alcance
<i>LPWAN</i>	Red de Amplia cobertura y baja energía
<i>LM</i>	Aprendizaje de Máquina
<i>MPR</i>	Material Potencialmente Reciclable
<i>MQTT</i>	Transporte de telemetría por cola de mensajes
<i>OWASP</i>	Proyecto abierto de seguridad aplicaciones web
<i>ORS</i>	Open Route Service
<i>PVRP</i>	Problema de enrutamiento de vehículos periódicos
<i>RIS</i>	Red Inalámbrica de Sensores
<i>RSU</i>	Residuo Sólido Urbano
<i>SVG</i>	Archivos Gráficos de vectores Escalables
<i>SO</i>	Sistema Operativo
<i>TTN</i>	The Things Network
<i>VRPTW</i>	Problema de enrutamiento de vehículos con ventanas de tiempo

Tabla 1-1 Abreviaturas

1. Introducción

Con la evolución de la tecnología, el crecimiento de la población y las dinámicas económicas globales, las ciudades se ven obligadas a adaptarse rápidamente para ofrecer buenas condiciones de vida a sus habitantes, usar de manera inteligente sus recursos y aprovechar sus ventajas competitivas para generar una economía sostenible. Parte de la infraestructura que requieren las ciudades para satisfacer estas condiciones implican contar con sistemas de abastecimiento suficientes, sistemas de transporte público organizados, redes de apoyo eficientes como departamentos de bomberos y policía, canales de comunicación directos entre las entidades de gobierno y la ciudadanía y la oferta de servicios públicos de calidad a bajo costo, entre ellos servicios de iluminación pública, alcantarillado y telecomunicaciones que sean accesibles para todos sus habitantes.

Este proceso de transformación es un reto al cual se suman otras tareas fundamentales tales como la gestión de residuos, que se ha convertido en una labor compleja debido al aumento apresurado en el volumen de residuos que se genera a causa de 3 factores: el rápido crecimiento demográfico, la producción industrial y los hábitos de consumo que hacen uso excesivo de material desecharable (Karthikeyan, Rani, Sridevi, & Bhuvaneswari, 2018). En un escenario deseable, los residuos deberían ser recogidos a diario para mantener la ciudad limpia y unas condiciones de vida higiénicas. Lamentablemente, en la realidad esta situación no es viable, por una parte, porque la operación de recolección es costosa y, por otra, porque no se cuenta con los recursos suficientes (carros y personal) para atender la demanda de recolección con la frecuencia deseada.

Por otra parte, el rápido desarrollo de Internet ha llevado la conectividad a un nivel asombroso que ha permitido conectar lugares remotos para transmitir datos de manera eficiente y confiable; en este proceso ha surgido un paradigma prometedor para automatizar y controlar muchas de las actividades de nuestra vida diaria llamado Internet de las Cosas (IoT, por su sigla en inglés), El término IoT está estrechamente relacionado con los ambientes inteligentes y agentes embebidos que surgieron de la visión de Mark Weiser en 1991, en un documento llamado "*The Computer for the 21st Century*" donde el autor describe un mundo compuesto por objetos comunes que pueden comunicar sus percepciones e interactuar con otros objetos con cierto grado de autonomía, locomoción y una intervención humana mínima, para mejorar la calidad de vida de múltiples maneras (S. Ray et al., 2018)

El concepto se ha venido transformando y refinando con el desarrollo, la oferta pública y reducción de costos en tecnologías como sistemas lógicos programables, sensores portables, y se ha aplicado en la solución de múltiples situaciones, por ejemplo, la reducción del consumo de la energía en edificios (Pocero, Amaxilatis, Mylonas, & Chatzigiannakis, 2017), operaciones logísticas (Zhong, Lan, Xu, Dai, & Huang, 2016), en cadenas de suministro (Verdouw, Beulens, & van der Vorst, 2013), en asistencia médica

(R. Kumar & Pallikonda Rajasekaran, 2016) y agricultura de precisión (P. P. Ray, 2017), también, más recientemente se ha aplicado a la resolución del problema de recolección eficiente de residuos en ciudades.

Una recolección eficiente de residuos se puede resumir en tres fases: la primera en la que los ciudadanos hagan una separación correcta de los residuos orgánicos y de los residuos que pueden ser reutilizados, la segunda consiste en una operación logística coherente que recolecta oportunamente los residuos y los gestione según la naturaleza del residuo llevando el material reciclabl e a plantas de procesamiento y el residuo a los rellenos sanitarios, y la tercera, un tratamiento adecuado de los residuos, especialmente de los aprovechables.

El enfoque más frecuente en los sistemas de recolección basados en IoT son las redes inalámbricas de sensores (RIS), con las que se supervisan las condiciones físicas de las unidades públicas de recolección, conocidas como contenedores de basura. Los volúmenes de información que se generan durante el monitoreo son considerables y deben ser gestionados adecuadamente para no saturar el sistema y permitir la planificación de una operación de recolección acorde con las necesidades reales de la ciudades. Se deben además evitar errores tales como dirigir los camiones a regiones en los que no existe un volumen de residuo que justifique el desplazamiento de los camiones, lo que podría por ejemplo impedir que estos lleguen a lugares donde si existe acumulación de residuo. La acumulación de residuos es un problema inicialmente de tipo estético por el desbordamiento de basuras y malos olores, pero que con el transcurrir de los días atrae animales como ratones y cucarachas, que en ocasiones pueden transmitir enfermedades convirtiendo la situación en un problema de salud pública.

Los sistemas de monitoreo pueden ser complementados anexando datos geográficos de los contenedores y los camiones para tener información completa y precisa, además permiten que la información que se calcule tenga un contexto espacial y, por lo tanto, que al momento de leerla su significado sea sencillo de interpretar usando herramientas como mapas y puntos de referencia.

En este trabajo se busca proponer un sistema de apoyo a la planificación de rutas de recolección de residuos sólidos urbanos (RSU) en la Ciudad de Bogotá, Colombia. Sin embargo, dada la magnitud de esta ciudad, en este estudio se decidió acotar a una sola de sus localidades. Vale anotar que en el momento en que se desarrolló esta investigación no se encontraron repositorios con información de las operaciones de recolección, ni datos históricos de la generación de residuos de la ciudad, por lo tanto para realizar una prueba de concepto del sistema fue necesario simular datos haciendo una regresión basada en un conjunto de datos de las rutas de recolección de residuos en la ciudad de Austin, Texas, Estados Unidos.

Considerando los factores previamente mencionados se hizo una revisión de literatura que buscaba identificar las tendencias y las soluciones que han reportado mejores resultados en la planificación y asistencia de operación de recolección de RSU, con el ánimo de encontrar las directrices que permitieran proponer un sistema de recolección que se adaptara a las condiciones de la ciudad de Bogotá. De acuerdo con esto, en este trabajo se planteó el siguiente objetivo general:

Objetivo general:

Desarrollar un prototipo de un sistema de monitoreo de contenedores de RSU, para planificar las rutas de recolección por demanda, usando como insumo información simulada.

Este objetivo general será desarrollado a través de la implementación de los siguientes objetivos específicos.

1. Desarrollar un dispositivo para medir de forma remota el nivel de llenado en un contenedor de Residuo Sólido Urbano.
2. Implementar una red inalámbrica de sensores con tres dispositivos, usando una arquitectura de Internet de las Cosas (IoT).
3. Simular la recolección de información sobre nivel de llenado de los contenedores de basura ubicados en un área seleccionada de Bogotá (en la localidad de Engativá)
4. Implementar una aplicación que permita generar y visualizar las rutas para la recolección y la localización de los contenedores que se deben atender.

El resto de este documento muestra un panorama completo que explica los detalles de implementación del sistema desarrollado según los objetivos propuestos, especialmente los detalles técnicos, demuestra las capacidades, dificultades y la viabilidad para operar en un ambiente real, demostrando cómo en este momento de la historia contamos con los recursos y herramientas tecnológicas para dar soluciones innovadoras y prácticas a problemas tan complejos como la recolección de residuos y a otros problemas complejos en el ámbito de la ciudad, los edificios y el hogar.

2. Marco Conceptual

2.1 Desarrollo Sostenible

El desarrollo sostenible en las ciudades es el objetivo número 11 y que debe ser implementado antes de 2030 según la agenda de la ONU (Moran, 2013) para lo cual se definieron políticas y objetivos que permitan hacer un uso eficiente y una distribución justa de los recursos del planeta, de manera que podamos satisfacer nuestras necesidades como especie y convivir con las demás especies sin afectar su entorno de vida. Por lo tanto, es necesario crear hábitos de explotación sostenibles para hacer uso de los recursos, con menor velocidad de la que el planeta está en capacidad de producir y así impedir una crisis por desabastecimiento de recursos vitales como agua y alimento.

Además de los esfuerzos por evitar el aumento desmedido de la población mundial y ocuparse de otros aspectos como el excesivo consumo de carnes, la deforestación, la emisión de gases de efecto invernadero y la conservación de especies en vía de extinción, el tratamiento correcto de los residuos generados por las actividades humanas es una de las principales preocupaciones de los expertos. El tema de la gestión de residuos sólidos abarca por lo menos 4 aspectos que se deben reestructurar para reducir el impacto de la contaminación: reducir la generación de desechos, el reciclaje y reuso de la mayor cantidad de material posible, el tratamiento correcto del material y la disposición segura del desecho.

2.2 Reciclaje

Se entiende por reciclaje la actividad de tratar materiales o productos, usando un proceso industrial especial para extender su vida útil o crear un nuevo producto a partir del material recuperado, para reducir la explotación de recursos y la acumulación de desechos en el mundo. El reciclaje tiene beneficios de tipo ambiental pero también de tipo económico; el material aprovechado tiene un valor importante en la industria y el costo de su tratamiento es mucho menor al costo de explorar recursos para producir nuevo material, esto ocurre con materiales como papel, cartón, vidrio o metal, esto crea un fenómeno conocido como economía circular, que genera oportunidades económicas de negocio y empleo lo cual tiene importantes beneficios sociales.

En Colombia se producen aproximadamente 12 millones de toneladas de RSU al año del cual se recicla sólo el 17%; solo en Bogotá se generan 6.300 toneladas de desechos a

diario. Esto quiere decir que la capital podría llenar cada día la torre Colpatria con las basuras que se producen (Morante, 2018). El 43% del material que llega al relleno sanitario doña Juana está compuesto por plástico, papel, cartón, vidrio y metales que aún pueden ser usados, este material está valuado en mil millones de pesos diarios pero terminan enterrados con el resto del desperdicio (Suarez, 2018).

Estas son unas cifras desalentadoras, si se comparan las cifras con países europeos, como Suecia que es un referente mundial, allí se producen 4,4 millones de toneladas de residuo al año, de los cuales solo el 1% termina en los rellenos sanitarios; esto gracias a las estaciones reciclaje de cada vecindario, en las cuales existen 7 contenedores para separar los residuos, de los cuales aproximadamente el 55% se procesa para crear nuevos productos, el resto de los residuos son llevados a unas fábricas donde se realiza un proceso de clasificación, en el que el material orgánico se usa para elaborar productos como fertilizantes y compost.

El resto de residuos se trata en un proceso llamado *waste-to-energy* (WTE) en el que los residuos se depositan en unos hornos enormes donde son incinerados a 850°C usando biogás, el calor producido se usar para hervir agua, su vapor tiene dos funciones: mover las turbinas para generar energía eléctrica que es el suministro en las escuelas y hospitales y alimentar la red de calefacción municipal. El 20% del material ingresado al horno termina convertido en cenizas que posteriormente son enterradas pero que se biodegradan relativamente rápido (Skoglund, 2019).

Dado el excelente manejo que los suecos le dan a los residuos, las fábricas no disponen de material suficiente para incinerar en los hornos, por lo que han optado por importar basura lista para incinerar (sin plásticos, metal o vidrio) de países como Inglaterra, Italia y Francia, a los cuales les cobran entre 30 y 40 euros por tonelada. Otros países que hacen una muy buena gestión de los residuos son: Suiza, Gales, Corea del sur, Austria y Alemania.

La baja eficiencia en el sistema de reciclaje en Bogotá obedece a tres factores. Primero, la baja cultura de separación de los residuos desde los hogares, se estima que solo el 22% de los ciudadanos separa los residuos correctamente, además el 78% restante que no recicla entorpece la labor de separación de quienes sí lo hacen, cuando depositan el residuo general en el contenedor de reciclaje. (Franca, Ribeiro, & Chaves, 2019).

Segundo, el esquema de recolección es informal, las agremiaciones de recicladores no tienen una regulación, no disponen del equipo adecuado, ni de condiciones de trabajo dignas para poder realizar eficientemente esta importante labor. Además, el mal uso de los contenedores ha dificultado el acceso de los recicladores al material reutilizable; el material que no es recuperado por los recicladores es recogido por el camión y mezclado por la compactadora con los demás residuos lo que lo hace inservible.

Por último, la poca cantidad de plantas de tratamiento de material reciclabl que existen en el país limita la capacidad de procesar los residuos recolectados, haciendo que la demanda del material reutilizable no sea alta y que el precio no sea suficiente para hacer de esto una actividad económica rentable (Marmolejo et al, 2011).

2.3 Sistemas de recolección de residuos.

De conformidad con el Decreto 605 de 1996 del Congreso de la República de Colombia, *“el servicio de aseo y saneamiento... debe prestarse de manera continua y sin interrupciones, en pro del bienestar del usuario.”* Este principio fue desacatado en febrero de 2018 debido a una protesta por parte de los trabajadores de la empresa de Acueducto y Alcantarillado de Bogotá (EAAB), quienes en ese momento eran los encargados de la recolección de residuos, pues entraron en cese de actividades, lo cual produjo una crisis sanitaria durante más de 15 días lo que obligó a declarar una emergencia de la ciudad (Rojas Calderón, 2018).

Como parte del nuevo esquema de recolección se definieron dos condiciones que los prestadores de servicio debían atender: el primero la eficiencia y garantía de la prestación del servicio óptimo e inteligente, esto obliga a los operadores a hacerse cargo de los costos de los nuevos equipos (barredoras, camiones recolectores y compactadores nuevos) para ofrecer una mejor prestación del servicio y, segundo, la inclusión de la población de recicladores que habían sido acogidos y formalizados como trabajadores durante la pasada administración, cuyas políticas tenían un fuerte enfoque social, pero que con el cambio de esquema corrían el riesgo de quedar desempleados (Rojas Calderón, 2018).

Tras la polémica social que ocasionó la adjudicación de los contratos de recolección de residuos en la ciudad de Bogotá, el nuevo esquema entró en vigencia el día 12 de febrero de 2018, este dividió la ciudad en 5 zonas que serían atendidas por las empresas privadas a quienes se les otorgaron los contratos, la estrategia de operación de estas empresas consiste en hacer rutas de recolección exhaustivas, zonificadas con una frecuencia de 2 a 3 veces por semana, según la demanda de la zona: Sin embargo, las rutas no están separadas como ocurre en otras ciudades donde se realiza una operación con la que se recolectan los residuos aprovechables que se llevan a puntos de acopio y una ruta diferente se encarga de recolectar el desperdicio que se lleva a los rellenos sanitarios, de forma que la recolección del material reciclabl queda delegada a los recicladores de oficio los cuales no tiene la capacidad operativa para recuperar todo el material aprovechable. El material que no es recolectado termina mezclado con el resto de los desechos, generando un desperdicio e impidiendo que se haga una labor de reciclaje eficiente.

2.4 Sistemas geo referenciados

La Universidad Politécnica de Valencia define la georeferenciación como “*el proceso por el cual se dota de un sistema de referencia de coordenadas terreno a una imagen digital que generalmente se encuentra en coordenadas pixel.*” Así, los elementos en las capas de un mapa tienen un componente que describe su ubicación geográfica, esto permite situarlos en el mapa de manera precisa, lo cual es fundamental en las representaciones cartográficas y en los Sistemas de Información geográfica, GIS por su sigla en inglés (ArcGIS, 2015)

Mientras que la geolocalización se define como “*la identificación de la ubicación de un dispositivo como un radar, un teléfono móvil o un aparato tecnológico conectado a internet*”, en el caso de google maps la aplicación accede a la ubicación de nuestro dispositivo y nos ofrece servicios como la medición de distancias, la ubicación de sitios de interés cercano o el cálculo de trayectorias entre una ubicación de origen y una de destino (DataCentric, 2018).

Vale anotar que se requiere de un marco de referencia para poder definir las ubicaciones en el mundo real, un sistema de coordenadas geográficas permite asignar dichas ubicaciones a los objetos, dentro de estos marcos se tienen los sistemas de coordenadas de latitud y longitud o los sistemas de coordenadas cartesianas planas que surgen de una proyección de la superficie terrestre sobre un plano.

La obtención de datos geográficos de dominio específico son útiles para determinar la correlación espacial entre los elementos para explicar fenómenos específicos, por ejemplo, la ubicación de los árboles de un fruto específico permite identificar el asentamiento de especies que se alimentan de este tipo de frutos y sus interacciones con el ecosistema, lo cual es potencialmente útil para la protección de zonas y la preservación de especies, la ubicación de fuentes hidrográficas y la medición de niveles de contaminación en los distintos tramos de las fuentes permiten la detección de agentes contaminantes, protección de la vida marina, la protección y estudio de lugares arqueológicos, estas son solo algunas de las muchas aplicaciones de los sistemas georeferenciados que los hacen importantes para la ciencia y la humanidad.

Latitud y Longitud.

Las mediciones de latitud y longitud son mediciones esféricas que se describen en ángulos (en grados) desde el centro de la tierra hasta un punto de la superficie terrestre la longitud mide los ángulos en dirección este-oeste, que normalmente toman como punto de partida el meridiano de Greenwich, que corresponde con la longitud cero y aumentan de manera positiva hacia el oriente y de forma negativa al occidente, tomando valores entre 180 y -180

De forma análoga latitud mide los grados desde la línea del ecuador hacia los polos, desde el ecuador hacia el norte aumentan de forma positiva y hacia el sur aumenta de forma negativa, tomando valores entre 90 y -90

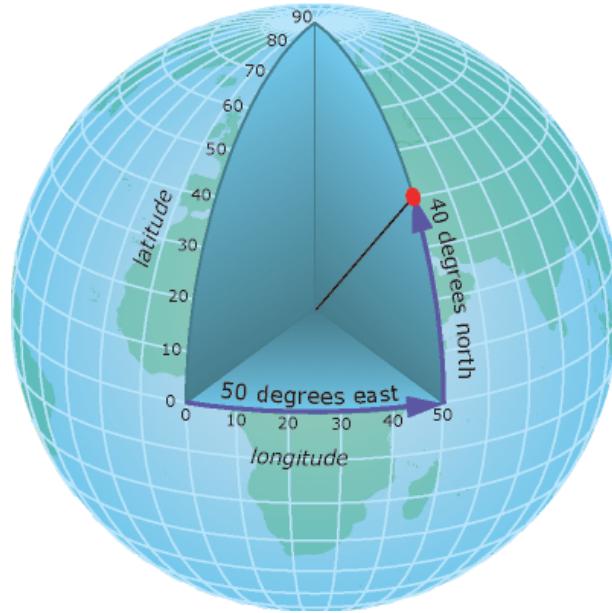


Figura 2-1 Proyección de coordenadas geográficas sobre la superficie terrestre, tomada de (ArcGIS, 2015)

Estas coordenadas no funcionan como unidad de medida pues no son uniformes pues solo a lo largo del ecuador un grado de longitud se aproxima a la distancia en un grado de latitud, pues los círculos que se forman por encima y debajo del Ecuador se van volviendo gradualmente más pequeños, según los estudios de Clark en 1866 un grado de longitud en el ecuador equivale a 111,321 Km, mientras que una latitud de 60° solo equivale a 55,802 Km, lo cual hace que no sean apropiados para medir distancias de forma precisa.

2.5 Sistema de posicionamiento global GPS

El Sistema de Posicionamiento Global GPS es un servicio de propiedad de los Estados Unidos que brinda información a los usuarios sobre su ubicación espacial y su navegación, el servicio está compuesto por los 3 segmentos que se abordan a continuación:

Segmento espacial:

Se refiere a la configuración de una constelación de satélites distribuidos alrededor de la tierra en 6 planos espaciales orbitales separados de forma equidistante, cada uno de los planos dispone de 4 márgenes atendidas por satélites de la línea base, esta

configuración de 24 márgenes garantiza que los usuarios se puedan comunicar por lo menos con 4 satélites para triangular su ubicación prácticamente desde cualquier punto sobre la superficie terrestre. (GPS-US, 2017)

Segmento de control:

Está constituido por un conjunto de sistemas que operan sobre la superficie terrestre para hacer seguimiento y brindar soporte a la constelación de satélites, están compuestos por estaciones de monitoreo, centros maestros de control y antenas en tierra, que operan para monitorear las transmisiones, ejecutar análisis, y enviar comandos a los satélites que están en órbita (GPS-US, 2017).

Segmento de usuario:

Los dispositivos están equipados con unos sistemas de recepción o antenas, capaces de capturar las señales emitidas por los satélites que reciben información de la hora de emisión de la señal y ubicación del satélite; al recibir las emisiones de 3 o 4 satélites se calcula el tiempo que tardó la señal desde que salió del satélite hasta que fue recibida por el dispositivo, dado que la velocidad de la señal es constante (aproximadamente unos 299,792 Km/s) el dispositivo puede calcular la distancia entre su ubicación y la posición del satélite. Luego se usa una técnica de triangulación en la que se traza una circunferencia con la distancia a cada satélite y se encuentra el punto en el que se cruzan todas las circunferencias lo que da una localización precisa de donde se encuentra el dispositivo.

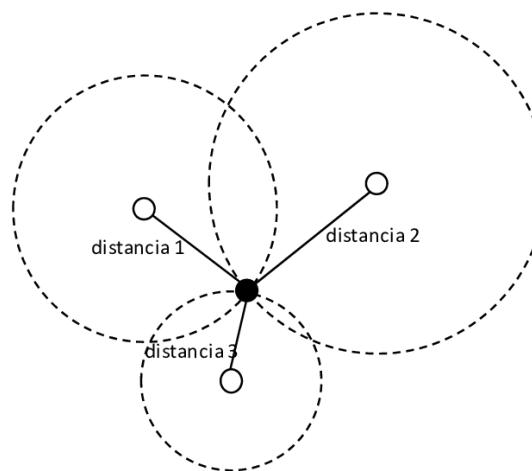


Figura 2-2 Trilateración de distancia para determinar posición relativa,
tomada de: (García et al., 2019)

De manera análoga al Internet, el GPS hace parte de la infraestructura de información global, su uso libre, abierto y confiable ha permitido la generación de nuevas tecnologías y la creación de cientos de aplicaciones, que han servido para transformar y mejorar diversos aspectos de la vida moderna (GPS-US, 2017).

También es importante señalar que el GPS no es el único sistema de posicionamiento que existe, ya que en órbita hay otras constelaciones de satélites que brindan servicios similares entre ellas GLONASS, que ha sido desplegado por el gobierno ruso, BeiDou por China y Galileo por Europa.

2.6 Internet de las Cosas

El término Internet de las Cosas (IoT por su sigla en inglés) está estrechamente relacionado con los ambientes inteligentes y agentes embebidos que surgieron de la visión de Mark Weiser en 1991, en un documento llamado “*The Computer for the 21st Century*” donde el autor describe un mundo compuesto por objetos comunes que pueden comunicar sus percepciones e interactuar con otros objetos con cierto grado de autonomía, locomoción y una intervención humana mínima, para mejorar la calidad de vida de múltiples maneras (S. Ray et al., 2018) Sin embargo, fue Kevin Ashton el primero que introdujo el término Internet de las Cosas (IoT) para referirse a la identificación de los productos de la compañía P&G¹ durante una reunión con los directores de esa empresa en 1999 (Gupta, Mudgal & Mehta, 2016)

Desde su concepción hasta el día de hoy, el término IoT se ha ido refinando y redefiniendo como un paradigma resultante de la integración de elementos tecnológicos y técnicas computacionales como: computación ubicua, sensores y sistemas embebidos, que se usan para construir ecosistemas donde asocian objetos del mundo real, estableciendo una comunicación, flujo de información y una interacción simbiótica entre ellos.

El principio básico para realizar esta asociación es integración de componentes electrónicos a objetos comunes, dotándolos de la capacidad de recolectar información de su entorno y comunicarse con otros sistemas usando internet (Borgia, 2014), estos objetos tienen una representación virtual que sirve para identificarlos, monitorearlos, controlarlos y permitirles interactuar en el mundo real, lo cual permite automatizar multitud de tareas, por lo que muchos lo describen como la evolución del Internet.

¹ Compañía estadounidense dedicada a la producción, distribución y venta de productos de cuidado personal, alimentos y bebidas.

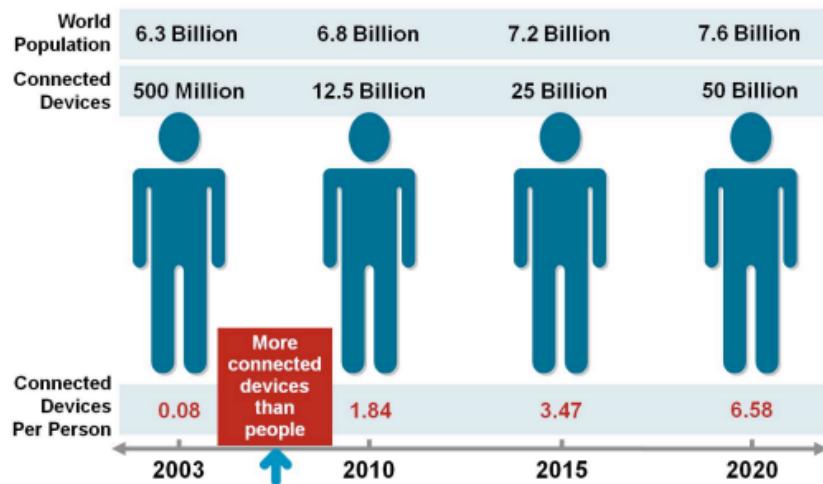


Figura 2-3 Evolución del número de dispositivos conectados a Internet, tomada de (Evans, 2011)

Pero la idea de que los objetos se comuniquen entre sí es una consecuencia directa del rápido aumento en el número de dispositivos que están conectados a Internet, en algún momento entre 2003 y 2010 la cantidad de dispositivos conectados a Internet excedió la población humana. Luego en 2009 los investigadores chinos encontraron un patrón similar a la ley de Moore que demostraba que la cantidad de dispositivos conectados a Internet se duplica cada 5,32 años. Haciendo una proyección de este patrón se predijo que para el año 2020 habría al menos 50 millones de dispositivos conectados a Internet. (Evans, 2011), lo cual hace que Big Data y la ciencia de datos, sean imprescindibles para poder administrar esta cantidad de dispositivos y los datos que estos generan.

Pero la importancia del IoT no está ni en su definición, ni en la cantidad de dispositivos que se conectan a internet, su potencial está en sus aplicaciones, pues los dispositivos IoT mejoran los procesos de transporte por medio del monitoreo de tráfico en tiempo real, en la infraestructura supervisando el flujo de energía en redes eléctricas, en la salud analizando los signos vitales y la evolución de los pacientes, en la industria optimizando las cadenas de producción, en el hogar con aplicaciones de domótica, en las ciudades regulando la iluminación pública, controlando automáticamente sistemas de alcantarillado y los semáforos, también haciendo más eficientes procesos como el de recolección de residuos, entre muchas otras posibles aplicaciones.

2.7 Sistemas embebidos

Un sistema embebido o incrustado es un componente de hardware administrado por un sistema de software, a diferencia los computadores de propósito general que pueden cumplir con un amplio rango de funciones, los sistemas embebidos están diseñados con unas funciones para suplir necesidades específicas. Esto los hace potencialmente útiles

para cumplir tareas sencillas manteniendo bajos sus costos de producción, evitando invertir dinero en procesadores, memorias o discos que exceden la capacidad requerida para una solución concreta tales como controlar las ventas de las máquinas expendedoras y determinar si requieren insumos, o controlar los sensores de humo para detectar incendios y activar los sistemas de riego evitando que el fuego se propague.

Las instrucciones de un sistema embebido se almacenan directamente en su micro controlador, en sus orígenes estas instrucciones se escribían en lenguaje *assembler*, en la actualidad se dispone de lenguajes de más alto nivel que simplifican la programación entre ellos C, C++ y JAVA; sin embargo, uno de los desafíos de los sistemas embebidos acoplados al mundo real es el tratamiento de la concurrencia. Normalmente los sistemas embebidos operan de forma sincrónica y secuencial, pero los eventos en el mundo real, algunas veces ocurren de forma simultánea lo que puede inducir el sistema a incurrir en fallos, de manera que es un aspecto que requiere de la experticia de los programadores para conciliar todos estos eventos en el software de un sistema embebido (Lee, 2017).

El flujo más común en un sistema embebido es tomar como entrada una señal análoga o digital y procesar dicha señal de acuerdo a las reglas y procedimientos que han sido configurados en su microprocesador para producir algún tipo de salida. Algunos sistemas embebidos pueden disponer de una interfaz de usuarios que le permite ajustar su comportamiento dependiendo de las instrucciones del usuario, pero también se suelen acoplar a sistemas eléctricos o mecánicos de mayor magnitud para controlar sus funciones, por ejemplo, cerrar una válvula cuando un contenedor alcance determinado peso o nivel. Comúnmente las señales que reciben los sistemas embebidos como entradas corresponden a cambios en su entorno, pero para que se puedan percibir dichos cambios se requiere de mecanismos auxiliares como sensores.

2.8 Sensores

Un sensor se define como un dispositivo que detecta los cambios de las propiedades, físicas, químicas o eléctricas y produce una señal eléctrica como respuesta al cambio que percibe. Algunas de las propiedades que un sensor puede medir son: luz, calor, movimiento, humedad o presión. Una analogía que facilita la comprensión son los órganos de los sentidos en el cuerpo animal, a través de estos órganos podemos percibir las propiedades externas y de nuestro entorno. Percibimos la luz con nuestros ojos, las ondas de sonido con nuestros oídos y la temperatura con nuestro tacto.

Cuando ponemos la mano en una superficie demasiado caliente, las terminales nerviosas en nuestra piel emiten una señal eléctrica que alertan a nuestro cerebro, este actúa como un procesador generando la orden de contraer los músculos del brazo y retirar la mano de la superficie protegiéndonos de lesiones. Los sensores son órganos artificiales que generan señales con las que se pueden monitorear y medir las propiedades del entorno.

Un ejemplo de sensor es una termocupla, un dispositivo que puede ser sometido a una alta temperatura y generar voltaje de manera proporcional, entre más alta sea la temperatura recibida mayor será el voltaje emitido por la termocupla. Se puede usar este dispositivo para regular la temperatura en multitud de procesos industriales.

2.9 Telecomunicaciones

La telecomunicación es la práctica de transmitir información por medios electromagnéticos, las telecomunicaciones modernas requieren la transmisión de grandes volúmenes de información a través de largas distancias, se requiere que los mecanismos de comunicación sean confiables y robustos, esto quiere decir que mantengan la estructura y contenido del mensaje desde el emisor hasta el receptor para lo cual deben proteger el mensaje del ruido y las interferencias.

Las telecomunicaciones modernas en gran medida transmiten señales de voz y video, que generalmente se reciben de forma análoga, por lo que se realiza un proceso de digitalización, que convierte la señal análoga en pulsos eléctricos binarios que resultan más eficientes al momento de ser transmitidos. El mensaje digital se pasa por un filtro de codificación que consiste en un algoritmo que por medio de fórmulas, elimina la redundancia de la información y reduce el tamaño del mensaje.

El siguiente paso es la modulación del mensaje, en esta fase se altera alguna de las características de una onda (amplitud o frecuencia) de alta frecuencia conocida como portadora para codificar la información y dicha onda se envuelve en una onda de menor frecuencia para proteger los datos y evitar que el mensaje se afecte en el trayecto; usando esta misma onda portadora se puede enviar más de un mensaje dando múltiples accesos, variando el canal de transmisión, este proceso se conoce como multiplexación (Stark, Borth, & Lehnert, 2019).

2.10 Comunicaciones inalámbricas

La tecnología inalámbrica es aquella que hace posible la comunicación entre dos o más dispositivos que se encuentran a distancia, sin que exista una unión cableada entre ellos, dado que las señales se pueden viajar a través del aire o incluso el vacío; en su mayor parte este tipo de comunicaciones usan las ondas de radiofrecuencia que tienen una forma sinusoidal y su frecuencia está definida como la cantidad de veces que se repite la onda en un segundo; esta frecuencia se expresa en Hertz. Por ejemplo, una onda que se repite 100 veces en un segundo es una onda con una frecuencia de 100 Hertz; esta frecuencia está estrechamente relacionada con la longitud de onda que es la distancia que hay entre la cresta de dos ondas.

La tecnología inalámbrica funciona porque una onda electromagnética que viaja por el aire puede generar o inducir una señal eléctrica en una antena, de manera que si se

puede controlar esta onda para codificar en ella los datos que se desean transmitir, también se puede usar para comunicarnos. Las señales de radiofrecuencia (RF) han ganado popularidad porque son relativamente fáciles de generar y tienen la característica de poder atravesar objetos sólidos como paredes y viajar ciertas distancias sin atenuarse. Algunas de las tecnologías inalámbricas basadas en radio más conocidas son: WiFi y Bluetooth, pero existen muchas más como ZigBee, WiMax, LTE y NB-IoT.

Es importante aclarar que las transmisiones de datos sobre redes inalámbricas son considerablemente más lentas, pues alcanza velocidades de hasta 150 Mbps en 4G, mientras que las conexiones por cable pueden alcanzar velocidades de 1Gbps.

Para emitir una señal por radiofrecuencia se necesita un transmisor, en donde un oscilador genera una onda eléctrica, esta señal se propaga por los componentes del circuito que se encargan de modular la onda codificando la información que se desea emitir hasta que llega a una antena. La antena es una pieza de un material conductor por el que la onda eléctrica viaja hasta el final de la antena en donde no tiene más material físico para fluir y necesita liberar la tensión por lo cual irradia la energía sobrante como una onda electromagnética y son estas ondas las que viajan por el aire llevando la información (Serrano, 2016).

El proceso anteriormente descrito es parecido a la amplificación de voz usando un micrófono; cuando se habla por un micrófono, las vibraciones del sonido hacen que vibre una membrana que comprime y descomprime un diafragma, esto causa que la corriente directa que circula por el micrófono varíe (modulación); estos pulsos son amplificados por un circuito y llevados hasta un altavoz que convierte la señal en ondas de sonido que viajan por el aire.

Como se mencionó, en el receptor las ondas crean o inducen pequeñas señales eléctricas en la antena, estas son interpretadas por el circuito eléctrico; sin embargo, la señal que se recibe suele ser tan débil que se necesita de un mecanismo de amplificación antes de procesarla; luego de esto la señal se demodula o decodifica usando un proceso inverso y se obtienen los datos emitidos por el transmisor, con lo cual se completa el envío del mensaje.

2.11 LoRa

Cuando se diseñan aplicaciones con transmisión de datos de forma inalámbrica, es importante reflexionar cuál es la tecnología que se amolda mejor a las necesidades de la aplicación que se desea implementar; por ejemplo, el sistema de WiFi es muy conveniente para transmitir grandes volúmenes de datos a alta velocidad, pero tiene un alcance limitado. Otras tecnologías como las redes Sistema Global para comunicaciones Móviles (GSM, por su sigla en inglés) superan la limitación de distancia pero a un costo energético elevado.

Para algunas de las aplicaciones como el monitoreo de variables ambientales en cultivos agrícolas, el monitoreo de la humedad en una región rural o del PH del agua en un afluente, en los que comúnmente los sitios que se deben monitorear se encuentran separados geográficamente y las variables que se desean medir se puede codificar en unos pocos bytes, la tecnología LoRa (abreviatura de *LOng RAnge modulation*) se adapta muy bien a estas necesidades. Especialmente, si los tiempos de medición de dichas variables no son muy frecuentes, considerando que los datos no varían significativamente en periodos de tiempo prolongados, además, si debido a la ubicación remota de los sensores, resulta difícil realizar operaciones de mantenimiento como el reemplazo de baterías de manera tal que la optimización de la energía se convierte en un aspecto crítico.

LoRa es una tecnología para la trasmisión de información por vía inalámbrica que pertenece a la capa física y es comparable con WiFi o con Bluetooth. Es un mecanismo para codificar información por medio de la modulación de onda diseñada con características de baja transmisión de datos usando bajo consumo energético y amplificando la cobertura a lo largo de redes que son redes conocidas como LPWAN (*Low Power Wide Area Network*).

LoRa es un tipo de modulación de espectro expandido que modula la información usando una señal que varía continuamente su frecuencia FSCM (*Frequency Shift Chirp Modulation*) (Vangelista, 2017), para lo cual usa los parámetros frecuencia inicial (FqI) y final (FqF), su espectro comienza en el valor del parámetro FqI, aumenta de manera lineal hasta alcanzar el parámetro FqF y repite el proceso de expandir la modulación en este rango de frecuencias, que están entre los 915 Mhz en América y Australia, en 868 Mhz en Europa y 433 Mhz en algunos países de Asia, como China, y los cuales corresponden a la banda de frecuencias abiertas para aplicaciones Industriales, Científicas y Médicas (ISM).

La cantidad de bits que se pueden codificar por símbolo depende de otro parámetro llamado factor de propagación SF que indica la cantidad de pitidos que el portador de datos emite por segundo, SF bajos significa que se pueden enviar mas pitidos por segundo; es decir se puede enviar más datos en menos tiempo, por el contrario los SF altos disponen de menos pitidos por segundo con lo cual se pueden codificar menos datos, comparados con los SF bajos, los SF altos requieren de más tiempo de transmisión para codificar el mismo mensaje, esto se conoce como tiempo en el aire (airtime). Los factores de transmisión deben ser seleccionados considerando las variables ambientales entre los dispositivos y el gateway, el beneficio de los SF alto es que extienden el tiempo en el aire lo cual le da mayores oportunidades al receptor de muestrear la potencia de la señal, lo cual produce una mejor sensibilidad este cobra importancia conforme aumenta la distancia de la transmisión (Qoitech, 2019). Los SF se pueden configurar usando 6 valores que van desde el 7 al 12 y estos operan en conjunto con el ancho de banda (BW) que se elija. entre los cuales están disponibles 125, 250 y

500 Mhz. La relación de tasa de transmisión se puede expresar usando la siguiente ecuación:

$$\text{Tasa de transmisión} = SF \times \frac{BW}{2^{\text{sf}}} \quad (1)$$

Algunas de las propiedades más relevantes de esta modulación son: (Solera, 2018)

Capacidad de la red: gracias al concepto de ortogonalidad (OFDM)², LoRa puede codificar varias señales usando para ello diferentes SF sin afectar la decodificación de los mensajes

Localización: esta tecnología fue concebida inicialmente para la exploración con radares por lo que tiene un excelente desempeño en aplicaciones de localización y ubicación.

Envolvente constante: gracias a esta propiedad, la demodulación de la señal es sencilla y puede reutilizar amplificadores con ganancias programables.

Ancho de banda escalable: puede ajustar su ancho de banda y su frecuencia para operar en diferentes canales.

Inmunidad contra el efecto doppler: la modulación que usa LoRa tiene una baja sensibilidad gracias al desplazamiento de frecuencia asociado al efecto doppler; como el receptor está configurado para recibir la señal en un rango de frecuencia deja sin efecto la alteración de la frecuencia cuando los dispositivos de emisión se encuentran en movimiento.

Robustez: esta modulación es altamente resistente ante mecanismos de interferencia pues el receptor tiene selectividad alrededor de los 90 dB y una repulsión de co-canal de 20 dB.

2.12 loraWAN

Como se ha mencionado, mientras que LoRa es una tecnología para la transmisión de datos que modula la frecuencia de la señal a nivel físico, LoRaWAN es una especificación para la implementación de LPWAN que, haciendo una clasificación según el estándar OSI, corresponde al nivel 2, Control de Acceso Medio (MAC, por su sigla en inglés), en la que se definen las secuencias para el intercambio de los paquetes a nivel físico y ofrece servicios para administración del canal, el ancho de banda, la autenticación de dispositivos y el cifrado de datos mediante AES128, para la conformación de redes.

² multiplexación por división de frecuencias ortogonales

Las redes LoRaWAN usan una topología de estrella en la que los nodos finales se comunican con un portal de acceso (*gateway*) que le da a la red acceso a Internet y por medio del cual se enrutan los paquetes enviados por los nodos hasta un servidor IoT, de manera que estos *gateways* conforman una segunda estrella cuyo núcleo es el servidor de IoT.

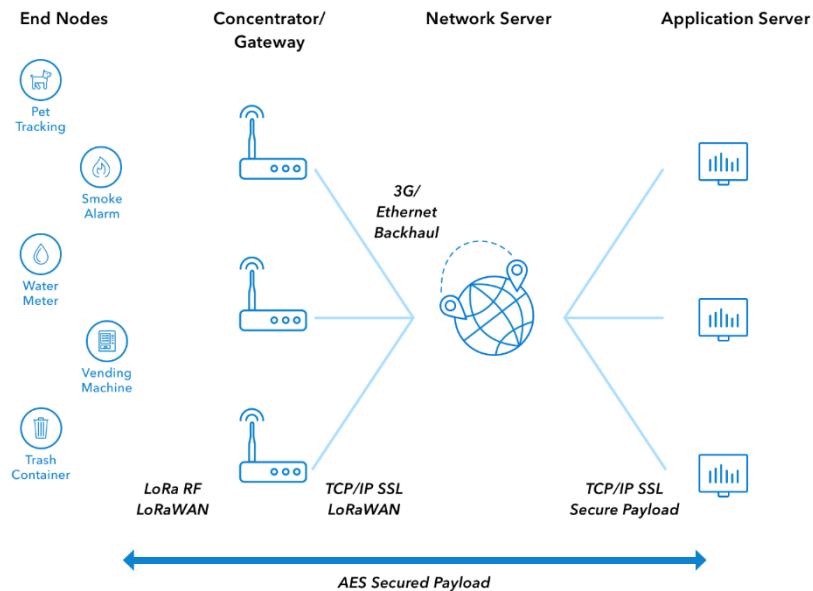


Figura 2-4 Arquitectura típica red LoRaWAN. Tomada de (The Things Network, 2017).

En este protocolo existen dos componentes principales para conectar la red, la primera correspondiente a infraestructura, compuesta por los *gateways* y las antenas que los nodos necesitan para comunicarse y la segunda una aplicación de administración, desde la cual se puedan registrar los nodos que hacen parte de esta aplicación, se pueda definir la estructura de la trama que remiten los nodos para poder decodificar la información y reescribirla en una codificación más simple de interpretar por un usuario final.

2.12.1 Tipos de dispositivos

Cuando se genera un envío de paquetes en una red LoRaWAN, si estos son emitidos desde los nodos hacia el Gateway se denominan mensajes ascendentes (Uplink) si los paquetes son enviados desde el Gateway hacia los nodos finales entonces los mensajes son descendentes (Downlink). Los nodos de la red pueden ser clasificados en 3 tipos según su configuración de comunicación.

Clase A:

Los dispositivos soportan comunicación bidireccional entre el dispositivo final y el *gateway* y pueden enviar mensajes deliberadamente luego el dispositivo entra en estado de recepción o escucha durante 1 o 2 segundos en los que espera confirmación del servidor, si el servidor no responde en este tiempo, aún tendrá una opción de confirmar la recepción del mensaje cuando se ejecute la próxima comunicación ascendente. Con esta configuración los dispositivos obtienen un mejor rendimiento de batería.

Clase B:

Los dispositivos de clase B establecen una ventanas de recepción con base a los tiempos que son establecidos por el *gateway*, se debe tener en cuenta que si los periodos de escucha configurados son largos esto impactará directamente en el rendimiento de la batería.

Clase C:

Los dispositivos se encuentran continuamente en modo de recepción y solamente cambian a modo de transmisión durante unas fracciones de segundo cuando deben emitir los datos, este tipo de dispositivos necesitan una fuente de alimentación externa.

2.12.2 Autenticación

El protocolo LoRaWAN define 2 mecanismos de autenticación como herramienta para asegurar las aplicaciones, evitando que sean vulnerables ante ataques de terceros.

Activación en el aire OTAA

Este método de autenticación usa una clave y una firma, cada nodo dispone de una clave de aplicación única de 128-bits AppKey, esta clave se usa para generar una petición de conexión, esta petición no está encriptada sino firmada con la clave de aplicación, la petición también contiene valores únicos para el dispositivo DevEUI y para la aplicación AppEUI y un valor generado de forma aleatoria llamado DevNonce, el AppKey es usado para crear un código de integridad de mensaje (MIC), cuando el mensaje llega a su destino el servidor usa la clave para validar la integridad del mensaje, en ese caso el servidor genera 2 nuevas claves de 128-bits que serán usadas para enviar los demás mensajes. Por lo que se necesita un mensaje de confirmación para transmitir las 2 claves generadas (Gresak & Voznak, 2020).

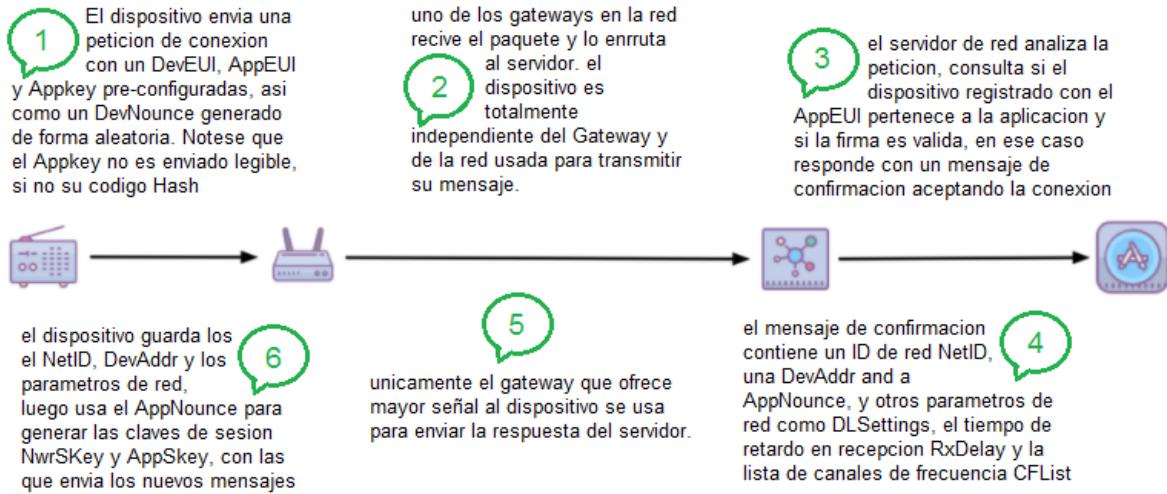


Figura 2-5 Activación en el aire OTAA. Traducida de (Raftery, 2018)

Autenticación por personalización ABP

En este método de autenticación, las claves de sesión (NwkSKey y AppSkey) junto con la dirección del dispositivo (DevAddr) están definidas en el dispositivo, estas claves se codifican directamente en el mensaje que se transmite y se enrutan por medio del Gateway que reciba la señal al servidor de IoT, en donde se verifican si las claves y la dirección recibida corresponden a uno de los dispositivos que pertenecen a la aplicación; en ese caso se procesan, además se puede retornar un mensaje de confirmación, en caso contrario se omite el mensaje recibido (Gresak & Voznak, 2020).

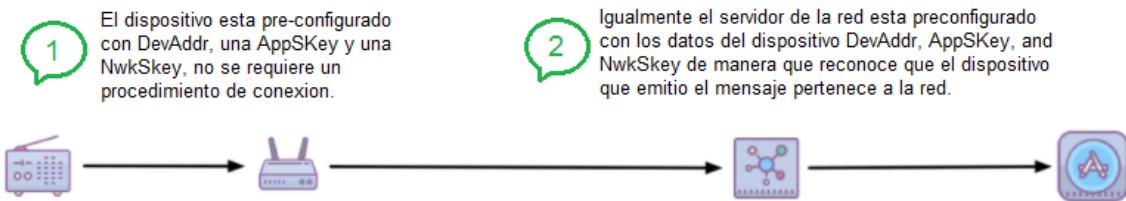


Figura 2-6 Autenticación por personalización ABP, Traducida de (Raftery, 2018).

2.13 Bandas ISM

Las bandas con licencia son de uso exclusivo de compañías y organizaciones que las usan para fines específicos pero que deben pagar una tarifa por ese derecho, mientras que las bandas libres son gratuitas y cualquier persona puede usarlas cumpliendo cierta normatividad.

Las bandas ISM son frecuencias del espectro radioeléctrico que están reservadas para el uso en aplicaciones de tipo industrial, científica y médica o doméstica; en general aplicaciones diferentes a las telecomunicaciones que ya tienen una asignación llamada IMT³ en dicho espectro y tanto las bandas ISM como las IMT tienen una estandarización internacional definida por el Reglamento de Radiocomunicaciones de la Unión Internacional de Telecomunicaciones (ITU, por su sigla en inglés).

La directora de la Agencia Nacional del Espectro (ANE) resaltó la importancia de la apertura de bandas de uso libre en cada rango de frecuencias afirmando que “*son un motor para la innovación ya que permiten crear desarrollos tecnológicos, sin la necesidad de pagar derechos por el uso del espectro.*” La resolución 711 de 2016 de la ANE atribuyen unas bandas de frecuencia para redes inalámbricas de área local que utilicen tecnologías de espectro ensanchado y modulación digital, de banda ancha y baja potencia. Los rangos de frecuencias aparecen en la siguiente tabla las cuales coinciden con las bandas definidas por la ITU.

Límite inferior (MHz)	Límite superior (MHz)
6.765	6.795
13.553	13.567
26.957	27.283
40.66	40.7
902	928
2400	2483.5
5725	5875
24000	24250
61000	61500
122000	123000
244000	246000

Tabla 2-1 Rangos de frecuencias autorizados para Banda ISM internacionalmente

Las operaciones sin licencia generalmente tienen permiso para usar estas bandas, pero debe ser tolerante a la interferencia por lo que la resolución establece que los dispositivos que operen en estas frecuencias deben aceptar la interferencia resultante de las demás aplicaciones que usen las mismas frecuencias. Por ejemplo, los hornos

³ International Mobile telecommunications

microondas que usan emisiones poderosas que pueden crear interferencia que puede interrumpir otras comunicaciones de otros usuarios en la misma banda.

2.14 Protocolos de comunicación nivel de aplicación.

Los protocolos de comunicación son estrategias que buscan que la comunicación entre las partes que componen un sistema sea eficiente, para ello la información debe ser clara, accesible, ordenada y que todas las partes del sistema puedan entenderla. De forma análoga en la comunicación humana cuando interactúan dos individuos fácilmente puede establecer una secuencia de emisión y recepción alternando el tiempo de locución para establecer un diálogo, pero cuando existen múltiples individuos es necesario delegar la palabra a uno solo de ellos con el propósito de evitar el ruido y la confusión como en el caso de los debates.

En el caso de los sistemas computacionales, el orden de la comunicación es fundamental para garantizar que las partes no solo reciben los mensajes que se les envían, sino además que los mensajes que reciben son solo los que les interesan y que tienen una secuencia lógica para evitar una mala interpretación de los mensajes y evitar el caos. A nivel de máquinas la importancia de los protocolos de comunicación es la interoperabilidad, que es la capacidad de comunicarse e intercambiar información para cooperar aunque estas no estén codificadas en el mismo lenguaje o estén programadas con la misma estructura. A continuación se abordarán tres de los protocolos más populares en el ámbito de IoT.

2.14.1 Protocolo avanzado para encolamiento de mensajes (AMQP)

Este protocolo centraliza su cadena en mensajes en un agente llamado bróker, que se encarga de realizar una retransmisión coherente de los datos, para ello crea una secuencia de mensajes permitiendo generar una operación asíncrona, esto quiere decir que el emisor y el receptor reciben los mensajes aunque estos no estén trabajando al mismo tiempo, de esta manera el destinatario pregunta por la información cuando tenga disponibilidad de hacerlo y el emisor puede continuar trabajando sin esperar que exista confirmación del receptor.

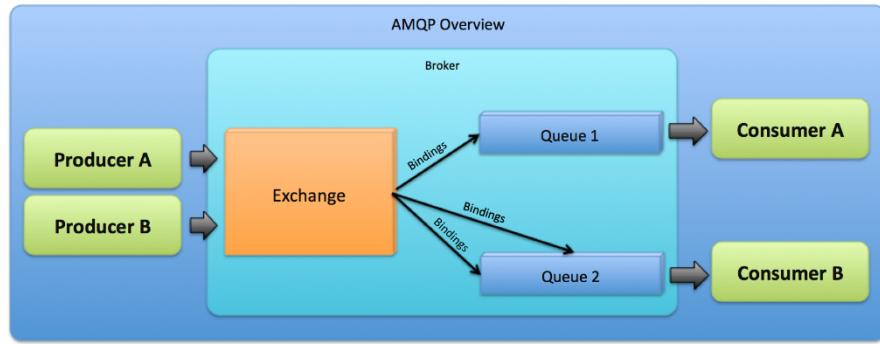


Figura 2-7 Secuencia de comunicación AMQP, tomada de (VOLOV, 2016).

En AMQP la emisión de un mensaje desencadena la secuencia de operaciones, luego de que el productor envía un mensaje, el bróker es el encargado de procesarlo, categorizarlo según la cola a la que pertenezca y distribuirlo atendiendo a las reglas que hayan sido definidas, el otro actor en esta comunicación es el consumidor quien se encarga de consultar si tiene mensajes pendientes por recibir, para ellos notifica al bróker cuales son las colas de su interés y el código de identificación del último mensaje que recibió. A lo cual el bróker responde con una lista secuencial de mensajes que se han generado luego del último mensaje recibido por el consumidor para cada una de las colas de su interés (IONOS, 2020).

2.14.2 Transporte de telemetría por cola de mensajes (MQTT)

Este es un protocolo de mensajería estándar para IoT, diseñado como un sistema de publicación/ suscripción para el transporte de mensajes que resulta ideal para conectarse a dispositivos remotos, dejando una huella diminuta y aprovechando el ancho de banda (MQTT, 2020).

Bajo este esquema, la información del sistema se centraliza en un servidor llamado Bróker, para hacer uso de los temas que son similares a las colas, los clientes envían un mensaje de conexión con la información necesaria para establecer la conexión: usuario, contraseña, identificación del cliente, a lo cual el bróker retornará un mensaje indicando si la conexión es exitosa o fallo. Una vez establecida la conexión, el cliente puede decidir si desea publicar información, para lo cual envía el contenido del mensaje y el tema al que corresponde o decide si se suscribe o revocar la suscripción a alguno de los temas que está siguiendo. Independientemente de la operación que elija el cliente siempre recibe confirmación por parte del bróker indicando el estado de la petición.

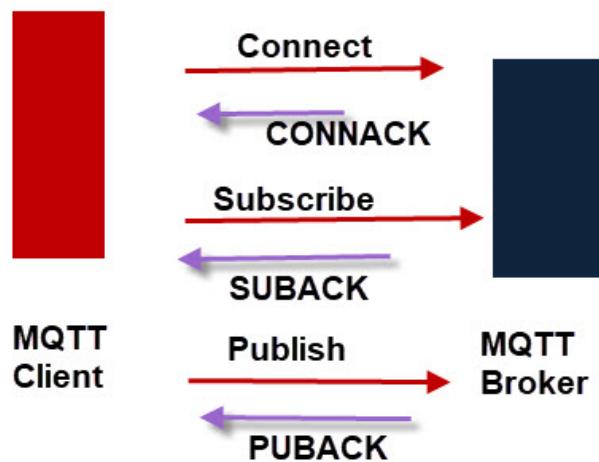


Figura 2-8 Esquema de interacción cliente bróker en MQTT, tomado de (Cope S, 2020).

El bróker MQTT usa un algoritmo llamado filtro con el cual selecciona los mensajes que son enviados a cada cliente basado en la organización jerárquica de los temas a los que los clientes se encuentran suscritos. el bróker mantiene los mensajes en la cola hasta que todos los clientes que se encuentran suscritos al tema confirmen que han recibido dicho mensaje.

A diferencia de otros protocolos MQTT mantiene la conexión entre los clientes abierta y recicla dicha conexión para el envío de nuevos mensajes evitando latencias en la entrega de mensajes, para verificar dicha conexión cada cierto periodo de tiempo los clientes envían un paquete PINGRESP que es respondido con un paquete del mismo tipo por parte del servidor, esto permite que aunque la comunicación entre las partes sea asincrónica los mensajes están disponibles muy rápidamente lo cual lo hace bastante bueno para sistemas que requieren actualización constante de datos (Hoplin, 2019).

2.14.3 Protocolo para aplicaciones con restricciones (CoAP)

Este protocolo fue diseñado para el intercambio de información en redes y dispositivos con restricciones de memoria, ancho de banda limitado baja potencia y baja disponibilidad, que se usa comúnmente en aplicaciones M2M, la analogía más inmediata para entender CoAP es pensar en una especie de HTTP para dispositivos IoT, es decir es una comunicación de tipo cliente servidor en el cual el cliente inicia una petición y espera una respuesta; sin embargo hay múltiples diferencias, la primera de ella es que este no usa TCP confirmar para el nivel de acceso a red, sino UDP que le permite el envío de mensajes asíncronos, los paquetes de CoAP son significativamente más

pequeños que los de HTTP y están compuestos por un encabezado binario, una sección compacta de opciones y el contenido del mensaje.

Modelo petición respuesta CoAP

Este modelo es el segundo nivel de abstracción del protocolo, las peticiones se emiten con un número de identificación y una etiqueta de confirmación (CON) o no confirmación (NON), de manera que si el servidor está en la capacidad de responder inmediatamente una petición etiquetada con confirmación (CON) retornará un mensaje reconocimiento (ACK) con la respuesta o el código de error generado.

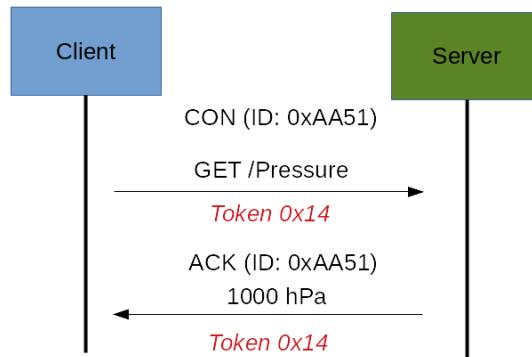


Figura 2-9 Ejemplo respuesta inmediata req/resp CoAP, tomado de (Azzola, F, 2019)

Como se observa en la imagen en CoAP un mensaje puede contener un token, que es una forma diferente de ID de mensaje que se usa para comparar la correspondencia entre la petición y la respuesta.

Si el servidor no puede dar respuesta inmediata a la petición, envía un mensaje de reconocimiento (ACK) vacío, y apila la petición en una pila de pendientes; luego cuando puede responder la petición envía un nuevo mensaje de confirmación con la información requerida, en este caso es el cliente quien debe confirmar la recepción de la respuesta.

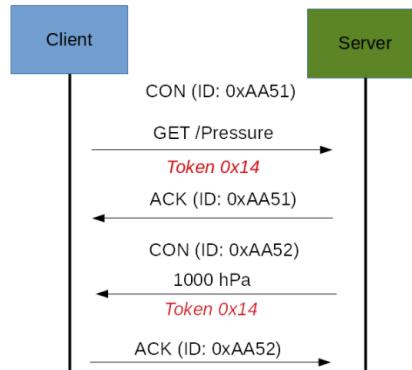


Figura 2-10 Ejemplo respuesta posterior req/resp CoAP, tomado de (Azzola, F, 2019).

2.15 Seguridad en IoT

La cantidad de dispositivos IoT que se ponen en funcionamiento cada año es abrumadora, pero este gran número de dispositivos y aplicaciones también son una oportunidad bastante atractiva para personas mal intencionadas; en 2016 se utilizaron 10 millones de dispositivos IoT secuestrados para lanzar una ataque de denegación de servicio distribuido a uno de los proveedores de DNS en estados unidos (Kshetri, 2017), hasta ese momento los mecanismos de autorización y autenticación resultaban ser una solución funcional para evitar estas amenazas, pero con la cantidad de dispositivos en aumento nuevos ataques se han ido desarrollando y estos mecanismos ya no son suficientes para proteger los sistemas. Otro componente que resulta comprometido por los ataques es la infraestructura, los ataques pueden alterar la configuración de los equipos y sacar a nodos de la red comprometiendo el funcionamiento de los demás dispositivos y de las aplicaciones (Shakdhe, Agrawal & Yang, 2019).

El proyecto abierto de seguridad en aplicaciones web, OWASP por su sigla en inglés, pertenece a una fundación homónima sin ánimo de lucro que trabaja para concebir, desarrollar, adquirir operar y mantener aplicaciones confiables. OWASP cuenta con una enorme comunidad de contribuyentes alrededor del mundo. Quienes lideran proyectos educativos y empresariales, para combatir los ataques y delitos informáticos por medio de la implementación de buenas prácticas, además ofrecen educación y entrenamiento en temas de seguridad así como recursos y herramientas para hacer que las aplicaciones sean seguras para todas las personas (OWASP).

Dentro de sus lineamientos y directivas OWASP ha publicado dos guías enfocadas a aspectos principales en el aseguramiento de sistemas IoT y una guía con los ataques más frecuentes. OWASP Internet of Things y OWASP IoT Analytics 4 Industry 4.0 donde se aborda un listado con los 10 problemas más frecuentes que deben ser atendidos para la creación de aplicaciones IoT confiables. La lista fue recuperada de (Smith, 2020):

1. **Problema de contraseñas débiles o predecibles:** el uso de mecanismos de fuerza bruta, contraseñas públicamente disponibles, credenciales invariables además de ingresos por puerta trasera en los sistemas o software de los clientes permiten el acceso y manipulación de los sistemas.
2. **Problema servicios de red inseguros:** servicios innecesarios e inseguros ejecutándose sobre el dispositivo mismo, especialmente aquellos expuestos a internet, que comprometen la confidencialidad, integridad, autenticidad o disponibilidad de la información o habilitan en control remoto no autorizado.
3. **Problema interfaces de ecosistemas inseguros:** interfaces del API de *backend*, en la nube, móviles del ecosistema externo que comprometen los dispositivos y los componentes relacionados. Inconvenientes comunes incluyen carencia de autenticación, autorización, falta de encriptación y falta de datos de entrada y filtrado.

4. **Problema carencias de mecanismos de actualización de seguridad:** la escasa capacidad de actualizar el dispositivo de forma segura, a causa de falta de la validación del firmware del dispositivo, el tránsito de las actualizaciones de entrega comúnmente no está encriptado, falta de mecanismos *anti-rollback* y carencia de notificaciones de cambios por efecto de las actualizaciones.
5. **Problema Uso de componentes desactualizados e inseguros:** el uso de componentes y librerías de software obsoletas que pueden comprometer el dispositivo, abarca la personalización de plataformas de sistemas operativos inseguros y el uso de software de terceros o componentes de hardware que comprometen la cadena de suministro.
6. **Problema protección insuficiente de la privacidad:** información personal del usuario almacenada en el dispositivo y es usada en un ecosistema inseguro o impropio o sin permisos.
7. **problema almacenamiento y transferencia insegura de datos:** carencia de encriptación y de control de acceso a datos sensibles en cualquiera de los puntos dentro del ecosistema, incluidos la suscripción, la entrega, en el tránsito o durante su procesamiento.
8. **falta de administración del dispositivo:** la falta de soporte de seguridad en dispositivos que ya se encuentran desplegados en producción, incluidos el manejo de activos, administración de actualizaciones, desmantelamiento seguro, sistemas de monitoreo, y capacidades de respuesta.
9. **problema configuraciones por defecto inseguras:** dispositivos o sistemas entregados con configuraciones iniciales inseguras o la falta de opciones de configuración para restringir ciertas operaciones como las modificaciones de acceso.
10. **problema falta de fortalecimiento a nivel físico:** la falta de medidas de fortalecimiento en la capa física esto facilita a los atacantes a tener acceso a información sensible que puede ser útil para ejecutar ataques remotos en el futuro o tomar control sobre el dispositivo (Miessler et al, 2018).

3. Antecedentes

A continuación se presentan diferentes investigaciones y estudios relacionados con estrategias de separación de residuos sólidos, recolección y aprovechamiento de material recicitable y planeación de rutas de recolección, que son relevantes y sirven como referencia para el sistema que se propone.

Tradicionalmente el problema de la recolección de basuras ha sido tratado desde un enfoque determinístico. De acuerdo con (Campbell & Wilson, 2014), el enrutamiento de los camiones recolectores de residuos se ha hecho de forma periódica PVRP⁴ durante más de 40 años, es decir los vehículos recorren rutas fijas uno o varios días por semana realizando siempre el mismo recorrido por las calles. Los autores abordan varios métodos heurísticos y exactos para resolver el problema de forma eficiente, por ejemplo: el método de búsqueda Tabú que consiste en asignar inicialmente las ubicaciones a una ruta, conforme nuevas ubicaciones se agregan a la ruta, se usa una inserción heurística con base a la similitud de factores como distancia y el tipo de carga, esto determina el orden de los puntos en la ruta, cuando la ruta llega al límite de su capacidad, se agrega una nueva ruta la ubicaciones insertadas previamente se distribuyen equitativamente e inserción de nuevos puntos se asigna a una de las rutas según su correspondencia (Brandão, 2004), solo para mencionar una de las estrategias. El artículo de (Campbell & Wilson, 2014) nos permite hacernos una idea general de la complejidad del problema de enrutamiento.

Casi simultáneamente se produjo otra investigación en la que se cambió la forma de ver el problema de enrutamiento que aunque también describió las rutas sobre un grafo, los valores del recorrido no se centraban en los vértices del grafo sino en sus arcos, además dejó enunciado de forma explícita el problema de los arcos forzados, que son arcos que se deben recorrer aún cuando sobre ellos no existen ubicaciones de interés, pero que son imprescindibles para llegar a los arcos prioritarios, es decir, arcos donde sí se necesita completar una tarea, allí se puede ver como un arco prioritario en una ruta puede resultar ser un arco forzoso de otra ruta y cómo aprovechar mejor el recorrido de estos arcos usando técnicas como la aumentación, la inversión y la reubicación (Lopes, Plastria, Ferreira, & Santos, 2014).

⁴ Periodic Vehicle Routing Problem

El problema se complica un poco más si se tiene en cuenta que se deben optimizar los recorridos para una flota entera, además de las restricciones que existen por ejemplo las ventanas de tiempo, para dichas restricciones (Babaee Tirkolaee, Abbasian, Soltani, & Ghaffarian, 2019) propusieron un algoritmo de simulación por ablandamiento térmico (*annealing*), su complejidad computacional resultó baja permitiendo una rápida convergencia y usa pasos de escalamiento para evitar quedar atrapado en óptimos locales. El algoritmo fue puesto a prueba con casos simples y casos complejos mostrando un desempeño superior comparado con sistemas para la resolución de problemas de optimización lineal como CPLEX de IBM, dando soluciones en el rango de los 77 segundos para problemas en los que CPLEX tardaba más de una hora en resolver, luego fue puesto a prueba como caso de estudio en la ciudad de Sanandaj en Irán, sobre un distrito de 330 km² con 43 nodos que debían ser recolectados, la solución del algoritmo dio el resultado que se muestra en la figura 3-1.

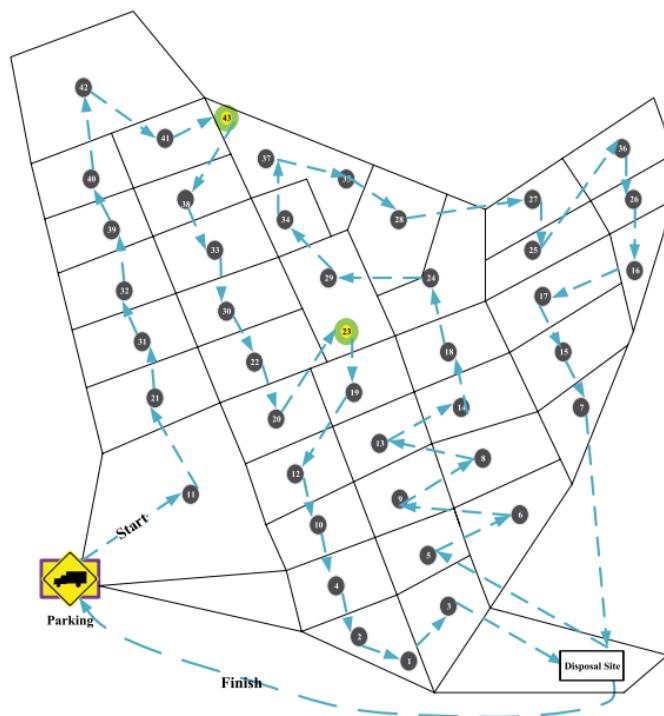


Figura 3-1 Solución generada por ablandamiento térmico en la ciudad de Sanandaj –Irán, tomada de (Babaee Tirkolaee et al., 2019).

Las ventanas de tiempo son relevantes en los esquemas de recolección de residuos, pues comúnmente estas operaciones se realizan en la noche, aprovechando el bajo tráfico vehicular, durante este periodo se deben atender de forma oportuna los contenedores, una alternativa de enrutamiento consiste en un algoritmo memético⁵

⁵ Memetico se refiere al estudio de la información y cultura desde un enfoque evolutivo, como el que propone Darwin en el que la información se transmite. La memética describe el proceso de

(Bustos Coral, Oliveira Santos, Toledo, & Fernando Niño, 2018), el cual ejecuta una fase análisis de agrupamiento (clustering) según la distancia espacial de los lugares objeto de la ruta, y luego una secuenciación de los lugares de cada clúster. La función objetivo es minimizar el tiempo total de viaje de los vehículos para atender a los clientes, en los que se usan 2 valores alfa-1 y alfa-2 con los que se penaliza el incumplimiento de restricciones a las franjas de tiempo y capacidades de carga de los vehículos, las fases siguiente del estudio incluyen la búsqueda de la población inicial, una fase de intercambio (*crossover*) y búsqueda local en la cual se realiza procedimientos exhaustivos de búsqueda intra e inter vecindarios y de variables.

La idea del algoritmo es encontrar una población inicial; es decir un conjunto de grupos, luego encontrar la mejor solución para ese conjunto y luego iterar generando nuevas poblaciones y encontrando la mejor solución para cada una de las nuevas poblaciones, en caso de encontrar una mejor solución a la inicial se guarda como solución, luego de iterar hasta que el número máximo de generaciones se alcance, se retorna la mejor solución hallada.

El algoritmo fue implementado en Kotlin y validado usando el conjunto de datos de muestra “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints,” Operations Research (INFORMS) el cual incluye 56 instancias del VRPTW con 100 clientes, el algoritmo fue capaz de encontrar la mejor solución reportada para 27 de las 56 instancias y la brecha general frente a las demás soluciones halladas por el MA es menor o igual al 1.0 % comparado con la mejor solución conocida, con un nivel de confianza del 95% lo cual demuestro una alta eficiencia y confiabilidad del Algoritmo para la resolución del VRPTW.

Es intuitivo pensar que una buena gestión de los residuos trae beneficios a nivel ambiental y social, pero también es importante considerar el valor económico que se puede extraer de los residuos, de allí la importancia de separarlos correctamente y darles una prioridad según su valor , así lo exponen (Jatinkumar Shah, Anagnostopoulos, Zaslavsky, & Behdad, 2018) quienes argumentan cómo los residuos electrónicos y aparatos del hogar contienen grandes cantidades de metales, plásticos y otros materiales que son potencialmente reutilizables, que disminuyen el costo de producción de nuevos artículos. Por esto es importante priorizar la recolección del residuo en función de su valor y su utilidad.

Para ello, (Jatinkumar Shah et al., 2018) propusieron un modelo de optimización estocástico basado en oportunidad y restricción, considerando el valor intrínseco del residuo recuperado. El modelo tiene 2 partes: la primera determina la asignación de recursos que deben ser trasladados a unidades de separación según el vecindario en el

como una idea puede propagarse exitosamente, sin que esto implique que se trate de un concepto basado en hechos o pruebas (Kantorovich, 2013)

que se encuentren. El segundo determina la cantidad de residuo recolectado de los depósitos y transferido a centros de recuperación por camiones de alta capacidad, el método de oportunidad y restricción se usó para convertir el modelo estocástico a un modelo de programación lineal. Maximizando así las utilidades económicas durante cada operación de recolección.

Así mismo, el desperdicio húmedo puede ser potencialmente aprovechado para la producción de fertilizantes orgánicos y la producción de biogás esto permite por una parte hacer una disposición del residuo y simultáneamente la generación de energías limpias como lo sustentan (Manglorkar, Sharma, Verma, & Rane, 2019), a través de la implementación de un sistema de monitoreo que usa una placa arduino uno y un método para el transporte y la separación del residuo orgánico.

3.1 Algoritmos genéticos y de enjambre

Además de las ventanas de tiempo se debe considerar la separación de residuos, para lo cual se han propuesto 2 alternativas: la primera usando rutas independientes dedicadas a recolectar exclusivamente el material recicitable y rutas dedicadas a recolectar el residuo sólido, lo cual generalmente aumenta las distancias y tiempo de trabajo. La segunda es usar camiones con compartimentos dobles pero independientes, donde se puedan separar los 2 tipos de residuo haciendo un solo recorrido.

Atacando problema desde el segundo enfoque la investigación propuesta por (Abdulkader, Gajpal, & Elmekkawy, 2015) usando una enfoque biológico combina un algoritmo de búsqueda local con un algoritmo inspirado en una colonia de hormigas, para el enrutamiento de vehículos dotados con compartimientos múltiples para transportar diversos tipos de carga, los investigadores hicieron validaciones numéricas para evaluar el desempeño del algoritmo y pudieron encontrar una solución para casi todas las pruebas de referencia, demostrando que el algoritmos basado en colonia de hormigas producen rutas más eficientes en menor tiempo que los métodos numéricos.

Los algoritmos genéticos son métodos que han demostrado un buen desempeño para resolver VRP, así lo demuestran (Fujdiak, Masek, Mlynek, Misurc, & Olshannikova, 2016) para esta investigación la información estaba estructurada en archivos Gráficos de vectores escalables SVG, los cuales contenían las coordenadas de los contenedores, esta información se transformó en grafos, cuyos vértices eran las ubicaciones de los contenedores y las aristas correspondían a las calles entre dichos vértices, cada arista tiene asociada un costo, el grafo resultante se escribía en un archivo XML sobre el cual el algoritmo ejecuta las operaciones como se puede observar en la figura 3-2.

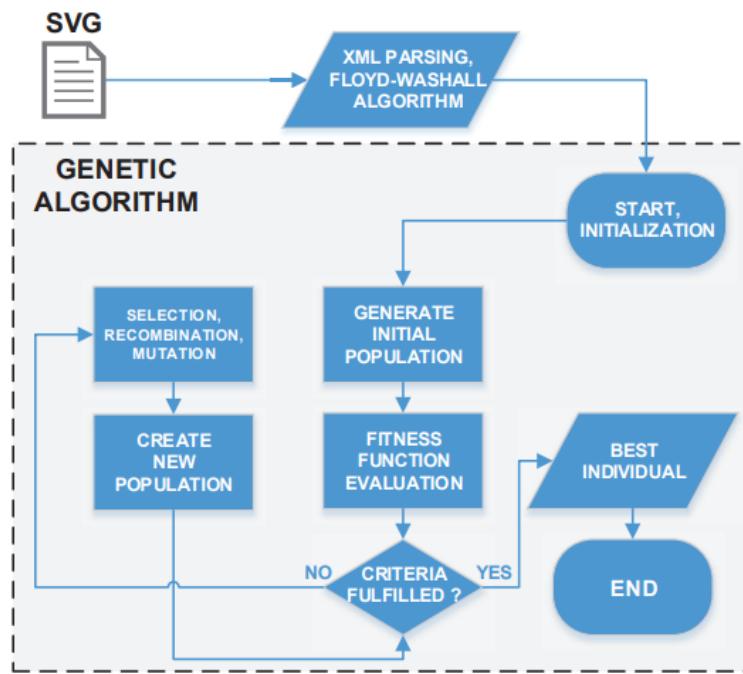


Figura 3-2 Diagrama de flujo general del algoritmo genético, tomado de (Fujdiak et al., 2016).

De la aplicación de este algoritmo sobre un conjunto de datos se pudo determinar que usando algoritmos genéticos es posible reducir los costos de la recolección en promedio un 15%, no obstante es necesario encontrar unos parámetros de población y recombinación, que permitan la implementación de estos algoritmos en dispositivos de bajo consumo.

Luego, en (Kuo & Zulvia, 2017) introdujeron el concepto de algoritmo híbrido en el cual se combinan técnicas bio-inspiradas como la búsqueda de colonia de hormigas y los algoritmos genéticos. Un modelo matemático describe el problema de recolección de RSU cuya función principal busca minimizar la distancia de los recorridos de los vehículos, sujeto a restricciones como no exceder la capacidad de carga del vehículo y visitar cada contenedor exactamente una vez.

Para este algoritmo bioinspirado se tienen en cuenta elementos como la búsqueda inicial de rutas usando el algoritmo de Prim, se inicializan los valores de la feromonía para los caminos encontrados, para cada solución encontrada se actualiza la cantidad de feromonía según la cantidad de hormigas que eligieron el camino y para el algoritmo genético cada uno de las soluciones encontradas por las hormigas se transforma en un cromosoma, pero se realiza recombinación de los cromosomas que corresponde a soluciones con mejor puntuación de feromonía, para mutación del cromosoma se realiza intercambiando el 30% de los bits del cromosoma, se encuentra la mejor solución global

y si cumple las restricciones se selecciona, de lo contrario se repite el ciclo desde el paso en el que los hormigas encuentran nuevos caminos.

El estudio se validó con información de la recolección de residuos reunida en enero de 2010 para una ciudad en Indonesia, para la aplicación del algoritmo se requirió analizar 5 secciones cada una de los cuales representa uno de los distritos de la ciudad, para la comparación se analizó el desempeño del algoritmo inspirado en colonia de hormigas, el algoritmo genético y el algoritmo híbrido, aunque el algoritmo híbrido toma mayor tiempo que el genético para converger a una solución, el algoritmo híbrido fue sustancialmente mejor con respecto al desempeño pues el genético tiende a quedar atrapado en óptimos locales. Los cromosomas representan las secuencias de ruta, si se altera el orden de los puntos visitados por las hormigas es posible encontrar trayectos que reducen la distancia y mejoran la solución general.

3.2 Soluciones basadas en IoT

Hasta este punto los algoritmos analizados usan información estática y las rutas calculadas se mantienen constantes en el tiempo, sin embargo el problema de la generación de RSU es muy variable y difícil de predecir por lo que algunos esfuerzos por hacer estimaciones más precisas del volumen de residuo han dado lugar a la instalación de dispositivos IoT, particularmente el uso de sensores remotos uno de los primeros estudios de este campo fue reportado por (N. S. Kumar, Vuayalakshmi, Prarthana, & Shankar, 2017) en un sistema compuesto por dos elementos: un sistema embebido ubicado en los contenedores de residuo y un servidor centralizado que se comunica con una base de datos y un servidor web. Mientras que el sistema embebido se ensambla sobre una placa Arduino Uno, con el cual se acciona un sensor de proximidad (HC-SR04) que opera con ultrasonido y una etiqueta de radio frecuencia (RFID), este sistema comunica la información recuperada por el sensor a través de un módulo GPRS que envía los datos al servidor central, con esto se logra informar a la municipalidad cuáles son los contenedores que están próximos a completar su capacidad de almacenamiento, para que se gestione su limpieza, así evitar la saturación y desbordamiento que generan múltiples problemas. Cuando la operación de limpieza se realiza se usa una etiqueta RFID como proceso de verificación para confirmar al servidor que el procedimiento se ha completado. Esta visión de cómo implementar sistemas inteligentes aprovechando el potencial de IoT, marcó el inicio de una gran número de publicaciones relacionadas,

GPRS no resulta ser una tecnología de transmisión muy eficiente, especialmente si se consideran las limitaciones energéticas de los sistemas embebidos; ante dicha incompatibilidad surgieron nuevas investigaciones como la de (Karthikeyan, Rani, Sridevi & Bhuvaneswari, 2017) quienes definieron una arquitectura IoT basada en una red ZigBee como sistema para la administración de residuos. El sistema se compone de tres secciones la primera encargada de la recolección de datos, la segunda encargada del procesamiento y la última correspondiente a la notificación. La parte que más interesa en

el presente trabajo es la primera luego de monitorear el nivel de llenado, la información se emite de forma inalámbrica usando ZigBee, la información es recibida en nodos coordinadores que se serializan dicha información y la transmiten a una Raspberry PI la cual no solo actúa como puerta de enlace, si no que además ejecuta scripts que repiten la información a un bróker MQTT usando internet, toda la información se encola en un mismo tema, el servidor se suscribe a ese tema y conforme nuevos mensajes llegan a la cola del tema el servidor es notificado con dichos mensajes, luego de este proceso inicia la etapa de procesamiento de datos. Los resultados del estudio son muy útiles porque muestran la confiabilidad de la trasmisión conforme aumenta la distancia como lo muestra la tabla 3-1.

Distancia (m)	potencia de la señal (dB)	Número de paquetes recibidos (por cada 100)
10	-35	99
25	-48	99
50	-51	99
65	-54	99
80	-55	97
100	-87	22
120	-92	13

Tabla 3-1 Pruebas de alcance y efectividad de trasmisión con ZigBee, tomada y traducida de (Karthikeyan, Rani, Sridevi & Bhuvaneswari, 2017)

Esto deja explícito el poder de ZigBee para transmisión de datos en distancias inferiores a los 80 metros, lo cual no se ajusta a las condiciones de Bogotá donde los contenedores se ubican a distancias mayores. En este sistema las rutas calculadas se les notifican a los conductores de los vehículos, usando Telegram configurada como cliente para recibir peticiones HTTP simples.

Un trabajo similar fue publicado por (Chaudhari & Bhole, 2018), esta investigación tiene 3 diferencias principales respecto al estudio anterior: la primera de ellas que la tecnología de transmisión que usan es WiFi, la cual tiene rangos de cobertura muy cercanos a ZigBee entre 100 y 120 metros, la segunda es el servidor de IoT en este caso el sistema se comunica con un software como servicio llamado ThingsSpeak y el tercero que las ubicaciones de los contenedores son mostradas por medio de una aplicación móvil que usa los servicios de Google mapas para geo-referenciar los coordenadas de los contenedores, lo que resulta en una interfaz mucho más explícita y fácil de interpretar para los usuarios. Una solución similar pero dedicada a la administración de residuos electrónicos es presentada en (Kang, Kang, Ilankoon, & Chong, 2020)

Una vez que la información es recolectada y transmitida al servidor IoT, uno de los desafíos consiste en procesar eficientemente esa información para encontrar rutas de recolección, lo que no es trivial dado que no existe una única solución por la naturaleza dinámica de los datos. Como respuesta a esta situación (Anagnostopoulos, Zaslavsky, Medvedev, & Khoruzhnicov, 2015) proponen un método de enrutamiento simple que

consiste en seleccionar los contenedores que reportan mayor nivel de llenado usando una consulta SQL sobre una base relacional, en la cual el criterio de ordenamiento es el volumen de llenado reportado por los contenedores, de esta manera se asignan a la ruta los contenedores que reportan mayor nivel, las coordenadas de los contenedores fueron suministradas a un servicio de Google mapas que se encarga de calcular una rutas teniendo en cuenta la configuración de las calles.

Un enfoque alternativo hace uso de un método combinatorio haciendo búsqueda por vecindarios con variable general GVNS (Papalitsas, Karakostas, Andronikos, Sioutas, & Giannakis, 2018) aplicado al enrutamiento sobre sistemas de información geográficos que se actualizan usando GPS. Para el componente geográfico los nodos de recolección se insertan en la ruta de acuerdo a las características de sus vecinos más cercanos, es decir aquellos que tienen una latitud y una longitud semejante de manera que se crea una secuencia de longitudes y latitudes ascendentes y espacialmente próximas hasta completar todos los puntos de la ruta. Luego se separan ordenadamente de acuerdo a la capacidad de carga de los camiones, puede ocurrir que el último camión no se aproveche completamente pues se le asigna la carga restante que puede ser poca comparada con las cargas de los demás camiones. El algoritmo fue codificado en fortran y se validó con datos de referencia del conjunto de datos (*dataset*) ofrecido por la universidad de Heidelberg TSPLIB (Heidelberg, 1997) mostrando desempeños muy cercanos a los de las a las rutas exactas, que computacionalmente son más costosas.

La ciudad India de Vellora usa un enfoque basado en un GIS, el cual ha sido un caso de estudio por sus métodos para la optimización de la recolección y transporte de RSU, (Lella, Mandla, & Zhu, 2017) usaron Dispositivos GPS Tremble Juno, con los que crearon los vectores espaciales de la ciudad y usando herramientas como los paquetes de sobreposición de capas de ArcGIS y las imágenes satelitales ofrecidas por Google mapas, así generaron la información geográfica de centros de reciclaje y compostaje de residuos, la información de las redes de calles, caminos y ubicación de contenedores.

Luego del levantamiento de información se gestionó la reubicación de contenedores usando el método de la media central, para reducir la distancia euclíadiana entre los contenedores de RSU. Para el análisis de rutas se crearon redes usando la extensión de análisis de redes de ArcGIS, La ruta de recolección óptima para cada zona fue hallada para las zonas 1-4, en las que se redujeron las distancias de viaje agilizando la recolección de residuos. Las entidades de gobierno pudieron replicar el sistema en varios distritos de forma económica. Finalmente el compostaje producido en los centros se usó para reforestar zonas de vegetación aledañas a la ciudad de las cuales se realizó un seguimiento por medio de imágenes satelitales que muestran cómo se recuperó una importante parte de la vegetación que había sido fuertemente afectada.

De manera análoga, el uso de sistemas de información geográfica (GIS) ha sido aplicado a la optimización de rutas de recolección por (Chaudhary, Nidhi, & Rawal, 2019) y (Jwad

& Hasson, 2018), quienes usan la funcionalidad de análisis de red de ArcGIS⁶ para determinar la zonificación y enrutamiento óptimos de los camiones recolectores, en los que se consideran factores como: costo de combustible, distancia, tiempo y número de vehículos; sin embargo, bajo estos enfoques el problema es observado desde una perspectiva estacionaria, cuando en realidad las variables que afectan la producción de residuo son de naturaleza estocástica.

Investigaciones más recientes integran elementos de ML con IoT para predecir los niveles de llenado, esta investigación fue presentada por (Anh Khoa et al., 2020) aquí no solo se define la arquitectura del nodo de medición usando LoRa para transmisión de datos, concretamente el módulo E32 que es mucho más potente pues alcanza cobertura de hasta de 2 Km, sino que se presenta un método de ML que usa probabilidad para predecir los volúmenes de recolección por zonas usando datos históricos y un modelo de regresión logística que aplica la función sigmoidea en el que S es el valor de llenado en el dominio de los reales entre (0,1)

$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s)$$

Ecuación 3-1: ecuación para la regresión logística que predice el nivel de llenado con salida binaria, tomada de (Anh Khoa et al., 2020).

La ruta óptima se calcula usando el algoritmo de Dijkstra, lo que convierte el problema en una combinatoria, en la cual se evalúan secuencias de recolección. La exactitud del sistema fue puesta a prueba en la universidad Ton Duc Thang en Vietnam, los nodos del sistema usan componentes económicos, haciendo que la implementación del sistema sea económicamente viable a gran escala. Pero se debe considerar que al tratar el problema de enrutamiento como un problema combinatoria, se limita la capacidad de encontrar soluciones computacionalmente eficientes a medida que el problema crece, de manera que el cálculo de ruta en áreas grandes en las que existen muchos contenedores como ciudades puede resultar ineficiente.

Dentro de las ventajas de LoRa está el uso de las bandas de comunicación ISM, las cuales es abierta y no requiere la autorización de ninguna entidad de gobierno para su operación, en ese sentido puede coexistir con la infraestructura celular, Además, LoRaWAN brinda soporte a la inscripción de datos, lo cual facilita la transferencia segura de paquetes (Bharadwaj, Rego, & Chowdhury, 2017).

⁶ ArcGIS es un conjunto herramientas para la captura, gestión, análisis, modificación, diseño, colaboración e impresión sobre esquemas información geográfica

3.3 Marcos de referencia para la implementación

En Indonesia (Fatimah, Govindan, Murniningsih, & Setiawan, 2020) se estudiaron 4 de las ciudades catalogadas como ciudades inteligentes en dicho país: Yakarta, Magelang, Semarang, Yogyakarta, para ello usaron cuestionarios que fueron diligenciados por las entidades de gobierno y observaciones directas, sobre los procesos de tratamiento y gestión de residuos de cada una de las ciudades, con lo cual pudieron determinar el nivel de madurez, las carencias y necesidades comunes para poder establecer un procesamiento estandarizado eficiente. A partir de ello propusieron un marco de referencia para la implementación de sistemas de gestión de residuos asistidos por IoT. El marco de referencia analiza el proceso desde 5 dimensiones: gobierno, economía, sociedad, ambiente y tecnología, también establecen los principios básicos para que una ciudad pueda adoptar el un sistema de recolección inteligente y estructurar una economía circular de forma exitosa: dinamismo, adaptación, flexibilidad e integración.

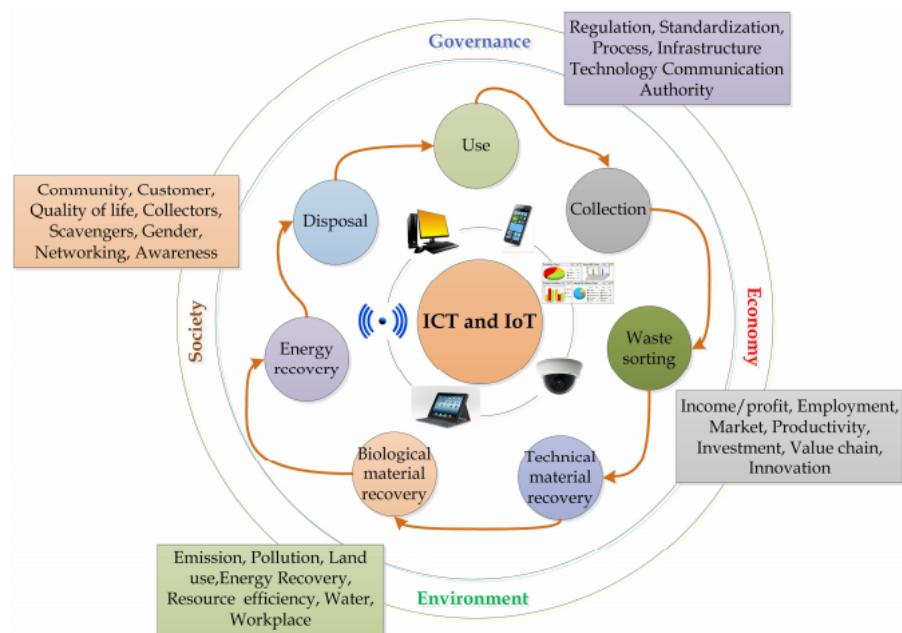


Figura 3-3 Fundamentos del marco de referencia de sistemas sostenibles e inteligentes de gestión de RSU, tomado de (Fatimah et al., 2020).

La implementación de dicho sistema impactó de forma positiva 5 aspectos de las ciudades: salud y bienestar social, saneamiento de agua potable, trabajo decente y economía creciente, consumo y producción responsable y, por último, acción contra el cambio climático.

3.4 Desafíos para la implementación

Además de los aspectos de alcance, precisión y el enrutamiento en sí mismo, un factor a considerar para poder implementar este tipo de soluciones en ambientes reales es la vida útil de la batería, dado que se requiere un mantenimiento mínimo de las unidades de medición en los lugares remotos, para ello (Kristanto, Yashiro, Koshizuka, & Sakamura, 2016) han propuesto un algoritmo de predicción basado en datos históricos con lo cual se reduce sustancialmente la transmisión de datos, pues es una de las operaciones que causa mayor consumo de energía, el algoritmo asume que cada uno de los contenedores alcanzará el límite en cierto periodo del día con esto se realiza una única medición diaria luego de dicho periodo, para evitar así múltiples mediciones y aun así mantener la información actualizada justo en el momento previo al cálculo de las rutas, el prototipo propuesto usa únicamente 3,6 mA por hora de manera que la duración de la batería podría prolongarse hasta los 33,6 años.

Otro de los aspectos a tener en cuenta para la implementación de soluciones IoT en el contexto de la ciudad inteligente es la seguridad para ello se deben garantizar por lo menos 4 principios: privacidad, confiabilidad, accesibilidad y autenticidad de manera que es preciso asegurar las interfaces de administración de aplicaciones, para enfrentar este desafío un modelamiento holístico que integra Big Data e inteligencia artificial es propuesto por (Chen, Hasan, & Mohan, 2018), bajo este esquema se crea una red de confianza distribuida entre usuarios y componentes del sistema, a cada usuario se le asigna una puntuación dependiendo de las interacciones con los otros usuarios y componentes del sistema. En este esquema la confiabilidad de un usuario está dada por la sumatoria del factor de confiabilidad que los demás usuarios del sistema reportan sobre el usuario en particular y basado en dicho puntaje el algoritmo clasifica si es posible crear un vínculo de colaboración con dicho usuario, denegar un servicio o incluso restringir su acceso a las interfaces de las aplicaciones. Esta lógica se implementa de forma transversal por medio de un servicio que analiza el tráfico de las peticiones y las puntuaciones de los usuarios con el fin de impedir que se ejecuten ataques dentro de la red que vulneren los datos de los usuarios y la integridad de los componentes del sistema.

Autores	Año	método	Enrutamiento	IoT	GIS	ML
Brandão	2004	Búsqueda Tabú	Inserción heurística	No	No	No
Abdulkader, Gajpal, & Elmekkawy	2015	colonia de hormigas	Algoritmo de enjambre	No	No	No
Anagnostopoulos, Zaslavsky, Medvedev, & Khoruzhnicov	2015	Top-K query	enrutamiento por prioridad	Si	Si	No
Fujdiak, Masek, Mlynek, Misurec, & Olshannikova	2016	algoritmos genéticos	Generación de Grafos XML	No	Si	No
		algoritmo	algoritmo biológico, genético			
Kuo & Zulvia	2017	híbrido		No	No	No
N. S. Kumar, Vuayalakshmi, Prarthana, & Shankar	2017	Monitoreo Remoto	NA	GPRS,R FID	No	No
Lella, Mandla, & Zhu	2017	vectores espaciales	análisis de redes de ArcGIS	No	Si	No
Karthikeyan, Rani, Sridevi & Bhuvaneswari	2017	Red Sensores Inalámbricos	NA	ZigBee	No	No
Jatinkumar Shah, Anagnostopoulos, Zaslavsky, & Behdad	2018	oportunidad y restricción	modelo de programación lineal	No	No	No
Papalitsas, Karakostas, Andronikos, Sioutas, & Giannakis	2018	búsqueda por vecindarios con variable general	longitudes y latitudes ascendentes	No	Si	No
Bustos, Oliveira, Toledo & Fernando Niño	2018	algoritmo memético	clúster y	No	No	No
Chaudhari & Bhole	2018	Red Sensores Inalámbricos	Google Mapas	WiFi	Si	No
Babaee Tirkolaee, Abbasian, Soltani, & Ghaffarian	2019	ablandamiento térmico	Si	No	No	No
Chaudhary, Nidhi, & Rawal	2019	Sistema IG	análisis de redes de ArcGIS	No	Si	No
Anh Khoa et al	2020	IoT basado en ML	algoritmo de Dijkstra	Si	No	Si
Fatimah, Govindan, Murniningsih, & Setiawan	2020	Marco de referencia IoT	NA	Si	No	No

Tabla 3-2 Síntesis de los métodos hallados en la revisión sistemática de literatura.

Los estudios e investigaciones revisadas nos permitieron entender varios aspectos entre ellos: que se debe considerar de la tecnología de transmisión seleccionada, la importancia de manejar información geográfica de los elementos físicos del sistema, que es preferible encontrar métodos de enrutamiento aproximados en los que no se sacrifique la eficiencia computacional, que para la presentación de rutas existen herramientas que son más fáciles de interpretar como los mapas, que tecnologías como ML nos pueden ayudar a predecir el comportamiento del sistema, que es se deben considerar los aspectos de seguridad y se debe ser cuidadoso con el uso de la energía pues es un recurso valioso ya que si se agota. los nodos requieren mantenimiento haciendo que se generen gastos adicionales y que el sistema sea ineficiente.

4. Implementación de Dispositivos de Medición

Uno de los elementos básicos en una red de sensores inalámbricos es el nodo de medición, que es un dispositivo electrónico que se debe adaptar a los contenedores que están dispuestos en las calles para almacenar el residuo sólido y con el cual se logra estimar con un alto grado de precisión el nivel de llenado en cada contenedor. Con esta información se puede determinar si es necesario que dicho contenedor sea incluido en la planificación de las próximas rutas de recolección que realiza la flota o su recolección puede ser postergada, para el dirigir el uso de los camiones a las zonas donde se produce mayor cantidad de residuo .

El dispositivo de medición propuesto se compone de 4 elementos, un sensor de distancia que se encarga de medir el nivel de llenado, un micro controlador en el que almacenan las instrucciones y procedimientos que debe realizar el dispositivo, un módulo de transmisión de radio encargado de modular y emitir la información de forma inalámbrica y una batería para alimentar el nodo. El siguiente diagrama de bloques muestra de manera abstracta la interacción de los componentes.

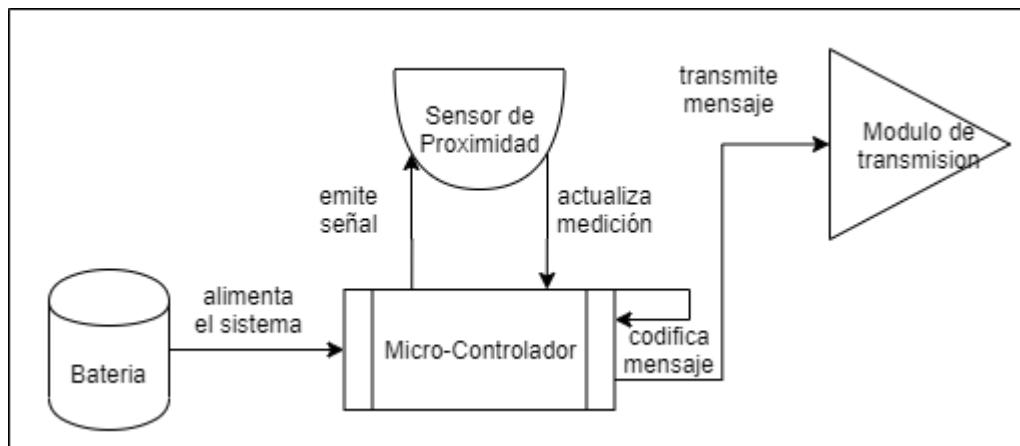


Figura 4-1 Diagrama de bloques arquitectura del dispositivo de medición.

Opcionalmente, el dispositivo puede disponer de un sensor GPS con el cual es posible determinar la ubicación del dispositivo sobre la superficie terrestre usando como referencia un sistema de coordenadas, pero analizando el hecho de que los contenedores tienden a permanecer en el mismo lugar, no se requiere actualización constante de su ubicación, y gracias a que la base de datos que gestiona la información del sistema integra un componente espacial, para el sistema sería suficiente registrar la ubicación geográfica del contenedor en el momento que se instala el dispositivo de

medición, para disponer de la geo-referenciación de los contenedores, lo cual disminuye aproximadamente en un 33% el costo de cada nodo.

A lo largo de este capítulo se abordan algunas configuraciones de estos 4 componentes y se comparan con soluciones que existen en el mercado para definir cuál de ellas es la que mejor se adapta a las condiciones del sistema que se propone haciendo un balance entre costo, rendimiento y precisión.

Sensor de distancia:

Es un dispositivo capaz de detectar la distancia hasta un obstáculo o cuerpo sólido en un ángulo dirigido a continuación se abordan 3 de los sensores disponibles en el mercado.

Sensor Laser VI53I0x-GyVI53I0xv2

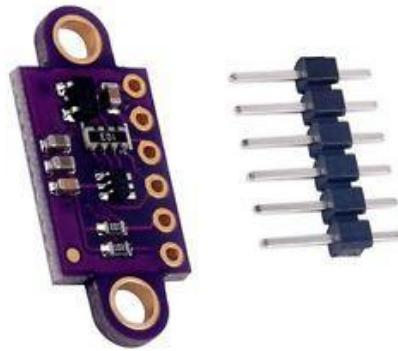
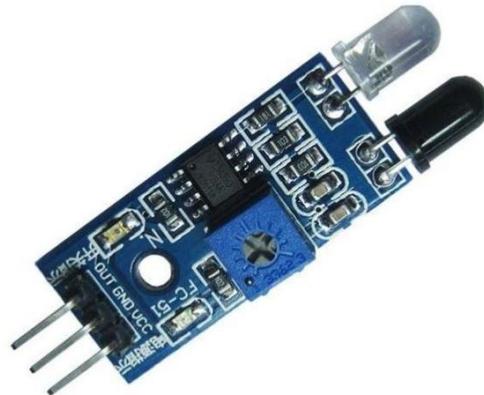


Figura 4-2 Sensor de proximidad laser GyVI53I0xv2

Este sensor emite una señal de láser⁷ y usa la reflexión para determinar la proximidad de los objetos, su alimentación es de 2,8 a 3,5 V y es un sensor de alta precisión que tienen una resolución de 1 milímetro y un rango entre 1 milímetro y 2 metros, pero su alcance efectivo depende de las condiciones ambientales como el ruido y la reflectancia del material del obstáculo, su costo aproximado es de 35.000 COP, es un circuito inter integrado I2C⁸, se considera que su estructura sin cable y sus pequeñas dimensiones hacen que sea algo difícil de configurar, para el sistema que se propone.

⁷ LASER: light amplification by stimulated emission of radiation

⁸ es un circuito que transporta datos en serie y que se usa para comunicar de forma sencilla los diferentes componentes de un circuito generalmente uno de los componentes actúa como master coordinando la transmisión de datos entre los demás componentes que se comunican por un bus de datos serial

Sensor de luz infrarroja (IR)**Figura 4-3 Sensor de proximidad infrarrojo FC-51**

Este sensor usa una señal de luz infrarroja para reflejar el objeto, el sensor consta de un bombillo led que emite la señal y un detector de monitoreo de posición, que recibe el pulso de la luz reflejada y genera una señal análoga que depende de la ubicación del objeto reflectante. Su alimentación es de 3,3 V, existen varios dispositivos en el mercado y su rango está asociado a su valor comercial, el más económico cuesta 5.000 COP y su rango está entre 2 y 30 cm, se pueden conseguir sensores con mejor alcance y calibración pero su costo es de 104.000 COP y su rango está entre 100cm y 550 cm. Estos rangos tampoco se ajustan muy bien a las dimensiones de los contenedores que se busca monitorear.

Sensor de distancia por ultrasonido:**Figura 4-4 Sensor de proximidad por ultrasonido HC-SR04.**

De manera análoga a los sensores basados en luz, este tipo de sensor usa la reflexión de la señal para medir la distancia a un obstáculo; este sensor emite un sonido en una frecuencia imperceptible para el oído humano y mide el tiempo que tarda en regresar el

sonido emitido convirtiendo el eco recibido en una señal eléctrica, para hallar la distancia se usa la constante de velocidad de propagación del sonido en el aire 334 m/s, empleando la siguiente fórmula:

$$\text{Distancia} = \frac{\text{Tiempo (s)} \times 334 \text{ m}}{1(\text{s})} \quad (1)$$

Ecuación 4-1: Medición de la distancia usando la velocidad del sonido.

La referencia más común de este sensor es HC-SR04, tiene una resolución de 1 cm y un rango entre 5 cm y 500 cm, su valor en el mercado es de 6.000 COP y dispone de 4 pines estándar que facilitan su conexión con placas de desarrollo como arduino o protoboard genéricas y existen varias librerías de código abierto que permiten que su configuración sea sencilla.

Para este trabajo se consideraron 3 configuraciones del nodo de medición en los cuales se usaron diferentes micro controladores y módulos de transmisión de radio frecuencia con el ánimo de encontrar de forma experimental cuál de estas configuraciones se adaptan mejor al sistema de monitoreo, teniendo en cuenta factores como la distancia de comunicación, la atenuación de la señal y el costo.

A manera de resumen, se extrajo la información en la siguiente tabla (tabla 4-1) para hacer un análisis más rápido de los sensores revisados.

Característica	Laser	Infrarrojo	Ultrasonido
Referencia	VI53I0x-GyVI53I0xv2	DS - IR	HC-SR04
Voltaje de entrada (V)	2,8 a 3,5	3,3	5
Resolución	1 milímetro	Depende del costo	1 cm
Precisión	Alta	NA	media
Rango (alcance)	1 milímetro hasta 2 metros	Rango está asociado a su valor comercial el más económico 2 hasta 30 cm, sin embargo otros ofrecen rangos de 1 hasta 5,5 metros.	entre 5 cm y 5 metros
Precio (COP)	\$ 35.000	el más económico cuesta \$5.000 otros de \$104.000	\$ 6.000

Tabla 4-1 Comparación sensores de proximidad.

4.1 Arduino

La página Web oficial de Arduino lo define como: una plataforma electrónica de código abierto, que tiene la capacidad de interpretar estímulos de entrada como la intensidad de luz por medio de un sensor, o la pulsación de un botón y a partir de esos estímulos desencadenar acciones, por ejemplo, encender un motor o enviar un mensaje (Arduino, 2018). All componente hardware de arduino lo llamamos tarjeta, a la tarjeta se le pueden indicar cómo operar ante determinado estímulo de entrada por medio de un conjunto de sentencias que se codifican en el micro controlador que viene integrado en la tarjeta, por medio de una interfaz de bus serial que se conecta fácilmente al computador a través del puerto USB.

Arduino no solo está compuesto por el hardware, dispone además de un lenguajes de programación de alto nivel y de un entorno de desarrollo llamado arduino estudio, basado en Processing⁹, una de las características más importantes de la plataforma es su comunidad, pues al tratarse de código abierto usuarios experimentados dan soporte a la plataforma y contribuyen desarrollado y publicado diferentes tipos de soluciones, para el uso y beneficio de la comunidad, que pueden ser instalados como librerías desde el gestor de paquetes del entorno de desarrollo. Otras de las ventajas al usar arduino son:

Económicas: las tarjetas de arduino están construidas con componentes de buena calidad y bajo costo lo que hace que su costo en el mercado esté por debajo de los 50 dólares, lo cual es económico si se compara con otras tarjetas de desarrollo disponibles en el mercado (Msv, J. 2019).

Multiplataforma: el hardware de arduino está diseñado para integrarse de forma sencilla con sistemas unix, Macintosh OSX y Windows.

Entorno de desarrollo: tiene un entorno de desarrollo muy sencillo, compuesto por una interfaz gráfica en donde se puede escribir el código y controles para configurar la tarjeta sobre la que se está desarrollando, y permite operación como compilar el código y cargarlo al microcontrolador, además dispone de un monitor para hacer seguimiento a las operaciones que se ejecutan en la tarjeta.

Software libre y extensible: las librerías y ejemplos de arduino son publicadas como código abierto de manera que todos los desarrolladores pueden abrir el código que esté en un lenguaje legible y modificarlo o ajustarlo de acuerdo a sus necesidades y el código estándar C puede ser incluido directamente en el programa de arduino.

⁹ Lenguaje de programación basado en java con una sintaxis simplificada y orientado a la programación de equipos electrónicos.

Hardware libre y extensible: todos los planos de las tarjetas arduino están publicados bajo la licencia de creative commons, para que los diseñadores de circuitos experimentados puedan hacer sus propias versiones del módulo extendiéndolas y mejorandolas, incluso usuarios sin muchas experiencia pueden revisar los planos de los circuitos y así entender mejor su funcionamiento (Arduino, 2018)-

Una de las configuraciones de dispositivo de medición se ha elegido el **arduino UNO**: es una tarjeta compuesta con un micro controlador ATMega328 tiene un conjunto de 14 pines de entrada o salida digital, 6 pines de entrada analoga y un resonador cerámico de 16 Mhz, un puerto de conexión USB, un puerto Jack para suministro de corriente, un puerto ICSP¹⁰, la tarjeta se puede conectar al computador por medio de un cable USB para empezar un prototipo de forma rápida (Arduino Uno Rev3, 2017).

Módulos de transmisión:

La compañía Semtech quien patentó la tecnología de modulación LoRa, desarrolló un módulo electrónico que recibe la señal en bits y la modula en una de las frecuencias de onda de la banda ISM predefinidas por LoRa, se trata de la familia SX-12xx (SX1276, SX1277, SX1278, SX1279) los cuales incorporan el módem de espectro ensanchado, como se había visto previamente, logrando distancias mayores a las que se consiguen con técnicas de modulación como FSK o OOK, su sensibilidad esta 8 dB por encima de FSK pero usando un amplificador de bajo costo se puede incrementar la sensibilidad del receptor en más de 20 dB, este dispositivo modula la frecuencia constantemente y con esto consigue tener un selectividad de la señal y un bloqueo de interferencia que mejora la confiabilidad de la comunicación.

Los módulos de Semtech ofrecen alta flexibilidad en la configuración permitiéndole decidir al usuario parámetros del espectro como el ancho de banda (BW), factor de propagación (SF), tasa de corrección de error (CR) (semtech, 2015)

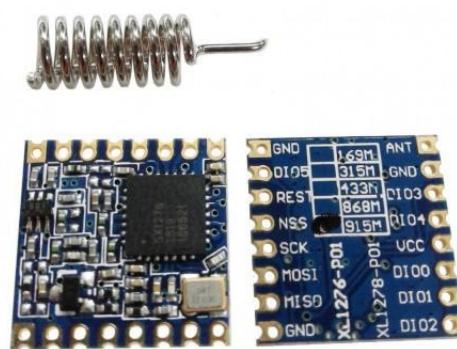


Figura 4-5 Módulo de transmisión SX1276

¹⁰ In-Circuit Serial Programming: interface que se usa para grabar desde el computador al microcontrolador un programa sin necesidad de usar el puerto USB

El módulo que se eligió para la primera configuración de dispositivos de medición es el SX1276 este módulo dispone de 14 pines dentro de los cuales se destacan los pines GND y 3,3V que son los que suministran energía al módulo, los pines de la interfaz periférica serial SPI¹¹: SCK reloj serial, MOSI salida del maestro entrada del esclavo, MISO entrada del maestro salida del esclavo, NSS selección del esclavo, el pin de reinicio Reset y los pines de entrada digital DIO0, DI1, DI2, además dispone de un espacio para la conexión de una antena. Lo particular en este módulo es que las resistencias, los diodos y los condensadores que la componen están calibrados para maximizar la señal en 915 Mhz.

El módulo es muy pequeño mide 2 cm X 2,7 cm, la distancia entre sus pines es muy poca y no se ajusta a una regleta de pines estándar lo que no solo dificulta su soldadura, sino también su montaje en instrumentos genéricos como la protoboard W-102 de manera que para poder ensamblar se requiere de adaptarla a una baquela universal. Para ensamblar el primer circuito del dispositivo de transmisión se usaron los 3 componentes mencionados en este capítulo: arduino uno, modulo SX1276 y sensor HC-SR04

Módulos de Geo localización:

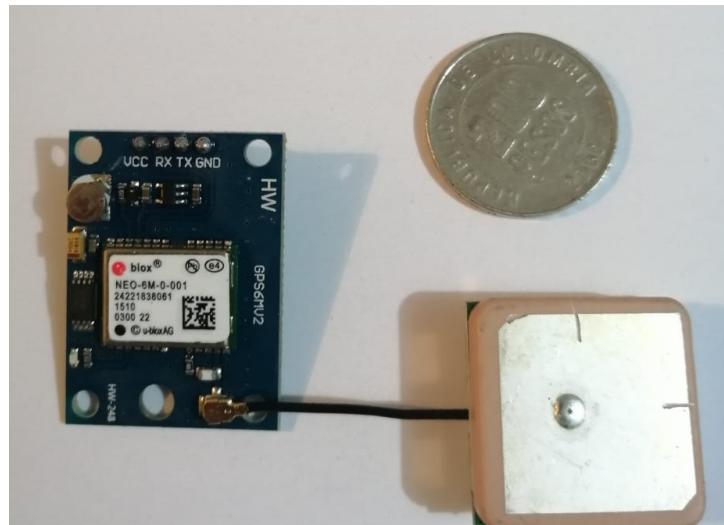


Figura 4-6 Módulo de localización Neo 6M Ublox

Este módulo utiliza tecnología Ublox de triangulación de señal satelital para detectar la ubicación espacial con un alto grado de precisión, dispone de una antena cerámica de 2,5 x 2,5 cm que genera alta ganancia, dicha antena se conecta al módulo por medio de un puerto UFL, el módulo dispone 4 pines TX, RX, VCC, GND, y tiene un voltaje de

¹¹ Especificación de la interfaz serial de comunicación síncrona usada para comunicaciones de corta distancia en sistemas embebidos

operación de 3,3 V se usa con mayor frecuencia el módulo TX, para conocer el posicionamiento solo se requiere la lectura de las sentencias NMEA¹² que genera el Chip, las cuales tienen la siguiente estructura.

\$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020620,,,A*44

Que según el protocolo NMEA ya mencionado se pueden interpretar así:

- **044235.000** representa la hora GMT (04:42:35)
- “**A**” es la indicación de que el dato de posición está fijado y es correcto. “**V**” sería no válido
- **4322.0289** representa la longitud ($43^{\circ} 22.0289'$)
- **N** representa el Norte
- **00824.5210** representa la latitud ($8^{\circ} 24.5210'$)
- **W** representa el Oeste
- **0.39** representa la velocidad en nudos
- **65.46** representa la orientación en grados
- **020620** representa la fecha (2 de Junio del 2020)

Información tomada de (Naylampmechatronics, 2016).

El módulo dispone de un bombillo led, el cual se mantiene apagado hasta que reconozca la localización, fecha y la hora, una vez encuentre esta información comenzará a prender y apagar de forma intermitente. Su velocidad de puerto serial es configurable desde los 4800 hasta los 115200 baudios, pero se recomienda operarlo en 9600 baudios.

¹² National Marine Electronics Association

4.1.1 Conexión de los componentes físicos

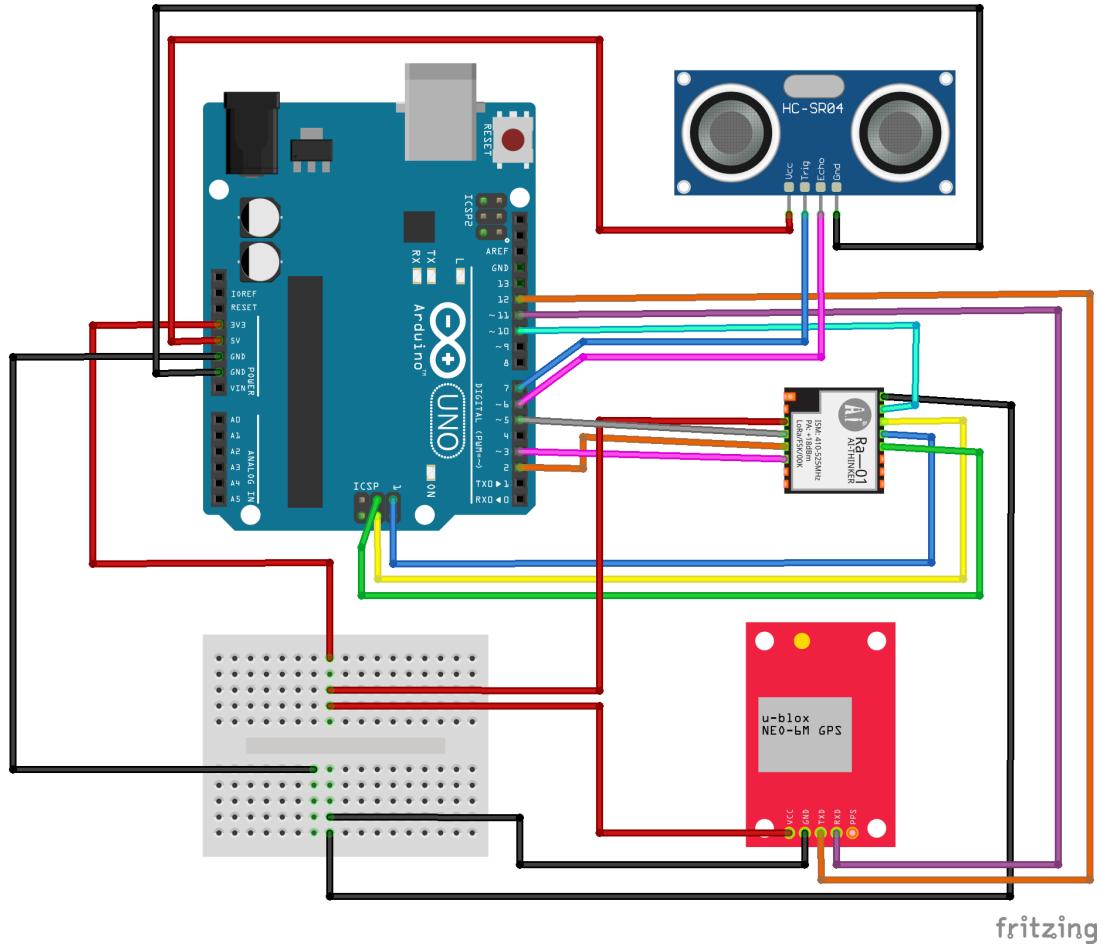


Figura 4-7 Esquema de conexión Arduino, SX1276 neo 6M y HC-SR04.

Para la conexión de los componentes del primer módulo de medición se usó la configuración que se muestra en la siguiente tabla (tabla 4-2):

Arduino	SX1276
3,3 V	3,3 V
GND	GND
D10	D2
D11	D3
MISO	MISO
MOSI	MOSI
SCK	SCK
D10	NSS
D5	Reset

Tabla 4-2 Conexiones de arduino uno a SX1276

Arduino	HC-SR04
5 V	5 V
GND	GND
D6	Echo
D7	Trigger

Tabla 4-3 Conexiones de arduino uno a HC-SR04

Arduino	Neo-6M
3,3 V	3,3 V
GND	GND
D11	RX
D12	TX

Tabla 4-4 conexiones de arduino uno a Neo-6M

Luego de haber realizado las conexiones con la configuración descrita anteriormente, se obtuvo un circuito ensamblado como se puede ver a continuación.

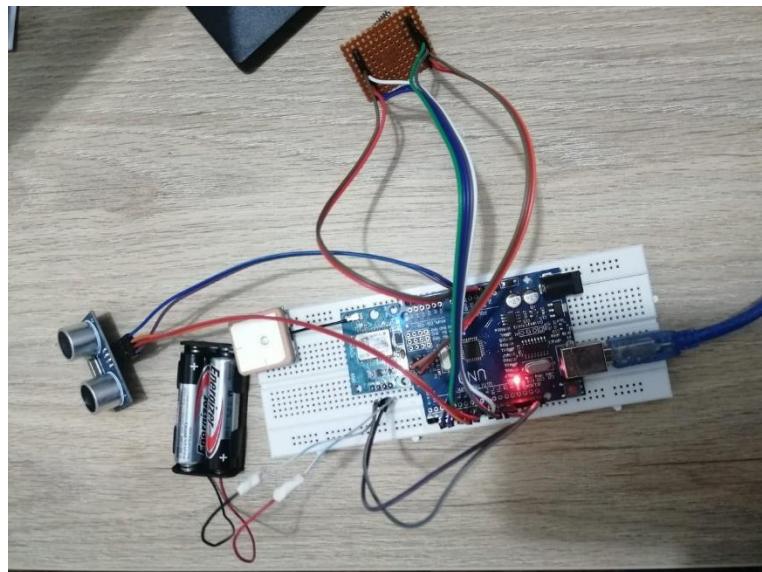


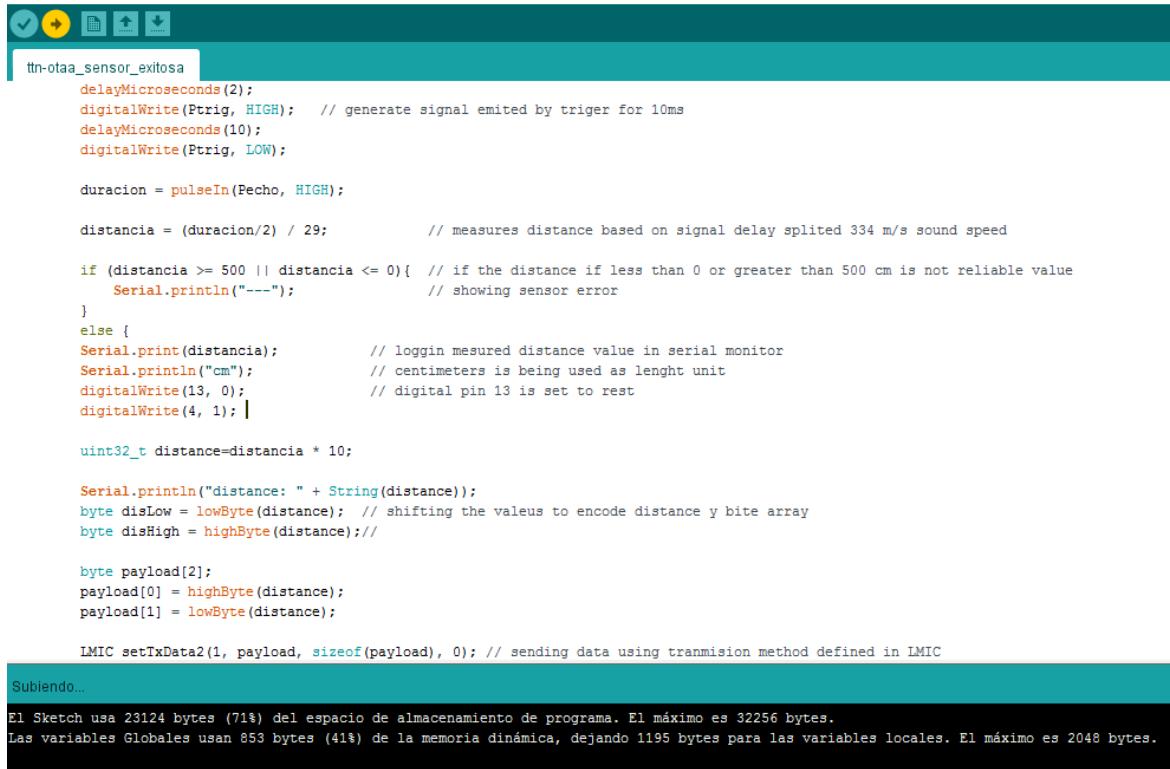
Figura 4-8 Ensamblado de los componentes físicos configuración con arduino UNO

4.1.2 Programación lógica del micro-controlador

Los programas en arduino se componen de dos secciones principales una llamada *setup*, donde se configuran las variables, librerías y objetos que se usarán en el programa, la otra llamada *loop*, donde se incluyen instrucciones que se ejecutan repetidamente mientras la placa esté conectada a una fuente de energía.

Para empezar, se importan las librerías que se instalaron LMIC y SPI. En la sección de *setup* se deben definir cuáles serán los pines de la tarjeta que usaremos para enviar o

recibir señales, para esta configuración se han definido los pines 6 como entrada y 7 como salida, que son los que están conectados al sensor de proximidad. La sección de *loop* ha sido reemplazada por una función llamada *do_send*, que será la encargada de enviar el mensaje. Esta sección contiene el código que se ve a continuación.



```

ttn-otaa_sensor_exitosa

delayMicroseconds(2);
digitalWrite(Ptrig, HIGH); // generate signal emitted by trigger for 10ms
delayMicroseconds(10);
digitalWrite(Ptrig, LOW);

duracion = pulseIn(Pecho, HIGH);

distancia = (duracion/2) / 29; // measures distance based on signal delay splitted 334 m/s sound speed

if (distancia >= 500 || distancia <= 0){ // if the distance is less than 0 or greater than 500 cm is not reliable value
    Serial.println("---");
    // showing sensor error
}
else {
    Serial.print(distancia); // logging measured distance value in serial monitor
    Serial.println("cm"); // centimeters is being used as length unit
    digitalWrite(13, 0); // digital pin 13 is set to rest
    digitalWrite(4, 1); |

    uint32_t distance=distancia * 10;

    Serial.println("distance: " + String(distance));
    byte disLow = lowByte(distance); // shifting the values to encode distance in byte array
    byte disHigh = highByte(distance);//

    byte payload[2];
    payload[0] = highByte(distance);
    payload[1] = lowByte(distance);

    LMIC_setTxData2(1, payload, sizeof(payload), 0); // sending data using transmission method defined in LMIC
}

Subiendo...
El Sketch uses 23124 bytes (71%) of program storage space. The maximum is 32256 bytes.
Global variables use 853 bytes (41%) of dynamic memory, leaving 1195 bytes for local variables. The maximum is 2048 bytes.

```

Figura 4-9 Segmento del programa en el entorno de arduino estudio.

El programa emite una señal durante 10 microsegundos y cuenta el tiempo que tarda esa señal en retornar al pin; la distancia se calcula dividiendo la duración en 2, pues esta tarda el doble, dado que debe ir hasta encontrar el obstáculo y regresar; luego este tiempo se multiplica por la velocidad del sonido. También se usó la librería TinyGPS para administrar los datos del sensor de ubicación y la librería SoftwareSerial de arduino para adaptar 2 pines como buses seriales para recibir y transmitir datos los cuales fueron el pin 11 para recepción (RX) y 12 para transmisión (TX).

La distancia obtenida es codificada en un arreglo que contiene el bit mayor y el bit menor, y las coordenadas fueron multiplicadas por 10 mil para quitar los puntos decimales y codificados con la misma estrategia en las posiciones 0 a la 5 del mismo arreglo; finalmente, este arreglo es el que se pasa como carga útil (*payload*) del paquete que será enviado usando la librería por medio de la función para asignar los datos de transición (*LMIC_setTxData2*).

Para poder almacenar el programa en el microcontrolador, se conectó la tarjeta Arduino uno al puerto USB de un computador, se inició el entorno de desarrollo y se configuró la tarjeta que se conectó, en este caso Arduino Uno, y se seleccionó el puerto por el cual estaba conectado la tarjeta.

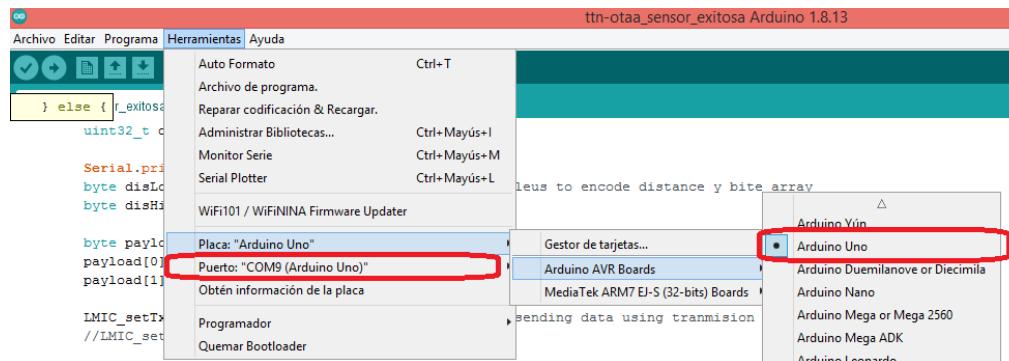


Figura 4-10 Configuración de la tarjeta en el entorno de desarrollo.

4.1.3 Pruebas de emisión de los mensajes

Asistidos por el monitor serial que provee el entorno de desarrollo como herramienta de visualización, vamos a revisar lo que ocurre dentro del micro controlador. En la siguiente imagen se pueden observar la distancia y las coordenadas geográficas obtenidas y su valor a codificar, luego se activa el módulo de radio en modo de transición y se emiten dos mensajes en la frecuencia de los 902 MHz uno con el factor de propagación (SF) en 7 y otro con este factor en 12, los paquetes quedan en cola, dado que aún no se ha activado ninguna puerta de enlace que comunique el nodo con internet, sin embargo podemos constatar que el nodo está intentando emitir los mensajes por la banda de frecuencias seleccionada, lo cual es el resultado esperado con esta prueba.

```

02:36:44.414 -> 169cm
02:36:44.414 -> distance: 1690
02:36:44.414 -> Latitud/Longitud: 4.64 , -74.09
02:36:44.448 -> Latitud/Longitud: 464448 , -7409277
02:36:44.462 -> 4490864: engineUpdate, opmode=0x808
02:36:44.550 -> 4493672: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
02:36:44.652 -> Packet queued
02:36:49.573 -> 4682174: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
02:36:47.640 -> 4685421: RXMODE SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
02:36:49.746 -> 4814105: EV_RXCOMPLETE (includes waiting for RX windows)
02:36:49.780 -> 4814151: engineUpdate, opmode=0x900
02:37:49.738 -> 121cm
02:37:49.738 -> distance: 1210
02:37:49.738 -> Latitud/Longitud: 4.64 , -74.09
02:37:49.773 -> Latitud/Longitud: 464448 , -7409277
02:37:49.807 -> 8566457: engineUpdate, opmode=0x808
02:37:49.875 -> 8569264: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
02:37:49.943 -> Packet queued
02:37:52.899 -> 8757830: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
02:37:52.967 -> 8761077: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
02:37:55.028 -> 8889761: EV_RXCOMPLETE (includes waiting for RX windows)
02:37:55.062 -> 8889807: engineUpdate, opmode=0x900
02:38:55.047 -> 26cm
02:38:55.047 -> distance: 260
02:38:55.047 -> Latitud/Longitud: 4.64 , -74.09
02:38:55.082 -> Latitud/Longitud: 464448 , -7409277
02:38:55.116 -> 12641634: engineUpdate, opmode=0x908
02:38:55.165 -> 12644507: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
02:38:55.253 -> Packet queued
02:38:58.174 -> 12833010: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
02:38:58.242 -> 12836192: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
02:39:00.353 -> 12964686: EV_RXCOMPLETE (includes waiting for RX windows)
02:39:00.387 -> 12964734: engineUpdate, opmode=0x900

```

Autoscroll Mostrar marca temporal Nueva línea 9600 baudio Limpiar salida

Figura 4-11 Verificación de las operaciones en el micro controlador desde el monitor serial en Arduino.

4.2 Dragino

Dragino es uno de los fabricantes de hardware mejor posicionado en transceptores con modulación LoRa, Dragino ofrece productos ensamblados que eventualmente podrían ser adaptados para la aplicación que se está proponiendo, por ejemplo: el sensor de distancia LDDS75 LoRaWan, que integra un sensor de ultrasonido y un módulo LoRa preconfigurados, además dispone de una batería de 4000 mA en litio pensada para durar hasta 10 años (*Dragino Distance*, 2020) o el dispositivo de seguimiento LGT-92, que integra un microcontrolador de bajo consumo, un módulo de posicionamiento L70, un acelerómetro de 9 ejes para identificar los tipos de movimientos que realizan los objetos todo esto alimentado por una batería de recargable 1000 mA diseñado para la implementación aplicaciones de cadena de suministro o control de tráfico en ciudades inteligentes. (*Dragino GPS*, 2019)

Lamentablemente, Dragino no ofrece un equipo que integre todos los sensores que se propusieron en la arquitectura del nodo de mediciones para este sistema; de manera que es posible simplemente comprar una solución ensamblada (*on the shell*), sin embargo Dragino ofrece equipos que se pueden acoplar fácilmente con placas de desarrollo estándar y facilitan la construcción de estas soluciones.

Dragino Shield: es una placa en la que viene integrada con un módulo SX12XX, un amplificador de 20 dB y una antena la función de la placa es la de hacer un puente entre los pines del módulo y los pines de arduino, la placa es compatible con las versiones: UNO, Mega, Leonardo y DUE, sobre esta placa será posible integrar los componentes definidos en la arquitectura para nuestro dispositivo de medición.

En este experimento se decidió usar un arduino Mega que dispone de un microprocesador más potente que el arduino uno y con un poco más de memoria la cual será necesaria para grabar las instrucciones que usaremos

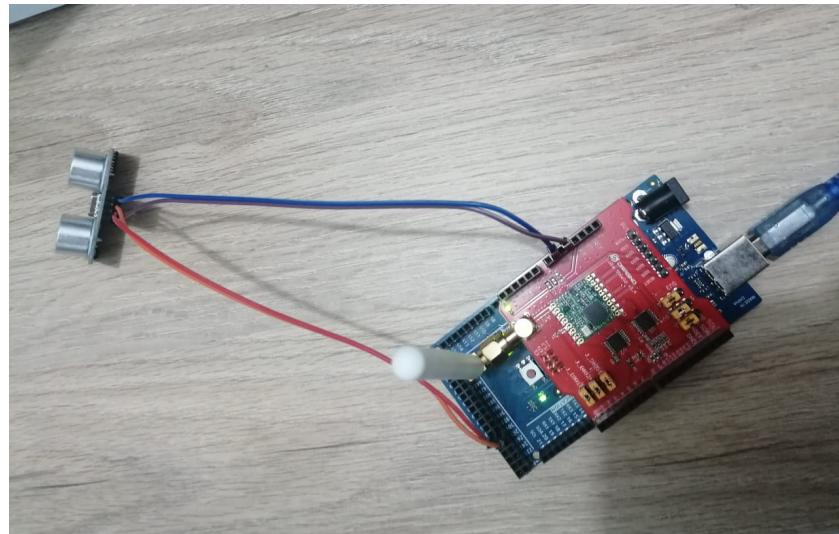


Figura 4-12 Integración de Dragino Shield y Arduino Mega 2560

4.2.1 Conexión de los componentes físicos

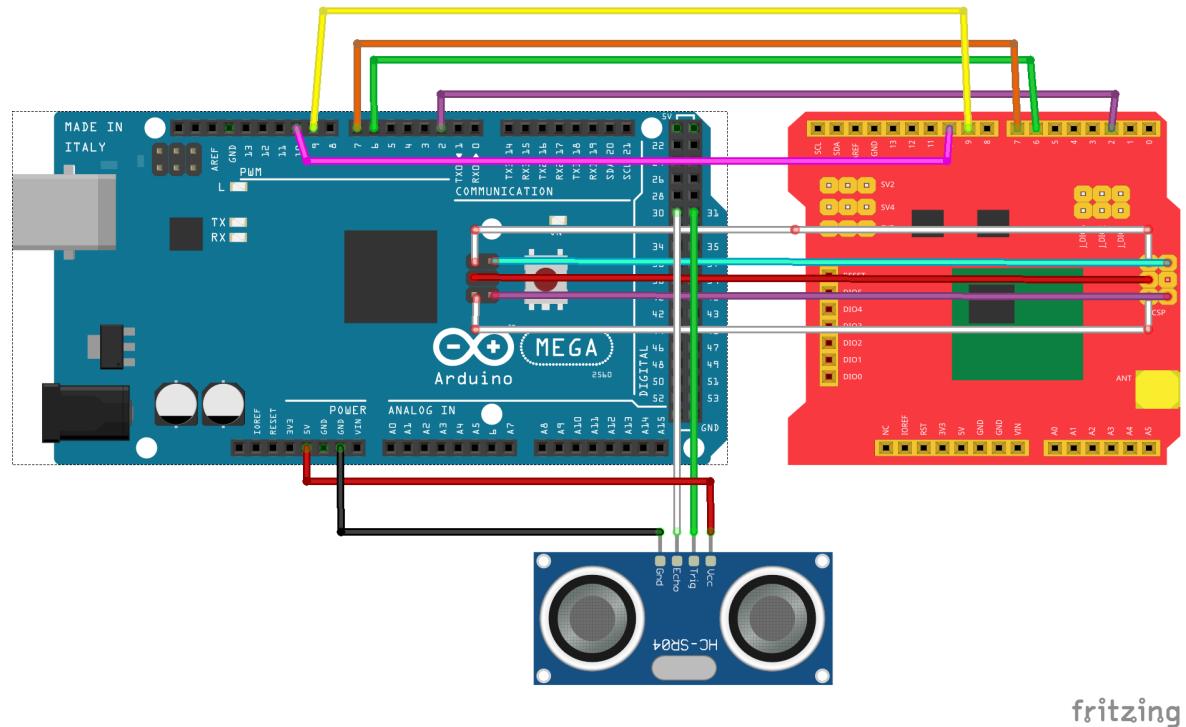


Figura 4-13 Esquema de conexión Arduino Mega, Dragino Shield y HC-SR04.

En este caso, la conexión de los componentes resulta más simple dado que se requirió únicamente la superposición de las placas arduino y dragino, sus pines se encuentran perfectamente acoplados; luego usando conectores Jumper Dupont de macho a hembra se conectó el sensor de Proximidad usando la siguiente configuración de pines (tabla 2-4).

Arduino	HC-SR04
5 V	5 V
GND	GND
D30	Echo
D31	Trigger

Tabla 4-5: Conexiones de Dragino a HC-SR04

Para simplificar esta configuración se prescindió de la conexión del módulo de ubicación GPS dado que las coordenadas pueden ser registradas cuando se instala el sensor en el contenedor la razones que serán expuestas en profundidad en la sección 4.6. Sin embargo, se obtuvieron las coordenadas aproximadas del sensor usando el servicio de mapas de Google y se codificó dicho dato en la trama que se va a enviar para poder reutilizar las funciones de codificación y decodificación que se implementaron en la primera prueba.

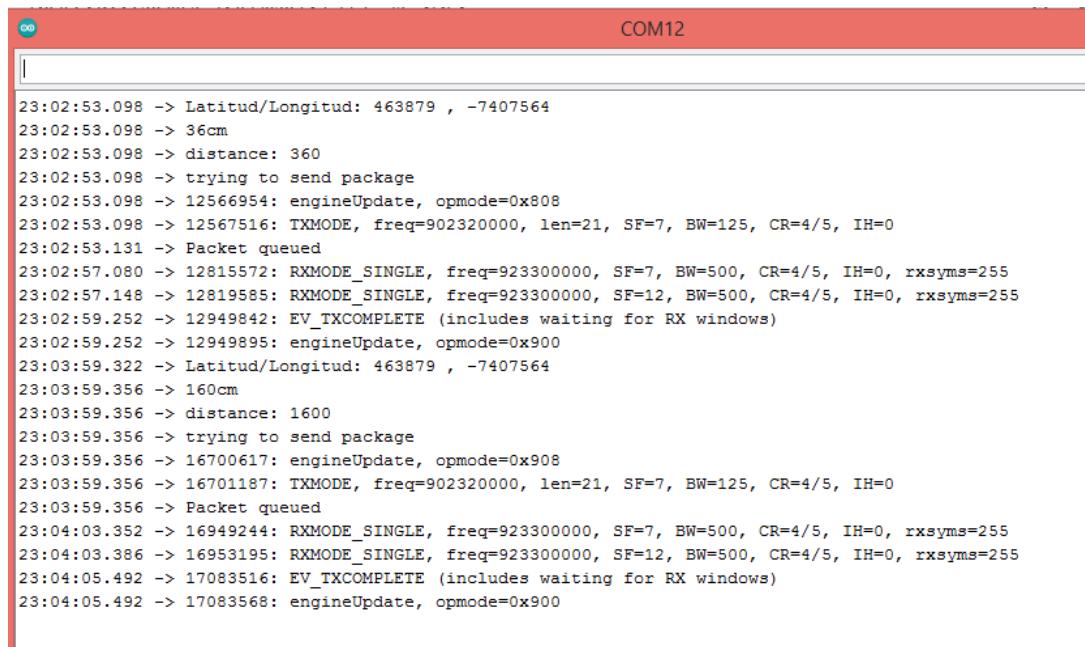
Programación lógica del microcontrolador: para programar el microcontrolador se usó la estructura que se utilizó en el primer nodo de medición con la diferencia que en este; se sustituyó la sección de la lectura de los datos de módulo de GPS por dos valores constantes de latitud y longitud, la definición de los pines del sensor de proximidad fueron ajustados por los numero 30 para *echo* y 31 para *trigger* y el mapeo de los pines entre la placa dragino y arduino quedó de la siguiente manera (tabla 4-5).

Arduino	Dragino
D2	DIO 0
D6	DIO 1
D7	DIO 2
MISO	MISO
MOSI	MOSI
SCK	SCK
D10	NSS
D9	Reset

Tabla 4-6 Conexiones entre arduino Mega 2560 y Dragino Shield

4.2.2 Pruebas de emisión de los mensajes

Nuevamente, asistidos por el monitor de puerto serial, identificamos las operaciones que ocurren dentro del micro controlar, en la parte superior se puede visualizar la coordenada geográfica que fue simulada, esta vez no fue necesario multiplicarla por 10 mil, dado que simplemente se incluyó sin cifras decimales. A continuación podemos observar el dato recuperado por el sensor de proximidad el cual multiplicamos por 10, con el propósito de eliminar las cifras decimales y hacer más fácil la codificación del dato, luego el módulo de radio entra en modo de trasmisión que emite en 902 MHz y usa un factor de propagación de 7, el paquete queda en estado encolado y el módulo entra en modo de recepción en la frecuencia de 923 MHz con factores de propagación en los 7 y los 12, genera una ventana de tiempo en la cual espera confirmación del servidor IoT, el cual aún no ha sido configurado, y se repite la secuencia, lo cual es el resultado esperado en esta prueba.



```
23:02:53.098 -> Latitud/Longitud: 463879 , -7407564
23:02:53.098 -> 36cm
23:02:53.098 -> distance: 360
23:02:53.098 -> trying to send package
23:02:53.098 -> 12566954: engineUpdate, opmode=0x808
23:02:53.098 -> 12567516: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
23:02:53.131 -> Packet queued
23:02:57.080 -> 12815572: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
23:02:57.148 -> 12819585: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
23:02:59.252 -> 12949842: EV_TXCOMPLETE (includes waiting for RX windows)
23:02:59.252 -> 12949895: engineUpdate, opmode=0x900
23:03:59.322 -> Latitud/Longitud: 463879 , -7407564
23:03:59.356 -> 160cm
23:03:59.356 -> distance: 1600
23:03:59.356 -> trying to send package
23:03:59.356 -> 16700617: engineUpdate, opmode=0x908
23:03:59.356 -> 16701187: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
23:03:59.356 -> Packet queued
23:04:03.352 -> 16949244: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
23:04:03.386 -> 16953195: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
23:04:05.492 -> 17083516: EV_TXCOMPLETE (includes waiting for RX windows)
23:04:05.492 -> 17083568: engineUpdate, opmode=0x900
```

Figura 4-14 Verificación de las operaciones en el micro controlador desde el monitor serial en Dragino.

4.3 TTGO

Lilygo es otra compañía productora de dispositivos electrónicos enfocada en soluciones con internet de las cosas, tiene su centro de operaciones en la provincia de Shenzhen en Hong Kong; en su página oficial encontramos una descripción de la compañía:

“dedicados al desarrollo de IoT, hacer que los desarrollos sean más fácil es el concepto de producto de Xinyuan, lanzamos series de productos de código abierto con diversos tipos de micro controladores y módulos IoT, hasta kits educativos para hacer posibles los desarrollo y convertir ideas creativas en realidades y hacer que se puedan compartir, hacer que todos podamos explorar el mundo y conectar los objetos es la visión de Xinyuan.” (Lilygo-Xinyuan,2017)

TTGO Para Arduino UNO LoRa MEGA328 433MHZ SX1278



Figura 4-15 Tarjeta TTGO compatible con arduino SX1278,
Tomada página oficial Lilygo

Es una tarjeta que es compatible con la configuración de pines de arduino uno y se puede programar desde el entorno de desarrollo de arduino uno, es interesante porque la misma placa integra en micro controlador y el módulo de transmisión, lo cual facilita mucho el trabajo, dado que viene ensamblado de la misma forma que se definió en la arquitectura del nodo de medición y los pines sobrantes pueden ser usados para conectar sensores como el de proximidad o el de GPS

Uno de los contratiempos fue que en el momento en el que se ordenó este artefacto aún no se tenía clara la frecuencia de las bandas autorizadas en Colombia y por un error se pidió la placa que integra el módulo SX1278, el cual opera en una frecuencia en la banda de 433 MHz, aun así se decidió usar este para hacer la prueba de concepto siendo conscientes que obtendremos buenos resultados únicamente cuando el nodo estuviese próximo al Gateway, pero que este mismo procedimiento puede ser replicado usando a una de las placas con módulos que operan a 915 MHz. Como el módulo “TTGO SX1276 LoRa ESP32 433/470MHz 868 / 915MHz Bluetooth WI-FI Internet Antenna Development Board”.

4.3.1 Conexión de los componentes físicos

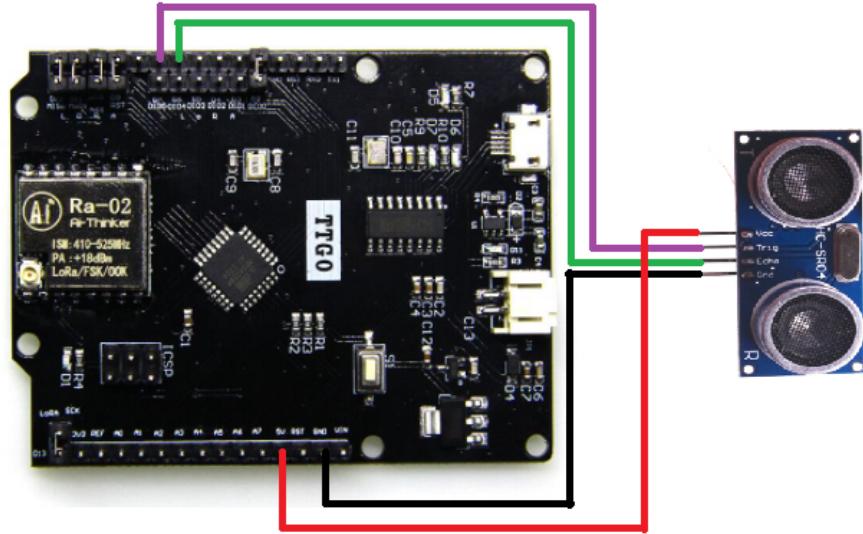


Figura 4-16 Esquema de conexión TTGO y HC-SR04

En esta configuración del Nodo de medición no hacen falta el ensamble de piezas adicionales, gracias a que el módulo de transmisión y el microcontrolador están conectados directamente en el circuito de la tarjeta de desarrollo lo único que hace falta para satisfacer la arquitectura propuesta es conectar el sensor de proximidad, lo cual podemos hacer fácilmente con la siguiente configuración de pines (tabla 4-6):

TTGO	HC-SR04
5 V	5 V
GND	GND
D6	Echo
D7	Trigger

Tabla 4-7 conexiones pines de TTGO y HC-SR04

Programación lógica del micro-controlador: en esta ocasión se usó el programa que se había escrito para el nodo uno, fue necesario hacer algunas adaptaciones; los pines del sensor de proximidad fueron definidos en 7 para echo y 6 para trigger, se reemplazó el código del sensor con el que se obtenían las coordenadas geográficas por 2 valores constantes para latitud y longitud, y el mapeo de pines entre el microcontrolador y el módulo de transmisión quedó de la misma manera que en el nodo 2 con la única diferencia que se deshabilitó el DIO 2.

4.3.2 Pruebas de emisión de datos

En este caso, podemos observar una operación muy similar a la de los nodos anteriores dado que el programa que se grabó en el microprocesador usa la misma secuencia para gestionar los recursos. En este caso, ocurre una pequeña variación que es muy importante en las próximas etapas del proceso de comunicación. Tras haber obtenido los datos del sensor de proximidad y simular las coordenadas geográficas, el módulo de radio encola el mensaje, se hace énfasis que es este caso se puede leer el mensaje **EV_JOINING**, esto indica que el mensaje ha sido recibido por un *gateway* y que se inició un protocolo de autenticación con el servidor de IoT para iniciar una comunicación, lo cual se abordará en profundidad en el capítulo 5.

```

COM11
Enviar

23:58:14.512 -> Starting
23:58:14.547 -> RXMODE_RSSI
23:58:14.547 -> 158cm
23:58:14.547 -> distance: 1580
23:58:14.581 -> Latitud/Longitud: 4.63 , -74.09
23:58:14.615 -> Latitud/Longitud: 463063 , -7408698
23:58:14.651 -> 3233: engineUpdate, opmode=0x8
23:58:14.685 -> Packet queued
23:58:17.679 -> 193723: EV_JOINING
23:58:17.679 -> 193749: engineUpdate, opmode=0xc
23:58:17.679 -> 194079: TXMODE, freq=902320000, len=23, SF=7, BW=125, CR=4/5, IH=0
23:58:22.689 -> 505145: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsyms=255
23:58:22.784 -> 508347: JaccRX1, dataLen=0
23:58:22.784 -> 508414: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsyms=255

```

Figura 4-17 Verificación de las operaciones en el micro controlador desde el monitor serial en TTGO

4.4 Comparación de las opciones:

Criterio	Arduino	Dragino	TTGO
Costo (COP)	\$97.000	\$170.000	\$80.000
disponibilidad	Disponible en Colombia	Importación de USA	Importación de China
Ensamble	Complejo	Medio	Sencilla
Funcionalidad	Bueno	Bueno	Bueno

Tabla 4-8 Tabla comparativa de los nodos de medición.

Haciendo un balance entre las opciones que se validaron, consideramos que resulta más conveniente importar placas de la marca Lilygo desde China no solo porque su costo es

inferior a las otras 2 opciones, sino porque facilita considerablemente el ensamblaje de los componentes gracias a que el microcontrolador y el módulo de transición vienen conectados de fábrica, el único aspecto que está en su contra es la disponibilidad, lo cual no se considera, un problema pues si bien la tarjeta puede tardar un poco en llegar al ser importada de China, se ha constatado que el proveedor es confiable y está en la capacidad de hacer entregas en Colombia. Una evaluación de los dispositivos se hace en el capítulo 5 en el cual se incluyen dentro de la decisión el criterio distancia de cobertura que es uno de los criterios más relevantes para el estudio.

4.5 Librería Arduino-LMIC de Matthijs Kooijman

La librería LMIC fue creada para manejar los parámetros de la comunicación entre los dispositivos de medición y el servidor de IoT del cual hablaremos en el capítulo 5, esta librería es de código libre bajo una licencia Eclipse Public License 1.0. La librería se puede instalar fácilmente desde el gestor de librerías en arduino studio. También está publicada en la cuenta de github del autor (Matthijs kooijman, 2015).

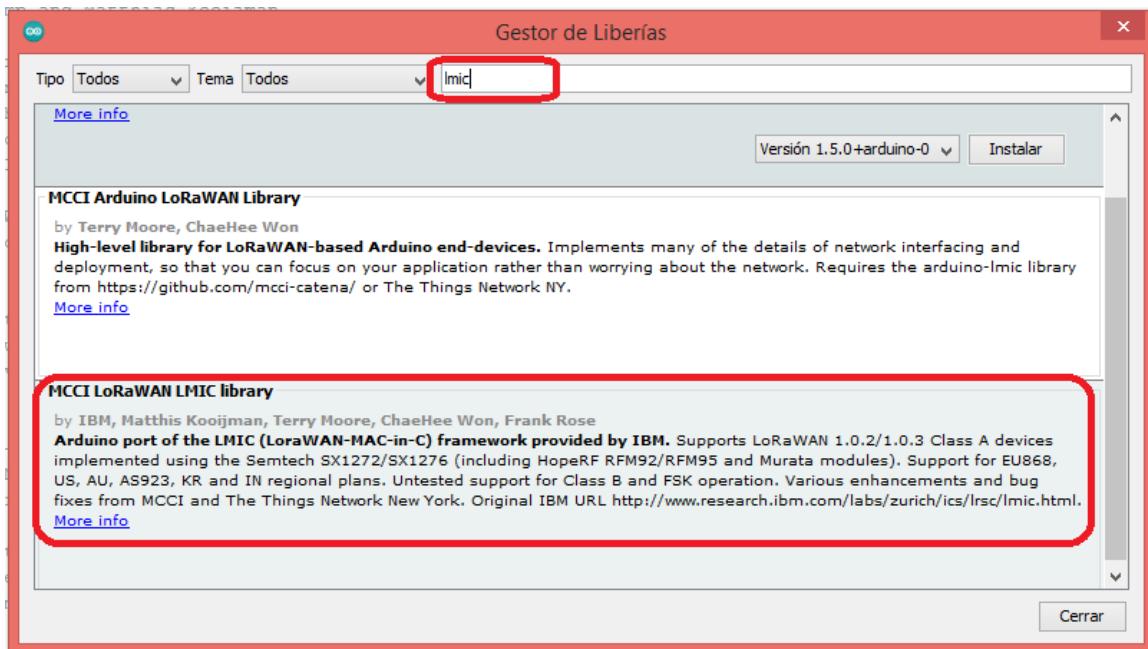


Figura 4-18 Instalación de la Librería desde el gestor de librerías de arduino studio.

Esta librería ofrece una implementación completa para dispositivos LoRaWAN de clase A y clase B, soportando las bandas de 868 y 915 dentro de las características que ya han sido validadas están:

- El envío de paquetes ascendentes, considerando los ciclos de trabajo.
- La encriptación de mensajes y verificación de integridad
- La recepción de mensajes de confirmación en la primera ventana de tiempo RX2

-
- Personalización de las frecuencias y parámetros de tasa de datos
 - La conexión por método OTAA

Hay algunas características que aún están pendientes por verificar, entre ellas: la recepción de paquetes descendentes en la primera ventana de tiempo RX1, la recepción y procesamiento de comandos MAC y la operación de dispositivos de clase B.

La comunidad generada en torno a la librería se ha encargado de mejorar continuamente no solo reportando los errores que obtienen con los diferentes tipos de hardware, si no registrando y documentando las soluciones que han encontrado para resolver los inconvenientes, esto ha hecho que la librería madure muy rápidamente, para ofrecer una operación robusta pero además para ganar versatilidad con otras tarjetas menos populares que arduino.

4.6 Consideraciones adicionales de nodo de medición

Las configuraciones y pruebas que se realizaron con los nodos de medición previamente expuestos son apenas una prueba de concepto para validar la funcionalidad y operatividad del dispositivo, sin embargo para adaptar esos nodos en un entorno productivo real, aún se requieren resolver los aspectos que se mencionan a continuación.

Necesidad de recubrimiento externo: dado que los dispositivos estarán expuestos a condiciones climáticas normales como el sol y la lluvia, que posiblemente causen daños en los circuitos. Se requiere diseñar un recubrimiento externo o empaque hermético para evitar filtraciones de agua, se sugiere que sea en una pasta dura para proteger el dispositivo de golpes y que tenga un compartimento independiente para poner las baterías y reemplazarlas fácilmente, sin tener que tocar los demás componentes del circuito.

Optimización del uso de la batería: dado que la batería es uno de los componentes que debe ser mejor administrado para reducir la frecuencia de mantenimiento de los nodos, es necesario analizar la forma de disminuir el consumo de la tarjeta, bien sea reemplazandola por una de menor consumo, poniéndola en modo de hibernación (sleep) durante los periodos que no se está realizando mediciones o sustituyendo el regulador de voltaje por un módulo de potencia de ultra bajo consumo encargado de activar la tarjeta únicamente para tomar y transmitir las mediciones, y luego apagar la placa completamente (Cerchecci et al., 2018) o por medio de un circuito que transforma la energía solar en corriente eléctrica por medio de una celda fotovoltaica como proponen en (Aslam et al., 2020).

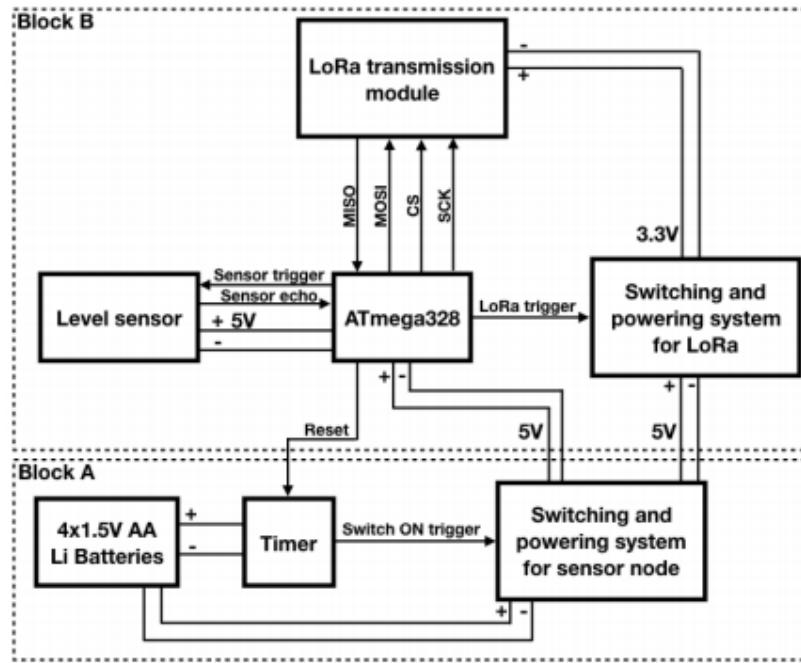


Figura 4-19 Diagrama de bloques para nodo de bajo consumo, tomado de (Cerchecci et al., 2018).

Prescindir del módulo de localización: teniendo en cuenta que los contenedores de basura raras veces cambian de ubicación, en lugar de ello tienden a mantenerse constantemente en el mismo sitio y que esquema de datos del sistema está dotado de un componente espacial, es posible emparejar la ubicación del nodo y la del contenedor en el momento en que se instale el dispositivo de medición y persistir esta información directamente en la base de datos, de esta manera se logra tener georeferenciación de los componentes del sistema, lo cual no solamente disminuye el costo de producción del nodo en un 33% aproximadamente, sino que permite ser más eficiente en el momento de tomar las mediciones, dado que el módulo de GPS puede tardar de 1 a 3 minutos, triangulando los datos de los satélites cuando lleva algún tiempo inactivo, afectando significativamente la duración de la batería.

5. Configuración de RIS y Arquitectura IoT

Indiscutiblemente el nodo de medición es un elemento importante en una red inalámbrica de sensores RIS, pero es solo uno de los eslabones de la cadena, además de los nodos existen otros 3 componentes imprescindibles en una LPWAN: primero, los *gateways* que se encargan de mantenerse escuchando los mensajes de radio emitidos por los nodos, se encargan de demodular dichas señales transformarlas en pulsos digitales, posteriormente la versión digital del mensaje es enrutada a través de internet hacia el siguiente componente. Segundo, el servidor que es el encargado de brindar servicios para gestionar la información recibida, validando su integridad, pertinencia y publicando esta información, en el tercer componente un intermediario que llamaremos bróker y cuya responsabilidad principal es la de notificar dicha información a las aplicaciones. En este capítulo se abordarán de manera detallada la configuración y puesta en marcha de cada uno de estos componentes como parte de la implementación del sistema de monitoreo para la planificación de rutas de recolección de RSU.

5.1 Gateway

También conocidos como puertas de enlace son dispositivos electrónicos que actúan como puentes que unen 2 mecanismos de comunicación por medio de la decodificación de señal. En el caso concreto de LoRaWAN los *gateways* reciben mensajes emitidos por los nodo finales que fueron modulados usando LoRa y que son recibidos usando componente físico llamado concentrador, cada uno de los mensajes es decodificado en forma de bits y se repite hacia el servidor usando un esquema IP (Santos Filho et al., 2020).

Existen varios tutoriales que indican detalladamente el procedimiento para construir *gateways* artesanales en los que con una tarjeta de desarrollo multipropósito como una Raspberry pi 3 y un concentrador de señal se puede ensamblar el dispositivo; además, se han creado software para permitir configurar las frecuencias de transmisión y la ruta servidor en el que se procesaran los mensajes. Con este tipo de soluciones también se hace necesario construir una antena con ciertas especificaciones técnicas para amplificar la ganancia de la señal en el *gateway*, no obstante se encontró una solución lista que ya viene ensamblada y dispone de una interfaz de configuración por un precio muy

razonable, por lo cual decidimos usar este componente dentro de nuestra red en lugar de gastar tiempo y dinero experimentando con la fabricación de un *gateway* artesanal.

5.1.1 LG02 de Dragino.

El dispositivo elegido para usar en nuestra red fue el *gateway* de 2 canales de LG02 de Dragino, el cual es capaz de enlazar una señal LoRa conectándose a Internet a través de WiFi, Ethernet, o de manera celular en la red 3G/4G usando un módulo LTE; esta cantidad de interfaces proveen buena flexibilidad para que los usuarios conecten sus nodos a internet, el dispositivo usa software Linux para controlar los 2 módulos 1276, este le permite operar en modo full dúplex¹³ incrementando la eficiencia de la comunicación.

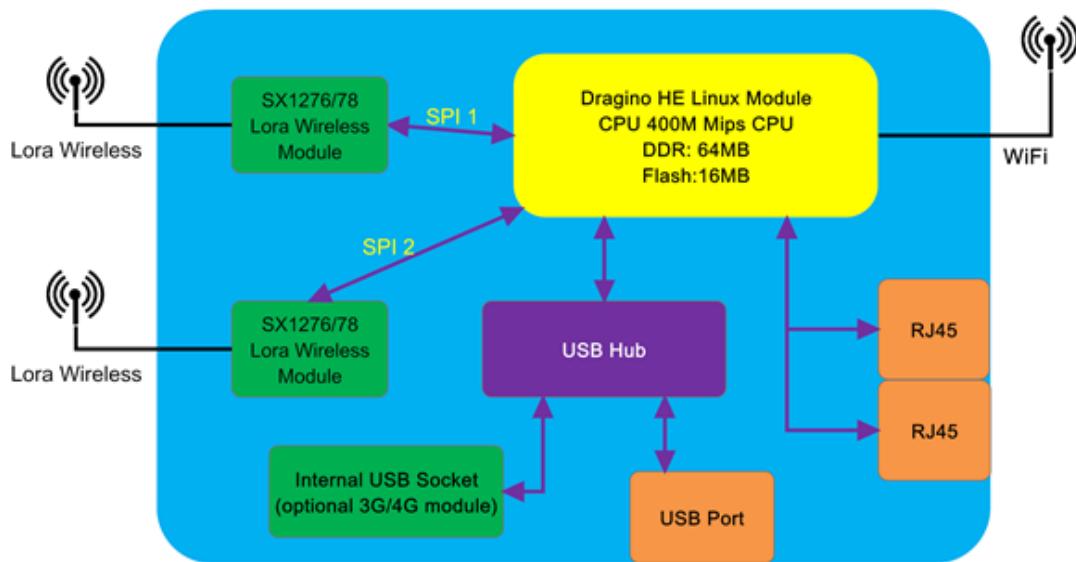


Figura 5-1 Diagrama general componentes Gateway LG02, tomada de (*Dragino LG02*).

Además de operar como *gateway*, el dispositivo es configurable en diferentes modos de operación para ajustarse a las diferentes necesidades de conexión IoT funcionando como: Repetidor LoRa, modo de receptor MQTT, como cliente TCP/IP o servidor TCP/IP, esta es una solución de bajo costo y por esto su capacidad de servicio está limitada a la conexión de hasta 300 nodos finales; sin embargo, su costo es aproximadamente la cuarta parte de lo que cuesta un *gateway* con un módulo concentrador potente, como el SX1301.

¹³ Full dúplex es una especificación que permite comunicación bidireccional gracias a que los dispositivos disponen de canales independientes para emitir y recibir señal, de manera que pueden realizar ambas operación de forma simultánea sin tener que generar interrupciones al alternar sus modos de operación .

5.1.2 Configuración del Gateway

Para la configuración del dispositivo se conectó vía Ethernet y se siguió el manual del fabricante, una vez el dispositivo estaba conectado a la red fue posible acceder a su interfaz de configuración ingresando por medio de un navegador web a la siguiente ruta: <http://10.130.1.1/cgi-bin/luci/admin>.

Esta ruta muestra una interfaz de autenticación a la que solo se puede acceder usando usuario y contraseña suministrados por el fabricante los cuales están en la etiqueta del producto. Una vez allí se selecciona el modo de operación; en este caso se usará como gateway LoRa.

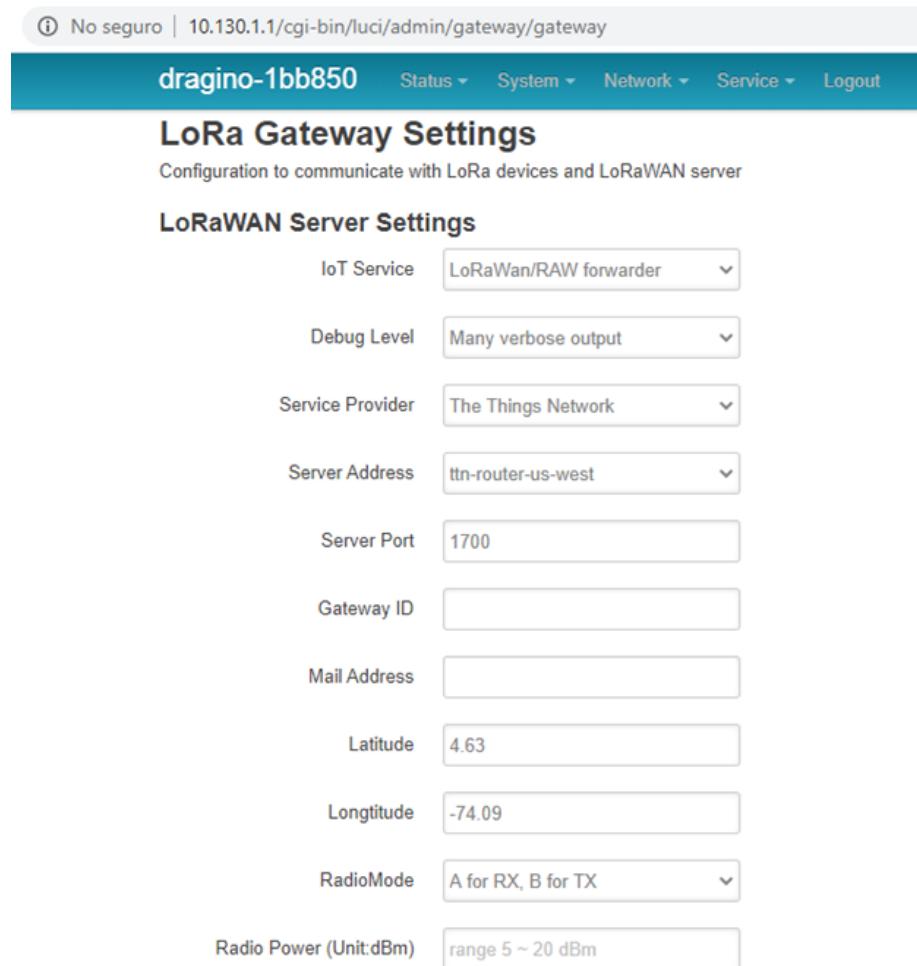


Figura 5-2 Interfaz de configuración del LG02.

Los parámetros que se observan en la imagen anterior incluyen el modo en el que se usará el *gateway*, en nuestro caso una repetidor de datos sin procesar, el nivel de información que queda en el registro del *gateway*, el servidor IoT que recibirá los

mensajes, la ubicación física del servidor, el puerto habilitado para la recepción de comunicaciones, las coordenadas geográficas, los canales de radio, se usará el canal A para recibir y el canal B para transmitir, la potencia de la señal de radio en Dbs y el correo electrónico de notificación

En la siguiente sección se explica cómo se deben configurar los parámetros con los que se están modulando las señales desde los nodos; es importante tener en cuenta que los parámetros con los que emiten los nodos deben coincidir exactamente con los que recibe el gateway en nuestro caso los parámetros de recepción quedarán de la siguiente manera.

Channel 1 Radio Settings

Radio settings for Channel 1

RadioA Frequency (Unit:Hz)	902320000
RadioA Spreading Factor	SF7
RadioA Coding Rate	4/5
RadioA Signal Bandwidth	125 kHz
RadioA Preamble Length	8 <small>Length range: 6 ~ 65536</small>
RadioA LoRa Sync Word	52 <small>Value 52(0x34) for LoRaWAN</small>
Encryption Key	Encryption Key

Figura 5-3 Configuración del canal de recepción LG02.

Aquí se pueden distinguir: la frecuencia de modulación en Hercios, el factor de propagación que varía entre 7 y 12, el radio de codificación que será de 4/5 y el ancho de banda que puede ser 125 kHz, 250 kHz o 500 kHz. De manera similar, se deben configurar los datos de transmisión, considerando nuevamente que estos parámetros serán los que se establezcan en los nodos como datos de recepción para que efectivamente reciban las señales descendentes.

Channel 2 Radio Settings

Radio settings for Channel 2

RadioB Frequency (Unit:Hz)	<input type="text" value="923300000"/>
RadioB Spreading Factor	<input type="text" value="SF7"/>
RadioB Coding Rate	<input type="text" value="4/5"/>
RadioB Signal Bandwidth	<input type="text" value="125 kHz"/>
RadioB Preamble Length	<input type="text" value="8"/>
	<small>Length range: 6 ~ 65536</small>
RadioB LoRa Sync Word	<input type="text" value="60"/>
	<small>Value 52(0x34) for LoRaWAN</small>
Encryption Key	<input type="text" value="Encryption Key"/>

Figura 5-4 Configuración del canal de transmisión LG02.

Para que los cambios tengan efecto, se debe reiniciar el *gateway*. En los nodos esta información se puede definir en el archivo de configuración de radio que se encuentra en la siguiente ruta `src\lmic\config.h`, como se puede ver las frecuencias y SF son inversas a las que se definieron en el *gateway*.

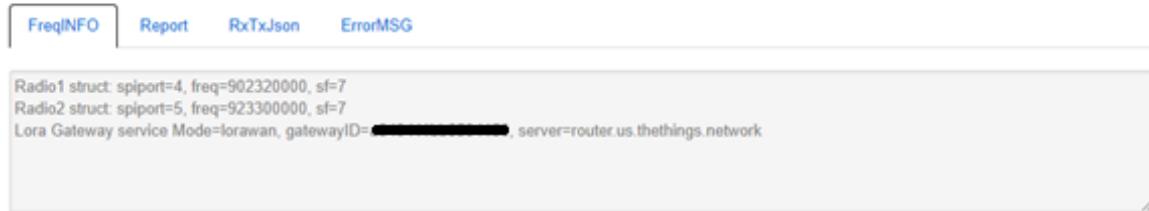
```
#define LG02_LG01 1

//US915: DR_SF10=0, DR_SF9=1, DR_SF8=2, DR_SF7=3, DR_SF8C=4
//          DR_SF12CR=8, DR_SF11CR=9, DR_SF10CR=10, DR_SF9CR=11, DR_SF
#if defined(CFG_us915) && defined(LG02_LG01)
// CFG_us915 || CFG_as923
#define LG02_UPFREQ    902320000
#define LG02_DNWREQ   923300000
#define LG02_RXSF      3      // DR_SF7  For LG01/LG02 Tx
#define LG02_TXSF      8      // DR_SF12CR For LG02/LG02 Rx
#elif defined(CFG_eu868) && defined(LG02_LG01)
```

Figura 5-5 Configuración de los parámetros de transmisión y recepción en la librería

Para validar los parámetros con los que opera el *gateway* podemos simplemente revisar el registro que se halla en la pestaña “información de frecuencias”, podemos observar cada uno de los canales, el modo de operación, el identificador del dispositivo, el cual dejará una marca en los paquetes que transitén a través de él y el servidor al cual se enviarán los mensajes.

Logread



The screenshot shows a log viewer titled "Logread". At the top, there are four tabs: "FreqINFO" (selected), "Report", "RxTxJson", and "ErrorMSG". The main area displays log entries:

```

Radio1 struct: spiport=4, freq=902320000, sf=7
Radio2 struct: spiport=5, freq=923300000, sf=7
Lora Gateway service Mode=lorawan, gatewayID=[REDACTED], server=router.us.thethings.network

```

Figura 5-6 Registro de parámetros de operación del LG02.

Algo que nos interesa revisar son los mensajes que recibe el *gateway*; para esta prueba, además de tener el *gateway* configurado y conectado a la red, activamos uno de los nodos que se configuraron en el capítulo 4 suministrando corriente, luego de un par de minutos en la pestaña de trasmisión se empezaron a obtener los mensajes que percibe el *gateway*.

Logread



The screenshot shows a log viewer titled "Logread". At the top, there are four tabs: "FreqINFO" (selected), "Report", "RxTxJson", and "ErrorMSG". The main area displays log entries, with the "RxTxJson" tab selected. The entries show multiple instances of retransmitted messages (RXPKT) over time:

```

Receive(HEX):00635a02d07ed5b370514150b81b4140a8be10699322fb
(RXPKT): [up] {"rxpk": [{"time": "2020-10-04T01:53:47.891865Z", "tmst": 2734587097, "chan": 0, "rfch": 1, "freq": 902.320000, "stat": 1, "modu": "LORA", "datr": "SF7BW125"}, {"down] {"bpk": {"imme": false, "tmst": 2740587097, "freq": 923.3, "rfch": 0, "powe": 20, "modu": "LORA", "datr": "SF12BW500", "codr": "4/5", "ipol": true, "size": 17, "nc}
Receive(HEX):40952902268000000102ccb96fb6077bca3bfad0dd
(RXPKT): [up] {"rxpk": [{"time": "2020-10-04T01:53:54.323107Z", "tmst": 2741018353, "chan": 0, "rfch": 1, "freq": 902.320000, "stat": 1, "modu": "LORA", "datr": "SF7BW125"}, {"down] {"bpk": {"imme": false, "tmst": 2743018353, "freq": 923.3, "rfch": 0, "powe": 20, "modu": "LORA", "datr": "SF12BW500", "codr": "4/5", "ipol": true, "size": 22, "nc}
Receive(HEX):00635a02d07ed5b370514150b81b4140a87a449cc6e60f
(RXPKT): [up] {"rxpk": [{"time": "2020-10-04T01:54:34.285414Z", "tmst": 2780980642, "chan": 0, "rfch": 1, "freq": 902.320000, "stat": 1, "modu": "LORA", "datr": "SF7BW125"}, {"down] {"bpk": {"imme": false, "tmst": 2786980642, "freq": 923.3, "rfch": 0, "powe": 20, "modu": "LORA", "datr": "SF12BW500", "codr": "4/5", "ipol": true, "size": 17, "nc}
Receive(HEX):4014220226800000016ac8e9caf28e9fc55386dbcc
(RXPKT): [up] {"rxpk": [{"time": "2020-10-04T01:54:40.741473Z", "tmst": 2787436705, "chan": 0, "rfch": 1, "freq": 902.320000, "stat": 1, "modu": "LORA", "datr": "SF7BW125"}, {"down] {"bpk": {"imme": false, "tmst": 2789436705, "freq": 923.3, "rfch": 0, "powe": 20, "modu": "LORA", "datr": "SF12BW500", "codr": "4/5", "ipol": true, "size": 22, "nc}
Receive(HEX):4014220226c201000307016bbf81628cd611bdbaf3888f
(RXPKT): [up] {"rxpk": [{"time": "2020-10-04T01:55:43.358444Z", "tmst": 2850053672, "chan": 0, "rfch": 1, "freq": 902.320000, "stat": 1, "modu": "LORA", "datr": "SF7BW125"}, {"down] {"bpk": {"imme": false, "tmst": 2852053672, "freq": 923.3, "rfch": 0, "powe": 20, "modu": "LORA", "datr": "SF12BW500", "codr": "4/5", "ipol": true, "size": 22, "nc}

```

Figura 5-7 Registro de mensajes retransmitidos por LG02.

Si estos mensajes corresponden con los cuales emitió el nodo que activamos, podríamos afirmar que la prueba es exitosa; sin embargo, dado que para efectos de seguridad la información del mensaje se encuentra codificada, es complicado identificar si estos mensajes que se muestran en el registro pertenecen al nodo que fue activado o pertenecen a un nodo desconocido en la misma zona de cobertura, lo cual es poco probable, pues para haber sido recibido, debe estar configurado con los mismos parámetros que nosotros usamos en el *gateway*. De cualquier manera, esto podemos validarla cuando decodifiquemos el mensaje una vez llegue al servidor.

5.2 Servidor IoT

Uno de los pilares de IoT es la capacidad de identificar objetos del mundo real y dándoles una identidad en internet a través de la virtualización para poder monitorearlos; un servidor IoT es el lugar donde se programan las reglas para que los dispositivos puedan interactuar o, mejor, cooperar para cumplir un objetivo común, sin necesidad de la intervención humana. Por otra parte, es aquí donde se agrupa la información de los dispositivos físicos y se analiza en conjunto para encontrar patrones o tendencias que permitan modelar e implementar soluciones prácticas y eficientes.

Para entender de forma concreta que es un servidor IoT, se puede pensar en un servicio publicado en internet que ofrece funcionalidades a los desarrolladores de sistemas de IoT para comunicar los dispositivos físicos y las aplicaciones; algunos de estos servicios incluyen el registro de dispositivos sean nodos finales o *gateways*, la creación de aplicaciones desde las cuales se puede hacer seguimiento a los mensajes enviados por los dispositivos. Uno de los aspectos importantes en una aplicación IoT es la seguridad, lo cual se consigue por medio de la autenticación de dispositivos y la verificación de integridad de los mensajes, existen servidores que permiten la decodificación de mensajes y algunos servicios para el post procesamiento de datos, como esquemas de visualización de información, integración con servidores de mapas o servicio de bases de datos en la Nube.

Algunos de los servidores IoT más populares en la Nube son Ubidots, ThingWorx, WSO2 IoT, ThingsBoard, AWS IoT Analytics y ThingSpeak; sin embargo, dependiendo de la tecnología de trasmisión que se haya elegido en la capa física, hay algunos que resultan integrarse mejor a nuestras aplicaciones en el caso de redes que usen el protocolo LoRaWAN unos de los servidores que mejor implementan los componentes de este protocolo es The Things Network TTN y por esta razón se ha elegido para el sistema propuesto.

5.2.1 The Things Network TTN

TTN suministra un servidor de red LoRaWAN que usa componentes minimalistas enfocados en los dispositivos; el servidor se encarga de la lógica de conexión o lo que conocemos popularmente como *backend*, algunas de las funciones que proporciona TTN están orientadas a los *gateways*, por ejemplo, el agendamiento que es la gestión de los dispositivos obedeciendo a los ciclos de trabajo y el enrutamiento de mensajes descendentes a través de los *gateways* que ofrezcan mejor señal al dispositivo final. Hay funciones orientadas a los dispositivos como la administración del estado de los dispositivos en la red, ofrece funcionalidades orientadas a las aplicaciones como los son los Brokers y los manejadores que se usan para procesar el tráfico en la red indicando

que información específica requiere ser enrutada a cada aplicación haciendo uso de protocolos como AMQP o MQTT.

TTN dispone de una comunidad en la que los usuarios pueden cooperar para crear aplicaciones, compartir dispositivos y encontrar la distribución de *gateways* que permita maximizar la cobertura en la ciudad, dándole mayor soporte a las aplicaciones, esto requiere ofrecer servicios de interoperabilidad en los que las aplicaciones se descentralicen, un servicio de descubrimiento para anexar a la red los nuevos nodos que se activen en el área de cobertura, ofrecer servicios que faciliten el enrutamiento de los mensajes para garantizar que son recibidos en el sitio que se requieren.

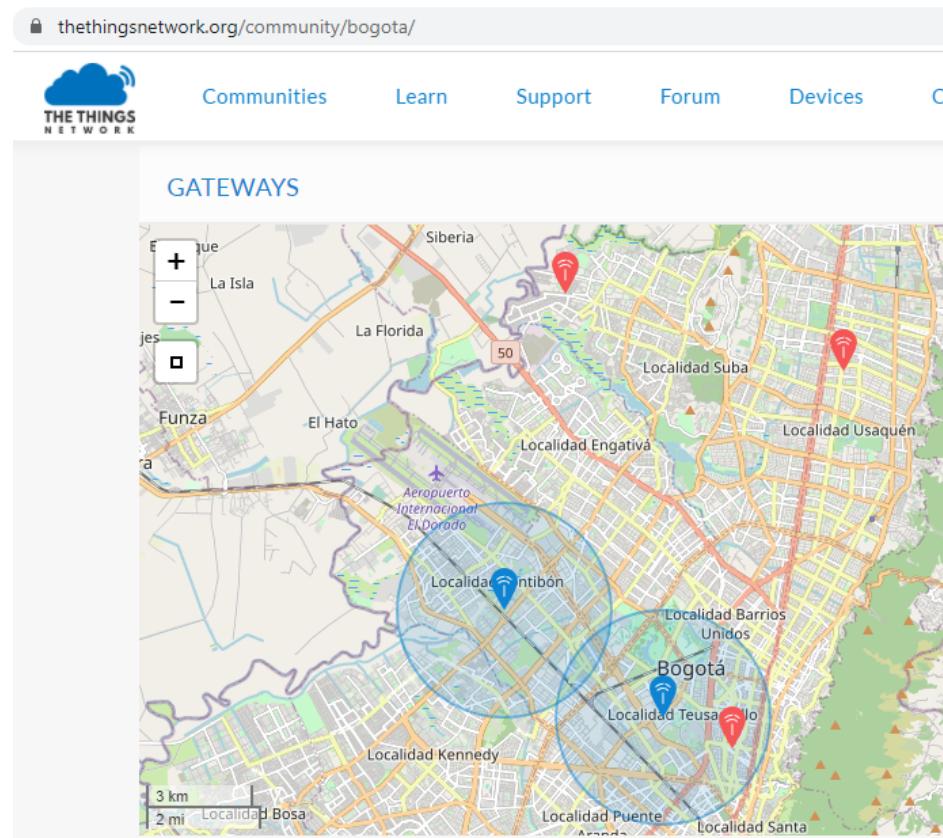


Figura 5-8 Localización de *gateways* registrados en Bogotá.

Para poder integrar los dispositivos en TTN se requieren algunas configuraciones preliminares que se explican a continuación

5.2.2 Registro de dispositivos

Antes de usar “The Things Network”, se debe crear una cuenta de usuario esto le permite tener control sobre y administrar los dispositivos que se creen desde su cuenta de

usuario y le garantiza que solo la persona con los datos de acceso a la cuenta podrá cambiar la información y la configuración de dichos dispositivos.

Registro del gateway:

TTN dispone de una funcionalidad para registrar el *gateway* en su consola de administración, para esta operación se requieren algunos datos como el código de identificación, el plan de frecuencias en el que opera, la ubicación, el tipo de antena que usa y una descripción, luego de registrado se puede observar desde la consola.

The screenshot shows the 'GATEWAY OVERVIEW' section of the TTN console. At the top, it displays the Gateway ID: eui-a840411bb8504150. Below this, the 'Description' is listed as 'recolección de basuras bogota'. The 'Owner' is 'montanarco' with a 'Transfer ownership' link. The 'Status' is 'connected'. The 'Frequency Plan' is 'United States 915MHz'. The 'Router' is 'ttn-router-us-west'. The 'Gateway Key' field is present. Below these details, it shows 'Last Seen' as '20 seconds ago', 'Received Messages' as '1754', and 'Transmitted Messages' as '1611'. At the bottom, the 'INFORMATION' section lists the 'Brand' as 'Dragino', 'Model' as 'LG02', and 'Antenna' as 'None'. There is also an 'edit info' link.

Figura 5-9 Registro exitoso del Gateway en la consola de TTN.

Una vez el servidor recibe una petición con el código de identificación con el que el *gateway* ha sido registrado, lo identifica y asocia y actualiza su estado. En la figura 5-9 se puede observar que el *gateway* está conectado y la última vez que el servidor recibió señal del *gateway*.

Creación de la aplicación:

Desde la consola de administración en la sección de aplicaciones también se creó una aplicación en la cual se recogerán los datos de los sensores que se reciban, para ello se requiere asignar un nombre a la aplicación; el servidor genera un código de aplicación, una descripción y la selección de un manejador de tráfico, del listado que se encuentra disponible. Se recomienda que el manejador sea el mismo que el servidor asignó durante

el registros del *gateway*, el cual está muy relacionado con el plan de frecuencias seleccionado.

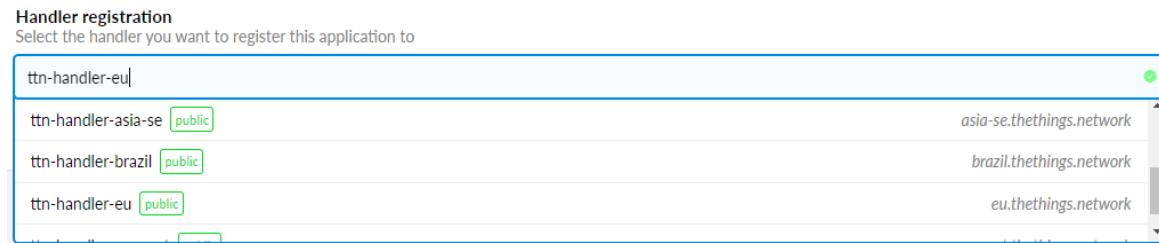


Figura 5-10 Selección de un manejador de tráfico para la aplicación.

Cuando la aplicación queda registrada, se podrá ver su información desde la consola.

The screenshot shows the "APPLICATION OVERVIEW" page for the application "01basurasbogota". The top navigation bar includes "Applications > 01basurasbogota". The main section displays the following details:

- Application ID:** 01basurasbogota
- Description:** recolección de basuras bogota
- Created:** 11 months ago
- Handler:** ttn-handler-us-west

Figura 5-11 Información de la aplicación creada.

Registro de los nodos:

El registro de nodos se hace en la sección dispositivos de la aplicación que se creó anteriormente; con esto se consigue que los dispositivos queden asociados a dicha aplicación. Dicho de otra manera, que los mensajes que se reciben desde los dispositivos que están asociados a la aplicación sean procesados con las mismas reglas que se programan a nivel de aplicación. Para registrar un dispositivo, es suficiente con darle un código de identificación, se sugiere que este código tenga un significado relevante que nos dé cierto grado de información acerca de qué función cumple el dispositivo en la red, dado que aunque esta información también puede ser enviada en la trama del mensaje aumenta el tamaño del paquete, que es algo que se desea evitar considerando las limitaciones de ancho de banda.

Una vez el dispositivo ha sido creado, se requiere configurar un mecanismo de activación que como vimos en los conceptos generales podía ser OTAA o ABP; para este caso, se eligió OTAA, para ello TTN genera las claves necesarias para poder procesar esta autenticación.

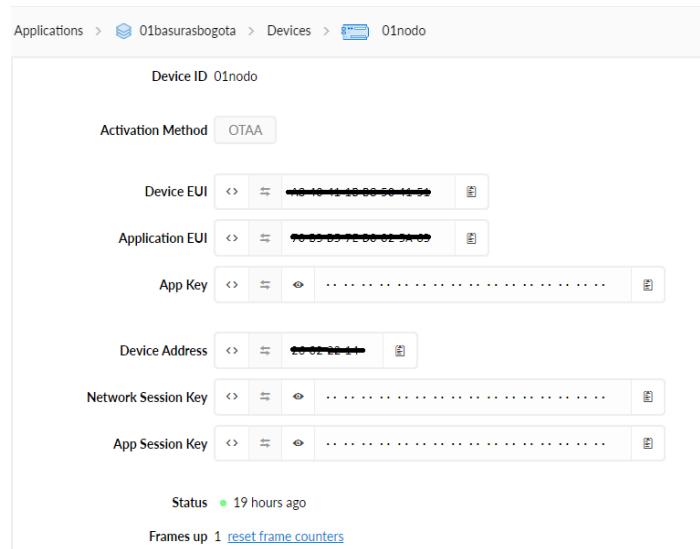


Figura 5-12 Registro del 01nodo en la aplicación TTN.

De forma análoga se procedió con el registro de los otros dos dispositivos que se ensamblaron en el capítulo 4, con lo cual quedaron registrados los tres dispositivos; el punto verde al final de cada línea indica que ya se ha establecido una conexión entre el servidor y el nodo.

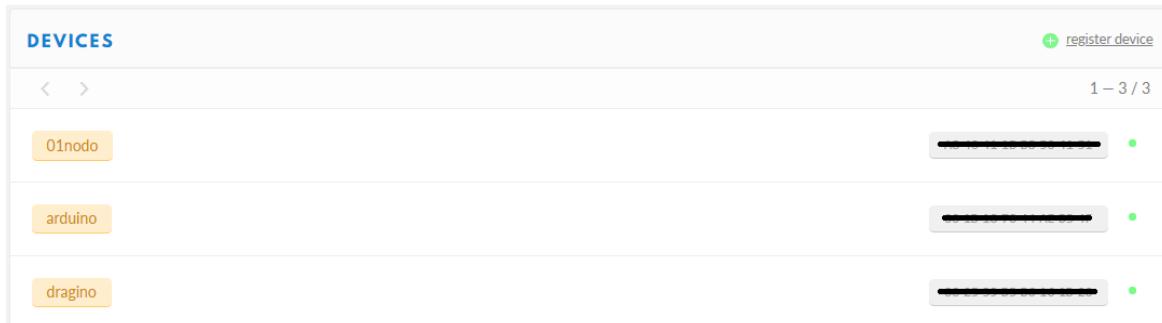


Figura 5-13 Lista de dispositivos asociados a la aplicación.

Autenticación OTAA:

Dado que ya disponemos de los códigos requeridos para gestionar la activación, Device EUI, App EUI y App Key, podemos hacer uso de dichos códigos para configurar el nodo y así poder establecer un canal de comunicación seguro entre el nodo y el servidor.

```

static const ul_t PROGMEM APPEUI[8]={████████████████, 0x70 };
void os_getArtEui (ul_t* buf) { memcpy_P(buf, APPEUI, 8);}

static const ul_t PROGMEM DEVEUI[8]={████████████████, 0xA8 };
void os_getDevEui (ul_t* buf) { memcpy_P(buf, DEVEUI, 8);}

static const ul_t PROGMEM APPKEY[16] = {████████████████████████████████, 0xEE, 0x7C };
void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}

```

Figura 5-14 Configuración de los códigos de autenticación OTAA en los nodos.

Estos códigos serán parte de la trama cuando se realice la petición de conexión, el servidor validará la correspondencia de estos datos y en caso de haber coincidencia retornará por medio de un mensaje descendente la dirección del dispositivo, que será usará por el Gateway para enrutar el mensaje de retorno, además de un ID de red y un AppNounce, que el nodo usará para generar el AppSKey y el NetSKey, con los que empezara a enviar los mensajes al servidor. La demostración de la comunicación se hará más adelante en este capítulo.

Función de decodificación:

Los datos emitidos por el nodo final se organizan en un arreglo de 8 bytes de los cuales se usan 3 para la coordenada de latitud, 3 para la coordenada de longitud y 2 para la distancia que obtiene el sensor de proximidad, es necesario hacer un corrimiento de bits para que los datos muy grandes se ajusten al tamaño del arreglo, esta codificación se realiza a nivel de nodo de la siguiente manera:

```

payload[0] = lat;
payload[1] = lat >> 8;
payload[2] = lat >> 16;

payload[3] = lon;
payload[4] = lon >> 8;
payload[5] = lon >> 16;

payload[6] = highByte(distance);
payload[7] = lowByte(distance);

```

Figura 5-15. Organización del arreglo con carga útil en los nodos finales.

Cuando esta información llega al servidor, los bytes están codificados en Hexadecimal y no son legibles de manera que para que sean útiles deben ser decodificadas; esta es una de las características que nos suministra TTN, nos permite escribir funciones JavaScript que sean comunes para decodificar los mensajes cuyo origen sean los dispositivos que están asociados a la aplicación esta funciones se llaman “payload formats”. Para poder decodificar el arreglo de bytes que estamos construyendo en los nodos, la función de decodificación queda de la siguiente manera.

```

function Decoder(bytes, port) {
    // Decode an uplink message from a buffer
    // (array) of bytes to an object of fields.
    var distance = ((bytes[6] << 8) | bytes[7])/10;
    var lat = (bytes[0] | bytes[1]<<8 | bytes[2]<<16 | (bytes[2] & 0x80 ? 0xFF<<24 : 0)) / 100000;
    var lng = (bytes[3] | bytes[4]<<8 | bytes[5]<<16 | (bytes[5] & 0x80 ? 0xFF<<24 : 0)) / 100000;

    return {
        distance: distance,
        location: {
            lat: lat,
            lng: lng
        }
    }
}

```

Figura 5-16 Función para decodificación de la carga útil recibida en el servidor.

De forma similar, TTN ofrece continuar procesando los datos luego de que estos son decodificados se tiene un conversor con la cual podemos hacer transformaciones de los datos, un validador para determinar si los datos recibidos tienen un nivel de integridad, cumple con los formatos y están dentro de los valores o rangos permitidos y, finalmente, dispone de una sección de codificación para los datos que se retornan a los nodos.

5.3 Prueba de recepción de datos:

Luego de haber configurado los dispositivos en el servidor, tanto el Gateway como los nodos finales, luego de configurar los datos de activación por mecanismo OTAA y de haber configurado las funciones de decodificación, veremos el flujo de datos desde los tres nodos hasta el servidor.

5.3.1 Pruebas con TTGO (01nodo):

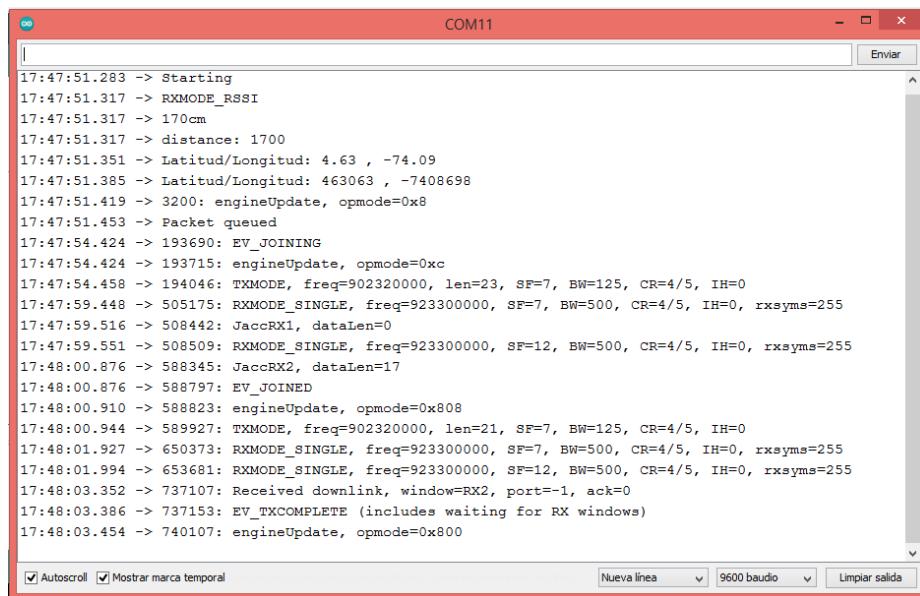


Figura 5-17. Inspección del registro del 01nodo usando el monitor serial.

Aquí podemos observar que luego de empaquetar el los datos obtenidos por los sensores de ubicación y proximidad en paquete, queda en espera para ser enviado; se puede ver el mensaje EV_JOINING que indica que el nodo inició una negociación con el servidor para establecer conexión; luego, 7 líneas más abajo se ve el mensaje EV_JOINED, lo cual indica que ya se estableció conexión con el servidor y 6 líneas más abajo se puede ver el mensaje EV_TXCOMPLETE, lo cual indica que el mensaje ha sido entregado y confirmado por el servidor. Si revisamos el flujo de datos de la aplicación en la interfaz del servidor podemos observar la recepción de los datos en el servidor (ver figura 5-18):

Applications > 01basurasbogota > Data						
time	counter	port	dev id:	payload	distance:	location.lat: location.lng:
▼ 17:49:04	0		dev id: 01nodo			
▲ 17:48:55	1	1	dev id: 01nodo	payload: D7 10 07 C6 F3 8E 01 40	distance: 32	location.lat: 4.63063 location.lng: -74.08698
▼ 17:48:01	0		dev id: 01nodo			
▲ 17:47:52	0	1	retry	dev id: 01nodo payload: D7 10 07 C6 F3 8E 06 A4	distance: 170	location.lat: 4.63063 location.lng: -74.08698
⚡ 17:47:54				dev id: 01nodo dev addr: 00-00-00-00-00-00 app eui: 70B5B57ED0025A08 dev eui: A0-A1-1D-B0-01-01		

Figura 5-18 Datos emitidos por el 01nodo en la aplicación TTN

Leyendo la secuencia de abajo hacia arriba, podemos observar que el servidor detectó que el dispositivo estaba encendido y realizó una solicitud de conexión enviando **dev addr**, **app eui** y **dev eui**, luego de establecer conexión se recibe un mensaje con carga útil codificada como D71007C6F38E06A4, y un mensaje de respuesta en el que no se está enviando ninguna información descendente, el mensaje se repite a las 17:48:55.

Inspeccionando el mensaje, podemos observar que la función de decodificación está operando correctamente pues se transforman los valores recibidos en hexadecimal y retorna los valores de distancia, latitud y longitud.

Applications > 01basurasbogota > Data						
time	counter	port	dev id:	payload	distance:	location.lat: location.lng:
▼ 17:48:01	0		dev id: 01nodo			
▲ 17:47:52	0	1	retry	dev id: 01nodo payload: D7 10 07 C6 F3 8E 06 A4	distance: 170	location.lat: 4.63063 location.lng: -74.08698
Uplink						
Payload						
D7 10 07 C6 F3 8E 06 A4						
Fields						
{ "distance": 170, "location": { "lat": 4.63063 , "lng": -74.08698 } }						

Figura 5-19 Salida de la función de decodificación en tiempo de ejecución.

5.3.2 Pruebas con Shield (Dragino):



```

18:08:35.729 -> 17207659: engineUpdate, opmode=0xc
18:08:35.776 -> 17207998: TXMODE, freq=902320000, len=23, SF=7, BW=125, CR=4/5, IH=0
18:08:40.754 -> 17520146: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsyms=255
18:08:40.849 -> 17523281: JaccRX1, dataLen=0
18:08:40.884 -> 17523357: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsyms=255
18:08:42.146 -> 17601725: JaccRX2, dataLen=17
18:08:42.180 -> 17602182: EV_JOINED
18:08:42.214 -> 17602213: engineUpdate, opmode=0x808
18:08:42.247 -> 17603703: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
18:08:43.194 -> 17664209: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsyms=255
18:08:43.288 -> 17666035: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsyms=255
18:08:44.666 -> 17749325: Invalid downlink, window=RX2
18:08:44.713 -> 17749361: EV_TXCOMPLETE (includes waiting for RX windows)
18:08:44.760 -> 17751483: engineUpdate, opmode=0x800

```

Figura 5-20 Inspección del registro del Dragino usando el monitor serial.

En la imagen se puede observar que luego de la conexión con el servidor la transmisión ha sido completada por el mensaje EV_TXCOMPLETE.



Figura 5-21 Datos emitidos por el Dragino en la aplicación TTN.

También se puede ver como los datos son recibidos y decodificados en el servidor.

5.3.3 Pruebas con Arduino y SX1276

La prueba se repitió con el tercer dispositivo y se obtuvieron los resultados esperados (ver figura 5-22).

```

18:23:44.152 -> Starting
18:23:45.791 -> RXMODE_RSSI
18:23:45.791 -> 142cm
18:23:45.791 -> distance: 1420
18:23:45.838 -> Latitud/Longitud: 4.64 , -74.09
18:23:45.838 -> Latitud/Longitud: 464448 , -7409277
18:23:45.884 -> 3216: engineUpdate, opmode=0x8
18:23:45.931 -> Packet queued
18:23:48.885 -> 193707: EV_JOINING
18:23:48.885 -> 193732: engineUpdate, opmode=0xc
18:23:48.932 -> 194062: TXMODE, freq=902320000, len=23, SF=7, BW=125, CR=4/5, IH=0
18:23:53.916 -> 505192: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
18:23:53.963 -> 508461: JaccRX1, dataLen=0
18:23:54.009 -> 508529: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255
18:23:55.341 -> 588803: JaccRX2, dataLen=17
18:23:55.341 -> 589255: EV_JOINED
18:23:55.341 -> 589281: engineUpdate, opmode=0x808
18:23:55.388 -> 590384: TXMODE, freq=902320000, len=21, SF=7, BW=125, CR=4/5, IH=0
18:23:56.376 -> 650895: RXMODE_SINGLE, freq=923300000, SF=7, BW=500, CR=4/5, IH=0, rxsysms=255
18:23:56.469 -> 654205: RXMODE_SINGLE, freq=923300000, SF=12, BW=500, CR=4/5, IH=0, rxsysms=255

```

Autoscroll Mostrar marca temporal Nueva línea 9600 baudio Limpiar salida

Figura 5-22 Inspección del registro del Arduino usando el monitor serial.

Applications > 01basurasbogota > Data						
▼ 18:24:58	0	dev id: arduino				
▲ 18:24:49	1	1	dev id: arduino	payload: 40 16 07 83 F1 8E 01 22	distance: 29	location.lat: 4.64448 location.lng: -74.09277
▼ 18:23:55	0		dev id: arduino			
▲ 18:23:47	0	1	dev id: arduino	payload: 40 16 07 83 F1 8E 05 8C	distance: 142	location.lat: 4.64448 location.lng: -74.09277
⚡ 18:23:49			dev id: arduino	dev addr: 00 02 20 05 app eui: 00 00 00 7E D0 02 5A 00 dev eui: 00 1D 10 78 11 AE 5B 1F		

Figura 5-23. Datos emitidos por el Arduino en la aplicación TTN.

5.4 Bróker MQTT

Tras recibir los datos al servidor IoT, se requiere de un mecanismo con el cual se comunique a las aplicaciones que usan de los datos, TTN soporta dos de los protocolos de la capa de comunicación MQTT y AMQP; sin embargo, para todas la aplicaciones TTN configura por defecto un bróker MQTT usando una implementación de Eclipse Mosquito, teniendo en cuenta que esto simplifica mucho el proceso de extracción de datos pues ya se cuenta con un bróker, hace falta solamente la implementación de un cliente para poder extraer los datos que se reciben en la aplicación. Para configurar un cliente bróker se realizaron los siguientes pasos:

- 1 se descargó e instaló mosquito de la página <https://mosquitto.org/download/>
2. se descargó el certificado de para añadir seguridad en la capa de transporte TLS <https://console.thethingsnetwork.org/mqtt-ca.pem> que ofrece TTN
3. se hizo una prueba para verificar que dispositivos estaban siendo activados en la aplicación para lo cual se usó la siguiente sentencia

```
mosquitto_sub -h us-west.thethings.network -t "+/devices/+events/activations" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnib_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
```

```
C:\Windows\System32\cmd.exe - mosquitto_sub -h us-west.thethings.network -t "+/devices/+events/activations" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnib_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Program Files\mosquitto>mosquitto_sub -h us-west.thethings.network -t "+/devices/+events/activations" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnib_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
01basurasbogota/devices/01nodo/events/activations {"app_eui":"70B3D57ED0025A63","dev_eui":"A840411BB8504151","dev_addr":26022CE4,"metadata":{"time":"2020-10-05T05:06:30.774Z4594Z","frequency":902.32,"modulation":"LORA","data_rate":"SF7BW125","airtime":25856000,"coding_rate":4/5,"gateways":[{"gtw_id":eui-a840411bb8504150,"timestamp":1913143898,"time":"2020-10-05T05:06:30.696Z","channel":0,"rssi":-93,"snr":7.8,"rf_chain":0}]}>
```

Figura 5-24. Datos recibidos tema de dispositivos en la app 01basurasbogota.

4. se hizo la suscripción al canal de los mensajes que ingresan a la aplicación en modo ascendente

```
mosquitto_sub -h us-west.thethings.network -t "+/devices/+/up" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnB_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
```

5. se compara la información de los mensajes que llegaron al servidor IoT y los que se encuentran en el Broker y se valida la correspondencia.

time	counter	port	dev id:	payload	distance:	location.lat:	location.lng:
24:11:09	0		01nodo				
24:11:00	2	1	01nodo	D7 10 07 C6 F3 8E 01 0E	27	4.63063	-74.08698
24:10:04	0		01nodo				
24:09:54	1	1	01nodo	D7 10 07 C6 F3 8E 06 86	167	4.63063	-74.08698

Figura 5-25. Mensajes recibidos en el servidor para la app 01basurasbogota.

```
C:\Windows\System32\cmd.exe - mosquitto_sub -h us-west.thethings.network -t "+/devices/+/up" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnB_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Program Files\mosquitto>mosquitto_sub -h us-west.thethings.network -t "+/devices/+/up" -u "01basurasbogota" -P "ttn-account-v2.RmXvJu3XZ3e1S3B9cpFGwnBTBrCSVnB_vMJEud3cA" -v --cafile mqtt-ca.pem -p 8883
01basurasbogota/devices/01nodo/up {"app_id": "01basurasbogota", "dev_id": "01nodo", "hardware_serial": "A840411BB8504151", "port": 1, "counter": 1, "payload_raw": "1xAHxv0OBoY=", "payload_fields": {"distance": 167, "location": {"lat": 4.63063, "lng": -74.08698}}, "metadata": {"time": "2020-10-05T05:09:54.726794639Z", "frequency": 902.32, "modulation": "LORA", "data_rate": "SF7BW125", "airtime": 61696000, "coding_rate": "4/5", "gateways": [{"gtw_id": "eui-a840411bb8504150", "timestamp": 2117099290, "time": "2020-10-05T05:09:54.651866Z", "channel": 0, "rssi": -93, "snr": 7.8, "rf_chain": 0, "latitude": 4.63475, "longitude": -74.09171}], "channel": 0}, "rssi": -93, "snr": 7.8, "rf_chain": 0, "latitude": 4.63475, "longitude": -74.09171}], "01basurasbogota/devices/01nodo/up {"app_id": "01basurasbogota", "dev_id": "01nodo", "hardware_serial": "A840411BB8504151", "port": 1, "counter": 2, "payload_raw": "1xAHxv0OBoQ4=", "payload_fields": {"distance": 27, "location": {"lat": 4.63063, "lng": -74.08698}}, "metadata": {"time": "2020-10-05T05:11:00.072213104Z", "frequency": 902.32, "modulation": "LORA", "data_rate": "SF7BW125", "airtime": 56576000, "coding_rate": "4/5", "gateways": [{"gtw_id": "eui-a840411bb8504150", "timestamp": 2182444462, "time": "2020-10-05T05:10:59.997044Z", "channel": 0, "rssi": -90, "snr": 7.8, "rf_chain": 0, "latitude": 4.63475, "longitude": -74.09171}]}]
```

Figura 5-26. Datos recibidos por mensajes ascendentes en la app 01basurasbogota.

5.5 Prueba Cliente MQTT

Este ejercicio fue suficiente para determinar que la información se podía extraer del servidor por medio de un cliente MQTT; sin embargo, se requiere una implementación más flexible que nos permita hacer un pos-procesamiento automático de los datos.

La comunidad de TTN también creó un paquete de desarrollo para embeber el cliente en una aplicación Java el código fuente es de libre acceso y se encuentra publicado en el un repositorio en github <https://github.com/TheThingsNetwork/java-app-sdk> El repositorio contiene una implementación del protocolo MQTT y una para el protocolo AMQP.

Se hizo una copia del kit de desarrollo ofrecido por TTN, se generó un cliente que se suscribe al canal de mensajes ascendentes, que contienen los datos de nuestra aplicación. En la función de recepción de mensajes se invoca una función pos procesamiento que se encarga de extraer la hora de la medición el dispositivo, y los valores que fueron retornados por la función de decodificación, usa estos valores y el tipo de contenedor al que está enlazado el sensor para estimar el nivel de llenado y persiste los datos en la base de datos de nuestra aplicación.

```
client.onMessage((String devId, DataMessage data) -> processData(devId, data));
```

Figura 5-27. Función SDK TTN que se suscribe a los mensajes ascendentes de la aplicación mediante mosquitto.

```
private static void processData(String devId, DataMessage data) {
    System.out.println("Message: " + devId + " " + ((UplinkMessage) data).getPayloadFields());
    Double distance = (Double) ((UplinkMessage) data).getPayloadFields().get("distance");
    Object location = (Object) ((UplinkMessage) data).getPayloadFields().get("location");

    System.out.println("location: " + location.toString());
    PostgreSQLConnection dao;
    try {
        dao = new PostgreSQLConnection();
        Dumpster dumpster;
        MeasureDumpster measureDump;
        dumpster = dao.searchDevID(devId);
        int idDumpsterMeasure = dao.findLastID();
        idDumpsterMeasure++;
        if (dumpster!=null) {
            measureDump = new MeasureDumpster();
            measureDump.setId(idDumpsterMeasure);
            measureDump.setIdDumpster(dumpster.getId());
            Integer level = utilities.caltulateLevel(distance, dumpster.getIdDumpsterType());
            measureDump.setLevel(level);
            measureDump.setMeasureDate(new Date());
            measureDump.computePriority();
            dao.saveMeasureDumpster(measureDump);
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Figura 5-28. Función pos procesamiento de datos.

Esta aplicación se compiló como una aplicación java estática. Luego de conectar los nodos finales y tener el *gateway* escuchando, se obtuvieron los siguientes resultados (ver figura 5-29):

time	counter	port	dev id	payload	distance	location.lat	location.lng
21:37:00		0	dragino				
21:36:59	6	1	dragino	07 14 07 34 F8 8E 01 5E	25.7	4.63879	-74.0756
21:36:42		0	O1nodo				
21:36:40	7	1	O1nodo	D7 10 07 C6 F3 8E 06 54	154.2	4.63063	-74.086
21:36:20		0	arduino				
21:36:18	6	1	arduino	40 16 07 83 F1 8E 06 A4	154.2	4.64448	-74.092

Figura 5-29 Registro de datos de la aplicación TTN.

Los datos que se estaban obteniendo en el servidor IoT estaban siendo recuperados con apenas un par de segundos de retraso por la implementación del cliente MQTT que se creó en java, los mensajes ascendentes que transmite el bróker son muy complejos, pero para simplificar la lectura fueron reducidos a un mensaje que contiene la siguiente estructura:

Identificador del dispositivo {distance = medición Distancia, location = {lat= coordenada Latitud, lon= coordenada Longitud}}.

```
App [Java Application] C:\Program Files\Java\jdk1.8.0_73\bin\javaw.exe (23/09/2020, 9:36:15 p. m.)
|connected !
Message: arduino {distance=154.2, location={lat=4.64448, lng=-74.09277}}
location: {lat=4.64448, lng=-74.09277}
Dumpster{id=28255, location=null, idDumpsterType=1, idPhysicalState=1, idvia=82796, deviceID=arduino}
Message: O1nodo {distance=154.2, location={lat=4.63063, lng=-74.08698}}
location: {lat=4.63063, lng=-74.08698}
Dumpster{id=28253, location=null, idDumpsterType=2, idPhysicalState=1, idvia=102470, deviceID=O1nodo}
Message: dragino {distance=25.7, location={lat=4.63879, lng=-74.07564}}
location: {lat=4.63879, lng=-74.07564}
Dumpster{id=28254, location=null, idDumpsterType=1, idPhysicalState=2, idvia=14065, deviceID=dragino}
Message: arduino {distance=154.2, location={lat=4.64448, lng=-74.09277}}
location: {lat=4.64448, lng=-74.09277}
Dumpster{id=28255, location=null, idDumpsterType=1, idPhysicalState=1, idvia=82796, deviceID=arduino}
Message: O1nodo {distance=154.2, location={lat=4.63063, lng=-74.08698}}
location: {lat=4.63063, lng=-74.08698}
Dumpster{id=28253, location=null, idDumpsterType=2, idPhysicalState=1, idvia=102470, deviceID=O1nodo}
Message: dragino {distance=25.7, location={lat=4.63879, lng=-74.07564}}
location: {lat=4.63879, lng=-74.07564}
Dumpster{id=28254, location=null, idDumpsterType=1, idPhysicalState=2, idvia=14065, deviceID=dragino}
```

Figura 5-30. Registro de datos del cliente MQTT en java.

Estos datos serán un recurso valioso que se usará en las aplicaciones de enrutamiento a las que nos referiremos en los próximos capítulos, por esta razón fue necesario almacenar estos datos en un entorno desde el cual fuera fácil de acceder desde dichas aplicaciones, se decidió que los datos reposen en una base de datos por la seguridad que ofrece y la capacidad que tiene para integrarse con otros sistemas.

Los registros quedaron almacenados en una tabla con la siguiente estructura

	123 id	123 id_dumpster	123 level	measure_date	123 priority	abc device_id
1	96,168	28,253 ↗	0	2020-09-23 21:38:53	5 ↗	01nodo
2	96,169	28,254 ↗	0	2020-09-23 21:39:11	5 ↗	dragino
3	96,178	28,255 ↗	0	2020-09-23 22:16:39	5 ↗	arduino

Figura 5-31. Consulta de los últimos registros recibidos en la tabla de mediciones.

5.6 Base de Datos

El motor de base de datos que se eligió fue postres SQL, que es una base de datos relacional con un lenguaje de consulta estructurado, pero que opera con una licencia libre que establece lo siguiente:

“permiso para usar, copiar, modificar y distribuir este software y su documentación para cualquier propósito, sin pago de cuotas” (postgreSQL ORG, 2021)

Pero además de la licencia, una de las características que nos interesan de Postgres es el complemento para manejo de datos espaciales postGIS, dado que en nuestro sistema queremos poder referencias ubicaciones de las calles por las que los camiones recolectores deben hacer sus recorridos, dependiendo de la ubicación de los contenedores, lo cual será tema del capítulo 6. Por lo pronto, se desea mostrar la estructura de datos que se está usando para almacenar los datos de los sensores.

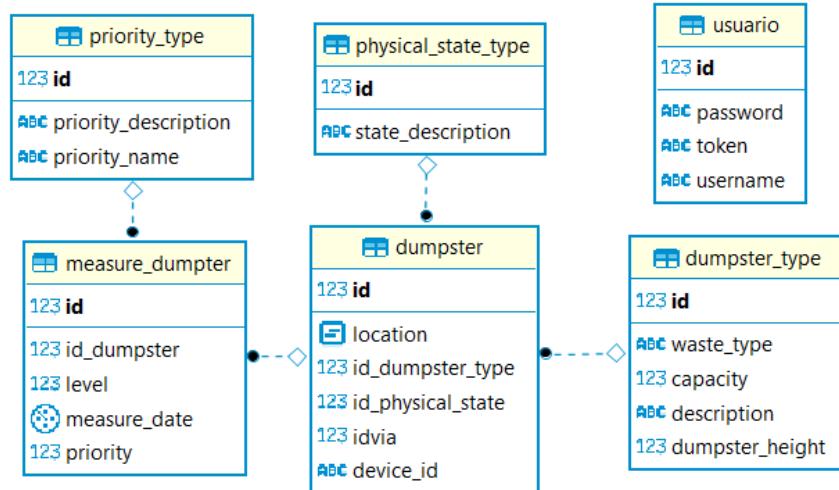


Figura 5-32 Modelo relacional del segmento de BD relacionado con contenedores y mediciones, generado usando DBeaver.

La figura 5-33 muestra seis tablas de las cuales se hace una breve descripción:

Tipo de contenedor: (dumpster_type) en la cual se caracterizan los tipos de contenedores que dispone la ciudad y que varían según su capacidad y el tipo de residuo al que están destinados.

Estado físico: (physical state) que caracteriza cuál es la condición del contenedor, indicando si se encuentra en buen estado, si requiere mantenimiento o está dañado y debe ser reemplazado.

Contenedor: (dumpster) es el registro correspondiente a la virtualización de un contenedor físico, en el cual se registra su tipo, su estado, su ubicación, la vía sobre la cual está ubicado y el identificador del nodo de medición que tiene asociado y que se registró en el servidor IoT.

Tipo de prioridad: (priority_type) en esta tabla se caracterizan las urgencias de atender un contenedor con base a su nivel de llenado.

Medición de contenedor: (measure dumpster) en esta tabla persisten las mediciones registradas para cada contenedor, asociando el contenedor al que le pertenece la medición, el instante en el que se reportó dicha medición, el nivel de llenado reportado y la prioridad asociada a ese nivel.

5.7 Visualización de los datos obtenidos.

Para que los datos recogidos tengan un significado más claro, se ha creado una aplicación básica, que consulta los valores de las mediciones más recientes y usa las coordenadas de los contenedores asociados a los nodos que recogieron esas mediciones, estos datos se muestran en interfaz gráfica que usa el API de Google maps para representar en un mapa la ubicación de los contenedores y su nivel de llenado.

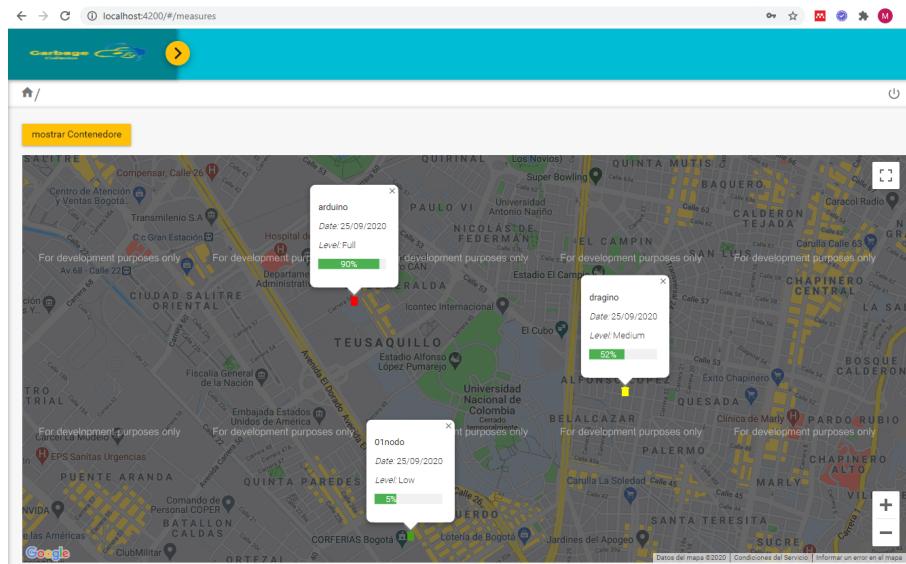


Figura 5-33 Interfaz web representación geográfica de los contendores y las mediciones más recientes.

Se debe aclarar que en esta prueba los únicos datos reales son los que reportó el 01nodo, el cual está integrado con el módulo GPS, las coordenadas geográficas de los otros 2 dispositivos son datos presuntivos, que se colocaron de manera aleatoria para poder visualizar los contendores en el mapa pero que no reflejan su ubicación real. Sin embargo, con esta misma estrategia se puede registrar la ubicación de los nodos dependiendo de las coordenadas del contenedor, en el caso que se decida no integrar el sensor GPS en los nodos.

5.8 Alcance de la señal

Para medir el alcance de la señal de los dispositivos, se conectó el *gateway* en una estación fija y se iniciaron los dispositivos de medición, usando para su alimentación una fuente de energía alterna, luego de recibir la señal del dispositivo de medición se fue separando progresivamente de la ubicación del *gateway* para verificar cual era la distancia de alcance máximo para la cual el *gateway* seguía recibiendo los datos del dispositivo.

Arduino

Para la primera prueba se usó el dispositivo ensamblado sobre el arduino, el *gateway* se conectó en una vivienda próxima a las instalaciones de la universidad, que corresponde con la ubicación marcada con el puntero verde sobre el mapa en la figura 5-34, los marcadores rojos indican los lugares desde los cuales se recibieron las coordenadas geográficas captadas por el sensor de GPS en el dispositivo de medición.

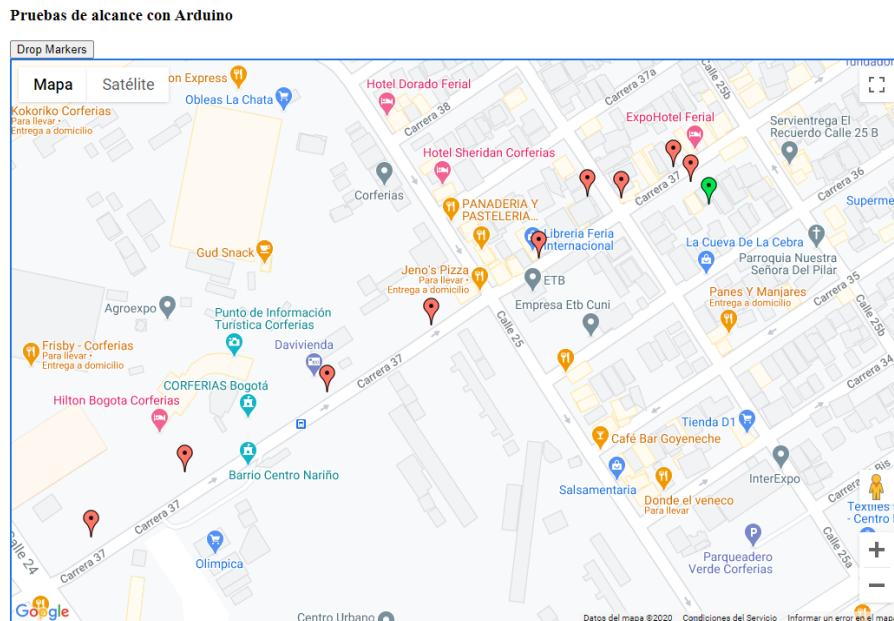


Figura 5-34. Coordenadas reportadas por el arduino en pruebas de alcance.

Una de las primeras observaciones que se realizó es que cuando se giraba por una de las calles y quedaba un muro entre el Gateway y el dispositivo de medición no se recibían los mensajes lo cual indica los muros interrumpen el paso de la señal. Sin embargo, mientras se tuvo una línea de visión directa entre el gateway y la señal fue constante. En la primera prueba se logró una cobertura aproximada de 460 metros lineales, como lo muestra la figura 5-35.

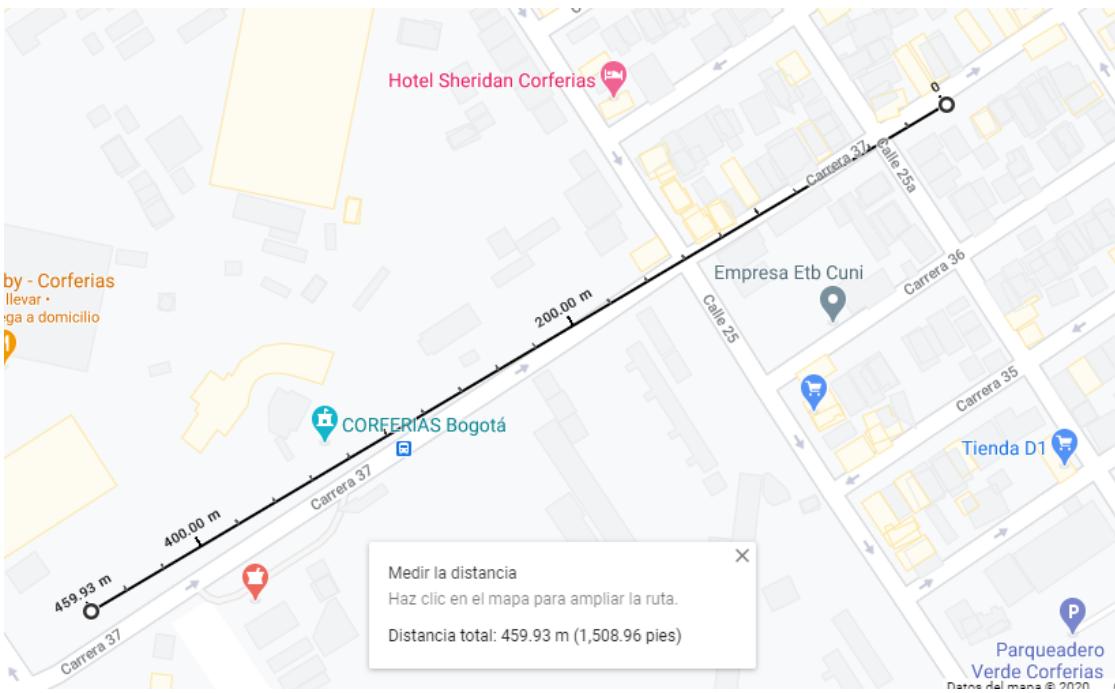


Figura 5-35 Distancia aproximada de alcance usando el circuito de arduino.

De estos resultados se puede afirmar que es una distancia corta, teniendo en cuenta que esta modulación está pensada para viajar distancias superiores a los 10 Km.

Dragino

Teniendo en cuenta los resultados de las primeras pruebas y de la interferencia de los muros se decidió buscar una región que tuviera una baja densidad de edificios, y se buscó ubicar el gateway en un lugar alto por encima del promedio de los edificios de la zona para poder evaluar el alcance del dispositivo con una línea de visión directa. Para ello se ubicó en el piso 18 de uno de los edificios aledaños a la universidad nacional.

Esta vez se usó el dispositivo Dragino, que trae incorporada una antena directiva tipo ápex y se realizó una operación similar alejando el dispositivo del gateway progresivamente para evaluar cuál sería la distancia máxima obtenida.



Figura 5-36. Imagen de Dragino emitiendo señal y antena apuntando en dirección al Gateway ubicado en el edificio que se ve en la parte posterior.

Pruebas de alcance con Dragino

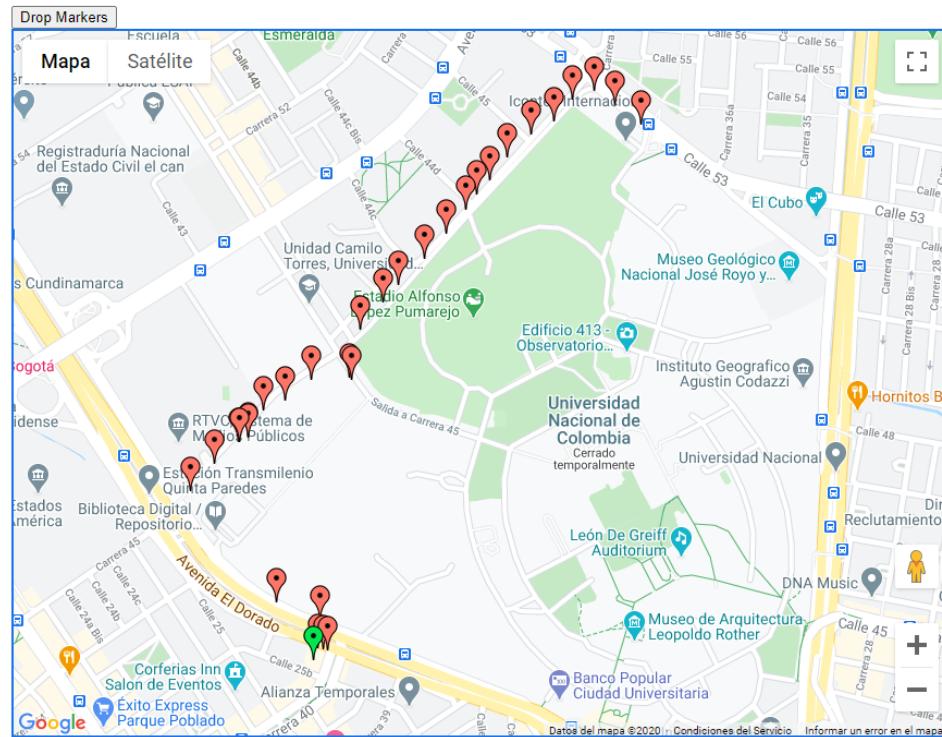


Figura 5-37. Coordenadas reportadas por Dragino en pruebas de alcance.

Se realizó un recorrido por la parte externa de la Universidad Nacional de Colombia sede Bogotá sobre la carrera 45 y se obtuvo reporte de las coordenadas del sensor GPS en todas las ubicaciones marcadas con punteros rojos. El último de los datos se obtuvo sobre la calle 53, sitio donde el dispositivo se apagó pues la batería se agotó, no obstante en este caso el alcance de la señal mejoró sustancialmente alcanzando una distancia máxima de 1380 metros aproximadamente, como se puede observar en la figura 5-38.



Figura 5-38 Distancia aproximada de alcance usando el circuito Dragino.

Con el TTGO no se realizaron pruebas, debido a la que la frecuencia de transmisión del módulo que trae integrado, es diferente a la del *gateway* por lo cual probablemente no tendrá un rango de alcance superior a los obtenidos anteriormente adicionalmente su circuito no está diseñado para adaptar una batería externa, lo cual dificulta acoplar un fuente de alimentación, para hacerlo portable.

Algunas conclusiones de la prueba son: los *gateways* deben ser ubicados en lugares altos, por encima la altura promedio de los edificios de la zona, para que exista una línea de visión directa entre ellos y los dispositivos de medición, de manera tal que se aumenten la cobertura de servicio. Los dispositivos no deben ser dotados con sensores GPS, pues no solo aumentan el costo del dispositivo, sino que también consumen demasiada energía. En lugar de ellos se puede indexar directamente su coordenada geográfica en la base de datos en el momento en que se instalan en los contenedores de residuo.

6. Simulación de Datos

Uno de los problemas que se enfrentaron durante la formulación de este sistema, fue la carencia de información relacionada con la producción de basuras en Bogotá, si esta información existe, no es de acceso público y es custodiada por las entidades que administran el sistema de recolección actual de basuras UAESP¹⁴, que son empresas privadas; por otra parte, la información que tenemos se limita a los tres nodos físicos que se implementaron y se escribieron en el capítulo 4 y que no han sido instalados a contendores reales, por lo tanto, no disponemos de información suficiente; es decir múltiples contenedores que reporten datos para calcular una ruta, lo cual nos impide evaluar el desempeño del sistema en una región más amplia.

La simulación es un instrumento que permite recrear de forma aproximada el comportamiento de un fenómeno o proceso del mundo real y crear un modelo para conducir experimentos comúnmente numéricos, que nos permitan entender este comportamiento, dada una serie de condiciones y generar estrategias para obtener mejores resultados del procedimiento o fenómeno que se estudia (Cabrera, 2010).

La simulación que se propone consiste en usar las ubicaciones de los contenedores reales y asumir que cada uno de ellos tiene instalado un nodo de medición, que se encuentran en el rango de cobertura de un *gateway*, conformando así una RIS que reporta el nivel de llenado de dichos contenedores una vez al día, con el fin de usar esta información poder planificar la operación de recolección. La región de estudio ha sido limitada a una sola de las localidades de Bogotá, porque la manipulación de información geográfica es costosa en términos de procesamiento; sin embargo, esta prueba de concepto en una zona limitada nos permite definir la validez del sistema en una región más amplia dado que la estrategia es replicable de forma sistemática a las demás localidades de la ciudad, siempre y cuando disponga de la información de ubicación de los contenedores.

Se eligió la Localidad de Engativá pues junto con la localidad de Barrios Unidos acumulan el 28,68% de todos los contenedores de acuerdo con información del Bogotá Limpia.

¹⁴ Unidades administradoras especiales de servicios públicos.

“Se tiene prevista la ubicación de 2.868 contenedores en dos cuadrantes. El primero irá desde la avenida Boyacá hasta la carrera 96, entre la calle 80 y el humedal Juan Amarillo. El segundo será desde la avenida Boyacá hasta el río Bogotá, entre la calle 80 y la calle 72. El operador Área Limpia anunció que instalará 2.726 en Suba.”

(El Espectador, 2020)

“Para tener una idea del impacto, solo en localidades como Engativá se van a ubicar 1.434 cajones como estos para residuos no aprovechables, de 2.400 litros, y otros 1.434 para residuos aprovechables con una capacidad de 1.100 litros. Es decir que el consorcio Bogotá Limpia instalará 2.868 contenedores solo en esa zona, una de las que más residuos produce.” (Parra, 2018)

6.1 Conjunto de datos

Debido a que no fue posible encontrar información referente a la recolección de RSU en Bogotá, dada la carencia de registros de acceso público, se optó por analizar un conjunto de datos de una ciudad que tuviese algunas características similares a las de Bogotá como: su densidad poblacional, la distribución de calles y esquemas de recolección de basuras, lo cual resultó ser complejo dado que cada ciudad es única y que exista una coincidencia para los aspectos mencionados entre dos ciudades es improbable.

Sin embargo en Kaggle, que es uno de los repositorios de datos más grandes que existe en Internet, se encontró un conjunto de datos generado por la gobernación de Austin Texas, USA que fue publicado para uso general, contiene información de los lugares de descarga de RSU, con identificaciones, tipos y pesos de las cargas, las fechas y descripción de las rutas en las que se recolectaron estos residuos entre 2008 y 2016. La información del conjunto de datos indica donde se genera el RSU, quien se hace cargo de él y a dónde va.

De acuerdo con el repositorio de datos, esta información fue recolectada con el ánimo de responder a las siguientes preguntas:

- ¿Que tanto RSU está generando la ciudad de Austin Texas?
- ¿Cuáles son las rutas que mayor cantidad de RSU producen? y ¿quiénes hacen un mejor proceso de reciclaje?
- ¿existen cambios o comportamientos periódicos?
- Intentar predecir las rutas de recolección a partir de los datos históricos.

Estas preguntas atrajeron bastante nuestra atención ya que giran en torno a aspectos muy relacionados con los que se pretenden modelar en nuestra simulación y tienen una motivación en común; tratar de predecir las rutas de recolección; por esta razón, se eligió este conjunto de datos para ser analizado.

Los datos están organizados en dos archivos separados por comas llamados rutas y desviación de cargas, los cuales están asociadas por una relación de uno a muchos que indica que para una misma ruta puede haber varios registros de recolección de residuos, la tabla de desviación de cargas contiene 564,403 registros.

routes		loads	
LANDFILL	varchar	dropoff_site	varchar
the_geom	varchar	load_id	int
GARB_RT	varchar	load_time	date
GARB_DAY	varchar	load_type	varchar
GARB_SUP	varchar	load_weight	int
SUPER_NUM	int	report_date	date
OP_TYPE	varchar	route_number	varchar
RT_OLD	varchar	route_type	varchar

Figura 6-1 Estructura del conjunto de datos.

Para la manipulación de datos se decidió hacer uso de Python, debido a las múltiples librerías que ofrece para el manejo de datos como Pandas, que permiten trabajar con estructuras de llamadas DataFrames que son similares a las tablas y sobre las cuales se pueden ejecutar operaciones sobre todos los elementos de una columna y operaciones entre las mismas columnas con relativa simplicidad, además de las funciones reconstruidas que generan análisis estadísticos rápidos sobre los datos agregados.

6.2 Depuración de datos

Distancia:

Como primer punto teniendo en cuenta que los aspectos geográficos son relevantes para nuestro modelo de datos pero las coordenadas geográficas no nos dan mucha información relacionada con la generación de RSU, se creó una nueva columna en la tabla de rutas transformando las coordenadas de la ruta en una distancia, para ello usamos la fórmula de Haversine que nos permite hallar la distancia en la superficie de una esfera.

$$\text{haversin}\left(\frac{d}{R}\right) = \text{haversin}(\varphi_1 - \varphi_2) + \cos \cos (\varphi_1) \cos \cos (\varphi_2) \text{haversin} (\Delta\lambda)$$

De la cual, haciendo algunas transformaciones matemáticas para despejar la distancia, se obtiene la siguiente fórmula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\lambda_2 - \frac{\lambda_1}{2}\right)}\right)$$

Esta función se codificó usando python de la siguiente manera

```
# distance function implements haversine formula to find distance
# between 2 coordinates locations
import math
from decimal import Decimal

def distance(origin, destination):
    lat1, lon1 = origin
    lat2, lon2 = destination
    radius = 6371 # km
    dlat = math.radians(lat2-lat1)
    dlon = math.radians(lon2-lon1)

    a = math.sin(dlat/2) * math.sin(dlat/2) + math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) * math.sin(dlon/2) * math.sin(dlon/2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = radius * c
    return d
```

Figura 6-2 Codificación de la función para hallar la distancia en una superficie esférica.

Reducción de Variables: Algunos de los datos que contiene el conjunto de datos **no** son relevantes para encontrar tendencias en la generación de RSU, entre ellos el nombre y código del supervisor por esta razón nos deshicimos de estas columnas, además existían dos columnas muy similares llamadas tipo de carga (*load_type*) y tipo de ruta (*route_type*), por lo cual se decidió remover *route_type*; finalmente, tras calcular las distancias, se eliminó la columna con la información espacial “*the_geom*”, pues no sería usada para el resto del análisis.

Unión de los Data-Frames: Para poder analizar los datos en conjunto, fue necesario unir las tablas usando una columna en común las cuales eran (*garb_rt* y *route_number*) en este paso hubo una leve reducción de registro debido a que no había plena integridad entre las rutas de algunas de las cargas y, por lo tanto, estas columnas fueron eliminadas.

Transformación de tipos: Dado que se quieren ejecutar algunos cálculos matemáticos sobre las columnas para obtener mediciones estadísticas, tales como la media, la desviación estándar, los valores máximos y mínimo, se convirtieron las columnas peso de la carga (load_weight) y distancia de la ruta (route_distance) a valores numéricos.

Llenado de valores nulos: algunos de los registros estaban incompletos especialmente en las columnas peso de la carga (load_weight) y distancia de la ruta (route_distance); para poder poblar los datos faltantes se agruparon estas dos columnas por la ruta a la que pertenecían y, para cada uno de los grupos, calculamos el promedio, luego reemplazamos el valor faltante por valor promedio, de acuerdo al grupo o la ruta a la que pertenecía ese peso o esa distancia.

Normalización de valores: por último, se deseaba tener valores numéricos que estuviesen en rangos similares, para que fuera más simple realizar comparaciones entre ellos, de forma que los valores muy grandes no generan sesgos en columnas en los que los valores eran pequeños en comparación, por esta razón, hicimos una normalización de variables usando la función Z Score de Scipy la cual calcula

$$z = \frac{x - \mu}{\delta}$$

en donde X es el valor a normalizar, μ es la media y δ es la desviación estándar.

6.3 Análisis exploratorio

Antes de iniciar una análisis estadístico profundo es conveniente establecer cuáles son las rutas que muestran un desempeño óptimo, o dicho de otra manera en cuáles de las rutas se recolecta una mayor cantidad de residuo recorriendo distancias más cortas, pues sería conveniente que las demás rutas tengan un comportamiento similar, de manera que las variables más relevantes a considerar son: pesos de la carga y distancias de las rutas, tomando esto como punto de partida se extrajeron medidas centrales generales y de desviación, se obtuvieron los resultados que se muestran a continuación.

Distancias de ruta

```
dfRoutesVsDistances['RT_DST'].describe()

count    122950.000000
mean      8.518411
std       6.052477
min       2.620150
25%       4.887549
50%       6.264389
75%       9.914222
max      43.805445
Name: RT_DST, dtype: float64
```

Figura 6-3. Resumen distancias por ruta.

Este resumen muestra una media de rutas de recolección de 8.5 Km, la ruta más larga tiene una longitud de 43.8 Km, lo que probablemente es un valor aislado; para que estos valores no nos induzcan a errores, removemos todos los recorridos cuya distancias superen los 35 Km. Otro factor inesperado es que la desviación estándar es muy alta alcanzando un valor cercano a la media este genera una XCDS a pensar que las distancias no obedecen a una distribución gaussiana.

Pesos de la carga

Para analizar las cargas en lugar de realizar las mediciones sobre los pesos en general, se agruparon las rutas de manera que pudiéramos ver los promedios y medidas por cada ruta. Lo cual nos dará un mejor panorama para determinar si el área de recolección tiene alguna influencia en el volumen del RSU generado.

```
dfAvgroute.describe()

count    184.000000
mean     18654.136201
std      1667.942316
min      14939.262295
25%      17518.058773
50%      18448.600383
75%      19749.847568
max      23295.255875
Name: load_weight, dtype: float64
```

Figura 6-4 Resumen de los pesos de la carga por ruta.

Comparando los pesos de las rutas, vemos que la media que se acerca a los 18,654 litros mientras que el promedio mínimo de una de las rutas que es de 14,936 vemos que

la diferencia es de 3,715 litros, por lo cual esperaríamos tener una distancia similar hacia dirección opuesta, es decir que tuviésemos un valor máximo de 22,369 litros pero en realidad el valor promedio en la ruta máxima es de 23295, lo cual no es muy habitual, pero para validar el comportamiento se graficaron estos promedios en un histograma de frecuencias.

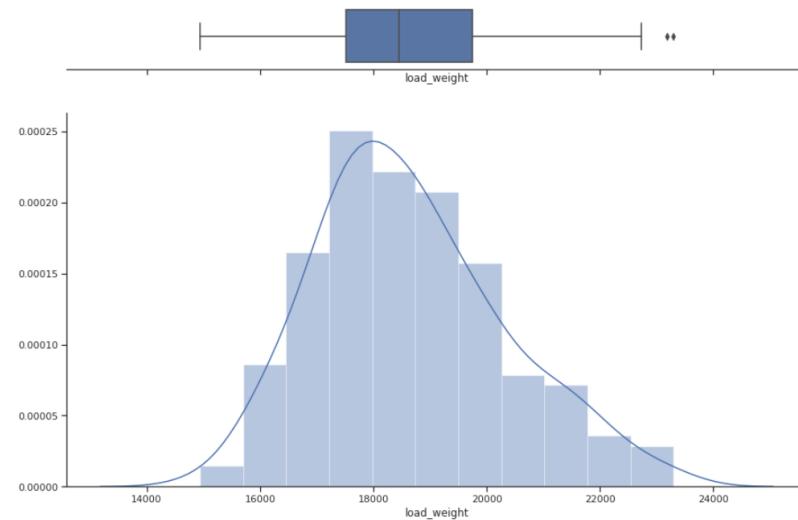


Figura 6-5 Histograma general de cargas.

Este histograma demuestra que los datos están densamente concentrados en la región correspondiente al cuartil del 25% de manera que podemos afirmar que la distribución de la generación de cargas está sesgada a la izquierda, la gráfica de caja que se encuentra en la parte superior del histograma nos ayuda a visualizar mucho mejor la diferencia entre la media y los cuartiles.

El histograma de las cargas agrupadas por ruta posiblemente muestre algunos patrones de comportamientos referentes a la generación de residuos para entender la causa de la alta desviación que obtuvimos del gráfico mostrado en la Figura 6-5.

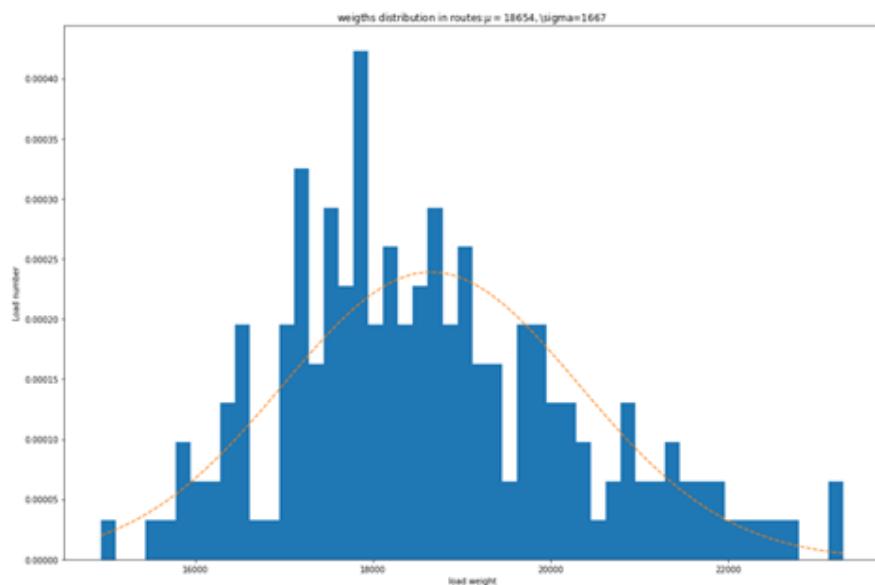


Figura 6-6 Histograma general de cargas.

En la gráfica anterior se usó una línea que describe una distribución normal para observar por una parte que tan segada se encuentra la información y por otra para confirmar que los valores por debajo de los 22,000 litros y por encima de los 35,000 son datos aislados o fuera de rango.

Una de las hipótesis que se hicieron a priori al obtener la información, fue que existía una relación directa entre el volumen de las cargas y las distancias de las rutas , lo que se pensaba es que entre más larga fuese la ruta mayor cantidad de residuos debía recolectar de manera que organizamos las ruta de forma ascendente según sus distancias (línea roja) y las pusimos junto al valor normalizado de los pesos promedios de las cargas en estas rutas (línea azul); se pensaba que existía una correlación, sin embargo, la gráfica demostró que no es así pues mientras las distancias están continuamente en aumento, las cargas se mantienen oscilando en los mismo rangos.

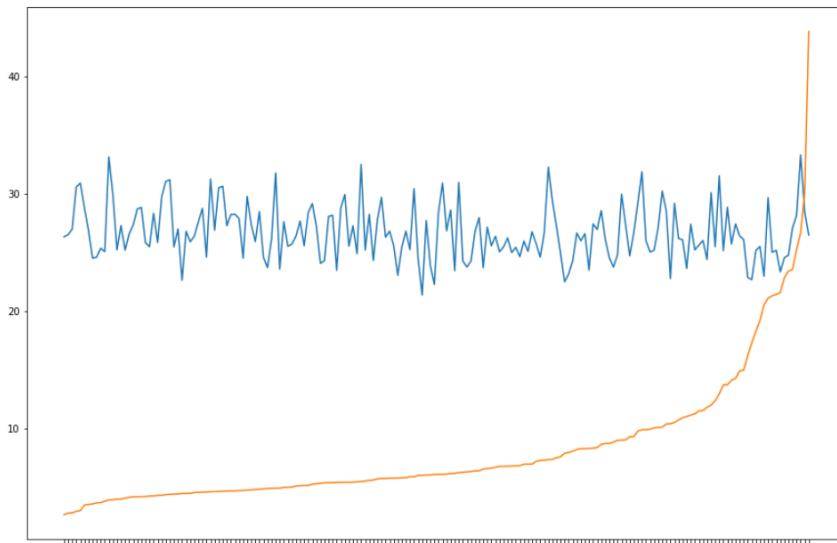


Figura 6-7. Relación entre las longitudes de las rutas (Km) y el volumen promedio recolectado en la misma (L).

Otro de los análisis que se hicieron fue la agrupación del promedio de volumen de las cargas por años, de donde surgió un patrón interesante: en los últimos años la cantidad de RSU ha ido en constante aumento, teniendo un aumento drástico en 2008 y continúa creciendo esto que indica que no estamos haciendo un uso responsable del residuos y es preocupante si se consideran los efectos ambientales que produce este tipo de contaminación, afectando negativamente el cambio climático.

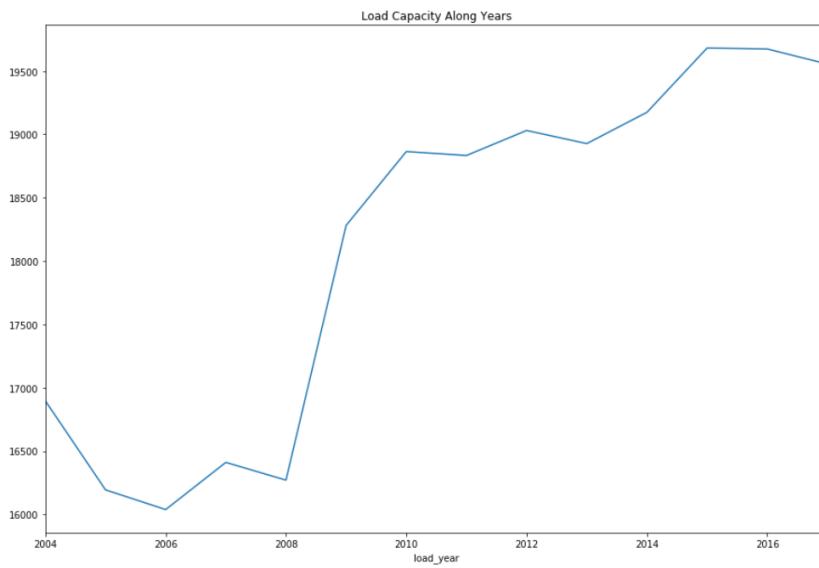


Figura 6-8. Evolución en la generación de Residuos en la ciudad de Austin entre 2004 y 2017.

6.4 Representación geográfica de las rutas

Buscando otros patrones se recurrió a la información geográfica con la esperanza de encontrar alguna relación entre las ubicaciones espaciales de las rutas y la producción de desechos, posiblemente exista una mayor generación de residuos en áreas residenciales o industriales. Para poder evaluarlo es necesario calcular una nueva variable que nos indique qué tan distante se encuentra la ruta de lugares de interés como: el centro geográfico de la ciudad, zonas de producción y zonas residenciales.

Inicialmente se realizó un análisis visual, graficando en un mapa algunas de las rutas anteriormente clasificadas, las rutas que mostraron un rendimiento bueno se dibujaron de color verde, las de rendimiento normal en color amarillo y las de rendimiento bajo en color rojo, esto con el fin de comparar e identificar si existe alguna clase de distribución o concentración geográfica, para no saturar el mapa de líneas lo que puede hacer que la representación sea difícil de interpretar se tomaron solo 10 rutas de cada tipo.

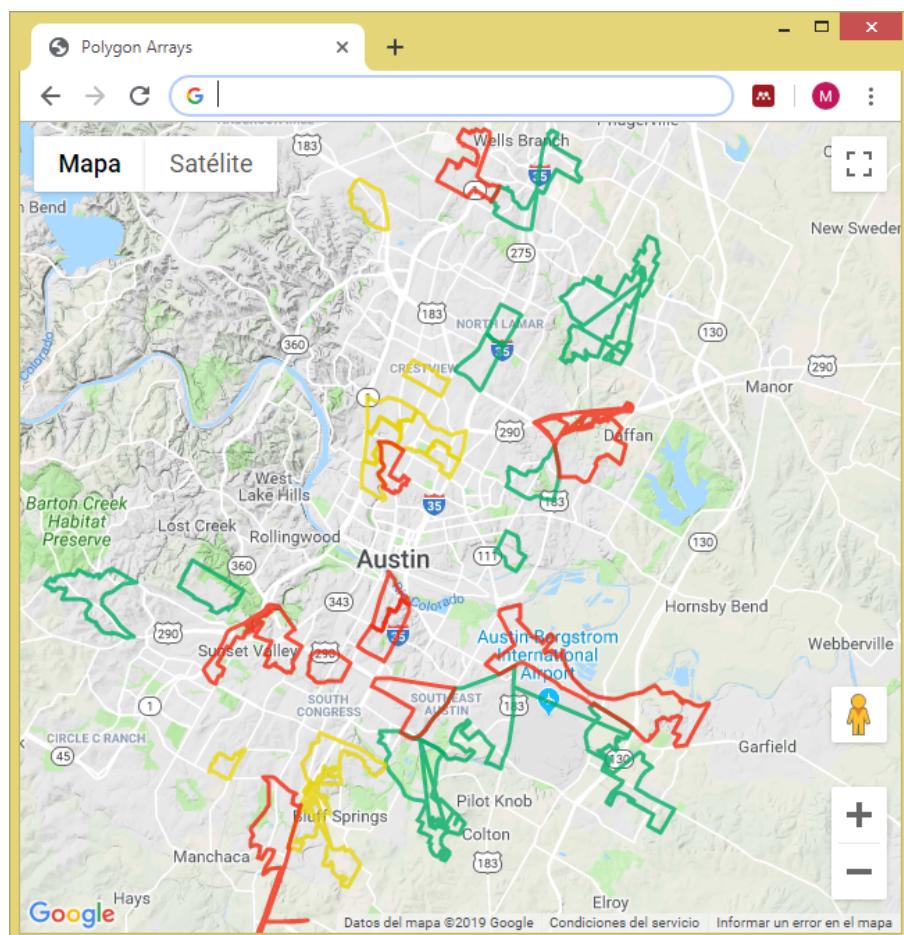


Figura 6-9 Mapa con rutas clasificadas según el promedio de carga recolectado.

En la visualización se pudo observar que existe un conjunto de rutas con comportamientos similares, un desempeño normal, en la parte central del mapa, pero para las rutas verdes y rojas aunque se variaron las rutas fue difícil determinar un patrón esto sirvió para definir que continuar con los análisis geográficos de las rutas no nos aportaría información relevante para la simulación que se desea obtener.

6.5 Simulación

Luego de las labores de limpieza de datos y encontrar algunos aspectos relativos a la generación de RSU, y las relaciones entre las variables que afectan los procesos de recolección en Austin Texas, se intentó reproducir el patrón de generación en una de las localidades de Bogotá.

Para poder lograr esta reproducción de datos se requiere información geográfica propia de Bogotá, que contenga las vías y ubicaciones de las calles, y los polígonos que muestran los contornos de los vecindarios de la ciudad y otros datos importantes de la ciudad son de acceso público y están disponibles en la página Web de la Unidad Administrativa Especial de Catastro Distrital. (Ideca-Mapas, 2019)

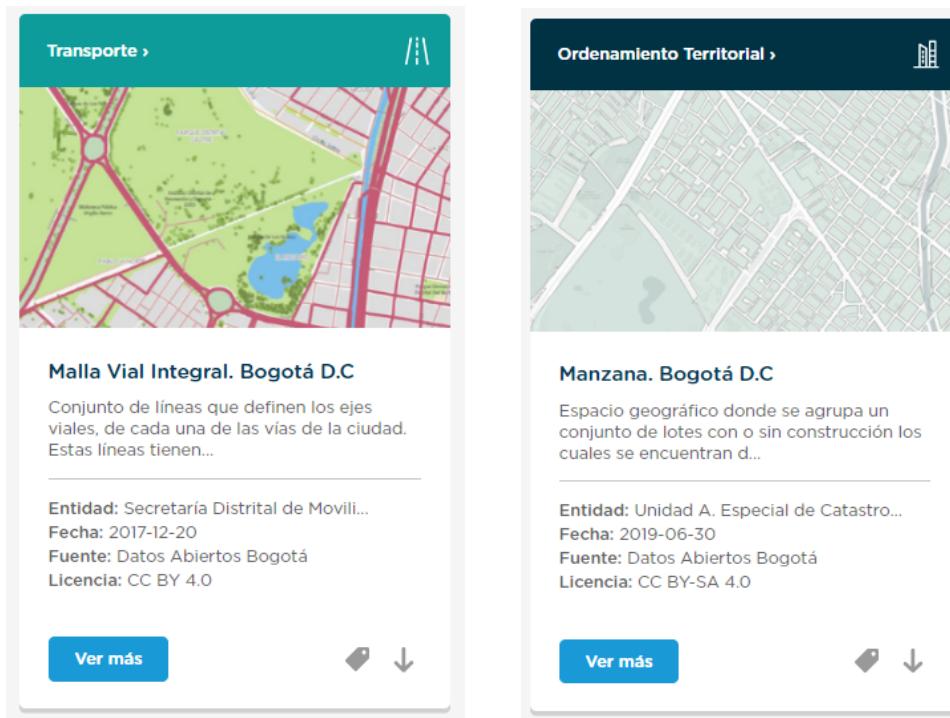


Figura 6-10. Mapas correspondientes a las mallas viales y manzanas, barrios de Bogotá tomado de (Ideca-Mapas)

Los datos descargados inicialmente tenían la estructura que se muestra a continuación (ver figura 6-11):

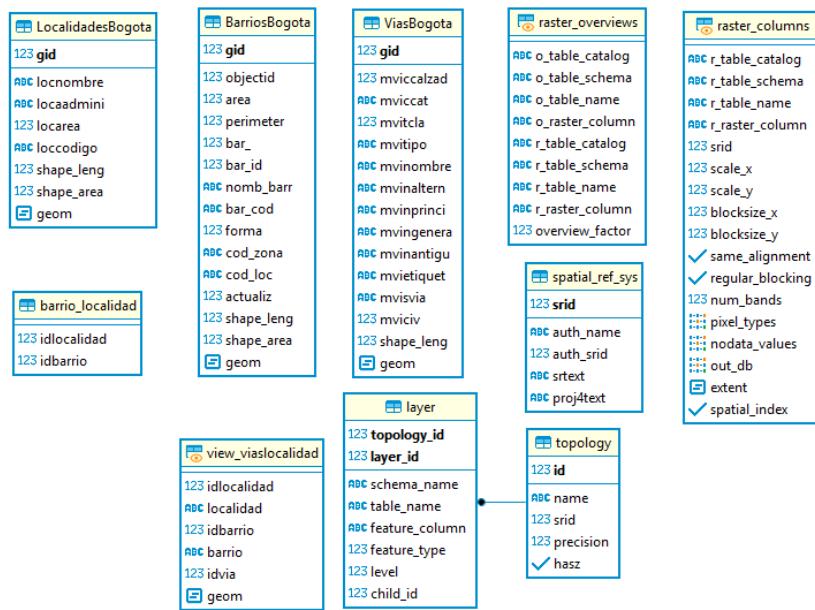


Figura 6-11 Estructura del conjunto de datos geográficos catastrales de Bogotá.

Como se puede ver, en este esquema de datos se dispone de los barrios, las localidades, las vías y una tabla donde se relacionan los barrios con la localidad en la que están contenidos, los datos están en el sistema de referencia Magna-Sirgas definido por el Instituto Geográfico Agustín Codazzi.

Los datos fueron cargados en una base de datos que dispone de un complemento geográfico llamado postgis, luego partiendo de estos datos se usaron las funciones geográficas de postgres para detectar qué rutas pertenecían a un barrio en particular y cuáles de estos barrios pertenecen a la localidad de nuestro interés, Engativá marcada con ID 19 y por medio de la conexión entre los barrios pudimos establecer las vías que pertenecen a la localidad. De la tabla barrios localidad se extrajeron los barrios que pertenecían a la localidad con id 19, luego haciendo uso de la función que compara los datos de tipo geográfico “**ST_Contains(Vías.geom, Barrios.geom)**” se logró determinar cuáles son las vías que se encuentran dentro de la región de los barrios se generó una nueva tabla llamada barrio_vía que contiene la relación entre la vía y el barrio al que pertenece, lo cual es de nuestro interés por 2 aspectos el primero porque sobre las calles se ubicaran los contenedores simulados y el segundo que las rutas de recolección deben ser entregadas en función de las calles que se deben recorrer para recoger los contenedores.

Solo los datos requeridos fueron migrados a la base de datos del proyecto para no saturar con información innecesaria. La estructura de la base de datos quedó de la siguiente manera (ver figura 6-12):

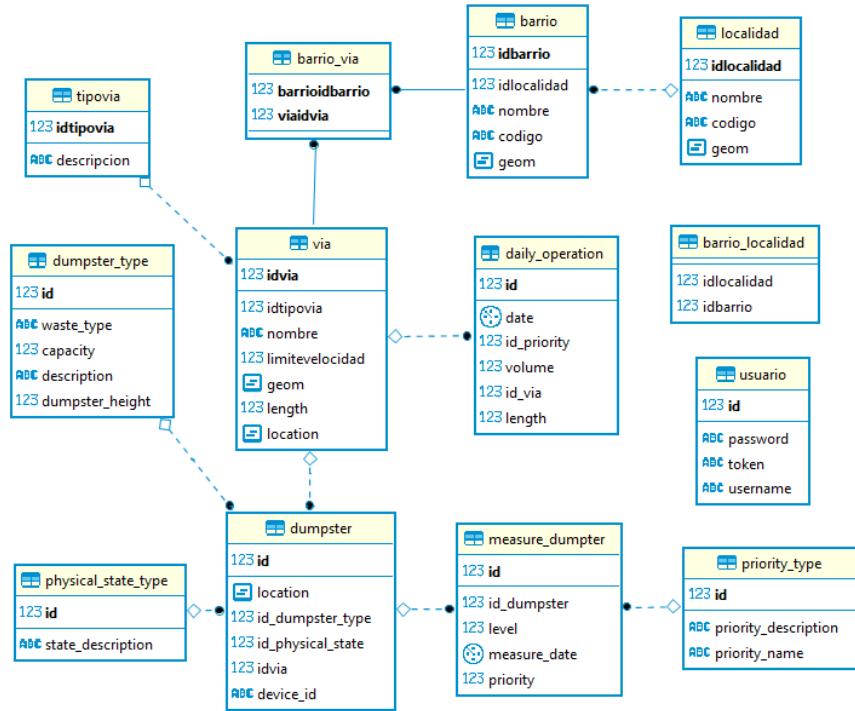


Figura 6-12 Estructura DB del proyecto luego de Migrar las tablas con la información espacial de catastro distrital.

6.6 Generación de datos

La generación de datos se dividió en dos fases. La primera, por medio de la cual se generó la ubicación virtual de los contenedores, considerando que deben estar ubicados sobre las calles. Se utilizó el listado de las calles que pertenecían a los barrios de la localidad que tuviesen las etiquetas de calles primarias y secundarias únicamente. se extrajo un punto intermedio a lo largo de la calle, esa coordenada corresponde a la ubicación de un contenedor, en dicha ubicación se pusieron 2 contenedores el primero simula el contenedor para desecho ordinario y el segundo para almacenar el material reciclable, tal como están ubicados actualmente, se incluyó una restricción adicional en la función; que debe existir un espacio de por lo menos 65 metros entre los contenedores colocados y un nuevo contenedor, lo cual se logró usando la función “*ST_Distance_Sphere*”, con esta estrategia se generaron 2868 contenedores en total este número coincide con los contenedores que habían sido instalados en la localidad de Engativá al 28 de febrero de 2019.

La segunda fase está relacionada con la generación de niveles de llenado, la función para generar tales datos se implementó teniendo en cuenta las distribuciones de los cargas del conjunto de datos encontrado para la ciudad de Austin - Texas, el cual se usó

como referencia, pues se comprende bien que las 2 ciudades tienen factores como densidad poblacional, hábitos de consumo, estrategias de reciclaje diferentes y por lo tanto su comportamiento también será diferente. Se generaron números entre 1 y 100 que representan el porcentaje de llenado en los contenedores, para ello se usaron la función Next Gaussian de Java y se sesgaron usando la misma media y la misma desviación estándar hallada en el conjunto de datos de Austin.

6.7 Representación geográfica de los datos generados

Los datos son importantes, pero encontrar patrones en un volumen grande de datos puede resultar difícil; los seres humanos tendemos a comprender mejor la información si se encuentra en una representación visual, comúnmente los datos geográficos se representan sobre un mapa. Como los datos que se generaron tienen un componente geográfico, esto puede ser aprovechado para efectos de vizualización usando un mapa, por lo que se desarrolló una sencilla aplicación Web que consume el servicio de mapas de Google y representa los datos de los contenedores simulados, para mostrar su ubicación aproximada, para representar el nivel de llenado, se usó un código de color que se describe a continuación.

nivel	Prioridad	Color
menor al 10%	Vacio	negro
entre el 11% y el 30%	bajo	verde
entre el 31% y el 55%	Medio	amarillo
entre el 56% y el 80%	alto	morado
mayor al 81%	Lleno	rojo

Tabla 6-1. Código de color para representación de prioridades.

Los niveles de llenado fueron establecidos según los porcentajes usando los análisis reportados por (Meesala et al, 2018), los cuales fueron validados usando un dispositivo de medición que integra un sensor de ultrasonido que considera la directividad y el ángulo de la señal emitida por el sensor.

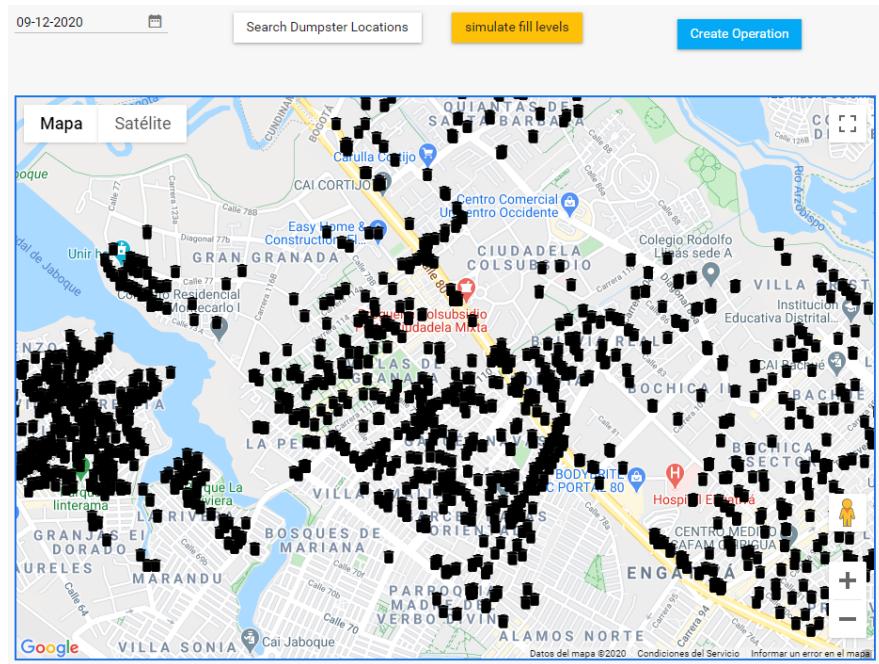


Figura 6-13. Fase 1, simulación de las ubicaciones de los contenedores.

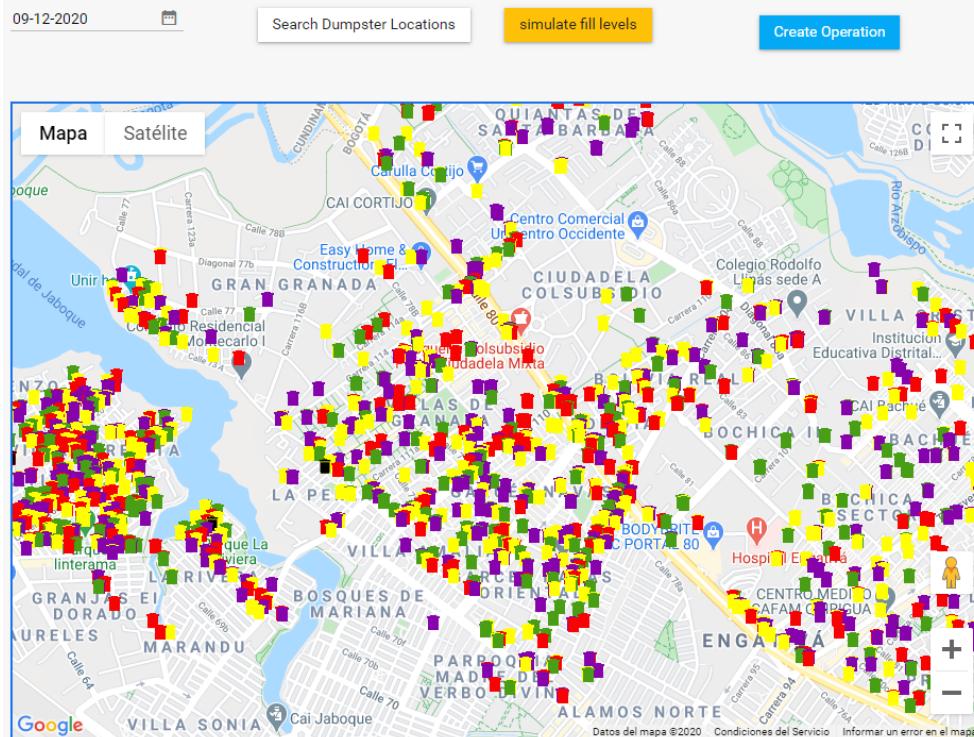


Figura 6-14. Fase 2, simulación de los niveles de llenado en los contenedores.

Teniendo en cuenta que la función objetivo busca maximizar el volumen recolectado, reduciendo la distancia recorrida y los recursos usados, una manera de conseguirlo es determinar la cantidad mínima de vehículos que se requieren para recolectar el volumen de RSU reportado por los sensores, luego establecer cuáles son las calles que deben recorrer los camiones, para atender los contenedores y las conexiones más cortas entre esas calles, con lo cual se consigue conformar un grafo sobre el cual se pueden conformar caminos Eulerianos¹⁵.

Para lograr esto se generó un arreglo por cada calle, en él se describe el volumen de residuo o su prioridad calculada a partir de 2 factores: primero la fecha del último registro de recolección y segundo las prioridades de los contenedores en esa calle, como se describe en la tabla 6-2.

Prioridad de calle	nivel de contenedor	Descripción
1	lennos > 0	existe por lo menos un contenedor lleno, o no se recolecta hace más de 3 días
2	Altos > 0	existe al menos un contenedor nivel alto, o no se recolecta hace más de 2 días
3	Medios > 0	solo hay contenedores de nivel medio o bajo en la calle
4	Bajos > 0	solo hay contenedores de nivel bajo en la calle

Tabla 6-2 Definición de prioridades de las calles según niveles reportados.

6.8 Técnicas de agrupamiento

En (Vu & Kaddoum, 2017) describen la importancia del preprocesamiento antes de la generación de rutas, ellos trabajan con datos de la ciudad de Philadelphia, Pennsylvania, USA. Para ello usan el método de las K-medias (k-means), para lo cual toman las coordenadas geográficas de los contenedores; latitud - longitud, como característica de agrupación y aplican el método del codo como estrategia de convergencia, además de una regresión logística para predecir aquellos contenedores que alcanzarían su nivel de llenado completo durante las próximas 6 horas y que por lo tanto debían ser incluidos en la operación.

Usando este precedente como referencia seleccionamos el método de K medias como estrategia de agrupación, la idea de este método de agrupación es encontrar una distribución equitativa entre la cantidad de elementos en cada grupo, tomando como

¹⁵ Es un trayecto en un grafo finito en el que si visitan todas las aristas del grafo exactamente una vez, se distingue de un ciclo Euleriano porque no necesariamente el nodo de partida y el de llegada deben ser el mismo, esta no es una restricción para este problema.

función objetivo minimizar la distancia de cada elemento al punto central o punto de referencia del grupo; reduciendo estas distancias individuales también se logra reducir la distancia global. El punto de referencia comúnmente es la media aritmética del atributo de comparación de los elementos que componen el grupo. Para este análisis el criterio seleccionado es la ubicación de las calles descritas en términos de su latitud y longitud.

$$\frac{\sum_{i=1}^n lat_i}{n}, \quad \frac{\sum_{i=1}^n lng_i}{n}$$

Con este enfoque, el punto de partida del camión recolector debe ser el centroide de cada grupo y las calles que conforman el grupo conformarían la ruta de recolección que el camión debe atender, sin embargo en ningún caso la sumatoria del volumen en un grupo debe exceder la capacidad del camión, de manera que para determinar el número K, es decir del número de camiones que realizan la operación, se debe dividir la sumatoria del volumen reportado (volumen total) entre la capacidad de los camiones y tomar el siguiente valor entero, teniendo en cuenta que si el valor decimal es próximo a la unidad se debe añadir un camión adicional pues durante la recolección se seguirán generando residuos.

“Este camión compactador es más grande que sus predecesores, con lo que se pasó de una capacidad de 22,86 metros cúbicos de basuras a 29 metros cúbicos. En términos de peso, esto significa que ahora el automotor podrá cargar entre 18 y 20 toneladas, en relación con las 12 o 13 que antes podía mover.”

(Franco, F. M, 2018)

Como en las demás fases de la simulación se ha usado java, se buscó una implementación de código libre que implementara en método de las K-medias, en este mismo lenguaje, se encontró el código disponible en el repositorio (<https://github.com/xetorthio/kmeans>), el cual se ajustó según las restricciones que se habían definido, sobre la versión original del código, se hicieron las modificaciones correspondientes, entre ellas el cálculo de las distancias en 2 dimensiones y el análisis de convergencia por el método del codo y las pendientes de los resultados entre K = X y K = X + 1.

En este punto de la agrupación, el desafío consiste en encontrar un número de grupos, que reduzca la distancia global, sin sobre ajustar el modelo (*overfitting*), la función de eficiencia se calculó como la suma de las distancias euclidianas entre los centroides de las calles y el centroide de cada grupo. Con esto se busca validar si hay una correspondencia cantidad de camiones obtenida a partir del volumen total de residuo y si este valor de K es óptimo reduciendo la distancia total que los camiones deben recorrer en las rutas asignadas.

De este análisis se pudo deducir que las diferencias entre las funciones de eficiencia disminuyen sustancialmente durante las primeras iteraciones, pero con aumento de la

iteración el mejoramiento del rendimiento no evoluciona significativamente. Por lo tanto, se deben detener las iteraciones en el momento en el que no se obtenga una mejora significativa entre una iteración y la siguiente. Para identificar este cambio del rendimiento se usó un enfoque analítico que evalúa la pendiente de la recta que se forma para las iteración $K = X$ y $K = X + 1$ que será la recta A y la recta que se forma para la iteración de $K = X+1$ y $K = X+2$, lo cual ocurre cuando la pendiente de la función se encuentra entre $\frac{1}{2}$ y $-\frac{1}{2}$, en el análisis realizado está pendiente se obtuvo cuando el valor de K estaba entre 9 y 10. Una explicación más rigurosa del análisis convergencia por método del codo y su implementación en javascript se puede encontrar en (Gove, 2017), en el caso de nuestra implementación este código fue transscrito en lenguaje java e integrado con el código que se encontró en el repositorio público de github.

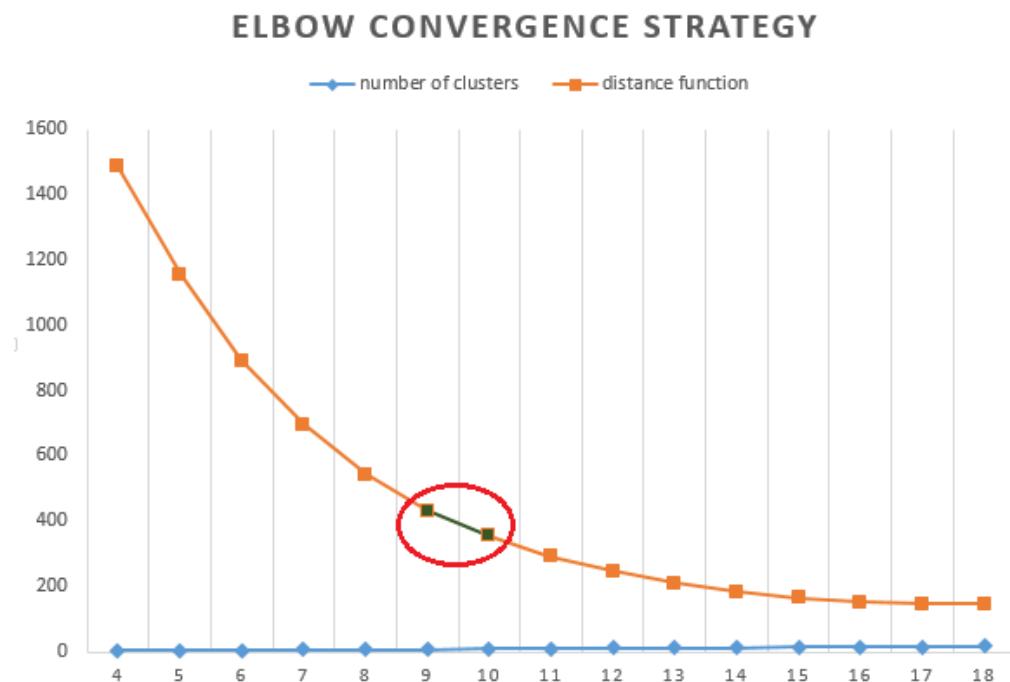


Figura 6-15. Análisis de convergencia cantidad de grupos basado en el método del codo.

El resultado del agrupamiento se puede observar en las figuras 6 - 16 y 6 - 17:

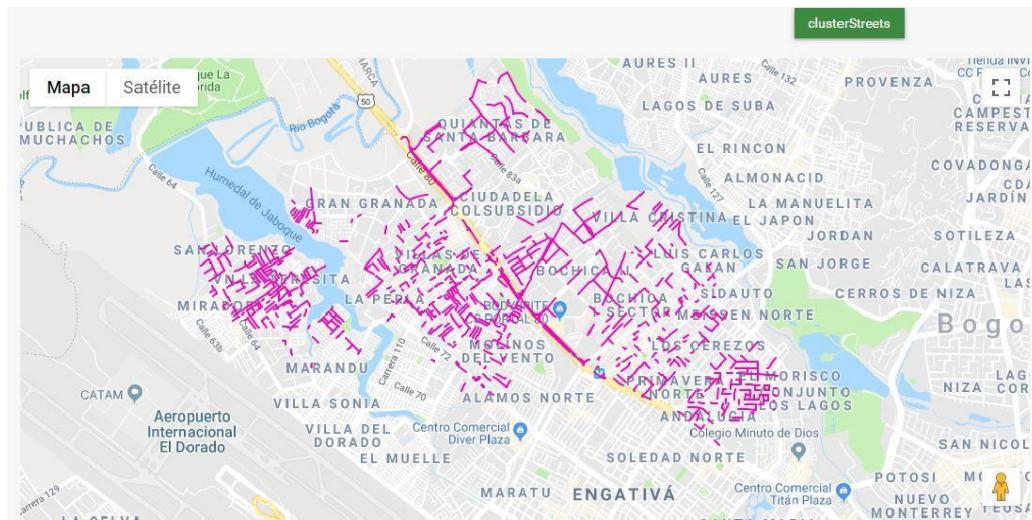


Figura 6-16. Calles priorizadas agrupadas usando $K = 1$.

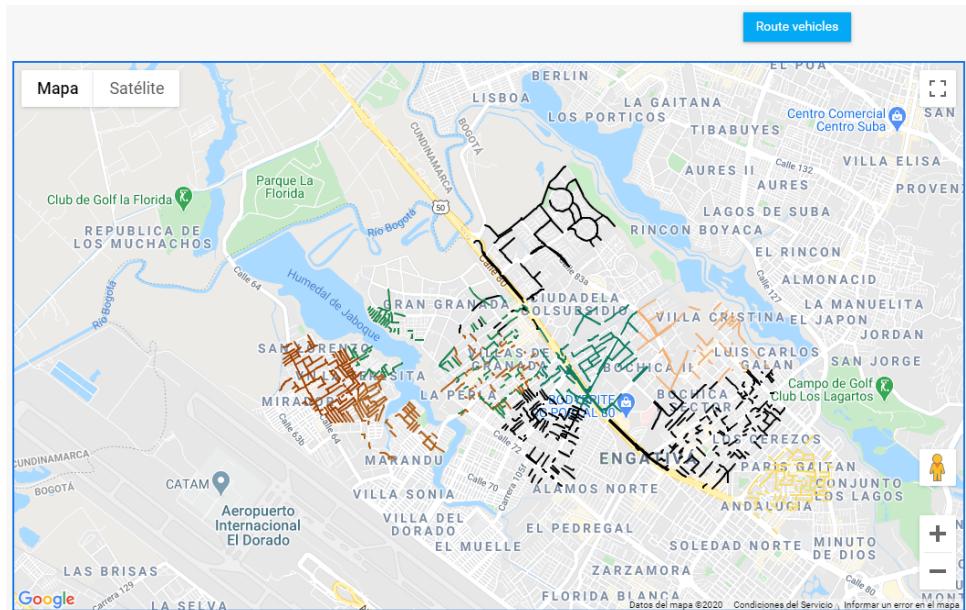


Figura 6-17. Calles priorizadas agrupadas usando $K = 9$.

En este momento se puede ver más claramente que las calles priorizadas no están completamente conectadas, bajo el esquema del cálculo de rutas propuesto que se basa en caminos euclidianos, los camiones deben viajar a través de calles, en las que no se requiere realizar una recolección, sólo para poder llegar a las calles donde se encuentran los contenedores etiquetados con mayor prioridad, lo cual tiene un impacto negativo en el función objetivo, porque incrementa la distancia recorrida por los camiones, considerando que el objetivo es recolectar mayor volumen de residuos, reduciendo la distancia recorrida.

7. Generación de Rutas de Recolección

7.1 Problema del enrutamiento de vehículos

El problema de enrutamiento de vehículos fue descrito originalmente por (Beltrami & Bodin, 1974), quienes tratan detalladamente las prioridades, entre tiempo, volumen, distancia y sus relaciones dinámicas, pero además como estas pueden ser conciliadas en una función objetivo, esto con el ánimo de reducir los costos de transporte y definir una estrategia que permita gestionar de forma organizada y oportuna un esquema de entrega o recolección de productos, mejorando los sistemas logísticos que para ese tiempo eran un tanto rudimentarios considerando que no se disponía de herramientas como internet o GPS.

Posteriormente (Gottinger, 1986) uso los postulados de Beltrami & Bodin para proponer un modelo computacional aplicado a la gestión de residuos, transformando el modelo matemático inicial en un problema de flujo a lo largo de una red y resuelta por medio de un algoritmo, pero lo más relevante del estudio es que fue aplicado a la resolución del problema de recolección de basuras a nivel regional en el área metropolitana de Múnich Alemania. Como conclusión, el autor encuentra una combinación de camiones dirigidos a rellenos que permiten un costo mínimo de recolección.

El procedimiento de recolección que se pretende resolver en este estudio tiene muchos aspectos en común con el estudio propuesto por Gottinger, pero con varias ventajas considerables incluidas: información muy precisa de la configuración de las calles basados en sistemas de referencia geográficos semejantes a la realidad, computadores potentes capaces de procesar operaciones complejas en unos pocos minutos y sistemas de monitoreo de tráfico que nos permiten conocer la densidad del tráfico en un la zona de recolección, haciendo que el proceso de calcular la ruta sea mucho más eficiente y preciso que el propuesto hace 34 años.

El enrutamiento de vehículos en sí mismo es un problema computacional complejo como se ha podido deducir en la literatura revisada; resolver el problema de una manera diferente a las propuestas consultadas e implementar el algoritmo correspondiente es una tarea difícil pues requiere validación mediante pruebas rigurosas y análisis comparativos frente a los recursos de referencia (*benchmarks*), lo cual que excede el alcance de este proyecto. Sin embargo, existen algunos servicios en internet que implementan los algoritmos de enrutamiento y los exponen a través de una API, para que personas que requieren integración con sus sistemas puedan hacer uso de dichas soluciones; de esta manera se pueden resolver dichos problemas haciendo uso de estrategias que ya han sido verificadas ahorrando tiempo y recursos.

7.2 Open Route Service ORS

ORS ofrece varios servicios pensados para facilitar la planeación de viajes e incluidos servicios de mapas compuestos por un gran conjunto de datos de imágenes satelitales, así como una amplia gama de capas vectoriales, con las ubicaciones de elementos como: fuentes hidrográficas, regiones de vegetación, puntos de interés y mallas viales etc; geocodificación, para transformar direcciones en coordenadas geográficas y viceversa; isócronos para determinar la región de impacto de un fenómeno, mapas de elevación para poder ejecutar estudios de terreno en geografías complejas. Sin embargo, el servicio que se usará en este proyecto es el de optimización.

7.3 Servicio de rutas

De acuerdo con la descripción de la página ORS, el servicio de optimización es una interfaz de aplicación que puede resolver distintos tipos de problemas de enrutamiento de vehículos. Este servicio suministra respuestas rápidas y permite personalizar las condiciones de los vehículos, las tareas que ejecutan y tiempos de acuerdo a las necesidades del usuario. (O.R.S. 2019)

el servicio de ORS usa un proyecto llamado VROOM enrutamiento de vehículos por una máquina se optimización de código libre; en este proyecto se resuelve el problema de enrutamiento usando una estrategia de agrupación heurística llamada expansión de árboles, la estrategia fue planteada originalmente para el enrutamiento en telecomunicaciones, sin embargo (Wu et al., 2000) se percataron que considerando las distancias de las calles o el tiempo para recorrerlas como factor aplicado al costo de ruta también podía ser aplicado al enrutamiento de vehículos. El servicio ejecuta operadores de modificación sobre la solución inicial, para reducir las distancias de los recorridos, algunos operadores son: reubicación, intercambio, inversión y alteración, la estrategia de solución ha sido descrita detalladamente en (Zhou, et al. 2018); los desarrolladores del servicio han publicado un video que explica el uso y funcionamiento del servicio y su estrategia de solución el cuál puede ser consultado en (Coupey, J. 2018).

Para el caso particular del enrutamiento de vehículos recolectores, se deben enviar múltiples vehículos de manera que el resultado del servicio sea la ruta optimizada para toda la flota, cada una de las paradas que indican un punto de recolección o entrega se debe definir como una tarea. El servicio permite definir los siguientes atributos para las rutas y los vehículos:

- **Capacidades:** definen los volúmenes de la carga en cada punto lo cual permite identificar la capacidad de carga del camión en cada tramo del recorrido.

- **Duraciones y ventanas de tiempo:** cada tarea puede traer asociado una duración y un periodo específico en el que debe ser realizada, por ejemplo las horas de cierre de los establecimientos.
- **Habilidades:** que se refieren a las necesidades específicas que requiere un conductor o un vehículo para que le sea asignada una tarea. Por ejemplo sólo vehículos con acondicionamiento frío podrán transportar alimentos que requieran conservar una cadena de frío.
- **Prioridad:** las tareas pueden ser priorizadas según su importancia de manera que el servicio intentará ejecutarlas a la brevedad posible.
- **Restricciones:** para poder prestar los servicios a toda la comunidad el servicio de Optimización permite un máximo de 500 peticiones diarias por usuario y máximo 40 peticiones por minuto.

7.3.1 Implementación.

La generación de rutas fue generada a partir de la información simulada, por lo cual se creó un nuevo botón en la aplicación Web que se usó para simular la información. Este botón genera una comunicación con el servidor del sistema, el servidor recupera la información simulada de las calles que cumple con el criterio de prioridad definido. Para este caso, se tomaron solo calles cuyos contenedores tuvieran niveles de llenado superiores al 55%

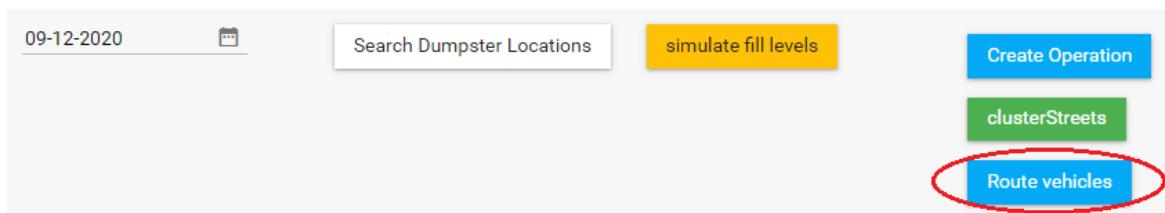


Figura 7-1. Inclusión del botón para calcular rutas en la aplicación Web.

Luego se totaliza el volumen aproximado (VolAp) como la sumatoria del volumen en cada contenedor:

$$VolAp = \sum_{i=1}^n Porcentaje_i \times capacidad_i$$

Con base a ellos se calculan la cantidad de camiones requeridos para la operación dividiendo el volumen aproximado entre la capacidad estándar de un camión, la cual se definió como 37.5 toneladas.

Petición: usando esos datos se genera la petición enviando el conjunto de camiones y el conjunto de tareas, correspondiente a los puntos medio de la calle por donde deben pasar los camiones para recoger el contenido de los contenedores.

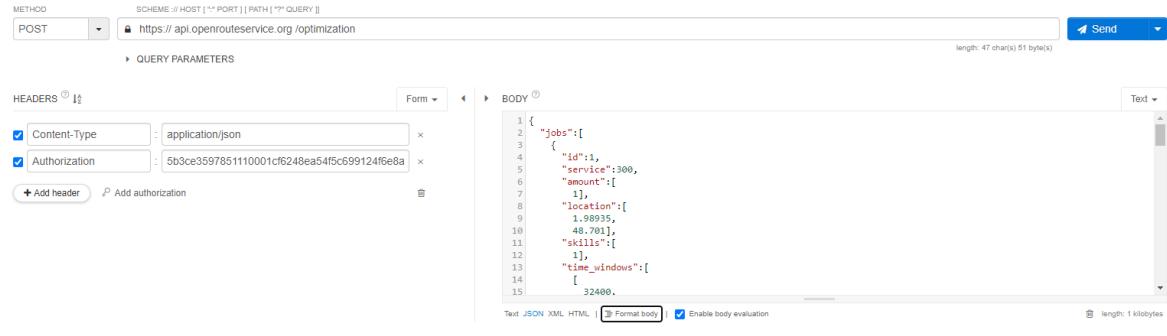


Figura 7-2 Configuración de prueba del servicio de enrutamiento.

Para esta prueba se obtuvo la respuesta que se observa a continuación.

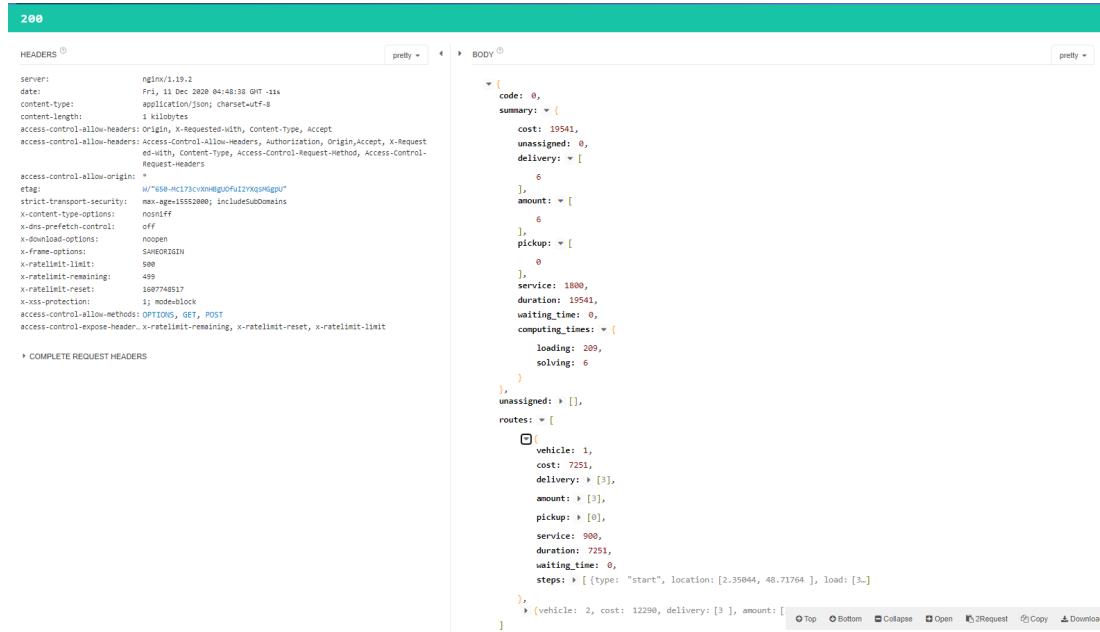


Figura 7-3 Respuesta del servicio de enrutamiento.

Estas imágenes corresponden a las pruebas realizadas, sin embargo, este procedimiento fue codificado en el servidor de aplicación del sistema respondiendo a la acción del botón **enrutar vehículos**. Las respuestas obtenidas del servicio se almacenan en una tabla del esquema de datos, de manera que la próxima vez que requieran ser consultar para una fecha en la que las rutas ya han sido generadas, estas se puedan reutilizar, recuperandolas directamente de la base de datos, sin tener que volver a usar el servicio de enrutamiento, con el ánimo de economizar recursos de cómputo.

8. Programación de la Interfaz Visualización de Rutas

Como interfaz para visualización de las rutas se desarrolló una aplicación móvil, considerando que actualmente disponemos de gran variedad de dispositivos móviles como teléfonos inteligentes o tabletas que pueden ser mucho más ligeros, prácticos, económicos y fáciles de usar, comparados, por ejemplo, con una computadora que están pensadas para desempeñar tareas más complejas.

Si bien una aplicación web se puede ejecutar desde un dispositivo móvil, resulta mucho más cómodo y adecuado usar una aplicación móvil, pues se pueden usar controles los nativos del sistema operativo a los que el usuario que está acostumbrado, enriqueciendo su experiencia de usuario pero principalmente por su capacidad de acceder y gestionar otros recursos físicos como la cámara, el micrófono, almacenamiento o la ubicación del dispositivo.

Dada la naturaleza geográfica del sistema propuesto, y que los puntos de la vía se encuentran en coordenadas geográficas, resulta intuitivo para el usuario ver las coordenadas de los puntos de recolección y las rutas calculadas sobre un mapa, el cual le indique las vías que debe recorrer hasta los puntos de recolección de cada ruta, y que además que le muestre lugares que sean familiares y que puedan servir como puntos de referencia.

Flutter:

Flutter es un kit de desarrollo de aplicaciones móviles libre, creado por Google que busca permitir la construcción de aplicaciones de nativas usando un recurso de código compartido para las distintas plataformas, este código compila directamente en lenguaje de máquina y utiliza un motor gráfico llamado SKIA, que optimiza los gráficos haciéndolo muy rápido, dado que no usa intérpretes en tiempo de ejecución.

Los bloques básicos de construcción en Flutter se llaman *widgets* y son componentes modulares que se pueden reutilizar a lo largo de la aplicación; el kit dispone de una gran cantidad de *widgets* con los que se puede empezar a construir aplicaciones móviles, pero le da la opción al usuario crear sus propios widgets según sus necesidades haciendo que el *framework* sea versátil y ajustable a las necesidades del proyecto.

Una de las ventajas principales de Flutter es su carácter libre, su código es gratuito y está disponible *online*, de manera que cualquier desarrollador puede ver cómo está construido y adaptarlo para obtener mejores resultados, esta característica atrae muchos desarrolladores generando una comunidad.

Autenticación JWT

La seguridad es un aspecto importante en cualquier tipo de aplicación, por lo menos un nivel básico de seguridad se logra aplicando los principios Confiabilidad, Integridad y Disponibilidad (CIA), esto se consigue en cierta medida aplicando 2 técnicas: la autenticación y el cifrado, garantizando que la información esté disponible solo para aquellas personas que tienen permisos de acceso a ella y el cifrado evitando que la información sea interpretable por terceros no autorizados o peor aún alterada.

Una herramienta que nos permite integrar estos 2 técnicas en aplicaciones Web y móviles es Json Web Token JWT: es un mecanismo para transmitir la identidad de un usuario usando un formato JSON y codificado en base 64¹⁶ y este mensaje es firmado digitalmente, este objeto de intercambio de información se llama Token y está compuesto por 3 partes el encabezado, el cuerpo y la firma los cuales van separados por punto.

El encabezado contiene el algoritmo de cifrado y el tipo de token, el cuerpo contiene el mensaje y la firma contiene el resumen matemático de la información y que verifica que el remitente sea quien dice ser; esta se verifica cuando se recibe y se vuelve a generar luego se comparan la firma recibida y la firma generada, así se puede determinar si el mensaje ha sufrido alguna alteración en el trayecto. En la figura 8-1 se puede observar un ejemplo de Token y su respectiva información decodificada.

The screenshot shows a web-based JWT decoder interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), there is a large text area containing a long, base64-encoded string of characters. On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), the token is broken down into three main sections:

- HEADER: ALGORITHM & TOKEN TYPE**: Contains the JSON object: { "alg": "HS256", "typ": "JWT" }
- PAYOUT: DATA**: Contains the JSON object: { "sub": "1234567890", "name": "John Doe", "iat": 1516239022, "email": "jondoe@mail.com", "vehicle": { "branch": "toyota", "plate": "HFW-786", "origin": "villavicencio" } }
- VERIFY SIGNATURE**: Shows the HMACSHA256 formula: HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret). There is also a checkbox labeled "secret base64 encoded".

Figura 8-1. Ejemplo de información transmitida usando el estándar JWT.

¹⁶ Usando esta base se puede codificar el mensaje usando una cantidad menor de caracteres lo cual disminuye la longitud del mensaje y ahorra ancho de banda.

Para la aplicación que se desarrolló, se usó una autenticación básica de usuario y contraseña que se verifica con una tabla en el modelo de datos. Si existe correspondencia de dicho usuario y contraseña, se emite un token, el cual se almacena en las variables globales de la aplicación, para autenticar en el servidor las demás peticiones que se van a hacer; de esta manera, el servidor puede llevar un registro de los usuarios que solicitan esta información y comprobar si dicho usuarios tienen permiso para acceder a la información que están solicitando.

```
RaisedButton(  
    onPressed: () async {  
        var username = _usernameController.text;  
        var password = _passwordController.text;  
        var response = await attemptLogIn(username, password);  
        if (response != null) {  
            var resp = jsonDecode(response);  
            storage.write(key: "jwt", value: resp['token']);  
            Navigator.push(  
                context,  
                MaterialPageRoute(  
                    builder: (context) =>  
                        HomePage.fromBase64(resp['token'])); // MaterialPageRoute  
            } else {  
                displayDialog(context, "An Error Occurred",  
                    "No account was found matching that username and password");  
            }  
        },
```

Figura 8-2. Implementación de la solicitud de token usando usuario y contraseña.

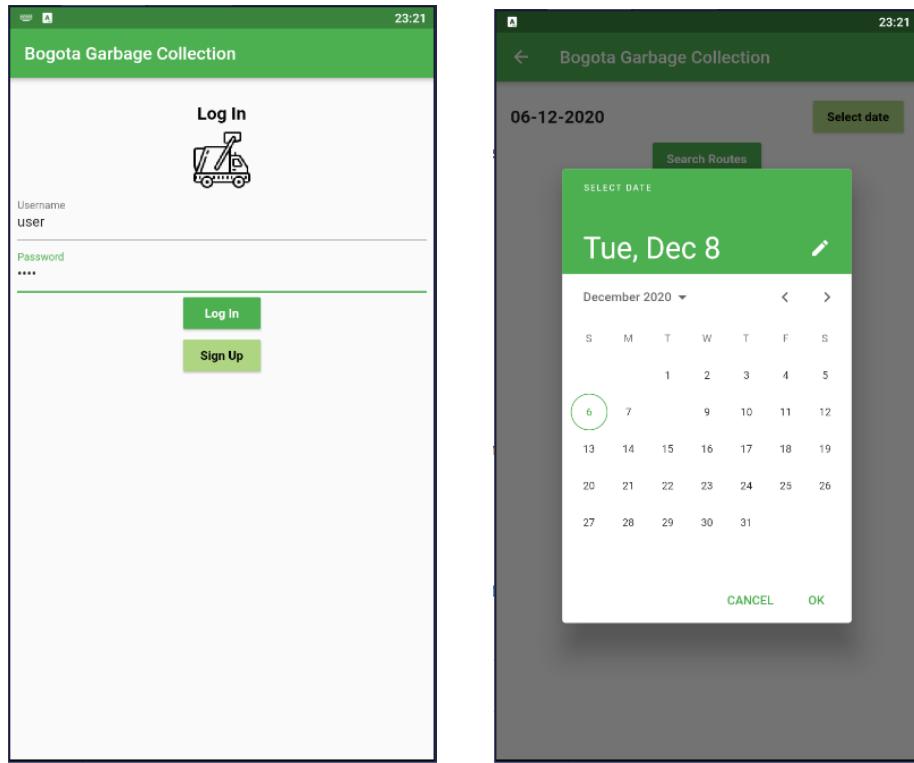


Figura 8-3. Interfaz de autenticación y consulta de rutas, selector de fechas.

La primera pantalla de la aplicación es la pantalla de autenticación, cuando la autenticación es exitosa se dirige a la pantalla de consulta de rutas.

8.1 Consumo de servicios Rest

Una aplicación móvil comúnmente requiere estar conectada a Internet para operar con información actualizada, una estrategia para lograr esta conexión es exponer servicios del lado del servidor para establecer una comunicación con los dependientes de estos servicios usando un protocolo de comunicación; Rest implementa un protocolo consolidado y reconocido por la comunidad llamado HTTP.

Para las operaciones de consulta, y más precisamente cualquier petición, Flutter proporciona una librería llamada HTTP, muy ligera y sencilla de usar, esta solo necesita la especificación del método que usaremos (get, post, put, delete) la dirección destino de la petición, y permite especificar encabezados y cuerpo.

```
Future<bool> searchRoutes() async {
    String stringDate = DateFormat('dd-MM-yyyy').format(selectedDate);
    String token = await storage.read(key: "jwt");
    if (stringDate != null) {
        var response = await http.get(
            Globals.urlApi + Globals.endPoinRutas + '/' + stringDate,
            headers: <String, String>{
                'Content-Type': 'application/json; charset=UTF-8',
                'Authorization': token,
            },
        );
    }
}
```

Figura 8-4. Fragmento de código implementación de HTTP para consulta de rutas.

En el caso de la aplicación móvil para la visualización de rutas, se realizó una solicitud de información al servidor de aplicación en la cual se consultaron las rutas, que se generaron usando el servicio ORS y la información simulada para una fecha específica; una vez se obtiene respuesta de la petición, la información recuperada se muestra en forma de lista en la pantalla de consulta de rutas.

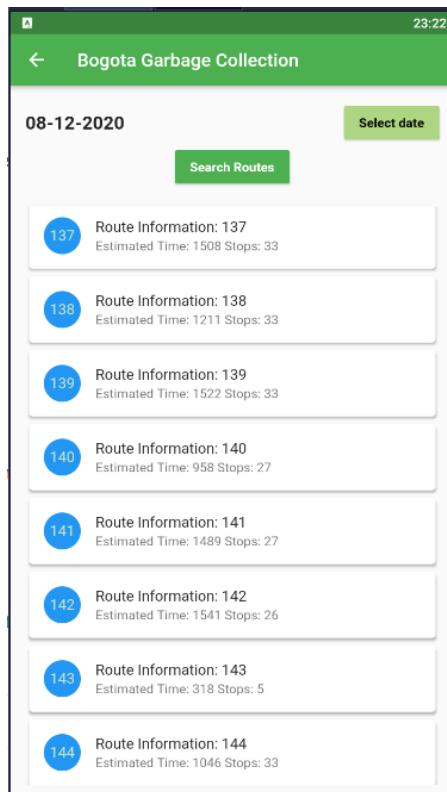


Figura 8-5. Interfaz con la lista de rutas encontradas para una fecha específica.

8.2 Integración con Google Maps

Existen varios proveedores de mapas, como Openstreetmap y Mapbox, para consultar mapas y tener referencias geográficas virtuales; para este caso, se decidió hacer uso de Google Maps, dado que la empresa creadora de Flutter es Google y dispone de un complemento para la integración de Flutter con las interfaces de Google Maps; además, porque consideramos que Google maps es un servicio preciso y confiable pues es actualizado a diario, gracias a los reportes de información de las personas que hacen uso de estos servicios.

8.2.1 Configuración del Servicio

Para poder hacer uso de los servicios de Google Maps se debe crear una cuenta en la plataforma de Google en la Nube GCP. Una vez la cuenta ha sido creada, se debe asociar una cuenta de facturación, la cual requiere del registro de una tarjeta de crédito, sobre la cual se realizarán cobros únicamente si se excede el monto mensual gratuito que GCP ofrece para efectos de desarrollo y pruebas.

Para ello, se debe ir a la sección de credenciales y una vez allí, se genera una clave de API

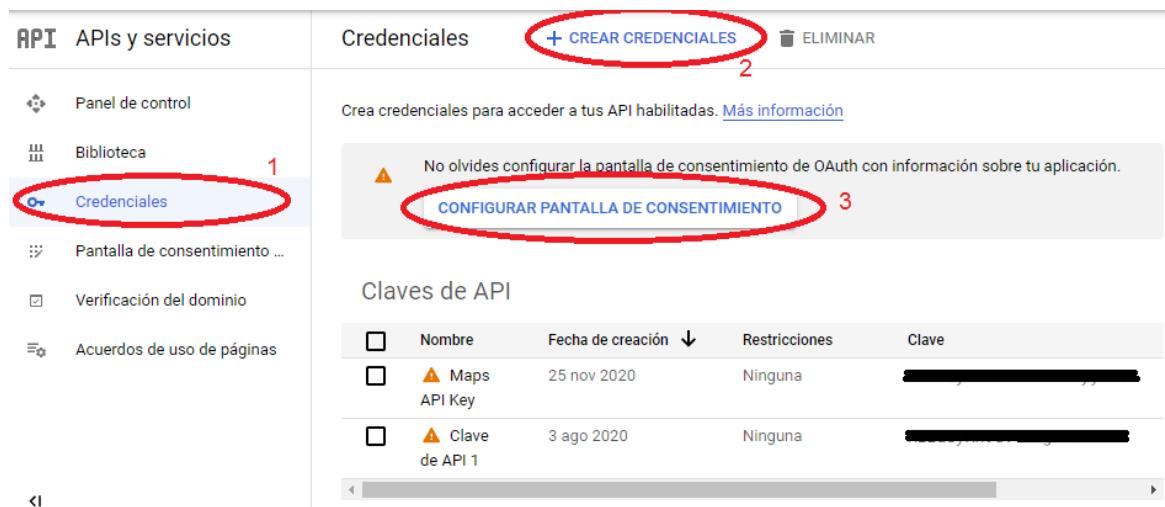


Figura 8-6 Generación de la clave para el consumo de API.

Una vez creada la clave, se le deben asociar permisos para especificar cuáles de los servicios ofrecidos por Google pueden ser consumidos haciendo uso de ella, esto con el fin de restringir el uso de servicios costosos que no son necesarios en la aplicación. Es importante configurar la pantalla de consentimiento OAuth; esta pantalla requiere cierta información asociada a la aplicación que hará uso de la clave, esto con el fin de evitar que terceros no autorizados que obtengan su clave hagan uso de ella para evitar gastos de facturación no deseados, es básicamente una manera de asegurar la clave.

Pantalla de consentimiento de OAuth

Tipo de usuario

Interna

Nombre de la aplicación

garbage_bogota

Correo electrónico de asistencia

mmontanezg@unal.edu.co

Logotipo de la aplicación

Omitido

Figura 8-7. Configuración para el consentimiento de OAuth.

Configuración del complemento

La librería `google_maps_flutter: 1.0.6` debe ser declarada como dependencia en el archivo `pubspec.yaml` de la aplicación; como la aplicación que se está desarrollando se generará para SO android, se debe agregar la clave para consumo del API, generada en la consola de Google, en un archivo que se encuentra en la siguiente ruta: (`android/app/src/main/AndroidManifest.xml`), para ello basta con incluir la siguiente etiqueta, reemplazando el valor del tag `android:value` por la clave generada.

```
<meta-data android:name="com.google.android.geo.API_KEY"  
        android:value="aquí va su clave de servicio"/>
```

8.2.2 Implementación del mapa

Para poder hacer uso del mapa, se debe declarar el componente de la librería

```
import 'package:google_maps_flutter/google_maps_flutter.dart';
```

El componente que construye el mapa se llama `GoogleMap`, el siguiente fragmento de código es el que genera (*render*) el mapa en la aplicación.

```

GoogleMap(
    markers: markers != null ? Set<Marker>.from(markers) : null,
    initialCameraPosition: _initialLocation,
    myLocationEnabled: true,
    myLocationButtonEnabled: false,
    mapType: MapType.normal,
    polylines: Set<Polyline>.of(polylines.values),
    onMapCreated: (GoogleMapController controller) {
        mapController = controller;
        _centerView();
    },
),

```

Figura 8-8. Segmento código del Constructor Componente Google Mapas App Móvil.

Markers: son un conjunto de iconos, a los cuales se les determina, una coordenada de latitud y longitud y un texto que se muestran sobre el mapa, son útiles para indicar puntos de interés.

initialCameraPosition: establece el lugar en el mapa donde se situará la imagen cuando se renderiza en la aplicación.

myLocationEnabled: permite añadir al mapa un símbolo que muestra la ubicación y la dirección reportada por el dispositivo móvil.

myLocationButtonEnabled: es un botón que cambia la ubicación de la cámara en el mapa y se centra en la ubicación reportada por el dispositivo móvil.

mapType: define el tipo de imagen con la cual se proyectará el mapa en la pantalla: existen mapas normales que son simplificados en colores, existen mapas satelitales que sobreponen las imágenes raster y que son mucho más costosos en términos de memoria y ancho de banda y mapa híbridos que usan componentes normales y los mapas satelitales.

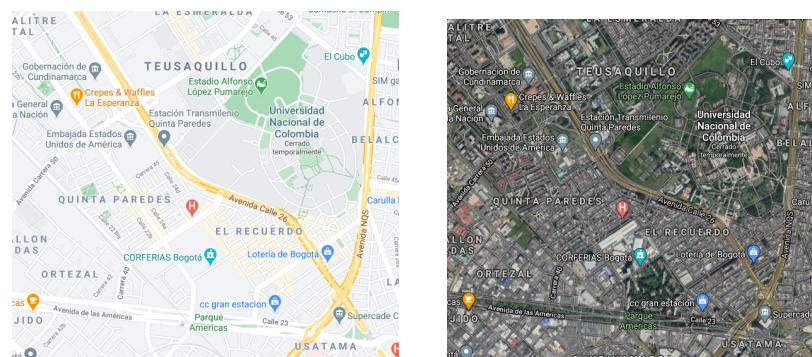


Figura 8-9. Tipos de mapas ofrecidos google maps (normal izquierda, satelital derecha).

Polylineas: son componentes vectoriales que se pueden describir como una secuencia de coordenadas geográficas y que forman líneas simples como una línea recta o tan compleja como la configuración de calles en un barrio, se puede usar por ejemplo para mostrar el recorrido de un vehículo entre un punto de origen y un punto de destino.

onMapCreated: es una función que se ejecuta cuando el mapa ha sido cargado y sirve para ejecutar operaciones que pueden actualizar la vista del mapa, o añadir otros elementos gráficos y controles en la interfaz del mapa.

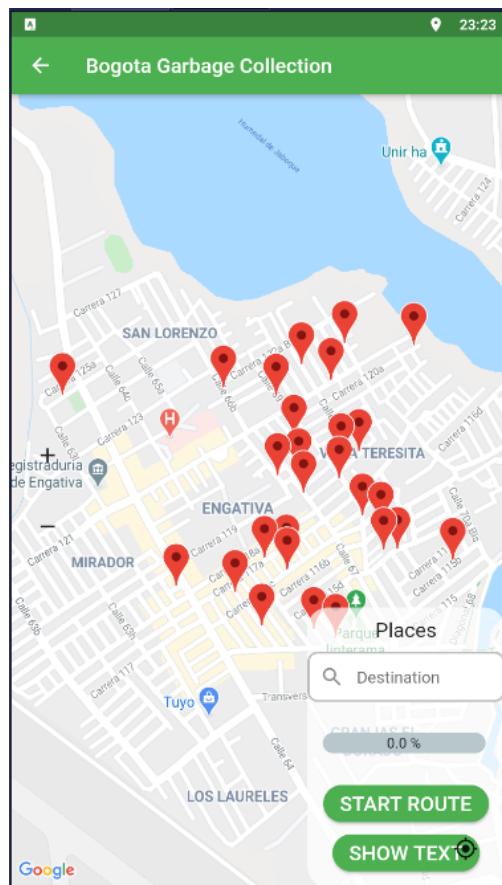


Figura 8-10. Interfaz del mapa suministrado por Google con marcadores para los contenedores de una Ruta.

La figura 8-10 muestra el mapa para una de las rutas consultadas, usando las coordenadas geográficas de las calles donde se ubican los contenedores que deben ser atendidos según el servicio; para cada uno de los puntos que componen el recorrido se generaron marcadores que se muestran sobre el mapa en color rojo, además, se recalculó el centro del mapa de manera que las coordenadas que componen la ruta queden incluidas del rango de visual del dispositivo móvil.

8.3 Directions Service

El sitio oficial define el servicio como una interfaz que calcula la ruta entre dos ubicaciones usando una petición HTTP; este servicio es muy flexible pues está en la capacidad de buscar direcciones para diferentes medios de transporte como: vehículo, transporte público, bicicleta o a pie. El servicio retorna una ruta conformada por una serie de puntos de intermedios (*waypoints*), que son los puntos por los que se deben pasar para llegar al destino y también se pueden especificar los lugares de origen y destino usando direcciones como calles y carrera o como coordenadas geográficas.

El servicio busca la ruta más eficiente usando como criterio de decisión primario el tiempo, pero puede ser configurado para tener en cuenta otros factores como distancia, cantidad de giros según las preferencias del usuario.

El servicio tiene en cuenta información del tráfico reciente con la cual busca evitar atascos en el tráfico, sin embargo, se debe tener en cuenta que el servicio no dispone de actualización en tiempo real, quiere decir que cuando se calcula la ruta el servicio retorna una ruta fija, constante y no se ajusta si las condiciones del tráfico cambian; para ello se debe calcular la ruta nuevamente.

Un ejemplo del formato de petición que recibe el servicio es el siguiente:

```
https://maps.googleapis.com/maps/api/timezone/json?origin=39.6034810,-119.6822510&destination=37.654871,-119.864059&key=YOUR_API_KEY
```

La integración de este servicio en la aplicación móvil se encuentra en un botón llamado “START ROUTE/ NEXT CONTAINER”; la estrategia de recolección consiste en iniciar el recorrido yendo de la ubicación del transportador al primer contenedor que debe ser recolectado. Luego de alcanzar dicho punto y recolectar el material, el botón NEXT CONTAINER calculará nuevamente la ruta entre el último contenedor visitado y el siguiente contenedor en la lista según el orden generado por el servicio de enrutamiento, a medida que se van recolectando los contenedores se calcula el porcentaje de progreso de la ruta hasta que el camión complete el recorrido; estas operaciones se pueden apreciar en las figuras 8-12 8-13.

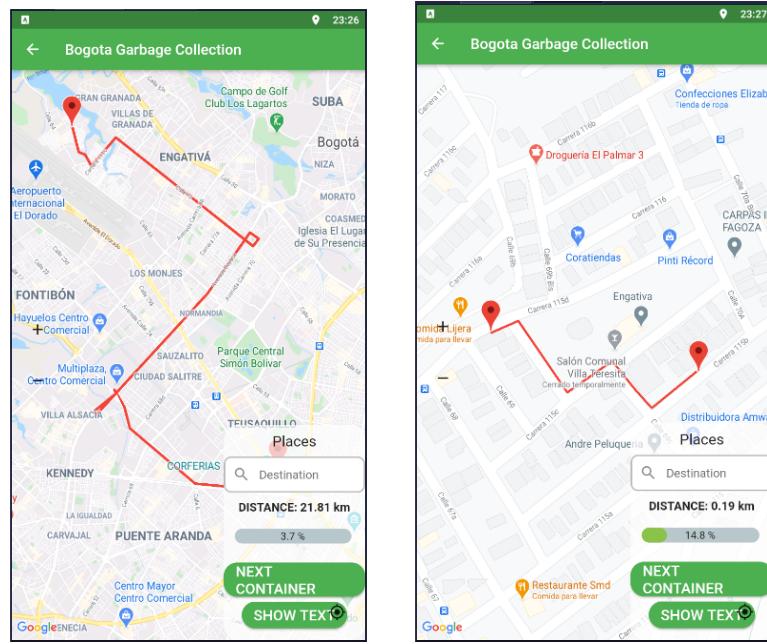


Figura 8-12. Interfaz de direccionamiento calculado por el servicio de Google. (entre la ubicación inicial del transportador y el primer contenedor, progreso de ruta 14,8%).

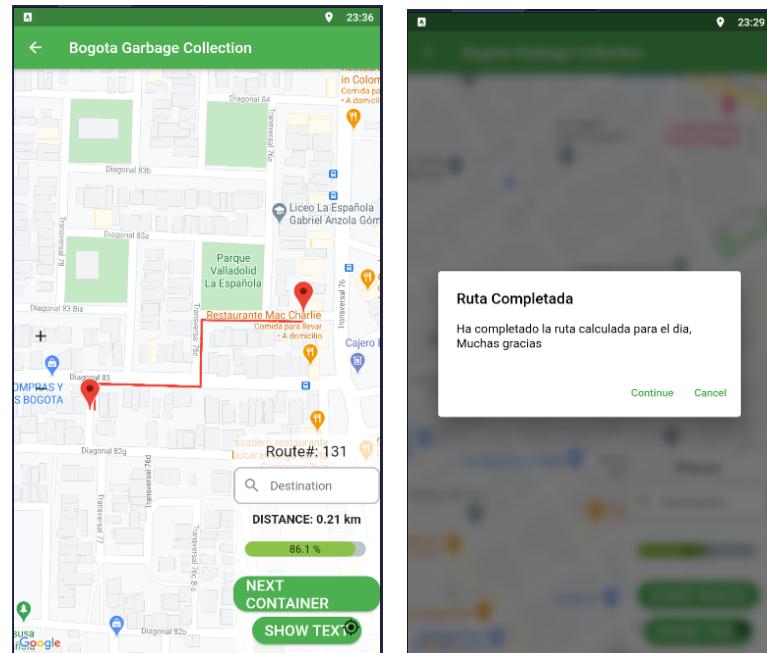


Figura 8-13. Interfaz de direccionamiento entre dos contenedores de la ruta 131. (Ruta alcanzando el 86,1% del recorrido, mensaje de notificación ruta completada)

8.4 Generación, firma del APK.

Para que nuestra aplicación sea publicada en una tienda de aplicaciones se necesita firmar la aplicación; para ello podemos ejecutar el siguiente comando:

```
keytool -genkey -v -keystore c:\Users\USER_NAME\key.jks -storetype JKS  
-keyalg RSA -keysize 2048 -validity 10000 -alias key
```

El cual genera un archivo llamado **key.jks**; una vez generado se debe referenciar desde la aplicación se debe crear en la siguiente ruta **app-folder/android** un archivo **key.properties**:

```
storePassword=<contraseña del store>  
keyPassword=<contraseña de la key>  
keyAlias=key  
storeFile= c:\Users\USER_NAME\key.jks
```

Este proceso se puede automatizar modificando el archivo **android\app\build.gradle** agregando el procedimiento para que la firma se genere cada vez que se empaqueta la aplicación, para lo cual resulta útil revisar esta documentación (<https://flutter.dev/docs/deployment/android>)

Para generar el archivo que instalará y ejecutará la aplicación en nuestro dispositivo móvil, usando la consola de comandos se puede ejecutar el comando **flutter build apk** el cual compila y empaqueta la aplicación

```
Calling mockable JAR artifact transform to create file: C:\Users\Miguel\.gradle\caches\transforms-2\fil  
es-2.1\d99aac6cf42f23ead5fa38f496cda169\android.jar with input C:\Users\Miguel\AppData\Local\Android\sd  
k\platforms\android-28\android.jar  
  
Removed unused resources: Binary resource data reduced from 256KB to 229KB: Removed 10%  
Running Gradle task 'assembleRelease'...  
✓ Built build\app\outputs\flutter-apk\app-release.apk (18.7MB).  
  
Miguel@MiguelM MINGW64 ~/Documents/Miguel/Flutter_courses/garbage_bogota
```

Figura 8-14. Resultado de la generación del APK.

Finalmente, para instalar la aplicación en nuestro dispositivo móvil basta con conectar el dispositivo y desde el directorio en el cual quedó alojado el archivo APK ejecutar el comando **Flutter Install** solo quedaría pendiente probar la aplicación y validar su desempeño en el dispositivo móvil.

Cuando la aplicación es pensada para un público muy grande o se trata de una aplicación de uso público, conviene gestionar la publicación de la aplicación en la tienda

de aplicaciones del SO en el caso de android en el Google Play Store; esto hace que el proceso de instalación sea mucho más simple, especialmente cuando los usuarios no son especialistas en tecnología, porque la tienda se encarga de gestionar la instalación de la aplicación en el dispositivo del usuario y gestionar los permisos. Este procedimiento de publicación consta de varios pasos y se debe cumplir con ciertos requisitos el procedimiento completo se puede consultar en el siguiente enlace (<https://developer.android.com/distribute/best-practices/launch>)

8.5. Consideraciones para Despliegue del Software en Entornos de Producción

introducción

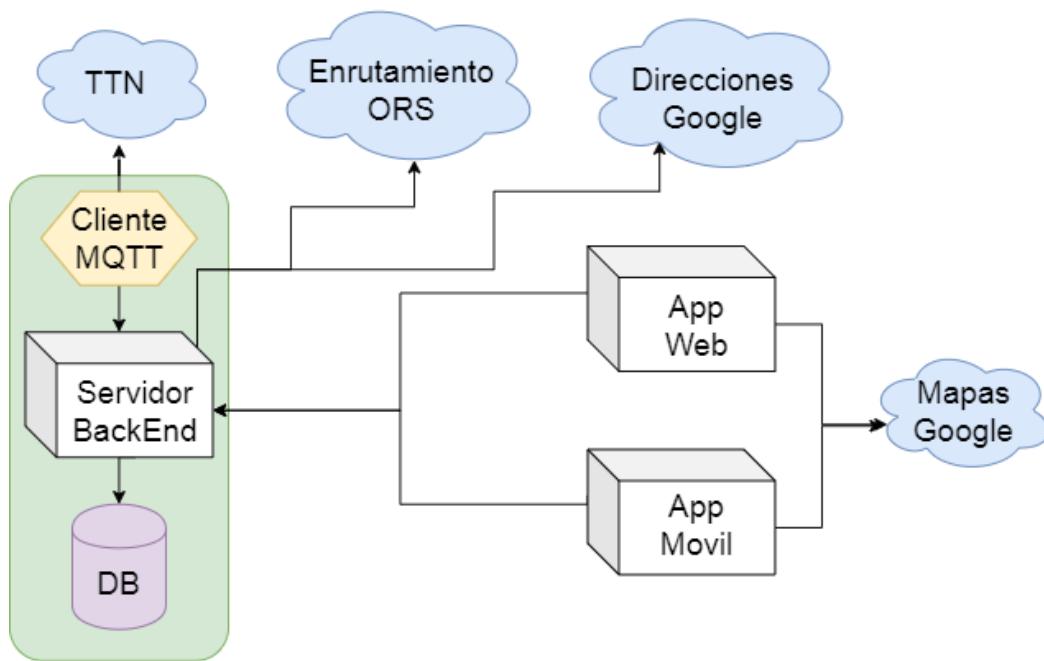


Figura 8 - 15. Diagrama de componentes de Software.

Servicios Adicionales: dado que la aplicación usa los servicios de direcciones y mapas de Google, se debe crear una cuenta de producción y asociar una tarjeta de crédito a la cuenta, se deben generar las llaves de API para cada uno de los servicios y reemplazarlos en las aplicaciones móvil y web, también se requiere configurar una cuenta de usuario en la página de ORS y generar una clave para usar el API; esta clave debe ser reemplazada en la variable BEARER_KEY que se encuentra en el paquete "com.example.apiGarbageSimulation.constants" del servidor de respaldo.

Configuración de una máquina virtual: para ejecutar el software es necesario crear una máquina virtual con un sistema operativo para instalar la máquina virtual de java y un servidor de aplicación, si el sistema va a tener una alta demanda se sugiere el uso de servidores de aplicación compatibles con Java como Nginx o WildFly, se deben a configurar los puertos para la comunicación con las instancias de base de datos y configurar las aplicaciones desde las que la máquina responderá las peticiones y restringir para denegar los servicios en cualquier otro caso.

Puesta en marcha de la base de datos: la aplicación debe persistir la información en la base de datos esta puede alojarse en un servicio de base de datos en la nube, debe ser compatible con postgresql y se debe instalar el complemento para gestión de información geográfica postGis, una vez configurado el motor se puede restaurar el back-up de la base de datos que se utilizó en el desarrollo y sobre el resultado se deben mantener los datos en tablas de configuración y parametrización, las tablas transaccionales deben ser purgadas para no almacenar información de prueba.

Despliegue del Servidor de Respaldo: se requiere la generación en un archivo ejecutable de java con la extensión EAR que contendrá todo la lógica con las que se atenderán las peticiones; el sistema usa algunas dependencias que están siendo gestionadas con Apache Maven, por lo que se requiere dar permisos a la aplicación para descargar y crear paquetes en la máquina en la que se está ejecutando, una vez compilado el ejecutable con sus dependencias; se debe poner en la carpeta de despliegue del servidor seleccionado e iniciar el servidor. En él se deben configurar los puentes de acceso a datos que comunican la aplicación y la base de datos, cuando el servidor inicia se debe validar que la comunicación entre la aplicación y la base de datos sea exitosa.

Cliente MQTT: el cliente MQTT también opera como una aplicación simple e independiente de java, basta con generar un paquete del programa con extensión JAR y ponerlo en el servidor de aplicación seleccionado, este será el encargado de interactuar y suscribirse a la información que está siendo recibido en el Servidor TTN. Es conveniente generar un servicio adicional que permita reiniciar la imagen del cliente MQTT de forma remota con un simple llamado HTTP, pues luego de mucho tiempo de funcionamiento es posible que se generen bloqueos y el cliente deje de responder.

Aplicación Web: para poder ejecutar la aplicación web se requiere instalar un servidor web. Uno bastante ligero y fácil de configurar es Apache Tomcat, para desplegar la aplicación web se deben generar los archivos de ejecución estos quedan almacenados en la carpeta **dist** del proyecto, el contenido de esta carpeta se debe colocar en la carpeta HTTP del servidor Apache. Para que la aplicación pueda ser accedida desde internet se debe comprar y enlazar un dominio de red con la máquina en la que se desplegó el servidor web. Los puntos de enlace (*endpoints*) de la aplicación web deben ajustarse según el dominio del servidor de respaldo para que exista comunicación.

Aplicación Móvil: en cuanto a la aplicación móvil, también deberán ajustarse los archivos de configuración donde se definieron los puntos de enlace según el dominio del servidor de Respaldo, se debe seguir el procedimiento definido en la sección 8.4 para la generación y firma del APK, para la distribución se sugiere gestionar la publicación en la tienda de aplicación de Android (Play Store). Solo los usuarios registrados pueden acceder a la aplicación, actualmente no se dispone de una funcionalidad para el registro de usuarios.

El código fuente con el que se desarrolló este proyecto puede ser consultado en los repositorios que se relacionan a continuación:

Contenido	Dirección
servidor de back-end	https://github.com/montanarco/Tesis-IOT-unal/tree/master/AppBackEnd
aplicación web	https://github.com/montanarco/Tesis-IOT-unal/tree/master/garbageCollectionWeb
aplicación móvil	https://github.com/montanarco/Tesis-IOT-unal/tree/master/App%20Mobil/garbage_bogota
configuración Broker	https://github.com/montanarco/Tesis-IOT-unal/tree/master/Broker%20MQTT
interfaz de simulación	https://github.com/montanarco/Tesis-IOT-unal/tree/master/apiGarbageSimulation

Tabla 8 -1 directorio al repositorio y código fuente de las aplicaciones.

9. Conclusiones y recomendaciones

9.1 Conclusiones

- Los logros obtenidos en el desarrollo de este proyecto fueron: la integración y configuración de 3 circuitos que operan como dispositivos de medición; la configuración de una red de sensores inalámbrica a través de la cual se transporta la telemetría obtenida por los dispositivos de medición; la simulación de un conjunto de sensores reportando niveles de llenado de los contenedores usando información geográfica real; la agrupación de las mediciones simuladas de acuerdo a la ubicación geográfica de los contenedores, su prioridad y la capacidad de carga de los camiones; la integración con un servicio web donde se usa la información simulada para calcular rutas eficientes de recolección y la implementación de una aplicación móvil para mostrar las rutas que generó el servicio de enrutamiento mediante un proveedor de mapas. Con lo cual podemos afirmar que los objetivos establecidos para este trabajo se cumplieron satisfactoriamente.
- El sistema desarrollado logra articular de forma coherente las 4 capas de una solución de IoT, incluyendo la capa física o de sensado, la de transmisión, la de procesamiento de datos y la capa de aplicación. El potencial de esta solución está en la sinergia de las cuatro capas que lo componen pues haciendo uso de las tecnologías y protocolos adecuados que se ensamblan de forma armónica, dando una solución innovadora que deja de percibir la operación de recolección de residuos sólidos como un problema estático y concilia la naturaleza y dinámica y difícil de predecir de la generación de basuras, su valor se puede medir por su aplicabilidad al contexto real.
- Al seleccionar LoRa como tecnología de transmisión, se debe encontrar una configuración que maximice la región de cobertura disminuyendo la cantidad de gateways y que los nodos de medición estén en la región de cobertura de por lo menos dos gateways; esto genera un sistema tolerante a fallos, pues si los nodos quedan en el rango de cobertura de un solo gateway y este falla, por ejemplo; en ausencia de Internet, los nodos en su región de cobertura quedarán aislados. Los gateways deben ser ubicados en lugares elevados por encima de la altura promedio de los edificios circundantes para que exista una línea de visión directa entre ellos y los nodos de medición de manera que se pueda garantizar la recepción debido a que los muros interrumpen la señal.

- La información espacial que se consideró para la agrupación de contenedores no estaba completa porque no se consideraron las fuentes hídricas como humedales y ríos, esto generó problemas de agrupación pues dejó en un mismo grupo contenedores que estaban espacialmente cerca, pero separados por un humedal lo cual aumenta la distancia de las rutas que se calculan en esos grupos y afecta negativamente el rendimiento de la operación. Por lo tanto, se deben tener en cuenta las demás condiciones geográficas del terreno para que las rutas se adapten mejor a las circunstancias reales y sean más eficientes.
- Los datos usados para la generación de rutas son simulados y se usaron solo con fines académicos como pruebas de concepto, al usar información real de los niveles de llenado, las rutas que se obtendrán serán muy diferentes y probablemente sea necesario ajustar los parámetros de agrupación y las restricciones de capacidad para que las rutas calculadas sean eficientes. Por esta razón, no se realizaron análisis comparativos de las rutas generadas frente a las rutas existentes.
- El prototipo que se desarrolló para esta investigación demuestra la viabilidad técnica para la implementación de un esquema de recolección de RSU habilitado por tecnologías IoT e información geográfica en un ambiente real para Bogotá. Se deben evaluar la viabilidad logística y financiera para su implementación, realizar una campaña educativa y de sensibilización, en la que se enseñe a los ciudadanos el uso correcto y los cuidados que se deben tener con los contenedores y los dispositivos de medición instalados en ellos. Una inversión en este tipo de solución genera múltiples beneficios para los ciudadanos y empresas de recolección, algunos de ellos son: calles limpias, separación y aprovechamiento de material reciclable, trabajo digno y economía sostenible, consumo y producción responsable con el medio ambiente, salud y bienestar social. El estudio se limitó a una sola localidad pero su extensión a las demás 19 localidades resulta relativamente sencillo.

9.2 Recomendaciones

Los nodos de medición deben ser mejorados: se debe trabajar en una caja que los proteja de las condiciones climáticas y golpes, se debe diseñar en un módulo de potencia que use de manera eficiente la energía reduciendo el consumo en los momentos en los que el nodo no esté operando; se sugiere que el nodo no integre un módulo de GPS dado que la coordenada geográfica del contenedor puede ser indexada directamente a la base de datos cuando se instala y registra el dispositivo, lo cual reduce el costo de cada nodo en un 33% aproximadamente.

En este proyecto el enrutamiento de los vehículos se realiza por medio de un algoritmo de expansión de árboles que selecciona el orden de recolección desde el centro del grupo hacia las partes externas, este procedimiento fue delegado a un servicio de uso libre, sin embargo existen múltiples algoritmos de enrutamiento que pueden ser implementados y comparar su eficiencia con el fin de encontrar soluciones que disminuyan las distancias globales de los recorridos y la cantidad de camiones, de manera que se aprovechen mejor los recursos y se reduzca el costo de la operación.

Este sistema puede ser complementado usando etiquetas de identificación por radio frecuencia RFID, para que cuando el camión entre en el rango de acción de la etiqueta puesta en el contenedor, el camión emita una señal de confirmación que indique que el proceso de recolección fue completado y actualice la información del contenedor en el servidor sin activar el nodo de medición.

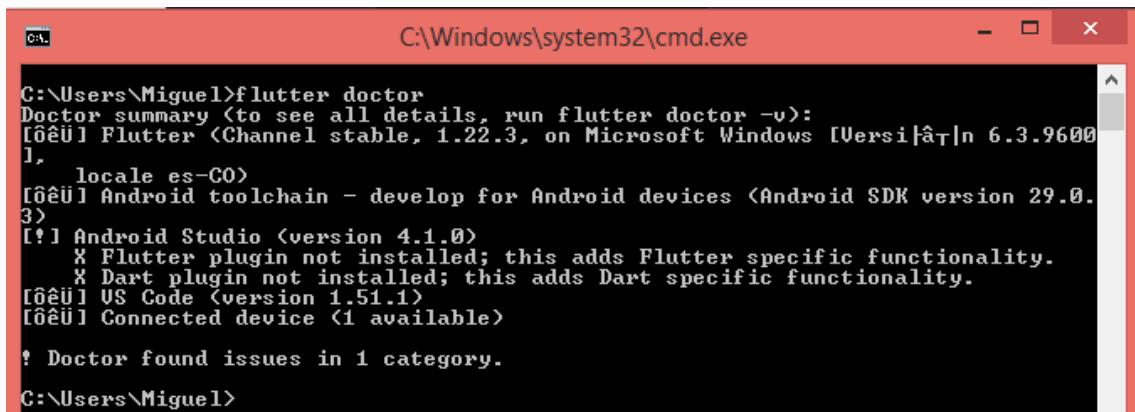
Antes de la puesta en marcha del sistema es necesaria una reubicación de los contenedores; para que por una parte sean de fácil acceso tanto para los ciudadanos como para el personal de reciclaje, pero que además se ubiquen en vías de fácil acceso sin que haya lugar a muchos desvíos, ni pasos por vías estrechas de dificultan el paso del camión y retrasan la operación.

Existen camiones recolectores con compartimentos independientes; la adquisición de esos camiones puede permitir que la separación y transporte de residuos mejore sustancialmente y que la recolección sea más eficiente gracias a que ambos tipos de residuos se pueden recoger en un solo recorrido. También es necesario que se publique información de los depósitos de reciclaje y puntos de acopio autorizados por el gobierno, para poder complementar las rutas dirigiendo los camiones a dichos sitios para que dejen allí el material aprovechable antes de ir a los campos de relleno.

Se deben completar las implementaciones y pruebas para que la aplicación móvil funcione en SO como IOS, para ello también se requiere de dispositivos para desarrollo como equipos MacBook y de prueba como teléfonos con SO IOS.

A. Anexo: Instalación y configuración Flutter

El kit se puede descargar de la página oficial¹⁷ y descomprimirse en una de los directorios de nuestro equipo de cómputo, en SO como Windows es necesario actualizar las variables de entorno para poder ejecutar las sentencia propias del Kit. Flutter dispone de una herramienta de diagnóstico que nos indica si nuestro equipo y sistema dispone de todo los requisitos para ejecutarse correctamente o si es necesario realizar configuraciones adicionales antes de iniciar, el comando es: **Flutter Doctor** y se ejecuta desde la interfaz de comando de nuestro sistema operativo



```
C:\Windows\system32\cmd.exe
C:\Users\Miguel>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[!] Flutter <Channel stable, 1.22.3, on Microsoft Windows [Versi n 6.3.9600]>
  locale es-CO
[!] Android toolchain - develop for Android devices <Android SDK version 29.0.3>
  ! Android Studio <version 4.1.0>
    X Flutter plugin not installed; this adds Flutter specific functionality.
    X Dart plugin not installed; this adds Dart specific functionality.
[!] VS Code <version 1.51.1>
[!] Connected device (1 available)

? Doctor found issues in 1 category.

C:\Users\Miguel>
```

Figura 10-1 Comprobación de la instalación y los requisitos usando flutter doctor.

Adicional a la instalación del Kit, para el desarrollo de aplicaciones es necesario instalar un emulador en el cual podamos probar el código que estamos haciendo, dado que en este caso se está desarrollando una aplicación para android usaremos un emulador con soporte para ese SO.

¹⁷ <https://flutter.dev/docs/get-started/install>

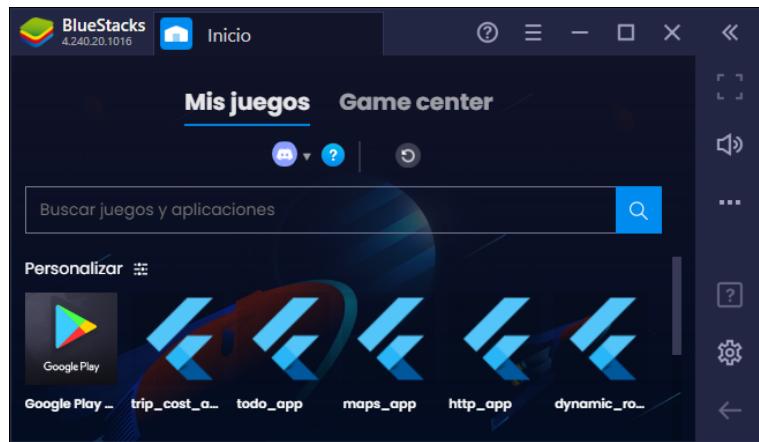


Figura 10-2 Emulador de Android elegido para el desarrollo.

Finalmente se debe establecer un puerto de comunicación entre el compilador de flutter y la máquina virtual del emulador para lo cual se usa un puente llamado Android Debug Bridge (ADB)

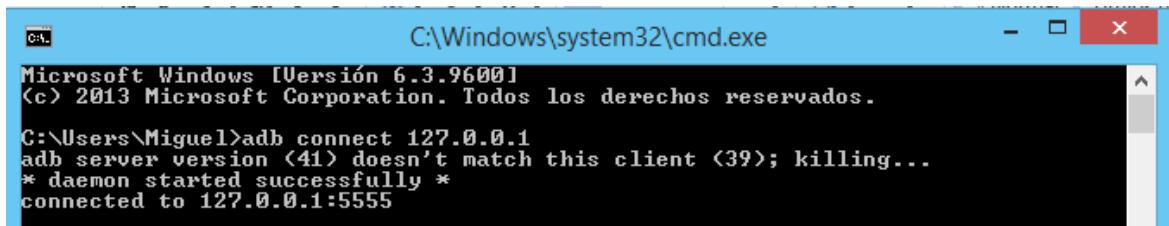


Figura 10-3 Conexión del emulador usando ADB.

Una vez configurado el ambiente de desarrollo se puede crear la primera aplicación flutter ejecutando el comando **flutter create <nombre_aplicacion>** lo cual genera un directorio con todos los archivos para iniciar un proyecto. Para esto resulta útil revisar la documentación de la herramienta disponible en el siguiente enlace:

<https://flutter.dev/docs/get-started/codelab>

B. Anexo: funcionamiento de los nuevos contenedores en Bogotá



Figura 10-4 Infografía funcionamiento de los nuevos contenedores en Bogotá, tomada de (ksestupinan, 2018)

10. Bibliografía

- Abdulkader, M. M. S., Gajpal, Y., & Elmekkawy, T. Y. (2015). Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem. *Applied Soft Computing Journal*, 37, 196–203. <https://doi.org/10.1016/j.asoc.2015.08.020>
- Anagnostopoulos, T., Zaslavsy, A., Medvedev, A., & Khoruzhnicov, S. (2015). Top - k Query Based Dynamic Scheduling for IoT-enabled Smart City Waste Collection. *Proceedings - IEEE International Conference on Mobile Data Management*, 2, 50–55. <https://doi.org/10.1109/MDM.2015.25>
- Anh Khoa, T., Phuc, C. H., Lam, P. D., Nhu, L. M. B., Trong, N. M., Phuong, N. T. H., ... Duc, D. N. M. (2020). Waste Management System Using IoT-Based Machine Learning in University. *Wireless Communications and Mobile Computing*, 2020. <https://doi.org/10.1155/2020/6138637>
- Aslam, A., Mehmood, U., Arshad, M. H., Ishfaq, A., Zaheer, J., Ul Haq Khan, A., & Sufyan, M. (2020, September 1). Dye-sensitized solar cells (DSSCs) as a potential photovoltaic technology for the self-powered internet of things (IoTs) applications. *Solar Energy*, Vol. 207, pp. 874–892. <https://doi.org/10.1016/j.solener.2020.07.029>
- ArcGIS. Georreferenciación y sistemas de coordenadas | ArcGIS Resource Center. (2015, february 12). ArcGIS. Retrieved October 12, 2020, from <https://resources.arcgis.com/es/help/getting-started/articles/026n0000000s000000.htm>
- Arduino Uno Rev3 | Arduino Official Store. (2017). Arduino UNO 2017. <https://store.arduino.cc/arduino-uno-rev3>
- Arteaga Mejia, L. M. (2019, July 30). ¿Qué es el software libre? - Proyecto GNU - Free Software Foundation. GNU. <https://www.gnu.org/philosophy/free-sw.es.html>
- Azzola, F. (2019, May 28). CoAP Protocol: Step-by-Step Guide. Dzone-Coap. <https://dzone.com/articles/coap-protocol-step-by-step-guide>
- Babaei Tirkolaee, E., Abbasian, P., Soltani, M., & Ghaffarian, S. A. (2019). Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study. *Waste Management and Research*, 37(1_suppl), 4–13. <https://doi.org/10.1177/0734242X18807001>
- Beltrami, E. J., & Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, 4(1), 65–94. <https://doi.org/10.1002/net.3230040106>
- Bharadwaj, A. S., Rego, R., & Chowdhury, A. (2017). IoT based solid waste management system: A conceptual approach with an architectural solution as a smart city application. *2016 IEEE Annual India Conference, INDICON 2016*. <https://doi.org/10.1109/INDICON.2016.7839147>
- Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54, 1–31. <https://doi.org/10.1016/j.comcom.2014.09.008>

- Brandão, J. (2004). A tabu search algorithm for the open vehicle routing problem. European Journal of Operational Research, 157(3), 552–564. [https://doi.org/10.1016/S0377-2217\(03\)00238-8](https://doi.org/10.1016/S0377-2217(03)00238-8)
- Bustos Coral, D., Oliveira Santos, M., Toledo, C., & Fernando Niño, L. (2018). Clustering-Based Search in a Memetic Algorithm for the Vehicle Routing Problem with Time Windows. 2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings, 8. <https://doi.org/10.1109/CEC.2018.8477710>
- Cabrera, A. G. (2010.). Simulación de procesos constructivos Simulation of constructive processes * Pontificia Universidad Javeriana. COLOMBIA. Retrieved from www.ing.puc.cl/ric
- Campbell, A. M., & Wilson, J. H. (2014). Forty years of periodic vehicle routing. Networks, 63(1), 2–15. <https://doi.org/10.1002/net.21527>
- Cerchecci, M., Luti, F., Mecocci, A., Parrino, S., Peruzzi, G., & Pozzebon, A. (2018). A low power IoT sensor node architecture for waste management within smart cities context. Sensors (Switzerland), 18(4). <https://doi.org/10.3390/s18041282>
- Chaudhari, S. S., & Bhole, V. Y. (2018). Solid Waste Collection as a Service using IoT-Solution for Smart Cities. 2018 International Conference on Smart City and Emerging Technology, ICSCET 2018. <https://doi.org/10.1109/ICSCET.2018.8537326>
- Chaudhary, S., Nidhi, C., & Rawal, N. R. (2019). Gis-based model for optimal collection and transportation system for solid waste in allahabad city. Advances in Intelligent Systems and Computing, 814, 45–65. https://doi.org/10.1007/978-981-13-1501-5_5
- Chen, C. Y., Hasan, M., & Mohan, S. (2018). Securing real-time internet-of-things. Sensors (Switzerland), 18(12), [4356]. <https://doi.org/10.3390/s18124356>
- Cope, S. (2020, 10 September). How MQTT Works -Beginners Guide, steves internet guide. <http://www.steves-internet-guide.com/mqtt-works/>
- Coupey, J. (2018, August 17). Solving vehicle routing problems with OpenStreetMap and VROOM. Stateofthemap. https://2018.stateofthemap.org/2018/T053-Solving_vehicle_routing_problems_with_OpenStreetMap_and_VROOM/
- DataCentric. Cómo distinguir entre geolocalización y georeferenciación. (2018, March 3). <https://www.datacentric.es/blog/geomarketing/diferencia-entre-geolocalizacion-y-geo-referenciacion/>
- Dragino GPS. LoRaWAN GPS Tracker with 9-axis accelerometer-LGT92. (2019, August 11). Dragino GPS. <https://www.dragino.com/products/lora-lorawan-end-node/item/142-lgt-92.html>
- Dragino Distance. LDDS75 LoRaWAN Distance Detection Sensor. (2020, June 10). <https://www.dragino.com/products/lora-lorawan-end-node/item/161-dds75.html>
- Dragino LG02. LG02 Dual Channels LoRa IoT Gateway. (2019, October 31). <https://www.dragino.com/products/lora-lorawan-gateway/item/135-lg02.html>
- El Espectador -Redacción Bogotá - bogota@elespectador.com. (2020, July 2). Listos los

- contenedores, solo falta saber dónde ubicarlos.
<https://www.elespectador.com/noticias/bogota/listos-los-contenedores-solo-falta-saber-donde-ubicarlos/>
- Evans, D. (2011). The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Retrieved from
https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
- Fatimah, Y. A., Govindan, K., Murniningsih, R., & Setiawan, A. (2020). Industry 4.0 based sustainable circular economy approach for smart waste management system to achieve sustainable development goals: A case study of Indonesia. *Journal of Cleaner Production*, 269, 122263. <https://doi.org/10.1016/j.jclepro.2020.122263>
- Franca, L. S., Ribeiro, G. M., & Chaves, G. de L. D. (2019). The planning of selective collection in a real-life vehicle routing problem: A case in Rio de Janeiro. *Sustainable Cities and Society*, 47, 101488. <https://doi.org/10.1016/j.scs.2019.101488>
- Franco, F. M. (2018, August 13). Así funcionan los nuevos camiones del esquema de aseo de Bogotá. *El Tiempo- Capacidad Camiones*.
<https://www.eltiempo.com/bogota/caracteristicas-de-los-camiones-de-basura-nuevos-en-bogota-255354>
- Fujdiak, R., Masek, P., Mlynek, P., Misurec, J., & Olshannikova, E. (2016). Using genetic algorithm for advanced municipal waste collection in Smart City. 2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2016. <https://doi.org/10.1109/CSNDSP.2016.7574016>
- García, S., Larios, D. F., Barbancho, J., Personal, E., Mora-Merchán, J. M., & León, C. (2019). Heterogeneous LoRa-Based Wireless Multimedia Sensor Network Multiprocessor Platform for Environmental Monitoring. *Sensors*, 19(16), 3446. <https://doi.org/10.3390/s19163446>
- GNU-Copyleft. ¿Qué es el copyleft? - Proyecto GNU - Free Software Foundation. (2019, September 15). <https://www.gnu.org/licenses/copyleft.es.html>
- Gottinger, H. W. (1986). A computational model for solid waste management with applications. *Applied Mathematical Modelling*, 10(5), 330–338. [https://doi.org/10.1016/0307-904X\(86\)90092-2](https://doi.org/10.1016/0307-904X(86)90092-2)
- Gove, R. (2017, December 26). Using the elbow method to determine the optimal number of clusters for k-means clustering. Robert Gove's Block.
<https://blockrocks.org/rpgove/0060ff3b656618e9136b>
- GPS-US.Gov. Bienvenidos a GPS.gov. Retrieved October 14, 2020, from
<https://www.gps.gov/spanish.php>
- Gresak, E., & Voznak, M. (2020). Protecting gateway from ABP replay attack on LoRaWAN. *Lecture Notes in Electrical Engineering*, 554, 400–408. https://doi.org/10.1007/978-3-030-14907-9_39
- Heidelberg University. (1997, February 19). TSPLIB. TSPLIB.
<http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html>

- Hoplin, I. (2019, August 6). MQTT: un protocolo abierto de red y su importancia en el IoT. Open-MQTT. <https://www.hwlibre.com/mqtt/>
- Ideca-Mapas. Infraestructura de Datos Espaciales de Bogotá. (2019, June 10) Retrieved October 7, 2020, from
https://www.ideca.gov.co/busador?topic=All&metadata=All&newest=All&entity=All&resource=All&content_type=map&res=true
- IONOS España S.L.U. (2020, September 30). Advanced Message Queuing Protocol (AMQP). IONOS Digital Guide.
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/advanced-message-queuing-protocol-amqp/>
- Jatinkumar Shah, P., Anagnostopoulos, T., Zaslavsky, A., & Behdad, S. (2018). A stochastic optimization framework for planning of waste collection and value recovery operations in smart and sustainable cities. *Waste Management*, 78, 104–114.
<https://doi.org/10.1016/j.wasman.2018.05.019>
- Jie Chen, Dr. L Ramanathan, Dr. Mamoun Alazab, Holistic Big Data Integrated Artificial Intelligent Modeling to Improve Privacy and Security in Data Management of Smart Cities, *Microprocessors and Microsystems*, 2020, 103722, ISSN 0141-9331,
<https://doi.org/10.1016/j.micpro.2020.103722>
- Jwad, Z. A., & Hasson, S. T. (2018). An Optimization Approach for Waste Collection Routes Based on GIS in Hillah-Iraq. ICOASE 2018 - International Conference on Advanced Science and Engineering, 60–63.
<https://doi.org/10.1109/ICOASE.2018.8548889>
- Kang, K. D., Kang, H., Ilankoon, I. M. S. K., & Chong, C. Y. (2020). Electronic waste collection systems using Internet of Things (IoT): Household electronic waste management in Malaysia. *Journal of Cleaner Production*, 252.
<https://doi.org/10.1016/j.jclepro.2019.119801>
- Kantorovich, A. (2013). An Evolutionary View of Science: Imitation and Memetics.
- Karthikeyan, S., Rani, G. S., Sridevi, M., & Bhuvaneswari, P. T. V. (2018). IoT enabled waste management system using ZigBee network. RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings, 2018-Janua, 2182–2187.
<https://doi.org/10.1109/RTEICT.2017.8256987>
- Kristanto, S., Yashiro, T., Koshizuka, N., & Sakamura, K. (2016). Dynamic polling algorithm for low energy garbage level measurement in smart trash bin. ACM International Conference Proceeding Series, 24-25-May-2016, 92–94.
<https://doi.org/10.1145/2962735.2962748>
- ksestupinan. (2018, October 28). Bogotá está instalando 10.000 contenedores. Bogota-Asi-Vamos.
<https://bogota.gov.co/asi-vamos/contenedores-para-residuos-en-bogota>
- Kshetri, N. (2017). An opinion on the “Report on Securing and Growing the Digital Economy.” *IEEE Security and Privacy*, 15(1), 80–85.

- <https://doi.org/10.1109/MSP.2017.10>
- Kumar, N. S., Vuayalakshmi, B., Prarthana, R. J., & Shankar, A. (2017). IOT based smart garbage alert system using Arduino UNO. IEEE Region 10 Annual International Conference, Proceedings/TENCON, 1028–1034.
<https://doi.org/10.1109/TENCON.2016.7848162>
- Kumar, R., & Pallikonda Rajasekaran, M. (2016). An IoT based patient monitoring system using raspberry Pi. 2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE 2016.
<https://doi.org/10.1109/ICCTIDE.2016.7725378>
- Kuo, R. J., & Zulvia, F. E. (2017). Hybrid genetic ant colony optimization algorithm for capacitated vehicle routing problem with fuzzy demand - A case study on garbage collection system. 2017 4th International Conference on Industrial Engineering and Applications, ICIEA 2017, 244–248. <https://doi.org/10.1109/IEA.2017.7939215>
- Lee, R. (2017). Build Lora node using Arduino Uno and SX1276. Retrieved September 21, 2020, from mobilefish website:
https://www.mobilefish.com/developer/lorawan lorawan_quickguide_build_lora_node_rfm95_arduino_uno.html
- Lella, J., Mandla, V. R., & Zhu, X. (2017). Solid waste collection/transport optimization and vegetation land cover estimation using Geographic Information System (GIS): A case study of a proposed smart-city. Sustainable Cities and Society, 35, 336–349.
<https://doi.org/10.1016/j.scs.2017.08.023>
- Lilygo-Xinyuan. Company Profile - Shenzhen Xin Yuan Electronic Technology Co., Ltd. (2017, May 12). <http://www.lilygo.cn/about.aspx?TypeId=1&FId=t1:1:1>
- Lopes, R. B., Plastria, F., Ferreira, C., & Santos, B. S. (2014). Location-arc routing problem: Heuristic approaches and test instances. Computers and Operations Research, 43, 309–317. <https://doi.org/10.1016/j.cor.2013.10.003>
- Manglorkar, S. S., Sharma, A. O., Verma, D. S., & Rane, S. B. (2019). Optimization of Organic Waste Collection for Generation of Bio Gas using IoT Techniques. IOP Conference Series: Materials Science and Engineering, 594(1).
<https://doi.org/10.1088/1757-899X/594/1/012026>
- Marmolejo, L. F. (2011). Análisis del funcionamiento de plantas de manejo de residuos sólidos en el norte del valle del cauca, Colombia. Retrieved October 12, 2020, from revista EIA website:
http://www.scielo.org.co/scielo.php?pid=S1794-12372011000200013&script=sci_abs tract&tlang=es
- Matthijs kooijman. (2015). arduino-lmic: LoraWAN-in-C library. Retrieved September 21, 2020, from <https://github.com/matthijskooijman/arduino-lmic>
- Meesala, Sirisha & Manoj Kumar, Nallapaneni & Parimala, Sugathi & Jyothi, N. Arun. (2018). Monitoring the smart garbage bin filling status: An IoT application towards waste management. International Journal of Civil Engineering and Technology. 9. 373-381.

- Miessler, D., Guzman, A., Rudresh, V., Smith, C. (2018) OWASP Internet of Things, owasp.org from: <https://owasp.org/www-project-internet-of-things/>
- Moran, M. (2013, March 20). Ciudades. ONU-Desarrollo Sostenible-cuidades. <https://www.un.org/sustainabledevelopment/es/cities/>
- MORANTE, A. (2018, February 23). Las basuras, un asunto de toda la ciudadanía. El Tiempo. <https://www.eltiempo.com/bogota/reciclaje-y-recolección-de-basura-en-bogot-a-186476>
- MQTT - The Standard for IoT Messaging. (2020, May 3). MQTT. <https://mqtt.org/>
- MSV, J. (2019, 10 mayo). 10 DIY Development Boards for IoT Prototyping. The New Stack. <https://thenewstack.io/10-diy-development-boards-iot-prototyping/>
- Naylampmechatronics. Tutorial Módulo GPS con Arduino. (2016, June 25). https://naylampmechatronics.com/blog/18_Tutorial-M%C3%B3dulo-GPS-con-Arduino.html
- OWASP Foundation | Open Source Foundation for Application Security. OWASP. Retrieved October 16, 2020, from <https://owasp.org/about/>
- O.R.S. (2019, May 11). Services | Openrouteservice. ORS-Services. Retrieved November 6, 2020, from: <https://openrouteservice.org/services/>
- PARRA, H. (2018, July 20). *5.000 contenedores en calle serán para material reciclado*. El Tiempo. <https://www.eltiempo.com/bogota/instalaran-contenedores-de-basura-en-las-calles-de-bogota-245866>
- Papalitsas, C., Karakostas, P., Andronikos, T., Sioutas, S., & Giannakis, K. (2018). Combinatorial GVNS (General Variable Neighborhood Search) Optimization for Dynamic Garbage Collection. *Algorithms*, 11(4), 38. <https://doi.org/10.3390/a11040038>
- Pocero, L., Amaxilatis, D., Mylonas, G., & Chatzigiannakis, I. (2017). Open source IoT meter devices for smart and energy-efficient school buildings. *HardwareX*, 1, 54–67. <https://doi.org/10.1016/j.hx.2017.02.002>
- PostgreSQL Global Development Group. (2021, February 11). *PostgreSQL: License*. PostgreSQL ORG. <https://www.postgresql.org/about/licence/>
- Qoitech, T. (2019, November 18). How Spreading Factor Affects LoRaWAN device battery life. The Things Network. <https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life>
- Raftery, H. (2018, 26 febrero). LoRaWAN: OTAA or ABP? NewieVentures. <https://www.newieventures.com.au/blogtext/2018/2/26/lorawan-otaa-or-abp>
- Ray, P. P. (2017). Internet of things for smart agriculture: Technologies, practices and future direction. *Journal of Ambient Intelligence and Smart Environments*, 9(4), 395–420. <https://doi.org/10.3233/AIS-170440>

- Ray, S., Tapadar, S., Chatterjee, S. K., Karlose, R., Saha, S., & Saha, H. N. (2018). Optimizing routine collection efficiency in IoT based garbage collection monitoring systems. 2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018, 2018-Janua, 84–90.
<https://doi.org/10.1109/CCWC.2018.8301629>
- Rojas Calderón, L. F. (2018). 123. El nuevo modelo de recolección de basuras en Bogotá ¿populismo saliente o monopolización entrante? - FCE. Retrieved October 14, 2020, from Econografos Escuela de Economía No 123 website:
<http://www.fce.unal.edu.co/centro-editorial/documentos/econografos-escuela-economia/2040-123-el-nuevo-modelo-de-recoleccion-de-basuras-en-bogota-populismo-saliente-o-monopolizacion-entrante.html>
- Santos Filho, F. H. C. dos, Dester, P. S., Stanganelli, E. M. G., Cardieri, P., Nardelli, P. H. J., Carrillo, D., & Alves, H. (2020). Performance of LoRaWAN for Handling Telemetry and Alarm Messages in Industrial Applications. *Sensors*, 20(11), 3061.
<https://doi.org/10.3390/s20113061>
- semtech. (2015). LoRa family tecnical detail. Retrieved September 21, 2020, from https://cdn-shop.adafruit.com/product-files/3179/sx1276_77_78_79.pdf
- Serrano, A. (2016, March 21). *Los Sistemas de Comunicación*. Area Tecnologia-Comunicaciones.
<https://www.areatecnologia.com/los-sistemas-de-comunicacion.htm>
- Shakdhe, Arjun & Agrawal, Suyash & Yang, Baijian. (2019). Security Vulnerabilities in Consumer IoT Applications. 1-6. 10.1109/BigDataSecurity-HPSC-IDS.2019.00012.
- Skoglund, P. (2019, March 31). Suecia recicla un asombroso 99 % de su basura. EcoInventos. <https://ecoinventos.com/suecia-recicla-un-asombroso-99-de-su-basura/>
- Smith, C. (2020, April 5). *OWASP Internet of Things*. OWASP-IOT.
<https://owasp.org/www-project-internet-of-things/>
- Solera, E. (2018, 27 agosto). Modulación LoRa: Long Range Modulation - Exploración de aplicaciones usando tecnología LoRa. Medium.
<https://medium.com/pruebas-de-laboratorio-de-la-modulacion-lora/modulacion-lora-4-ad74cabd59e>
- Stallman, R. M. (2019, December 30). Free Hardware and Free Hardware Designs - GNU Project - Free Software Foundation. GNU-Free-Hardware.
<https://www.gnu.org/philosophy/free-hardware-designs.html>
- Stark, W. E., Borth, D. E., & Lehnert, J. S. (2019). telecommunication. Retrieved October 15, 2020, from Enciclopedia Britannica website:
<https://www.britannica.com/technology/telecommunication>
- Suarez, C. (2018, September 3). Bogotá produce 6.300 toneladas de basura al día. Residuos y Reciclaje Bogotá.
<http://concejodebogota.gov.co/bogota-produce-6-300-toneladas-de-basura-al-dia/cbogota/2018-09-03/134429.php>
- Tecnología Inalámbrica. Qué es, Funcionamiento, Tipos y Comunicación Inalámbrica.

- (2017, November 19).
<https://www.atecnologia.com/informatica/tecnologia-inalambrica.html#:~:text=La%20tecnolog%C3%A1%20inal%C3%A1mbrica%20es%20la,tipo%20ni%20otro~s%20medios%20f%C3%ADos%20c%C3%ADasicos>.
- The Things Network. (2017, August 30). *LoRaWAN Architecture*.
<https://www.thethingsnetwork.org/docs/lorawan/architecture/index.html>
- Vangelista, L. (2017). Frequency Shift Chirp Modulation: The LoRa Modulation. *IEEE Signal Processing Letters*, 24(12), 1818–1821.
<https://doi.org/10.1109/LSP.2017.2762960>
- Verdouw, C. N., Beulens, A. J. M., & van der Vorst, J. G. A. J. (2013). Virtualisation of floricultural supply chains: A review from an internet of things perspective. *Computers and Electronics in Agriculture*, 99, 160–175.
<https://doi.org/10.1016/j.compag.2013.09.006>
- VOLOV, Alexander, V. (2016, June 13). Advanced Message Queuing Protocol (AMQP) | Alexander Volov's homepage. <http://alexvolov.com/2016/06/amqp/>
- Vu, D. D., & Kaddoum, G. (2017). A waste city management system for smart cities applications. *Proceedings - 2017 Advances in Wireless and Optical Communications, RTUWO 2017*, 2017-January, 225–229.
<https://doi.org/10.1109/RTUWO.2017.8228538>
- What is Arduino? (2018, 5 febrero). Arduino cc.
<https://www.arduino.cc/en/Guide/Introduction>
- Wu, B. Y., Lancia, G., Bafna, V., Chao, K. M., Ravi, R., & Tang, C. Y. (2000). A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM Journal on Computing*, 29(3), 761–778.
<https://doi.org/10.1137/S009753979732253X>
- Zhong, R. Y., Lan, S., Xu, C., Dai, Q., & Huang, G. Q. (2016). Visualization of RFID-enabled shopfloor logistics Big Data in Cloud Manufacturing. *International Journal of Advanced Manufacturing Technology*, 84(1–4), 5–16.
<https://doi.org/10.1007/s00170-015-7702-1>
- Zhou, Qing & Benlic, Una & Wu, Qinghua & Hao, Jin-Kao. (2018). Heuristic search to the capacitated clustering problem. *European Journal of Operational Research*. 273.
<https://doi.org/10.1016/j.ejor.2018.08.043>.