UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
CARRERA EN TELECOMUNICACIONES

Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario "Vida y Salud".

Trabajo de Titulación para optar al título de Ingeniero en Telecomunicaciones

Autor:
Arévalo Ocaña, Alex Dario
Robalino Navarrete, Edwin Miguel

Tutor:
Phd. Leonardo Fabián Rentería Bustamante.

Riobamba, Ecuador. ***2025***

DECLARATORIA DE AUTORÍA

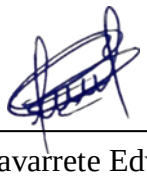
Nosotros, Arévalo Ocaña Alex Dario, con cédula de ciudadanía 0605414507 y Robalino Navarrete Edwin Miguel, con cédula de ciudadanía 0605791698, autores del trabajo de investigación titulado: **Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario "Vida y Salud"**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 12 de mayo de 2025.

Arévalo Ocaña Alex Dario

C.I: 0605414507



Robalino Navarrete Edwin Miguel

C.I: 0605791698

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 12 días del mes de MAYO de 2025, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante ALEX DARIO AREVALO OCAÑA con CC: **0605414507,** de la carrera de **INGENIERIA EN TELECOMUNICACIONES** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado "**Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario Vida y Salud"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
LEONARDO FABIAN
RENTERIA BUSTAMANTE
Validar únicamente con FirmaEC

_____

PhD. Leonardo Rentería
TUTOR(A)

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 12 días del mes de MAYO de 2025, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante EDWIN MIGUEL ROBALINO NAVARRETE con CC: **0605791698,** de la carrera de **INGENIERIA EN TELECOMUNICACIONES** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado "**Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario Vida y Salud"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
LEONARDO FABIAN
RENTERIA BUSTAMANTE
Validar únicamente con FirmaEC

_____

PhD. Leonardo Rentería
TUTOR(A)

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "**DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA MONITOREO VETERINARIO DE SIGNOS VITALES Y RASTREO DE MASCOTAS EMPLEANDO LA TECNOLOGÍA LORAWAN EN EL CENTRO CLÍNICO VETERINARIO VIDA Y SALUD**", presentado por **ARÉVALO OCAÑA ALEX DARIO**, con cédula de identidad número **0605414507**, bajo la tutoria de **PHD. LEONARDO FABIÁN RENTERÍA BUSTAMANTE**.; certificamos que recomendamos la **APROBACIÓN** de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a los 09 días del mes de junio de 2025.

PhD. Manuel Antonio Meneses Freire
PRESIDENTE DEL TRIBUNAL DE GRADO

Firma

Ing. José Luis Jinez Tapia
MIEMBRO DEL TRIBUNAL DE GRADO

Firma

Ing. Luis Gonzalo Santillán Valdiviezo
MIEMBRO DEL TRIBUNAL DE GRADO

Firma

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "**DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA MONITOREO VETERINARIO DE SIGNOS VITALES Y RASTREO DE MASCOTAS EMPLEANDO LA TECNOLOGÍA LORAWAN EN EL CENTRO CLÍNICO VETERINARIO VIDA Y SALUD**", presentado por **ROBALINO NAVARRETE EDWIN MIGUEL**, con cédula de identidad número **0605791698**, bajo la tutoría de **PHD. LEONARDO FABIÁN RENTERÍA BUSTAMANTE**.; certificamos que recomendamos la **APROBACIÓN** de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba el 09 de junio de 2025.

PhD. Manuel Antonio Meneses Freire
PRESIDENTE DEL TRIBUNAL DE GRADO

Firma

Ing. José Luis Jinez Tapia
MIEMBRO DEL TRIBUNAL DE GRADO

Firma

Ing. Luis Gonzalo Santillán Valdiviezo
MIEMBRO DEL TRIBUNAL DE GRADO

Firma

C E R T I F I C A C I Ó N

Que, **ARÉVALO OCAÑA ALEX DARIO** con CC: **0605414507**, estudiante de la Carrera de **TELECOMUNICACIONES** , Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario Vida y Salud",** cumple con el **10%**, de acuerdo al reporte del sistema Anti plagio **COMPILATIO**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 02 de junio de 2025.

Firmado electrónicamente por:
LEONARDO FABIAN
RENTERIA BUSTAMANTE
Validar únicamente con FirmaEC

Phd. Leonardo Fabián Rentería Bustamante
TUTOR

C E R T I F I C A C I Ó N

Que, **ROBALINO NAVARRETE EDWIN MIGUEL** con CC: **0605791698**, estudiante de la Carrera de **TELECOMUNICACIONES** , Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"Diseño e implementación de un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario Vida y Salud",** cumple con el **10%**, de acuerdo al reporte del sistema Anti plagio **COMPILATIO**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 02 de junio de 2025.

Firmado electrónicamente por:
LEONARDO FABIAN
RENTERIA BUSTAMANTE
Validar únicamente con FirmaEC

Phd. Leonardo Fabián Rentería Bustamante
TUTOR

DEDICATORIA

A mi madre, Glenda Navarrete, cuyo amor incondicional, fortaleza y paciencia han sido el motor que me impulsó a seguir adelante. Gracias por estar siempre a mi lado, por tus palabras de aliento en los momentos difíciles y por enseñarme a nunca rendirme. Este logro es tan tuyo como mío. A mi hermana, Lizbeth Robalino, por ser mi compañera en cada paso del camino, por tu apoyo constante y por recordarme siempre la importancia de luchar por mis sueños.

A ambas, les dedico este trabajo con profundo agradecimiento y con el deseo de que vean en este logro un reflejo del esfuerzo y amor que me han brindado.

Miguel Robalino

A mi madre, Luz Ocaña, cuya determinación, apoyo constante y sabiduría han sido el cimiento sobre el cual se ha construido este logro. Su ejemplo de fortaleza y entrega ha guiado mi camino con claridad y propósito. A mi tía, Rebeca Manzano, por ser presencia firme y voz de aliento en momentos clave. Su confianza en mí ha sido impulso silencioso pero decisivo. A mi hermana, Daniela Estévez, que desde lo eterno habita en mi memoria y mi causa. Su recuerdo no solo me acompaña, sino que me inspira a seguir con firmeza, en honor a todo lo compartido y a lo que aún se construye desde el amor. A mí mismo, por mantenerme de pie con integridad, por avanzar con visión y resiliencia, y por demostrar que la disciplina y la convicción abren caminos incluso en medio de la incertidumbre.

A cada uno, gracias por formar parte de esta conquista.

Alex Arévalo

AGRADECIMIENTO

Agradezco primeramente a Dios, por brindarme salud, sabiduría y fortaleza a lo largo de este proceso. Sin Su guía y bendiciones, no hubiera sido posible alcanzar esta meta.

A mi madre, Glenda Navarrete, quien ha sido mi mayor inspiración y ejemplo de vida. Tu amor incondicional, tu sacrificio constante y tu paciencia infinita me enseñaron a nunca rendirme, incluso en los momentos más difíciles. Gracias por ser mi refugio en los días oscuros y mi mayor motivación en los días de triunfo. Cada palabra de aliento, cada abrazo y cada gesto de apoyo quedarán grabados en mi corazón como recordatorio del inmenso amor que me has brindado. Este logro es tan tuyo como mío. A mi hermana, Lizbeth Robalino, por ser mi compañera y amiga incondicional. Tu presencia y palabras de aliento fueron esenciales para mantenerme firme y enfocado en cada paso de este camino. A mi familia, por su constante apoyo, sus muestras de cariño y por ser una fuente inagotable de motivación. Gracias por estar presentes en cada etapa de mi formación.

A mi tutor de tesis, PhD. Leonardo Rentería, por su orientación, paciencia y valiosos aportes durante el desarrollo de este trabajo.

A todos ustedes, mi más sincero agradecimiento por ser parte fundamental de este logro.

Miguel Robalino

Expreso mi sincero reconocimiento al PhD Leonardo Rentería, tutor de esta tesis, por su orientación académica, su criterio técnico y el acompañamiento riguroso que fortaleció cada etapa de este trabajo. A la Magíster Deisy Inca, quien, más allá de su rol como directora, representó un respaldo humano invaluable. Su liderazgo cercano y sus palabras oportunas fueron sostén en momentos clave. A la Universidad, por brindar un espacio de formación sólida, y a los docentes que aportaron con su conocimiento a lo largo del camino académico.

A mi madre, Luz Ocaña, y a mi tía, Rebeca Manzano, por su apoyo firme y constante, y a mi hermana, Daniela Estévez, cuya memoria sigue siendo fuerza y guía.

A todos quienes, de forma directa o indirecta, contribuyeron a la culminación de este proceso, les extiendo mi respeto y gratitud.

Alex Arévalo

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Existe una necesidad inevitable de herramientas, mediante las cuales se pueda controlar de manera remota los parámetros fisiológicos y localización de los canes. Por tal motivo, en este trabajo se presentó el diseño e implementación de un prototipo de monitoreo veterinario, incluyendo sensores biomédicos, y ubicación de mascotas, añadiendo un módulo de geolocalización, utilizando la tecnología LoRaWAN.

El sistema está conformado de una malla ajustable integrada a un chaleco adaptado para la inclusión del transmisor con los sensores con el objetivo de medir en todo momento la frecuencia cardíaca y SpO2 con el sensor MAX30102, la temperatura corporal con el sensor MLX90614, además de la actividad de la mascota con el MPU6050. Asimismo, se incorpora un módulo GPS para la ubicación en tiempo real. Todos los datos se procesan en un ESP32, donde se verifican, filtran y organizan para su posterior transmisión a través de LoRaWAN. La información captada por el receptor es almacenada en la plataforma Firebase y posteriormente se accede a ella mediante una aplicación móvil desarrollada en Android Studio, lo que facilita el monitoreo en tiempo real del estado fisiológico y ubicación de la mascota.

Fue importante conocer el desempeño del prototipo, por este motivo, se efectuaron pruebas en la clínica veterinaria, y los datos medidos se compararon con los del dispositivo veterinario certificado. Además, se realizaron pruebas de localización y se registraron coordenadas geográficas que posteriormente se verificaron con los datos de Google Maps. Se utilizaron 10 canes, de los cuales se registraron 1200 datos en la medición de signos vitales y en la geolocalización se obtuvieron 16 datos de latitud y longitud, para ser después evaluados estadísticamente.

Por medio del análisis estadístico, los datos recopilados por el prototipo coincidieron en gran medida con los del dispositivo veterinario certificado, ya que fue capaz de medir los signos vitales y ubicación de una manera eficiente. Esto asegura la confiabilidad del prototipo en el monitoreo, además, la tecnología LoRaWAN facilitó la comunicación gracias a su amplia cobertura y bajo consumo energético.

Palabras claves: Monitoreo veterinario, Sensores biomédicos, LoRaWAN, Geolocalización.

ABSTRACT

Dogs often require remote monitoring of their physiological parameters and location, necessitating the development of appropriate tools. To address this need, this research presents a veterinary monitoring prototype. This system integrates biomedical sensors and a geolocation module utilizing LoRaWAN technology. The prototype consists of an adjustable mesh vest designed to house the transmitter and its sensors. Specifically, a MAX30102 sensor continuously measures heart rate and SpO2, an MLX90614 sensor monitors body temperature, and an MPU6050 tracks the pet's activity. A GPS module is also incorporated for real-time location tracking. An ESP32 processes all captured data, where it is verified, filtered, and organized before transmission via LoRaWAN. The receiver then captures this information, which is stored on the Firebase platform and made accessible through a mobile application developed in Android Studio. This facilitates real-time monitoring of the pet's physiological state and location. To evaluate the prototype's performance, tests were conducted at a veterinary clinic. Measured data were compared against those from a certified veterinary device. Furthermore, location tests were performed, with recorded geographic coordinates subsequently verified using Google Maps. The study involved ten canines, yielding 1200 vital sign measurements and 16 latitude and longitude data points for geolocation, all of which underwent statistical analysis. Statistical analysis revealed a strong correlation between the data collected by the prototype and that from the certified veterinary device, demonstrating the prototype's efficient measurement of vital signs and location. This confirms the prototype's reliability for remote monitoring. Moreover, the use of LoRaWAN technology proved beneficial due to its wide coverage and low power consumption, enhancing communication capabilities.

Keywords: Veterinary remote monitoring, Biomedical sensors, LoRaWAN, Geolocation.

Reviewed and improved by Jacqueline Armijos

CAPÍTULO I.

1.1 INTRODUCCIÓN

En los últimos años, con el avance continuo de la tecnología y la necesidad de una conectividad de bajo costo, ha surgido el concepto de Internet de las Cosas (IoT) [1] para desarrollar soluciones de conectividad inalámbrica que sean capaces de enlazar múltiples dispositivos. Con capacidad de tener un alcance amplio a muchos dispositivos, un bajo costo de comunicación y bajo consumo energético se destacan las tecnologías de redes de área amplia de baja potencia (LPWANs) [2].

Una de las tecnologías que destacan en el grupo de las LPWANs y bajo el contexto mencionado anteriormente es la red de área amplia de largo alcance (LoRaWAN). Su funcionamiento se basa en estaciones base conectadas para enviar datos desde sensores o dispositivos de seguimiento. Cabe señalar que utiliza la modulación Long Range (LoRa) [3] que permite la transmisión de datos a través de largas distancias con una buena eficiencia energética y bajo impacto económico. De esta manera, la tecnología LoRa es la responsable de permitir la conexión de larga distancia en la comunicación, mientras que LoRaWAN establece el protocolo de comunicación y la estructura del sistema de red [4].

Gracias a estos avances uno de los sectores más beneficiado ha sido el de atención y salud de mascotas, por el simple hecho de que forman parte de la familia y es necesario valorar su estado de salud y bienestar para una mejor calidad de vida. En este contexto, la tecnología LoRaWAN [5] ha creado nuevas oportunidades para la monitorización y el cuidado de los animales domésticos, proporcionando a los propietarios una mayor tranquilidad y control.

Por lo tanto, el presente proyecto consistió en diseñar e implementar un prototipo orientado al monitoreo veterinario de signos vitales y rastreo de mascotas empleando tecnología LoRaWAN. La finalidad fue brindar una solución de monitoreo en tiempo real para que los dueños de mascotas puedan obtener información detallada sobre la salud de sus animales y, por consiguiente, tomar medidas preventivas ante cualquier problema que se detecte. Asimismo, el sistema de rastreo permitió conocer la ubicación de los animales en todo momento, lo cual resulta especialmente valioso en situaciones de extravío o robo.

Realizar un estudio profundo sobre la tecnología LoRaWAN y sus diversas aplicaciones para el monitoreo de signos vitales, además de aprender conceptos sobre la medicina veterinaria, supuso un reto, sobre todo este último por su complejidad al tratar de comprender la fisiología de los animales y métodos adecuados de medición, por ello, al inicio se presentaron ciertos desafíos con respecto al contacto adecuado de los sensores con la piel de la mascota sin afectar el bienestar de esta.

Al entender lo anterior mencionado, fue indispensable diseñar e implementar un sistema que mida y transmita signos vitales de forma correcta, precisa y de manera remota, sin que exista pérdidas de información que podrían resultar catastróficas para una mascota si estuviese enferma, asimismo asegurar que el prototipo sea portátil y cómodo para los canes. Un punto esencial fue conseguir el chaleco adecuado y ergonómico donde se pudiera acoplar el transmisor con sus sensores biomédicos, para disminuir molestias y así evitar lo más posible las interferencias al momento de medir los parámetros de temperatura, frecuencia cardíaca y saturación de oxígeno.

En la parte técnica, se integraron componentes fundamentales como el módulo GPS [6], el microcontrolador [7], el módulo LoRa [8] y el dispositivo Gateway LoRaWAN [9], seleccionados conforme a los requerimientos de precisión, bajo consumo energético y conectividad de largo alcance definidos en el diseño general. En definitiva, la complejidad del proyecto residió en armonizar estos elementos dentro de una solución funcional, ergonómica y adecuada con base en los fundamentos veterinarios estudiados para su aplicación en el monitoreo veterinario en condiciones reales.

1.2 PLANTEAMIENTO DEL PROBLEMA

La evaluación de las constantes fisiológicas en animales de compañía, particularmente la frecuencia cardíaca, la temperatura corporal y la saturación de oxígeno en sangre, constituye un procedimiento esencial para la valoración integral de su condición fisiológica y la identificación oportuna de patologías o estados críticos. A nivel internacional, la vigilancia permanente de los parámetros vitales en mascotas presenta todavía limitaciones importantes, derivadas principalmente de la predominancia de técnicas de medición manual. En el panorama ecuatoriano, pese a evidenciar cierta evolución tecnológica en el

ámbito de la medicina veterinaria, carece aún de la implementación generalizada de dispositivos automatizados para el control remoto de estos indicadores.

La situación se torna más evidente en el entorno específico de Riobamba, donde la monitorización de signos vitales se ejecuta exclusivamente mediante procedimientos presenciales en establecimientos veterinarios, circunstancia que incrementa significativamente los requerimientos temporales, la asignación de personal sanitario y la necesidad de supervisión ininterrumpida. En el establecimiento veterinario donde se implementó la presente iniciativa, la afluencia diaria de pacientes caninos resulta considerable, registrándose entre ellos diversos casos de gravedad asociados a trastornos cardíacos, complicaciones respiratorias o desequilibrios metabólicos, lo que evidencia la imperiosa necesidad de establecer mecanismos de vigilancia más eficientes. Adicionalmente, la capacidad de determinar la posición geográfica de los animales domésticos adquiere particular relevancia en escenarios de tratamiento ambulatorio o situaciones de pérdida, contextos en los cuales la precisión en la localización se transforma en un factor determinante para su recuperación o auxilio.

Cabe destacar que la infraestructura LoRaWAN, reconocida mundialmente por su bajo consumo energético y amplio alcance de transmisión, mantiene una presencia incipiente en territorio ecuatoriano, con particular escasez de conocimientos técnicos e infraestructura física y tecnológica en Riobamba. Esta insuficiencia tecnológica obstaculiza considerablemente el desarrollo e implementación de soluciones innovadoras en el ámbito veterinario, limitando el aprovechamiento de recursos técnicos potencialmente beneficiosos para la optimización de la atención animal.

La implementación de un dispositivo de monitoreo veterinario fundamentado en la tecnología LoRaWAN se orienta a subsanar las deficiencias identificadas en el ámbito de la supervisión de parámetros fisiológicos y la localización de animales domésticos. La presente iniciativa contempla el diseño e implementación de un instrumento técnico capaz de capturar y transmitir de forma continua los indicadores biomédicos y coordenadas geográficas, facilitando la identificación temprana de variaciones significativas en el estado fisiológico de los pacientes veterinarios.

Esta solución tecnológica permitirá optimizar los recursos disponibles, reducir la frecuencia de desplazamientos presenciales a instalaciones veterinarias y disminuir la sobrecarga laboral del personal médico veterinario. Adicionalmente, la automatización del proceso de monitorización mejorar significativamente la capacidad de reacción ante emergencias veterinarias, contribuyendo al establecimiento de protocolos asistenciales más expeditos y eficaces.

La propuesta aspira, asimismo, a introducir y validar estadísticamente la aplicación de la tecnología LoRaWAN en el contexto veterinario en la ciudad de Riobamba, estableciendo un antecedente significativo para su futura implementación tanto en el ámbito local como a escala nacional, promoviendo la modernización tecnológica del sector veterinario ecuatoriano y estableciendo nuevos estándares en el cuidado animal.

1.3 OBJETIVOS

1.3.1 Objetivo General

- Diseñar e implementar un prototipo para monitoreo veterinario de signos vitales y rastreo de mascotas empleando la tecnología LoRaWAN en el Centro Clínico Veterinario “Vida y Salud”.

1.3.2 Objetivos Específicos

- Investigar el estado del arte de la tecnología LoRaWAN y sus aplicaciones en el monitoreo de signos vitales y rastreo de mascotas, identificando las limitaciones y oportunidades.

- Diseñar e Implementar un prototipo y una red LoRaWAN para el monitoreo veterinario de signos vitales y rastreo de mascotas, integrando sensores y dispositivos de seguimiento.

- Evaluar la funcionalidad y confiabilidad del prototipo, llevando a cabo pruebas en un entorno controlado en el Centro Clínico Veterinario “Vida y Salud”.

CAPÍTULO II.

2.1 MARCO TEÓRICO.

2.1.1 LoRa

LoRaWAN es una red de área amplia de baja potencia que utiliza la tecnología de modulación de radiofrecuencia conocida como LoRa. En otras palabras, LoRa constituye la capa física de la red LoRaWAN.

2.1.2 Arquitectura de red

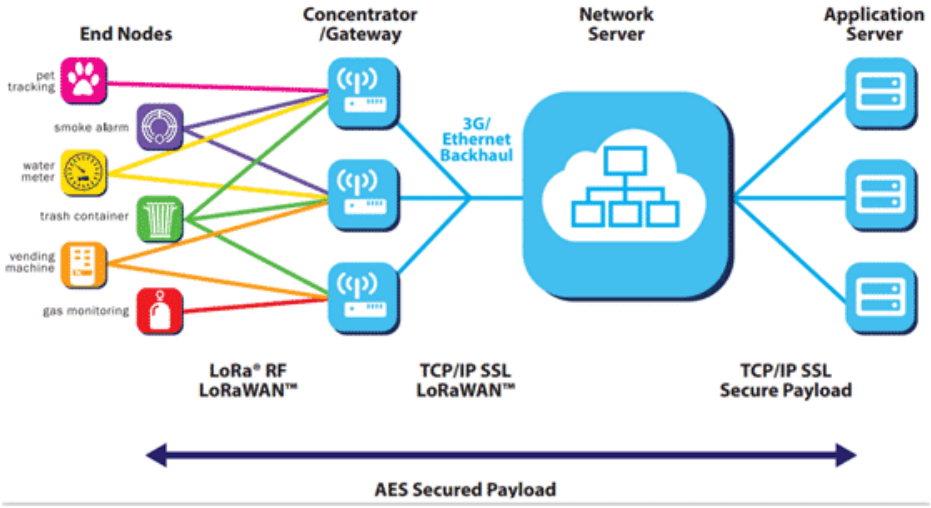


Figura 1. Arquitectura de red.

Fuente: [10]

LoRa utiliza una topología de red en forma de estrella de un solo salto, lo que significa que no requiere elementos de enrutamiento complicado [11]. Los componentes principales de una red LoRa son los siguientes:

- Dispositivos finales: Estos son los dispositivos utilizados para conectar objetos a la red LoRa. Los dispositivos finales recopilan información específica de los objetos y la transmiten a través de comunicaciones inalámbricas de salto único hacia una o varias pasarelas [11].

- Gateway o Pasarela: Las pasarelas, también conocidas como gateways, actúan como estaciones base LoRa. Su función es actuar como puentes transparentes que transmiten de manera bidireccional las comunicaciones realizadas por múltiples dispositivos finales hacia los servidores de red [11].

- Servidores de red: Los servidores de red se conectan a varias pasarelas a través de una conexión TCP/IP segura, ya sea mediante cables o de forma inalámbrica. Estos servidores son responsables de recibir y procesar la información proveniente de los dispositivos finales. Realizan tareas como la eliminación de datos duplicados, la gestión y configuración de la red y los dispositivos finales [11].

- Servidor de aplicaciones: Los servidores de aplicaciones recopilan y analizan los datos recibidos de los dispositivos finales. Estos servidores determinan las acciones o respuestas adecuadas basadas en los datos recopilados. Por ejemplo, pueden generar alertas, notificaciones o activar acciones específicas en los dispositivos finales [11].

Los gateways retransmiten la información proveniente de los dispositivos finales a un servidor de red utilizando una conexión IP estándar. Esto implica que las estaciones base actúan como puentes, lo que facilita el diseño de la red y sus componentes [11]. Por otro lado, los gateways, como no desempeñan funciones de enrutamiento, no se retransmite el tráfico, por tal motivo se obtiene un ahorro energético.

En la comunicación de extremo a extremo, una red LoRa normalmente es bidireccional. No obstante, LoRa sí admite una funcionalidad de multidifusión, que permite enviar información de manera simultánea a varios dispositivos. Esto es especialmente útil para tareas como actualizaciones de software en varios dispositivos de la red.

2.1.3 LoRaWAN

LoRaWAN es un protocolo de red que usa la tecnología LoRa para comunicar y administrar dispositivos LoRa, por tanto, es la capa de acceso al medio y se desarrolla de forma abierta por LoRa Alliance.

2.1.3.1 Clases LoRaWAN

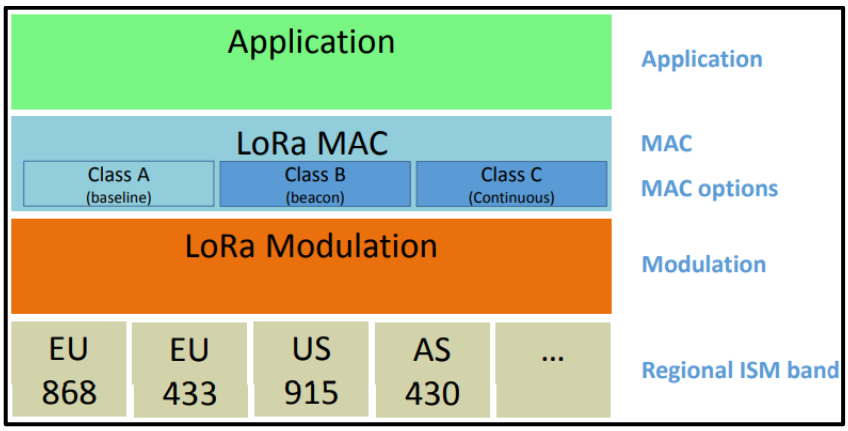


Figura 2. Relación LoRa-LoRaWAN.

Fuente: [10]

- En la red LoRaWAN, los dispositivos se clasifican en tres clases según las funcionalidades que admiten: Clase A, Clase B y Clase C. Las tres clases permiten la comunicación bidireccional y la capacidad de iniciar una transmisión hacia los servidores a través del gateway[10]. Sin embargo, difieren en cuanto a cuándo aceptan mensajes entrantes del servidor:
- Clase A: Estos dispositivos ofrecen el mayor ahorro de energía. Después de enviar un mensaje al gateway, entran en un período de espera (ventana de recepción) durante un tiempo predeterminado. Solo escuchan los mensajes entrantes durante esta ventana después de cada transmisión realizada. Después de la ventana de recepción, vuelven al modo de bajo consumo de energía. La comunicación en Clase A se empieza siempre desde el dispositivo final [10].

- Clase B: Estos dispositivos eliminan la restricción de recepción de datos solo después de la transmisión de un paquete. Además de la ventana de recepción, como en Clase A, tienen ventanas adicionales de recepción programadas en momentos

específicos sincronizados con el gateway. Esto permite una mayor flexibilidad para recibir datos del servidor en momentos predefinidos, aunque también consume algo más de energía que la Clase A [10].

- Clase C: Estos dispositivos ofrecen el menor ahorro de energía, ya que están en modo de escucha continuamente, excepto durante la transmisión. En Clase C, la ventana de recepción está siempre abierta, lo que permite una comunicación bidireccional más rápida y una menor latencia. Estos dispositivos son ideales para casos de uso que requieren una respuesta rápida en ambos sentidos, pero a costa de un mayor consumo de energía en comparación con las otras clases [10].

2.1.4 Medicina veterinaria

La medicina veterinaria, además de ser una ciencia que utiliza los principios de la medicina en el cuidado de los animales. Su función cubre desde la prevención, diagnóstico y tratamiento de enfermedades, hasta la atención quirúrgica y nutrición. En la actualidad, existe un rol más extenso para el médico veterinario, como el control de alimentos y centros sanitarios para los animales, prevenir la zoonosis, entre otros. En la medicina veterinaria se encuentran especialidades como: Medicina interna, Dermatología, Odontología, Oftalmología y muchos más. [12].

2.1.4.1 Anatomía y Fisiología

La anatomía animal estudia el número, estructura, tamaño, forma, disposición, situación y relaciones de las diferentes partes internas y externas de los animales [12]. Por otro lado, la fisiología estudia el funcionamiento del organismo de los animales.

En los canes es importante entender los principios básicos de la anatomía, por ello se inicia con las células que es la unidad funcional de todos los tejidos y tienen la habilidad para realizar todas las funciones vitales esenciales. Además, tenemos a los tejidos básicos los cuales son conformados por: Tejido muscular, tejido epitelial, tejido conjuntivo, sangre, cartílago, huesos, tejido hemopoyético y tejido nervioso. Por último, las cavidades corporales que son muy importantes y están divididas en tres: Cavidad torácica, cavidad abdominal y cavidad pélvica [12].

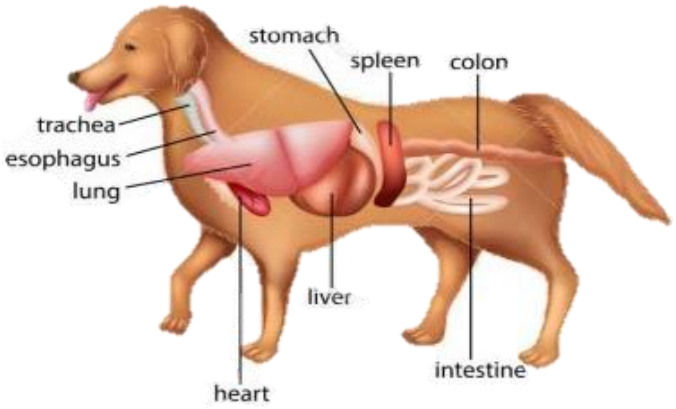


Figura 3. Anatomía del perro

Fuente:[12]

Por otra parte, en la fisiología todas las áreas son importantes, aunque la fisiología cardiovascular, función respiratoria y homeostasis son las más relevantes.

- El sistema cardiovascular es fundamental ya que aquí se transporta nutrientes, oxígeno, desechos, hormonas y calor por todo el organismo, esto asegura que los tejidos obtengan lo necesario para mantenerse funcionales. Es decir, que en un tejido si no llega la sangre suficiente se produce isquemia, que puede llegar hasta la necrosis, la cual deriva en un infarto. Por ello, es esencial la circulación constante ya que puede provocar hasta daños irreversibles. Cualquier cambio en una parte en específico afecta todo el sistema, y es crucial el entendimiento de este sistema [13].
- En el sistema respiratorio existen vías que conceden el paso del aire estas son las narinas, fosas nasales, faringe y laringe. Además, con los alvéolos sucede el intercambio gaseoso, que es el paso del aire a la sangre y el dióxido de carbono se descarta en sentido contrario. Este sistema es interesante, ya que según el tamaño o nivel de actividad del animal se adapta al metabolismo, por eso en perros el consumo de oxígeno es alto cuando realizan una actividad física esto por su gran concentración mitocondrial muscular. Por último, se le atribuyen la regulación térmica, vocalización, entre otros [13].

2.1.4.2 Exámenes Físicos

Estado de conciencia

En un animal el estado de conciencia es primordial para obtener información. Debido a que, si el paciente responde, está alerta, consciente con relación a un paciente que no responde, está somnoliento o en coma, emitir un diagnóstico preciso es esencial para dar una atención inmediata. No se debe dejar de lado distinguir los estados mentales, ya que podría deberse a enfermedades neurológicas primarias o secundarias, como podría ser una toxina o una mala perfusión por un estado de shock [14].

Temperatura

La temperatura corporal es una base importante para valorar el estado del paciente, se mide generalmente de formal rectal, lo cual refleja la temperatura central de cuerpo. Además, estudios indican que es un estándar de oro para la medición de temperatura central. Existen otros métodos para una medición de temperatura como son la axilar o auricular, pero son menos precisas. Para los canes las mediciones axilares y auriculares son diferentes en un 98% y 56% respectivamente, en comparación con las rectales (± 0.5 °C). La medición de temperatura brinda información sobre un paciente que puede estar con hipotermia que es causada por la exposición al ambiente, shock u otras condiciones, o hipertermia, que se suelen relacionar con el ambiente o actividad intensa. Además, en perros que sufren enfermedades respiratorias u obesidad no pueden jadear de manera correcta para disminuir el calor. Otro factor está relacionado a la fiebre ya que es un síntoma de infecciones o algunas otras enfermedades que no necesariamente se relacionan con la disipación de calor [14].

Frecuencia cardíaca, ritmo e intensidad del pulso

La frecuencia cardíaca se puede evaluar por medio de la auscultación torácica y palpitación simultánea de los pulsos periféricos. En la auscultación también se caracterizan los ruidos cardíacos, como los soplos. Palpar los pulsos periféricos es una forma complementaria en la auscultación para una mejor evaluación. La valoración del estado cardiovascular en el paciente decide su estabilidad ya que, si se identifica frecuencias cardíacas extremadamente altas, bajas, así como pulsos bajos, se necesita de tratamiento urgente [14].

Saturación de oxígeno

La saturación de oxígeno normal en un paciente que esté respirando el aire del ambiente oscila entre 96% y 98%, y esto representa a una presión arterial de 85 a 100 mmHg. Un término importante que se relaciona fuertemente con la saturación es la hipoxemia ya que si el SpO2 está por debajo de 95% o si es menor que 90% se consideran hipoxémicos o tienen una hipoxemia severa. En el oxímetro existe una precisión de ±2-3% a diferencia de una gasometría arterial, y en pacientes con una SaO2 menor a 90%, el oxímetro tiene una variación de hasta un ±5%, por esta razón los valores deben ser más de interpretación que centrarse en un valor exacto [14].

2.1.5 Monitoreo de signos vitales en medicina veterinaria

Es importante una adecuada interpretación y la actuación oportuna en ciertas situaciones de salud de las mascotas que requieren un análisis preciso de los signos vitales, ya que son indicadores fisiológicos que muestran el estado de los pacientes.

Esto es igualmente aplicable al monitoreo de los signos vitales de los perros, que incluyen la temperatura corporal, la frecuencia respiratoria y el pulso arterial. Sin embargo, medir estos signos en perros puede presentar desafíos debido a su tamaño, forma y pelaje, lo que dificulta la obtención de señales precisas [15]. Por eso se debe tener en cuenta factores muy importantes como el peso, el sexo, la edad y la alimentación porque pueden influir en estas medidas.

Por tal razón, un objetivo clave en el desarrollo de las tecnologías para las mediciones de los signos vitales en mascotas es que sean fáciles y efectivas. Es fundamental reconocer que los métodos de medición convencionales establecen una base sólida para el desarrollo de nuevas técnicas en este campo.

Figura 4. Monitoreo de signos vitales.

Fuente: [16]

2.1.6 Dispositivos y sensores para monitoreo veterinario

2.1.6.1 Sensores para monitoreo:

Estos sensores se utilizarán para recopilar datos sobre los signos vitales[17] de la mascota, como la saturación de oxígeno [18], la frecuencia cardíaca [19] y la temperatura [20]. Hay una variedad de sensores que se pueden usar para este propósito.

2.1.6.2 Sensor de temperatura:

Este sensor se utilizará para medir la temperatura corporal de la mascota. Por lo general, en los animales, la temperatura se mide con los termómetros que son introducidos mediante vía rectal, esto puede producir una reacción agresiva por el malestar del procedimiento. Existe otro menos invasivo e igual de confiable, es mediante un sensor infrarrojo que permite realizar mediciones de temperatura sin ese tipo de contacto [21].

2.1.6.3 Sensor de concentración de oxígeno y frecuencia cardíaca:

Este sensor mide la saturación de oxígeno en la sangre de la mascota y está ideado para cumplir con las exigencias de los dispositivos portátiles. Además, destaca por tener un tamaño compacto, sin involucrar sus funciones eléctricas y ópticas [22].

Por otro lado, este sensor también mide el número de veces que late el corazón por minuto. La frecuencia cardíaca, es otro signo vital que es importante controlar, ya que sus cambios pueden ser un signo de enfermedad o lesión, esta se estima midiendo los intervalos de tiempo entre los latidos [21]. El sensor funciona detectando las señales

eléctricas que genera el corazón. Estas señales se utilizan luego para calcular la frecuencia cardíaca.

2.1.6.4 Módulo GPS:

Este módulo es un dispositivo electrónico compacto y su objetivo es recibir señales satelitales, de tal manera que logre calcular de una forma precisa la posición geográfica de un objeto o persona en la tierra mediante un proceso de triangulación. En este contexto, el Sistema de Posicionamiento Global (GPS) sustituyó métodos tradicionales de navegación [23].

2.1.6.5 Módulo LoRa:

Este módulo es un dispositivo que opera en bandas de frecuencia ISM como 433, 868 y 915 MHz. Permite la comunicación inalámbrica de largo alcance y bajo consumo de energía, por tal motivo son apropiados para aplicaciones IoT. Además, pueden conectarse directamente con sensores mediante microcontroladores como Arduino uno, Mega o ESP32, por ende, facilita la obtención de datos físicos como temperatura, presión, corriente o voltaje [24].

2.1.6.6 Microcontrolador:

Esta es la unidad central de procesamiento del sistema. Será responsable de recopilar datos de los sensores, transmitir los datos a la puerta de enlace y recibir comandos de la puerta de enlace [25].

2.1.6.7 Gateway:

El gateway LoRaWAN permite una fácil interconexión de la comunicación entre los sensores de una red LoRaWAN y un servidor de red. De tal forma que se reciben datos de los sensores en formato LoRaWAN y son transformados a paquetes UDP, en donde el gateway trabaja como un puente entre ambos mundos. Por otro lado, es el encargado de transmitir los comandos y configuraciones del servidor a los sensores, de esta manera se completa el ciclo de comunicación. Con respecto a la capacidad del gateway, depende mucho de la antena, ya que se puede gestionar una gran cantidad de nodos además de su

alcance de comunicación, por ello es un componente clave que define el desempeño de la red [26].

2.1.6.8 Flutter

Flutter es un framework de desarrollo multiplataforma propuesto en el año 2016 por Google. El principal objetivo es facilitar el desarrollo de aplicaciones móviles de alto rendimiento a partir de un único código base, y que son compatibles con varios sistemas operativos como: iOS y Android. La característica más relevante de Flutter es que no utiliza los componentes nativos del sistema operativo, sino que utiliza su propio motor gráfico para generar cada elemento de la interfaz.

El lenguaje que se utiliza es Dart, este se compila a código nativo mediante herramientas específicas según el sistema operativo de destino. Se distingue por su funcionalidad denominada Stateful hot Reload, que permite realizar correcciones en el código fuente para verlos al instante, sin perder el estado actual ni reiniciar la aplicación [27].

2.1.7 Rastreo de mascotas

2.1.7.1 Tecnología de seguimiento y localización:

El rastreo de mascotas se ha vuelto más accesible gracias al desarrollo de tecnologías como los dispositivos GPS [28], los microchips y los collares inteligentes. Estos dispositivos permiten a los dueños de mascotas localizar a sus animales en tiempo real a través de aplicaciones móviles o plataformas en línea. Algunos dispositivos incluso ofrecen características adicionales, como cercas virtuales que alertan al dueño cuando la mascota sale de un área predefinida.

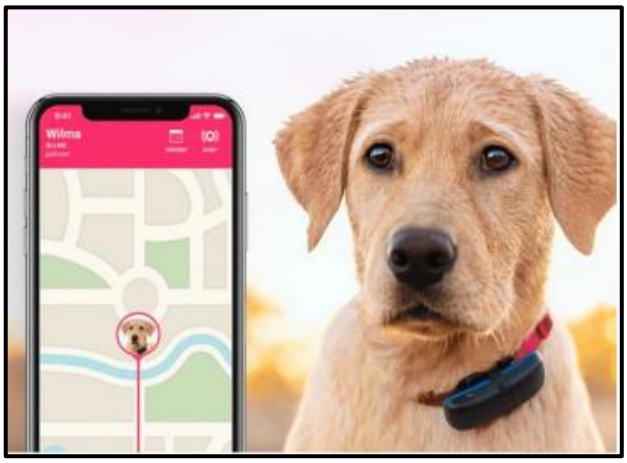


Figura 5. Referencia de Rastreo Satelital.

Fuente: [28]

2.1.7.2 Seguridad de las mascotas:

El rastreo de mascotas puede contribuir a mejorar su seguridad. Al contar con dispositivos de seguimiento, los dueños pueden encontrar rápidamente a sus mascotas en caso de que se pierdan o escapen. Esto puede reducir el riesgo de accidentes o robos de animales. Según la compañía PetHub que ofrece etiquetas de identificación, brinda algunas estadísticas con respecto a las mascotas extraviadas donde menciona que una de cada tres mascotas se extravía en algún momento de sus vidas, de los cuales, menos del 2% de gatos y solo del 15 al 20% de los perros extraviados logran regresar a su hogar [29].

CAPÍTULO III.

3.1 METODOLOGÍA.

3.1.1 Tipo de Investigación

Esta investigación es de carácter experimental y cuantitativo. Esto se debe a que su enfoque radica en la recopilación y análisis de datos numéricos obtenidos de mediciones específicas de la temperatura, frecuencia cardíaca, SpO2 y ubicación. La precisión de estas mediciones es esencial para validar la eficacia del dispositivo, por lo que se han diseñado experimentos controlados en los que se realizan múltiples mediciones en períodos definidos para cada sujeto de prueba. Este procedimiento experimental permite analizar la variabilidad de los datos y determinar la consistencia de los resultados.

El tipo de investigación adoptado es también de carácter comparativo, ya que se realiza una comparación sistemática entre dos métodos de medición: por un lado, el dispositivo de monitoreo propuesto y también los instrumentos tradicionales empleados en la práctica veterinaria.

Finalmente, el análisis comparativo de los datos obtenidos permitirá determinar la concordancia y validez del dispositivo basado en LoRaWAN como alternativa a los métodos convencionales.

3.1.2 Diseño de Investigación

Tabla 1. Etapas del Desarrollo del Prototipo.

Etapa	Descripción
1. Investigación y revisión bibliográfica	Revisión de la literatura relacionada con el monitoreo veterinario de signos vitales, rastreo de mascotas y tecnología LoRaWAN.
2. Definición de requisitos y diseño del sistema	Establecimiento de los requisitos del proyecto. Diseño de la arquitectura y elaboración de un diagrama de bloques.
3. Selección y adquisición de	Investigación y selección de los componentes electrónicos necesarios y su adquisición.

componentes	
4. Programación	Programación de microcontroladores y sensores para la obtención y procesamiento de datos. Implementación de la tecnología LoRaWAN.
5. Implementación del hardware	Montaje físico de los componentes. Conexiones y soldaduras, verificación de su funcionamiento y comunicación entre los componentes.
6. Pruebas y validación del sistema	Realización de pruebas de rendimiento de la comunicación inalámbrica entre los distintos módulos del prototipo.
7. Análisis de resultados y conclusiones	Análisis de los datos obtenidos. Evaluación de los objetivos y requisitos cumplidos. Evaluación crítica del sistema y áreas de mejora identificadas.

3.1.3 Técnicas de Recolección de Datos

3.1.3.1 Observación

Este trabajo utiliza la técnica de observación para la recopilación de información de manera directa en un entorno real, de esta forma se podrá observar:

- El uso del prototipo.
- Registro de datos en un ambiente controlado.
- La evaluación del rendimiento del equipo desarrollado en entornos clínicos.

3.1.3.2 Adquisición Instrumental Automatizada

Se utiliza también esta técnica para la recolección de datos a través de sensores. De esta manera se obtiene información de forma continua y en tiempo real, sin intervención humana directa.

3.1.4 Población de Estudio y Tamaño de Muestra

3.1.4.1 Población

Las mediciones se realizaron en intervalos de tiempo, donde se obtuvieron datos de cada dispositivo para las variables siguientes:

- Temperatura (°C)
- Frecuencia cardíaca (lpm)
- Saturación de oxígeno (%)

Es importante tener en cuenta que las variables pueden ser medidas de una manera continua en el tiempo, por esta razón se considera una población de datos infinita.

3.1.4.2 Tamaño de la muestra

En esta investigación, no se usará la estimación estadística con intervalos de confianza, en su lugar, se realizaron pruebas de hipótesis. En vista de que no se conoce la varianza de la población, la muestra se calcula con base al árbol de decisiones [30], de la siguiente manera, donde:

$n_0 \rightarrow$ Es el tamaño de muestra requerido.

$t_{\propto/2,n_p-1} \rightarrow$ Es el valor t de Student asociado al nivel de significancia.

$t_{\beta,n_p-1} \rightarrow$ Es el valor t asociado al poder estadístico deseado.

$S_p \rightarrow$ es la desviación estándar de las diferencias.

$\Delta \rightarrow$ es la diferencia mínima que desea detectar.

$$n_0 = \left[\frac{\left(t_{\propto/2,n_p-1} + t_{\beta,n_p-1}\right) s_{\mathrm{p}}}{\Delta}\right]^2$$

Temperatura:

$$n_0 = \left[\frac{(1.990 + 0.847) * 0.318}{0.1}\right]^2$$

$$n_0 = 81.3 \approx 81\ pares\ de\ datos.$$

Tras realizar iteraciones aplicando la distribución t de Student, el valor se mantuvo estable. Por lo tanto, se concluye que la muestra mínima necesaria para el análisis de temperatura es de 81 pares de datos.

Frecuencia Cardíaca:

$$n_0 = \left[\frac{(1.989 + 0.847) * 6.546}{2}\right]^2$$

$$n_0 = 86.1 \approx 86\ pares\ de\ datos.$$

Del mismo modo, luego de aplicar iteraciones con la fórmula ajustada por t de Student, el valor final se mantuvo constante. En consecuencia, la muestra mínima requerida para frecuencia cardíaca es de 86 pares de datos.

SpO2:

$$n_0 = \left[\frac{(2.131 + 0.866) * 1.424}{1}\right]^2$$

$$n_0 = 37.2 \approx 37\ pares\ de\ datos.$$

En este caso, el cálculo inicial arrojó un tamaño de muestra de 37 pares. Sin embargo, al aplicar las iteraciones con la distribución t de Student, el tamaño estimado de muestra se estabilizó en 18 mediciones tras varias iteraciones. Por tanto, la muestra mínima necesaria para el análisis del SpO_2 es de 18 pares de datos.

Para efectos del estudio, se tomó como referencia el tamaño muestral más alto obtenido entre los tres parámetros analizados. Este corresponde a la frecuencia cardíaca, con 86 mediciones. Por lo tanto, se consideró como muestra mínima general 86 mediciones por cada parámetro y por cada dispositivo.

3.1.5 Operacionalización de las variables

Tabla 2. Parámetros y Métodos de Evaluación.

VARIABLE	CONCEPTO	INDICADORES	TÉCNICAS E INSTRUMENTACIÓN
DEPENDIENTE			

Temperatura	Valor de temperatura corporal normal en canes.	Grados Celsius (°C)	Medición directa con termómetro digital clínico y medición automática mediante el sensor de temperatura integrado.
Saturación de oxígeno	Saturación de oxígeno en sangre.	Porcentaje (%)	Medición directa con equipo multiparámetros veterinario mediante sonda y medición automática a través del sensor de SpO_2 incorporado.
Frecuencia cardíaca	Ritmo de pulsaciones normales en canes.	Pulsaciones por minuto (BPM)	Medición directa con equipo multiparámetros veterinario mediante sonda y medición automática mediante el sensor de frecuencia cardíaca integrado.
Ubicación (GPS)	Determinación de la posición geográfica en tiempo real para rastreo.	Coordenadas geográficas (Lat/Long)	Medición directa con equipo GPS certificado y medición automática a través del módulo GPS
INDEPENDIENTE			
Tipo de Dispositivo	Implementado y tradicional	Tipo	Observación directa

3.1.6 Primera fase

Esta fase se inicia con la revisión de fundamentos teóricos tanto de la parte técnica como veterinaria. Aquí se comparan las tecnologías de comunicación inalámbrica y los datos

fisiológicos estándar de los canes, ya que esta es la base para la implementación del prototipo.

La selección de la tecnología de comunicación inalámbrica es un aspecto clave en el diseño de sistemas eficientes para el monitoreo veterinario. Por este motivo, se realizó una comparación entre diversas tecnologías utilizadas en aplicaciones inalámbricas, específicamente Wi-Fi 6, Bluetooth 5.0, ZigBee y LoRa. En la Tabla 3 se resumen sus principales características, abordando aspectos como consumo de energía, alcance, velocidad de transmisión y otras propiedades relevantes para el monitoreo remoto y el uso en dispositivos portátiles

Tabla 3. Comparativa de Tecnologías de Comunicación Inalámbricas.

	LoRa	**Bluetooth 5.0**	**Wi-Fi 6**	**Zigbee**
Banda de Frecuencia	863-870 MHz, 902-928 MHz, 433 MHz.	2.4 GHz	2.4 GHz, 5GHz	2.4GHz, 868 MHz, 915 MHz
Velocidad de transmisión	50 kbps	2 Mbps	10 Gbps	250 kbps
Consumo de energía	TX: 120-150 mA; inactivo profundo: <1µA	TX: 3-10 mA; inactivo: 1-50 µA	TX: 300-500 mA; inactivo: 100-200 mA	TX: 40-50 mA; inactivo 1A
Cobertura	Hasta 20 Km rural y 5Km urbano	Hasta 240 m exterior y 60 m interior	Hasta 250 m exterior y 70 m interior	Hasta 1600 m exterior y 90 m interior
Potencia de transmisión	20 dBm, 14 dBm, 10 dBm	20-30 dBm	20-30 dBm	0-20 dBm
Ventajas principales	Muy bajo consumo, excelente alcance, ideal para zonas rurales y con	Bajo consumo, fácil integración, económico	Alta velocidad y gran capacidad de ancho de banda.	Bajo consumo y buena escalabilidad.

	edificios

Fuente: [31]

Luego del análisis comparativo realizado, se seleccionó la tecnología LoRa como la opción más adecuada para el desarrollo del sistema de monitoreo veterinario portátil, debido a sus destacadas características técnicas. Entre estas se encuentran su gran alcance de transmisión, su excelente eficiencia energética y su capacidad para operar eficazmente en entornos rurales o urbanos. Estas ventajas son especialmente importantes en sistemas que requieren autonomía prolongada y conectividad confiable. Además, a diferencia de otras alternativas inalámbricas, LoRa posibilita la transmisión continua de datos fisiológicos sin necesidad de una conexión eléctrica constante, haciéndola particularmente apropiada para dispositivos alimentados mediante baterías.

De este modo, su estructura de red permite enviar todos los datos recolectados hacia un único receptor o gateway, facilitando así el monitoreo remoto, permitiendo observar en tiempo real los signos vitales de las mascotas a través de la aplicación móvil.

En el ámbito del monitoreo fisiológico, resulta imprescindible disponer de parámetros de referencia que permitan una adecuada interpretación de los datos a recopilar. En este sentido, la Tabla 4 presentada a continuación expone los rangos fisiológicos considerados normales en perros adultos, los cuales sirven como fundamento para la configuración de umbrales de alerta, así como para evaluar la precisión y fiabilidad de los sensores incorporados en el dispositivo desarrollado.

Tabla 4. Valores fisiológicos normales en perros.

Parámetro	Rango normal	Unidad
Temperatura	37.38 - 39.17	°C
Frecuencia Cardíaca	60-150	latidos/minuto
SpO2	96-98	%

Fuente: [14]

3.1.7 Segunda fase

La segunda fase se enfocó en el diseño, implementación y validación del sistema de monitoreo portátil, integrando sensores biomédicos, módulos de comunicación y la estructura física del dispositivo. Esta etapa implicó seleccionar los componentes electrónicos más adecuados, programar el microcontrolador principal, y adaptar el diseño a un chaleco ajustable para asegurar la correcta ubicación de los sensores en el cuerpo del animal.

Los componentes seleccionados, como los sensores tanto de temperatura, frecuencia cardíaca y SpO2 se configuraron para evaluar su precisión en la obtención de datos y su compatibilidad con LoRaWAN. Además, se programó el firmware del transmisor para mejorar la comunicación y envío de datos con el gateway LoRaWAN que posteriormente se observa en la aplicación. Por último, se comprueba el dispositivo y los datos medidos en condiciones reales, analizando eficiencia y autonomía.

En el diagrama de bloques de la Figura 6, se muestra la arquitectura de hardware de la implementación propuesta. Se comienza con la alimentación que contiene una batería que pasa por un filtrado y protección para seguridad de todos los componentes, también tiene un divisor y un regulador que nos ayudan a controlar el nivel de voltaje que reciben los componentes y esto permite que el suministro de energía sea estable. Cabe recalcar que el bloque de alimentación es el mismo tanto para el transmisor como para el receptor. Posteriormente, en el bloque de sensores se encuentra el acelerómetro, oxímetro y de temperatura, estos conectados a la ESP32, que funciona como emisor principal que también se conecta con el módulo GPS. La información que procesa la ESP32 se muestra localmente por medio de una pantalla OLED e indicadores LED, esto facilita el acceso a los datos de manera inmediata. A través del módulo LoRa, los datos que posteriormente fueron recopilados se mandan a un receptor que cuenta con otra ESP32, este a su vez, se enlaza a la nube usando el servicio Firebase para almacenar la información y que se encuentre segura y que sea accesible en tiempo real. Finalmente, se presentan los datos al usuario mediante una aplicación móvil, de esta manera se facilita el monitoreo constante de la mascota y la toma de decisiones según la información obtenida.

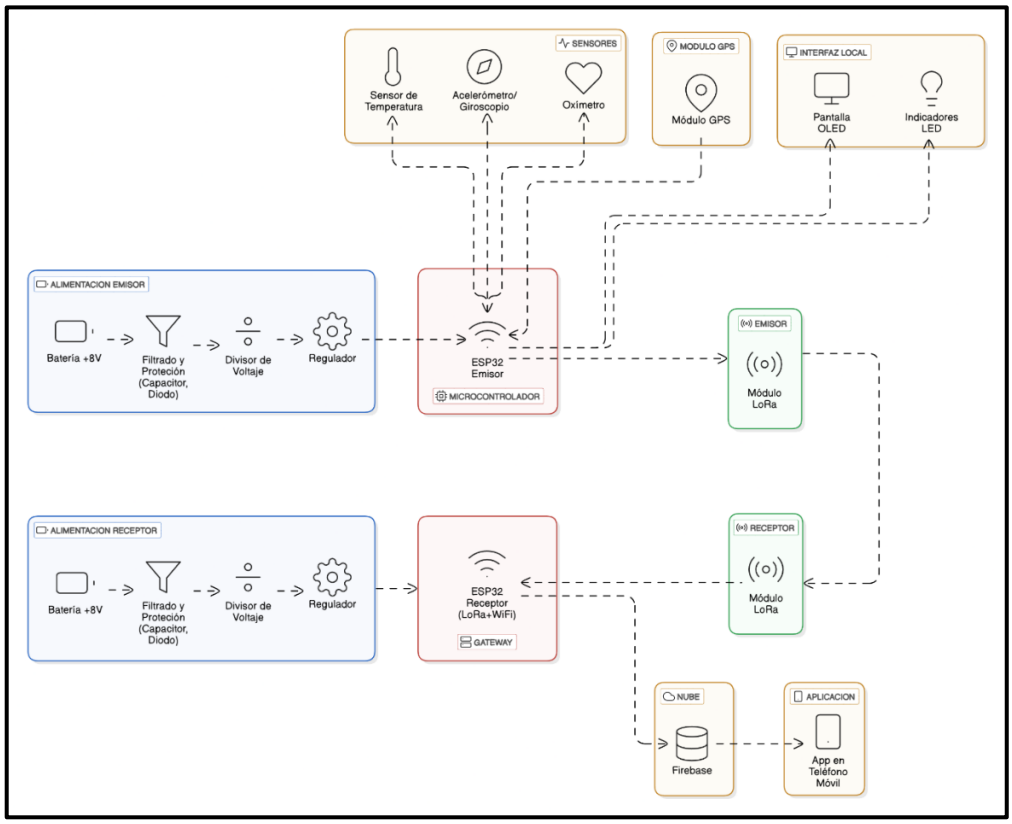


Figura 6. Diagrama de bloques del dispositivo.

3.1.8 Diseño del Esquemático Electrónico

- **Transmisor**

En la Figura 7 se muestra el diagrama esquemático del circuito, donde se observan los componentes y sus conexiones correspondientes con etiquetas.

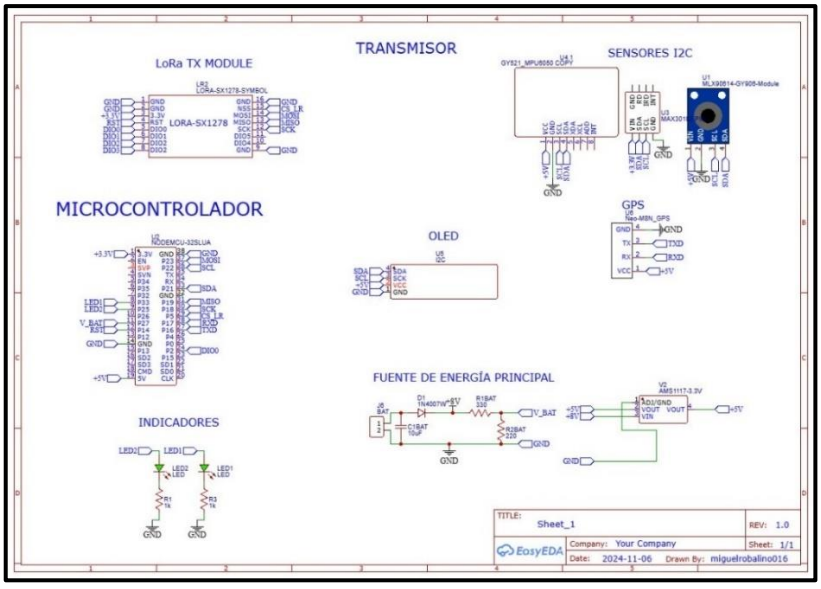


Figura 7. Circuito General del Transmisor en el software EasyEDA.

Además, se presenta el diseño PCB del circuito en la Figura 8, donde se visualizan las conexiones que se realizaron entre componentes.

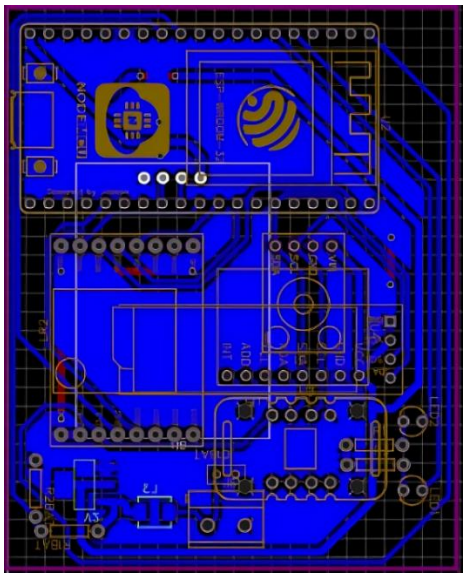


Figura 8. Diseño PCB del circuito Transmisor.

- **Receptor**

De la misma forma se muestra en la Figura 9 el diagrama esquemático del circuito, este módulo tiene menos componentes que transmisor.

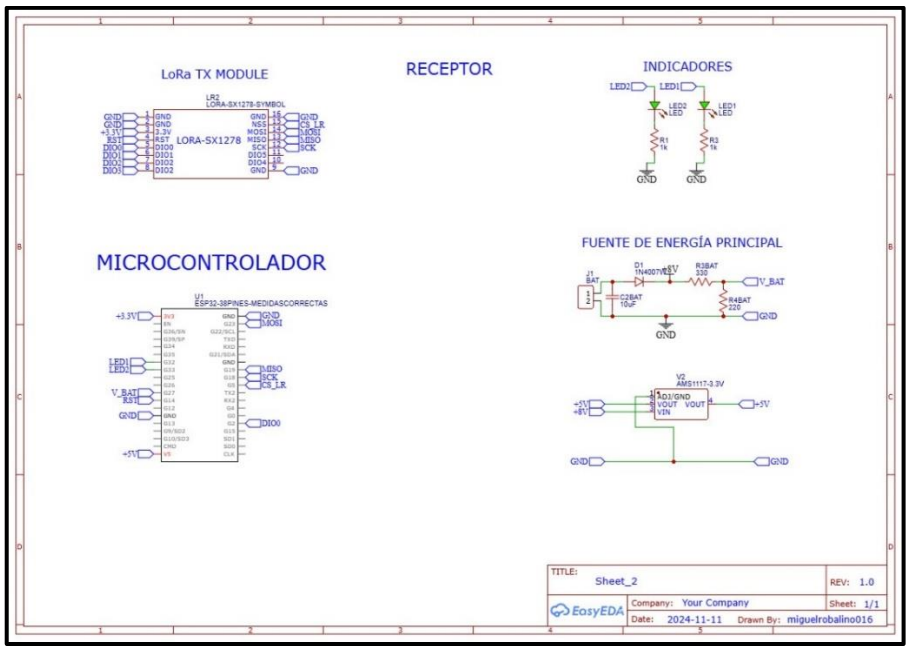


Figura 9. Circuito General del Receptor en el software EasyEDA.

Por último, en la Figura 10 se muestra el diseño PCB del receptor, ahí se pueden apreciar las conexiones de los componentes.

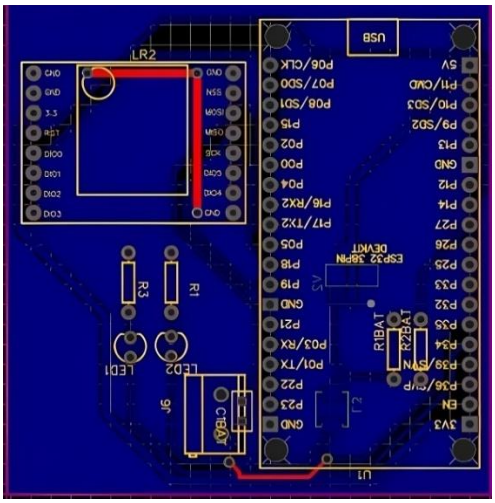


Figura 10. Diseño PCB del circuito Receptor.

3.1.9 Representación y Modelado 3D del Prototipo

En el modelado del prototipo de la Figura 11 se muestra la carcasa para el circuito, la cual tiene ciertos orificios en lugares estratégicos para la entrada de la alimentación o la salida de los sensores. Además, cuenta con unas pequeñas carcasas de los sensores para su protección, ya que van a estar en contacto con las mascotas.

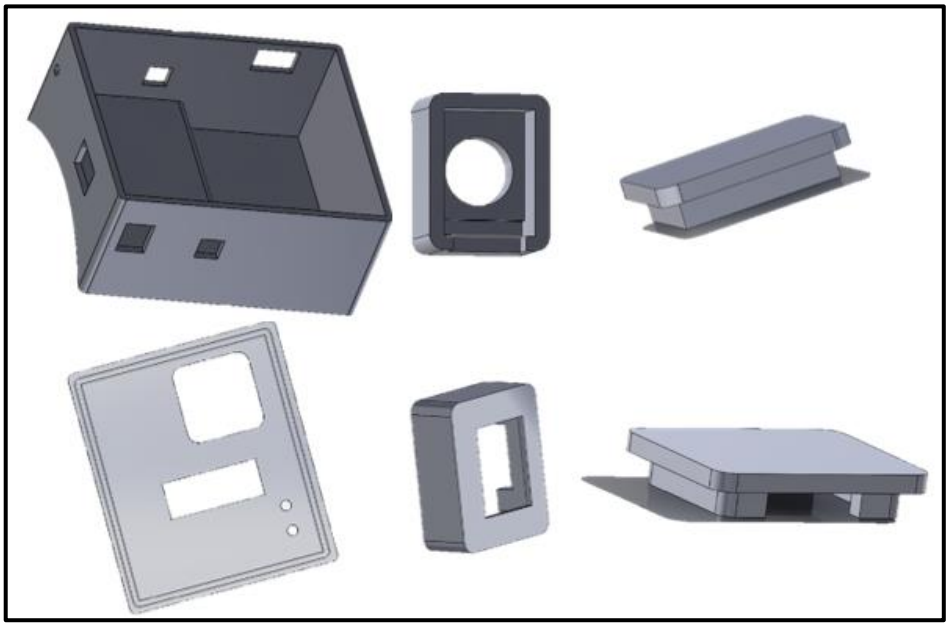


Figura 11. Partes del dispositivo y Gateway.

Como se observa en la Figura 12, se realizó el ensamble del circuito en la carcasa para que se pudiera previsualizar cómo iba a quedar el dispositivo y los sensores. Crear de esta manera las carcasas permite que en mantenimientos futuros sea más sencillo el acceso a los componentes.

Figura 12. Ensamble del dispositivo Transmisor y Gateway.

3.1.10 Transmisor

3.1.10.1 Diseño del bloque de alimentación

El bloque de la alimentación para el circuito, tal como se muestra en la Figura 13 cuenta con la fuente de alimentación, en este caso, de 7.4 V. La siguiente fase es la de filtrado, protección y división del voltaje, en donde se coloca un capacitor de 10 uF que filtra la señal, para la eliminación de ruidos y picos donde se obtiene un voltaje más limpio. Asimismo, se incorpora un diodo 1n4007 que sirve como protector y solo permite el paso de la corriente únicamente cuando se polariza positivamente, evitando así cortocircuitos. Además, se añadió un divisor de voltaje conformado por una resistencia de 330 Ω y 220 Ω, el cual regula el voltaje de 7.4 V a aproximadamente 3.3 V, valor que se utiliza para medir el nivel de la batería a través del ESP32. Seguidamente, el voltaje filtrado se dirige al

circuito de regulación, la cual consta del regulador AMS1117 5 V que transforma los 7.4 V que va a la entrada en una salida de 5 V. En este punto se verifica que la salida es estable; de no ser el caso, se realizan ajustes o se corrigen las conexiones y componentes implicados. Ya que se encuentre estable el voltaje, este se distribuye a los módulos y sensores.

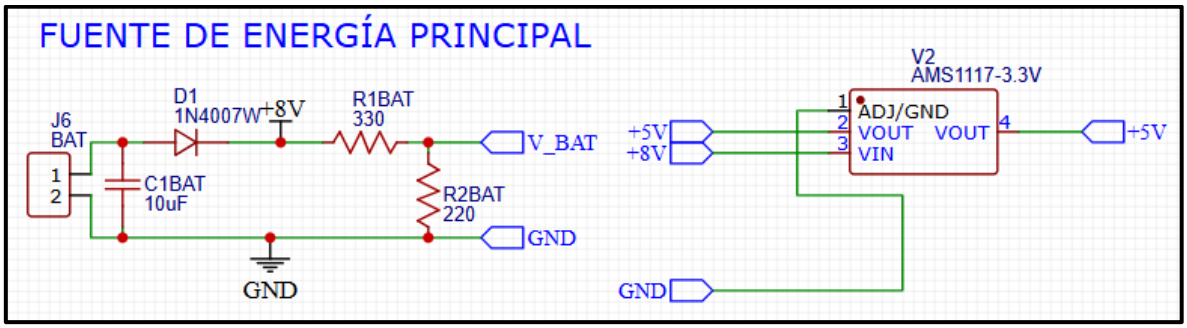


Figura 13. Bloque de alimentación del transmisor

Finalmente, incluye un indicador de estado de la batería, donde se encuentra un LED que indica si el nivel de la batería es bajo, como se puede observar en la Figura 14.

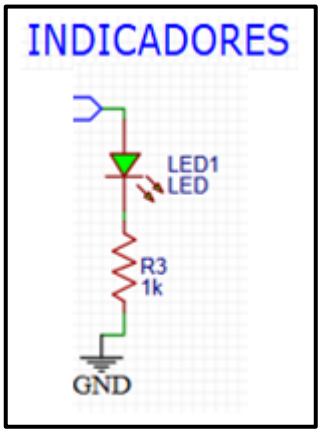


Figura 14. Indicador LED de nivel de batería.

A continuación, en la Figura 15 se presenta este bloque ya ensamblado, se puede mirar los componentes encargados de suministrar y regular la energía del sistema.

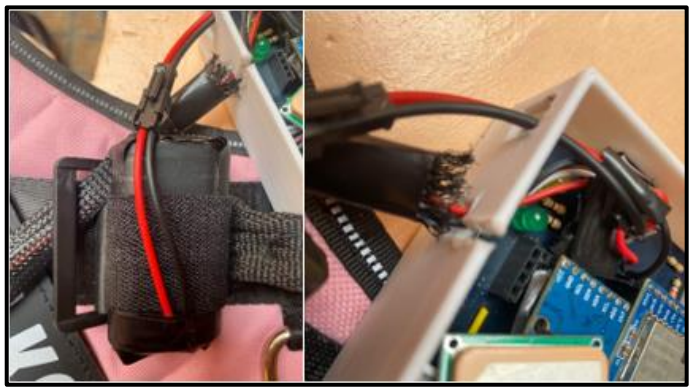


Figura 15. Bloque de alimentación ensamblado.

3.1.10.2 Diseño del bloque de sensores, GPS y OLED

En el bloque de sensores del circuito que se observa en la Figura 16 inicia con la incorporación de los módulos a la ESP32 por medio de los buses de comunicación UART e I2C, esto permite la obtención de datos en tiempo real. Primero se conecta el sensor de temperatura infrarrojo MLX90614 a través del protocolo I2C, garantizando una configuración correcta de sus líneas de datos (SDA/SCL) y la alimentación de 5 V. Conjuntamente, el sensor MAX30102, que se encarga de medir la saturación de oxígeno en la sangre (SpO2) y el ritmo cardíaco, se conecta a la misma interfaz de comunicación, con la configuración incluida de sus pines adicionales para el control del LED IR y la detección de señal.

Por otro lado, se añade el acelerómetro y giroscopio MPU6050, este también opera bajo el protocolo I2C y dispone de un pin de interrupción que posibilita la detección de cambios en la orientación o de movimiento. Este módulo necesita una calibración inicial para que se garanticen unas mediciones precisas. Asimismo, se incorpora el módulo GPS Neo-M8N, el cual se comunica UART utilizando los pines TX y RX de la ESP32.

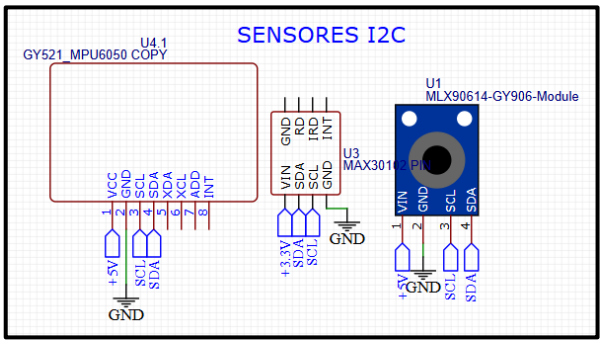


Figura 16. Bloque de sensores del circuito.

A su vez, se conecta la OLED por medio del mismo bus I2C, utilizando los pines SCL y SDA de la ESP32, como se muestra en la Figura 17. El módulo GPS brinda una visualización en tiempo real de los datos recolectados por los sensores, de esta manera aseguramos y comprobamos que el dispositivo funcione de manera local. Por último, se verifica que la resolución del OLED y la configuración del software sean compatibles y que se vayan actualizando los datos en la pantalla de manera correcta.

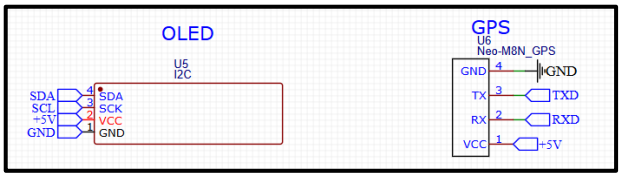


Figura 17. La OLED y GPS con sus respectivas conexiones.

En la siguiente Figura 18 se observa imágenes del dispositivo, donde se pueden ver los sensores, el módulo LoRa y la OLED.

Figura 18. Bloque de sensores ensamblado.

3.1.10.3 Diseño del bloque de procesamiento

Este bloque es el núcleo del sistema del transmisor, el cual se encarga de gestionar, procesar y empaquetar los datos obtenidos de los sensores, garantizando que la información enviada a través de la red LoRa sea precisa. Se organiza en las siguientes etapas:

- **Inicialización y configuración**

  Al encender el dispositivo, el ESP32 inicia la configuración de ADC, temporizadores y buffers, estableciendo además la comunicación serial.

- **Adquisición y validación de Datos**

  El microcontrolador se conecta a los sensores de temperatura, acelerómetro/giroscopio, oxímetro y GPS mediante I2C y UART, recibiendo sus mediciones y validándolas para asegurar que estén dentro de los rangos esperados; en caso contrario, se repite la lectura.

- **Procesamiento y filtrado**

  Una vez validados, los datos se filtran mediante algoritmos que eliminan valores atípicos y reducen el ruido, garantizando que las mediciones reflejen de forma precisa el estado de los signos vitales.

- **Integración y empaquetado**

  Los datos filtrados se consolidan en un único paquete de información estructurada, que integra temperatura, frecuencia cardíaca, SpO_2, datos del acelerómetro/giroscopio y ubicación GPS.

- **Registro y actualización de indicadores**

  El ESP32 mantiene registros internos de las mediciones y actualiza indicadores visuales, como la pantalla OLED y los LED.

- **Preparación para la transmisión**

  Finalmente, el paquete de datos procesados se envía al módulo LoRa, que lo transmite al gateway, garantizando la comunicación.

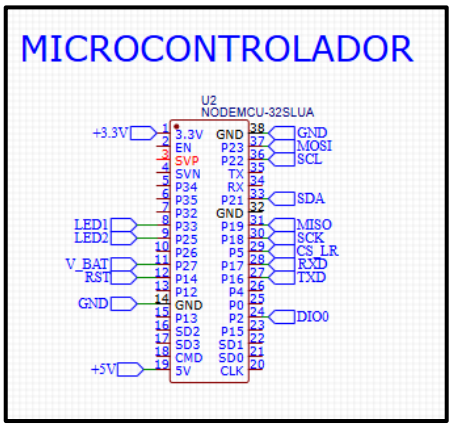


Figura 19. Bloque de procesamiento del circuito.

En la siguiente Figura 20 se muestra la ESP32 como microcontrolador principal y su conexión con los sensores y demás módulos del sistema**.**

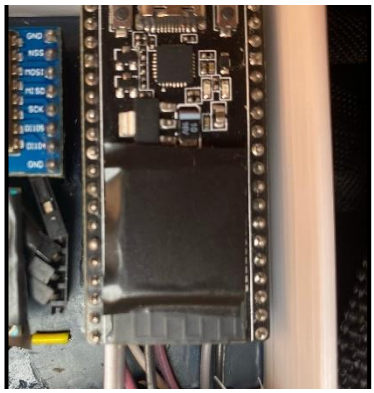


Figura 20. ESP32 y sus respectivas conexiones.

3.1.10.4 Diseño del bloque de comunicación

El bloque de comunicación se encarga de traducir los datos digitales procesados en una señal de radio a través del módulo LoRa, permitiendo que la información se transmita hacia el gateway para su posterior procesamiento y almacenamiento en la nube. Una vez que el ESP32 ha integrado, validado y empaquetado la información proveniente de los sensores, esta se envía al módulo LoRa. El microcontrolador configura en el módulo LoRa la frecuencia de operación de 433 MHz, asegurando que la señal se emita de forma óptima.

El proceso inicia con la preparación del paquete de datos. Una vez que el mismo esté listo, el ESP32 lo envía al módulo LoRa, el cual modula la información y la transmite de forma inalámbrica.

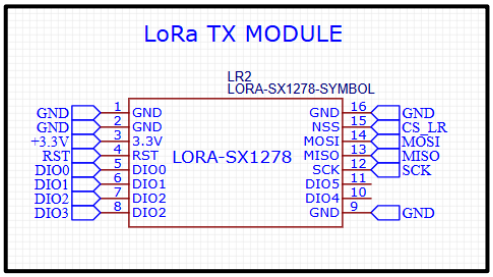


Figura 21. Bloque de comunicación del circuito

Se muestra la Figura 22 donde se aprecia la conexión del módulo LoRa, el cual nos proporciona el envío de datos.

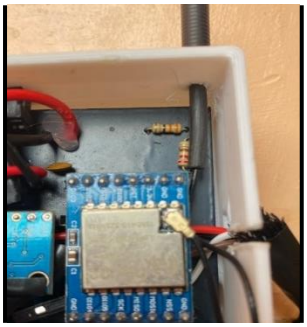


Figura 22. Módulo LoRa utilizado para el envío de datos.

3.1.11 Receptor

3.1.11.1 Diseño del bloque de alimentación

El bloque de alimentación del receptor se emplea de la misma manera que en el transmisor descrito en la sección 3.1.10.1, dado que ambos utilizan la misma configuración. En la siguiente Figura 23 se muestra la conexión de los componentes que conforman este bloque.

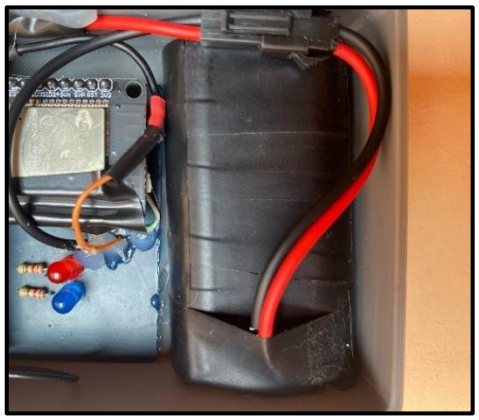


Figura 23. Bloque de alimentación del receptor.

3.1.11.2 Diseño del bloque de procesamiento

Para este diseño en el receptor se realiza el mismo procedimiento que en el transmisor que se encuentra en la sección 3.1.10.3, pero teniendo en cuenta que ya no existen las conexiones del SCL, SDA, TXD y RXD, que son netamente para los sensores. Podemos observar en la Figura 24, la ESP32 ya colocada en su respectivo módulo.

Figura 24. ESP32 y sus respectivas conexiones.

3.1.11.3 Diseño del bloque de comunicación

Para el bloque de Comunicación en el Receptor se implementa de manera similar al del transmisor descrito en la sección 3.1.10.4, ya que ambos cuentan con la misma configuración. En la Figura 25 se muestra el módulo LoRa conectado correctamente.

Figura 25. Módulo LoRa utilizado para la recepción de datos.

3.1.12 Implementación de la base de datos

Este apartado integra la gestión y visualización remota de los datos recolectados por el dispositivo, combinando la capacidad de almacenamiento en la nube con la interfaz de la aplicación móvil. En primer lugar, el gateway recibe y procesa el paquete de datos transmitido por el dispositivo transmisor a través de la red LoRaWAN, y luego mediante la conexión WiFi lo reenvía a Firebase, una plataforma que permite la sincronización en tiempo real de la información y su almacenamiento seguro en la nube. Firebase se configura para recibir datos estructurados y se encarga de mantener actualizados los registros, lo que es crucial para el monitoreo continuo de los signos vitales y la ubicación de las mascotas.

3.1.13 Desarrollo de la aplicación móvil

El desarrollo de la aplicación móvil se realiza en Flutter, utilizando Android Studio como entorno de desarrollo integrado. La aplicación se divide en dos secciones como se muestra en la Figura 26. La parte superior de la pantalla se destina a la visualización de una interfaz compuesta por tarjetas informativas de los datos que incluyen el nivel de batería del dispositivo, la temperatura corporal, el tipo de movimiento, la frecuencia cardíaca y la saturación de oxígeno (SpO2).

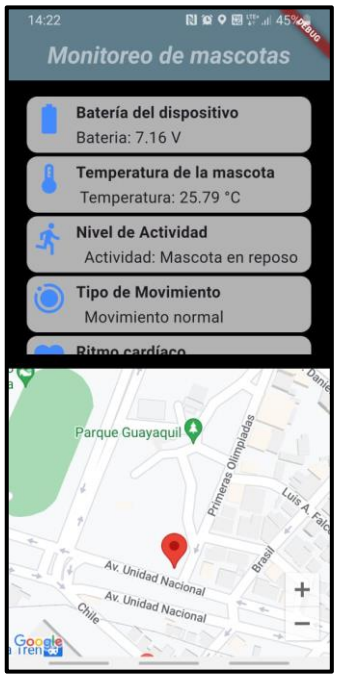


Figura 26. Aplicación final en Android Studio

En la parte inferior de la pantalla se incorpora un mapa utilizando la API de Google Maps, por la integración con Google Cloud. La ubicación de la mascota se representa con un marcador rojo que se actualiza constantemente a partir de los datos enviados por el transmisor.

Para finalizar en Android Studio, se configuran las dependencias necesarias para integrar Firebase y Google Maps, lo que permite a la aplicación conectarse a la nube, obtener actualizaciones automáticas y representarlas en la interfaz de usuario. Los datos se recuperan mediante consultas en tiempo real a Firebase, y la sincronización se gestiona mediante flujos de datos que actualizan dinámicamente los widgets en Flutter. Este proceso asegura que la información mostrada en la aplicación sea actual y precisa, facilitando la toma de decisiones por parte del usuario.

3.1.14 Tercera fase

Para validar el funcionamiento del sistema desarrollado, se procedió a realizar pruebas controladas en un entorno clínico veterinario. El objetivo principal fue verificar la estabilidad del chaleco, la precisión de los sensores de temperatura, SpO_2, frecuencia cardíaca y la correcta transmisión de datos hacia la aplicación móvil mediante el gateway LoRa.

Figura 27. Colocación de la malla ajustable.

El dispositivo está compuesto por dos elementos fundamentales: una malla interna ajustable como se observa en la Figura 27 y un chaleco externo diseñado para adaptarse al cuerpo del animal. Su estructura permite envolver de forma segura y ergonómica el pecho, las patas delanteras, el abdomen y el lomo del perro. Esta malla cumple una función crítica, ya que permite mantener en posición los sensores biomédicos evitando desplazamientos durante el movimiento.

Figura 28. Colocación del chaleco.

Posteriormente, se coloca el chaleco exterior como se muestra en la Figura 28, el cual es también regulable y contiene un compartimento para alojar el transmisor LoRa. Este componente cuenta con el cableado de los sensores que se extiende desde la parte superior del dispositivo y desciende por ambos laterales. En el costado derecho se sitúa el sensor de temperatura, mientras que en el izquierdo se dispone el sensor combinado de frecuencia cardíaca y saturación de oxígeno.

Figura 29. Perro equipado correctamente con el prototipo.

Los sensores deben ubicarse en zonas estratégicas para garantizar lecturas confiables. En el caso de la temperatura corporal, se posiciona el sensor a la altura del pecho o axila del animal, ya que esta región presenta una mayor estabilidad térmica y cercanía a vasos sanguíneos superficiales, lo cual la convierte en un punto anatómicamente óptimo para mediciones no invasivas. Para lograr esta fijación, los sensores se introducen por debajo de la malla interna, que actúa como soporte de presión ligera, manteniéndolos en su sitio sin causar incomodidad.

Figura 30. Funcionamiento del prototipo de monitoreo.

Una vez colocado el sistema completo, se procede al encendido del dispositivo transmisor que se aprecia en la Figura 30, seguido por la activación del gateway receptor, el cual envía los datos capturados hacia la base de datos en la nube mediante Firebase. Esta información puede visualizarse en tiempo real en la aplicación móvil desarrollada como parte de este proyecto. La aplicación muestra variables como temperatura corporal, frecuencia cardíaca, nivel de actividad, movimiento detectado, saturación de oxígeno, y estado del GPS.

CAPÍTULO IV.

4.1 RESULTADOS Y DISCUSIÓN

En el capítulo siguiente se realiza la presentación y análisis de los resultados conseguidos con el equipo desarrollado de monitoreo durante las pruebas efectuadas. Se describen los datos obtenidos por los sensores de frecuencia cardíaca, SpO2 y temperatura, así como la ubicación obtenida por el GPS.

4.1.1 Prueba de normalidad, análisis comparativo y resultados de la temperatura, frecuencia cardíaca, SpO2 entre el prototipo y el dispositivo veterinario.

Para proceder al análisis de resultados, se llevó a cabo una prueba de normalidad para determinar si los datos medidos a partir del prototipo y del dispositivo veterinario, corresponden a una distribución normal.

Ho: Los datos medidos tienen una distribución normal.
Hi: Los datos medidos no tienen una distribución normal.

Tabla 5. P-Valor para los datos del Prototipo y Dispositivo Veterinario.

	P-Valor	
Variables	**Prototipo**	**Dispositivo veterinario**
Temperatura	0.000	0.000
Frecuencia Cardíaca	0.000	0.000
SpO2	0.000	0.000

Según se muestra en la Tabla 5, el p-valor obtenido para las variables de temperatura, frecuencia cardíaca, y SpO2 es menor que 0.05 en ambos dispositivos. Esto indica que los datos en estas variables no se distribuyen normalmente. Por tal motivo, no se acepta la hipótesis nula y la comparación de medias se ejecuta aplicando pruebas no paramétricas como la prueba de Wilcoxon para muestras relacionadas.

Prueba de Wilcoxon para muestras relacionadas en temperatura.

Ho: La mediana de diferencias entre la temperatura del prototipo y la del dispositivo veterinario es igual a 0.

Hi: La mediana de diferencias entre la temperatura del prototipo y la del dispositivo veterinario no es igual a 0.

Tabla 6. Prueba de Wilcoxon para muestras relacionadas. P-Valor de temperatura.

Variable	Mediana	P-Valor
Temperatura Prototipo	38.66	0.612
Temperatura dispositivo veterinario	38.61	

La Tabla 6 presenta los resultados de la comparación entre los datos del prototipo y del dispositivo veterinario. Dado que el p-valor es mayor a 0.05, se acepta la hipótesis nula, lo que significa que no hay diferencias significativas entre las mediciones realizadas por ambos dispositivos.

Prueba de Wilcoxon para muestras relacionadas en frecuencia cardíaca.

Ho: La mediana de diferencias entre la frecuencia cardíaca del prototipo y la del dispositivo veterinario es igual a 0.
Hi: La mediana de diferencias entre la frecuencia cardíaca del prototipo y la del dispositivo veterinario no es igual a 0.

Tabla 7. Prueba de Wilcoxon para muestras relacionadas. P-Valor de frecuencia cardíaca.

Variable	Mediana	P-Valor
Frec. Cardíaca Prototipo	89	0.493
Frec. Cardíaca dispositivo veterinario	88	

Asimismo, como en la tabla anterior, en la Tabla 7, se compara los datos y se muestra que el p-valor es mayor a 0.05, de esta manera, se acepta la hipótesis nula, dado que no hay diferencias considerables entre las mediciones de los dos dispositivos.

Prueba de Wilcoxon para muestras relacionadas en SpO2.

Ho: La mediana de diferencias entre el SpO2 del prototipo y la del dispositivo veterinario es igual a 0.
Hi: La mediana de diferencias entre el SpO2 del prototipo y la del dispositivo veterinario no es igual a 0.

Tabla 8. Prueba de Wilcoxon para muestras relacionadas. P-Valor de SpO2.

Variable	Mediana	P-Valor
SpO2 Prototipo	96	0.109
SpO2 dispositivo veterinario	96	

Por último, tenemos la comparación de los datos de SpO2 en la Tabla 8, donde se observa que el p-valor también es mayor a 0.05, y se acepta la hipótesis nula, sabiendo que no existen diferencias significativas entre mediciones.

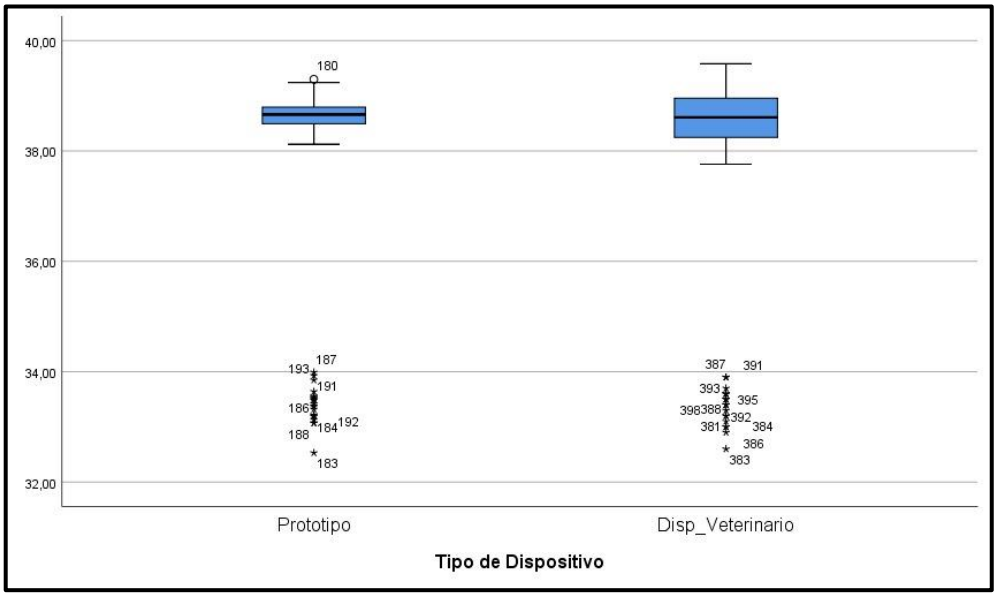


Figura 31. Diagrama de caja para la temperatura registrada por el prototipo y el dispositivo veterinario en °C.

Se observa en la Figura 31 que los dos dispositivos registran valores de temperatura con comportamientos notablemente parecidos. No obstante, aparecen algunos valores atípicos, esto se debe a ciertos motivos, uno de ellos es por los movimientos del animal durante la

medición y variaciones momentáneas por estrés o agitación. Otro motivo muy importante es que en el estudio realizado contó con un perro con un estado de salud complicado que se analiza en la sección 4.1.2. Por tal motivo, ambos dispositivos entregaron valores por debajo de la media.

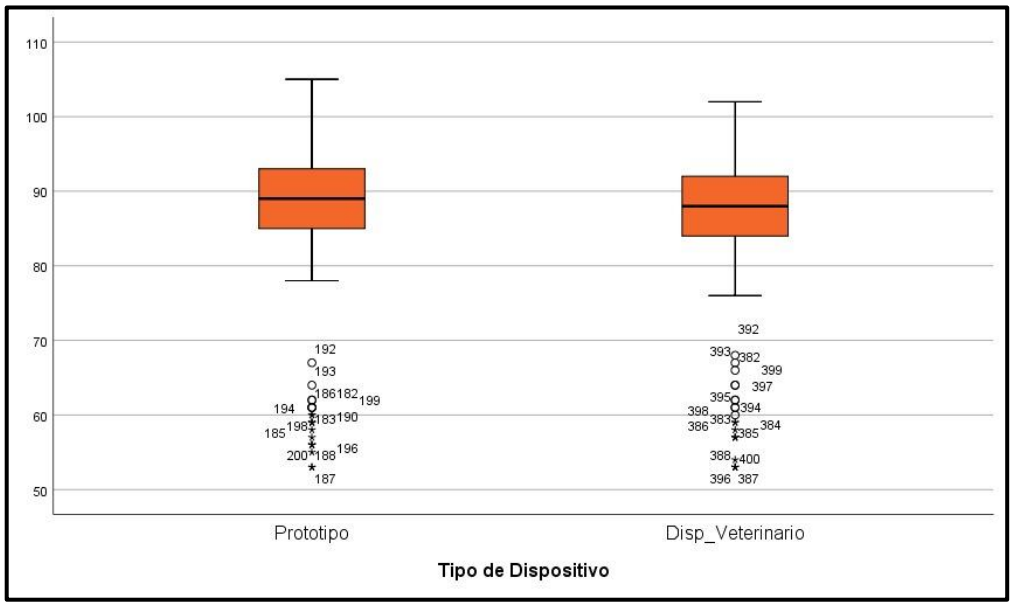


Figura 32. Diagrama de caja para la frecuencia cardíaca registrada por el prototipo y el dispositivo veterinario (lpm).

En la Figura 32 y Figura 33 se presentan valores de frecuencia cardíaca y SpO2 de ambos dispositivos, estos datos tienen comportamientos bastante similares como en la figura anterior. Además, aquí también aparecen algunos valores atípicos, y se debe al mismo motivo hablado anteriormente.

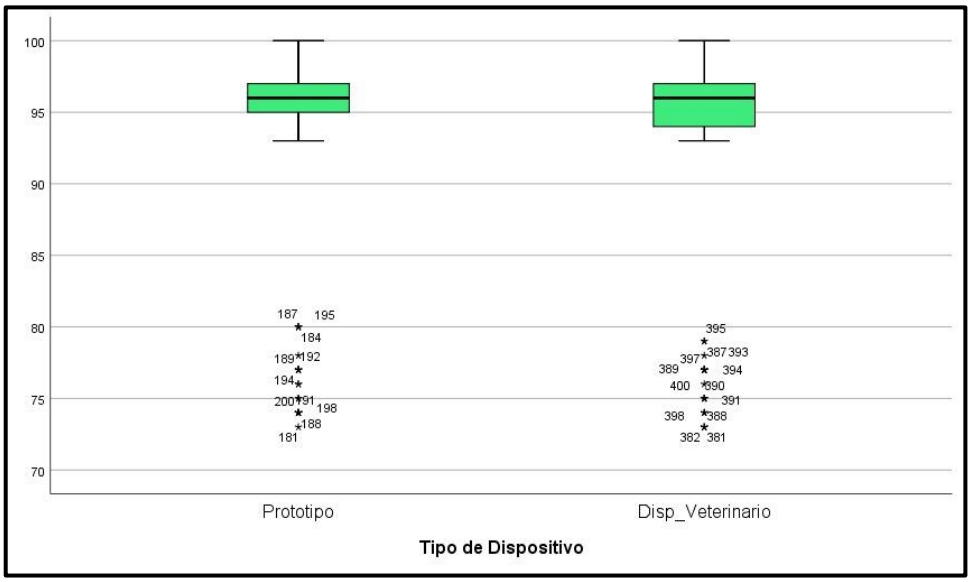


Figura 33. Diagrama de caja para el SpO2 registrado por el prototipo y el dispositivo veterinario.

4.1.2 Prueba de normalidad, análisis comparativo y resultados de la temperatura y frecuencia cardíaca en perros sanos y enfermos.

Adicionalmente, se realizó el estudio de la distribución de la temperatura y frecuencia cardíaca. Con la finalidad de entender el comportamiento de ambas variables en perros sanos y enfermos.

Ho: La temperatura y la frecuencia cardíaca en perros sanos sigue una distribución normal.
Hi: La temperatura y la frecuencia cardíaca en perros sanos no sigue una distribución normal.

Ho: La temperatura y la frecuencia cardíaca en el perro enfermo sigue una distribución normal.
Hi: La temperatura y la frecuencia cardíaca en el perro enfermo no sigue una distribución normal.

Se realizó la prueba de Kolmogorov para los perros sanos y Shapiro-Wilk para el perro enfermo. En la Tabla 9 los resultados mostraron que los p-valores son mayores a 0.05, por ende, se acepta la hipótesis nula en ambos casos.

Tabla 9. P-valor de los datos de los perros sanos y del perro enfermo.

	P-Valor	
Variables	**Perros Sanos**	**Perro Enfermo**
Temperatura	0.200	0.478
Frecuencia Cardíaca	0.055	0.807

Prueba T para muestras independientes

Ho: No existe diferencia significativa entre las medias de la temperatura y Frecuencia Cardíaca de los perros sanos y el perro enfermo.

Hi: Sí existe diferencia significativa entre las medias de la temperatura y Frecuencia Cardíaca de los perros sanos y el perro enfermo.

Tabla 10. Prueba T para muestras independientes.

Variable	Media	P-Valor
Temperatura Perros Sanos	38.69	0.000
Temperatura Perro Enfermo	33.39	
Frecuencia Cardíaca Perros Sanos	89.49	0.000
Frecuencia Cardíaca Perro Enfermo	58.95	

En la Tabla 10 se presentan los resultados de la comparación entre los perros sanos y el enfermo. El p-valor es inferior a 0.05 y las medias son significativamente diferentes, lo que indica que sí existe una diferencia estadísticamente relevante entre las mediciones. Esto quiere decir que, tanto la temperatura como la frecuencia cardíaca tienen un impacto considerable en el estado de salud del can. Si estos parámetros se encuentran fuera de los rangos fisiológicos normales, podría tratarse de un indicio de alguna alteración o problema en su condición.

En la Figura 34 y Figura 35 se presentan los diagramas de caja que corresponden a las variables de la temperatura y frecuencia cardíaca en función del estado de salud de los perros. Se evidencia que los perros sanos presentan valores más altos tanto en temperatura como en frecuencia cardíaca, en comparación con el perro enfermo. Esto nos indica que el perro enfermo presenta bradicardia e hipotermia. Estas condiciones reflejan un estado clínico comprometido.

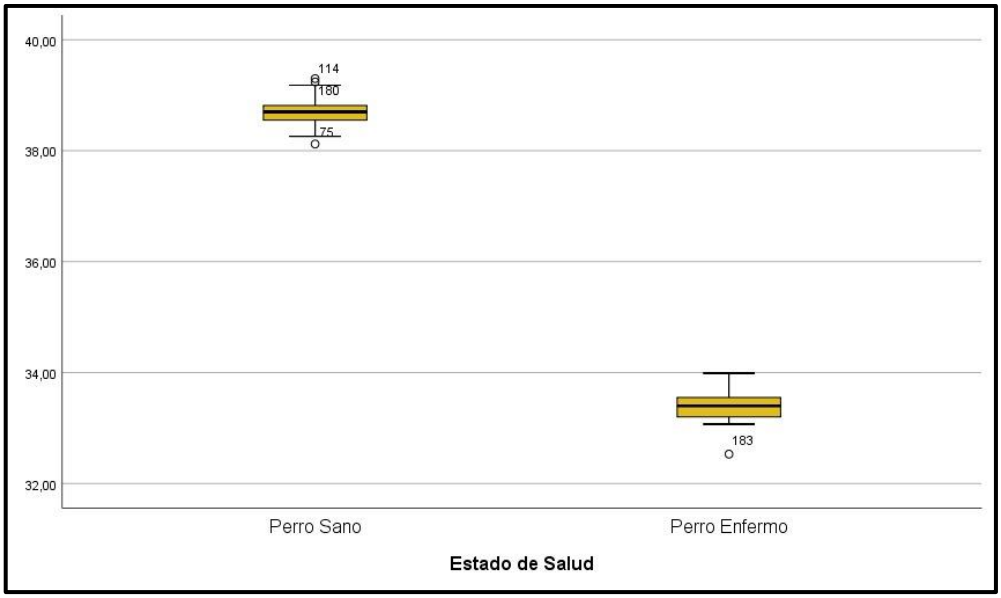


Figura 34. Diagrama de caja para la temperatura registrada por los perros sanos y enfermo en °C.

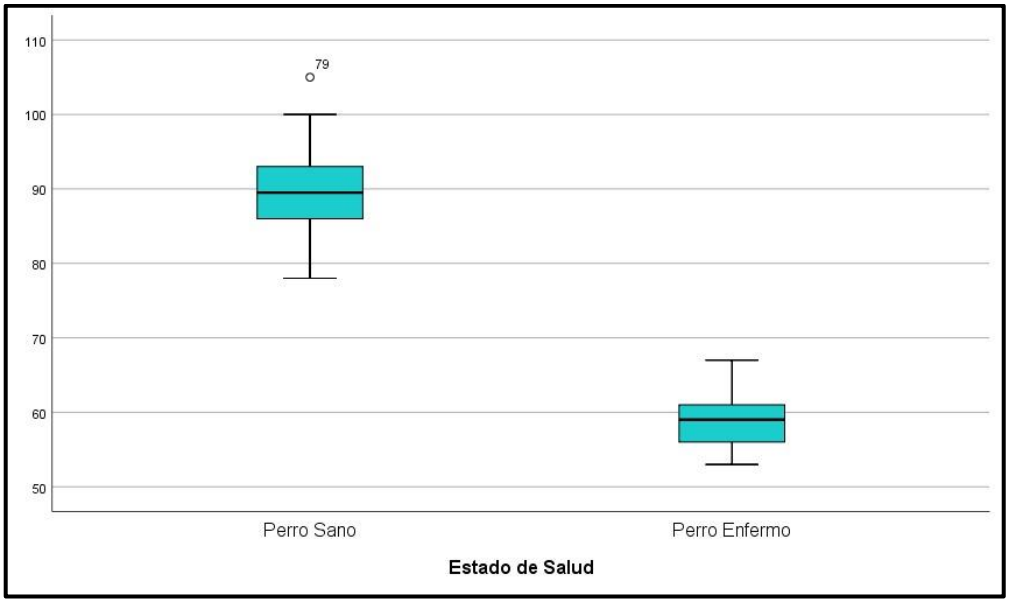


Figura 35. Diagrama de caja para la frecuencia cardíaca registrada por los perros sanos y enfermo en (lpm).

4.1.3 Prueba de normalidad, análisis comparativo y resultados de la ubicación GPS entre el prototipo y Google Maps.

A continuación, se realizó una prueba de normalidad con el objetivo de determinar qué tipo de distribución se presenta en las mediciones de latitud y longitud obtenidas con el prototipo y con Google Maps.

Ho: Los datos medidos tienen una distribución normal.
Hi: Los datos medidos no tienen una distribución normal.

Tabla 11. P-Valor para los datos GPS.

	P-Valor	
Variables	**Prototipo**	**Google Maps**
Latitud	0.006	0.006
Longitud	0.982	0.397

En la Tabla 11, se representan los valores de latitud obtenidos tanto por el prototipo como por Google Maps, que tienen un p-valor menor a 0.05, lo cual indica que los datos no siguen una distribución normal. Por tal motivo, se recurre a la prueba de Wilcoxon para muestras relacionadas.

Asimismo, en la variable longitud, ambos dispositivos presentan p-valores mayores a 0.05, lo cual sugiere que los datos se distribuyen normalmente. Por ello, se utiliza la prueba t para muestras relacionadas para comparar las mediciones de longitud entre el prototipo y Google Maps.

Prueba T para muestras Relacionadas de Longitud

Ho: No existe diferencia significativa entre las longitudes registradas por el prototipo y Google Maps.
Hi: Sí existe una diferencia significativa entre las longitudes registradas por el prototipo y Google Maps.

Tabla 12. Prueba T para muestras Relacionadas de la Longitud.

Variable	Media	P-Valor (t-test)
Longitud Prototipo	-78.6587	0.656
Longitud Google Maps	-78.6587	

A partir de la Tabla 12, se observa que el p-valor es mayor a 0.05, lo que indica que no existen diferencias entre las mediciones de ambos dispositivos. Esto indica que ambos sistemas presentan un comportamiento similar en la detección de la ubicación en términos de longitud.

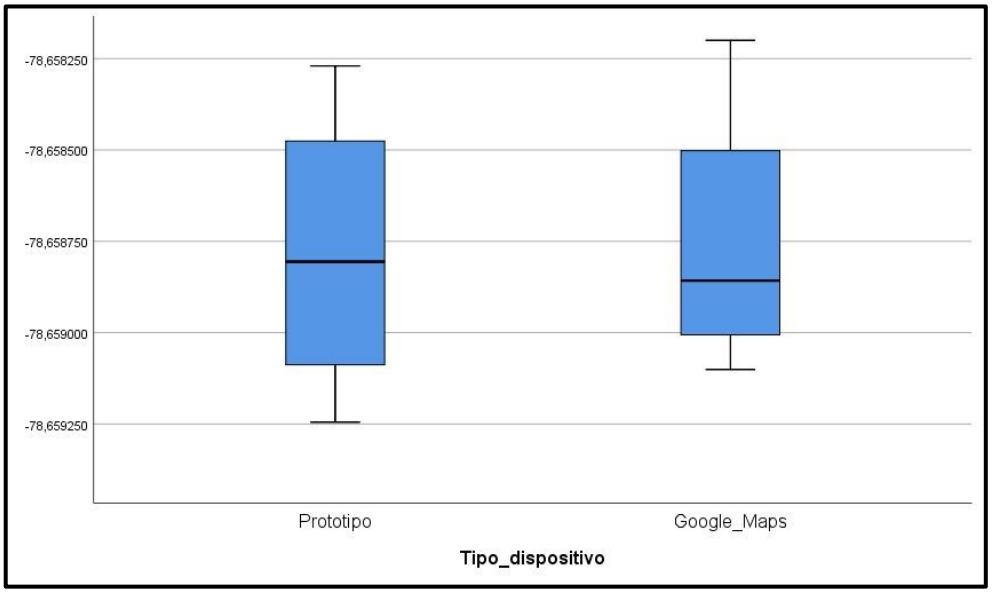


Figura 36. Diagrama de caja para la longitud registrada por el prototipo y Google Maps.

Asimismo, como en las figuras anteriores**,** la Figura 36 indica el diagrama de caja que corresponde a los datos de longitud obtenidos tanto por el prototipo como Google Maps.

Prueba de Wilcoxon para muestras relacionadas en Latitud

Ho: La mediana de diferencias entre la latitud del prototipo y la latitud de Google Maps es igual a 0.
Hi: La mediana de diferencias entre la latitud del prototipo y la latitud de Google Maps no es igual a 0.

Tabla 13. P-valor de la Latitud. Prueba Wilcoxon.

Variable	Mediana	P-Valor
Latitud Prototipo	-1.66	0.715
Latitud Google Maps	-1.66	

Se muestra en la Tabla 13 el p-valor obtenido en la prueba de Wilcoxon que es mayor a 0.05. Como en las otras pruebas, se acepta la hipótesis nula, ya que no existen diferencias significativas entre los valores de latitud medidos por los dos dispositivos. En la Figura 37 se observa el diagrama de cajas de los valores obtenidos de la latitud.

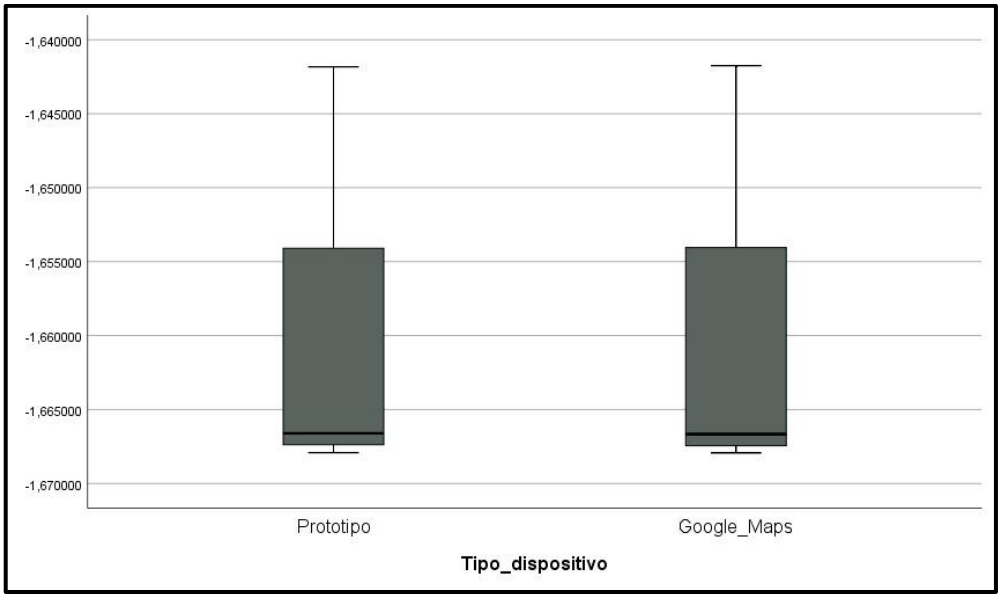


Figura 37.Diagrama de caja para la latitud registrada por el prototipo y Google Maps.

En resumen, los resultados obtenidos demuestran que el prototipo desarrollado posee la capacidad de medir de manera eficiente los signos vitales de temperatura, frecuencia cardíaca y saturación de oxígeno, evidenciando una elevada concordancia con los dispositivos veterinarios convencionales, conforme a las pruebas estadísticas realizadas. De igual modo, el sistema de rastreo basado en GPS mostró una precisión adecuada al ser comparado con la ubicación proporcionada por Google Maps, lo cual valida su operatividad en condiciones reales. Estos hallazgos sustentan la confiabilidad del prototipo para el monitoreo remoto de mascotas, consolidando la tecnología LoRaWAN como una alternativa viable para la transmisión de datos biomédicos y geográficos en aplicaciones veterinarias.

CAPÍTULO V.

5.1 CONCLUSIONES

- Con el estudio realizado entre diversas tecnologías de comunicación inalámbrica se determinó que LoRaWAN es la opción más adecuada para este proyecto, debido a su amplio alcance, eficiencia y bajo consumo energético. Asimismo, el estudio de los parámetros fisiológicos desde un enfoque veterinario permitió orientar el diseño del prototipo hacia necesidades reales y específicas del monitoreo en canes, asegurando así su aplicabilidad.

- A partir del análisis técnico y médico realizado, se diseñó e implementó un prototipo funcional que incorpora varios sensores como el de temperatura, frecuencia cardíaca, SpO2, y un módulo GPS, enlazados mediante una red LoRaWAN. La arquitectura desarrollada permitió la obtención, procesamiento y el envío de datos hacia Firebase, lo que agilizó su posterior visualización en la aplicación móvil. Como resultado, se obtuvo un dispositivo ergonómico, funcional y portátil, adecuado para el monitoreo continuo del estado físico y entorno de la mascota.

- Las pruebas realizadas en la Clínica Veterinaria contribuyeron a la evaluación del rendimiento del sistema desarrollado. Estas permitieron comprobar y verificar el funcionamiento correcto del dispositivo en condiciones reales, por medio de mediciones fisiológicas de los animales. Asimismo, la aplicación de las pruebas estadísticas sirvió para validar la fiabilidad y precisión de los datos obtenidos y respalda la efectividad del prototipo respecto a equipos comerciales.

5.2 RECOMENDACIONES

- Para una mejor comodidad y funcionalidad del prototipo, se puede realizar un diseño más ergonómico para la sujeción de los sensores, tomando en cuenta este criterio, se puede buscar opciones que reduzcan el impacto sobre la mascota sin dejar de lado la estabilidad de los dispositivos durante el monitoreo.

- Pese a que el prototipo ha demostrado tener un buen desempeño en las pruebas y ser confiable, es importante realizar un estudio más extenso en donde se pueda incluir diferentes factores para validar la aplicabilidad del sistema en distintas condiciones de salud o actividades.

- Para mejorar la precisión en la detección de anomalías en los signos vitales, se recomienda integrar técnicas de aprendizaje automático, específicamente redes neuronales artificiales (RNA) o modelos de detección de anomalías. Estos algoritmos pueden entrenarse con los datos recopilados por el prototipo y el dispositivo veterinario de referencia para identificar patrones normales y detectar valores fuera de rango que podrían indicar problemas de salud.

BIBLIOGRAFÍA

[1] D. Evans, “The Internet of Things: How the Next Evolution of the Internet Is Changing Everything”, 2011.

[2] E. Carrasco, “Metodología para selección de tecnologías LPWAN para diversas aplicaciones de internet de las cosas”, pp. 15–16, 2020.

[3] M. Andrés, M. Quimbita, : Carlos, y E. P. Salvador, “Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos”, 2018.

[4] J. Haxhibeqiri, E. De Poorter, I. Moerman, y J. Hoebeke, “A survey of LoRaWAN for IoT: From technology to application”, 2018, *MDPI AG*. doi: 10.3390/s18113995.

[5] S. Mohandass, S. Sridevi, y R. Sathyabama C A Assistant, “Animal Health Monitoring and Intrusion Detection System based on LORAWAN”, 2021.

[6] G. Sastre, “Localizador GPS basado en Arduino. Diseño y fabricación de prototipo para su aplicación en el análisis de parámetros de rutas.”, UNIVERSIDAD DE VALLADOLID, Valladolid, 2024.

[7] R. K. Raghunathan, “History of Microcontrollers: First 50 Years”, *IEEE Micro*, vol. 41, núm. 6, pp. 97–104, nov. 2021.

[8] C. Ndukwe, M. T. Iqbal, X. Liang, J. Khan, y L. Aghenta, “LoRa-based communication system for data transfer in microgrids”, *AIMS Electronics and Electrical Engineering*, vol. 4, núm. 3, pp. 311–312, sep. 2020, doi: 10.3934/ElectrEng.2020.3.303.

[9] Ricardo. Chuqui y Diego. Castro, “DESPLIEGUE Y CONFIGURACIÓN DE UNA RED LoRaWAN USANDO UNA PLATAFORMA THE THINGS NETWORK (TTN) PARA DISPOSITIVOS DE INTERNET DE LAS COSAS (IoT)”, Tesis de ingeniería, Universidad Politécnica Salesiana, Cuenca, 2022. Consultado: el 23 de abril de 2025. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/23944

[10] B. Cevallos y S. Rubio, “DESARROLLO DE UNA RED IOT CON TECNOLOGÍA LORA PARA GESTIÓN DE INVERNADEROS.”, UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO, Quito, 2021. Consultado: el 23 de

abril de 2025. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/20297

[11] S. Hernández, "Estudio en detalle de LoRaWAN. Comparación con otras tecnologías LPWAN considerando diferentes patrones de tráfico.", Master Tesis, Universitat Oberta de Catalunya., 2020. Consultado: el 23 de abril de 2025. [En línea]. Disponible en: http://hdl.handle.net/10609/106369

[12] O. Criollo, D. Chávez, M. Inapanta, M. Pérez, y A. Valencia, *Fundamentos Básicos de la Medicina Veterinaria*, 1a ed. Ecuador: Instituto de Investigaciones Transdisciplinarias, 2022. doi: https://doi.org/10.56846/bin.ec.YJMY4674.

[13] B. G. Klein, *Cunningham's Textbook of Veterinary Physiology*, 6th ed. St. Louis, Missouri, USA: Elsevier Saunders, 2021. [En línea]. Disponible en: http://evolve.elsevier.com/Klein/physiology/

[14] E. Thomovsky, P. Johnson, y A. Brooks, *Basic Monitoring in Canine and Feline Emergency Patients*. Wallingford, Oxfordshire, UK ; Boston, MA: CABI, 2020. Consultado: el 4 de mayo de 2025. [En línea]. Disponible en: https://lccn.loc.gov/2019039193

[15] V. Mina y A. Restrepo, "Desarrollo de un Prototipo de Monitoreo No Invasivo de Signos Vitales y Localización para Caninos", Tesis de Ingeniería, Institución Universitaria Antonio Jose Camacho, Cali, 2019. Consultado: el 23 de abril de 2025. [En línea]. Disponible en: https://repositorio.uniajc.edu.co/handle/uniajc/1138

[16] Y. León, "SISTEMA ELECTRÓNICO DE MONITOREO DE MASCOTAS PARA LA GESTIÓN DE CLÍNICAS VETERINARIAS UTILIZANDO VOIP E IOT.", Tesis de Ingeniería, Universidad Técnica de Ambato, Ambato, 2022. Consultado: el 23 de abril de 2025. [En línea]. Disponible en: https://repositorio.uta.edu.ec/handle/123456789/34895

[17] L. F. Tull Soriano, "PROTOTIPO DE MONITOR DE SIGNOS VITALES EN PACIENTES VETERINARIOS DE ESPECIE CANINA UTILIZANDO IOT", Tesis de Ingeniería, Universidad Nacional Pedro Henríquez Ureña, Santo Domingo, 2021.

[18] Y. Nakazawa, T. Ohshima, M. Kitagawa, T. Nuruki, y A. Fujiwara-Igarashi, "Relationship between Respiratory Rate, Oxygen Saturation, and Blood Test Results in Dogs with Chronic or Acute Respiratory Disease: A Retrospective Study", *Vet Sci*, vol. 11, núm. 1, p. 2, ene. 2024, doi: 10.3390/vetsci11010027.

[19] S. G. Alejandra, “Uso de la Azaperona como tranquilizante en perros y su efecto sobre la Frecuencia Cardíaca y Presión Arterial.”, Tesis Médica Veterinaria, Universidad Nacional Autónoma de México, Cuautitlán, 2024.

[20] B. Cugmas, P. Šušterič, N. R. Gorenjec, y T. Plavec, “Comparison between rectal and body surface temperature in dogs by the calibrated infrared thermometer”, *Vet Anim Sci*, vol. 9, p. 100120, jun. 2020, doi: 10.1016/J.VAS.2020.100120.

[21] F. Suárez, C. Santamaria, Y. Avila, y W. Cuevas, “Instrumentación de Variables Biomédicas en Animales Domésticos para Mediciones no Invasivas”, en *Congreso Internacional de Ciencias de la Computación y la Ingeniería (CICI 2018)*, Bogotá: ESCOM - Escuela de Comunicaciones del Ejército Nacional, oct. 2018, pp. 4–8.

[22] F. Reboll, “Diseño y desarrollo de un dispositivo electrónico de uso personal para la monitorización continuada de la saturación de oxígeno en sangre.”, Tesis ingeniería, Universidad Politécnica de Valencia, 2020.

[23] A. Villacorte, “Construcción de un prototipo de localizador GSM para la ubicación de mascotas.”, Tesis ingeniería, Universidad Mariana, San Juan de Pasto, 2024.

[24] C. Ndukwe, M. T. Iqbal, X. Liang, J. Khan, y L. Aghenta, “LoRa-based communication system for data transfer in microgrids”, *AIMS Electronics and Electrical Engineering*, vol. 4, núm. 3, pp. 303–325, sep. 2020, doi: 10.3934/ElectrEng.2020.3.303.

[25] R. Pereira, C. de Souza, D. Patino, y J. Lata, “Plataforma de enseñanza a distancia de microcontroladores e internet de las cosas”, *Ingenius*, vol. 2022, núm. 28, p. 54, jul. 2022, doi: 10.17163/ings.n28.2022.05.

[26] P. A. Daniel y R. Risco, “Implementación de Lora y Lorawan como escenario futuro de la industrias 4.0 en el sector agroindustrial peruano”, *Campus*, vol. 25, núm. 29, pp. 133–147, dic. 2019, doi: 10.24265/campus.2019.v25n29.10.

[27] A. Tashildar, N. Shah, R. Gala, T. Giri, y P. Chavhan, “APPLICATION DEVELOPMENT USING FLUTTER”, *International Research Journal of Modernization in Engineering Technology and Science @International Research Journal of Modernization in Engineering*, vol. 2, núm. 8, pp. 2582–5208, ago. 2020, [En línea]. Disponible en: www.irjmets.com

[28] A. Paz, “Análisis e Implementación de un Sistema de Rastreo Satelital aplicado a mascotas mediante software libre con tecnología GPS y GSM.”, Tesis Magister, Universidad Católica de Santiago de Guayaquil, Guayaquil, 2021. Consultado: el 23

de abril de 2025. [En línea]. Disponible en: http://repositorio.ucsg.edu.ec/handle/3317/16332

[29] J. Gavilanes Bayas, “SISTEMA ELECTRÓNICO DE DETECCIÓN Y RASTREO DE MASCOTAS”, Tesis de ingeniería, Universidad Técnica de Ambato, Ambato, 2018.

[30] C. E. Valdivieso Taborga, R. Valdivieso Castellón, y O. Á. Valdivieso Taborga, “DETERMINACIÓN DEL TAMAÑO MUESTRAL MEDIANTE EL USO DE ÁRBOLES DE DECISIÓN”, *INVESTIGACION & DESARROLLO*, vol. 11, núm. 1, pp. 53–80, jul. 2011, doi: 10.23881/idupbo.011.1-4e.

[31] R. Rentería, C. Ortiz, y O. Gaxiola, “ANÁLISIS COMPARATIVO DE TECNOLOGÍAS BLUETOOTH, WIFI, ZIGBEE Y LORA CON UN ENFOQUE EN LA IMPLEMENTACIÓN DE UNA RED DE MALLA.”, *Revista ELECTRO*, vol. 46, Culiacán, pp. 148–153, 2024. Consultado: el 1 de mayo de 2025. [En línea]. Disponible en: https://itchihuahua.mx/revista_electro

ANEXOS

Anexo A. Código Arduino Transmisor y Gateway.

```
//EMISOR// - Código para el dispositivo emisor.
#include <SPI.h>   // Comunicación SPI (LoRa).
#include <LoRa.h>  // Control del módulo LoRa.
#include <Wire.h>  // Comunicación I2C (OLED, sensores).
#include <Adafruit_MLX90614.h> // Sensor de temperatura infrarrojo.
#include <TinyGPS++.h>        // Librería GPS.
#include <Adafruit_MPU6050.h>  // Acelerómetro y giroscopio.
#include <Adafruit_Sensor.h> // Base para sensores Adafruit.
#include <Adafruit_SSD1306.h>  // Librería OLED.
#include "MAX30105.h"        // Sensor HR/SpO2.
#include "spo2_algorithm.h"  // Algoritmo SpO2.

// Configuración OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET); // Objeto OLED.

// Configuración sensores
Adafruit_MLX90614 mlx = Adafruit_MLX90614(); // Objeto temp.
TinyGPSPlus gps;                          // Objeto GPS.
Adafruit_MPU6050 mpu;                       // Objeto acel/giro.
MAX30105 particleSensor;                  // Objeto HR/SpO2.

// Configuración pines LoRa
#define LORA_CS 5   // Pin CS LoRa.
#define LORA_RST 14  // Pin RST LoRa.
#define LORA_DIO0 2  // Pin DIO0 LoRa.

// Configuración ADC batería
#define PIN_BATERIA 27      // Pin ADC batería.
#define VOLTAJE_REF 3.3     // Voltaje ref. ESP32.
#define RESOLUCION_ADC 4095 // Resolución ADC (12 bits).
#define R1 330.0            // Resistencia R1 divisor.
#define R2 220.0            // Resistencia R2 divisor.

// Niveles batería críticos
#define NIVEL_BATERIA_PLENA 8   // Voltaje máximo batería.
#define NIVEL_BATERIA_BAJA 5.5  // Voltaje crítico batería.

// Configuración pines GPS
```

```
static const int RXPin = 17; // Pin RX GPS.
static const int TXPin = 16; // Pin TX GPS.
static const uint32_t GPSBaud = 9600; // Baudrate GPS.

// Variables
bool sensorTempConectado = false; // Estado sensor temperatura.
bool gpsConectado = false;        // Estado GPS.
int32_t bufferLength;
int32_t spo2;          // Valor SpO2.
int32_t heartRate;     // Valor HR.
int8_t validSPO2;      // Validez SpO2.
int8_t validHeartRate; // Validez HR.
//unsigned long previousMillis = 0;
//const long interval = 1000;

// Variables control tiempo
unsigned long previousMillisBateria = 0;
unsigned long previousMillisTemp = 0;
unsigned long previousMillisAcelerometro = 0;
unsigned long previousMillisHR = 0;
unsigned long previousMillisGPS = 0;

// Intervalos
const unsigned long intervaloBateria = 1000;    // Intervalo batería (1s).
const unsigned long intervaloCategorias = 1000; // Intervalo sensores (1s).

// Variables detección actividad
#define MAX_DATOS 10
float aceleracionBuffer[MAX_DATOS];
int bufferIndex = 0;
bool bufferLleno = false;

// Pines LEDs
int leda = 25;
int ledb = 33; // Pin para el segundo LED.

void setup() {
  Serial.begin(115200); // Inicializa la comunicación serial.

  //Estados de leds
  pinMode(leda, OUTPUT); // Configura el pin del LED 'a' como salida.
  pinMode(ledb, OUTPUT); // Configura el pin del LED 'b' como salida.

  // Inicializar OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Inicializa la OLED con
dirección I2C 0x3C.
    Serial.println(F("Error al iniciar OLED"));
    //while (true);
  }
```

```
  display.clearDisplay(); // Limpia la pantalla OLED.
  display.display();      // Muestra el buffer en la pantalla.

  // Mensaje a mostrar
  String inicio = "Iniciando..."; // Mensaje de inicio.
  int size = 1;                   // Tamaño del texto.

  // Calcular posición centrada
  int x_centrada = (128 - (inicio.length() * 6 * size)) / 2; // Calcula la posición X
centrada.
  int y_centrada = (32 - (8 * size)) / 2;                    // Calcula la posición Y centrada.

  // Configurar y mostrar el mensaje centrado
  display.setTextSize(size);        // Establece el tamaño del texto.
  display.setTextColor(SSD1306_WHITE); // Establece el color del texto a blanco.
  display.setCursor(x_centrada, y_centrada); // Establece la posición del cursor.
  display.println(inicio);            // Imprime el mensaje.
  display.display();                  // Muestra el mensaje.
  delay(1000);                        // Espera 1 segundo.

  // Configurar ADC para leer batería
  analogReadResolution(12); // Establece la resolución del ADC a 12 bits.

  int line = 0; // Contador de líneas para la OLED.

  display.clearDisplay(); // Limpia la pantalla.
  display.display();      // Muestra (limpio).
  display.setTextSize(1); // Establece el tamaño de texto predeterminado.

  // Configuración pines e Inicialización LoRa
  LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0); // Define los pines de LoRa.
  if (!LoRa.begin(433E6)) {               // Inicializa LoRa a 433MHz.
    Serial.println("Error al iniciar LoRa");
    displayMessage("Error LoRa", line);
    //while (1);
  } else {
    Serial.println("LoRa iniciado correctamente");
    displayMessage("LoRa iniciado", line);
    LoRa.setTxPower(20); // Establece la potencia de transmisión de LoRa a 20 dBm.
  }

  // Inicializar sensor temperatura MLX90614
  sensorTempConectado = mlx.begin(); // Inicializa el sensor de temperatura.
  if (sensorTempConectado) {
    Serial.println("Sensor de Temperatura iniciado correctamente.");
    displayMessage("Temp iniciado", line);
  } else {
    Serial.println("Error al iniciar sensor de Temperatura.");
```

```
    displayMessage("Error Temp", line);
  }

  // Inicializar MPU6050
  if (!mpu.begin()) { // Inicializa el acelerómetro/giroscopio.
    Serial.println("Error al iniciar Acelerometro");
    displayMessage("Error Acelerometro", line);
    //while (1);
  } else {
    Serial.println("Acelerometro iniciado correctamente");
    displayMessage("Acelerometro iniciado", line);
    mpu.setAccelerometerRange(MPU6050_RANGE_8_G);   // Rango acelerómetro +/-
8g.
    mpu.setGyroRange(MPU6050_RANGE_500_DEG);       // Rango giroscopio +/- 500
deg/s.
    mpu.setFilterBandwidth(MPU6050_BAND_21_HZ); // Ancho de banda del filtro.
  }

  // Inicialización GPS (Serial2 en ESP32)
  Serial2.begin(GPSBaud, SERIAL_8N1, RXPin, TXPin); // Inicializa la serial para el
GPS.
  gpsConectado = true;                          // Marca el GPS como conectado.

  // Inicializar sensor MAX30102
  if (particleSensor.begin(Wire, I2C_SPEED_FAST)) { // Inicializa el sensor HR/SpO2
(I2C rápido).
    Serial.println("Oximetro iniciado correctamente.");
    displayMessage("Oximetro iniciado", line);
    byte ledBrightness = 40;
    byte sampleAverage = 2;
    byte ledMode = 2;
    byte sampleRate = 100;
    int pulseWidth = 411;
    int adcRange = 8192;
    particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange); // Configura el sensor HR/SpO2.
  } else {
    Serial.println("Error al iniciar Oximetro.");
    displayMessage("Error Oximetro", line);

  }

  delay(1000);            // Espera 1 segundo.
  display.clearDisplay(); // Limpia la pantalla OLED.
  display.display();      // Muestra (limpio).
}

// Función para mostrar mensajes en la OLED
void displayMessage(const char* message, int &line) {
```

```
  display.setCursor(0, line * 10); // Posición del texto en la OLED.
  display.println(message);       // Imprime el mensaje.
  display.display();              // Muestra el mensaje.
  line++;                         // Incrementa el contador de líneas.
  if (line >= 3) {                // Si se alcanzan 3 líneas.
    delay(500);
    display.clearDisplay(); // Limpia la OLED.
    line = 0;             // Reinicia el contador de líneas.
  }
  delay(1000); // Espera 1 segundo después del mensaje.
}

void loop() {
  unsigned long currentMillis = millis(); // Tiempo actual.
  String mensaje = "";                    // Mensaje a enviar por LoRa.

  // Valores de umbral
  float tempMin = 20;  // Temp mínima.
  float tempMax = 30;  // Temp máxima.
  int heartRateMin = 60; // HR mínima.
  int heartRateMax = 160; // HR máxima.
  int spo2Min = 80;    // SpO2 mínima.
  int spo2Max = 110;   // SpO2 máxima.
  int lineS = 0;       // Contador de líneas OLED (loop).

  if (currentMillis - previousMillisBateria >= intervaloBateria) { // Control de tiempo
batería.
    previousMillisBateria = currentMillis;

    // Leer nivel de batería
    int lecturaADC = analogRead(PIN_BATERIA);            // Lee el ADC.
    float voltajePin = (lecturaADC * VOLTAJE_REF) / RESOLUCION_ADC; //
Convierte a voltaje.
    float voltajeBateria = voltajePin * ((R1 + R2) / R2);      // Calcula voltaje real.

    // Mostrar nivel de batería
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("Voltaje Bateria:");
    display.print(voltajeBateria);
    display.println(" V");

    if (voltajeBateria <= NIVEL_BATERIA_BAJA) {
      mensaje += "Bateria BAJA: " + String(voltajeBateria) + " V ;";
      digitalWrite(leda, HIGH); // Enciende LED bajo batería.
      delay(500);
      digitalWrite(leda, LOW);  // Apaga LED.
      display.println("Bateria BAJA");
      //Serial.println("¡Advertencia! Nivel de batería bajo.");
```

```
} else if (voltajeBateria >= NIVEL_BATERIA_PLENA) {
  mensaje += "Bateria ALTA: " + String(voltajeBateria) + " V; ";
  display.println("Bateria CARGADA");
  //Serial.println("Batería completamente cargada.");
} else {
  mensaje += "Bateria: " + String(voltajeBateria) + " V; ";
  display.println("Bateria OK");
  //Serial.println("Batería en buen estado.");
}

display.display();
delay(2000);

//Limpiar OLED
display.clearDisplay();
display.setCursor(0, 0);

// Lectura del MLX90614
if (sensorTempConectado) {
  float temperatura = mlx.readObjectTempC(); // Lee temperatura.
  if (temperatura < tempMin && temperatura >= 5) {
    mensaje += "¡Alerta! " + String(temperatura) + " °C. " + "Temperatura baja;";
    display.print("Temp: ");
    display.print(temperatura);
    display.println(" C.");
  } else if (temperatura > tempMax && temperatura <= 40) {
    mensaje += "¡Alerta! " + String(temperatura) + " °C. " + "Temperatura alta;";
    display.print("Temp: ");
    display.print(temperatura);
    display.println(" C.");
  } else if (temperatura < 5 || temperatura > 40 ){
    mensaje += "Esperando temperatura adecuada;";
    display.print("Temp: ");
    display.println("Invalida");
  } else {
    mensaje += "Temperatura: " + String(temperatura) + " °C; ";
    display.print("Temp: ");
    display.print(temperatura);
    display.println(" C.");
  }
} else {
  mensaje += " Alerta: Sensor Temp no conectado; ";
  display.println("Temp: Desconectado");
}

// Lectura del GPS
while (Serial2.available() > 0) {
  gps.encode(Serial2.read()); // Lee datos GPS.
}
```

```
  if (gps.location.isValid()) {
    mensaje += " Lat: " + String(gps.location.lat(), 6) + ", Lon: " +
String(gps.location.lng(), 6) + "; ";
    display.println("GPS: Normal Signal;");
  } else {
    mensaje += " Alerta: GPS sin señal; ";
    display.println("GPS: No Signal");
  }

  // Lectura del MPU6050
  if (mpu.begin()) {
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp); // Lee eventos del sensor.

    // Detección de actividad
    float aceleracionTotal = sqrt(a.acceleration.x * a.acceleration.x + a.acceleration.y *
a.acceleration.y + a.acceleration.z * a.acceleration.z);
    aceleracionBuffer[bufferIndex] = aceleracionTotal;
    int prevIndex = (bufferIndex == 0) ? MAX_DATOS - 1 : bufferIndex - 1;

    bufferIndex++;
    if (bufferIndex >= MAX_DATOS) {
      bufferIndex = 0;
      bufferLleno = true;
    }

    if (bufferLleno || bufferIndex > 1) {
      float diferencia = abs(aceleracionBuffer[prevIndex] - aceleracionTotal);
      if (diferencia > 1.0) {
        mensaje += " Actividad: Mascota corriendo; ";
        display.println("Actividad: Corriendo ");
      } else if (diferencia > 0.30 && diferencia <= 1.0) {

          mensaje += " Actividad: Mascota caminando; ";

        display.println("Actividad: Caminando ");
      } else {
        mensaje += " Actividad: Mascota en reposo; ";
        display.println("Actividad: Descanso ");
      }
    }

    // Detección de movimientos bruscos
    if (abs(g.gyro.x) > 2.0 || abs(g.gyro.y) > 2.0 || abs(g.gyro.z) > 2.0) {
      mensaje += " Movimiento brusco detectado; ";
      display.println("Movimiento: Brusco ");
    } else {
      mensaje += " Movimiento normal; ";
      display.println("Movimiento: Normal ");
    }
```

```
    } else {
      mensaje += " Alerta: Acelerómetro no conectado; ";
      display.println("Mov: Desconectado");
    }

    display.display();

    // Lectura sensor MAX30102 (HR y SpO2)
    if (particleSensor.begin(Wire, I2C_SPEED_FAST)) {
      int32_t heartRate, spo2;
      int8_t validHeartRate, validSPO2;
      uint32_t redBuffer[100], irBuffer[100];

      for (byte i = 0; i < 100; i++) {
        while (particleSensor.available() == false) {
          particleSensor.check();
        }
        redBuffer[i] = particleSensor.getRed();
        irBuffer[i] = particleSensor.getIR();
        particleSensor.nextSample();
      }

      maxim_heart_rate_and_oxygen_saturation(irBuffer, 100, redBuffer, &spo2,
&validSPO2, &heartRate, &validHeartRate);

      display.clearDisplay();
      display.setCursor(0, 0);

      if (irBuffer[99] < 50000) {
        mensaje += " ¡Oxímetro mal ubicado!; ";
        display.println("Oximetro mal ubicado ");
      } else {
        if (heartRate < heartRateMin && heartRate >= 40) {
          mensaje += " ¡Alerta! " + String(heartRate) + " lpm. " + " Frecuencia cardiaca
baja; ";
          display.print("HR: ");
          display.print(heartRate);
          display.println(" lpm.");
        } else if (heartRate > heartRateMax && heartRate <= 200) {
          mensaje += " ¡Alerta! " + String(heartRate) + " lpm. " + " Frecuencia cardiaca
alta; ";
          display.print("HR: ");
          display.print(heartRate);
          display.println(" lpm.");
        } else if (heartRate < 40 || heartRate > 200 ) {
          mensaje += " Esperando pulso adecuado; ";
          display.println("HR: Invalido");
        } else {
          mensaje += " Frecuencia cardiaca: " + String(heartRate) + " lpm; ";
```

```
      display.print("HR: ");
      display.print(heartRate);
      display.println(" lpm.");
    }

    // Lectura de SpO2
    if (spo2 < spo2Min && spo2 >= 60 ) {
      mensaje += " ¡Alerta! " + String(spo2) + "%. " + " SpO2 bajo; ";
      display.print("Spo2: ");
      display.print(spo2);
      display.println(" %.");
    } else if (spo2 > spo2Max && spo2 <= 125 ) {
      mensaje += " ¡Alerta! " + String(spo2) + "%. " + " SpO2 alto; ";
      display.print("Spo2: ");
      display.print(spo2);
      display.println(" %.");
    } else if (spo2 < 60 || spo2 > 125) {
      mensaje += " Esperando SpO2 adecuado; ";
      display.println("Spo2: Invalido");
    } else {
      mensaje += " SpO2: " + String(spo2) + "%; ";
      display.print("Spo2: ");
      display.print(spo2);
      display.println(" %.");
    }
  }
} else {
  display.clearDisplay();
  display.setCursor(0, 0);
  mensaje += " Alerta: Oxímetro no conectado; ";
  display.println("Oximetro: Desconectado");
}

display.display();
Serial.println(mensaje);

// Enviar mensaje por LoRa
LoRa.beginPacket();
LoRa.print(mensaje);
int resultado = LoRa.endPacket(); // Finaliza y envía el paquete

// Validar envío LoRa
if (resultado == 0) {
  Serial.println("Error: No se pudo enviar el paquete.");
} else {
  Serial.println("Paquete enviado correctamente.");
  digitalWrite(ledb, HIGH); // Enciende LED envío OK.
  delay(200);
  digitalWrite(ledb, LOW);  // Apaga LED.
```

```
    }
    delay(3000); // Espera 3 segundos.
  }
}


//GATEWAY// - Código para el dispositivo Gateway.

#include <SPI.h>   // Comunicación SPI (LoRa).
#include <LoRa.h>  // Control del módulo LoRa.

#if defined(ESP32)
  #include <WiFi.h> // Librería WiFi para ESP32.
#elif defined(ESP8266)
  #include <ESP8266WiFi.h> // Librería WiFi para ESP8266.
#endif
#include <Firebase_ESP_Client.h> // Cliente Firebase para ESP.

// Info para generación de tokens Firebase.
#include "addons/TokenHelper.h"
// Funciones helper para RTDB Firebase.
#include "addons/RTDBHelper.h"

// Configuración red WiFi
#define WIFI_SSID1 "Tesis1"       // Nombre de la primera red WiFi.
#define WIFI_PASSWORD1 "*010213*" // Contraseña de la primera red WiFi.
#define WIFI_SSID2 "FibraTeleCoM_2.4GHz_DANY" // Nombre de la segunda red
WiFi.
#define WIFI_PASSWORD2 "Dany1978"           // Contraseña de la segunda red WiFi.

// Configuración proyecto Firebase
#define API_KEY "AIzaSyBqB6Z3dy65cAy1VO2Rj4bgDPcxUAX1sqM" // Clave API
de Firebase.
#define DATABASE_URL "https://base-de-datos-9445f-default-rtdb.firebaseio.com/" //
URL de la base de datos Firebase.

// Pines módulo LoRa
#define LORA_SCK 18  // Pin SCK LoRa.
#define LORA_MISO 19 // Pin MISO LoRa.
#define LORA_MOSI 23 // Pin MOSI LoRa.
#define LORA_SS 5    // Pin SS LoRa.
#define LORA_RST 14  // Pin RST LoRa.
#define LORA_DI0 2   // Pin DIO0 LoRa.

// Configuración ADC batería (igual que emisor)
#define PIN_BATERIA 27
#define VOLTAJE_REF 3.3
#define RESOLUCION_ADC 4095
#define R1 330.0
```

```
#define R2 220.0

// Niveles batería críticos (igual que emisor)
#define NIVEL_BATERIA_PLENA 8
#define NIVEL_BATERIA_BAJA 5.5

// Objeto Firebase
FirebaseData fbdo;      // Objeto para datos de Firebase.
FirebaseAuth auth;      // Objeto para autenticación Firebase.
FirebaseConfig config;    // Objeto para configuración Firebase.

unsigned long sendDataPrevMillis = 0; // Tiempo de la última publicación.
bool signupOK = false;              // Estado del registro en Firebase.

// Variables control tiempo
unsigned long previousMillisBateria = 0; // Última lectura de batería.

// Intervalos
const unsigned long intervaloBateria = 30000; // Intervalo lectura batería (30s).

// Pines LEDs
int leda = 26; // LED azul.
int ledb = 32; // LED rojo.

void conectarWiFi() {
  Serial.println("Intentando conectar a WiFi...");

  WiFi.begin(WIFI_SSID1, WIFI_PASSWORD1); // Intenta conectar a la primera red.
  int intentos = 0;

  while (WiFi.status() != WL_CONNECTED && intentos < 20) { // Espera conexión o
20 intentos.
    Serial.print(".");
    delay(500);
    intentos++;
  }

  if (WiFi.status() != WL_CONNECTED) { // Si no se conecta a la primera red.
    Serial.println("\nNo se pudo conectar a la primera red, intentando con la segunda...");
    WiFi.begin(WIFI_SSID2, WIFI_PASSWORD2); // Intenta conectar a la segunda red.
    intentos = 0;
    while (WiFi.status() != WL_CONNECTED && intentos < 20) { // Espera conexión o
20 intentos.
      Serial.print(".");
      delay(500);
      intentos++;
    }
  }
```

```
  if (WiFi.status() == WL_CONNECTED) { // Si la conexión WiFi es exitosa.
    Serial.println("\nConexión establecida.");
    Serial.print("Conectado a: ");
    Serial.println(WiFi.SSID());
    Serial.print("Con IP: ");
    Serial.println(WiFi.localIP());
  } else { // Si no se pudo conectar a ninguna red.
    Serial.println("\nNo se pudo conectar a ninguna red.");
  }
}

void setup() {
  // Configuración del monitor serie
  Serial.begin(115200);

  //Estados de leds
  pinMode(leda, OUTPUT); // Configura pin LED azul como salida.
  pinMode(ledb, OUTPUT); // Configura pin LED rojo como salida.

  // Configurar ADC para leer batería
  analogReadResolution(12); // Resolución ADC a 12 bits.

  // Intentar conectar a las redes WiFi
  conectarWiFi();

  // Configuración de Firebase
  config.api_key = API_KEY;       // Asigna la clave API.
  config.database_url = DATABASE_URL; // Asigna la URL de la base de datos.

  if (Firebase.signUp(&config, &auth, "", "")) { // Intenta registrarse en Firebase (sin
email/password).
    Serial.println("Conexión a Firebase exitosa");
    signupOK = true;
  } else {
    Serial.printf("Error en Firebase: %s\n", config.signer.signupError.message.c_str());
  }

  config.token_status_callback = tokenStatusCallback; // Callback para el estado del
token.

  Firebase.begin(&config, &auth); // Inicializa la conexión Firebase.

  Firebase.reconnectWiFi(true); // Intenta reconectar WiFi automáticamente.

  // Configuración de los pines del módulo LoRa
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_SS); // Inicializa la
comunicación SPI para LoRa.
  LoRa.setPins(LORA_SS, LORA_RST, LORA_DI0);       // Define los pines de control
de LoRa.
```

```
  // Inicialización de LoRa
  if (!LoRa.begin(433E6)) { // Configura la frecuencia LoRa a 433 MHz.
    Serial.println("No se pudo iniciar el módulo LoRa.");
    while (1); // Bucle infinito si falla la inicialización de LoRa.
  }

  Serial.println("Receptor LoRa iniciado correctamente.");
}

void loop() {

  unsigned long currentMillis = millis(); // Tiempo actual.
  if (currentMillis - previousMillisBateria >= intervaloBateria) { // Control de tiempo
batería.
    previousMillisBateria = currentMillis;

// Leer nivel de batería
    int lecturaADC = analogRead(PIN_BATERIA);            // Lee el valor ADC.
    float voltajePin = (lecturaADC * VOLTAJE_REF) / RESOLUCION_ADC; //
Convierte a voltaje.
    float voltajeBateria = voltajePin * ((R1 + R2) / R2);      // Calcula voltaje real.

    if (voltajeBateria <= NIVEL_BATERIA_BAJA) {
      digitalWrite(ledb, HIGH); // Enciende LED rojo (batería baja).
      delay(1000);
      digitalWrite(ledb, LOW);  // Apaga LED rojo.
      Serial.println("¡Advertencia! Nivel de batería bajo.");
      delay(1000);
    }
  }

  // Verifica si hay paquete LoRa disponible
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.print("Paquete recibido: ");
    digitalWrite(leda, HIGH); // Enciende LED azul (paquete recibido).
    delay(1000);
    digitalWrite(leda, LOW);  // Apaga LED azul.
    String received = "";     // Almacena el mensaje LoRa.

    while (LoRa.available()) {
      received += (char)LoRa.read(); // Lee byte y añade al string.
    }
    Serial.println(received);

    // Divide el mensaje por ';'
    int pos = 0;
    String token;
```

```
  int index = 0;
  String parts[10]; // Array para partes del mensaje.

  while ((pos = received.indexOf(';')) != -1) {
    token = received.substring(0, pos); // Extrae la parte.
    parts[index++] = token;           // Guarda la parte.
    received.remove(0, pos + 1);      // Elimina la parte procesada.
  }
  parts[index] = received; // Última parte (sin ';').

  // Envía cada parte a Firebase
  for (int i = 0; i <= index; i++) {
    if (Firebase.ready() && signupOK) {
      String path = "LoRa/part" + String(i + 1); // Ruta en Firebase.
      if (Firebase.RTDB.setString(&fbdo, path, parts[i])) {
        Serial.print("Parte ");
        Serial.print(i + 1);
        Serial.println(" enviada a Firebase exitosamente");
        digitalWrite(leda, HIGH); // Enciende LED azul (envío OK).
        delay(500);
        digitalWrite(leda, LOW);  // Apaga LED azul.
      } else {
        Serial.print("Error al enviar la parte ");
        Serial.print(i + 1);
        Serial.println(" a Firebase:");
        Serial.println(fbdo.errorReason()); // Imprime el error.
      }
    }
  }
}
```

Anexo B. Código Android Studio.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.unido"> <!-- Nombre del paquete de la aplicación -->

    <!-- Permite ejecutar servicios en primer plano, necesarios por ejemplo para rastreo o
notificaciones persistentes -->
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

    <!-- Permite acceso a la ubicación en segundo plano (cuando la app no está en uso) -->
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>

    <application
        android:label="unido" <!-- Nombre visible de la aplicación -->
        android:name="${applicationName}" <!-- Nombre de la clase Application (puede ser
configurado en Gradle) -->
        android:icon="@mipmap/ic_launcher"> <!-- Icono de la app -->

        <!-- Clave de API para usar Google Maps u otros servicios de geolocalización -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyAjpsJEf3xA0AAWNfCmh9H5NE38wMiCf8g"/>

        <activity
            android:name=".MainActivity" <!-- Actividad principal -->
            android:exported="true" <!-- Indica que esta actividad puede ser lanzada por otras
apps -->
            android:launchMode="singleTop" <!-- Reutiliza la actividad si ya está en la cima --
>
            android:taskAffinity="" <!-- Afinidad vacía: se comporta como parte de otra tarea -
->
            android:theme="@style/LaunchTheme" <!-- Tema usado al iniciar la app -->

android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreen
Size|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
            android:hardwareAccelerated="true" <!-- Usa aceleración por hardware -->
            android:windowSoftInputMode="adjustResize"> <!-- Ajusta el contenido al
mostrar el teclado -->

            <!-- Tema normal para Flutter -->
            <meta-data
              android:name="io.flutter.embedding.android.NormalTheme"
              android:resource="@style/NormalTheme" />

            <!-- Filtro de intentos para definir esta actividad como la lanzadora principal -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
```

```
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>

    <!-- Configuración interna para la versión de Flutter embedding -->
    <meta-data
        android:name="flutterEmbedding"
        android:value="2" />
  </application>

  <!-- Declaración de intenciones para consultar texto desde otras apps -->
  <queries>
    <intent>
      <action android:name="android.intent.action.PROCESS_TEXT"/>
      <data android:mimeType="text/plain"/>
    </intent>
  </queries>
</manifest>
```

OrderTraking

```
import 'dart:async';
// Manejo de tareas asincrónicas: delays, timers, y futuros.

import 'package:flutter/material.dart';
// Widgets y componentes visuales de Flutter.

import 'package:flutter_polyline_points/flutter_polyline_points.dart';
// Para generar rutas (polylines) entre puntos geográficos.

import 'package:google_maps_flutter/google_maps_flutter.dart';
// Permite mostrar y controlar mapas de Google en Flutter.

import 'package:location/location.dart';
// Maneja ubicación GPS del dispositivo, incluyendo permisos y actualizaciones.

import 'package:firebase_database/firebase_database.dart';
// Acceso a Firebase Realtime Database en tiempo real.

// WIDGET PRINCIPAL

class GoogleMapPage extends StatefulWidget {
  // Widget con estado, ya que se actualiza dinámicamente (mapa, ubicación, datos).
  const GoogleMapPage({super.key});

  @override
```

```
  State<GoogleMapPage> createState() => _GoogleMapPageState();
}

// ESTADO DEL WIDGET

class _GoogleMapPageState extends State<GoogleMapPage> {
  GoogleMapController? mapController;
  // Controlador del mapa, permite interactuar programáticamente con Google Maps.

  final DatabaseReference _databaseRef = FirebaseDatabase.instance.ref('LoRa');
  // Referencia a la ruta 'LoRa' en Firebase.

  final locationController = Location();
  // Controlador para gestionar la ubicación del dispositivo.

  LatLng? currentPosition;
  // Guarda la última ubicación conocida del dispositivo.

  Map<PolylineId, Polyline> polylines = {};
  // Almacena las líneas de ruta trazadas en el mapa.

  Map<String, String> firebaseData = {};
  // Almacena datos recibidos de Firebase.

  @override
  void initState() {
    // Método que se ejecuta al iniciar el widget (antes de mostrar en pantalla).
    super.initState();

    WidgetsBinding.instance.addPostFrameCallback((_) async => await initializeMap());
    // Ejecuta la inicialización del mapa después del primer renderizado del widget.

    fetchFirebaseData();
    // Comienza la escucha de cambios en Firebase en tiempo real.
  }

  Future<void> initializeMap() async {
    // Inicializa el mapa: obtiene ubicación y dibuja ruta.
    await fetchLocationUpdates();
    final coordinates = await fetchPolylinePoints();  // Coordenadas de la ruta
    generatePolyLineFromPoints(coordinates);          // Dibuja la ruta en el mapa
  }

  Future<void> fetchFirebaseData() async {
    // Escucha cambios en múltiples nodos de Firebase (ej. part1 a part6)


_databaseRef.onValue.listen((event) {

  // Escucha en tiempo real los cambios en la base de datos de Firebase
```

```
  if (event.snapshot.exists) {
    final data = event.snapshot.value as Map<dynamic, dynamic>;
    setState(() {
      firebaseData = data.map((key, value) => MapEntry(key, value.toString()));
      // Actualiza el estado con los nuevos datos convertidos a String
    });
  }
});

@override
Widget build(BuildContext context) => Scaffold(
  backgroundColor: Colors.black12, // Fondo general oscuro para la pantalla
  appBar: AppBar(
    title: const Text(
      'Monitoreo de mascotas',
      style: TextStyle(
        fontSize: 24,
        fontWeight: FontWeight.bold,
        fontFamily: 'Roboto',
        fontStyle: FontStyle.italic,
        color: Colors.white54,
      ),
    ),
    centerTitle: true,
    backgroundColor: Colors.blueGrey, // Color del AppBar
  ),
  body: currentPosition == null
      ? const Center(child: CircularProgressIndicator())
      // Si la posición aún no está disponible, muestra un spinner de carga
      : Column(
          children: [
            Expanded(
              flex: 1, // Mitad superior: vista de datos
              child: Container(
                padding: const EdgeInsets.all(16.0),
                child: firebaseData.isEmpty
                    ? const Center(child: Text('Cargando datos...'))
                    // Mensaje de espera si no hay datos aún
                    : ListView(
                        children: [
                          _buildMessageCard('Batería del dispositivo', firebaseData['part1'] ?? 'Sin
datos'),
                          _buildMessageCard('Temperatura de la mascota', firebaseData['part2'] ??
'Sin datos'),
                          _buildMessageCard('Nivel de Actividad', firebaseData['part4'] ?? 'Sin
datos'),
                          _buildMessageCard('Tipo de Movimiento', firebaseData['part5'] ?? 'Sin
datos'),
                          _buildMessageCard('Ritmo cardíaco', firebaseData['part6'] ?? 'Sin datos'),
```

```
                    _buildMessageCard('Concentración de oxígeno', firebaseData['part7'] ??
'Sin datos'),
                    // Se muestran los datos principales de monitoreo en tarjetas
                  ],
                ),
              ),
            ),
            Expanded(
              flex: 1, // Mitad inferior: mapa
              child: Stack(
                children: [
                  GoogleMap(
                    initialCameraPosition: CameraPosition(
                      target: const LatLng(-1.655760, -78.671113),
                      zoom: 5,
                    ),
                    onMapCreated: (GoogleMapController controller) {
                      setState(() {
                        mapController = controller;
                        // Se guarda el controlador del mapa
                      });
                    },
                    markers: _buildMarkers(firebaseData['part3']),
                    // Coloca el marcador con la posición actual
                  ),
                  if (_isGpsSignalLost(firebaseData['part3']))
                    Positioned(
                      top: 16.0,
                      left: 16.0,
                      right: 16.0,
                      // Si se detecta pérdida de señal GPS, muestra alerta visual
                    child: Container(
  padding: const EdgeInsets.all(12.0),
  decoration: BoxDecoration(
    color: Colors.redAccent, // Fondo rojo para indicar alerta
    borderRadius: BorderRadius.circular(8.0),
  ),
  child: const Text(
    'Alerta: GPS sin señal',
    style: TextStyle(
      color: Colors.white,
      fontWeight: FontWeight.bold,
    ),
    textAlign: TextAlign.center,
  ),
),
```

```dart
// ====================================
// FUNCIONES DE UBICACIÓN Y RUTAS
// ====================================

Future<void> fetchLocationUpdates() async {
  // Solicita permisos y actualiza posición del usuario en tiempo real
  bool serviceEnabled;
  PermissionStatus permissionGranted;

  serviceEnabled = await locationController.serviceEnabled();
  if (!serviceEnabled) {
    serviceEnabled = await locationController.requestService();
  }
  if (!serviceEnabled) return; // Si no se habilita servicio, se termina

  permissionGranted = await locationController.hasPermission();
  if (permissionGranted == PermissionStatus.denied) {
    permissionGranted = await locationController.requestPermission();
    if (permissionGranted != PermissionStatus.granted) return;
  }

  // Se actualiza `currentPosition` cada vez que cambia la ubicación
  locationController.onLocationChanged.listen((currentLocation) {
    if (currentLocation.latitude != null && currentLocation.longitude != null) {
      setState(() {
        currentPosition = LatLng(
          currentLocation.latitude!,
          currentLocation.longitude!,
        );
      });
    }
  });
}

Future<List<LatLng>> fetchPolylinePoints() async {
  // Llama a la API de Polyline para obtener una ruta entre dos puntos
  final polylinePoints = PolylinePoints();

  final result = await polylinePoints.getRouteBetweenCoordinates(
    'AIzaSyCRQm4PkcXsCVIQUt-wEK4k2JhC5XX0Czs', // API Key (reemplazable)
    PointLatLng(googlePlex.latitude, googlePlex.longitude),
    PointLatLng(mountainView.latitude, mountainView.longitude),
  );

  if (result.points.isNotEmpty) {
    // Convierte los puntos obtenidos en LatLng para usar en el mapa
    return result.points
        .map((point) => LatLng(point.latitude, point.longitude))
        .toList();
```

```
  } else {
    debugPrint(result.errorMessage); // Si hay error, lo imprime
    return [];
  }
}

Future<void> generatePolyLineFromPoints(List<LatLng> polylineCoordinates) async {
  // Crea una polyline (línea en el mapa) a partir de una lista de coordenadas
  const id = PolylineId('polyline');

  final polyline = Polyline(
    polylineId: id,
    color: Colors.blueAccent,
    points: polylineCoordinates,
    width: 5,
  );

  setState(() => polylines[id] = polyline); // Guarda la línea en el mapa
}

Widget _buildMessageCard(String title, String message) {
  // Construye una tarjeta visual con ícono y texto basado en el título
  IconData iconData;

  switch (title) {
    case 'Batería del dispositivo':
      iconData = Icons.battery_full;
      break;
    case 'Temperatura de la mascota':
      iconData = Icons.thermostat;
      break;
    case 'Nivel de Actividad':
      iconData = Icons.directions_run;
      break;
    case 'Tipo de Movimiento':
      iconData = Icons.motion_photos_on;
      break;

      case 'Ritmo cardíaco':

        iconData = Icons.favorite; // Ícono de corazón para FC
        break;
      case 'Concentración de oxígeno':
        iconData = Icons.water_drop; // Ícono de gota para SpO2
        break;
      default:
        iconData = Icons.info; // Ícono genérico por defecto
    }

    return Card(
```

```
    color: Colors.white70, // Fondo semitransparente
    margin: const EdgeInsets.symmetric(vertical: 2.0, horizontal: 5.0),
    elevation: 5.0, // Altura de sombra
    child: Padding(
      padding: const EdgeInsets.all(5.0),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Icon(
            iconData,
            color: Colors.blueAccent,
            size: 40.0,
          ),
          const SizedBox(width: 10.0),
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  title,
                  style: const TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 16.0,
                  ),
                ),
                const SizedBox(height: 3.0),
                Text(
                  message,
                  style: const TextStyle(
                    fontSize: 16.0,
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
}

Set<Marker> _buildMarkers(String? part3) {
  // Genera un marcador en el mapa si hay coordenadas válidas
  if (part3 == null || _isGpsSignalLost(part3)) {
    return {};
  }

  try {
    // Extrae latitud y longitud desde una cadena tipo: "Lat: xxx, Lon: xxx"
```

```
      final latMatch = RegExp(r'Lat:\s*(-?\d+\.\d+)').firstMatch(part3);
      final lonMatch = RegExp(r'Lon:\s*(-?\d+\.\d+)').firstMatch(part3);

      if (latMatch != null && lonMatch != null) {
        final latitude = double.parse(latMatch.group(1)!);
        final longitude = double.parse(lonMatch.group(1)!);

        return {
          Marker(
            markerId: const MarkerId('gps_location'),
            position: LatLng(latitude, longitude),
            infoWindow: const InfoWindow(title: 'Ubicación actual'),
          ),
        };
      }
    } catch (e) {
      print('Error al procesar las coordenadas: $e');
    }

    return {};
  }

  bool _isGpsSignalLost(String? part3) {
    // Verifica si la cadena contiene latitud y longitud
    return part3 == null || !part3.contains('Lat:') || !part3.contains('Lon:');
  }

}
```

Build.gradle

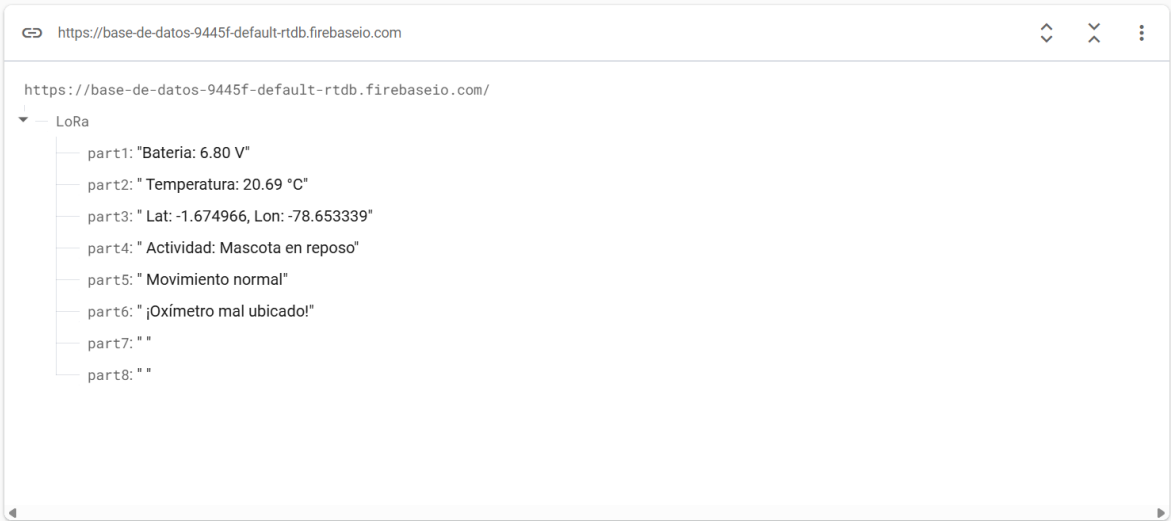
```
plugins {
    id "com.android.application"                  // Plugin base para aplicaciones Android
    id "kotlin-android"                           // Habilita soporte para Kotlin en Android
    id 'com.google.gms.google-services' version '4.4.2' apply false
    // Plugin de servicios de Google (Firebase, Maps). Se aplica al final.

    id "dev.flutter.flutter-gradle-plugin"        // Plugin específico de Flutter para Gradle
}

android {
    namespace = "com.example.unido"               // Nombre de espacio para la app
    compileSdkVersion 34                          // Versión del SDK usada para compilar
    buildToolsVersion "34.0.0"                    // Herramientas de compilación (build tools)
    ndkVersion = flutter.ndkVersion               // Versión del NDK usada por Flutter
```

```
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8  // Compatibilidad Java 8
        targetCompatibility = JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = JavaVersion.VERSION_1_8          // Compatibilidad con la JVM de
Java 8
    }

    defaultConfig {
        applicationId = "com.example.unido"          // ID único de la aplicación
        minSdk = flutter.minSdkVersion               // Versión mínima soportada por Flutter
        targetSdk = flutter.targetSdkVersion         // Versión objetivo del sistema operativo
        versionCode = flutter.versionCode            // Código de versión para despliegue
        versionName = flutter.versionName            // Nombre de la versión (ej: "1.0.0")

        buildConfigField "String", "MAPS_API_KEY",
"\"AIzaSyAjpsJEf3xA0AAWNfCmh9H5NE38wMiCf8g\""
        // Clave API de Google Maps inyectada en tiempo de compilación

        buildFeatures {
            buildConfig true                         // Habilita la generación de BuildConfig.java
        }
    }

    buildTypes {
        release {
            signingConfig = signingConfigs.debug      // Firma la app release con clave debug
por defecto
            // Reemplazar con configuración real para producción
        }
    }
}

flutter {
    source = "../.."                             // Ruta raíz del proyecto Flutter
}

dependencies {
    implementation platform('com.google.firebase:firebase-bom:33.7.0')
    // Firebase BoM permite manejar versiones compatibles de múltiples SDKs
automáticamente

    // SDKs opcionales de Firebase (descomentarlos según necesidad):
    // implementation "com.google.firebase:firebase-analytics"
    // implementation "com.google.firebase:firebase-auth"

    // Recomendación: no especificar versiones individuales al usar BoM
```

}

apply plugin: "com.google.gms.google-services" // Aplica el plugin de servicios de Google (Firebase)

Anexo C. Base de Datos en FIREBASE.



Anexo D. Aplicación en Android Studio.