UNIVERSIDAD TECNOLÓGICA METROPOLITANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRICIDAD
ESCUELA DE ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE LOCALIZACIÓN Y MONITOREO GEORREFERENCIAL EN TIEMPO REAL BASADO EN RED MESH LORA

TRABAJO DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN ELECTRÓNICA

AUTOR:
MELCHIORI PINTO, CHARLIE JOST

PROFESOR GUÍA:
DURNEY WASAFF, HUGO IVÁN

SANTIAGO - CHILE
2023

Agradecimientos

En este momento crucial, deseo expresar mi profundo agradecimiento a todas las personas e instituciones por su contribución al éxito de este trabajo de título. Su apoyo constante ha sido vital en cada etapa.

En primer lugar, este trabajo se enmarca dentro del proyecto FIC-R (Fondo de Innovación para la Competitividad Regional) bajo el título *Mejoramiento de la experiencia turística mediante un sistema físico-digital en áreas silvestres protegidas de la Región Metropolitana*, con el código BIP 40026822-0. Este proyecto es el resultado de una colaboración valiosa entre dos instituciones clave: la Universidad Tecnológica Metropolitana (UTEM) y la Corporación Nacional Forestal (CONAF). Mi más sincero agradecimiento a ambas instituciones por proporcionar el entorno y los recursos necesarios para que este proyecto cobrara vida.

Un especial agradecimiento a la UTEM por su apoyo fundamental y recursos. Agradezco a los programas de la VTTE (Vicerrectoría de Transferencia Tecnológica y Extensión), especialmente a PERNC (Programa de Energías Renovables y No Convencionales para el Desarrollo Sustentable) y ProteinLab (Programa de Prospectiva e Innovación Tecnológica), por su respaldo constante en espacio y apoyo profesional.
Mención especial a mi Profesor Guía, Hugo Durney Wasaff, por su orientación y apoyo continuo en este viaje académico. Reconozco la valiosa contribución de los profesores Héctor Torres y Marfilda Sandoval, en el marco del Proyecto FIC.

Quiero expresar mi aprecio y gratitud a mi familia por su inquebrantable apoyo y comprensión a lo largo de este arduo camino. Su constante aliento y paciencia han sido la base que me ha permitido alcanzar este logro.

Por último, rindo homenaje a los amigos que compartieron este viaje universitario. Aunque ya no están con nosotros, sus vivencias son invaluables.

En resumen, este trabajo de título es el resultado del esfuerzo colectivo y el apoyo de numerosas personas e instituciones. Mi más sincero agradecimiento a cada uno de ustedes por su contribución al éxito de este proyecto.

Autorización para la reproducción del trabajo de titulación

1. Identificación del trabajo de titulación

   Nombre del alumno: Charlie Jost Melchiori Pinto

   E-mail: charlie.melchiorip@utem.cl

   Título de la tesis: Desarrollo de un prototipo de sistema de localización y monitoreo georreferencial en tiempo real basado en red mesh LoRa

   Escuela: Escuela de Electrónica

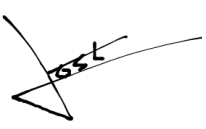
   Carrera: Ingeniería Civil Electrónica

   Título al que opta: Ingeniero Civil en Electrónica

2. Autorización de reproducción

   Se autoriza la reproducción total o parcial de este trabajo de titulación, con fines académicos por cualquier medio o procedimiento, incluyendo la cita bibliográfica que acredita al trabajo y a su autor.
   En consideración a lo anterior, se autoriza su reproducción de forma

X	Inmediata
.....	A partir de la siguiente fecha .....(mes/año)

Fecha: 23-AGO-2023 Firma: 

Esta autorización se otorga en el marco de la ley N°17.336 sobre propiedad intelectual, con carácter gratuito y no exclusivo para la institución.

NOTA OBTENIDA: _____

_____

Firma y timbre autoridad responsable

Índice de Contenido

Índice de figuras

Página

Índice de tablas

Página

Resumen

El presente trabajo de titulación aborda el diseño e implementación de un prototipo altamente escalable de un sistema de localización y monitoreo dinámico georreferenciado basado en una red mesh LoRa, para el seguimiento de personas u objetivos que pueden estar en puntos fijos o desplazándose a velocidad normal de caminata dentro de un área abierta, relativamente extensa, donde no se cuente con alternativas de comunicaciones móviles masivas convencionales. El objetivo principal es mejorar la eficiencia y seguridad en diversos entornos mediante la transmisión en tiempo real de datos de ubicación. La metodología incluyó la selección de tecnologías apropiadas (hardware y software), pruebas en entorno controlado de laboratorio, desarrollo de sistemas escalables y una interfaz de visualización para una interacción intuitiva por parte del usuario final. En este sentido, se consideraron diversos aspectos técnicos relacionados con la tecnología y criterios de diseño relacionados con las condiciones particulares del escenario experimental de aplicación. Las pruebas en terreno consideraron un entorno desafiante, de gran interés para las comunicaciones inalámbricas (relieves, vegetación, condiciones climáticas, etc.), y demostraron la viabilidad y robustez del sistema, donde se adoptó un enfoque modular para garantizar la escalabilidad. Se destacan, además, medidas de ciberseguridad implementadas para proteger la integridad y privacidad de los datos. Este trabajo culminó en un prototipo funcional con potencial para mejoras futuras.

Palabras clave: *IoT, redes mesh, LoRa, Meshtastic, tracking GNSS*

Abstract

The present thesis work addresses the design and implementation of a highly scalable prototype for a georeferenced dynamic localization and monitoring system based on a LoRa mesh network. This system is intended for tracking individuals or targets that might be stationary or moving at a normal walking speed within a relatively extensive open area where conventional mobile communication alternatives are not available. The primary objective is to enhance efficiency and safety in various environments by enabling real-time transmission of location data. The methodology involved selecting appropriate technologies (both hardware and software), conducting laboratory tests for the first controlled environment testing stage, developing a scalable model for the systems, and creating an intuitive visualization interface to improve final-user interaction. Various technical aspects related to the technology were taken into account. Field tests, carried out in challenging environments, demonstrated the system's feasibility. A modular approach was adopted to ensure scalability. Additionally, the implemented security measures to safeguard data integrity and privacy are highlighted. This work concludes with a functional prototype that holds potential for future enhancements.

Keywords: *IoT, mesh networks, LoRa, Meshtastic, tracking GNSS*

Capítulo 1

Introducción

En la actualidad, la conectividad se ha vuelto esencial tanto para los dispositivos como para las personas, permitiendo la comunicación en diversos ámbitos. Sin embargo, existen sectores y entornos críticos en los cuales aún no es posible establecer una comunicación efectiva y continua. Estos entornos incluyen áreas rurales, zonas remotas o lugares con cobertura celular limitada o inexistente.

Dentro de este ámbito, es necesario generar redes fuera de los estándares convencionales, que puedan brindar soluciones alternativas y resilientes de comunicación. Para estos casos, integrar tecnologías como LoRa (Long Range) que generen redes mesh, junto con el uso de software libre se presenta como una alternativa prometedora y de bajo costo para abordar esta necesidad de conectividad en entornos desafiados por la falta de cobertura convencional.

La tecnología LoRa, es ampliamente usada en el ámbito del Internet de las Cosas o Internet of Things[1] [2] [3], IoT para abreviar, debido a su fiabilidad para la transmisión de cadenas cortas de datos en largas distancias [4]; sumado al uso de la topología mesh que permite la ampliación de la cobertura utilizando los mismos dispositivos como estaciones repetidoras móviles distribuidas [5].

Es precisamente en sectores carentes de cobertura por parte de operadores masivos de comunicaciones, que un teléfono móvil que es actualmente un elemento esencial para la comunicación, se vuelve ineficiente, ya que no permite cumplir su función principal.

La información georeferencial y trackeo (también referido como tracking, rastreo o seguimiento) en tiempo real de personas en diversos entornos outdoor puede

dar origen a sistemas complejos y de alto costo de despliegue y mantenimiento operacional (CAPEX y OPEX) para potenciar la toma de decisiones, optimizar cadenas logísticas, efectuar análisis de comportamiento y, algo muy relevante , contribuir a la búsqueda, localización y rescate de personas en situaciones de emergencia.

En la última década el desarrollo de tecnologías y aplicaciones IoT ha dado lugar al surgimiento de soluciones para muy diversos propósitos [6] [7] basadas en recursos de hardware cada vez mas accesibles y redes inalámbricas, de muy similares características, con alta compatibilidad e interoperabilidad.

En base a lo mencionado previamente, el propósito central de este trabajo de titulación es la creación de un prototipo de sistema destinado a habilitar comunicación inalámbrica y seguimiento georeferenciado en áreas con cobertura limitada o inexistente por parte de los operadores masivos de infraestructura basada en tecnologías comerciales actuales como 4G LTE y 5G. Este objetivo se logrará mediante la implementación de una red mesh LoRa, lo que permitirá enfrentar el desafío de lograr ubicación en tiempo *real*. Todo este proceso de desarrollo se enmarca en el contexto de un proyecto FIC-R (Fondo de Innovación para la Competitividad Regional) financiado por el Gobierno Regional (GORE) Metropolitano, proyecto que une a la Corporación Nacional Forestal (CONAF) y a la Universidad Tecnológica Metropolitana (UTEM), incorporando al menos a cuatro de sus Programas de I+D+i (Programa de Energías Renovables no Convencionales - PERNC, Programa de Prospectiva e Innovación Tecnológica - ProteinLab, Programa de Ciencia de Datos, Innovación y Tecnología - PRODIT y Programa de Competitividad Turística - PCT) lo cual dió origen a un trabajo interdisciplinario que aportó mayor interés y complejidad al proyecto. De este modo, el marco de este trabajo es una colaboración interdisciplinaria e interinstitucional UTEM-CONAF-GORE, destinada a abordar diversas problemáticas a través de la innovación tecnológica aplicada a un escenario abierto y de escala real, constantemente visitado por personas que practican el senderismo, como es la Reserva Nacional Río Clarillo[1] ubicada en a comuna de Pirque, Región Metropolitana de Santiago (RM), Chile.

Siguiendo la metodología de desarrollo del proyecto mismo, el presente trabajo de título ha sido estructurado de acuerdo al siguiente orden lógico:

[1] https://www.conaf.cl/parques/parque-nacional-rio-clarillo/

CAPÍTULO 2. Estado del arte y técnico

Donde se abordan los conocimientos teóricos y se describirán los fundamentos para el uso de LoRa como tecnología. También se describirán otros proyectos similares o soluciones comerciales. Además, se enuncian las características de hardware y software utilizados para la solución del sistema propuesto.

CAPÍTULO 3. Pruebas de laboratorio

Se indican las pruebas tanto para el hardware como para el software seleccionado. Adicionalmente, se evalúan los métodos de adquisición de datos disponibles y se describe el diseño de un primer prototipo para la validación de la comunicación punto a punto.

CAPÍTULO 4. Diseño e Infraestructura de red

Se describen la estructura y características de la infraestructura de red que se conforma, así como los diferentes componentes contemplados en ella y su interoperabilidad.

CAPÍTULO 5. Desarrollo de prototipo del sistema

Se detalla el prototipado de los terminales y el sistema en general, indicando los diferentes requisitos para cada uno de los componentes hardware del sistema. También se describen las herramientas de software y servicios necesarios para el funcionamiento del prototipo y su correcto despliegue.

CAPÍTULO 6. Interfaz de Usuario

Se describen los pasos necesarios para montar una interfaz de usuario utilizando la herramienta de código abierto Grafana. Además, se describen los diferentes componentes que permiten presentar la información adquirida durante el desarrollo del sistema al usuario.

CAPÍTULO 7. Pruebas en terreno

Se describen las instalaciones donde se lleva a cabo la implementación a escala real del prototipo, junto con las consideraciones específicas para el terreno. También se evalúan los resultados de esta implementación.

1.1 Objetivos

Objetivo General

Desarrollar un prototipo de sistema de localización y monitoreo basado en una red mesh LoRa para el seguimiento georeferencial de personas, el cual permitirá la transmisión eficiente de datos de ubicación en tiempo real, mejorando aspectos de seguridad y eficiencia en diversos entornos.

Objetivos Específicos

*OE*1. Diseñar la arquitectura del sistema de localización y monitoreo basado en la red mesh LoRa, considerando los requisitos técnicos y funcionales necesarios para garantizar una transmisión confiable de datos de ubicación en tiempo real.

*OE*2. Implementar una infraestructura LoRa eficiente y escalable que permita la adquisición suficientemente precisa de datos de ubicación y una transmisión efectiva en el área de monitoreo.

*OE*3. Desarrollar una plataforma de monitoreo y seguimiento intuitiva, que permita la visualización en tiempo real de la ubicación de las personas, así como la generación de alertas y análisis de datos relevantes para facilitar la toma de decisiones.

*OE*4. Realizar pruebas de campo en un entorno con condiciones de interés para evaluar la precisión y confiabilidad del sistema de localización y monitoreo basado en LoRa, realizando ajustes y optimizaciones necesarias para su correcto desempeño.

*OE*5. Evaluar la seguridad y privacidad del sistema, implementando medidas adecuadas para proteger la integridad de los datos de ubicación y garantizar la privacidad de las personas involucradas en el proceso de monitoreo.

1.2 Alcances y justificación

Este trabajo de título tiene como enfoque principal las aplicaciones de rastreo y geolocalización en entornos al aire libre. Todas las aplicaciones se centrarán en objetivos que se asume pueden desplazarse a velocidades pedestres (p.ej. personas caminando en senderos abiertos), por lo que la precisión de la localización a altas velocidades no será un aspecto relevante para este prototipo.

Se considerarán los siguientes parámetros para un usuario convencional:

- Latitud y longitud para indicar una posición en un mapa.
- Altitud: proporcionando una altitud aproximada para cada localización en metros sobre el nivel medio del mar (msnm).
- Nivel de batería: indicando el estado actual de los dispositivos.

En cuanto al administrador del sistema, se contemplarán los siguientes parámetros:

- airUtilTx: este valor, que si bien no es de interés para el usuario final, pero sí lo es para efectos de evaluación técnica del sistema, indicará el porcentaje de ocupación en la red mesh.
- Número de satélites: indicador de la cantidad de satélites detectados durante la adquisición de coordenadas, lo cual permitirá inferir el nivel de precisión obtenido en la geolocalización.

Capítulo 2

Estado del arte y técnico

En este capitulo se abordará el estado del arte y los recursos seleccionados para el desarrollo del prototipo de sistema de localización y monitoreo georeferencial en tiempo real basado en redes mesh LoRa. El capítulo comienza con un análisis detallado de la tecnología LoRa, explorando sus características y aplicaciones en el contexto del Internet de las Cosas (IoT). A continuación, se examinan soluciones similares, incluyendo tecnologías como Sigfox, BLE y 3G/4G, y se comparan con LoRa en términos de alcance, eficiencia energética y capacidad de transmisión de datos. Además, se presentan los recursos hardware y software seleccionados, destacando la placa de desarrollo LilyGO T-Beam y la plataforma Meshtastic. Estos recursos han sido cuidadosamente elegidos para garantizar la implementación efectiva del prototipo. En conjunto, este capítulo sienta las bases teóricas y prácticas necesarias para el desarrollo del sistema de localización y monitoreo propuesto.

2.1 Estado del Arte

2.1.1 Análisis y características de la tecnología LoRa para IoT

LoRa, un acrónimo para Long Range, es una tecnología de comunicación inalámbrica diseñada específicamente para aplicaciones de IoT que requieren largos alcances de transmisión y bajo consumo de energía. Desarrollada y patentada por la empresa Semtech [8], LoRa se ha convertido en una solución popular para la conectividad en el ámbito del IoT debido a sus características particulares en el área.

Tecnología	Rango (km)	Ancho de Banda (Mbps)	Sistema de Pago	Eficiencia Energética
3G	5	5	Pago	Alta eficiencia
4G	10	100	Pago	Media eficiencia
LoRa	15	0.3	Libre de costo	Alta eficiencia
Sigfox	50	0.03	Pago	Alta eficiencia
BLE	0.1	1	Libre de costo	Baja eficiencia

Tabla 1: Comparación de tecnologías

LoRa utiliza una técnica de modulación de espectro ensanchado llamado Chirp Spread Spectrum (CSS o espectro ensanchado con dispersión de frecuencia) [1], que permite una mayor resistencia a las interferencias y una mejor capacidad de penetración en entornos urbanos densos. Esta técnica de modulación permite que las señales LoRa se propaguen eficientemente a través de obstáculos físicos y alcancen distancias significativas sin requerir una potencia de transmisión elevada [9].

Además, LoRa opera en bandas de frecuencia sin licencia como las bandas ISM (Industrial, Scientific and Medical), lo que significa que se puede utilizar de manera gratuita y sin requerir concesiones de banda de frecuencia en la mayoría de las regiones del mundo. En la Tabla 2 aparecen algunos ejemplos. Esto facilita su adopción y despliegue en diversas aplicaciones de IoT.

Ubicación/País	Banda/Canales	Frecuencia (MHz)
Europa	863 - 870 MHz	868
América del Norte	902 MHz - 928	915
China	433 MHz	433
China	470 - 510 MHz	470
Corea	779 - 787 MHz	779
Chile	902 - 928 MHz	915

Tabla 2: Frecuencias internacionales utilizadas en el estándar de LoRa [10]

Una de las características distintivas de LoRa es su capacidad para crear redes mesh, también conocidas como redes de malla [5]. En una red mesh LoRa, los dispositivos pueden actuar tanto como transmisores como repetidores, lo que permite la comunicación directa entre dispositivos en lugar de depender únicamente de una infraestructura centralizada. Esto proporciona una mayor redundancia y escalabilidad a la red, ya que cada dispositivo puede retransmitir los datos y extender el alcance de

la señal. Además, la topología de red mesh permite la auto-organización y el enrutamiento flexible de los datos, lo que mejora la cobertura y la capacidad de recuperación de la red.

En resumen, LoRa es una tecnología inalámbrica diseñada para el IoT que se destaca por su largo alcance, bajo consumo de energía y capacidad de creación de redes mesh. Estas características la convierten en una opción atractiva para aplicaciones de localización y monitoreo georreferencial en tiempo real, ya que permite la transmisión eficiente de datos de ubicación y la mejora de la seguridad y eficiencia en diversos entornos de monitoreo de personas.

2.1.2 Soluciones similares

GoTenna

GoTenna[1] es un dispositivo de comunicación que permite enviar mensajes de texto y compartir ubicaciones sin necesidad de una conexión celular o Wi-Fi. El dispositivo se conecta a un teléfono inteligente a través de Bluetooth (BLE) y utiliza una red de malla para enviar y recibir mensajes.

	Gotenna CB[2] (MURS[3])	**Gotenna Mesh**	
		FCC/IC	**CE**
Frecuencias (MHz)	151-154 MHz	902-928 MHz	869.4-869.65MHz
Máxima Potencia	1W	1W	0.5 W
Mesh Protocol		Aspen Grove™	

Tabla 3: Frecuencias de GoTenna (first-gen) y GoTenna Mesh

Sonnet

Sonnet[4] es una aplicación de mensajería que permite a los usuarios enviar mensajes de texto y compartir ubicaciones sin necesidad de una conexión celular o

[1] https://gotenna.com/
[2] Citizens Band Radio
[3] Multi-Use Radio Service
[4] https://www.kickstarter.com/projects/sonnet/sonnet-decentralized-mobile-communication

Wi-Fi. Para la comunicación se utiliza la tecnología LoRa. Además, agrega un cifrado de extremo a extremo para garantizar la privacidad de los mensajes.

	Sonnet	**goTenna**
Alcance máximo	30 Km	10 Km
Tipo de comunicación	Mesh Network	Mesh Network
Compatibilidad con smartphones	Sí	Sí
Modo de comunicación	Texto y voz	Texto
Resistente al agua	Sí	Sí
Precio (por dispositivo)	$199	$149

Tabla 4: Comparación entre Sonnet y goTenna (Precios en USD)

2.2 Recursos Hardware y Software seleccionados para el prototipo

2.2.1 Hardware

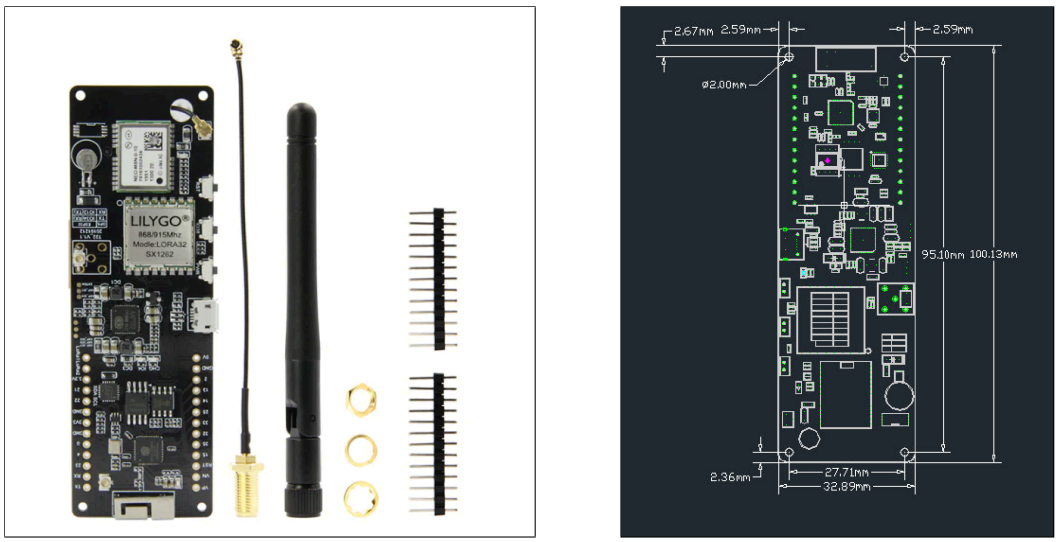
En esta sección, se describe el hardware seleccionado para el desarrollo del prototipo de sistema de localización y monitoreo georreferencial en tiempo real basado en red mesh LoRa. A continuación, se detalla la placa de desarrollo utilizada:

LilyGo T-Beam

La placa de desarrollo LilyGO T-Beam, fabricada por LilyGo[5], se ha seleccionado como el componente principal para la implementación del prototipo de sistema de localización y monitoreo georreferencial en tiempo real basado en red mesh LoRa. Esta placa proporciona un conjunto de características y funcionalidades que son adecuadas para el propósito del proyecto. A continuación, se destacan algunas de las características principales de la LilyGO T-Beam:

- Microcontrolador: La placa está equipada con un microcontrolador ESP32, que combina un procesador de doble núcleo, conectividad Wi-Fi y Bluetooth, y una amplia gama de periféricos integrados.

[5]https://www.lilygo.cc/



(a) Fotografía (b) Dimensiones

Figura 1: Placa de desarrollo LilyGo T-Beam v1.1

- Módulo LoRa: Incorpora un módulo LoRa SX1276, que permite la comunicación de largo alcance y bajo consumo de energía.
- Antena GPS: Cuenta con una antena GPS integrada para la obtención de la ubicación geográfica precisa.
- Batería recargable 18650: Dispone de una batería recargable de alta capacidad que permite el funcionamiento autónomo del sistema.
- Conectividad: Proporciona conectividad USB para la programación y carga de firmware, así como pines de E/S adicionales para la conexión de sensores externos.

2.2.2 Software

En esta sección, se describe el software seleccionado para el desarrollo del prototipo de sistema de localización y monitoreo georreferencial en tiempo real basado en red mesh LoRa. A continuación, se detalla la plataforma utilizada:

Meshtastic

Meshtastic es una plataforma de software de código abierto diseñada para facilitar la comunicación y el seguimiento georreferencial en redes mesh basadas en tecnología LoRa. Esta plataforma proporciona una serie de características y funcionalidades que son relevantes para el proyecto. A continuación, se destacan algunas de ellas:

- Funcionalidades de red mesh: Meshtastic facilita la creación y gestión de una red mesh utilizando dispositivos compatibles con radio LoRa. Permite la comunicación entre los nodos de la red, así como la transmisión de datos de ubicación en tiempo real.
- Localización y seguimiento: Meshtastic utiliza algoritmos de trilateración junto con GPS para calcular y visualizar la ubicación de los dispositivos dentro de la red mesh. Esto proporciona un seguimiento de los dispositivos en tiempo real.
- Interfaz de usuario intuitiva: La plataforma ofrece una interfaz de usuario amigable y de fácil comprensión, que permite la configuración de los nodos, la visualización de la ubicación en mapas interactivos y la gestión de la red.
- Código abierto y personalizable: Meshtastic se basa en un enfoque de código abierto, lo que significa que el software es libremente accesible y modificable. Esto permite adaptar la plataforma a las necesidades específicas del proyecto.

Capítulo 3

Pruebas de laboratorio

En el presente capítulo se evaluará el desempeño del firmware que el proyecto Meshtastic dispone para su integración en el hardware LilyGo T-Beam. Se entregan detalles de la instalación del firmware (proceso de descarga y carga del firmware en la placa) y las configuraciones necesarias para transmisión de datos en la red mesh.

Se abordan dos configuraciones de transmisión de datos: Transmisión a un servidor MQTT mediante Wi-Fi y la lectura de datos a través del puerto serie utilizando Python.

Se desarrolla y valida un sistema mínimo de comunicación PtP, diferenciando los eventos relevantes para el sistema. (Detalles de eventos en la Tabla 5). Se implementa el almacenamiento de eventos en un archivo CSV para análisis.

Finalmente, en este capítulo se plantea visión integral de las pruebas para evaluar el hardware y software en el proyecto. La instalación y configuraciones adecuadas son cruciales para el funcionamiento de la red mesh. La validación del sistema PtP demuestra la viabilidad de la solución propuesta.

3.1 Pruebas de Hardware y Software

A continuación se presentarán las pruebas de funcionamiento realizadas para evaluar el desempeño de hardware y software utilizando el firmware Meshtastic. Se destacarán dos aspectos clave: la instalación del firmware Meshtastic y las diferentes configuraciones de transmisión de datos en la red mesh.

3.1.1 Instalación de firmware Meshtastic y configuración de la región

La primera prueba consistió en la instalación del firmware Meshtastic en el hardware objetivo (LilyGo T-Beam v1.1). Para ello, en el repositorio[1] del proyecto Meshtastic se puede descargar el firmware en las diferentes versiones disponibles. Debido a la naturaleza del proyecto y su constante evolución, es probable que las configuraciones y/o protocolos sean modificados en el tiempo, para limitar los errores durante el desarrollo del prototipo y las pruebas se limitará a la versión estable v1.3.41.

Para la instalación / carga de firmware seguiremos los siguientes pasos:

Paso 1. Descarga de firmware: Acceder al repositorio en GitHub para descargar el firmware.

Paso 2. Instalar la herramienta esptool (disponible para Windows, Mac y Linux). Esta herramienta permite cargar el firmware en el microcontrolador de la LilyGo T-Beam.

Paso 3. Descomprimir el archivo descargado.

Paso 4. Abrir una terminal en la carpeta donde se encuentra el firmware.

Paso 5. Conectar la placa de desarrollo LilyGo T-Beam mediante el uso de un cable microUSB.

Paso 6. Ejecutar el comando “device-install.sh“ en Linux o “device-install.bat“ en Windows, seguido del argumento “-f {nombre_del_firmware}.bin“, esto iniciará la carga del firmware en la placa (Tiempo aproximado 1min).

Para realizar la configuración inicial, se debe instalar la librería de Meshtastic para Python, puede ser instalada siguiendo estos pasos:

Paso 1. Instalar la librería de Meshtastic para Python.

```
pip install meshtastic==1.3.41
```

Paso 2. Para validar la correcta instalación, utilizar el siguiente comando: `meshstatic --version` (ejemplo en la Fig. 2)

[1] https://github.com/meshtastic/firmware

Figura 2: Interfaz terminal de Linux, ejecucion de comando validando la instalación correcta de la versión especifica de Meshtastic, se ejecuta dentro de un entorno virtual de Python (pipenv)

Paso 3. Al iniciar, la placa requiere que sea seteada la región para su funcionamiento, para aplicar la región correspondiente a Chile que corresponde al uso de la frecuencia de 915 MHz[10], para esto se utilizara el siguiente comando:

```
meshtastic --set region ANZ
```

Además, se realizó una configuración adecuada del dispositivo en la región geográfica correspondiente. Esto permitió que el dispositivo se comunicara correctamente con otros nodos de la red mesh ubicados en la misma región. Durante esta prueba, se verificó que la instalación del firmware se completara sin errores y que el dispositivo pudiera unirse exitosamente a la red de comunicación de malla. Se comprobó la estabilidad y confiabilidad de la conexión, así como la capacidad del dispositivo para transmitir y recibir datos dentro de la red.

3.1.2 Diferentes configuraciones de transmisión de datos de red mesh

En esta etapa, se exploraron diferentes configuraciones de transmisión de datos dentro de la red de comunicación de malla. Se probaron dos configuraciones específicas:

Transmisión de datos a servidor MQTT

Para lograr una transmisión exitosa de datos al servidor MQTT, es necesario seguir una serie de pasos de configuración en el dispositivo LilyGo T-Beam. Primero, se debe establecer una conexión con una red Wi-Fi cercana utilizando el módulo integrado de Wi-Fi.

A continuación, se muestra un ejemplo de cómo configurar la red Wi-Fi en el dispositivo utilizando el código de configuración:

```
# Habilitación de WiFi
meshtastic --set network.wifi_enabled true
# Modo de Conexión WiFi (Modo Cliente o Hotspot)
meshtastic --set network.wifi_mode CLIENT
# Nombre de la red WiFi
meshtastic --set network.wifi_ssid "WIFI_SSID"
# Contraseña red WiFi
meshtastic --set network.wifi_psk "WIFI_PASS"
```

Código 1: Configuración de Red WiFi

Luego de esto es necesario configurar los parámetros de comunicación MQTT, en el Código 2 se puede ver como configurar algunos de estos parámetros como la dirección IP del broker, el nombre de usuario y la contraseña, entre otras configuraciones.

```
# Habilitación de MQTT
meshtastic --set mqtt.enabled true
# Direccion Host del Broker MQTT
meshtastic --set mqtt.address "BROKER_IP"
# Desactivar encriptación por MQTT
meshtastic --set mqtt.encryption_enabled false
# Activar formato JSON en payload MQTT
meshtastic --set mqtt.json_enabled true
# Configurar Usuario y Cotraseña MQTT
meshtastic --set mqtt.password "MQTT_USER"
meshtastic --set mqtt.username "MQTT_PASS"
```

Código 2: Configuración de comunicación MQTT

Una vez que se ha logrado establecer una conexión exitosa entre el dispositivo

y el servidor MQTT, los datos comienzan a circular a través de la red mesh. Luego, se realiza una evaluación para asegurar la recepción adecuada de los datos transmitidos por el dispositivo en el servidor MQTT, validando así la efectividad de este método de transmisión de datos.

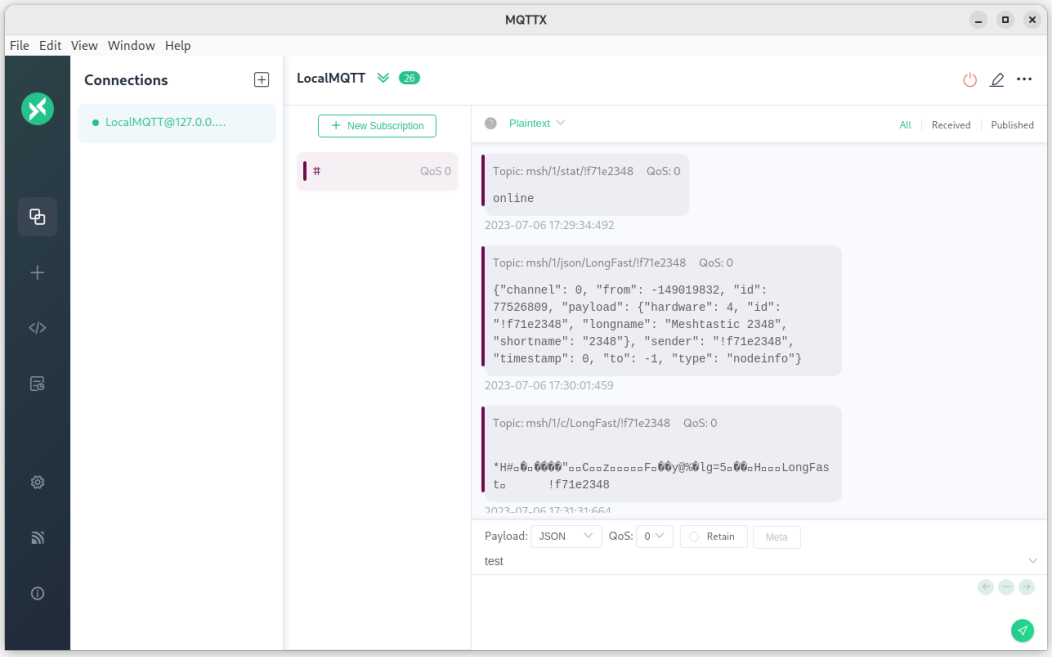


Figura 3: Registro de comunicaciones MQTT. (Software MQTTX, de la empresa EMQX)

En la Figura 3, se puede apreciar que los mensajes son recibidos periódicamente en el Broker[2], el cual se implementa utilizando Docker[3] y Mosquitto[4]. Estos mensajes se presentan en dos formatos distintos: JSON[5], que es un formato legible para los humanos y de fácil comprensión, y protobuf[6], que es un formato binario optimizado para el rendimiento y requiere conocer la estructura de los datos para poder interpretarlo adecuadamente.

[2]Un servidor MQTT.
[3]`https://www.docker.com/`
[4]`https://mosquitto.org/`
[5]JavaScript Object Notation
[6]Un formato de serialización binaria eficiente y de alto rendimiento para el intercambio de datos estructurados

Dado que el proyecto Meshtastic se encuentra en constante desarrollo, es posible que tanto los objetos JSON como los datos en formato protobuf sufran cambios en versiones futuras. Es importante tener en cuenta que para utilizar este enfoque de transmisión de datos se requiere una conexión WiFi, lo cual puede presentar desafíos adicionales. Esta dependencia de WiFi introduce dificultades para mantener una conectividad estable y puede aumentar la latencia en la entrega de los mensajes. Además, al no contar con un sistema de respaldo en caso de pérdida de la conexión WiFi, existe el riesgo de interrupciones en la comunicación.

Lectura de datos por medio de puerto serie

Para interactuar con el dispositivo y utilizar su puerto serie, es necesario utilizar la librería de Meshtastic que se ha instalado previamente. Esta librería no solo actúa como una interfaz de línea de comandos (CLI), sino también como una API que permite trabajar con el dispositivo utilizando el lenguaje de programación Python. Utilizando a esta funcionalidad, es posible desarrollar aplicaciones personalizadas, realizar consultas y enviar comandos al dispositivo a través de la API proporcionada por la librería de Meshtastic.

A continuación, se desarrolló un código (Anexo B) utilizando la librería en Python, con el objetivo de analizar diversos eventos posibles. Este código permite la detección de nuevos eventos en el puerto serie, así como la recepción de mensajes por parte del dispositivo LilyGo T-Beam. El script permite identificar diferentes tipos de eventos implementados por los desarrolladores de Meshtastic y almacenarlos como un texto plano. Sin embargo, dado que el alcance de este proyecto no busca abarcar todas las posibilidades, se discriminarán los eventos necesarios para el presente trabajo y se omitirán los demás.

Evento	Descripción
ADMIN_APP	Aplicación de Meshtastic
TEXT_MESSAGE_APP	Mensajes de texto
POSITION_APP	Localizaciones GPS
NODEINFO_APP	Información de la placa en la red Mesh
TELEMETRY_APP	Telemetría de la placa LilyGo T-BEAM

Tabla 5: Resumen de eventos detectados en la red mesh

En la Tabla 5 se presentan algunos de los diversos eventos detectados durante

las pruebas utilizando el script. Tras revisar cada *payload*[7] desencriptado, se determinó que este método era más seguro y viable para manejar los datos en comparación a la prueba realizada en la sección 3.1.2.

3.2 Desarrollo y validación de sistema mínimo (Point to Point)

A continuación, se llevará a cabo el desarrollo y la validación del sistema mínimo. Para lograr este objetivo, se analizarán los datos recopilados durante las pruebas realizadas en el capítulo anterior y se procesarán para obtener un control completo sobre la información recibida. Antes de comenzar con el desarrollo, es fundamental establecer un entorno de desarrollo adecuado para evitar posibles problemas o conflictos derivados de diferentes versiones de software en uso.

3.2.1 Entorno de desarrollo

Para organizar y gestionar las dependencias del proyecto, utilizaremos *pipenv*[8], una herramienta popular para la creación y gestión de entornos virtuales y paquetes en Python. Utilizar pipenv permite mantener un control más estricto sobre las dependencias y las versiones utilizadas en nuestro proyecto.

Adicionalmente, será necesario tener una estructura de carpetas bien definida. En la Figura 4 se muestra la estructura base del entorno de desarrollo.

En donde:

- **Pipfile** y **Pipfile.lock** corresponden a archivos generados por pipenv.
- **src**, nomenclatura para referirse a source (fuente), será el directorio donde se encontrara el código de fuente.
- **docs**, directorio para archivos o documentación necesaria, se incluirán listado de comando e instrucciones para poder hacer funcionar el sistema.

Además, se utilizará la herramienta Git para llevar a cabo la gestión de versiones del proyecto.

[7] Carga de datos
[8] https://pipenv-es.readthedocs.io/es/latest/

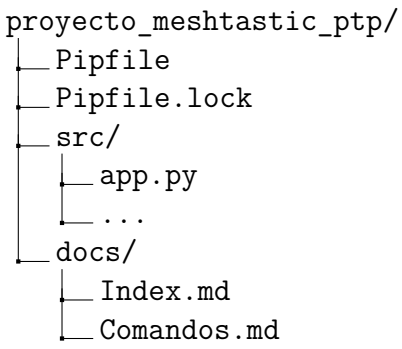


Figura 4: Estructura de carpeta para el entorno de desarrollo.

3.2.2 Tipo de eventos y Tipado de datos

Utilizando el código desarrollado en la sección anterior, se extenderán sus capacidades para permitir desarrollar un prototipo que permita diferenciar los diferentes eventos, correspondientes a POSITION_APP y TELEMETRY_APP y procesar su información según sus características.

Para desarrollar este sistema, se deben tener disponibles dos dispositivos Lily-GO T-Beam. Ambos dispositivos deben tener instalada la misma versión del firmware Meshtastic y pertenecer a la misma red mesh. Además, es importante asegurarse de que ambos dispositivos tengan la misma configuración de telemetría (configuración utilizada en Anexo C y Anexo D).

En la Sección 3.1.2, se han identificado algunos de los eventos que circulan a través de la red mesh y que son decodificados mediante el puerto serie utilizando la librería de Meshtastic. En los Código 3 y Código 4, se representa la estructura de datos, incluyendo nombre de variables y tipo de dato, solo se considerarán a los eventos que se estimen necesarios para el desarrollo principal del trabajo.

Ambas estructuras comparten una configuración similar, pero empiezan a mostrar diferencias cuando se descodifica la información. Para facilitar el desarrollo, los datos se convertirán en estructuras con tipos definidos, lo que permitirá trabajar con ellos como objetos orientados a la programación (POO), convirtiendo cada mensaje en un objeto con tipos y estructuras personalizadas.

Se utiliza la biblioteca de Pydantic, la cual aprovecha las nuevas características de Python para permitir la tipificación y realizar comprobaciones, con el fin de reducir errores en la programación. Se crean diferentes clases basadas en los modelos base (BaseModel) de Pydantic, que tomarán el objeto recibido por la biblioteca de Mesh-

```
from: int
to: int
decoded:
    portnum: str
    payload: bytes
    telemetry:
        time: int
        deviceMetrics:
            batteryLevel: int
            voltage: float
            airUtilTx: float
            raw: object
id: int
rxSnr: float
hopLimit: int
rxRssi: int
raw: object
fromId: str
toId: str
```

Código 3: Estructura de datos de TELEMETRY_APP

tastic y lo convertirán en una estructura de llamada más sencilla. Con esta finalidad, se podrá trabajar de manera más dinámica con las diferentes estructuras.

En él Anexo E, se puede observar que el código corresponde al tipado utilizando la librería Pydantic. En este código, se define la clase *BasePacket* como la base principal de los mensajes. Esta clase depende de un atributo llamado *decoded*, el cual puede contener propiedades opcionales como *position* y *telemetry*. La presencia o ausencia de estas propiedades en el mensaje recibido determinará si son consideradas en el atributo *decoded*. En caso de existir, se utilizarán los modelos *Position* y *Telemetry* respectivamente para asignarles los valores correspondientes. Si alguna de estas propiedades no está presente en el mensaje, se les asignará un valor nulo o *None*, lo cual facilitará su identificación.

3.2.3 Desarrollo y validación de sistema mínimo

Para validar el funcionamiento del sistema mínimo o prueba Point to Point (PtP), es requerido que el programa sea capaz de identificar el origen del mensaje, como se puede ver en las estructuras de datos descritas en la sección anterior; *fromId* es una variable dentro del mensaje que contiene la información con el autor del mensaje

```
from: int
to: int
decoded:
    portnum: str
    payload: bytes
    position:
        time: int
        timestamp: int
        latitudel: int
        longitudel: int
        latitude: float
        longitude: float
        altitude: int
        satsInView: int
id: int
rxSnr: float
hopLimit: int
rxRssi: int
raw: object
fromId: str
toId: str
```

Código 4: Estructura de datos de POSITION_APP

original. Por lo tanto, si deseamos validar este sistema, se debe ser capaz de recibir información desde un emisor en específico en una primera instancia.

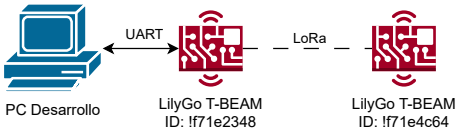


Figura 5: Diagrama de prueba PtP

Utilizando el código de la Sección 3.1.2, se agregarán modificaciones como el tipado descrito en la sección anterior a los mensajes emergentes y según el tipo de mensaje que llegue se ejecutaran diferentes acciones o se imprimirán diferentes mensajes en consola. Adicionalmente, se agregó la capacidad de registrar en CSV los eventos, para ir almacenando y validando el correcto funcionamiento del sistema.

Como se puede apreciar en las Figura 6 y Figura 7 el dispositivo con ID ***!f71e4c64*** emite constantemente en un intervalo de 2 min, tiempo que fue definido anteriormente cuando se realizó la configuración para esta prueba (120 segundos). Esto valida la comunicación inalámbrica entre ambas LilyGo T-Beam.

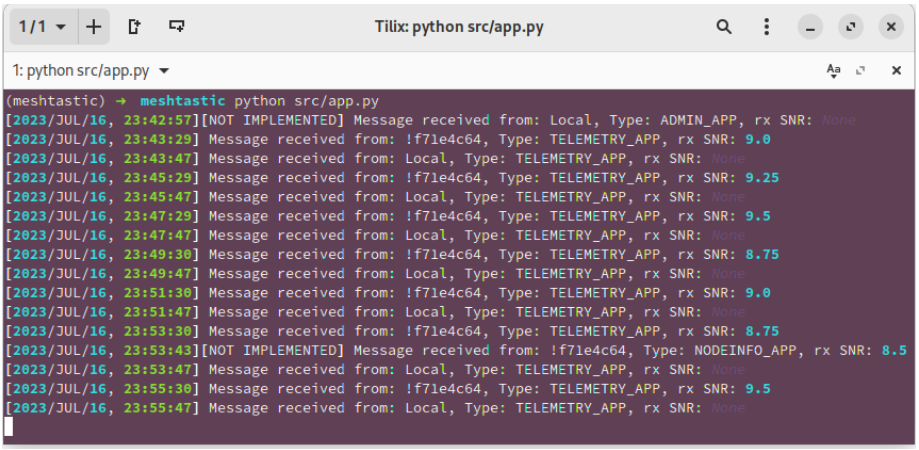


Figura 6: Ejecución de prueba de verificación PtP

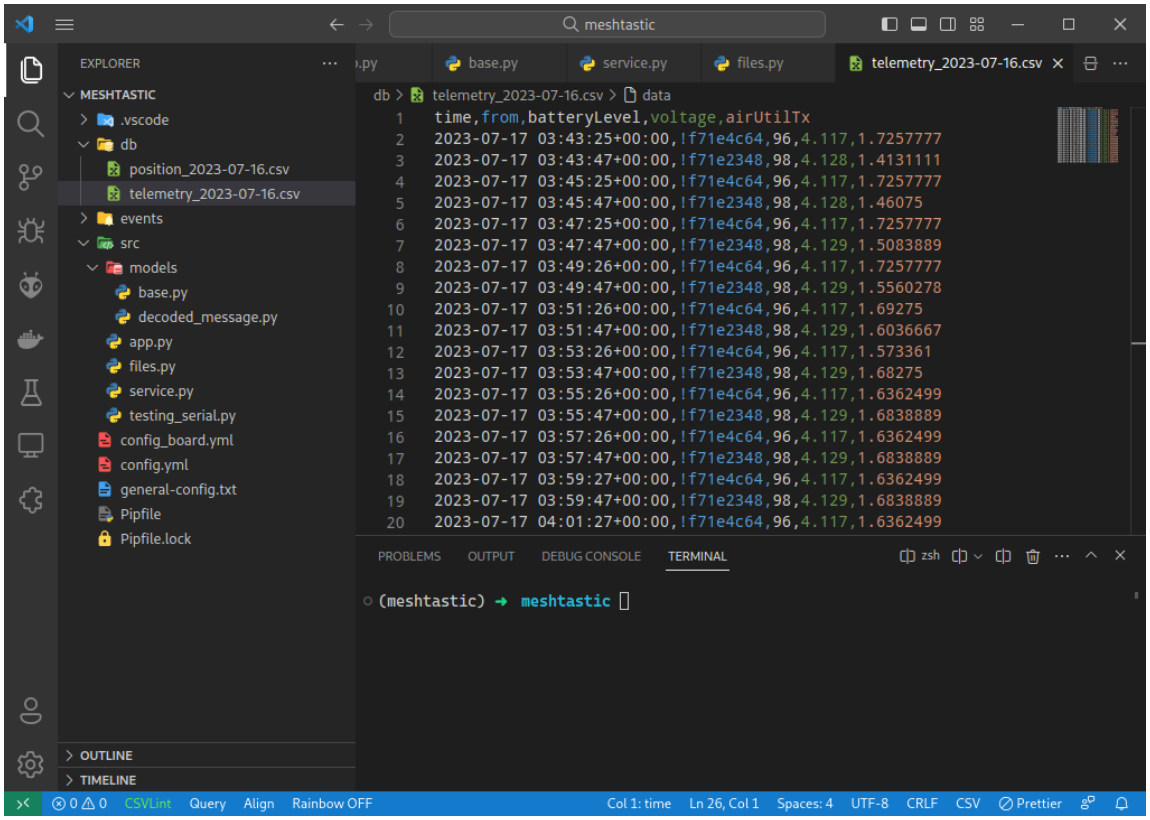


Figura 7: Entorno de Desarrollo y Resultados CSV

Capítulo 4

Diseño e Infraestructura de red

En el siguiente capítulo se abordan aspectos cruciales para el funcionamiento eficiente y escalable de la red mesh utilizando la tecnología LoRa y el firmware Meshtastic.

Se propone una infraestructura que incluye un servidor central y un Gateway para mejorar la gestión de la información en la red.

Se describen las características y funciones de las diferentes estaciones y nodos en la red mesh, como la Estación Base (Meshtastic Gateway) y las Estaciones Fijas Repetidoras con alimentación fotovoltaica. También se detalla el diseño de los Nodos Móviles con batería Li-ion para permitir su movilidad y autonomía en la red.

4.1 Infraestructura

Al trabajar con la tecnología LoRa, una de sus ventajas es que los protocolos de comunicación son definidos por el usuario. Pero para esto, es importante seguir ciertas recomendaciones para evitar problemas y garantizar un funcionamiento eficiente. En este sentido, se pueden tomar en consideración las recomendaciones del estándar IEEE 802.11s[5], que ofrecen directrices para el diseño de redes inalámbricas de tipo malla o Mesh, aunque su aplicación es estrictamente para el funcionamiento de redes inalámbricas de tipo WiFi.

Dentro de los alcances del trabajo se plantea desarrollar un prototipo escalable, este sistema debe ser capaz de mantener su rendimiento a pesar del número creciente de usuarios o en este caso de diferentes fuentes emisoras de información.

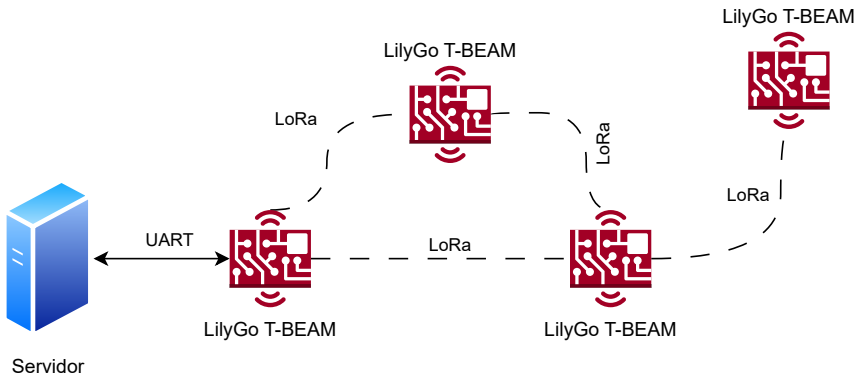


Figura 8: Infraestructura Red Mesh LilyGo T-Beam, escenario individual

En un primer escenario, se plantea la configuración descrita en la Figura 8. Se presenta un Servidor central que procesa toda la red Mesh a través de su conexión mediante puerto serie. Este sistema es útil para el monitoreo de datos a pequeña escala, pero a medida que aumenta la cantidad de emisiones simultáneas, también aumenta la probabilidad de pérdida de información.

Con el fin de reducir esta carga en el servidor, se propone agregar un elemento adicional: un Gateway que cumple la misma función de recibir la información de la red Mesh y procesarla. Sin embargo, en lugar de procesarla internamente, se encarga de enviarla a un servidor dedicado para su almacenamiento, como se aprecia en la Figura 9. Adicionalmente en la Figura 10 se describe el proceso. Este enfoque plantea el desafío de la duplicidad de información, pero los servicios dentro servidor, como por ejemplo las bases de datos utilizadas, se encargarán de filtrar y gestionar dicha información.

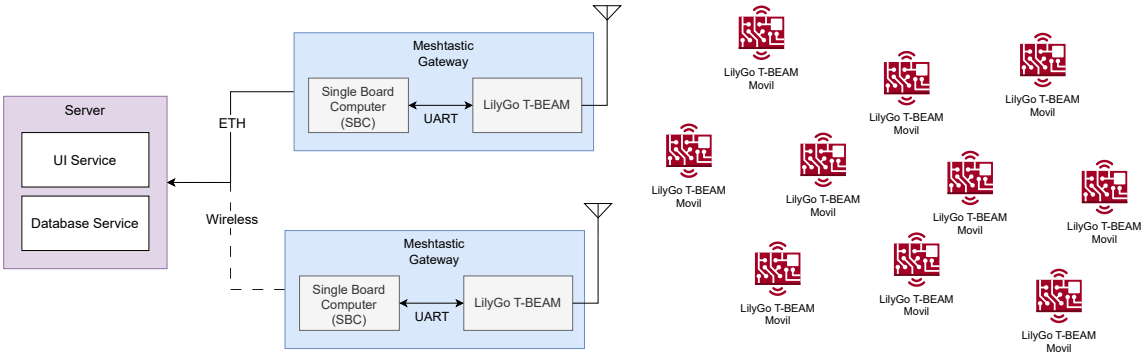


Figura 9: Infraestructura Red Mesh LilyGo T-Beam, escenario escalable

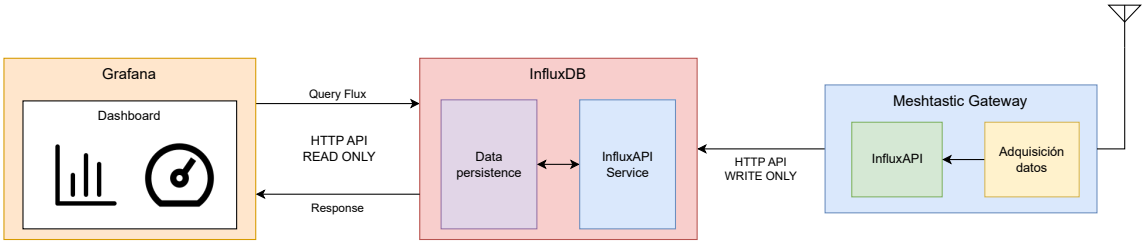


Figura 10: Diagrama de procesamiento, inyección y procesamiento de datos.

4.2 Diseño de Estaciones y Nodos

4.2.1 Estación Base (Meshtastic Gateway)

Corresponde a una abstracción de una parte del sistema individual, de la sección anterior, donde se captura la información de la red mesh y procesa.
Debido a que comercialmente no existe un elemento que satisfaga estas necesidades, puede ser desarrollado utilizando una Single Board Computer[1] y una LilyGo T-Beam. Dentro de los parámetros de diseño para esto, se debe considerar que la SBC debe funcionar con un sistema de operativo con compatibilidad nativa para Python, para esto cualquier placa que soporte un sistema operativo Linux y que tenga arquitectura ***AMD64*** o ***AARCH64***[2] se consideraría válida en este escenario.
La transmisión entre la SBC y la LilyGo T-Beam se mantendrá mediante el puerto serie. No obstante, la comunicación con el servidor deberá ser establecida mediante conexiones directas a la red (Ethernet) o a través de conexiones inalámbricas (Wi-Fi). Considerando lo anterior, la Figura 11 representa en forma de bloques los elementos necesarios para el desarrollo de este elemento fundamental en la red Mesh.

[1] SBC
[2] ARM64

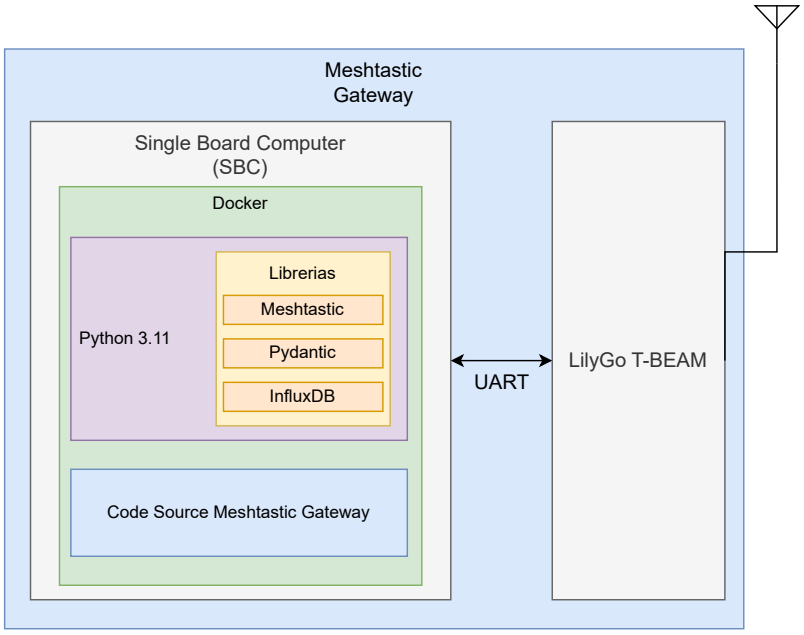


Figura 11: Diagrama de Meshtastic Gateway

4.2.2 Estación Fija Repetidora

Además, el desafío inicial en este tipo de redes mesh radica en la cantidad limitada de usuarios iniciales, lo que resulta en una cobertura restringida o limitada. Para solventar este problema en la realización de este prototipo, se implementarán estaciones fijas repetidoras cuyo rol será participar como usuarios activos dentro de la red Mesh en ubicaciones determinadas para mantener una cobertura estable.

Estas estaciones ampliarán la cobertura de la red Mesh y necesitarán alimentación constante. Por lo tanto, se ha diseñado un sistema de alimentación basado en energía fotovoltaica para garantizar su funcionamiento ininterrumpido. La utilización de energía solar permitirá que las estaciones fijas repetidoras sean autosuficientes y respetuosas con el medio ambiente, asegurando así la disponibilidad y estabilidad del sistema en ubicaciones donde no sea factible contar con una conexión eléctrica convencional. Con este enfoque sostenible, se busca optimizar el rendimiento y la eficiencia de la red Mesh, facilitando su despliegue en áreas remotas o de difícil acceso.

Como se ilustra en la Figura 12, la estación fija repetidora incorpora un sistema de alimentación fotovoltaica. Los componentes fundamentales de esta estación son:

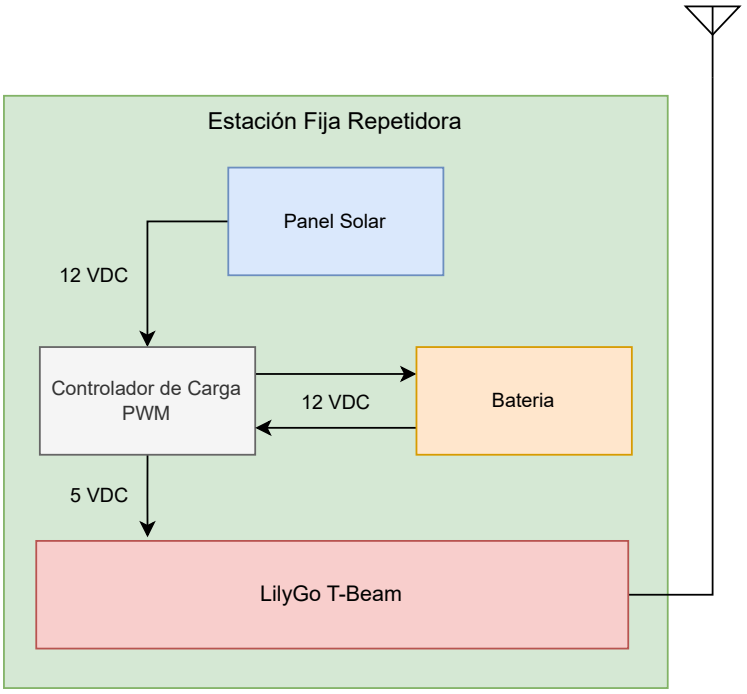


Figura 12: Diagrama de Estación Fija Repetidora

- Un panel solar destinado a la captación de energía solar.
- Un controlador de carga PWM encargado de regular el flujo energético y gestionar la carga de la batería recargable compatible con dicho controlador.
- Una batería recargable.
- La LilyGO T-Beam, desempeñando la función de nodo intermedio para la comunicación dentro de la red Mesh.

4.2.3 Nodos Móviles

Estos corresponden a las unidades de LilyGo T-Beam distribuidas entre los usuarios del sistema, cuentan con un sistema de batería integrado que utiliza una batería de Li-ion (Litio) 18650. Esta tecnología de baterías proporciona la energía necesaria para el funcionamiento del Nodo Móvil y asegura su autonomía, permitiendo así su movilidad y despliegue en diferentes ubicaciones dentro de la red Mesh. De esta manera, los Nodos Móviles pueden operar de forma independiente y mantener la conectividad y la captura de posiciones de los usuarios de manera eficiente.

Debido a que la compra de los insumos no contempla, la compra de elementos protectores como carcasas para los equipos se solicitó al equipo de ProteinLab[3], el desarrollo de un case protector para la LilyGo T-Beam. En la Figura 13 se pueden observar los render de la cubierta para la protección, incluyendo iconografía relacionada con el proyecto y Universidad.

(a) *Render de case 3D (frontal)* (b) *Render de case 3D (trasera)*

Figura 13: Encapsulados para dispositivos Móviles. Cortesía de ProteinLab

[3]ProteinLab: `https://proteinlab.cl/`

Capítulo 5

Desarrollo de prototipo del sistema

En el siguiente capítulo se abordarán los preparativos para poder desarrollar el prototipo de sistema, para esto es necesario acondicionar o preparar los equipos para el correcto funcionamiento del sistema.

Se abordarán los requisitos mínimos de funcionamiento y se describirá el uso de algunas tecnologías para permitir un trabajo en un ambiente de desarrollo y se describirán los pasos para realizar el despliegue del sistema.

5.1 Acondicionamiento de equipos

Los diversos equipos deben ser preparados para permitir su correcto funcionamiento como parte del sistema. Dentro de los equipos del sistema se encuentran:

1. Las tarjetas de desarrollo LilyGo T-Beam v1.1
2. El Servidor de Desarrollo
3. Single Board Computer, para el desarrollo del Gateway

A continuación sé irá detallando los requerimientos y procesos para acondicionar cada uno de estos elementos.

5.1.1 LilyGo T-Beam

Para permitir la interoperabilidad con los diferentes sistemas, los equipos T-BEAM deben cumplir con los siguientes requisitos mínimos:

- Misma versión de Firmware Meshtastic (v1.3.41)
- Configuraciones de banda base / región (AU915 / 915MHz)
- Asignación de canal público (Predeterminado Meshtastic)

En este proceso se utilizaron los pasos descritos en la Sección 3.1.1 y las configuraciones del Anexo C.

5.1.2 Servidor

Para asegurar el adecuado procesamiento, almacenamiento y visualización de la información proveniente de la red Mesh, se ha incorporado un servidor en la arquitectura del sistema.

A fin de alcanzar un rendimiento óptimo, es imprescindible que este servidor cumpla con ciertos requisitos mínimos de hardware:

- Procesador de 64 bits (x64, ARM64 o arquitectura similar).
- Memoria RAM con una capacidad mínima de 2GB.
- Sistema Operativo basado en Kernel Linux, como Debian, Ubuntu, Fedora, u otros, que sean compatibles con las siguientes herramientas requeridas:
  - Git[1]
  - Docker y Docker Compose[2]

El servidor es esencial para el funcionamiento del sistema, ya que se encarga de gestionar los datos y asegurar que las aplicaciones se ejecuten de manera eficiente. La instalación de Git y Docker en el servidor es crucial debido a su importancia en el desarrollo y funcionamiento del sistema:

[1] https://git-scm.com/download/linux
[2] https://docs.docker.com/engine/install/

- **Git**: Proporciona un sistema de control de versiones distribuido que permite rastrear y gestionar los cambios en el código fuente y la documentación. Facilita la colaboración entre desarrolladores, la creación de ramas para nuevas funcionalidades y la gestión de fusiones para mantener una base de código coherente.
- **Docker**: Es una plataforma de contenedores que simplifica la creación, distribución y ejecución de aplicaciones en entornos independientes. Los contenedores de Docker ofrecen aislamiento, portabilidad y escalabilidad, permitiendo desplegar aplicaciones de manera eficiente en diferentes entornos y gestionarlas de forma sencilla.

En resumen, al cumplir con los requisitos de hardware y tener instalados Git y Docker, el servidor estará preparado para enfrentar las demandas del sistema y asegurar un desarrollo colaborativo eficiente, así como un despliegue de aplicaciones más ágil y escalable.

5.1.3 Single Board Computer

Debido a que se requiere interoperabilidad entre las diferentes partes del sistema se recomienda que la SBC cumpla con unos requisitos mínimos similares a la contra-parte del servidor.

- Procesador de 64 bits (x64, ARM64 o arquitectura similar).
- Memoria RAM de 512MB como mínimo.
- Sistema Operativo basado en Kernel Linux (Debian, Ubuntu, Fedora, etc.)
  Este sistema debe ser compatible y tener instaladas las siguientes herramientas:
  - Git
  - Docker y Docker Compose

5.2 Almacenamiento de datos

La elección de utilizar la base de datos InfluxDB para el almacenamiento en el servidor es una decisión fundamentada y considerando los requerimientos técnicos

del proyecto, y dada la naturaleza de los diferentes tipos de datos recibidos y la necesidad de almacenar y procesar una gran cantidad de información con una resolución de tiempo precisa, es esencial seleccionar una base de datos que permita manejar estos desafíos.

La base de datos InfluxDB es una opción destacada debido a sus características y ventajas en comparación con sus competidores[11], como TimeScaleDB y Prometheus. Las razones para elegir InfluxDB incluyen:

1. Implementación rápida: InfluxDB ofrece una configuración sencilla y una puesta en marcha rápida, lo que permite reducir significativamente el tiempo de desarrollo necesario para poner en funcionamiento el sistema de almacenamiento de datos.

2. Alto rendimiento: InfluxDB está diseñado para manejar grandes volúmenes de datos con alta velocidad de escritura y lectura. Esto es crucial para gestionar los flujos continuos de datos que pueden provenir de múltiples fuentes en tiempo real.

3. Resolución de tiempo: InfluxDB es especialmente adecuado para series de tiempo, lo que lo convierte en una opción idónea para datos que requieren una resolución de tiempo precisa.

4. Soporte para datos de series temporales: InfluxDB está optimizado para manejar datos de series temporales, lo que lo hace altamente eficiente en el almacenamiento y consulta de datos que evolucionan con el tiempo.

5. Comunidad y soporte: InfluxDB cuenta con una comunidad activa y un buen soporte técnico, lo que facilita la resolución de problemas y la obtención de ayuda en caso de ser necesario.

Considerando lo anterior, InfluxDB como la base de datos para el almacenamiento en el servidor cumple con los requerimientos técnicos del proyecto. Su facilidad de implementación, alto rendimiento y capacidad para manejar datos de series temporales hacen que InfluxDB una solución adecuada para satisfacer las necesidades específicas de almacenamiento y procesamiento de datos del sistema.

5.3 Desarrollo de Meshtastic Gateway

Con el desarrollo realizado en la sección 3.2.3, se realizará una refactorización, permitiendo no solo reutilizar el código, sino volviéndolo más legible, mantenible y eficiente, reduciendo la complejidad y aumentando la calidad general del software.

Dentro de este proceso algunos aspectos importantes fueron:

1. Lectura y configuración del servicio del Gateway desde variables de entorno: Se brinda la posibilidad de realizar configuraciones sin tener que intervenir directamente en el código fuente. Al utilizar variables de entorno, el sistema se vuelve más flexible y adaptable a distintos entornos de despliegue.

2. Modularización del código del servicio para permitir agregar funciones según el tipo de mensaje: Al dividir el código en módulos y permitir la adición de funciones según el tipo de mensaje, se obtiene un diseño más flexible y escalable. Esto facilita la incorporación de nuevas características relacionadas con diferentes tipos de mensajes o la adaptación a distintos protocolos de comunicación. La modularidad también favorece la legibilidad del código, al agrupar lógicas relacionadas en módulos separados.

3. Otros aspectos abordados durante el proceso de refactorización incluyeron la optimización de algoritmos y estructuras de datos para mejorar la eficiencia del servicio del Gateway, la identificación y corrección de posibles problemas de rendimiento y de integración para asegurar el correcto funcionamiento del código en diferentes escenarios.

Luego de tener cubierto los aspectos principales en el código se procede a integrar la funcionalidad de almacenamiento en la base de datos seleccionada, para esto se puede seguir la documentación oficial desde la página de InfluxData[3], donde se provee información para trabajar usando la librería proporcionada por ellos.

[3]https://docs.influxdata.com/influxdb/v2.7/api-guide/client-libraries/python/

5.4 Despliegue del sistema

Para realizar el despliegue efectivo del sistema, se utilizará Docker y Docker Compose. Estas herramientas permiten una implementación rápida y ágil, asegurando la portabilidad y consistencia de las aplicaciones en diferentes entornos.

Como se mencionó en la sección 5.1.2, Docker es una plataforma de contenedores que proporciona aislamiento y portabilidad a las aplicaciones (ver Figura 14). Al utilizar Docker Compose, se podrán definir y gestionar todos los servicios y sus relaciones de manera declarativa en un archivo YAML, lo que simplifica enormemente el despliegue y la gestión de aplicaciones complejas.

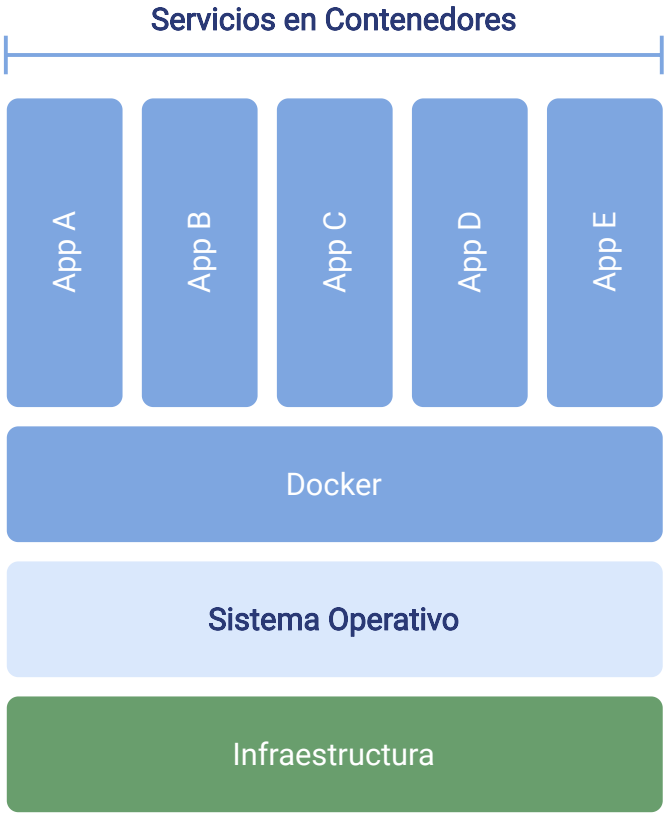


Figura 14: Diagrama de Arquitectura de funcionamiento Docker

El despliegue del sistema se realizará siguiendo los siguientes pasos:

1. Creación de archivo de construcción de las imágenes de docker para el servicio de Meshtastic Gateway utilizando el Dockerfile, esto permite construir una imagen para diferentes arquitecturas. Esto asegurará que el servicio se ejecute en un entorno controlado y replicable.

2. Creación de los archivos de configuración Docker Compose[4] que describa los servicios necesarios para el funcionamiento del sistema, incluido el servicio Meshtastic Gateway y la base de datos InfluxDB.

3. Distribuir los archivos de configuración Docker Compose entre los diferentes entornos de trabajo, Servidor y Single Board Computer para el Meshtastic Gateway.

4. Construcción de la imagen Meshtastic Gateway en la SBC, mediante el comando `docker compose build`, esto permite crear una imagen Docker correspondiente a la arquitectura del entorno de trabajo.

5. Ejecución del comando `docker-compose up -d` para levantar los servicios definidos en el archivo Docker Compose. Esto desplegará los diferentes servicios del sistema y permitirá que los servicios se comuniquen entre sí dentro de la red Docker.

Con esta configuración, el sistema estará listo para funcionar, permitiendo una gestión eficiente de la información generada por la red mesh y facilitando la interacción y comunicación entre los diferentes elementos del sistema.

[4] docker-compose.yml

Capítulo 6

Interfaz de usuario

En este capítulo, se abordará el proceso de desarrollo de la interfaz de usuario (UI) para el sistema de red mesh, utilizando la plataforma Grafana. A lo largo del capítulo, explicaremos los pasos para la instalación de Grafana, así como la conexión con la base de datos InfluxDB. Además, presentaremos el desarrollo de una interfaz centrada en el principio KISS (Keep It Simple, Stupid) para asegurar una experiencia de usuario efectiva y satisfactoria.

6.1 Instalación de la plataforma Grafana

Grafana es una herramienta de software libre diseñada para la visualización de datos. Permite conectarse a diversas bases de datos y se destaca por su capacidad de personalización y la posibilidad de extender sus funciones mediante plugins.

Existen dos versiones de Grafana: Enterprise y OSS (Open Source Software). La versión Enterprise requiere una licencia para su uso y ofrece funciones, plugins y servicios exclusivos, mientras que la versión OSS es completamente gratuita y cuenta con el respaldo de una amplia comunidad que proporciona plugins para mejorar la experiencia. En este proyecto, utilizaremos Grafana OSS, ya que cumple con los requerimientos específicos necesarios.

Para realizar la instalación, optaremos por utilizar Docker y su complemento Docker Compose. GrafanaLabs proporciona una imagen del contenedor de Docker

que permite ejecutar Grafana OSS sin necesidad de realizar instalaciones a nivel del sistema.

En él (`docker-compose.yml`), se definen los `volumes` para facilitar la actualización de la versión de Grafana, instalar plugins y guardar los cambios fuera del contenedor. También se establecen las `networks` que permiten la comunicación con la base de datos InfluxDB u otros contenedores, así como un puerto (3000) para la comunicación externa con Grafana:

Al utilizar el comando `docker compose up -d`, debería ser descargada la última versión de Grafana OSS desde el registro de contenedores en DockerHub, que es actualizada constantemente por GrafanaLabs y una vez descargada debería levantarse el servicio y ejecutarse en el puerto previamente indicado. Al tratarse de una aplicación web se debe utilizar la IP y puerto en el navegador para acceder a Grafana, como se puede ver en la Figura 15. (`http://{IP}:{PUERTO}/`)

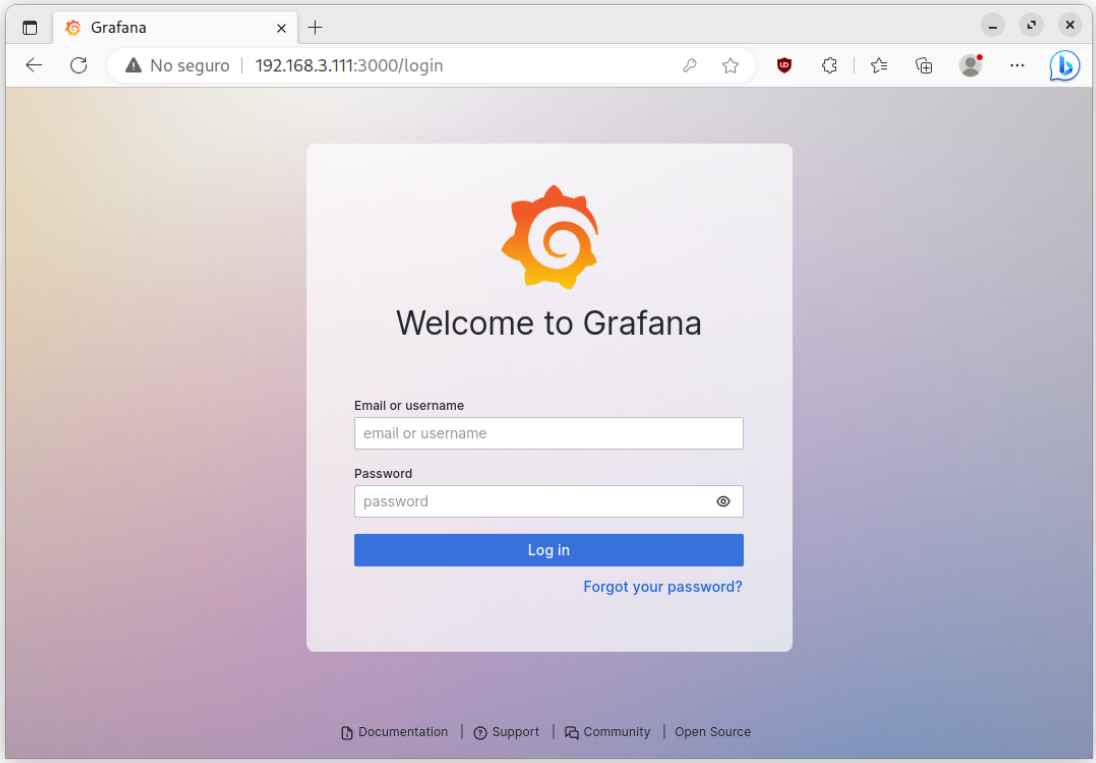


Figura 15: Servicio de Grafana ejecutado correctamente.

Al iniciar tendrá que iniciar con las credenciales por defecto de la aplicación, estas pueden ser modificadas en cualquier momento.

Cuando inicie sesión se encontrará con el Panel Principal de Grafana, en este panel se indican algunos pasos básicos para continuar con la configuración de Grafana, como se visualiza en la Figura 16.

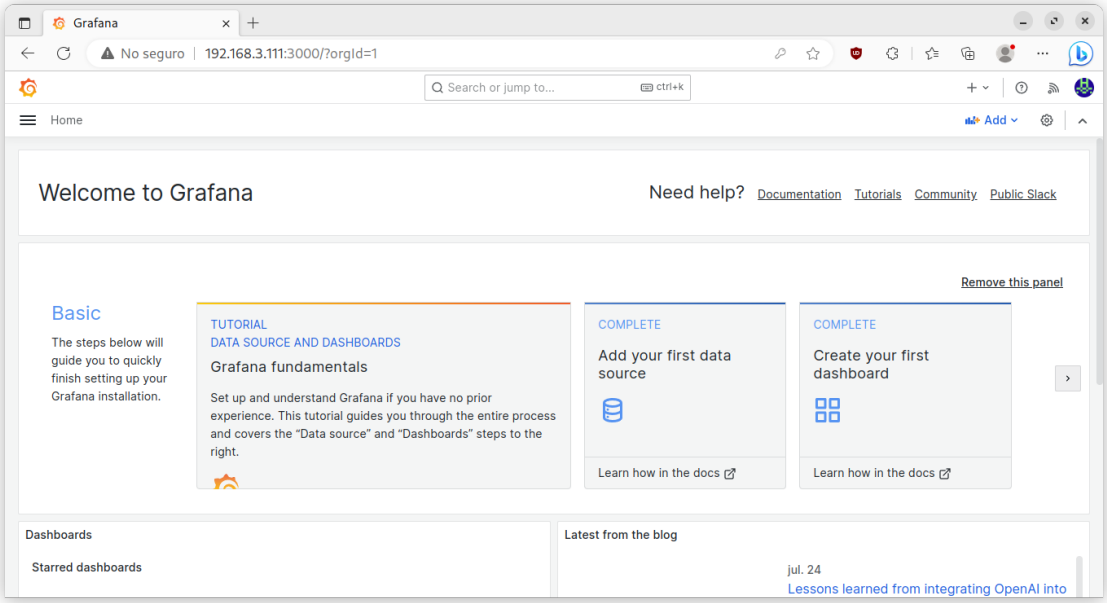


Figura 16: Vista inicial de Grafana

6.1.1 Configuración y conexión con InfluxDB

Para realizar la conexión con la base de datos en Grafana se debe crear una `Data source` dentro de las configuraciones de Grafana, vista en la Figura 17, estas corresponden a la definición de la conexión con la base de datos. Para esto anteriormente debe haberse levantado la base de datos InfluxDB (Anexo G) y haber realizado la configuración inicial que entregará las credenciales para continuar en este proceso.

La configuración para InfluxDB requiere de lo siguiente:

- Query Language: Desde la versión 2.0 InfluxDB funciona con un nuevo lenguaje de scripting para realizar consultas de manera eficiente, esto permitiendo extender las funcionalidades y entregar resultados mas directos sin tener que ser procesados con otro software.

- URL: InfluxDB posee una API basada en el protocolo HTTP, se indica la direccion Host donde se encuentra la instancia de InfluxDB, en caso de utilizar Docker se puede indicar el nombre del contenedor siempre que se encuentren en la misma subred de Docker.
- Organization: InfluxDB realiza por defecto separaciones de organizaciones para sus datos. A la hora de realizar la configuración inicial se ingresa una por defecto.
- Token: Corresponde a la llave de seguridad para hacer uso de la API, InfluxDB permite crear Token con diferentes niveles de acceso para resguardar datos. En caso de Grafana lo importante es tener un Token con acceso de solo lectura (read only)
- Default Bucket: Son el equivalente a tablas en otras base de datos. Por defecto, utilizando un Token se puede utilizar para realizar lecturas a múltiples Buckets siempre que se posean los permisos, pero indicar uno por defecto permite validar el funcionamiento de la base de datos en Grafana.

La interfaz de Grafana ofrece la ventaja de agregar dinámicamente diversas fuentes de base de datos sin necesidad de reiniciar el servicio. Esta característica permite una configuración ágil y en tiempo real, facilitando la integración y validación de nuevas conexiones sin interrupciones. La flexibilidad de Grafana para adaptarse a cambios y experimentar con distintos datos la convierte en una poderosa herramienta de visualización para analistas y científicos de datos.

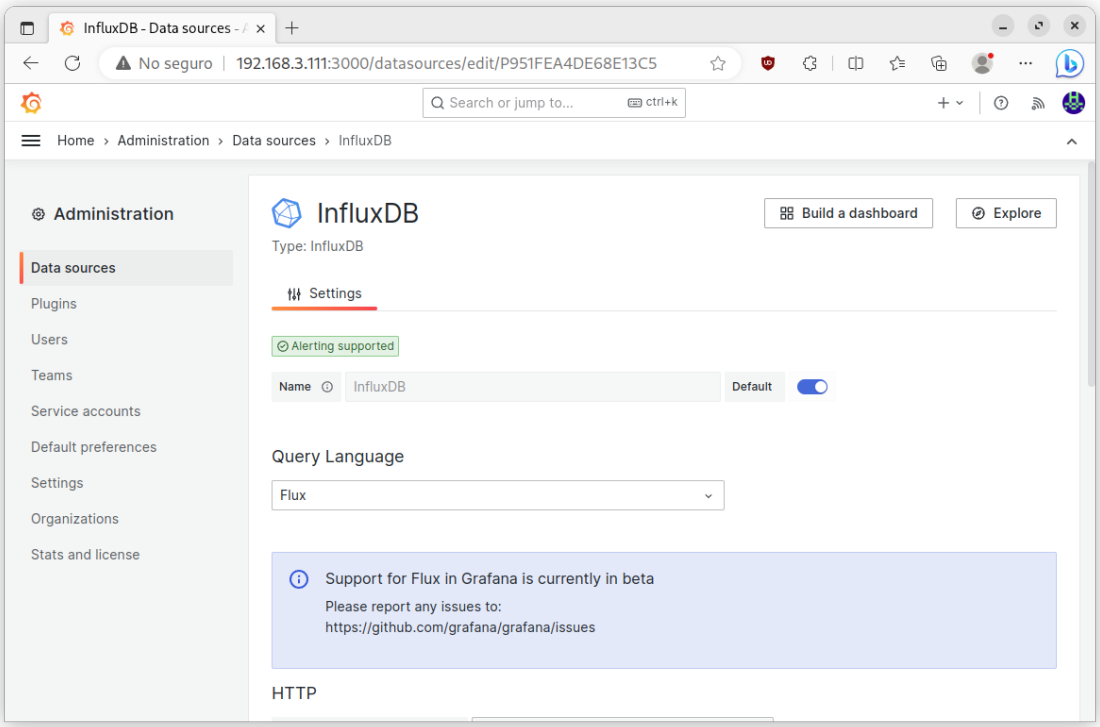


Figura 17: Interfaz de configuración de base de datos, Grafana

6.2 Desarrollo de Interfaz de Usuario

En esta sección se detallarán las consideraciones para el desarrollo de los diferentes paneles (dashboard) en la plataforma de Grafana, a esto se suma la elección de los diferentes componentes de Grafana, algunos pudiendo ser terceros.

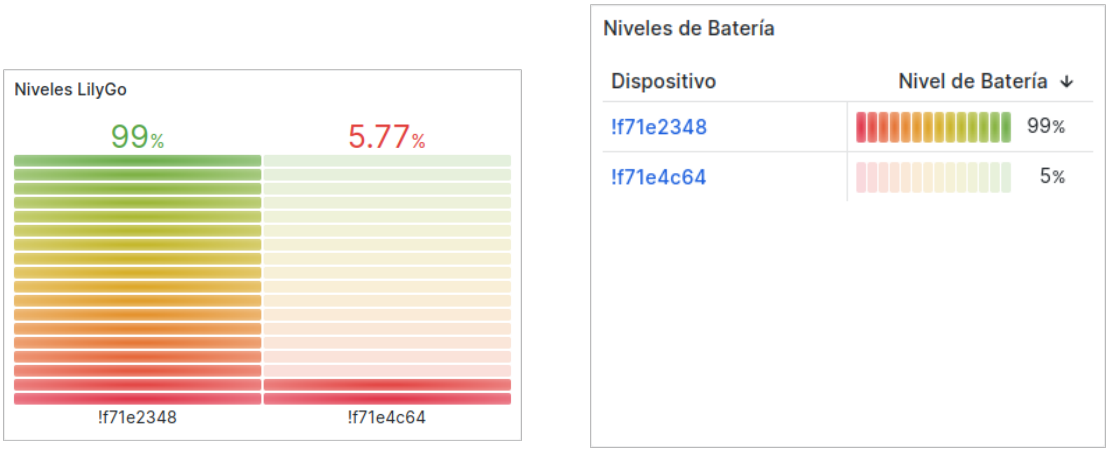
6.2.1 Panel General

En este dashboard, se prioriza la visualización de datos generales del sistema, como las últimas posiciones emitidas por las LilyGo T-Beam, así como sus niveles de batería.

Niveles de batería

En este panel es importante la visualización de los niveles de batería de cada uno de los LilyGo T-Beam móviles, incluyendo su ID dentro de la red mesh.

Dentro de las opciones provistas por los componentes de Grafana las opciones que cumplirían con la correcta visualización de datos serían los componentes **Bar Gauge** y **Table**, como se ven configurados en la Figura 18. Cada uno de estos componentes será capaz de mostrar las variables requeridas y de generar elementos nuevos según el crecimiento de la cantidad de dispositivos, esto disminuye la necesidad de estar creando nuevos Dashboards.



(a) *Bar Gauge* (b) *Table*

Figura 18: Componentes para representar el nivel de batería de los dispositivos.

Adicionalmente, Grafana permite agregar características al componente como la implementación de *panel links* dentro del mismo utilizando las variables devueltas por la base de datos o variables relacionadas con el panel general, esto permite en sí permite crear paneles dinámicos, de los que se hablaran en la sección Panel Individual.

Dado que ambas opciones, el *Bar Gauge* y la *Table*, satisfacen los requisitos necesarios para mostrar los niveles de batería de manera efectiva, la elección final dependerá de su capacidad para manejar un incremento en la cantidad de dispositivos dentro de la red mesh, además es importante no comprometer la claridad en la interpretación de la información.

Mapa de últimas ubicaciones

El propósito de este mapa consiste en mostrar de manera visual los puntos más recientes correspondientes a cada una de las posiciones registradas por los dispositivos LilyGo T-Beam. Esta representación proporciona un panorama general del estado de los dispositivos móviles en diversas ubicaciones, permitiendo obtener una comprensión completa de su distribución geográfica.

Para este mapa se utilizó el componente *Geomap*, como se visualiza en Figura 19, implementado en las últimas versiones de Grafana, este mapa utiliza el proyecto colaborativo OpenStreetMap para la visualización. Los datos requeridos desde la base de datos serían *latitud*, *longitud*, *altitud*, donde solo la latitud y la longitud son necesarios para señalar un punto en el mapa.

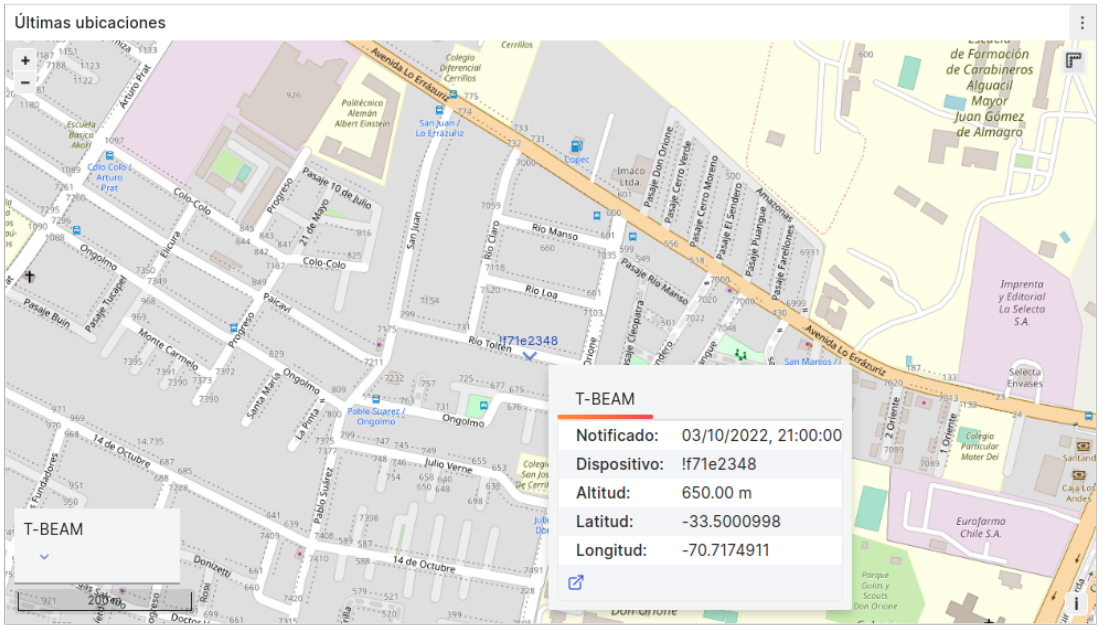


Figura 19: Componente Geomap mostrando posiciones desde la base de datos. (Por defecto se utilizan coordenadas WGS84 en formato decimal.)

Dispositivos en red mesh

Para este caso se utiliza el componente *tabla* dentro de los plug-ins de Grafana; como se visualiza en la Figura 20, en él aparecerán todos los dispositivos que estén transmitiendo información durante un periodo de tiempo acotado (10 min), utilizando el parámetro `airUtilTx` es posible validar que los dispositivos se encuentran activos en la red y por otra es un indicador de nivel posible congestión en el uso del tiempo de canal aun cuando no han emitido niveles de batería o posiciones.

Figura 20: Componente de Grafana para visualización de dispositivos activos dentro de la red mesh.

Panel General

Finalmente, estos elementos se unirán para conformar el panel general, armonizando la presentación de la información de manera visualmente agradable. En este contexto, se dispondrán los componentes de forma equilibrada, teniendo en cuenta tanto la claridad informativa como la estética general. En primer lugar, se visualizarán los niveles de batería de los dispositivos móviles, proporcionando una instantánea de su energía restante. Luego, se presentarán las ubicaciones más recientes de estos dispositivos en un mapa, ofreciendo una perspectiva geográfica de su distribución. Por último, se listará la totalidad de dispositivos en la red mesh. En la Figura 21, se

puede apreciar un ejemplo de la distribución, pero para efectos prácticos es suficiente para esta primera propuesta, ya que se cumple con lo explicado anteriormente.

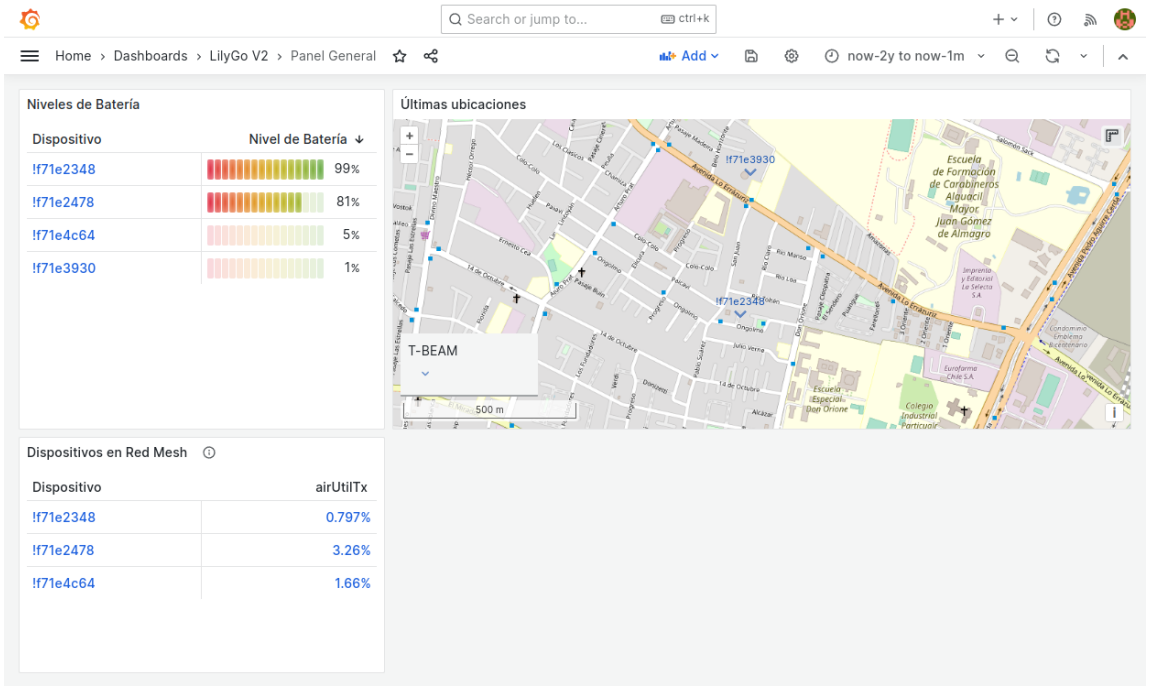


Figura 21: Panel General con todos sus componentes.

Panel Individual

A diferencia del panel general, el panel individual, como su nombre sugiere, se enfoca en presentar información específica de una LilyGo en particular. Este enfoque permite un análisis más profundo de los registros de ubicaciones, otorgando al usuario la capacidad de tomar decisiones fundamentadas en estos datos detallados.

Selección de Dispositivo

Más que ser simplemente un componente, consiste en una variable del panel que posibilita la selección de dispositivos sin la necesidad de regresar al panel general. Esta variable es un selector poblado con los identificadores de aquellos dispositivos que hayan registrado información en la base de datos de InfluxDB. Estas variables necesitan ser configuradas en la sección de ajustes específicos del panel en Grafana.

Para llevar a cabo este proceso, es necesario incorporar una consulta que permita obtener las variables correspondientes. Como alternativa, también existe la opción de generar variables con listados estáticos, simplificando la elección de dispositivos específicos.

La consulta Flux que sigue es empleada para obtener los identificadores desde InfluxDB, como se puede observar en la Figura 22.

```
1 import "influxdata/influxdb/schema"
2
3 schema.tagValues(bucket: "meshtastic", tag: "device")
```

Código 5: Consulta Flux para la obtención de identificadores desde InfluxDB

Figura 22: Selector de Variable dentro de Panel Individual.

Mapa de Ubicaciones

En este mapa, como se visualiza en la Figura 23, se representarán visualmente las ubicaciones que han sido registradas por el dispositivo. La consulta realizada a InfluxDB se encargará de seleccionar únicamente los puntos más recientes que hayan registrado una posición relevante dentro del periodo de tiempo específico elegido.

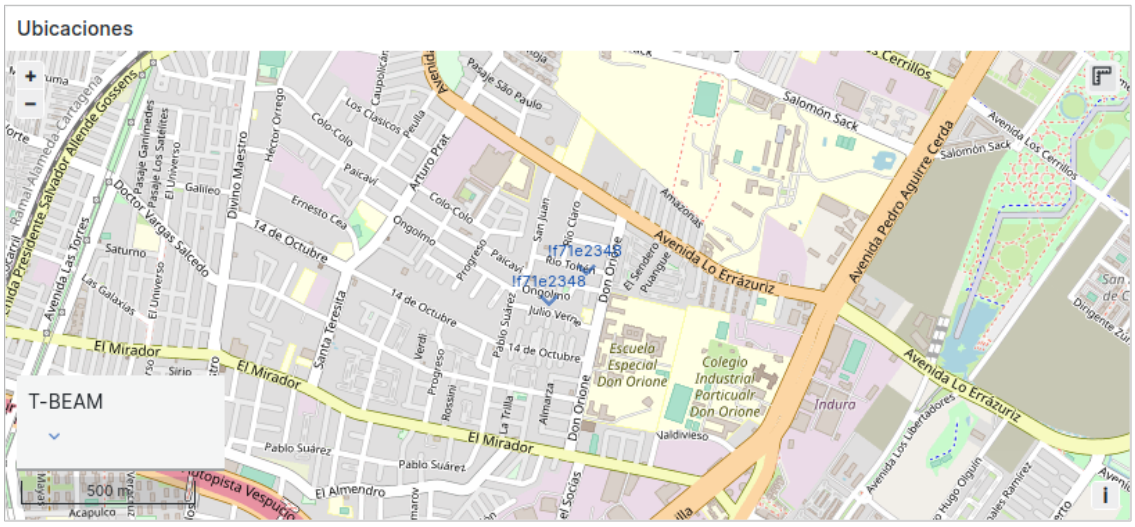


Figura 23: Componente Mapa para las ubicaciones del panel individual

Mapa de Seguimiento

Este mapa (ver Figura 24) utiliza todos los puntos disponibles en la base de datos para generar un seguimiento. Se traza una línea conectando cada uno de los puntos según su marca de tiempo, lo que brinda una representación visual del trayecto recorrido. Es importante tener en cuenta que esta representación de seguimiento es una estimación basada en los datos disponibles, y su precisión depende de la cantidad y la calidad de los datos registrados. La incorporación de funciones de predicción está más allá de los alcances de este trabajo de título, pero aun así desarrollar esto permitiría una visualización aún más lógica y precisa en casos futuros.

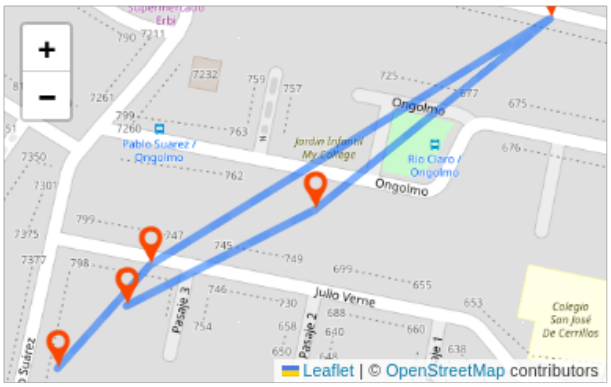


Figura 24: Componente Mapa de Seguimiento en el panel individual.

Tabla de Ubicaciones

Esta componente tabla, como se ve en la Figura 25, presenta un registro histórico de ubicaciones, detallando la longitud, latitud y altitud correspondientes. Además, proporciona la marca de tiempo en la cual el dispositivo notificó cada una de estas posiciones.

Ubicaciones

Notificado	Altitud	Latitud	Longitud
30/09/2022, 23:48:58	479.00 m	-33.5013943	-70.7198716
30/09/2022, 23:50:59	487.00 m	-33.5014286	-70.7198606
30/09/2022, 23:52:59	493.00 m	-33.5013618	-70.7198420

‹ 1 2 3 4 5 6 7 ··· 32 › 1 - 3 of 94 rows

Figura 25: Componentes tabla para el registro historico de ubicaciones.

Nivel de Batería

En este panel, como se ve en la Figura 26, se presentan dos componentes destinados a mostrar el nivel de batería. El primero de ellos indica el nivel de batería actual de un dispositivo móvil de manera instantánea, exhibiendo un porcentaje acompañado de una barra de colores que resalta de forma visualmente atractiva el estado actual de la batería.

Además, el segundo componente consiste en un gráfico que visualiza el historial del nivel de batería a lo largo del tiempo. Este gráfico proporciona una visión más completa del comportamiento de la batería tipo 18650 que está integrada en el dispositivo móvil.

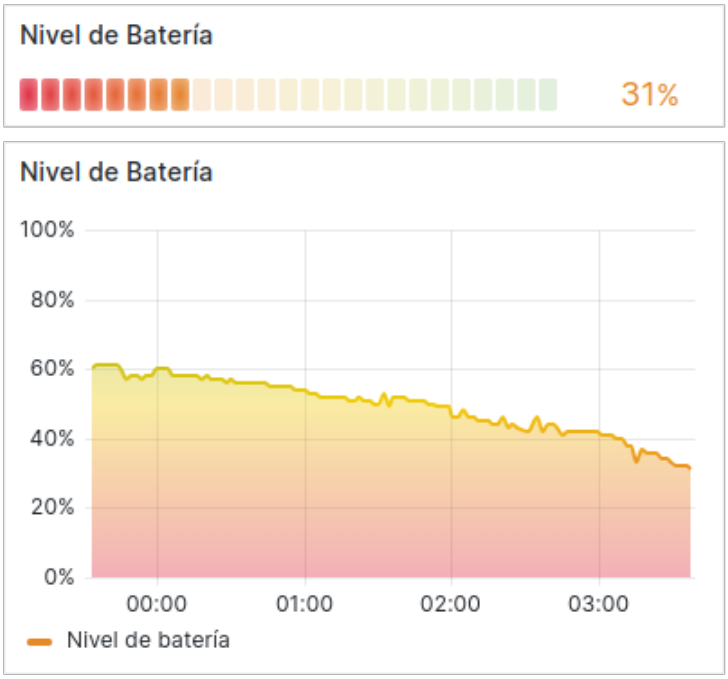


Figura 26: Componentes para el nivel de batería.

Panel Individual

Siguiendo la misma idea que en el panel general, es esencial disponer los componentes de manera que logren una estética armónica y presenten la información del dispositivo de forma clara. Todo esto debe realizarse sin abrumar al usuario, de manera que pueda navegar por la plataforma sin dificultad.

La inclusión de la variable de Dispositivo dentro del panel proporcionará la comodidad de no tener que retroceder para acceder a otros dispositivos disponibles. El Mapa de Ubicaciones permitirá visualizar las últimas ubicaciones registradas por los usuarios, mientras que el mapa de seguimiento permitirá examinar con mayor precisión el historial de ubicaciones. Finalmente, se puede apreciar la inclusión de todos los componentes previamente descritos en la Figura 27.

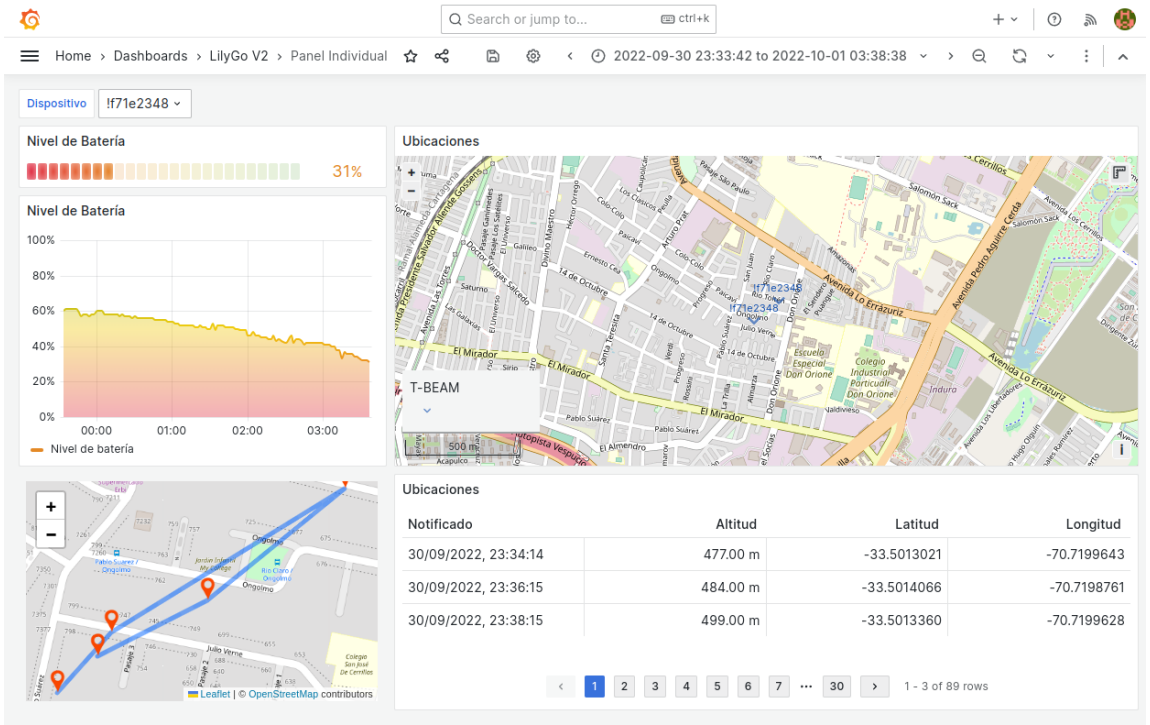


Figura 27: Panel Individual con todos los componentes asociados a un mismo dispositivo.

Capítulo 7

Pruebas en terreno

En este capítulo, se presenta la ejecución de pruebas en terreno para el proyecto “Mejoramiento de la experiencia turística mediante un sistema físico-digital en áreas silvestres protegidas de la Región Metropolitana” en la Reserva Nacional Río Clarillo. Este proyecto, resultado de la colaboración entre UTEM y CONAF, tiene como objetivo abordar las problemáticas de los guardaparques y mejorar la experiencia turística en el área protegida.

Además, se utilizarán los datos recopilados durante las pruebas en terreno para analizar el rendimiento y la eficacia de los dispositivos. Estas pruebas de los dispositivos móviles concluyeron con una duración promedio de batería de hasta 36 horas, aunque su variabilidad depende de la interacción del usuario y junto con las condiciones del terreno. Las ubicaciones geográficas demostraron un nivel satisfactorio de precisión (10m), con algunos errores detectados en situaciones específicas debido a la disponibilidad de satélites.

7.1 Implementaciones

Para llevar a cabo estas pruebas, se seleccionó la Reserva Nacional Río Clarillo como el área de estudio. Este trabajo se enmarca dentro del Fondo de Innovación para la Competitividad Regional (FIC-R).

El proyecto FIC, denominado “Mejoramiento de la experiencia turística mediante un sistema físico-digital en áreas silvestres protegidas de la Región Metropolitana” Código BIP 40026822-0, surgió de una colaboración entre dos partes: UTEM y CO-

NAF siendo este último el beneficiario principal del proyecto.

Dentro de los alcances de este proyecto, se tuvo en consideración las problemáticas que enfrentan los guardaparques en su labor diaria. La implementación de este sistema de seguimiento se concibió como una potencial solución directa, que podría proporcionar una herramienta valiosa para la administración y el mantenimiento de la Reserva Nacional. Asimismo, este sistema tiene el potencial de contribuir a la mejora de la experiencia turística en el área protegida, impulsando el turismo de manera efectiva.

A continuación se mostraran las implementaciones de este sistema realizadas dentro de la Reserva Nacional Río Clarillo.

7.1.1 Oficina de Administración

En el contexto del Proyecto FIC, se procedió a instalar un servidor en las instalaciones de la oficina de administración de CONAF, instalación mostrada en la Figura 28. Se utilizará como base para implementar toda la infraestructura necesaria, incluyendo Grafana e InfluxDB. Además, cumplirá el papel de ser el Gateway receptor de toda la red involucrada en el sistema.

Dado que no se dispone de una conexión directa a este servidor para efectuar cambios de manera remota, se ha optado por la utilización de ZeroTier. Esta herramienta permitirá establecer un acceso remoto mediante la creación de una red virtual.

Figura 28: Rack instalado en Oficina de Administración, Río Clarillo

7.1.2 Centro de Información Ambiental (CIA)

En la cúspide de este punto, se encuentra un mirador que ha servido como ubicación ideal para la instalación de una torre galvanizada, fotografía en la Figura 29, en la cual se han montado diversas antenas. Dentro de esta área, se ha colocado una Estación Repetidora para potenciar las comunicaciones. Además, se tiene previsto aprovechar esta estructura para instalar una antena de mayor ganancia ($5dBi$).

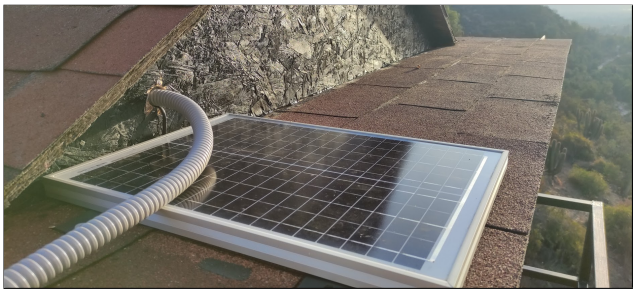
Un aspecto particularmente beneficioso es que esta ubicación dispone de suministro eléctrico, lo cual ha eliminado la necesidad de emplear paneles solares y baterías para el abastecimiento energético. De todas maneras, tratándose de un sector rural cuyo abastecimiento de energía eléctrica es muy inestable, resulta muy aconsejable el uso de estabilizadores y/o reguladores de voltaje para evitar daño o desconfiguraciones en los dispositivos microelectrónicos.

Figura 29: Torre en Centro de Información Ambiental, Río Clarillo

7.1.3 Torre de Vigilancia

Esta ubicación está designada como un punto de vigilancia crucial para los guarda-parques. La particularidad de esta estructura reside en su capacidad de admitir la instalación de una estación repetidora. Para ello, se implementará un sistema fotovoltaico completo, el cual incluirá un regulador de carga, una batería y paneles solares. Este enfoque asegurará un suministro de energía constante y sostenible para la operación de la estación repetidora en esta torre de vigilancia. Instalaciones visibles en la Figura 30.

(a) *Módulo solar*



(b) *Panel Solar*

Figura 30: Instalación de estación repetidora con módulo solar en Torre de Vigilancia, Río Clarillo.

7.1.4 Dispositivos Móviles

Los dispositivos móviles empleados en este proyecto están equipados con baterías de litio tipo 18650, que proporcionan la capacidad de alimentación necesaria para su funcionamiento. Estas baterías pueden recargarse mediante la conexión de un cable microUSB, lo que facilita la recarga y el mantenimiento.
Para garantizar la protección y el funcionamiento óptimo de estos dispositivos, se diseñó y fabricó una carcasa específica utilizando tecnología de impresión 3D. Esta carcasa, elaborada con materiales resistentes y duraderos, ofrece una cubierta protectora que asegura la integridad de los dispositivos móviles en diversos entornos y condiciones.

Figura 31: Dispositivo Móvil con su respectiva *Case* en Reserva Nacional Río Clarillo.

7.2 Resultados

Es fundamental resaltar la importancia de las pruebas en terreno como un paso crucial para validar el sistema propuesto. No obstante, es necesario tener en cuenta las limitaciones geográficas que han impactado las oportunidades para llevar a cabo múltiples pruebas en el área de estudio, específicamente en Río Clarillo. Estas pruebas se han restringido a las salidas contempladas en el marco del Proyecto FIC.
A pesar de las restricciones geográficas que han influido en la cantidad de pruebas realizadas, es esencial reconocer que estas pruebas han generado información sumamente valiosa. Esta información no solo ha permitido comprender las capacidades tanto del software como del hardware, sino también ha manifestado el potencial de este proyecto.

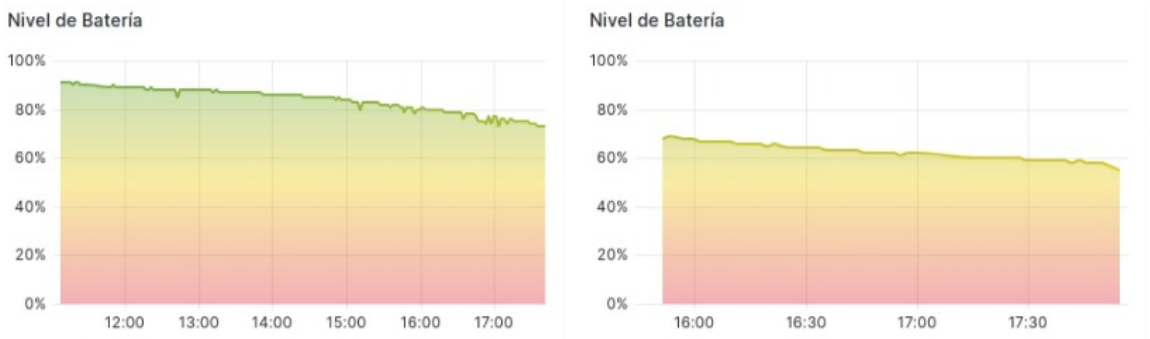
En este sentido, los diversos registros recopilados abarcan:

- Niveles de batería
- Ubicaciones geográficas (Latitud y Longitud)
- Altitud
- Número de satélites

Niveles de batería

Con los mismos parámetros de configuración que se utilizaron durante pruebas PtP (Point to Point), se determinó mediante el análisis de los datos de funcionamiento que la batería del dispositivo es capaz de mantener el funcionamiento desde el encendido durante un período continuo promedio de hasta 36 horas. Sin embargo, es crucial tener en cuenta que esta duración está sujeta a la interacción y manipulación del usuario. La realización de acciones como la presión de botones o el encendido y apagado frecuente del dispositivo puede reducir el intervalo de funcionamiento efectivo.
En la Figura 32, se puede apreciar la gráfica de los niveles de baterías de dos dispositivos, en estas se representa el decaimiento del nivel de batería con el transcurso de las horas.

Además de la influencia directa del usuario, existen otros factores que deben considerarse. La vida útil de la batería es un aspecto importante que puede afectar la



(a) *Dispositivo !f71e49d4* (b) *Dispositivo !f71e4c64*

Figura 32: Niveles de batería durante pruebas de funcionamiento en Río Clarillo.

capacidad de mantenimiento de energía a lo largo del tiempo. Asimismo, el estado de congestión de la red mesh también puede tener un impacto en la duración de la batería, ya que la comunicación constante puede requerir un mayor consumo energético. Es necesario destacar que, si bien se logró identificar un intervalo aproximado de funcionamiento de hasta 36 horas, la realidad operativa podría variar debido a la interacción del usuario y las diversas condiciones ambientales y de uso.

Ubicaciones

Durante las pruebas en terreno, se enfrentó la limitación de no poder realizar evaluaciones extensivas, específicamente aquellas que simularían el movimiento de una persona caminando a una velocidad normal. Esta restricción se debió a la realización simultánea de actividades relacionadas con el Proyecto FIC. A pesar de esta circunstancia, las ubicaciones registradas presentaron un nivel satisfactorio de precisión, lo que permitió capturar los datos sin mayores contratiempos, en la Figura 33 se aprecian algunas de las ubicaciones capturadas durante el transcurso de las pruebas. Es relevante resaltar que, si bien se han observado algunos errores en los registros de ubicación, estos errores parecen coincidir con situaciones en las que el módulo GPS detectó la existencia de un número limitado de satélites. Esto pudo afectar la trilateración precisa y, por ende, generar desviaciones en la precisión de las ubicaciones registradas.

No obstante, es importante resaltar que para fines prácticos, estos datos pueden someterse a un proceso de filtrado posterior y/o pueden ser mejorados mediante la implementación de algoritmos específicos. El uso de técnicas de post-procesado en

la base de datos puede ayudar a reducir la tasa de error y mejorar la calidad de las ubicaciones registradas.

(a) *Dispositivo !f71e49d4*

(b) *Dispositivo !f71e4c64*

Figura 33: Ubicaciones durante pruebas de funcionamiento Río Clarillo.

Altitud

En relación con la altitud, es importante destacar que se han identificado ciertos errores en situaciones específicas, tal como se mencionó en la sección anterior.
Este error en particular se refiere al incremento de la altitud sobrepasando las condiciones geográficas de Río Clarillo. (línea verde)
Estos errores están contemplados y documentados tanto en la hoja técnica de los módulos GPS utilizados como en la documentación proporcionada por Meshtastic.
La Figura 34 presenta una comparativa entre dos dispositivos: **!71e2348** y **!71e4c64**. Ambos dispositivos son modelos LilyGo T-Beam, sin embargo, se diferencian en los módulos LoRa y GPS que incorporan. Específicamente, el primer dispositivo utiliza el módulo GPS NEO 6M, que solo ofrece soporte para GPS, mientras que el segundo dispositivo utiliza el módulo NEO M8N, el cual incluye soporte para GPS, GLONASS, Galileo y BeiDou. Esta mejora en la versión avanzada del LilyGo T-Beam resulta en una mayor precisión y una reducción de los errores en la determinación de la altitud.

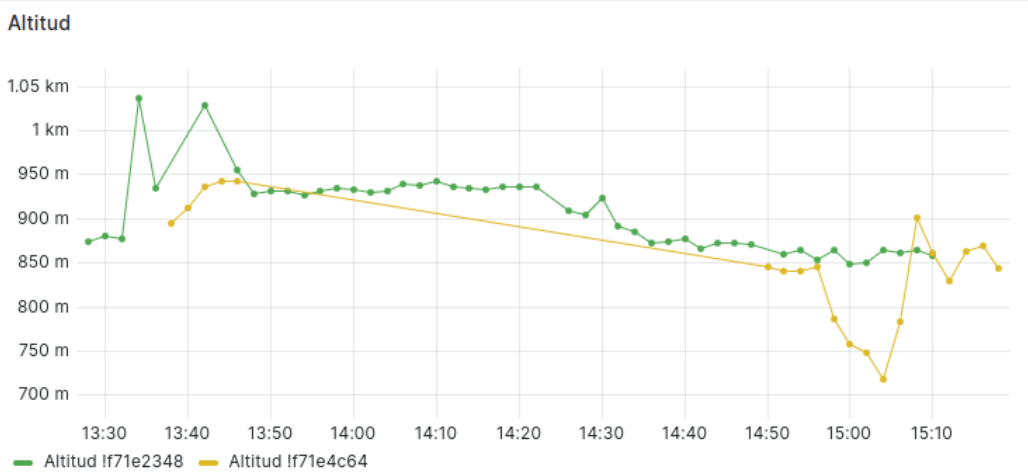


Figura 34: Comparativa de Altitud entre dispositivos con diferentes módulos GPS

Número de satélites

Considerando la Figura 35 y tomando en cuenta que ambos dispositivos difieren en sus módulos GPS, utilizando como referencia el mismo período de análisis en el que uno de los dispositivos experimentó un incremento visible en los errores en la altitud, solo tenía en cuenta 4 satélites a la vista al momento de realizar el cálculo de

posición. Sin embargo, en contraste, el segundo dispositivo, equipado con el módulo GPS mejorado, captó 5 satélites, lo que contribuyó a reducir el error en la altitud.

En un período posterior en el que ambos dispositivos funcionaron simultáneamente, al final de la gráfica se observa que el dispositivo con la actualización del módulo GPS fue capaz de detectar hasta 12 satélites en vista. Mientras tanto, el primer dispositivo solo logró un máximo de 8 (sin considerar el período anterior en el que obtuvo 9 debido a la separación de los equipos de trabajo en terreno). Estos resultados indican que la incorporación de múltiples tecnologías puede aumentar la precisión, especialmente cuando las ubicaciones de los satélites de posicionamiento no se encuentran en el hemisferio de trabajo.

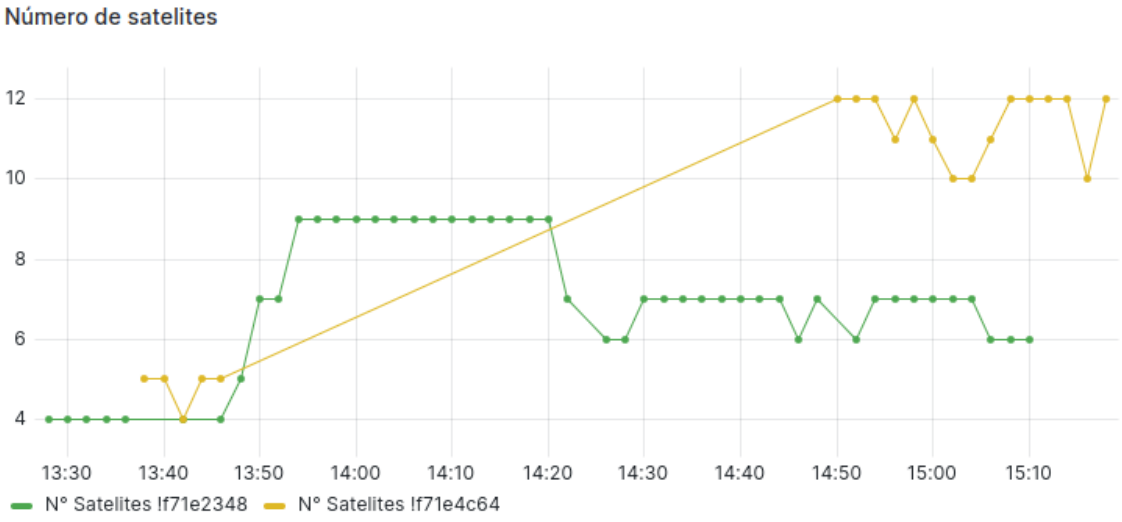


Figura 35: Comparativa de número de satélites detectados (en vista) entre dispositivos con diferentes módulos GPS

Capítulo 8

Conclusiones y Trabajos futuros

En este capítulo se presentan las conclusiones derivadas del uso de tecnología preexistente (LilyGo T-Beam y Meshtastic 1.3.41). Se resaltan las características del enfoque adoptado al momento de desarrollar, permitiendo la escalabilidad del sistema y la facilidad de uso de Grafana. La implementación en el terreno enfrentó desafíos logísticos. Además se considera la seguridad de datos mediante el formato *protobuf* provista por Meshtastic y la capa de seguridad por parte de la base de datos InfluxDB.

En cuanto a trabajos futuros, se sugiere extender la cobertura de red mediante diversas pruebas, mejorar la experiencia de los visitantes con informes y rutas, ampliar la funcionalidad de comunicación, agregar diferentes tipos de dispositivos y crear un sistema de sensores y actuadores distribuidos en la red mesh.

8.1 Conclusiones

Respecto del *OE*1., es posible señalar que se tomaron en cuenta los diferentes aspectos técnicos tanto del hardware (LilyGo T-Beam) como del software (Meshtastic) en su versión 1.3.41. Esto posibilitó agilizar los procesos de desarrollo y generar una infraestructura basada en la tecnología preexistente. Además, se consideraron las problemáticas señaladas en la comunidad de Meshtastic, utilizando estas experiencias para evitar la repetición de los distintos inconvenientes, tales como la demora en la obtención de la posición en los LilyGO T-Beam, así como la implementación de ajustes para optimizar el progreso del desarrollo en un entorno controlado. Dentro de este enfoque, también se incluyó la revisión constante de la documentación disponi-

ble en el proyecto Meshtastic junto con la documentación de la API. Los requisitos funcionales básicos, a su vez, fueron establecidos sobre la base de entrevistas con potenciales usuarios en el entorno del parque nacional utilizado como escenario del pilotaje a escala real.

En relación al *OE*2., con el propósito de garantizar la escalabilidad del sistema en todo el proceso de desarrollo, se adoptó un enfoque modular. Este enfoque permite la incorporación de nuevos dispositivos al sistema o la introducción de nuevos servicios sin afectar la coexistencia de los dispositivos ya existentes. Además, el uso de servicios y conexiones a través de diversas API brinda la posibilidad de crear futuras aplicaciones, como la generación de informes y análisis de información en tiempo real mediante la detección de la inserción de datos en la base de datos. La precisión georreferenciada obtenida en las pruebas arrojó una precisión con una deriva inferior a 10 metros lo cual, teniendo en cuenta el objetivo de la aplicación piloteada, constituye un valor de precisión adecuado a efectos de, por ejemplo, posicionar a una persona en el contexto de los senderos del parque y trackear su desplazamiento.

Respecto de lo planteado en el *OE*3., aunque se optó por la herramienta de código abierto Grafana, el desarrollo de los distintos componentes para cada tipo de panel dentro de esta herramienta permitió ofrecer una representación clara de los elementos que el usuario en general esperaría y debería poder interpretar de manera sencilla e intuitiva. Esta simplicidad en la experiencia del usuario aumenta la afinidad con el prototipo desarrollado, el cual por supuesto puede ser mejorado iterativamente aún más a través de metodologías de evaluación de usabilidad.

Por otro lado, en relación al *OE*4., realizar pruebas en el entorno del parque nacional Río Clarillo presentó un desafío, no solo debido a las grandes distancias entre los puntos, sino también a la ubicación remota y alejada de la llamada “civilización”. Por lo tanto, planificar meticulosamente antes de llevar a cabo implementaciones en el terreno resultó crucial. Esto es especialmente importante ya que, en caso de faltar algún implemento, insumo o material de carácter técnico específico, la ejecución en terreno tendría que ser pospuesta hasta que se disponga de todo lo necesario, generando costos adicionales en la implementación. Estos costos podrían abarcar elementos como mano de obra, transporte, herramientas, entre otros. Asimismo, se debe tener en cuenta que durante la implementación podrían surgir cambios, como el caso ocurrido en el Centro de Información Ambiental, donde el Sistema Fotovoltaico

fue sustituido al contar con red de energía eléctrica de baja tensión en el área, pero al costo de perder la estabilidad y continuidad que podría haber provisto un buen sistema de respaldo fotovoltaico y de enfrentar constantes cortes o inestabilidad en la energía eléctrica del sector.

Finalmente, sobre los aséctos de seguridad de los datos y del sistema mismo planteados en el *OE*5., es posible indicar que dado que el proyecto Meshtastic trabaja con el formato *protobuf*, se requiere que el usuario esté familiarizado con el protocolo y el tipo de mensaje para llevar a cabo una correcta desencriptación. Además, la base de datos InfluxDB brinda una capa adicional de seguridad para los datos. Esta capa se encuentra protegida mediante un token y, además, es posible integrar seguridad SSL mediante la utilización de certificados de seguridad y la gestión de puertos seguros para la interacción con la API. Se agrega, además, que para la comunicación entre Grafana e InfluxDB, toda la información compartida entre estos elementos se realiza a través de un adaptador de red privada generado mediante Docker.

8.2 Trabajos futuros

Dentro de los márgenes de este trabajo de título se establecieron diversos límites con el fin de salvaguardar los tiempos de desarrollo al mínimo, asegurando así la creación de un prototipo viable y escalable.

En caso de continuar con esta línea de investigación aplicada y desarrollo tecnológico, podrían definirse las siguientes áreas de trabajo para complementar lo desarrollado en este trabajo de título, entre las cuales se incluyen:

- Realizar pruebas de alcance y cobertura en la red mesh LoRa implementada, además de generar un mapa que identifique las zonas sin cobertura dentro del parque, con el objetivo de incorporar estaciones repetidoras equipadas con sistemas fotovoltaicos.

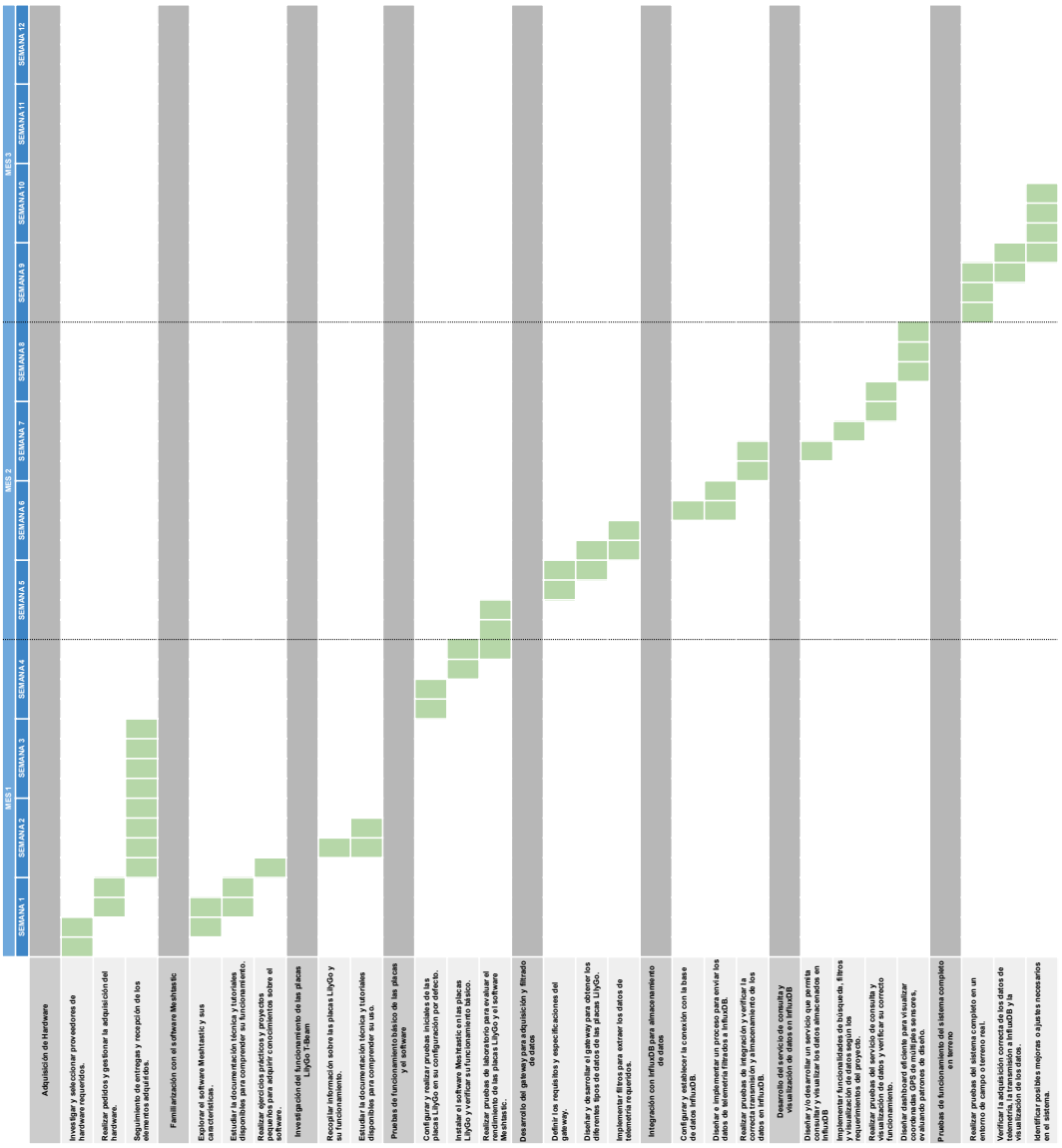
- Implementar la detección de dispositivos LilyGo en tiempo real a través de la actualización a la versión 2.0 de Meshtastic, permitiendo el rastreo indirecto mediante el seguimiento de mensajes por medio de un sistema de ping en la red mesh.

- Generar informes y proporcionar rutas de seguimiento para los visitantes, con el propósito de mejorar la experiencia turística en el parque.
- Ampliar las capacidades del prototipo inicial, permitiendo el envío de mensajes desde los terminales portátiles a los guardaparques. Además, integrar la funcionalidad de recibir alertas en los dispositivos para lograr una comunicación bidireccional, ya sea a través de indicaciones visuales en pantalla u otros métodos de interacción con el usuario.
- Integrar otros dispositivos compatibles con Meshtastic (por ejemplo brazaletes), lo cual reduciría costos en situaciones en las que no se requiera el uso de LilyGo T-Beam para la función de posicionamiento.
- Establecer un sistema distribuido de sensores y actuadores utilizando la red mesh preexistente. Esto permitiría el control y la creación de un sistema interconectado de sensores para, por ejemplo, la detección de posibles incendios, medición del caudal de agua del río o nivel de agua en estanques distribuidos por el parque, entre otros escenarios.

Anexos

Anexo A

Carta Gantt



Anexo B
Obtención de Eventos Serial Asíncrono

```python
import os
import sys
from signal import SIGINT, signal
from time import sleep, time
from meshtastic.serial_interface import
   SerialInterface
from pubsub import pub
from rich import print

PATH = "./events"


def get_name_file(portnum):
    timestamp = int(time())
    name_file = f"{PATH}/{portnum}_{timestamp}"
    i = 2
    while os.path.exists(name_file):
        name_file = f"{name_file}_{i}"
        i += 1
    return name_file

def save_event(packet):
    portnum = packet["decoded"]["portnum"]
    name_file = get_name_file(portnum)
    with open(f"{name_file}.txt", "w") as archivo:
        archivo.write(str(packet))

def onReceive(packet, interface):
    print(f"Tipo de mensaje: {packet['decoded']['
       portnum']}")
    print(f"Payload: {packet['decoded']['payload']}")
    save_event(packet)

pub.subscribe(onReceive, "meshtastic.receive")
signal(SIGINT, lambda signal, frame: sys.exit(0))

interface = SerialInterface()

while True:
    sleep(0.1)
```

Anexo C

Comandos de configuración LilyGo para pruebas PtP

```
# Configuracion Meshtastic Pruebas PtP

# Habilitación GPS
meshtastic --set position.gps_enabled true

# Intervalo de tiempo entre solicitud de actualizacion
# posicion (s)
meshtastic --set position.gps_update_interval 120

# Intervalo de tiempo entre cada intento de actualizacion
# de posición (s)
meshtastic --set position.gps_attempt_time 120

# Intervalo de emision de posicion a la red mesh (s)
meshtastic --set position.position_broadcast_secs 120

# Solo se enviara la posicion si ha sido un cambio relevante
meshtastic --set position.position_broadcast_smart_enabled true

# Definicion de que valores se emiten durante la
# transmision de mensajes en la red mesh

meshtastic --pos-fields ALTITUDE ALTITUDE_MSL SATINVIEW TIMESTAMP
```

Anexo D
Configuración de Position Flags

Index	Valor	Descripción
0	UNSET	Requerido para la compilación
1	ALTITUDE	Incluir un valor de altitud (si está disponible)
2	ALTITUDE_MSL	El valor de altitud es MSL
3	GEOIDAL_SEPARATION	Incluir separación geoidal
4	DOP	Incluir el valor DOP; PDOP utilizado por defecto, ver más abajo
5	HVDOP	Si se establece POS_DOP, enviar valores separados de HDOP / VDOP en lugar de PDOP
6	SATINVIEW	Incluir el número de "satélites en vista"
7	SEQ_NO	Incluir un número de secuencia incrementado por paquete
8	TIMESTAMP	Incluir marca de tiempo de posición (desde la solución GPS)
9	HEADING	Incluir rumbo de posición (desde la solución GPS)
10	SPEED	Incluir velocidad de posición (desde la solución GPS)

Anexo E

Tipado de datos recibidos a través de puerto serie Meshtastic

```python
from pydantic import BaseModel
from datetime import datetime

class Position(BaseModel):
    time: datetime | None = None
    latitude: float | None = None
    longitude: float | None = None
    altitude: int | None = None
    altitudeHae: int | None = None
    satsInView: int | None

    class Config:
        json_encoders = {datetime: lambda v: datetime.
            fromtimestamp(v)}

class DeviceMetrics(BaseModel):
    batteryLevel: int | None
    voltage: float | None
    airUtilTx: float
    channelUtilization: float | None

class Telemetry(BaseModel):
    time: datetime | None
    deviceMetrics: DeviceMetrics | None

    class Config:
        json_encoders = {datetime: lambda v: datetime.
            fromtimestamp(v)}

class Decoded(BaseModel):
    portnum: str
    payload: bytes
    position: Position | None
    text: str | None
    telemetry: Telemetry | None

class BasePacket(BaseModel):
    decoded: Decoded
    rxTime: datetime | None
    rxSnr: float | None
    fromId: str | None
    toId: str | None

    class Config:
        json_encoders = {datetime: lambda v: datetime.
            fromtimestamp(v)}
```

Anexo F

`docker-compose.yml` para la declaración declaración del servicio de Grafana

```yaml
version: "3.6"

services:
  grafana:
    container_name: grafana
    image: grafana/grafana
    volumes:
      - ${PWD}/data:/var/lib/grafana
      - ${PWD}/config.ini:/etc/grafana/grafana.ini
      - ${PWD}/provisioning:/etc/grafana/provisioning
      - ${PWD}/provisioning/plugins:/etc/grafana/provisioning/
          plugins
    networks:
      - internal
    restart: unless-stopped
    ports:
      - 3000:3000
    user: "0"
    logging:
      driver: json-file
      options:
        max-size: 5k

networks:
  internal:
    name: internal
    external: true
```

Anexo G

`docker-compose.yml` para levantar servicio de InfluxDB

```yaml
version: '3.9'

services:
  influxdb:
    container_name: influxdb
    image: influxdb:latest
    volumes:
      - ${PWD}/data:/var/lib/influxdb2
    networks:
      - internal
    restart: unless-stopped
    logging:
      driver: json-file
      options:
        max-size: 5k
    ports:
      - 8086:8086

networks:
  internal:
    name: internal
    external: true
```

Bibliografía

[1] H. Huh y J. Y. Kim, "LoRa-based Mesh Network for IoT Applications", en *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, págs. 524-527. DOI: 10.1109/WF-IoT.2019.8767242.

[2] J. Fox, A. Donnellan y L. Doumen, "The deployment of an IoT network infrastructure, as a localised regional service", en *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, págs. 319-324. DOI: 10.1109/WF-IoT.2019.8767188.

[3] M. Liya y M. Aswathy, "LoRa technology for Internet of Things(IoT):A brief Survey", en *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020, págs. 8-13. DOI: 10.1109/I-SMAC49090.2020.9243449.

[4] C.-W. Liang, Y.-L. Wu, C.-Y. Shi, S.-M. Lu y H.-C. Lee, "Poster Abstract: Evaluation of a LoRa Mesh Wireless Networking System Supporting Time-Critical Transmission and Data Lost Recovery", en *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2019, págs. 317-318. DOI: 10.1145/3302506.3312607.

[5] G. R. Hiertz, D. Denteneer, S. Max et al., "IEEE 802.11s: The WLAN Mesh Standard", *IEEE Wireless Communications*, vol. 17, n.º 1, págs. 104-111, 2010. DOI: 10.1109/MWC.2010.5416357.

[6] R. K. Kodali, K. Y. Borra, S. S. G. N. y H. J. Domma, "An IoT Based Smart Parking System Using LoRa", en *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2018, págs. 151-1513. DOI: 10.1109/CyberC.2018.00039.

[7] A. O. Silalahi, A. Sinambela, J. T. Pardosi y H. M. Panggabean, “Automated Water Quality Monitoring System for Aquaponic Pond using LoRa TTGO SX1276 and Cayenne Platform”, en *2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM)*, 2022, págs. 1-6. DOI: 10.1109/ICOSNIKOM56551.2022.10034916.

[8] L. Y. Astutik y R. Yusuf, “Performance Study of LoRa IoT Technology”, en *2022 12th International Conference on System Engineering and Technology (ICSET)*, 2022, págs. 31-35. DOI: 10.1109/ICSET57543.2022.10011110.

[9] Q. Yu, H. Wang, Z. Lu y S. An, “Group-Based CSS Modulation: A Novel Enhancement to LoRa Physical Layer”, *IEEE Wireless Communications Letters*, vol. 11, n.º 3, págs. 660-664, 2022. DOI: 10.1109/LWC.2022.3140860.

[10] L. A. Lora Alliance, *Rp2-1.0.1 LoRaWANő regional parameters*, nov. de 2022. dirección: https://lora-alliance.org/resource_hub/rp2-101-lorawan-regional-parameters-2/.

[11] S. Di Martino, L. Fiadone, A. Peron, A. Riccabone y V. N. Vitale, *Industrial Internet of Things: Persistence for Time Series with NoSQL Databases*, 2019. DOI: 10.1109/WETICE.2019.00076.