

**Universidad Nacional del Este.  
Facultad Politécnica.**

---



**Aplicación web asistida por  
inteligencia  
artificial para diagnosticar  
fracturas en imágenes  
radiológicas caninas**

**Ana Cecilia Barreto Jara  
Matias Alejandro Garay Morinigo**

**Año 2025.**

**Universidad Nacional del Este.**  
**Facultad Politécnica.**

---

**Carrera Ingeniería de Sistemas.**  
**Cátedra Trabajo Final de Grado.**

# **Aplicación web asistida por inteligencia artificial para diagnosticar fracturas en imágenes radiológicas caninas**

**Por: Ana Cecilia Barreto Jara  
y Matias Alejandro Garay Morinigo**

**Profesor Orientador: Ing. Katia Andrea Ayala Díaz**

Trabajo final de grado presentado a la Facultad Politécnica de la Universidad Nacional del Este como parte de los requisitos para optar al título Ingeniero de Sistemas.

**Ciudad del Este, Alto Paraná. Paraguay.**

**Año 2025**

# Índice general

Índice de Figuras	xii
Índice de Tablas	xii
Acrónimos y símbolos	xii
<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	2
1.2. Definición del problema . . . . .	3
1.3. Objetivos . . . . .	3
1.3.1. Objetivo General . . . . .	3
1.3.2. Objetivos Específicos . . . . .	4
1.4. Hipótesis . . . . .	4
1.5. Justificación . . . . .	5
1.6. Delimitación del alcance del trabajo . . . . .	6
<b>2. Conceptos fundamentales, teoría y antecedentes</b>	<b>8</b>
2.1. Inteligencia Artificial . . . . .	8
2.1.1. Evolución histórica . . . . .	8
2.1.2. Concepto de inteligencia artificial . . . . .	9
2.1.3. Modelos de Inteligencia Artificial . . . . .	9
2.2. <i>YOLO</i> ( <i>You Only Look Once</i> ) . . . . .	9
2.2.1. Concepto . . . . .	9
2.2.2. Características . . . . .	10
2.2.3. Arquitectura de YOLOv11 . . . . .	10
2.2.4. Arquitectura de YOLOv12 . . . . .	11
2.3. Visión por Computadora . . . . .	13
2.3.1. Evolución histórica . . . . .	13
2.3.2. Concepto . . . . .	14
2.3.3. Fases de un Sistema de Visión por Computadora . . . . .	14
2.4. Radiografías . . . . .	15

2.5.	Fracturas óseas . . . . .	16
2.6.	Diagnóstico por imágenes en veterinaria . . . . .	16
2.7.	Anatomía Ósea Canina . . . . .	17
2.7.1.	Húmero . . . . .	17
2.7.2.	Radio y Cúbito . . . . .	18
2.7.3.	Fémur . . . . .	18
2.7.4.	Tibia y Peroné . . . . .	18
2.8.	Clasificación de Fracturas . . . . .	18
2.8.1.	Sin Desplazamiento . . . . .	19
2.8.2.	Desplazada . . . . .	19
2.8.3.	Desplazada sin Punta . . . . .	19
2.8.4.	Desplazada con Punta . . . . .	19
2.8.5.	Desplazada con Fragmento . . . . .	20
2.9.	Localización Anatómica . . . . .	20
2.9.1.	Tercio Proximal . . . . .	20
2.9.2.	Tercio Medio . . . . .	20
2.9.3.	Tercio Distal . . . . .	21
2.10.	Aplicación web . . . . .	21
2.11.	Herramientas utilizadas . . . . .	21
2.12.	Antecedentes . . . . .	22
2.12.1.	Detección de fracturas en radiografías de cadera mediante el uso de inteligencia artificial (Año 2024) . . . . .	22
2.12.2.	Detection of Hand Bone Fractures in X-ray Images using Hybrid YOLO NAS (Año 2024) . . . . .	23
2.12.3.	DW-YOLO: Improved YOLO for Bone Fracture Detection (Año 2024) . . . . .	24
2.12.4.	Inteligencia Artificial en Radiología (Año 2021) . . . . .	25
2.12.5.	Implementation of Personal Protective Equipment Detection Using Django and Yolo Web at Paiton Steam Power Plant (PLTU) (Año 2023) . . . . .	27
<b>3.</b>	<b>Tecnologías utilizadas</b> . . . . .	<b>28</b>
3.1.	Componentes de <i>Hardware</i> . . . . .	28
3.1.1.	Placa de Desarrollo WiFi LoRa 32 (V3) . . . . .	28
3.1.2.	Lector RFID MFRC522 . . . . .	31
3.1.3.	Módulo GNSS GP-02 Kit . . . . .	32
3.1.4.	Placa de Desarrollo ESP-WROOM-32 . . . . .	33
3.2.	Componentes de <i>Software</i> . . . . .	34
3.2.1.	<i>Helium Console</i> . . . . .	34
3.2.2.	Thingsboard . . . . .	36
3.2.3.	Arduino IDE . . . . .	40

3.2.4. Fritzing . . . . .	41
3.2.5. Blender . . . . .	42
<b>4. Método</b>	<b>44</b>
4.1. Definición de arquitectura . . . . .	44
4.1.1. Requisitos clave . . . . .	45
4.1.2. Estructura interna del prototipo . . . . .	45
4.1.3. Arquitectura de comunicación externa . . . . .	46
4.2. Módulo de autenticación y control . . . . .	47
4.2.1. Requisitos del sistema . . . . .	47
4.2.2. Conexión del módulo <i>RFID</i> . . . . .	48
4.2.3. Primera versión del código: implementación básica en el bucle principal . . . . .	48
4.2.4. Segunda versión del código: integración de FreeRTOS y modularización . . . . .	49
4.3. Módulo de captura y transmisión de datos . . . . .	50
4.3.1. Implementación básica para obtención de datos de las coordenadas en tiempo real . . . . .	51
4.3.2. Implementación y configuración del módulo LoRaWAN para la transmisión de datos . . . . .	53
4.3.3. Implementación de la comunicación entre los módulos <i>GNSS</i> y <i>LoRaWAN</i> . . . . .	55
4.4. Módulo de interfaz gráfica y monitoreo . . . . .	57
4.4.1. Configuración de <i>ThingsBoard</i> . . . . .	57
4.4.2. Integración de <i>ThingsBoard</i> con <i>Helium</i> . . . . .	64
4.5. Integración de componentes y encapsulamiento . . . . .	68
4.5.1. Implementación Básica con Integración Inicial de los Módulos <i>RFID</i> y <i>LoRaWAN</i> . . . . .	68
4.5.2. Implementación alternativa con la Integración ajustada entre los Módulos . . . . .	68
4.5.3. Incorporación de Corte de Energía y de Arranque . . . . .	71
4.5.4. Modelado y Diseño de Encapsulado 3D. . . . .	72
<b>5. Pruebas y resultados</b>	<b>76</b>
5.1. Pruebas en entornos controlado . . . . .	76
5.1.1. Pruebas del Módulo de autentificación y control . . . . .	77
5.1.2. Pruebas del Módulo de captura y transmisión de datos	79
5.1.3. Pruebas del Módulo de Interfaz Gráfica y Monitoreo .	87
5.1.4. Integración del sistema completo . . . . .	92
5.2. Pruebas de Campo . . . . .	95
5.2.1. Evaluación Inicial . . . . .	95

5.2.2. Pruebas en Movimiento . . . . .	96
5.2.3. Resultados Cuantitativos . . . . .	97
5.2.4. Interpretación de los Resultados . . . . .	99
5.3. Prototipo del sistema desarrollado . . . . .	99
5.3.1. <i>Hardware</i> del prototipo . . . . .	99
<b>6. Conclusión</b>	<b>104</b>
6.1. Discusión . . . . .	104
6.2. Análisis de la hipótesis . . . . .	105
6.3. Principales logros alcanzados . . . . .	105
6.4. Sugerencias para futuras investigaciones . . . . .	106
<b>Glosario</b>	<b>107</b>
<b>Referencias bibliográficas</b>	<b>108</b>

# Índice de figuras

2.1. <i>Diagrama de flujo del modelo YOLO</i> . . . . .	11
2.2. <i>Diagrama de flujo del modelo YOLO – versión completa</i> . . . . .	13
2.3. <i>Tipos de huesos en extremidades caninas</i> . . . . .	17
2.4. <i>Tipos de fracturas óseas caninas según clasificación morfológica</i> . . . . .	19
2.5. <i>Ubicación de la fractura en el esqueleto canino según regiones anatómicas</i> . . . . .	20
3.1. Placa de desarrollo Heltec . . . . .	29
3.2. Lector <i>RFID</i> . . . . .	31
3.3. Módulo de desarrollo <i>GP-02 Kit</i> . . . . .	33
3.4. Placa de desarrollo <i>ESP32</i> . . . . .	34
3.5. Diagrama de Arquitectura ThingsBoard . . . . .	37
3.6. Cadena de reglas raíz ThingsBoard . . . . .	38
3.7. Interfaz del Arduino IDE 2 [56]. . . . .	41
3.8. Entorno de diseño de Fritzing mostrando el esquema de una placa [58]. . . . .	42
3.9. Entorno de desarrollo en Blender. . . . .	43
4.1. Fases principales del sistema. . . . .	44
4.2. Estructura del prototipo. . . . .	45
4.3. Arquitectura de comunicación y monitoreo. . . . .	46
4.4. Diagrama de flujo de autenticación y control. . . . .	47
4.5. Diagrama de conexiones entre WiFi LoRa 32 V3 y RFID RC522. . . . .	48
4.6. Diagrama de conexiones entre WiFi LoRa 32 V3 y GNSS. . . . .	51
4.7. Entorno para crear dispositivo y configuración de las credenciales necesarias para el nodo. . . . .	54
4.8. Ajuste de los parámetros necesarios en el dispositivo para la comunicación con el servidor <i>LoRaWAN</i> . . . . .	55
4.9. Diagrama de flujo de transmisión de datos por <i>LoRaWAN</i> . . . . .	55
4.10. Detalles del dispositivo ThingsBoard. . . . .	58

4.11.	Perfil de dispositivo ThingsBoard. . . . .	59
4.12.	Cliente con usuario creado en ThingsBoard. . . . .	59
4.13.	Asignación de dispositivos a clientes en ThingsBoard. . . . .	60
4.14.	<i>Widgets</i> disponibles en <i>ThingsBoard</i> . . . . .	61
4.15.	<i>Widgets</i> utilizados en <i>ThingsBoard</i> . . . . .	62
4.16.	Configuraciones en <i>Widgets</i> de Mapa. . . . .	63
4.17.	Diagrama del motor de reglas <i>MQTT</i> en <i>ThingsBoard</i> . . . . .	64
4.18.	Menú de integraciones en la consola de <i>Helium</i> . . . . .	65
4.19.	Detalles de conexión <i>MQTT</i> configurados en la consola de <i>Helium</i> . . . . .	66
4.20.	Flujo de integración entre el dispositivo PruebaHeltec y <i>MQTT_TB</i> . . . . .	66
4.21.	Configuraciones en el <i>router</i> de la red local. . . . .	67
4.22.	Diagrama de conexiones entre RFID RC522, ESP-WROOM-32, WiFi LoRa 32 V3 y GNSS. . . . .	69
4.23.	Diagrama de conexiones de todos los módulos. . . . .	72
4.24.	Disposición estructural de los componentes. . . . .	73
4.25.	Base del encapsulado. . . . .	74
4.26.	Parte superior del encapsulado. . . . .	74
4.27.	Modelado 3D del encapsulado para el módulo RFID. . . . .	75
5.1.	Prueba del módulo RFID. . . . .	78
5.2.	Pruebas del módulo GNSS. . . . .	79
5.3.	Comprobación de las coordenadas del módulo GNSS. . . . .	80
5.4.	Comprobación con teléfono convencional. . . . .	81
5.5.	Medición de distancias con Google Earth - Coordenadas específicas. . . . .	82
5.6.	Distancia calculada con Omni Calculator. . . . .	82
5.7.	Pruebas de ubicación en Google Earth con múltiples puntos adquiridos. . . . .	83
5.8.	Visualización en la consola helium, mensaje de conexión y subida. . . . .	83
5.9.	Visualización en la consola helium. . . . .	84
5.10.	Visualización de cobertura en la consola helium. . . . .	84
5.11.	Visualización en la página helium maps. . . . .	85
5.12.	Estructura del Mensaje enviado, donde payload serían las coordenadas en Base64. . . . .	86
5.13.	Interpretación de Base64 a Hexadecimal. . . . .	87
5.14.	Interpretación de las coordenadas de latitud y longitud transmitidas. . . . .	87

5.15. Visualización de la ubicación en el mapa <i>OpenStreet</i> con trayectoria. . . . .	88
5.16. Series temporales de telemetría. . . . .	89
5.17. Prueba de decodificación de mensajes en formato <i>Base64</i> a valores hexadecimales. . . . .	90
5.18. Prueba de interpretación de valores hexadecimales según su tamaño. . . . .	91
5.19. Uso de <i>Mosquitto</i> para simular mensajes durante las pruebas. .	91
5.20. Integración Helium-ThingsBoards. . . . .	92
5.21. Visualización en la consola Helium de los paquetes de datos recibidos del UID. . . . .	94
5.22. Visualización en la consola Helium de los paquetes de datos recibidos de las coordenadas. . . . .	95
5.23. Diagrama esquemático de conexiones. . . . .	96
5.24. Trayectoria registrada durante las pruebas de campo. . . . .	97
5.25. Disposición estructural. . . . .	100
5.26. Encapsulado 3D. . . . .	101
5.27. Encapsulado 3D para el módulo <i>RFID</i> . . . . .	101
5.28. Prototipo Final. . . . .	102
5.29. Visualización de datos en tiempo real. . . . .	103

# Índice de Tablas

3.1.	Tabla comparativa entre Placas de Desarrollo . . . . .	30
3.2.	Tabla comparativa entre módulos <i>RFID</i> . . . . .	32
3.3.	Comparativa de plataformas LoRaWAN . . . . .	36
3.4.	Tabla comparativa de Plataformas IoT . . . . .	39
5.1.	Resultados de las pruebas de detección y comparación del UID	78
5.2.	Resultados de las Pruebas por Versión del Código . . . . .	94
5.3.	Resultados de las pruebas de campo . . . . .	98

# Acrónimos y símbolos

$\pi$  raz la circunferencia del culo a su ditro. 5

**LAN** Local Area Network. 9, 10, 12

**SOR** Sistema Operativo de Red. 10, 11

**SVM** Support Vector Machine. 5

# Capítulo 1

## Introducción

En la medicina veterinaria contemporánea, el diagnóstico preciso de fracturas óseas constituye un elemento fundamental para garantizar la salud y bienestar de los animales de compañía. Sin embargo, la escasez de especialistas en diagnóstico por imágenes en Paraguay obliga a los veterinarios generales, sin formación específica en esta área, a interpretar radiografías, aumentando significativamente el riesgo de errores diagnósticos. Un diagnóstico impreciso o incorrecto puede derivar en tratamientos inadecuados, prolongando el sufrimiento del animal y comprometiendo su recuperación.

La presente investigación se centra en el desarrollo de una aplicación web asistida por inteligencia artificial para diagnosticar fracturas en imágenes radiológicas caninas, empleando tecnologías de inteligencia artificial, visión por computadora y aprendizaje profundo, integradas en sistemas web modernos.

La inteligencia artificial ha demostrado capacidades sobresalientes en el análisis y interpretación de imágenes médicas, permitiendo la detección automática de patologías con niveles de precisión comparables e incluso superiores a los especialistas humanos [?]. En este contexto, las técnicas de detección de objetos como YOLO permiten la identificación y localización de fracturas óseas en imágenes radiológicas. YOLO es un sistema de detección de objetos en tiempo real que procesa imágenes completas en una sola evaluación de red neuronal, dividiendo la imagen en regiones y prediciendo simultáneamente cuadros delimitadores y probabilidades de clase para cada región [?].

Las aplicaciones web modernas proporcionan una plataforma accesible y escalable para la implementación de soluciones de inteligencia artificial en entornos clínicos, permitiendo a los veterinarios cargar imágenes radiológicas, procesarlas mediante algoritmos de inteligencia artificial y obtener diagnósticos asistidos de manera eficiente.

## **1.1. Motivación**

La motivación para la realización de este Trabajo Final de Grado surge de la convergencia de varios factores académicos, tecnológicos y sociales que justifican la importancia de desarrollar herramientas de diagnóstico asistido por inteligencia artificial en el ámbito veterinario paraguayo.

A través de la observación en un centro veterinario local, se pudo constatar la escasez de médicos veterinarios que cuentan con los conocimientos suficientes para realizar diagnósticos precisos basados en imágenes de radiografías caninas. Esta realidad evidencia la necesidad urgente de herramientas que asistan a los profesionales en la interpretación radiológica.

En primer lugar, el desarrollo de esta aplicación web permite aplicar tecnologías emergentes de inteligencia artificial en el ámbito de la medicina veterinaria, utilizando YOLO, un algoritmo de visión por computadora basado en redes neuronales convolucionales, para detectar y diagnosticar patologías en imágenes radiológicas de forma precisa y rápida. La implementación de estas tecnologías en un contexto veterinario real ofrece la oportunidad de expandir el conocimiento y habilidades técnicas en áreas clave de la ingeniería de sistemas.

Además, este proyecto sirve como un medio para alcanzar el objetivo académico de culminar la carrera de Ingeniería de Sistemas, demostrando la capacidad para integrar conocimientos teóricos y prácticos adquiridos a lo largo de la formación universitaria. La realización de un proyecto de esta envergadura no solo contribuye al crecimiento profesional, sino que también fortalece la preparación para enfrentar desafíos futuros en el campo de la inteligencia artificial y el desarrollo de aplicaciones web especializadas.

Desde una perspectiva social, la motivación se ve impulsada por el deseo de contribuir al bienestar animal en Paraguay, donde la falta de especialistas en diagnóstico por imágenes veterinarias representa un desafío significativo para la práctica clínica. Este componente humanitario y social añade un valor adicional al proyecto, haciendo que el esfuerzo invertido tenga un impacto tangible y positivo en la calidad de atención veterinaria disponible en el país.

Finalmente, la oportunidad de aplicar inteligencia artificial en un dominio específico como el diagnóstico radiológico canino representa un campo de investigación prometedor que puede sentar las bases para futuras innovaciones en telemedicina veterinaria y sistemas de apoyo diagnóstico automatizado.

## **1.2. Definición del problema**

La medicina veterinaria en Paraguay enfrenta desafíos significativos en el área del diagnóstico por imágenes, particularmente en la interpretación de radiografías para la detección de fracturas óseas en caninos. La escasez de radiólogos veterinarios especializados obliga a los profesionales generales a realizar interpretaciones radiológicas sin la formación específica necesaria, lo que incrementa considerablemente el riesgo de errores diagnósticos.

Según datos de la práctica veterinaria local, la mayoría de las clínicas veterinarias en Paraguay no cuentan con especialistas en radiología animal, lo que obliga a los veterinarios generales a depender de profesionales externos para la interpretación de estudios radiológicos complejos. Esta limitación provoca incertidumbre diagnóstica, retrasos en la atención y, en casos graves, diagnósticos erróneos que pueden comprometer la salud y el bienestar de los animales.

Las fracturas óseas representan una de las patologías más comunes en la práctica veterinaria de pequeños animales, especialmente en caninos, y requieren un diagnóstico preciso y oportuno para determinar el tratamiento adecuado. La interpretación incorrecta de radiografías puede llevar a tratamientos inadecuados, cirugías innecesarias o, por el contrario, a la falta de intervención cuando es requerida.

A pesar de los avances significativos en inteligencia artificial aplicados al diagnóstico médico humano, su adopción en la medicina veterinaria paraguaya aún es limitada. La falta de herramientas tecnológicas accesibles y específicamente diseñadas para el contexto veterinario local representa una oportunidad para el desarrollo de soluciones innovadoras que puedan asistir a los profesionales en la toma de decisiones diagnósticas.

### *Definición del problema de investigación*

¿Cómo desarrollar una aplicación web asistida por inteligencia artificial que pueda diagnosticar fracturas en imágenes radiológicas caninas con un nivel de precisión que asista efectivamente a los veterinarios en la toma de decisiones clínicas?

## **1.3. Objetivos**

### **1.3.1. Objetivo General**

Desarrollar una aplicación web asistida por inteligencia artificial para diagnosticar fracturas en imágenes radiológicas caninas.

### **1.3.2. Objetivos Específicos**

1. Comprender el uso de tecnologías para clasificar imágenes radiológicas mediante el estudio de algoritmos de visión por computadora y aprendizaje profundo.
2. Seleccionar las herramientas de software más adecuadas para integrar el modelo de inteligencia artificial en una aplicación web funcional.
3. Diseñar la lógica integral de la aplicación web considerando aspectos de usabilidad, escalabilidad y seguridad de datos médicos.
4. Obtener imágenes radiológicas caninas que contienen fracturas óseas mediante colaboración con profesionales veterinarios autorizados.
5. Disponer las imágenes obtenidas para su análisis, a través del preprocesamiento con la colaboración de expertos en diagnóstico veterinario.
6. Seleccionar el algoritmo de inteligencia artificial más adecuado para el diagnóstico de fracturas mediante evaluación comparativa de diferentes arquitecturas.
7. Programar la lógica de las funcionalidades para cada módulo de la aplicación incluyendo carga de imágenes, procesamiento y visualización de resultados.
8. Desarrollar la aplicación completa integrando las funcionalidades programadas en una plataforma web accesible y eficiente.
9. Evaluar el desempeño de cada funcionalidad del sistema mediante la aplicación de métricas correspondientes y validación con profesionales veterinarios.

### **1.4. Hipótesis**

La aplicación web propuesta utilizando técnicas de inteligencia artificial entrenadas con un conjunto de datos diversos de imágenes radiológicas caninas alcanzará una precisión al menos del 80 % en el diagnóstico de fracturas, proporcionando una herramienta de apoyo diagnóstico confiable para veterinarios en Paraguay.

## 1.5. Justificación

En el contexto actual de la medicina veterinaria, donde la precisión diagnóstica es fundamental para el bienestar animal y la eficiencia de los tratamientos, el desarrollo de herramientas asistidas por inteligencia artificial representa una necesidad crítica y una oportunidad significativa para mejorar la calidad de atención veterinaria en Paraguay.

Este trabajo se basa en la necesidad identificada de abordar la problemática de la interpretación radiológica en medicina veterinaria, como lo evidencia la escasez de especialistas en diagnóstico por imágenes en el país. La propuesta de investigación se centra en la implementación de algoritmos de aprendizaje profundo para el análisis automatizado de imágenes radiológicas, aprovechando los avances recientes en visión por computadora y su aplicación exitosa en el diagnóstico médico humano.

Los avances en inteligencia artificial, particularmente en el procesamiento de imágenes médicas, han demostrado resultados prometedores en la detección y clasificación de patologías. El modelo YOLO ha mostrado capacidades excepcionales en tareas de detección de objetos en tiempo real, características que lo hacen ideal para la identificación de fracturas en imágenes radiológicas [?].

Este enfoque busca no solo generar conocimiento académico relevante en el área de inteligencia artificial aplicada a la medicina veterinaria, sino también tener un impacto positivo en la práctica clínica local. Se espera que la implementación de este sistema contribuya a:

- Mejorar la precisión diagnóstica en la detección de fracturas óseas caninas
- Reducir el tiempo de interpretación radiológica
- Proporcionar una segunda opinión automatizada que asista en la toma de decisiones clínicas
- Facilitar la formación y educación continua de veterinarios generales en diagnóstico por imágenes
- Establecer una base tecnológica para futuras aplicaciones de inteligencia artificial en medicina veterinaria paraguaya

Desde una perspectiva tecnológica, la motivación para llevar a cabo este proyecto surge de la oportunidad de aplicar tecnologías emergentes en un contexto local específico, demostrando la viabilidad y efectividad de soluciones de inteligencia artificial en entornos con recursos limitados. La integración

de estas tecnologías en una aplicación web accesible representa un paso importante hacia la digitalización y modernización de los servicios veterinarios en Paraguay.

En síntesis, este proyecto busca integrar la generación de conocimiento académico con la solución de problemas prácticos en la medicina veterinaria, con el objetivo de ofrecer herramientas tecnológicas innovadoras que mejoren la calidad de atención y el bienestar animal en la comunidad paraguaya.

## 1.6. Delimitación del alcance del trabajo

La investigación se centró en el desarrollo de una aplicación web para el diagnóstico asistido de fracturas óseas en imágenes radiológicas caninas. El proyecto integra algoritmos de visión por computadora basados en redes neuronales convolucionales, reconocidos por su capacidad de realizar detección de objetos en tiempo real con alta precisión. El estudio se llevó a cabo considerando las siguientes delimitaciones específicas:

**Delimitación temática:** El trabajo se limita al diagnóstico de fracturas óseas en la especie canina, excluyendo otras especies animales y otros tipos de patologías radiológicas. El análisis se enfoca específicamente en imágenes radiológicas convencionales (rayos X) de las extremidades, descartando radiografías de otras regiones del cuerpo.

**Delimitación tecnológica:** La implementación se basa en algoritmos de detección de objetos para el análisis de imágenes radiológicas. Roboflow se utiliza para la gestión, anotación y preprocesamiento de los datasets, facilitando la preparación de imágenes para el entrenamiento de modelos de visión por computadora. La aplicación web se desarrolla empleando tecnologías estándar como HTML, CSS y JavaScript para el frontend, y Python para el backend.

**Delimitación geográfica:** El sistema está diseñado para su aplicación en el contexto veterinario paraguayo, considerando las limitaciones de recursos y infraestructura tecnológica locales, aunque su arquitectura permite escalabilidad para otros contextos similares.

**Delimitación temporal:** El desarrollo del prototipo y las pruebas de validación se realizan en un período definido, con evaluaciones en entornos controlados que permiten la medición del rendimiento del sistema bajo condiciones específicas.

El trabajo incluye la implementación de un sistema que permite la carga de imágenes radiológicas, su procesamiento mediante algoritmos de inteligencia artificial y la visualización de resultados diagnósticos de manera accesible para profesionales veterinarios. Además, el sistema posibilita la generación

automática de reportes con los resultados obtenidos. Esta investigación establece una base sólida para futuras exploraciones y desarrollos en el ámbito de la inteligencia artificial aplicada a la medicina veterinaria.

# **Capítulo 2**

## **Conceptos fundamentales, teoría y antecedentes**

En este capítulo se desarrolla el marco teórico conceptual y referencial de este trabajo de investigación. En primer lugar, se presentan las definiciones básicas asociadas al estudio, como las relacionadas con la tecnología de geolocalización y radiofrecuencia. Por último, se presentan los antecedentes de trabajos anteriormente realizados en el área sobre el cual está basado este trabajo final de grado.

### **2.1. Inteligencia Artificial**

#### **2.1.1. Evolución histórica**

La inteligencia artificial (IA) tiene sus orígenes en los trabajos pioneros de Alan Turing, quien en 1950 publicó su artículo “*Computing Machinery and Intelligence*” donde propuso el ahora famoso *Test de Turing* como método para evaluar la capacidad de una máquina para exhibir un comportamiento inteligente. El término “Inteligencia Artificial” fue acuñado formalmente en 1956 durante la conferencia de Dartmouth, organizada por John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon, estableciendo así el nacimiento de este campo como disciplina académica [?].

Las décadas de 1960 y 1970 vieron el desarrollo de los primeros sistemas expertos como *DENDRAL* y *MYCIN*, que demostraron la capacidad de las computadoras para simular el razonamiento humano en dominios específicos. Sin embargo, las limitaciones técnicas y teóricas condujeron a lo que se conoce como el “invierno de la IA” en los años 80, un período de reducción de financiación e interés en la investigación [?].

El resurgimiento de la IA comenzó en los años 90 con avances en *machine learning*, particularmente con el desarrollo de métodos estadísticos y probabilísticos. El verdadero punto de inflexión llegó en 2012, cuando una red neuronal profunda desarrollada por Krizhevsky et al. redujo drásticamente la tasa de error en el desafío de reconocimiento visual *ImageNet*, inaugurando la era moderna del *deep learning* [?].

### **2.1.2. Concepto de inteligencia artificial**

La IA puede ser definida desde cuatro perspectivas diferentes: sistemas que piensan como humanos, sistemas que actúan como humanos, sistemas que piensan racionalmente y sistemas que actúan racionalmente. Esta multidimensionalidad refleja la complejidad inherente al campo [?].

La IA es la capacidad de un sistema para interpretar correctamente datos externos, aprender de dichos datos y utilizar esos aprendizajes para lograr objetivos y tareas específicas mediante adaptación flexible. Esta definición enfatiza las capacidades de aprendizaje y adaptación que caracterizan a los sistemas de IA modernos [?].

La inteligencia artificial representa la convergencia de diversos campos como la computación, la estadística, la neurociencia y la psicología cognitiva, con el objetivo de crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, incluyendo el aprendizaje, la resolución de problemas, el reconocimiento de patrones y la toma de decisiones en entornos complejos y dinámicos.

### **2.1.3. Modelos de Inteligencia Artificial**

La IA puede clasificarse en IA estrecha o débil (diseñada para tareas específicas), IA general (capaz de realizar cualquier tarea intelectual humana) e IA superinteligente (hipotética, superaría la inteligencia humana). Actualmente, todos los sistemas desplegados son ejemplos de IA estrecha, aunque la investigación avanza hacia desarrollos más generales [?].

## **2.2. *YOLO* (*You Only Look Once*)**

### **2.2.1. Concepto**

*YOLO* es un sistema de detección de objetos en tiempo real que procesa imágenes completas en una sola evaluación de red neuronal, dividiendo la

imagen en regiones y prediciendo simultáneamente cuadros delimitadores y probabilidades de clase para cada región [?].

*YOLO* es un enfoque unificado para la detección de objetos que trata la detección como un problema de regresión espacial, donde una única red neuronal predice directamente cuadros delimitadores y probabilidades de clase simultáneamente. Este diseño posibilita una optimización directa de extremo a extremo y posibilita la detección en tiempo real mientras mantiene alta precisión [?].

### 2.2.2. Características

*YOLO* destaca por su velocidad de procesamiento, alcanzando 45 fotogramas por segundo en su configuración base y hasta 155 fotogramas por segundo en su versión más rápida, lo que posibilita aplicaciones en tiempo real. Su arquitectura unificada optimiza directamente el rendimiento de detección al tratar la detección como un problema de regresión único [?].

La arquitectura de *YOLO* divide la imagen de entrada en una cuadrícula  $S \times S$  y, para cada celda de la cuadrícula, predice  $B$  cuadros delimitadores junto con sus puntuaciones de confianza y probabilidades de clase. Esta formulación posibilita al modelo considerar simultáneamente características visuales globales y contexto espacial, contribuyendo a su capacidad para generalizar bien a nuevas situaciones y dominios [?].

### 2.2.3. Arquitectura de YOLOv11

La arquitectura de YOLOv11 sigue el paradigma modular *backbone-neck-head*, optimizado para lograr un alto rendimiento en tareas de detección de objetos en tiempo real. Desarrollado por Ultralytics, este modelo introduce mejoras en la extracción de características, mecanismos de atención y predicción multi-escala que permiten mantener una elevada precisión (*mAP*) sin comprometer la velocidad de inferencia, incluso en entornos con recursos computacionales limitados [?].

#### Backbone – Extracción de características

El *backbone* de YOLOv11 emplea una arquitectura tipo *DarkNet/DarkFPN* para extraer tres niveles de características (P3, P4 y P5), representando patrones visuales desde detalles finos hasta información semántica de alto nivel. Incorpora bloques optimizados para eficiencia computacional y culmina con el módulo *SPPF* (*Spatial Pyramid Pooling Fast*), que fusiona información de contexto a múltiples escalas mediante *max-pooling*, mejorando la detección de objetos pequeños sin degradar el rendimiento en tiempo real.

#### Neck – Fusión y atención espacial

El *neck* está diseñado para combinar y refinar las características extraídas en el *backbone*. YOLOv11 introduce el bloque *C2PSA* (*Cross Stage Partial con Spatial Attention*), que aplica atención espacial en sub-rutas parciales del flujo de datos, destacando regiones relevantes y atenuando el ruido. Posteriormente, se emplea un esquema tipo FPN con *upsampling* y concatenación para mantener la coherencia entre distintas resoluciones, optimizando así la detección en múltiples escalas con un coste computacional controlado.

#### **Head – Predicción multi-escala**

El *head* opera sobre los tres mapas de características (P3, P4, P5) generados tras el *neck*, permitiendo detectar objetos pequeños, medianos y grandes de forma simultánea. Cada salida genera predicciones de cajas delimitadoras y categorías, ajustadas según el *stride* correspondiente. En la etapa final de inferencia se aplican algoritmos de *Non-Maximum Suppression (NMS)* para eliminar detecciones redundantes y conservar las más probables.

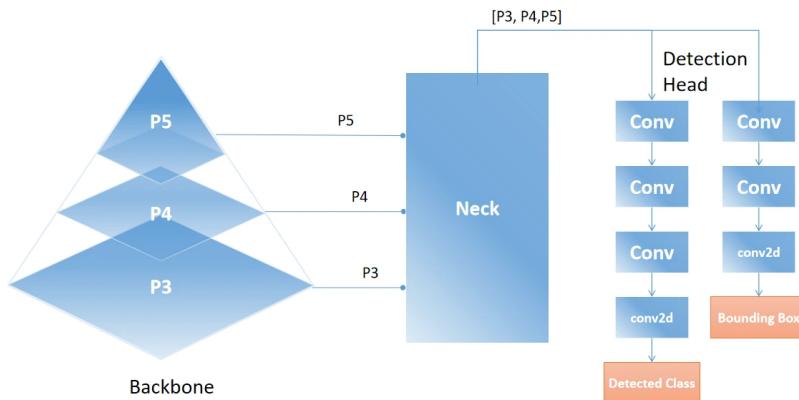


Figura 2.1: *Diagrama de flujo del modelo YOLO*.

#### **2.2.4. Arquitectura de YOLOv12**

*YOLOv12* es una evolución de la familia *YOLO* que integra mecanismos de atención en un modelo que mantiene la rapidez característica de los detectores basados en *CNN* (redes convolucionales). Su diseño busca balancear precisión y velocidad, superando las limitaciones tradicionales de los mecanismos de atención, que suelen ser lentos y consumir mucha memoria. [?].

##### **Módulo de Atención por Áreas (*Area Attention Module - A2*)**

- Divide el mapa de características en segmentos locales, en lugar de aplicar atención global costosa.
- Este método conserva un campo receptivo grande (aunque menor que

la atención global) pero con una complejidad computacional mucho menor.

- Gracias a esta división sencilla (reorganización del tensor) se reduce el costo computacional a la mitad, acelerando el proceso sin perder precisión significativa.

### Redes de Agregación Residual Eficiente (*Residual Efficient Layer Aggregation Networks - R-ELAN*)

- Evolución del módulo *ELAN* usado en YOLO anteriores, que mejora la agregación de características.
- Introduce conexiones residuales a nivel de bloque con un factor de escala para estabilizar el entrenamiento y evitar problemas de convergencia, especialmente con atención.
- Cambia la forma de agregar características, usando un solo mapa de características procesado mediante bloques tipo *cuello de botella* para mejorar eficiencia y rendimiento.

### Mejoras Arquitectónicas

- **Flash Attention:** Optimiza el acceso a memoria para acelerar la atención, cerrando la brecha de velocidad entre atención y convoluciones.
- **Eliminación del Codificado Posicional:** Se suprime el uso de codificaciones posicionales explícitas en la atención para simplificar y acelerar el modelo sin perder precisión.
- **Ajuste en la Expansión del MLP:** Reduce la relación de expansión del perceptrón multicapa de 4 a 1.2 para equilibrar la carga computacional entre atención y redes *feed-forward*.
- **Reducción de la Profundidad de Bloques:** Disminuye la cantidad de bloques apilados en la última etapa para facilitar la optimización y mejorar la velocidad de inferencia.
- **Uso Extensivo de Operadores Convolucionales:** Prefiere convoluciones con normalización por lotes en lugar de capas lineales con normalización de capas, aprovechando su eficiencia computacional.

### Resultados

- YOLOv12 mejora la precisión (*mAP*) respecto a versiones anteriores (YOLOv10, YOLOv11) y supera a otros modelos recientes, manteniendo o mejorando la velocidad de inferencia.
- Estas mejoras son escalables para distintos tamaños de modelo (desde versiones pequeñas a grandes).
- Sin embargo, para obtener la máxima velocidad, YOLOv12 requiere GPUs modernas que soporten *Flash Attention* (arquitecturas NVIDIA Turing en adelante).

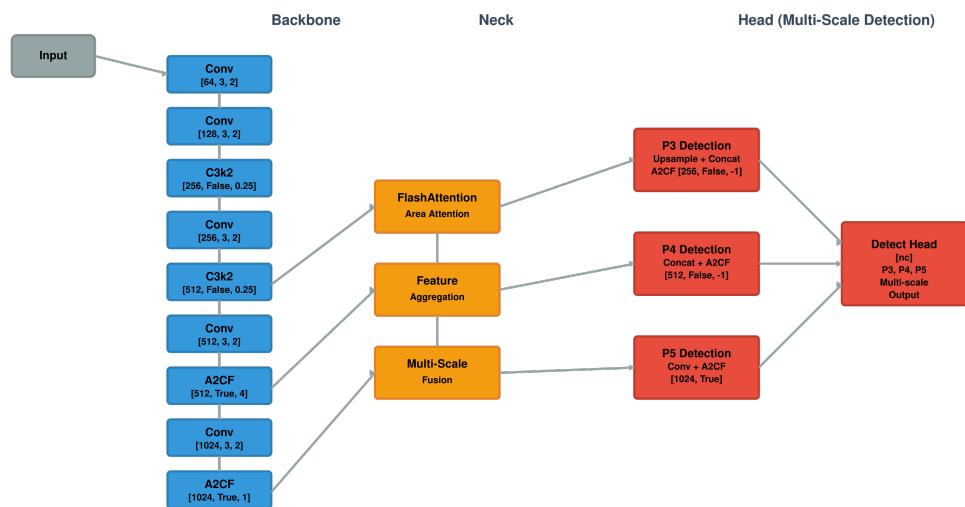


Figura 2.2: *Diagrama de flujo del modelo YOLO – versión completa.*

## 2.3. Visión por Computadora

### 2.3.1. Evolución histórica

La visión por computadora como disciplina formal comenzó a desarrollarse en los años 60, cuando investigadores como Larry Roberts exploraron la posibilidad de reconstruir objetos 3D a partir de imágenes 2D en su tesis doctoral del *MIT*. Durante la década siguiente, David Marr estableció un marco teórico influyente que conceptualizaba la visión como un proceso de información multinivel [?].

### 2.3.2. Concepto

La visión por computadora es la construcción de descripciones explícitas y significativas de objetos físicos a partir de imágenes. Esta disciplina busca dotar a las máquinas de la capacidad de interpretar y comprender información visual de manera similar a la visión humana [?].

### 2.3.3. Fases de un Sistema de Visión por Computadora

Un sistema de visión por computadora se estructura desde la adquisición de datos visuales hasta la interpretación de los resultados. Estas fases conforman un flujo modular, donde cada una contribuye de manera específica al procesamiento automatizado de imágenes. La correcta definición e implementación de estas fases es esencial para garantizar la eficiencia, precisión y trazabilidad del sistema [?].

**Adquisición de Imágenes:** La primera fase consiste en la obtención de imágenes digitales a través de sensores o dispositivos de captura. Esto puede incluir cámaras digitales, dispositivos móviles, escáneres, sensores térmicos, tomografías computarizadas o equipos de rayos X. El resultado es una imagen digital en formato estándar (por ejemplo, *PNG*, *JPEG* o *DICOM*), que servirá como entrada para las fases posteriores del sistema [?].

La calidad de la imagen capturada incide directamente en la efectividad del análisis visual. Por ello, durante esta fase también pueden realizarse tareas de conversión de formato, validación de resolución y verificación de parámetros como la nitidez, exposición y escala, garantizando así una base adecuada para los procesos siguientes [?].

**Preprocesamiento:** El preprocesamiento de imágenes tiene como finalidad mejorar la calidad visual y estandarizar las características técnicas de las imágenes antes de su análisis. Esta fase no altera el contenido semántico, pero sí lo optimiza, eliminando interferencias que podrían afectar negativamente la precisión del sistema [?]. Entre las técnicas más comunes se incluyen:

- *Redimensionamiento:* ajustar todas las imágenes a dimensiones uniformes compatibles con el modelo.
- *Normalización:* escalar los valores de los píxeles a un rango determinado (por ejemplo, de 0 a 1).
- *Reducción de ruido:* aplicación de filtros como el gaussiano o de media-n.
- *Mejoramiento de contraste:* mediante técnicas como la ecualización del histograma.

- *Alineación o rotación*: estandarizar la orientación de las imágenes.

Estas operaciones preparan las imágenes para su procesamiento eficiente, posibilitando que los algoritmos se enfoquen en patrones estructurales relevantes.

**Segmentación y Extracción de Características:** La tercera fase implica dos procesos complementarios:

- *Segmentación*: consiste en dividir la imagen en regiones homogéneas o de interés, separando estructuras relevantes del fondo [?].
- *Extracción de características*: convierte dichas regiones en representaciones numéricas o vectores de atributos que puedan ser utilizados por algoritmos de clasificación [?].

Las características extraídas pueden ser geométricas (forma, contorno, área), texturales (rugosidad, patrones repetitivos), colorimétricas (intensidad, distribución tonal) o espaciales (posición y relación con otras regiones). El éxito de esta fase depende de la pertinencia de los atributos seleccionados, ya que determinan el rendimiento del modelo en la siguiente etapa.

**Clasificación o Detección:** En esta fase se aplican modelos de inteligencia artificial para identificar, clasificar o detectar automáticamente patrones presentes en las imágenes [?]. La elección del modelo depende del objetivo del sistema: si se desea clasificar una imagen completa o localizar múltiples objetos específicos dentro de ella [?].

**Interpretación y Salida del Sistema:** En la última fase, los resultados obtenidos por el modelo son traducidos a un formato comprensible y útil para el usuario o sistema final. Esta fase implica la visualización de resultados, generación de informes y la toma de decisiones basada en la interpretación de los datos procesados [?].

## 2.4. Radiografías

Las radiografías son una técnica de imagen médica que utiliza rayos X para visualizar las estructuras internas del cuerpo. Funcionan mediante la emisión de radiación electromagnética de alta energía que atraviesa el cuerpo y es absorbida en diferentes grados según la densidad de los tejidos [?]. Estas diferencias de absorción crean un patrón de sombras que se registra en un detector, generando una imagen bidimensional donde las estructuras densas como los huesos aparecen más claras (radiopacas) y las menos densas como el aire aparecen más oscuras (radiolúcidas) [?].

Las radiografías siguen siendo una herramienta diagnóstica fundamental en medicina debido a su accesibilidad, rapidez, bajo costo relativo y capacidad para proporcionar información anatómica esencial. Son ampliamente utilizadas para examinar fracturas óseas, evaluar alteraciones estructurales en órganos internos, detectar cuerpos extraños y diagnosticar diversas condiciones patológicas.

La tecnología radiográfica ha evolucionado desde las placas fotográficas tradicionales hasta los sistemas digitales modernos, que posibilitan el procesamiento, almacenamiento y transmisión electrónica de imágenes, facilitando el análisis asistido por computadora y la aplicación de técnicas de inteligencia artificial para mejorar la precisión diagnóstica.

## 2.5. Fracturas óseas

Las fracturas óseas representan interrupciones en la continuidad estructural de un hueso, producidas cuando las fuerzas aplicadas superan la resistencia del tejido óseo [?]. Radiográficamente, se manifiestan como líneas radiolúcidas (oscuras) que atraviesan la estructura ósea, a menudo acompañadas de desplazamiento de fragmentos, cambios en la alineación anatómica y, en casos agudos, edema de tejidos blandos circundantes [?].

El diagnóstico radiográfico de fracturas requiere la identificación de signos directos (línea de fractura, desplazamiento) e indirectos (reacción periódica, edema). Las radiografías convencionales siguen siendo la modalidad de imagen de primera línea para la evaluación inicial de fracturas, aunque en casos complejos pueden requerir técnicas avanzadas como tomografía computarizada o resonancia magnética.

Las fracturas pueden clasificarse según múltiples criterios: por su comunicación con el exterior (cerradas o abiertas), por su patrón (transversales, oblicuas, espirales, comminutas), por su localización anatómica o por el mecanismo de lesión subyacente. El diagnóstico preciso y la correcta clasificación son esenciales para determinar el tratamiento adecuado y predecir el pronóstico.

## 2.6. Diagnóstico por imágenes en veterinaria

El diagnóstico por imágenes en medicina veterinaria comprende un conjunto de técnicas no invasivas que posibilitan visualizar las estructuras anatómicas internas de los animales con fines diagnósticos y terapéuticos. La radiografía convencional constituye el pilar fundamental de esta disciplina, par-

ticularmente en la evaluación del sistema musculoesquelético canino, donde posibilita la detección de fracturas, luxaciones, neoplasias óseas y alteraciones articulares degenerativas [?].

En el contexto veterinario, la interpretación radiográfica presenta desafíos específicos relacionados con la variabilidad anatómica entre razas caninas, las limitaciones en el posicionamiento del paciente y la necesidad frecuente de sedación. La radiología digital ha transformado significativamente esta práctica al mejorar la calidad de imagen, reducir la exposición a radiación y facilitar el almacenamiento y transmisión de estudios [?].

## 2.7. Anatomía Ósea Canina

El sistema esquelético canino presenta características anatómicas específicas que difieren del esquelético humano, tanto en proporciones como en funcionalidad. Los perros poseen aproximadamente 319 huesos, variando según la raza y el tamaño [?]. En esta investigación se analizaron cuatro tipos principales de huesos que representan las estructuras óseas más comúnmente afectadas por fracturas traumáticas en la práctica veterinaria.



Figura 2.3: *Tipos de huesos en extremidades caninas.*

### 2.7.1. Húmero

El húmero constituye el hueso principal de la extremidad anterior canina, ubicado entre la escápula y el antebrazo [?]. Incluye cabeza humeral, tuberosidades mayor y menor, y región distal con cóndilos humerales como

características anatómicas relevantes. Las fracturas humerales son frecuentes en perros jóvenes y activos, representando aproximadamente el 8-12 % de todas las fracturas de huesos largos [?].

### **2.7.2. Radio y Cúbito**

El radio y cúbito forman el esqueleto del antebrazo canino, estando más fusionados funcionalmente que en otros mamíferos [?]. El radio es el hueso principal de carga mientras el cúbito proporciona estabilidad articular y puntos de inserción muscular. Las fracturas de radio-cúbito son comunes en razas pequeñas y representan el 8-18 % de todas las fracturas en perros [?].

### **2.7.3. Fémur**

El fémur es el hueso más largo y fuerte de la extremidad posterior del perro, extendiéndose desde la cadera hasta la rodilla [?]. Presenta cabeza femoral, cuello femoral, trocánteres mayor y menor, y diáfisis como estructuras principales. Las fracturas femorales representan aproximadamente el 20-25 % de todas las fracturas óseas en la práctica veterinaria [?].

### **2.7.4. Tibia y Peroné**

La tibia y peroné constituyen el esqueleto de la pierna canina, siendo la tibia el hueso principal de soporte del peso corporal [?]. La tibia presenta una cresta tibial prominente para inserción del ligamento rotuliano y los músculos extensores. El peroné, más delgado, proporciona estabilidad articular y puntos de inserción muscular específicos [?].

## **2.8. Clasificación de Fracturas**

El sistema de diagnóstico desarrollado identifica cinco tipos principales de fracturas basándose en las características morfológicas observables en imágenes radiográficas caninas, siguiendo los principios de clasificación AO modificados para medicina veterinaria [?].



Figura 2.4: *Tipos de fracturas óseas caninas según clasificación morfológica.*

### **2.8.1. Sin Desplazamiento**

Fracturas en las que los fragmentos óseos mantienen su alineación anatómica normal con separación mínima (menos de 2mm) [?]. La línea de fractura es visible radiográficamente pero no existe separación significativa entre los fragmentos. Este tipo de fractura generalmente presenta mejor pronóstico y menor tiempo de consolidación.

### **2.8.2. Desplazada**

Fracturas caracterizadas por la pérdida de alineación entre los fragmentos óseos principales con separación superior a 2mm o angulación visible [?]. Los segmentos fracturados muestran separación, angulación o cabalgamiento visible en las proyecciones radiográficas estándar.

### **2.8.3. Desplazada sin Punta**

Fracturas desplazadas donde los fragmentos principales presentan bordes relativamente lisos sin formación de fragmentos triangulares o puntiagudos [?]. Este patrón facilita la reducción quirúrgica y presenta menor riesgo de complicaciones de tejidos blandos.

### **2.8.4. Desplazada con Punta**

Fracturas que combinan desplazamiento con la presencia de fragmentos óseos de morfología triangular o puntiaguda [?]. Estos fragmentos puntiagudos son fácilmente identificables en las imágenes radiográficas y pueden causar lesiones de tejidos blandos adyacentes.

### 2.8.5. Desplazada con Fragmento

Fracturas caracterizadas por desplazamiento de los fragmentos principales acompañado de fragmentos óseos adicionales de diversos tamaños [?]. Representa el patrón más complejo de fractura en términos de número de fragmentos y requiere técnicas de fijación más sofisticadas.

## 2.9. Localización Anatómica

La ubicación de la fractura dentro del hueso se clasifica en tres regiones principales que determinan las características biomecánicas, el pronóstico de la lesión y el método de tratamiento más apropiado [?].

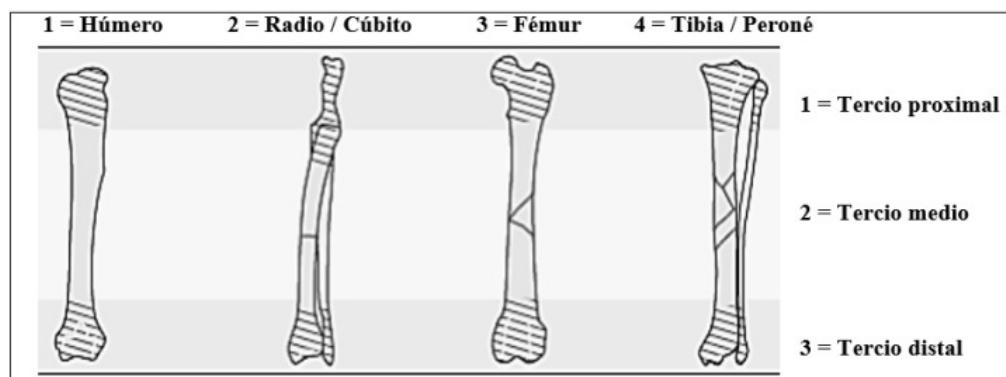


Figura 2.5: *Ubicación de la fractura en el esqueleto canino según regiones anatómicas.*

### 2.9.1. Tercio Proximal

Región que comprende el extremo del hueso más cercano al tronco del animal [?]. Incluye las estructuras articulares proximales, áreas de inserción de los principales grupos musculares y zonas de alta vascularización. Las fracturas en esta región frecuentemente involucran componentes articulares.

### 2.9.2. Tercio Medio

Corresponde a la diáfisis o cuerpo central del hueso largo [?]. Esta región se caracteriza por su cortical gruesa y representa la principal área de soporte estructural del hueso. Las fracturas diafisarias tienen generalmente buen pronóstico de consolidación debido a la rica vascularización perióstica.

### **2.9.3. Tercio Distal**

Región que abarca el extremo del hueso más alejado del tronco [?]. Frecuentemente incluye componentes articulares distales y estructuras importantes para la biomecánica articular. Requiere reducción anatómica precisa para mantener la función articular normal [?].

## **2.10. Aplicación web**

Una aplicación web constituye un *software* alojado en un servidor y accesible a través de internet mediante un navegador, sin requerir instalación local en el dispositivo del usuario. A diferencia de los sitios web estáticos, las aplicaciones web posibilitan la interactividad usuario-sistema y el procesamiento dinámico de datos, características esenciales para herramientas de diagnóstico asistido [?].

La arquitectura típica de una aplicación web moderna comprende tres componentes principales: el *frontend* (interfaz con la que interactúa el usuario), el *backend* (lógica de procesamiento y acceso a datos) y la base de datos (almacenamiento persistente). Este paradigma de desarrollo conocido como arquitectura de tres capas facilita el mantenimiento, escalabilidad y seguridad del sistema [?].

Las aplicaciones web médicas presentan requisitos específicos relacionados con la privacidad de datos, alta disponibilidad, interfaces intuitivas adaptadas al contexto clínico y capacidad para gestionar archivos de imagen de gran tamaño. En el ámbito veterinario, estas aplicaciones están experimentando una rápida adopción, particularmente aquellas que implementan algoritmos de inteligencia artificial para asistir en tareas diagnósticas complejas.

## **2.11. Herramientas utilizadas**

El desarrollo de sistemas de diagnóstico asistido por computadora para radiología veterinaria integra diversas tecnologías y herramientas especializadas. En el ámbito del *machine learning*, las redes neuronales convolucionales (*CNN*) han demostrado eficacia superior en tareas de procesamiento de imágenes médicas, siendo *YOLO* las arquitecturas seleccionadas para este trabajo debido a su capacidad para detectar y clasificar anomalías en imágenes radiográficas caninas [?].

Para la implementación del *backend*, *frameworks* como *TensorFlow*, *PyTorch* y *Keras* facilitan el desarrollo y despliegue de modelos de inteligencia

artificial, mientras que tecnologías como *Flask*, *Django* o *Node.js* proporcionan la infraestructura para la creación de *APIs* y servicios web [?].

**Python:** Lenguaje de programación interpretado de alto nivel utilizado ampliamente en el desarrollo de aplicaciones de inteligencia artificial, procesamiento de imágenes y *backend* de aplicaciones web.

**HTML:** Lenguaje de marcado utilizado para estructurar el contenido de las páginas web, definiendo elementos como encabezados, párrafos, imágenes y enlaces que conforman la interfaz visual de la aplicación.

**CSS:** Lenguaje de estilos que determina la presentación visual de los elementos *HTML*, posibilitando definir colores, tipografías, espaciados y diseños responsivos que se adaptan a diferentes dispositivos y tamaños de pantalla.

**JavaScript:** Lenguaje de programación que añade interactividad a las interfaces web, posibilitando manipular el contenido dinámicamente, procesar eventos del usuario, realizar validaciones y comunicarse con el servidor a través de peticiones asincrónicas.

**MySQL:** Sistema de gestión de bases de datos relacional de código abierto que posibilita almacenar, organizar y recuperar datos de manera eficiente. En aplicaciones de diagnóstico médico, facilita el almacenamiento de información de pacientes, resultados de análisis e historial clínico con alta integridad y seguridad.

## 2.12. Antecedentes

### 2.12.1. Detección de fracturas en radiografías de cadera mediante el uso de inteligencia artificial (Año 2024)

**Desarrollado por:** Ramón Martínez Oliva, bajo la tutoría de Germán González Serrano, Universidad de Alicante - Escuela Politécnica Superior.

En [?] se desarrolló un sistema de inteligencia artificial capaz de analizar radiografías de cadera para detectar fracturas de fémur, identificar su tipo y localización específica. Este sistema fue diseñado para mejorar la precisión y eficiencia del diagnóstico médico en el contexto de fracturas de cadera, las cuales representan una patología frecuente con alta morbilidad, especialmente en población de edad avanzada. La metodología empleada consistió en la comparación de seis arquitecturas de redes neuronales convolucionales (DenseNet121, Xception, InceptionV3, EfficientNetB4, ResNet50 y VGG16) aplicando *Transfer Learning* con pesos preentrenados de ImageNet. Se utilizó un conjunto de datos de 1,454 imágenes de fémures obtenidas de 986 estudios radiográficos del Hospital Universitario San Juan de Alicante, clasifi-

ficadas en seis categorías: sin fractura, fracturas de cuello femoral tipos I-II y III-IV según Garden, fracturas trocantéricas tipos I-II y III-V según Evans, y fracturas subtrocantéricas.

Los principales parámetros evaluados fueron la exactitud (*accuracy*) y el área bajo la curva ROC (AUC) para la clasificación multiclase. Para la detección y segmentación de fémures se empleó un modelo YOLO preentrenado, seguido de técnicas de preprocessamiento como inversión cromática y extracción de región de interés. Se implementó *Grid Search* para la optimización de hiperparámetros (*learning rate, dropout, batch size*) y técnicas de aumento de datos (*Data Augmentation*) para mejorar la generalización del modelo.

Los resultados mostraron que DenseNet121 alcanzó el mejor rendimiento con una exactitud del 78.6 % en el conjunto de prueba y un AUC de 0.985 para la detección binaria de fracturas, superando significativamente el rendimiento de otras arquitecturas evaluadas. El sistema demostró particular eficacia en la clasificación de fémures sanos (98.5 % de precisión) y fracturas de cuello femoral tipo III-IV (95.1 %), aunque presentó limitaciones en fracturas de cuello femoral tipo I-II (6.25 %) y subtrocantéricas (41.7 %).

Ambos proyectos utilizan inteligencia artificial para el diagnóstico automatizado de fracturas en imágenes radiológicas, empleando arquitecturas de *deep learning* para la clasificación de diferentes tipos de fracturas. Ambos sistemas buscan asistir a profesionales médicos en el diagnóstico, especialmente en contextos donde existe escasez de especialistas en radiología. Asimismo, ambos proyectos implementan YOLO para la detección y localización de estructuras óseas de interés en las radiografías.

El antecedente se enfoca exclusivamente en fracturas de cadera humana utilizando múltiples arquitecturas CNN (principalmente DenseNet121), mientras que el proyecto actual se especializa en diagnóstico veterinario de fracturas caninas empleando la arquitectura ViT para clasificación. El proyecto de referencia clasifica fracturas según sistemas médicos establecidos (Garden, Evans), mientras que el actual se centra en la ubicación anatómica de fracturas en diferentes tipos de huesos caninos. Además, el trabajo actual incluye la generación automática de reportes diagnósticos, funcionalidad no presente en el antecedente analizado.

### **2.12.2. Detection of Hand Bone Fractures in X-ray Images using Hybrid YOLO NAS (Año 2024)**

**Desarrollado por:** Sai Charan Medaramatla, Chennupati Veda Samhitha, Sagar Dhanraj Pande, Surendar Reddy Vinta.

En [?] se desarrolló un modelo híbrido para la detección automática de

fracturas en huesos de la mano utilizando imágenes de rayos X. El sistema fue diseñado para asistir a profesionales médicos y radiólogos en el diagnóstico temprano y preciso de fracturas, abordando la problemática de la escasez de especialistas en radiología y los posibles errores humanos en la interpretación de imágenes médicas. La metodología empleada consistió en el desarrollo de un modelo híbrido que combina los algoritmos YOLO NAS, EfficientDet y DETR3 para aprovechar las fortalezas de cada uno en la detección de objetos. El dataset utilizado fue una colección híbrida de 4,736 imágenes de rayos X de huesos de mano, clasificadas en 6 categorías: fracturas de dedos, muñeca, antebrazo, codo, húmero y hombro. Los principales parámetros evaluados fueron: precisión, sensibilidad (recall), puntuación F1 y precisión media promedio (mAP). El modelo propuesto logró una precisión del 99.20 % en entrenamiento y 98.10 % en pruebas, con un mAP de 97.85 %, superando significativamente a algoritmos individuales como InceptionV3 (72.50 %), VGG19 (74.30 %) y ResNet50 (71.21 %).

Ambos proyectos utilizan técnicas de inteligencia artificial para asistir en el diagnóstico médico mediante análisis de imágenes, específicamente para la detección de fracturas óseas. Sin embargo, presentan diferencias fundamentales en su enfoque: el trabajo de referencia se centra exclusivamente en fracturas de huesos humanos de la mano utilizando un modelo híbrido que combina múltiples algoritmos (YOLO NAS, EfficientDet, DETR3), mientras que el proyecto actual se especializa en fracturas óseas caninas empleando YOLO para detección de objetos (tipo de hueso y fractura) y ViT para clasificación de ubicación de la fractura. Además, el trabajo de referencia utiliza un dataset de imágenes estáticas para entrenamiento, mientras que el proyecto actual incluye la generación automática de reportes diagnósticos, lo que representa una funcionalidad adicional orientada a la práctica veterinaria. Estas diferencias subrayan la adaptabilidad de las tecnologías de visión por computadora en distintos contextos médicos, tanto humanos como veterinarios.

### **2.12.3. DW-YOLO: Improved YOLO for Bone Fracture Detection (Año 2024)**

**Desarrollado por:** Bo Liu.

Se desarrolló DW-YOLO, una adaptación mejorada del modelo de detección de objetos YOLO, fortificada con capas convolucionales depthwise (DWConv) para mejorar la identificación de fracturas en imágenes médicas. [?]. El sistema fue diseñado para abordar la necesidad de detección rápida y precisa de fracturas óseas, especialmente en instalaciones con pocos

diagnósticos experimentados. La metodología empleada consistió en la integración de convoluciones depthwise en la arquitectura YOLOv5 para reducir el número de parámetros y mejorar la velocidad de inferencia, manteniendo alta precisión. El dataset utilizado comprendió 4,148 imágenes craneales categorizadas en seis tipos distintos de fracturas, distribuidas metodológicamente en 70 % para entrenamiento, 20 % para validación y 10 % para pruebas. Los principales parámetros evaluados fueron: precisión media promedio (mAP), precisión, sensibilidad (recall) y puntuación F1. El modelo DW-YOLO logró un mAP de 0.889 en el conjunto de prueba con un tamaño de modelo de solo 7.5MB, demostrando capacidades de procesamiento rápido de imágenes mientras mantiene alta precisión. La arquitectura incorpora técnicas de aumento de datos Mosaic, cajas de anclaje adaptativas y utiliza una función de pérdida compuesta que combina pérdida de clasificación, pérdida de objetividad y pérdida de regresión de cajas (utilizando CIoU loss).

Ambos proyectos utilizan arquitecturas YOLO para la detección automática de fracturas óseas en imágenes médicas, abordando la problemática de la escasez de especialistas en diagnóstico por imágenes. Sin embargo, presentan diferencias significativas en su enfoque técnico y aplicación: el trabajo de referencia se especializa en fracturas craneales humanas utilizando DW-YOLO con convoluciones depthwise para optimizar velocidad y eficiencia computacional, mientras que el proyecto actual se enfoca en fracturas óseas caninas empleando una combinación de YOLO para detección de objetos y ViT para clasificación de ubicación de fracturas. Además, el trabajo de referencia se centra únicamente en la detección y localización de fracturas con un dataset de 4,148 imágenes craneales, mientras que el proyecto actual incluye funcionalidades adicionales como la generación automática de reportes diagnósticos orientados a la práctica veterinaria. El trabajo de referencia logra un modelo compacto de 7.5MB optimizado para aplicaciones clínicas de tiempo real, mientras que el proyecto actual realiza una aplicación web asistida por inteligencia artificial para proporcionar una solución más completa para veterinarios generales. Estas diferencias subrayan la versatilidad de las arquitecturas YOLO en diferentes contextos médicos y la importancia de la optimización específica según las necesidades clínicas particulares.

#### **2.12.4. Inteligencia Artificial en Radiología (Año 2021)**

**Desarrollado por:** José Antonio Marín Rodríguez y Germán Lucini Pelayo.

En [?] se realizó una revisión exhaustiva sobre la aplicación de la inteligencia artificial en el campo de la radiología, explorando tanto sus aplicaciones prácticas como sus implicaciones ético-legales. El sistema fue concebido pa-

ra analizar el impacto transformador de las tecnologías de IA en el flujo de trabajo radiológico y su potencial para mejorar la atención médica. La metodología empleada consistió en una revisión bibliográfica sistemática utilizando motores de búsqueda científicos como PubMed, Scopus, Google Scholar y ResearchGate, complementada con formación especializada en cursos de inteligencia artificial y entrevistas con profesionales del sector. El estudio abarcó tres áreas principales: utilidades de la IA en radiología (adquisición de imagen, procesamiento e informe), radiómica como nueva especialidad médica, y aspectos ético-legales del desarrollo de tecnologías de IA. Los principales hallazgos incluyeron aplicaciones exitosas como la detección automática de neumotórax con sensibilidad del 93.15 % y especificidad del 92.99 %, detección de fracturas de muñeca que mejora la sensibilidad diagnóstica del 80.8 % al 91.5 %, y el desarrollo de la radiómica para análisis cuantitativo de imágenes médicas. El trabajo también identificó desafíos críticos como el problema de la “caja negra” del deep learning, sesgos en datasets de entrenamiento, y la necesidad de marcos regulatorios robustos como el propuesto por la Unión Europea para IA de alto riesgo en aplicaciones médicas.

Ambos proyectos abordan la aplicación de inteligencia artificial en el diagnóstico médico por imágenes, reconociendo la problemática de la escasez de especialistas en radiología y el potencial de la IA para mejorar la precisión diagnóstica. Sin embargo, presentan diferencias significativas en su alcance y enfoque: el trabajo de referencia constituye una revisión teórica amplia que abarca todo el espectro de aplicaciones de IA en radiología humana, incluyendo aspectos técnicos, éticos y regulatorios, mientras que el proyecto actual se enfoca específicamente en el desarrollo práctico de una aplicación de IA para diagnóstico de fracturas en radiología veterinaria canina. El trabajo de referencia analiza múltiples algoritmos y aplicaciones existentes sin implementar una solución específica, mientras que el proyecto actual desarrolla una herramienta funcional que combina YOLO para detección de objetos y ViT para clasificación, incluyendo generación automática de reportes diagnósticos. Además, el trabajo de referencia se centra en el impacto de la IA en la práctica radiológica humana y sus implicaciones para el futuro de la profesión, mientras que el proyecto actual aborda una necesidad específica en medicina veterinaria donde los profesionales generales requieren asistencia de IA para tomar decisiones diagnósticas en ausencia de especialistas. Estas diferencias subrayan la evolución de la IA médica desde análisis teóricos hacia implementaciones prácticas específicas por especialidad.

## **2.12.5. Implementation of Personal Protective Equipment Detection Using Django and Yolo Web at Paiton Steam Power Plant (PLTU) (Año 2023)**

**Desarrollado por:** Khoirun Nisa, Fathorazi Nur Fajri, Zainal Arifin.

En [?] se desarrolló un sistema de detección de Equipos de Protección Personal (EPP) en tiempo real utilizando YOLOv8 y Django Web en la Planta de Energía a Vapor Paiton (PLTU). El sistema fue diseñado para mejorar la seguridad laboral y prevenir accidentes mediante el monitoreo del cumplimiento de los requisitos de EPP. La metodología empleada consistía en la recolección de datos de imágenes, preprocesamiento, entrenamiento del modelo y despliegue del sistema usando el framework Django. Los principales parámetros evaluados fueron: la precisión de detección (90.3 %), el valor de recall (75.1 %) y el mAP50 (81.6 %). El sistema logró detectar cuatro clases de EPP (casco, chaleco, no-casco, no-chaleco) con una precisión promedio del 82.3 % en 230 datos de prueba. Los resultados mostraron que YOLOv8 es efectivo para la detección de EPP en entornos industriales, aunque se encontraron factores de error relacionados con la iluminación y especificaciones de la cámara.

Ambos trabajos utilizan la tecnología YOLO para detección de objetos en tiempo real y se enfocan en aplicaciones de seguridad y salud. Además, ambos emplean frameworks web para la interfaz de usuario y utilizan Roboflow para el manejo de datasets y preprocesamiento de datos.

Este trabajo se centra en la detección de EPP para seguridad laboral usando YOLOv8 con cuatro clases específicas (casco/chaleco), mientras que la propuesta actual utiliza YOLO para detectar tipos de huesos y fracturas en imágenes radiológicas caninas, complementado con un algoritmo de clasificación ViT para localizar fracturas. El enfoque del trabajo referenciado es preventivo en seguridad industrial, mientras que el proyecto actual tiene un enfoque diagnóstico en medicina veterinaria para asistir a profesionales en la toma de decisiones clínicas.

# Capítulo 3

## Tecnologías utilizadas

En este capítulo se presentan las herramientas utilizadas para el desarrollo de este trabajo. En primer lugar, para asegurar que las herramientas seleccionadas sean adecuadas, se realiza una comparación entre alternativas considerando los factores y características más importantes dentro de las actividades.

### 3.1. Componentes de *Hardware*

Los componentes de hardware abarcan los dispositivos físicos necesarios para la ejecución del sistema, incluyendo unidades de procesamiento gráfico especializadas que se encargan de realizar las tareas de entrenamiento y procesamiento de modelos de inteligencia artificial requeridas.

#### 3.1.1. Tarjeta NVIDIA® GeForce RTX™ 5090

La *NVIDIA® GeForce RTX™ 5090* (véase Figura 3.10) es una unidad de procesamiento gráfico () de alta gama diseñada específicamente para aplicaciones de inteligencia artificial, aprendizaje automático y computación de alto rendimiento. Esta tarjeta representa el estado del arte en tecnología de procesamiento paralelo, ofreciendo capacidades excepcionales para el entrenamiento de modelos de *deep learning* y procesamiento de grandes volúmenes de datos.

La *RTX™ 5090* incorpora la arquitectura más avanzada de *NVIDIA*, con miles de núcleos *CUDA* especializados que permiten la ejecución paralela de operaciones matemáticas complejas. Esta característica es fundamental para el entrenamiento de redes neuronales profundas, donde se requiere procesar matrices de gran tamaño y realizar millones de operaciones por segundo.

Además, cuenta con núcleos *Tensor* de tercera generación que aceleran específicamente las operaciones de inteligencia artificial, reduciendo significativamente los tiempos de entrenamiento [?].

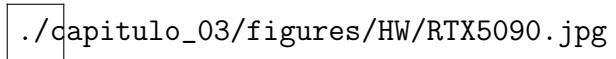


Figura 3.1: Tarjeta gráfica NVIDIA® GeForce RTX™ 5090

Una de las ventajas más destacadas de esta es su amplia memoria *VRAM* de alta velocidad, que permite cargar y procesar *datasets* de gran tamaño sin limitaciones de memoria. Esto resulta especialmente importante cuando se trabaja con modelos de visión computacional que requieren procesar imágenes de alta resolución o grandes cantidades de datos simultáneamente.

Para este proyecto, la obtención de la tarjeta *NVIDIA® GeForce RTX™ 5090* se realizó a través de la plataforma *NiceGPU*, un servicio en línea que proporciona acceso gratuito a recursos de computación de alto rendimiento para fines académicos y de investigación. Esta plataforma permite a estudiantes e investigadores acceder a hardware especializado sin incurrir en los altos costos asociados con la adquisición de equipos de gama profesional.

La selección de esta se fundamentó en un análisis comparativo de diferentes opciones disponibles en el mercado. Se consideraron criterios técnicos específicos para determinar la opción más adecuada para las demandas computacionales del proyecto.

Los criterios de evaluación establecidos fueron:

- **Rendimiento computacional:** Capacidad para manejar operaciones de matriz y álgebra lineal de forma eficiente.
- **Memoria disponible:** Cantidad de *VRAM* necesaria para cargar modelos complejos y *datasets* grandes.
- **Soporte para frameworks:** Compatibilidad nativa con *TensorFlow*, *PyTorch* y otras herramientas de *ML*.
- **Eficiencia energética:** Relación optimizada entre rendimiento y consumo energético.
- **Disponibilidad:** Accesibilidad a través de plataformas de computación en la nube.
- **Costo:** Consideración de los gastos asociados al uso del hardware.

Para reflejar el desempeño relativo de cada dispositivo, se empleó una escala del 1 al 3:

- **3:** Indica el mejor rendimiento o la opción más favorable en ese criterio.
- **2:** Representa un rendimiento intermedio, adecuado pero no el mejor.
- **1:** Refleja un rendimiento limitado en comparación con los otros dispositivos.

A continuación, se presenta la tabla 3.5 comparativa que resume las características técnicas más relevantes de cada dispositivo, facilitando así la selección de la opción más adecuada para este proyecto.

Tabla 3.1: Tabla comparativa entre Tarjetas Gráficas

Criterio/GPU	RTX™ 4080	RTX™ 5090	RTX™ 4090
Rendimiento computacional	2	3	2
Memoria disponible	2	3	3
Soporte para frameworks	3	3	3
Eficiencia energética	3	2	1
Disponibilidad	2	3	2
Costo	2	3	1
<b>Total</b>	<b>14</b>	<b>17</b>	<b>12</b>

De la tabla comparativa se determina que la *NVIDIA® GeForce RTX™ 5090* es el dispositivo más adecuado para la investigación. Este dispositivo fue seleccionado para las pruebas debido a su superior rendimiento computacional y disponibilidad gratuita a través de *NiceGPU*, lo que lo convierte en una opción ideal para el entrenamiento de modelos de inteligencia artificial complejos.

### 3.2. Componentes de *Software*

Los componentes de software proporcionan las herramientas y algoritmos necesarios para diseñar, entrenar y evaluar los modelos de inteligencia artificial. Además, facilitan la preparación y gestión de *datasets*, permitiendo un flujo de trabajo eficiente desde la recolección de datos hasta la implementación del modelo final.

### 3.2.1. Algoritmos de Entrenamiento

Para el desarrollo del modelo de inteligencia artificial se seleccionaron tres arquitecturas de redes neuronales convolucionales reconocidas por su eficacia en tareas de visión computacional. La selección se basó en un análisis comparativo considerando factores como precisión, eficiencia computacional y capacidad de generalización.

#### ResNet50

*ResNet50* (*Residual Network* de 50 capas) es una arquitectura de red neuronal convolucional profunda que introduce el concepto de conexiones residuales para resolver el problema de degradación en redes muy profundas. Esta arquitectura permite el entrenamiento efectivo de redes con mayor número de capas sin sufrir el problema del gradiente que desaparece.

Las conexiones residuales permiten que la información fluya directamente a través de la red mediante atajos, facilitando el entrenamiento de arquitecturas más profundas y complejas. *ResNet50* ha demostrado excelente rendimiento en tareas de clasificación de imágenes y transferencia de aprendizaje, convirtiéndose en una opción robusta para aplicaciones de visión computacional [?].

La arquitectura *ResNet50* se caracteriza por:

- **Estabilidad en el entrenamiento:** Las conexiones residuales facilitan la convergencia del modelo durante el proceso de entrenamiento.
- **Transferencia de aprendizaje:** Disponibilidad de pesos pre-entrenados en *ImageNet* que aceleran el desarrollo.
- **Rendimiento comprobado:** Resultados consistentes y reproducibles en *benchmarks* académicos.
- **Eficiencia computacional:** Balance adecuado entre precisión y recursos computacionales necesarios.

#### EfficientNetB3

*EfficientNetB3* forma parte de la familia *EfficientNet*, que se caracteriza por optimizar simultáneamente la profundidad, anchura y resolución de la red neuronal mediante un método de escalado compuesto. Esta arquitectura logra un balance superior entre precisión y eficiencia computacional.

*EfficientNetB3* utiliza bloques de convolución eficientes y técnicas de escalado que permiten obtener mejor rendimiento con menor cantidad de

parámetros comparado con arquitecturas tradicionales. Su diseño optimizado resulta especialmente valioso cuando se trabaja con recursos computacionales limitados o se requiere implementación en dispositivos con restricciones de memoria [?].

Las ventajas principales de *EfficientNetB3* incluyen:

- **Eficiencia de parámetros:** Menor número de parámetros para rendimiento equivalente o superior.
- **Escalado balanceado:** Optimización simultánea de múltiples dimensiones de la red neuronal.
- **Velocidad de inferencia:** Tiempos de procesamiento reducidos sin comprometer la precisión.
- **Versatilidad:** Adaptabilidad a diferentes tamaños de *dataset* y dominios de aplicación.

## Vision Transformer (ViT)

*Vision Transformer (ViT)* representa un paradigma diferente en el procesamiento de imágenes, aplicando la arquitectura *Transformer* (originalmente diseñada para procesamiento de lenguaje natural) al dominio de la visión computacional. Esta aproximación divide las imágenes en *patches* que son tratados como *tokens* de secuencia.

*ViT* ha demostrado capacidades superiores cuando se dispone de grandes cantidades de datos de entrenamiento, ofreciendo una alternativa prometedora a las redes convolucionales tradicionales. Su mecanismo de atención permite capturar relaciones globales en la imagen de manera más directa que las convoluciones [?].

Los beneficios de *ViT* incluyen:

- **Atención global:** Capacidad para modelar relaciones a larga distancia en imágenes de forma directa.
- **Escalabilidad:** Rendimiento mejorado con *datasets* más grandes y diversos.
- **Flexibilidad:** Menor *inductive bias* comparado con *CNNs* tradicionales.
- **Innovación arquitectural:** Representación del estado del arte en visión computacional moderna.

Con el fin de seleccionar los algoritmos más adecuados, se establecieron criterios de evaluación que permiten analizar y comparar las opciones disponibles:

- **Precisión:** Capacidad del modelo para realizar clasificaciones correctas en datos de prueba.
- **Eficiencia computacional:** Relación entre rendimiento y recursos computacionales requeridos.
- **Velocidad de entrenamiento:** Tiempo necesario para convergencia del modelo.
- **Generalización:** Capacidad para mantener rendimiento en datos no vistos durante el entrenamiento.
- **Transferencia de aprendizaje:** Disponibilidad y efectividad de modelos pre-entrenados.
- **Documentación y soporte:** Calidad de recursos disponibles y comunidad de desarrolladores.

La tabla 3.6 presenta la comparación cuantitativa de los algoritmos seleccionados:

Tabla 3.2: Tabla comparativa entre Algoritmos de Entrenamiento

Criterio/Algoritmo	ResNet50	EfficientNetB3ViT	
Precisión	3	3	3
Eficiencia computacional	2	3	2
Velocidad de entrenamiento	3	2	1
Generalización	3	3	2
Transferencia de aprendizaje	3	3	2
Documentación y soporte	3	2	3
<b>Total</b>	<b>17</b>	<b>16</b>	<b>13</b>

De acuerdo con los resultados obtenidos en la tabla comparativa, los tres algoritmos presentan características complementarias que justifican su selección conjunta. *ResNet50* destaca por su estabilidad y soporte, *EfficientNetB3*

por su eficiencia, y *ViT* por su innovación arquitectural, proporcionando un conjunto diverso de aproximaciones para el entrenamiento del modelo.

### 3.2.2. Plataforma Roboflow

*Roboflow* (ver Figura 3.11) es una plataforma en línea especializada en la gestión, preparación y *augmentación* de *datasets* para proyectos de visión computacional. Esta herramienta facilita todo el flujo de trabajo desde la recolección de imágenes hasta la preparación de datos listos para el entrenamiento de modelos de inteligencia artificial.

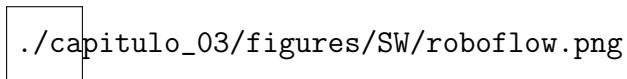


Figura 3.2: Interfaz de la plataforma Roboflow

La plataforma ofrece funcionalidades esenciales para el procesamiento de *datasets*:

**Gestión de Datos:** *Roboflow* permite cargar, organizar y etiquetar imágenes de manera eficiente. Su interfaz intuitiva facilita la anotación de objetos en imágenes, generando automáticamente los archivos de etiquetas en formatos compatibles con los principales *frameworks* de *machine learning*.

**Augmentación Automática:** Una de las características más valiosas es su capacidad para aplicar técnicas de *augmentación* de datos de forma automática. Esto incluye rotaciones, cambios de iluminación, recortes, y otras transformaciones que incrementan la diversidad del *dataset* y mejoran la generalización del modelo.

**Preprocesamiento:** La plataforma automatiza tareas de preprocesamiento como redimensionamiento, normalización y conversión de formatos, asegurando que los datos estén optimizados para el entrenamiento.

**Control de Calidad:** Herramientas integradas para detectar imágenes duplicadas, etiquetas inconsistentes y otros problemas comunes que pueden afectar la calidad del *dataset*.

**Exportación Flexible:** Capacidad para exportar *datasets* en múltiples formatos compatibles con *TensorFlow*, *PyTorch*, *YOLOv5*, y otros *frameworks* populares [?].

Para la selección de la plataforma de gestión de *datasets*, se evaluaron diferentes opciones considerando criterios específicos para el desarrollo del proyecto. Se compararon *Roboflow*, *Labelbox* y *V7* bajo los siguientes parámetros:

- **Facilidad de uso:** Intuitividad de la interfaz y curva de aprendizaje requerida.
- **Automatización:** Capacidades de automatización en el procesamiento de datos.
- **Calidad:** Herramientas disponibles para asegurar la consistencia del *dataset*.
- **Compatibilidad:** Soporte nativo para múltiples formatos y *frameworks*.
- **Costo:** Disponibilidad de tier gratuito adecuado para proyectos académicos.
- **Documentación:** Calidad de recursos y soporte de la comunidad.

La tabla 3.7 presenta la comparación entre las plataformas evaluadas:

Tabla 3.3: Tabla comparativa entre Plataformas de Gestión de Datasets

Criterio/Plataforma	Roboflow	Labelbox	V7
Facilidad de uso	3	2	2
Automatización	3	2	3
Calidad	3	3	2
Compatibilidad	3	2	2
Costo	3	1	2
Documentación	3	3	2
<b>Total</b>	<b>18</b>	<b>13</b>	<b>13</b>

De acuerdo con los resultados obtenidos en la tabla comparativa, *Roboflow* ha sido seleccionado como la plataforma más adecuada para la gestión de *datasets* en este proyecto. Su combinación de facilidad de uso, capacidades de automatización avanzadas, y disponibilidad de un tier gratuito robusto lo posicionan como la opción óptima para el desarrollo académico.

Estas tecnologías, tanto de *hardware* como *software*, fueron seleccionadas considerando su complementariedad y capacidad para soportar los objetivos específicos del proyecto. La combinación de la potencia computacional de la *NVIDIA® GeForce RTX™ 5090*, la diversidad algorítmica de *ResNet50*, *EfficientNetB3* y *ViT*, junto con las capacidades de gestión de datos de *Roboflow*, proporcionan un ecosistema tecnológico robusto y eficiente para el desarrollo del sistema de inteligencia artificial propuesto.

# Capítulo 4

## Método

Este capítulo describe el desarrollo del trabajo de investigación, enfocado en la creación de un prototipo de control para motocicletas mediante tecnologías *RFID* y *LoRaWAN*, con el propósito de implementar un sistema de autenticación y localización para prevenir robos. El trabajo constituye una investigación aplicada al desarrollo tecnológico de un sistema de seguridad antirrobo, con un enfoque cuantitativo y un diseño experimental y transversal en el tiempo, abarcando el desarrollo de un prototipo funcional. En las siguientes secciones se detallan los módulos desarrollados, cada uno representando una fase clave en el funcionamiento del sistema (ver Figura 4.1).



Figura 4.1: Fases principales del sistema.

### 4.1. Definición de arquitectura

En esta primera fase, se definieron los requisitos técnicos orientados a proporcionar seguridad y monitoreo remoto en tiempo real mediante tecnologías *IoT*. El objetivo principal fue desarrollar un sistema que autentique al propietario de la motocicleta a través de un módulo *RFID* y, en caso de intento de robo o alejamiento del vehículo sin el *tag* autorizado, interrumpa el encendido y envíe una alerta a un servidor remoto. Para satisfacer estos requisitos, se diseñó una arquitectura que integra comunicación de baja potencia me-

diente *LoRaWAN* y un dispositivo de geolocalización *GNSS*, permitiendo el monitoreo y la visualización de datos en una plataforma gráfica accesible.

#### 4.1.1. Requisitos clave

- Autenticación del propietario mediante *RFID*.
- Corte de corriente en el sistema de encendido de la motocicleta cuando el *tag* autorizado se encuentre fuera de rango.
- Envío de datos de geolocalización (coordenadas *GNSS*) y estado de autenticación mediante *LoRaWAN* hacia la red Helium.
- Monitoreo remoto de la ubicación y estado de la motocicleta a través de una interfaz gráfica.

Con base en estos requisitos, se definieron dos aspectos principales: la estructura interna del prototipo y la arquitectura de comunicación externa hacia el servidor.

#### 4.1.2. Estructura interna del prototipo

La Figura 4.2 ilustra el diseño del hardware y el flujo de control dentro del prototipo.

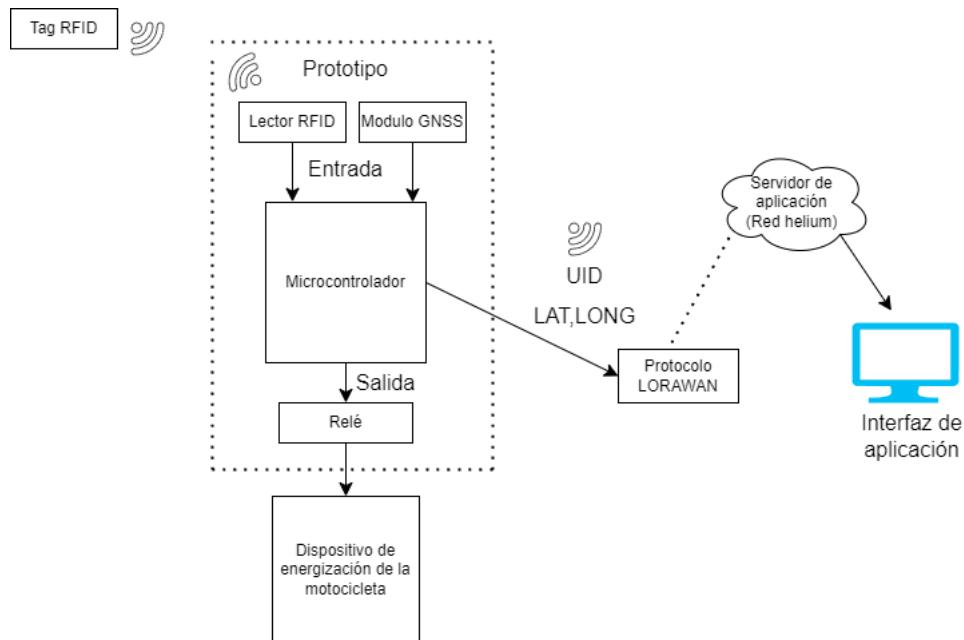


Figura 4.2: Estructura del prototipo.

- **Microcontrolador (Heltec WiFi LoRa 32 V3):** Procesa las entradas de los módulos *RFID* y *GNSS*, y envía datos mediante *LoRaWAN*.
- **Módulo *RFID*:** Detecta y autentica el *tag* del usuario. Si el *UID* del *tag* coincide con uno de los usuarios autorizados, se habilita el sistema de arranque.
- **Módulo *GNSS*:** Proporciona las coordenadas de geolocalización, que se envían junto con el *UID* autenticado para monitoreo remoto.
- **Relé:** Controla el flujo de energía hacia el encendido de la motocicleta. Si el *tag* *RFID* no es reconocido o se retira del rango, el relé corta la energía, evitando que la motocicleta arranque.

#### 4.1.3. Arquitectura de comunicación externa

La Figura 4.3 muestra cómo los datos capturados por el prototipo se procesan para cumplir con los requisitos de monitoreo remoto en tiempo real.

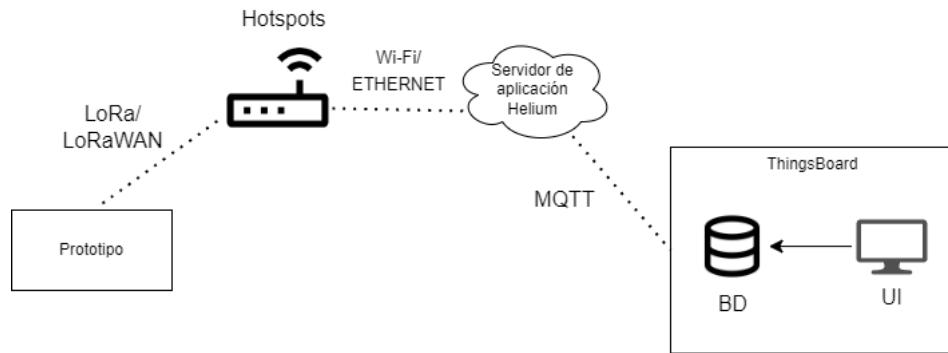


Figura 4.3: Arquitectura de comunicación y monitoreo.

- **Prototipo:** Recopila datos del *RFID* y las coordenadas *GNSS*. El microcontrolador envía esta información utilizando el protocolo *LoRaWAN*.
- **Hotspots Helium:** Actúan como puntos de acceso, recibiendo los datos enviados por el prototipo y transmitiéndolos al servidor de aplicación.
- **Servidor Helium:** Procesa los datos recibidos y los envía mediante el protocolo *MQTT* a la plataforma *ThingsBoard*.

- **ThingsBoard (UI):** Plataforma que permite la visualización gráfica de los datos en tiempo real. Los usuarios pueden monitorear la ubicación de la motocicleta, su estado de autenticación, y recibir alertas en caso de eventos inusuales, como intentos de robo.

## 4.2. Módulo de autenticación y control

En esta fase, se desarrollaron las funciones principales para garantizar la autenticación del usuario antes de permitir el arranque de la motocicleta. El sistema está diseñado para validar la presencia de un *tag RFID* autorizado; solo cuando se detecta un *tag* registrado se permite el arranque del motor de la motocicleta. Si el *tag* no es detectado o es removido en algún momento, el sistema corta el suministro de corriente al motor, bloqueando su funcionamiento. Este mecanismo proporciona una primera capa de seguridad para prevenir el uso no autorizado de la motocicleta.

### 4.2.1. Requisitos del sistema

Para implementar esta funcionalidad, se definieron los siguientes requisitos:

- **Comparación de UID:** El sistema debe comparar el UID del *tag* detectado con una lista de UID autorizados.
- **Verificación en rango:** Una vez autorizado el *tag*, el sistema realiza una lectura constante para verificar que el *tag* permanezca en rango.

Considerando estos requisitos, se diseñó el diagrama de flujo que representa el funcionamiento del sistema de autenticación y control (Figura 4.4).

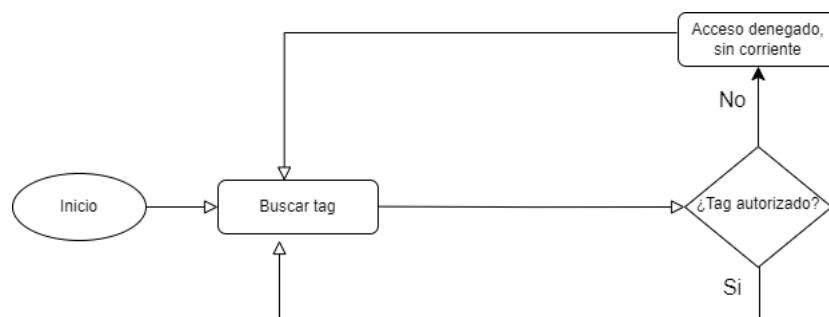


Figura 4.4: Diagrama de flujo de autenticación y control.

#### 4.2.2. Conexión del módulo *RFID*

Para la comunicación entre el módulo *RFID RC522* y la placa de desarrollo *WiFi LoRa 32 V3* se empleó el protocolo SPI. Los pines de conexión utilizados fueron:

- **CS:** Pin 34.
- **MOSI:** Pin 35.
- **CLK:** Pin 36.
- **MISO:** Pin 26.
- **RESET:** Pin 33.
- **VCC y GND:** Pines de alimentación.

En la Figura 4.5 se muestra el diagrama de conexiones utilizadas para esta configuración.

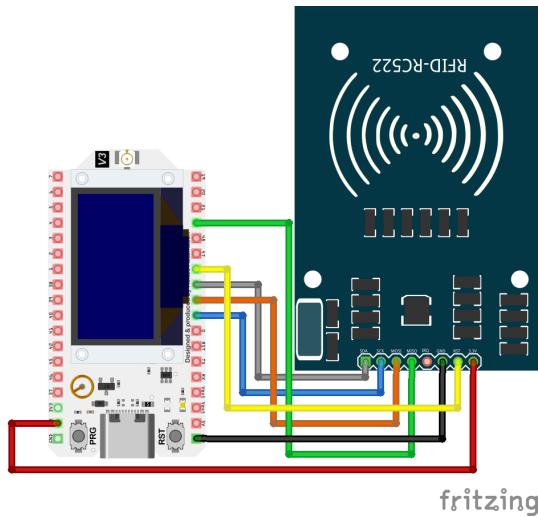


Figura 4.5: Diagrama de conexiones entre WiFi LoRa 32 V3 y RFID RC522.

#### 4.2.3. Primera versión del código: implementación básica en el bucle principal

La primera versión del código implementa un sistema de autenticación y monitoreo continuo de un *tag RFID*, permitiendo el acceso únicamente cuando se detecta la presencia de un *tag* autorizado. En este caso, se definieron

dos UIDs correspondientes a *Usuario1* y *Usuario2*. Al leer el UID del *tag*, el sistema lo compara con estos valores autorizados para conceder o denegar el acceso.

### Autenticación del *tag*

- Si **tagAutenticada** es **falso**, el sistema permanece en espera, buscando la presencia de un *tag RFID*.
- Cuando se detecta un *tag*, se lee su UID y se utiliza la función **comparaUID()** para verificar si coincide con uno de los usuarios autorizados. Si coincide, se muestra un mensaje de bienvenida y **tagAutenticada** se establece en **verdadero**.
- Si el UID no es reconocido, se muestra un mensaje indicando que el *tag* no es válido.

### Verificación de presencia del *tag*

- Si **tagAutenticada** es **verdadero**, el sistema verifica constantemente si el *tag* autorizado permanece en rango.
- Si el tiempo desde la última detección del *tag* excede un límite definido (**tiempoCorte**), el sistema corta el suministro de corriente al motor.
- Si el *tag* sigue en rango, se actualiza el tiempo de última lectura y el motor continúa en funcionamiento.

#### 4.2.4. Segunda versión del código: integración de FreeRTOS y modularización

En esta versión, se incorporó FreeRTOS para gestionar de manera concurrente la autenticación del *tag RFID* y la verificación de su presencia. Además, el código se modularizó en archivos específicos para facilitar su mantenimiento y escalabilidad:

- **Archivo rfidfunciones.h:** Declaraciones de variables y funciones relacionadas con el módulo *RFID*.
- **Archivo rfidfunciones.cpp:** Implementación de funciones y tareas de FreeRTOS para autenticación y monitoreo.
- **Archivo principal RFIDFreeRTOS.ino:** Inicialización de tareas de FreeRTOS y configuración de parámetros principales.

## Tareas implementadas con FreeRTOS

### Tarea 1: tareaDeteccionTag

- Busca la presencia de un *tag RFID*. Si se detecta, verifica su autenticidad mediante la función `autenticarTag()`.
- Monitorea la permanencia del *tag* en rango. Si el *tag* es detectado nuevamente, actualiza el tiempo de última lectura.
- Notifica a la tarea de verificación del tiempo mediante un semáforo (`tagDetectadaSemaphore`).

### Tarea 2: tareaVerificarTiempo

- Espera la señal del semáforo para ejecutar la función `verificarTiempo()`, que comprueba si el tiempo transcurrido desde la última detección supera el límite definido (`tiempoCorte`).
- Si el tiempo excede el límite, se corta el suministro de corriente y se establece `tagAutenticada` en falso.

Esta versión proporciona una estructura modular, mejorando la eficiencia y permitiendo un control más preciso del sistema. El uso de *FreeRTOS* optimiza la gestión de tareas críticas, asegurando la sincronización y escalabilidad del sistema.

## 4.3. Módulo de captura y transmisión de datos

Para llevar a cabo la captura y transmisión de datos en tiempo real, se desarrollaron códigos específicos que permiten la adquisición de coordenadas de ubicación mediante el módulo *GNSS* y su transmisión eficiente a través de *LoRaWAN*, integrando ambos subsistemas en el funcionamiento general del prototipo.

Este proceso se dividió en tres fases:

- Implementación básica para obtención de datos de las coordenadas en tiempo real.
- Implementación y configuración del módulo *LoRaWAN* para la transmisión de datos.

- Integración de la comunicación entre los módulos *GNSS* y *LoRaWAN*.

Para la comunicación entre el módulo *GNSS* y la placa de desarrollo *WiFi LoRa 32 V3*, se empleó el protocolo *UART*. Los pines de conexión utilizados fueron el pin 45 (RX), pin 46 (TX), y los pines de alimentación *VCC* y *GND*. En la Figura 4.6, se muestra el diagrama de conexiones utilizado.

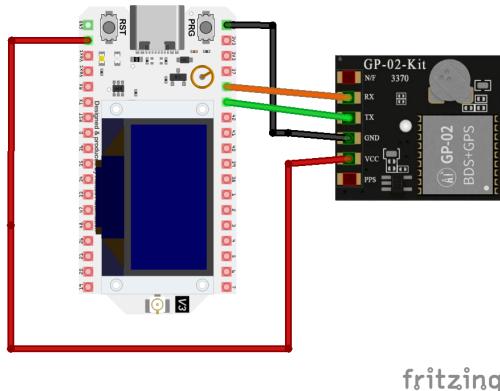


Figura 4.6: Diagrama de conexiones entre *WiFi LoRa 32 V3* y *GNSS*.

#### 4.3.1. Implementación básica para obtención de datos de las coordenadas en tiempo real

Una vez conectado el módulo *GNSS* a la placa de desarrollo, se implementó un código para obtener y mostrar en tiempo real los datos de ubicación (latitud y longitud), junto con la fecha y hora actual. Este código se basa en la librería *TinyGPSPlus*, que permite interpretar los datos en formato *NMEA* recibidos desde el GPS. A continuación, se describe la implementación:

##### Función `setup()`

Durante la fase de inicialización, la función `setup()` se encarga de:

- Configurar el monitor serial (`Serial`) a 115200 baudios para facilitar la visualización de los datos en una terminal serial.
- Configurar `Serial1`, el puerto serial dedicado a la comunicación con el GPS, con los pines y velocidad previamente definidos.

## Proceso de captura de datos en la función `loop()`

En la función `loop()`, el sistema verifica continuamente si hay datos disponibles en el puerto `Serial1`, donde el módulo *GPS* envía sus mensajes de datos. El flujo de trabajo en esta función es el siguiente:

### Recepción y decodificación de datos:

- Cuando `Serial1` tiene datos disponibles, estos se procesan utilizando la función `gps.encode()`. Esta función decodifica los mensajes de datos en formato *NMEA* provenientes del *GPS* y almacena la información en la instancia `gps` de la clase *TinyGPSPlus*.

### Verificación de la validez de los datos:

- Si `gps.encode()` detecta un mensaje completo de datos *GPS* válido, se llama a la función `displayInfo()`, que muestra los datos procesados (ubicación, fecha y hora) en el monitor serial.
- Si no se reciben suficientes caracteres de *GPS* dentro de los primeros 5 segundos (menos de 10 caracteres), se muestra un mensaje de error en el monitor serial indicando que no se detecta el *GPS*. Esto podría deberse a un problema de cableado, en cuyo caso el sistema se detiene para evitar lecturas incorrectas.

### Función `displayInfo()`

La función `displayInfo()` se encarga de presentar los datos de ubicación, fecha y hora en el monitor serial.

### Visualización de la ubicación (latitud y longitud):

- Si los datos de ubicación son válidos (`gps.location.isValid()`), se muestran la latitud y longitud con una precisión de hasta seis decimales. Esta precisión permite obtener coordenadas exactas del dispositivo en el mapa.
- Si los datos no son válidos, se muestra el mensaje “**INVALIDO**” en lugar de las coordenadas.

### **Visualización de la fecha:**

- Si el *GPS* proporciona una fecha válida (`gps.date.isValid()`), esta se presenta en formato MM/DD/AAAA.
- Si la fecha no es válida o no se ha recibido, se muestra “INVALIDO” en su lugar.

### **Visualización de la hora:**

- La hora se muestra en formato HH:MM:SS.CC (horas, minutos, segundos y centésimas de segundo) si los datos son válidos (`gps.time.isValid()`).
- Si la hora no es válida, se muestra “INVALIDO”.

### **4.3.2. Implementación y configuración del módulo LoRaWAN para la transmisión de datos**

En esta fase, se llevó a cabo la configuración del sistema de transmisión de datos mediante la consola *Helium (Legacy)* [61], junto con la placa de desarrollo Heltec WiFi LoRa 32 V3. A continuación, se detallan los pasos y configuraciones necesarios para establecer la comunicación entre el dispositivo y la red *Helium*, así como los procesos requeridos para su correcto funcionamiento.

#### **Uso de la consola *Helium***

La consola *Helium* se empleó para configurar los parámetros de los nodos, verificar el estado de la red y monitorear las transmisiones de datos. A través de esta plataforma, se prepararon los entornos necesarios para visualizar la conectividad entre los dispositivos y la consola, incluyendo el registro de los dispositivos y la verificación de su comunicación con los *gateways*.

El proceso comenzó con el acceso a una organización en la que se habían otorgado permisos de usuario. Tras recibir estos permisos, se procedió a crear una cuenta en la plataforma *Helium*. Completado el registro y acceso a la cuenta, fue posible crear un nodo en la red *Helium*, configurado específicamente para recibir los datos transmitidos por el dispositivo. La Figura 4.7 ilustra el entorno de configuración en la consola *Helium*, donde se han resaltado con recuadros negros los elementos clave, incluyendo las credenciales principales necesarias para establecer la conexión.

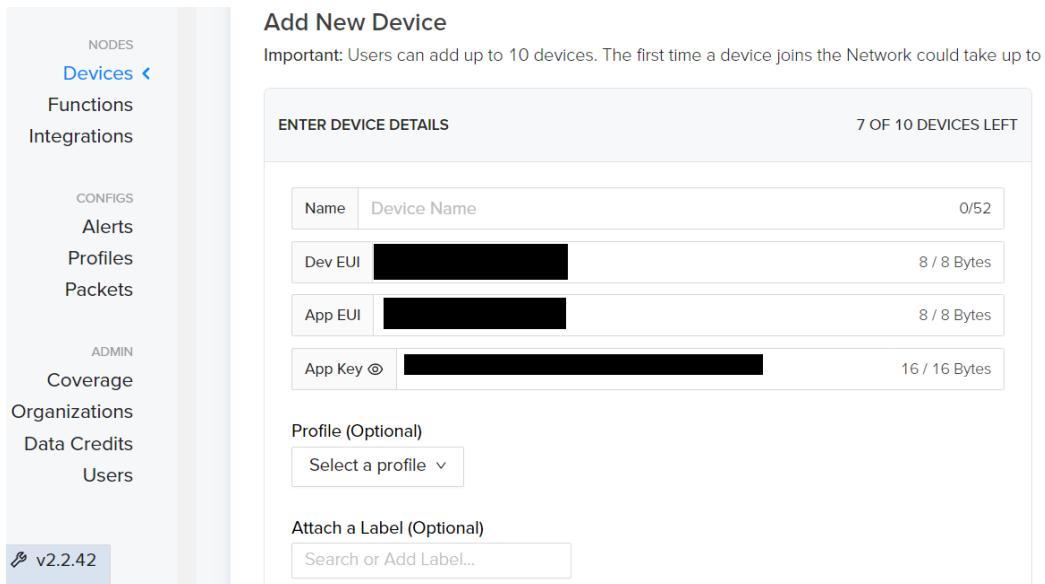


Figura 4.7: Entorno para crear dispositivo y configuración de las credenciales necesarias para el nodo.

### Configuración del dispositivo Heltec WiFi LoRa 32 V3

Luego de crear el nodo y obtener las credenciales necesarias, el siguiente paso fue la configuración de la placa de desarrollo Heltec WiFi LoRa 32 V3. Para la implementación inicial, se utilizó la documentación oficial de *Heltec Automation* [62], que proporciona instrucciones detalladas paso a paso sobre la configuración del módulo *LoRaWAN*, así como las librerías requeridas para su funcionamiento.

Una vez completada esta configuración, se consultó la guía oficial de *Helium* [63], para su integración en el entorno Arduino IDE, adaptándola a las especificaciones particulares de la placa. Esta guía proporcionó los pasos necesarios para registrar las credenciales generadas, permitiendo que el dispositivo enviara datos en *uplink* a la red *Helium*. En la Figura 4.8, se muestran los ajustes de los parámetros necesarios para la comunicación con el servidor *LoRaWAN*.

```
2  
3 #include "LoRaWan_APP.h"  
4  
5 /* OTAA para */  
6 uint8_t devEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };  
7 uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };  
8 uint8_t appKey[] = { 0x88, 0x66, 0x01 };  
9  
10 /* ABP para */  
11 uint8_t nwkKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d, 0x11, 0x18, 0x1e, 0x1e, 0xc7, 0xda, 0x85 };  
12 uint8_t appKey[] = { 0xd2, 0x7c, 0x79, 0x75, 0x9c, 0x9d, 0x9a, 0x9f, 0x95, 0x9e, 0x4a, 0x77, 0x9d, 0x16, 0xf, 0x71 };
```

Figura 4.8: Ajuste de los parámetros necesarios en el dispositivo para la comunicación con el servidor *LoRaWAN*.

#### **4.3.3. Implementación de la comunicación entre los módulos *GNSS* y *LoRaWAN***

Una vez configurados y validados de manera independiente tanto el módulo *GNSS* como el sistema de comunicación *LoRaWAN*, se procedió con la integración de ambos componentes. El objetivo de esta fase fue garantizar que las coordenadas obtenidas por el módulo *GNSS* se transmitieran de manera eficiente a través de la red *LoRaWAN*. Para ello, se realizaron los ajustes necesarios en el código, asegurando que la captura de los datos de ubicación y su envío a través de *LoRaWAN* ocurrieran de manera sincronizada y sin pérdida de información.

En esta sección, se describen en detalle los procesos y el funcionamiento del código ajustado para integrar los sistemas de geolocalización y transmisión de datos.

Como primer paso, se diseñó un diagrama de flujo para representar el funcionamiento del sistema integrado, el cual se puede observar en la Figura 4.9.

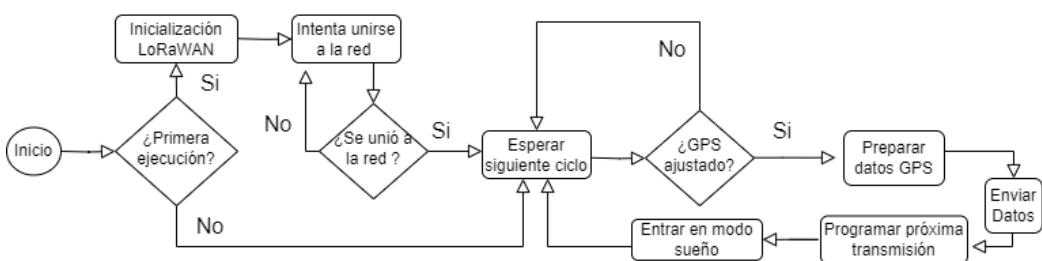


Figura 4.9: Diagrama de flujo de transmisión de datos por *LoRaWAN*.

Una vez diseñado el diagrama de flujo, se procedió a desarrollar una nueva versión del código, reutilizando los fragmentos de código de las fases

anteriores e incorporando los ajustes necesarios. A continuación, se describe la implementación de este código.

El programa inicia con la inclusión de las librerías esenciales: `LoRaWANAPP.h`, para gestionar la comunicación mediante *LoRaWAN*, y `TinyGPSPlus.h`, para manipular los datos obtenidos del módulo *GNSS*. Luego, se configuran los pines de acuerdo con las definiciones establecidas en la primera fase del proyecto.

A continuación, se definen los parámetros de conexión de *LoRaWAN*. La conexión puede configurarse de dos maneras: mediante *OTAA* (*Over-The-Air Activation*) o *ABP* (*Activation By Personalization*). Para ello, se especifican las claves necesarias para cada método (`devEui`, `appEui`, `AppKey` para *OTAA*, y `NwkSKey`, `AppSKey`, `devAddr` para *ABP*). También se definen los canales de comunicación y la región de operación.

Posteriormente, se establece un ciclo de transmisión de datos cada 60 segundos (`appTxDutyCycle` a 60000) y se especifican otros parámetros, como el uso de *ADR* (*Adaptive Data Rate*) y la configuración de los mensajes como confirmados o no confirmados, según los requisitos del sistema.

La función `prepareTxFame` se encarga de preparar los datos *GPS* para su envío. Primero, el programa espera hasta obtener una ubicación válida (`gps.location.isValid()`). Una vez que la señal es válida, la función extrae la latitud y longitud y las almacena en un *buffer* de datos (`appData`) en el formato adecuado para su transmisión mediante *LoRaWAN*.

Al comenzar el ciclo `setup`, el programa inicializa la comunicación serie y *LoRaWAN*. Adicionalmente, se utiliza una variable (`firstrun`) para verificar si es la primera vez que el dispositivo se configura en el *MCU*, mostrando un mensaje en pantalla para confirmar la inicialización.

En el ciclo `loop`, la lógica del programa sigue una estructura de máquina de estados, que permite gestionar cada fase de la operación *LoRaWAN*. Los estados principales son:

- **DEVICE STATE INIT:** Inicializa el módulo *LoRaWAN* y realiza las configuraciones básicas.
- **DEVICE STATE JOIN:** Intenta unirse a la red *LoRaWAN*, mostrando mensajes de estado.
- **DEVICE STATE SEND:** Llama a `prepareTxFame` para obtener y empaquetar los datos de *GPS*, y luego envía la información.
- **DEVICE STATE CYCLE:** Calcula el tiempo hasta la próxima transmisión, usando `appTxDutyCycle` como base.

- **DEVICE STATE SLEEP:** Pone al dispositivo en modo de bajo consumo para conservar energía hasta la siguiente transmisión.

Finalmente, la función `displayInfo` muestra en el monitor serie la información de ubicación, fecha y hora del *GPS*. En caso de que los datos no estén disponibles, se muestra un mensaje de *INVALIDO*.

## 4.4. Módulo de interfaz gráfica y monitoreo

El módulo de interfaz gráfica y monitoreo constituye un componente esencial del sistema, proporcionando una plataforma interactiva *IoT* para visualizar en tiempo real la ubicación de la motocicleta y otros parámetros relevantes. Para esta implementación, se utilizó *ThingsBoard Community Edition*, que se conectó a la consola *Helium*, permitiendo la recepción y visualización de los datos transmitidos desde la red. La estructura del módulo incluye un servidor local que procesa y organiza la información en una interfaz gráfica accesible para el usuario final.

La implementación de este módulo se dividió en dos fases principales. La primera fase consistió en la configuración de *ThingsBoard* como receptor de datos, habilitando el procesamiento y la visualización de la información recibida. La segunda fase abordó la integración de la consola *Helium* con *ThingsBoard* mediante el protocolo *MQTT*, configurando un flujo de datos en tiempo real entre el dispositivo y el servidor local.

En las siguientes secciones, se detallan cada una de estas fases:

### 4.4.1. Configuración de *ThingsBoard*

En esta fase, se llevó a cabo la instalación y configuración de *ThingsBoard* de manera local, siguiendo las instrucciones de la documentación oficial para asegurar una implementación inicial estable [64]. Una vez completada esta instalación, se realizaron configuraciones personalizadas con el objetivo de adaptar la plataforma a los requisitos específicos del sistema de monitoreo.

Los pasos realizados en esta etapa incluyeron:

#### Creación de dispositivos y configuración de conectividad

En *ThingsBoard*, se añadieron los dispositivos que transmitirían datos al sistema. En este caso, se creó un dispositivo llamado *Poli\_Moto* (ver Figura 4.10) y se le asignó un perfil *MQTT*. Esta configuración permite que el dispositivo envíe datos de telemetría a través de la consola *Helium* hacia *ThingsBoard*.

Se definió un perfil de transporte *MQTT* con los temas de telemetría y atributos configurados para recibir datos en formato *JSON* (ver Figura 4.11). Esto asegura que los datos de ubicación y estado se transmitan de manera continua y en tiempo real.

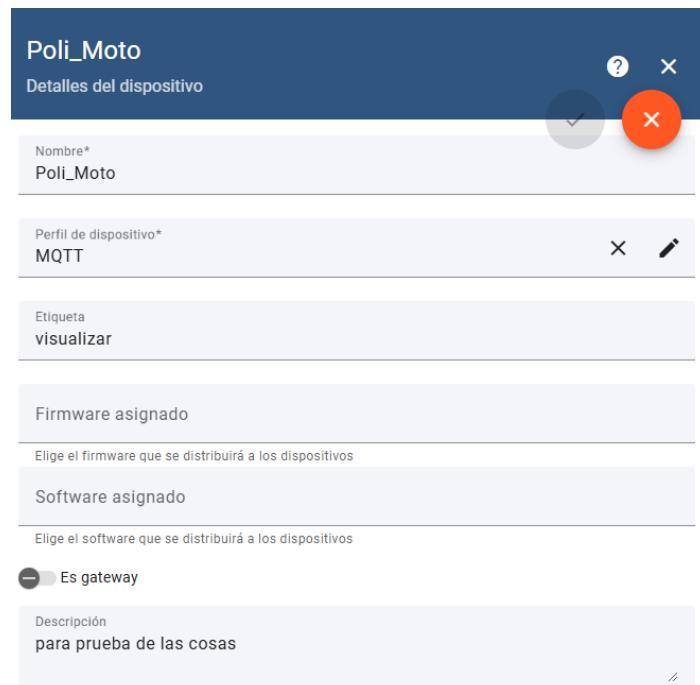


Figura 4.10: Detalles del dispositivo ThingsBoard.

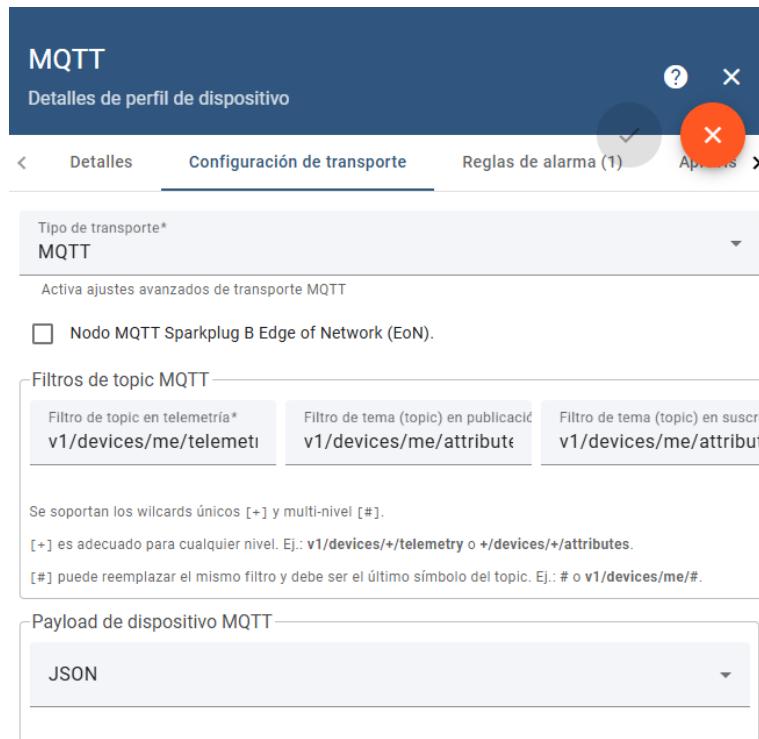


Figura 4.11: Perfil de dispositivo ThingsBoard.

### Gestión de Usuarios y Roles de Acceso

Se crearon diferentes usuarios y clientes para facilitar el acceso y administración de los datos en el sistema. En la configuración de clientes, se creó un cliente de prueba llamado *Cliente\_Prueta*, al cual se le asignaron dispositivos específicos y se configuraron sus permisos de acceso (ver Figuras 4.12 y 4.13).

Se configuraron usuarios con permisos exclusivos para acceder a dispositivos asignados, permitiendo una gestión controlada y personalizada de la información disponible.

	Fecha de creación	Nombre	Apellido	Email
<input type="checkbox"/>	2024-09-21 23:01:15	Mario	Barreto	kutumaio@gmail.com

Figura 4.12: Cliente con usuario creado en ThingsBoard.

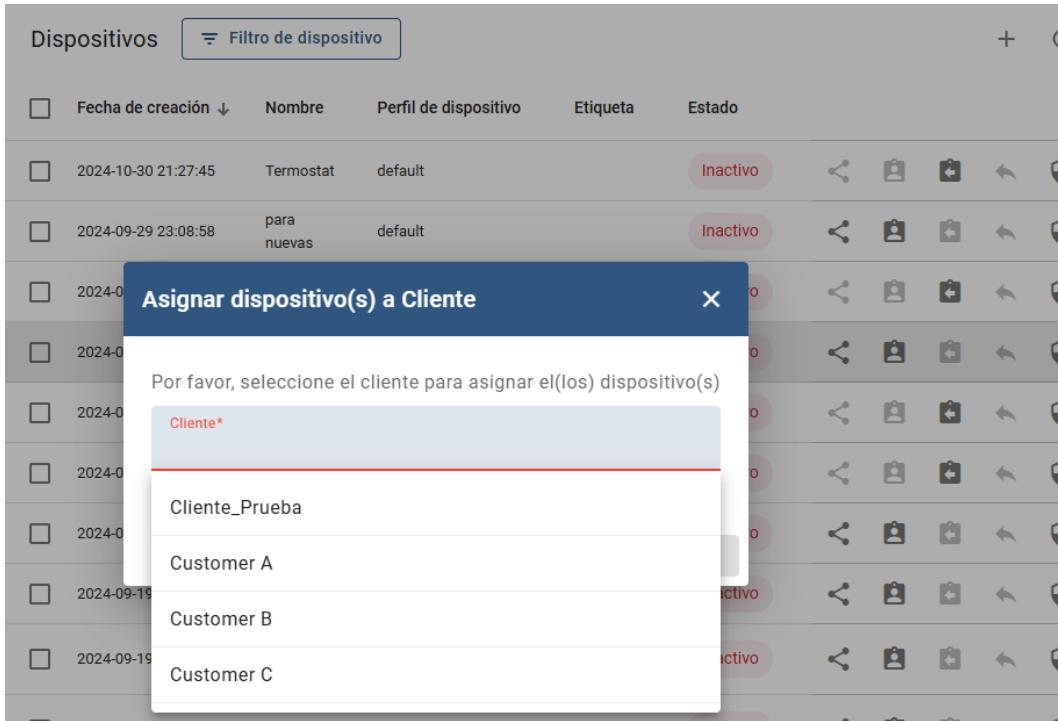


Figura 4.13: Asignación de dispositivos a clientes en ThingsBoard.

### Diseño del Tablero Interactivo

Se diseñó un tablero personalizado titulado *Mapa\_Mot*, donde se agruparon los datos de los dispositivos en widgets específicos para su monitoreo en tiempo real. Este tablero permite centralizar y visualizar la información de ubicación y otros datos relevantes de los dispositivos conectados.

Se exploraron los *widgets* disponibles en ThingsBoard para elegir aquellos que mejor se adaptaran a las necesidades del sistema (ver Figura 4.14). Se seleccionaron los *widgets* de mapas y gráficos de series de tiempo, esenciales para monitorear la posición geográfica y el historial de movimiento del dispositivo en tiempo real.

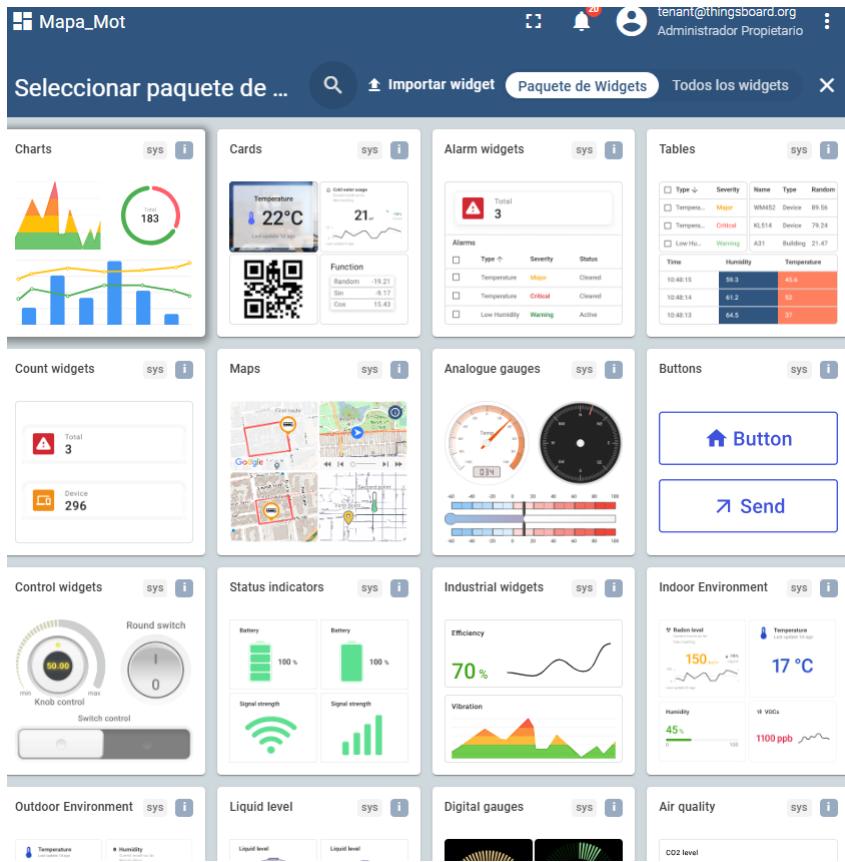


Figura 4.14: *Widgets* disponibles en *ThingsBoard*.

Se añadieron *widgets* de mapa, como *Route Map - OpenStreet* y *Route Map - Google*, que ofrecen una visualización interactiva de la ubicación en tiempo real (ver Figura 4.15). Además, se incorporó un *widget* de tabla de series temporales para analizar datos históricos como latitud y longitud.

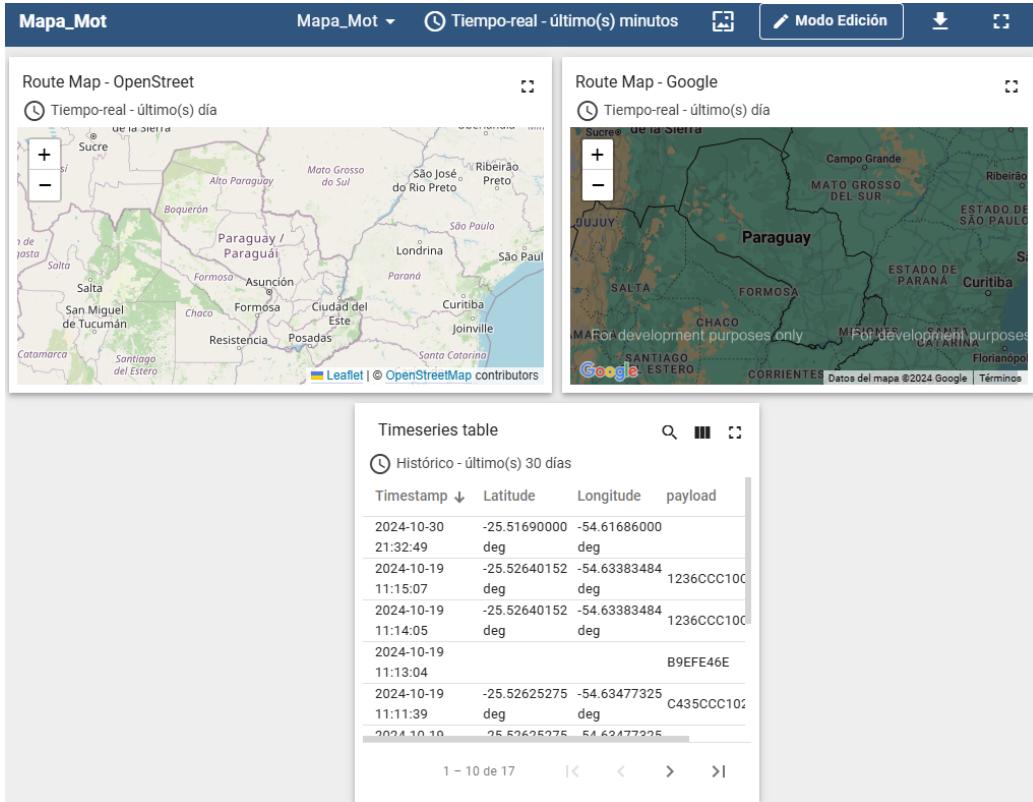


Figura 4.15: *Widgets* utilizados en *ThingsBoard*.

Para garantizar la precisión, se configuró una ventana de tiempo en los *widgets*, visualizando información del último día. Se definieron claves como latitud, longitud y estado de actividad para asegurar que los datos mostrados sean precisos (ver Figura 4.16).

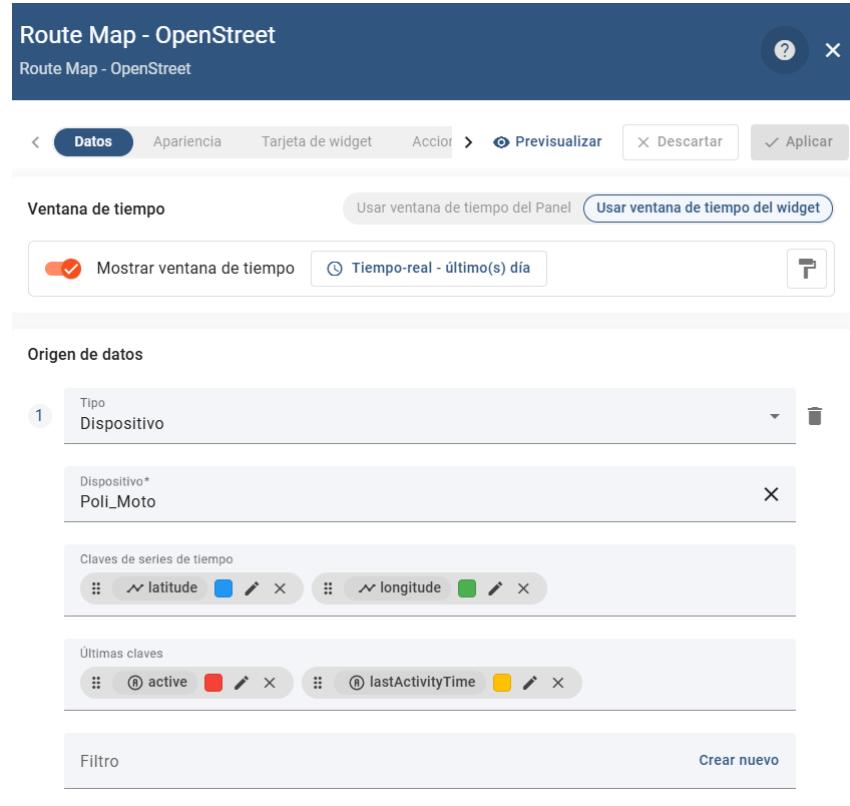


Figura 4.16: Configuraciones en *Widgets* de Mapa.

### Definición de las *Rule Chains* en el Motor de Reglas

En el motor de reglas (*Rule Engine*), se configuraron cadenas de reglas personalizadas para procesar los mensajes recibidos. Se creó una cadena llamada *MQTT*, con nodos como un decodificador base64 y hexadecimal, para transformar los datos antes de almacenarlos en series temporales (ver Figura 4.17).

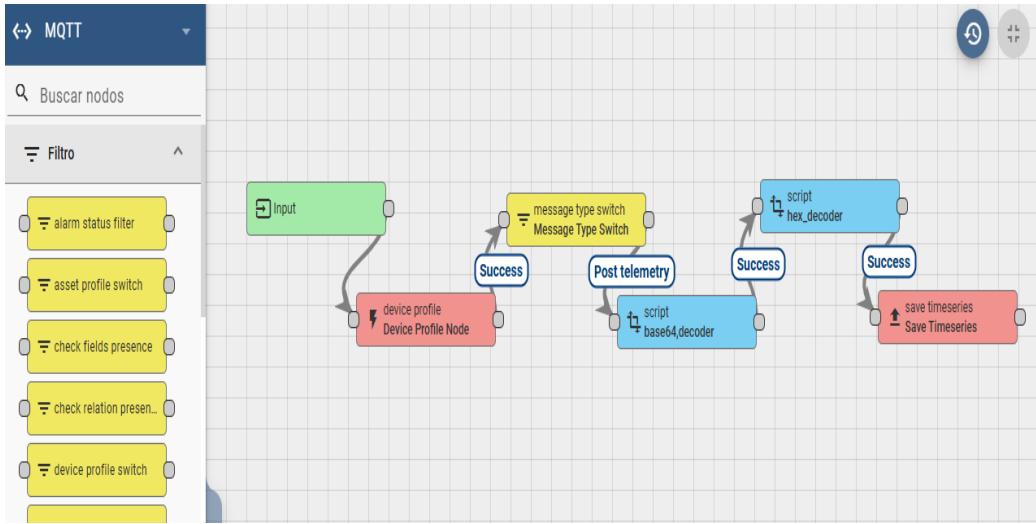


Figura 4.17: Diagrama del motor de reglas *MQTT* en *ThingsBoard*.

### Configuración de Alarmas y Notificaciones

Se configuraron alarmas para generar alertas en eventos específicos, como desconexiones del dispositivo o valores de telemetría fuera de parámetros. Estas alarmas mejoran la capacidad de respuesta del sistema ante eventos críticos.

Todas las configuraciones siguieron las mejores prácticas de la documentación oficial de *ThingsBoard* [65].

#### 4.4.2. Integración de *ThingsBoard* con *Helium*

La siguiente etapa consistió en la integración de *ThingsBoard* con la consola *Helium*. Para integrar estas herramientas y lograr una transmisión de datos en tiempo real, se configuró una integración a través del protocolo *MQTT*. Este proceso permitió recibir datos de telemetría desde dispositivos conectados a la red *Helium* y visualizarlos en el tablero de *ThingsBoard*. A continuación, se detallan los pasos realizados para establecer esta integración.

Primero, se accedió al menú de *Integrations* en la consola de *Helium*, señalado con la flecha (A) en la Figura 4.18. En este menú, se seleccionó la opción para añadir una nueva integración, lo que despliega un conjunto de opciones básicas de integración. Entre estas opciones, se encuentra *MQTT*, identificado con el círculo (B) en la Figura 4.18, que fue el protocolo elegido para transmitir los datos de los dispositivos a *ThingsBoard*.

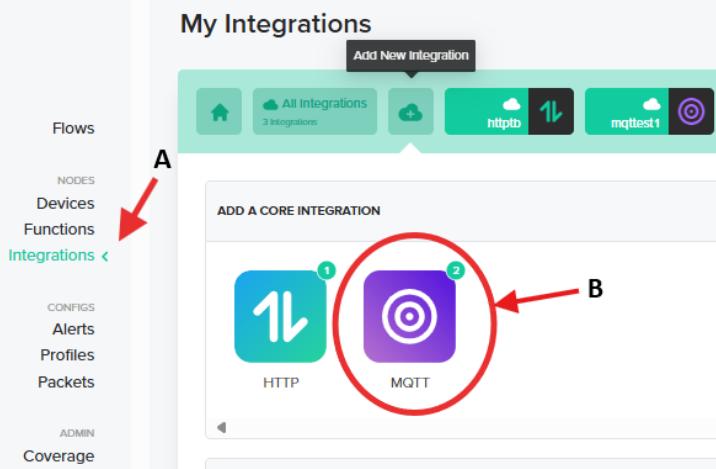


Figura 4.18: Menú de integraciones en la consola de *Helium*.

Una vez seleccionada la opción *MQTT*, se procedió a configurar los detalles de conexión en la consola de *Helium*, como se muestra en la Figura 4.19. En la sección *MQTT Connection Details*, se definieron varios parámetros clave para establecer la comunicación con el servidor de *ThingsBoard*:

En el campo Endpoint, señalado con la etiqueta (A) en la Figura 4.19, se incluyó el token único del dispositivo generado previamente en *ThingsBoard*, el cual se añadió directamente en la URL como credencial de acceso. La dirección IP pública del servidor y el puerto estándar para *MQTT* (1883), marcados con la etiqueta (B) en la Figura 4.19, fueron configurados para garantizar la correcta vinculación con el servicio remoto. Finalmente, se establecieron los temas necesarios para la comunicación señalados con la flecha (C) en la Figura 4.19. El tema *Uplink Topic*, fue configurado como v1/devices/me/telemetry, permitiendo el envío de datos de telemetría desde los dispositivos, el tema *Downlink Topic*, se configuró para recibir comandos y configuraciones desde *ThingsBoard* hacia los dispositivos. Con esta configuración, se garantizó una comunicación bidireccional entre ambos sistemas, facilitando tanto la gestión remota como el monitoreo en tiempo real de los dispositivos conectados.

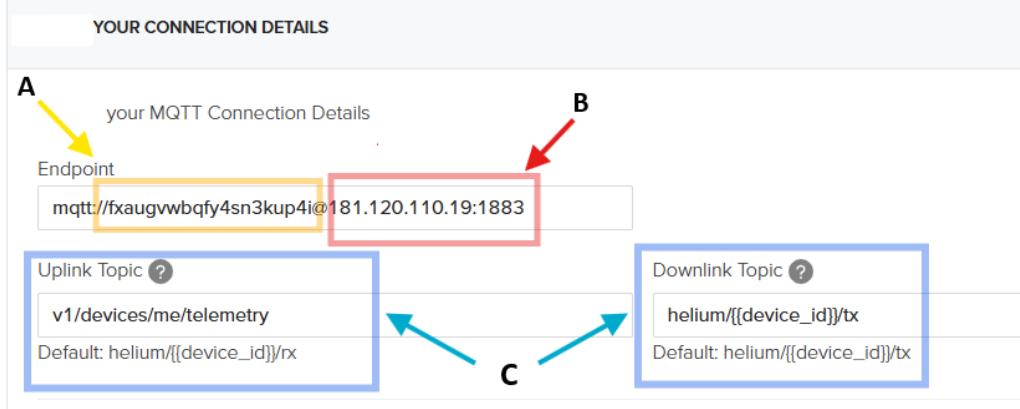


Figura 4.19: Detalles de conexión *MQTT* configurados en la consola de *Helium*.

Después de configurar los detalles de conexión, se procedió a crear un flujo en la consola de *Helium* que conecta los dispositivos a la integración *MQTT\_TB*, como se observa en la Figura 4.20. En el menú Flows (A), se seleccionó la integración previamente configurada, etiquetada como *MQTT\_TB* (B) en la Figura 4.20, y se vinculó con el dispositivo PruebaHeltec. Este flujo garantiza que los datos capturados por el dispositivo se envíen automáticamente a *ThingsBoard* a través de la red *Helium*.

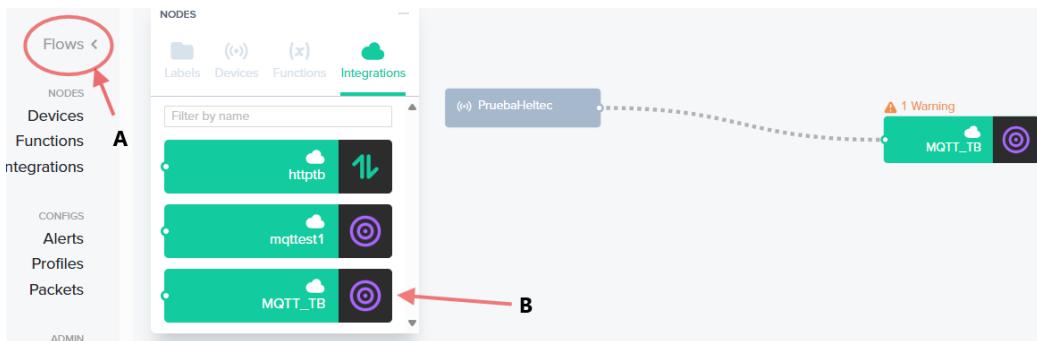


Figura 4.20: Flujo de integración entre el dispositivo PruebaHeltec y *MQTT\_TB*.

Para completar la integración, fue necesario configurar la redirección de puertos en el router de la red local donde se encuentra alojado el servidor **ThingsBoard**. En la sección *IPv4 Port Mapping* del panel de administración

del *router* (ver Figura 4.21, se realizó el mapeo correspondiente para permitir el acceso externo al servidor *MQTT*. Se definió un nombre de mapeo denominado *TBMQTT*, señalado en (A) de la Figura 4.21, que asocia la dirección IP interna del servidor (192.168.100.5) con la IP pública asignada a la red (181.94.228.197) indicado con (B) de la Figura 4.21). Asimismo, identificado con el círculo (C) en la Figura 4.21 ,se estableció el puerto 1883 como punto de entrada para las conexiones externas, y se configuró el protocolo TCP, necesario para garantizar la correcta transmisión de los datos.

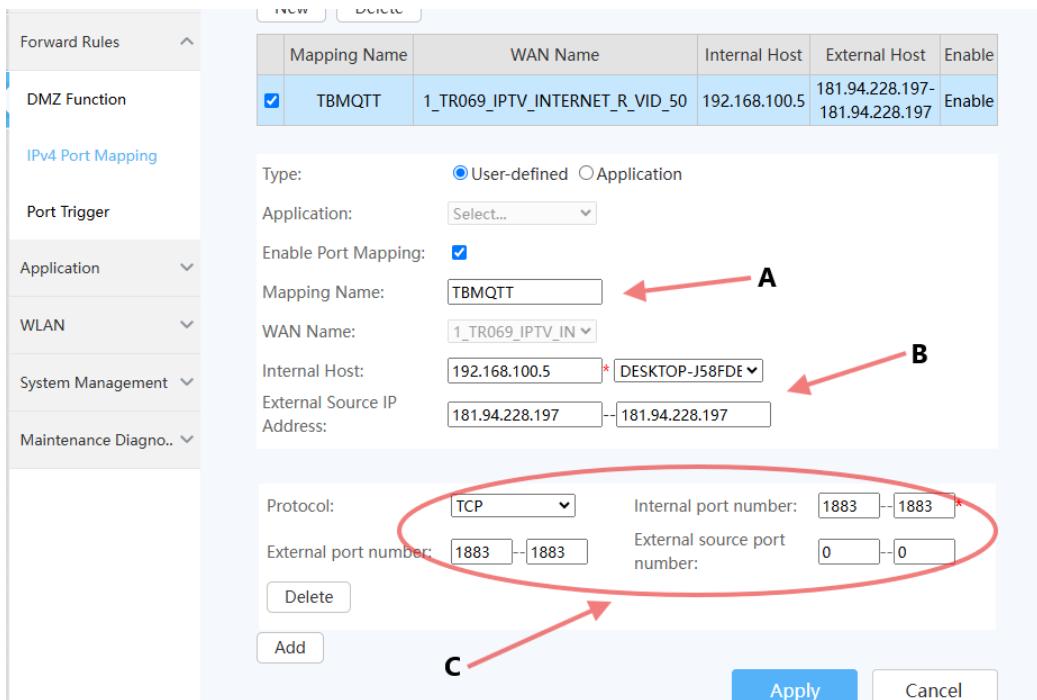


Figura 4.21: Configuraciones en el *router* de la red local.

Con estos ajustes, se completó la integración de *ThingsBoard* con *Helium*, permitiendo que los datos capturados por los dispositivos en la red *Helium* sepuedan transmitir de manera funcional al servidor *ThingsBoard* para su visualización y monitoreo en tiempo real.

## **4.5. Integración de componentes y encapsulamiento**

En esta sección, se describe el proceso de integración de los distintos módulos desarrollados en el sistema y el encapsulamiento final del dispositivo. El objetivo de esta fase es lograr que los módulos de autenticación, control, captura y transmisión de datos trabajen de forma conjunta. Además, se diseñaron un encapsulado protector y estético para asegurar la robustez y usabilidad del dispositivo en su entorno final. A continuación, se detallan las etapas de integración y encapsulado del sistema.

Este proceso se dividió en cuatro etapas:

- Implementación Básica con la Integración Inicial entre los Módulos RFID y LoRaWAN.
- Implementación alternativa con la Integración ajustada entre los Módulos.
- Incorporación de Corte de Energía y Simulación de Arranque
- Modelado y Diseño de Encapsulado 3D.

### **4.5.1. Implementación Básica con Integración Inicial de los Módulos RFID y LoRaWAN**

En esta primera etapa, se integraron los módulos de autenticación RFID y el sistema de comunicación LoRaWAN. El objetivo de esta fase fue asegurar que el sistema pudiera autenticar correctamente los UIDs autorizados y transmitirlos de manera confiable a través de la red LoRaWAN.

Para establecer la comunicación entre el módulo RFID y la placa de desarrollo WiFi LoRa 32 V3, se emplearon las mismas conexiones definidas previamente en el módulo de autenticación y control, lo cual permitió mantener una estructura coherente en el diseño. Esta configuración inicial proporcionó las bases para evaluar el correcto funcionamiento de los módulos integrados y realizar los ajustes necesarios en las etapas siguientes.

### **4.5.2. Implementación alternativa con la Integración ajustada entre los Módulos.**

Tras implementar la primera versión del código, se identificaron conflictos en la comunicación SPI entre los módulos RFID y LoRaWAN. Se observó

que el módulo LoRaWAN utilizaba el protocolo SPI con conexiones definidas internamente en la placa de desarrollo, mientras que el módulo RFID, al conectarse externamente mediante SPI, generaba un solapamiento que impedía el funcionamiento coordinado de ambos sistemas.

Para abordar los problemas detectados, se diseñó una nueva versión del código que incorporó una integración alternativa con el uso de un ESP-WROOM-32 dedicado exclusivamente a gestionar el módulo RFID. Este ajuste tuvo como propósito lograr que los componentes funcionaran sin interferencias en la comunicación, permitiendo una transmisión estable y una sincronización efectiva entre los módulos.

Para establecer la comunicación entre el módulo RFID RC522 y la placa de desarrollo ESP-WROOM-32, se utilizó el protocolo SPI. Las conexiones se realizaron de la siguiente manera: el pin 4 se utilizó como CS (Chip Select), el pin 23(MOSI), el pin 18(SCK), el pin 19(MISO) y el pin 15(RESET). La alimentación se proporcionó a través de los pines VCC y GND.

Para la comunicación entre la placa ESP-WROOM-32 y la Heltec WiFi LoRa 32 V3, se empleó el protocolo UART, con las siguientes conexiones: el pin 48 de la WiFi LoRa 32 V3 se conectó al RX del ESP-WROOM-32, el pin 47 de la WiFi LoRa 32 V3 al TX del ESP-WROOM-32, y los pines GND de ambas placas se unieron para asegurar una referencia común de tierra.

El módulo GNSS se conectó utilizando las mismas conexiones definidas previamente en el módulo de captura y transmisión de datos. En la figura 4.22, se presenta el diagrama de conexiones empleado para esta configuración.

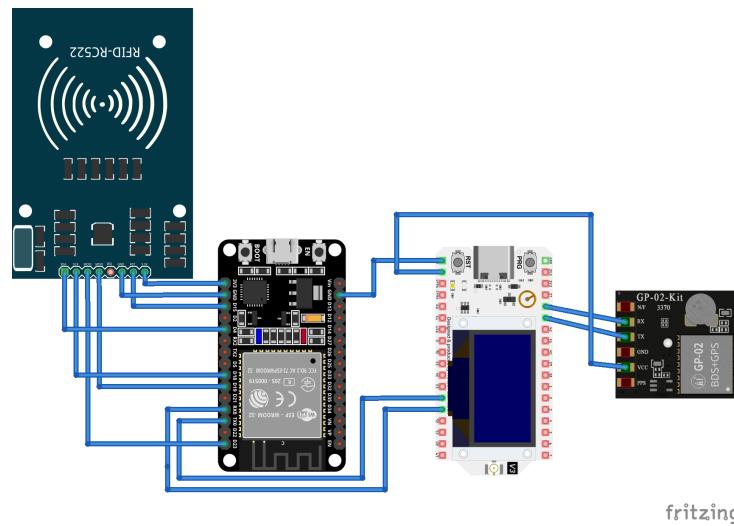


Figura 4.22: Diagrama de conexiones entre RFID RC522, ESP-WROOM-32, WiFi LoRa 32 V3 y GNSS.

Luego de realizar las conexiones, se procedió a ajustar los códigos. En la placa de desarrollo ESP-WROOM-32, se cargó una segunda versión del código previamente desarrollado para el módulo de autenticación y control. Se realizaron modificaciones en el archivo rfidfunciones.cpp, específicamente en la función mostrarUID, que fue adaptada para crear una cadena llamada uidString, la cual almacena el UID en un formato legible.

Durante el procesamiento de cada byte del UID, la función verifica si el valor es menor a 0x10, en ese caso, agrega un 0 al inicio para mantener un formato uniforme. Cada byte se convierte luego a su representación hexadecimal y se añade a uidString, utilizando : como separador entre los bytes. Además, la función almacena cada byte en un array llamado LecturaUID, permitiendo que el UID esté disponible para otras partes del programa cuando sea necesario.

Finalmente, la función imprime el uidString completo en el monitor serial, mostrando el UID del tag RFID en el formato deseado para su fácil identificación y verificación.

En esta fase, se añadieron múltiples tareas al sistema para gestionar de manera coordinada la comunicación entre módulos. Estas tareas, cargadas en la placa de desarrollo Heltec WiFi LoRa 32 V3, se organizaron de la siguiente manera:

- **Tarea UART (Recepción de Datos RFID):** Esta tarea permanece en escucha constante de los datos recibidos desde el módulo RFID a través de la UART. Al recibir un UID, este se compara con los UID previamente autorizados. En caso de que el tag recibido sea autorizado, la tarea ajusta el estado del sistema, habilitando el envío del UID por LoRaWAN y activando la tarea de transmisión de LoRaWAN.
- **Tarea LoRaWAN:** La tarea de LoRaWAN sigue una máquina de estados que controla las diferentes etapas de transmisión, incluyendo la inicialización, la unión a la red (join), el envío de datos y el ciclo de suspensión (sleep). Esta estructura permite que la tarea transmita los datos de forma programada y que el dispositivo entre en suspensión tras cada transmisión, optimizando así el consumo de energía.
- **Tarea GPS:** Esta tarea permanece inactiva hasta que se obtienen coordenadas GPS válidas. Una vez que las coordenadas de latitud y longitud han sido obtenidas y verificadas, la tarea las almacena y las marca como listas para ser enviadas en el siguiente ciclo de transmisión de LoRaWAN. Esto asegura que el sistema envíe únicamente datos de localización precisos.

Con los módulos RFID, GNSS y LoRaWAN completamente integrados, se realizaron ajustes finales en el código para definir el funcionamiento general del sistema, que quedó organizado de la siguiente manera:

- **Detección RFID:** El sistema identifica si un tag RFID autorizado está presente. Al detectar un UID autorizado, el sistema se encarga de enviarlo a través de LoRaWAN.
- **Envío de Datos por LoRaWAN:** Según el tipo de mensaje seleccionado, puede enviar el UID del RFID, otros datos relevantes o, si están disponibles, las coordenadas GPS.
- **Ciclo de LoRaWAN:** El sistema opera en un ciclo de transmisión periódico. Durante cada ciclo, envía los datos y luego entra en modo de suspensión (sleep) hasta el siguiente ciclo, optimizando así el consumo energético.
- **Validación de Coordenadas GPS:** El sistema realiza intentos continuos de obtener coordenadas GPS válidas. Una vez que estas coordenadas son capturadas con éxito, se preparan para ser enviadas en el próximo ciclo de transmisión de LoRaWAN.

#### 4.5.3. Incorporación de Corte de Energía y de Arranque

En esta fase, se llevaron a cabo los ajustes finales mediante la incorporación de un módulo de relé, el cual se sincronizó con el módulo RFID para gestionar el corte de energía y el arranque del sistema. Para implementar esta funcionalidad, se añadió una nueva tarea al código, denominada vTaskCorriente. Esta tarea permanece a la espera de recibir una notificación que indique el estado del relé. Cuando la notificación recibida desde la tarea UART (Recepción de Datos RFID) es positiva, el relé se activa, permitiendo el arranque del sistema, en caso contrario, el relé se desactiva, interrumpiendo la corriente y apagando el sistema. Para ilustrar la estructura y conexiones del sistema, se diseñó un diagrama esquemático completo, que se presenta en la Figura 4.23.

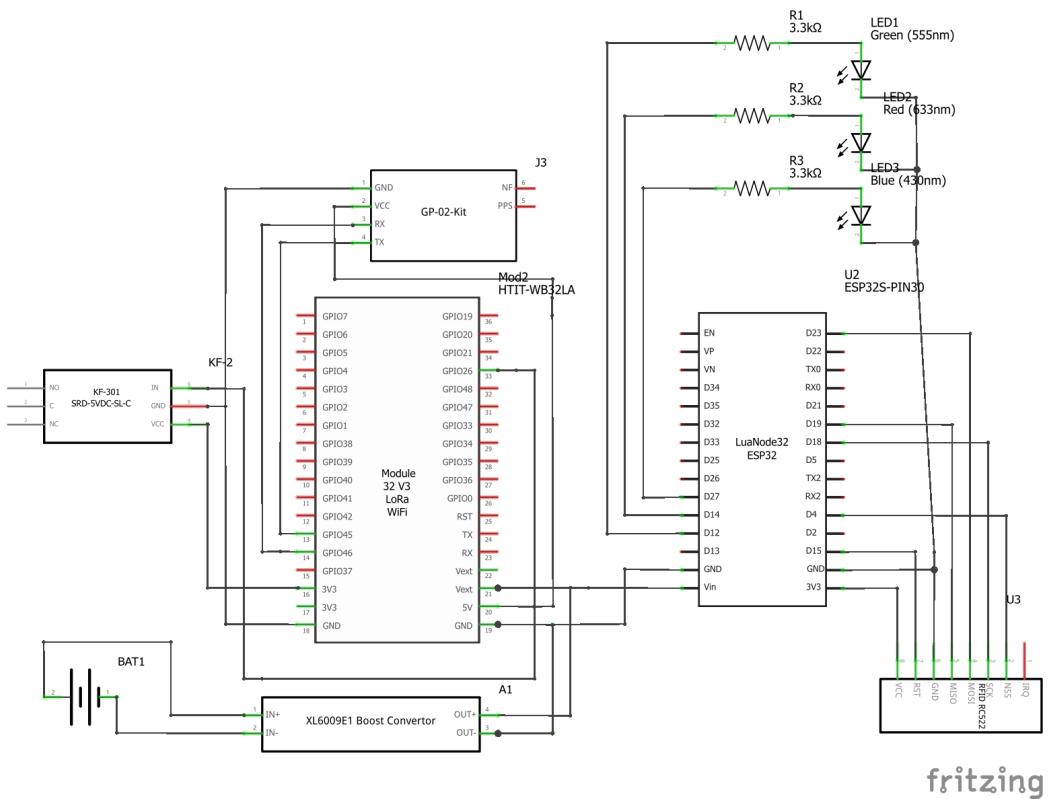


Figura 4.23: Diagrama de conexiones de todos los módulos.

Además, se integraron tres LEDs para indicar los estados del sistema. El LED rojo realiza un “self-check” del módulo RFID, indicando que este aún no está listo. Una vez completada la inicialización del RFID, el LED verde se enciende, señalizando que el sistema está preparado para la lectura. Cuando un tag es detectado y autorizado, el LED azul se enciende para confirmar el acceso. Esta señalización mediante LEDs facilita el monitoreo visual del estado y funcionamiento del sistema en tiempo real.

Por último, se añadió un nuevo tipo de mensaje al sistema de transmisión por LoRaWAN para alertar en caso de desconexión, informando sobre el estado de corte de energía.

#### 4.5.4. Modelado y Diseño de Encapsulado 3D.

En esta fase, se comenzó definiendo la disposición y estructura de los componentes en el sistema. En primer lugar, se determinó la ubicación óptima de cada módulo, de manera que los elementos clave estuvieran organizados

de forma compacta y funcional. Este paso fue esencial para facilitar el acceso a los conectores y puertos de cada componente, optimizando el espacio y garantizando una configuración estructurada.

En la Figura 4.24, se presenta la distribución de los componentes, donde puede observarse cómo cada módulo está colocado estratégicamente para minimizar el uso de cables y asegurar una accesibilidad adecuada. Esta disposición garantiza que los puertos y conexiones principales sean accesibles para su mantenimiento.

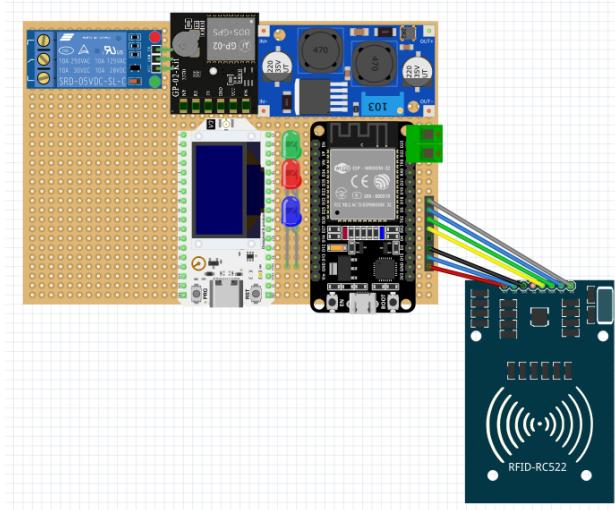


Figura 4.24: Disposición estructural de los componentes.

Con la disposición estructural establecida, se avanzó a la etapa de diseño del encapsulado 3D mediante el software Blender. Este encapsulado fue modelado específicamente para proteger los componentes, adaptándose a sus dimensiones y ubicación en el shield. La estructura fue diseñada para proporcionar una cobertura robusta y duradera, evitando el movimiento de los componentes internos y protegiéndolos de factores externos.

En la figura 4.25, se puede observar el modelo 3D del encapsulado final, que incluye aperturas específicas para cada puerto y conector, así como detalles adicionales para facilitar el montaje y desmontaje de componentes en caso de futuras modificaciones o mantenimientos.

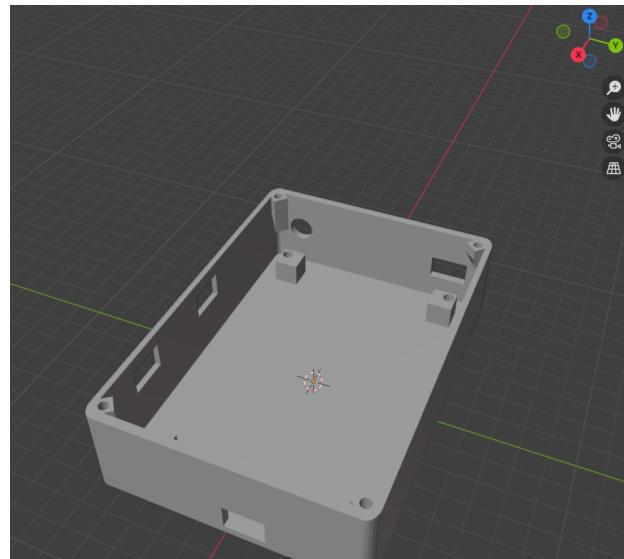


Figura 4.25: Base del encapsulado.

En la figura 4.26, se observa el diseño de una tapa destinada a cubrir el encapsulado, proporcionando una capa adicional de protección para los componentes internos. Esta tapa fue modelada para ajustarse de forma precisa al contorno del encapsulado, incluyendo accesos a los conectores que permiten la salida de los cables necesarios para el funcionamiento del sistema.

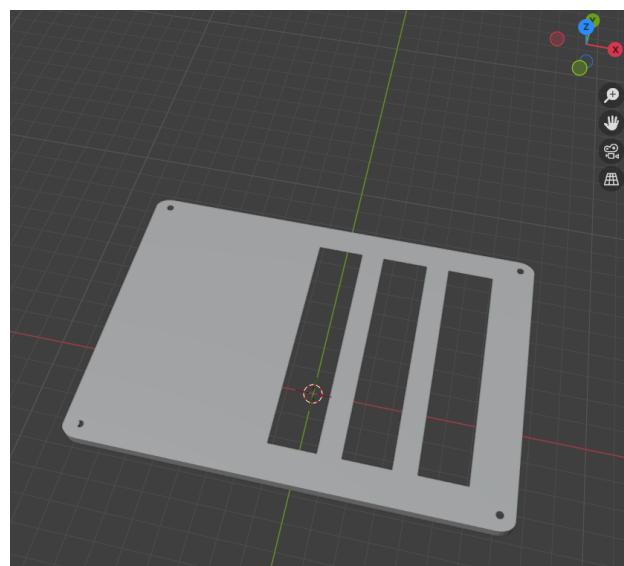


Figura 4.26: Parte superior del encapsulado.

Adicionalmente, se desarrolló un diseño 3D complementario para el módu-

lo RFID con el propósito de satisfacer la necesidad de identificación del usuario y garantizar una lectura continua del tag. En la Figura 4.27, se muestra el diseño del encapsulado, el cual incorpora un orificio estratégicamente ubicado para alojar el tag en una posición fija. Este diseño asegura su permanencia dentro del rango de lectura, optimizando la funcionalidad del sistema y mejorando su integración con los demás componentes.

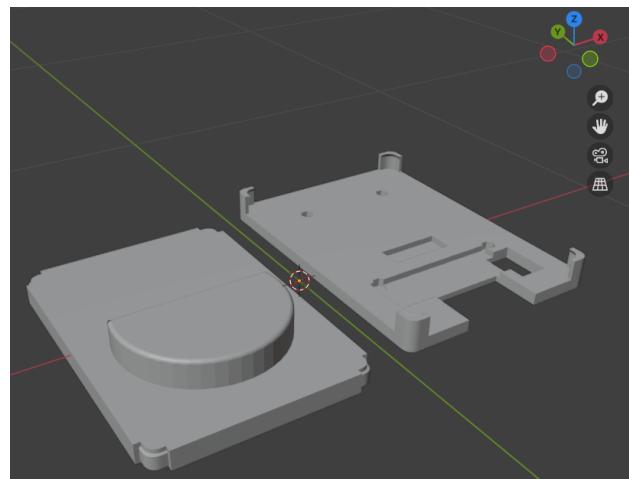


Figura 4.27: Modelado 3D del encapsulado para el módulo RFID.

# **Capítulo 5**

## **Pruebas y resultados**

En este capítulo se presentan las pruebas realizadas y los resultados obtenidos para cada uno de los módulos descritos en el Capítulo 4, evaluando su desempeño tanto de forma individual como integrada dentro del sistema. Estas pruebas permitieron validar la efectividad de cada componente y del sistema completo en condiciones reales. Posteriormente, se aborda el diseño y construcción del prototipo, destacando las características del hardware y software, la disposición estructural de los componentes y el encapsulado 3D, así como su montaje final en una motocicleta.

### **5.1. Pruebas en entornos controlado**

Se aplicaron pruebas en entornos controlado con la finalidad de evaluar el funcionamiento de los distintos módulos del sistema en condiciones ideales, libres de interferencias externas o factores no previstos. Este entorno permitió realizar un análisis más preciso de la autenticación de los usuarios, la captura y transmisión de datos, así como la integración completa del sistema, garantizando que todas las funciones operen correctamente antes de ser sometidas a condiciones reales. Para la ejecución de estas pruebas, se utilizaron los siguientes materiales:

- Notebook Dell Inspiron 15 3535 equipada con un procesador AMD Ryzen 5 7530U, gráficos integrados Radeon, 8 GB de memoria RAM y cuenta con sistema operativo Windows 11 Pro.
- Entorno de desarrollo Arduino IDE version 2.3.3.
- Módulo RFID RC522.
- Módulo GNSS GP-02-KIT.

- Placa de desarrollo Heltec WiFi LoRa 32 V3.
- Placa de desarrollo ESP-WROOM-32.
- Consola Helium (Legacy).
- Thingsboard.

### **5.1.1. Pruebas del Módulo de autentificación y control**

Se llevaron a cabo pruebas con el módulo RFID RC522, utilizando la placa desarrollo WiFi LoRa 32 V3 y el entorno de desarrollo Arduino IDE. Estas pruebas se realizaron con el objetivo de verificar el funcionamiento del módulo RFID y evaluar su capacidad para leer y comparar los identificadores únicos (UID) de tarjetas y llaveros RFID.

La prueba de la primera versión consta en tres partes, descritas a continuación:

#### **Lectura del UID de la tarjeta/llavero**

En esta fase, se acercó una tarjeta o llavero RFID al módulo RC522, permitiendo que el sistema leyera y mostrara el UID en el monitor serie en formato hexadecimal. Esta etapa inicial confirmó la capacidad del módulo para detectar y visualizar los datos de identificación de los dispositivos RFID.

#### **Comparación de UID y control de acceso**

Posteriormente, se implementó una función para comparar el UID leído con los valores previamente almacenados en el sistema. Esta verificación fue clave para determinar si el módulo podía diferenciar entre tarjetas o llaveros autorizados y no autorizados. El sistema respondió adecuadamente, permitiendo el acceso solo cuando el UID coincidía con los registros autorizados.

#### **Combinación de lectura y comparación constante**

Finalmente, se desarrolló una prueba de combinación, en la cual el sistema leía y comparaba de forma constante el UID de la tarjeta o llavero, verificando su presencia continua. Este proceso fue diseñado para probar una futura funcionalidad del sistema, en la que se definirá un tiempo específico para asegurar la autenticación continua del usuario. La Figura 5.1, muestra cómo el sistema monitorea constantemente la presencia del UID, lo cual es fundamental para mantener la autenticación activa mientras el usuario permanezca en proximidad.

The screenshot shows a terminal window titled "Serial Monitor". It displays a series of messages from a WiFi LoRa module. The messages indicate successful detections of an RFID tag within a range, followed by a final message about the tag going out of range and disconnecting the motorcycle.

```

Output  Serial Monitor X
Message (Enter to send message to 'WiFi LoRa 32(V3)' on 'COM8')
17:51:56.877 -> UID: B9 EF E4 6E      Bienvenido Usuario 1
17:51:59.931 -> Tag sigue dentro del rango.
17:52:00.615 -> Tag sigue dentro del rango.
17:52:00.853 -> Tag sigue dentro del rango.
17:52:01.071 -> Tag sigue dentro del rango.
17:52:01.872 -> Tag sigue dentro del rango.
17:52:02.119 -> Tag sigue dentro del rango.
17:52:02.554 -> Tag sigue dentro del rango.
17:52:12.663 -> Tag fuera de rango, desconectando la motocicleta...

```

Figura 5.1: Prueba del módulo RFID.

En la Tabla 5.1, se puede observar el número de pruebas realizadas para cada versión del código, así como los resultados obtenidos en términos de detección y comparación del UID de las tarjetas y llaveros RFID. Esta información proporciona una visión general del desempeño y funcionamiento del sistema en cada fase del desarrollo.

Tabla 5.1: Resultados de las pruebas de detección y comparación del UID

Versión del Código	Total de Pruebas	Pruebas Exitosas	Pruebas Fallidas
Primera versión	10	7	3
Segunda versión	30	25	5

Las fallas observadas durante las pruebas fueron esporádicas y se debieron a errores humanos, problemas de conexión y ajustes manuales. En algunos casos, los cables se desconectaron accidentalmente durante el proceso, interrumpiendo temporalmente la comunicación del sistema. Otras fallas, aunque puntuales, estuvieron relacionadas con errores en el código que afectaron la detección y comparación del UID. Además, se identificaron dificultades al mantener el tag en posición fija con la mano, lo que ocasionó lecturas inestables en ciertos momentos.

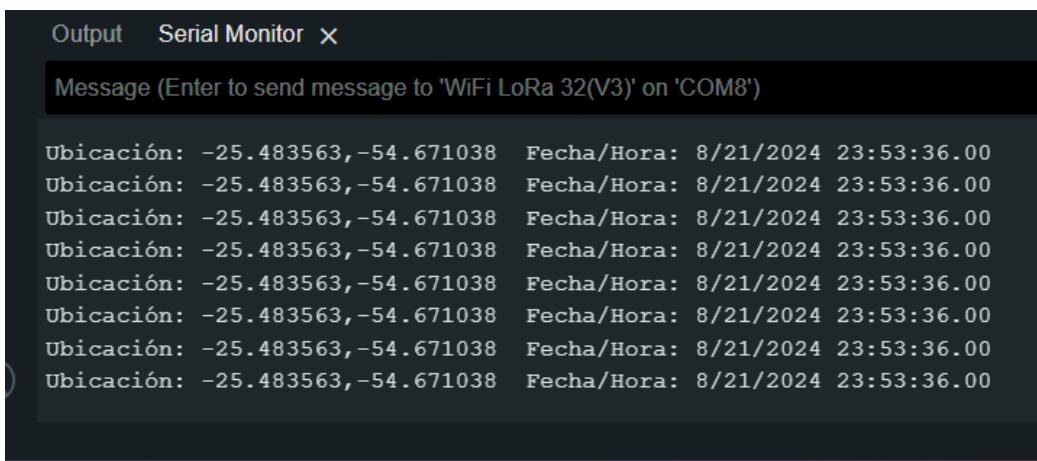
### 5.1.2. Pruebas del Módulo de captura y transmisión de datos

Se llevaron a cabo pruebas específicas por separado para evaluar la capacidad del sistema en la captura de las coordenadas del módulo GNSS y su transmisión a través de la red LoRaWAN utilizando la consola Helium. En esta fase, se validó la correcta conexión a un hotspot, así como la correcta recepción de las coordenadas y la recepción de paquetes de datos en condiciones de laboratorio. Las pruebas se dividieron en tres partes, descritas a continuación:

- Prueba del módulo GNSS.
- Prueba del módulo LoRa del Heltec.
- Prueba de captura y transmisión de datos de localización.

#### Prueba del módulo GNSS

En la primera implementación del código, se utilizó únicamente el módulo GNSS conectado a la placa de desarrollo Heltec WiFi LoRa 32 V3. El objetivo fue obtener datos de ubicación y verificar que el sistema pudiera capturar correctamente las coordenadas en tiempo real. Durante esta prueba, se evaluó la precisión de las coordenadas obtenidas, asegurando que fueran consistentes y confiables. En la figura 5.2, se puede observar como se obtuvo las coordenadas por monitor serial de la ubicación estática.



The screenshot shows a terminal window titled "Serial Monitor". The title bar includes "Output" and "Serial Monitor" with a close button "X". Below the title bar is a dark grey header bar with the text "Message (Enter to send message to 'WiFi LoRa 32(V3)' on 'COM8')". The main area of the window displays repeated lines of text in white on a black background. Each line consists of two columns separated by a tab character. The first column contains the text "Ubicación: -25.483563,-54.671038" and the second column contains the text "Fecha/Hora: 8/21/2024 23:53:36.00". This pattern repeats eight times.

Ubicación:	Fecha/Hora:
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00
-25.483563,-54.671038	8/21/2024 23:53:36.00

Figura 5.2: Pruebas del módulo GNSS.

Luego, se tomaron las coordenadas obtenidas del módulo GNSS y se verificó su precisión utilizando Google Maps [66]. Este paso permitió corroborar la ubicación detectada por el sistema y evaluar la exactitud de la geolocalización en comparación con un dispositivo de referencia. Para ello, se utilizó un teléfono convencional de la marca Xiaomi con modelo Mi 11 Lite, mediante el cual se consultó la ubicación actual del lugar. Este proceso brindó un punto de comparación en cuanto a la precisión del módulo GNSS frente al teléfono, permitiendo analizar posibles diferencias en la exactitud de las coordenadas reportadas por ambos dispositivos. En las figuras 5.3 y 5.4, se muestran, respectivamente, las ubicaciones obtenidas con el módulo GNSS y con el teléfono, facilitando así la visualización comparativa de la precisión entre ambas lecturas.

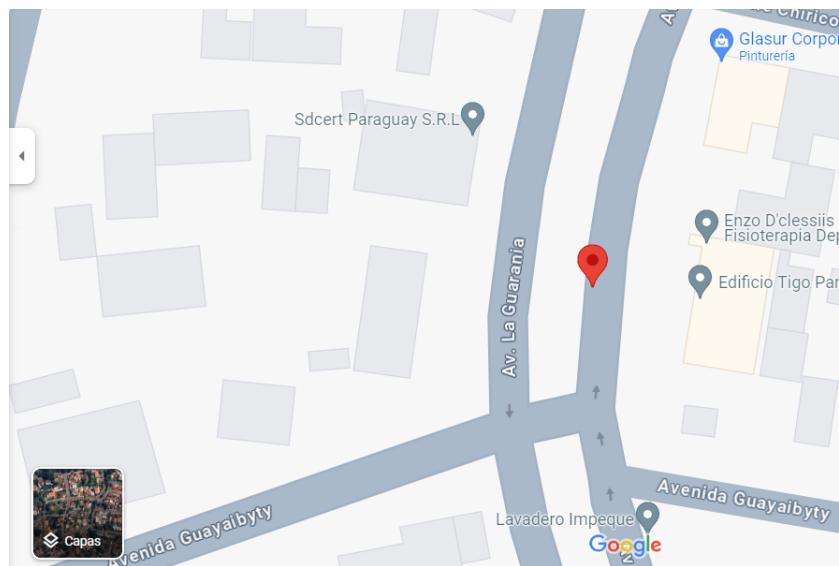


Figura 5.3: Comprobación de las coordenadas del módulo GNSS.

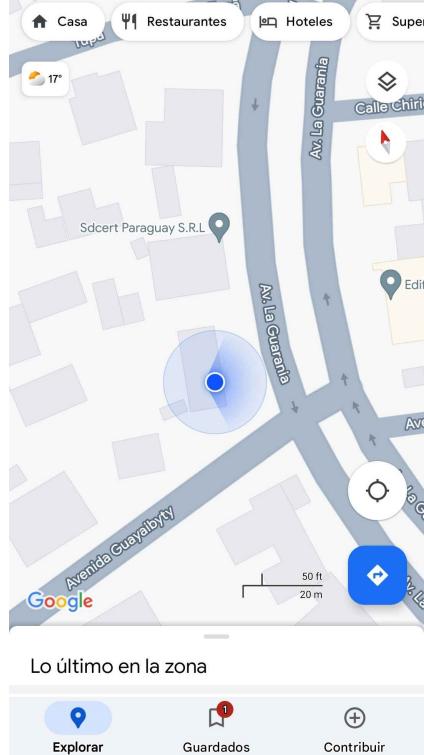


Figura 5.4: Comprobación con teléfono convencional.

A partir de la comparación de las ubicaciones obtenidas, se pudo observar una diferencia de aproximadamente 33 metros entre las coordenadas reportadas por el módulo GNSS y el teléfono Xiaomi Mi 11 Lite. Esta variación es una medida importante que permite evaluar la precisión del módulo GNSS en relación con un dispositivo de uso cotidiano, como el teléfono, que emplea servicios de localización basados en múltiples satélites y redes de datos.

Es importante destacar que estas pruebas iniciales se realizaron dentro de un recinto cerrado, lo cual pudo haber limitado la precisión del módulo GNSS debido a interferencias o bloqueos de señal. Posteriormente, se realizaron pruebas en un espacio abierto, donde la precisión del módulo mejoró notablemente. La exactitud rondó entre los 4 y 6 metros en promedio, con un error máximo de 8 metros, como se observa en la Figura 5.5, que muestra la distancia medida en Google Earth [67], para dos puntos obtenidos.



Figura 5.5: Medición de distancias con Google Earth - Coordenadas específicas.

Para validar estas distancias, se utilizó la herramienta Omni Calculator [68], con la cual se calculó la separación exacta entre las coordenadas obtenidas. El resultado de esta validación se detalla en la Figura 5.6, que confirma una diferencia de 4.9 metros entre las posiciones reportadas.

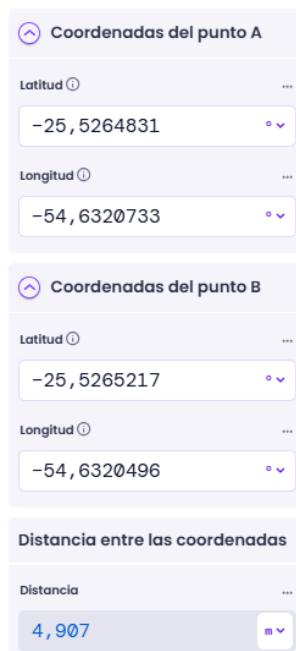


Figura 5.6: Distancia calculada con Omni Calculator.

En la Figura 5.7 se ilustran varias ubicaciones adquiridas durante las pruebas en un espacio abierto, demostrando la consistencia de las coordenadas registradas por el módulo GNSS. Estas mediciones fueron esenciales para comprobar la precisión del módulo .



Figura 5.7: Pruebas de ubicación en Google Earth con múltiples puntos adquiridos.

### Prueba del módulo LoRa

En la figura 5.8, se puede observar como se establecio la conexión exitosamente, se pudo observar en la consola como se realizaban las peticiones y los envíos uplink. La consola mostró en tiempo real los datos transmitidos desde el dispositivo, confirmando que los mensajes uplink eran recibidos y procesados correctamente por el servidor. En la segunda implementación, se centró exclusivamente en la funcionalidad de LoRaWAN. Se verificó si el sistema podía enviar correctamente solicitudes de conexión a la red y transmitir mensajes. Esta prueba fue crucial para asegurarse de que el módulo LoRa estuviera configurado correctamente y pudiera establecer comunicación con la red Helium [61], garantizando así que el sistema estuviera listo para la transmisión de datos.

Event	Type	No. of Hotspots	Time
+	Uplink ↗ 0	1	Aug 6, 2024 11:05:43.716 PM
+	Join Accept	1	Aug 6, 2024 11:05:37.635 PM
+	Join Request	1	Aug 6, 2024 11:05:35.635 PM

Figura 5.8: Visualización en la consola helium, mensaje de conexión y subida.

La consola también permitió visualizar a qué hotspot se conectó el dispositivo y con cuál realizó el envío de mensajes. Como se muestra en la figura 5.9.

Hotspots (1)					
Hotspot Name	RSSI	SNR	Frequency	Spreading	Time
feisty-bronze-crane	-116	-6.00	917.00	SF11BW125	Aug 6, 2024 11:05:43.716 PM

Figura 5.9: Visualización en la consola helium.

Con esta información, se utilizó el mapa de Helium [69], para identificar la ubicación exacta del hotspot, lo que permitió estimar la distancia desde la cual el dispositivo podía conectarse. Esto facilitó la evaluación de la cobertura y el alcance de la red en el área de prueba, como se puede observa en las siguientes figuras 5.10 y 5.11.

<input type="checkbox"/>	Hotspot Name	Alias	Signal	Packets	# of Devices	Status
<input type="checkbox"/>	Unknown Hotspot ⓘ			14	1	
<input type="checkbox"/>	Faint Green Hippo			2	1	
<input type="checkbox"/>	Feisty Bronze Crane			1	1	

Figura 5.10: Visualización de cobertura en la consola helium.

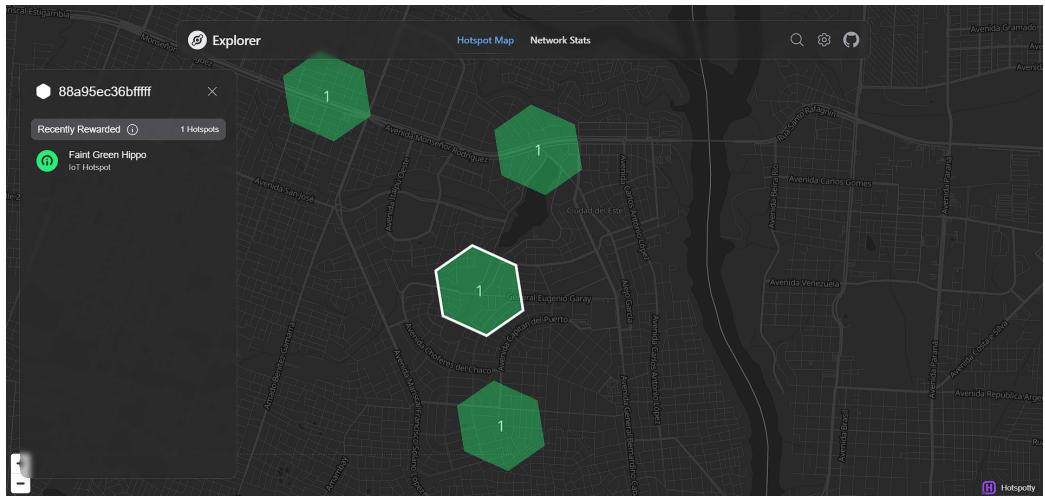


Figura 5.11: Visualización en la página helium maps.

Esta información adicional facilitó la verificación del funcionamiento de la red y permitió asegurar que la comunicación entre el dispositivo y el hotspot se estaba realizando de manera efectiva.

### Prueba de captura y transmisión de datos de localización

En la tercera implementación, se combinó la obtención de datos de localización del módulo GNSS con el envío de estos datos a través de LoRaWAN. Durante esta prueba, se verificó la correcta transmisión de los payloads que contenían las coordenadas GNSS, asegurando que no hubiera pérdida de datos en el proceso. Además, se realizó una decodificación manual del payload para validar su contenido. Se analizó la recepción de los datos en el servidor, confirmando que las coordenadas se enviaran y registraran sin errores. En la Figura 5.12, se muestra la estructura del archivo JSON que se descargó de la consola Helium, donde se registran todas las acciones realizadas.

```

"hold_time": 0,
"hotspot": {
    "channel": 3,
    "frequency": 917.4,
    "id": "11cde3uWPkobSSjXNPsNzpkZbpLBRdP64eFkWb8yrhpeJh74ZUh",
    "lat": -25.5341156144535,
    "long": -54.63307668405634,
    "name": "beautiful-cinnamon-cobra",
    "rssi": -130,
    "snr": -12.199999809265137,
    "spreading": "SF10BW125"
},
"mac": [],
"payload": "gDTMwYaIWsI=",
"payload_size": 8,
"port": 2,
"raw_packet": "QC4AAEgAAAACj31oyNl8/IbPy0H6"
},
"description": "Unconfirmed data up received",

```

Figura 5.12: Estructura del Mensaje enviado, donde payload serían las coordenadas en Base64.

Para validar la precisión de las coordenadas GNSS transmitidas, se siguieron los siguientes pasos para decodificar el mensaje recibido:

Por ejemplo, el mensaje gDTMwYaIWsI= se decodifica a: Latitud: -25.5256, Longitud: -54.6333. Para la decodificación Base 64, se utilizó una herramienta en línea, como Base64 Decode [70], para decodificar el mensaje. Este proceso convierte el mensaje codificado en una serie de bytes en formato hexadecimal, lo que facilita su interpretación. Luego se dividieron los primeros 4 bytes del mensaje para obtener la latitud y los siguientes 4 bytes para la longitud. Para convertir estos bytes en números flotantes (float), se utilizó una calculadora en línea, como Hex Convert [71]. Esta herramienta permite convertir el formato hexadecimal en un valor numérico que puede ser interpretado como coordenadas.

En las Figuras 5.13 y 5.14, se muestra el proceso de conversión utilizando la herramienta mencionada, ilustrando cómo se obtuvieron las coordenadas a partir del mensaje recibido.

Coordenadas en Base64 gDTMwYaIWsI=

Text ▾

Bytes ▾ Latitud y Longitud en 4bytes

FORMAT Hexadecimal

GROUP BY 4 Bytes

8034ccc1 86885ac2

Figura 5.13: Interpretación de Base64 a Hexadecimal.

Float - LittleEndian (DCBA)		
#	Raw	Float
0	C1 CC 34 80	-25.5256348

Float - LittleEndian (DCBA)		
#	Raw	Float
0	C2 5A 88 86	-54.6333237

Latitud y Longitud convertidas

Figura 5.14: Interpretación de las coordenadas de latitud y longitud transmitidas.

### 5.1.3. Pruebas del Módulo de Interfaz Gráfica y Monitoreo

El módulo de Interfaz Gráfica y Monitoreo se configuró para recibir y procesar datos en tiempo real de la ubicación y estado de un dispositivo prototipo. Durante las pruebas, se verificaron las siguientes funcionalidades clave: transmisión de datos, visualización en mapas interactivos, almacenamiento de datos en series temporales, y decodificación de mensajes en formato *Base64*.

#### Transmisión y visualización de datos

El dispositivo prototipo fue configurado en la consola de *ThingsBoard* para enviar datos de telemetría, tales como latitud, longitud, y eventos de usuario. Los datos se transmitieron a través del protocolo *MQTT* y fueron correctamente registrados en el tablero de control del sistema, donde se visualizan en tiempo real en un mapa *OpenStreet* (ver Figura 5.15).

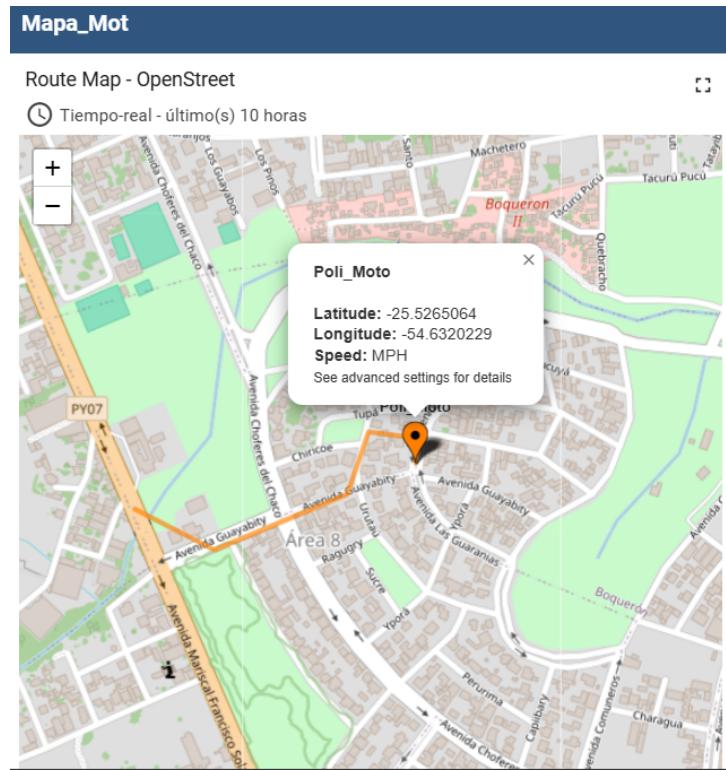


Figura 5.15: Visualización de la ubicación en el mapa *OpenStreet* con trayectoria.

### Almacenamiento en tablas de series temporales

La telemetría se almacenó en una tabla de series de tiempo que registró cada evento, permitiendo un análisis detallado de la trayectoria y actividad del dispositivo (ver Figura 5.16). Esto proporcionó un historial de datos que facilita la trazabilidad y auditoría de la ubicación del dispositivo en distintos momentos del día.

Timeseries table			
<span>⌚ Tiempo-real - último(s) día</span>			
Timestamp ↓	latitude	longitude	payload
2024-11-15 17:41:11	-25.5138053894 deg	-54.6414070129 deg	461CCCC1CD905AC2
2024-11-15 17:40:49	-25.5161609650 deg	-54.6415176392 deg	1921CCC1EA905AC2
2024-11-15 17:40:27	-25.5162410736 deg	-54.6414985657 deg	4321CCC1E5905AC2
2024-11-15 17:40:05	-25.5165843964 deg	-54.6414947510 deg	F721CCC1E4905AC2
2024-11-15 17:39:43	-25.5175304413 deg	-54.6413383484 deg	E723CCC1BB905AC2
2024-11-15 17:39:22	-25.5176429749 deg	-54.6412849426 deg	2224CCC1AD905AC2
2024-11-15 17:39:00	-25.5181102753 deg	-54.6411170959 deg	1725CCC181905AC2
2024-11-15 17:38:40	-25.5189476013 deg	-54.6407203674 deg	CE26CCC119905AC2
2024-11-15 17:37:58	-25.5224742889 deg	-54.6390914917 deg	072ECCC16E8E5AC2
2024-11-15 17:37:36	-25.5225334167 deg	-54.6390609741 deg	262ECCC1668E5AC2
2024-11-15 17:37:14	-25.5225753784 deg	-54.6390495300 deg	3C2ECCC1638E5AC2

Powered by [Thingsboard v.3.8.1](#)

Figura 5.16: Series temporales de telemetría.

## Pruebas de decodificación y transformación de datos

Para evaluar la correcta interpretación de mensajes enviados al sistema, se llevaron a cabo una serie de pruebas utilizando dos scripts desarrollados en *JavaScript*. Estas pruebas se diseñaron para asegurar la decodificación precisa de los datos transmitidos en formato *Base64* y su posterior interpretación en valores útiles para el sistema.

El proceso se dividió en dos etapas principales de prueba. En la primera etapa, se probó un script encargado de transformar los datos enviados en formato *Base64* a valores hexadecimales (ver Figura 5.17). Este script se verificó con mensajes simulados de distintos tamaños y contenido. Se realizaron 5 pruebas, cada una con un mensaje diferente, asegurando que los valores obtenidos fueran consistentes con los datos esperados.

En la segunda etapa, se utilizó un script adicional que interpretaba los valores hexadecimales obtenidos en la etapa previa. Este script clasificaba los datos según el tamaño del mensaje decodificado:

- Mensajes de 2 *bytes* representaban eventos, como alertas de desconexión o intentos de acceso no autorizado.

- Mensajes de 4 *bytes* representaban eventos, como de acceso de usuarios autorizados.
- Mensajes de 8 *bytes* correspondían a telemetrías de ubicación, como coordenadas de latitud y longitud.

Durante esta etapa, se realizaron 10 pruebas para verificar la capacidad del sistema de manejar una variedad de tamaños y contenidos diferentes en los mensajes. Los resultados de todas las pruebas fueron exitosos, demostrando es capaz de interpretar y clasificar correctamente los datos enviados. Un ejemplo de este proceso puede observarse en la Figura 5.18, donde se muestra la interpretación de un mensaje hexadecimal como denegación de acceso no autorizado.

The screenshot shows a user interface for message processing. On the left, under 'Mensaje', there is a code editor containing the following JSON-like code:

```

1 {
2   "payload": "/wE"
3 }

```

A red arrow points from the text "Mensaje de entrada en Base64" to the line "payload: "/wE"".

In the center, a 'Transformer' component is shown with the following JavaScript code:

```

function Transform(msg, metadata, msgType) {
  // Función para convertir Base64 a Hexadecimal
  function base64ToHex(base64) {
    var chars =
      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    var hex = '';
    // Tabla de decodificación Base64
    var binaryString = [];
    var buffer = 0;
    var bitsInBuffer = 0;
    for (var i = 0; i < base64.length; i++) {
      var index = chars.indexOf(base64[i]);
      if (index === -1) {
        throw new Error(`Character ${base64[i]} is not a valid Base64 character`);
      }
      if (bitsInBuffer === 0) {
        buffer = index;
      } else {
        buffer = buffer * 64 + index;
        bitsInBuffer++;
        if (bitsInBuffer === 8) {
          binaryString.push(buffer.toString(2));
          buffer = 0;
          bitsInBuffer = 0;
        }
      }
    }
    if (bitsInBuffer > 0) {
      binaryString.push(buffer.toString(2));
    }
    return binaryString.join('');
  }
  var hexMessage = base64ToHex(msg.payload);
  var msgType = 'POST_TELEMETRY_REQUEST';
  return {
    msg: {
      payload: hexMessage,
      metadata: {}
    },
    msgType: msgType
  };
}

```

On the right, under 'Salida', the transformed message is displayed:

```

1 {
2   "msg": {
3     "payload": "FF01"
4   },
5   "metadata": {},
6   "msgType": "POST_TELEMETRY_REQUEST"
7 }

```

A red circle highlights the entire 'msg' object, and a red arrow points from the text "Mensaje de salida en Hexadecimal" to the line "payload: FF01".

Figura 5.17: Prueba de decodificación de mensajes en formato *Base64* a valores hexadecimales.

```

Mensaje
1 ~ {
2   "payload": "FF0001"
3 }

function Transform(msg, metadata, msgType) {
  1 // Función para convertir hexadecimal a float usando little-endian
  2 function hexToFloat(hex) {
  3   if (hex.length !== 8) return null; // Validación para 8
      caracteres
  4
  5   var buffer = new ArrayBuffer(4); // Crear un buffer de 4 bytes
  6   var view = new DataView(buffer);
  7
  8   for (var i = 0; i < 4; i++) {
  9     view.setUint8(i, parseInt(hex.substr(i * 2, 2), 16));
 10   }
 11
 12   return view.getFloat32(0, true); // true para little-endian
 13 }
}

Test

```

Mensaje de entrada en Hexadecimal

Mensaje de salida interpretado

Figura 5.18: Prueba de interpretación de valores hexadecimales según su tamaño.

Para garantizar la precisión de las pruebas, se utilizó *Mosquitto* como herramienta para enviar mensajes simulados al sistema (ver Figura 5.19). Esta herramienta permitió verificar el flujo completo de transmisión.

Todos los resultados confirmaron el correcto funcionamiento del sistema, logrando interpretar los datos en tiempo real de manera eficiente y confiable.

```

mario@DESKTOP-J58FDBR:/mnt/c/Users/Hp$ mosquitto_pub -d -q 1 -h 172.21.240.1 -p 1883 -t v1/devices/me/telemetry -u "fxau...
gwbqfy4sn3kup4i" -m "{"
  "latitude": -25.5153,
  "longitude": -54.61558
}"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (48 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
mario@DESKTOP-J58FDBR:/mnt/c/Users/Hp$ mosquitto_pub -d -q 1 -h 172.21.240.1 -p 1883 -t v1/devices/me/telemetry -u "fxau...
gwbqfy4sn3kup4i" -m "{"
  "latitude": -25.5155,
  "longitude": -54.61574
}"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (48 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

```

Publicación exitosa

Figura 5.19: Uso de *Mosquitto* para simular mensajes durante las pruebas.

## Resultados de la integración entre ThingsBoard y Helium

En la Figura 5.20 se observa el resultado del proceso de integración entre ThingsBoard y la red Helium mediante el protocolo *MQTT*. Este proceso consistió en la recepción y procesamiento de cada mensaje enviado desde el dispositivo conectado.

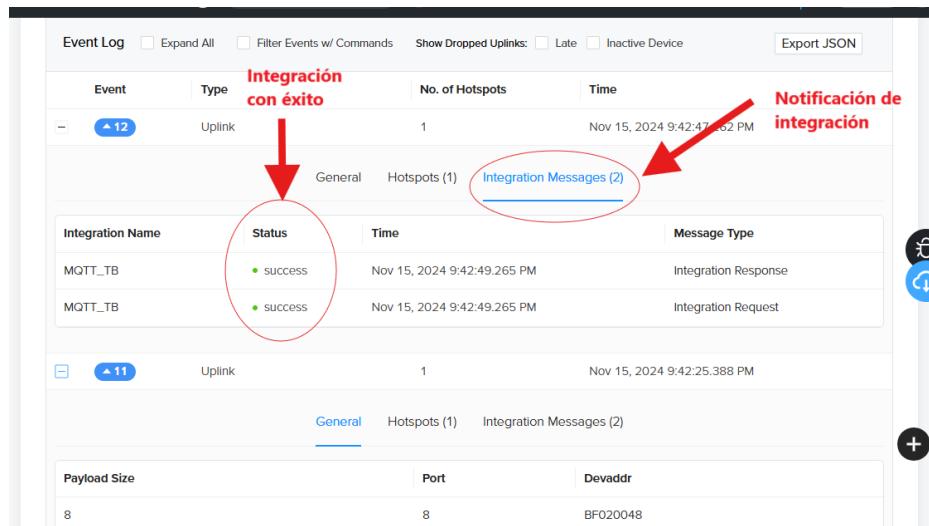


Figura 5.20: Integración Helium-ThingsBoards.

De los más de 120 mensajes transmitidos, un total de 8 mensajes fallaron al momento de integrarse, lo que representa un porcentaje de efectividad del 93.33 % en la integración.

A pesar de estas fallas, los mensajes que sí lograron ser transmitidos exitosamente fueron interpretados y decodificados con un 100 % de efectividad. Esto incluye tanto la transformación de los datos de formato *Base64* a hexadecimales como su interpretación en valores útiles para el sistema, validando el correcto funcionamiento del módulo.

### 5.1.4. Integración del sistema completo

En esta fase, se realizaron pruebas específicas para validar la integración y el correcto funcionamiento de todos los módulos necesarios para el sistema: RFID, GNSS y LoRaWAN, utilizando las placas de desarrollo ESP-WROOM-32 y Heltec WiFi LoRa 32 V3. El objetivo principal fue asegurar la operación conjunta de estos módulos en el sistema, comprobando la funcionalidad de identificación mediante RFID, la transmisión de datos de ubicación

y el envío de información por la red LoRaWAN. Las pruebas se desarrollaron en tres etapas, detalladas a continuación:

### **Primera Prueba: Comunicación entre las placas de desarrollo (ESP-WROOM-32 y Heltec WiFi LoRa 32 V3)**

Esta prueba inicial consistió en establecer la comunicación entre ambas placas mediante el protocolo UART, con el módulo RFID conectado al ESP-WROOM-32. El sistema transmitió datos de identificación desde el módulo RFID al Heltec WiFi LoRa 32 V3, validando la presencia del UID autorizado. Además, se comprobó que esta transmisión no interrumpiera la conexión con la red LoRaWAN, manteniendo la integridad de los datos y la estabilidad de la comunicación entre los dispositivos.

### **Segunda Prueba: Integración del Módulo GNSS**

En esta etapa, se implementó la versión 2 del código, que integraba el módulo GNSS junto con el módulo RFID. La prueba se orientó a evaluar la transmisión de los datos recolectados, donde:

El UID autorizado fue enviado en formato hexadecimal. Otros datos de ejemplo fueron incluidos en la transmisión para verificar la estructura de los mensajes. Las coordenadas GPS fueron convertidas a un formato adecuado, preparadas como un arreglo de bytes para ser enviadas por LoRaWAN.

### **Tercera Prueba: Evaluación de las Funciones de Envío de Datos Completos**

En la última fase, se probó la versión 3 del código, que incluyó el flujo completo de detección RFID, obtención de coordenadas GNSS y envío de datos a través de LoRaWAN. En esta prueba se evaluaron las siguientes condiciones:

- Detección RFID: El sistema identificó la presencia de un tag RFID autorizado, enviando el UID a través de LoRaWAN en el caso de una identificación válida.
- Envío de Datos por LoRaWAN: Dependiendo del tipo de mensaje seleccionado, el sistema envió el UID del RFID, otros datos de prueba o, en caso de estar disponibles, las coordenadas.
- Ciclo de LoRaWAN: El dispositivo siguió un ciclo de transmisión periódica, entrando en modo de suspensión tras cada envío, optimizando así el consumo energético.

- Validación de Coordenadas: Se verificó que el dispositivo obtuviera coordenadas válidas y las transmitiera en el siguiente ciclo de envío, asegurando que no hubiera pérdida de datos en la transmisión.

Los resultados de las pruebas realizadas para las tres versiones del código se resumen en la Tabla 5.2.

Tabla 5.2: Resultados de las Pruebas por Versión del Código

Versión del Código	Total de Pruebas	Pruebas Exitosas	Pruebas Fallidas
Primera Versión	10	8	2
Segunda Versión	20	15	5
Tercera Versión	25	24	1

A lo largo de las pruebas realizadas, la evolución del sistema mostró mejoras significativas. En la primera versión, se obtuvieron un 80 % de éxito y un 20 % de fallos en 10 pruebas. La segunda versión alcanzó un 75 % de éxito y un 25 % de fallos en 20 pruebas. Finalmente, la tercera versión logró una notable eficiencia con un 96 % de éxito y solo un 4 % de fallos en 25 pruebas.

En la Figura 5.21, se presenta la correcta recepción de los datos de UID, transmitidos como payloads de 4 bytes. En la Figura 5.22, se muestra cómo se recibieron los datos de las coordenadas en los intervalos de tiempo definidos.

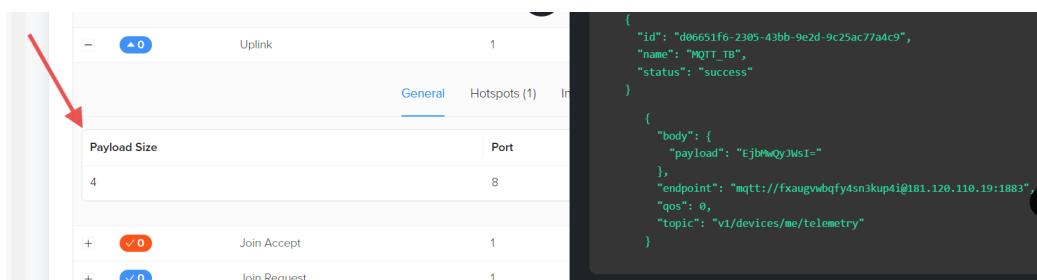


Figura 5.21: Visualización en la consola Helium de los paquetes de datos recibidos del UID.

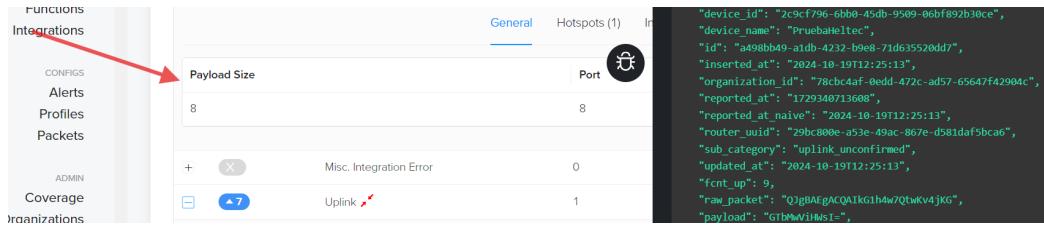


Figura 5.22: Visualización en la consola Helium de los paquetes de datos recibidos de las coordenadas.

## 5.2. Pruebas de Campo

En esta sección, se documentan las pruebas de campo realizadas con una motocicleta Kenton, modelo GL150, equipada con un sistema de encendido *CDI* (Ignición por Descarga de Condensador) y una batería de 12 voltios. Estas pruebas se llevaron a cabo utilizando la última versión del código ajustado, que integra la funcionalidad de corte de energía y arranque controlado mediante el sistema *RFID* y el módulo relé. Además, se evaluó el rendimiento del sistema en movimiento.

### 5.2.1. Evaluación Inicial

El propósito de las pruebas iniciales fue verificar la integración del sistema y su capacidad para realizar autenticaciones fluidas y sincronizarse adecuadamente con el módulo de corte de energía. Se comprobó que el sistema respondiera de manera precisa a la presencia de un *tag* autorizado, ejecutando el corte de energía al retirarse o ausentarse dicho *tag*. Además, se llevaron a cabo varias pruebas para medir la respuesta del sistema y garantizar su fiabilidad.

En la Figura 5.23, se presenta el diagrama esquemático de las conexiones realizadas para la integración del sistema desarrollado en la motocicleta. Este esquema representa la interacción entre los componentes eléctricos de la motocicleta y el prototipo diseñado.

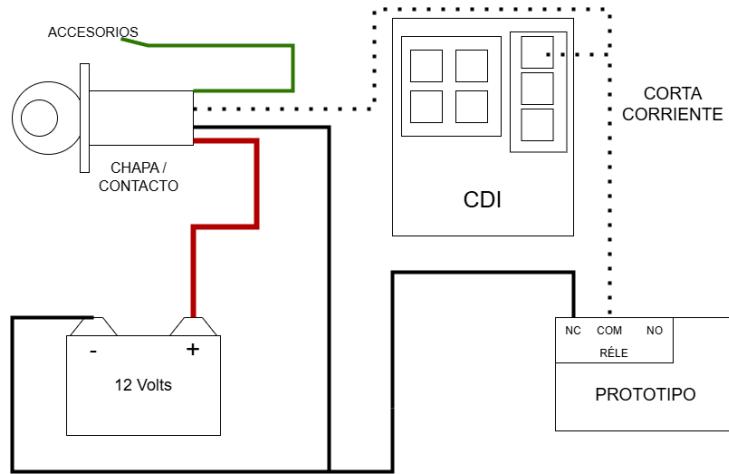


Figura 5.23: Diagrama esquemático de conexiones.

### 5.2.2. Pruebas en Movimiento

Posteriormente, se realizaron 9 pruebas de campo adicionales para evaluar el comportamiento del sistema en condiciones reales de movimiento. En estas pruebas, se evaluaron las siguientes fases:

- **Conexión a *LoRaWAN* en movimiento:** En tres pruebas específicas, se inició el sistema fuera del área de cobertura de la red *LoRaWAN*, y se comprobó que, al entrar en el rango de la red, el dispositivo lograba realizar el proceso de *join* exitosamente. Todas estas pruebas fueron satisfactorias.
- **Autenticación y arranque del motor:** En cada prueba, se aproximó el *tag RFID* para autenticar al usuario. Aunque en dos pruebas iniciales la lectura del *UID* falló, provocando el apagado inmediato del motor, las pruebas posteriores fueron exitosas, logrando una autenticación fluida y un arranque sin interrupciones.
- **Estabilidad del sistema durante el movimiento:** Durante todas las pruebas, el sistema permaneció operativo, sin desconexiones ni interrupciones, incluso al enfrentarse a cambios de velocidad.
- **Transmisión de datos en tiempo real:** El sistema transmitió correctamente las coordenadas *GPS* y los eventos críticos (autenticación, desconexión, etc.) a la plataforma *ThingsBoard*, donde se visualizaron en tiempo real.

En la Figura 5.24, se ilustra una de las trayectorias recogidas durante las pruebas de movimiento, mostrando cómo el sistema registró y transmitió la ubicación de la motocicleta en tiempo real.

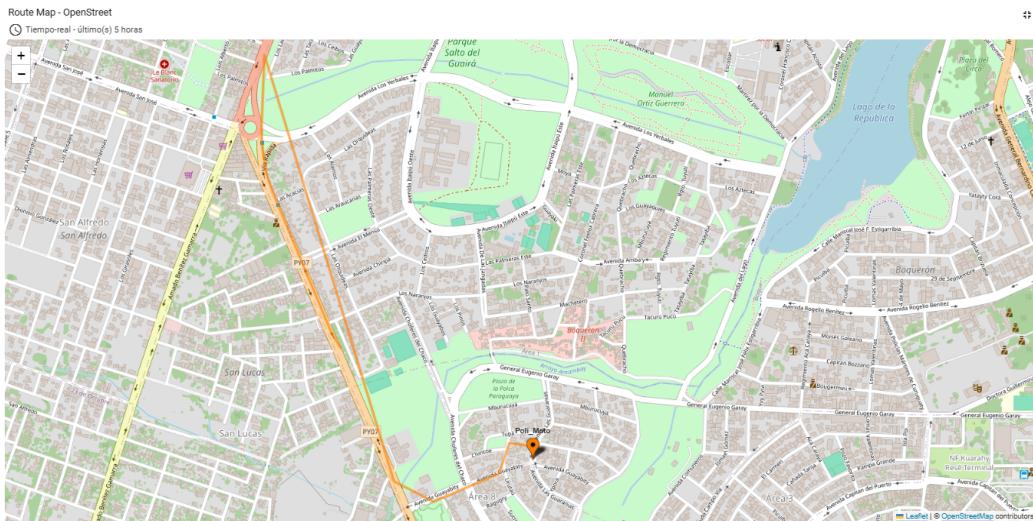


Figura 5.24: Trayectoria registrada durante las pruebas de campo.

### 5.2.3. Resultados Cuantitativos

En la Tabla 5.3, se resumen los resultados de las pruebas realizadas, incluyendo autenticaciones exitosas, conexiones a *LoRaWAN* y transmisión de datos durante el movimiento.

Tabla 5.3: Resultados de las pruebas de campo

Nº de Prueba	Autenticación exitosa	Conexión a LoRa-WAN	Transmisión de datos	Sistema estable	Corte al alejar el tag
1	No	Sí	Sí	Sí	Sí
2	No	Sí	Sí	Sí	Sí
3	Sí	Sí	Sí	Sí	Sí
4	Sí	Sí	Sí	Sí	Sí
5	Sí	Sí	Sí	Sí	Sí
6	Sí	Sí	Sí	Sí	Sí
7	Sí	Sí	Sí	Sí	Sí
8	Sí	Sí	Sí	Sí	Sí
9	Sí	Sí	Sí	Sí	Sí

### Porcentajes de Éxito y Fallo

- **Autenticación exitosa:** De las 9 pruebas realizadas, 7 lograron autenticar correctamente al usuario mediante el *tag RFID*, lo que representa un 77.78 % de éxito. Las dos fallas iniciales, que corresponden al 22.22 % de fallo, se debieron a errores humanos durante el proceso de manejo del sistema, como una posición incorrecta del tag RFID o fallos en su aproximación al lector. Estas situaciones fueron identificadas y corregidas en las pruebas posteriores, logrando resultados exitosos en el resto de las evaluaciones.
- **Conexión a *LoRaWAN*:** En el 100 % de las pruebas, el dispositivo logró conectarse exitosamente a la red *LoRaWAN*, incluso en condiciones de movimiento y al iniciar fuera del rango de cobertura. Esto demuestra una fiabilidad completa (100 % de éxito) en la capacidad de conexión del sistema.
- **Transmisión de datos:** En todas las pruebas, los datos del sistema, como coordenadas *GPS* y eventos críticos, fueron transmitidos correctamente a la plataforma *ThingsBoard*. Esto refleja un 100 % de éxito en la transmisión de datos.
- **Sistema estable:** Durante todas las pruebas, el sistema permaneció operativo, sin apagarse ni desconectarse de manera inesperada. Esto

indica una estabilidad del 100 %.

- **Corte de energía al alejar la tarjeta:** En el 100 % de los casos, el sistema ejecutó correctamente el corte de energía al detectar la ausencia del *tag* autorizado. Este resultado asegura una respuesta fiable del sistema en situaciones de seguridad.

#### 5.2.4. Interpretación de los Resultados

Los resultados generales demuestran un desempeño sólido del sistema en las fases probadas. A pesar de las fallas iniciales en el proceso de autenticación, el prototipo mostró un comportamiento consistente y confiable tras los ajustes realizados.

- **Éxito global:** Considerando todos los criterios evaluados, el sistema presentó un éxito promedio de 95.56 % (calculado como el promedio ponderado de los porcentajes de éxito en cada criterio).
- **Fallo global:** Las fallas observadas se limitaron al proceso de autenticación inicial y representaron un promedio de 4.44 % del total de las pruebas, indicando que los problemas fueron corregidos de manera efectiva.

Este análisis cuantitativo confirma que el prototipo cumple con los requerimientos definidos para garantizar la autenticación de usuarios, la conectividad a la red y la transmisión de datos en tiempo real. Además, su estabilidad y capacidad de respuesta en condiciones reales refuerzan su viabilidad como solución para la seguridad vehicular.

### 5.3. Prototipo del sistema desarrollado

En este apartado se describen las principales características del prototipo desarrollado, el cual integra *hardware* y diseño estructural. El prototipo se divide en dos partes principales: el *hardware* del sistema y el *software* del sistema.

#### 5.3.1. *Hardware* del prototipo

El *hardware* del prototipo está compuesto por dos módulos principales. El módulo de autenticación y control identifica un *UID* autorizado mediante un lector *RFID* y realiza una lectura constante para verificar si el *UID*

permanece dentro del rango establecido. En caso de que el *UID* autorizado salga del rango, el sistema corta automáticamente la energía de la motocicleta, impidiendo su funcionamiento. Adicionalmente, el sistema incorpora un módulo de captura y transmisión de datos, diseñado para transmitir información mediante tecnología *LoRaWAN*. Incluye el envío del *UID* autorizado y las coordenadas de ubicación de la motocicleta, además de enviar mensajes de desconexión del usuario.

### Modelado y Diseño de Encapsulado 3D

A continuación, en la Figura 5.25, se presenta el diseño de la disposición estructural de los componentes, donde se han colocado los cables de conexión por debajo del *shield*, optimizando así el orden y la accesibilidad de cada módulo en el sistema.

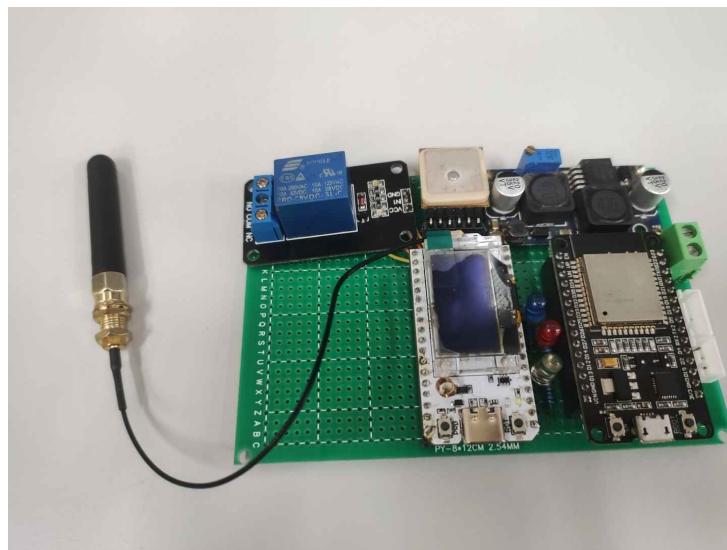


Figura 5.25: Disposición estructural.

En la siguiente Figura 5.26, se presenta el encapsulado en 3D, elaborado mediante impresión en una impresora 3D, el cual proporciona protección y organización a los componentes del sistema. En la Figura 5.27, se muestra un encapsulado adicional para el módulo *RFID*.



Figura 5.26: Encapsulado 3D.

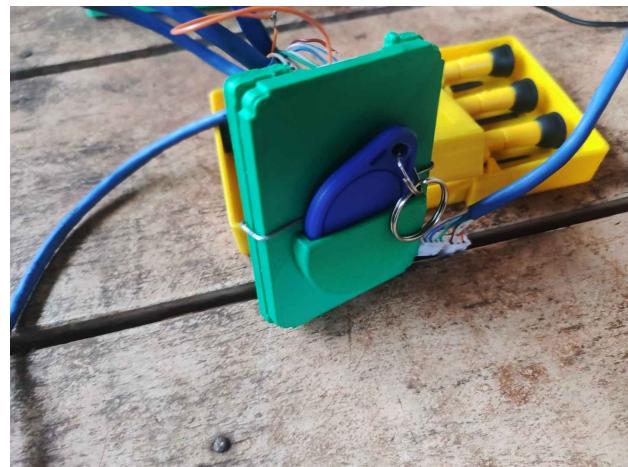


Figura 5.27: Encapsulado 3D para el módulo *RFID*.

Por último, en la Figura 5.28, se presenta el montaje final del sistema, con todos los componentes integrados y dispuestos en la motocicleta. La Figura 5.28 incluye flechas que destacan la ubicación precisa de los prototipos, facilitando su identificación y visualización en el contexto del montaje completo.



Figura 5.28: Prototipo Final.

### **Software del prototipo**

El *software* desarrollado para el prototipo habilita las funciones principales del sistema, asegurando la integración de los módulos de *hardware* con un control eficiente y coordinado. Este *software* gestiona la interacción entre los componentes, facilitando la autenticación de usuarios, el monitoreo continuo y la transmisión de datos mediante *LoRaWAN*.

Entre sus funcionalidades más relevantes, se encuentra la validación de usuarios mediante el lector *RFID*, supervisando de forma constante la presencia del *UID* autorizado. En caso de desconexión, el *software* ejecuta automáticamente acciones críticas, como el corte de energía de la motocicleta, reforzando la seguridad del sistema.

Adicionalmente, el *software* es responsable de la recopilación y transmisión de datos relevantes, como las coordenadas *GPS* y el *UID* autenticado, a través de *LoRaWAN*. Esto asegura que el usuario pueda recibir alertas en tiempo real, incluyendo notificaciones de eventos críticos como intentos de arranque no autorizados.

Una de las características más destacadas es la capacidad del *software* para integrar y mostrar estos datos en una plataforma gráfica basada en

*ThingsBoard*. En esta interfaz, el usuario puede visualizar la ubicación en tiempo real de la motocicleta sobre un mapa interactivo, además de consultar una tabla de series temporales que detalla las coordenadas transmitidas y otros parámetros relevantes.

En la Figura 5.29 se presenta un ejemplo de la visualización en la plataforma, destacando cómo el *software* permite un monitoreo continuo y detallado. La figura muestra la ubicación de la motocicleta y los datos transmitidos mediante *LoRaWAN*.

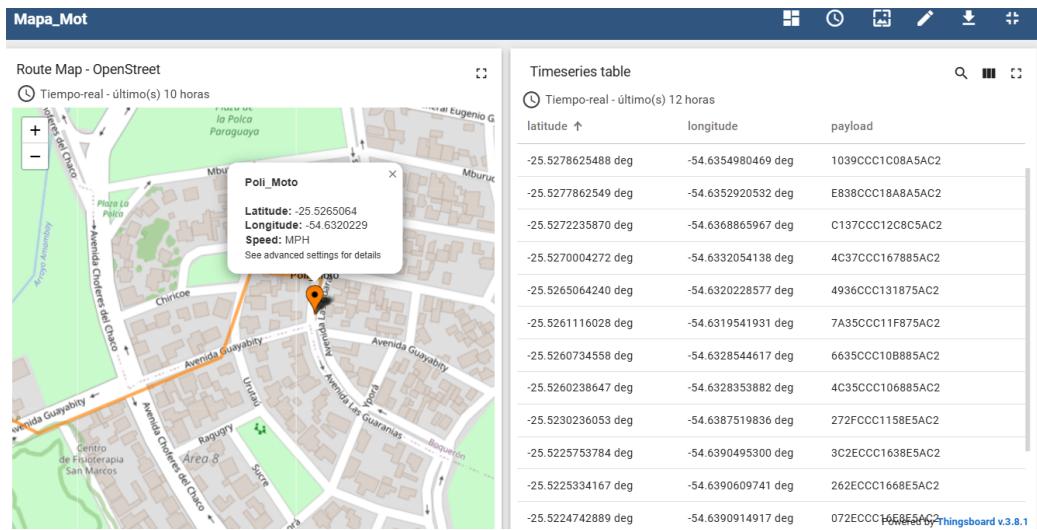


Figura 5.29: Visualización de datos en tiempo real.

Con estas funcionalidades, el *software* habilita una solución tecnológica que combina seguridad, monitoreo y gestión de datos en tiempo real, respondiendo a las necesidades específicas del prototipo desarrollado.

# Capítulo 6

## Conclusión

En este capítulo se interpretan y sintetizan los resultados expuestos previamente, además de evaluar la validez de la hipótesis, el logro de los objetivos y realizar recomendaciones para futuros estudios.

### 6.1. Discusión

Se ha desarrollado un sistema de seguridad para motocicletas basado en tecnologías de autenticación *RFID* y transmisión de datos mediante *LoRaWAN*. Este sistema, tras varias etapas de desarrollo e investigación en el ámbito *IoT*, ha sido capaz de autenticar usuarios y rastrear la ubicación de la motocicleta en tiempo real.

Durante el proceso de desarrollo, se lograron integrar exitosamente los módulos de autenticación, comunicación y control, permitiendo una funcionalidad de seguridad que restringe el uso de la motocicleta solo a usuarios autenticados y proporciona un monitoreo continuo en caso de situaciones de riesgo.

Además, se tuvieron en cuenta aspectos como la alimentación del sistema en entornos de prueba y la integración de sensores que permitieran tanto la autenticación mediante *RFID* como la recolección de datos de ubicación a través del módulo *GNSS*. Esto aseguró que el prototipo respondiera adecuadamente en condiciones reales de uso en motocicletas.

Las pruebas realizadas arrojaron resultados favorables, evidenciando la efectividad del sistema en el control de acceso y el rastreo de ubicación. Comparado con métodos de seguridad vehicular que solo emplean bloqueos mecánicos o electrónicos, este sistema proporciona una capa adicional de seguridad, permitiendo la autenticación continua del usuario y la desconexión

automática en caso de que el usuario se aleje del vehículo. Esta solución, probada en entornos urbanos simulados, ha mostrado un grado de efectividad del 95.56 % en las situaciones de prueba.

En comparación con otros estudios y tecnologías de seguridad, como los sistemas de rastreo *GPS* convencionales o las aplicaciones de bloqueo remoto, nuestro sistema destaca por su integración de una red de baja potencia (*LoRaWAN*) que permite un monitoreo constante sin afectar significativamente la autonomía de la batería del vehículo. De este modo, el sistema propuesto ofrece un enfoque novedoso y efectivo en términos de seguridad y rastreo.

## 6.2. Análisis de la hipótesis

A partir de los resultados obtenidos, la hipótesis planteada inicialmente, que sugiere que la integración de tecnologías *RFID* y *LoRaWAN* para la seguridad y rastreo de motocicletas puede ofrecer una alternativa eficaz a los sistemas actuales, ha sido validada. Los datos recopilados durante las pruebas en entornos controlados y en campo confirman que el sistema desarrollado proporciona una capa adicional de seguridad y monitoreo en tiempo real.

## 6.3. Principales logros alcanzados

En consecuencia al cumplimiento de los objetivos también presentados en el primer capítulo, han sido alcanzados los siguientes logros:

- Se definió una estructura para el sistema de seguridad, integrando tecnologías de comunicación y control.
- Se desarrolló un *firmware* eficiente para el microcontrolador que permite la interacción fluida entre los dispositivos de seguridad y comunicación.
- Se adaptó y aprovechó el uso de *ThingsBoard* como interfaz de usuario, permitiendo la visualización clara de alertas y datos en tiempo real.
- Se logró ensamblar un prototipo de *hardware* que integra sensores, actuadores y un sistema de alimentación adecuado, garantizando su funcionalidad en el entorno de prueba.
- Se logró diseñar y fabricar un encapsulado *3D* personalizado que asegura la protección eficaz de los componentes del sistema.

- Se llevaron a cabo pruebas de laboratorio y de campo que confirmaron el rendimiento esperado del sistema, validando su efectividad en escenarios reales y obteniendo datos clave para futuros desarrollos.

## 6.4. Sugerencias para futuras investigaciones

Con base en este trabajo, se sugieren las siguientes áreas de mejora y futuras investigaciones:

- Optimizar el sistema de energía mediante algoritmos de bajo consumo para prolongar la vida útil de la batería de respaldo en períodos de inactividad.
- Desarrollar una aplicación móvil que permita monitorear el estado del vehículo en tiempo real y recibir alertas inmediatas.
- Evaluar la cobertura de la red *Helium* en distintos entornos geográficos y analizar su rendimiento en áreas rurales y urbanas para validar la robustez del sistema.



# Referencias bibliográficas

- [1] Ikpehai, Augustine, Adebisi, Bamidele, Rabie, K. M., A. Kelvin, Ande, R. E., Hammoudeh, Mohammad, Gacanin, Haris, Mbanaso, y U. M., “Low-power wide area network technologies for internet-of-things: A comparative review,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2225–2240, 2019.
- [2] X. Jia, Feng, Quanyuan, Fan, Taihua, Lei, y Quanshui, “Rfid technology and its applications in internet of things (iot),” *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*, 04 2012.
- [3] “What is lorawan,” 2020, loRa Alliance. Disponible en: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan>. [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>
- [4] D. de Helium, “Descripción general de la red de helium.” [Online]. Available: <https://docs.helium.com/>
- [5] Dirección del registro de automotores. Estadística de matriculaciones correspondiente al periodo: octubre de 2000 a febrero de 2024. [Online]. Available: <https://www.pj.gov.py/contenido/155-direccion-del-registro-de-automotores/1331>
- [6] Denuncias registradas y aclaradas en las comisarías del país, según tipo de delito. aÑo 2019. Datos.gov.py - Anuario Estadístico 2019. [Online]. Available: <https://www.datos.gov.py/dataset/anuario-estad-stico-2019-estad-sticas-policiales-y-accidentes-de-tr-nsito/resource/c73bf2da>
- [7] A. N. de Tránsito y Seguridad Vial (ANTS). (Año) Informe anual 2021 - observatorio. [Online]. Available: [https://www.antsv.gov.py/application/files/8716/6004/9458/INFORME\\_ANUAL\\_2021\\_OBSERVATORIO.pdf](https://www.antsv.gov.py/application/files/8716/6004/9458/INFORME_ANUAL_2021_OBSERVATORIO.pdf)

- [8] N. Rana, P. Khatta, y A. D. Mishra, “Smart security system for two-wheelers,” *International Journal of Technical Research & Science (Special Issue)*, 2020. [Online]. Available: <https://doi.org/10.30780/specialissue-ICACCG2020/019>
- [9] M. Sathiyanarayanan, S. Mahendra, y R. B. Vasu, “Smart security system for vehicles using internet of things (iot),” in *2018 Second International Conference on Green Computing and Internet of Things (ICG-CIoT)*, 2018, pp. 430–435.
- [10] A. N. de Tránsito y Seguridad Vial (ANTSV). (Año) Informe anual 2020. [Online]. Available: [https://www.antsv.gov.py/application/files/2716/1943/8500/INFORME\\_ANUAL\\_2020.pdf](https://www.antsv.gov.py/application/files/2716/1943/8500/INFORME_ANUAL_2020.pdf)
- [11] I. Quesada. (2021) Proyecto control y seguimiento automático de vagones rfid-lorawan. <https://doi.org/10.13140/RG.2.2.36244.01925>.
- [12] A. Cama, E. D. la Hoz, y D. Cama, “Las redes de sensores inalámbricos y el internet de las cosas,” *Revista INGE CUC*, vol. 8, no. 1, pp. 163–172, 2012, artículo que analiza la importancia de las *WSN* en el contexto del IoT, destacando estándares como *IEEE 802.15.4*, *6LoWPAN* y *RPL*. [Online]. Available: <https://www.cuc.edu.co/revistas>
- [13] J. G. R. Berrio y R. A. L. Sánchez, *Internet de las Cosas*. Córdoba, España: Red Educativa Digital Descartes, 2024, libro que ofrece una introducción al IoT y su aplicación en diversos sectores. [Online]. Available: <https://proyectodescartes.org/>
- [14] S. M. González, “El internet de las cosas: Definición y aplicaciones,” *Investiga TEC*, Septiembre 2015, el artículo explora las aplicaciones potenciales del IoT y los desafíos que plantea en términos de privacidad y seguridad.
- [15] J. A. Arévalo, “El internet de las cosas,” *DesiderataLAB*, no. 1, pp. 24–25, 2016, análisis sobre la evolución del IoT y su impacto en la transformación de entornos urbanos e industriales.
- [16] G. Martínez Muñoz, “Red de sensores para localización basada en lora y fiware,” 2019. [Online]. Available: <https://hdl.handle.net/11441/92169>
- [17] D. de semtech, “¿qué son lora® y lorawan®?” [Online]. Available: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>

- [18] Semtech, “Semtech lora technology overview,” 2021, disponible en: <https://semtech.com/lora/lora-technology-overview>.
- [19] P. Bertoletti, R. Paiotti, y R. Leca, *Proyectos con ESP32 y LoRa*. Editora NCB, 2019. [Online]. Available: <https://books.google.es/books?id=Doi0DwAAQBAJ>
- [20] T. T. Network, “Frequency plans — the things network,” 2020, the Things Network. Disponible en: <https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/>.
- [21] L. Alliance, “Rp002-1.0.1 lorawan regional parameters,” 2015, parámetros regionales, especificación RP002, noviembre 2015.
- [22] Semtech, “Lora phy — semtech,” 2021, disponible en: <https://semtech.com/lora/lora-phy>.
- [23] L. Alliance, “About lorawan® - lora alliance®,” 2020, LoRa Alliance. Disponible en: <https://lora-alliance.org/about-lorawan/>.
- [24] Semtech. (2021) Lorawan standard — lora — semtech. Semtech. Disponible en: <https://semtech.com/lora/lora-and-lorawan>. [Online]. Available: <https://semtech.com/lora/lora-and-lorawan>
- [25] D. A. Sierra Perera, G. Milián Teijeiro, I. Quesada Hernández, L. Pérez Roche, y P. M. Perera Miniet, “Marco teórico para la implementación de la geolocalización en redes lpwan,” *Revista Telemática*, vol. 22, pp. 45–61, 2023. [Online]. Available: <https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/629/507>
- [26] T. things network, “Arquitectura lorawan,” fecha de acceso, 2024. [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/architecture/>
- [27] J. P. M. Álvarez, V. I. R. Abdalá, F. M. R. M. Barboza, y F. R. C. Soria, “Estudio descriptivo de lorawan y aplicaciones específicas,” *difu100cia*, vol. 15, no. 1, pp. 8–17, Apr. 2021.
- [28] “Security — the things network,” <https://www.thethingsnetwork.org/docs/lorawan/security/>, 2024, explica la implementación de claves de seguridad y las medidas contra ataques de repetición en LoRaWAN.
- [29] S. Gemalto, Actility, “Lorawan™ security white paper,” 2017, describe los principios de autenticación, integridad y confidencialidad en la seguridad de LoRaWAN, incluyendo la encriptación de extremo a extremo

- para datos de aplicaciones IoT. [Online]. Available: [https://lora-alliance.org/wp-content/uploads/2020/11/lorawan\\_security\\_whitepaper.pdf](https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf)
- [30] D. de teleónica, “¿qué es rfid?” [Online]. Available: <https://telelectronica.com/rfid-faq/>
- [31] Mohammad, Gouse, Shitharth, S. Syed, Dugyala, Raman, Rao, K.Sreenivasa, Alenezi, Fayadh, Althubiti, Sara, Polat, y Kemal, “Mechanism of internet of things (iot) integrated with radio frequency identification (rfid) technology for healthcare system,” *Mathematical Problems in Engineering*, vol. 2022, pp. 1–8, 03 2022.
- [32] J. FOMBONA y E. VÁZQUEZ, “Posibilidades de utilización de la geolocalización y realidad aumentada en el ámbito educativo,” 2017, citado el: 18 de Junio de 2021.
- [33] L. I. Benavides Segura y B. D. Cárdenas Espinoza, “Implementación de un dispositivo iot, basado en tecnología lora para la geolocalización y monitoreo fisiológico de personas en lugares turísticos,” dec 2021, escuela Superior Politécnica de Chimborazo. Riobamba. [Online]. Available: <http://dspace.espoch.edu.ec/handle/123456789/21248>
- [34] concepto de servicio, “Gestión de la entrega de servicios.” [Online]. Available: <https://www.nomadia-group.com/es/recursos/blog/la-gestion-de-la-entrega-de-servicios-optimizando-la-satisfaccion-del-cliente/>
- [35] J. M. V. S. Hole y N. Wetterskog, “Lokalisering av sensorer med lorawan på kalmar länssjukhus,” Ph.D. dissertation, Dissertation, 2021. [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:1612938/FULLTEXT01.pdf>
- [36] S. Ponis, G. Plakas, E. Aretoulaki, D. Tzanetou, y T. N. Maroutas, “Lorawan for tracking inland routes of plastic waste: Introducing the smart trackplast bottle,” *Cleaner Waste Systems*, vol. 4, p. 100068, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2772912522000689>
- [37] C. Civelek, “Development of an iot based (lorawan) tractor tracking system,” *Tarım Bilimleri Dergisi*, 08 2021.
- [38] M. Hamidu, “Use of rfid technology as a reporting mechanism in vehicle tracking system,” *Advances in Wireless Communications and Networks*, vol. Volume 2, pp. Pages: 1–10, 12 2016.

- [39] Heltec, “Wifi lora 32 v3,” 2023. [Online]. Available: <https://heltec.org/project/wifi-lora-32-v3/>
- [40] Semtech, “Sx1262,” 2023. [Online]. Available: <https://www.semtech.com/products/wireless-rf/lora-connect/sx1262>
- [41] A. Cherkaoui, S. Merzouk, A. Marzak, y M. Hain, “Review on embedded systems and the internet of things: Comparative study,” *Proceedings of the 4th International Conference on Networking, Information Systems & Security*, 2021.
- [42] A. Latifov y A. Pradeep, “Design and implementation of a lora-based home monitoring system with heltec esp32 gateway,” *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 01–06, 2023.
- [43] N. Jengsriwong y S. Chansareewittaya, “Lorawan gps tracker,” *2023 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, pp. 199–202, 2023.
- [44] N. Semiconductors, “Mfrc522 datasheet,” 2023. [Online]. Available: <https://www.elecrow.com/download/MFRC522%20Datasheet.pdf>
- [45] *GP-02 GPS Development Board Specifications*, Shenzhen Ai-Thinker Technology Co., Ltd, 2021, accessed: Oct. 02, 2024. [Online]. Available: [https://cdn.robotshop.com/media/A/AIT/RB-Ait-24/pdf/ai\\_thinker\\_gp\\_02\\_gps\\_development\\_board\\_gp\\_02\\_kit\\_specifications.pdf](https://cdn.robotshop.com/media/A/AIT/RB-Ait-24/pdf/ai_thinker_gp_02_gps_development_board_gp_02_kit_specifications.pdf)
- [46] E. Systems, “Esp32-wroom-32, datasheet,” 2019, [Online]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/download/1179101/ESPRESSIF/ESP-WROOM-32.html>.
- [47] (2024) Thingsboard - open-source iot platform. ThingsBoard, Inc. Accessed: Oct. 02, 2024. [Online]. Available: <https://thingsboard.io/>
- [48] A. Alquhali, M. Roslee, M. Y. Alias, y K. S. Mohamed, “Iot based real-time vehicle tracking system,” *2019 IEEE Conference on Sustainable Utilization and Development in Engineering and Technologies (CSU-DET)*, pp. 265–270, 2019.
- [49] O. U. Nwankwo, C. I. Nwakanma, D. S. Kim, y J.-M. Lee, “Iot-assisted intelligent vehicle tracking system using cloud computing,” in *2022 13th*

*International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 1677–1679.

- [50] J. Santa, R. Sanchez-Iborra, P. Rodriguez-Rey, L. Bernal-Escobedo, y A. Gómez-Skarmeta, “Lpwan-based vehicular monitoring platform with a generic ip network interface,” *Sensors (Basel, Switzerland)*, vol. 19, 2019.
- [51] M. Suwaid, M. Habaebi, y S. Khan, “Embedded lorawan for agricultural sensing applications,” in *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 2019, pp. 1–5.
- [52] S. A. Alavi, M. Javadipour, y K. Mehran, “State monitoring for situational awareness in rural microgrids using the iot infrastructure,” *ArXiv*, vol. abs/1906.00437, 2019.
- [53] S. K, J. S, S. B, y G. P, “Iot based smart helmet for workers in mines using lorawan,” *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, pp. 1–6, 2023.
- [54] (2024) Mqtt: The standard for iot messaging. MQTT.org. [Online]. Available: <https://mqtt.org/>
- [55] Arduino, “Arduino ide,” [Online]. Disponible: <https://docs.arduino.cc/software/ide/>.
- [56] ——, “Getting started with arduino ide 2,” [Online]. Disponible: <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/>.
- [57] Fritzing, “Fritzing: Software para la automatización del diseño electrónico (eda),” [Online]. Disponible: <https://fritzing.org/>.
- [58] ——, “Fritzing: Primeros pasos,” [Online]. Disponible: <https://fritzing.org/media/uploads/learning/translations/Fritzing-PrimerosPasos.pdf>.
- [59] (2024) About blender. Blender Foundation. [Online]. Available: <https://www.blender.org/about/>
- [60] (2024) Blender. Blender Foundation. [Online]. Available: <https://www.blender.org/>
- [61] (2024) Helium console. Helium Foundation. [Online]. Available: <https://console.helium.com/>

- [62] D. de Heltec, “Documentación de heltec oficial para el uso de wifi lora 32.” [Online]. Available: [https://docs.heltec.org/en/node/esp32/wifi\\_lora\\_32/index.html#](https://docs.heltec.org/en/node/esp32/wifi_lora_32/index.html#)
- [63] D. de Helium, “Guía de arduino para heltec wifi lora 32 v2.” [Online]. Available: <https://docs.helium.com/network-iot/devices/development/heltec/wifi-lora-32-v2/arduino/>
- [64] (2024) Installing thingsboard using docker (windows). ThingsBoard, Inc. Accessed: Oct. 20, 2024. [Online]. Available: <https://thingsboard.io/docs/user-guide/install/docker-windows/>
- [65] (2024) Installing thingsboard doc). ThingsBoard, Inc. Accessed: Nov. 01, 2024. [Online]. Available: <https://thingsboard.io/docs/guides/>
- [66] (2024) Google maps. Google LLC. [Online]. Available: <https://maps.google.com>
- [67] G. Earth, “Google earth - visualización de coordenadas y medición de distancias,” <https://earth.google.com/web/>, 2024, imágenes utilizadas para validar las coordenadas obtenidas con el módulo GNSS y medir distancias entre puntos.
- [68] O. Calculator, “Calculadora de distancia entre coordenadas (latitud y longitud),” <https://www.omnicalculator.com/es/otros/calculadora-latitud-longitud-distancia>, 2024, herramienta utilizada para calcular distancias precisas entre puntos GPS adquiridos.
- [69] (2024) Helium explorer. Helium Foundation. [Online]. Available: <https://explorer.helium.com/>
- [70] (2024) Base64 to hex converter. Cryptii. [Online]. Available: <https://cryptii.com/pipes/base64-to-hex>
- [71] (2024) Online hex converter. SCADACore. [Online]. Available: <https://www.scadacore.com/tools/programming-calculators/online-hex-converter/>