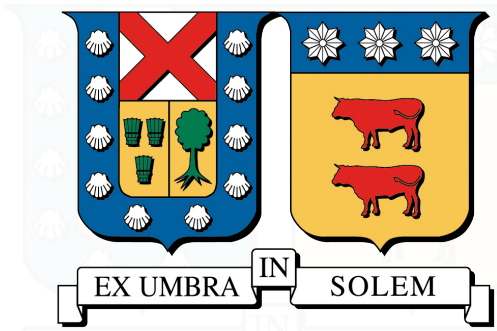
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



LoRa Based Communication for Internet of Electric Vehicles

RUDOLF ALEXANDER HARTMANN VALENZUELA

Memoria para optar el título de
INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA	:	MOHAMED ABDELHAMID
PROFESOR CORREFERENTE	:	JOSÉ MANUEL MARTINEZ

DICIEMBRE 2023

A mis padres, que me ayudaron e impulsaron a lograr mis metas.

A mis amigos, que compartieron mis alegrías y frustraciones.

A mi pareja Judit, que estuvo ahí tanto en los momentos buenos y como en los difíciles.

A todas aquellas personas que sin importar las tempestad, siempre están ahí, que dan apoyo y alegría incondicional. Para cada uno de ellos, muchas gracias

Agradecimientos

Agradezco al profesor Mohamed por su enorme paciencia, voluntad y compromiso al guiarme durante este trabajo de título. Agradezco también al centro estudiantil Burschenschaft Ripuaria, por entregarme herramientas y compañeros increíbles en esta gran etapa de mi vida y por ser una de los lugares más extraordinarios para vivir la vida universitaria. Agradezco igualmente a todas las personas que me acompañaron durante este proceso de estudio, todos aquellos que lograron ayudarme, motivarme y enseñarme a descubrir cosas nuevas, las que comparten mis pasiones y sueños. A todos aquellos que en el viaje me acompañaron a disfrutar de los desvíos, y mostrarme que estos pueden llegar a ser más importantes que el destino mismo. A todas ellas, muchas gracias.

Además me gustaría agradecer a la Agencia Nacional de Investigación y Desarrollo (ANID), Proyecto Fondecyt de Iniciación en Investigación 2020, Project ID 11200178, por el apoyo a esta investigación.

RESUMEN EJECUTIVO

Tener un vehículo eléctrico hoy en día es una alternativa poco conveniente en un país como Chile, esto se debe a que el proceso de masificación de los centros de carga (electrolineras) a lo largo del país se ha ralentizado debido a la pandemia del 2020-2021 y la recesión económica que le siguió en consecuencia. Sin embargo, el interés de la población en disminuir la emisión de gases de efecto invernadero y reducir la huella de carbono individual sigue motivando a personas para comprar un vehículo eléctrico a pesar de todos estos obstáculos.

Durante este trabajo de título se analizará el estado actual de Chile en relación a la electromovilidad, y el efecto que tiene este sobre los usuarios de vehículos eléctricos. La solución propuesta consta de una implementación en hardware y software que trabajarán en conjunto para llevar a cabo al comunicación y procesamiento de datos necesarios para entregar un servicio satisfactorio a los usuarios de vehículos eléctricos del futuro. Este servicio permitirá a los usuarios buscar, comparar y escoger un centro de carga que se acomode a sus necesidades a través de un ”recomendador” alojado en el servidor web de la aplicación.

Para implementar la solución propuesta se dividió el trabajo en tres secciones claves:

La primera sección consiste en la implementación de un sistema de comunicación inalámbrica entre los vehículos eléctricos y/o estaciones de carga con las puertas de enlace (gateways) que permiten la salida y entrada de la información hacia internet basado en tecnologías LoRa. Este sistema utiliza un controlador Heltec LoRa 32 V2 que tiene integrado un módulo LoRa, este es necesario para realizar la comunicación entre vehículos. Por otro lado está el “LoRa gateway” que usando el protocolo LoRaWAN permitirá recibir y convertir esta información al formato estándar de la internet y vice versa.

La segunda sección consiste en el desarrollo de una API REST la cual permitirá responder las peticiones de los usuarios y acceder a la información almacenada en la base de datos. Además tendrá alojado el servicio de recomendación, este permitirá al usuario

buscar las electrolineras cercanas que se ajusten a sus intereses teniendo en consideración los siguientes parámetros: el costo monetario, el tiempo de espera (en la cola), velocidad de carga y la distancia. Las electrolineras candidatas primero son filtradas por la distancia máxima que podría recorrer el vehículo con su energía restante y los tipos de conectores que son compatibles con este. A todo esto se le suma un simulador que permitirá hacer las pruebas de concepto y de esta manera validar el trabajo realizado.

Finalmente la tercera sección consiste en el desarrollo de una aplicación móvil que permitirá acceder a un mapa nacional con todas las electrolineras disponibles a lo largo del país, además del acceso y creación a perfiles personalizados que permitirán modificar los valores de los parámetros del ”recomendador”. Así mismo, luego de solicitar la recomendación permitirá al usuario seleccionar entre las 10 electrolineras con menor costo (general) e indicar la ruta hacia estos nuevos destinos.

NaVE (Navegación para Vehículo eléctrico), como fue nombrada la aplicación, considera los principales problemas que enfrentan los usuarios de vehículos eléctricos y presenta una solución que no se encuentra vigente hoy en Chile, permitiendo a las personas que utilicen vehículos eléctricos disfrutar de un servicio que facilitará sus vidas, disminuirá el nivel ansiedad experimentado y con ello motivará a más personas a cambiar su vehículo convencional por uno eléctrico o híbrido.

Este trabajo de título se enfocará en abordar la primera sección a fondo, junto a todas sus problemáticas y desafíos.

ABSTRACT

Having electric vehicles today is an alternative not so convenient in a country as Chile, this is because the massification process of the electric charging stations across the country has been slowed due to the 2020-2021 pandemic and in consequence, the economic recession. However, the interest for lowering the greenhouse gases emissions and reducing the individual carbon footprint in the community is motivating more people to take the decision of buying an electric vehicle despite the obstacles.

During this work, we will be analyzing the current state of electromobility in Chile and the effects that it haves on electric vehicle users. The proposed solution consists of a hardware and software implementation that will work together to carry out the communication and processes needed to deliver a satisfactory service to electric vehicle users. This service will allow users to search, compare and choose a charging station that suits their needs through a "recommender" hosted on the application's web server.

To implement the proposed solution, the work was divided into three key sections:

The first section involves the implementation of a wireless communication network between electric vehicles and/or chargin stations with gateways that allows receiving and transmiting information to the Internet based on LoRa technology. This infrastructure considered the use of a Heltec LoRa 32 V2 micro-controller that has an integrated LoRa module, needed for communication. On the other hand there is the LoRa gateway that using the LoRaWAN protocol wich will convert this information to the standard Internet format and vice versa.

The second section consists of the development of a REST API which will allow users to respond to requests and access the information stored in the database. In addition, the referral service will be available, which will allow the user to search for the electric charging station that suit their interests in terms of charging cost, waiting time (in line), charging speed and distance. However, filtering the charging stations first by the maximum distance that the user's vehicle could travel with the remaining battery of their vehicle and the types of connectors that are compatible with their vehicle.

Finally, the third section consists of the development of a user interface that will allow access to a national map with all the electric charging stations available throughout the country, as well as access to profiles and creation of this profiles where it will be possible to modify the level of interest of each of the characteristics considered by the referral service. It will also allow the user to choose among the 10 charging stations with the lowest cost (general) and generate the route to these new destinations in addition to the final destination of the journey.

NaVE, as the app was named, considers the main problems faced by electric vehicle users and presents a solution that is not currently in force in Chile, allowing these people to use a service that will make their lives easier, decrease their anxiety levels and therefor motivate more people to switch from conventional to electric and/or hybrid vehicles.

This work will focus on the first section in depth, along with all its problems and challenges.

Glosario

- **Back-End:** Sector de un sistema informático responsable de almacenar y manipular datos sensibles, y a su vez ofrecer servicios útiles para el usuario final.
- **Front-End:** Sector de un sistema informático que muestra la información al usuario e interactúa directamente con este a través de aplicaciones con interfaz gráfica.
- **Heltec LoRa 32 v2:** Microcontrolador con un microchip ESP32 + SX127x, el cual tiene integrado módulos Wi-Fi, LoRa, BLE y una pantalla OLED de 0,96".
- **LoRa:** LoRa es una tecnología inalámbrica que utiliza un tipo de modulación llamado Chirp Spread Spectrum (CSS). Este tipo de modulación le da la capacidad de cubrir grandes distancias y ser altamente tolerable a interferencias.
- **LoRaWAN:** LoRaWAN es un protocolo de red punto a multipunto que utiliza el esquema de modulación LoRa de Semtech.
- **GPS:** Sistema de Posicionamiento Global, permite fijar a escala mundial la posición de un objeto, una persona o vehículo.
- **PostgreSQL:** Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.
- **Raspberry Pi:** Ordenador de bajo costo y de tamaño compacto con la posibilidad de conexión Wi-Fi, y con la habilidad de interactuar con el mundo exterior por medio de módulos.
- **Flutter:** Framework de Google basado en el lenguaje de programación Dart, pensado para programación de aplicaciones móviles y web basado en Widgets.
- **Django:** Entorno de ejecución de Python de código abierto y multiplataforma.
- **API:** Conjunto de funciones y procedimientos que permiten acceder a las características o datos de un servidor web.
- **JSON:** Estándar de formato de archivos e intercambio de información.

- **Endpoint:** URL a través del que se accede a alguna API.
- **HTTP:** Protocolo que permite realizar una petición de datos y recursos a algún servidor web.
- **Gateway:** Dispositivo utilizado para conectar dos redes diferentes, especialmente una conexión a Internet. Su traducción: Puerta de enlace.
- **TTN:** *The Things Network*. Plataforma de código abierto para redes de área amplia de bajo consumo de energía (LPWAN) que permite la conectividad de dispositivos de Internet de las cosas (IoT)
- **OTAA:** $Over - The - Air - Activation$. Método de activación en redes LoRaWAN que permite la configuración remota y segura de dispositivos IoT. El dispositivo envía una solicitud de activación al servidor LoRaWAN, que responde con información de configuración necesaria para la conexión.
- **ABP:** $Activation - By - Personalitation$. Método de activación en redes LoRaWAN que permite la configuración remota y segura de dispositivos IoT. Utiliza claves preconfiguradas almacenadas en el dispositivo y en el servidor.
- **DevAddr:** Dirección de dispositivo de 32 bits.
- **DevEUI:** Identificador único de 64 bits para nodos LoRa.
- **AppEUI:** Identificador único de 64 bits para aplicaciones LoRa.
- **GatewayEUI:** Identificador único de 64 bits para gateways LoRa.
- **AppSKey:** Llave de sesión de la aplicación. Llave privada usada para encriptación y desencriptación del mensaje en conexiones nodo y aplicación LoRa en los servidores de The Things Network.
- **NwkSKey:** Llave de sesión de la red. Llave pública usada para verificar integridad de cada mensaje entre conexiones nodo y servidor de red LoRa en The Things Network.

- **AppKey:** Llave de aplicación. Llave solo conocida por el nodo y la aplicación LoRa en los servidores de The Things Network. Usada para derivar las llaves AppSKey y NwkSKey en cada sesión.
- **SF:** *Spreading Factor*, traducido al español es factor de dispersión.
- **BW:** *Band Width*, traducido al español es ancho de banda.
- **RSSI:** *Received Signal Strength Indicator*, traducido al español es indicador de fuerza de señal recibida.
- **SNR:** *Signal to Noise Ratio*, traducido al español es ratio de señal a ruido.
- **MQTT:** *Message Queuing Telemetry Transport*. Se trata de un protocolo de mensajería ligero para usar en casos de clientes que necesitan una huella de código pequeña, que están conectados a redes no fiables o con recursos limitados en cuanto al ancho de banda
- **Node-Red:** Herramienta construida sobre Node.js para conectar dispositivos, servicios y APIs de manera visual y sencilla a través de nodos y flujos visibles.

Índice de Contenidos

1. Introducción **1**
- 1.1. ¿Como es actualmente? . . . 2
- 1.2. ¿Cuál es nuestro desafío? . . . 2
  - 1.2.1. Análisis del desafío y alternativas de problemas específicos por resolver . . . 2
- 1.3. Acercamiento a la solución . . . 5
- 1.4. Objetivo General . . . 6
- 1.5. Objetivos Específicos . . . 7
- 1.6. Estructura del documento . . . 7

2. Marco Teórico **9**
- 2.1. ¿Qué es una plataforma de navegación? . . . 9
- 2.2. Automóviles eléctricos en Chile . . . 9
- 2.3. Electrolineras en Chile . . . 10
- 2.4. Servicio de recomendación . . . 12

3. Estado del arte **14**
- 3.1. Comunicación . . . 14
- 3.2. Middleware . . . 21
- 3.3. Interfaz de usuario . . . 30

4. Arquitectura de la Plataforma **36**
- 4.1. Capa de Percepción . . . 37
- 4.2. Capa de Comunicación . . . 37
- 4.3. Capa Middleware . . . 38
- 4.4. Capa de Aplicación . . . 39
- 4.5. Requisitos del Sistema . . . 40
  - 4.5.1. Requisitos Funcionales . . . 40
  - 4.5.2. Requisitos No Funcionales . . . 40
  - 4.5.3. Requisitos de Interfaces . . . 41
  - 4.5.4. Requisitos de Ambiente . . . 43
- 4.6. Perfiles de Usuario . . . 44
- 4.7. Propuesta de Solución . . . 44

5. Diseño de prototipo y resultados **47**

5.1. Implementación . . . 48
5.1.1. Hardware . . . 53
5.1.1.1. Gateway LoRa . . . 53
5.1.1.2. Nodo LoRa . . . 54
5.1.2. Software . . . 55
5.1.2.1. *Gateway* . . . 55
5.1.2.2. Servidores . . . 62
5.1.2.3. Nodo LoRa . . . 63
5.2. Resultados . . . 64
5.2.1. Escenario 1: . . . 64
5.2.2. Escenario 2: . . . 68
5.2.3. Escenario 3: . . . 71
5.2.4. Escenario 4: . . . 73

6. Simulación y resultados **76**
6.1. Desarrollo . . . 78
6.1.1. Primer objetivo: . . . 78
6.1.2. Segundo objetivo: . . . 83
6.1.2.1. Escenario 1 . . . 85
6.1.2.2. Escenario 2 . . . 86
6.1.2.3. Escenario 3 . . . 87
6.2. Resultados . . . 89
6.2.1. Primer objetivo: . . . 89
6.2.2. Segundo objetivo: . . . 90
6.2.2.1. Escenario 1: . . . 90
6.2.2.2. Escenario 2: . . . 92
6.2.2.3. Escenario 3: . . . 93

7. Conclusión **96**

Bibliografía **100**

A. Apéndice **103**
A.1. Códigos . . . 103

Índice de Tablas

3.1. Trabajos previos capa percepción 20
3.2. Trabajos previos capa middleware 28
3.3. Trabajos previos capa middleware 29
3.4. Trabajos previos capa middleware 30
3.5. Trabajos previos capa aplicación 34
3.6. Trabajos previos capa aplicación 35

4.1. Eventos externos 42
4.2. Respuestas del sistema 43
4.3. Perfiles de Usuario 44

5.1. Escenarios y sus diseños 48
5.2. Resultados RSSI en dBm para escenario 1 65
5.3. Resultados SNR en dBm para escenario 1 66
5.4. Resultados RSSI promedio, SNR promedio y paquetes recibidos para escenario 1 68
5.5. Resultados filtrados de RSSI promedio, SNR promedio y paquetes recibidos para escenario 1 68
5.6. Resultados RSSI en dBm para escenario 2 69
5.7. Resultados SNR en dBm para escenario 2 69
5.8. Resultados RSSI promedio, SNR promedio y paquetes recibidos para escenario 2 70
5.9. Resultados RSSI en dBm para escenario 3 71
5.10. Resultados SNR en dBm para escenario 3 72
5.11. Resultados RSSI en dBm para escenario 4 73
5.12. Resultados SNR en dBm para escenario 4 74

6.1. Escenarios y sus diseños 84
6.2. Resultados de paquetes recibidos según distancia para escenario 1 89
6.3. Perdida de paquete por nodo 94

Índice de Figuras

1.1. Topología General del Sistema 3
1.2. Visualización de los centros de carga cercanos 6

2.1. Participación de vehículos eléctricos en el mercado automotriz [8] 10
2.2. Ventas anuales de vehículos eléctricos en Chile [9] 10
2.3. Metas y avances en electromovilidad en Chile [7] 11
2.4. Electrolineras públicas en Chile [10]. 12

3.1. Posibles mensajes que puede enviar un EV a un CS [11]. 15
3.2. Posibles mensajes que puede responder un CS a un EV [11]. 15
3.3. Node-Red Flow que lee el mensaje, lo desencripta y luego gráfica [12]. . 16
3.4. V2X en [13]. 16
3.5. VXG en [14]. 17
3.6. Arquitectura descrita en [15]. 18
3.7. Topología lograda para servicio de seguimiento de transporte publico [16]. 19
3.8. Modelo de negocio presentado por el equipo de WeCharge 3.8 21
3.9. Estados de transición del agente [18] 22
3.10. Diagrama estructural de navegación de cuatro dimensiones [19] 23
3.11. Diagrama de flujo para la navegación de carga óptima [19] 24
3.12. Arquitectura del sistema (Framework) [20] 25
3.13. Modelos de colas para EVs [20] 26
3.14. Proceso de navegación en el sistema integrado [21] 26
3.15. Mecanismo de oportunidad de reservación [21] 27
3.16. Mapa desarrollado en Framework multiplataforma Flutter [22]. 31
3.17. Diagrama de bloques del sistema de BRAZ [23] 31
3.18. Mapa con la ubicación de la persona en aplicación desarrollada [24] . . . 32
3.19. Arquitectura del sistema de seguimiento de autobuses [25] 33
3.20. Compasión de React Native y Flutter para escenarios simples. [26]. . . . 33

4.1. Tecnologías propuestas 36
4.2. Arquitectura final del sistema con sus tecnologías elegidas para IoEV . . . 45

5.1. Estacionamientos en campus central UTFSM 48
5.2. Escenario 1 49
5.3. Escenario 2 49
5.4. Escenario 3 50

5.5. Escenario 4 50
5.6. Puntos donde se hicieron la mediciones para escenarios 1 y 2 52
5.7. Puntos donde se hicieron la mediciones para escenarios 3 y 4 52
5.8. Raspberry Pi 4B con Dragino LoRa/GPS Hat 53
5.9. Dragino LG308-EC25 54
5.10. Nodo LoRa con GPS 55
5.11. Esquema de conexión 55
5.12. Archivo global_conf.json 56
5.13. *Gateway ID* 57
5.14. Registrando *gateway* en TTN 58
5.15. Frecuencias y SF usadas por TTN tanto de carga y descarga de mensajes . 59
5.16. Configuración de parámetros en *gateway* LG308 60
5.17. Configuración de servidores en *gateway* LG308 61
5.18. Incorporación de llaves secretas de protocolo ABP al *gateway* LG308 . . 61
5.19. Configuración cliente MQTT del *gateway* LG308 62
5.20. Agregación del canal MQTT 62
5.21. Flujo Node-Red para transmisión nodo a nodo usado en escenario 2 . . . 63
5.22. Flujo Node-Red para transmisión nodo a *gateway* usado en escenarios 3 y 4 63
5.23. Valores RSSI vs Distancia para las configuraciones usadas en el escenario 1 67
5.24. Valores SNR vs Distancia para las configuraciones usadas en el escenario 1 67
5.25. Valores RSSI vs Distancia para las configuraciones usadas en el escenario 2 70
5.26. Valores SNR vs Distancia para las configuraciones usadas en el escenario 2 70
5.27. Valores RSSI vs Distancia para las configuraciones usadas en el escenario 3 72
5.28. Valores SNR vs Distancia para las configuraciones usadas en el escenario 3 73
5.29. Valores RSSI vs Distancia para las configuraciones usadas en el escenario 4 74
5.30. Valores SNR vs Distancia para las configuraciones usadas en el escenario 4 75

6.1. Mapa satelital con *gateways* ubicados en estacionamientos en campo central UTFSM 76
6.2. 77
6.3. Mapa vial con *gateways* ubicados en estacionamientos en campo central UTFSM 78
6.4. Nodos transmisor y receptor en sector plano 79
6.5. Código SenScript para nodo receptor para primer objetivo 81
6.6. Código SenScript para nodo transmisor para primer objetivo 82
6.7. Código SenScript para nodo receptor para segundo objetivo 84
6.8. Código SenScript para nodo transmisor para segundo objetivo 85
6.9. Nodo estático 86
6.10. Ruta 1 seguida por nodo móvil N°1 86
6.11. Ruta 2 seguida por nodo móvil N°2 87
6.12. Ruta 3 seguida por nodo móvil N°3 88
6.13. Ruta 4 seguida por nodo móvil N°4 88
6.14. Mensajes recibidos por cada *gateway* para escenario 1 91
6.15. Consumo de energía del nodo estático por una hora para escenario 1 . . . 91

6.16. Nivel de energía almacenada en nodo estático a lo largo de una hora escenario 1 92
6.17. Mensajes recibidos por cada *gateway* para escenario 2 93
6.18. Mensajes enviados y mensajes recibidos en total para el escenario 2 . . . 93
6.19. Mensajes recibidos por cada *gateway* por nodo para escenario 2 94
6.20. Mensajes enviados y mensajes recibidos en total por cada nodo para el escenario 3 95

7.1. Análisis de niveles de radio 97
7.2. Casos de prueba de algoritmo simulados en la interfaz de usuario 98

1 | Introducción

Hoy en día la crisis climática impacta gravemente en las vidas de todos los seres vivos, lo que ha llevado a un aumento de desarrollos tecnológicos destinados a aportar al cuidado del medio ambiente y/o resolver problemas sociales que generan un impacto favorable al mismo.

Actualmente existe un gran interés en los países por ser parte del proceso de adaptación hacia un mundo más ecológico y sustentable. Chile no se queda atrás, y junto a ello surgen problemas para satisfacer las necesidades de las personas que se adhieren a estos cambios. Uno de estos cambios es el reemplazo de los combustibles fósiles como fuente energética en automóviles por una alternativa más sustentable, como es la energía eléctrica, dando como resultado el desarrollo de automóviles eléctricos e híbridos disponibles para el público general.

Los vehículos eléctricos son un gran avance tecnológico, y han traído consigo una gran cantidad de desafíos. Uno de estos desafíos consiste en encontrar la ruta y la estación de carga óptima para evitar la sobrecarga de la red eléctrica del sector y consigo garantizar el bienestar de los clientes considerando sus prioridades. Este trae consigo varias problemáticas a resolver, como por ejemplo:

- Agotamiento de energía.
- El desconocimiento del usuario de los centros de carga que tiene a su alrededor y/o de sus características

- Además de la distribución de los vehículos eléctricos para evitar la sobrecarga de la red eléctrica

En este informe se abordarán la problemáticas que surgen de este desafío, buscando satisfacer tanto las necesidades de los actuales clientes, como de los futuros.

1.1. ¿Como es actualmente?

Actualmente existen plataformas para vehículos eléctricos que solo proveen la información parcial de las electrolineras en el país. Estas plataformas no proveen servicios completos e información detallada de lo disponible en cada estación de carga. Con ello tampoco proveen un servicio de recomendación, y aunque algunas cuentan con comunidades virtuales donde es posible compartir información de las electrolineras, no logran satisfacer las necesidades de los usuarios.

1.2. ¿Cuál es nuestro desafío?

El desafío del presente trabajo es diseñar una plataforma capaz de mostrarle a los usuarios de vehículos eléctricos, las electrolineras que se encuentran a su alrededor, recomendar la que se ajuste a sus necesidades y generar la ruta hacia ese destino, contribuyendo de esta manera a su bienestar, por medio de la satisfacción de una de sus necesidades fundamentales.

1.2.1. Análisis del desafío y alternativas de problemas específicos por resolver

El desafío apunta a diseñar la creación de una comunicación inalámbrica entre vehículos y estaciones de carga. El criterio utilizado para seleccionar los centros de carga (Charging

Stations: CSs) dependerá de las prioridades y necesidades de cada usuario en relación a las características de cada CS. Los componentes principales del sistema son: los vehículos eléctricos (Electric Vehicles: EVs), puntos de carga para vehículos eléctricos (Electric Vehicle Charging Station: EVCS), centro de sistema inteligente de transporte (Inteligent Transport System: ITS) y centro de control de navegación (Navegation Control Center: NCC), tal como se puede ver en la figura 1.1.

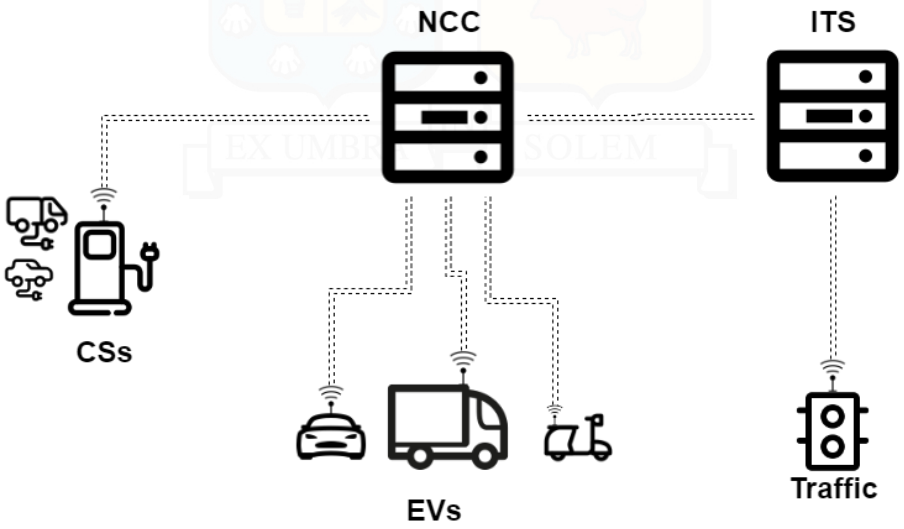


Figura 1.1: Topología General del Sistema

Cuando un usuario necesite cargar su vehículo solicitará la búsqueda de los CSs que más se acomoden sus demandas. Esta información será transmitida al NCC, el cual comparará las características de los EVCS para obtener la ruta hacia el primer candidato con ayuda del ITS. Dentro de las características más importantes en consideración se encuentran: los tipos de conectores en los CS, valor de la energía por kWh, la distancia entre el vehículo y el CS, el tiempo de espera en la cola, la potencia de los cargadores en kW (que determinarán la velocidad de carga) y la cantidad de energía en la batería del EV.

El uso de APIs para obtener la información de tráfico y generar las rutas siguiendo las direcciones y sentidos permitidos en las calles, ayudará a obtener resultados más precisos para el usuario. Dentro de las opciones para efectuar esta tarea se encuentra WazeRouteCal-

culator [2] y Tomtom Search API [3].

También se necesita construir un servidor web que permita resolver solicitudes de los usuarios (API) y que ejecute el algoritmo de optimización que compare y evalué los centros de carga adaptándose a las necesidades del usuario. Esto permitirá entregar la información solicitada de forma rápida y en un formato sencillo.

Debido a la naturaleza del problema, donde tenemos una población de soluciones no aleatorias y se debe seleccionar una de ellas o las que mejor se adapten a las necesidades del usuario, se decide implementar una heurística [4], una función objetivo que permita minimizar los costos para el usuario. Esto es posible usando la programación lineal [5], es decir, identificando las restricciones del problema, las constantes y variables dentro del mismo. Las electrolineras o centros de carga que estén alrededor del usuario serán evaluadas una por una hasta dar con las que obtengan el menor costo entre las opciones. Las prioridades del usuario podrán ser modificables añadiendo una variable que acompañe a los atributos normalizados de cada electrolinera.

En relación al almacenamiento de la información se utilizará una base de datos SQL como PostgreSQL o MySQL. Esto nos permitirá acceder a cada CS/electrolinera de forma individual, obtener la información actualizada manteniendo un esquema estático y aprovechar de mejor manera la estructura de los datos obtenidos [6].

1.3. Acercamiento a la solución

Como se mencionó en la sección anterior, para definir una solución primero se debe identificar el problema. Este consiste en la búsqueda constante que los usuarios de vehículos eléctricos realizan para encontrar centros de carga durante sus viajes. A esto se le suma que no todos los EV tienen las mismas necesidades, hablamos tanto del almacenamiento de energía en kWh (baterías), como del tipo de conectores, entre otros. Estas restricciones significan una complicación constante para los usuarios y provoca que las demás personas no quieran adquirir un vehículo eléctrico debido a todos estos inconvenientes. Sin mencionar que a medida que se implementan nuevas tecnologías a lo largo de las ciudades, nacen con ellas complicaciones que no afectan solo al usuario final, sino que también a los proveedores de energía eléctrica como, por ejemplo, la sobrecarga de las líneas eléctricas al no distribuir los vehículos entre los puntos de carga disponibles. Sin embargo, el desarrollo de NaVE trae una solución a estas problemáticas, pues propone desarrollar una plataforma que disponga la información fundamental al usuario para realizar la búsqueda de las electrolineras, con el fin de disminuir su nivel de estrés y distribuir los vehículos entre los centros de carga dentro de las ciudades y comunas.

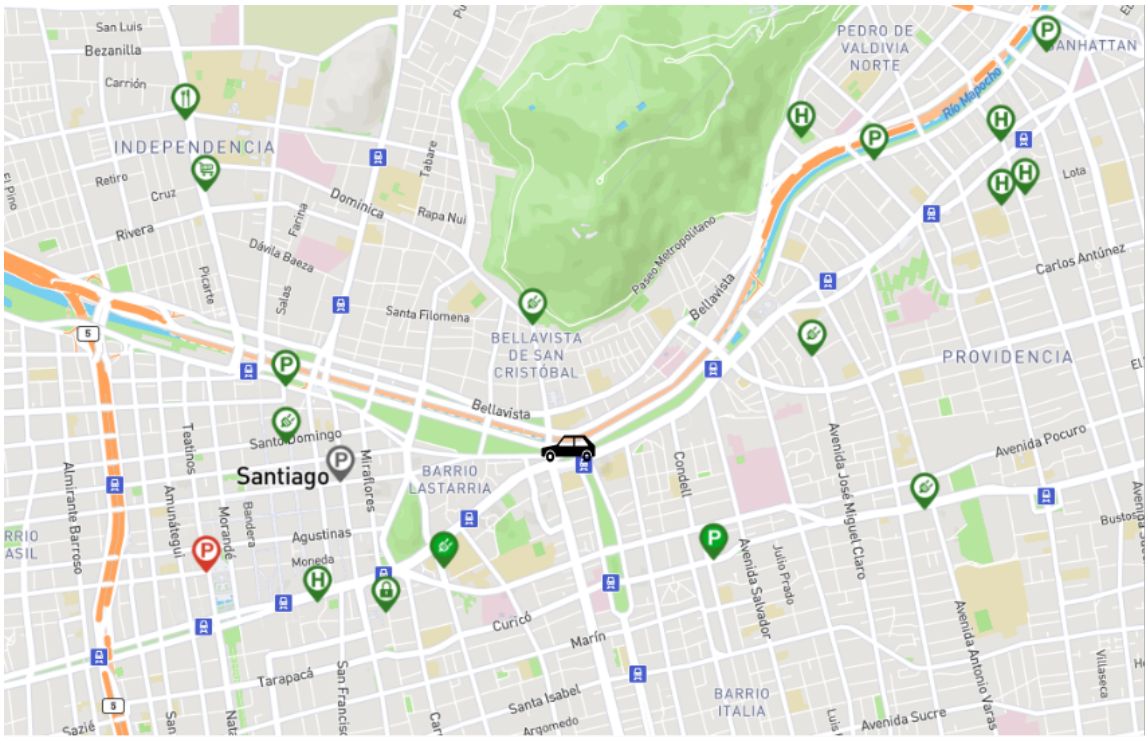


Figura 1.2: Visualización de los centros de carga cercanos

En la figura 1.2 apreciamos un bosquejo de la aplicación mostrando la ubicación del vehículo y los centros de carga que le rodean.

1.4. Objetivo General

Desarrollar una plataforma de navegación para vehículos eléctricos que incluya un sistema de recomendación de centros de carga.

1.5. Objetivos Específicos

Definir la generación de data para los componentes principales del sistema eléctrico (EV, EVCS)	:	Definir cómo generar datos de EV, buscar donde conseguir los EVCS.
Definir una topología de red y la relación entre los componentes	:	Definir como será el comportamiento y la comunicación entre los componentes el sistema.
Desarrollo de comunicación inalámbrica entre EVs y EVCS	:	Comunicar a través de una tecnología inalámbrica a los vehículos eléctricos y los centros de carga.

1.6. Estructura del documento

Este documento se estructura de la siguiente forma:

- En el Capítulo 2 se estudian los conceptos referentes a plataformas de navegación, automóviles eléctricos y electrolineras en Chile, como así también, el servicio de recomendación ofrecido en la plataforma.
- En el Capítulo 3 se realiza una revisión de los trabajos previos referentes al uso de hardware y tecnología de comunicación y frameworks de visualización para Android e iOS.
- En el Capítulo 4 se estructura el modelo de capas del desafío, explicando sus componentes y la función que cumplen dentro del sistema. Además, se señala la arquitectura de componentes propuesta de la solución, en base a lo previamente estudiado en el Capítulo 3.

- En el Capitulo 5 se describe la implementación de hardware y los resultados de lo que respecta a la capa de comunicación.
- En el Capitulo 6 se describe el desarrollo de la simulación usando CupCarbon y los resultados de lo que también respecta a la capa de comunicación. En el capitulo anterior y este se elabora todo lo relacionado con la tecnología inalámbrica.
- Finalmente, en Capítulo 7 se genera un análisis del trabajo realizado junto a un plan de trabajos futuros para mejorar el funcionamiento de lo implementado en la capa de comunicación.

2 | Marco Teórico

2.1. ¿Qué es una plataforma de navegación?

Una plataforma de navegación es una aplicación web y/o móvil que permite a las personas explorar los productos y/o servicios que ofrece una empresa o organización sobre un tema o problemática en específico.

Estos productos y/o servicios pueden ser de toda clase de rubro, y dirigidos a cualquier grupo social. Simplemente deben apuntar a resolver un problema o necesidad de la comunidad involucrada, de esta manera llama la atención de las personas que eventualmente se convertirán en clientes.

En este caso la plataforma de navegación, está destinada a un público que tiene como medio de transporte un vehículo eléctrico.

Los problemas y soluciones propuestas serán expuestas a lo largo de este documento

2.2. Automóviles eléctricos en Chile

En Chile, cada vez hay más vehículos eléctricos en circulación, es decir, que cada año se comercializa más vehículos con esta tecnología. En las Figuras 2.1 y 2.2 se puede visibilizar la variación en la compra de vehículos eléctricos en Chile durante los últimos años.

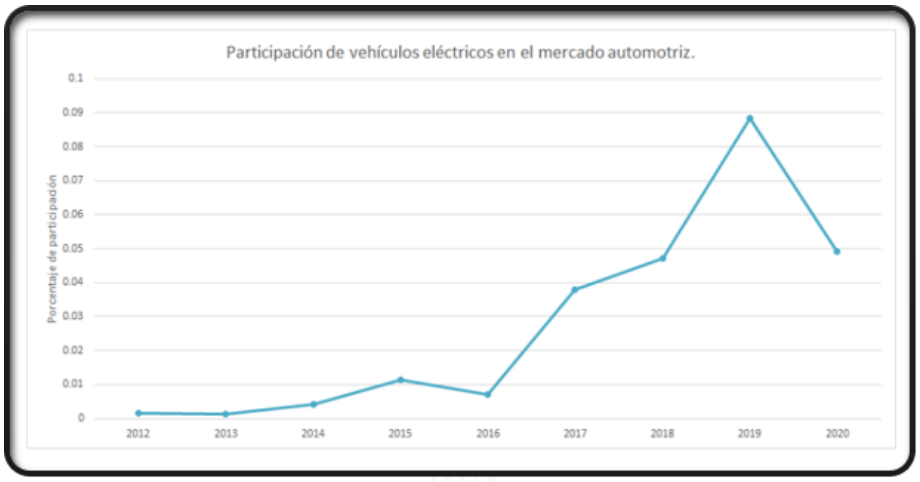


Figura 2.1: Participación de vehículos eléctricos en el mercado automotriz [8]

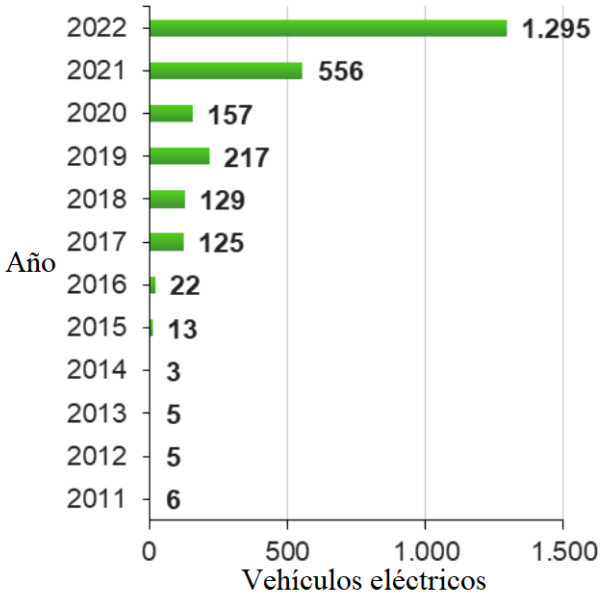


Figura 2.2: Ventas anuales de vehículos eléctricos en Chile [9]

2.3. Electrolineras en Chile

La expansión hacia la electromovilidad es tan deseada que el gobierno chileno trabaja actualmente por medio del ministerio de energía para fomentar su uso, instalando nuevos centros de carga a lo largo del país (Figura 2.3).



Figura 2.3: Metas y avances en electromovilidad en Chile [7]

Junto a esto podemos observar en la Figura 2.4 la distribución por regiones de las estaciones de carga, estando la mayoría de estas en la región Metropolitana. Esto último es un problema a la hora de viajar a lo largo del país con este tipo de vehículo, ya que aumenta el nivel de ansiedad experimenta en e los usuarios debido a la inseguridad de encontrar un centro de carga antes de que la batería del vehículo se agote. Este problema será detalladamente analizado durante este informe.

Regiones	N° Electrolineras
Metropolitana	526
Valparaíso	69
Coquimbo	44
Biobío	38
La Araucanía	35
Libertador Bernardo OHiggins	32
Maule	30
Los Lagos	28
Ñuble	14
Antofagasta	14
Tarapacá	8
Arica y Parinacota	8
Magallanes y de la Antartica Chilena	6
Atacama	6
De los Ríos	6
Araucanía	4
O´Higgins	4
Aysén	2
Aisén del General Carlos Ibáñez del Campo	2
Los Ríos	2
Total	**878**

Figura 2.4: Electrolineras públicas en Chile [10].

2.4. Servicio de recomendación

Un servicio de recomendación permite ofrecer alternativas al usuario y destacar las que se adecuen mayormente a sus necesidades e intereses.

En el caso de este trabajo de memoria, se enfocará en comunicar los vehículos con el centro de carga. En cuanto el servicio de recomendación, el cual no estará implementado en este informe, se encargará de entregar al usuario centros de carga que mejor se adapten a sus intereses y mostrarlos en alguna interfaz de usuario. Estos intereses estarán reflejados en el peso que previamente el usuario asignó a cada característica de las electrolineras a través de la plataforma. Estas características son:

- Costo de la energía.
- Velocidad de carga.
- Tiempo de espera.

- Distancia.

El usuario podrá manipular el peso de cada característica a través de un perfil personalizado y enviar eventualmente esta información hacia el servidor.

3 | Estado del arte

En la actualidad existen distintas plataformas que facilitan al usuario de automóviles eléctricos la búsqueda de centros de carga. Sin embargo, en Chile, aún no existe nada parecido. Su ausencia provoca un estrés adicional hacia los usuarios de este país, lo que se traduce como un desincentivo para comprar un vehículo eléctrico.

El trabajo se divide en tres secciones principales: comunicación, middleware e interfaz de usuario. Cada una de estas secciones requirió investigación para validar las decisiones y tecnologías usadas. El trabajo realizado a lo largo de este proyecto fue desarrollado por Mario Araya, quien se encargó del Middleware. Y Rudolf Hartmann, quien se encargó de la comunicación y la interfaz de usuario. En este informe se analizará en detalle la sección de comunicación desarrollada por Rudolf Hartmann, sin embargo, de todas formas se cubrirán las diversas investigaciones que se realizaron en cada sección.

3.1. Comunicación

Tras analizar “An Efficient Electric Vehicle Charging Architecture based on LoRa Communication” [11], se reconoce una forma de evitar la aglomeración de automóviles en un punto de carga, y la sobrecarga de la red eléctrica. Esto se sería posible al usar LoRa como tecnología de comunicación inalámbrica entre el CS y el EV, donde el primero (el cual es alimentado por un red de paneles solares) le informa al vehículo si tiene la carga necesaria para reservar un cupo, de lo contrario, le notifica que vuelva mas tarde. Esta lógica de comunicación se puede utilizar para calcular e informar al usuario del EV sobre la cola de espera de cada electrolinera.

Esta comunicación entre estas entidades se logra bajo una cierta codificación y lógica en los bits enviados. Como se puede ver en el la Figura 3.1, esta seria los posibles mensajes que el centro de carga podría recibir de un vehículo eléctrico. Por otro lado, podemos ver la Figura 3.2, la cual muestra la variedad de mensajes que puede responder el centro de carga al vehículo, según la disponibilidad e información que maneja.

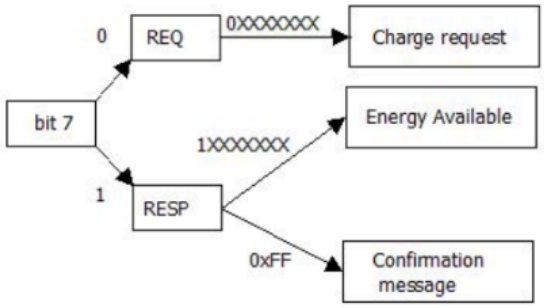


Figura 3.1: Posibles mensajes que puede enviar un EV a un CS [11].

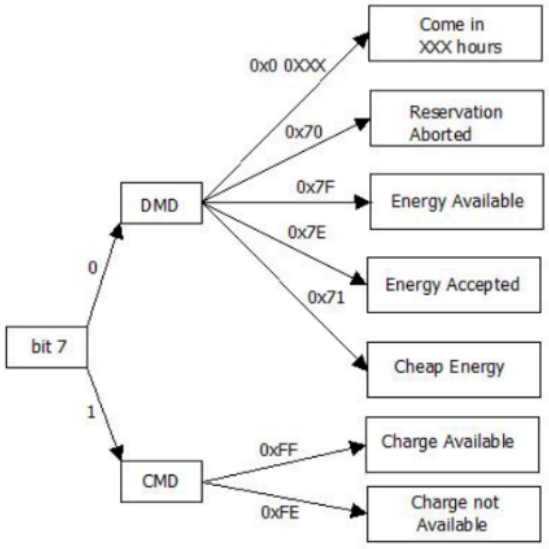


Figura 3.2: Posibles mensajes que puede responder un CS a un EV [11].

En “LoRa-based communication system for data transfer in microgrids” [12] se estudia la comunicación entre módulos LoRa y una puerta de enlace. Esta última utiliza la programación de bloques de código en Node-Red para desencriptar el mensaje enviado a través de LoRa para visualizar los voltajes medidos por cada nodo (Fig. 3.3). Esta arquitectura de programación y flujo de datos es ideal para lo que se quiere lograr en nuestro proyecto,

como la toma de datos a ciertas distancias y la lectura de la información codificada.

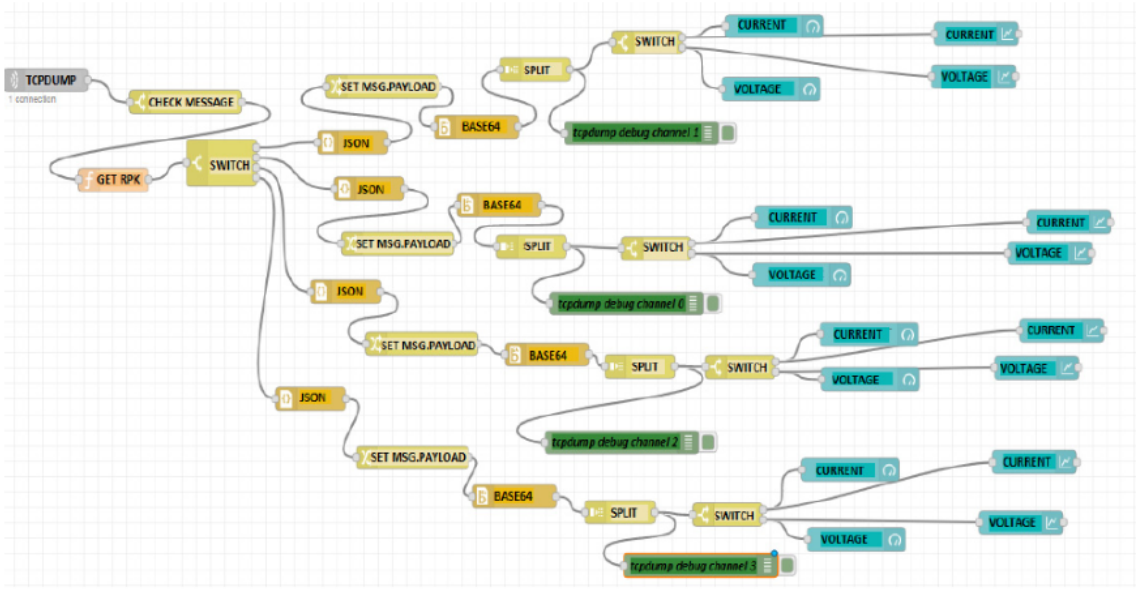


Figura 3.3: Node-Red Flow que lee el mensaje, lo desencripta y luego gráfica [12].

El sistema propuesto en "LoRa Architecture for V2X Communication" [13] adopta un enfoque de comunicación LoRa D2D para reducir la latencia al ofrecer comunicación directa de vehicle-to-vehicle (V2V) y de vehicle-to-infrastructure (V2I), en lugar de enrutar los datos a través del servidor LoRaWAN (Fig.3.4). Las pruebas y análisis verifican que el sistema propuesto puede comunicarse de manera confiable con infraestructuras y otros vehículos a velocidades entre 15 a 50 [km/h].

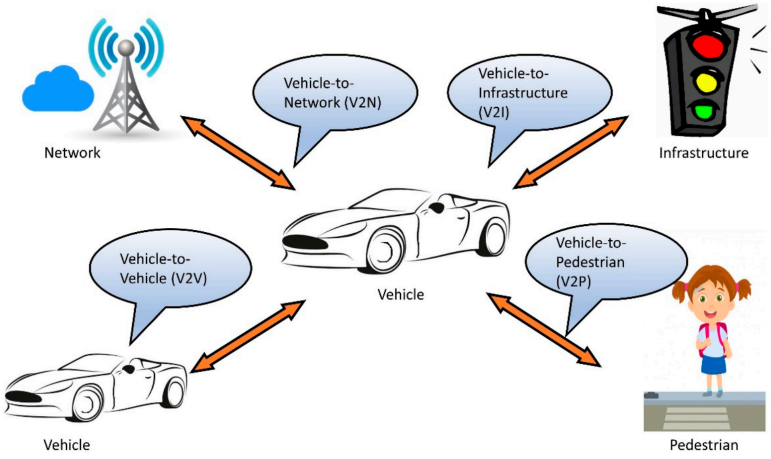


Figura 3.4: V2X en [13].

El articulo “Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration” [14] presenta una revisión exhaustiva y una evaluación del avance de los vehículos eléctricos (EV) y su interacción con el *Smart Grid*. Tanto en la estrategia para integrar los EV a la red eléctrica, tecnologías inteligentes de carga de los EV con la perspectiva de V2G (Vehicle-to-Grid) y con todo esto también la influencia de las fuentes de energía renovable en este tema.

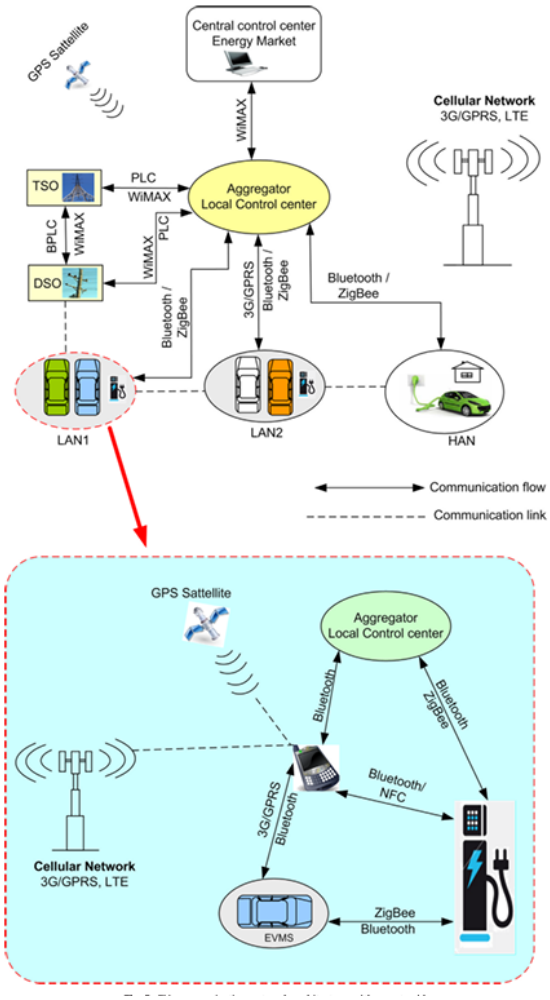


Figura 3.5: VXG en [14].

En el artículo “On the Mobile Communication Requirements” [15] se discuten los requisitos de comunicación para el gestión de la demanda de los EV en entornos urbanos, centrándose en la comunicación móvil entre estos y los *Smart Grids*. Donde se propone el uso de una solución LPWAN (Low-Power Wide-Area Network) como una posible

alternativa a las soluciones. Concluyendo que los resultados muestran que la tecnología LPWAN es capaz de manejar una cantidad adecuada de información para el escenario propuesto con la arquitectura descrita en la Figura 3.6.

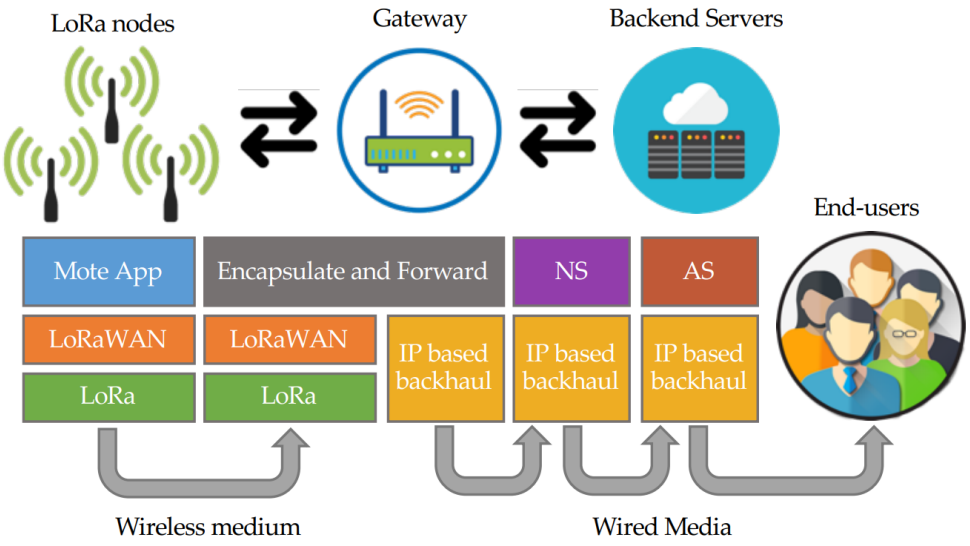


Figura 3.6: Arquitectura descrita en [15].

Los autores de "Proof of Concept of an IoT-Based Public Vehicle Tracking System, Using LoRa (Long Range) and Intelligent Transportation System (ITS) Services" [16] desarrollaron un servicio de seguimiento basado en IoT para el transporte público utilizando LoRa y una arquitectura ITS. Usando nodos de prueba, experimentan con la comunicación entre nodos y envían información GPS al servidor. Finalmente, recomiendan los parámetros de configuración de red óptimos en función de sus experimentos.

La topología utilizada en este paper se acerca a la deseada en la plataforma, por lo que resulta un buen ejemplo a seguir. Como se puede ver en la Figura 3.7.

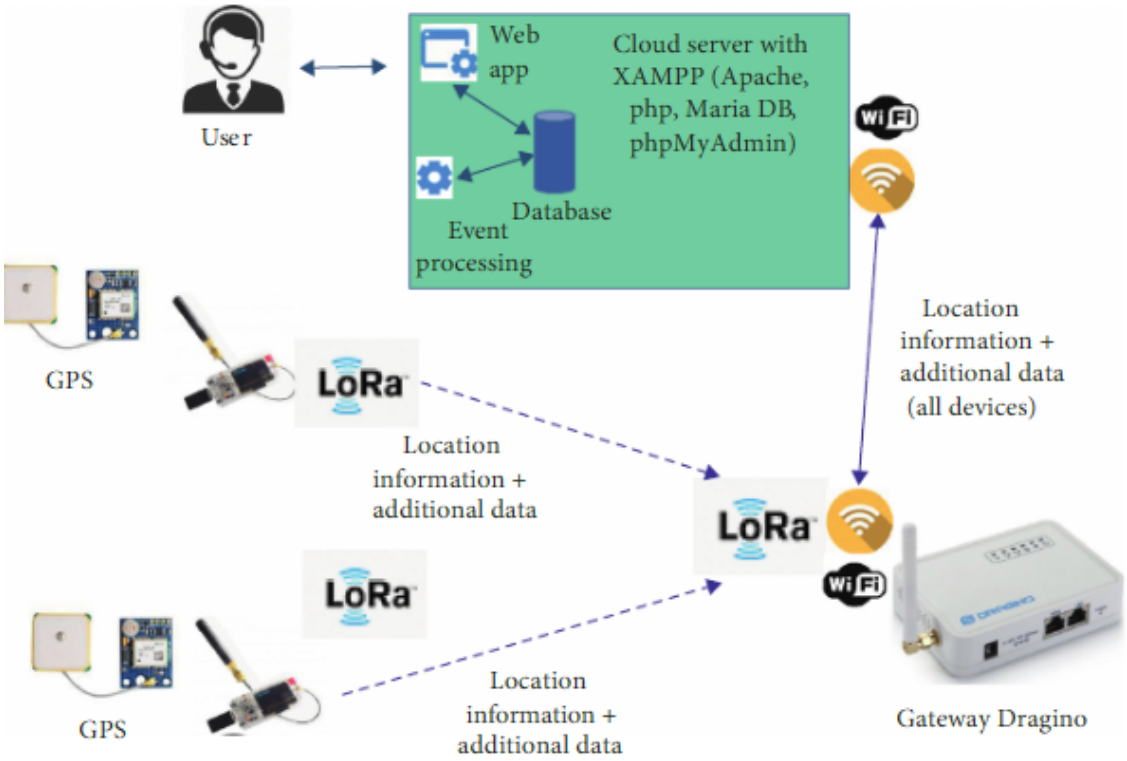


Figura 3.7: Topología lograda para servicio de seguimiento de transporte publico [16].

Tabla 3.1: Trabajos previos capa percepción

Paper	Tipo	Sensores	Red	App	Contribución
[11]	Técnico	No	Si	No	Presenta un diseño para el manejo de energía dentro de la comunicación entre EVs y CSs alimentadas por energía solar.
[12]	Técnico	Si	Si	Si	Presenta una comunicación inalámbrica basada en LoRa para transferencia de datos en microgrids.
[13]	Técnico	No	Si	No	Presenta tecnologías de comunicación como LoRa que juegan un papel importante en desarrollar una arquitectura de comunicación segura y confiable.
[14]	Estudio	No	Si	No	Analiza y evalúa la interacción de EVs con Smart-Grid, retratando el futuro del modelo de la energía eléctrica.
[15]	Estudio	No	Si	No	Analiza las necesidades de comunicaciones para la gestión de la demanda de EVs en el entorno urbano. Centrándose en la comunicación móvil entre los EVs y Smart-Grid.
[16]	Técnico	Si	Si	Si	Se propone un sistema de seguimiento de vehículos públicos basado en IoT, utilizando LoRa(Long Range) y servicios ITS, para la mejora del transporte público.

3.2. Middleware

WeCharge es un modelo de negocios desarrollado en Bélgica el 2022 que presenta una solución a la problemática [17]. Primero identifican el desconocimiento del usuario sobre la infraestructura de las estaciones de carga, es decir, si hay conectores disponibles, tipo de conector, tipo de carga, localización, tarifas de carga y penalizaciones. La penalización se efectúa cuando el usuario no retira su vehículo una vez cargado. Su objetivo es disminuir el tiempo sin utilización de los centros de carga, tanto públicos como privados, y redistribuir su uso. El modelo incluye servicios como reservas en los puestos de carga, monitoreo de la batería, tarifas a tiempo real, demanda a tiempo real y participación como centro de carga. Luego diseñan un algoritmo de optimización que separa las restricciones "ligeras"de las más importantes o "pesadas", esto permite establecer opciones de exclusión y prioridad en la búsqueda, lo que mejora la experiencia del usuario. De este trabajo se logró rescatar el modelo de negocio, la forma con la que la plataforma interactúa con los usuarios y como estos deciden a través de ella los valores de los parámetros del algoritmo. En la figura 3.8 se puede ver el modelo de negocio propuesto por el equipo de WeCharge, el mismo que inspiró el de este trabajo.

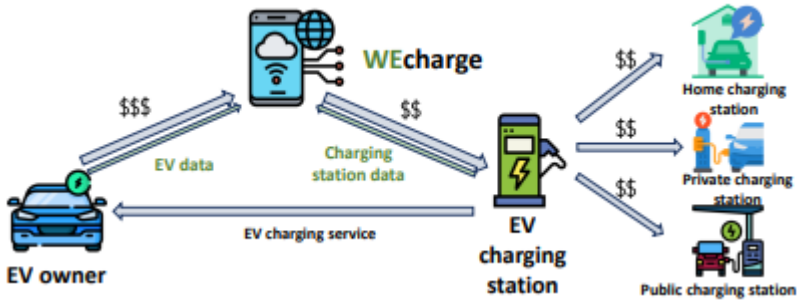


Figura 3.8: Modelo de negocio presentado por el equipo de WeCharge 3.8

Otro punto de vista para [18], resolver la problemática de elegir una estación de carga óptima para el usuario según el costo total. Se muestran los distintos factores que pueden influenciar en la selección de estación de carga. Alguno de ellos son: consumo de energía al manejar hacia la estación de carga, la energía demandada, el tiempo extra de carga debido

a la sobrecarga de la red (aglomeración de vehículos en un mismo CS) y el consumo de energía usada para viajar al destino final luego de cargar el vehículo. Todo estos valores son representados con su respectiva función de costo. Esto permite el cálculo de varios métodos de comparación y de esta manera minimizar el costo del usuario en su plan de movilidad. Este trabajo inspiró la idea de un agente, el cual transiciona por diferentes estados hasta llegar a su destino (como se ve en la figura 3.9).

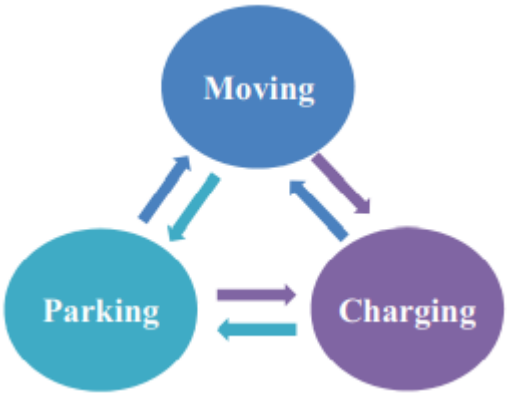


Figura 3.9: Estados de transición del agente [18]

La simulación, como prueba de concepto, es una herramienta muy útil que permite explorar y corregir los comportamientos del software a lo largo de diferentes casos de uso y contextos, por esta razón también fue incluida dentro del proyecto.

En el trabajo "Optimal Charging Navigation Strategy for Electric Vehicles" [19] existe un enfoque en la optimización en la navegación para carga del vehículo, considerando cuatro aspectos: vehículos eléctricos, red eléctrica, trafico y centros de carga. Se utiliza MATLAB para simular entornos donde 5000 vehículos conducen y cargan simultáneamente con el objetivo de obtener los valores que permiten navegar y cargar el vehículo óptimamente. Para esto, los aspectos mencionados anteriormente de desglosan en parámetros que vienen a calcular la eficiencia óptima de la red eléctrica, la seguridad óptima del tráfico y los gastos óptimos, tal como se ve en la figura 3.10.

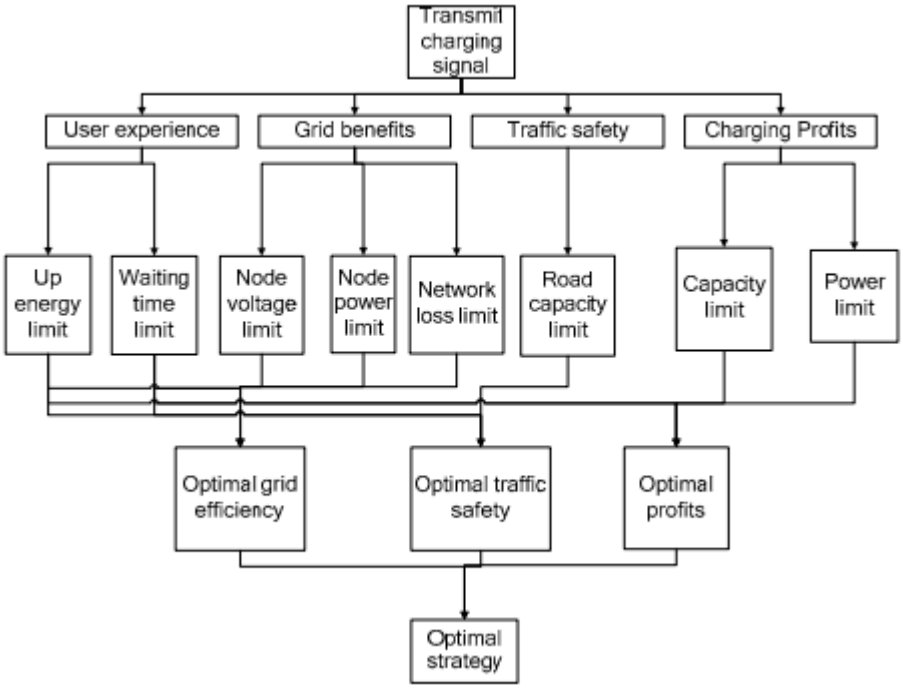


Figura 3.10: Diagrama estructural de navegación de cuatro dimensiones [19]

Teniendo la estructura del modelo de optimización proceden a crear el diagrama de flujo que permitirá a los agentes, que representan a los EVs de la simulación, moverse y buscar centros de carga. Ese diagrama es representado en la figura 3.11

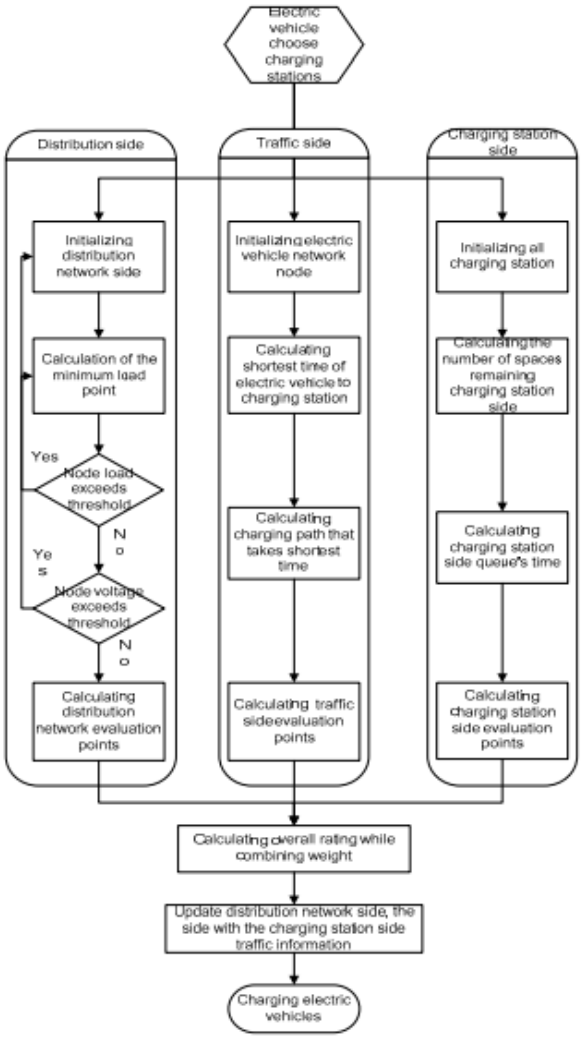


Figura 3.11: Diagrama de flujo para la navegación de carga óptima [19]

Luego, integrando lo estructurado proceden a ejecutar la simulación utilizando individualmente cada sección de optimización por separado, y finalmente un caso con todas las secciones unidas, obteniendo los mejores resultados. Durante el trabajo de título los parámetros del modelo no fueron subdivididos en grupos de optimización, pero si se trabajó en un diagrama de flujo que permitiera describir el algoritmo y el comportamiento de los agentes que interactúan en la simulación.

El siguiente trabajo “Real-Time Charging Navigation of Electric Vehicles to Fast Charging Stations: A Hierarchical Game Approach” [20] tiene como objetivo considerar

el sistema de energía y de transporte con el fin de proponer un enfoque que sea efectivo para asegurar la disponibilidad de la red eléctrica y minimizar los costos de carga del vehículo. Esto se aborda utilizando un modelo de juego jerárquico, donde en el primer nivel se presenta un juego no competitivo que modela el comportamiento entre EVCSs. Por otro lado, en los niveles más bajos, el resultado del juego no competitivo es utilizado para alimentar juegos evolutivos, y de esta forma encontrar una solución rápida y eficaz al problema.

En la figura 3.12 podemos ver la arquitectura del sistema y la forma en la que interactúan los componentes. Dentro de esta arquitectura se ven comprometidas cuatro partes, las cuales son: Power system operation center (PSOC), EVCSs, Electric vehicle navigation system (EVNS) y EVs.

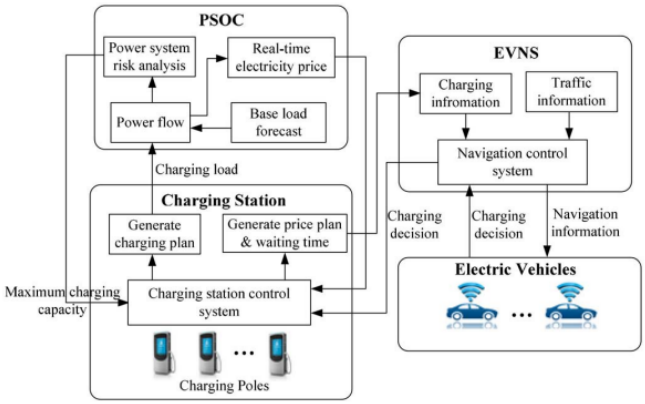


Figura 3.12: Arquitectura del sistema (Framework) [20]

Los resultados de este proyecto a pesar de ser certeros, se enfoca en solucionar el problema de distribución en específico. Por esta razón, toma un enfoque totalmente distinto, pero aún así permite analizar desde otra perspectiva las diferentes formas con las que se puede abordar una problemática que involucra vehículos eléctricos y centros de carga. En el trabajo de título también se busca balancear la carga de la línea eléctrica y distribuir los vehículos entre los EVCSs, por lo que dentro de todo se puede rescatar la idea de un modelo de cola (ver figura 3.13) para los EVs en cada EVCSs.

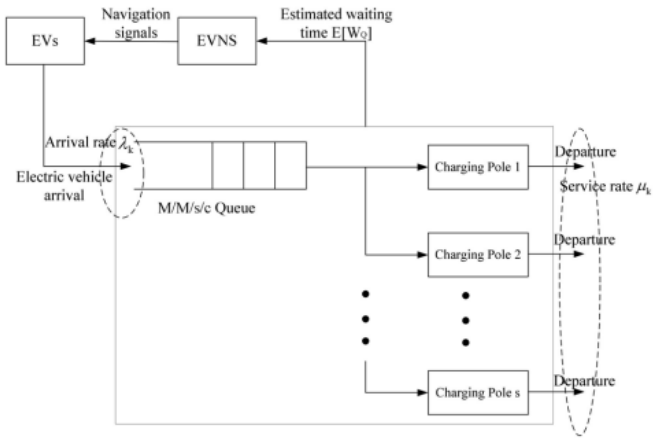


Figura 3.13: Modelos de colas para EVs [20]

En el trabajo de investigación "Price Incentive-Based Charging Navigation Strategy for Electric Vehicles" [21] se propone una estrategia basada en el costo de carga. Esta estrategia considera la influencia espacio temporal de los EVs, donde un modelo de colas junto a un mecanismo de reservación permite la minimización de estos costos. El proceso de navegación dentro del sistema se puede ver la en la figura 3.14, donde interactúan cuatro entidades entre sí, estas son: EVs, Information exchange center (IEC), ITS y CSs.

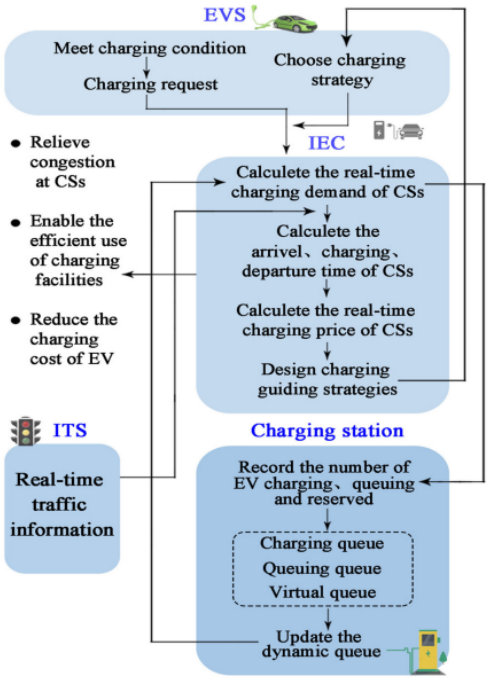


Figura 3.14: Proceso de navegación en el sistema integrado [21]

El mecanismo de oportunidad de reservación (ver figura 3.15) permite calcular los costos de carga en base al precio y el tiempo de espera. Este mecanismo entrega un enfoque bastante interesante para abordar el problema de los tiempos de espera en carga de vehículos eléctricos, que también se debe abordar en el trabajo de título. Por lo tanto, el modelo será útil para desarrollar una estrategia propia, y así abordar el problema de recomendación de centros de carga para EVs.

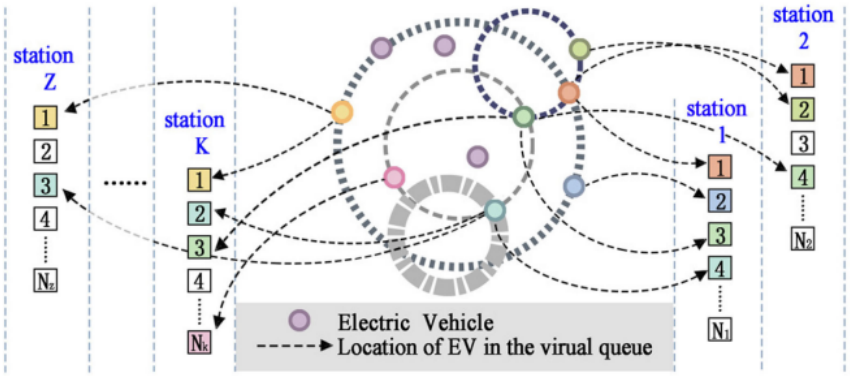


Figura 3.15: Mecanismo de oportunidad de reservación [21]

Tabla 3.2: Trabajos previos capa middleware

Paper	Enfoque	Variables y parámetros	Contribución
[17]	Plataforma de navegación hacia centros de carga y propuesta de modelo de negocio.	▪ Tipo de cargador del EV. ▪ Batería restante. ▪ Disponibilidad de los cargadores. ▪ Distancia hacia el CS. ▪ Duración de la carga. ▪ Tiempo de espera. ▪ Costo de la recarga.	El estudio ofrece un sistema con interfaz de usuario donde este pueda seleccionar la prioridad de las variables y de esta forma obtener resultados personalizados.
[18]	Estrategia de navegación hacia centros de carga.	▪ Consumo de energía hasta el CS. ▪ La energía demandada por el EV. ▪ El tiempo de espera para acceder al CS. ▪ Tiempo de espera extra por las bajas de potencia del CS. ▪ La energía requerida para llegar al destino del EV.	El estudio compara el rendimiento de métodos simples con el método que considera todas las variables anteriormente mencionadas obteniendo resultados favorables.

Tabla 3.3: Trabajos previos capa middleware

Paper	Enfoque	Variables y parámetros	Contribución
[19]	Estrategia de navegación hacia centros de carga.	▪ Tráfico (Velocidad del viaje). ▪ Centro de carga con menos desviación de voltaje (desbalance de la red). ▪ Perdida de conexión en la red (Saturación). ▪ Tiempo de espera.	El estudio utiliza una estrategia considerando 4 variables, la cual entrega resultados favorables para encontrar un CS con carga segura.
[20]	Estrategia de navegación hacia centros de carga.	▪ Densidad de tráfico. ▪ Velocidad de tráfico. ▪ Flujo de tráfico. ▪ Estado de carga (SoC) del EV. ▪ Precio de carga. ▪ Distancia del EV al CS. ▪ Tiempo de espera. ▪ Número de cargadores en la CS. ▪ Lista de espera para usar cargador de la CS.	El estudio modela el tráfico y la cola de los cargadores, además de definir las estrategias de los centros de carga y los vehículos eléctricos. El fin es lograr una estrategia óptima para cargar el vehículo teniendo en cuenta el sistema eléctrico y de transporte. También se utiliza un agente para simular y ejecutar los algoritmos usando teoría de juegos.

Tabla 3.4: Trabajos previos capa middleware

Paper	Enfoque	Variables y parámetros	Contribución
[21]	Estrategia de navegación hacia centros de carga.	▪ Costo de la energía en el CS. ▪ Tiempo de carga. ▪ Congestión del CS. ▪ Lista dinámica de reservaciones en CSs.	Estudio simula un entorno donde múltiples vehículos compiten para cargar su batería. Estos vehículos evalúan y reservan un puesto en la electrolinera que prometa un menor costo de precio y tiempo para el usuario. También añaden un Route manager o ITS para calcular las rutas a destino.

3.3. Interfaz de usuario

Finalmente el usuario tiene que ser capaz de ver y usar estos resultados, esto nos conlleva a la ultima sección. Esta se trata de la muestra de los datos, una plataforma de visualización, la cual interactuará con el usuario y sus elecciones, mostrando los datos analizados y calculados en la sección anterior.

Tras leer y analizar el estudio del paper “Technologies and implementation of an integration with geolocation and maps in mobile application development” [22], se puede ver que solo una tecnología de las estudiadas es multiplataforma, esta es Flutter. En la Figura 3.16 se puede ver una implementación usando este framework.

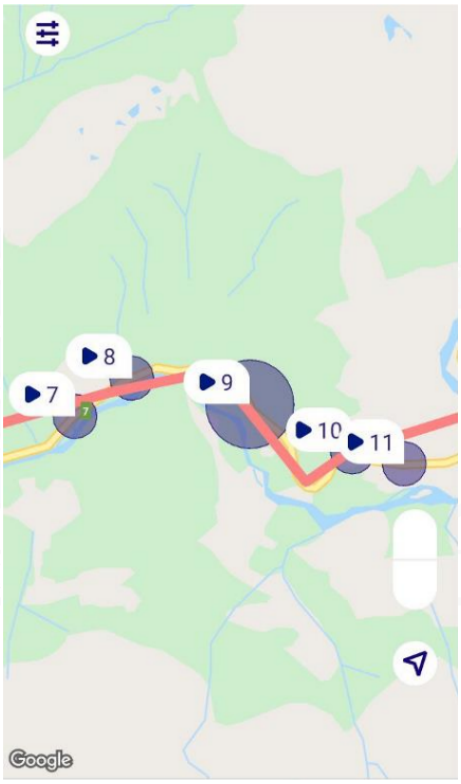


Figura 3.16: Mapa desarrollado en Framework multiplataforma Flutter [22].

El texto “Development of an Android App for Monitoring PMS in Gas Stations” [23] presenta una aplicación Android llamada BRAZ, diseñada para monitorear el suministro de combustible en estaciones de servicio. La investigación surgió de la necesidad de ayudar a los consumidores a encontrar la estación de servicio más cercana y asequible para comprar Gasolina Premium (PMS). Esta App ofrece una lista en tiempo real de estaciones con PMS, incluyendo precios por litro, densidad de vehículos y ubicación GPS para navegación. Esto beneficia especialmente a los visitantes desconocidos con la red vial local.

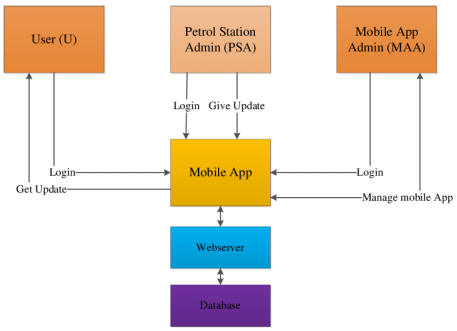


Figura 3.17: Diagrama de bloques del sistema de BRAZ [23]

En “Development of Android Based Real Location Tracking App” [24] se muestra como se desarrolló una aplicación para dispositivos Android que permite rastrear la ubicación de una persona en tiempo real (3.19) y proporcionar informes periódicos. Además, cuenta con una base de datos donde los usuarios pueden almacenar su información personal y compartirla con otros que se encuentren en el mismo lugar. Esta aplicación facilita el registro de nuevos usuarios y ofrece un inicio de sesión sencillo en cualquier momento, utilizando una base de datos en tiempo real de Firebase.



Figura 3.18: Mapa con la ubicación de la persona en aplicación desarrollada [24]

El paper “Design and development of android mobile based bus tracking system” [25] habla de un sistema de seguimiento de autobuses móvil que ofrece una solución que permite a las personas obtener la ubicación del autobús sin molestar a los pasajeros. Requiere que los pasajeros y coordinadores del autobús tengan un teléfono móvil Android con conexión a internet. Este utiliza GPS y GSM para enviar la ubicación del autobús a los servidores, mostrándola en un mapa y calculando el tiempo de llegada y distancia a la parada de espera.

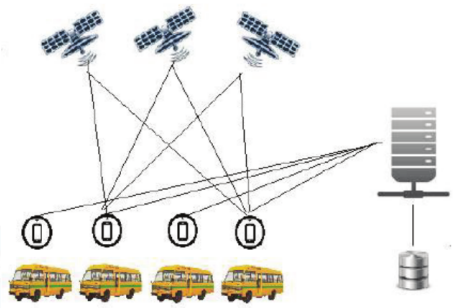


Figura 3.19: Arquitectura del sistema de seguimiento de autobuses [25]

En “React Native vs Flutter, cross-platform mobile application frameworks” [26] se compara, analiza y estudia en profundidad dos frameworks cross-plataform: Flutter y React Native, ambos creados y desarrollados por grandes entidades del Internet: Google y Facebook respectivamente. Aquí se analiza los distintos enfoques que tienen ambas tecnologías para desarrollar aplicaciones móviles, como por ejemplo Props y States de React Native, y los Widget de Flutter. También se compara el rendimiento de cada Framework para varios escenarios. Desgraciadamente no incluyen ningún escenario con un mapa de navegación. Cómo se puede ver en la Figura 3.20, React Native tiene una mayor velocidad de actualización para escenarios simples.

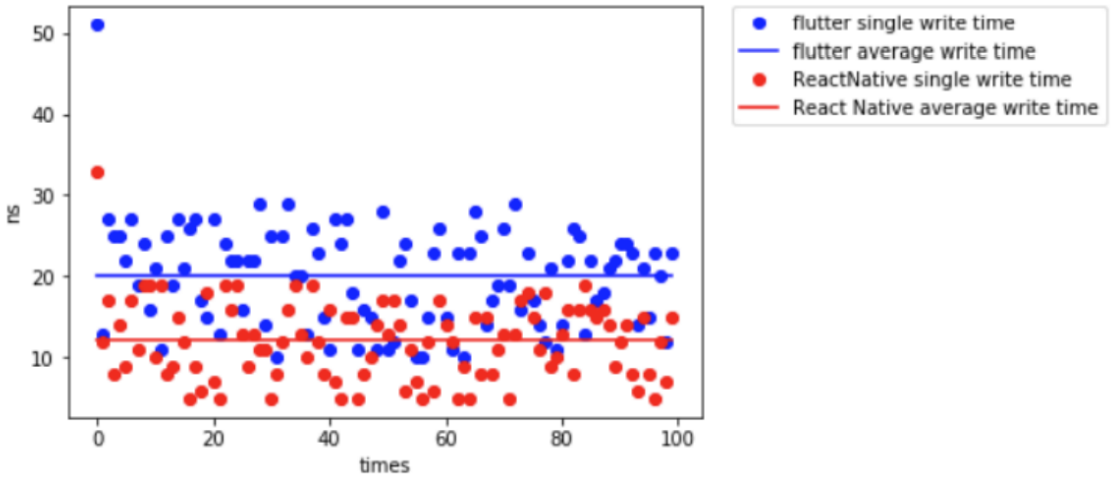


Figura 3.20: Compasión de React Native y Flutter para escenarios simples. [26].

Por otro lado, Flutter tiene un tiempo de reacción mas lento, pero sigue siendo estable

y totalmente compatible con tecnologías de Google, lo cual nos facilitaría el uso de APIs como la de Google-Maps.

Tabla 3.5: Trabajos previos capa aplicación

Paper	Tipo	GPS	Plataforma	Tecnología	Contribución
[22]	Estudio	Si	iOS/Android	▪ Swift ▪ Objective-C ▪ Kotlin ▪ Flutter ▪ React Native	Analiza y describe las tecnologías nativas y multiplataforma utilizadas para implementar la funcionalidad de geolocalización y mapas geográficos en el desarrollo de aplicaciones móviles para Plataformas iOS y Android.
[23]	Técnico	Si	Android	Java	Presenta una aplicación para Android que monitorea el estado de la gasolina en gasolineras.
[24]	Técnico	Si	Android	Java	Presenta el desarrollo de una App Android basada en el seguimiento en tiempo real de la ubicación actual de una persona.

Tabla 3.6: Trabajos previos capa aplicación

Paper	Tipo	GPS	Plataforma	Tecnología	Contribución
[25]	Técnico	Si	Android	-	Presenta una aplicación con un sistema de seguimiento de buses.
[26]	Estudio	No	iOS/Android	▪ Flutter ▪ React Native	Analiza y estudia algunas de las características más importantes de React Native y Flutter. Esta investigación experimenta e investiga las diferencias entre esas características e intenta comprender la razón que hay detrás de ellos.

Tras analizar los trabajos descritos e identificar los principales problemas y restricciones de las soluciones propuestas, se procederá a definir, proponer e implementar una solución al desafío abordado en este trabajo de título.

4 | Arquitectura de la Plataforma

Antes de decidir la arquitectura, se han propuesto varias tecnologías para utilizar en cada una de las capas. Estas tecnologías fueron categorizadas por capas, como se muestra en la figura 4.1.

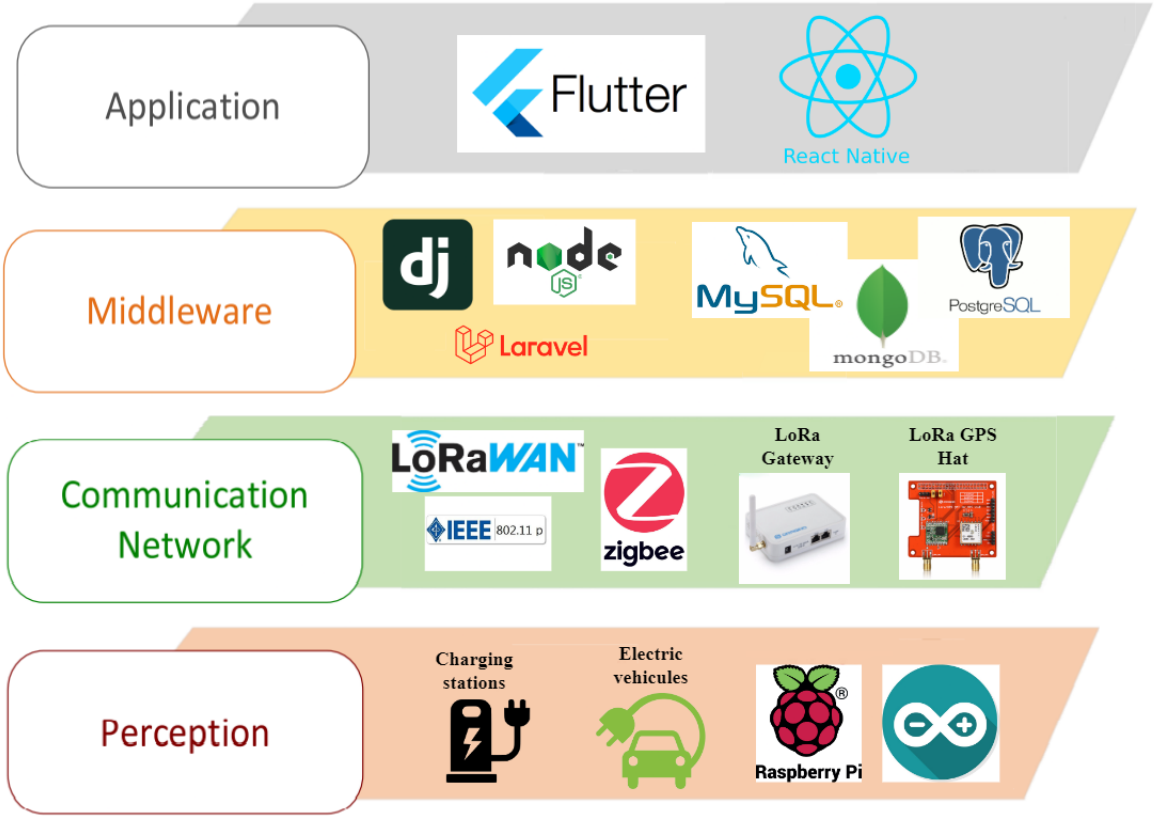


Figura 4.1: Tecnologías propuestas

4.1. Capa de Percepción

La capa más baja de la arquitectura representa todo el hardware periférico que interactúa con el sistema. Estas entidades corresponden al vehículo eléctrico y los centros de carga, siendo ambas muy importantes para la generación de datos y toma de decisiones.

Descripción de componentes

- **EVs (Electric Vehicles):** Vehículos que son impulsados completa o parcialmente por un motor eléctrico. Su batería tiene que ser cargada por una fuente externa.
- **CSs (Charging Stations)**: Estaciones donde se establecen conectores especiales para cargar baterías de vehículos eléctricos.
- **NCC (Navigation Control Center)**: Servidor o ambiente donde toda la información necesaria es recopilada, y donde se ejecutan todas las tareas y algoritmos correspondientes.
- **ITS (Intelligent Traffic Station center)**: Centro de información que provee el estado del tráfico, accidentes, rutas, etc.

4.2. Capa de Comunicación

Esta capa considera el hardware y el software que se encargan de transmitir e interpretar la información enviada de forma inalámbrica, como así también se especifican las tecnologías utilizadas para enviar y recibir información de los vehículos eléctricos y estaciones de carga. Diferentes tecnologías puedes ser utilizadas, tanto inalámbricas como alambicas.

Descripción de componentes

- **Heltec LoRa v2:** Microcontrolador el cual simulará la posición, movimiento y conexión de un vehículo eléctrico.

- **Raspberry Pi**: Computador de placa simple que cumplirá la función de puerta de enlace entre los vehículos eléctricos y el servidor.
- **Dragino LG308-EC25**: Gateway LoRaWan con capacidad de conexiones Lora, WiFi y 4G, cumplirá como puerta de enlace entre EV y servidores.

4.3. Capa Middleware

Middleware hace referencia al software que se encuentra en medio de la comunicación entre el usuario y los datos. En esta capa encontramos a la API, entidad que permitirá extraer los datos desde la base de datos hacia la interfaz de usuario por medio de consultas HTTP, la cual será implementada utilizando Django. Este último es un framework de python que permite hacer APIs robustas con bastante facilidad, además de ser compatible con las numerosas bibliotecas científicas del lenguaje de programación más versátil y popular del mundo. Junto a ella se encuentra el servidor de la base de datos, en este caso, PostgreSQL es una buena opción ya que no tiene una diferencia significativa para este problema en relación a las demás bases de datos relacionales (SQL), pero cuenta con una interfaz más intuitiva y limpia que permite una mejor experiencia de trabajo. Todo lo anterior será 'dockerizado', es decir, se montará en un contenedor Docker para asegurar el correcto funcionamiento del servidor considerando posibles cambios de maquina anfitriona. Finalmente el contenedor será levantado en un servidor en la nube de Amazon Web Service (AWS), quienes ofrecen maquinas gratuitas para realizar prototipos como este.

Descripción de Módulos

La capa de Middleware se separa en los siguientes módulos:

- **API:** Servidor HTTP que administra el flujo de información de la base de datos al usuario y viceversa. También administra la información consultada a APIs de terceros para satisfacer los requerimientos del usuario.

- **Base de datos:** Servidor donde se encuentra toda la información de los usuarios, centros de carga y cualquier otra entidad que participe dentro del sistema. En este caso se cuenta con una base de datos relacional, lo que quiere decir que es capaz de recibir consultas SQL ya que cuenta con una estructura basada en tablas que se relacionan entre sí a partir de claves foráneas.
- **Algoritmo de recomendación:** Luego de describir el problema de optimización utilizando programación lineal se procederá a implementar la función objetivo y los procesos que llevarán a cabo la recolección de información sobre la ubicación del vehículo y las electrolineras cercanas y dentro del rango.

4.4. Capa de Aplicación

La última capa de la arquitectura se relaciona con el software utilizado para crear la interfaz usuario, siendo este Flutter. Flutter es un framework escrito en el lenguaje de programación Dart, desarrollado por Google, que permite el desarrollo de aplicaciones móviles con gran facilidad y rapidez, lo cual se ajusta a lo requerido para este proyecto.

Descripción de Módulos

La capa de aplicación se divide en diferentes módulos, entre los cuales están:

- **Login/Logout:** Una forma de LogIn y LogOut con un ciertos perfiles creados por el usuario.
- **Mapa/Visualizador:** Vista principal de la aplicación que contará con un mapa centrado en la ubicación actual del vehículo vinculado, donde las electrolineras en el área serán mostradas junto al destino con su respectivo marcador, y la ruta hacia este.
- **Seleccionador de electrolinera:** El usuario debe tener la oportunidad de seleccionar una electrolinera de las cercanas y las que puede alcanzar, obteniendo como respuesta la información de esta.

4.5. Requisitos del Sistema

Antes de comenzar a describir las tecnologías que se utilizaron para la resolución del problema, es necesario conocer los requisitos descritos por el cliente para así obtener una plataforma que permita satisfacer las necesidades del usuario.

4.5.1. Requisitos Funcionales

Los requisitos funcionales definen el comportamiento del sistema, es decir, describen lo que este debe hacer. En otras palabras, interpreta lo que espera el usuario objetivo de la plataforma. Con esto en mente los requisitos funcionales definidos son los siguientes:

- **RF1** → El sistema ser capaz de conectarse a internet y a el/los servidor/es.
- **RF2** → El sistema debe ser capaz de entregar los resultados de interés, una lista de puntos carga cercanos en orden de preferencia según el usuario en menos de 3[s].
- **RF3** → Cada punto de carga de la lista debe tener un información compacta y comprensible para el usuario.
- **RF4** → La ruta hacia el destino elegido por el usuario debe ser calculada y mostrada en menos de 3[s].
- **RF5** → La navegación del sistema puede ser detenida por el usuario o automáticamente por el sistema cuando se llegue al destino.

4.5.2. Requisitos No Funcionales

Los requisitos no funcionales del sistema, definen atributos de calidad con un valor para el usuario. A continuación, se definieron los siguientes:

- **RNF1** → Interfaz accesible, sencilla e intuitiva para el usuario.

- **RNF2** → El rango de atraso de actualización de los datos de cada punto de carga mostrado debe ser menor a un día.
- **RNF3** → El sistema debe tener una disponibilidad del 99 %.
- **RNF4** → Se busca que el usuario use este sistema estando integrado al automóvil. Nos limitaremos a que los datos técnicos de este son proporcionados por el sistema mismo.
- **RNF5** → Se busca que el usuario use este sistema donde haya conectividad, con el fin de comunicarse con los servidores necesarios. De esta manera se puede establecer un ambiente de laboratorio donde se espera un funcionamiento ideal para el proyecto.
- **RNF6** → El sistema debe ser confiable en términos de seguridad.

4.5.3. Requisitos de Interfaces

Ya que ahora se conocen los requisitos funcionales y no funcionales del desafío, se procede a definir los requisitos de interfaces. Estos describen como espera el usuario que responda el sistema frente a diferentes eventos.

En la siguiente tabla se listan y describen los distintos eventos que el usuario y/o el sistema puede ejecutar.

Tabla 4.1: Eventos externos

Evento	Descripción	Iniciador	Parámetros	Respuesta
Consulta por necesidad de carga	Consulta por algún punto de carga por falta de batería en el vehículo	Usuario/ Sistema	Estado de carga de la batería; Preferencias del usuario para ordenar la lista; Posición usuario	Lista de puntos de carga en rango alcanzable
Elección de punto de carga	Usuario elige un punto de carga hacia donde quiere viajar	Usuario	ID punto de carga; ID usuario; Posición usuario;	Ruta hacia el punto de carga seleccionado
Cancelación o finalización de navegación	Se termina la navegación en vivo que esta en pantalla por elección del usuario o por llegada al destino	Usuario/ Sistema	ID viaje; ID usuario;	Pantalla de termino

La siguiente tabla muestra de manera detallada las respuestas del sistema frente a eventos externos.

Respuesta	Descripción	Parámetros
Lista de puntos de carga en rango alcanzable	Lista de puntos de carga a los que puede llegar el vehículo según el estado de carga de la batería	ID; Nombre; Dirección; Distancia; Precio; Tipos de enchufe; Tiempo de espera; Potencia carga.
Ruta hacia el punto de carga seleccionado	Se muestra la ruta hacia el punto de carga cercano y sus respectivas indicaciones para llegar a él.	ID viaje; Ruta en vivo
Pantalla de termino	Se cancela o se finaliza la navegación hacia el destino y se muestra algún resultado	Texto/Imagen

Tabla 4.2: Respuestas del sistema

4.5.4. Requisitos de Ambiente

Hardware de desarrollo

- Arduino y/o RasperryPi con sus módulos LoRa
- Dragino LG308-EC25
- Computador/Notebook
- Smartphone con Android

Software de desarrollo

- Python 3.8
- C
- JavaScript

- Microsoft Excel
- SQL/noSQL
- Dart

4.6. Perfiles de Usuario

Tabla 4.3: Perfiles de Usuario

Perfil	Socioeconómico y Cultural	Ocupación	Rango etario	Características fisiológicas/psicológicas
Conductor comercial	Trabajador independiente o en alguna empresa de transporte	Conductor Profesional/ Chofer	20 a 65	Busca eficientar su trabajo y tiempo, como así también evitar percances con su auto
Conductor común	Persona común y corriente que usa su auto para transportarse	Cualquiera	18 a 75	Busca eficientar su tiempo y dinero, como así también evitar percances con su auto

4.7. Propuesta de Solución

En este trabajo de título se pretende desarrollar una plataforma de navegación para vehículos eléctricos que no solo permita a los usuarios visualizar los distintos centros de carga a lo largo de país junto con sus características, sino que también ofrecerles el servicio de recomendación de centros de carga para facilitar sus viajes fuera de la capital y a lo largo de todo el país, evitando y previniendo los problemas de batería que son tan temidos dentro de la comunidad.

En relación a la capa de aplicación, se optó por el uso de Flutter como framework debido a la fácil integración con otras tecnologías de Google y la experiencia previa que

tenía el equipo utilizándolo.

En cuanto a la base de datos, se debía elegir una SQL debido a la estructura estática de los datos, sin embargo, MySQL y PostgreSQL no cuentan con diferencias significativas con respecto al uso que se le dará a lo largo de este proyecto. En consecuencia, se eligió PostgreSQL por la experiencia previa del equipo de desarrollo con este. Para el caso del framework del Back-End, se eligió Django, ya que al estar escrito en Python permite integrar todas las bibliotecas de Data-Science y de manejo de datos fácilmente, ayudando con la implementación de los algoritmos en el proyecto.

Finalmente en lo que la tecnología de comunicación inalámbrica se refiere, se eligió LoRa, debido a las ventajas que presenta en cuanto al gran alcance en distancia, una tasa de bits suficiente para enviar los datos que se necesitan entre el NCC y los EVs (como la ubicación GPS) bajo consumo energético y su gran masificación en el mundo de la tecnología IoT.

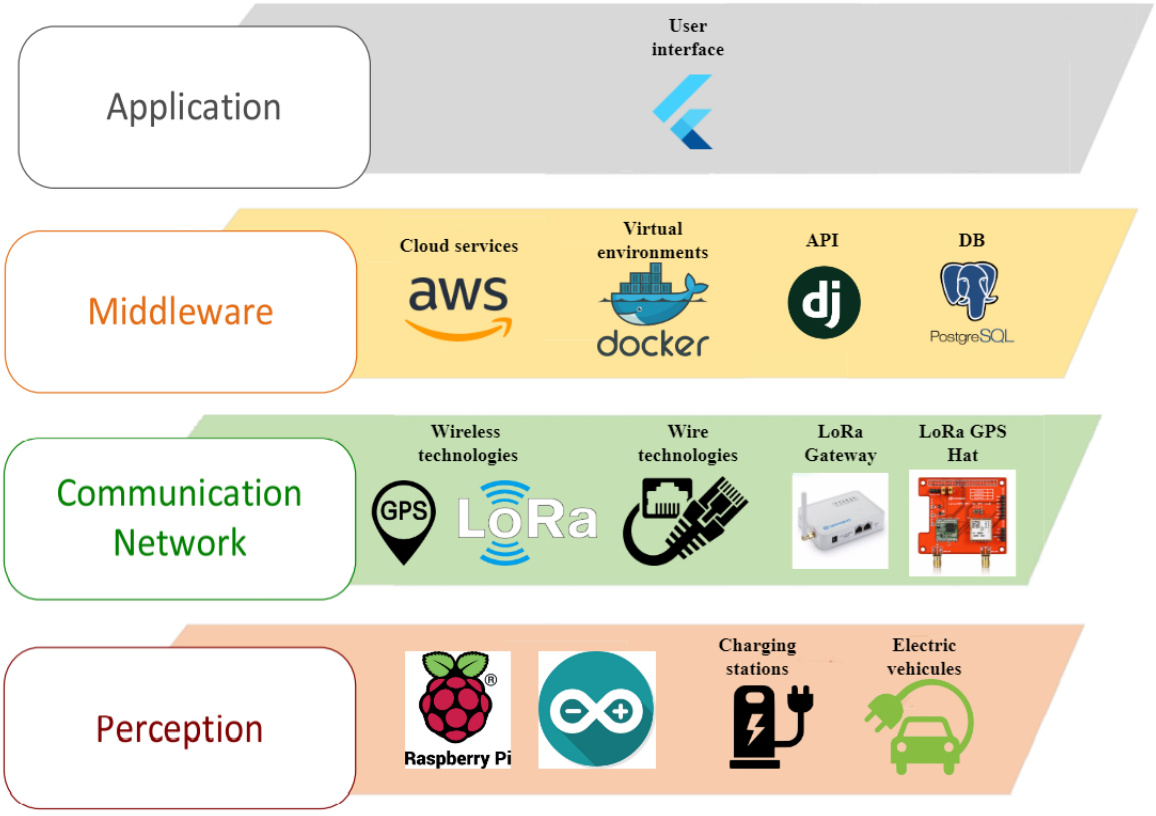


Figura 4.2: Arquitectura final del sistema con sus tecnologías elegidas para IoEV

A continuación se describirá la propuesta de solución del problema con más detalle.

- **Topología Centralizada:** Topología donde toda la información de los componentes principales llega a un servidor central (NCC). Este recopila, calcula y distribuye la información del sistema.
- **Aplicación móvil:** Una aplicación compatible con los sistemas operativos de los vehículos eléctricos o celulares de los usuarios que permita acceder a la información y servicios de la plataforma.
- **GPS:** Tecnología inalámbrica que comparte la información de posicionamiento de un individuo o cosa a lo largo y ancho del globo terráqueo.
- **LoRa:** Tecnología de comunicación inalámbrica de largo alcance, resistente al ruido y de bajo consumo energético, que permitirá comunicar los vehículos eléctricos con el servidor (NCC).
- **Servidor web y base de datos:** Servidor HTTP o API donde se alojará el algoritmo de recomendación, y que además se comunicará directamente con la base de datos para obtener la información requerida por el usuario.

Utilizando las tecnologías y sistemas mencionados anteriormente (ver figura 4.2), se pretende desarrollar la solución a la problemática que hoy en día afecta a miles de usuarios de vehículos eléctricos.

5 | Diseño de prototipo y resultados

Como se mencionó anteriormente, en este trabajo de memoria se trabajo e investigó sobre la movilización y carga de autos eléctricos, concentrándonos sobre la capa de comunicación y percepción. Por lo que esto lleva a la implementación y evaluación de esta misma. Por lo que este capítulo se concentrará en mostrar el desarrollo y resultados del prototipo creado para este trabajo.

Para evaluar la tecnología seleccionada en este trabajo. Se ideo el escenario futuro de prueba el cual será búsqueda de espacios libres en un estacionamiento. Específicamente el campus central de la Universidad Técnica Federico Santa María en Valparaíso, V Región, Chile. En la Figura 5.1 podemos ver que este tiene cinco estacionamientos marcados con morado y su respectivas numeraciones repartidos por el campus. Teniendo esta información, podremos crear y probar un modelo centrado en la capa de comunicación diseñado para este paisaje.

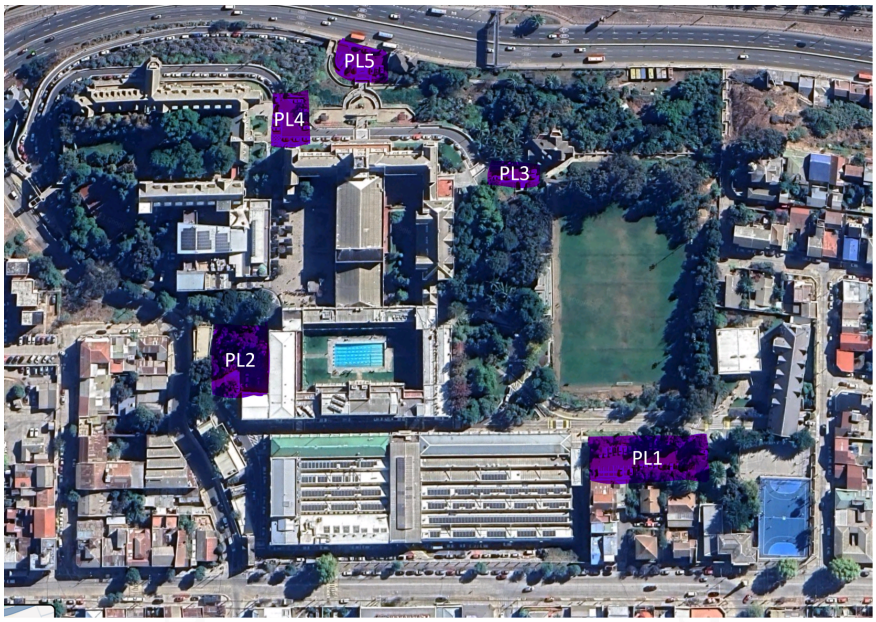


Figura 5.1: Estacionamientos en campus central UTFSM

5.1. Implementación

Para validar la propuesta y arquitectura presentada, se diseñaron e implementaron distintos escenarios a posteriormente ser evaluados. Acá distintos parámetros en función de calidad de señal fueron evaluados, estas son el *Received Signal Strength Indicator* (RSSI), *Signal to Noise Ratio* (SNR) y la perdida de paquetes. Los escenarios considerados son los siguientes:

Tabla 5.1: Escenarios y sus diseños

Escenario	Diseño
1	Comunicación nodo-nodo SIN almacenamiento de datos
2	Comunicación nodo-nodo CON almacenamiento de datos
3	Comunicación nodo-*gateway* con almacenamiento de datos y SIN módulo GPS
4	Comunicación nodo-*gateway* con almacenamiento de datos y CON módulo GPS



Figura 5.2: Escenario 1

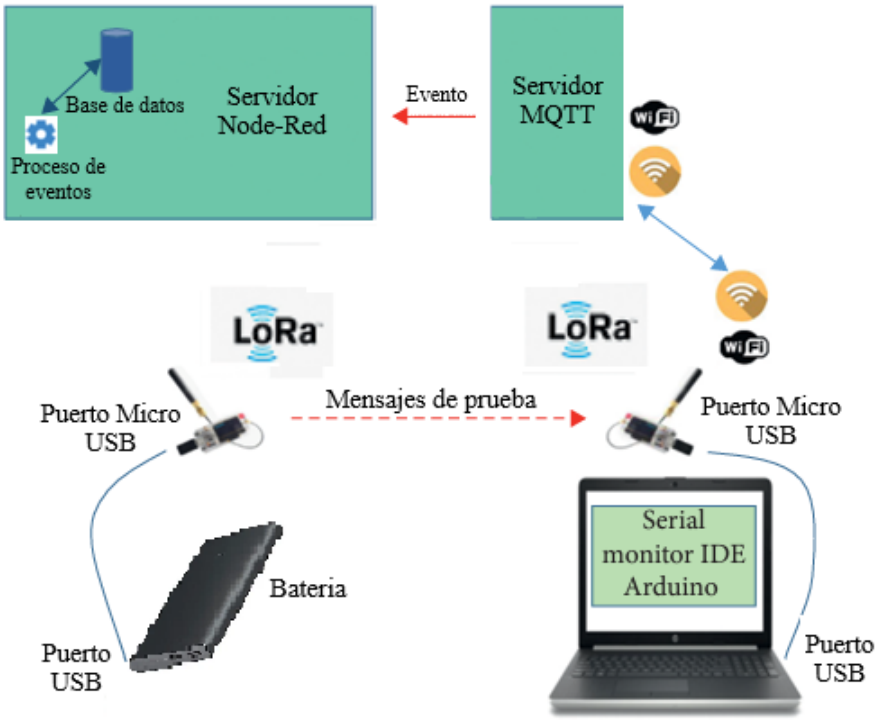


Figura 5.3: Escenario 2

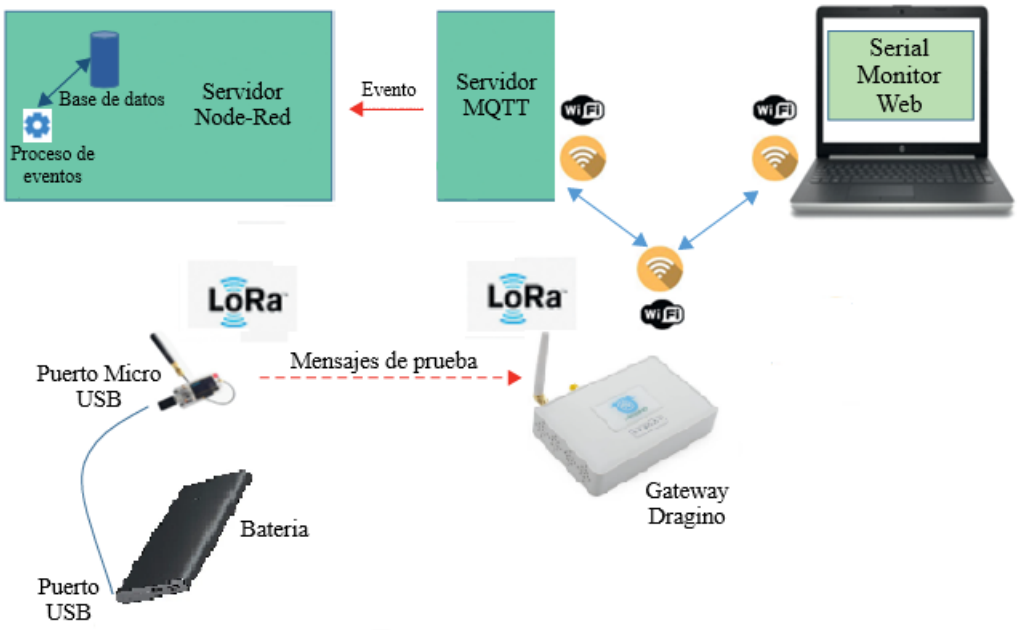


Figura 5.4: Escenario 3

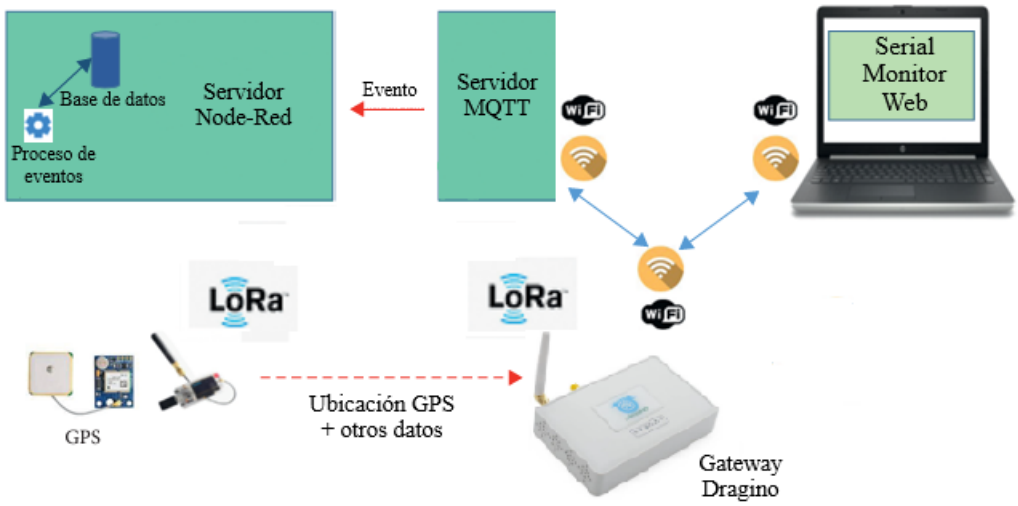


Figura 5.5: Escenario 4

Como se puede apreciar en la figura 5.2, la primera configuración corresponde a enviar y recibir mensajes de prueba en una conexión entre nodos unidireccional.

La segunda situación (Fig. 5.3) es similar que la primera pero con almacenamiento de los mensajes en una base de datos de texto plano. Esto se logró con un servidor MQTT y un servidor Node-Red los cuales serán aclarados más adelante.

El escenario 3 (Fig. 5.4) compete ya de una conexión nodo-*gateway*, y los datos siendo almacenados en una base de datos usando la misma arquitectura del escenario 2.

Y ya el último escenario que se muestra en la figura 5.5 incumbe lo mismo que su predecesor, difiriendo en que los mensajes que se envían corresponden a la ubicación del nodo, esto se consigue con un módulo GPS conectado emisor.

Las pruebas fueron tomadas cada 6 [m], lo que corresponde a la longitud a lo largo de una estacionamiento para autos comunes. Para estas pruebas se usaron varias configuraciones, alterando los parámetros *Spreading Factor* (SF) y *Band Width* (BW). Dando un total de 18 configuraciones distintas. Cabe mencionar que el *Coding Rate* (CR) fue dejado con un valor estático de 4/5, el cual es él más estable dentro de sus opciones según la conclusión llegada en el paper "Experimental Evaluation of LoRa in Transit Vehicle Tracking Service Based on Intelligent Transportation Systems and IoT" [30]. Y no es redundante mencionar que estas pruebas fueron ejecutadas en ambientes donde hubo vista clara entre emisor y receptor, de esta manera evitando, lo más posible, factores externos de interferencia.



Figura 5.6: Puntos donde se hicieron la mediciones para escenarios 1 y 2



Figura 5.7: Puntos donde se hicieron la mediciones para escenarios 3 y 4

Para realizar estas pruebas, se tiene que configurar tanto el hardware como el software

del *gateway* y los nodos LoRa. Lo que será explicado a continuación.

5.1.1. Hardware

Para esta capa se tuvo que conectar ciertos dispositivos con sus respectivos módulos tanto para la puerta de enlace como en los nodos ocupados.

5.1.1.1. Gateway LoRa

Para este caso se intentó usar una Raspberry Pi 4B conectada con un Dragino LoRa/GPS Hat (Fig. 5.8). Este dispositivo fue seleccionado por su disponibilidad en el laboratorio y sobre todo la versatilidad que tiene este dispositivo con tecnologías IoT, y con esto la cantidad de librerías disponibles para ser utilizadas. Desgraciadamente, el uso de este aparato se vio perjudicado por la configuración necesitada en los casos de prueba que se intentaron hacer, la posibilidad de crear un servidor LoRaWan y con ello, el poder usar un BW especifico, no fue posible en la Raspberry Pi.



Figura 5.8: Raspberry Pi 4B con Dragino LoRa/GPS Hat

Con esto en mente se eligió, por disponibilidad, versatilidad y facilidad de uso, el *gateway* LoRaWan Dragino LG308-EC25 (5.9). Con este fue posible cambiarle los pa-

rámetros anteriormente mencionados, como así también posibilidad de conexión con un servidor MQTT y con todo esto llevar a cabo los 18 casos de prueba con sus respectivas configuraciones.



Figura 5.9: Dragino LG308-EC25

5.1.1.2. Nodo LoRa

Para implementar los nodos de comunicación y ubicación se utilizó el microcontrolador Heltec LoRa V2 conectado con un módulo GPS (Fig. 5.10). Este fue seleccionado, ya que la disponibilidad de este abundaba en el laboratorio, y con ello se puede tener la posibilidad de poder de implementar varios y así probar conexión con más de un dispositivo a la vez. Además de esto, la disponibilidad en línea de recursos como librerías y ejemplos para este dispositivo es abundante y variada, dando un mayor valor al uso de este dispositivo como prototipo de implementación en este trabajo de memoria.

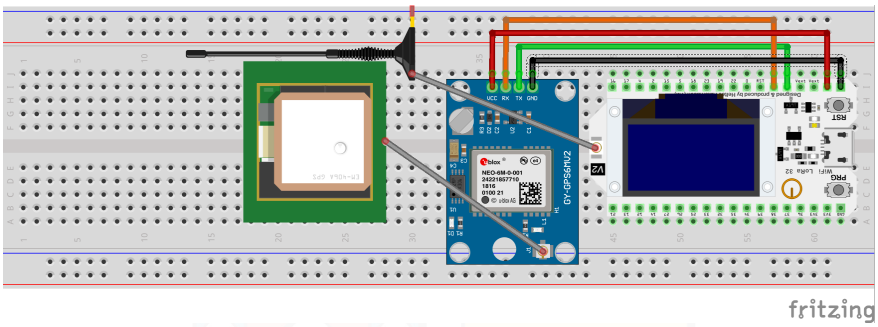


Figura 5.10: Nodo LoRa con GPS

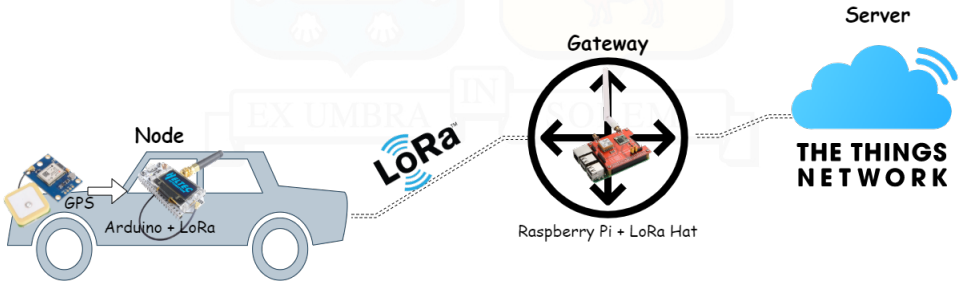


Figura 5.11: Esquema de conexión

5.1.2. Software

En esta sección se describe la implementación paso a paso del software configurado en los nodos, *gateway* y servidores que habrá en la red.

5.1.2.1. *Gateway*

a.- Raspberry Pi 4B

Como anteriormente se mencionó, como primera opción se usó como puerta de enlace una Raspberry Pi 4B con un Dragino Lora/GPS HAT. Para configurar este dispositivo se siguió el documento oficial de Dragino [27]:

- Después de conectar el HAT a la Raspberry, hubo que configurarla, empezando con la activación del módulo del kernel SPI. De esta manera se puede comunicar el computador con el módulo HAT.
- Después se instaló las distintas librerías con los siguientes comandos:

- `sudo apt-get install git`
- `sudo apt-get install wiringpi`
- `cd /home/pi`

  `git clone https://github.com/dragino/dual_chan_pkt_fwd`

  `make`

Ya instalada la librería de Dragino se configuró la frecuencia LoRa, SF y los pines a utilizar en el archivo *global_conf.json* como lo muestra la figura 5.12

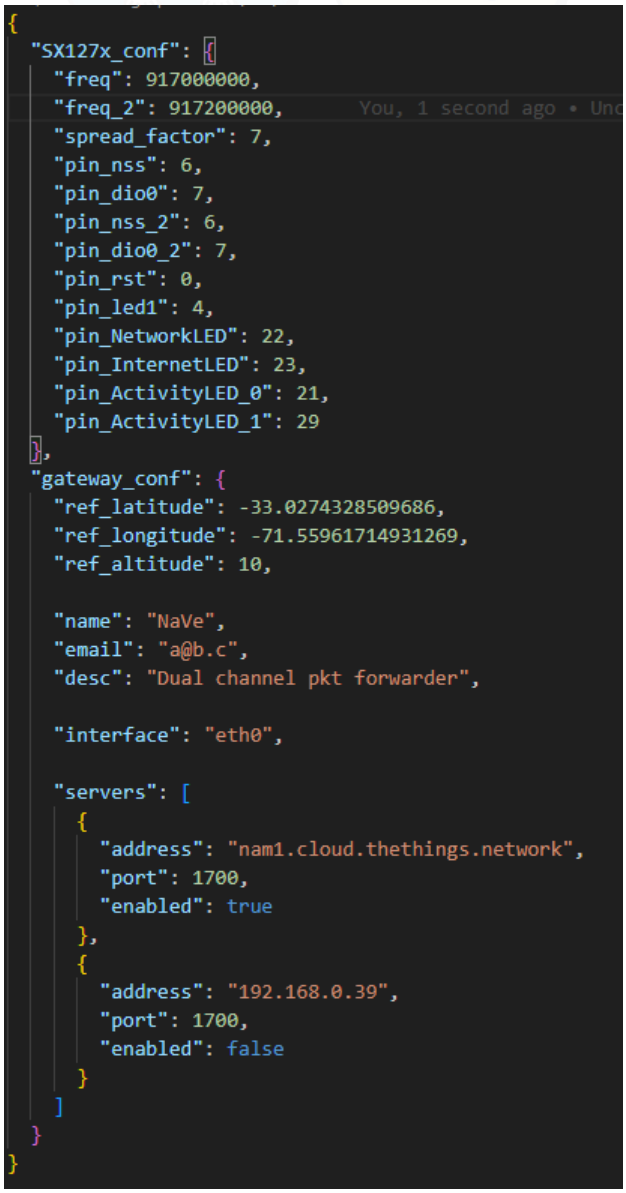


Figura 5.12: Archivo global_conf.json

Después se corrió el programa *dual_chan_pkt_fwd*, para así conseguir el *Gateway ID*, cual equivale a GateEUI en TTN.

```
pi@nave:~/dual_chan_pkt_fwd $ ./dual_chan_pkt_fwd
server: .address = nam1.cloud.thethings.network; .port = 1700; .enable = 1
server: .address = 127.0.0.1; .port = 1700; .enable = 1
Gateway Configuration
  your name (a@b.c)
  Dual channel pkt forwarder
  Latitude=0.00000000
  Longitude=0.00000000
  Altitude=10
  Interface: eth0
Trying to detect module CE0 with NSS=6 DIO0=7 Reset=3 Led1=unused
SX1276 detected on CE0, starting.
Trying to detect module CE1 with NSS=6 DIO0=7 Reset=3 Led1=unused
SX1276 detected on CE1, starting.
Gateway ID: e4:5f:01:ff:ff:37:54:60
Listening at SF9 on 917.100000 Mhz.
Listening at SF9 on 917.100000 Mhz.
```

Figura 5.13: *Gateway ID*

Teniendo configurado nuestro servicio LoRa, hubo que registrarlo en TTN. Al momento de querer registrar un *gateway*, este pide el GateEUI, el cual se consiguió anteriormente en la figura 5.13. Con esto se registró la Raspberry Pi como *gateway* en los servidores TTN. Después se solicitó el espectro de frecuencia y el *Spreading Factor*, acá se indicó que serán las frecuencias entre 915-928MHz, ya que las frecuencias que están cercanas a 433 MHz y entre a 902 MHz a 915 MHz [33] se usan en dispositivos de control remoto y vigilancia respectivamente en Chile. Se eligió FSB 2, como se puede ver en la figura 5.14, porque este es el que se usa en TTN y son las frecuencias que podemos ver en la figura 5.15 que envuelven la ya configurada en la Raspberry Pi.

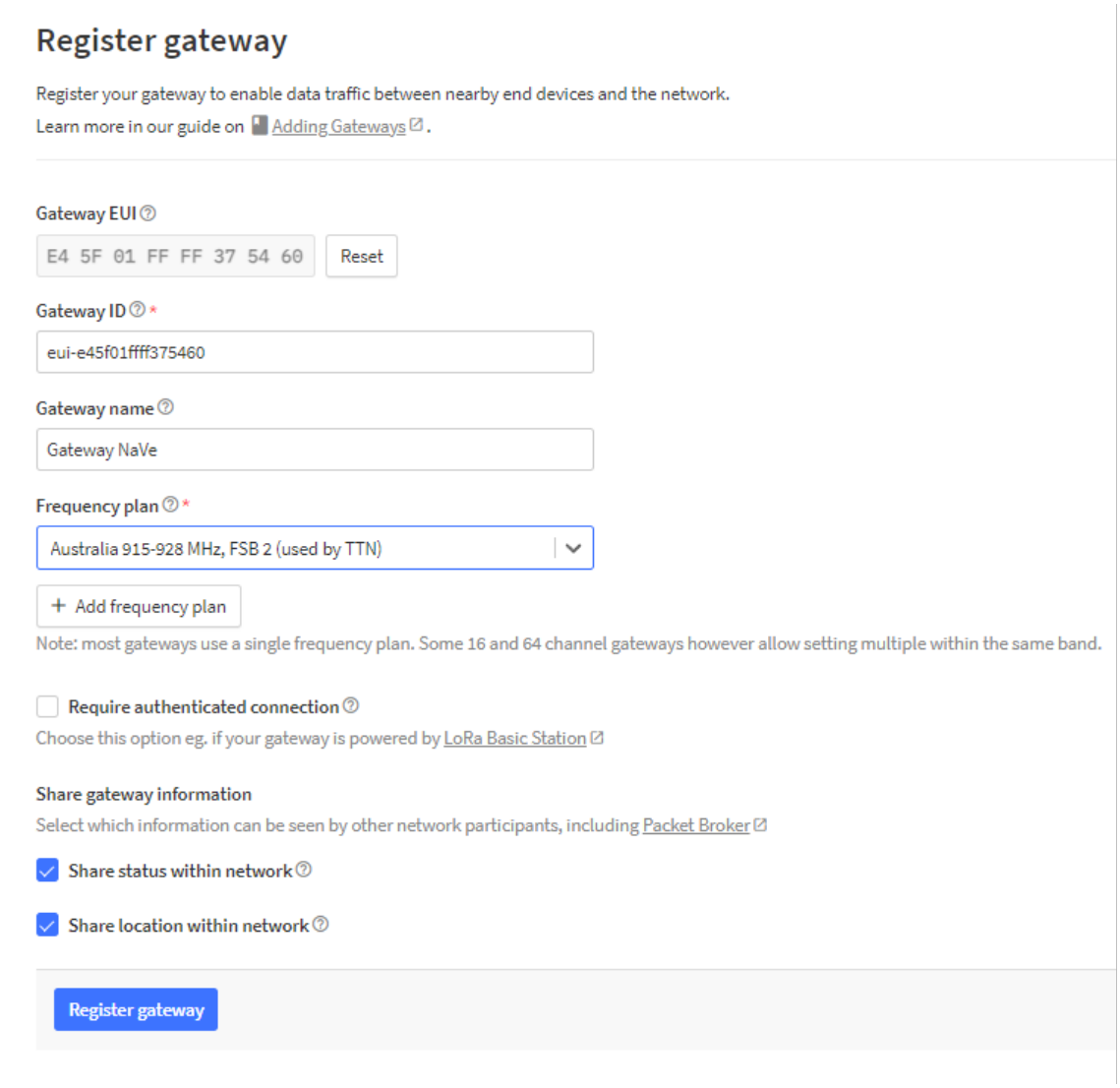


Figura 5.14: Registrando *gateway* en TTN



Figura 5.15: Frecuencias y SF usadas por TTN tanto de carga y descarga de mensajes

Con esto tenemos el *gateway* con la Raspberry Pi ya configurada y funcionando.

Como nos podemos fijar en la figura 5.15, hay una variedad de frecuencias que se usan para carga y otra variedad que se usan para descarga. Esto también envuelve el uso de distintos SF y BW de manera ya preconfigurada. Ahora, como estas variables no pueden ser configuradas de manera estática, es decir no se puede configurar una frecuencia en particular para recibir y/o enviar mensajes, ni tampoco el *Spreading Factor* y mucho menos el *Band Width*. El poder ejecutar los casos de prueba con las 18 configuraciones distintas, no es posible con TTN, y por ello tampoco con la Raspberry Pi.

b.- Dragino LG308-EC25

Ahora como segunda opción, usamos el *gateway* Dragino LG308-EC25. Para configurar este hay que entrar a su consola, esto se logra a través del navegador accediendo a la IP 10.130.1.1 con su respectivo usuario y contraseña. Como se requiere poder configurar algunos parámetros, como es la frecuencia, factor de dispersión y ancho de banda, en este aparato tenemos la opción de configurar estos, eligiendo a la frecuencia 917.1 MHz como la única frecuencia que utilizaremos (Fig. 5.16).

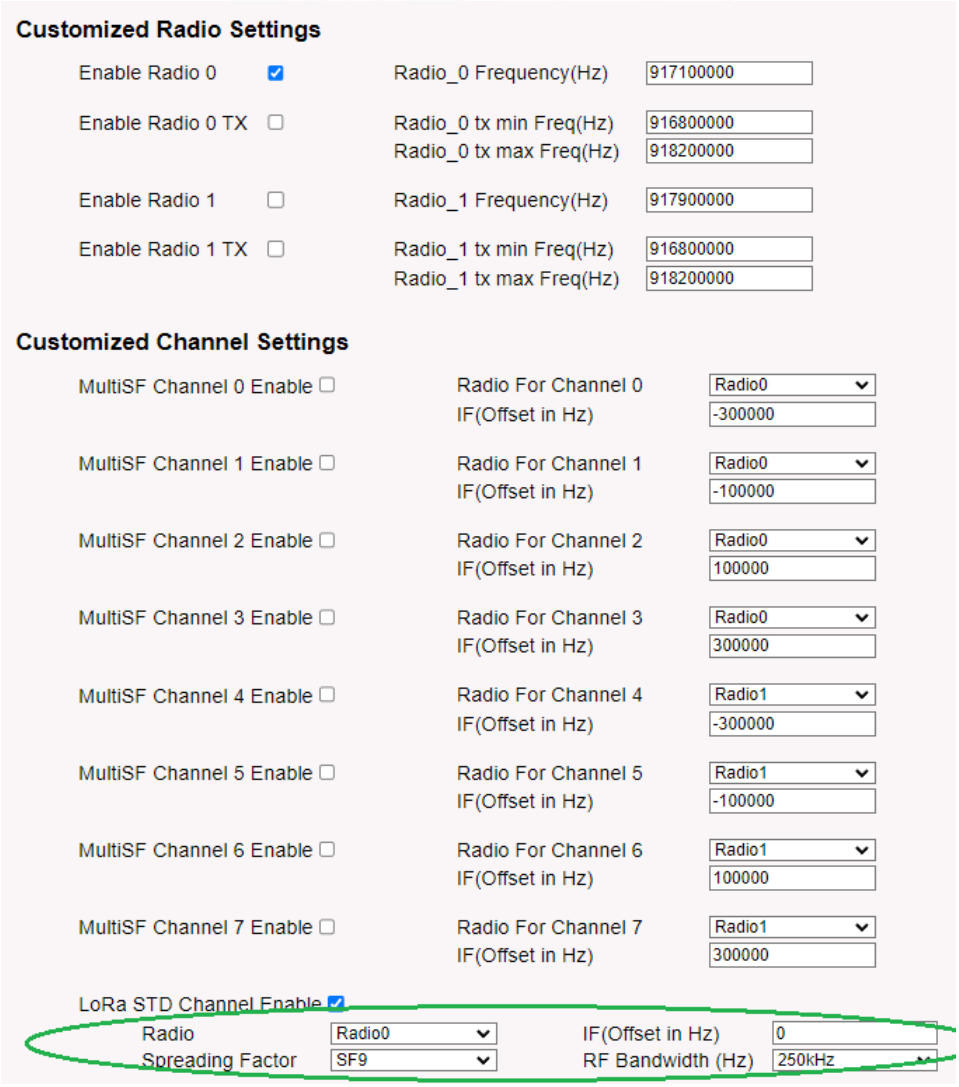


Figura 5.16: Configuración de parámetros en *gateway* LG308

Como podemos ver, esta configuración no es la que se usa en TTN, por lo que no podemos usar este como servidor LoRaWan. Pero este dispositivo nos permite decodificar

los mensajes que llegan por el protocolo ABP y enviarlos a través de MQTT a algún servidor privado, como se puede apreciar en la figura 5.17

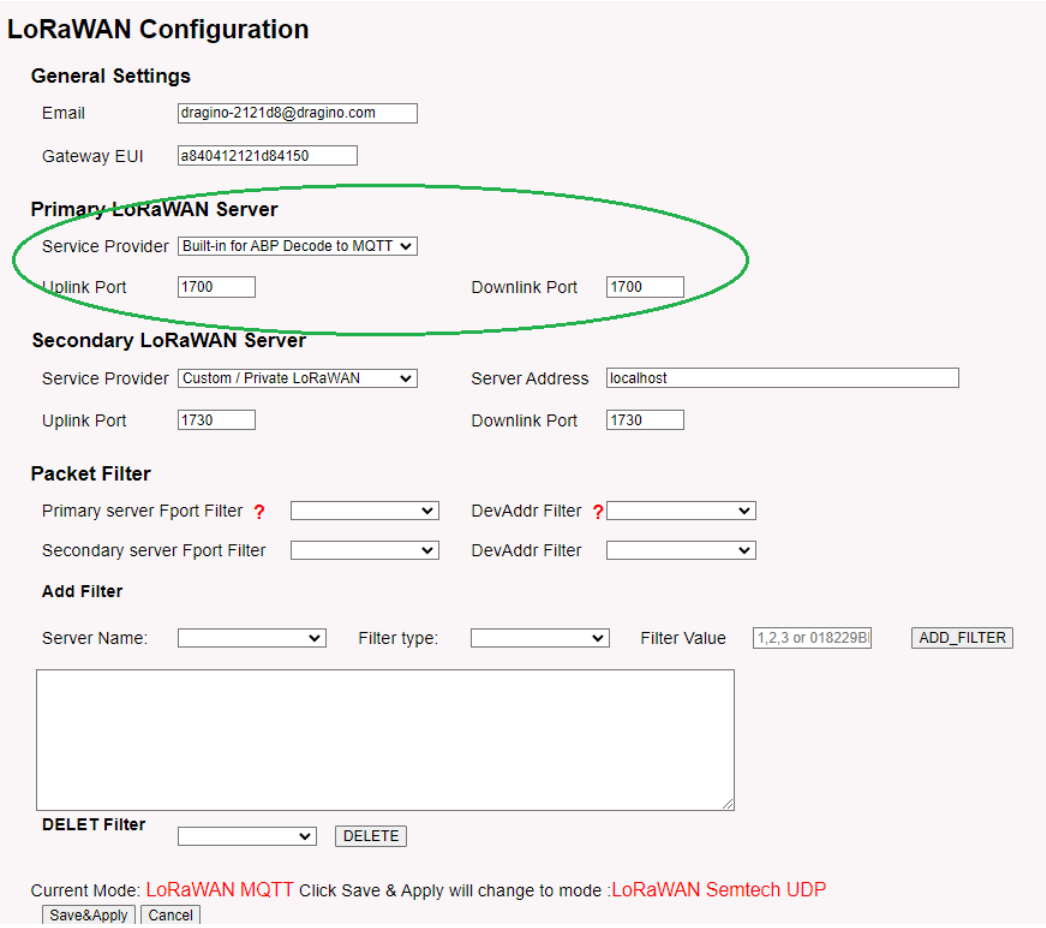


Figura 5.17: Configuración de servidores en *gateway* LG308

Se activó el protocolo ABP y se añadieron sus respectivas llaves, de esta manera los mensajes pueden ser recibidos y decodificado con "*ASCII String*", como muestra en la figura 5.18.



Figura 5.18: Incorporación de llaves secretas de protocolo ABP al *gateway* LG308

También se configuró la conexión al servidor MQTT ya creado y montado, cambiando los valores de la dirección IP del *Broker*, el ID del cliente, el formato del tópico al cual se publicará cada mensaje recibido, como así también el formato de este mensaje, como se ve en la imagen 5.19.

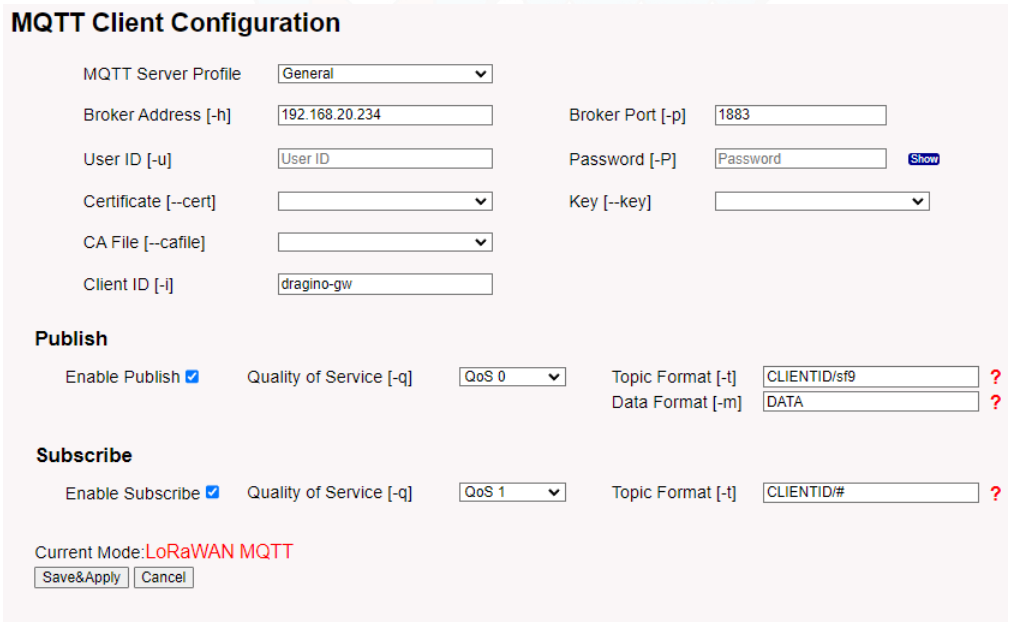


Figura 5.19: Configuración cliente MQTT del *gateway* LG308

Finalizando, se configuró también el canal usado en MQTT, esto se refiere a que solo los mensajes de los dispositivos con los IDs ingresados serán enviados (Fig. 5.20).

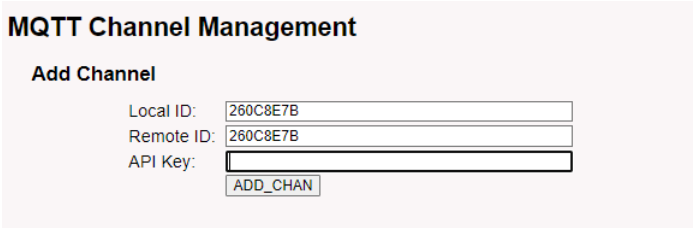


Figura 5.20: Agregación del canal MQTT

5.1.2.2. Servidores

En las siguientes figuras tenemos flujos de Node-Red, los cuales están suscritos a tópicos, en donde el primero (Fig.5.21) es un flujo que está suscrito a un tópico de MQTT en donde llegan los mensajes de las conexiones nodo-nodo, específicamente para el escenario

2. Y el segundo (Fig.5.22) es un flujo el cual está suscrito al tópico donde existe la conexión nodo-*gateway*, por el cual cada mensaje que llega es desencriptado y guardado en una base de datos de texto plano. Los mensajes de los primeros 3 escenario tienen el formato «$Mensaje\ <N\_de\_mensaje>$». Y para el cuarto, el formato es «$<Latitud>, <Longitud>, <N\_de\_mensaje>$»

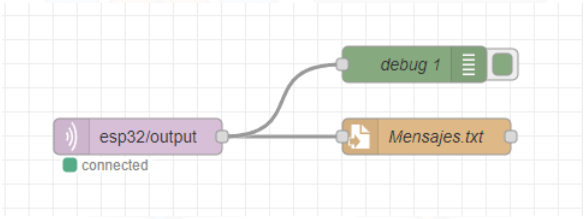


Figura 5.21: Flujo Node-Red para transmisión nodo a nodo usado en escenario 2

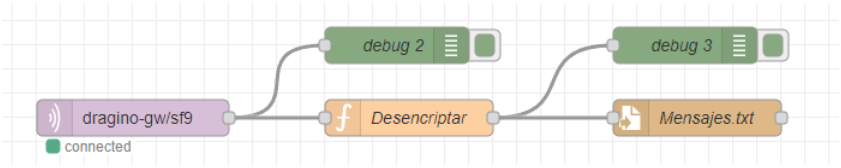


Figura 5.22: Flujo Node-Red para transmisión nodo a *gateway* usado en escenarios 3 y 4

5.1.2.3. Nodo LoRa

Para los nodos en nuestros casos de prueba hay cuatro configuraciones distintas:

- Nodo transmisor a otro nodo: equivale a un nodo transmisor de mensajes de prueba sin protocolo de seguridad, enviando un mensaje sin encriptar en un frecuencia, factor de dispersión y ancho de banda determinados y modificables. Utilizado en escenarios 1 y 2.
- Nodo receptor: es un nodo receptor sin protocolo de seguridad, recibiendo mensajes de prueba en una frecuencia, factor de dispersión y ancho de banda determinados y modificables. Utilizado en escenarios 1 y 2.
- Nodo transmisor a *gateway* TTN con protocolo OTAA: se trata de un nodo con llaves de seguridad del protocolo OTAA otorgadas por TTN, enviando mensajes en un rango

de frecuencias, con un factor de dispersión y ancho de banda predeterminados por TTN y no modificables. No utilizado en ningún escenario.

- Nodo sin ubicación GPS transmisor a *gateway* con protocolo ABP: nodo transmisor de mensajes de prueba con llaves de protocolo ABP extraídas de TTN, pero con frecuencia, factor de dispersión y ancho de banda determinados y modificables. Utilizado en el escenario 3.
- Nodo con ubicación GPS transmisor a *gateway* con protocolo ABP: nodo transmisor de mensajes con ubicación GPS en tiempo real con llaves de protocolo ABP extraídas de TTN, con frecuencia, factor de dispersión y ancho de banda determinados y modificables. Utilizado en el escenario 4.

Los códigos de cada configuración mencionada estarán disponibles en el apéndice.

5.2. Resultados

En esta sección se mostrará los resultados obtenidos por la implementación anteriormente descrita.

5.2.1. Escenario 1:

Este escenario fue llevado a cabo el 17 de enero del 2024 entre 2 a 8 P.M., aquí fueron probadas las 18 configuraciones. Enviando 10 mensajes cada 6 metros a lo largo de 60 metros en el pasillo del edificio B de Universidad Federico Santa María en Valparaíso (Fig. 5.6). Donde cada mensaje fue enviado del nodo transmisor cada 3 segundos al nodos receptor, el cual estaba conectado al computador vía USB, de esta manera podríamos leer los mensajes que llegaron y extraer la información necesaria, la cual se puede ver en la siguientes tablas:

Tabla 5.2: Resultados RSSI en dBm para escenario 1

Configuración		6 [m]	12 [m]	18 [m]	24 [m]	30 [m]	36 [m]	42 [m]	48 [m]	54 [m]	60 [m]
SF=7	BW=125 [kHz]	-87.3	-85.8	-93.8	-92.2	-90.1	-104.5	-103.1	-96.3	-91.3	-99.1
SF=7	BW=250 [kHz]	-76	-79.5	-79	-81.1	-86.1	-89	-84.2	-87.5	-83.9	-83.9
SF=7	BW=500 [kHz]	-81.2	-79.2	-80.1	-92.6	-91.5	-90.7	-91.7	-89.4	-90.1	-91.7
SF=8	BW=125 [kHz]	-81.3	-87.2	-92.9	-90.8	-90.6	-97.4	-99	-103.5	-105.2	-109.6
SF=8	BW=250 [kHz]	-94.1	-93.4	-100.2	-95.3	-96.3	-98.4	-95.4	-93.4	-97.7	-95.2
SF=8	BW=500 [kHz]	-83.6	-81.7	-89.2	-90.3	-93.9	-94.4	-95.4	-94.1	-96.1	-96.8
SF=9	BW=125 [kHz]	-86.4	-93.3	-87.9	-87.5	-95.3	-101	-108.1	-107.9	-100.3	-100.3
SF=9	BW=250 [kHz]	-84.7	-79.6	-100	-88.9	-93.9	-91.7	-90.5	-87.9	-89.4	-90
SF=9	BW=500 [kHz]	-91.8	-92.4	-95.7	-92.7	-97.7	-98	-100.6	-103.9	-104	-104.9
SF=10	BW=125 [kHz]	-83.8	-90.4	-100.8	-90.8	-100.6	-100.9	-105.3	-96.6	-96.9	-101.3
SF=10	BW=250 [kHz]	-89.5	-82.1	-93.2	-87.6	-88.4	-94.5	-91	-96.5	-106.7	-99.9
SF=10	BW=500 [kHz]	-80.9	-88.2	-91.4	-89.6	-95.3	-93.4	-92.8	-92.7	-102.9	-100.1
SF=11	BW=125 [kHz]	-92.4	-97.8	-97.7	-94.4	-100.2	-104.4	-108	-103.2	-102.4	-98.1
SF=11	BW=250 [kHz]	-92.8	-92.3	-93.5	-90.6	-101.9	-100	-98.5	-100.9	-107.1	-100.6
SF=11	BW=500 [kHz]	-87.4	-83.3	-98.3	-87.2	-98.2	-94.2	-96.3	-98.6	-106.3	-102.6
SF=12	BW=125 [kHz]	-91.1	-90.2	-91.7	-92.9	-97.2	-97.1	-96.3	-95.9	-100.2	-94.9
SF=12	BW=250 [kHz]	-84.7	-82.8	-91.6	-89.3	-97.3	-94.1	-95.5	-106	-101.3	-97.4
SF=12	BW=500 [kHz]	-85.6	-90.9	-92.7	-89.2	-97.5	-95.3	-92.6	-93	-102.1	-103.9

Tabla 5.3: Resultados SNR en dBm para escenario 1

Configuración		6 [m]	12 [m]	18 [m]	24 [m]	30 [m]	36 [m]	42 [m]	48 [m]	54 [m]	60 [m]
SF=7	BW=125 [kHz]	6.8	6.7	5.8	6.5	6.9	6.1	6.6	6.7	6.8	6.6
SF=7	BW=250 [kHz]	6.6	7.1	7.1	7.2	7.1	7	7.2	7.1	7	7.1
SF=7	BW=500 [kHz]	6	6	6	5.5	5.8	5.4	5.5	5.6	5.3	5.2
SF=8	BW=125 [kHz]	7.8	7.7	7.7	7.4	7.6	7.9	8	7	6.2	3.1
SF=8	BW=250 [kHz]	7.9	8	6.8	7.7	7.2	7.2	7.6	7.6	6.8	7.9
SF=8	BW=500 [kHz]	6.8	7.2	6.7	7.1	6.9	6.8	6.3	6.8	6.3	6.3
SF=9	BW=125 [kHz]	7.4	7.6	7.9	7.5	7.9	7.2	6.9	6.1	7.8	8
SF=9	BW=250 [kHz]	7.9	7.9	7.1	7.7	7.7	7.4	7.3	8.5	7.8	8.7
SF=9	BW=500 [kHz]	6.9	6.9	6.4	7.3	6.1	6	5.4	3.8	4.1	1.8
SF=10	BW=125 [kHz]	7.9	7.4	7.4	6.9	7.4	7.4	7.1	7.5	7.7	7.5
SF 10	BW=250 [kHz]	7.8	7.4	7.4	7.6	7.4	7.5	7.8	7.6	6.3	6.6
SF=10	BW=500 [kHz]	7.8	7.2	7.7	6.7	7	6.5	6.7	7	5.8	6.2
SF=11	BW=125 [kHz]	8.5	8.5	8.9	9	9.2	8.8	7.7	8.8	8.8	9.1
SF=11	BW=250 [kHz]	7.5	7.1	7.9	7.5	6	6.9	7	6.6	5.3	6.4
SF=11	BW=500 [kHz]	6.9	7.4	6.4	6.7	6.2	6.7	6.6	6	3.8	5.9
SF=12	BW=125 [kHz]	5.9	6.6	6.9	6.4	6.5	6.6	6	6.7	6.4	6.9
SF=12	BW=250 [kHz]	8.1	8.6	8	8.3	7.8	8.2	7.1	6.4	6.7	8.2
SF=12	BW=500 [kHz]	6.4	5.9	6.6	6	5.8	6.1	6.2	6.6	4.5	3.7

Dando como resultado los siguientes gráficos:

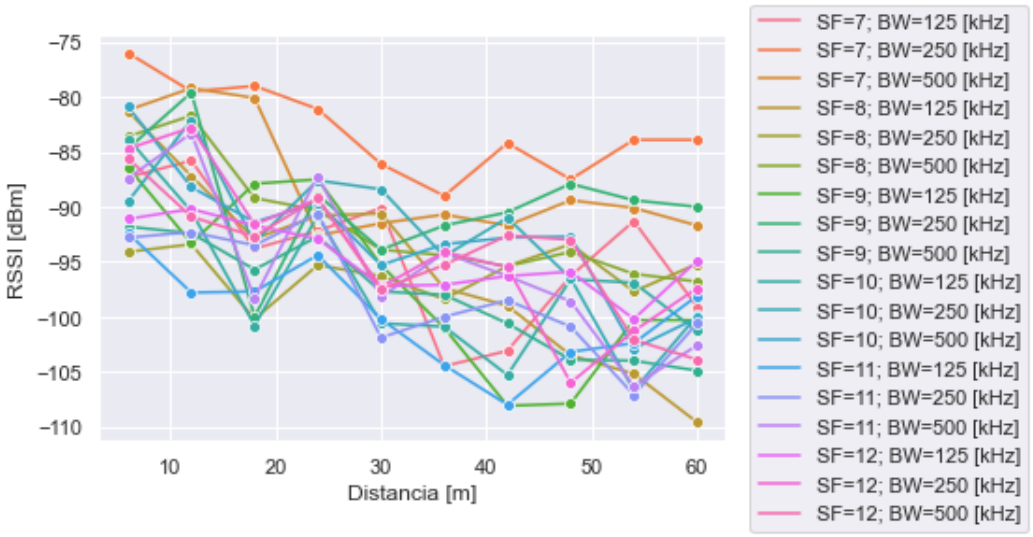


Figura 5.23: Valores RSSI vs Distancia para las configuraciones usadas en el escenario 1

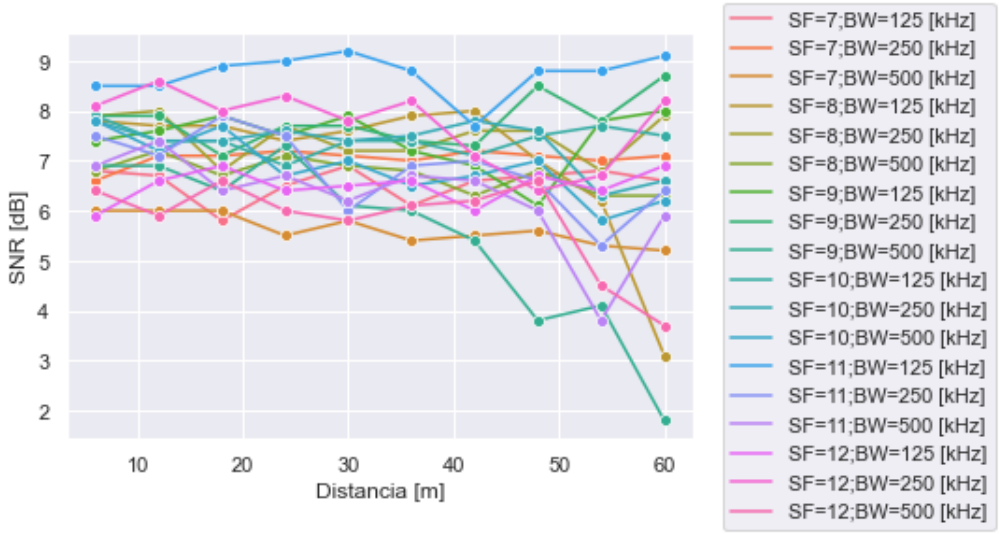


Figura 5.24: Valores SNR vs Distancia para las configuraciones usadas en el escenario 1

Para poder evaluar las configuraciones se toma el promedio para cada configuración de las tablas anteriores, se le agrega la perdida de paquetes para cada configuración y se crea la consecuente tabla:

Teniendo esta tabla con el promedio de los valores obtenidos, se procedió a sacar la media para cada columna y de esta manera poder encontrar las configuraciones que estaban

Tabla 5.4: Resultados RSSI promedio, SNR promedio y paquetes recibidos para escenario 1

Configuración		RSSI Prom.	SNR Prom	Packet Received
SF=7	BW=125 [kHz]	-94.1	6.6	97.0 %
SF=7	BW=250 [kHz]	-83	7	100.0 %
SF=7	BW=500 [kHz]	-87.8	5.6	100.0 %
SF=8	BW=125 [kHz]	-95.8	7	100.0 %
SF=8	BW=250 [kHz]	-95.9	7.5	100.0 %
SF=8	BW=500 [kHz]	-91.5	6.7	100.0 %
SF=9	BW=125 [kHz]	-96.8	7.4	100.0 %
SF=9	BW=250 [kHz]	-89.7	7.8	100.0 %
SF=9	BW=500 [kHz]	-98.2	5.5	100.0 %
SF=10	BW=125 [kHz]	-96.7	7.4	100.0 %
SF=10	BW=250 [kHz]	-92.9	7.3	100.0 %
SF=10	BW=500 [kHz]	-92.7	6.9	100.0 %
SF=11	BW=125 [kHz]	-99.7	8.7	98.0 %
SF=11	BW=250 [kHz]	-97.7	6.8	99.0 %
SF=11	BW=500 [kHz]	-95.2	6.3	99.0 %
SF=12	BW=125 [kHz]	-94.8	6.5	100.0 %
SF=12	BW=250 [kHz]	-94	7.7	100.0 %
SF=12	BW=500 [kHz]	-94.3	5.8	100.0 %

sobre esta. Dándonos como resultado la consiguiente lista:

Tabla 5.5: Resultados filtrados de RSSI promedio, SNR promedio y paquetes recibidos para escenario 1

Configuración		RSSI Prom.	SNR Prom	Packet Received
SF=7	BW=250 [kHz]	-83	7	100.0 %
SF=9	BW=250 [kHz]	-89.7	7.8	100.0 %
SF=10	BW=250 [kHz]	-92.9	7.3	100.0 %

5.2.2. Escenario 2:

Con las configuraciones que resaltaron sobre la media en el escenario anterior se efectuaron las prueba en este escenario. Este fue llevado a cabo el 18 de enero del 2024 entre 3 a 4 P.M. en la misma ubicación del escenario anterior (Fig. 5.6). En este escenario el nodo receptor está conectado por WiFi a un servidor MQTT corriendo en un notebook, el cual está conectado a la misma red WiFi, y usando el flujo de Node-Red mostrado en la figura 5.21 se guardó cada mensaje en el dispositivo. Cabe mencionar que se usó el hotspot

de un celular como punto de acceso WiFi para cada dispositivo. Dando los resultados a continuación:

Tabla 5.6: Resultados RSSI en dBm para escenario 2

Distancia	SF=7; BW=250 [kHz]	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	-74	-89.8	-82.3
12 [m]	-82.9	-81.4	-83.4
18 [m]	-80.6	-84.5	-86.8
24 [m]	-91.6	-86.3	-85.1
30 [m]	-83.9	-91.7	-89.5
36 [m]	-84.6	-86.7	-82.3
42 [m]	-87.2	-87.4	-91.2
48 [m]	-97.6	-92.4	-92.7
54 [m]	-85.8	-102.1	-96.5
60 [m]	-86.1	-95.3	-95.6

Tabla 5.7: Resultados SNR en dBm para escenario 2

Distancia	SF=7; BW=250 [kHz]	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	7.1	8.1	8.1
12 [m]	7	6.8	7.7
18 [m]	7	8.5	8
24 [m]	6.8	8.1	8
30 [m]	7.1	8	7.6
36 [m]	7	7.7	7.9
42 [m]	7.1	8.6	8
48 [m]	6.3	7	7.9
54 [m]	7	7	7
60 [m]	6.9	7.6	7.4

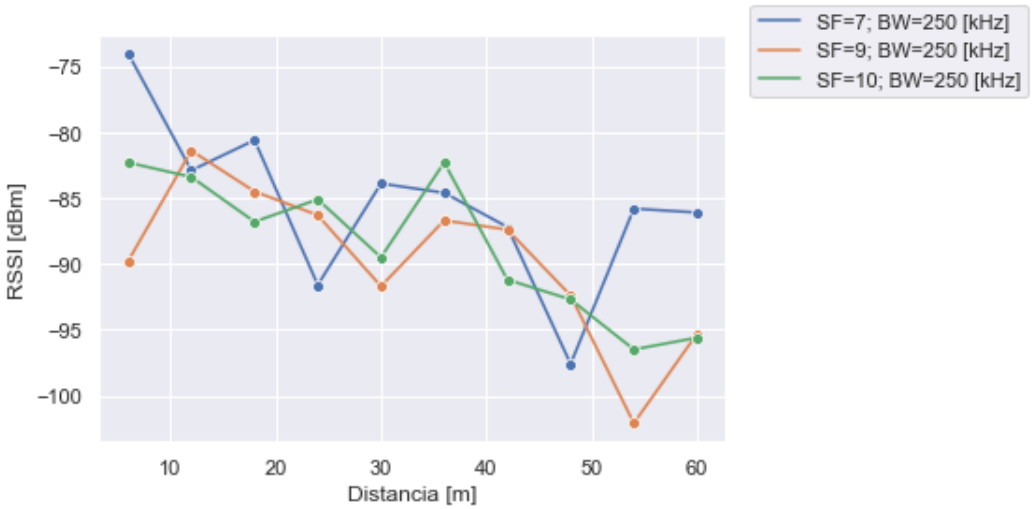


Figura 5.25: Valores RSSI vs Distancia para las configuraciones usadas en el escenario 2

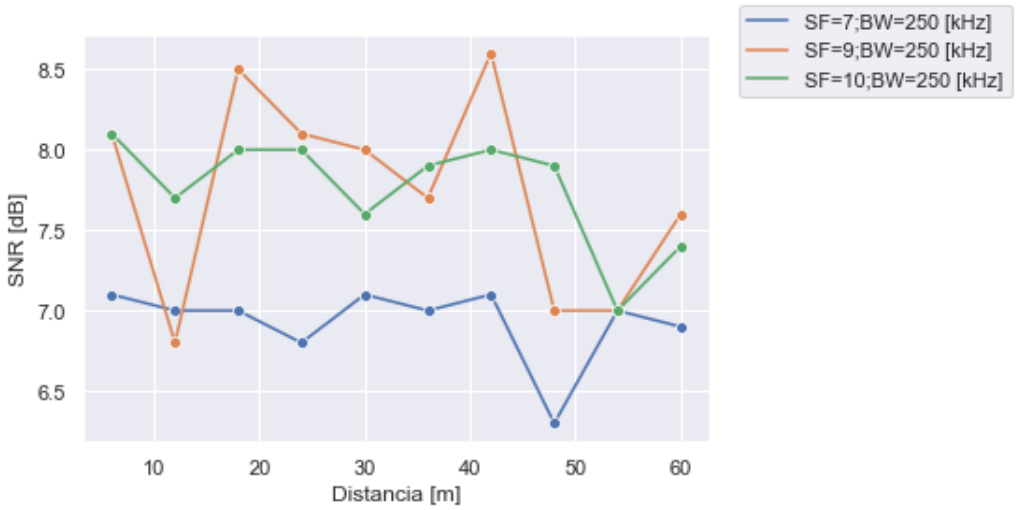


Figura 5.26: Valores SNR vs Distancia para las configuraciones usadas en el escenario 2

Repitiendo lo que se hace en el escenario 1, se crea una tabla con sus valores promedio y se le agrega la perdida de paquetes para cada configuración:

Tabla 5.8: Resultados RSSI promedio, SNR promedio y paquetes recibidos para escenario 2

Configuración	RSSI Prom.	SNR Prom	Packet Received
SF=7 BW=250 [kHz]	-85.6	6.9	97.0 %
SF=9 BW=250 [kHz]	-89.8	7.7	100.0 %
SF=10 BW=250 [kHz]	-88.5	7.8	99.0 %

Dado que la primera configuración (SF=7 con BW=250 [kHz]) tiene una pérdida de paquetes mayor y un SNR menor al resto, se descarta.

5.2.3. Escenario 3:

Con las configuraciones restantes del escenario anterior se llevarán a cabo las pruebas en este marco. Fue realizado el 7 de febrero del 2024 entre 4 a 6 P.M. en la comuna de Pirque, Santiago (Fig. 5.7). En este escenario ya no hay nodo receptor, sino que existe un *gateway* conectado a la misma red WiFi que el servidor MQTT y Node-Red corriendo en un notebook. Los mensajes son enviados cada 10 segundos para el caso de la configuración con SF=9 y cada 18 segundos para la configuración con SF=10, los cuales son recibidos por el *gateway*, este publica los mensajes a un tópico al servidor MQTT, y con el flujo Node-Red 5.22 es recibido, desencriptado y guardado. Cabe mencionar que de nuevo se usó el hotspot de un celular como punto de acceso WiFi para cada dispositivo. Dando los resultados a continuación:

Tabla 5.9: Resultados RSSI en dBm para escenario 3

Distancia	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	-60.3	-64.3
12 [m]	-63.2	-82.7
18 [m]	-66.7	-79.2
24 [m]	-64.3	-78.7
30 [m]	-63.6	-80.8
36 [m]	-65.5	-81.6
42 [m]	-73.1	-91.1
48 [m]	-73.2	-79
54 [m]	-75.4	-79.3
60 [m]	-80.2	-86.3

Tabla 5.10: Resultados SNR en dBm para escenario 3

Distancia	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	11	10.6
12 [m]	10.2	10.7
18 [m]	11	10.5
24 [m]	10.2	11
30 [m]	10.7	10.7
36 [m]	10.6	10.9
42 [m]	11.1	10
48 [m]	10.9	10.5
54 [m]	11	10.8
60 [m]	10.9	10.4

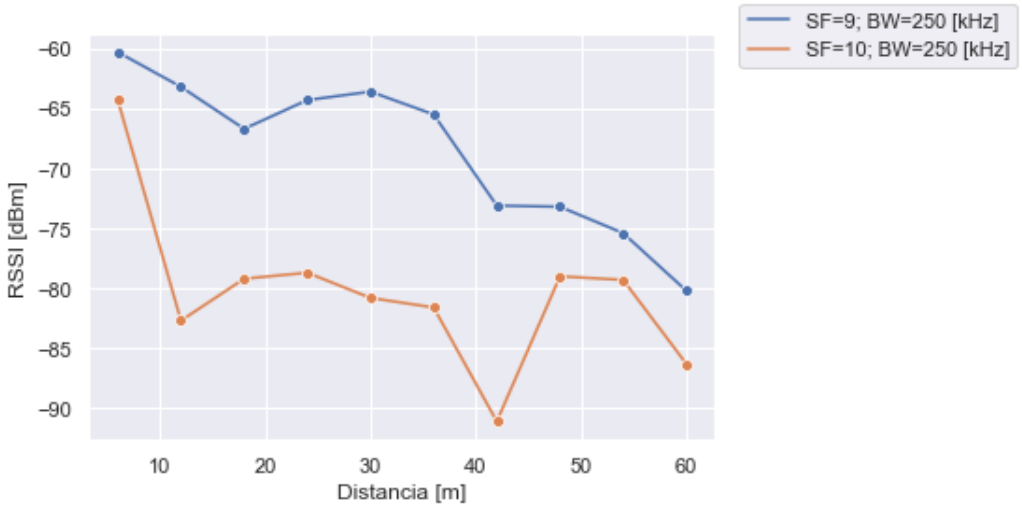


Figura 5.27: Valores RSSI vs Distancia para las configuraciones usadas en el escenario 3

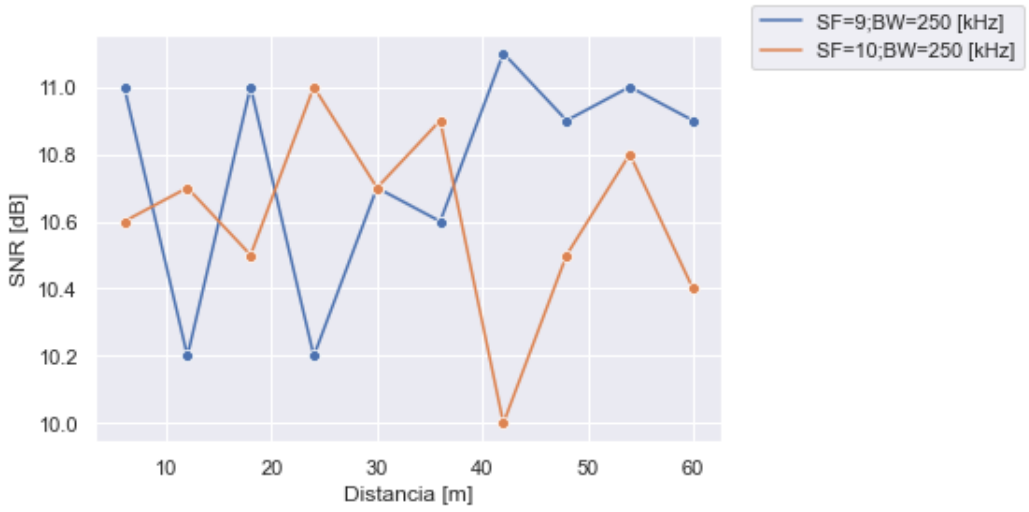


Figura 5.28: Valores SNR vs Distancia para las configuraciones usadas en el escenario 3

5.2.4. Escenario 4:

Este escenario fue concretado el 13 de febrero del 2024 entre 5 a 7 P.M. en el mismo lugar del escenario anterior (Fig. 5.7). Lo que cambia en este escenario respecto al anterior, es que los mensajes que se envían, poseen la latitud y longitud del nodo en tiempo real.

Tabla 5.11: Resultados RSSI en dBm para escenario 4

Distancia	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	-54.7	-57.8
12 [m]	-64.3	-62.5
18 [m]	-71.2	-74.1
24 [m]	-71	-73.8
30 [m]	-81	-75.1
36 [m]	-79.6	-77.3
42 [m]	-76.6	-75.9
48 [m]	-78.3	-77.8
54 [m]	-87.7	-75.8
60 [m]	-83.1	-84.5

Tabla 5.12: Resultados SNR en dBm para escenario 4

Distancia	SF=9; BW=250 [kHz]	SF=10; BW=250 [kHz]
6 [m]	10.9	10.9
12 [m]	10.6	11
18 [m]	10.6	11.1
24 [m]	11.2	11
30 [m]	10.1	11.2
36 [m]	11.1	11
42 [m]	10.9	11.1
48 [m]	11.2	11.1
54 [m]	10.5	11.2
60 [m]	11	10.9

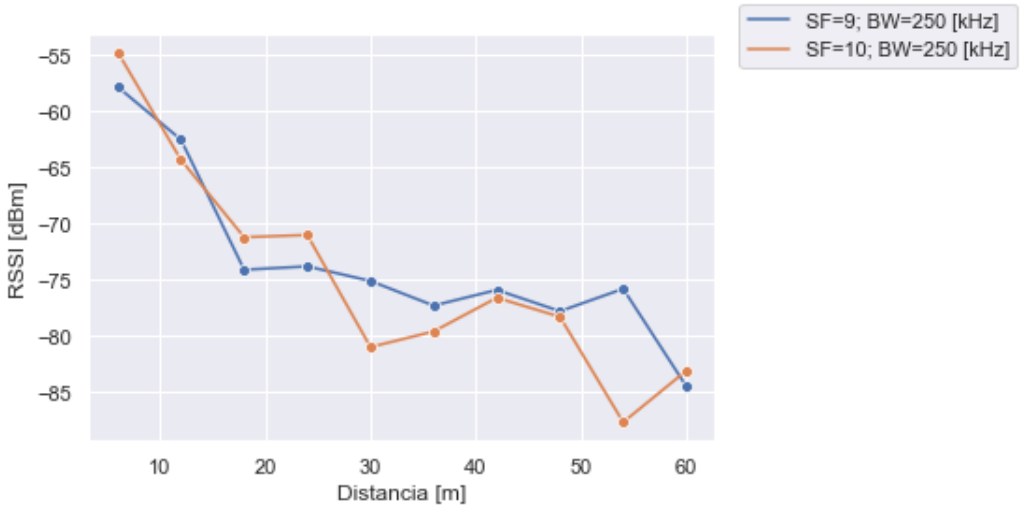


Figura 5.29: Valores RSSI vs Distancia para las configuraciones usadas en el escenario 4

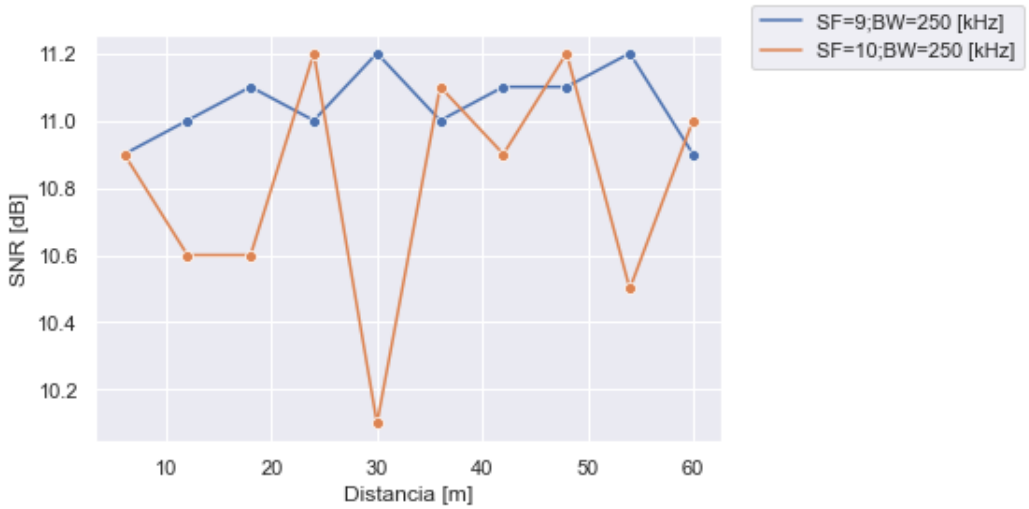


Figura 5.30: Valores SNR vs Distancia para las configuraciones usadas en el escenario 4

6 | Simulación y resultados

Este capítulo está enfocado en mostrar el desarrollo y resultados de la simulación para la capa de comunicación. Con esto, podremos demostrar la posibilidad de comunicación de un y varios nodos (vehículos) dentro del campus de la Universidad Federico Santa María en Valparaíso.

Como podemos ver en la figura 5.1, hay cinco estacionamientos, por lo tanto, se decidió que cada uno de estos tendrá su respectivo unidad colectora de información o gateway. Esto se demuestra en la siguiente imagen:

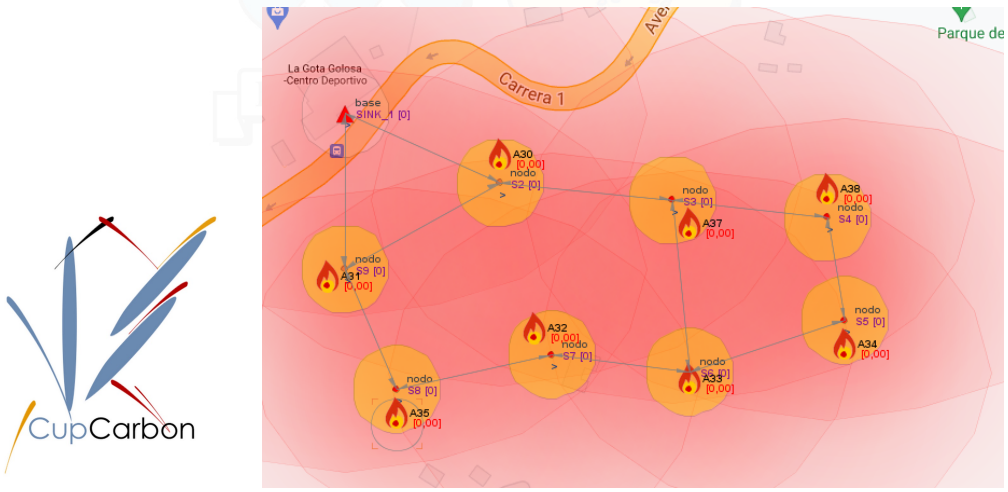


Figura 6.1: Mapa satelital con *gateways* ubicados en estacionamientos en campo central UTFSM

Esta simulación será generada y probada por CupCarbon. Esta es una plataforma de

simulación de redes inalámbricas y móviles. Es una herramienta muy útil tanto para investigadores como para profesionales que trabajan en el campo de las redes de comunicación inalámbrica.

Una de las características principales de CupCarbon es que permite simular diferentes tipos de redes inalámbricas, como redes de sensores, redes LoRa, WiFi, ZigBee, entre otras. Los usuarios pueden definir la estructura de la red, del entorno, como son los edificios, clima, humedad, temperatura, etc, y simular su comportamiento en diferentes variadas situaciones.



(a) Logo CupCarbon **(b)** Ejemplo red de sensores de incendios con CupCarbon

Figura 6.2

Además de las redes fijas, CupCarbon también permite simular dispositivos móviles como teléfonos inteligentes y vehículos. Esto es muy útil para estudiar el rendimiento de las redes móviles y los protocolos de enrutamiento en entornos dinámicos.

Una de las ventajas de CupCarbon es que ofrece herramientas de visualización interactiva que facilitan el análisis de los resultados de la simulación. Esto incluye la representación gráfica de la estructura de la red, la cobertura de la señal, los enlaces entre dispositivos y otros parámetros relevantes.

En cuanto a sus aplicaciones, CupCarbon se utiliza principalmente en la investigación y desarrollo de protocolos de enrutamiento eficientes para redes inalámbricas y móviles. También es muy útil para el diseño y optimización de redes de sensores en diferentes

entornos.

La siguiente imagen podemos ver nuestro escenario ya implementado en este gran simulador:

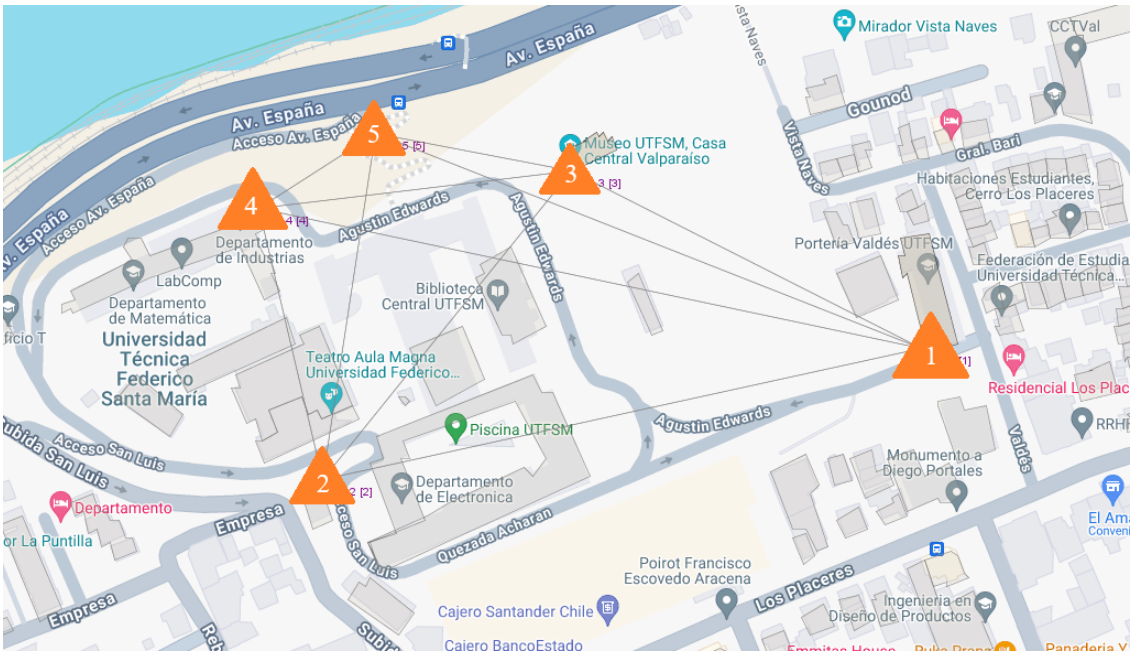


Figura 6.3: Mapa vial con *gateways* ubicados en estacionamientos en campo central UTFSM

Acá nos centraremos en dos objetivos, primero probar las 18 configuraciones vistas en el capitulo anterior, pero con mayores distancias. Y el segundo objetivo es probar la comunicación de nodos estáticos y móviles dentro del campus, teniendo cada *gateway* posicionado en el mapa dentro de CupCarbon como se puede ver en la imagen 6.3.

6.1. Desarrollo

6.1.1. Primer objetivo:

Para abarcar el este objetivo, se usó un sector plano, sin edificios ni interferencia climática (Fig. 6.4). Esto se abordó usando dos nodos, uno transmisor y otro receptor, dónde el primero enviará mensajes cada 5 segundos y el otro cambiará de ubicación cada 100 mensajes recibidos, abarcando las distancias de 3, 5, 10, 20, 50, 100, 200,

500, 1000, 2000, 5000 metros del transmisor. Todo esto repetido por cada una de las 18 configuraciones.

Para las configuraciones de los nodos, hay un archivo SenScript, donde la lógica ocupada por este es simple de utilizar y entender (Fig. 6.5 y Fig. 6.6).

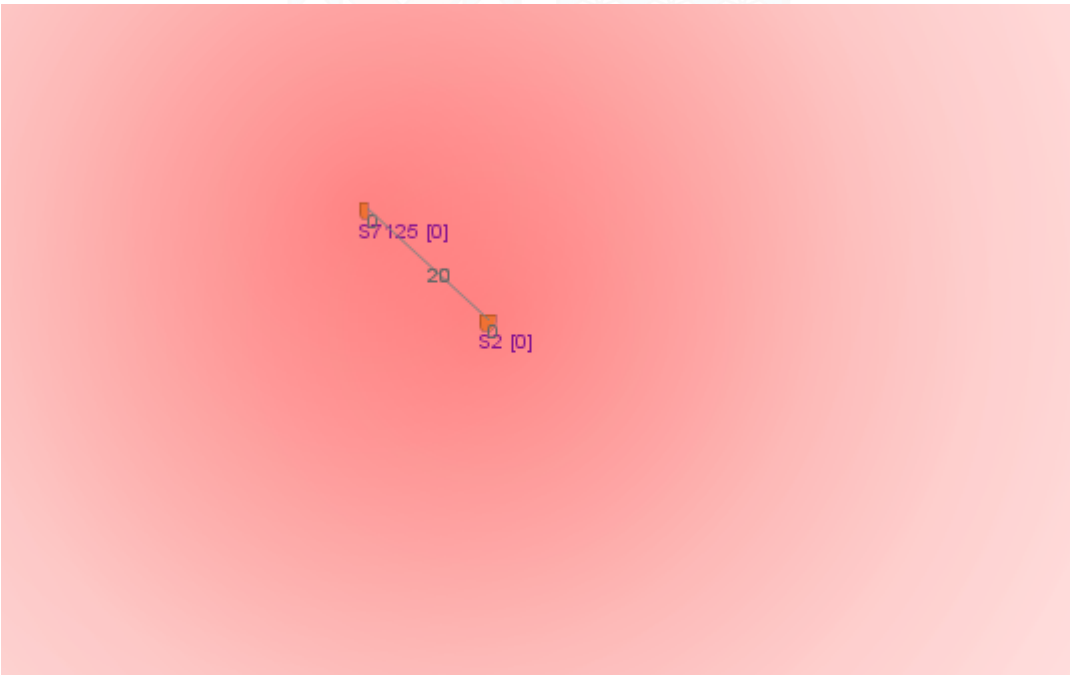


Figura 6.4: Nodos transmisor y receptor en sector plano

```
//Receptor

// Contador para cambio de posición del nodo
int count 0

// Inicio de loop de recepcion
loop
        // Se mueve el nodo receptor cada 100 mensajes recibidos
        if (count==0)
                // 3 metros del transmisor
                move -71.5895909070968 -33.03657250269227 0 100000
                int d 3
        end
        if (count==1)
                // 10 metros del transmisor
                move -71.5895265340805 -33.036608479595166 0 10000
                int d 10
        end
        if (count==2)
                // 20 metros del transmisor
                move -71.58944070339203 -33.0366601963713 0 10000
                int d 20
        end
        if (count==3)
                // 50 metros del transmisor
                move -71.58919662237167 -33.03683783199614 0 10000
                int d 50
        end
        if (count==4)
                // 100 metros del transmisor
                move -71.58878356218338 -33.03712564598528 0 10000
                int d 100
        end
        if (count==5)
                // 200 metros del transmisor
                move -71.58786088228226 -33.03759334167648 0 10000
                int d 200
        end
        if (count==6)
                // 500 metros del transmisor
                move -71.58502846956253 -33.03887949176265 0 10000
                int d 500
        end
        if (count==7)
                // 1000 metros del transmisor
                move -71.58026486635208 -33.04094808538134 0 10000
                int d 1000
        end
```

```
if (count==8)
        // 2000 metros del transmisor
        move -71.57120436429977 -33.0457573883525 0 10000
        int d 2000
end
if (count==9)
        // 5000 metros del transmisor
        move -71.55553758144373 -33.07123169022656 0 10000
        int d 5000
end

// Recibe el mensaje
receive msj

// Tranforma el mensaje a un int
int num msj

// Agrega linea a archivo log con la distancia
// y el numero de mensaje
printfile d num

// Incrementa el count para incrementar la distancia
// al llegar a mensaje número 100
if (num==100)
        inc count
end
```

Figura 6.5: Código SenScript para nodo receptor para primer objetivo

```
//Transmisor

// Contador de mensajes enviados
int num 0

// Contador de distancia
int d 0

// Se demora 5 segundos en empezar el loop para transmitir
delay 5000

// Empieza el loop de transmisión
loop
        //  Incrementa contador de mensaje
        inc num
        // Envia mensaje, el cual es el contador
        send num

        // Si ya se enviaron 100 mensajes,
        // se resetea el valor del contador de mensajes
        // y se incrementa el contador de distancia
        if (num==100)
                int num 0
                inc d
        end
        // Si ya fueron 10 ditancias distintas,
        // la simulación se detiene
        if (d==10)
                stop
        end

        // Espera 5 segundos para volver a transmitir
        delay 5000
```

Figura 6.6: Código SenScript para nodo transmisor para primer objetivo

Como en CupCarbon no es posible configurar directamente el BW, si no que el *Data Rate* (DR), este tuvo que ser calculado por cada configuración necesitada, con la siguiente formula:

$$R_b = SF * \frac{\left[\frac{4}{4+CR}\right]}{\left[\frac{2^{SF}}{BW}\right]} * 1000$$

SF = Spreading Factor (6,7,8,9,10,11,12)
CR = Code Rate (1,2,3,4)
BW = Bandwidth in KHz
(10.4,15.6,20.8,31.25,41.7,62.5,125,250,500)
Rb = Data rate or Bit Rate in bps

Dándonos como resultado, para cada configuración, la siguiente tabla:

	125 [kHz]		250 [kHz]		500 [kHz]	
SF	Data Rate	Distancia	Data Rate	Distancia	Data Rate	Distancia
7	5468 [bit/s]	2900 [m]	10937 [bit/s]	2380 [m]	21875 [bit/s]	1720 [m]
8	3125 [bit/s]	3410 [m]	6250 [bit/s]	2800 [m]	12500 [bit/s]	2020 [m]
9	1757 [bit/s]	4020 [m]	3515 [bit/s]	3300 [m]	7031 [bit/s]	2380 [m]
10	976 [bit/s]	4730 [m]	1953 [bit/s]	3890 [m]	3906 [bit/s]	2800 [m]
11	537 [bit/s]	5570 [m]	1074 [bit/s]	4580 [m]	2148 [bit/s]	3300 [m]
12	292 [bit/s]	6560 [m]	585 [bit/s]	5390 [m]	1171 [bit/s]	3880 [m]

6.1.2. Segundo objetivo:

Para simular este objetivo, se ajustó cada *gateway* y nodo utilizado con una sola configuración (SF=7 y DR=2500), la cual no fue cambiada, ya que no nos centraremos en esto en este objetivo. Acá, nos centraremos en buscar si hay perdida de paquetes y ver la energía consumida nodo transmisor (vehículo).

Como fue mostrado en las Figuras 6.1 y 6.3, se ocupará el campus central de la UTFSM, simulando también la existencia de los edificios que existen en el lugar. Como en el capitulo anterior, usaremos escenarios, acá específicamente tres:

Para cada nodo, el código SenScript utilizado fue el mismo para cada escenario:

Tabla 6.1: Escenarios y sus diseños

Escenarios	Diseño
1	Comunicación de un nodo estático con los *gateways*
2	Comunicación de un nodo móvil con los *gateways*
3	Comunicación de 4 nodos móviles con los *gateways*

```
//Receptor

// Empieza loop para recibir mensajes
loop
        // Recibe mensaje, si es que hay
        receive v

        // Transforma mensaje lat#long#id#num a vectores
        vdata data v

        // Asigna cada vector a una variable
        vget lat data 0
        vget long data 1
        vget id data 2
        vget count data 3

        // Añade una fila a archivo log
        printfile "lat:"lat "long:"long "id:"id "count:"count
```

Figura 6.7: Código SenScript para nodo receptor para segundo objetivo

```
//Transmisor

// Contador de mensajes enviados
int num 0

// Se demora 5 segundos en empezar el loop para transmitir
delay 5000

// Empieza el loop de transmisión
loop

        // Obtiene la posicion del nodo con la estructura lat#long
        getpos pos

        // Obtiene el id del nodo
        atget id id

        // Estrucutra el mensaje a enviar como lat#long#id#num
        data data pos id num

        // Envia el mensaje
        send data

        // Incremente el contador de mensajes
        inc num

        // Si ya pasó una hora, se detiene la simulación
        if (num==720)
                stop
        end

        // Espera 5 segundos para transmitir de nuevo
        delay 5000
```

Figura 6.8: Código SenScript para nodo transmisor para segundo objetivo

6.1.2.1. Escenario 1

En este escenario se ocupó un solo nodo, ubicado abajo del mapa, cerca del estacionamiento PL1 (Fig. 6.9). Acá se simuló una comunicación de nodo-*gateways* por una hora, enviando mensajes cada 5 segundos, donde el mensaje del nodo conformaba de el siguiente formato: «< *lat* > # < *long* > # < *id_nodo* > # < *contador_mensajes* >». Dando un máximo de 43 bytes en cada mensaje.

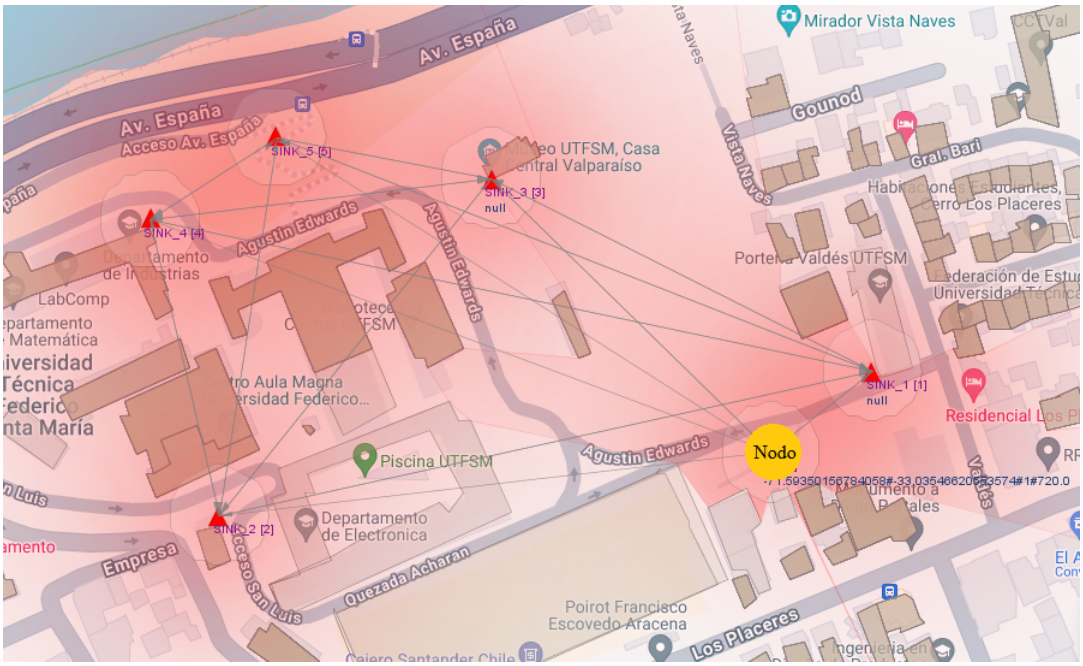


Figura 6.9: Nodo estático

6.1.2.2. Escenario 2

En este escenario se ocupo un solo nodo, el cual seguirá un ruta especifica, con una media de 20 km/h (Fig. 6.10). Como en el escenario anterior, los mensajes son enviados cada 5 segundos, siguiendo el mismo formato.

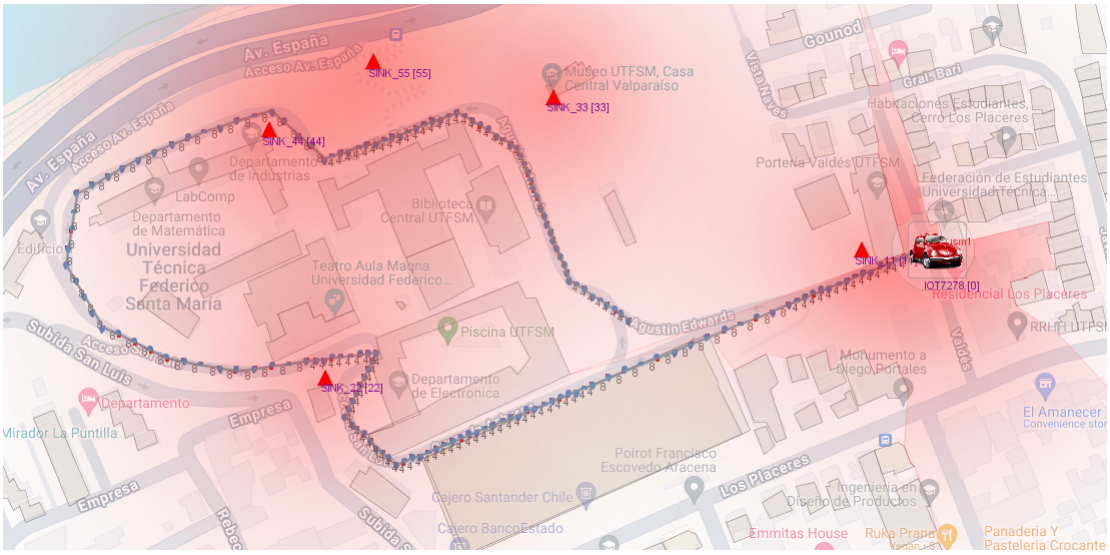


Figura 6.10: Ruta 1 seguida por nodo móvil N°1

6.1.2.3. Escenario 3

En este escenario se ocuparon cuatro nodos, donde cada uno, al mismo tiempo, seguirá una ruta distinta mostrados en las figuras 6.10, 6.11, 6.12 y 6.13. Cada ruta inicia donde corresponde cada entrada al campus. Como en el escenario anterior, la velocidad media es la misma para cada nodo y los mensajes son enviados cada 5 segundos, siguiendo el mismo formato.

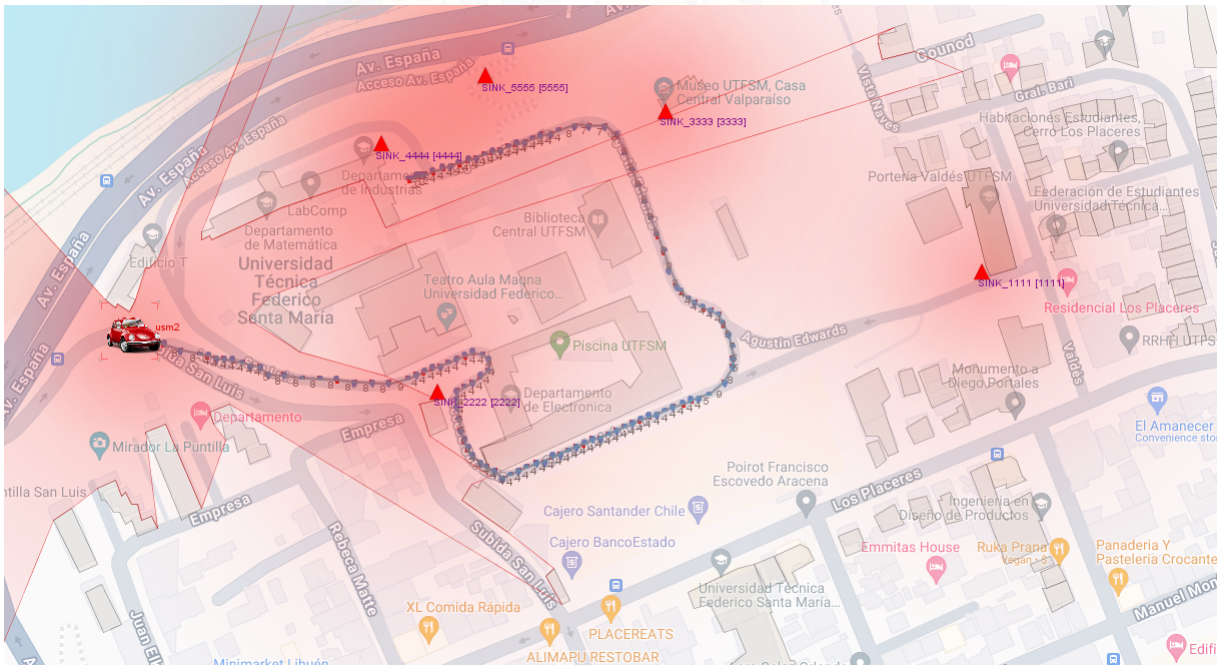


Figura 6.11: Ruta 2 seguida por nodo móvil N°2

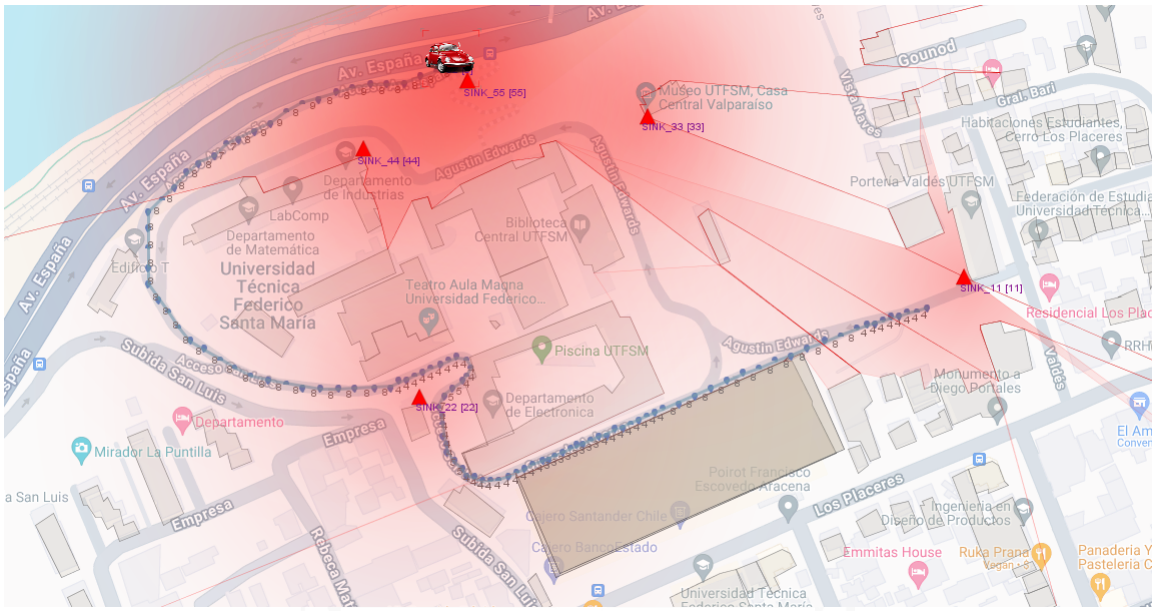


Figura 6.12: Ruta 3 seguida por nodo móvil N°3

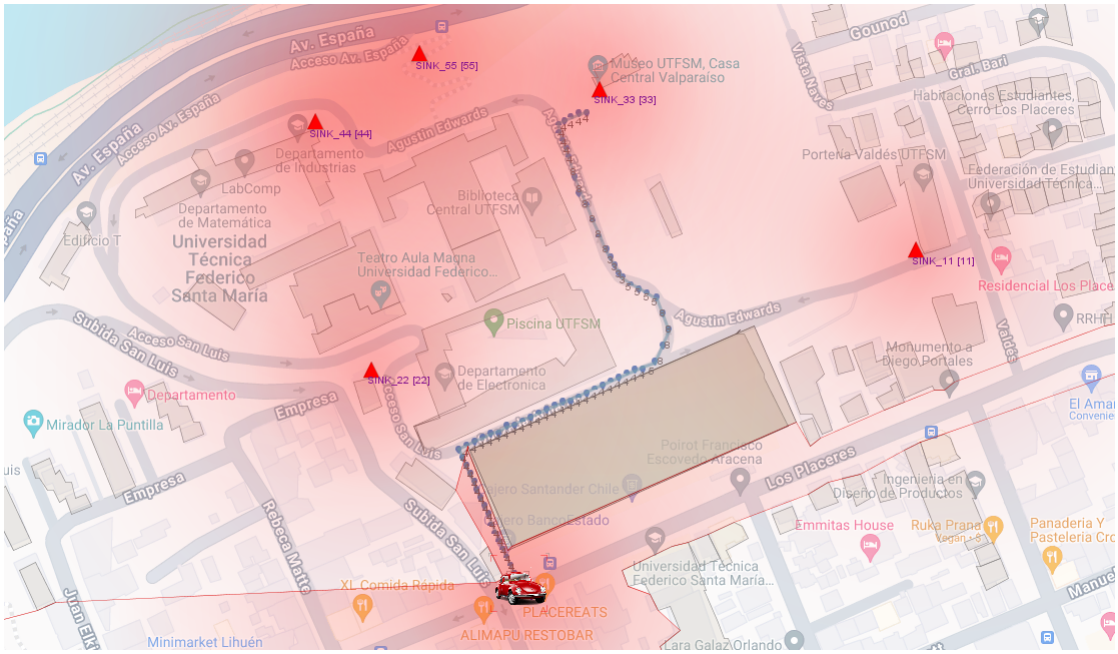


Figura 6.13: Ruta 4 seguida por nodo móvil N°4

6.2. Resultados

En esta sección se mostrará los resultados obtenidos por el desarrollo anteriormente descrito.

6.2.1. Primer objetivo:

Como se describió en la sección anterior, en este objetivo se evalúan y prueban las 18 configuraciones del capítulo 5, con la excepción que solo se puede comprobar si el mensaje fue recibido con éxito o no. Los resultados se pueden ver en la siguiente tabla:

Tabla 6.2: Resultados de paquetes recibidos según distancia para escenario 1

Configuración		3 [m]	10 [m]	20 [m]	50 [m]	100 [m]
SF=7	BW=125 [kHz]	98 %	100 %	100 %	100 %	100 %
SF=7	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=7	BW=500 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=8	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=8	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=8	BW=500 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=9	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=9	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=9	BW=500 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=10	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=10	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=10	BW=500 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=11	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=11	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=11	BW=500 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=12	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=12	BW=250 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=12	BW=500 [kHz]	99 %	100 %	100 %	100 %	100 %

Configuración		200 [m]	500 [m]	1000 [m]	2000 [m]	5000 [m]
SF=7	BW=125 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=7	BW=250 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=7	BW=500 [kHz]	100 %	100 %	100 %	0 %	0 %
SF=8	BW=125 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=8	BW=250 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=8	BW=500 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=9	BW=125 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=9	BW=250 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=9	BW=500 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=10	BW=125 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=10	BW=250 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=10	BW=500 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=11	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=11	BW=250 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=11	BW=500 [kHz]	100 %	100 %	100 %	100 %	0 %
SF=12	BW=125 [kHz]	100 %	100 %	100 %	100 %	100 %
SF=12	BW=250 [kHz]	100 %	99 %	100 %	100 %	100 %
SF=12	BW=500 [kHz]	100 %	100 %	100 %	100 %	0 %

6.2.2. Segundo objetivo:

Acá se mostrarán los resultados para los 3 escenarios implementados.

6.2.2.1. Escenario 1:

En el primer escenario considera un nodo fijo como se mostró en la figura 6.9, se enviaron mensajes durante una hora cada 5 segundos, dando un total de 720 mensajes.

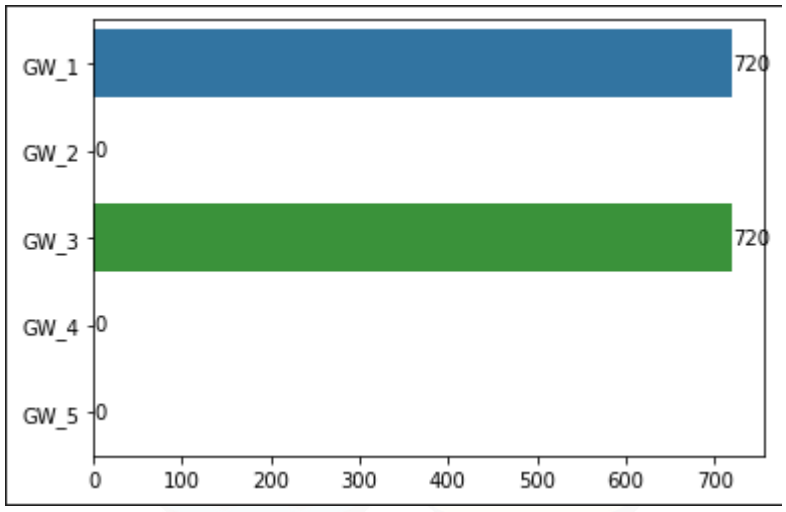


Figura 6.14: Mensajes recibidos por cada *gateway* para escenario 1

Los resultados salieron como se esperaba, solo los *gateways* que estaban a la vista del nodo, GW_1 y GW_3, fueron capaces de recibir todos los mensajes enviados. También podemos darnos cuenta que los mensajes recibidos por una puerta de enlace pueden estar duplicados en otra, esto se debe que los mensajes enviados por el nodo son en modo *broadcast*, lo que habilita que cualquiera *gateway* pueda recibirlos.

Para este escenario también se midió el consumo de energía al transmitir desde el nodo:

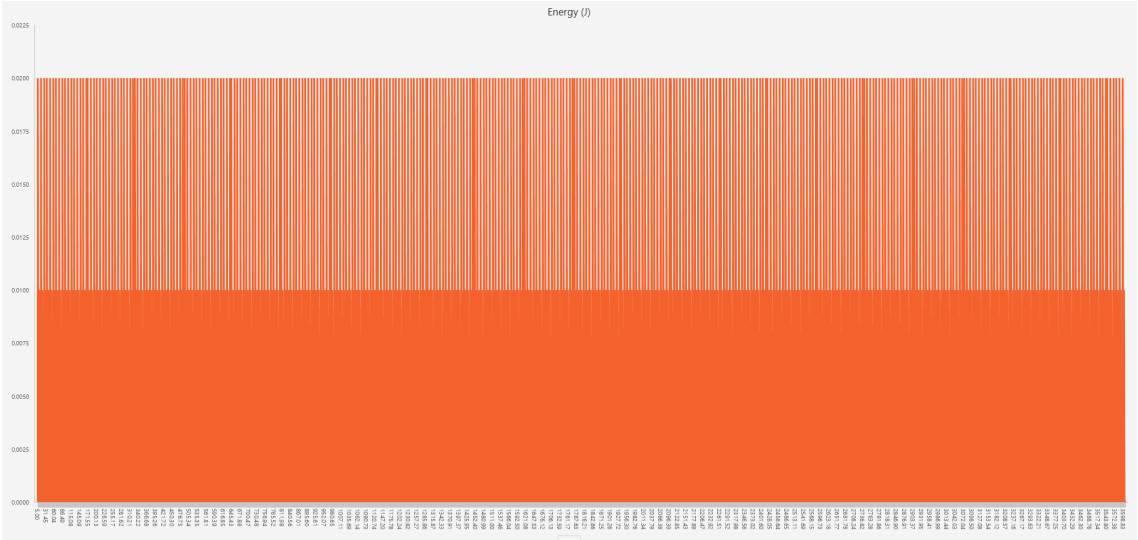


Figura 6.15: Consumo de energía del nodo estático por una hora para escenario 1

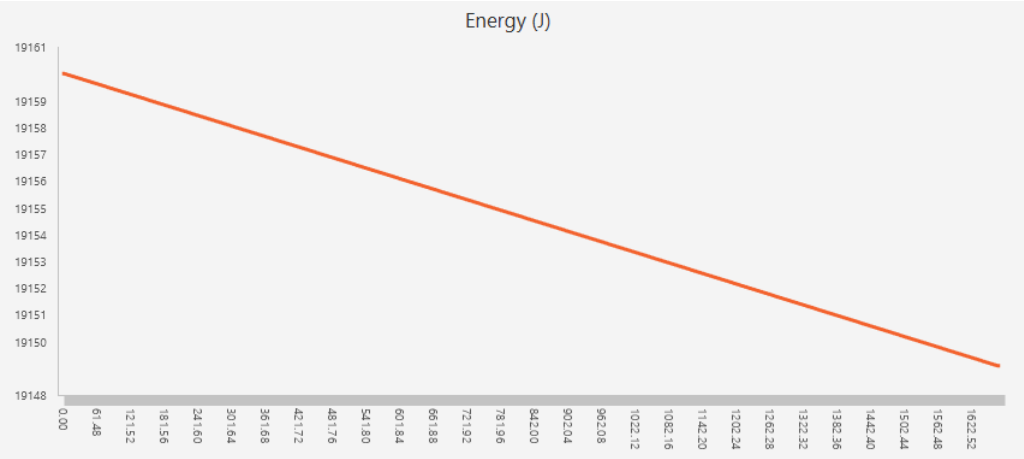


Figura 6.16: Nivel de energía almacenada en nodo estático a lo largo de una hora escenario 1

El nodo tiene un pico de consumo de 0.02 Joules por mensaje enviado como se ve en la figura **??**, esto es igual a 0.0005555556 Watt/h. También nos podemos fijar en la figura 6.16 que tuvo un consumo aproximado de 11 Joules en una hora, eso equivale a 0.00305556 Watts.

6.2.2.2. Escenario 2:

Para este escenario se utilizó un solo nodo en movimiento, donde en los siguientes gráficos podemos observar la participación de cada *gateway* en la recepción de los mensajes (Fig. 6.17). En la figura 6.18, podemos observar la cantidad de mensajes enviados por el nodo, la cantidad total de mensajes recibidos por los *gateways* y la cantidad de los mensajes no duplicados que fueron recibidos. De esta manera podemos calcular que para esta prueba hubo un 16.13 % de perdida de paquetes.

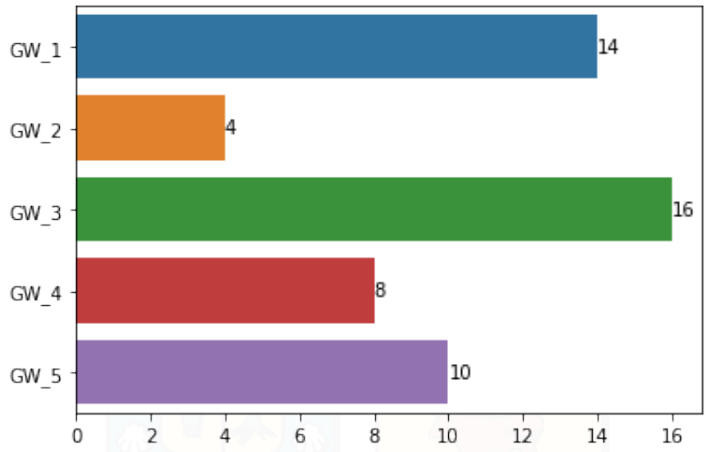


Figura 6.17: Mensajes recibidos por cada *gateway* para escenario 2

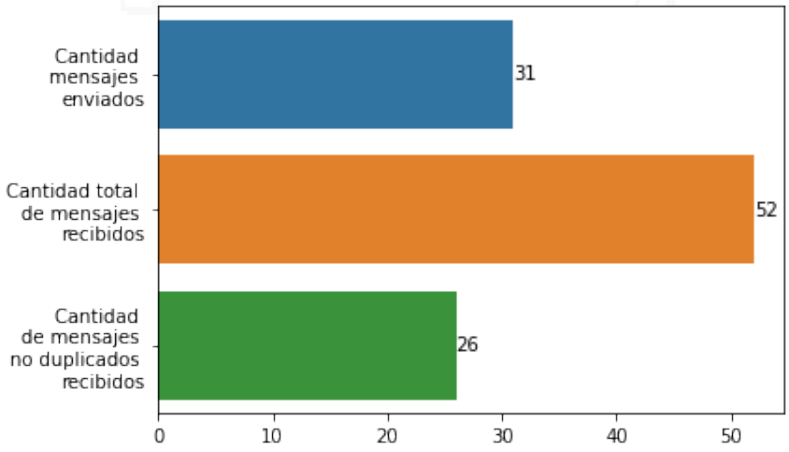


Figura 6.18: Mensajes enviados y mensajes recibidos en total para el escenario 2

6.2.2.3. Escenario 3:

Este escenario se simuló con cuatro nodos en movimiento, donde en los siguiente gráficos podemos observar la participación de cada *gateway* en la recepción de los mensajes para cada nodo en particular (Fig. 6.19). En la figura 6.20 podemos observar la cantidad de mensajes enviados, la cantidad total de mensajes recibidos y cantidad de mensajes no duplicados recibidos por los *gateways*. Con esto podemos calcular la perdida de paquete para cada nodo, lo cual nos da la tabla 6.3.

Tabla 6.3: Perdida de paquete por nodo

N° nodo	Perdida de paquete
1	16.13 %
2	12.5 %
3	13.64 %
4	0 %

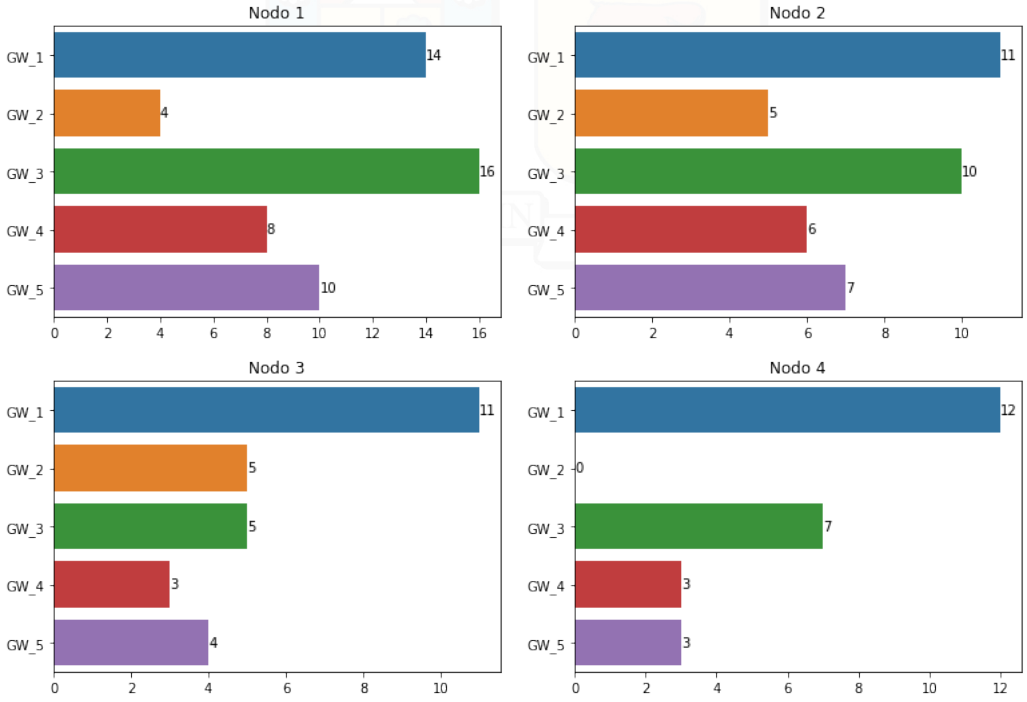


Figura 6.19: Mensajes recibidos por cada *gateway* por nodo para escenario 2

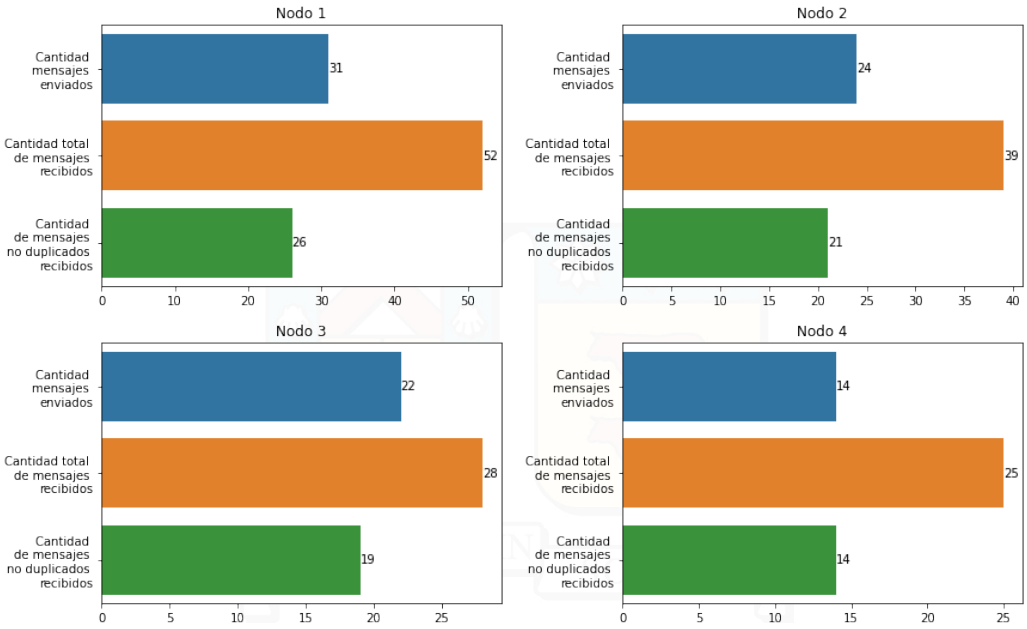


Figura 6.20: Mensajes enviados y mensajes recibidos en total por cada nodo para el escenario 3

7 | Conclusión

En relación a las decisiones tomadas, es importante destacar que los enfoques utilizados para definir la arquitectura IoT han sido fundamentales para darle dirección y certeza a la solución propuesta y desarrollada.

Respecto a la implementación, se puede concluir que la configuración SF=9 y BW=250 kHz es ligeramente mejor que su competencia. En los gráficos podemos ver que los valores de SNR y RSSI a distancias grandes tiende a ser más estable y con mayor valor. Esto concuerda con lo leído en el paper "Experimental Evaluation of LoRa in Transit Vehicle Tracking Service Based on Intelligent Transportation Systems and IoT" [30]. Según la página oficial de Sensing Labs, el valor RSSI mínimo para considerar una buena conexión es de -105 dB y para el caso de SNR un valor mínimo de 7 dBm (Fig. 7.1). Con esto podemos concluir también que para cada distancia probada, los valores de SNR y RSSI están sobre lo mínimamente necesario para una conexión estable.

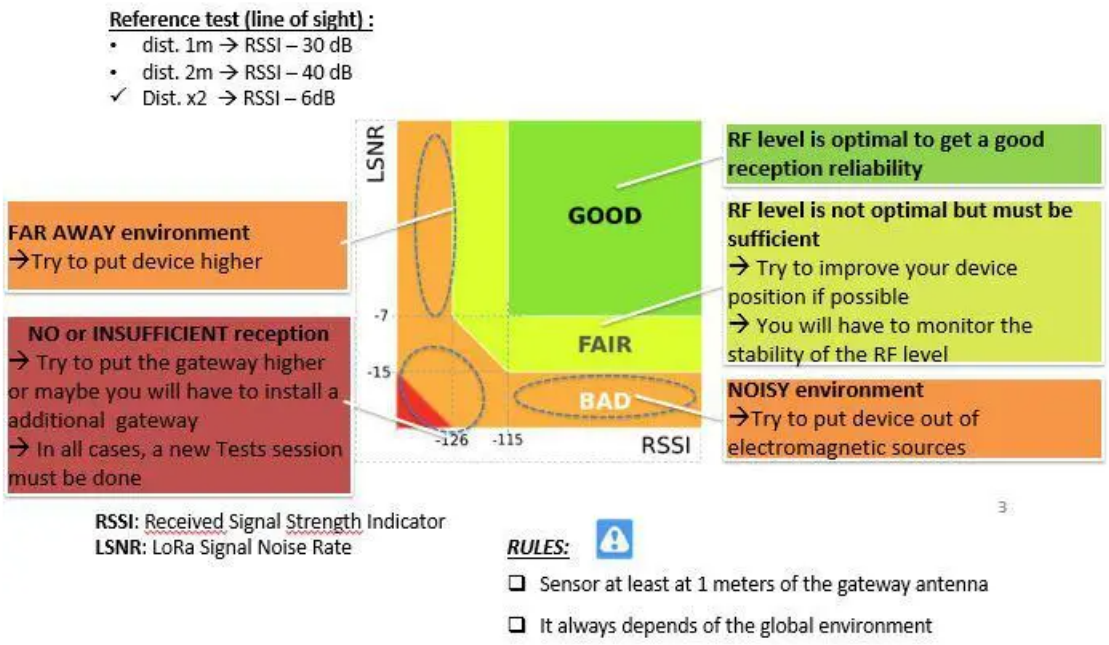


Figura 7.1: Análisis de niveles de radio

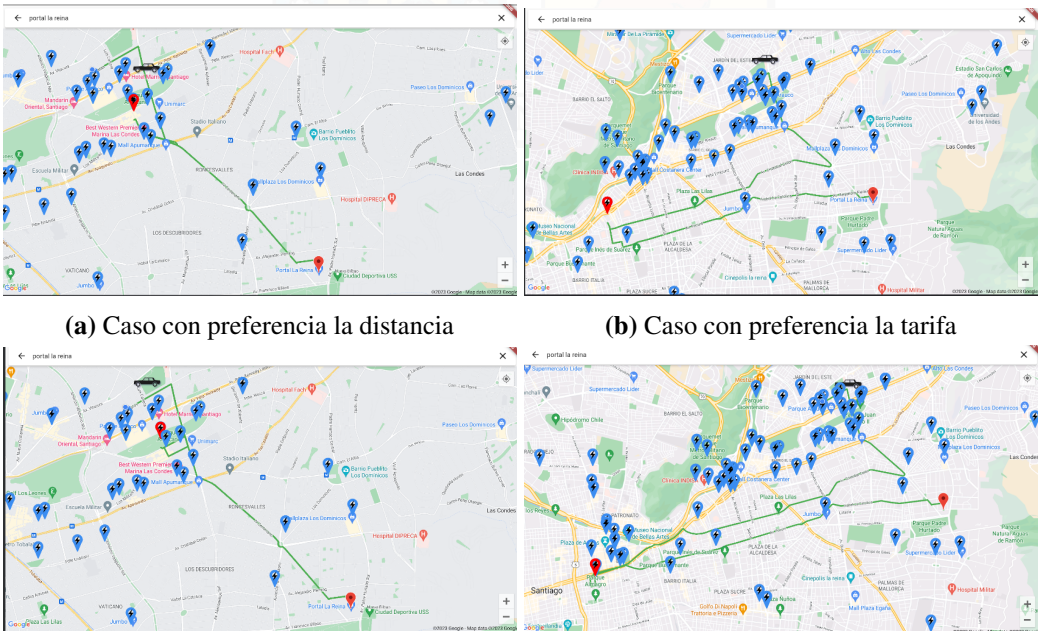
En la simulación, ya que la ruta y la prueba hecha en el escenario 3 es la misma que el nodo 1 del escenario 4, y dado que la perdida de paquetes para este nodo en ambos escenarios es la misma. Se puede concluir que la perdida de paquetes que hay no es por colisión entre paquetes, si no que por interferencia de objetos, en este caso edificios. También podemos observar que la perdida de paquetes es menor a un 20 % para cada nodo, concluyendo que, dentro del campus, la comunicación entre nodo y *gateways* es buena, ya que este no es un servicio donde necesites obtener datos de manera continua.

Viendo el consumo del nodo transmisor (vehículo) a lo largo de una hora, se ve que la transmisión no consume mucha energía, lo que para un auto eléctrico, este no sería un consumo significativo, mas bien casi nulo.

En el caso de la capa de middleware, se ha logrado crear una plataforma de navegación con un algoritmo que se adapta aún teniendo limitaciones tecnológicas y la densidad baja de electrolineras en el territorio nacional chileno. A través de diferentes configuraciones en los parámetros y una cuidadosa implementación del diagrama de flujo por medio de un algoritmo de optimización utilizando programación lineal, se han obtenido resultados que

demuestran el cumplimiento del objetivo principal.

En los casos de prueba simulados al integrar todo lo que compone el middleware con la interfaz de usuario (Fig. 7.2), se pudo evidenciar que la simulación ha proporcionado resultados claros sobre cómo el algoritmo podría comportarse en un escenario real, esto ha permitido identificar sus debilidades. Y gracias a esto, se ha logrado optimizar y corregir cualquier aspecto necesario. Esta simulación fue fundamental para mejorar la eficiencia y confiabilidad del algoritmo en situaciones reales.



(a) Caso con preferencia la distancia **(b)** Caso con preferencia la tarifa

(c) Caso con preferencia el tiempo de espera **(d)** Caso con preferencia la velocidad de carga

Figura 7.2: Casos de prueba de algoritmo simulados en la interfaz de usuario

Aún estando lejos de ser perfecto, el algoritmo de recomendación ha logrado abordar un problema que aún persiste en el país. Esto brinda esperanzas de que el proyecto avance en la dirección correcta y en el futuro se pueda presentar al público como una opción de navegación de calidad y gratuita para vehículos eléctricos. Desde la perspectiva de un ingeniero, esto es muy motivador, ya que incentiva a las personas a adquirir vehículos eléctricos y ser parte del cambio en nuestra sociedad.

Al fin y al cabo, como se pudo demostrar empíricamente a lo largo de este trabajo, es posible implementar la solución propuesta, y que los resultados obtenidos a partir de ella son

lo suficientemente adecuados como para entregarle un sistema y servicio de calidad a los usuarios de esta aplicación, asegurando que sus problemas se harán cada vez menores. Ya para terminar, se puede decir que el trabajo en su totalidad se presenta como una alternativa viable para resolver el problema de la navegación de vehículos eléctricos descrito en este trabajo.

En relación a los trabajos futuros en relación a la capa de comunicación se consideran los siguientes puntos:

- Conexiones bilaterales entre nodo y gateway.
- Optimizar rapidez de envió entre paquetes.
- Realizar más pruebas con distintos *Coding Rate*.
- Considerar conducción del vehículo dentro del monitoreo del uso de energía.
- Hacer realidad los casos de prueba simulados.
- Integrar la capa de comunicación con el resto de la plataforma.

Bibliografía

[1] Estudio de Movilidad Eléctrica en Chile, AChEE https://drive.google.com/file/d/1vtMMWX6w6vTEgDI6RIBZN5XVu131LYGY/view

[2] WazeRouteCalculator. https://github.com/kovacsbalu/WazeRouteCalculator 1.2.1

[3] Tomtom: Location technology for developer. https://developer.tomtom.com/ 1.2.1

[4] Sörensen, K. (2015), Metaheuristics—the metaphor exposed. Intl. Trans. in Op. Res., 22: 3-18. https://doi.org/10.1111/itor.12001 1.2.1

[5] Guerrero Salas, H. (2009). Programación lineal aplicada. Bogotá, Colombia: Segunda edición.https://www.ecoeediciones.com/wp-content/uploads/2017/07/Programación-lineal-2ed.pdf 1.2.1

[6] Palanisamy, S., & SuvithaVani, P. (2020). A survey on RDBMS and NoSQL Databases MySQL vs MongoDB. "2020 International Conference on Computer Communication and Informatics (ICCCI)", 1-7. https://doi.org/10.1109/ICCCI48352.2020.9104047 1.2.1

[7] Ministerio de Energía, Gobierno de Chile. (s.f.). Plataforma de Electromovilidad.https://energia.gob.cl/sites/default/files/estrategia-nacional-electromovilidad_ministerio-de-energia.pdf (document), 2.3

[8] Montoya, A. (2021, febrero 1). Electromovilidad en Chile 2020. ASOCIACIÓN GREMIAL DE VEHÍCULOS ELÉCTRICOS DE CHILE. https://www.avec.cl/wp-content/uploads/2021/03/Electromovilidad-en-Chile-2020.pdf (document), 2.1

[9] ANAC, Asociación Nacional Automotriz de Chile A.G. (2022). Informe de ventas vehículos Cero y Bajas emisiones Diciembre 2022.https://www.anac.cl/wp-content/uploads/2023/01/12-ANAC-Informe-vehiculos-cero-y-bajas-emisiones-Diciembre-2022.pdf (document), 2.2

[10] Excel Ecocarga. https://drive.google.com/file/d/1pUIMOnyTcMKMepTbDQw6ulnVlaS-FXy0/view (document), 2.4

[11] Habault, G., Montavont, N., Papadopoulos, G. Z., Ouya, A., Aragon, B., & Bouette, C. (2017). An Efficient Electric Vehicle Charging Architecture based on LoRa Communication. https://doi.org/10.1109/SmartGridComm.2017.8340723 (document), 3.1, 3.1, 3.2, 3.1

[12] Ndukwe, C. I., Iqbal, T., Liang, X., Khan, J., & Aghenta, L. (2020). LoRa-based communication system for data transfer in microgrids. “AIMS Electronics and Electrical Engineering”, 4, 303–325. https://doi.org/10.3934/ElectrEng.2020.3.303 (document), 3.1, 3.3, 3.1

[13] Haque, K. F., Abdelgawad, A., Yanambaka, V. P., & Yelamarthi, K. (2020). LoRa Architecture for V2X Communication: An Experimental Evaluation with Vehicles on the Move. “Sensors”, 20(23), 6876. https://doi.org/10.3390/s20236876 (document), 3.1, 3.4, 3.1

[14] Mwasilu, F., Justo, J., Kim, E.-K., Do, T., & Jung, J.-W. (2014). Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration. “Renewable and Sustainable Energy Reviews”, 34, 501–516.https://doi.org/10.1016/j.rser.2014.03.031 (document), 3.1, 3.5, 3.1

[15] Rinaldi, S., Pasetti, M., Sisinni, E., Bonafini, F., Ferrari, P., Rizzi, M., & Flammini, A. (2018). On the Mobile Communication Requirements for the Demand-Side Management of Electric Vehicles. “Energies”, 11(5), 1220. https://doi.org/10.3390/en11051220 (document), 3.1, 3.6, 3.1

[16] Salazar-Cabrera, R., Pachón de la Cruz, A., & Madrid Molina, J. M. (2019). Proof of Concept of an IoT-Based Public Vehicle Tracking System, Using LoRa (Long Range) and Intelligent Transportation System (ITS) Services. “Journal of Computer Networks and Communications”, 2019, 9198157. https://www.hindawi.com/journals/jcnc/2019/9198157/ (document), 3.1, 3.7, 3.1

[17] Hashmi, M. U., Alam, M. M., Ramarozatovo, O. L. V., & Alam, M. S. (2022). WEcharge: democratizing EV charging infrastructure. arXiv preprint arXiv:2204.01478. https://arxiv.org/pdf/2204.01478.pdf 3.2, 3.2

[18] Alsabbagh, A., Li, Z., & Ma, C. (2021). Electric Vehicle Charging Navigation Strategy Considering Multiple Customer Concern. “2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)”. https://doi.org/10.1109/ICPS49255.2021.9468147 (document), 3.2, 3.9, 3.2

[19] Guo, X., Liu, J., & Fan, H. (2016). Optimal Charging Navigation Strategy for Electric Vehicles. “2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)”, 222-225. https://doi.org/10.1109/ICIICII.2016.0061 (document), 3.2, 3.10, 3.11, 3.3

[20] Tan, J., & Wang, L. (2017). Real-Time Charging Navigation of Electric Vehicles to Fast Charging Stations: A Hierarchical Game Approach. “IEEE Transactions on Smart Grid”, 8(2), 846-856. https://doi.org/10.1109/TSG.2015.2458863 (document), 3.2, 3.12, 3.13, 3.3

[21] Li, X., Song, Y., Kang, C., Cheng, H., & Lian, J. (2020). Price Incentive-Based Charging Navigation Strategy for Electric Vehicles. “IEEE Transactions on Industry Applications”, 56(5), 5762-5774. https://doi.org/10.1109/TIA.2020.2981275 (document), 3.2, 3.14, 3.15, 3.4

[22] Kalyuzhny, E. R., Krasnousov, V. M., Bukreev, L. V., Shpakovsky, G. A., & Zarikovskaya, N. V. (2021). Technologies and implementation of an integration with geolocation and maps in mobile application development. “Journal of Physics: Conference Series”, 2094(3), 032029. https://doi.org/10.1088/1742-6596/2094/3/032029 (document), 3.3, 3.16, 3.5

[23] Audu, W., Okwori, M., Aliyu, S., Badmus, R., Abubakar, Z., & Abdullahi, U. (2016). Development of an Android App for Monitoring PMS in Gas Stations. https://www.researchgate.net/publication/325737158_Development_of_an_Android_App_for_Monitorin_PMS_in_Gas_Stations (document), 3.3, 3.17, 3.5

[24] Sharma, R., Sharma, P., & Sachdeva, R. (2018). Development of Android Based Real Location Tracking App. Journal of Emerging Technologies and Innovative Research. Recuperado de https://api.semanticscholar.org/CorpusID:215851091. (document), 3.3, 3.18, 3.5

[25] Sujatha, K., Nageswara Rao, P. V., Sruthi, K. J., & Arjuna Rao, A. (2014). Design and development of android mobile based bus tracking system. “2014 First International Conference on Networks & Soft Computing (ICNSC2014)”, 231–235. https://doi.org/10.1109/CNSC.2014.6906694 (document), 3.3, 3.19, 3.6

[26] Wu, W. (2018). React Native vs Flutter, Cross-platforms mobile application frameworks.ttps://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf (document), 3.3, 3.20, 3.6

[27] Dragino. (2019). LoRa GPS HAT Single Channel LoRa & GPS module User Manual.https://www.dragino.com/downloads/downloads/LoRa_GPS_HAT/LoRa_GPS_HAT_UserManual_v1.4.pdf 5.1.2.1

[28] Subsecretaría de Telecomunicaciones. (2017, octubre 6). Resolución 1985 EXENTA [Fija norma técnica de equipos de alcance reducido]. “Biblioteca del Congreso Nacional de Chile”.https://bcn.cl/3gztv

[29] FreeNwork. (s.f.). Manual de usuario de CupCarbon.https://freenwork.com/cupcarbon/cupcarbon_user_guide.pdf

[30] Jurado Murillo F, Quintero Yoshioka JS, Varela López AD, Salazar-Cabrera R, Pachón de la Cruz Á, Madrid Molina JM. Experimental Evaluation of LoRa in Transit Vehicle Tracking Service Based on Intelligent Transportation Systems and IoT. Electronics. 2020; 9(11):1950. https://doi.org/10.3390/electronics9111950 5.1, 7

[31] Cattani M, Boano CA, Römer K. An Experimental Evaluation of the Reliability of LoRa Long-Range Low-Power Wireless Communication. Journal of Sensor and Actuator Networks. 2017; 6(2):7. https://doi.org/10.3390/jsan6020007

[32] Karthigaichelvi, R., & Balakumar, B. (2023). Analysis of Coverage in LoRa using secured IoT Systems. E3S Web Conf., 399, 04001. DOI: https://doi.org/10.1051/e3sconf/202339904001

[33] SUBTEL. (2014). Oficio N° 14427 (Informe mensual de ejecución presupuestaria).https://www.subtel.gob.cl/transparencia/Perfiles/Transparencia20285/Normativas/Oficios/14oo_14427.pdf 5.1.2.1

A | Apéndice

A.1. Códigos

```
1
/*
3 * HelTec Automation(TM) LoRaWAN 1.0.2 OTAA example use
    OTAA, CLASS A
 *
5 * Function summary:
 *
7 * - use internal RTC(150KHz);
 *
9 * - Include stop mode and deep sleep mode;
 *
11 * - 15S data send cycle;
 *
13 * - Informations output via serial(115200);
 *
15 * - Only ESP32 + LoRa series boards can use this library,
    need a license
 *   to make the code run(check you license here: http://
    www.heltec.cn/search/);
17 *
```

```
 * You can change some definition in "Commissioning.h" and
   "LoRaMac-definitions.h"
 *
 * HelTec AutoMation, Chengdu, China.
 *
 * https://heltec.org
 * support@heltec.cn
 *
 *this project also release in GitHub:
 *https://github.com/HelTecAutomation/ESP32_LoRaWAN
*/

#include "LoRaWan_APP.h"
#include "Arduino.h"
#include "HT_SSD1306Wire.h"

#define RF_FREQUENCY 915000000 // Hz

#define TX_OUTPUT_POWER 22 // dBm

#define LORA_BANDWIDTH 2 // [0: 125 kHz,
                         //  1: 250 kHz,
                         //  2: 500 kHz,
                         //  3: Reserved]
#define LORA_SPREADING_FACTOR 12 // [SF7..SF12]
#define LORA_CODINGRATE 1 // [1: 4/5,
                          //  2: 4/6,
                          //  3: 4/7,
                          //  4: 4/8]
#define LORA_PREAMBLE_LENGTH 8 // Same for Tx and Rx
```

```
#define LORA_SYMBOL_TIMEOUT 0 // Symbols
#define LORA_FIX_LENGTH_PAYLOAD_ON false
#define LORA_IQ_INVERSION_ON false


#define RX_TIMEOUT_VALUE 1000
#define BUFFER_SIZE 30 // Define the payload size here

char txpacket[BUFFER_SIZE];
char rxpacket[BUFFER_SIZE];
char Repacket[BUFFER_SIZE];

static RadioEvents_t RadioEvents;
void OnTxDone( void );
void OnTxTimeout( void );
void displayMcuInit();
void displaySend();

typedef enum
{
    STATUS_LOWPOWER,
    STATUS_RX,
    STATUS_TX
}States_t;


int16_t txNumber;
States_t state;
bool sleepMode = false;
int16_t Rssi,rxSize;
```

```
77
uint32_t  license[4] = {0x4D0177F3,0x8E701F25,0x58ECC2A8,0
   x1C5E8D49}; // Heltec 1
79 extern SSD1306Wire display;

81 void setup()
{
83   displayMcuInit();

85   Serial.begin(115200);
   while (!Serial);
87   SPI.begin(SCK,MISO,MOSI,SS);
    Mcu.begin();
89


91     txNumber=0;
     Rssi=0;
93

     RadioEvents.TxDone = OnTxDone;
95     RadioEvents.TxTimeout = OnTxTimeout;

97     Radio.Init( &RadioEvents );
     Radio.SetChannel( RF_FREQUENCY );
99     Radio.SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0,
   LORA_BANDWIDTH,
                                    LORA_SPREADING_FACTOR,
   LORA_CODINGRATE,
101                                 LORA_PREAMBLE_LENGTH,
   LORA_FIX_LENGTH_PAYLOAD_ON,
```

```
                                        true, 0, 0,
   LORA_IQ_INVERSION_ON, 3000 );
103
    state=STATUS_TX;
105 }

107
void loop()
109 {
  switch(state)
111  {
    case STATUS_TX:
113      txNumber++;
        sprintf(txpacket,"%s","Message ");
115     sprintf(txpacket+strlen(txpacket),"%d",txNumber);

117     Serial.printf("\r\nSending Packet \"%s\" , length %
   d\r\n",txpacket, strlen(txpacket));

119     Radio.Send( (uint8_t *)txpacket, strlen(txpacket) )
   ;
        displaySend();
121     delay(3000);
        break;
123   case STATUS_LOWPOWER:
            LoRaWAN.sleep( CLASS_A );
125     break;
    default:
127     break;
  }
```

```
}

void OnTxDone( void )
{
  Serial.print("TX done......");
  sprintf(Repacket,"%s","");
  displaySend();
  state=STATUS_RX;
}

void OnTxTimeout( void )
{
    Radio.Sleep( );
    Serial.print("TX Timeout......");
    sprintf(Repacket,"%s","TX Timeout,Retransmission");
    displaySend();
    state=STATUS_TX;
}


void displayMcuInit()
{
  display.wakeup();
  display.init();
  delay(100);
  display.flipScreenVertically();
  display.setFont(ArialMT_Plain_10);
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.clear();
  display.drawString(18, 11, "ESP32_SENDER");
```

```
159   display.drawString(40, 31, "STARTING");


161   display.display();
      delay(100);
163   display.clear();
}
165
void displaySend()
167 {
      display.drawString(0,0,"Sending packet:");
169   display.drawString(0,11,(String)txpacket);
      display.drawString(0,25,(String)Repacket);
171   display.display();
      delay(100);
173   display.clear();
}
```

Listing A.1: Código .ino para nodo transmisor para escenarios 1 y 2 del capitulo 5

```
/*
2   This is a simple example show the Heltec.LoRa recived
     data in OLED.


4   The onboard OLED display is SSD1306 driver and I2C
     interface. In order to make the
    OLED correctly operation, you should output a high-low-
     high(1-0-1) signal by soft-
6   ware to OLED's reset pin, the low-level signal at least 5
     ms.


8   OLED pins to ESP32 GPIOs via this connecthin:
```

```
  OLED_SDA -- GPIO4
  OLED_SCL -- GPIO15
  OLED_RST -- GPIO16

  by Aaron.Lee from HelTec AutoMation, ChengDu, China

  www.heltec.cn

  This project is also available on GitHub:
  https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series
*/
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>

#include "LoRaWan_APP.h"
#include "Arduino.h"
#include "string.h"
#include "stdio.h"
#include "HT_SSD1306Wire.h"
#include "images.h"
extern SSD1306Wire display;
#define RF_FREQUENCY 915000000 // Hz

#define TX_OUTPUT_POWER 22 // dBm


#define LORA_BANDWIDTH 2 // [0: 125 kHz,
                         //  1: 250 kHz,
                         //  2: 500 kHz,
```

```
                            //  3: Reserved]
#define LORA_SPREADING_FACTOR 12 // [SF7..SF12]
#define LORA_CODINGRATE 1 // [1: 4/5,
                          //  2: 4/6,
                          //  3: 4/7,
                          //  4: 4/8]
#define LORA_PREAMBLE_LENGTH 8 // Same for Tx and Rx
#define LORA_SYMBOL_TIMEOUT 0 // Symbols
#define LORA_FIX_LENGTH_PAYLOAD_ON false
#define LORA_IQ_INVERSION_ON false


#define RX_TIMEOUT_VALUE 1000
#define BUFFER_SIZE 30 // Define the payload size here

char txpacket[BUFFER_SIZE];
char rxpacket[BUFFER_SIZE];

const char ssid[] = "SSID";  //  your network SSID (name)
const char password[] = "****";  // your network password

const char* mqtt_server = "192.168.85.234";

WiFiClient espClient;
PubSubClient client(espClient);

static RadioEvents_t RadioEvents;

int16_t txNumber, rxSize;

```

```
String  rssi;
String  snr;
char buffer[1024];
bool lora_idle = true;

void logo(){
 display.clear();
 display.drawXbm(0,5,logo_width,logo_height,logo_bits);
 display.display();
}

void LoRaData(){
 display.clear();
 display.setTextAlignment(TEXT_ALIGN_LEFT);
 display.setFont(ArialMT_Plain_10);
 display.drawString(0 , 15 , "Received "+ String(rxSize) +
   " bytes");
 display.drawStringMaxWidth(0 , 26 , 128, rxpacket);
 display.drawString(0, 0, rssi);
 display.display();
}

void setup() {
   //WIFI Kit series V1 not support Vext control
  Mcu.begin();

  display.init();
  display.flipScreenVertically();
  display.setFont(ArialMT_Plain_10);
  logo();
```

```
  delay(100);
  display.clear();

  display.drawString(0, 0, "Heltec.LoRa Initial success!");
  display.drawString(0, 10, "Wait for incoming data...");
  display.display();
  delay(100);
  //LoRa.onReceive(cbk);
  Serial.begin(115200);
  Mcu.begin();

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  RadioEvents.RxDone = OnRxDone;
  Radio.Init( &RadioEvents );
  Radio.SetChannel( RF_FREQUENCY );
  Radio.SetRxConfig( MODEM_LORA, LORA_BANDWIDTH,
   LORA_SPREADING_FACTOR,
                     LORA_CODINGRATE, 0,
   LORA_PREAMBLE_LENGTH,
                     LORA_SYMBOL_TIMEOUT,
   LORA_FIX_LENGTH_PAYLOAD_ON,
                     0, true, 0, 0, LORA_IQ_INVERSION_ON,
   true );
}

void loop() {
 delay(10);
```

```
124   if (!client.connected()) {
          reconnect();
126     }
      client.loop();
128
 if(lora_idle)
130   {
        lora_idle = false;
132     Radio.Rx(0);
      }
134   Radio.IrqProcess( );
}
136
void setup_wifi() {
138   delay(10);
      Serial.println();
140   Serial.print("Connecting to ");
      Serial.println(ssid);
142
      WiFi.begin(ssid, password);
144
      while (WiFi.status() != WL_CONNECTED) {
146     delay(500);
        Serial.print(".");
148   }

150   Serial.println("");
      Serial.println("WiFi connected");
152   Serial.println("IP address: ");
      Serial.println(WiFi.localIP());
```

```
154 }

156 void callback(char* topic, byte* message, unsigned int
      length) {
    Serial.print("Message arrived on topic: ");
158 Serial.print(topic);
    Serial.print(". Message: ");
160 String messageTemp;

162 for (int i = 0; i < length; i++) {
      Serial.print((char)message[i]);
164   messageTemp += (char)message[i];
    }
166 Serial.println();

168 // If a message is received on the topic esp32/output,
    // you check if the message is either "on" or "off".
170 // Changes the output state according to the message
    if (String(topic) == "esp32/output") {
172   Serial.print("Changing output to ");
      if(messageTemp == "on"){
174     Serial.println("on");
      }
176   else if(messageTemp == "off"){
        Serial.println("off");
178   }
    }
180 }

182 void reconnect() {
```

```
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void OnRxDone( uint8_t *payload, uint16_t size, int16_t
   rssi, int8_t snr )
{
    rssi=rssi;
    snr=snr;
    rxSize=size;
    memcpy(rxpacket, payload, size );
    rxpacket[size]='\0';
    Radio.Sleep( );
    Serial.printf("\r\nreceived packet \"%s\" with rssi %d,
    snr %d, length %d\r\n",rxpacket,rssi,snr,rxSize);
    LoRaData();
    snprintf(buffer, sizeof(buffer), "\r\n\"%s\" with rssi
   %d, snr %d",rxpacket,rssi,snr);
```

```
210     client.publish("esp32/output", buffer);
        lora_idle = true;
212 }
```

Listing A.2: Código .ino para nodo receptor para escenarios 1 y 2 del capitulo 5

```
/*
2  * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
   * Copyright (c) 2018 Terry Moore, MCCI
4  *
   * Permission is hereby granted, free of charge, to anyone
6  * obtaining a copy of this document and accompanying files
     ,
   * to do whatever they want with them without any
     restriction,
8  * including, but not limited to, copying, modification and
      redistribution.
   * NO WARRANTY OF ANY KIND IS PROVIDED.
10 *
   * This example sends a valid LoRaWAN packet with payload "
     Hello,
12 * world!", using frequency and encryption settings
     matching those of
   * the The Things Network.
14 *
   * This uses ABP (Activation-by-personalisation), where a
     DevAddr and
16 * Session keys are preconfigured (unlike OTAA, where a
     DevEUI and
   * application key is configured, while the DevAddr and
     session keys are
```

```
18  * assigned/generated in the over-the-air-activation
      procedure).
    *
20  * Note: LoRaWAN per sub-band duty-cycle limitation is
      enforced (1% in
    * g1, 0.1% in g2), but not the TTN fair usage policy (
      which is probably
22  * violated by this sketch when left running for longer)!
    *
24  * To use this sketch, first register your application and
      device with
    * the things network, to set or generate a DevAddr,
      NwkSKey and
26  * AppSKey. Each device should have their own unique values
       for these
    * fields.
28  *
    * Do not forget to define the radio type correctly in
30  * arduino-lmic/project_config/lmic_project_config.h or
      from your BOARDS.txt.
    *
32 */

34  // References:
    // [feather] adafruit-feather-m0-radio-with-lora-module.
      pdf
36

#include <lmic.h>
38 #include <hal/hal.h>
#include <SPI.h>
```

```
#include "heltec.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

static const int RXPin = RX, TXPin = TX;
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

//
// For normal use, we require that you edit the sketch to replace FILLMEIN
// with values assigned by the TTN console. However, for regression tests,
// we want to be able to compile these scripts. The regression tests define
// COMPILE_REGRESSION_TEST, and in that case we define FILLMEIN to a non-
// working but innocuous value.
//

// LoRaWAN NwkSKey, network session key
// This should be in big-endian (aka msb).
static const PROGMEM u1_t NWKSKEY[16] = {
  0x70, 0xFC, 0x6C, 0x66, 0x91, 0x34, 0x84, 0x3B,
  0xD1, 0x84, 0xAA, 0xE5, 0x2E, 0xAB, 0x44, 0xB1
  };

// LoRaWAN AppSKey, application session key
```

```
// This should also be in big-endian (aka msb).
static const u1_t PROGMEM APPSKEY[16] = {
  0xE7, 0x09, 0xFC, 0x20, 0x0A, 0x4E, 0x1C, 0x3A,
  0x25, 0x4C, 0x56, 0x20, 0x46, 0xFE, 0x13, 0xD5
  };

// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
// The library converts the address to network byte order as needed,
// so this should be in big-endian (aka msb) too.
static const u4_t DEVADDR = 0x260C8E7B ;

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in arduino-lmic/project_config/lmic_project_config.h,
// otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }
#define BUFFER_SIZE  25 // Define the payload size here

String txpacket = "";
static uint8_t mydata[BUFFER_SIZE];
static osjob_t sendjob;
int16_t txNumber;
float latitude = 0.0;
```

```
92 float longitude = 0.0;

94 // Schedule TX every this many seconds
   // (might become longer due to duty cycle limitations).
96 const unsigned TX_INTERVAL = 10;

98 // Pin mapping
   const lmic_pinmap lmic_pins = {
100     .nss = 18,
        .rxtx = LMIC_UNUSED_PIN,
102     .rst = 14,
        .dio = {26, 34, 35},
104 };

106 void onEvent (ev_t ev) {
        switch(ev) {
108         case EV_TXCOMPLETE:
                Serial.println(F("EV_TXCOMPLETE (includes
    waiting for RX windows)"));
110             if (LMIC.txrxFlags & TXRX_ACK)
                  Serial.println(F("Received ack"));
112             if (LMIC.dataLen) {
                  Serial.println(F("Received "));
114               Serial.println(LMIC.dataLen);
                  Serial.println(F(" bytes of payload"));
116             }
                // Schedule next transmission
118             os_setTimedCallback(&sendjob,
                                    os_getTime()+sec2osticks(
    TX_INTERVAL),
```

```
                                    do_send);
            break;
        case EV_TXSTART:
            Serial.println(F("EV_TXSTART"));
            break;
        default:
            Serial.print(F("Unknown event: "));
            Serial.println((unsigned) ev);
            break;
    }
}

void do_send(osjob_t* j){

    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next
   possible time.
        txpacket = "";
        latitude = gps.location.lat();
        longitude = gps.location.lng();
        txpacket.concat(String(gps.location.lat(),6));
        txpacket.concat(",");
        txpacket.concat(String(gps.location.lng(),6));
        txpacket.concat(",");
        txpacket.concat(String(txNumber));
        txpacket.getBytes(mydata, txpacket.length() + 1  );
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
```

```
        Serial.printf("\r\nPacket %d queued\n",txNumber);
        Serial.printf("\r\nMessage: %s\n",mydata);

        Heltec.display->clear();
        Heltec.display->setFont(ArialMT_Plain_16);
        Heltec.display->drawString(0, 0, "Sending Packet:");
        Heltec.display->drawString(50, 25, String(txNumber));
        Heltec.display->setFont(ArialMT_Plain_16);
        Heltec.display->display();
        txNumber++;
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    while (!Serial); // wait for Serial to be initialized
    Serial.begin(115200);
    delay(100);     // per sample code on RF_95 test
    Serial.println(F("Starting"));

    txNumber=0;

    Heltec.display->init();
    Heltec.display->setContrast(255);
    Heltec.display->flipScreenVertically();
    Heltec.display->clear();
    Heltec.display->setFont(ArialMT_Plain_24);
    Heltec.display->drawString(23, 15, "Starting");
```

```
    Heltec.display->display();
178 delay(1500);

180 ss.begin(GPSBaud);

182 // LMIC init
    os_init();
184 // Reset the MAC state.
    // Session and pending data transfers will be discarded
    .
186 LMIC_reset();

188 // Set static session parameters.
    // Instead of dynamically establishing a session
190 // by joining the network,
    // precomputed session parameters are be provided.
192 #ifdef PROGMEM
    // On AVR, these values are stored in flash and only
    copied to RAM
194 // once. Copy them to a temporary buffer here,
    LMIC_setSession will
    // copy them into a buffer of its own again.
196 uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
198 memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
200 LMIC_setSession (0x13, DEVADDR, nwkskey, appskey);
    #else
202 // If not running an AVR with PROGMEM, just use the
    arrays directly
```

```
    LMIC_setSession (0x13, DEVADDR, NWKSKEY, APPSKEY);
204 #endif

206 #if defined(CFG_eu868)
    LMIC_setupChannel(0,
208                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7),
210                   BAND_CENTI);      // g-band
    LMIC_setupChannel(1,
212                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7B),
214                   BAND_CENTI);      // g-band
    LMIC_setupChannel(2,
216                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7),
218                   BAND_CENTI);      // g-band
    LMIC_setupChannel(3,
220                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7),
222                   BAND_CENTI);      // g-band
    LMIC_setupChannel(4,
224                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7),
226                   BAND_CENTI);      // g-band
    LMIC_setupChannel(5,
228                   917100000,
                      DR_RANGE_MAP(DR_SF12, DR_SF7),
230                   BAND_CENTI);      // g-band
    LMIC_setupChannel(6,
232                   917100000,
```

```
                        DR_RANGE_MAP(DR_SF12, DR_SF7),
234                     BAND_CENTI);      // g-band
    LMIC_setupChannel(7,
236                     917100000,
                        DR_RANGE_MAP(DR_SF12, DR_SF7),
238                     BAND_CENTI);      // g-band
    LMIC_setupChannel(8,
240                     917100000,
                        DR_RANGE_MAP(DR_FSK,  DR_FSK),
242                     BAND_MILLI);      // g2-band
    #endif
244
    // Disable link check validation
246 LMIC_setLinkCheckMode(0);

248 // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9
250
    // Set data rate and transmit power for uplink
252 LMIC_setDrTxpow(DR_SF9,22);

254 // Start job
    do_send(&sendjob);
256 }

258 void loop() {

260   if (ss.available() > 0){
      if (gps.encode(ss.read())){
262     if (gps.location.isValid()){
```

```
        os_runloop_once();
      }
      else {
        Heltec.display->clear();
        Heltec.display->setFont(ArialMT_Plain_16);
        Heltec.display->drawString(0, 20, "Seaching GPS..."
  );
        Heltec.display->display();
      }
    }
  }

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Heltec.display->clear();
    Serial.println(F("No GPS detected: check wiring."));
    Heltec.display->setFont(ArialMT_Plain_16);
    Heltec.display->drawString(0, 20, "No GPS detected");
    Heltec.display->display();
  }
}
```

Listing A.3: Código .ino para nodo transmisor para escenarios 3 y 4 del capitulo 5

```
/*
* Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
*
* Permission is hereby granted, free of charge, to anyone
* obtaining a copy of this document and accompanying files
  ,
* to do whatever they want with them without any
```

```
 restriction,
* including, but not limited to, copying, modification and
   redistribution.
* NO WARRANTY OF ANY KIND IS PROVIDED.
*
* This example sends a valid LoRaWAN packet with payload "
 Hello,
* world!", using frequency and encryption settings
 matching those of
* the The Things Network.
*
* This uses OTAA (Over-the-air activation), where where a
 DevEUI and
* application key is configured, which are used in an over
 -the-air
* activation procedure where a DevAddr and session keys
 are
* assigned/generated for use with all further
 communication.
*
* Note: LoRaWAN per sub-band duty-cycle limitation is
 enforced (1% in
* g1, 0.1% in g2), but not the TTN fair usage policy (
 which is probably
* violated by this sketch when left running for longer)!

* To use this sketch, first register your application and
 device with
* the things network, to set or generate an AppEUI, DevEUI
   and AppKey.
```

```
 * Multiple devices can use the same AppEUI, but each
   device has its own
 * DevEUI and AppKey.
 *
 * Do not forget to define the radio type correctly in
   config.h.
 *
*/

#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include "heltec.h"

static const int RXPin = RX, TXPin = TX;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

// This EUI must be in little-endian format, so least-
   significant-byte
// first. When copying an EUI from ttnctl output, this
   means to reverse
// the bytes. For TTN issued EUIs the last bytes should be
```

```
   0xD5, 0xB3,
// 0x70.
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0
   x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={ 0xE0, 0x07, 0xD0, 0
   x60, 0x00, 0x1F, 0x70, 0x05 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is
    not really a
// number but a block of memory, endianness does not really
    apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] = { 0x0D, 0x00, 0xD7,
   0x02, 0x30, 0xF7, 0x03, 0xF0, 0x5C, 0x08, 0xC0, 0xA4, 0
   x0E, 0x90, 0x54, 0x01 };
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16)
   ;}


static uint8_t mydata[] = "-00.000000,-00.000000,000";
static osjob_t sendjob;
float latitude = 0.0;
float longitude = 0.0;
String latString;
String longString;

```

```
String resultado = "";
int count = 0;
String countString = "000";

float glat = -33.02757338952090;
float glon = -71.55951529741289;
float x1;
float x2;
float a;
int R = 6371;
float d;
// Schedule TX every this many seconds (might become longer
    due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 10;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 34, 35},
};

void onEvent(ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
```

```
102             break;
        case EV_BEACON_FOUND:
104             Serial.println(F("EV_BEACON_FOUND"));
                break;
106     case EV_BEACON_MISSED:
                Serial.println(F("EV_BEACON_MISSED"));
108             break;
        case EV_BEACON_TRACKED:
110             Serial.println(F("EV_BEACON_TRACKED"));
                break;
112     case EV_JOINING:
                Serial.println(F("EV_JOINING"));
114             break;
        case EV_JOINED:
116             Serial.println(F("EV_JOINED"));

118             // Disable link check validation (automatically
    enabled
                // during join, but not supported by TTN at
    this time).
120             LMIC_setLinkCheckMode(0);
                break;
122     case EV_RFU1:
                Serial.println(F("EV_RFU1"));
124             break;
        case EV_JOIN_FAILED:
126             Serial.println(F("EV_JOIN_FAILED"));
                break;
128     case EV_REJOIN_FAILED:
                Serial.println(F("EV_REJOIN_FAILED"));
```

```
130             break;
                break;
132         case EV_TXCOMPLETE:
                Serial.println(F("EV_TXCOMPLETE (includes
    waiting for RX windows)"));
134
                if (LMIC.txrxFlags & TXRX_ACK)
136               Serial.println(F("Received ack"));
                if (LMIC.dataLen) {
138               Serial.println(F("Received "));
                  Serial.println(LMIC.dataLen);
140               Serial.println(F(" bytes of payload"));
                }
142             // Schedule next transmission
                os_setTimedCallback(&sendjob,
144                                 os_getTime()+sec2osticks(
    TX_INTERVAL)
                                    , do_send
146                                 );
                break;
148         case EV_LOST_TSYNC:
                Serial.println(F("EV_LOST_TSYNC"));
150             break;
            case EV_RESET:
152             Serial.println(F("EV_RESET"));
                break;
154         case EV_RXCOMPLETE:
                // data received in ping slot
156             Serial.println(F("EV_RXCOMPLETE"));
                break;
```

```
158         case EV_LINK_DEAD:
                Serial.println(F("EV_LINK_DEAD"));
160             break;
            case EV_LINK_ALIVE:
162             Serial.println(F("EV_LINK_ALIVE"));
                break;
164          default:
                Serial.println(F("Unknown event"));
166             break;
        }
168 }

170 void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
172     const uint8_t* lastMyData;
    if (LMIC.opmode & OP_TXRXPEND) {
174         Serial.println(F("OP_TXRXPEND, not sending"));
    } else if(lastMyData != mydata){
176         lastMyData = mydata;
            // Prepare upstream data transmission at the next
   possible time.
178         count += 1;
            if (count<10){
180           countString = "00";
              countString.concat(String(count));
182         }else if (count<100){
              countString = "0";
184           countString.concat(String(count));
            }else if (count<1000){
186           countString = "";
```

```
            countString.concat(String(count));
188         }

190         LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
            Serial.print("DATA: ");
192         Serial.println(F(mydata));
            Serial.println(F("Packet queued"));
194
    }
196 // Next TX is scheduled after TX_COMPLETE event.
}
198
void displayInfo(uint8_t* mydata)
200 {

202   if (gps.location.isValid())
  {
204     latitude = gps.location.lat();
      longitude = gps.location.lng();
206   }
  else
208   {
      latitude = 0.0;
210     longitude = 0.0;
  }
212
  x1 = glat-latitude;
214   x2 = glon-longitude;
  a = sin(
216       ((x1 * 71) / 4068)/2) * sin(((x1 * 71) / 4068)/2)
```

```
      + cos((latitude*71) / 4068) * cos((glat*71) / 4068)
      * sin(((x2 *71) / 4068)/2) * sin(((x2 *71) / 4068)/2)
  ;
  d = R * 2 * atan2(sqrt(a), sqrt(1-a)) * 1000; // en
   Metros

  resultado = "";
  resultado.getBytes(mydata, 0);
    if (latitude == 0.0){
      resultado.concat("-00.000000");
      resultado.concat(",");
    }else {
      latString = String(latitude, 6);
      resultado.concat(latString);
      resultado.concat(",");
    }
    if (longitude == 0.0){
      resultado.concat("-00.000000");
      resultado.concat(",");
    }else {
      longString = String(longitude, 6);
      resultado.concat(longString);
      resultado.concat(",");
    }
  resultado.concat(countString);
  resultado.concat(",");
  resultado.getBytes(mydata, resultado.length() + 1  );
}

String getValue(String data, char separator, int index)
```

```
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;

  for(int i=0; i<=maxIndex && found<=index; i++){
    if(data.charAt(i)==separator || i==maxIndex){
        found++;
        strIndex[0] = strIndex[1]+1;
        strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }

  return found>index ? data.substring(strIndex[0], strIndex
   [1]) : "";
}


void setup() {

    Serial.begin(115200);
    Heltec.begin(true /*DisplayEnable Enable*/,
                 false /*LoRa disabled*/,
                 true /*Serial Enable*/);
    Serial.println(F("Starting"));
    Heltec.display->init();
    Heltec.display->setContrast(255);
    Heltec.display->flipScreenVertically();
    Heltec.display->clear();
    Heltec.display->setFont(ArialMT_Plain_24);
```

```
274     Heltec.display->drawString(23, 15, "Starting");
        Heltec.display->display();
276     delay(1500);

278     ss.begin(GPSBaud);

280     #ifdef VCC_ENABLE
        // For Pinoccio Scout boards
282     pinMode(VCC_ENABLE, OUTPUT);
        digitalWrite(VCC_ENABLE, HIGH);
284     delay(1000);
        #endif
286
        // LMIC init
288     os_init();
        // Reset the MAC state. Session and pending data
    transfers will be discarded.
290     LMIC_reset();

292     // Start job (sending automatically starts OTAA too)
        LMIC_setClockError (MAX_CLOCK_ERROR * 10/100);
294     do_send(&sendjob);
}
296
void loop() {
298   const uint8_t* lastMyData;

300   if (ss.available() > 0 && bateria > 0){
        if (gps.encode(ss.read())){
302       displayInfo(mydata);
```

```
      if (lastMyData != mydata) {
        lastMyData = mydata;
        Heltec.display->clear();
        Heltec.display->setFont(ArialMT_Plain_10);
        Heltec.display->drawString(0, 0, "Sending Packet:")
  ;
        Heltec.display->drawString(90, 0, countString);
        Heltec.display->setFont(ArialMT_Plain_16);
        Heltec.display->drawString(35, 20, String((int)
  round(d)));
        os_runloop_once();
      }
    }
  }

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Heltec.display->clear();
    Serial.println(F("No GPS detected: check wiring."));
    Heltec.display->setFont(ArialMT_Plain_16);
    Heltec.display->drawString(0, 20, "No GPS detected");
  }

  Heltec.display->display();


}
```

Listing A.4: Código .ino para nodo con GPS transmisor a gateway Raspberry Pi en TTN