
# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

## CARRERA DE INGENIERÍA EN MECATRÓNICA

## TRABAJO DE GRADO

## TEMA:

"NOTIFICACIÓN REMOTA DE FALLOS EN ROBOTS MÓVILES MEDIANTE INTEGRACIÓN DE LoRaWAN Y ROS"

Trabajo de titulación previo a la obtención del título de Ingeniero en Mecatrónica

**Línea de investigación:** Producción industrial y tecnología sostenible

**AUTOR:**

Antony Bolivar Chávez Arias

**DIRECTOR:**

PhD. Carlos Xavier Rosero Chandi

**Ibarra – Ecuador 2026**

# UNIVERSIDAD TÉCNICA DEL NORTE

## DIRECCIÓN DE BIBLIOTECA

### AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

## 1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

| DATOS DE CONTACTO | |
|---|---|
| **APELLIDOS Y NOMBRES:** | Chávez Arias Antony Bolivar |

| DATOS DE LA OBRA | |
|---|---|
| **TÍTULO:** | "Notificación remota de fallos en robots móviles mediante integración de LoRaWAN y ROS" |
| **AUTOR:** | Chávez Arias Antony Bolivar |
| **FECHA:** | 16/04/2026 |
| SOLO PARA TRABAJOS DE GRADO | |
| **PROGRAMA:** | ■ PREGRADO ☐ POSGRADO |
| **TÍTULO POR EL QUE OPTA:** | Ingeniero Mecatrónico |
| **ASESOR / DIRECTOR:** | PhD. Carlos Xavier Rosero Chandi<br>PhD. Brizeida Nohemí Gámez Aparicio |

## 2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 16 días del mes de abril de 2026

**EL AUTOR:**

Firmado electrónicamente por:
ANTONY BOLIVAR CHAVEZ ARIAS
Validar únicamente con FirmaEC

Nombre: Chávez Arias Antony Bolivar

C.I: 1755026497

# CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 16 de abril de 2026

PhD. Carlos Xavier Rosero Chandi

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI
Validar únicamente con FirmaEC

*PhD. Carlos Xavier Rosero Chandi*

*C.I:1002515821*

*Director de Tesis*

# APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificado del trabajo de Integración Curricular “NOTIFICACIÓN REMOTA DE FALLOS EN ROBOTS MÓVILES MEDIANTE INTEGRACIÓN DE LoRaWAN Y ROS” elaborado por Antony Bolivar Chávez Arias, previo a la obtención del título de Ingeniero Mecatrónico, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI
Validar únicamente con FirmaEC

*PhD. Carlos Xavier Rosero Chandi*

*C.I:1002515821*

*Director de Tesis*

Firmado electrónicamente por:
BRIZEIDA NOHEMI GAMEZ APARICIO
Validar únicamente con FirmaEC

*PhD. Brizeida Nohemí Gámez Aparicio*

*C.I:1758387383*

*Asesor de Tesis*

# Dedicatorias

Dedico este logro a mi familia, que ha sido mi fuerza en cada paso del camino. A mi mamá, por su ternura infinita, sus consejos sabios y por enseñarme a nunca rendirme. A mis hermanos, por sus risas, su apoyo y por recordarme siempre de dónde vengo.

A mi papá que, aunque esté lejos, su amor me acompaña cada día. Gracias por tus sacrificios silenciosos, por creer en mí incluso en la distancia y por ser mi ejemplo de trabajo, valentía y responsabilidad. Este logro también es tuyo, porque tu esfuerzo vive en mis sueños cumplidos.

Y, con esperanza en el corazón, dedico también estas palabras a mi futura familia y a mis hijos: que encuentren en mí el amor, la guía y el ejemplo que yo recibí, y que juntos construyamos un hogar lleno de fe, respeto y felicidad. Gracias T.R.

# Agradecimientos

Para comenzar, quiero agradecer a Dios por darme salud y fortaleza. También a mis padres, Ana y Santiago, y a mis hermanos, Brayan y Kandy, por el apoyo incondicional que me brindaron a lo largo de estos años de carrera y que creyeron en mi más que nadie y nunca me dejaron solo.

A mi mejor amiga Alisson, por su apoyo constante y por haber creído en mí cuando ni yo mismo lo hacía, así como a mis demás amigos que estuvieron presentes en este proceso.

Al ingeniero Xavier Rosero, por su guía durante todo este camino y por los conocimientos que me brindó.

A la ingeniera Brizeida Gámez, por su orientación al momento de redactar este trabajo.

Finalmente, agradezco a todas las personas que me acompañaron y guiaron en este proceso, y que no permitieron que me rindiera.

# Índice general

# Índice de figuras

# Índice de tablas

# Resumen

En este estudio se aborda el desarrollo e implementación de un sistema para el envío de notificaciones remotas de eventos de alto impacto en un robot móvil autónomo. La solución propuesta integra el estándar ROS 2 con comunicación LoRaWAN, facilitando la detección de anomalías en el voltaje, la estabilidad mecánica y el posicionamiento global. Mediante el envío de alertas en tiempo real, se garantiza que un operador reciba datos críticos y eventos de alto impacto de forma inmediata; el diseño se apoya en nodos de procesamiento de datos y un módulo de comunicación de largo alcance y bajo consumo energético, complementados por una interfaz de gestión. Las pruebas realizadas demuestran la eficacia del sistema al identificar condiciones de riesgo como sobrevoltaje, vuelcos accidentales o a su vez pérdida de la señal GPS y salida del área permitida. Esta combinación tecnológica fortalece la supervisión en zonas de baja conectividad, elevando los niveles de seguridad y autonomía del vehículo. Esto permite que el operador conozca el estado actual del robot, aunque no se encuentre en la zona de operación y evitando revisiones de forma presencial, permitiendo un sistema de monitoreo remoto eficiente para la detección de fallas en robots móviles, basado en tecnologías de IoT.

***Palabras clave:*** ROS 2, LoRaWAN, monitoreo remoto, detección de fallas, IoT, robots móviles, notificaciones remotas.

# Abstract

This study examines the development and implementation of a system for sending remote notifications of high-impact events on an autonomous mobile robot. The proposed solution integrates the ROS 2 standard with LoRaWAN communication, facilitating the detection of anomalies in voltage, mechanical stability and global positioning. By sending real-time alerts, it ensures that an operator receives critical data and high-impact events immediately; the design relies on data processing nodes and a long-range, low-power communication module, complemented by a management interface. The tests carried out demonstrate the system's effectiveness in identifying risk conditions such as overvoltage, accidental overturning, loss of GPS signal and departure from the permitted area. This combination of technologies enhances monitoring in areas with poor connectivity, increasing the vehicle's safety and autonomy levels. This allows the operator to know the robot's current status, even when they are not in the operational area, and avoids the need for on-site inspections, enabling an efficient remote monitoring system for faults detection in mobile robots, based on IoT technologies.

***Keywords:*** ROS 2, LoRaWAN, remote monitoring, faults detection, IoT, mobile robots, remote notifications.

# CAPÍTULO I: EL PROBLEMA

## 1.1. Planteamiento del problema

Los avances recientes en el ámbito de la robótica móvil han tenido un efecto importante en la ingeniería contemporánea, estableciéndose como un área de crecimiento tanto en investigación como en implementación tecnológica [1]. Los robots móviles autónomos necesitan sistemas que garanticen, en este escenario, no solamente su funcionamiento y navegación, sino también el monitoreo constante de su estado funcional. No obstante, el prototipo que existe en el laboratorio de Mecatrónica de la Universidad Técnica del Norte [2–4] no incluye un sistema que pueda identificar y avisar automáticamente sobre sucesos críticos mientras se encuentra funcionando, como oscilaciones anómalas, voltaje bajo o pérdida de señal. Esto reduce su capacidad real de autonomía.

La falta de un sistema de monitoreo limita su implementación en contextos no supervisados, aumenta el peligro de daños al equipo e impide prever fallos operativos. Esta falta disminuye la capacidad del robot para operar como plataforma experimental y como instrumento para tareas de campo realizadas autónomamente. La labor del robot estará sujeta a la supervisión constante del operador si no se instala un sistema de alerta apropiado, lo que contradice el principio de autonomía que caracteriza a este tipo de máquinas. Por otro lado, el uso de redes basadas en Internet se usa comúnmente en aplicaciones de monitoreo remoto, su vulnerabilidad ante diversos ataques informáticos genera riesgos importantes para la integridad de los sistemas conectados [5]. En este sentido, tecnologías de comunicación como LoRaWAN ofrecen alternativas más seguras y de bajo consumo energético, adecuada para la transmisión de datos críticos en entornos de difícil conectividad.

Frente a esta problemática, se plantea el diseño e implementación de un sistema para monitorear y notificar anomalías en el robot móvil autónomo, utilizando LoRaWAN para la comunicación y el framework ROS. Esta solución permite una posible identificación de eventos críticos y enviar notificaciones al operador de manera remoto en tiempo real, lo que mejorará la seguridad durante la operación, la autonomía del robot y la fiabilidad del sistema diseñado.

## 1.2. Objetivos

### 1.2.1. General

Desarrollar un sistema de notificaciones de eventos críticos para robots móviles autónomos mediante ROS y LoRaWAN.

### 1.2.2. Específicos

- Determinar los eventos críticos de mayor prioridad, así como las especificaciones técnicas necesarias para notificar mediante ROS y LoRaWAN
- Diseñar un sistema de notificación de eventos críticos basado en ROS y LoRaWAN para robots móviles autónomos.
- Implementar el sistema de notificación de eventos críticos en el robot móvil autónomo.

## 1.3. Alcance

Con este trabajo se pretende desarrollar un sistema de notificación de eventos críticos, utilizando el framework ROS para la gestión de datos y LoRaWAN como medio de comunicación remota de largo alcance; informando al operador de fallas en el robot.

El desarrollo del trabajo iniciará con la identificación de eventos críticos prioritarios y la determinación de las especificaciones técnicas necesarias para el manejo de los aplicativos ROS y LoRaWAN, se prosigue con la creación del sistema que será capaz de reportar y detectar anomalías durante la operación del robot de manera remota, para lograr esto se integrarán sensores específicos, también se establecerá una arquitectura entre la placa principal del robot y una placa auxiliar.

Por último, se realizará la validación del sistema por medio de pruebas experimentales, simulando una situación de falla para evaluar el correcto envío y recepción de alertas al operador de manera remota mediante LoRaWAN. Para este estudio no se contemplan aspectos como el control del robot, tampoco la integración de plataformas basadas en internet.

## 1.4. Justificación

El robot móvil autónomo del laboratorio de Mecatrónica no dispone actualmente de un sistema de notificación de eventos críticos, lo que contradice su principio de operación sin supervisión directa y aumenta el riesgo de fallos no detectados [6]. La implementación de un mecanismo de monitoreo y alerta permitiría identificar anomalías como oscilación, sobreconsumo energético, bajo voltaje o pérdida de señal GPS, todas ellas indicativas de posibles fallas operativas relevantes [7–10].

Para el estudiante de Ingeniería en Mecatrónica, este proyecto representa una oportunidad de integrar conocimientos en comunicación, automatización y sistemas embebidos, desarrollando una solución real a partir de tecnologías como LoRaWAN y ROS. Para la comunidad, el sistema mejora la confiabilidad de plataformas robóticas en entornos sin supervisión continua. A nivel de ingeniería, contribuye al diseño de sistemas autónomos más seguros, robustos y adaptados a escenarios con restricciones de conectividad, utilizando LoRaWAN como alternativa segura frente a redes inalámbricas vulnerables [5].

# CAPÍTULO II: MARCO TEÓRICO

## 2.1. Antecedentes

La automatización es el medio por el cual se aumenta la efectividad operativa en los sectores de servicios, sociales e industriales. Esta habilidad es fundamental porque posibilita conservar los niveles de productividad en circunstancias desfavorables, como la experimentada durante la pandemia. En ese periodo, muchas empresas logran continuar con sus operaciones por medio de procesos automatizados y la supervisión remota [11, 12]. Los robots móviles autónomos se consolidan como una de las áreas de mayor interés en la ingeniería, aplicándose en campos como la medicina, donde asisten a personas con movilidad reducida en tareas cotidianas [4]. Dado que no requieren operación directa, es fundamental incorporar sistemas de monitoreo que permitan detectar anomalías o fallos en tiempo real [13].

En el ámbito de la comunicación remota, se destaca la necesidad de redes seguras, estables y de largo alcance. Tecnologías como LoRa ofrecen transmisiones confiables en rangos de hasta un kilómetro, siendo ideales para entornos con conectividad limitada [14]. En [15], se demuestra la utilidad de robots autónomos en tareas de delivery urbano durante la pandemia, reduciendo el riesgo de contagios mediante entregas sin contacto. La investigación presentada en [16] desarrolla una interfaz para robots hospitalarios con módulos de navegación, trayectoria y visualización 3D, lo cual refuerza la necesidad de sistemas integrados que gestionen información crítica del entorno. De forma complementaria, [17] propone un modelo de automatización con sensores inteligentes basados en IoT, que incrementa la autonomía operativa sin intervención directa.

ROS 2 destaca como plataforma clave para el desarrollo de dispositivos autónomos debido a su flexibilidad y capacidad de integración modular. En [18], se documenta su aplicación en cobots, los cuales interactúan con humanos de forma segura. En el contexto del robot móvil de alto torque, los estudios [2, 3, 19] describen el diseño mecánico, la visión artificial implementada y el sistema electrónico con ROS 2, sentando una base sólida para la incorporación de funciones avanzadas como la notificación remota de eventos críticos.

Respecto a la transmisión de datos, LoRa es reconocido por su eficiencia energética y seguridad, aunque limitado en capacidad de transferencia. Su integración con LoRaWAN mejora notablemente el flujo de información sin sacrificar alcance [20]. En [21], se diseña un sistema multi-robot en ROS 2 con navegación autónoma basada en visión, destacando el papel de la percepción en entornos dinámicos. Finalmente, [22] enfatiza la utilidad de la visión artificial para reconocimiento y diferenciación de objetos, aplicada con éxito en sistemas de control de tránsito.

## 2.2. Bases teóricas

Esta base teórica respalda el desarrollo de un sistema para notificar fallos en robots móviles autónomos mediante la integración de LoRaWAN y ROS 2. Se abordan temas clave como los robots móviles, los tipos de fallas que pueden afectar su funcionamiento, las ventajas de la comunicación inalámbrica de largo alcance con LoRaWAN, y el uso de ROS 2 para el control y procesamiento de datos. Además, se detalla el modelo de integración entre LoRaWAN y ROS 2 a través de MQTT, necesario para la transmisión eficiente de información entre sensores remotos y el sistema robótico.

### 2.2.1. Robots móviles autónomos

Los robots móviles autónomos cada vez son más avanzados, debido a que estos son capaces de navegar y operar en un entorno de trabajo sin intervención humana constante. Estos se desplazan en un entorno físico, interactuando con él para realizar diversas tareas, esto también les permite adaptarse a diversos escenarios, ya que la navegación define la eficiencia, la seguridad y la efectividad con la que el robot puede operar en su entorno [23]. La evaluación del desempeño de robots móviles es un campo crucial para el desarrollo y optimización de estos sistemas como el caso de los robots móviles autónomos de almacenamiento y transporte seguro como se observa en la Fig. 2.1.


**Fig. 2.1:** Robots móviles autónomos usados para almacenamiento seguro [24].

### 2.2.2. Fallas en sistemas robóticos

Dada su creciente complejidad e interacción con ambientes dinámicos, los sistemas robóticos pueden ser vulnerables a múltiples fallos que pueden poner en riesgo su rendimiento, seguridad y confiabilidad. En términos generales, se puede describir una falla como una desviación inaceptable del comportamiento previsto de un componente o del sistema en general. Estas fallas pueden presentarse de diversas formas, entre las más comunes se encuentran propuestas en [25].

#### 2.2.2.1. Fallas en sensores

Información incorrecta, lecturas incoherentes o pérdida de datos esenciales para la percepción del ambiente o la condición interna del robot. Esto puede provocar equivocaciones en las decisiones de navegación o movimientos inestables.

#### 2.2.2.2. Fallas en actuadores

La falla en los motores, servomecanismos o sistemas hidráulicos/neumáticos, que obstaculizan que el robot realice los movimientos o fuerzas requeridos. Esto puede provocar cambios en la trayectoria o incapacidad para llevar a cabo tareas.

#### 2.2.2.3. Fallas en componentes electrónicos y eléctricos

Cortocircuitos, averías en la fuente eléctrica o deterioro de la circuitería que inciden en la comunicación, el procesamiento de datos o la energía de otros subsistemas.

#### 2.2.2.4. Fallas mecánicas

Desgaste, holguras, roturas o atascos en elementos estructurales, engranajes o articulaciones del robot, afectando su precisión y capacidad de movimiento.

#### 2.2.2.5. Fallas de software

Errores en la codificación, fallos en los algoritmos de control, navegación u organización de actividades, que pueden provocar comportamientos imprevistos o la parada del sistema.

### 2.2.3. Comunicación inalámbrica

Para la creación de sistemas que necesitan una comunicación eficaz en ambientes de escaso consumo energético y extenso alcance, las tecnologías LPWAN (Low Power Wide Area Network) han emergido como una solución esencial. LoRaWAN sobresale por proporcionar características que la convierten en la opción perfecta para aplicaciones de Internet de las Cosas (IoT) y sistemas en los que la autonomía y el alcance son esenciales. Como se ha mencionado LoRaWAN es un sistema de comunicación inalámbrica muy eficiente tanto en el rendimiento, conexión, costos, robustez y resistencia. A continuación se revisan varios parámetros, propuestos en [26].

#### 2.2.3.1. Consumo ultra bajo de energía

Una de las particularidades más destacadas de LoRaWAN es su notable disminución en el uso de energía. Esta característica posibilita que los aparatos que funcionan a batería puedan mantenerse en funcionamiento durante periodos extremadamente extensos, con autonomías que pueden exceder los 10 o incluso 15 años. Esta durabilidad de la batería es vital para la factibilidad de despliegues masivos de sensores y aparatos en lugares remotos o de acceso complicado, disminuyendo significativamente los gastos de mantenimiento y sustitución.

#### 2.2.3.2. Conectividad de largo alcance (Wide Area Network)

LoRaWAN se ha diseñado para ofrecer una conexión de larga distancia, superando las restricciones de otras tecnologías inalámbricas en este sentido. En contextos urbanos, su alcance

puede llegar a 3 km, mientras que, en zonas rurales o no urbanas, su rango de acción puede superar los 15 km. Esta habilidad de larga distancia es esencial para aplicaciones que demandan la comunicación entre dispositivos dispersos geográficamente o que funcionan en infraestructuras amplias, como la agricultura de precisión, la vigilancia del medio ambiente o la logística en grandes depósitos.

#### 2.2.3.3. Bajo coste de despliegue y operación

La tecnología LoRaWAN facilita el establecimiento de comunicaciones bidireccionales entre objetos, evitando la necesidad de efectuar despliegues de infraestructura de gran envergadura y complejidad. Esto resulta en una disminución considerable de los gastos relacionados con la instalación y el cuidado de la red. Además, su autonomía respecto a las redes de operadores móviles tradicionales puede proporcionar una mayor adaptabilidad y reducir los gastos operativos a largo plazo.

#### 2.2.3.4. Robustez y resistencia a interferencias

LoRaWAN está concebida para funcionar de manera eficiente en ambientes con interferencias y una amplia variedad de dispositivos desplegados, incluyendo la habilidad de infiltrarse en interiores y entre construcciones. A pesar de que el artículo se centra en los sistemas de seguridad, la solidez de la comunicación es un beneficio general para cualquier aplicación que necesite confiabilidad en la transmisión de información.

### 2.2.4. ROS y ROS 2

ROS, o Sistema de Operación para Robots (Robot Operating System), es una plataforma de software de código abierto que proporciona a los programadores el conjunto de herramientas y bibliotecas requeridos para la creación de aplicaciones robóticas. Open Robotics lo define como un “kit de desarrollo de software” que satisface todas las demandas de un proyecto de robótica, desde la codificación de controladores hasta la aplicación de algoritmos avanzados[27]. Además, dispone de herramientas sólidas para desarrolladores, además de integrarse con Gazebo, lo que lo convierte en todo de código abierto. Como un avance esencial, ROS 2 emergió en su segunda versión, diseñada para superar las falencias de la versión inicial y proporcionar progresos en aspectos como la comunicación, la compatibilidad con múltiples plataformas y la mejora del desempeño.

#### 2.2.4.1. Nodos ROS 2

En ROS 2, se describe a un nodo como una unidad modular responsable de llevar a cabo una función determinada dentro del sistema robótico. Cada nodo opera de manera autónoma, lo que posibilita segmentar el comportamiento global del robot en elementos más reducidos, ver Fig. 2.2, sencillos de administrar, incrementar y reutilizar de acuerdo con las demandas del sistema [28].

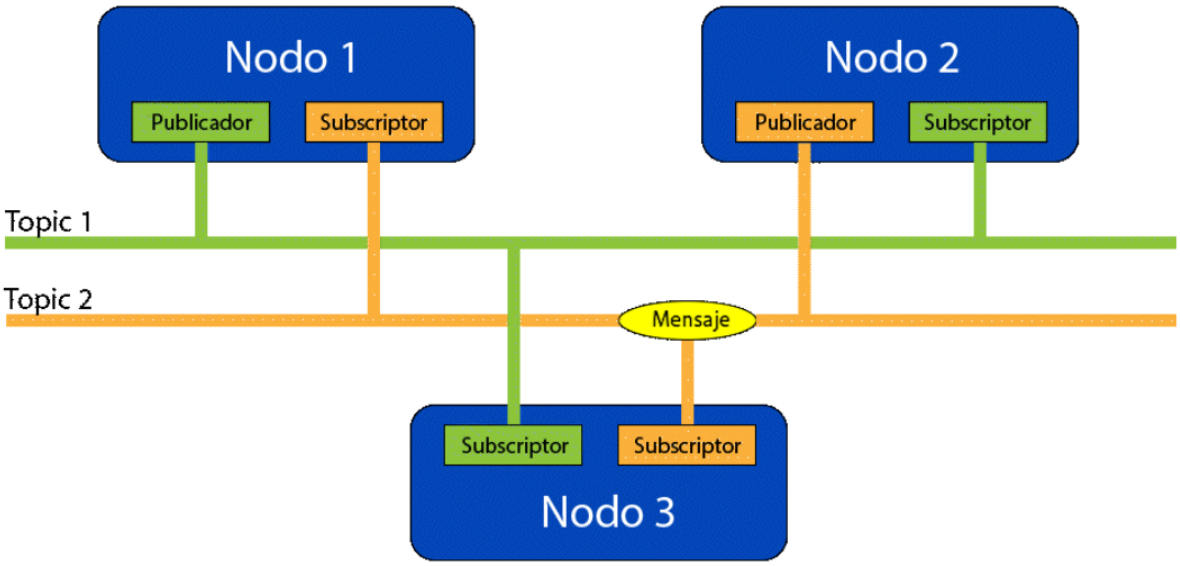



**Fig. 2.2:** Diagrama del funcionamiento de los nodos en ROS 2 [29].

#### 2.2.4.2. Comunicación entre módulos

Estas interfaces se definen utilizando un lenguaje de definición de interfaz (IDL) y se manifiestan principalmente a través de tres tipos y son *mensajes* (.msg), *servicios* (.srv) y *acciones* (.action). *mensajes*se utilizan para que los datos unidireccionales se comuniquen de manera asincrónica, esta función permite que los nodos publiquen información en topics (temas), para que estos reciban información los nodos deben suscribirse [30].

Los *servicios* son los encargados de facilitar la comunicación síncrona de solicitud/respuesta. Un nodo puede "ofrecer" un servicio y otro nodo puede "llamar" a ese servicio, esperando una respuesta inmediata. Las *acciones* ofrecen un sistema para actividades de larga duración que implican un "objetivo" (goal) que se plantea, una "retroalimentación" (feedback) que se obtiene conforme la tarea avanza, y un "resultado" (result) final al finalizar la tarea [30].

### 2.2.5. Integración entre LoRaWAN y ROS

La estrategia de integración se fundamenta en un método modular que vincula la infraestructura de la red de sensores IoT con el ecosistema de ROS, facilitando un intercambio de datos eficientes y bidireccionales [31]. Existen varios modelos de integración, pero analizaremos la arquitectura de integración IoT-ROS basada en MQTT con nodo puente que se define posteriormente.

#### 2.2.5.1. Capa de recopilación de datos IoT

La red de sensores IoT (tecnologías de largo alcance y bajo consumo como LoRaWAN) tiene la tarea de recolectar datos ambientales y de estado (como la temperatura, la humedad, la detección de presencia). Estos sensores están diseñados para transmitir sus datos a un servidor de mensajería simple, como MQTT (Message Queuing Telemetry Transport), este procedimiento se selecciona debido a su eficacia en contextos definidos con su modelo de publicación/suscripción, perfecto para la propagación de datos provenientes de diversas fuentes. La información obtenida de los sensores se divulga en "topics" determinados dentro del servidor [31].

#### 2.2.5.2. TTN (The Things Network)

The Things Network (TTN) es una infraestructura de red a nivel mundial y abierta que se ha creado especialmente para funcionar con redes LoRaWAN. TTN actúa como una red central que posibilita la conexión de dispositivos del Internet de las cosas (IoT) a gran escala, lo cual elimina la obligación de que los usuarios creen y sostengan sus propias redes privadas. Este servicio funciona como el punto de agregación en el que las gateways (o pasarelas) LoRaWAN retransmiten los datos recolectados por los nodos finales, dirigiendo la información hacia las aplicaciones finales para que sean monitoreadas, analizadas y visualizadas [32].

#### 2.2.5.3. Capa de acople ROS-MQTT

Para que el robot opere bajo ROS y pueda obtener la información es necesaria la creación de un nodo como puente de comunicación. Este nodo ROS posee la función de suscriptor MQTT: se vincula al servidor MQTT y se adhiere a los tópicos en los que los sensores IoT comparten sus datos. Cuando el nodo recibe estos datos, los procesa y los transforma en un formato que se alinea con los mensajes de ROS. Este mismo nodo ROS desempeña el papel de publicador ROS, reutilizando los datos obtenidos de MQTT en temas de ROS que son accesibles para los

demás nodos del sistema del robot. Esta revisión garantiza que los datos externos se incorporen sin dificultades en el esquema de comunicación de ROS del robot [31].

#### 2.2.5.4. Capa de procesamiento y decisión robótica

Cuando los datos sensoriales del ambiente se encuentran accesibles en los hilos ROS, los algoritmos de control y procesamiento de datos del robot pueden acceder a ellos. El artículo detalla un "sistema de reglas" que emplea estos datos para producir "alertas o situaciones de interés", lo que facilita al robot la toma de decisiones basadas en información y la adaptación de su comportamiento al ambiente del usuario [31].

### 2.2.6. Tecnologías implementadas en servidores web

Los servidores web necesitan de varias tecnologías que permitan gestionar y sincronizar la comunicación, garantizando eficiencia, seguridad y disponibilidad de los servicios que se mencionan en los siguientes apartados.

#### 2.2.6.1. Linux

Se considera que la distribución Ubuntu 20.04, que es parte del sistema operativo Linux, proporciona el mejor ambiente para la implementación de servidores web. Los estudios indican que, en comparación con Windows, la arquitectura Linux (que se encuentra en las pilas LWMP y LAMP) permite una programación paralela de mejor calidad y tiempos de ejecución más veloces. Los hallazgos experimentales mostraron que este sistema operativo es capaz de lograr un rendimiento "perfecto" (con una precisión del 100 % en los ensayos de carga), al ser superior a su homólogo en lo que respecta a estabilidad bajo estrés y tiempo de respuesta [33].

#### 2.2.6.2. Apache

Apache es el servidor web estándar que se utiliza en la investigación como interfaz de puerta de enlace en la arquitectura del middleware, a menudo vinculado con PHP y pilas de desarrollo como XAMPP. En una arquitectura de tres niveles, su tarea primordial es administrar la conexión entre el backend y el frontend. El análisis se vale de Apache como punto de referencia para cotejar el desempeño del tiempo de respuesta (Response Time) en relación con la interfaz WSGI, examinando su habilidad para gestionar las consultas y la transferencia de archivos en sistemas de aprendizaje electrónico [33].

#### 2.2.6.3. MySQL

En la arquitectura del sistema sugerido, MySQL se utiliza como la base de datos web compartida en el backend para las dos tecnologías analizadas. Actúa como el almacén de datos clave para las aplicaciones web dinámicas, y lo hace sin depender del middleware (Apache o WSGI) ni del sistema operativo (Linux o Windows). El documento lo define como un elemento permanente en las ecuaciones de prueba (WAMP/LAMP y WWMP/LWMP), lo que posibilita que los cambios en los resultados de rendimiento se asignen a los sistemas operativos y servidores web, no al administrador de base de datos [33].

#### 2.2.6.4. PHP

PHP es un lenguaje de programación y una tecnología web dinámica que se emplea en la capa de middleware. El informe concluye que, a pesar de que es más "ligero" en cuanto a la capacidad inicial (35 MB antes de agregar datos, comparado con los 42 MB de Python), su rendimiento en términos de tiempo es menos eficiente cuando se trata de cargas altas. En esta tecnología, se notó una relación indirecta entre el tiempo de respuesta y la capacidad: si la carga de datos crece (particularmente cuando alcanza 1024 MB), el desempeño de PHP se reduce considerablemente, lo que provoca tiempos de respuesta más largos y menos eficientes que los de su contraparte rival [33].

### 2.2.7. Tecnologías adicionales para comunicación M2M

Para garantizar que los dispositivos conectados transmitan datos de manera eficaz y fiable, se utilizan tecnologías complementarias que hacen más fácil la comunicación máquina a máquina (M2M). Los más importantes es explican a continuación.

#### 2.2.7.1. Webhook

Un Webhook es un procedimiento de comunicación asíncrona que posibilita que una aplicación, a través del envío de una solicitud HTTP POST, avise a otra automáticamente sobre la ocurrencia de un evento particular. El Webhook funciona como una "interfaz de empuje" (push) que envía datos en tiempo real a una dirección URL que ha sido configurada anteriormente, lo cual lo distingue de las interfaces tradicionales. En el marco de la integración con plataformas de mensajería como WhatsApp Business API, esta tecnología es esencial para que los mensajes de los clientes se reciban al instante. Esto elimina la necesidad de hacer preguntas al servidor continuamente y facilita una administración de datos más eficaz y escalable [34].

#### 2.2.7.2. Firebase

Firebase es una plataforma de desarrollo de aplicaciones (Backend as a Service) que tiene como objetivo simplificar la creación de soluciones web y móviles, utilizando una infraestructura en la nube. La Realtime Database, una base de datos NoSQL que guarda la información en formato JSON y posibilita que todos los usuarios conectados se sincronicen automáticamente,uno de sus componentes principales [35]. Además, el documento subraya su habilidad para persistir, asegurando que las aplicaciones puedan seguir funcionando y respondiendo incluso si la conectividad es escasa o inexistente.

### 2.2.8. Código de colores y señalizaciones

El empleo de colores en la señalización tiene como objetivo conseguir una comprensión fácil y rápida con el fin de evitar riesgos. En este sistema, se utiliza el color rojo para señalar la prohibición o el paro, así como para distinguir los equipos contra incendios, lo que en una interfaz se traduce en errores críticos o acciones que interrumpen un procedimiento. El color amarillo es una señal de advertencia de riesgo, y se utiliza para demarcar zonas; en el diseño digital, actúa como un aviso de precaución o verificación previo a un posible error. En último lugar, el color verde simboliza una situación segura y se emplea para indicar rutas de evacuación o áreas de ayuda. En contextos interactivos, esto transmite éxito, procesos que han terminado sin inconvenientes o navegación segura [36].

### 2.2.9. Usabilidad

La usabilidad se refiere a la habilidad de un software para ser entendido, aprendido y utilizado, siendo atractivo para el usuario en condiciones específicas. La norma ISO 9241-11 define específicamente como el nivel en que un producto posibilita que usuarios específicos logren metas particulares de manera eficaz, eficiente y satisfactoria en un contexto de uso determinado. Este rasgo es crucial en la creación de sistemas interactivos para asegurar que el producto final no solo opere adecuadamente, sino que además satisfaga las expectativas y requerimientos del usuario [37].

# CAPÍTULO III: MARCO METODOLÓGICO

## 3.1. Enfoque y tipos de investigación

La presente investigación se centra en un enfoque ingenieril, ya que busca desarrollar, implementar y validar un sistema de notificación remota de fallos en un robot móvil autónomo [38]. Esta es una investigación de campo, porque se lleva a cabo en un entorno real de operación del robot, donde se aplican y prueban directamente los conocimientos teóricos y tecnológicos para resolver un problema específico relacionado con la detección y comunicación de fallos en tiempo real [38].

Se desarrollará una investigación aplicada, ya que se busca resolver una necesidad específica relacionada con la notificación remota de fallos en sistemas robóticos [39]. A continuación, se implementará una fase descriptiva para caracterizar el comportamiento del sistema en distintos escenarios [38]. Posteriormente, se ejecutará una base experimental en pruebas tanto en entornos controlados como en condiciones reales, con el fin de validar el funcionamiento del sistema de notificación basado en ROS y LoRaWAN [40]. Este proceso será reforzado con una revisión documental, en la que se recopilarán antecedentes, bases teóricas y experiencias previas en la aplicación de tecnologías similares [41]. Adicionalmente, para evitar comprometer el equipo principal con actualizaciones o reprogramaciones futuras para lograr compatibilidad con distintos protocolos de conexiones tal como explica el fabricante en [42], se optará por descartar adaptadores basados en el chip CH340 para la comunicación.

## 3.2. Diseño de la investigación

La investigación se estructura en tres fases metodológicas siendo estas diseño, implementación y validación, con la finalidad de cumplir con los objetivos del trabajo de integración curricular. Todos los datos obtenidos serán registrados y documentados para el análisis y estudio del caso.

### 3.2.1. Fase 1: Análisis e identificación de eventos críticos y requerimientos técnicos

Se definen los fallos que deben ser detectados y notificados por el sistema, así como los sensores y módulos necesarios. En esta etapa se establecen las bases técnicas del proyecto tanto en el manejo de software como del robot.

***Actividad 1.1: Búsqueda y estudio de información sobre el manejo de ROS 2 y LoRaWAN con sus diversas funciones.***

Recopilación de datos técnicos acerca del funcionamiento de ROS 2 y LoRaWAN, esto con el fin de detectar y reconocer el rango de comunicación, la capacidad de envío de información y sus limitaciones, como la latencia y el alcance de estas mismas.

***Actividad 1.2: Investigación de los eventos críticos y umbral que dispone el robot actualmente.***

Identificación de los eventos críticos que impactan en el funcionamiento del robot, tales como pérdida de comunicación o conducta inusual. Se establecen los niveles límite de funcionamiento que ponen en marcha las alertas en el sistema, basándose en el comportamiento presente del prototipo robótico.

***Actividad 1.3: Evaluación de los sensores y su compatibilidad con la placa Jetson Orin nano y placa auxiliar***

Evaluación de los sensores adquiridos tras la investigación respectiva, esto con el fin de determinar su compatibilidad con la placa principal, ya que esta no permite comunicación con dispositivos que incorporen el protocolo CH340 sin realizar modificaciones.

***Actividad 1.4: Adquisición de los componentes y dispositivos que son necesarios para el desarrollo e implementación del sistema.***

Adquisición de todos los componentes que son necesarios para el diseño de esta interfaz, en su selección se consideran todos los parámetros establecidos para evitar la existencia de errores tanto en comunicación, compatibilidad, entre otros antes mencionados.

### 3.2.2. Fase 2: Diseño e implementación del sistema de notificación basado en LoRaWAN y ROS

Se desarrolla la arquitectura del sistema integrando sensores, nodos ROS y comunicación LoRaWAN. Todo se configura para detectar y enviar alertas de forma eficiente de manera simulada y con situaciones especiales.

***Actividad 2.1: Elaboración de un diagrama de bloque referente al sistema de notificación***

***evaluando: conexiones, componentes, transmisión de datos y comunicación.***

Construcción de un esquema de bloques que establece la interacción entre los elementos del sistema, que abarcan sensores, microcontroladores, nodos ROS y el módulo LoRaWAN. El diseño abarca la lógica de transferencia de información, a su vez la administración de alertas a distancia.

***Actividad 2.2: Planteamiento de varias alternativas de solución al problema.***

Planteamiento de diversas alternativas de diseño para el sistema de alertas, teniendo en cuenta diferentes sensores, compatibilidad, estructuras de comunicación, modelos de software y protocolo de comunicación. Las opciones se analizan basándose en la factibilidad, el consumo de energía, la facilidad de implementación y la cobertura de comunicación.

***Actividad 2.3: Selección de la mejor alternativa de solución.***

Selección de la opción más apropiada para el sistema, dando preferencia a la que proporcione mayor estabilidad, compatibilidad con ROS y eficiencia en la alerta de fallos. Esta decisión se toma tras el análisis de consumo energético, adaptabilidad y compatibilidad.

***Actividad 2.4: Cálculo de las variables a medir con sus respectivos puntos críticos.***

Proceso de toma de datos sin codificación de los respectivos sensores y seleccionar los datos de mayor importancia que serán censados y recolectados respectivamente en este caso el Balanceo (acelerómetro), también Latitud, Longitud y Velocidad (GPS).

***Actividad 2.5: Comprobación, mediante la placa auxiliar, de la correcta medición de los datos y notificación de fallos detectados.***

Ejecución de un diagnóstico con ayuda de la placa auxiliar, con ellos se logrará ver que datos llegan a esta misma, la latencia y el orden respectivo.

***Actividad 2.6: Implementación de la programación y comunicación entre modulo LoRaWAN con la placa auxiliar.***

Implementación de la programación tanto en el módulo LoRaWAN como de la placa auxiliar, configuraciones respectivas para que esta se comunique con la placa auxiliar para que puedan enviar y recibir información entre ellas.

***Actividad 2.7: Vinculación con el servidor remoto mediante webhook y apache, mediante la aplicación RealVNC Viewer***

Ejecución del proceso de vinculación entre el servidor remoto y los datos obtenidos de TTN (The Things Network) para ser almacenados.

***Actividad 2.8: Creación de base de datos, unión con firebase.***

Creación de la base de datos usando la plataforma de servicio en la nube firebase, con el fin de recibir datos en tiempo real desde el servidor remoto y almacenarlo en la nube.

***Actividad 2.9: Verificación del envío de datos y recolección en firebase.***

Verificación en la plataforma firebase de la correcta recepción y envío de información desde el servidor remoto, con el fin de detectar posibles fallas o errores en la comunicación y evitar pérdida de información.

***Actividad 2.10: Diseño de la interfaz gráfica de notificaciones remotas en base a los datos que recopila en tiempo real.***

Desarrollo de la interfaz gráfica, teniendo presente el código de color, usabilidad, claridad visual, feedback (retroalimentación), simplicidad y HMI. Esta misma es capaz de tomar los datos directamente de firebase, procesarlos y mostrarlo al operador.

***Actividad 2.11: Testeo del aplicativo mediante pruebas simuladas.***

Realización de pruebas con la finalidad de verificar todos los parámetros establecidos, detección de posibles fallos, reajustes e interactividad con esta misma comprobando su funcionamiento y descarte de posibles errores y objetos y funciones innecesarias.

***Actividad 2.12: Creación del nodo ROS y programación en la placa Jetson Nano.***

Creación de los respectivos nodos en la placa principal en el sistema Ubuntu con ROS 2 Humble, se debe adaptar la programación de cada sensor a esta nueva arquitectura, al ser Python no existen cambios mayores. Los sensores son nodos publicadores y la placa auxiliar recibirá datos de manera serial por un nodo suscriptor.

***Actividad 2.13: Validación del enlace entre la placa, microcontrolador y nodo ROS mediante prueba.***

Validación de los enlaces, frecuencia, latencia, envío y recepción de información, esto asegura que todos los dispositivos tengan buena comunicación y decodifiquen de manera correcta los datos según el protocolo para cada caso.

***Actividad 2.14: Simulaciones en entornos de prueba (sin uso del robot)***

Evaluación del sistema en diversos entornos, sin montarlo en el robot, esto permite la detección de fallos como cables rotos, puertos averiados, mala recepción o sincronización entre los dispositivos.

### 3.2.3. Fase 3: Validación del sistema de notificación remota y documentación

Se realizan pruebas con fallos simulados para verificar la correcta detección y envío de notificaciones. Se ajusta el sistema según los resultados obtenidos y su documentación respectiva.

***Actividad 3.1: Montaje del sistema en el robot***

Integración de todo el sistema dentro del robot móvil autónomo, esto incluye todos los sensores, módulos, microcontroladores, placa principal y el respectivo cableado, asegurando el

montaje fijo y adecuado de los componentes con su correcta sujeción evitando la interferencia entre elementos ya existentes.

***Actividad 3.2: Configuración de la comunicación serial entre placa auxiliar y Jetson Orin nano***

Configuración de la comunicación serial entre la placa auxiliar y la Jetson Orin Nano. Se determinan los parámetros como velocidad de transmisión, puerto de conexión y control de errores, garantizando una transferencia segura de los datos del sistema.

***Actividad 3.3: Integración y adaptación de los nodos ROS 2 en el robot para recibir datos en tiempo real.***

Adaptación de los nodos ROS 2 previamente desarrollados al entorno operativo del robot. Se prueba su ejecución en tiempo real, verificando que los datos de sensores se lean, procesen y publiquen de manera adecuada para el envío de notificaciones.

***Actividad 3.4: Sincronización de los mensajes de notificación de eventos críticos y verificación del correcto envío ante anomalías.***

Sincronización de los mensajes enviados por el robot ante la detección de eventos críticos. Se comprueba que cada fallo genere una notificación clara, que esta llegue correctamente al receptor y que no exista pérdida de información durante la transmisión.

***Actividad 3.5: Prueba del funcionamiento simulando fallos para comprobar la reacción del sistema.***

Simulación de fallos en tiempo real como pérdida de señal o inclinaciones peligrosas. Se observa la reacción del sistema y se verifican los tiempos de respuesta, garantizando que las alertas sean generadas y enviadas oportunamente al operador.

***Actividad 3.6: Evaluación del alcance, latencia y estabilidad en la comunicación mediante LoRaWAN***

Evaluación de la cobertura, latencia y estabilidad del enlace LoRaWAN entre el robot y el receptor. Se realizan pruebas en diferentes distancias y condiciones ambientales para validar el alcance efectivo del sistema de notificación.

***Actividad 3.7: Registros de los resultados, ajustes y validación final del sistema***

Análisis de los registros de varios resultados de todas las pruebas realizadas. Se documentan los fallos detectados, la efectividad de las notificaciones, así como los ajustes realizados al sistema. Estos datos permiten validar técnicamente el funcionamiento general.

***Actividad 3.8: Reajuste en caso de algún error o anomalía.***

Implementación de los reajustes necesarios en el sistema, corrigiendo errores detectados durante la validación. Se mejora la lógica de los nodos, la configuración del hardware o la calibración del módulo LoRaWAN si fuera necesario.

***Actividad 3.9: Documentación y preparación para la presentación.***

Elaboración del documento final del proyecto, incluyendo diagramas, configuraciones, código fuente y resultados de pruebas. Esta información se organiza para su presentación formal como parte del trabajo de integración curricular.

# CAPÍTULO IV: RESULTADOS

En este capítulo se presentan los resultados obtenidos durante la puesta en marcha y las pruebas del sistema de notificación remota de fallos en robots móviles mediante la combinación de LoRaWAN y ROS. Se explican los métodos utilizados para validar el sistema, la información recolectada en las pruebas y el análisis de su rendimiento con respecto a los objetivos establecidos en este estudio. El objetivo de este apartado es mostrar, de modo específico, cuán efectivo y fiable es el sistema, evidenciando cómo la fusión de la plataforma ROS 2, los módulos de sensores y la comunicación LoRaWAN permite detectar y notificar eventos críticos en tiempo real. Se incorporan resultados cualitativos y cuantitativos que posibilitan la evaluación de la aplicación del sistema en entornos controlados, además de comprobar si se han cumplido los objetivos particulares. Cuando se habla de pruebas en robots o prototipos robótico se debe recopilar toda la información posible durante el testeo [43]. De este modo, se logra una perspectiva precisa de cómo opera el sistema y de las ventajas que ofrece para la vigilancia a distancia de robots móviles, estableciendo así los cimientos para las conclusiones finales del estudio.

## 4.1. Características del sistema desarrollado

El sistema de notificación remota de eventos críticos para el robot móvil autónomo se diseña bajo los principios de eficiencia, modularidad y bajo consumo. A continuación, se destacan las características principales:

- **Notificación en tiempo real:** El sistema permite la transmisión inmediata de alertas cuando se detectan anomalías en los sensores integrados, mejorando la seguridad del robot durante su operación autónoma.
- **Integración modular mediante ROS 2:** La arquitectura basada en nodos ROS permite escalar el sistema o modificarlo según nuevas necesidades o sensores adicionales.

- **Comunicación LoRaWAN:** Se emplea un protocolo de largo alcance y bajo consumo, ideal para entornos sin cobertura de internet ni redes convencionales.
- **Interfaz de usuario remota (HMI):** Permite al operador monitorear en tiempo real el estado del robot y recibir notificaciones visuales e informativas desde un punto de control externo.
- **Compatibilidad con el microcomputador:** El sistema es implementado como unidad principal de procesamiento, lo que permite un alto rendimiento y flexibilidad.

Tras el análisis de las variables disponibles en el robot, se seleccionan tres para su monitoreo, entre ellas están *balanceo*, *GPS* y *voltaje*.

## 4.2. Sistema implementado

Entre las diferentes alternativas, se elige el sistema que cumple con todas las características técnicas requeridas para la implementación del proyecto, como se muestra en la Fig. 4.1.

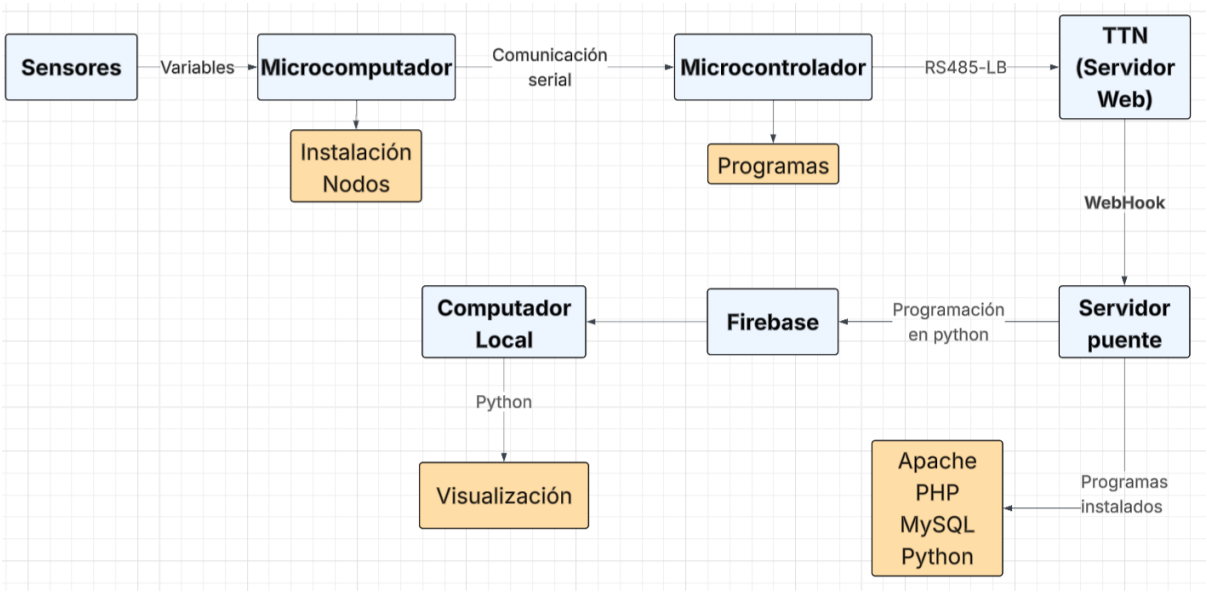



**Fig. 4.1:** Arquitectura del sistema.

Esta alternativa presenta el mayor nivel de eficiencia, compatibilidad y viabilidad para el desarrollo del sistema de notificación remota.

## 4.3. Arquitectura

La arquitectura del sistema describe la estructura general y la interacción entre los diferentes componentes de hardware y software. En esta sección se presenta la organización de los elementos que permiten el envío y recepción de información, su procesamiento y la comunicación para el sistema propuesto.

### 4.3.1. Sistema de nodos

El código fuente desarrollado para el presente proyecto se encuentra alojado en un repositorio público de la plataforma GitHub en https://github.com/VIRTUALTONY/Notificacion_Fallos_Robot.git.

Los archivos que integra la lógica de programación están organizados en el directorio principal `Notificacion_Fallos_Robot`. La mayoría de programas están instalados junto con los nodos dentro del microcomputador, mientras que otros en el microcontrolador, servidor web o servidor puente. En la siguiente sección se describen los archivos asociados a cada sensor o dispositivo utilizado en la arquitectura del sistema.

### 4.3.2. Unidades de control y procesamiento

Los componentes que se encargan de implementar la lógica del sistema, manejar la comunicación entre los diferentes módulos y procesar los datos que provienen de los sensores son las unidades de control y procesamiento. En esta sección se muestran cada uno de los elementos que se emplean para estas tareas, como el microcontrolador y el microcomputador. Estos dispositivos hacen posible la obtención, procesamiento y transmisión de información en el interior de la arquitectura del sistema.

#### 4.3.2.1. Microcomputador

La NVIDIA Jetson Orin Nano es una computadora embebida de alto rendimiento que se ha creado para ser utilizada en aplicaciones de inteligencia artificial, robótica y visión por computadora en el borde (edge computing). Esta placa combina CPU, GPU y memoria en un módulo compacto único, esta información junto a la imagen referente se encuentra en [44], lo que posibilita la ejecución de algoritmos sofisticados de aprendizaje automático y procesamiento en tiempo real con bajo consumo energético, ver Fig. 4.2.

**Tabla 4.1:** Especificaciones de la placa SBC Jetson Orin Nano [44].

| **NVIDIA Jetson Orin Nano 8 GB** | |
|---|---|
| **Característica** | **Detalle** |
| CPU | 6-core Arm Cortex-A78AE v8.2 64-bit, 1.5 MB L2 + 4 MB L3 |
| GPU | NVIDIA Ampere con 1024 CUDA cores y 32 Tensor Cores |
| Memoria | 8 GB LPDDR5 128-bit, 68 GB/s |
| Almacenamiento | Ranura microSD; soporte NVMe externo |
| Consumo energético | 7 W – 15 W |
| Interfaces USB | 4× USB 3.2 Gen2; 1× USB-C |
| Conectividad | 1× Gigabit Ethernet |
| Cámara | 2× MIPI CSI-2 (22 pines) |
| Display | DisplayPort 1.2 |
| Expansión | Header de 40 pines GPIO |
| Dimensiones | 100 mm × 79 mm × 21 mm |

El microcomputador es seleccionado porque cumple con las características explicadas en la sección 4.1.


**Fig. 4.2:** Placa SBC (Single Board Computer) Jetson Orin Nano [44].

- **Nodo 1 "*publicador*":** En la Fig. 4.3 se presenta el diagrama de bloques de la arquitectura de funcionamiento del nodo *(tm171_roll_node)* relacionado con el acelerómetro. Este nodo se encarga de publicar los datos en formato de mensaje correspondientes al *balanceo* en el topic *(/tm171/roll)* del paquete *(tm171_accel)*.


**Fig. 4.3:** Diagramas de bloques del nodo publicador acelerómetro.

- **Nodo 2 "*publicador*":** Para el sistema de posicionamiento global se implementa un segundo nodo como se observa en Fig. 4.4 denominado *(gps)*, encargado de publicar la información en el topic *(/gps/status)* dentro del mismo paquete *(tm171_accel)*. Esta separación permite evitar conflictos o choque de datos entre los distintos flujos de información del sistema.


**Fig. 4.4:** Diagramas de bloques del nodo publicador gps.

- **Sistema de nodo suscriptor:** Como se observa en la Fig. 4.5, se representa el diagrama de bloqueos del nodo suscriptor *(roll_subscriber_node)* que se registra en DDS y es el encargado de suscribirse a los topics *(/tm171/roll y /gps)* especificados, los escucha, recibe el mensaje, ejecuta un callback, en este punto los datos se convierten en variables usables en el código.

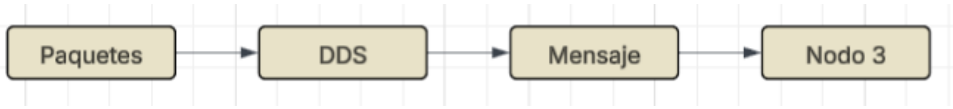



**Fig. 4.5:** Diagrama de bloques del nodo suscriptor.

La programación correspondiente a cada caso respectivo se encuentra en los archivos del repositorio con los nombres `Nodo_GPS.py` y `Acc_Balanceo.py`, que son los encargados de realizar la interfaz de comunicación con el hardware, procesando los flujos de datos entrantes mediante algoritmos de conversión de tipos. De este modo, la información sensorial es codificada en formatos decimales y enteros, asegurando la integridad de los datos durante la etapa de procesamiento digital. El programa del nodo suscriptor se encuentra en el directorio

con el nombre de `Nodo_Suscriptor.py` que es el encargado de tomar los datos de los diferentes nodos en términos de 10 caracteres, unirlos con una coma y posteriormente enviarlos al microcontrolador en un intervalo de 1 segundo.

#### 4.3.2.2. Microcontrolador

El Arduino Uno R3, que se basa en el microcontrolador ATmega328P, es una plataforma de desarrollo que se emplea mucho en prototipos electrónicos y sistemas embebidos. En el sistema propuesto, este dispositivo es el encargado de recopilar datos de los sensores y realizar un primer procesamiento de la información, posterior se envíe a otros módulos del sistema[45], como se muestra en Fig. 4.6.

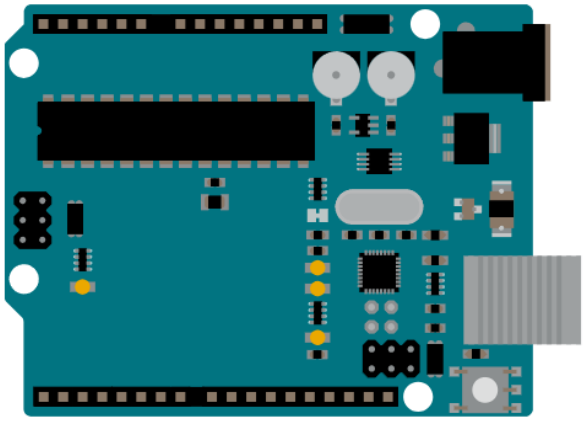


**Fig. 4.6:** Microcontrolador Arduino UNO R3 [45].

- **Configuración:** Para el manejo ágil y preciso del trabajo se realiza la configuración de los diferentes dispositivos con el fin de que no exista errores como colisiones de datos, perdida de información, o datos corruptos. En la Tabla 4.2 se procede a mostrar la configuración del microcontrolador.

**Tabla 4.2:** Configuración del microcontrolador

| Parámetro | Valor / Descripción |
|---|---|
| Microcontrolador | Arduino UNO (ATmega328P) |
| Comunicación serial 1 (Hardware) | 9600 baudios (Pines 0 RX, 1 TX) |
| Comunicación serial 2 (Software) | 9600 baudios (Pines 2 RX, 3 TX) |
| Intervalo de envío (Trigger) | 15,000 ms (15 segundos) |
| Ventana de procesamiento | 10,000 ms (10 segundos) |
| Pin del disparador (Trigger) | Pin digital 13 |
| Duración del pulso trigger | 100 ms |
| Entrada del sensor de voltaje | Pin analógico A0 |
| Muestreo de voltaje | Promedio de 100 lecturas (intervalo 1 ms) |
| Resolución del buffer | 10 caracteres por variable (Relleno con ceros) |
| Protocolo de control | Comandos AT y caracteres de control (1-5) |

Con esta configuración se garantiza un óptimo funcionamiento en el procesamiento de la información.

- **Algoritmo de adquisición y transmisión de datos del nodo:** Para el manejo ágil y preciso del trabajo se realiza la configuración de los diferentes dispositivos con el fin de que no exista errores como colisiones de datos, perdida de información, o datos corruptos.

  En el repositorio la programación del microcontrolador está ubicada con el nombre de `Arduino.ino` que está basado en el trabajo [46] que es la encargada de tomar los datos del puerto serial que envía el nodo suscriptor, medir el voltaje del puerto *A0*, ponerlo en una cadena de 10 caracteres, enviar el *trigger* cada 15 segundo para esperar la respuesta del módulo RS485-LB para enviarlos a cada canal de este mismo, tal como se puede observar en la Fig. 4.7.

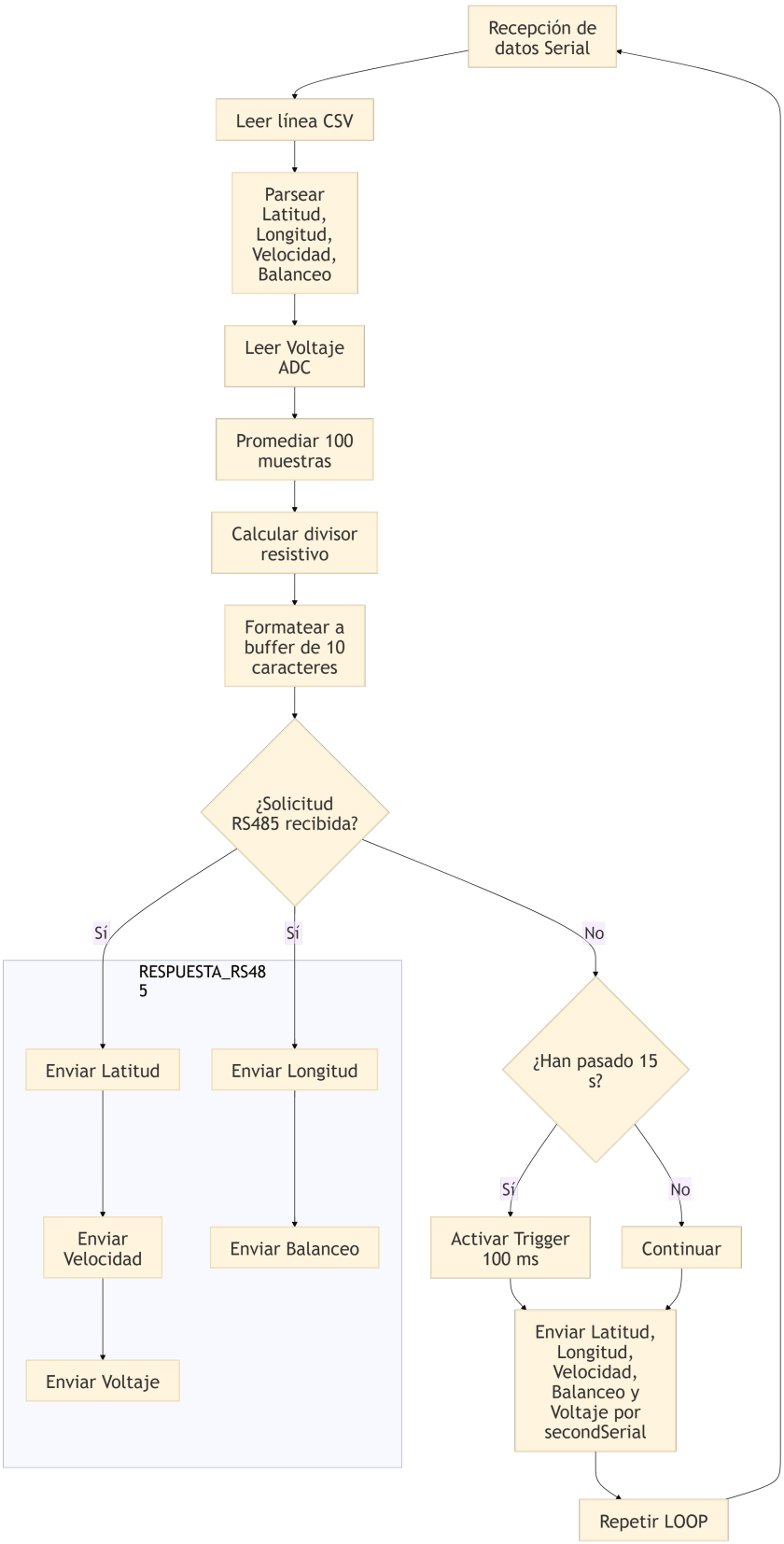



**Fig. 4.7:** Diagrama de flujo de la programación del microcontrolador.

Como se observa en caso de no recibir la solicitud cada 15 segundos se envía un disparador para forzar al módulo RS485-LB al envió de esta misma.

### 4.3.3. Sensores

El sistema de monitoreo requiere sensores de alta precisión, que además sean compatibles con el microcomputador, eficientes, compactos y reemplazables en caso de fallas mecánicas o electrónicas.

#### 4.3.3.1. Acelerómetro

El sensor seleccionado para cumplir con la función del acelerómetro es el AHRS/IMU TM171 de SYD Dynamics[47] es un sistema de referencia de actitud y dirección (AHRS) que incluye una unidad de medición inercial de nueve ejes, la cual está formada por un magnetómetro triaxial, un giroscopio y un acelerómetro. Este sensor posibilita no solo la estimación confiable de los ángulos de yaw, pitch y roll. El módulo está concebido para aplicaciones dinámicas, con un error bajo en la estimación de orientación y una baja deriva angular; por lo tanto, es apropiado para plataformas móviles robóticas. Ofrece interfaces de comunicación tipo USB C y UART, lo que permite integrarlo con sistemas embebidos y ambientes que funcionan con ROS 2.

**Tabla 4.3:** Especificaciones del transducerM 9-Axis AHRS / IMU [47].

| **Característica** | **Detalle** |
|---|---|
| Tipo de sistema | AHRS con IMU de 9 ejes |
| Sensores integrados | Acelerómetro (3 ejes), giroscopio (3 ejes), magnetómetro (3 ejes) |
| Variables estimadas | Roll, pitch, yaw, cuaterniones y datos IMU crudos |
| Interfaces de comunicación | UART serial y USB tipo C (Virtual COM Port) |
| Compatibilidad de sistemas | Linux (/dev/ttyACMx) y Windows (COMx) |
| Frecuencia de salida | Hasta 800 Hz |
| Rango del giroscopio | $\pm$1000°/s |
| Resolución del giroscopio | 0.01°/s |
| No linealidad del giroscopio | <0.2 % FS |
| Rango del acelerómetro | $\pm$10 g |
| Error dinámico roll/pitch (RMS) | <1.0° |
| Deriva de yaw | Aprox. 2.6° cada 25 minutos |
| Consumo de corriente | 80 mA a 5 V |
| Configuración | Software gráfico ImuAssistant (Windows) |
| Protección eléctrica | Protección ESD en líneas TXD y RXD |
| Calidad de datos | Indicador QoS integrado |

La Tabla 4.3 resume sus especificaciones técnicas más relevantes y en la Fig. 4.8 el sensor. Se selecciona por cumplir con la compatibilidad y lo demás mencionado en la Sección 3.2.1 en la Actividad 3.2.1 .


**Fig. 4.8:** TransducerM 9-Axis AHRS / IMU [47].

#### 4.3.3.2. Conversor RS485-LB de Dragino

El convertidor RS485-LB de Dragino es un nodo final que se creó para incorporar dispositivos con interfaz UART o RS485 en redes inalámbricas que utilizan el protocolo LoRaWAN. Este equipo es reconocido por su resistencia, ya que tiene una batería de Li-SOCl2 de 8500 mAh y una carcasa con certificación IP67 de impermeabilidad, lo cual asegura su autonomía en aplicaciones externas durante varios años. Ofrece la posibilidad de leer sensores a través de comandos AT específicos para el protocolo Modbus (RS485) o comunicación serie TTL, como las versiones previas, lo que brinda versatilidad para la supervisión remota en entornos industriales [48]. La conexión usa el protocolo UART, como se muestra en la Fig. 4.9.

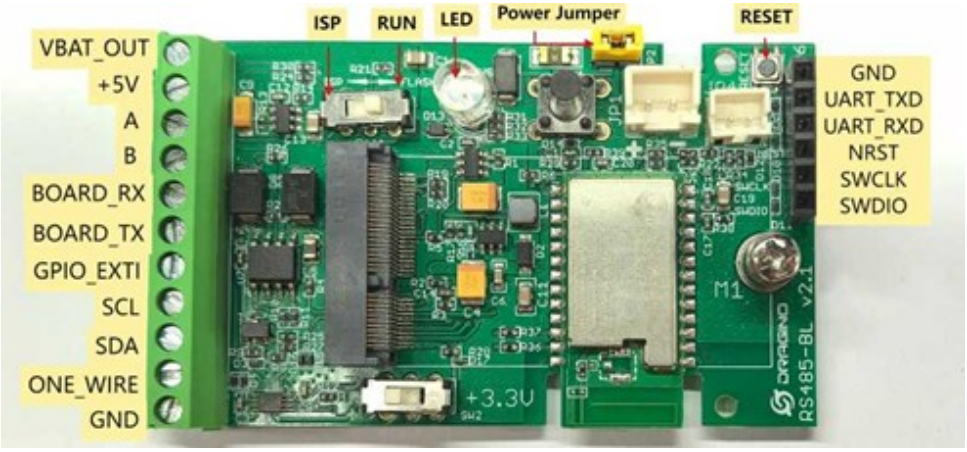



**Fig. 4.9:** Distribución y configuración de los terminales de conexión del RS485-LB [48].

Es esencial que el RS485-LB se aplique en este proyecto, pues tiene la capacidad de actuar como un 'puente' tecnológico. Este dispositivo posibilita beneficiarse de la exactitud de sensores industriales que ya están presentes y que operan con el estándar RS485, a diferencia de otros nodos que exigen sensores analógicos directos. Su diseño "plug-and-play" y su gran resistencia ambiental disminuyen considerablemente los gastos de mantenimiento y hacen más sencilla la infraestructura requerida para recoger datos en el campo.

En la Tabla 4.4 se presenta la configuración de los comandos AT empleados en el módulo RS485-LB.

**Tabla 4.4:** Configuración de Comando AT.

| Comandos AT | |
|---|---|
| AT+COMMAND1=31 ,0 | AT+SEARCH1=0,0 |
| AT+DATACUT1=10,2,1~10 | AT+CMDDL1=0 |
| AT+COMMAND2=32 ,0 | AT+SEARCH2=0,0 |
| AT+DATACUT2=10,2,1~10 | AT+CMDDL2=0 |
| AT+COMMAND3=33 ,0 | AT+SEARCH3=0,0 |
| AT+DATACUT3=10,2,1~10 | AT+CMDDL3=0 |
| AT+COMMAND4=34 ,0 | AT+SEARCH4=0,0 |
| AT+DATACUT4=10,2,1~10 | AT+CMDDL4=0 |
| AT+COMMAND5=35 ,0 | AT+SEARCH5=0,0 |
| AT+DATACUT5=10,2,1~10 | AT+CMDDL5=0 |
| AT+PWORD=000000 | AT+MOD=2 |

Es importante mencionar que la contraseña se edita para agilizar el trabajo, pasa de `123456` a `000000`, para futuras configuraciones.

#### 4.3.3.3. El módulo GPS

Se selecciona el módulo u-blox NEO-6M V2 como se observa en la Fig. 4.10 como sensor de posicionamiento debido a s u alta sensibilidad y bajo consumo energético, factores críticos para garantizar la autonomía del nodo LoRaWAN en exteriores. Su integración es ideal para este proyecto ya que su comunicación nativa es de nivel lógico TTL (Transistor-Transistor Logic), lo que lo hace directamente compatible con el microcontrolador Arduino y facilita la etapa de depuración mediante un conversor USB-TTL [49]. Este adaptador actúa como un puente de comunicación que permite la lectura directa de las sentencias NMEA desde una computadora, asegurando que los datos de latitud, longitud y velocidad sean válidos antes de ser procesados por el algoritmo de derivadas y enviados a través del conversor LoRaWAN.

**Tabla 4.5:** Especificaciones del módulo u-blox NEO-6M V2 [49].

| Característica | Detalle |
|---|---|
| Modelo | u-blox NEO-6M V2 |
| Interfaz de comunicación | UART (Serial) |
| Velocidad por defecto | 9600 baudios |
| Voltaje de operación | 3.3 V – 5 V (Incluye regulador) |
| Protocolo de datos | NMEA 0183 |
| Frecuencia de actualización | 1 Hz (Hasta 5 Hz configurable) |
| Precisión de posición | Aprox. 2.5 metros |

La Tabla 4.5 presenta las principales especificaciones técnicas y configuraciones del módulo

GPS utilizado en el sistema.

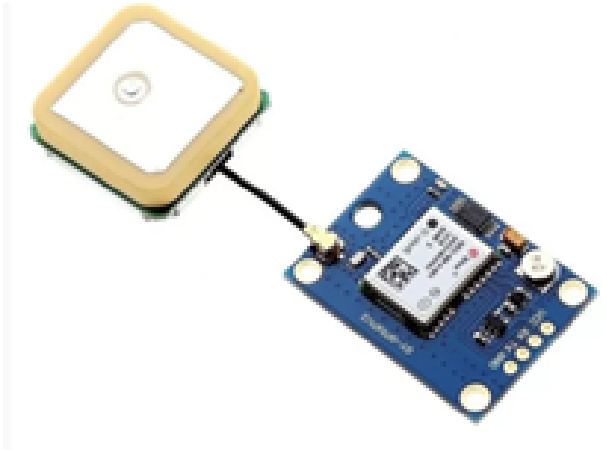


**Fig. 4.10:** Módulo u-blox NEO-6M V2 [49].

La lógica de programación correspondiente se encuentra en el archivo `Nodo_GPS.py`, el cual gestiona la comunicación con el módulo GPS y realiza el procesamiento inicial de los datos recibidos. De esta manera, la información de posicionamiento es convertida y estructurada para su posterior utilización dentro de la arquitectura del sistema.

#### 4.3.3.4. Divisor de voltaje

Para llevar a cabo la medición del nivel de las baterías que es de 24V a 18 Ah se diseña un divisor de voltaje que bajo el nivel en un 89.1 % pasando a 2.61V teóricos. Con el fin de proteger el microcontrolador, también se le incorpora un diodo Zener de 5.1v para mayor protección en caso de un pico de voltaje inesperado como se observa en la Fig. 4.11.

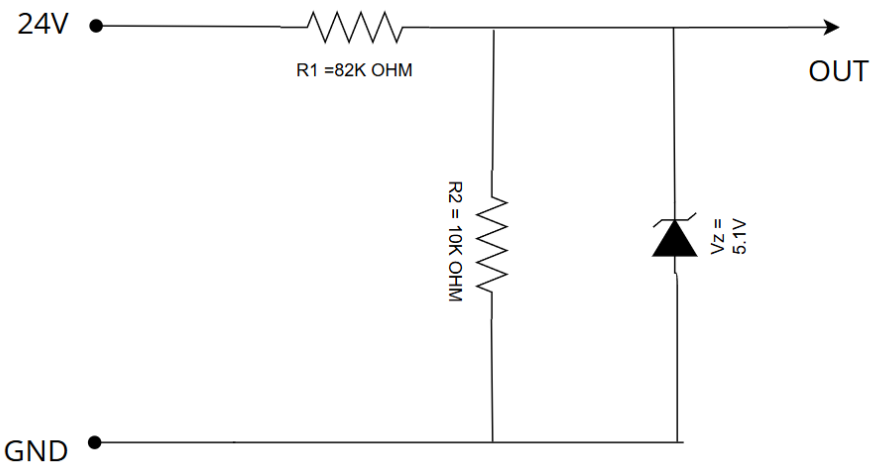



**Fig. 4.11:** Divisor de voltaje.

Esto asegura que la tensión de salida se reduzca correctamente sin que el microcontrolador sufra daño al momento de realizar las respectivas mediciones.

### 4.3.4. Comunicación entre nodos

En la Fig. 4.12 es la representación de la comunicación entre los diferentes nodos. El microcontrolador se comunica con el RS485-LB mediante conexión UART.

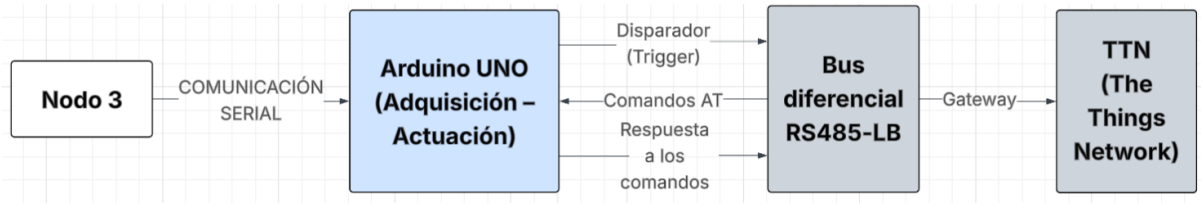



**Fig. 4.12:** Diagrama de bloques del nodo publicador.

El flujo de operación del sistema comienza con la adquisición de señales mediante comunicación serial desde el Nodo 3 hacia un microcontrolador *Arduino UNO*. Este último gestiona la lógica de control mediante comandos AT y señales de disparo hacia el convertidor *RS485-LB* (ver Fig. 4.9). Finalmente, el dispositivo Dragino actúa como interfaz de comunicación inalámbrica, transmitiendo los paquetes de datos vía *LoRaWAN* hacia un *Gateway* para su posterior recepción en la plataforma en la nube *The Things Network (TTN)* [32]. La comunicación se realiza por UART entre el módulo *(BOARD_RX y BOARD_TX)* [48] y el microcontrolador *(D2_RX y D3_TX)*.

## 4.4. The Things Network y servidor puente

### 4.4.1. Protocolo de conectividad con The Things Network

Es esencial configurar la red de manera que garantice la sincronización entre el Gateway y el dispositivo final para poder incluir el nodo sensor en la infraestructura de The Things Network (TTN) [32]. Esta parametrización incluye la elección del plan de frecuencias regional, el modelo de hardware del módulo LoRaWAN y las identificaciones únicas de sesión para encender el aparato.

**Tabla 4.6:** Configuración y especificaciones del RS485-LB en TTN.

| **Características de Configuración** | |
|---|---|
| End device ID | rs485-lb-robot |
| Frequency plan | United States 902-928 MHz, FSB 2 (used by TTN) |
| LoRaWAN version | LoRaWAN Specification 1.0.3 |
| Regional Parameters version | RP001 Regional Parameters 1.0.3 revision A |
| Created at | Sep 19, 2025 13:51:27 |

La Tabla 4.6 describe esta configuración técnica, la cual asegura la vinculación entre el módulo y The Things Network.

### 4.4.2. Algoritmo de decodificación de datos en TTN

El algoritmo llamado `Decoder.txt` implementado permite la recepción, decodificación y procesamiento de tramas provenientes del módulo RS485-LB. Cada trama contiene información transmitida en formato ASCII, organizada en bloques de 10 caracteres que representan variables críticas del sistema. En la Fig. 4.13 se puede observar la estructura de procesamiento del payload para obtener valores legibles

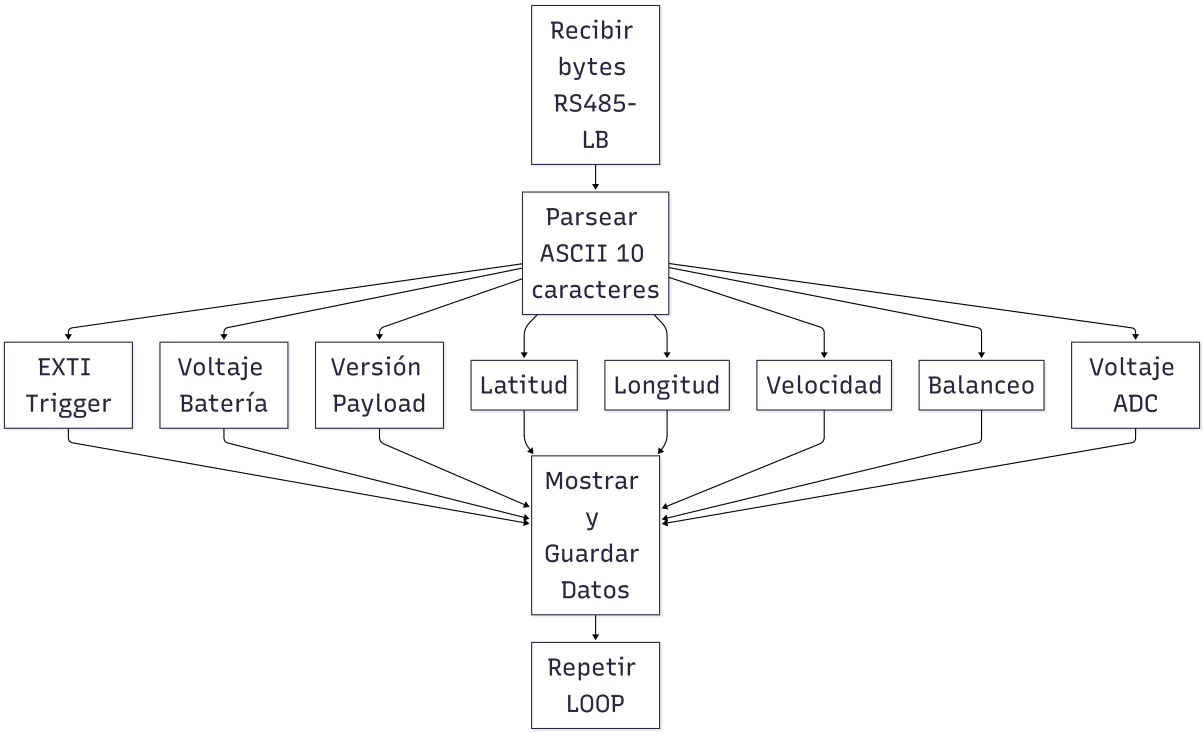



**Fig. 4.13:** Diagrama de flujo del decoder en The Things Network.

### 4.4.3. Payload y decodificación a caracteres

Un ejemplo real del payload que el dispositivo RS485-LB ha recibido en formato hexadecimal se ilustra en la Fig. 4.14. En la primera fila se observan los bytes que transmite el sistema de comunicación LoRaWAN. Cada uno de ellos es el código hexadecimal ASCII de un carácter que manda el robot móvil. El decodificador que se ha implementado en el servidor traduce estos bytes, transforma cada bloque de 10 caracteres ASCII a su valor numérico correspondiente y vuelve a crear las variables del sistema, por ejemplo: latitud, longitud, estado de interrupción, voltaje de la batería, velocidad, balanceo y voltaje que mide el ADC. Así, el procedimiento de decodificación convierte el payload hexadecimal recibido en valores numéricos que son comprensibles para la vigilancia y almacenamiento del sistema. Este se almacena en un archivo con formato JSON.

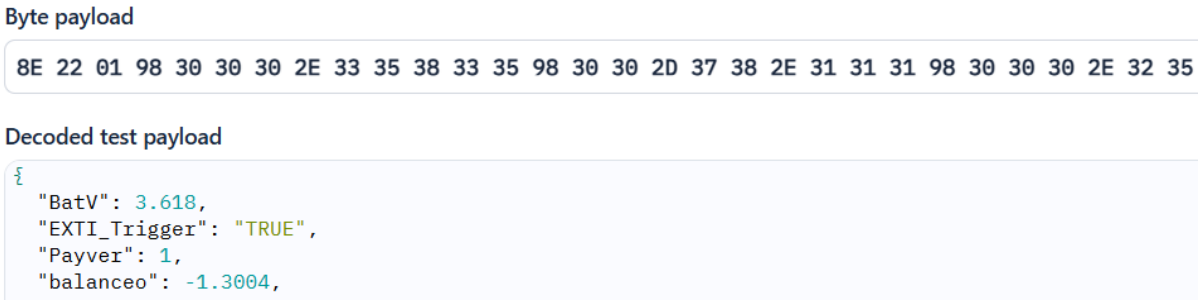



**Fig. 4.14:** Ejemplo de payload recibido en formato ASCII hexadecimal y su decodificación a valores numéricos mediante el algoritmo implementado.

### 4.4.4. Flujo de gestión y sincronización de datos

La arquitectura de comunicación creada para este proyecto tiene en cuenta un flujo de datos multinivel, asegurando que la información sea procesada y respaldada antes de ser visualizada por última vez. El procedimiento comienza en la plataforma The Things Network (TTN), que hace las veces de receptor principal de los paquetes de datos en formato *JSON*.

Tomando como referencia el trabajo [50] la información se transfiere a un servidor puente que está basado en Ubuntu y es administrado de manera remota como se muestra en Fig. 4.1. El servidor web Apache es el que recibe y codifica las tramas dentro de este servidor. Luego, los datos se almacenan físicamente en una base de datos *MySQL*. Este paso intermedio posibilita un control local de la información y garantiza que los datos se mantengan a lo largo del tiempo. Finalmente se implementó un *script* de automatización denominado `Firebase.py`. Este componente se encarga de extraer los registros almacenados en *MySQL* y transmitirlos hacia la nube de Firebase, logrando así que la aplicación final refleje los movimientos del nodo de manera dinámica y sin latencia perceptible para el usuario.

## 4.5. Almacenamiento local

La base de datos SQLite se emplea como repositorio local de los datos del robot, permitiendo conservar un historial de eventos críticos y mediciones de sensores incluso cuando la conexión a Firebase no está disponible. Cada registro almacena información de *latitud, longitud, velocidad, balanceo y voltaje*, garantizando que la interfaz pueda acceder a datos recientes de manera inmediata, dicha información almacenada será usada para el sistema de alertas y el respaldo de todos los errores que ocurrieron, para evitar pérdida de datos cuando el sistema no esté habilitado los datos quedaran resguardados.

## 4.6. Interfaz gráfica

Para el monitoreo y la visualización de las notificaciones de eventos críticos de forma remota, se desarrolla una interfaz con el nombre `Interfaz_Notificaciones_Remotas.py` con los principales eventos críticos cada uno con su respectiva sección como se observa en la Fig. 4.15, se emplea el concepto de *Cloud Computing*.

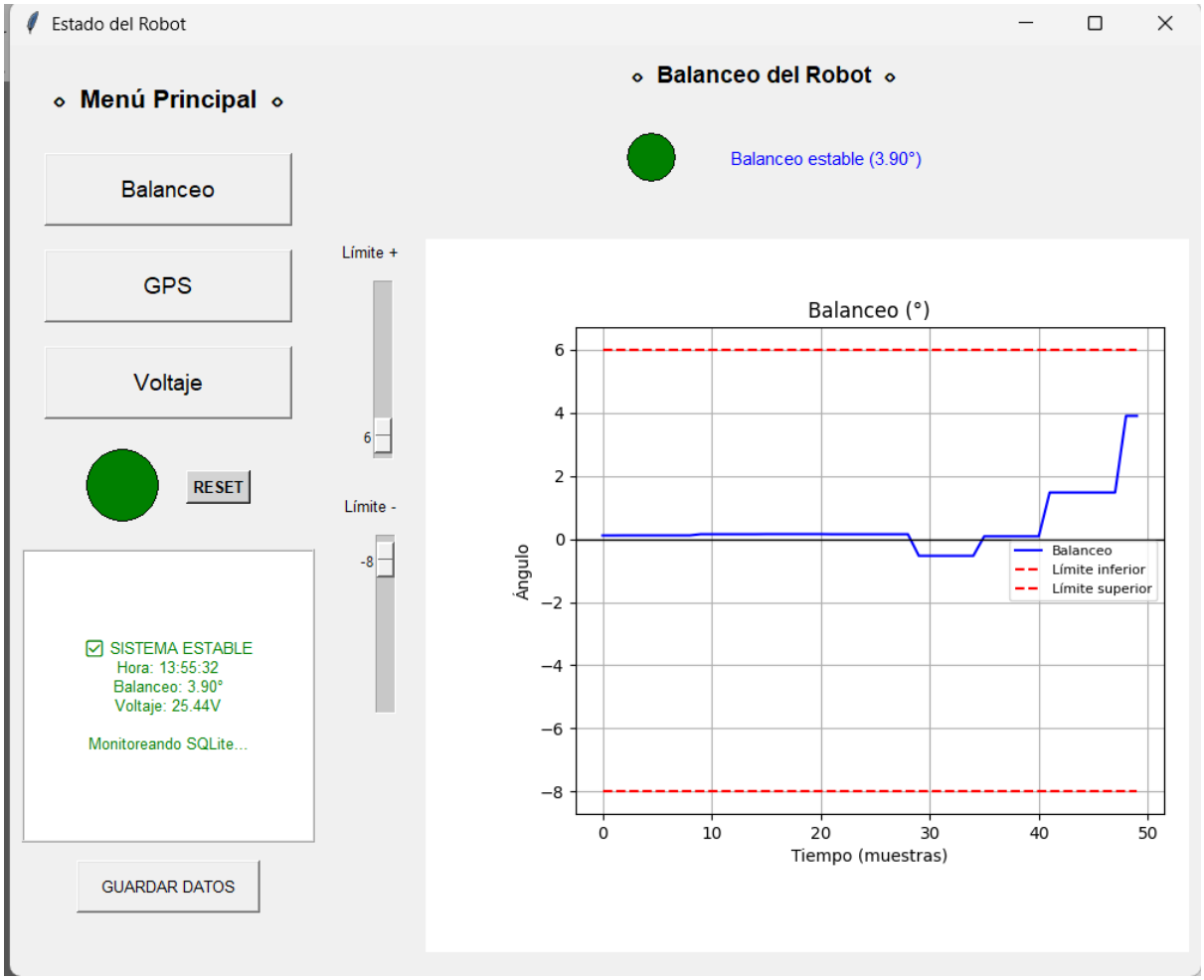



**Fig. 4.15:** Interfaz gráfica para la visualización de eventos críticos y el estado actual del robot.

Como se observa la interfaz no tiene secciones complejas debido a que emplea el código de colores y señalización que se explica en la sección 2.2.8, y la usabilidad descrita en la sección 2.2.9. Esto facilita la interacción entre el operador sea o no que tenga experiencia en el campo.

### 4.6.1. Captura y gestión de datos de robot mediante Firebase

La biblioteca *requests* establece la conexión con Firebase, utilizando solicitudes HTTP para adquirir en tiempo real la información del robot. La función *update_status* interroga a la base de datos en la nube cada segundo, adquiere voltaje, balanceo y coordenadas GPS y renueva automáticamente los gráficos, LEDs y labels. Esto hace posible que la GUI opere como un sistema de monitoreo en tiempo real, exhibiendo alertas y datos visuales cuando los valores sobrepasan los límites establecidos.

### 4.6.2. Menú y gráficos de la interfaz

La interfaz principal como se observa en la Fig. 4.15 se estructura con *Tkinter*, que posibilita la creación de elementos visuales como ventanas, sliders, botones, etiquetas y lienzos. Un menú lateral con botones que posibilitan la alternancia entre las distintas vistas se encuentra en el lado izquierdo de la ventana: Voltaje, GPS y balanceo. Todos los botones están vinculados a la función *show_frame*, que enseña el marco correspondiente y esconde el resto, lo que posibilita una navegación fácil dentro de una misma ventana. Se emplean *Canvas* para representar elementos visuales, como los LEDs que señalan el estado de balanceo, la posición GPS o el voltaje, dentro de cada cuadro. Asimismo, para los gráficos de voltaje y balanceo se emplean *Matplotlib* y *FigureCanvasTkAgg*, permitiendo así la incorporación de los gráficos a la interfaz gráfica de usuario (GUI) de *Tkinter* y su actualización en tiempo real durante el envío de datos por parte del robot.

### 4.6.3. Monitoreo de balanceo y voltaje

Se emplean estructuras *deque* de la biblioteca *collections* para llevar a cabo el monitoreo de voltaje y balanceo, las cuales guardan los 50 valores más recientes para graficar en tiempo real evitando que la memoria se vea sobrecargada. La función *update_status*, que se ejecuta en un hilo paralelo (threading), tiene la responsabilidad de adquirir información de Firebase y actualizar constantemente los LEDs, etiquetas y gráficos.

Los límites superior e inferior para el balanceo como se muestra en la Fig. 4.15 se establecen por medio de sliders, permitiendo al usuario modificar los rangos de alerta. De acuerdo al valor de balanceo, el LED cambia su color entre rojo amarillo y verde, también aparece un texto que señala si el robot está estable, en riesgo de caerse, sobreesfuerzo de motores por alta inclinación o se ha caído. Para el voltaje que se observa en la Fig. 4.16, se presentan cuadros de batería en 4 barras que cada una representa un 25 % de la batería nominal que se pintan con colores

dependiendo del nivel actual, así como una gráfica de voltaje con líneas que señalan cuando la batería está baja o sobrecargada.

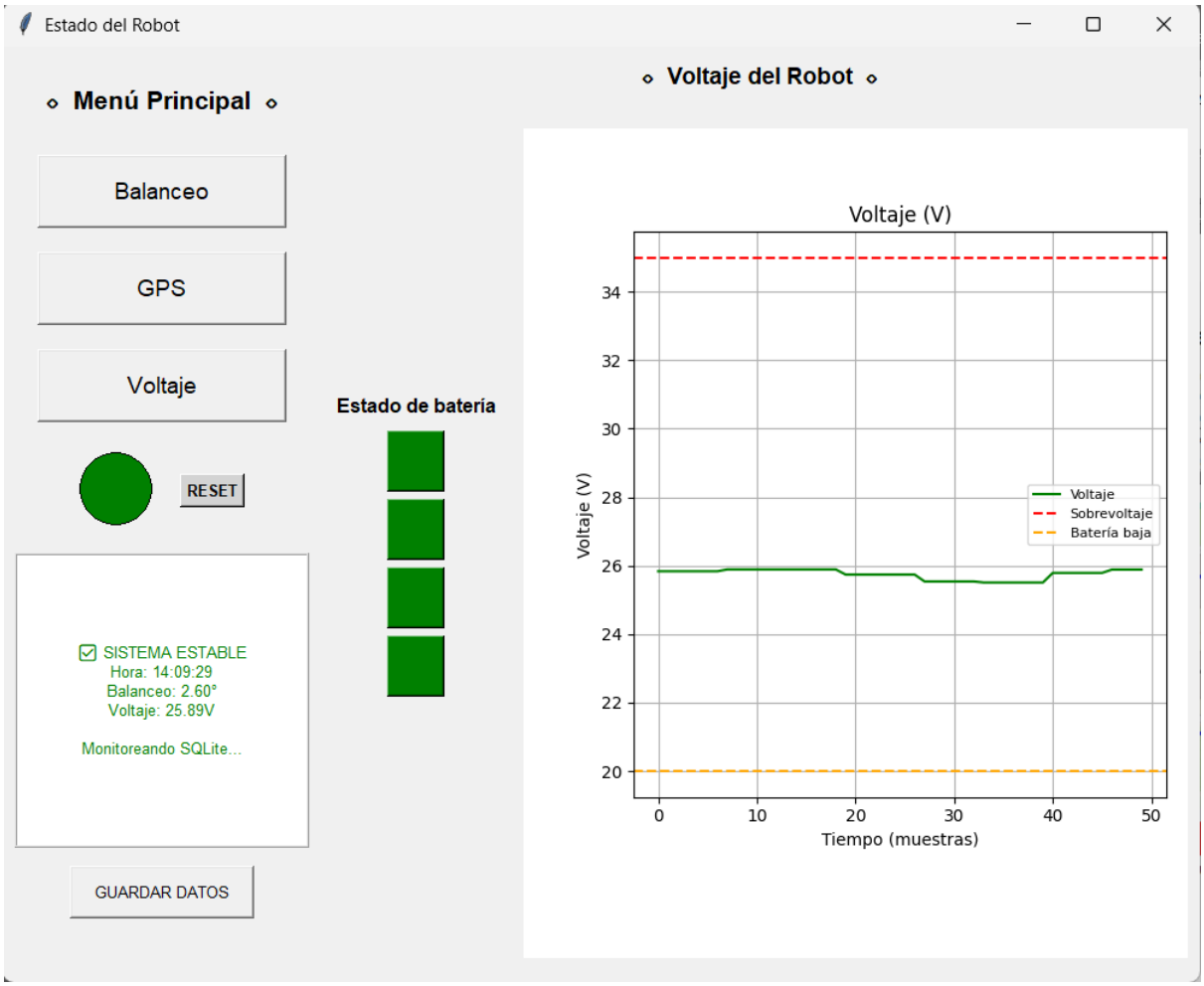



**Fig. 4.16:** Sección del sistema voltaje que muestra los límites de la tensión de las baterías, su estado de carga y la gráfica de los posibles picos que puede llegar a tener.

### 4.6.4. Visualización y control de GPS

La librería *tkintermapview*, que posibilita la visualización de un mapa en la ventana de Tkinter, es empleada por la sección de GPS como se observa en la Fig. 4.17. En este lugar, se puede observar la ubicación del robot a través de un marcador y es posible seguir su trayectoria con una línea roja identificando las zonas por donde el robot se ha movilizado. Además, se traza un cuadrado azul que señala el rango de movimiento que tiene permitido el robot en una escala de 0.1km a 5km. El usuario tiene la opción de modificar el centro del mapa introduciendo manualmente los valores de longitud y latitud o emplear los valores predeterminados. También,

el LED en rojo o verde señala si el robot está dentro del rango permitido, lo cual brinda una retroalimentación visual y rápida.

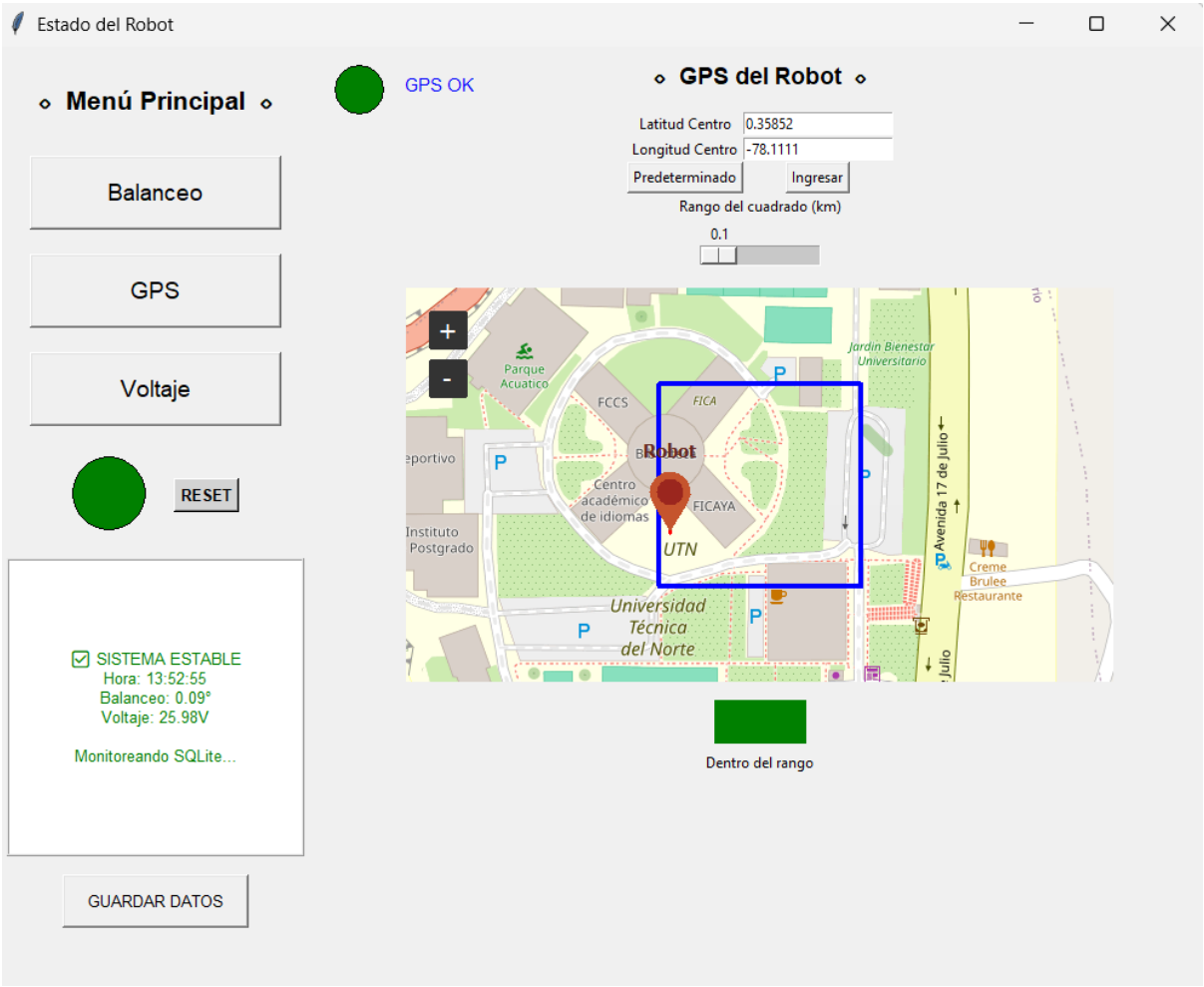
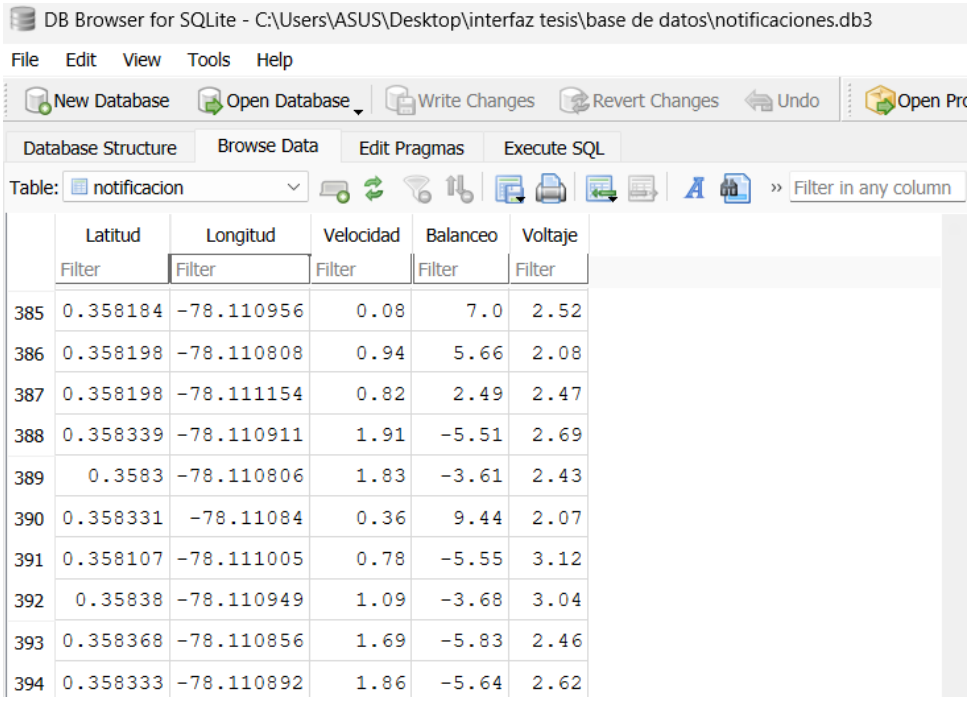



**Fig. 4.17:** Sección del menú GPS con sus respectivas funciones y datos editable como rango o punto central.

### 4.6.5. Base de datos SQLite

Para el almacenamiento se emplea *SQLite (sqlite3)* para almacenar los datos de manera local como se observa en Fig. 4.18 en el equipo que esta operado la sistemas, estos llegan directamente de firebase. Otra función importante de esta librería es la que nos permite acceder a los datos históricos, lo que posibilita leer el último registro de la base local y presentarlo en la interfaz. La función *leer_ultimo_registro_sqlite* obtiene la longitud, latitud, velocidad, voltaje y balanceo; por su parte, actualizar_label_sqlite presenta esta información en la etiqueta de estado principal y se encarga de determinar si se debe activar alguna alerta. Esto garantiza

que la interfaz gráfica siga mostrando datos recientes, aun cuando la conexión con Firebase no funcione en el programa principal .




| | Latitud | Longitud | Velocidad | Balanceo | Voltaje |
|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter |
| 385 | 0.358184 | -78.110956 | 0.08 | 7.0 | 2.52 |
| 386 | 0.358198 | -78.110808 | 0.94 | 5.66 | 2.08 |
| 387 | 0.358198 | -78.111154 | 0.82 | 2.49 | 2.47 |
| 388 | 0.358339 | -78.110911 | 1.91 | -5.51 | 2.69 |
| 389 | 0.3583 | -78.110806 | 1.83 | -3.61 | 2.43 |
| 390 | 0.358331 | -78.11084 | 0.36 | 9.44 | 2.07 |
| 391 | 0.358107 | -78.111005 | 0.78 | -5.55 | 3.12 |
| 392 | 0.35838 | -78.110949 | 1.09 | -3.68 | 3.04 |
| 393 | 0.358368 | -78.110856 | 1.69 | -5.83 | 2.46 |
| 394 | 0.358333 | -78.110892 | 1.86 | -5.64 | 2.62 |

**Fig. 4.18:** Base de datos local encargada de colectar los datos directamente de Firebase y encargada de proporcionar datos históricos al sistema principal.

### 4.6.6. Etiqueta de estado, botón de guardar y reseteo

La etiqueta principal se observa en Fig. 4.19 *(voltaje_led_label)* se utiliza para presentar el estado general del robot, incluyendo el voltaje, alertas que se hayan detectado y balanceo. En caso de que se identifique un evento crítico, la etiqueta se actualiza con los datos del error y cambia a rojo, al igual que el LED correspondiente. Los datos son tomados directamente de la base de datos local con el programa `Firebase_a_SQL.py` El botón de guardar (*GUARDAR DATOS*) posibilita almacenar en un archivo de texto todas las alertas que se han detectado durante la operación del dispositivo, haciendo uso de la librería *os* para administrar rutas y archivos. Esto facilita la creación de un historial que se puede revisar después fuera de la aplicación, conservando un registro de los acontecimientos críticos por los que ha pasado el robot.

El botón de reseto (*RESET*) que se encuentra en la interfaz posibilita que el usuario reinicie manualmente las alertas visuales sin modificar los datos de Firebase o de la base de datos. El LED que señala el estado del sistema se vuelve verde y en la etiqueta principal muestra un mensaje informando al operador que el sistema ha sido reiniciado y espera nuevos datos.

Adicionalmente, el botón cambia a verde con letras blancas para indicar que el usuario ha limpiado la alerta. Mientras esté en esta condición, la interfaz no sobrescribirá el mensaje hasta que se detecte una nueva alerta; esto ocurre cada 10 segundos para darle tiempo de restablecer o solucionar las fallas.


**Fig. 4.19:** Sección de la etiqueta (label) que se encuentra en el menú principal donde se observa el estado del robot o si existe alguna posible alerta junto al botón que permite resetear el sistema de alertas y el botón que permite guardar los fallos que ocurrieron durante el monitoreo.

## 4.7. Pruebas y montaje del sistema

### 4.7.1. Pruebas simuladas y montaje de los componentes

Para probar las efectividad y correcto funcionamiento del sistema se desarrolla un programa que simula diversas situaciones que pueden ocurrir durante la operación del robot móvil, para ello se emplea el programa denominado `Datos_Simulados.py` que se encargan de enviar datos de forma directa a firebase cada 10 segundos para simular los datos reales que ingresarán en la nube. La simulación es beneficiosa para el reajuste del sistema y cambios de

los diversos umbrales, con esto se logró una efectividad del 99 % en lo que respecta al correcto funcionamiento. Tras la comprobación del correcto funcionamiento se procede a la instalación de todos los componentes dentro del robot móvil sin comprometer los dispositivos previamente instalados y sus respectivos funcionamientos como se observa en la Fig. 4.20 y se los numera de tal manera que su ubicación sea reconocible en la Tabla 4.7

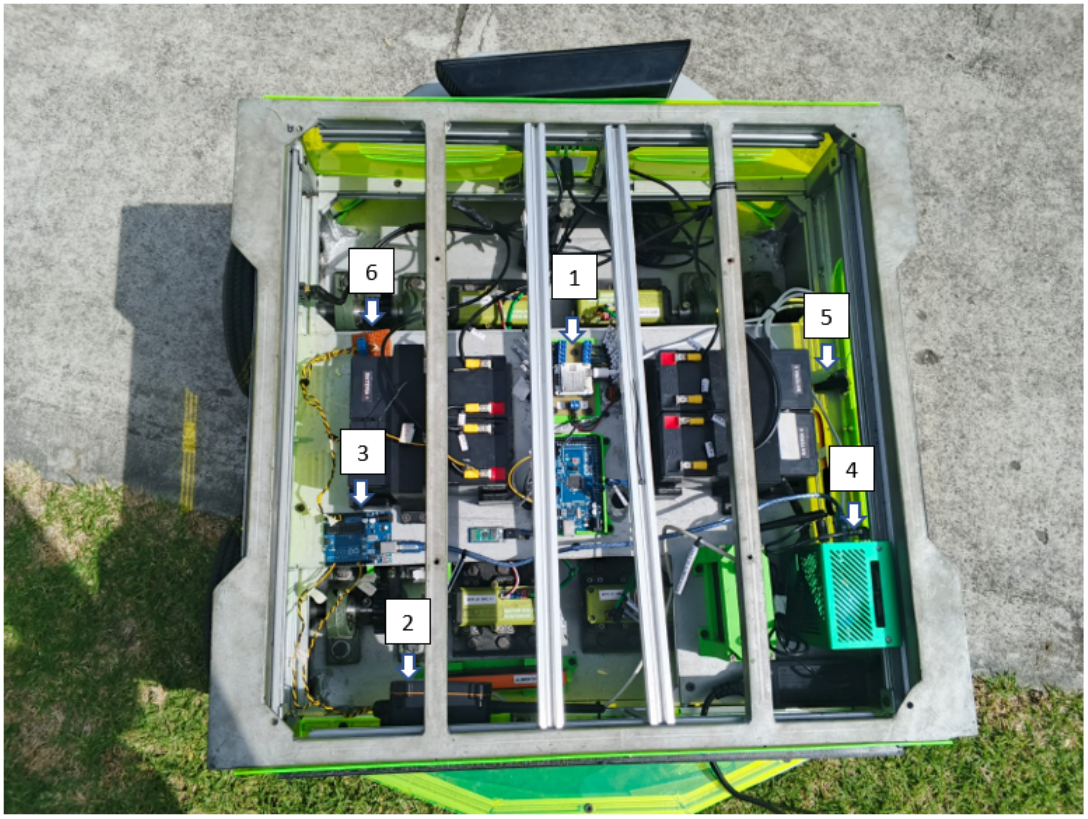



**Fig. 4.20:** Montaje interno de componentes eléctricos.

**Tabla 4.7:** Numeración de los componentes instalados.

| **Numeración de los componentes instalados** | |
|---|---|
| 1 | TransducerM 9-Axis AHRS / IMU |
| 2 | Conversor RS485-LB de Dragino |
| 3 | Microcontrolador Arduino UNO |
| 4 | NVIDIA Jetson Orin Nano 8 GB |
| 5 | El módulo u-blox NEO-6M V2 |
| 6 | Divisor de voltaje |

### 4.7.2. Pruebas de Campo

En esta sección se presenta el funcionamiento del sistema de notificaciones remotas basado en Cloud Computing, con el propósito de validar el funcionamiento en diversas situaciones que se pueden presentar en el robot móvil de alto torque, eventos críticos como el balanceo, voltaje de las baterías, ubicación mediante GPS. El sistema de monitoreo remoto esta nombrado como `Notificaciones_Remotas_de_Eventos_Criticos.py` fue diseñado para ser ejecutado en cualquier ordenador siempre y cuando éste tenga conexión a internet y tenga instalado Python, de preferencia la versión 3.14, o un software gratuito como Visual Studio Code con sus respectivas librerías que se ubican en el archivo `Librerias.txt` para una correcta instalación y funcionamiento del sistema. La descripción de cada una de las alertas se encuentra en la Tabla 4.8.

**Tabla 4.8:** Códigos de errores del sistema.

| **CÓDIGOS DE ERRORES** | |
|---|---|
| `BAL-01` | POSIBLE CAÍDA POR INERCIA", "El robot se está inclinando demasiado. Revisar inclinación y trayectoria. |
| `BAL-02` | SOBREESFUERZO EN LOS MOTORES", "Los motores están trabajando demasiado. Verificar balanceo y límites. |
| `BAL-03` | EL ROBOT SE CAYÓ", "El robot se volcó. Enviar a un operador a revisar inmediatamente. |
| `VOL-01` | SOBREVOLTAJE DETECTADO", "El voltaje ha superado 35V. Posible riesgo de daño al robot. |
| `VOL-02` | BATERÍA BAJA", "El voltaje está por debajo de 20V. Cargar o reemplazar batería. |

### 4.7.3. Pruebas de balanceo

Un evento crítico de alto valor dentro del sistema es el balanceo, debido a que una inclinación excesiva en uno de sus ejes es capaz de comprometer la estabilidad y la integridad estructural del robot móvil de alto torque. Se emplea el acelerómetro por las características que ofrece y se describen en la sección 4.3.3.1. Se emplea el eje X para la respectiva recopilación de información que es procesada en el sistema y visualizada en la interfaz mostrada en la Fig. 4.15.

En la sección del balanceo existen 3 posibles alertas: *INCLINACION SUPERIOR*, *INCLINACIÓN INFERIOR*, *CAÍDA DEL ROBOT*.

#### 4.7.3.1. Notificación con respecto al eje X positivo.

En la Fig. 4.21 se muestra como el sistema ejecuta la alerta cuando el valor del balanceo supera el limite que se le ha establecido con respecto al eje X positivo. El led indicador principal de la sección balanceo pasa de color verde a un tono amarillo junto con el mensaje de estado en la parte superior, el led general cambia a un color rojo informando un evento critico, la etiqueta principal muestra la alerta, la hora a la que ocurre, el codigo del error *BAL-01* mostrado en la Tabla 4.8, el evento, el valor y las cordenadas donde se encunetra el robot. Todo esto queda registrado en el archivo de texto junto a todo el historial de alertas registrados.

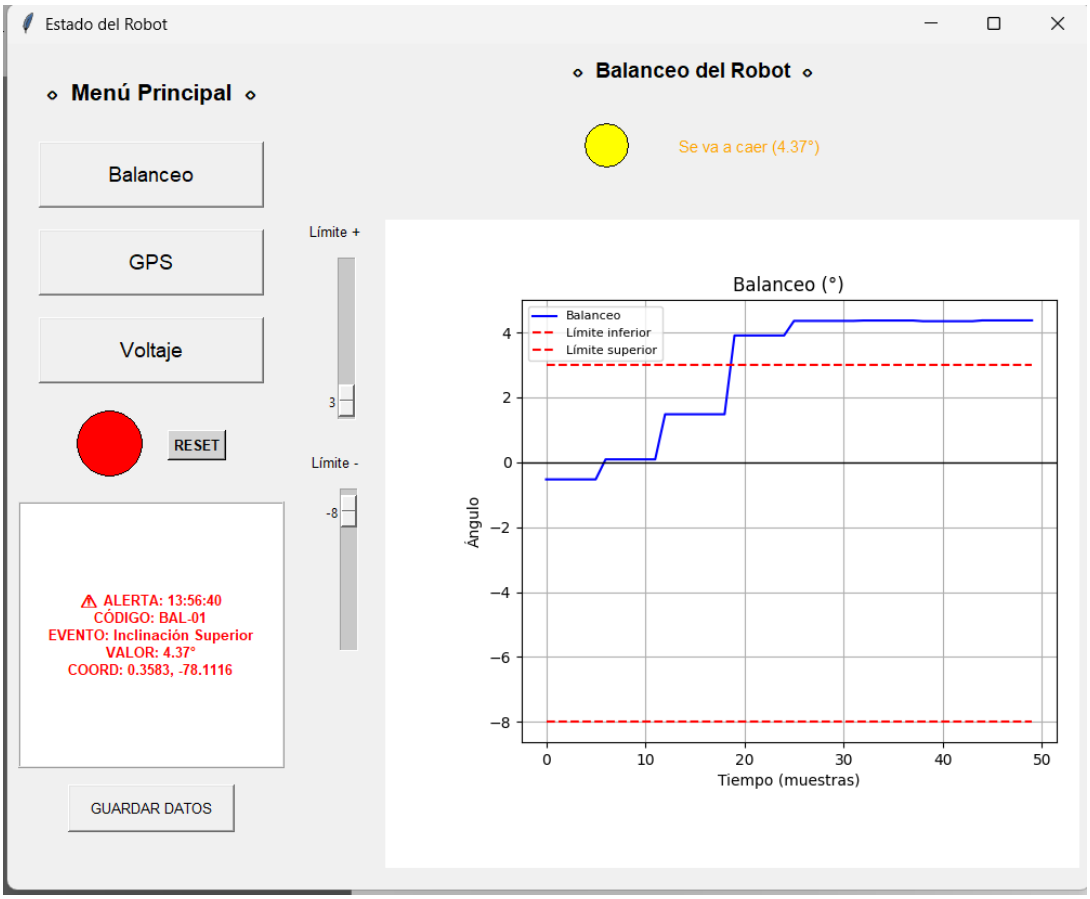



**Fig. 4.21:** Captura de la interfaz mostrando las alertas cuando ocurre un evento crítico relacionado al eje X positivo.

#### 4.7.3.2. Notificación con respecto al eje X negativo.

En la Fig. 4.22 se muestra cómo el sistema ejecuta la alerta cuando el valor del balanceo supera el límite que se le ha establecido con respecto al eje X negativo. El LED indicador principal y el mensaje de estado, el LED general junto con la etiqueta de estado general, pero

con sus respectivos datos de caso, se comportan de manera similar que en la sección 4.7.3.2, solo variando el código de error *BAL-02* mostrado en la Tabla 4.8 y el evento.

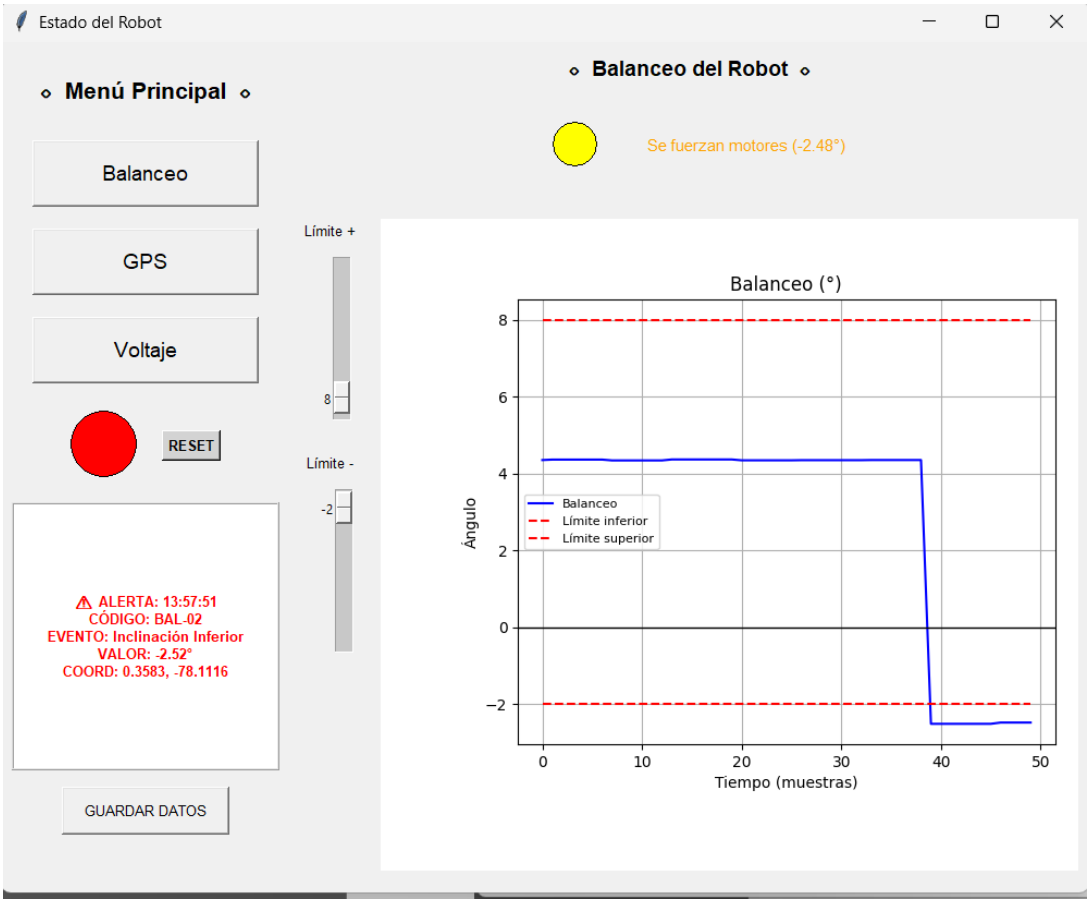



**Fig. 4.22:** Captura de la interfaz mostrando las alertas cuando ocurre un evento crítico relacionado al eje X negativo.

### 4.7.3.3. Notificación de volcamiento

En la Fig. 4.23 se observa lo que ocurre cuando el robot se cae; el LED principal cambia a color rojo junto con el mensaje de estado, debido a que es una alerta crítica. Esto ocurre cuando sobrepasa el balanceo de ±80°. En ese caso, registran dos alertas que se guardan en el archivo de texto, mostrando los códigos de error *BAL-01, BAL-03* cuando se cae con respecto al eje X positivo, o *BAL-02, BAL-03* con respecto al eje X negativo, que están explicados en la Tabla 4.8.

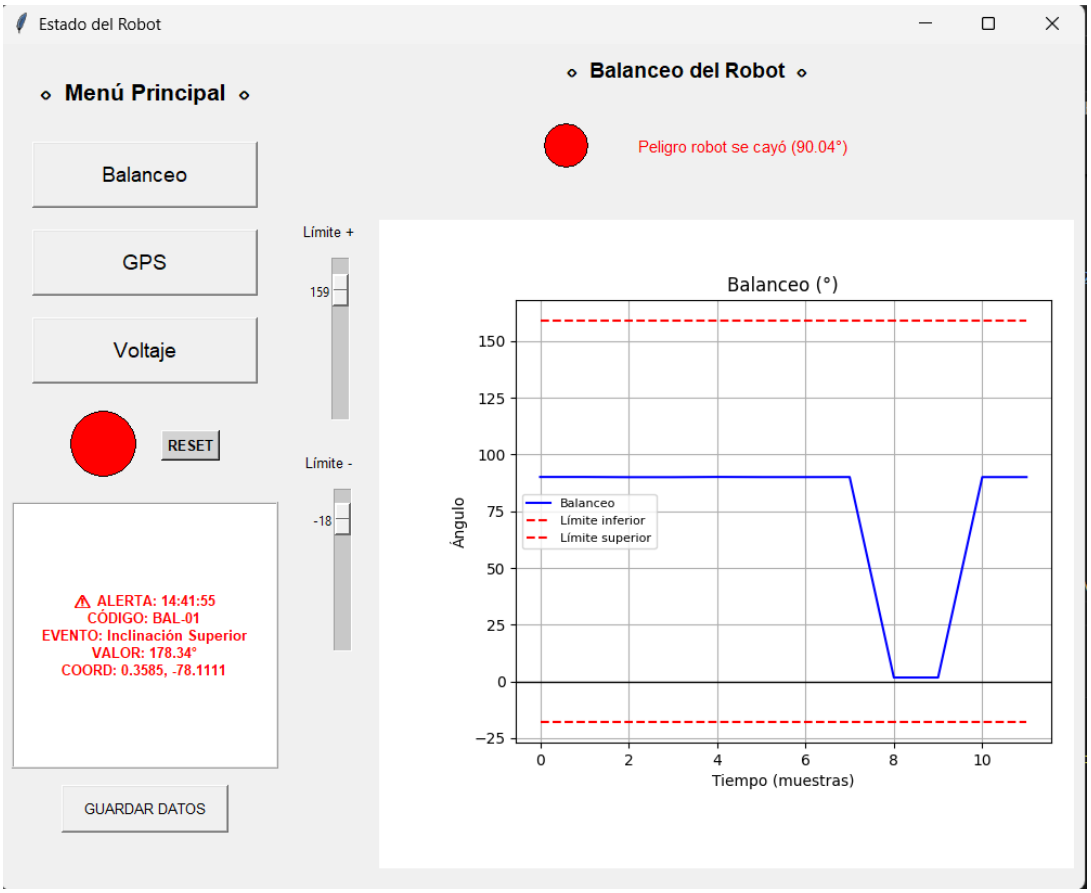



**Fig. 4.23:** La interfaz mostrando las alertas cuando ocurre un volcamiento del robot con respecto a cada uno de los polos del eje X.

### 4.7.4. Notificación del GPS

Cuando se trabaja con un robot móvil, es necesario saber su ubicación; un evento crítico se ejecuta si este no se encuentra en un rango seguro o perímetro permitido. Se emplea un módulo GPS por las características que ofrece y se describen en la sección 4.6.4. Para este apartado de la interfaz se usan los datos de *latitud, longitud y velocidad* para su ubicación y desplazamiento, como se muestra en la Fig. 4.17.

En la Fig. 4.24 se observa cómo funciona el sistema de alertas: cuando el robot se encuentra en coordenadas que no están dentro del perímetro asignado, representado con el recuadro de líneas azules, el LED indicador en la parte inferior del mapa se coloca de color rojo con un mensaje de alerta; el LED general cambia de tonalidad verde a rojo y la etiqueta de estado general muestra una alerta con el mensaje *"Robot fuera de rango"* con sus respectivas coordenadas. También permite visualizar el recorrido que realiza el robot con la guía de color rojo; el LED en la parte superior muestra la conexión con el GPS, ya sea activa o desconectada.

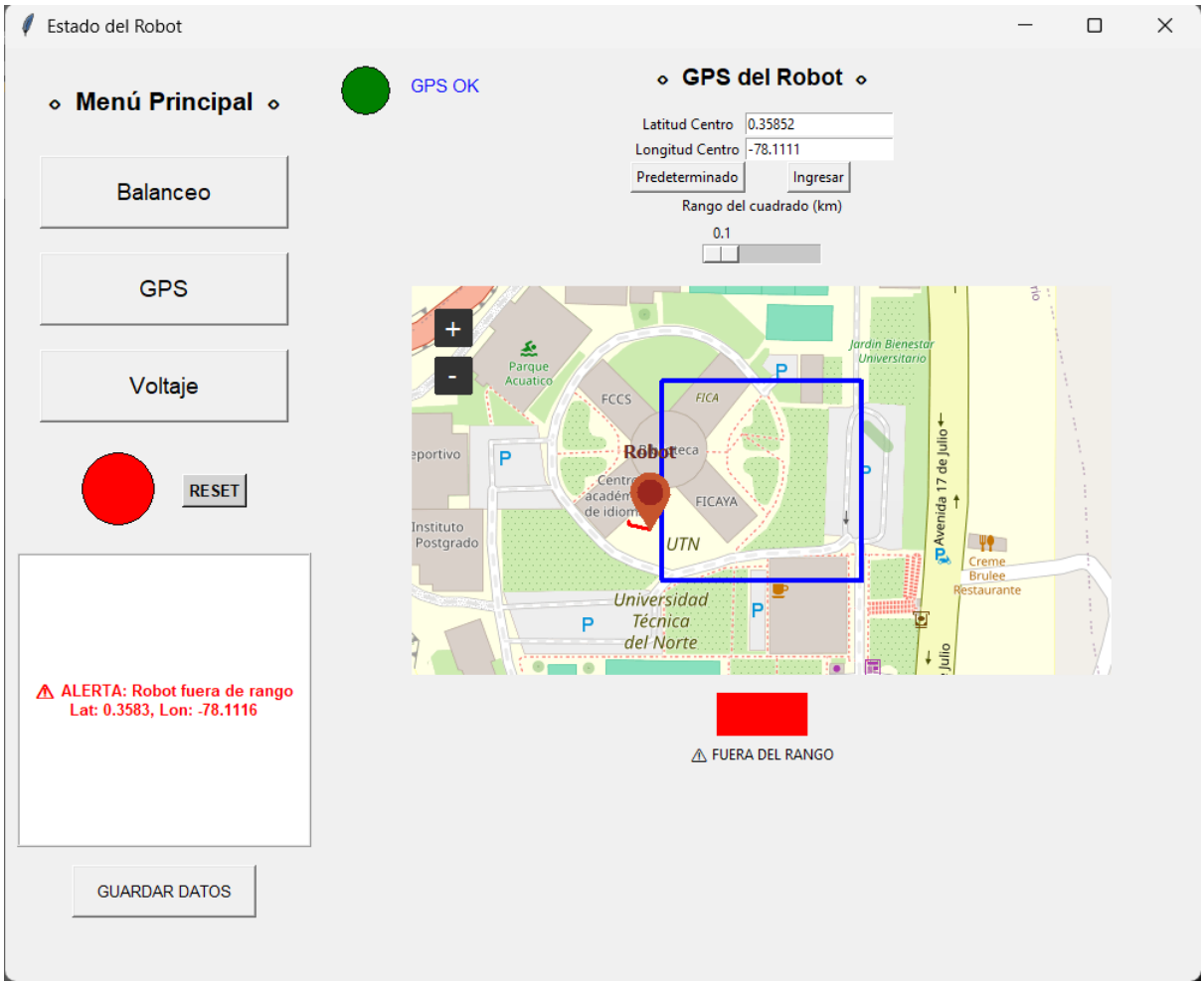



**Fig. 4.24:** Captura de la interfaz mostrando la ubicación actual del robot, el trayecto recorrido, fuera del rango establecido, con su respectiva alerta.

### 4.7.5. Pruebas de voltaje

Un evento crítico de alto impacto, cuando se estudian o diseñan sistemas electrónicos, es el voltaje de alimentación. El sistema se encarga de medir continuamente el voltaje de las baterías mediante un circuito divisor resistor; debido a que el microcontrolador empleado soporta un voltaje pico de 5.1 V, se explica en la sección 4.3.3.4. En la Fig. 4.16 se observa la interfaz del voltaje junto a sus componentes visuales. En este apartado existen 2 posibles alertas; entre ellas están: *SOBREVOLTAJE* y *BATERÍA BAJA*.

#### 4.7.5.1. Notificación de sobrevoltaje

El sobrevoltaje constituye una condición peligrosa para los equipos electrónicos, por lo que es fundamental identificarlo a tiempo para evitar averías. El sistema se encarga de medir el voltaje y ajustarlo a su valor real para procesarlo. El sistema, al momento de detectar valores superiores a 35 V, determinado al medir las baterías en una máxima capacidad de uso recomendado, activa el protocolo de alerta, como se muestra en la Fig. 4.25. Como en los anteriores casos, cambiará el color de la etiqueta de estado general y el LED general. El código de error es *VOL-01*, explicado en la Tabla 4.8.

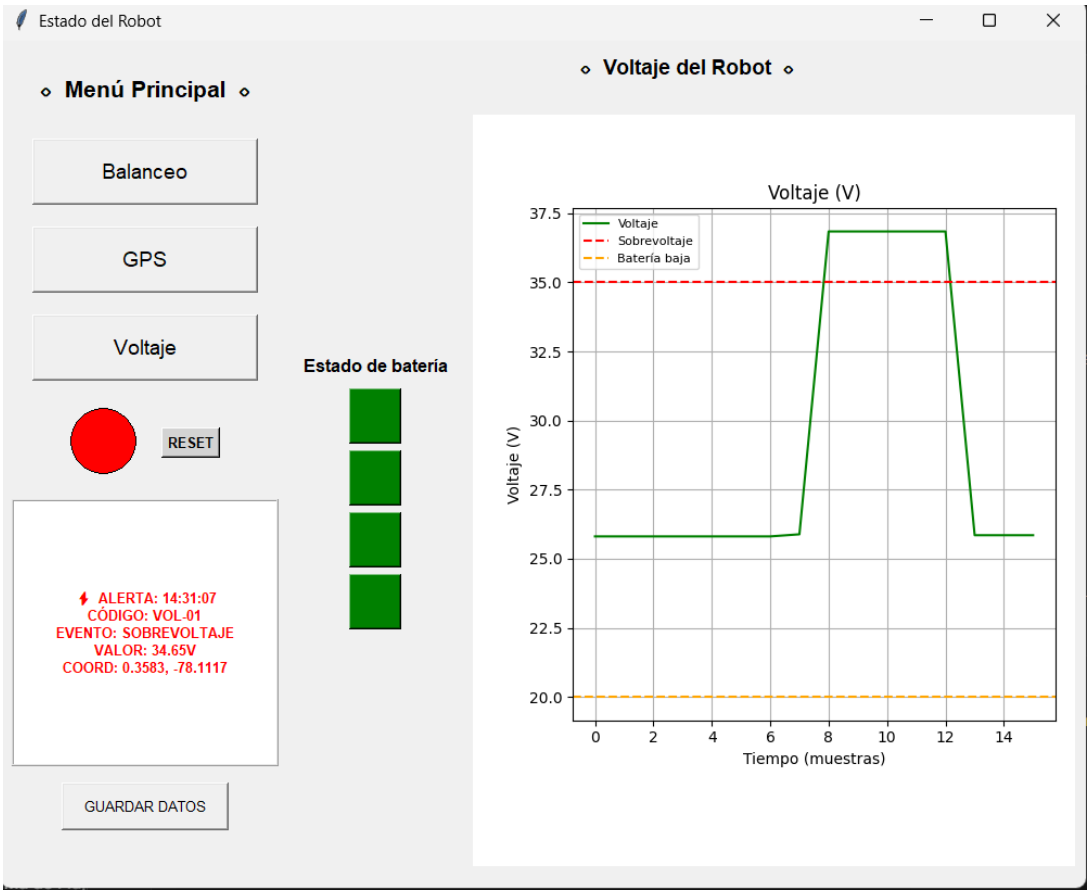



**Fig. 4.25:** Interfaz mostrando el comportamiento de un pico de voltaje.

#### 4.7.5.2. Notificación de batería baja

De manera similar, el sistema monitorea los niveles de voltaje de la batería para evitar fallos por descarga excesiva. Cuando el voltaje medido desciende por debajo del umbral establecido en este caso 20V, este es el nivel de voltaje en donde el robot movil presenta problemas de deplazamiento, el sistema genera una notificación de nivel bajo de batería, activando el protocolo

de alerta correspondiente.

Esta alerta modifica el estado visual de la aplicación, cambiando el color de la etiqueta general y del LED indicador, además de enviar una notificación remota al operador observado en Fig. 4.26 El código de error asignado a este evento es *VOL-02*, descrito en la Tabla 4.8.

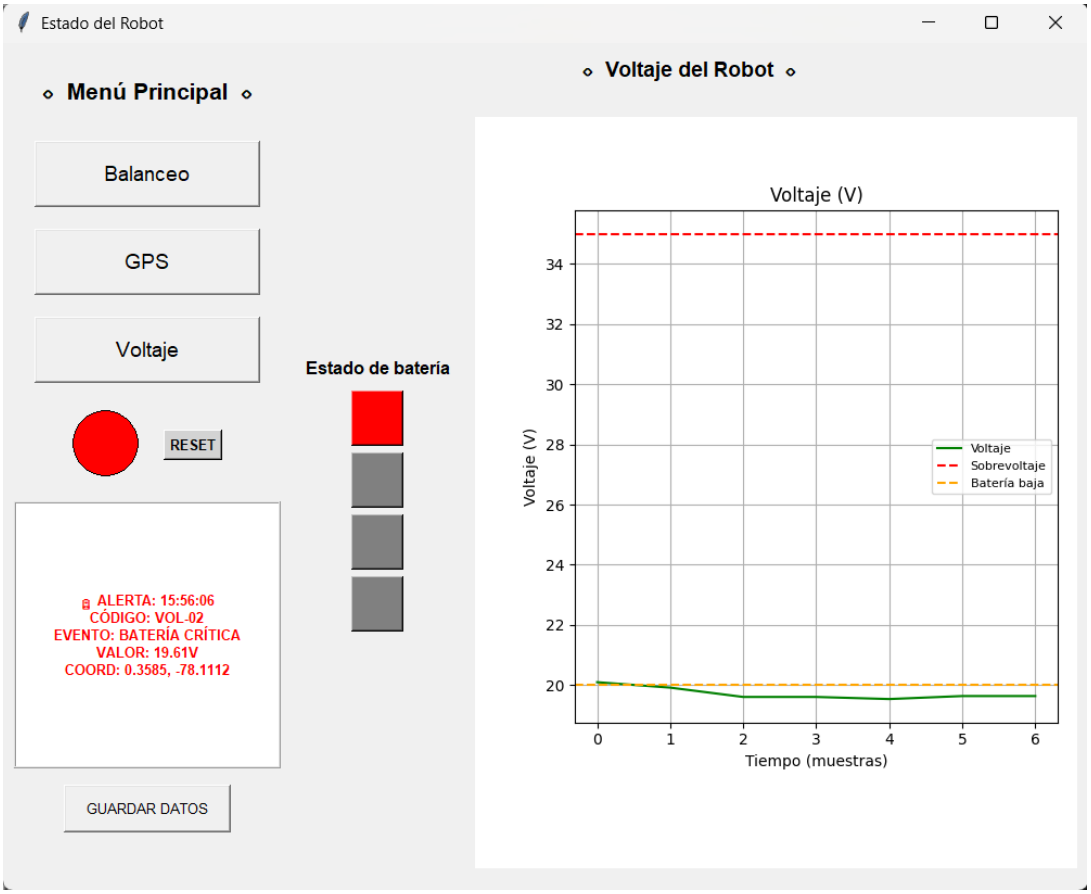



**Fig. 4.26:** Interfaz mostrando el comportamiento de bajo voltaje.

#### 4.7.5.3. Niveles de batería

En la Fig. 4.27 se visualizan los diversos estados de los niveles de las baterías, dependientes del voltaje que estas envíen al microcontrolador. En el primer caso tenemos bajo voltaje, que ocupa una carga de 0 % a 25 % en color rojo, siendo un nivel crítico; en el segundo caso, un valor entre 26 % y 50 % en color naranja, que representa una carga media; para el tercer caso, una carga que oscila entre 51 % y 75 %, mostrando un nivel de batería aceptable; y por último, una carga de 76 % a 100 % en color verde para indicar que las baterías poseen una carga completa o excelente.

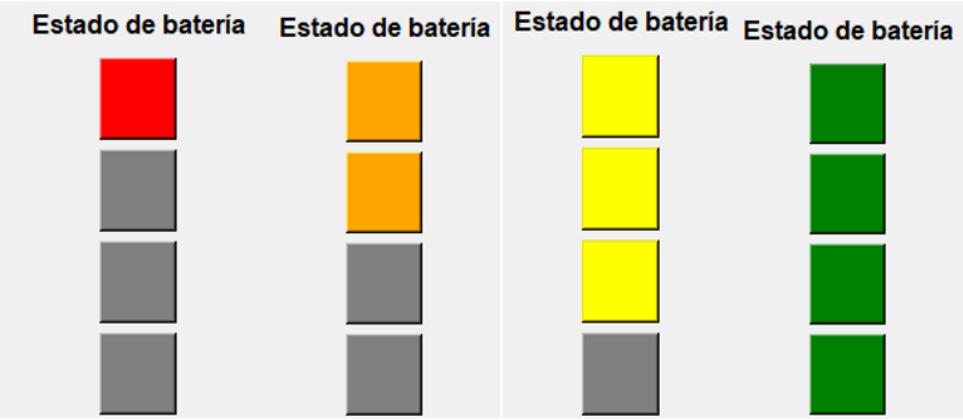



**Fig. 4.27:** Imagen de los diversos ciclos de carga de las baterías del robot móvil.

### 4.7.6. Sección de monitoreo general

En la Fig. 4.19 se tiene varios objetos entre ellos un botón que permite guarda el registro de todos los fallos que se suscitaron durante el periodo de monitoreo, la etiqueta y un led general que se encarga de mostrar en todo momento el estado general del robot, un botón de reseteo que se encarga de pausar la notificacion en un periodo de 10 segundos hasta esperar otro evento critico. En la Fig. 4.28 podemos observar que cuando se presiona el botón de reseteo se cambia de color y la etiqueta de estado general muestra el mensaje de reseteo y la hora a la que se realiza este mismo.

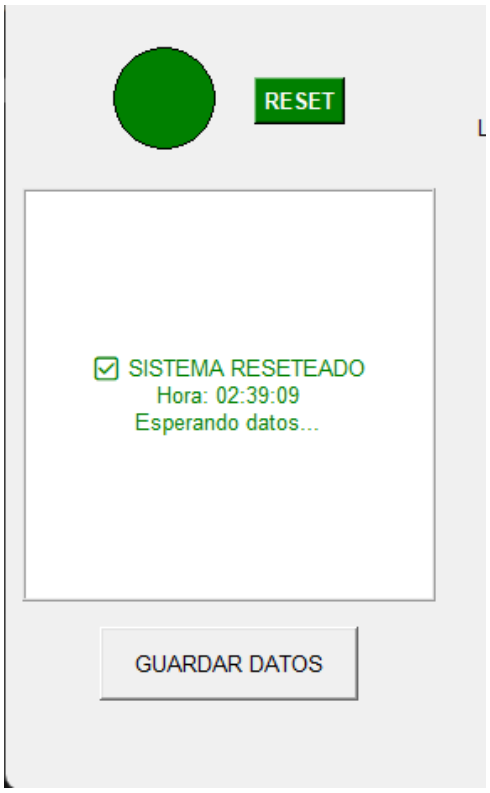



**Fig. 4.28:** Sección del monitoreo general.

El archivo de texto anteriormente mencionado, que se genera cada vez que seleccionamos la opción de *GUARDAR DATOS*, se muestra en la Fig. 4.29 y se organiza de la siguiente manera: la fecha y la hora que inicia el sistema; posteriormente, la hora con su respectivo código de falla o evento crítico y los valores que se obtuvieron durante ese lapso de tiempo.

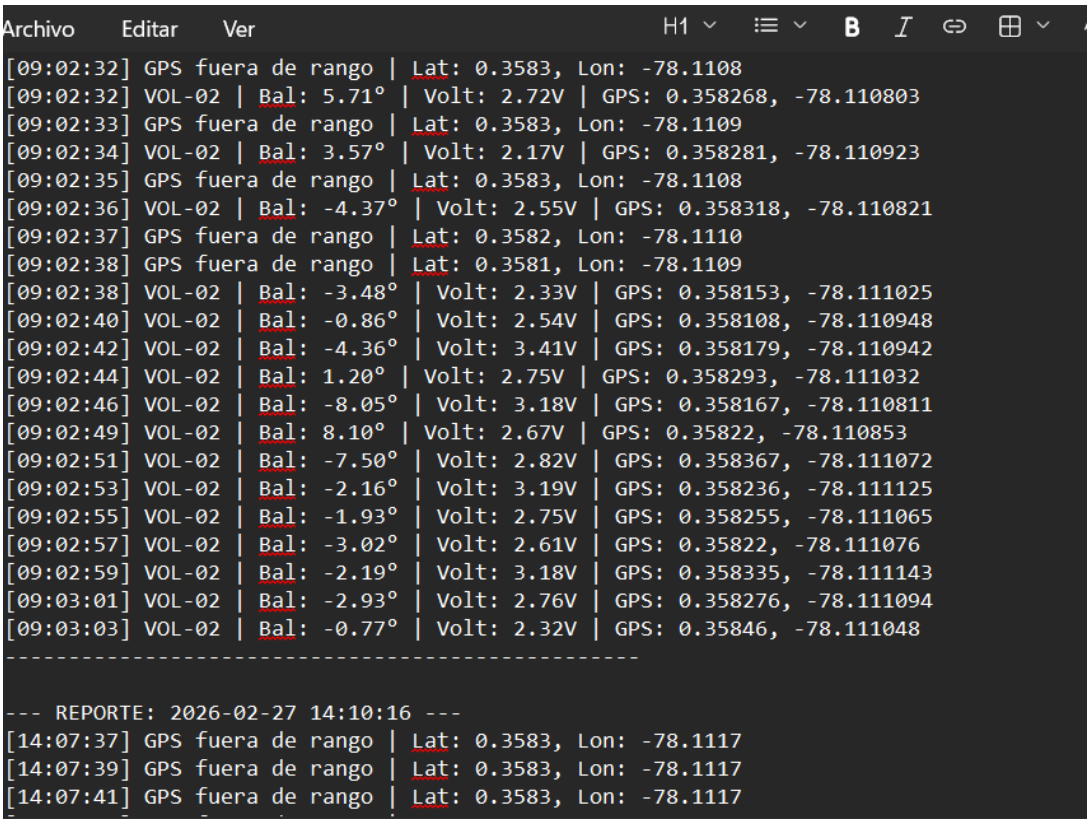



**Fig. 4.29:** Imagen de los registros de eventos críticos actualizable.

## 4.8. Evaluación de desempeño del enlace LoRaWAN

Se evalúa el rendimiento del enlace de comunicación *LoRaWAN* en un entorno abierto durante las pruebas de campo, donde la distancia entre el nodo final y la puerta de enlace era de alrededor de 100 metros. Estos resultados se obtienen mediante las pruebas con ayuda de *MySQL* y *TTN* para verificar el envío de datos. El periodo de envío de datos es de aproximadamente 21 segundos, y el tiempo que tarda en llegar al ordenador remoto es de 2 segundos, dando un total de 23 segundos desde la lectura y emisión de la información. Los resultados de confiabilidad se analizan en la Tabla 4.9.

**Tabla 4.9:** Parámetros de latencia de envío de información.

| **Parámetros** | |
|---|---|
| Paquetes enviados ($N\_env$) | 100 |
| Paquetes recibidos ($N\_rec$) | 96 |
| Paquetes perdidos ($N\_perd$) | 4 |
| Packet Delivery Ratio (PDR) | 96 % |
| Packet Loss Rate (PLR) | 4 % |
| Intervalo de transmisión | 21 segundos |
| Latencia promedio | 2 segundos |
| Tiempo total ciclo | 23 segundos |
| Distancia nodo–Gateway (cerrado) | 100 metros |
| Entorno | Abierto |

# CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

## 5.1. Conclusiones

El desarrollo del sistema mostró que es posible combinar ROS con LoRaWAN para monitorear de manera remota los robots móviles y recibir alertas de fallos críticos en tiempo real. Esto permitió garantizar que los eventos importantes fueran notificados de manera rápida y confiable, reduciendo la necesidad de supervisión constante y mejorando la seguridad en la operación del robot. Además, se demostró que esta integración puede escalarse a más robots o entornos industriales, haciendo el sistema versátil y eficiente para futuras aplicaciones.

Se identificaron claramente los eventos que requieren atención inmediata, como fallas de sensores, perdida de localización, balanceo, bloqueos de movimiento o caídas de voltaje, lo que permitió priorizar la información más relevante para el operador. También se definieron las especificaciones técnicas necesarias para que ROS y LoRaWAN trabajen de manera coordinada, asegurando que los mensajes lleguen de forma correcta y oportuna. Esto facilitó que el sistema sea confiable y minimice la pérdida de datos durante la operación.

El diseño del sistema permitió organizar de manera clara cómo se detectan los eventos en el robot y cómo se envían las notificaciones a través de LoRaWAN. Se pensó en la modularidad y la escalabilidad, de modo que sea fácil agregar más sensores o nodos en el futuro. Además, se incluyeron medidas de seguridad y protección ante posibles fallos de voltaje o pérdidas de señal, asegurando que el sistema sea robusto y confiable incluso en condiciones adversas.

La implementación del sistema permitió comprobar que el diseño funcionaba de manera efectiva en un robot real, transmitiendo alertas en tiempo real y con alta confiabilidad. Durante las pruebas, el robot pudo notificar fallos críticos. Esta experiencia confirmó que el sistema es práctico y útil, aportando seguridad y control sobre los robots móviles autónomos en entornos industriales.

## 5.2. Recomendaciones

Se propone implementar un método de confirmación donde el robot se asegure de que la información llegó a su destino y, si no es así, la reenvíe automáticamente. Esto es clave en lugares con muchas interferencias. Además, hacer pruebas a distancias más largas nos ayudará a conocer el límite real del sistema y garantizar que el monitoreo no se corte cuando el robot se aleje.

Un punto muy importante es que el robot no dependa siempre de estar conectado. Si añadimos una memoria interna para guardar eventos y sensores de respaldo, el equipo podrá seguir trabajando aunque la señal falle por un momento; los datos se guardarán localmente y se enviarán en cuanto la conexión vuelva. Al mismo tiempo, limpiar y actualizar el código hará que todo funcione más rápido y consuma menos batería, lo que facilita que en el futuro podamos añadir más robots o sensores sin que el sistema se vuelva lento.

Por último, el éxito a largo plazo depende de un buen plan de mantenimiento. No basta con instalar el equipo; hay que revisar periódicamente que las baterías estén en buen estado, que los módulos de radio funcionen bien y que los sensores midan correctamente. Al adelantarnos a estos pequeños problemas antes de que el robot falle en plena tarea, aseguramos que todo el sistema sea estable, seguro y útil durante mucho tiempo.

## 5.3. Trabajo futuro

El desarrollo de este sistema sienta las bases para diversas líneas de optimización que buscan elevar la autonomía y la robustez del robot en entornos operativos reales. A continuación, se detallan las propuestas de mejora prioritarias:

- **Migración a comunicación inalámbrica integral:** Se proyecta eliminar por completo la dependencia de cables mediante la implementación de un enlace de datos vía Bluetooth entre el microcontrolador y el microcomputador. Esta transición no solo busca una estética más limpia, sino que es fundamental para evitar desconexiones accidentales durante el desplazamiento. Para asegurar la integridad de la señal en distancias mayores, se contempla la integración de un *gateway* adicional que actúe como puente de comunicación, minimizando la latencia y la pérdida de paquetes.

- **Optimización del software y modernización de nodos:** El rendimiento del sistema puede potenciarse significativamente mediante la refactorización del código actual. Se propone la actualización de los nodos de ROS (*Robot Operating System*) y el uso de librerías

de bajo consumo de recursos. La implementación de protocolos más robustos y técnicas de depuración de memoria (*memory debugging*) permitirá que el procesamiento sea más ágil, garantizando que el hardware responda con precisión en escenarios industriales exigentes.

- **Gestión inteligente de datos y eventos críticos:** Una vía de investigación clave es el desarrollo de algoritmos de transmisión selectiva. En lugar de saturar el ancho de banda con telemetría redundante, el sistema debe ser capaz de priorizar eventos críticos y alertas de error sobre los datos de menor relevancia. Esta jerarquización de la información asegurará que las notificaciones de seguridad se transmitan en tiempo real, consolidando una arquitectura de supervisión remota mucho más eficiente y segura.

# Bibliografía


[1] J. J. P. Montes, J. González-Monroy, and C. G. Andrades, "Gamificación en robótica móvil usando ros2 y coppeliasim," *Jornadas de Automática*, 7 2024.

[2] G. C. J. Andrew, "Robot mÓvil de alto torque: Desarrollo mecÁnico," Ph.D. dissertation, Universidad Técnica del Norte, 2 2025.

[3] E. S. M. Borja, "Robot mÓvil de alto torque: Desarrollo electrÓnico," Ph.D. dissertation, Universidad Técnica del Norte, 2 2025.

[4] M. G. Gordillo, "Análisis legal del uso de los robots en la medicina legal analysis of the use of robots in medicine resumen palabras clave," 2023. [Online]. Available: https://dx.doi.org/10.12795/IETSCIENTIA

[5] D. A. G. Betancourt, "Análisis de seguridad en redes lora," Ph.D. dissertation, Universidad de Manizales, 2024.

[6] D. Azcurra and S. Rodríguez, "Arquitecturas de control para robots autÓnomos mÓviles didÁcticos basadas en sistemas embebidos," Universidad Nacional de Tres de Febrero, Tech. Rep., 2012. [Online]. Available: https://sedici.unlp.edu.ar/bitstream/handle/10915/19208/Documento_completo.pdf?sequence=1&isAllowed=y

[7] G. Murillo, "Sistema de control para la coordinación del movimiento grupal de robots autónomos móviles omnidireccionales de tres llantas," Universidad Iberoamericana Puebla, Tech. Rep., 2024. [Online]. Available: http://repositorio.iberopuebla.mx/licencia.pdf

[8] A. F. T. Pilay, "Diseño de un sistema de control autónomo para estabilizar un robot submarino en la captura de imágenes y reconocimiento de especie marina estrella de mar mediante pixhawk y visión artificial," UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA, Tech. Rep., 3 2025. [Online]. Available: https://repositorio.upse.edu.ec/bitstream/46000/13200/1/UPSE-TET-2025-0002.pdf

[9] M. A. Altamirano, M. Ángel Quiroz García, C. R. G. Escarpeta, C. V. P. Balderas, and A. P. Velázquez, "Estudio de cortocircuito mediante la simulación de sistemas eléctricos utilizando matlab, para mejorar la evaluación de impactos en la red. una contribución al manual de prácticas de ingeniería eléctrica en el tecnm," *South Florida Journal of Development*, vol. 6, p. e5043, 3 2025. [Online]. Available: https://ojs.southfloridapublishing.com/ojs/index.php/jdev/article/view/5043

[10] D. M. C. Martínez, "Sistema de emulación por software embebido de baterías li-ion con bajo error," Ph.D. dissertation, Universidad Autónoma de Querétaro, 8 2023.

[11] R. F. Hurtado-Guevara, "Impacto de la automatización en la auditoría: Ventajas y desafíos," *Revista Científica Zambos*, vol. 3, pp. 30–43, 9 2024. [Online]. Available: https://revistaczambos.utelvtsd.edu.ec/index.php/home/article/view/56

[12] M. A. O. VIÑAN and J. M. P. RAMOS, "Dispositivos utilizados en ingeniería electrónica para el control de la automatizacion industrial," *E-IDEA 4.0 Revista Multidisciplinar*, vol. 3, pp. 21–31, 3 2021. [Online]. Available: https://revista.estudioidea.org/ojs/index.php/mj/article/view/152

[13] F. Kleinubing and F. H. V. Okulczyk, "Diseño y desarrollo de sistema de monitoreo y registro de eventos en generadores de energía eléctrica," *Facultad de Ingeniería, Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina*, 2024. [Online]. Available: https://autoresjidetev.fio.unam.edu.ar/index.php/jidetev/article/view/69/854

[14] V. Hernandez-Rodriguez, D. Kairuz-Cabrera, A. Martinez-Laguardia, P. Merino-Laso, and O. S. RESUMEN, "Estación meteorológica iot basada en ttgo t-beam y comunicación lora title: Iot weather station based on ttgo t-beam and lora communication," *RIELAC*, vol. 44, p. 2302, 2023.

[15] J. E. Q. Vega, "Diseño e implementación de un robot móvil autónomo para reducir contagios de covid-19 en el servicio urbano de delivery de comida con carga máxima de hasta 10 kg," Ph.D. dissertation, Universidad Ricardo Palma, 2023.

[16] A. Tirado-Bou, R. Marín-Prades, L. Baiguera-Tambutti, P. J. Sanz, and J. V. Martí, *Desarrollo de una interfaz para el prototipado y validación de un robot móvil autónomo de uso hospitalario*. Servizo de Publicacións da UDC, 9 2022, pp. 156–164.

[17] J. A. G. Vasquez, M. E. C. Arroyo, M. M. H. Maguiña, M. K. L. Asto, L. M. S. Quincho, and A. F. V. Cabrera, "Internet of things technologies applied in the supply chain. a systematic review," *Gestión de Operaciones Industriales*, vol. 2, pp. 8–26, 7 2023.

[18] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, "Robot operating system 2 (ros2)-based frameworks for increasing robot autonomy: A survey," 12 2023.

[19] P. S. H. Rojas, "Robot mÓvil de alto torque: VisiÓn artificial," Ph.D. dissertation, Universidad Técnica del Norte, 2 2025.

[20] D. M. Calle, "Seguridad en dispositivos iot para redes lora," Ph.D. dissertation, Universidad de los Andes, 2023.

[21] A. E. P. Toala and D. S. R. Manosalvas, "Diseño e implementación de un sistema multi-robot de código abierto para ambientes colaborativos en ros2." Ph.D. dissertation, Escuela Superior Politécnica del Litoral, 2023.

[22] B. Martillo and E. Daniel, "Implementación de un robot móvil para el control de tránsito y movilidad usando visión artificial," Tech. Rep., 2023.

[23] N. D. Munoz-Ceballos and G. Suarez-Rivera, "Performance criteria for evaluating mobile robot navigation algorithms: a review," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 19, pp. 132–143, 2022.

[24] C. Juárez, "Continúa creciendo el uso de robots móviles en el almacenamiento y la logística," 12 2024. [Online]. Available: https://thelogisticsworld.com/actualidad-logistica/continua-creciendo-el-uso-de-robots-moviles-en-el-almacenamiento-y-la-logistica/

[25] J. Y. C. Castillo, "Diagnóstico de fallas en sistemas lpv basado en observadores intervalares," Ph.D. dissertation, Tecnológico Nacional de México, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), 2024.

[26] M. Mendiguren, "Lorawan, nuevo backup de comunicaciones para sistemas de seguridad," *SEGUROLATAM*, p. 115, 2021. [Online]. Available: https://alaisecure.co/pdfs/noticias/alai-secure-segurilatam-lorawan-backup-comunicaciones-seguridad.pdf

[27] Robotnik, "Ros 2 (robot operating system): visión general y puntos clave del software de robótica," 4 2025. [Online]. Available: https://robotnik.eu/es/ros-2-robot-operating-system-vision-general-y-puntos-clave-del-software-de-robotica/

[28] O. Robotics, "Understanding ros2 nodes," 2025. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html

[29] J. E. Cabrera, "Ros2. publicadores y subscriptores," 8 2022. [Online]. Available: https://robotica-facil-con-ros2.es/?p=3821

[30] O. Robotics, "About ros 2 interfaces," 2022. [Online]. Available: https://docs.ros.org/en/foxy/Concepts/About-ROS-Interfaces.html

[31] G. P. Garrido, Álvaro Castro González, J. C. C. Montoya, M. Ángeles Malfaz Vázquez, and M. S. Sánchez-Caballero, "Una propuesta de integración de una red de sensores iot en el robot mini para monitorización del usuario en su domicilio," *Jornadas de Automática*, 7 2024.

[32] M. Sanchez and J. Tutor, "Monitorización de rendimiento y diagnóstico de un nodo lorawan basado en esp32," Ph.D. dissertation, UNIVERSITAT POLITÈCNICA DE VALÈNCIA, 2025.

[33] H. J. Mohammed, K. Hama, and A. Faraj, "Python-wsgi and php-apache web server performance analysis by search page generator (spg)," *UKH Journal of Science and Engineering*, vol. 5, 2021.

[34] E. Gallego and Adrián, "Communicenter, una aplicación web para gestionar las comunicaciones de empresas con sus clientes." Ph.D. dissertation, Universidad Complutense de Madrid., 2024.

[35] M. I. Z. Hasibuan and T. Triase, "Implementasi sistem database nosql secara realtime menggunakan firebase realtime database pada aplikasi ourticle," *SIBATIK JOURNAL: Jurnal Ilmiah Bidang Sosial, Ekonomi, Budaya, Teknologi, dan Pendidikan*, vol. 2, pp. 1–24, 12 2022.

[36] HERNÁNDEZ-Verónica, K. J. Carlos, A. Héctor, R. Norma, and B. Irving, "Actualización y estandarización de señalización de acuerdo a la norma nom-026-stps-2008 en el área de almacén de recibo," *Tecnológico Nacional de México*, vol. 9, 4 2022.

[37] M. S. Martínez and . Autor, "Aplicación de norma iso 9241-11 para la evaluación de la usabilidad en simuladores de vuelo application of iso 9241-11 standard for the evaluation of usability in flight simulators," Tech. Rep., 2022.

[38] G. T. Grajales, "Tipos de investigacion," p. 1, 3 2000. [Online]. Available: https://cursa.ihmc.us/rid=1RM1F0L42-VZ46F4-319H/Investigaci%C3%B3n.pdf

[39] J. O. Lozada, "Investigación aplicada: Definición, propiedad intelectual e industria," Tech. Rep., 2014.

[40] C. Ramos-Galarza, "Editorial: Diseños de investigación experimental," *CienciAmérica*, vol. 10, pp. 1–7, 2 2021.

[41] Q. C. Tancara, "La investigacion documental," Tech. Rep., 2010. [Online]. Available: http://revistasbolivianas.umsa.bo/pdf/rts/n17/n17a08.pdf

[42] Nvidia, "Issue with ch340 usb-to-serial converter not creating device files on jetson orin nano super," 2025. [Online]. Available: https://forums.developer.nvidia.com/t/issue-with-ch340-usb-to-serial-converter-not-creating-device-files-on-jetson-orin-nano-super/326022

[43] O. Vargas, O. Flor, and F. Suárez, "Construcción y pruebas de funcionamiento de un prototipo robótico para prótesis humana," Tech. Rep., 2020. [Online]. Available: https://www.redalyc.org/articulo.oa?id=573261514005

[44] Y. Technology, "Yahboom jetson orin nano super ai large model developer kit 8gb," 2024. [Online]. Available: https://www.robotshop.com/products/yahboom-jetson-orin-nano-super-ai-large-model-developer-kit-8gb-sub-ai-large-model-kit

[45] Arduino, "Arduino uno rev3 documentation," 2026. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3/

[46] A. V. G. Rojas, C. X. R. C. Asesor, and M. A. C. Sortino, "Sistema de monitoreo remoto de oscilaciones en estructuras sometidas a carga dinámica utilizando tecnología iot," Ph.D. dissertation, UNIVERSIDAD T ´ECNICA DEL NORTE, 2025.

[47] R. Inc., "Syd dynamics transducerm tm171 9-axis ahrs w/ dual-port communication," 2026. [Online]. Available: https://ca.robotshop.com/products/syd-dynamics-transducerm-tm171-9-axis-ahrs-w-dual-port-communication?qd=caf276fb272dbd7931b33df3a8618732

[48] L. D. T. Co., "User manual for rs485-lb waterproof rs485/uart to lorawan converter," 12 2025. [Online]. Available: https://wiki.dragino.com/xwiki/bin/view/Main/

User%20Manual%20for%20LoRaWAN%20End%20Nodes/RS485-LB_Waterproof_RS485UART_to_LoRaWAN_Converter/

[49] Megatrónica, "Módulo gps ublox neo-6m v2 para arduino y raspberry," 2026. [Online]. Available: https://megatronica.cc/producto/modulo-gps-ublox-neo-6m-v2-arduino-raspberry/?srsltid=AfmBOoo2sUiHKouwxAS9wXhXTPQEMSXqFqoGzZH7Ms9V21qntdQgTObb

[50] I. A. L. Reinoso, "Red iot comunitaria con tecnología lora para cultivos urbanos inteligentes: Aplicación," Ph.D. dissertation, UNIVERSIDAD TÉCNICA DEL NORTE, 2024.