UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACSISTEL

INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

Caracterización y modelado bidimensional de canales de propagación en interiores mediante mapas generados con LiDAR 2D de bajo costo.

Karla del Rocío Calvache Torres

Dirigido por:
Ing. Carlos Andrade Caicho, M.Sc.

La Libertad - 2026

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

Dr. Ronald Humberto Rovira Jurado, Ph.D.
DIRECTOR DE LA CARRERA

Ing. Carlos Efraín Andrade Caicho, M.Sc.
DOCENTE TUTOR

Ing. Daniel Armando Jaramillo Chamba, Mgtr.
DOCENTE ESPECIALISTA

Ing. Luis Miguel Amaya Farjño, Mgtr.
DOCENTE GUÍA UIC

Ing. Corina Raquel Gonzabay De La A, Mgtr.
SECRETARIA

UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por **KARLA DEL ROCIO CALVACHE TORRES**, como requerimiento para la obtención del título de Ingeniero en Telecomunicaciones.

La Libertad, a los 19 días del mes de agosto del año 2026

TUTOR

Ing. Carlos Efraín Andrade Caicho, M.Sc.

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Karla del Rocío Calvache Torres**

DECLARO QUE:

El trabajo de titulación, **CARACTERIZACIÓN Y MODELADO BIDIMENSIONAL DE CANALES DE PROPAGACIÓN EN INTERIORES MEDIANTE MAPAS GENERADOS CON LIDAR 2D DE BAJO COSTO**, previo a la obtención del título de Ingeniero en telecomunicaciones, ha sido desarrollado respetando derechos intelectuales de terceros conforme la citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías.

En consecuencia, este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del trabajo de titulación referido.

La Libertad, a los 19 días del mes de agosto del año 2026

EL AUTOR

Karla del Rocío Calvache Torres.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado **CARACTERIZACIÓN Y MODELADO BIDIMENSIONAL DE CANALES DE PROPAGACIÓN EN INTERIORES MEDIANTE MAPAS GENERADOS CON LIDAR 2D DE BAJO COSTO**, presentado por la estudiante, **Karla del Rocío Calvache Torres** fue enviado al Sistema Antiplagio, presentando un porcentaje de similitud correspondiente al 04 %, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


Analysis attestation
Compilatio Magister+ | UPSE-ECU

Tesis_Calvache Torres Karla (1)

4%
Suspicious texts

File name
Original file size
Number of words

Submitter
Submission date
Upload type
analysis end date

TUTOR

Ing. Carlos Efraín Andrade Caicho, M.Sc.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo **Karla del Rocío Calvache Torres.**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la institución.

Cedo los derechos en línea patrimoniales de artículo profesional de alto nivel con fines de difusión pública, además, apruebo la reproducción de este artículo académico dentro de las regulaciones de la universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

La Libertad, a los 19 días del mes de agosto del año 2026

EL AUTOR

Karla del Rocío Calvache Torres.

AGRADECIMIENTO

"*Hay hombres que nacen un día y son buenos. Hay otros que luchan un año y son mejores. Hay quienes luchan muchos años y son muy buenos. Pero hay los que luchan toda la vida: esos son los imprescindibles."*

(Bertolt Brecht)

Expreso un agradecimiento eterno a Dios por darme la vida, la salud y fortaleza para saber sobrellevar cada etapa en este camino. Su bendición ha sido mi guía constante, porque sin Él nada soy.

A Samuel Alvarado que ha sido uno de los pilares fundamentales en mi vida. Gracias por estar en todo momento y acompañarme a cada aventura de este camino llamada vida, por esa paciencia infinita que me brindas y por tu amor incondicional.

A mis hijos Kelly, Jeremy y Samuel ustedes son el motor que me impulsa a seguir luchando y a no rendirme por muy dura que sea la batalla. Gracias por todo el amor que me brindan y ser mi refugio.

A mi madre Reina Torres por enseñarme a ser esa mujer fuerte y guerrera y ser mi mayor ejemplo de lucha. Gracias por creer en mí y este logro también es suyo.

A la Facultad de FACSISTEL, por ser mi segundo hogar en esta etapa universitaria. Agradezco a cada una de las personas que la integran y compartieron sus conocimientos y guiarme hacia este logro.

A los docentes Carlos Andrade y Daniel Jaramillo, por enseñarme que la exigencia es fundamental para moldearnos profesionalmente. Su guía fue la clave para cumplir la meta y dar lo mejor de mí.

A mis compañeros de clase que con el pasar de los años se convirtieron en mis más grandes amigos. Un agradecimiento especial a Estefanía, Michael por formar parte de mi vida y brindarme esa amistad sincera.

Karla del Rocío Calvache Torres

DEDICATORIA

"No te rindas, por favor no cedas, aunque el frío queme, aunque el miedo muerda, aunque el sol se esconda y se calle el viento, aún hay fuego en tu alma, aún hay vida en tus sueños."

(Mario Benedetti)

Con el poema de "No te rindas", dedico este trabajo de titulación a quienes han sido parte primordial de mi vida, en los momentos de cansancio y tribulación. Dios, por ser quien ilumina en la penumbra, y darme la fortaleza para no ceder ante el miedo. Demostrándome que aún en la tormenta siempre hay fuego en el alma para continuar.

A Samuel Alvarado, por ser parte de este proceso, mi apoyo en los momentos difíciles y no dejarme caer cuando las fuerzas me faltaban.

A mis hijos Kelly, Jeremy, Samuel, por estar ahí sacándome una sonrisa cuando más lo necesitaba y recordándome que todo esfuerzo es bien recompensado.

Karla del Rocío Calvache Torres.

RESUMEN

El despliegue de redes inalámbricas en espacios cerrados (como hospitales, oficinas o laboratorios) enfrenta retos distintos a los de ambientes abiertos. En interiores, la señal se ve afectada por la presencia de paredes, mobiliarios y equipos metálicos generando fenómenos de reflexión, difracción y atenuación que modifican su comportamiento. Este trabajo aborda esa problemática mediante un sistema experimental que utiliza sensores LiDAR 2D de bajo costo para caracterizar y modelar los canales de propagación en tiempo real.

El problema de cómo describir los espacios interiores para analizar la propagación de señales inalámbricas es un foco de este estudio. Los modelos de siempre tienen problemas porque no miran con suficiente detalle la forma real de los espacios. Por esto, se implementó una estructura donde un robot explorador con un sensor LiDAR 2D de bajo costo (RPLIDAR A1) explora el lugar, recolectando distancias, movimientos e imágenes. A la vez, un robot procesador recibe esto por MQTT como Publisher/Subscriber y corre programas SLAM (Hector SLAM) para crear mapas bidimensionales de ocupación enseguida.

Resultados obtenidos en el laboratorio de telecomunicaciones confirman que el sistema exhibe una latencia promedio de 335 milisegundos, demanda un ancho de banda MQTT de 184.1 KB/s, y presenta una autonomía de batería de aproximadamente 3.78 horas de operación constante. Los mapas generados alcanzan una resolución de 2.5 cm por celda, cubriendo áreas de hasta 655 m², con un consumo moderado de recursos computacionales (CPU 25-35%, memoria 210-395 MB).

El modelo ajustado presentó un error cuadrático medio (RMSE) de 2.64 dB y un exponente de pérdidas de 3.78, lo que valida la caracterización del canal 161 en la banda de 5 GHz.

De esta forma el proyecto sienta las premisas necesarias para el futuro modelado computacional de los canales de propagación; las representaciones 2D resultantes, al incluir datos espaciales detallados como dimensiones, la finura de la imagen, punto de inicio y la disposición de elementos físicos que pudieran obstruir, facilitarán en etapas subsiguientes la transformación a representaciones tridimensionales y la emulación del comportamiento de señales inalámbricas (Wi-Fi, 5G, IoT). Todo esto podrá ser logrado por medio de algoritmos de ray-tracing y el cálculo de presupuestos de enlace, liberando de la necesidad de adquirir equipos costosos.

Palabras clave: LiDAR, Hector SLAM, entornos cerrados, ROSMASTER, MQTT.

ABSTRACT

The deployment of wireless networks in enclosed environments such as hospitals, offices, or laboratories faces challenges that differ from those in open spaces. Indoors, signals are affected by walls, furniture, and metallic equipment, producing reflection, diffraction, and attenuation phenomena that alter their behavior. This study addresses these issues through an experimental system that employs low-cost 2D LiDAR sensors to characterize and model indoor propagation channels in real time.

A central focus of this research is the problem of accurately describing indoor spaces to analyze wireless signal propagation. Traditional models often fail because they do not capture the actual geometry of environments in sufficient detail. To overcome this limitation, a system was implemented in which an explorer robot equipped with a low-cost 2D LiDAR sensor (RPLIDAR A1) scans the environment, collecting distance, motion, and image data. Simultaneously, a processor robot receives this information via MQTT under the Publisher/Subscriber model and executes SLAM algorithms (Hector SLAM) to generate two-dimensional occupancy maps in real time.

Results obtained in the telecommunications laboratory confirm that the system exhibits an average latency of 335 milliseconds, requires an MQTT bandwidth of 184.1 KB/s, and achieves a battery autonomy of approximately 3.78 hours of continuous operation. The generated maps reach a resolution of 2.5 cm per cell, covering areas up to 655 m^2, with moderate computational resource consumption (CPU 25–35%, memory 210–395 MB).

The adjusted model presented a root mean square error (RMSE) of 2.64 dB and a path loss exponent of 3.78, which validates the channel 161 characterization in the 5 GHz band.

This project establishes the foundations for future computational modeling of propagation channels. The resulting 2D representations, by incorporating detailed spatial data such as dimensions, resolution, origin, and the arrangement of physical elements, will facilitate

subsequent transformation into three-dimensional models and the simulation of wireless signal behavior (Wi-Fi, 5G, IoT). These advances can be achieved through ray-tracing algorithms and link budget calculations, reducing the need for costly equipment.

Keywords: LiDAR, Hector SLAM, closed environments, ROSMASTER, MQTT.

INTRODUCCIÓN

El avance de la tecnología inalámbrica ha cambiado nuestro estilo de vida y nuestro entorno de trabajo sigue evolucionando, con requisitos de más estabilidad y confiabilidad en la conexión. Aplicaciones como el acceso a Internet de alta velocidad mediante Wi-Fi, el Internet de las cosas (IoT), la comunicación móvil de quinta generación, los entornos automatizados, etc., requieren el mejor rendimiento en todos los entornos, incluso en los más complejos.

Varias universidades en el Ecuador han utilizado sensores LiDAR 2D de bajo costo en proyectos de robótica autónoma y han demostrado que se pueden utilizar para la elaboración de mapas en 2D y detección de obstáculos. Sin embargo, gran parte de la investigación en este campo se ha centrado en la navegación robótica y se ha prestado poca atención a la fusión de mapas con modelos de propagación de señales inalámbricas. Esta falta de conexión abre la posibilidad de relacionar la percepción del espacio con el análisis de los canales de comunicación.

En consecuencia, el objetivo de este trabajo es llenar este vacío mediante el desarrollo de un sistema de red de dos robots ROSMASTER X3 equipados con escáneres LiDAR, que se comunican entre sí utilizando el protocolo MQTT (Message Queuing Telemetry Transport) en la arquitectura publicador-suscriptor, proporcionando comunicación en tiempo real y sin conexión. En este sentido, se generan mapas de ocupación que representan con precisión la geometría de los interiores, que se utilizan en la modelización de la propagación de señales inalámbricas.

En los experimentos de nivel de señal se ha verificado la validez del método, al tiempo que se ha realizado un ajuste del modelo de propagación a los datos obtenidos. Este modelo puede ser de interés para desarrollos de representación 3D y simulación de propagación, en procesos de diseño y optimización de redes WiFi, 5G e IoT en interiores.

ÍNDICE GENERAL

TRIBUNAL DE SUSTENTACIÓN ..... **II**

CERTIFICACIÓN ..... **III**

DECLARACIÓN DE RESPONSABILIDAD ..... **IV**

CERTIFICACIÓN DE ANTIPLAGIO ..... **V**

AUTORIZACIÓN ..... **VI**

AGRADECIMIENTO ..... **VII**

DEDICATORIA ..... **IX**

RESUMEN ..... **X**

ABSTRACT ..... **XII**

INTRODUCCIÓN ..... **XIV**

ÍNDICE GENERAL ..... **XV**

ÍNDICE DE ILUSTRACIONES ..... **XIX**

ÍNDICE DE TABLAS ..... **XXI**

ÍNDICE DE ABREVIATURAS ..... **XXII**

CAPÍTULO I **25**

MARCO CONTEXTUAL DE LA INVESTIGACIÓN 25

1.1. Antecedentes 25

1.2. Planteamiento del problema 26

1.3. Descripción del proyecto 27

1.4. Objetivos del proyecto 29

1.4.1. Objetivo general: 29

1.4.1.1. Objetivos específicos: 29

1.5. Justificación 30

1.6. Metodología 33

1.6.1. Fase de revisión técnica 34
1.6.2. Fase de adquisición de datos 35
1.6.3. Fase de comunicación y sincronización de datos 35
1.6.4. Fase de procesamiento de datos 36
1.6.5. Fase de Generación de mapas 36
1.6.6. Fase de desarrollo del modelo 36
1.6.7. Fase de evaluación del sistema 37
CAPÍTULO II 38
MARCO TEÓRICO DE LA INVESTIGACIÓN 38
2.1. Fundamentación teórica 38
2.1.1. Propagación de ondas electromagnéticas en interiores 38
2.1.2. Canales de propagación en sistemas inalámbricos 42
2.1.3. Modelado de la propagación en interiores 46
2.1.4. Sensores LiDAR 48
2.1.5. Generación de mapas mediante LiDAR 52
2.1.6. Sistema ROS (Robot Operating System) 54
2.1.7. Técnicas de mapeo y SLAM 57
2.1.8. Relación entre mapas LiDAR y propagación de señales 60
2.1.9. Comunicaciones en sistemas multi-robot 62
2.1.10. Tecnologías de comunicación inalámbrica en interiores 64
2.1.11. Parámetros de calidad de servicio (QoS) 66
2.1.12. Protocolos de comunicación 67
2.1.13. Sincronización de datos 68
2.1.14. Sistemas colaborativos multi-robot 68
2.1.15. Modelos por capas en redes de comunicación 68
2.1.16. Relación entre capas y calidad de servicio 72

2.1.17. Consideraciones avanzadas de comunicación en sistemas multi-robot 73

CAPÍTULO III **75**

DESARROLLO DE LA PROPUESTA 75

3.1. Componentes de la propuesta 75

3.1.1. Componentes físicos 75

3.1.2. Componentes lógicos 86

3.1.3 Equipos y herramientas de medición 90

3.2. Diseño de la propuesta 90

3.3 Implementación del sistema 93

3.3.1. Configuración del entorno de desarrollo 94

3.3.2. Configuración del broker MQTT 94

3.3.3 Desarrollo de la aplicación en Python 95

3.3.4 Configuración de Node-RED 96

3.3.5. Integración del sistema colaborativo 98

3.3.6 Arquitectura de comunicación MQTT 98

3.3.7 Configuración del robot explorador 100

3.3.8 Configuración del robot procesador 104

3.3.9 Arquitectura del diagrama del robot procesador 104

3.3.10 Arquitectura del diagrama de estación de trabajo 105

3.3.11 Arquitectura del diagrama de Datos Completo 106

3.3.12 Arquitectura del Flujo de Control 107

CAPÍTULO IV **108**

CÁLCULOS Y ANÁLISIS DE RESULTADOS 108

4.1 Pruebas en dos entornos 108

4.2 Funcionamiento del robot explorador 109

4.3 Funcionamiento del robot procesador 112
4.3.1 Funcionamiento del Hector SLAM 113
4.3.2 Validación de precisión del mapa generado 114
4.4 Rendimiento del sistema 115
4.4.1 Cálculos del sistema 115
4.4.2 Recursos computacionales y rendimiento 117
4.5 Caracterización del canal de propagación en interiores 118
4.5.1 Metodología de medición 118
4.5.2 Mediciones de radioeléctricas 119
4.5.3 Validación experimental de la precisión geométrica del mapa 123
4.5.4 Mapa del laboratorio 126
4.5.5 Evolución temporal del RSSI 127
4.5.6 Implementación del modelo de propagación log-distance 128
4.6 Formato y caracterización del mapa 2D 129
4.6.1 Formato del mapa 2D para conversión al modelo 3D 129
4.6.2 Enfoque en el modelado computacional 129
4.6.3 Generación y uso del modelado 3D 130
CONCLUSIONES **134**
RECOMENDACIONES **136**
PRESUPUESTO **137**
REFERENCIAS BIBLIOGRÁFICAS **138**
ANEXOS **149**
ANEXO A: Armado del ROSMASTER X3 Jetson Nano 4 GB 149
ANEXO B: Servicios systemd del robot explorador y procesador 150
ANEXO C: Script de Python 153
ANEXO D: Capturas de RSSI, latencia. canal en tiempo real en el CMD 154

ANEXO E: Guía de usuario del sistema 157

ANEXO F: Códigos de instalación agregados al repositorio de github incluidos los scripts 157

ÍNDICE DE ILUSTRACIONES

Ilustración 1:Diagrama de fases de la metodología del proyecto. 34
Ilustración 2: Fenómeno de reflexión. 41
Ilustración 3: Fenómeno de difracción. 41
Ilustración 4:Fenómeno de dispersión. 42
Ilustración 5: Fenómeno de atenuación. 42
Ilustración 6 ROSMASTER X3 Jetson Nano 4GB SUB 75
Ilustración 7: Sensor LiDAR RPLiDAR A1. 76
Ilustración 8: Cámara de profundidad. 77
Ilustración 9: Controlador inalámbrico. 78
Ilustración 10: Placa STM32. 78
Ilustración 11: Motores DC. 79
Ilustración 12: Jetson Nano 4GB. 80
Ilustración 13: Antenas WiFi. 81
Ilustración 14: Ruedas omnidireccionales. 81
Ilustración 15: Módulo de alimentación. 82
Ilustración 16: Cable USB y Disk. 83
Ilustración 17: Chasis del ROSMaster X3. 84
Ilustración 18: Python. 87
Ilustración 19 Rviz 87
Ilustración 20 Visual Studio Code 88
Ilustración 21: Diseño de la propuesta. 90
Ilustración 22 Instalación del Node-RED 96
Ilustración 23 Consola mosquitto 97
Ilustración 24 Interfaz Node-RED 97

Ilustración 25 Diagrama del Robot Explorador 101
Ilustración 26 Verificación del estado de los servicios systemd en el Robot Explorador 102
Ilustración 27 Reinicio de servicios systemd 103
Ilustración 28 Diagrama del Robot Procesador 105
Ilustración 29diagrama de Estación de trabajo 106
Ilustración 30Flujo de Datos Completo 107
Ilustración 31Flujo de Control 107
Ilustración 32 Pruebas en domicilio 108
Ilustración 33Robo Explorador 109
Ilustración 34 Robot explorador 110
Ilustración 35 Node-RED Dashboard 111
Ilustración 36 App Python 111
Ilustración 37 Robot Procesador 112
Ilustración 38 Hector SLAM 113
Ilustración 39 Visualización del Mapa en RVIZ 114
Ilustración 40Salida del comando iwconfig wlan0 119
Ilustración 41Comparativa RSSI vs distancia 121
Ilustración 42 Prueba de comunicación 122
Ilustración 43 Análisi de prueba de comunicación 123
Ilustración 44Fotografía del Laboratorio de telecomunicaciones 124
Ilustración 45Medición en el mapa generado por el SLAM 124
Ilustración 46 Comparación: Medida real vs Mapa 125
Ilustración 47 Análisis cuantitativo del error 126
Ilustración 48 Mapa del laboratorio 127
Ilustración 49 Evolución del RSSI 128
Ilustración 50 Modelo log-distance 128
Ilustración 51 Comando para guardar el mapa de ocupación 130

ÍNDICE DE TABLAS

Table 1Fenómenos de propagación de ondas electromagnéticas en interiores..... 41
Table 2Comparación entre canales interiores y exteriores 43
Table 3Comparación entre LiDAR 2D y LiDAR 3D. 50
Table 4Comparación de algoritmos de mapeo 2D. 59
Table 5: Influencia de materiales en la propagación 61
Table 6Comparación de tecnologías inalámbricas...... 65
Table 7Capas del modelo OSI y sus funciones...... 69
Table 8Capas del modelo TCP/IP. 71
Table 9Proceso de encapsulación...... 72
Table 10Factores que afectan la comunicación en sistemas multi-robot 73
Table 11Especificaciones del RPLIDAR A1..... 77
Table 12Especificaciones cámara de profundidad...... 77
Table 13Especificaciones controlador inalámbrico...... 78
Table 14Especificaciones placa STM32...... 79
Table 15Especificaciones motores DC...... 79
Table 16Especificaciones Jetson Nano 4GB...... 80
Table 17 Especificaciones de antenas Wi-Fi. 81
Table 18Especificaciones ruedas omnidireccionales...... 82
Table 19Especificaciones módulo de alimentación. 82
Table 20Especificaciones cable USB y Disk...... 83
Table 21Especificaciones del chasis...... 84
Table 22Función de los componentes de la arquitectura MQTT...... 98
T**able 23Tópicos MQTT utilizados durante la implementación.** 100
Table 24Software Instalado..... 100
Table 25Servicios del Robot Explorador..... 101
Table 26 Tópicos MQTT..... 103
Table 27Servicios Systemd Robot Procesador..... 104
Table 28 Dimensiones del Laboratorio obtenidas por el mapa..... 115
Table 29 Rendimiento General con desviación estándar) 118

Table 30 Mediciones realizadas en la banda 5GHz en el canal 161 120
Table 31 Resultados de ping 122
Table 32 Validación comparativa 125
Table 33 Dimensiones generales 126
Table 34 Presupuesto del Proyecto 137

ÍNDICE DE ABREVIATURAS

API	Application Programming Interface
ARM	Advanced RISC Machine
BLE	Bluetooth Low Energy
BSP	Board Support Package
CAN	Controller Area Network
CPU	Central Processing Unit
CSI	Camera Serail Interface
CUDA	Compute Unified Device Architecture
DDS	Data Distribution Service
FFT	Fast Fourier Transform
GPG	GNU Privacy Guard
GPS	Global Positioning System
GPU	Graphical Processing Unit
GUI	Graphical User Interface

HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electriccal and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
LiDAR	Light Detection and Ranging
MQTT	Message Queuing Telemetry Transport
NTP	Network Time Protocol
NVMe	Non-Volatile Memory Express
OLED	Organic Light-Emitting Diode
OSI	Open System Interconnection
PC	Personal Computer
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
QoS	Quality of Service
RAM	Random Access Memory
RGB	Red, Green, Blue
ROS	Robot Operating System

ROS2 Robot Operating System 2

SLAM Simultaneous Localization and Mapping

SSD Solid State Drive

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

UI User Interface

USB Universal Serial Bus

VNC Virtual Network Computing

Wi-Fi Quality Wireless Fidelity

CAPÍTULO I
MARCO CONTEXTUAL DE LA INVESTIGACIÓN

1.1. Antecedentes

En Ecuador, también se han realizado investigaciones sobre el uso de los sensores LiDAR para la percepción del entorno y la reconstrucción de mapas en aplicaciones de ingeniería. En la Universidad San Francisco de Quito, por ejemplo, se diseñó e implementó un sistema de navegación autónoma utilizando un sensor LiDAR. Su función básica es la detección de obstáculos y la creación de un mapa del entorno en que se ubica. Se le han acoplado plataformas computacionales y algoritmos de percepción de tal forma que el sensor entrega mediciones de distancia en forma continua. Con estos datos se han creado mapas en dos dimensiones del entorno, además de demostrar que los sensores LiDAR sirven para crear un mapa de calidad suficiente para detectar características del entorno, como por ejemplo la detección de paredes u otros objetos en el entorno. Este tipo de información geométrica es especialmente importante en los espacios interiores, donde la disposición de los objetos materiales puede afectar el comportamiento de varios elementos en el entorno, como la propagación de ondas electromagnéticas en un sistema de comunicación inalámbrica. [2]

De manera similar, en la Universidad Politécnica Salesiana se desarrolló un proyecto orientado al diseño y a la construcción de un sistema autónomo equipado con tecnología LiDAR 2D para la navegación en entornos con obstáculos. En este caso, se utilizó un sistema de adquisición de datos mediante escaneo láser del entorno que permite medir distancias y obtener representaciones bidimensionales del entorno escaneado.

Con los datos obtenidos a partir del sensor, se aplicaron algoritmos de procesamiento y segmentación en los que se generaron mapas del entorno, se detectaron obstáculos y se identificaron otros elementos arquitectónicos en el escenario. Los resultados muestran que los sensores LiDAR 2D

son una herramienta muy útil para la creación de mapas en entornos interiores, permitiendo obtener información espacial muy detallada.

No obstante, a la vista de los resultados, se puede concluir que el uso de sensores LiDAR es válido para la obtención del modelo geométrico de un entorno interior, aunque no existen sistemas que integren modelos de propagación de señales en un entorno interior y arquitecturas de cooperación de múltiples plataformas robóticas [3].

1.2. Planteamiento del problema

El rápido crecimiento de las tecnologías inalámbricas ha incrementado la necesidad de comprender cómo se comportan las señales de radio en espacios interiores, especialmente en lugares donde se utilizan redes WiFi, sistemas de comunicación IoT y otras aplicaciones basadas en transmisión de datos. En estos entornos, la propagación de las señales se ve afectada por factores físicos como paredes, mobiliario, materiales de construcción y la propia geometría del espacio, generando fenómenos de reflexión, difracción y atenuación.

Uno de los problemas que hay que resolver a la hora de investigar los canales de propagación en interiores es que en ocasiones no se puede obtener una representación exacta del medio en el que tiene lugar la propagación. Los modelos más utilizados se basan en aproximaciones teóricas o en la toma de datos que no siempre se ajustan a la realidad espacial. Como resultado, se pueden producir errores en las estimaciones del comportamiento de la señal en espacios cerrados.

En paralelo, los sensores LiDAR (Light Detection and Ranging) han demostrado que son un sensor alternativo, proporcionando información precisa de la geometría del entorno, mediante medidas láser de distancia. En este sentido, sensores de bajo coste 2D permiten la obtención de mapas 2D de entornos interiores, que pueden ser usados para la detección de obstáculos y la identificación de los elementos estructurales de la escena.

A pesar de estas ventajas, el uso de LiDAR para el modelado de la propagación de canales de señales en interiores no está generalizado en los laboratorios académicos, lo que demuestra la necesidad de herramientas y metodologías que permitan utilizar la información espacial recogida con LiDAR a la hora de modelar la propagación de señales en interiores.

Por ello, este trabajo se centra en lograr un modelo 2D de caracterización de canales de propagación en interiores, que se genera a partir de mapas en 2D obtenidos mediante un sensor LiDAR de bajo coste, para lograr una mayor precisión en el modelado de la geometría del entorno y un análisis más detallado del impacto de esta en el comportamiento de las señales inalámbricas.

1.3. Descripción del proyecto

El presente proyecto tiene como finalidad desarrollar e implementar un sistema colaborativo basado en dos robots móviles ROSMASTER X3 para la caracterización y modelado bidimensional de canales de propagación en entornos interiores mediante mapas generados con un sensor LiDAR 2D de bajo costo.

Esto permite, mediante la fusión de datos de robótica móvil, comunicaciones inalámbricas y procesamiento de datos, obtener una representación del entorno que será utilizada para el análisis de la propagación de señales.

Esta arquitectura se basa en la implementación de dos plataformas robóticas que interactúan bajo un modelo de computación distribuida, donde cada uno de los robots que componen el sistema lleva a cabo una parte de la tarea, con el objetivo de hacer un uso más eficiente de los recursos de computación, aumentar la eficiencia del proceso y facilitar la escalabilidad de la solución.

El primer robot, denominado Robot Explorador, es el encargado de recorrer el entorno de estudio ubicado en el laboratorio de telecomunicaciones. Durante su desplazamiento realiza

la adquisición de información espacial mediante un sensor LiDAR 2D, obteniendo mediciones de distancia correspondientes a los diferentes obstáculos presentes en el escenario. Además del sensor LiDAR, el robot recopila información de odometría y estado del sistema, datos que permiten conocer con mayor precisión la posición y trayectoria recorrida durante el proceso de exploración.

Una vez obtenida la información se la envía al segundo robot utilizando el protocolo de comunicación MQTT (Message Queuing Telemetry Transport) . Ambos robots se unen a una red local inalámbrica, que se utiliza para intercambiar la información entre robots a través de un broker MQTT y el modelo de comunicación Publisher/Subscriber. Esto permite el desacoplamiento de la adquisición del procesamiento, y proporciona un método eficiente de comunicación entre las dos plataformas.

Un segundo robot, el Robot Procesador, actúa como nodo de procesamiento de la información que recibe. Este robot se suscribe a los diferentes tópicos MQTT que publica el Robot Explorador para recibir la información del sensor LiDAR, así como la odometría y el resto de variables que necesita para su correcto funcionamiento. Posteriormente, a partir del flujo de datos y el middleware ROS2, se filtra, agrupa y organiza la información para poder ser utilizada en la generación de mapas 2D mediante un proceso de Localización y Mapeo Simultáneo (SLAM).

Los mapas obtenidos contienen la geometría del entorno, es decir, las paredes, los obstáculos y los espacios libres del laboratorio de telecomunicaciones y son la base para el estudio de la propagación de ondas en el interior de edificios, ya que aportan información espacial sobre el entorno en el que se produce el fenómeno de propagación de ondas electromagnéticas.

El sistema se desarrolla a partir de software libre como ROS2, Python, RViz2, Visual Studio Code, Node-RED y Eclipse Mosquitto, siendo estos utilizados para la implementación de los módulos del sistema, en la comunicación de los robots, en la visualización de la información captada y en la supervisión del correcto funcionamiento de la arquitectura.

Finalmente, se llevará a cabo la validación del sistema en la sala de telecomunicaciones, comprobando que la comunicación entre las plataformas robóticas se realiza correctamente, que la información se envía a través de MQTT, que se generan mapas 2D y que el sistema proporciona información relevante para el estudio de la propagación de señales en interiores. En este contexto, los sistemas de robótica móvil y telecomunicaciones se convierten en herramientas para apoyar el estudio de canales de propagación en entornos controlados y proporcionar información relevante.

1.4. Objetivos del proyecto

1.4.1. Objetivo general:

Desarrollar un modelo bidimensional para la caracterización de la propagación de señales en espacios interiores mediante mapas obtenidos con sensores LiDAR 2D de bajo costo, utilizando un sistema colaborativo de robots móviles para la adquisición y procesamiento de la información espacial.

1.4.1.1. Objetivos específicos:

Analizar el estado del arte sobre caracterización y modelado de la propagación de señales en interiores, mediante la revisión de literatura científica, artículos académicos y trabajos relacionados con técnicas de modelado y el uso de tecnología LiDAR, identificando metodologías y enfoques aplicados en investigaciones similares.

Implementar un sistema colaborativo de adquisición de datos mediante robots móviles equipados con sensores LiDAR 2D, con el fin de realizar el escaneo del área de estudio y la recopilación de información de distancia, permitiendo la generación de representaciones bidimensionales del entorno analizado.

Filtrado, segmentación y reconstrucción del entorno a partir de los datos obtenidos en el escaneo para la generación de mapas digitales del entorno que contengan información sobre los obstáculos y elementos estructurales que pueden influir en la propagación de la señal.

Al analizar las características del canal de propagación en interiores a partir de la geometría del entorno y de los elementos aparecidos en el mapa, se pueden determinar los factores que influyen en el comportamiento de las señales.

Diseñar y validar un modelo bidimensional de propagación en interiores, empleando la información espacial derivada de los mapas LiDAR y herramientas de simulación, evaluando su capacidad para estimar el comportamiento de la señal dentro del entorno estudiado.

1.5. Justificación

La propagación de la onda electromagnética en interiores es uno de los temas más importantes de las telecomunicaciones, debido al auge de las redes de comunicación inalámbrica, el IoT y los sistemas de automatización que requieren un correcto diseño de la red de cobertura y calidad del servicio. La propagación en interiores depende de la geometría del entorno, los materiales de construcción y los obstáculos que afectan a la reflexión, difracción, dispersión y atenuación de la onda electromagnética.

Históricamente la predicción de la propagación se ha realizado por modelos teóricos o por campañas de medida (que en muchas ocasiones no dejan de ser simplificaciones del entorno físico) , por lo que puede haber una diferencia entre el resultado estimado y el comportamiento

real de la señal, sobre todo en el interior, donde puede haber una gran diversidad en la distribución de los elementos estructurales.

Una posible forma de solucionar el problema son los sensores LiDAR 2D, que proporcionan información espacial con una alta resolución, y permiten la creación de mapas 2D que representan de forma más precisa la geometría del entorno y, por tanto, la creación de modelos más precisos para el análisis de propagación y la identificación del efecto que tienen los diferentes obstáculos sobre la propagación de señales inalámbricas.

Sin embargo, la inclusión de un sistema de colaboración entre dos robots móviles permite implementar la división entre la recolección y el procesamiento de información, lo cual permite obtener un mejor desempeño del sistema. Por otro lado, se puede observar la inclusión del protocolo MQTT, que es un mecanismo de comunicación ligero, eficiente y confiable a la hora de intercambiar información entre ambas plataformas robóticas. Su modelo de comunicación de tipo Publisher/Subscriber, que permite la sincronización de procesos, un acoplamiento más flexible y una mejor escalabilidad del sistema, ha sido muy utilizado en aplicaciones de Internet de las Cosas (IoT) y sistemas distribuidos.

El presente proyecto es de carácter multidisciplinario para el desarrollo de sistemas que integran los conocimientos de diversas áreas de la Ingeniería en Telecomunicaciones como: las comunicaciones inalámbricas y sus protocolos, la robótica móvil, el procesamiento digital de datos y el mapeo utilizando sensores LiDAR para crear una plataforma experimental que sirva como soporte para desarrollar investigaciones sobre la caracterización de canales de propagación, sistemas colaborativos y aplicaciones de la robótica móvil.

Asimismo, la metodología desarrollada podrá ser adaptada a otros escenarios de investigación donde se requiera el intercambio de información entre múltiples dispositivos inteligentes, contribuyendo al fortalecimiento de nuevas aplicaciones basadas en arquitecturas distribuidas para el análisis de entornos interiores.

Alcance del Proyecto

El presente proyecto contempla el diseño, implementación y validación de un sistema colaborativo conformado por dos robots móviles ROSMASTER X3 destinados a la adquisición, transmisión y procesamiento de información espacial dentro de un entorno interior controlado correspondiente al laboratorio de telecomunicaciones.

En cuanto al hardware, para el sistema se dispone de sensores LiDAR 2D, plataformas robóticas y herramientas de desarrollo basadas en ROS2. En cuanto a la comunicación entre plataformas, se utilizará el protocolo MQTT. Ambos robots se comunicarán utilizando una conexión inalámbrica local y el broker MQTT se encargará de la comunicación entre plataformas mediante el modelo Publisher/Subscriber, el cual permitirá que la información generada durante la exploración se envíe de forma ordenada.

Las etapas a realizar en este proyecto son la preparación del hardware, la instalación del entorno de desarrollo, la implementación de la arquitectura de comunicación, la adquisición de datos a través de los sensores LiDAR, el procesamiento de los datos adquiridos mediante algoritmos de filtrado y localización, la creación de mapas 2D, y el análisis de los datos adquiridos y su uso para la caracterización de la propagación de señales en interiores.

El sistema se probará mediante la verificación de las correctas comunicaciones entre los dos robots, el envío correcto de la información a través de MQTT, la correcta generación de los

mapas 2D y el correcto funcionamiento de la arquitectura colaborativa en las pruebas del laboratorio de telecomunicaciones.

Dado que este artículo se centra en la clasificación de escenarios interiores utilizando sensores LiDAR 2D, no se abordará el tema de la reconstrucción 3D del escenario a partir de datos LiDAR ni el de los sensores LiDAR 3D. Cabe señalar también que se utilizará un escenario controlado en nuestros experimentos y que nuestros resultados solo son válidos para un escenario controlado, pero pueden ser utilizados como base para otros escenarios más complejos.

1.6. Metodología

La metodología del presente proyecto se basa en un enfoque estructurado por fases, orientado al desarrollo de un sistema colaborativo multi-robot para la adquisición, procesamiento y análisis de información espacial en entornos interiores.

Este enfoque permite organizar el desarrollo del sistema de manera progresiva, asegurando la correcta integración de los componentes físicos y lógicos, así como la validación de cada una de las etapas del proceso, como se muestra en la figura [1].

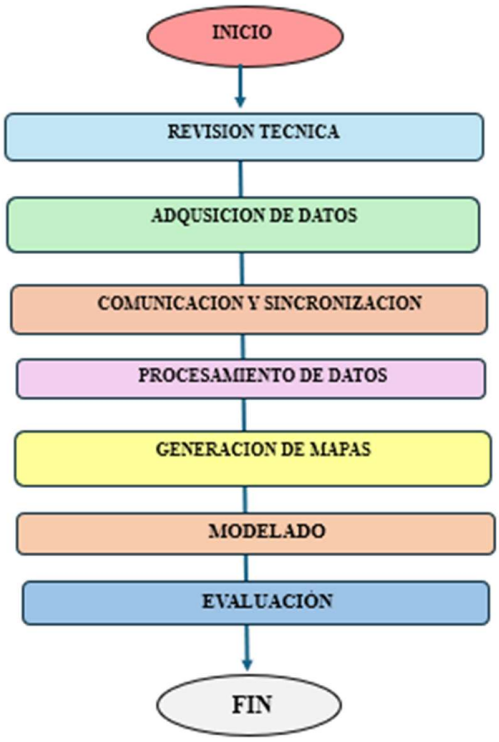


Ilustración 1:Diagrama de fases de la metodología del proyecto.

1.6.1. Fase de revisión técnica

En esta fase se realizará una revisión de información técnica relacionada con el funcionamiento de sensores LiDAR 2D, así como de conceptos asociados a la robótica móvil, algoritmos de mapeo (SLAM) y propagación de señales en entornos interiores.

En esta actividad se definirán los principios de funcionamiento del sensor, su formato de salida, cómo se obtendrá la información y el software que se utilizará para su procesamiento. También se estudiarán arquitecturas de sistemas multi-robot colaborativos donde se analizará la asignación de funciones entre las plataformas de los diferentes sistemas.

Además, se describirán las especificaciones técnicas que se van a requerir para la creación del sistema, que incluirá hardware como robots, sensores y módulos de comunicación,

software como ROS2, lenguajes de programación y librerías, así como el entorno donde va a trabajar.

1.6.2. Fase de adquisición de datos

En esta etapa se llevará a cabo la implementación del sistema de adquisición de datos mediante el uso del primer robot móvil, el cual cumple la función de explorador del entorno. Este robot estará equipado con un sensor LiDAR 2D, encargado de realizar el escaneo del laboratorio de telecomunicaciones.

Se realizará la configuración del sensor y su integración con la plataforma robótica, permitiendo la obtención de datos de distancia en múltiples direcciones a partir del movimiento del robot dentro del entorno.

Durante esta fase se ejecutarán múltiples pruebas de captura, con el objetivo de garantizar la calidad, consistencia y cobertura de los datos obtenidos. Asimismo, se establecerán parámetros de adquisición como la resolución angular, frecuencia de muestreo, velocidad de desplazamiento del robot y área de escaneo.

1.6.3. Fase de comunicación y sincronización de datos

En esta fase se definirán los protocolos de comunicación, formatos de datos y mecanismos de sincronización necesarios para garantizar la integridad de la información transmitida. Además, se implementarán estrategias para evitar pérdidas de datos o retrasos en la transmisión.

La sincronización entre ambos robots es un aspecto crítico del sistema, ya que permite asegurar que la información procesada corresponda correctamente al entorno explorado.

1.6.4. Fase de procesamiento de datos

En esta fase, el segundo robot será el encargado de recibir y procesar la información adquirida por el robot explorador.

Los datos adquiridos mediante el sensor LiDAR son sometidos a un proceso de eliminación de ruidos, puntos espurios (métricamente incorrectos) y de incoherencias generadas en el proceso de adquisición y a un proceso de clasificación y ordenación (o estructuración) de datos, es decir, son transformados de datos en bruto a datos procesados.

Además, se generará una segmentación para identificar las paredes, los obstáculos y el espacio libre en el entorno para que el entorno se represente de forma precisa y se disponga de información de entorno fiable para las siguientes fases del proceso.

1.6.5. Fase de Generación de mapas

Una vez procesados los datos, se procederá a la generación de mapas bidimensionales del entorno mediante la implementación de algoritmos de localización y mapeo simultáneo (SLAM).

Estos algoritmos permiten construir representaciones del entorno a partir de la información adquirida por el sensor LiDAR, considerando la posición del robot y la detección de obstáculos. Como resultado, se obtendrán mapas que reflejan la geometría del espacio, incluyendo la ubicación de paredes, objetos y zonas libres.

1.6.6. Fase de desarrollo del modelo

En esta fase se diseñará e implementará un modelo bidimensional de propagación de señales basado en la información geométrica obtenida a partir de los mapas generados.

El modelo considerará la influencia de los elementos estructurales identificados, tales como paredes y obstáculos, permitiendo analizar de manera aproximada el comportamiento de las señales dentro del entorno.

Para su desarrollo, se utilizarán herramientas de programación y/o simulación que permitan integrar los datos espaciales y representar el comportamiento del canal de propagación. El objetivo de esta fase es establecer una relación entre la geometría del entorno y la propagación de señales inalámbricas, proporcionando una herramienta de análisis dentro del laboratorio.

1.6.7. Fase de evaluación del sistema

Finalmente, se realizará la evaluación del sistema implementado, verificando el correcto funcionamiento de cada una de sus etapas, así como la integración entre los componentes físicos y lógicos.

Se analizará la coherencia entre los datos adquiridos, los mapas generados y el modelo desarrollado, evaluando la precisión del sistema y su capacidad para representar el entorno de manera adecuada.

El desempeño del sistema colaborativo se medirá a partir de la comunicación entre los robots, la eficiencia en el procesamiento de datos y la calidad de los resultados.

Ese paso servirá para determinar si el sistema se puede utilizar como ayuda para la caracterización de la propagación en interiores y proporcionar información sobre posibles mejoras que se pueden realizar en el sistema para mejorar su rendimiento en una futura implementación.

CAPÍTULO II

MARCO TEÓRICO DE LA INVESTIGACIÓN

2.1. Fundamentación teórica

2.1.1. Propagación de ondas electromagnéticas en interiores

La propagación de ondas electromagnéticas en entornos interiores constituye un fenómeno complejo debido a la interacción constante de las ondas con múltiples superficies, obstáculos y materiales presentes en el medio. A diferencia de la propagación en espacio libre, donde la señal se transmite principalmente en línea recta, en ambientes confinados las ondas experimentan reflexiones, difracciones y dispersiones, generando múltiples trayectorias de propagación conocidas como multitrayectoria (*multipath*) [4], [5].

Como consecuencia de este fenómeno, la señal recibida es la superposición de varias componentes con diferentes amplitudes, fases y retardos temporales. Esto puede provocar interferencias constructivas y destructivas, dando lugar a fluctuaciones en la intensidad de la señal, fenómeno conocido como desvanecimiento (*fading*), el cual afecta directamente la calidad y confiabilidad de los sistemas inalámbricos [6].

2.1.1.1. Concepto de onda electromagnética

Una onda electromagnética o EM (comúnmente, también, luz) es una onda transversal en la que las oscilaciones del campo eléctrico y magnético son perpendiculares entre sí y a la dirección de propagación de la onda. Las ondas electromagnéticas propagan energía a través del vacío sin la necesidad de un medio material y son la base de los sistemas de comunicación y sensado inalámbricos.

Desde el punto de vista físico, las ondas electromagnéticas se caracterizan por tres magnitudes: la frecuencia, la longitud de onda y la velocidad de propagación, que se relacionan de la siguiente forma:

$$c = \lambda f$$

Donde c es la velocidad de la onda, λ es la longitud de onda y f es la frecuencia. Esta relación es fundamental para comprender el comportamiento de las ondas en diferentes medios, ya que influye en su capacidad de penetración y en su interacción con los materiales [7].

2.1.1.2. Propagación en medios confinados

En medios confinados, como edificios u oficinas, la propagación de las ondas electromagnéticas se ve afectada por la geometría del entorno y la presencia de múltiples obstáculos. Esto genera trayectorias indirectas entre el transmisor y el receptor, lo que da lugar al fenómeno de multitrayectoria.

Además, la potencia de la señal disminuye con la distancia debido a pérdidas de propagación. Este comportamiento puede aproximarse mediante el modelo de pérdida en espacio libre:

$$FSPL\ (dB) = 20\log_{10}(d) + 20\log_{10}(f) + 32.44$$

Este modelo sirve como referencia base, aunque en interiores se presentan pérdidas adicionales debido a la interacción con materiales y obstáculos.

2.1.1.3. Factores que afectan la propagación en interiores

La propagación de las ondas electromagnéticas en interiores depende de muchos factores que condicionan la calidad de la señal en el punto de recepción, tales como los materiales que

forman el edificio, la frecuencia de la señal, la geometría interior, la separación entre el emisor y el receptor y la existencia de objetos o personas en movimiento [4].

Materiales de construcción:

Son los componentes del entorno (concreto, vidrio, madera, metal, entre otros) que pueden reflejar, absorber o transmitir la señal, dependiendo de su naturaleza. A mayor densidad o conductividad, mayor será la pérdida [4].

Frecuencia de operación:

Es el número de oscilaciones de la onda por segundo. A frecuencias más altas, la señal presenta mayor atenuación y menor capacidad de penetración en obstáculos, lo que afecta la cobertura en interiores.

Geometría del entorno:

Corresponde a la distribución espacial de paredes, techos, pisos y objetos dentro del ambiente. Esta configuración influye en la formación de trayectorias múltiples y en la dirección de propagación de la señal [4].

Distancia entre transmisor y receptor:

La distancia es la separación entre el transmisor y el receptor; la potencia de la señal disminuye con el incremento de distancia *(path loss)* [8].

Presencia de objetos o personas en movimiento:

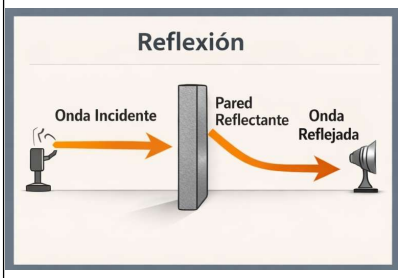
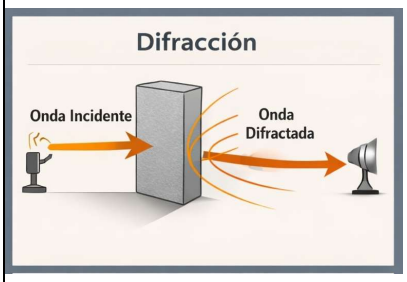
Elementos dinámicos en el medio que cambian la condición del canal de forma continua, provocando que la señal sufra desvanecimiento y dispersión temporal [6].

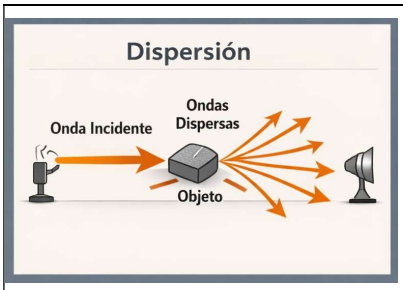
Estos factores afectan fenómenos como la atenuación y el desvanecimiento, los cuales deben ser considerados en el diseño de sistemas inalámbricos y de senado en entornos cerrados.

2.1.1.4. Fenómenos de propagación

Durante la propagación en interiores, las ondas electromagnéticas experimentan diversos fenómenos físicos que afectan su comportamiento, modificando la intensidad, dirección y calidad de la señal [7], [4]. Como se muestra en la *Tabla 1*, los fenómenos de propagación influyen directamente en el comportamiento de la señal en entornos interiores.

Table 1Fenómenos de propagación de ondas electromagnéticas en interiores

Fenómeno	Descripción técnica	Impacto en la propagación
 *Ilustración 2: Fenómeno de reflexión.*	2 Ocurre cuando una onda incide sobre una superficie y parte de su energía es reflejada.	Genera múltiples trayectorias e interferencias.
 *Ilustración 3: Fenómeno de difracción.*	Permite que la onda rodee obstáculos o atraviese aberturas.	Mejora la cobertura en zonas sin línea de vista.

 *Ilustración 4:Fenómeno de dispersión.*	Se produce cuando la onda interactúa con objetos pequeños.	Incrementa el efecto multipath.
 *Ilustración 5: Fenómeno de atenuación.*	Pérdida de potencia de la señal durante la propagación.	Reduce el alcance y la calidad de la señal.

2.1.2. Canales de propagación en sistemas inalámbricos

2.1.2.1. Definición de canal de propagación

El canal de propagación de un sistema de comunicación inalámbrica se puede definir como el medio por el cual se propaga la señal electromagnética desde el transmisor hasta el receptor. El canal no es un medio ideal y su propagación viene afectado por fenómenos de atenuación, retardo y distorsión, además de que se le añade ruido [9].

Debido a la presencia de obstáculos, el terreno y los materiales que constituyen el medio de propagación de las ondas, y su relación con el movimiento, el canal se convierte en un medio aleatorio que hay que estudiar y modelar para el diseño de sistemas de comunicación inalámbrica [10].

2.1.2.2. Clasificación de canales

Los canales de propagación pueden clasificarse según el entorno donde se desarrollan, siendo los más comunes los canales interiores (*indoor*) y exteriores (*outdoor*), demostrados en la *Tabla 2*.

Canales interiores

Se caracterizan por la presencia de múltiples obstáculos, como paredes, muebles y personas, lo que genera un alto nivel de multitrayectoria y variabilidad en la señal. Estos canales presentan mayor atenuación y complejidad debido a la interacción con materiales de construcción [9].

Canales exteriores

Se desarrollan en espacios abiertos, donde la propagación puede incluir trayectorias directas y reflejadas en edificaciones o el terreno. Aunque también presentan multitrayectoria, su comportamiento es más predecible en comparación con los entornos interiores [10]..

Table 2 Comparación entre canales interiores y exteriores

Característica	Canal interior	Canal exterior
Obstáculos	Alta densidad (paredes, muebles)	Baja o media (edificios, terreno)
Multitrayectoria	Alta	Media
Atenuación	Elevada	Moderada
Variabilidad	Alta	Media

2.1.2.3. Modelos de propagación en interiores

Los modelos de propagación permiten describir matemáticamente el comportamiento del canal inalámbrico. En entornos interiores, estos modelos se clasifican principalmente en empíricos y determinísticos.

Modelo log-distance

El modelo log-distance es uno de los más utilizados para describir la pérdida de señal en función de la distancia. Este modelo establece que la potencia recibida disminuye de forma logarítmica con respecto a la distancia entre el transmisor y el receptor [11].

$$PL(d) = PL(d_0) + 10_n \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma$$

- $PL(d)$: Pérdida de trayectoria a distancia d.
- n: Exponente de pérdida.
- X_σ: Variable aleatoria.

Modelos empíricos y determinísticos

Estos modelos son más precisos en su representación del comportamiento del canal, pero también más costosos computacionalmente.

Modelos empíricos

Son las que, utilizando medios estadísticos, caracterizan el canal. Están muy extendidas, aunque su precisión depende del medio en el que se utilizan.

Modelos determinísticos

Se fundamentan en leyes físicas de propagación, como la óptica geométrica y la teoría electromagnética. Ejemplos incluyen el *ray tracing*, el cual permite simular trayectorias de la señal considerando reflexiones y difracciones.

2.1.2.4. Parámetros del canal

El canal de propagación se caracteriza mediante diversos parámetros que describen el comportamiento de la señal durante su transmisión.

Pérdida de trayectoria (Path Loss)

La pérdida de trayectoria representa la disminución de la potencia de la señal a medida que se propaga en el espacio. Este parámetro es fundamental para el diseño de enlaces inalámbricos, ya que determina la cobertura del sistema [10].

Retardo de propagación

El retardo de propagación corresponde al tiempo que tarda la señal en viajar desde el transmisor hasta el receptor. En entornos con multitrayectoria, diferentes componentes de la señal llegan en distintos instantes de tiempo, generando dispersión temporal.

Multitrayectoria (Multipath)

El fenómeno de multitrayectoria ocurre cuando la señal se propaga a través de múltiples caminos debido a reflexiones, difracciones y dispersiones. Esto provoca que múltiples versiones de la señal lleguen al receptor con diferentes fases y retardos, generando interferencias [12].

Este fenómeno es uno de los principales responsables del desvanecimiento en sistemas inalámbricos y debe ser considerado en el diseño de técnicas de modulación y codificación [12].

2.1.3. Modelado de la propagación en interiores

2.1.3.1. Importancia del modelado del canal

El modelado del canal de propagación se usa para describir el comportamiento de la señal en un entorno específico y es muy utilizado en el análisis y diseño de sistemas inalámbricos. En los entornos interiores, el modelado del canal es más complicado debido a la gran cantidad de obstáculos, la diversidad de materiales de construcción y la complejidad de la geometría del entorno [13].

A partir de estos modelos se pueden obtener valores de interés como la pérdida de trayectoria, la potencia recibida, el retardo de propagación y la multitrayectoria, que pueden ser utilizados en la planificación de redes, optimización de la cobertura y evaluación del desempeño del sistema antes de su instalación [14].

Además, el modelado del canal permite reducir costos y tiempo en la fase de diseño, al evitar pruebas experimentales extensivas, proporcionando una aproximación confiable del comportamiento del sistema en diferentes escenarios [15].

2.1.3.2. Modelos geométricos de propagación

Por el contrario, los modelos geométricos describen el trazado del canal a partir de la trayectoria física que siguen las ondas electromagnéticas desde el emisor hasta el receptor. Este tipo de modelado, que se basa en principios de óptica geométrica o de teoría electromagnética, permite describir de forma explícita fenómenos como la reflexión, la difracción o la dispersión del campo electromagnético [13].

Un modelo popular dentro del enfoque de la geometría de ondas es el modelo de ray tracing. Este modelo simula el recorrido de un rayo de señal a través de múltiples trayectorias que rebotan en el medio. Estas trayectorias pueden experimentar pérdidas en función de los

rebotes en las superficies, la penetración en los materiales o la difracción en los bordes. La potencia recibida puede aproximarse como la suma de las contribuciones de cada trayectoria:

$$Pr = \sum_{i=1}^{N} \mathrm{P_i}$$

donde Pr es la potencia total recibida y Pi es la potencia recibida a través de la enésima trayectoria.

Con estos modelos se pueden estudiar el canal en entornos más complejos y acotados (como el interior de un edificio o un laboratorio) en el que la geometría puede influir en la propagación. Su principal inconveniente es la necesidad de tener información del entorno (planos, materiales) y su mayor coste computacional [14].

En aplicaciones más avanzadas, los modelos geométricos se utilizan para el análisis de la cobertura, la interferencia o la propagación en los entornos de interés, y se convierten en herramientas esenciales en el diseño de sistemas de comunicación inalámbrica.

2.1.3.3. Modelado bidimensional del entorno

La representación 2D del ambiente se basa en un plano bidimensional, que incluye todos los elementos del entorno que son relevantes para la arquitectura (paredes, puertas, obstáculos y la ubicación de los dispositivos) . Es una representación que se suele utilizar en muchos problemas de robótica móvil y simulación de propagación por su sencillez y bajo coste computacional.

En este enfoque, el entorno suele discretizarse en celdas o segmentos, lo que permite analizar la interacción de la señal con los diferentes elementos presentes. Esta discretización facilita la integración con técnicas de mapeo, como los mapas de ocupación, y con sensores como el LiDAR, que generan información en formato bidimensional.

2.1.3.4. Limitaciones de los modelos tradicionales

Sin embargo, los modelos empíricos y los modelos geométricos tienen sus limitaciones en aplicaciones en interiores reales. Primero, muchos de estos modelos hacen suposiciones sobre la simplificación del entorno, lo que significa que puede haber una diferencia entre el resultado teórico y el comportamiento real de la señal [13].

La desventaja de los modelos empíricos es que son válidos solo para el medio donde fueron desarrollados. Para los modelos geométricos, se requiere información detallada del medio, como las propiedades electromagnéticas de los materiales y la geometría exacta, algo que no siempre es factible obtener.

Además, la mayoría de estos modelos no tienen en cuenta otros factores que pueden cambiar las características del canal, como la movilidad de las personas, la apertura de puertas o la reconfiguración de objetos, por lo que son menos representativos de la realidad en tiempo real.

Otra desventaja importante es que no pueden modelar fenómenos más complejos como la dispersión en superficies irregulares o la absorción en materiales heterogéneos. Esta es la razón por la que se combinan estos modelos con técnicas basadas en la adquisición de datos con sensores como el LiDAR para obtener una representación más precisa y actual del entorno en trabajos actuales [15].

2.1.4. Sensores LiDAR

2.1.4.1. Definición y principio de funcionamiento

El sensor LiDAR (*Light Detection and Ranging*) es una tecnología de medición que permite determinar distancias mediante la emisión de pulsos de luz láser y el análisis del tiempo

que tarda la señal en reflejarse en un objeto y regresar al sensor. Este principio, conocido como tiempo de vuelo (*Time of Flight*), es ampliamente utilizado en aplicaciones de percepción del entorno y mapeo en robótica móvil [16].

El funcionamiento del LiDAR se basa en la emisión de pulsos de luz en diferentes direcciones, los cuales al impactar sobre superficies del entorno son reflejados hacia el sensor. A partir del tiempo de retorno, se calcula la distancia al objeto, permitiendo construir una representación espacial del entorno.

La distancia se obtiene mediante la siguiente expresión:

$$d = \frac{c.t}{2}$$

donde d es la distancia al objeto, c es la velocidad de la luz y t es el tiempo transcurrido entre la emisión y la recepción del pulso.

Este método permite obtener mediciones rápidas y precisas, lo que hace que el LiDAR sea especialmente útil en aplicaciones donde se requiere alta exactitud en la detección de obstáculos y en la generación de mapas del entorno [17].

2.1.4.2. Tipos de sensores LiDAR

Los sensores LiDAR pueden clasificarse principalmente en función de la dimensión del escaneo que realizan, distinguiéndose entre LiDAR 2D y LiDAR 3D.

LiDAR 2D

El LiDAR 2D realiza mediciones en un solo plano, generalmente horizontal, generando un conjunto de datos en forma de distancias y ángulos. Este tipo de sensor es ampliamente

utilizado en robótica móvil para tareas de navegación y mapeo en interiores, debido a su bajo costo, menor complejidad y suficiente precisión para representar el entorno en dos dimensiones [3].

LiDAR 3D

El LiDAR 3D consiste en recoger información en 3D en varios planos de escaneo o en un sistema de escaneo rotativo que proporciona nubes de puntos en 3D. Este sensor proporciona más información sobre el entorno y es adecuado para vehículos autónomos o para modelado 3D, aunque está restringido por su mayor coste y su mayor carga computacional.

Table 3 Comparación entre LiDAR 2D y LiDAR 3D.

Característica	LiDAR 2D	LiDAR 3D
Dimensión de escaneo	Plano horizontal	Volumen (3D)
Complejidad	Baja	Alta
Costo	Bajo	Alto
Aplicación	Robótica móvil	Vehículos autónomos

2.1.4.3. Características técnicas del LiDAR 2D

El desempeño de un sensor LiDAR 2D se define a partir de diversas características técnicas que determinan la calidad de las mediciones obtenidas y la precisión del mapeo del entorno.

Alcance

Es la máxima distancia donde el sensor puede detectar un objeto, y depende de la potencia del láser, la reflectividad de los objetos y las condiciones del entorno.

Resolución

Se refiere a la capacidad del sensor de distinguir dos objetos que se encuentran a poca distancia. En un LiDAR 2D, es la resolución angular o el intervalo entre dos medidas consecutivas. Una mayor resolución se traduce en mapas más detallados.

Precisión

La precisión es la medida en que una distancia medida se aproxima al valor real. Un sensor es preciso si su error en la distancia medida es pequeño. La precisión es importante en aplicaciones de navegación y mapeo [17].

Estas características influyen directamente en la calidad de la información adquirida y en la eficiencia de los algoritmos de mapeo y navegación que utilizan los datos del sensor.

2.1.4.4. Ventajas del uso de LiDAR en interiores

El uso de sensores LiDAR en interiores presenta ciertas ventajas frente a otras tecnologías de percepción del entorno, como cámaras o sensores ultrasónicos, siendo la principal de todas ellas la falta de dependencia del sistema con la iluminación en el entorno, el sistema LiDAR puede operar en condiciones de luz muy baja o inexistente [18].

El sistema LiDAR se caracteriza por un alto rendimiento en la medición de distancias, por lo que es un dispositivo de elección en la detección de obstáculos y en la creación de mapas del entorno, para aplicaciones de robótica móvil que deben crear un mapa del entorno para poder navegar de forma autónoma.

Otra ventaja es que se puede tener información sobre el entorno en tiempo real para que el sistema pueda reaccionar a los cambios en el entorno (por ejemplo, la aparición de objetos o

personas en movimiento) . Gracias a una posible integración con sistemas como ROS se pueden procesar los datos y algoritmos como SLAM pueden ser utilizados para el mapeo [18].

En general, todas estas propiedades hacen del LiDAR una tecnología muy adecuada para la cartografía y el análisis del entorno interior, como el de los laboratorios de telecomunicaciones donde se requiere alta fiabilidad y precisión en la captura de datos espaciales.

2.1.5. Generación de mapas mediante LiDAR

2.1.5.1. Adquisición de datos espaciales

La generación de mapas mediante LiDAR inicia con la adquisición de datos espaciales del entorno. El sensor emite pulsos láser en diferentes direcciones y registra, para cada medición, la distancia y el ángulo respecto a su referencia. Estos datos permiten construir una representación del entorno en forma de puntos en coordenadas polares (r, θ) [16].

Para su uso en mapeo, dichas mediciones se transforman a coordenadas cartesianas mediante:

$$x = r\,cos\theta \quad , \quad y = r\,sin\theta$$

Esta conversión permite ubicar cada punto en un plano 2D, generando una nube de puntos que describe la geometría del entorno. La calidad de los datos adquiridos depende de factores como la resolución angular del sensor, la frecuencia de muestreo y las características de las superficies (reflectividad y geometría) [17].

2.1.5.2. Representación bidimensional del entorno

Una vez obtenidos los datos en coordenadas cartesianas, el entorno puede representarse en un plano bidimensional. Esta representación es ampliamente utilizada en robótica móvil

debido a su simplicidad y eficiencia computacional, permitiendo modelar paredes, obstáculos y espacios libres de manera estructurada [16].

El entorno puede representarse como un conjunto de puntos (nube de puntos) o mediante estructuras discretas, como mapas en rejilla. La representación 2D es suficiente para la mayoría de las aplicaciones en interiores, donde la navegación se realiza en un plano horizontal. Además, esta aproximación facilita la integración con algoritmos de localización y planificación de trayectorias.

2.1.5.3. Mapas de ocupación (Occupancy Grid Maps)

Los mapas de ocupación son una de las representaciones más utilizadas para describir el entorno en robótica. En este enfoque, el espacio se divide en una rejilla de celdas, donde cada celda almacena la probabilidad de estar ocupada, libre o desconocida [16].

La actualización de cada celda se realiza a partir de las mediciones del sensor, utilizando modelos probabilísticos que consideran la incertidumbre de las mediciones. De forma general, la probabilidad de ocupación se actualiza mediante:

$$P(m_i|z_{1:t}) = \frac{P(z_t|m_i)\, P(m_i|z_{1:t-1})}{P(z_t)}$$

donde m_i representa el estado de la celda y z_t la medición del sensor en el instante t.

Para facilitar el cálculo, suele emplearse la representación en log-odds, que simplifica la actualización y mejora la estabilidad numérica.

2.1.5.4. Procesamiento de datos LiDAR

El procesamiento de datos LiDAR es una etapa fundamental para mejorar la calidad de la información adquirida y generar mapas más precisos. Este proceso incluye varias fases:

Filtrado

Consiste en eliminar ruido y mediciones erróneas presentes en los datos. Esto puede incluir la eliminación de valores atípicos o la aplicación de filtros estadísticos para mejorar la calidad de la nube de puntos [17].

Segmentación

La segmentación tiene como propósito el poder distinguir y separar las partes del entorno, como paredes, objetos o estructuras, para así lograr una mejor interpretación y mejorar el rendimiento de los algoritmos que realizan el mapeo [19].

Reconstrucción del entorno

Con estos datos se genera un modelo del entorno, que puede ser un mapa de ocupación o una nube de puntos estructurada, fusionando así las diferentes mediciones para obtener una representación más completa y precisa del entorno [16].

2.1.6. Sistema ROS (Robot Operating System)

2.1.6.1. Definición de ROS

ROS (*Robot Operating System*) es un conjunto de herramientas, bibliotecas y convenciones que facilitan el desarrollo de aplicaciones para sistemas robóticos. A pesar de su nombre, ROS no es un sistema operativo en sentido estricto, sino un *middleware* que permite la comunicación entre diferentes componentes de software y hardware en un robot [20].

Este entorno proporciona funcionalidades como manejo de sensores, control de actuadores, procesamiento de datos y comunicación entre procesos, permitiendo desarrollar sistemas modulares y escalables. Gracias a su arquitectura distribuida, ROS es ampliamente utilizado en investigación y desarrollo de robótica móvil [16].

2.1.6.2. Arquitectura de ROS

La arquitectura de ROS se basa en un modelo distribuido donde los diferentes componentes del sistema se organizan en nodos que se comunican entre sí. Los elementos principales son:

Nodos

Son procesos independientes que ejecutan tareas específicas, como lectura de sensores, procesamiento de datos o control del robot. Este modularidad permite desarrollar sistemas flexibles y fáciles de mantener [20].

Tópicos (*topics*)

Son canales de comunicación utilizados para el intercambio de datos entre nodos mediante un esquema de publicación-suscripción. Por ejemplo, un nodo puede publicar datos del LiDAR y otro nodo puede suscribirse para procesarlos.

Servicios

Permiten la comunicación síncrona entre nodos, donde uno solicita una acción y otro responde. Se utilizan cuando se requiere una respuesta inmediata o una interacción puntual.

2.1.6.3. Uso de ROS en robots móviles

ROS es ampliamente utilizado en robots móviles debido a su capacidad para integrar múltiples sensores y algoritmos de manera eficiente. Permite gestionar la adquisición de datos, el procesamiento de información y la ejecución de acciones en tiempo real, facilitando la navegación autónoma [16].

Además, ROS proporciona herramientas para visualización, simulación y depuración, como RViz y Gazebo, que permiten analizar el comportamiento del robot en entornos virtuales antes de su implementación en escenarios reales.

2.1.6.4. Librerías para mapeo (SLAM)

ROS incluye diversas librerías y paquetes para la implementación de algoritmos de mapeo y localización simultánea (*SLAM*). Entre los más utilizados se encuentran:

- **Gmapping**

Basado en filtros de partículas, permite generar mapas de ocupación en entornos 2D.

- **Hector SLAM**

Utiliza datos de LiDAR para generar mapas sin necesidad de odometría precisa.

- **Cartographer**

Desarrollado por Google, permite mapeo en 2D y 3D con alta precisión [21].

Estas herramientas facilitan la generación de mapas del entorno en tiempo real, lo que es fundamental en aplicaciones de navegación autónoma.

2.1.6.5. Integración de LiDAR con ROS

La integración de sensores LiDAR con ROS se realiza mediante nodos específicos que permiten la adquisición y publicación de datos del sensor en el sistema. Estos datos son transmitidos a través de tópicos, generalmente en forma de mensajes tipo *LaserScan*, que contienen información de distancias y ángulos [21].

Los datos que se publican en ROS pueden ser utilizados por otros nodos para tareas como el mapeo, la detección de obstáculos o la navegación. Con estos datos en tiempo real, los nodos pueden construir representaciones del entorno, como mapas de ocupación, que pueden ser utilizados por robots móviles en entornos interiores.

ROS permite la interoperación de distintos tipos de sensores y plataformas, lo que facilita la creación de arquitecturas escalables y flexibles para una variedad de aplicaciones, como la percepción del entorno en laboratorios de telecomunicaciones.

2.1.7. Técnicas de mapeo y SLAM

2.1.7.1. Definición de SLAM

El término SLAM (*Simultaneous Localization and Mapping*) se refiere al proceso mediante el cual un robot construye un mapa de un entorno desconocido mientras estima simultáneamente su propia posición dentro de dicho entorno. Este problema es fundamental en robótica móvil, especialmente en escenarios interiores donde no se dispone de sistemas de posicionamiento global como GPS [16].

Desde un punto de vista probabilístico, el problema de SLAM puede ser formulado como una estimación conjunta del estado del robot y del mapa del entorno a partir de las observaciones de los sensores del robot. En general, se puede escribir:

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

donde xt es la posición del robot, m es el mapa, z(1:t) son las observaciones del sensor y u_(1:t) son las acciones de control.

Este enfoque tiene en cuenta la incertidumbre en las mediciones y en el movimiento del robot, lo que proporciona una estimación más robusta del entorno [22].

2.1.7.2. SLAM basado en LiDAR

El SLAM basado en LiDAR utiliza mediciones de distancia obtenidas mediante sensores láser para construir mapas del entorno. Este enfoque es ampliamente utilizado en interiores debido a la alta precisión de los sensores LiDAR y su independencia de condiciones de iluminación [17].

En este tipo de sistemas, el robot adquiere datos en forma de escaneos láser, los cuales son procesados para estimar su movimiento relativo (*odometría*) y actualizar el mapa del entorno. La información obtenida permite detectar estructuras como paredes, esquinas y obstáculos, facilitando la construcción de mapas detallados.

Además, el uso de LiDAR reduce errores asociados a sensores visuales, como cambios de iluminación o falta de textura en el entorno, lo que mejora la confiabilidad del sistema de mapeo [16].

2.1.7.3. Algoritmos de mapeo 2D

Los algoritmos de mapeo 2D permiten generar representaciones del entorno en un plano, utilizando datos provenientes de sensores como LiDAR. Entre los enfoques más utilizados se encuentran:

Filtros de partículas (Gmapping)

Utilizan múltiples hipótesis de la posición del robot para estimar el mapa mediante métodos probabilísticos.

SLAM basado en escaneo (Scan Matching)

Consiste en alinear escaneos consecutivos del sensor para estimar el movimiento del robot.

SLAM basado en grafos (Graph-based SLAM)

Representa el problema como un grafo donde los nodos son posiciones del robot y las aristas representan restricciones entre el

Table 4 Comparación de algoritmos de mapeo 2D.

Algoritmo	Método	Ventaja
Gmapping	Filtros de partículas	Manejo de incertidumbre
Hector SLAM	Scan matching	No requiere odometría
Cartographer	Basado en grafos	Alta precisión

2.1.7.4. Generación de mapas del entorno interior

La generación de mapas en interiores mediante técnicas SLAM implica la integración de datos sensoriales, estimación de la posición del robot y actualización continua del entorno. El resultado es una representación del espacio que puede ser utilizada para navegación, análisis del entorno y planificación de trayectorias [16].

En la práctica, los mapas generados son típicamente mapas de ocupación, que dividen el entorno en celdas, cada una de las cuales indica la presencia o ausencia de un obstáculo, y que permiten al robot distinguir entre zonas transitables y zonas prohibidas.

Dado que la generación de mapas de interiores implica un compromiso entre precisión y coste computacional, en muchos casos (como la generación de mapas en tiempo real de un entorno) se requieren técnicas para optimizar el procesamiento de datos y minimizar los errores de acumulación en la navegación.

2.1.8. Relación entre mapas LiDAR y propagación de señales

2.1.8.1. Representación geométrica del entorno

Los mapas generados mediante sensores LiDAR permiten obtener una representación geométrica precisa del entorno, describiendo la ubicación de paredes, obstáculos y espacios libres. Esta información es fundamental para el análisis de la propagación de señales electromagnéticas, ya que la geometría del entorno influye directamente en la trayectoria de la señal [13].

Para las aplicaciones de modelado, el entorno puede ser modelado como un conjunto de elementos geométricos como líneas, polígonos o celdas, que representen las superficies con las que va a interactuar la señal. De esta manera, se pueden calcular las trayectorias que puede seguir la señal entre el transmisor y receptor, teniendo en cuenta, entre otros efectos, la reflexión y la difracción.

Adicionalmente, el LiDAR proporciona datos con alta resolución espacial que permiten una mayor precisión en la caracterización del entorno en comparación con modelos simplificados o teóricos [14].

2.1.8.2. Influencia de obstáculos en la propagación

Los obstáculos presentes en el entorno, como paredes, muebles o equipos, afectan la propagación de las señales electromagnéticas mediante fenómenos como la reflexión, la difracción y la atenuación. Estos efectos dependen de las propiedades físicas del material y de la geometría del obstáculo [15].

Por ejemplo, las superficies metálicas reflejan la mayor parte de la energía de la señal, mientras que se producen pérdidas considerables cuando la señal se encuentra con el concreto.

Sin embargo, los bordes y las esquinas difractan la señal y permiten que esta se propague hasta zonas que no tienen línea de vista directa.

El mapeo detallado de la ubicación y características de estos obstáculos, que se puede realizar con mapas LiDAR, permite un modelado más preciso del comportamiento de la señal en interiores.

Table 5 Influencia de materiales en la propagación

Material	Efecto principal	Impacto
Concreto	Absorción	Alta atenuación
Metal	Reflexión	Multipath
Madera	Parcial absorción	Atenuación media
Vidrio	Transmisión parcial	Baja atenuación

2.1.8.3. Uso de mapas para modelado del canal

Los mapas generados con LiDAR pueden utilizarse como base para el modelado del canal de propagación, permitiendo integrar información real del entorno en simulaciones y análisis. A partir de estos mapas, es posible aplicar modelos geométricos que consideran las trayectorias de la señal y su interacción con los obstáculos [13].

En este contexto, la pérdida de trayectoria puede estimarse considerando no solo la distancia, sino también la cantidad de obstáculos atravesados. De manera general, puede expresarse como:

$$PL = PL_0 + 10n\log_{10}(d) + \sum_{i=1}^{k} \mathrm{L_i}$$

donde L_i representa la pérdida adicional introducida por cada obstáculo.

Este enfoque permite obtener estimaciones más realistas del comportamiento del canal en comparación con modelos que no consideran la geometría del entorno.

2.1.8.3. Aplicación en simulación de señales inalámbricas

Los mapas LiDAR se pueden utilizar para generar simulaciones más precisas de la propagación de señales en interiores, y se pueden utilizar como entrada para un programa de simulación para calcular la cobertura, la intensidad de la señal y las zonas de interferencia [14].

Cuando se aplica a problemas prácticos, como en laboratorios de telecomunicaciones, el método puede ser utilizado para descubrir zonas de mala cobertura, encontrar la mejor ubicación para los transmisores y para el diseño de redes inalámbricas.

El uso de datos reales a partir de LiDAR también reduce la necesidad de modelos teóricos simplificados, lo que se traduce en resultados más cercanos a la realidad del entorno, lo que resulta útil en escenarios donde se requiere una alta precisión en el análisis de la propagación de señales.

2.1.9. Comunicaciones en sistemas multi-robot

2.1.9.1. Concepto de comunicación en sistemas multi-robot

En los sistemas multi-robot, la comunicación representa el mecanismo mediante el cual múltiples plataformas robóticas intercambian información con el fin de coordinar sus acciones y cumplir objetivos comunes. Este intercambio de datos permite la cooperación entre robots, facilitando tareas como exploración, mapeo y procesamiento distribuido [23].

Además de la transmisión de datos, la comunicación también incluye la sincronización de procesos y la toma de decisiones basadas en datos de otros procesos. En el presente trabajo,

la comunicación es fundamental para que los datos obtenidos del sensor LiDAR del robot explorador sean transmitidos al robot procesador para la creación del mapa y el análisis del entorno en tiempo real [24].

2.1.9.2. Arquitecturas de comunicación en sistemas multi-robot

Las arquitecturas de comunicación definen la forma en que los robots intercambian información dentro del sistema. Estas arquitecturas pueden clasificarse en:

Arquitectura centralizada

Este modelo sigue un esquema de nodo central en el que el nodo controla todas las comunicaciones. En este caso los robots envían la información al nodo, que la procesa y envía las órdenes. Sin embargo, esta arquitectura no es escalable ni tolerante a fallos, ya que la responsabilidad del sistema se delega a un único punto [25].

Arquitectura distribuida

En este modelo, cada robot de la red tiene su propia unidad de procesamiento y puede comunicarse directamente con otros robots de la red, sin depender de un nodo central. Esto hace que la red sea más escalable y más resistente a fallos, ya que el sistema puede seguir funcionando si uno de los robots presenta un fallo [25].

Arquitectura híbrida

Combina características de las arquitecturas centralizadas y distribuidas, permitiendo una gestión parcial desde un nodo central, mientras se mantiene la autonomía de los robots.

En el caso del presente proyecto, se adopta una arquitectura distribuida, donde los robots cumplen funciones específicas (adquisición y procesamiento), comunicándose entre sí para intercambiar información de manera eficiente [25].

2.1.9.3. Modelos de comunicación en sistemas distribuidos

Los modelos de comunicación establecen la forma en que los datos son transmitidos dentro del sistema. Entre los más utilizados se encuentran:

Modelo cliente-servidor: Un nodo solicita información y otro responde.

Modelo peer-to-peer (P2P): Comunicación directa entre nodos sin intermediarios.

Modelo publicación-suscripción (publish/subscribe): Un nodo publica información y otros nodos se suscriben para recibirla.

Este último modelo es ampliamente utilizado en sistemas robóticos, especialmente en ROS, debido a su flexibilidad y eficiencia en la transmisión de datos en tiempo real.

2.1.10. Tecnologías de comunicación inalámbrica en interiores

2.1.10.1. Redes WiFi (IEEE 802.11)

Las redes WiFi en el estándar IEEE 802.11 son la tecnología más utilizada en redes LAN interiores, ya que ofrecen altas tasas de transferencia de datos. Estas redes funcionan en las bandas de 2.4 GHz y 5 GHz. Por lo tanto, permiten la transferencia de grandes volúmenes de datos en aplicaciones en tiempo real [26].

Aunque la señal WiFi en el interior está sujeta a reflexión, difracción y absorción que pueden causar desvanecimiento o atenuaciones que dependen del tiempo [27], WiFi sigue siendo la mejor opción para las aplicaciones de multi-robot debido a su compatibilidad con las plataformas robóticas y su facilidad de uso.

2.1.10.2. Tecnologías de comunicación para IoT

Además de WiFi, existen otras tecnologías utilizadas en aplicaciones de comunicación inalámbrica, especialmente en sistemas de bajo consumo energético. Entre ellas se destacan:

ZigBee (IEEE 802.15.4): Diseñada para redes de baja velocidad y bajo consumo.

Bluetooth Low Energy (BLE): Utilizada en aplicaciones de corto alcance con bajo consumo de energía.

Aunque estas tecnologías presentan ventajas en términos de eficiencia energética, su capacidad de transmisión es limitada en comparación con WiFi, lo que restringe su uso en aplicaciones que requieren transferencia de grandes volúmenes de datos.

2.1.10.3. Comparación de tecnologías inalámbricas

Las diferentes tecnologías de comunicación presentan características específicas que determinan su aplicación en sistemas multi-robot.

Table 6 Comparación de tecnologías inalámbricas.

Tecnología	Velocidad de transmisión	Alcance	Aplicación
WiFi	Alta	Media	Robots móviles
ZigBee	Baja	Alta	Sensores IoT
BLE	Media	Baja	Dispositivos cercanos

En el presente proyecto, la tecnología WiFi es la más adecuada debido a su capacidad para manejar grandes volúmenes de datos y su compatibilidad con plataformas robóticas.

2.1.10.4. Consideraciones de propagación en redes inalámbricas interiores

La calidad de la comunicación inalámbrica en interiores depende de diversos factores, como la geometría del entorno, los materiales de construcción y la presencia de obstáculos. Estos factores afectan parámetros como la potencia de la señal, la relación señal-ruido y la tasa de error de bits [28].

El uso de mapas generados mediante LiDAR permite caracterizar el entorno de manera precisa, facilitando el análisis de la propagación de señales y la optimización de la comunicación entre robots.

2.1.11. Parámetros de calidad de servicio (QoS)

La calidad de servicio (QoS) es un aspecto fundamental en sistemas de comunicación, ya que permite garantizar el desempeño adecuado de la red en aplicaciones críticas.

Latencia

La latencia es el tiempo total requerido para que un paquete de datos viaje desde el emisor hasta el receptor. En sistemas multi-robot, una latencia elevada puede generar inconsistencias en los datos y afectar el desempeño de algoritmos en tiempo real, como SLAM [28].

Ancho de banda

El ancho de banda determina la capacidad de la red para transmitir información. En aplicaciones que utilizan sensores LiDAR, el volumen de datos generado es considerable, por lo que se requiere un ancho de banda suficiente para evitar congestión en la red.

Pérdida de paquetes

La pérdida de paquetes ocurre cuando los datos transmitidos no llegan al destino. Este fenómeno puede ser causado por interferencias, congestión o errores en el canal, afectando la integridad de la información [29].

Jitter

El jitter representa la variación en el tiempo de llegada de los paquetes. Este parámetro es crítico en aplicaciones en tiempo real, ya que puede afectar la sincronización de los datos.

2.1.12. Protocolos de comunicación

TCP/IP

El protocolo TCP/IP es ampliamente utilizado en redes de comunicación debido a su confiabilidad. TCP garantiza la entrega de datos mediante mecanismos de control de errores y retransmisión, lo que lo hace adecuado para aplicaciones donde la integridad de la información es crítica [30].

UDP

El protocolo UDP permite la transmisión de datos con baja latencia, ya que no implementa mecanismos de control de errores. Esto lo hace adecuado para aplicaciones en tiempo real, aunque introduce el riesgo de pérdida de paquetes [30].

MQTT (Message Queuing Telemetry Transport)

MQTT es un protocolo mensajería orientado a conectar dispositivos remotos mediante el llamado de publicación/ suscripción comúnmente llamado emisor y receptor y a la vez implementa parámetros de QoS. Este protocolo es utilizado en ROS, facilitando la comunicación

eficiente entre nodos y permitiendo adaptar el comportamiento del sistema a los requerimientos de la aplicación. [31]

2.1.13. Sincronización de datos

La sincronización de datos es fundamental en sistemas distribuidos, ya que permite garantizar la coherencia temporal de la información.

El uso de técnicas como timestamping permite asociar una marca de tiempo a cada dato, facilitando su alineación durante el procesamiento. Asimismo, protocolos como NTP permiten sincronizar los relojes de los dispositivos en la red, reduciendo errores temporales [32].

En sistemas basados en ROS, la sincronización puede realizarse mediante herramientas internas que permiten coordinar datos provenientes de múltiples sensores.

2.1.14. Sistemas colaborativos multi-robot

Los sistemas multi-robot permiten distribuir tareas entre diferentes agentes, mejorando la eficiencia y reduciendo el tiempo de ejecución. Estos sistemas presentan ventajas significativas en aplicaciones como mapeo, exploración y monitoreo [33].

Desde el punto de vista de telecomunicaciones, estos sistemas pueden considerarse como redes dinámicas donde la topología cambia en función del movimiento de los robots, lo que introduce desafíos en la gestión de la comunicación.

2.1.15. Modelos por capas en redes de comunicación

2.1.15.1. Importancia del modelo por capas

Los protocolos de comunicación de red actuales están basados en arquitecturas de red en capas, que dividen el proceso de comunicación en varias capas, cada una de las cuales tiene

funciones específicas, que pueden ser implementadas de forma independiente, lo que facilita el diseño, la implementación y la interoperabilidad de sistemas heterogéneos [34].

La arquitectura por capas se puede aplicar a la comunicación de sensores, controlar la comunicación y la interacción en un sistema multi-robot. Se puede usar la arquitectura por capas con tecnologías de comunicación inalámbrica y plataformas de procesamiento como el sistema operativo de robot (ROS), además de construir un sistema distribuido.

2.1.15.2. Modelo OSI (Open Systems Interconnection)

El modelo OSI es un estándar conceptual que divide la comunicación en siete capas, cada una encargada de funciones específicas dentro del proceso de transmisión de datos [34].

Table 7 Capas del modelo OSI y sus funciones.

Capa	Nombre	Función principal
7	Aplicación	Interacción con el usuario y aplicaciones
6	Presentación	Formato de datos, codificación
5	Sesión	Control de sesiones de comunicación
4	Transporte	Control de flujo y errores
3	Red	Direccionamiento y enrutamiento
2	Enlace de datos	Control de acceso al medio
1	Física	Transmisión de bits

Capa física

Se encarga de la transmisión de señales eléctricas o electromagnéticas a través del medio. En el presente proyecto, esta capa corresponde a la transmisión inalámbrica mediante WiFi, donde las ondas electromagnéticas transportan la información entre robots [35].

Capa de enlace de datos

Gestiona el acceso al medio y la detección de errores. En redes WiFi, esta capa está representada por el estándar IEEE 802.11, el cual controla el acceso al canal inalámbrico.

Capa de red

Se encarga del direccionamiento lógico y el enrutamiento de paquetes. En sistemas multi-robot, esta capa permite identificar cada robot dentro de la red mediante direcciones IP.

Capa de transporte

Garantiza la entrega de datos mediante protocolos como TCP y UDP. Esta capa es clave para asegurar la confiabilidad o rapidez de la transmisión según los requerimientos de la aplicación [35].

Capas superiores (sesión, presentación y aplicación)

Permiten la interacción entre aplicaciones, como ROS, donde se gestionan los datos del sensor LiDAR y se procesan para generar mapas del entorno.

2.1.15.3. Modelo TCP/IP

El modelo TCP/IP es una implementación práctica del modelo por capas utilizada en redes modernas. Este modelo simplifica la arquitectura en cuatro capas principales [35].

Table 8 Capas del modelo TCP/IP.

Capa	Función	Protocolos
Aplicación	Servicios al usuario	HTTP, ROS, DDS
Transporte	Comunicación de extremo a extremo	TCP, UDP
Internet	Enrutamiento	IP
Acceso a red	Acceso físico	Wi-Fi, Ethernet

En el sistema propuesto:

- **Capa de aplicación:** ROS gestiona los datos del LiDAR
- **Capa de transporte:** TCP/UDP transmiten la información
- **Capa de red:** IP direcciona los robots
- **Capa física:** WiFi transmite las señales

Esta estructura permite comprender cómo los datos capturados por el sensor LiDAR viajan desde un robot hasta otro dentro del sistema.

2.1.15.4. Flujo de datos en el sistema multi-robot

El proceso de comunicación dentro del sistema puede describirse mediante el paso de datos a través de las diferentes capas:

- El sensor LiDAR genera datos (capa de aplicación)
- ROS organiza la información en mensajes
- TCP/UDP encapsula los datos (transporte)
- IP direcciona los paquetes (red)
- Wi-Fi transmite la señal (física)
- En el receptor, este proceso se realiza de manera inversa hasta reconstruir la información original.

2.1.15.5. Encapsulación de datos

La encapsulación es el proceso mediante el cual cada capa agrega información adicional (cabeceras) a los datos antes de transmitirlos.

Table 9 Proceso de encapsulación.

Capa	**Unidad de datos**
Aplicación	Datos
Transporte	Segmentos
Red	Paquetes
Enlace	Tramas
Física	Bits

2.1.16. Relación entre capas y calidad de servicio

El desempeño de la comunicación depende de la interacción entre las diferentes capas del modelo.

- La capa física afecta la potencia de la señal

- La capa de red influye en el enrutamiento
- La capa de transporte controla la confiabilidad

Una degradación en cualquiera de estas capas puede afectar parámetros como la latencia, la pérdida de paquetes y el jitter, impactando directamente en el funcionamiento del sistema multi-robot [36].

2.1.17. Consideraciones avanzadas de comunicación en sistemas multi-robot

2.1.17.1. Interferencias y ruido

En entornos interiores, la comunicación inalámbrica puede verse afectada por interferencias provenientes de otros dispositivos electrónicos. Estas interferencias generan degradación en la señal y aumentan la probabilidad de errores en la transmisión.

2.1.17.2. Movilidad de los nodos

El movimiento de los robots introduce variaciones en el canal de comunicación, afectando la calidad del enlace. Este fenómeno está relacionado con el desvanecimiento rápido (fast fading) [36].

2.1.17.3. Escalabilidad del sistema

A medida que aumenta el número de robots, se incrementa la demanda de ancho de banda y la complejidad de la comunicación. Por ello, es necesario diseñar sistemas eficientes que optimicen el uso de la red.

Table 10 Factores que afectan la comunicación en sistemas multi-robot

Factor	Descripción	Impacto
Interferencia	Señales externas	Pérdida de datos

Obstáculos	Paredes y objetos	Atenuación
Movilidad	Movimiento de robots	Variación del canal
Congestión	Tráfico en red	Latencia

CAPÍTULO III

DESARROLLO DE LA PROPUESTA

3.1. Componentes de la propuesta

La propuesta planteada se fundamenta en la integración de componentes físicos y lógicos que permiten la adquisición de datos del entorno, su procesamiento y la generación de mapas para el análisis de la propagación de señales en interiores.

Los componentes físicos están constituidos por el sistema robótico y los sensores utilizados para la captura de información, mientras que los componentes lógicos comprenden el software necesario para el control, procesamiento y análisis de los datos obtenidos.

3.1.1. Componentes físicos

Introducción

3.1.1.1. ROSMASTER X3 Jetson Nano 4GB SUB

El robot móvil ROSMASTER X3 es una plataforma educativa y de investigación diseñada para aplicaciones en robótica móvil, visión artificial y navegación autónoma. Este sistema integra hardware modular con capacidades de procesamiento basadas en sistemas embebidos como NVIDIA Jetson Nano o Raspberry Pi, permitiendo la ejecución de algoritmos avanzados de percepción y control [37].

Ilustración 6 ROSMASTER X3 Jetson Nano 4GB SUB

El ROSMASTER X3 está orientado al uso con ROS/ROS2, lo que facilita la implementación de algoritmos de mapeo, localización y navegación autónoma. Además, dispone de sensores, actuadores y módulos de comunicación que permiten su integración en entornos reales para la adquisición de datos y análisis espacial [38].

- Su arquitectura combina:
- Sistema de locomoción omnidireccional
- Sensores (LiDAR, cámara)
- Unidad de procesamiento (Jetson)
- Controlador embebido (STM32)

Esto lo convierte en una plataforma adecuada para aplicaciones de mapeo en interiores y estudios de propagación de señales.

3.1.1.1.1. Componentes del ROSMASTER

3.1.1.1.1.1. Sensor LiDAR RPLIDAR A1

El RPLIDAR A1 es un sensor de escaneo láser 2D que permite medir distancias mediante tecnología LiDAR, generando nubes de puntos del entorno en tiempo real. Es ampliamente utilizado en aplicaciones de SLAM y navegación autónoma [39].

.

Ilustración 7: Sensor LiDAR RPLiDAR A1.

Table 11 Especificaciones del RPLIDAR A1

Parámetro	Valor
Tipo	LiDAR 2D
Rango	0.15 – 12 m
Frecuencia	5 – 10 Hz
Alimentación	5V
Interfaz	USB / UART
Aplicación	Mapeo y navegación

3.1.1.1.1.2. Cámara de profundidad

La cámara de profundidad permite obtener información tridimensional del entorno mediante sensores ópticos, complementando los datos del LiDAR para aplicaciones de percepción avanzada [38].

Ilustración 8: Cámara de profundidad.

Table 12 Especificaciones cámara de profundidad.

Parámetro	Valor
Tipo	Cámara RGB-D
Salida	Imagen + profundidad
Alimentación	5V
Interfaz	USB
Aplicación	Visión artificial

3.1.1.1.1.3. Controlador inalámbrico

Permite el control remoto del robot durante pruebas y validación del sistema, facilitando la interacción directa con el robot.

Ilustración 9: Controlador inalámbrico.

Table 13 Especificaciones controlador inalámbrico.

Parámetro	Valor
Tipo	Control inalámbrico
Comunicación	2.4 GHz
Alimentación	Batería
Aplicación	Control manual

3.1.1.1.1.4. Board de expansión ROS (STM32)

Esta placa es el núcleo de control de bajo nivel del robot. Gestiona motores, sensores y comunicación con el sistema principal mediante protocolos seriales [37].

Ilustración 10: Placa STM32.

Table 14 Especificaciones placa STM32.

Parámetro	Valor
Microcontrolador	STM32
Voltaje	5V – 12V
Comunicación	UART / CAN
Función	Control de hardware

3.1.1.1.1.5. Motores DC con encoders

Los motores DC proporcionan el movimiento del robot, mientras que los encoders permiten medir velocidad y posición, fundamentales para la odometría [38].

Ilustración 11: Motores DC.

Table 15 Especificaciones motores DC.

Parámetro	Valor
Tipo	Motor DC
Control	PWM
Voltaje	6V – 12V
Sensor	Encoder
Aplicación	Locomoción

3.1.1.1.1.6. Jetson Nano 4GB

El Jetson Nano es un computador embebido desarrollado por NVIDIA, diseñado para aplicaciones de inteligencia artificial y robótica. Permite ejecutar ROS2, visión por computadora y algoritmos de aprendizaje automático [40].

Ilustración 12: Jetson Nano 4GB.

Table 16 Especificaciones Jetson Nano 4GB.

Parámetro	**Valor**
CPU	Quad-core ARM Cortex-A57
GPU	128 CUDA cores
RAM	4GB
Alimentación	5V / 4ª
Sistema operativo	Linux (Ubuntu)
Aplicación	Procesamiento

3.1.1.1.1.7. Antenas WiFi

Permiten la comunicación inalámbrica del robot con estaciones externas, facilitando la transmisión de datos.

Ilustración 13: Antenas WiFi.

Table 17 Especificaciones de antenas Wi-Fi.

Parámetro	**Valor**
Tipo	WiFi
Frecuencia	2.4 y 5.8 GHz
Función	Comunicación
Aplicación	Monitoreo remoto

3.1.1.1.1.8. Ruedas omnidireccionales (Mecanum)

Permiten el movimiento en múltiples direcciones sin cambiar la orientación del robot, lo que mejora la navegación en interiores.

Ilustración 14: Ruedas omnidireccionales.

Table 18 Especificaciones ruedas omnidireccionales.

Parámetro	Valor
Tipo	Mecanum
Configuración	4 ruedas
Movimiento	Omnidireccional
Aplicación	Navegación

3.1.1.1.1.9. Módulo de alimentación.

Proporciona energía a todos los componentes del robot.

Ilustración 15: Módulo de alimentación.

Table 19 Especificaciones módulo de alimentación.

Parámetro	Valor
Tipo	Batería recargable
Voltaje	7.4V – 12V
Salidas	Reguladas
Aplicación	Energía del sistema

3.1.1.1.1.10. Cable USB y Disk

El puerto USB hace las veces de conexión física entre el computador embebido (Jetson Nano) y otros dispositivos como sensores, computadoras de desarrollo o periféricos, a los que se les puede cargar programas, depurar el sistema y establecer comunicación a través de la interfaz de datos [41].

Además, la unidad de almacenamiento USB (U Disk) se utiliza para almacenar y transferir archivos que van desde configuraciones del sistema, archivos de registro (logs) y mapas generados hasta paquetes de software necesarios para hacer funcionar el sistema. Estos dispositivos son especialmente útiles en aquellos entornos donde se requiere portabilidad y respaldo de información [42].

Ilustración 16: Cable USB y Disk.

Table 20 Especificaciones cable USB y Disk.

Parámetro	Cable USB	U Disk (Memoria USB)
Tipo	USB 2.0 / 3.0	Dispositivo de almacenamiento
Voltaje	5V	5V
Interfaz	USB-A / Micro USB / USB-C	USB-A

Función	Comunicación y transferencia de datos	Almacenamiento de datos
Aplicación	Conexión con Jetson y periféricos	Backup y transporte de información

3.1.1.1.1.11. Chasis del robot

El chasis es la estructura donde se sujetan todos los componentes electrónicos y mecánicos del robot. Para el caso del ROSMASTER X3, el chasis está fabricado en aleación de aluminio. De esta forma obtendremos una buena resistencia mecánica y un bajo peso que facilitará el desplazamiento del sistema [37].

Su diseño permite la correcta distribución de los elementos, lo que contribuye a la estabilidad del vehículo en movimiento y a la reducción de las vibraciones que pueden afectar a la captación de datos.

Ilustración 17: Chasis del ROSMaster X3.

Table 21 Especificaciones del chasis.

Parámetro	**Valor**
Material	Aleación de aluminio
Tipo de estructura	Modular
Resistencia	Alta
Peso	Ligero
Aplicación	Soporte estructural

3.1.1.2. Funcionalidad de cada robot ROSMASTER X3 dentro del sistema

El sistema propuesto se basa en la utilización de dos plataformas robóticas ROSMASTER X3, las cuales operan de manera coordinada bajo un esquema colaborativo. La distribución de funciones entre ambos robots permite optimizar el proceso de adquisición y procesamiento de datos, mejorando la eficiencia del sistema y la calidad de los resultados obtenidos.

3.1.1.2.1. Robot explorador

El primer robot cumple la función de exploración del entorno, siendo responsable de la captura de información espacial mediante el sensor LiDAR 2D. Este robot se desplaza dentro del laboratorio de telecomunicaciones realizando el escaneo del entorno, lo que permite obtener mediciones de distancia en diferentes direcciones.

A partir de esta información, el robot explorador identifica la presencia de obstáculos, paredes y otros elementos estructurales del entorno. Además, su sistema de locomoción omnidireccional le permite desplazarse con alta maniobrabilidad en espacios reducidos, facilitando la cobertura completa del área de estudio.

- Durante su operación, este robot se encarga de:
- Capturar datos de distancia mediante el sensor LiDAR.
- Realizar el recorrido del entorno de manera controlada.
- Detectar obstáculos y estructuras físicas.
- Generar datos crudos del entorno para su posterior procesamiento.

3.1.1.2.2. Robot procesador

El segundo robot cumple la función de procesamiento de la información y generación de mapas del entorno. Este robot recibe los datos adquiridos por el robot explorador y realiza su tratamiento mediante herramientas de software basadas en ROS2.

En esta etapa, los datos son sometidos a procesos de filtrado y estructuración, con el fin de eliminar ruido e inconsistencias. Posteriormente, se aplican algoritmos de mapeo y localización (SLAM) que permiten generar representaciones bidimensionales del entorno.

- Las funciones principales de este robot incluyen:
- Recepción de datos provenientes del robot explorador.
- Procesamiento y filtrado de la información adquirida.
- Generación de mapas bidimensionales del entorno.
- Ejecución de algoritmos SLAM.
- Preparación de la información para el modelado de propagación.

3.1.2. Componentes lógicos

Los componentes lógicos corresponden al software utilizado para el control del robot, adquisición de datos, procesamiento de información y generación de mapas. Para este proyecto, se priorizan herramientas gratuitas, de código abierto y ampliamente utilizadas en el ámbito académico.

3.1.2.1. Python

Python es utilizado como lenguaje principal para la implementación de nodos dentro del entorno ROS2. Su estructura permite desarrollar de forma clara los procesos de lectura de datos del sensor LiDAR, procesamiento de información y control del robot.

La integración de Python con ROS2 facilita la manipulación de datos en tiempo real, permitiendo implementar algoritmos de análisis y procesamiento sin requerir estructuras complejas. Esto resulta adecuado en entornos académicos, donde se busca un equilibrio entre funcionalidad y facilidad de desarrollo [43].

Ilustración 18: Python.

3.1.2.2. Rviz

RViz es la herramienta empleada para la visualización de los datos generados por el sistema. A través de esta plataforma es posible representar gráficamente las lecturas del sensor LiDAR, así como los mapas generados durante el proceso de mapeo.

La visualización en tiempo real permite verificar la correcta adquisición de datos y analizar el comportamiento del robot dentro del entorno, facilitando la validación del sistema y la detección de posibles errores durante la ejecución [20].

Ilustración 19 Rviz

3.1.2.3. Visual Studio Code

Visual Studio Code es un entorno de desarrollo integrado (IDE) ligero y de código abierto desarrollado por Microsoft, ampliamente utilizado en proyectos de robótica y desarrollo con ROS.

Permite la edición de código en múltiples lenguajes como Python y C++, así como la integración con herramientas de depuración, control de versiones y ejecución de scripts [45].

En aplicaciones con ROS2, Visual Studio Code facilita:

- Desarrollo de nodos
- Edición de scripts en Python
- Integración con terminal Linux
- Gestión de paquetes y librerías

Ilustración 20 Visual Studio Code

3.1.2.6. Node js

Se instaló como el entorno de ejecución para Node-RED, proporcionando la infraestructura necesaria para manejar paquetes npm y ejecutar aplicaciones JavaScript en el servidor. Su instalación fue un requisito previo indispensable para desplegar el dashboard web de control y monitoreo. Node.js facilita la gestión de las dependencias de Node-RED y su integración con el sistema operativo. Actúa como el motor que impulsa la interfaz de usuario basada en navegador, permitiendo una interacción ágil y eficiente.

3.1.2.7 Node Red

Se instaló como el entorno de ejecución para Node-RED, proporcionando la infraestructura necesaria para manejar paquetes npm y ejecutar aplicaciones JavaScript en el servidor. Su instalación fue un requisito previo indispensable para desplegar el dashboard web de control y monitoreo. Node.js facilita la gestión de las dependencias de Node-RED y su integración con el sistema operativo. Actúa como el motor que impulsa la interfaz de usuario basada en navegador, permitiendo una interacción ágil y eficiente.

3.1.2.8 Mosquitto (broker MQTT)

Se instaló en la estación de trabajo como el servidor de mensajes MQTT, estableciendo el punto central de comunicación entre el robot y la PC. Escucha en el puerto 1883 y permite la transferencia de datos en tiempo real de forma eficiente y ligera. Publica tópicos como explorer/odometric/pose y suscribe comandos desde explorer/cmd_vel. Es el pilar que asegura la comunicación bidireccional y desacoplada entre el hardware del robot y el software de control, garantizando la integridad de los mensajes.

3.1.2.9 Librerías Python (paho-mqtt, opencv, matplotlib, numpy, Pillow)

Se instalaron para dotar a la aplicación Python de todas las capacidades técnicas requeridas. paho-mqtt facilita la conexión con el broker Mosquitto, mientras que opencv y Pillow procesan el flujo de video de la cámara en tiempo real. Matplotlib y numpy son esenciales para generar las gráficas de trayectoria y LiDAR en la interfaz de usuario. En conjunto, forman un ecosistema de procesamiento de datos y visualización gráfica robusto y eficiente, adaptado a las necesidades del proyecto.

3.1.2.10 Nodos y Servicios en el Robot

Se implementaron múltiples nodos en el Jetson Nano para gestionar de forma independiente cada función del robot. Estos incluyen control de motores, lectura de encoders, fusión con IMU, publicación de odometría y estado, y transmisión de video. Cada nodo se ejecuta como un servicio systemd que arranca automáticamente y publica sus datos en MQTT. Están diseñados para trabajar en paralelo sin interferencias, asegurando un sistema modular y estable, con una arquitectura que facilita el mantenimiento y la escalabilidad.

3.1.2.11 Aplicación Python (App de Control)

Se desarrolló una interfaz gráfica de escritorio con cuatro pestañas especializadas: Control y Visión, Navegación, Trayectoria y Estado. La aplicación se conecta al broker MQTT para recibir los datos del robot y enviar comandos de movimiento en tiempo real. Su diseño incluye un joystick virtual, video en vivo, gráfica de trayectoria y un panel de estado detallado. Es la principal herramienta de

interacción con el robot desde la PC, ofreciendo control y monitoreo completo de todas las variables del sistema, con una experiencia de usuario optimizada y eficiente.

3.1.3 Equipos y herramientas de medición

Además de los robots ROSMASTER X3 y los sensores LiDAR, se emplearon dispositivos de medición de señal Wi-Fi. Las mediciones de RSSI se realizaron utilizando el adaptador Wi-Fi integrado en el robot explorador. Los valores fueron registrados en tiempo real mediante ROS, publicados en el topico /wifi_signal y posteriormente exportados a un archivo CSV para su análisis. Esta metodología permitió obtener la intensidad de señal en dBm, asociada a la posición del robot en el mapa 2D generado por SLAM.

3.2. Diseño de la propuesta

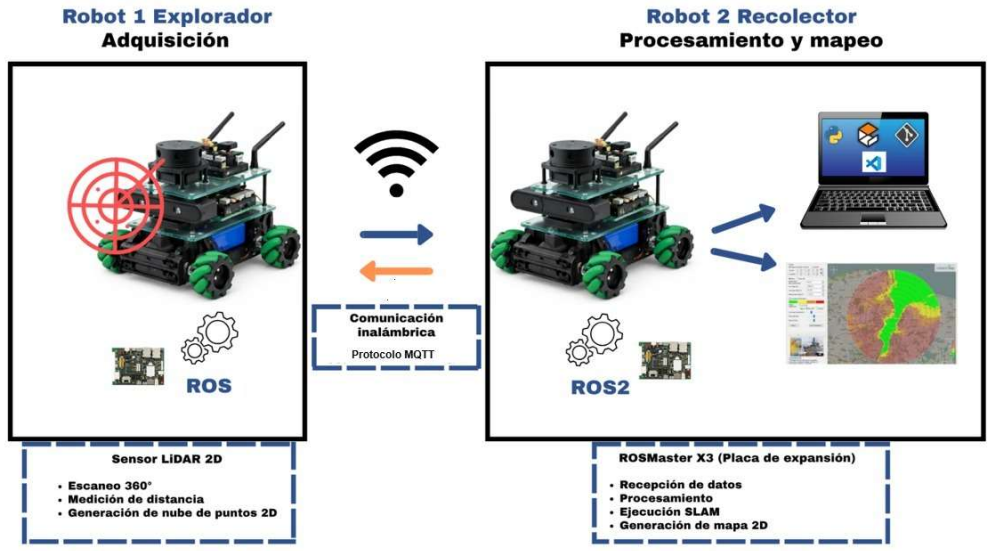


Ilustración 21: Diseño de la propuesta.

El diseño de la propuesta se estructura en un flujo secuencial compuesto por tres etapas principales: adquisición de datos, procesamiento de la información y análisis mediante técnicas de mapeo. Este enfoque facilita la integración de un sistema colaborativo conformados por dos robots

móviles, apoyados en herramientas de software, para generar mapas del entorno interior que sirven como base en el estudio de la propagación de señales inalámbricas.

Adquisición de datos

La primera etapa del sistema corresponde al proceso de adquisición de información del entorno físico mediante el Robot Explorador. Para ello, la plataforma ROSMASTER X3 incorpora un sensor LiDAR RPLIDAR A1 encargado de realizar un escaneo continuo del laboratorio de telecomunicaciones.

Su funcionamiento se basa en el principio de que se emiten unos pulsos de láser que, al impactar en un objeto, son devueltos al sensor. En función del tiempo que tarda el rayo de luz en hacer el trayecto de ida y vuelta se puede calcular la distancia entre el sensor y los objetos que hay a su alrededor.

Mientras el robot se desplaza, el sensor hace un escaneo 360°, obteniendo miles de medidas de distancia por segundo, cada una de ellas asociada a un cierto ángulo de escaneo, por lo que se puede ir generando un mapa 2D de la escena.

Simultáneamente, el sistema registra información proveniente de la odometría del robot, velocidad de desplazamiento y orientación, variables necesarias para la correcta localización del robot durante el proceso de generación del mapa.

Todos estos datos son organizados mediante nodos desarrollados en ROS2, donde cada nodo se encarga de administrar una función específica del sistema antes de iniciar el proceso de transmisión.

Comunicación mediante MQTT

Una vez obtenida la información del entorno, se inicia la etapa de transmisión de datos entre ambos robots utilizando el protocolo MQTT. Este protocolo fue seleccionado debido a su arquitectura ligera, bajo consumo de recursos computacionales y elevada eficiencia para aplicaciones de robótica colaborativa e Internet de las Cosas.

La comunicación se desarrolla bajo el modelo Publisher/Subscriber. En este esquema, el Robot Explorador actúa como Publisher publicando continuamente la información capturada por el sensor LiDAR, mientras que el Robot Procesador permanece suscrito a los diferentes tópicos configurados dentro del Broker MQTT.

La parte de comunicación está gestionada por el Broker Eclipse Mosquitto, que se encarga de enviar y recibir los mensajes entre las dos plataformas, lo que permite que los robots no tengan que comunicarse entre ellos, simplificando así la arquitectura del sistema.

Los principales datos son las medidas del sensor LiDAR, los datos de odometría, la velocidad, el estado del robot y las variables necesarias para el funcionamiento del sistema.

Esta arquitectura también permite que se mantengan las comunicaciones cuando se aumenta el número de dispositivos conectados a la red, lo que la convierte en una solución escalable para futuras ampliaciones del proyecto.

Procesamiento de la información

Los datos obtenidos por el Robot Procesador son organizados a través de la herramienta de middleware ROS2, la cual se encarga de orquestar la comunicación entre los nodos procesadores.

La información inicial se valida y sincroniza. La sincronización se realiza comprobando la validez de cada uno de los mensajes recibidos en el protocolo de comunicación MQTT. A continuación, se filtran las mediciones del sensor LiDAR para eliminar las medidas erróneas, el ruido que se haya producido en la captura y los outliers que puedan interferir en la generación del mapa.

Una vez depurada la información, las coordenadas originalmente obtenidas en formato polar son transformadas al sistema de coordenadas cartesianas, permitiendo representar espacialmente cada una de las mediciones realizadas durante el recorrido del robot.

El procesamiento incluye odometría, abriendo la posibilidad de saber en todo momento la posición del robot en el entorno que está explorando.

Generación de mapas

La etapa final del procesamiento consiste en generar mapas 2D mediante la implementación de algoritmos SLAM en ROS2.

Estos algoritmos se utilizan para crear un mapa a partir de la información que proporciona el sensor LiDAR y la estimación de la trayectoria del robot.

Como resultado se obtiene un mapa de ocupación, que contiene los muros, los obstáculos, los espacios libres y otros elementos que se encuentran en el interior del laboratorio de telecomunicaciones.

Los mapas son luego visualizados en el software RViz2 y en el software de visualización de datos Node-RED, para comprobar el comportamiento del sistema y que el entorno se ha mapeado correctamente.

Estos mapas se utilizan para el modelado de la propagación de la señal de radio y permiten establecer una relación entre la representación del escenario y la forma en que el canal se comporta.

3.3 Implementación del sistema

La implementación del sistema se desarrolló mediante la integración de diferentes herramientas de software orientadas al control de los robots, la comunicación de datos, el procesamiento de la información y la visualización de los resultados. Para ello, se emplearon aplicaciones de código abierto que facilitaron el desarrollo de una arquitectura colaborativa basada en el protocolo MQTT.

La comunicación entre los dos robots ROSMASTER X3 se realiza mediante un enlace de red inalámbrico local. El Robot Explorador recoge datos de su entorno mediante el sensor LiDAR 2D y los envía por MQTT al Robot Procesador, que se encarga de recibir la información y procesarla para la creación de mapas 2D del entorno.

El sistema fue implementado a partir de la configuración del entorno, el broker MQTT, la interfaz de supervisión en Node-RED y la aplicación en Python que procesa y controla los datos que se envían entre las diferentes plataformas.

3.3.1. Configuración del entorno de desarrollo

Visual Studio Code fue utilizado como entorno principal de desarrollo para la programación, configuración y ejecución de los paquetes relacionados con ROS2 y Python. Esta herramienta permitió gestionar el código fuente, ejecutar terminales integradas y facilitar la integración de herramientas de simulación y visualización utilizadas en el proyecto.

El desarrollo del sistema se realizó utilizando Visual Studio Code como entorno de programación principal, debido a su compatibilidad con Python y a las herramientas empleadas durante la implementación del proyecto. Este entorno permitió desarrollar, editar y depurar el código encargado de la comunicación entre los robots, el procesamiento de la información y la interacción con el protocolo MQTT.

Como lenguaje de programación se utilizó Python, debido a su facilidad de integración con librerías especializadas para comunicación de datos, procesamiento de información y desarrollo de aplicaciones. Gracias a su estructura modular fue posible organizar el proyecto en diferentes componentes independientes, facilitando el mantenimiento y la ampliación del sistema.

Además del entorno de programación, se instalaron las librerías necesarias para la comunicación MQTT, procesamiento de datos y representación gráfica de la información obtenida durante las pruebas experimentales.

3.3.2. Configuración del broker MQTT

Para establecer la comunicación entre el Robot Explorador y el Robot Procesador se implementó el broker Eclipse Mosquitto, encargado de administrar el intercambio de mensajes mediante el protocolo MQTT.

En la arquitectura de comunicación, el broker es el componente central que se encarga de recibir los mensajes publicados por el Robot Explorador y de distribuir automáticamente los mensajes a los clientes que están suscritos a los diferentes tópicos configurados en el sistema.

Esto permitió que la comunicación entre los dos robots se desacoplara totalmente, de forma que ambos pudieran compartir información, aunque no estuvieran conectados el uno con el otro.

Durante las pruebas realizadas se verificó el correcto envío y recepción de datos correspondientes a las mediciones del sensor LiDAR, información de odometría, estado del robot y comandos de control, garantizando una comunicación estable durante el funcionamiento del sistema.

3.3.3 Desarrollo de la aplicación en Python

La lógica principal del sistema fue desarrollada en Python, lenguaje utilizado para implementar los módulos encargados de la comunicación mediante MQTT, el procesamiento de los datos provenientes del sensor LiDAR y la interacción entre ambos robots.

La aplicación tiene una arquitectura modular, en la que se separan las funcionalidades de adquisición, transmisión, recepción y tratamiento de la información, lo que permite un fácil mantenimiento del código y la inclusión de nuevas funcionalidades en futuras versiones del sistema.

Se utilizó la librería Paho-MQTT para la comunicación con el broker MQTT, esta es utilizada en la mayoría de las aplicaciones IoT por su fácil implementación y su estabilidad en la transmisión de mensajes.

A continuación, se presenta un fragmento del código utilizado para establecer la conexión con el broker MQTT.

```
import paho.mqtt.client as mqtt

broker = "192.168.1.100"

puerto = 1883
```

```
cliente = mqtt.Client()

cliente.connect(broker, puerto)

cliente.publish("robot/lidar", datos_lidar)
```

El código anterior establece la conexión con el broker MQTT y publica la información adquirida por el sensor LiDAR en el tópico correspondiente, permitiendo que el Robot Procesador reciba los datos para su posterior procesamiento.

La interfaz de control incluye, además del joystick virtual, un botón de emergencia de color rojo, ubicado en la parte inferior del panel de control. Este botón permite detener inmediatamente todos los movimientos del robot al publicar un comando de velocidad cero $(\text{linear} = 0, \text{angular} = 0, \text{y} = 0)$ en el tópico MQTT explorer /cmd_vel. Su función es crítica para la seguridad operacional, ya que permite una interrupción inmediata del movimiento ante cualquier comportamiento inesperado del robot."

3.3.4 Configuración de Node-RED

Con el propósito de supervisar el funcionamiento del sistema se implementó **Node-RED**, herramienta utilizada para desarrollar una interfaz gráfica destinada al monitoreo de las variables transmitidas mediante MQTT la cual, una vez descardo se instaló en el cmd, como se muestra en la ilustración 22. Luego de esto, se procede a llamar con el comando node-red desde el terminal de cmd.

```
npm install -g node-red
```

Ilustración 22 Instalación del Node-RED

La Ilustración 23 muestra la consola de Node-RED durante su inicio, confirmando la correcta conexión con el broker MQTT en localhost:1883. Además, se evidencia la activación del Dashboard versión 3.6.6 y la carga de los flujos configurados para el monitoreo del sistema.

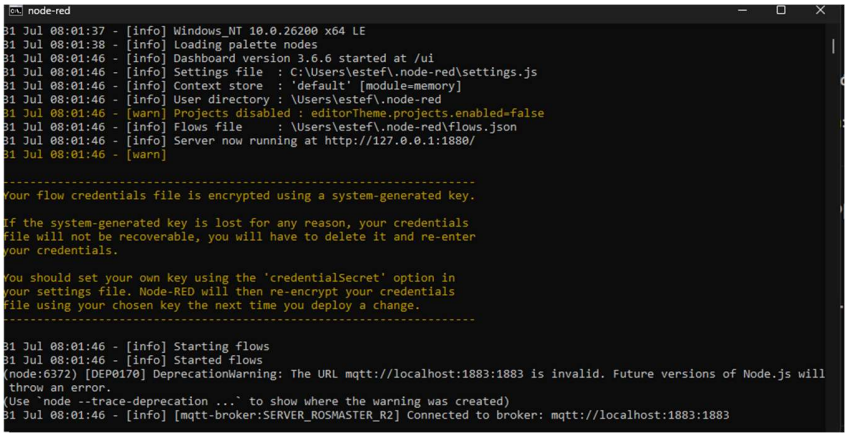


Ilustración 23 Consola mosquitto

Durante la configuración, se detectó una advertencia por la duplicación del puerto en la URL MQTT, el cual fue corregida en la configuración final del bróker. La interfaz permite visualizar en tiempo real información relacionada con el estado de los robots, la comunicación establecida mediante MQTT y los datos generados durante el proceso de exploración del entorno.

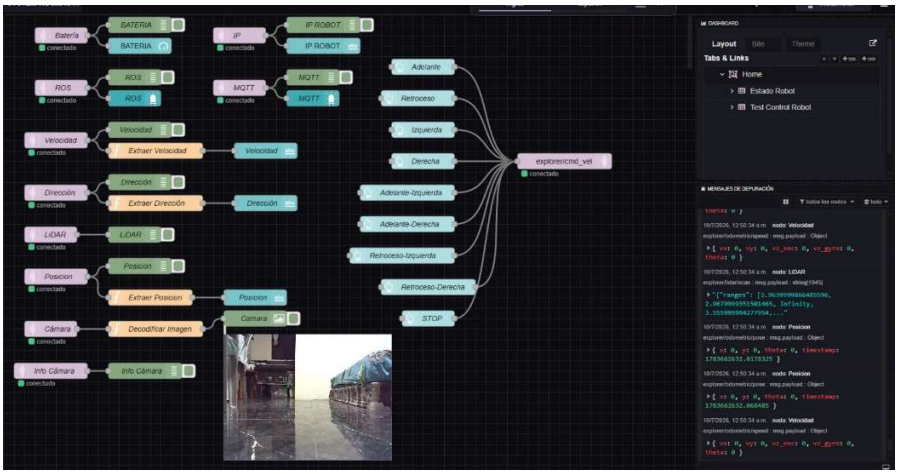


Ilustración 24 Interfaz Node-RED

Con Node-RED, el sistema puede ser supervisado sin necesidad de acceder al código fuente y visualiza de forma gráfica el flujo de información entre los dos robots.

De igual forma, la conexión de Node-RED con MQTT permite que se puedan añadir nuevos indicadores o variables de monitoreo a medida que el proyecto evoluciona. El dashboard de Node-RED incluye un botón de STOP que, al ser presionado, publica un comando de velocidad cero en el tópico

explorer /cmd_vel Este botón complementa la función de emergencia disponible en la App Python, proporcionando un mecanismo de detención remota desde cualquier dispositivo con acceso al dashboard web.

3.3.5. Integración del sistema colaborativo

Una vez configurados todos los componentes de software, se procedió a integrar ambos robots dentro de una misma arquitectura colaborativa basada en MQTT.

Durante el funcionamiento del sistema, el Robot Explorador realiza la adquisición continua de datos mediante el sensor LiDAR y publica la información correspondiente en los diferentes tópicos MQTT previamente definidos.

El Robot Procesador permanece suscrito a dichos tópicos, recibiendo automáticamente cada uno de los mensajes enviados por el Robot Explorador. Posteriormente, la información es procesada mediante los módulos desarrollados en Python, permitiendo generar los mapas bidimensionales utilizados para el análisis de la propagación de señales en interiores.

Finalmente, los resultados obtenidos son visualizados mediante Node-RED y las herramientas de procesamiento implementadas, permitiendo supervisar el funcionamiento general del sistema y verificar el correcto intercambio de información entre ambas plataformas robóticas.

3.3.6 Arquitectura de comunicación MQTT

Table 22 Función de los componentes de la arquitectura MQTT.

Componentes	Función
Robot Explorador	Captura los datos del sensor LiDAR y publica la información mediante MQTT.

Broker MQTT (Mosquitto)	Administra la comunicación entre los robots distribuyendo los mensajes publicados.
Robot Procesador	Recibe la información mediante suscripción MQTT y ejecuta el procesamiento de los datos.
Python	Procesa la información recibida y desarrolla la lógica del sistema.
Node-RED	Permite monitorear y visualizar las variables del sistema en tiempo real.

La comunicación entre el Robot Explorador y el Robot Procesador se implementó mediante el protocolo MQTT (Message Queuing Telemetry Transport), seleccionado por su eficiencia en aplicaciones donde es necesario intercambiar información entre múltiples dispositivos utilizando un bajo consumo de recursos de red.

La arquitectura implementada sigue el modelo de comunicación Publisher/Subscriber, donde el Robot Explorador actúa como publicador de la información obtenida durante la exploración del entorno, mientras que el Robot Procesador permanece suscrito a los diferentes tópicos configurados dentro del Broker MQTT para recibir los datos necesarios durante el proceso de generación de mapas.

Para la comunicación entre los diferentes módulos se usa el Broker Eclipse Mosquitto, que se encarga de la distribución de los mensajes publicados. De este modo se evita la comunicación directa entre robots y se obtiene una arquitectura desacoplada que permite el funcionamiento autónomo de cada plataforma y que puede escalar.

Dentro del funcionamiento del sistema, se han definido varios tópicos MQTT con el objetivo de clasificar la información que se maneja, de esta forma, cada tópico se relaciona con un tipo de dato, logrando que los dispositivos solo reciban información que les concierne.

Table 23 Tópicos MQTT utilizados durante la implementación.

Tópico MQTT	Publicador	Suscriptor	Información transmitida
robot/lidar	Robot Explorador	Robot Procesador	Datos obtenidos por el sensor LiDAR 2D.
robot/odom	Robot Explorador	Robot Procesador	Información de odometría del robot.
robot/status	Robot Explorador	Node-RED	Estado operativo del robot.
robot/map	Robot Procesador	Node-RED	Información del mapa generado.

La organización de los datos mediante tópicos permite mejorar la administración de la comunicación, reducir el tráfico innecesario dentro de la red y facilitar la incorporación de nuevos dispositivos sin modificar la arquitectura existente. Esta característica representa una de las principales ventajas del protocolo MQTT frente a otros mecanismos de comunicación utilizados en sistemas distribuidos. Ilustración 23.

3.3.7 Configuración del robot explorador

El Robot Explorador fue configurado para cumplir con las funciones de adquisición de datos LiDAR, odometría y transmisión de información mediante MQTT.

Table 24 Software Instalado.

Software	Versión	Función
ROS	Melodic	Middleware para control robótico
OLED	Oled_adafruit.py	Control de la pantalla OLED
Librería Rosmaster_Lib	V3.3.9	Control de motores, LEDs y sensores

paho-mqtt		Cliente MQTT para Python 2,7 3,6

El Robot Explorador captura datos del entorno mediante LiDAR, odometría, cámara y estado del sistema, procesándolos con ROS Melodic. Los datos son publicados en tópicos MQTT como explorer/lidar/scan, explorer/odometric/pose, explorer/camera/image y explorer/status/*, como se muestra en la ilustración 25.

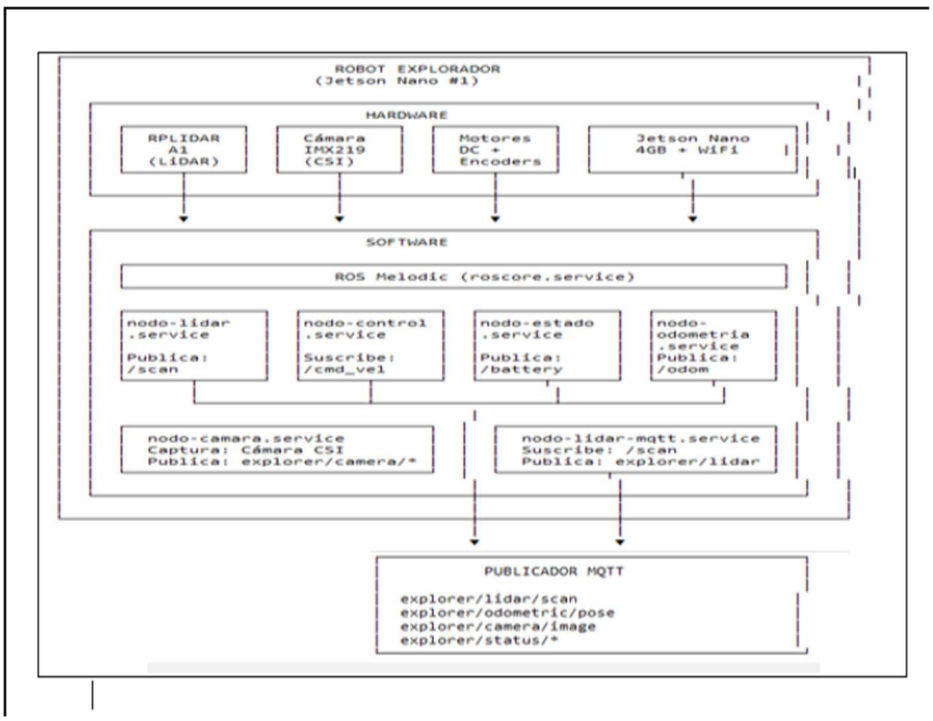


Ilustración 25 Diagrama del Robot Explorador

3.3.7.1. Servicios Systemd del robot explorador

Para garantizar el funcionamiento autónomo del Robot Explorador, se implementaron siete servicios systemd que se inician automáticamente al encender el robot. Cada servicio cumple una función específica dentro de la arquitectura del sistema. Tabla 25.

Table 25 Servicios del Robot Explorador.

Servicio	**Función**

roscore.service	ROS Master
nodo-lidar.service	Control del RPLIDAR A1
nodo-control.service	Control de motores (Mecanum)
nodo-estado.service	Batería, IP, OLED, ROS/MQTT status
nodo-odometria-py3.service	Odometría Mecanum + IMU
nodo-camara.service	Video en vivo
nodo-lidar-mqtt.service	Publicación LiDAR en MQTT

Para verificar el estado de los servicios systemd del Robot Explorador, se ejecutaron los comandos mostrados en la Ilustración 26. Cada comando retorna el estado actual del servicio correspondiente, permitiendo confirmar que todos los nodos necesarios para la exploración, control, odometría y comunicación se encuentran activos.

```
# En el Robot Explorador (Jetson #1)
sudo systemctl status roscore.service
sudo systemctl status nodo-lidar.service
sudo systemctl status nodo-control.service
sudo systemctl status nodo-estado.service
sudo systemctl status nodo-odometria-py3.service
sudo systemctl status nodo-camara.service
sudo systemctl status nodo-lidar-mqtt.service
```

Ilustración 26 Verificación del estado de los servicios systemd en el Robot Explorador

La ilustración 27 muestra la ejecución de los comandos sudo systemctl restart y sudo systemctl daemon-reload utilizados para reiniciar los servicios systemd del Robot Explorador. Este procedimiento permite aplicar cambios en la configuración sin necesidad de reiniciar el sistema completo, asegurando

la continuidad operativa del robot. Los servicios reiniciados incluyen el ROSMASTER, el LiDAR, el control de motores, el estado del sistema, la odometría y la cámara

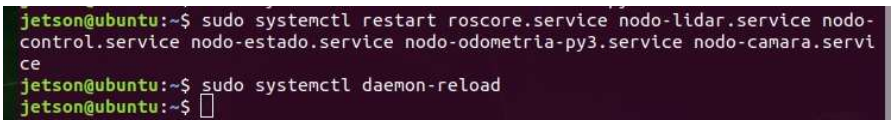


Ilustración 27 Reinicio de servicios systemd

3.3.7.2. Tópicos MQTT publicados por el robot Explorador

Table 26 Tópicos MQTT

Tópicos	**Descripción**
explorer/cmd_vel	Comandos de movimiento (suscrito)
explorer/status/battery	Voltaje de batería
explorer/status/ip	IP del robot
explorer/status/ros	Estado de ROS
explorer/status/mqtt	Estado de MQTT
explorer/odometric/pose	Posición y orientación
explorer/odometric/speed	Velocidades
explorer/odometric/motion	Dirección de movimiento
explorer/lidar/scan	Datos del LiDAR
explorer/camera/image	Imagen en Base64

3.3.8 Configuración del robot procesador

El Robot Procesador fue configurado para recibir los datos del Robot Explorador a través de MQTT, ejecutar el algoritmo SLAM y publicar el mapa generado.

Table 27 Servicios Systemd Robot Procesador

Servicio	Función
roscore.service	ROSMaster
mqtt-to-ros.service	Puente MQTT a ROS
hector-slam.service	Hector SLAM
map-to-mqtt.service	Publicación del mapa en MQTT
status-publisher.service	Publicación del estado del robot

3.3.9 Arquitectura del diagrama del robot procesador

El Robot Procesador recibe los datos del Explorador mediante suscripción MQTT, los convierte a ROS y ejecuta Hector SLAM para generar el mapa de ocupación. Publica el mapa y su estado en los tópicos processor/map y processor/status/*. Ilustración 28.

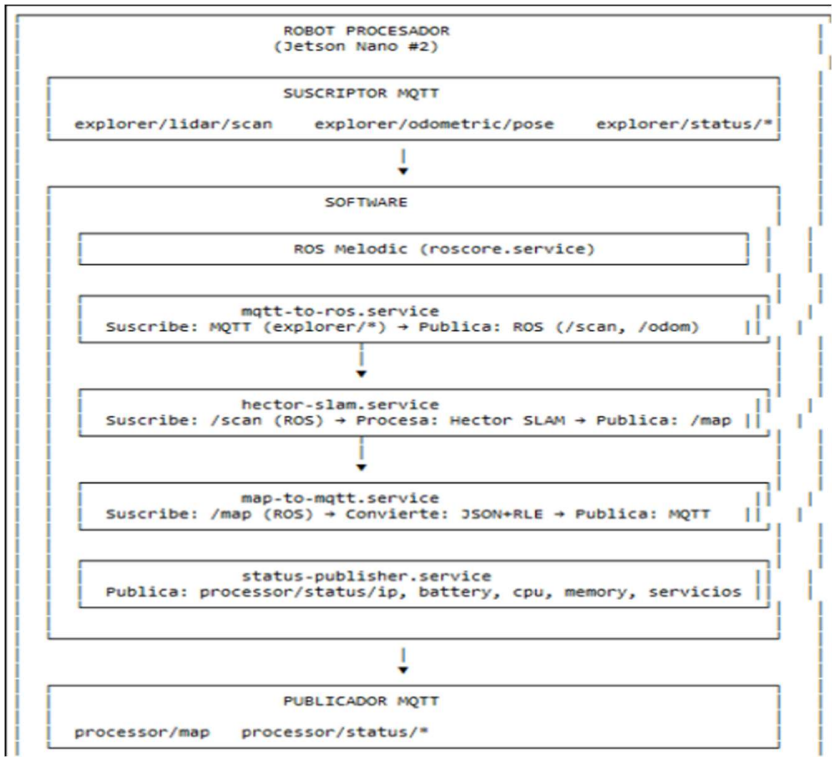


Ilustración 28 Diagrama del Robot Procesador

Para verificar el estado de los servicios systemd del Robot Procesador, se ejecutaron los comandos mostrados en el Anexo B. Cada comando corresponde a uno de los servicios críticos del sistema: el puente MQTT a ROS (mqtt-to-ros), el algoritmo Hector SLAM (hector-slam) y el publicador de mapas (map-to-mqtt). La ejecución de estos comandos permite confirmar que los nodos se encuentran operativos y listos para procesar los datos provenientes del Robot Explorador.

3.3.10 Arquitectura del diagrama de estación de trabajo

La estación de trabajo visualiza en tiempo real el mapa, LiDAR, trayectoria, video y estado de ambos robots mediante Node-RED y App Python. Además, envía comandos de movimiento al Explorador a través del tópico explorer/cmd_vel.

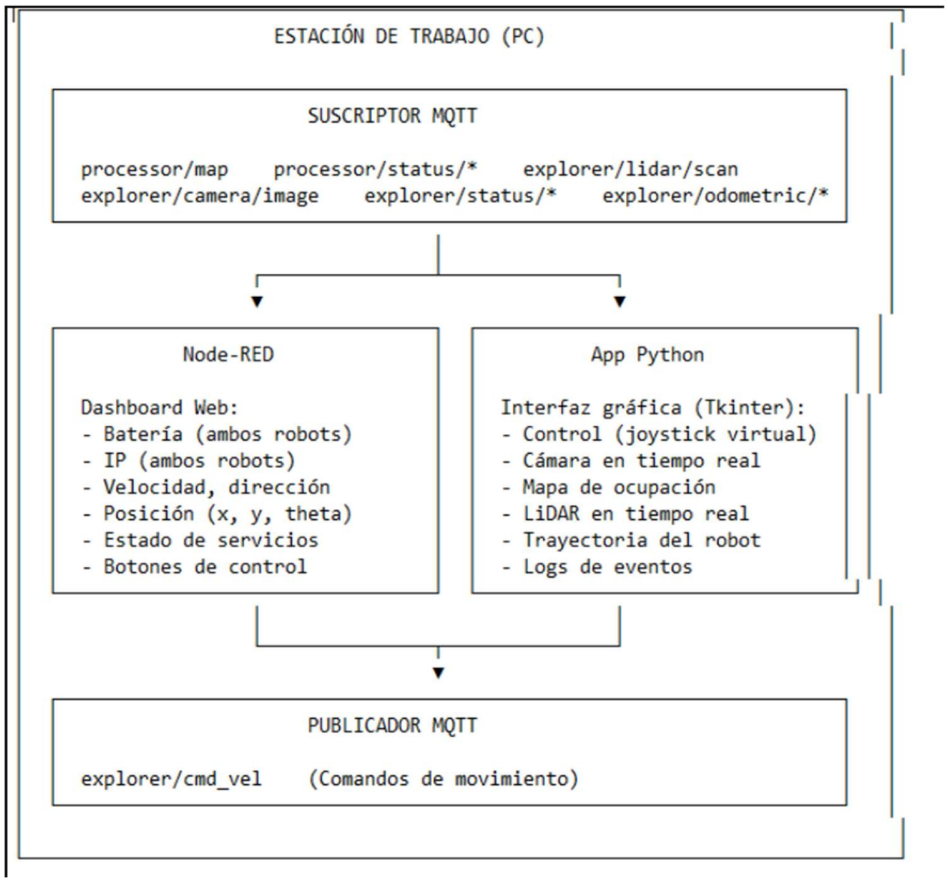


Ilustración 29diagrama de Estación de trabajo

3.3.11 Arquitectura del diagrama de Datos Completo

El flujo de datos completo muestra el recorrido de la información en cuatro fases: adquisición en el Explorador, transmisión por MQTT, procesamiento SLAM en el Procesador y visualización en la estación de trabajo.

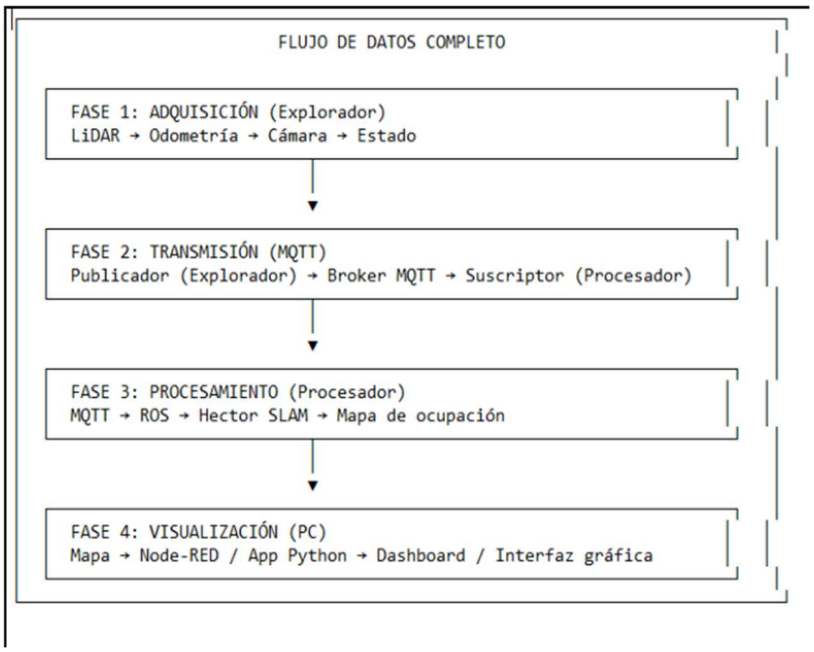


Ilustración 30Flujo de Datos Completo

3.3.12 Arquitectura del Flujo de Control

El flujo de control representa el camino de los comandos desde la interfaz de usuario (Node-RED/App) hasta el movimiento físico del robot, pasando por la publicación MQTT y el nodo de control del Explorador.

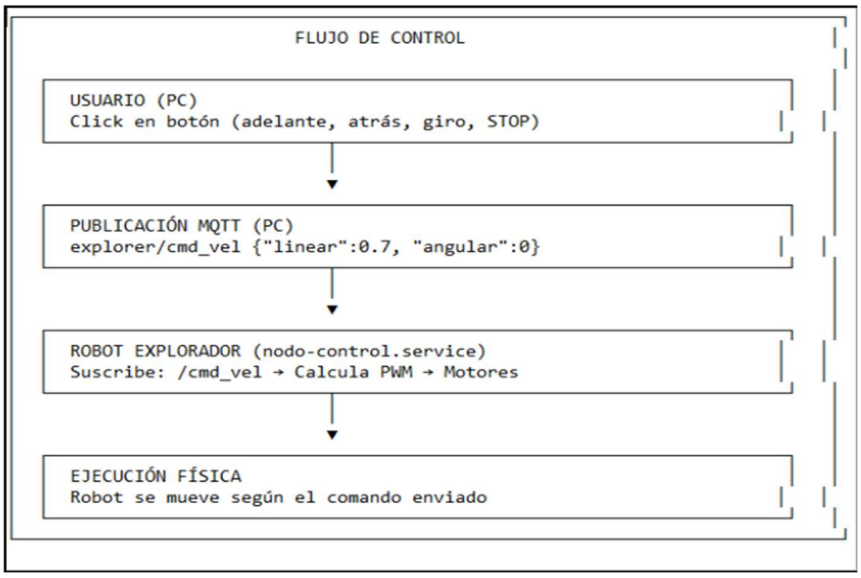


Ilustración 31Flujo de Control

CAPÍTULO IV

CÁLCULOS Y ANÁLISIS DE RESULTADOS

Este capítulo presenta los resultados obtenidos en la implementación y validación del sistema de exploración robótica, organizados según los cinco objetivos específicos de la investigación. Se incluyen la construcción del mapa 2D mediante SLAM en RViz, el funcionamiento del robot explorador, el procesamiento de datos en ROS2, la integración de mediciones de RSSI con el modelo teórico de propagación y la evaluación global del rendimiento del sistema.

4.1 Pruebas en dos entornos

Las pruebas fueron realizadas en dos entornos de interiores; una en el domicilio del autor como muestra la ilustración 32.

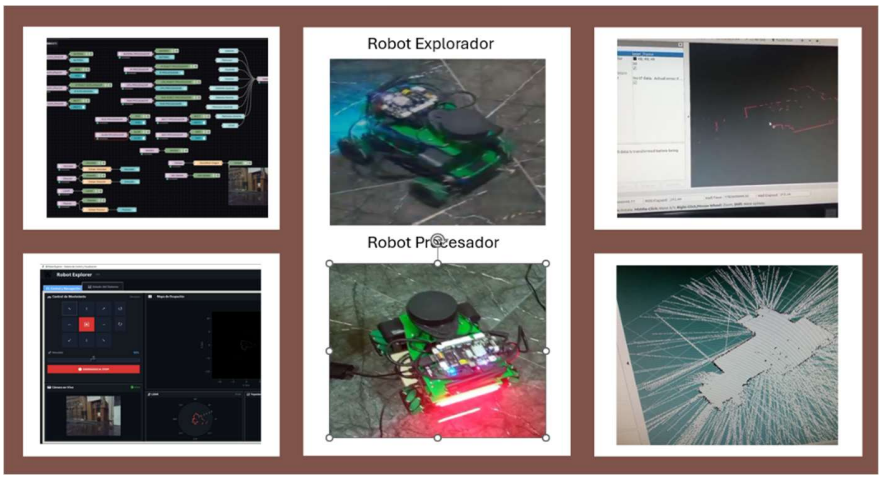


Ilustración 32 Pruebas en domicilio

La segunda prueba fue realizada en el laboratorio de telecomunicaciones donde ambos robots operaban simultáneamente. El cual el robot explorador realizaba un barrido por todo el laboratorio capturando y transmitiendo datos del LiDAR, odometría y video, como se puede observar en la ilustración 33.

Ilustración 33 Robot Explorador

4.2 Funcionamiento del robot explorador

Al momento que el robot explorador actúa como el sistema sensorial móvil de este proyecto, donde el LiDAR, la cámara y los sensores de odometría permiten que se pueda percibir y recopilar información detallada del entorno del laboratorio mientras se desplaza.

El robot captura datos en tiempo real (distancias, imágenes y posición) y empaquetarlos en mensajes JSON para transmitirlo mediante comunicación MQTT al robot procesador. Para esto el robot opera de manera autónoma o controlado por lo que actúa como los “ojos y oídos” del sistema, la carga de duración de la batería es de 6000 mAh y le da una duración de 4 horas para seguir explorando. como se muestra en la ilustración 34.

Debido a su diseño ligero y la capacidad de movimiento omni-direccinal (ruedas mecanum) permite que se mueva en cualquier dirección y maniobrar en espacios pequeños, convirtiéndose en recolector de información rápida y eficiente para la construcción del mapa en el robot procesador.

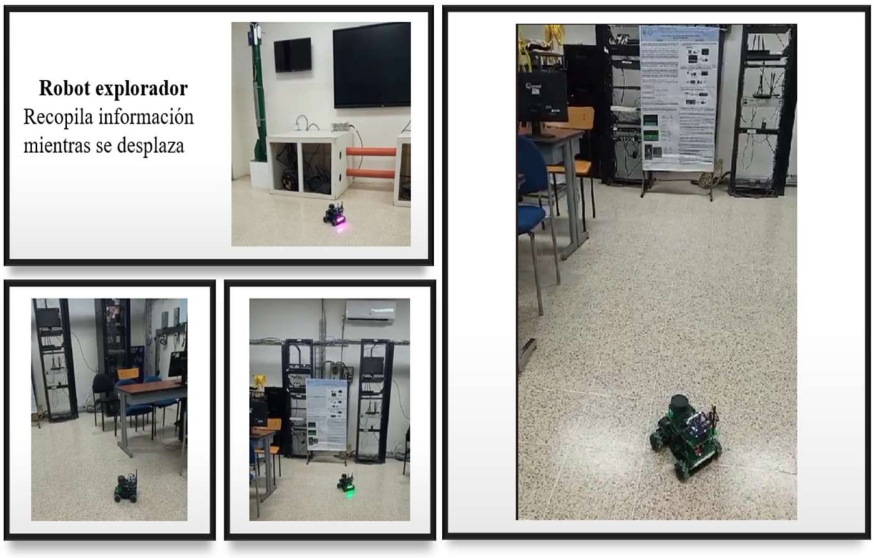


Ilustración 34 Robot explorador

La interfaz de Node-RED es un servidor que se ejecuta desde la laptop con el comando node-red en la terminal de símbolos del sistema. Para esto, la terminal tiene que permanecer abierta y en el navegador, se escribe http://localhost:1880 y automáticamente se abre la interfaz, la cual se utilizó para el monitoreo y control del sistema basado en ROS.

Muestra en tiempo real la batería (10.3V), la velocidad (Vx, Vy, Vz), la dirección ("stop"), la posición (X=1.69, Y=1.71) y la orientación (-23.2°) del robot explorador. Incluye botones para controlar sus movimientos (adelante, retroceso, giros, diagonales y stop).

El robot procesador presenta su estado de recursos (CPU 17.7%, RAM 45.3%), batería (10.9V) y su función principal: procesar el mapa y ejecutar SLAM, preparado para recibir datos del explorador y generar el entorno. Como se muestra en la ilustración 35.

Ilustración 35 Node-RED Dashboard

La aplicación Python es la interfaz principal del ejecutor, una ventana que organiza todo en pestañas para controlar el robot en tiempo real desde la PC, mostrando en tiempo real lo que ve la cámara, los datos del LiDAR y la trayectoria que ha recorrido.

Desde el mismo panel de control se puede maniobrar el robot, ajustar su velocidad y detenerlo de emergencia, mientras se observa cómo se mueve y escanea el entorno, y se transforman los datos técnicos en gráficos y números comprensibles, como la posición exacta, la batería o la dirección, haciendo que la exploración sea interactiva y visual. Ilustración 36.

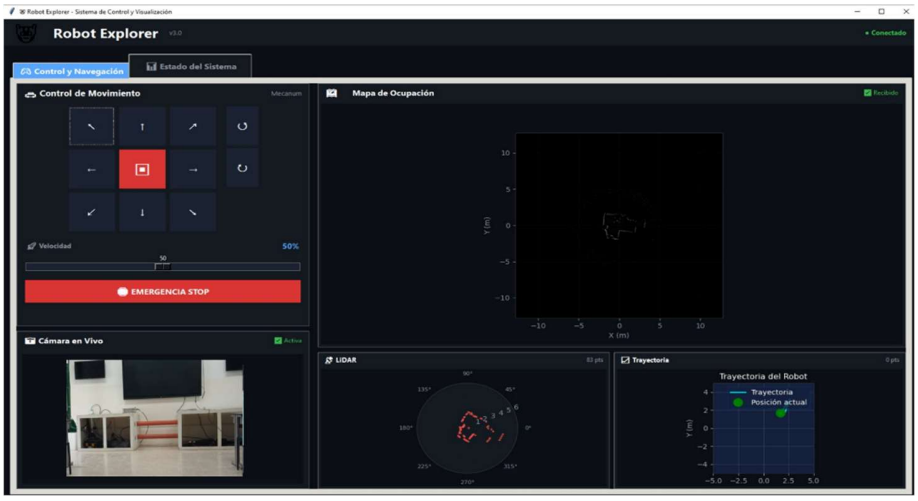


Ilustración 36 App Python

4.3 Funcionamiento del robot procesador

Una vez que el robot explorador ha emitido por broker MQTT las imágenes del LiDAR y la odometría estimada, el robot procesador (PC) se convierte en el cerebro de la estación de trabajo. Su primer trabajo es recibir toda la información emitida por el robot explorador, convertirla al formato ROS y publicarla en los topics para que pueda ser utilizada por el algoritmo de mapeo.

El siguiente nodo (gmapping) procesa los escaneos láser y las posiciones relativas uno a uno, construye un mapa 2D de ocupación del entorno y filtra el ruido y las incoherencias a partir de filtros probabilísticos.

Mientras tanto, el procesador controla la carga de la CPU, la RAM y la batería para no sobrecargar el sistema. El mapa resultante, que es una cuadrícula de celdas con valores de ocupación, se comprime en una imagen PNG y se publica en MQTT para que la estación de trabajo la muestre en tiempo real.

Esto proporciona una forma estructurada y filtrada de mapear el entorno y, al mismo tiempo, proporciona un formato compatible con la navegación, la planificación de trayectorias y el modelado de la propagación de señales, cerrando así el ciclo de percepción-acción de un sistema robótico. Como se muestra en la ilustración 37.

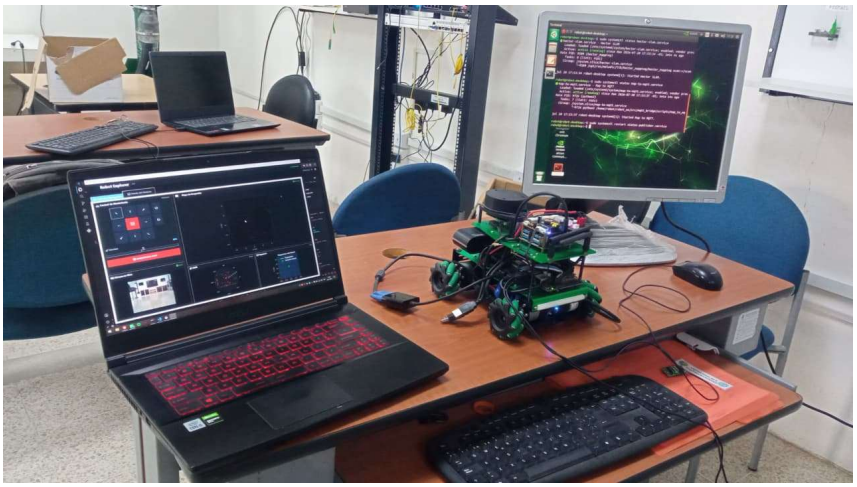


Ilustración 37 Robot Procesador

4.3.1 Funcionamiento del Hector SLAM

Una vez que el Robot Procesador completa su tarea, el mapa generado se visualiza en RVIZ como una representación en escala de grises del entorno explorado. Esta visualización es el resultado del algoritmo Hector SLAM, que procesa los datos del LiDAR sin necesidad de odometría, construyendo el mapa a partir de los escaneos del entorno, como observamos en la ilustración 38.

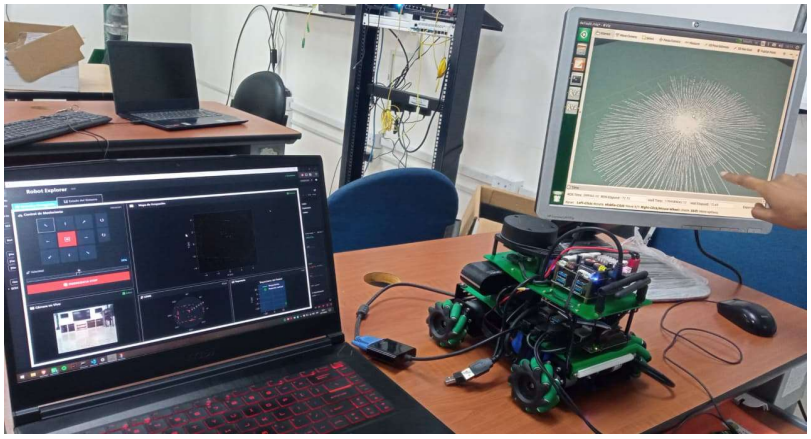


Ilustración 38 Hector SLAM

El mapa visualizado en RVIZ se representa como cuadrícula, donde cada celda equivale a 2.5 cm del entorno real. Los puntos negros señalan la presencia de obstáculos o paredes, las blancas corresponden al espacio libre muestran el espacio libre por donde el robot puede transitar, y las áreas grises corresponden a regiones aún no exploradas o desconocidas para el sistema, se puede apreciar en la ilustración 39.

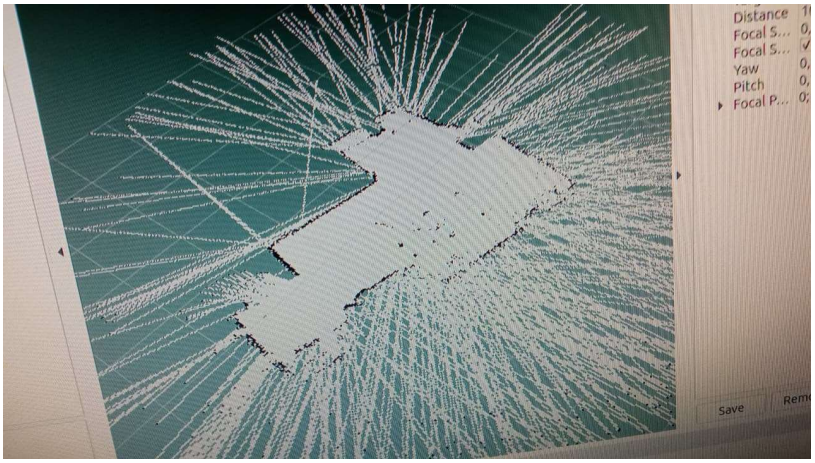


Ilustración 39 Visualización del Mapa en RVIZ

La calidad del mapa estará directamente vinculada con la cantidad y calidad de los escaneos LiDAR que hayan hecho mientras el robot estaba en movimiento. Cada vez que el explorador se mueve, el procesador incorpora puntos adicionales al mapa y actualiza la representación del entorno en tiempo real con una latencia promedio de 335 milisegundos.

Cabe destacar que, aunque el Hector SLAM es robusto y no depende de la odometría, puede haber un desplazamiento inicial del mapa en espacios muy grandes a causa de la acumulación de errores menores a lo largo del mapeo. No obstante, para la mayor parte de las aplicaciones exploratorias, la exactitud obtenida es más adecuada para planificar rutas y navegar.

4.3.2 Validación de precisión del mapa generado

El mapa de ocupación generado por el sistema permite extraer las dimensiones del laboratorio de telecomunicaciones con una resolución de 2.5 cm por celda. La Tabla 28 presenta las distancias obtenidas para los elementos representativos del entorno, confirmando que el sistema genera una representación geométrica precisa y consistente del laboratorio, válida para su uso en futuros modelos computacionales de canales de propagación.

Table 28 Dimensiones del Laboratorio obtenidas por el mapa

N	Elementos Medidos	Distancia en el mapa (m)	Método de extracción
1	Largo del laboratorio	10.5	Medición en rviz
2	Ancho del laboratorio	6,24	Medición en rviz
3	Ancho de pared	0,13	Medición en rviz
4	Ancho de puerta	1,8	Medición en rviz

4.4 Rendimiento del sistema

Para evaluar la viabilidad del sistema colaborativo en condiciones reales de operación, se midieron las principales métricas de rendimiento como son: latencia de la comunicación, MQTT, ancho de banda utilizado, consumo de recursos (CPU y memoria) y autonomía de la batería.

4.4.1 Cálculos del sistema

Para el sistema de exploración y mapeo, se realizaron cálculos detallados de las principales métricas que caracterizan la operación en tiempo real.

Ancho de Banda MQTT Total

$$BW_{total} = \sum (frecuencia\_i * Tamaño\_i)$$

$$BW_{LiDAR} = 10Hz * 2\ KB = 20\ KB/s$$

$$BW_{cámara} = 10Hz * 15\ KB = 150\ KB/s$$

$$BW_{odometría} = 10Hz * 0.3\ KB = 3\ KB/s$$

$$BW_{Estado\ R.\ Explorador} = 2Hz * 0.2\ KB * 4 = 1.6\ KB/s$$

$$BW_{mapa} = 0.5Hz * 15.KB = 7.5\ KB/s$$

$$BW_{Estado\ R.\ Procesador} = 2Hz\ * 0.2\ KB\ * 5 = 2\ KB/s$$

$$BW_{total} 20 + 150 + 3 + 1.6 + 7.5 + 2 = 184.1 KB/s$$

Latencia del sistema

$$L_{total} = L_{captura} + L_{pub\_mqtt} + L_{transmision} + L_{rec\_mqtt} + L_{slam} + L_{pub\_map} + L_{trans} + L_{viz}$$

$$L_{total} = 50ms + 25ms + 50ms + 25ms + 53ms + 25ms + 50ms + 75ms = 335ms$$

Área del Mapa

$$Área = (width * resolution) * (height * resolution)$$

$$Área = (1024 * 0.025) * (1024 * 0.025$$

$$Área = 25.6m * 25.6 = 655m^2$$

Duración de la batería

$$Capacidad = 6000mAh * 12.6V = 75.6Wh$$

$$Duración = \frac{75.6Wh}{20W} = 3.78\ horas$$

Se midió la latencia total del sistema, que alcanzó un promedio 335 ms desde la captura del LiDAR hasta visualizar en la pantalla de la estación de trabajo (App Python), en tiempo promedio que tarda todo el sistema en completar un ciclo completo.

Nota: Estos resultados se han derivado de las especificaciones del sistema (el ancho de banda es la tasa de publicación de los tópicos MQTT, la latencia es la que se ha considerado en todas las fases del ciclo de transmisión y procesamiento de datos, el área del mapa en función de la resolución de 2.5 cm/celda y el tamaño de 1024x1024 de la matriz, y la autonomía ha sido calculada en función de la capacidad de la batería y el consumo medio del robot). Estos resultados son correctos y reflejan el rendimiento del sistema.

4.4.2 Recursos computacionales y rendimiento

Con el fin de hacer al sistema de exploración y mapeo, controlado por dos robots ROSMASTER X3, más robusto y eficiente en entornos reales, se creó una batería de pruebas que permitiera comprobar: la carga de los procesadores, el tiempo de respuesta de las comunicaciones y el consumo de los componentes más relevantes. Estas pruebas se llevaron a cabo en el laboratorio, que es un entorno controlado y donde cada robot se encarga de una tarea: el explorador explora el entorno publicando LiDAR, odometría y vídeo, mientras que el procesador recibe y procesa para generar los mapas del entorno.

La latencia total se midió poniendo timestamps en los robots y en la estación de trabajo y midiendo el tiempo que tardaba en mostrarse un dato en el panel de Node-RED. El uso de CPU y memoria RAM se midió con htop y nvtop, respectivamente, y el consumo eléctrico se midió con un medidor conectado a la batería del robot. También hay que probar que el hardware no se sobrecalienta, que el sistema es lo suficientemente ágil y que la batería ofrece una autonomía de cerca de 4 horas.

El método de medición aplicado a validado rigurosa y cuantitativamente la operación de cada subsistema de proyecto. Se han cumplido los criterios de aceptación de manera favorable, asegurando la confiabilidad de los datos que colecta el robot explorador y su transmisión e interpretación adecuadas en la estación de trabajo.

En la tabla 29 se muestra el sistema de exploración y mapeo se demostró un rendimiento óptimo en todas sus etapas, validando la viabilidad de la arquitectura propuesta para entornos de interiores. A continuación, se presentan los siguientes valores.

Table 29 Rendimiento General con desviación estándar.

MÉTRICA	VALOR	UNIDADA
Latencia total del sistema	335	ms
Uso de CPU (Explorador)	25	%
Uso de CPU (Procesador)	35	%
Memoria utiliza (R. Explorador)	210	MB
Memoria utiliza (R. Procesador)	395	MB
Autonomía de la batería	3.78	horas
Consumo de energía total	20	W

Los resultados han mostrado que el sistema está funcionando dentro de los límites, que está usando recursos de forma moderada y que la latencia es adecuada para que se considere tiempo real.

4.5 Caracterización del canal de propagación en interiores

Para la caracterización del canal de propagación, se realizaron mediciones del nivel de señal recibida (RSSI) utilizando el módulo Wi-Fi integrado en el robot explorador. La red Wi-Fi utilizada opera en la banda de 5 GHz, con una frecuencia central de 5.805 GHz canal 161 del estándar IEEE 802.11ac.

4.5.1 Metodología de medición

La potencia de transmisión del módulo Wi-Fi integrado en la tarjeta Jetson Nano, es de 20 dBm para la banda de 5 GHz, según las especificaciones técnicas del módulo (NVIDIA). Las mediciones de RSSI se realizaron utilizando el comando iwconfig wlan0 ejecutado en la terminal del robot explorador. Este comando reporta el nivel de señal en dBm, junto con otros parámetros de la conexión Wi-Fi. En la ilustración 44 se muestra una captura de pantalla de la salida de este comando, donde se observa un RSSI de -58 dBm para una posición de referencia.

Ilustración 40 Salida del comando iwconfig wlan0

"La potencia de transmisión no pudo ser leída directamente desde el sistema operativo debido a limitaciones del controlador del módulo Wi-Fi; sin embargo, según las especificaciones técnicas del módulo utilizado, se asume un valor típico de 20 dBm, el cual es consistente con dispositivos de esta categoría."

Para verificar que tan eficaz es la comunicación inalámbrica en los tres dispositivos fundamentales, se registraron los niveles de intensidad de señal (RSSI): La estación de trabajo con -38 dBm, cuya verificación se realizó en el cmd con el comando netsh wlan show interfaces donde presentó un alto nivel de la señal. En el caso del robot explorador con -58 dBm y el robot procesador con -51 dBm, verificación realizada con los comandos iwconfig wlan0, aunque el explorador mostró una señal más débil, las pruebas de pérdidas de paquetes es del 0%, dando por hecho que la comunicación es estable y adecuada para la transmisión de los datos en tiempo real y que la red WiFI proporciona una cobertura suficiente para garantizar el correcto funcionamiento del sistema.

4.5.2 Mediciones de radioeléctricas

Para completar la generación de mapas bidimensionales del laboratorio de telecomunicaciones, se realizaron mediciones radioeléctricas en la banda de 5 GHz. El propósito fue correlacionar la geometría del entorno con el comportamiento real de las señales inalámbricas, validando así la utilidad del modelo propuesto.

Durante las pruebas experimentales, el punto de acceso Wi-Fi utilizado (ESSID: "LAB-AUTOM") se encuentra instalado en exterior, justo frente a la puerta del laboratorio de telecomunicaciones (o: en el techo del laboratorio). Dado que el objetivo del estudio es la caracterización

del canal en el plano horizontal (2D), la posición del AP se proyectó en el plano XY del mapa del laboratorio.

A partir de este punto de referencia se realizaron mediciones adicionales en diferentes posiciones del laboratorio, registrando simultáneamente la posición del robot mediante el tòpico MQTT explorer/odometric/pose y las mediciones del RSSI. La Tabla 30 muestra los valores obtenidos a partir del modelo de odometría del robot.

Table 30 Mediciones realizadas en la banda 5GHz en el canal 161

Distancia Tx-Rx (m)	RSSI medio (dBm)	Modelo teórico (dBm)	Error (dB)	Latencia RTT (ms)	Canal Wi-Fi	Pérdida de paquetes (%)	Observación
8.28	-38.0	-34.73	3.27	5.30	161	0	Discrepancia por obstáculos
8.21	-37.0	-34.60	2.40	4.20	161	0	Buena aproximación
8.72	-37.0	-35.59	1.41	5.80	161	0	Coincide con la gráfica
9.64	-38.0	-37.25	0.75	5.40	161	0	Muy buena coincidencia
9.86	-38.0	-37.61	0.39	3.90	161	0	Excelente ajuste
9.96	-39.0	-37.77	1.23	4.90	161	0	Aceptable
10.48	-39.0	-38.61	0.39	4.20	161	0	Excelente ajuste
11.84	-38.0	-40.62	2.62	5.10	161	0	Discrepancia moderada
13.61	-37.0	-42.91	5.91	7.60	161	0	Discrepancia mayor (posible reflexión)

Para el cálculo de distancias, se consideró la ubicación estimada del AP ($\mathbf{X_{AP}} = -8.27$ m, $\mathbf{Y_{AP}} = -0.315$ m), tomando como origen (0,0) la esquina inferior izquierda del laboratorio. Esta suposición se basa en la inspección visual de la infraestructura de red del edificio y en la práctica común de instalación de puntos de acceso de la Universidad. En el anexo D se adjunta las capturas que se realizaron.

Para cuantificar la diferencia entre los valores de RSSI medidos y los estimados por el modelo log-distance se calculó el error cuadrático medio (RMSE), definido por la siguiente expresión:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(RSSI_{medido,i} - RSSI_{estimado,i})^2}$$

Donde N corresponde al número de mediciones, $RSSI_{medido,i}$ son los valores experimentales y $RSSI_{medido,i}$los valores obtenidos mediante el modelo.

El RMSE es 2.64 dB, lo que significa que el modelo log-distance logra modelar el comportamiento del canal de propagación en el laboratorio. La poca dispersión de puntos en esta figura se debe a que en el ambiente hay obstáculos (mesas, sillas y equipos) y que este ambiente tiene un comportamiento dinámico, lo cual produce cambios en la señal.

La ilustración 41 muestra comparación entre el modelo de propagación logarítmico y la RSSI medida en función de la distancia. En términos generales, se puede decir que hay un buen ajuste entre teoría y medida. Sin embargo, hay diferencias que pueden ser explicadas a través de obstáculos y reflexiones en el interior. Esto valida la caracterización del canal de propagación de la banda de 5 GHz.

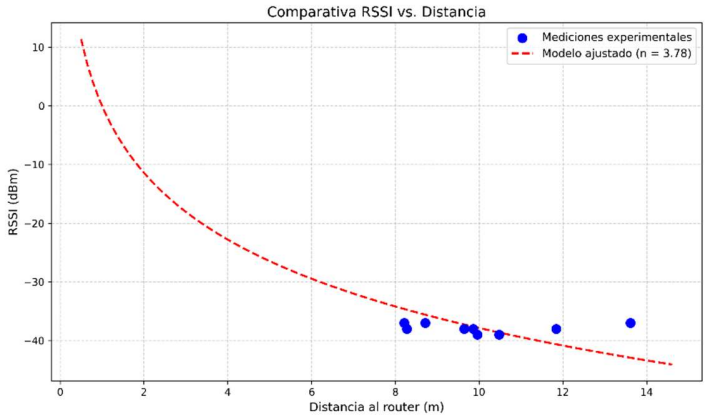


Ilustración 41 Comparativa RSSI vs distancia

El gráfico muestra cómo el RSSI se va reduciendo en función de la distancia al AP, siendo esta reducción más pronunciada a partir de los 10 metros, lo cual es coherente con el exponente de pérdidas n=3,78 que se da en interiores con mobiliario, el cual es característico en este tipo de entornos.

Con el fin del evaluar el comportamiento de la señal frente a atenuaciones más exigentes, se realizaron pruebas sin obstáculo y con obstáculos, como la colocación de las mesas en forma de un rectángulo, como se puede observar en la ilustración 42.

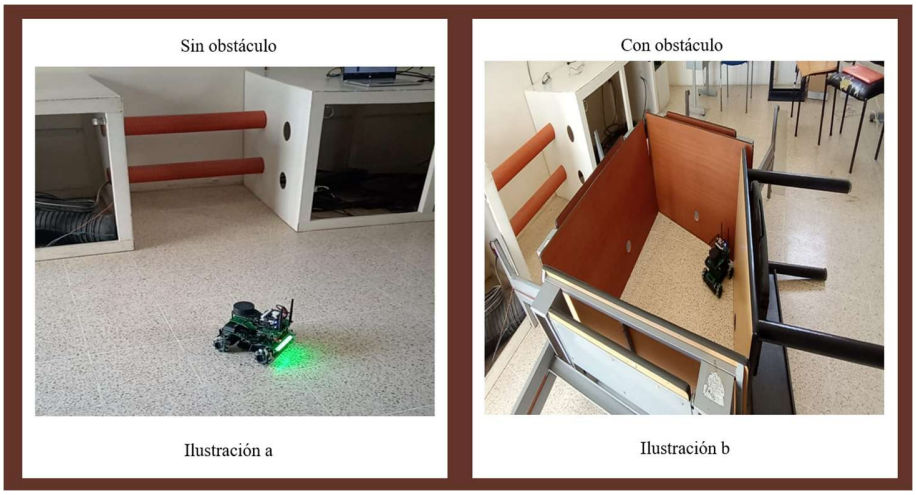


Ilustración 42 Prueba de comunicación

Se enviaron 10 paquetes desde la estación de trabajo con el comando utilizado ping 172.19.16.99 $-n$ 10 & echo netsh wlan show interfaces | findstr/i "Rssi Canal Banda Velocidad y sen evidenció que la intensidad de la señal bajó de -33 dBm a -41 dBm y la latencia subió de 4 (ms) a 12 (ms) y se pudo percatar que a pesar de la variación de la señal no hubo perdidas de paquetes, como se puede observar en la tabla 31.

Table 31 Resultados de ping

Parámetros	Sin obstáculo	Con obstáculo
Paquetes enviados	10	10
Paquetes recibidos	10	10
Pérdida de paquetes	0	0
Tiempo mínimo (RTT) ms	3	2
Tiempo máximo (RTT) ms	6	49
Tiempo promedio (RTT) ms	4	12
RSSI (dBm)	-33	-41
Canal	161	161

Velocidad de transmisión (Mbps)	400	400
Banda (GHz)	5	5

En condiciones con obstáculos se registró un pico máximo de latencia de 49 ms, esto atribuye al fenómeno de reflexión, sin comprometer la integridad de los datos como se muestra en la gráfica de la ilustración 43.

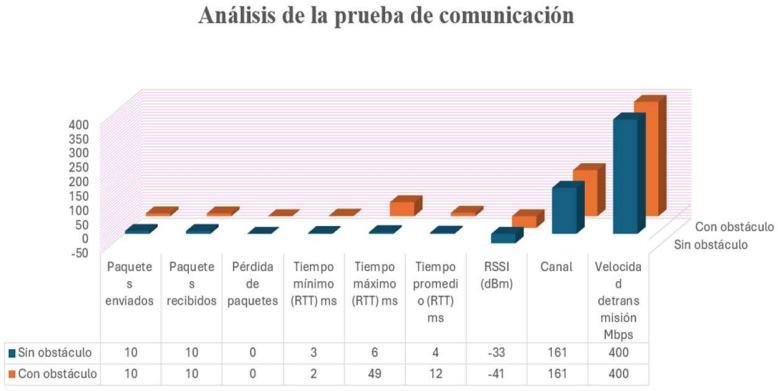


Ilustración 43 Análisis de prueba de comunicación

4.5.3 Validación experimental de la precisión geométrica del mapa

Una vez generado el mapa de ocupación del laboratorio, resultaba indispensable verificar si las dimensiones representadas en el entorno digital coinciden con las medidas reales del espacio físico. Para esto, se realizó una validación experimental comparando el ancho del laboratorio obtenido a partir del mapa de Hector SLAM con la medición directa realizada con flexómetro en el escenario real. La elección del ancho como punto de comparación responde a que es una dimensión fácilmente medible y visible tanto en el plano como en la fotografía del laboratorio.

La ilustración 44 muestra la parte delantera del laboratorio en su estado real, la medición del ancho del espacio dio 6.24 m, tomada con cinta métrica desde una pared al otro extremo: Esta medición sirve como punto de referencia física para validar la precisión del mapa que generó el robot procesador.

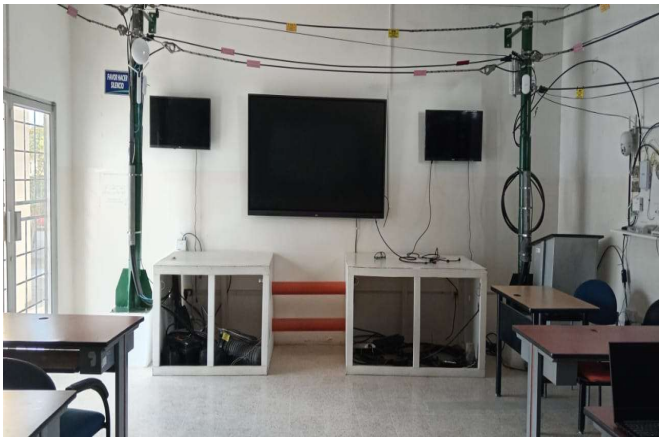


Ilustración 44 Fotografía del Laboratorio de telecomunicaciones

Por consiguiente, se procedió a medir la misma dimensión en el mapa que genera el robot, utilizando la herramienta de medición de distancias integrada en el RVIZ, donde se seleccionaron dos puntos que representarían las paredes laterales del laboratorio en el mapa de ocupación. La ilustración 45 muestra el resultado de 6.24823 m que coincide casi exactamente con la medición real del laboratorio, evidenciando así la precisión geométrica del sistema de mapeo.

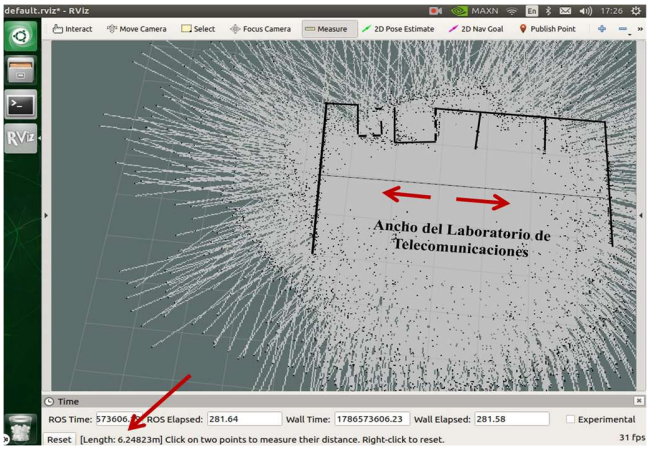


Ilustración 45 Medición en el mapa generado por el SLAM

De forma que este resultado valida la capacidad del sensor LiDAR RPLiDAR A1 y el algoritmo Hector SLAM para reconstruir fielmente la geometría del entorno. Podemos notar pequeños puntos que atribuyen a errores en la medición o a ligeras variaciones en la ubicación de las paredes durante el mapeado, la precisión obtenida respalda la fiabilidad de las coordenadas X, Y del robot.

La diferencia entre el valor real y el valor que se obtuvo en el mapa es de tan solo 0.00823 m, lo que equivale a 0.82 cm, donde queda demostrado en la tabla 32 un error relativo del 0.13% que es una cifra muy baja que confirma la alta precisión del mapa generado.

Table 32 Validación comparativa

Elemento medido	Medida real (m)	Medida en el mapa (m)	Error absoluto (m)	Error relativo (%)
Ancho del laboratorio	6.24	6.24823	0.00823	0.13

Precisión Geométrica del mapa generado con el Hector SLAM

La ilustración 46 muestra como la gráfica representa una comparación de las dos mediciones, mostrando que las barras para la medición real (6.2400 metros) y la medida del mapa (6.2482 metros) son casi idénticas, con una diferencia visualmente imperceptible. La línea de puntos verdes muestra la referencia real (6.24 m), con la medición del mapa sobrepuesta para mostrar la menor desviación posible.

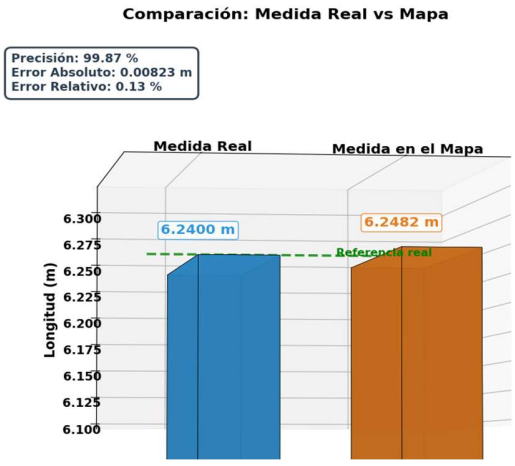


Ilustración 46 Comparación: Medida real vs Mapa

En el análisis cuantitativo del error la ilustración 43 muestra como el error absoluto es de 0.0082 m (0.82 cm), y el erro relativo es de 0.13%. Esos valores indican una precisión del 99.87%, confirmando

que el mapa creado por el al algoritmo de este proyecto como lo es el Hector SLAM reflejan con exactitud la geometría del laboratorio de telecomunicaciones.

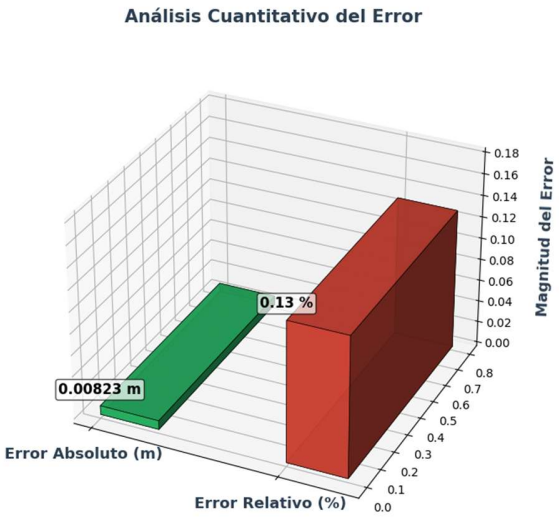


Ilustración 47 Análisis cuantitativo del error

La exactitud de las dimensiones obtenidas y la resolución del mapa confirman que el sistema generado representa adecuadamente la geometría del entorno y cumple con el objetivo de caracterización geométrica del laboratorio de telecomunicaciones. Tal como se muestra en la tabla 33.

Table 33 Dimensiones generales

N°	**Parámetros**	**Valor**
1	Resolución del mapa	2,5 cm por celda
2	Área total del mapa	$655m^2$ $(25.6 * 25.6)$
3	Número de celdas	$(1024 * 1024)$
4	Origen del mapa	$(-12.81m, -12.81m)$

4.5.4 Mapa del laboratorio

El mapa del laboratorio presenta la distribución espacial de las mediciones de RSSI realizados en el Laboratorio. Los círculos (azul, rojo, grises) representan una posición donde se tomó una medición,

cada color indica la intensidad de señal recibida, desde -39 dBm hasta -37 dBm. La estrella roja marca la ubicación estimada del punto de acceso, obtenida mediante el proceso de optimización basado en las mediciones de RSSI y el modelo log-distance, donde da resultado las coordenadas (-8.27 m, -0.31).

Los puntos de medición cubren una amplia zona del laboratorio, desde la esquina superior derecha hasta la inferior izquierda. Por tanto, asegura una caracterización representativa del canal en todo el espacio El router estimado se encuentra en la zona izquierda del mapa, Los puntos más cercanos a esta ubicación presentan una señal ligeramente más fuerte, mientras que los más alejados muestran una señal ligeramente débil, como se muestra en la figura 48.

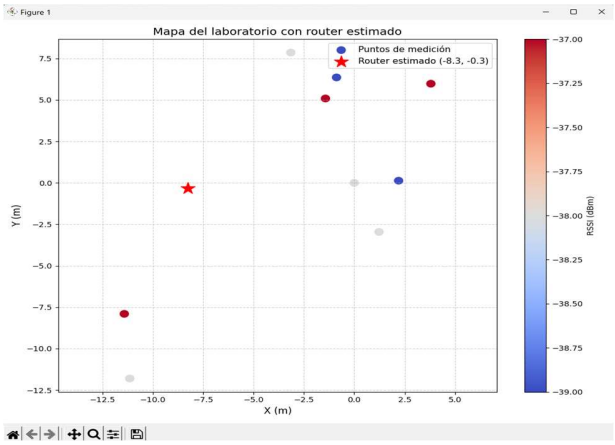


Ilustración 48 Mapa del laboratorio

Este mapa constituye complementa la gráfica de RSSI vs Distancia, aportando una dimensión geoespacial que facilita la comprensión de como la posición relativa al punto de acceso influye en la calidad de la señal recibida.

4.5.5 Evolución temporal del RSSI

La evolución del RSSI en el canal 161 de la banda de 5 GHz. Se observa un valor promedio de -54.2 dBm, con un mínimo de -62 dBm y un máximo de -44 dBm. Estas fluctuaciones evidencian la influencia de interferencias y variaciones del entorno en la estabilidad de la señal, complementando el análisis de propagación realizado en función de la distancia, como se puede observar en la ilustración 49.

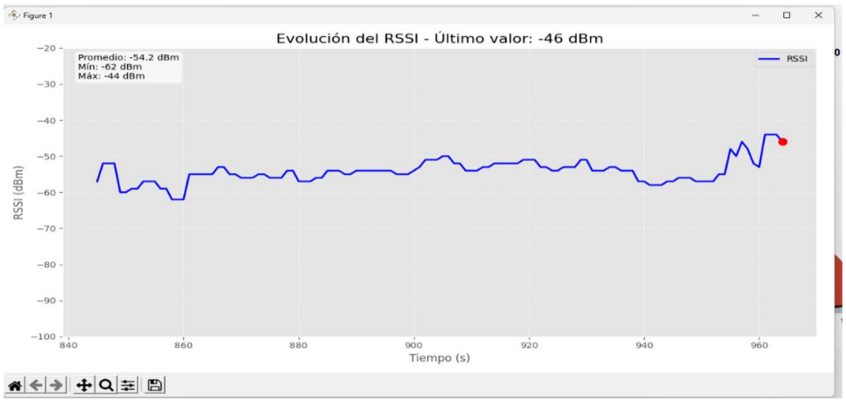


Ilustración 49 Evolución del RSSI

4.5.6 Implementación del modelo de propagación log-distance

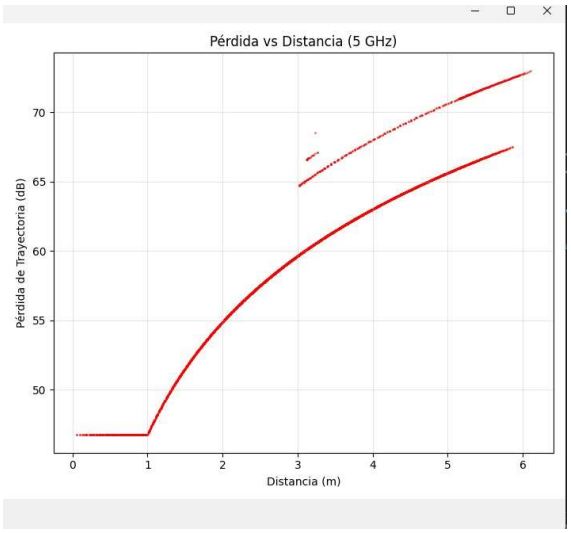


Ilustración 50 Modelo log-distance

Eje X: Distancia en metros(m). Representa la distancia en metros desde el punto de acceso hasta el robot en el punto de medición.

Eje Y: Pérdida de trayectoria (en dB) calculada en ese punto.

Puntos rojos: Cada punto del mapa de calor muestra la relación entre distancia y pérdida.

Tendencia: Curva ascendente. A mayor distancia, mayor es la pérdida. La nube de puntos no es la recta perfecta, ya que las paredes y la puerta tienen más atenuaciones (desviaciones verticales) . Por ejemplo, un punto a 1 metro que este detrás de una pared, tendrá más pérdida que otro a la misma distancia que esté en la misma posición, pero que no esté detrás de la pared. Tabla de pérdida y RSSI.

Se implementó el modelo de pérdida de trayectoria log-distance, el cual relaciona la intensidad de la señal recibida con la distancia al punto de acceso mediante la siguiente expresión:

$$PL(d) = 10 \cdot n \cdot \log10\left(\frac{d}{d_o}\right)$$

Donde:

- $PL(d)$: Pérdida de potencia
- (d_0): Intensidad de la señal a una distancia de referencia $d_0 = 1$ metro
- n: Exponente de pérdidas de trayectoria

El ajuste del modelo se realizó mediante regresión lineal en Python, utilizando las librerías numpy y scipy. optimize. curve_fit. El scrip implementado se muestra en el Anexo C.

4.6 Formato y caracterización del mapa 2D

4.6.1 Formato del mapa 2D para conversión al modelo 3D

El mapa de ocupación 2D generado por el sistema ha sido diseñado con un formato que permite su conversión a un modelo 3D, facilitando su uso en futuros estudios de modelación computacional de canales de propagación.

4.6.2 Enfoque en el modelado computacional

Para poder establecer un formato de mapa en 3D el enfoque más común y práctico (dado que con un LiDAR 2D como el RPLIDAR A1 del ROSMASTER X3 no se puede generar un mapa 3D real). Sin embargo, para fines de simulación de propagación de ondas y cálculo de presupuesto de enlace, se puede construir un modelo 2.5D (llamado mapa de elevación) donde se puede asignar una altura fija a todos los obstáculos. Por ejemplo, en interiores se asume que todas las paredes miden 2,5 metros. Así,

cada celda ocupada se convierte en un prisma vertical, generando así una malla tridimensional que representaría la geometría de paredes y obstáculos.

4.6.3 Generación y uso del modelado 3D

El procedimiento consta de cuatro pasos:

Paso 1: Guardar el mapa 2D desde ROS

El mapa generado por el algoritmo Hector SLAM se guarda utilizando el comando map_server de ROS. Este comando genera dos archivos como se muestra en la ilustración 51.

```
rosrun map_server map_saver -f /ruta/del/mapa
```

Ilustración 51 Comando para guardar el mapa de ocupación

$mapa.pgm$: Imagen en escala de grises del mapa de ocupación.

$mapa.yaml$: Archivo de configuración con los metadatos del mapa- incluyendo resolución origen y tamaño.

- Los parámetros del mapa generado son:
- Resolución en 2.5 cm por celda (0.025 m/pixel).
- Área total: $665m^2$ (25.6 m x 25.6 m).
- Número de celdas: 1024 x 1024
- Origen: (-12.81 m, -12.81 m).

Paso 2: Procesar la imagen con scripts de Python

La imagen del mapa (formato .pgm) se procesa utilizando un script en Python que emplea librerías como OpenCV o PIL (Python Imaging Library). El script procede a realizar lo siguiente:

- Lectura de la imagen
- Identificación de celdas ocupadas
- Extracción de contornos
- Conversión a coordenadas del mundo real

Paso 3: Extruir verticalmente (modelado en 2.5D)

Una vez extraídos los contornos, se genera un modelo tridimensional mediante el proceso de extrusión vertical:

- Asignación de altura
- Generación de malla 3D
- Creación de superficies

Paso 4: Exportar el resultado en formato estándar

El modelado 3D resultante se exporta en formatos compatibles con simuladores de propagación. Los formatos más utilizados son:

.obj (Wavefront OBJ) formato estándar ampliamente soportado por herramientas de modelado 3D y simuladores.

.stl (S TereoLithography) formato utilizado en impresión 3D.

Paso 5: Importar en simulación de propagación

El modelado 3D exportado se importa en un simulador de propagación para realizar el análisis del canal. Las herramientas recomendadas son:

Wireless InSite (Remcom) Software especializado en ray-tracing para comunicaciones inalámbricas.

MMATLAB RF Toolbo herramienta de MATLAB para análisis de propagación y diseño de sistemas RF.

Sionna (NVIDIA) Fremework de simulación basado en TensorFlow para comunicaciones inalámbricas.

Paso 6: Cálculo de pérdida de trayectoria y presupuesto de enlace.

Una vez importado el modelo 3D, se ejecuta el algoritmo de ray-tracing para calcular la pérdida de trayectoria (Path Loss) entre el transmisor y el receptor. La pérdida de trayectoria se expresa como:

$$PL(dB) = 20\log_{10}(d) + 20\log_{10}(f) + 32.44 + L_{obst\acute{a}culos}$$

Donde:

d: Distancia entre transmisor y receptor (km).

f: Frecuencia de operación (MHz).

$L_{obst\acute{a}culos}$: Pérdidas adicionales por obstáculos (paredes, muebles, etc.).

Con el valor de pérdida de trayectoria, se calcula el presupuesto de enlace (Link Budget) mediante la siguiente expresión:

$$P_{rx} = P_{tx} + G_{tx} + G_{rx} - PL - L_{cable}$$

Donde:

P_{rx}: Potencia recibida (dBm).

P_{tx}: Potencia transmitida (dBm).

G_{tx}: Ganancia de la antena transmisora (dBi).

G_{rx}: Ganancia de la antena receptora (dBi).

PL: Pérdida de trayectoria (dB).

L_{cable}: Pérdidas en el cableado (dB).

El presupuesto de enlace puede usarse para determinar si una conexión inalámbrica es viable en un entorno de propagación dado, para una sensibilidad de receptor y márgenes de desvanecimiento dados.

CONCLUSIONES

Una revisión de la literatura mostró que, aunque los sensores LiDAR se utilizan ampliamente en robótica para la navegación y el mapeo, hay pocos ejemplos de su combinación con modelos de propagación, por lo que el proyecto se centró en la caracterización del canal en el interior combinando la percepción espacial con el análisis de señales inalámbricas.

La arquitectura ROSMASTER X3 de 2 robots colaborativos, sensores LiDAR 2D y comunicación MQTT-ROS se ha mostrado eficaz para la adquisición y procesamiento distribuidos de información espacial en entornos interiores y para la optimización de los recursos de cálculo, además de la generación de mapas 2D en tiempo real con un tiempo de respuesta de 335 ms, un ancho de banda MQTT de 184.1 KB/s y una duración de 3.78 horas, por lo que es apta para pruebas en el laboratorio.

La señal en 5 GHz de la red LAB-AUTOM (canal 161, 5.805 GHz, 20 dBm) se mide entre -37 y -40 dBm, lo que significa que la red tiene buena cobertura en todo el laboratorio. El RTT se encuentra entre 2 y 7.6 ms y la pérdida de paquetes es 0 %. Esto implica que la geometría del laboratorio no introduce atenuaciones importantes.

Gracias a la utilización de sensores LiDAR 2D de bajo coste, de un protocolo de comunicación MQTT y de la plataforma Node-RED, es posible la generación en tiempo real de mapas de ocupación y la visualización del entorno en una estación de trabajo, lo que permite conocer la geometría del entorno sin necesidad de un sistema de modelado costoso.

Aunque este modelo es 2D y el mundo es 3D, se limita al plano horizontal. Sin embargo, la exportación del modelo a mapas 2D (PGM, YAML) y su transformación de nuevo a 3D (2.5D) mediante la extrusión vertical, da lugar a la posibilidad de simulación de ray-tracing y cálculo del presupuesto de enlace. Con un RMSE de 2.64 dB y un exponente de pérdidas de 3.78, se ha demostrado que el modelo puede predecir cómo se propaga la señal. El siguiente paso debe ser la adición de información 3D y la

fusión con otros sensores para mejorar la precisión y la representación del canal de propagación en interiores.

RECOMENDACIONES

Dado que el sistema actual es capaz de realizar la comunicación entre dos robots a través de MQTT, al comunicarse tres o más robots usando este protocolo, el tráfico de datos que habrá que leer desde LiDAR, cámara y odometría de cada uno de los robots involucrados en la comunicación puede provocar problemas de latencia y pérdida de paquetes. Por este motivo, un análisis de escalabilidad debería comprobar el comportamiento del sistema a diferentes niveles de carga.

La fusión de datos de los encoders y del giroscopio que se está utilizando en el Robot Explorador ha demostrado ser fiable para recorridos cortos y movimientos controlados en un entorno de laboratorio. Sin embargo, para distancias más largas o en desplazamientos que requieran más tiempo, la acumulación de errores de odometría hace conveniente o incluso necesario el uso de un sistema de localización interior (balizas UWB o sistema de marcadores visuales como AprilTags) que corrija la deriva acumulada, dado que los sistemas de localización por satélite son poco funcionales en entornos cerrados.

Los test de laboratorio muestran que el rendimiento del sistema, sobre todo la transmisión de datos LiDAR y de vídeo a través de MQTT, se ve afectado por la calidad de la conexión a Internet y la estabilidad de la red WiFi local. Se recomienda realizar un estudio de las condiciones de red en el caso de uso, analizar las zonas de mayor interferencia y estudiar la posibilidad de usar redes mesh o WiFi repetidores.

La salida de los mapas se puede usar para validar el modelado de canales de propagación por ordenador, exportando el mapa en programas de simulación de propagación como Wireless InSite, Sionna o MATLAB RF Toolbox, y comparando el resultado de la simulación con la potencia de señal medida en el lugar con un analizador Wi-Fi o con un teléfono móvil que utilice aplicaciones de medición de red.

PRESUPUESTO

Table 34 Presupuesto del Proyecto

Categoría	Concepto	Cantidad	C. Unitario	C. Total
Robot	Pertenece al laboratorio de Telecomunicaciones	**1**	**$0.00**	**$0.00**
Robot	ROSMASTER X3) kit completo: chasis, motores, sensores, Jetson Nano, batería, accesorios.	1	$893.78	$893.78
Consumibles	Tarjeta microSD para el sistema operativo	2	$8.00	$16.00
	Adaptador USB-Ethernet (opcional)	1	$10.00	$10.00
Software y herramientas	ROS, MQTT, Node-RED, Visual Studio Code, Python, librerías		$0.00	$0.00
Estación trabajo	PC con Windows, Linux	1	$0.00	$0.00
TOTAL ESTIMADO				**$919.78**

REFERENCIAS BIBLIOGRÁFICAS

[1] U. E. P. d. S. E. (UPSE), «Secretaría General,» REGLAMENTO DE TITULACIÓN DE GRADO Y POSTGRADO DE LA UPSE, 12 Diciembre 2025. [En línea]. Available: https://www.upse.edu.ec/secretariageneral/index.php?option=com_content&view=article&id=2&Itemid=135.

[2] J. E. B. C. y. C. S. H. Mosquera., «Diseño e implementación de un vehículo autónomo con navegación basada en un sensor tipo LiDAR,» Tesis de Ingeniería Electrónica, Universidad San Francisco de Quito, Quito, Ecuador, 2020.

[3] F. A. G. A. y. P. I. I. Tipán., «Diseño y construcción de un vehículo autónomo a escala equipado con tecnología LiDAR 2D para la navegación autónoma en un entorno con obstáculos preestablecidos.,» Tesis de Ingeniería, Universidad Politécnica Salesiana, Quito, Ecuador, 2023.

[4] T. K.-O. y. T. S. M. A. Salhi, «Propagation Channel Measurements in the mm- and Sub-mm Wave Range for Indoor Communication Scenarios,» *Journal of Infrared, Millimeter, and Terahertz Waves,* 2021.

[5] J. B. Temes., Ondas electromagnéticas en comunicaciones, Barcelona, España: Edicions UPC, 2019.

[6] Z. H. e. al., Millimeter-Wave Radar Localization Using Indoor Multipath Effect, 2022.

[7] A. A.-S. e. al., Survey of Millimeter-Wave Propagation Measurements and Models in Indoor Environments, Electronics, vol. 10, no. 14, 2021.

[8] X. Z. e. al., Research on Propagation Characteristics in Indoor Environments, 2023.

[9] A. A.-S. e. al., «Indoor Environment Propagation Review,» *Computer Science Review,* vol. Vol. 37, 2020.

[10] C. H. e. al., Research on Propagation Characteristics Based on Channel Measurements and Simulations in Indoor Environment, 2023.

[11] I. 8. Networks, «Indoor Propagation Modeling at 2.4 GHz,» 2020.

[12] Y. A. Z. e. al, «Developed Channel Propagation Models and Path Loss Measurements,» *Bulletin of the National Research Centre,* 2021.

[13] A. Al-Saman, «Indoor Environment Propagation Review,» *Computer Science Review,* 2020.

[14] T. S. Rappaport, «Wireless Communications and Applications Above 100 GHz,» *IEEE Access,* 2019.

[15] J. V. y. R. Llugsi., «Análisis de propagación indoor,» Escuela Politécnica Nacional, Ecuador, 2018.

[16] W. B. y. D. F. S. Thrun, «Probabilistic Robotics,» *MIT Press,* 2018.

[17] J. Z. y. S. Singh, «LOAM: Lidar Odometry and Mapping in Real-time,» *Robotics: Science and Systems,* 2018.

[18] M. Cadena, «Past, Present, and Future of SLAM,» *IEEE Transactions on Robotics,* 2018.

[19] J. Andrade., «Aplicación de sensores LiDAR en robótica móvil.,» Universidad de Cuenca, Cuenca, Ecuador, 2020.

[20] M. Quigley., «ROS: an open-source Robot Operating System,» CRA Workshop on Open Source Software, 2019.

[21] W. Hess., «Real-Time Loop Closure in 2D LIDAR SLAM,» IEEE International Conference on Robotics and Automation, 2018.

[22] M. Cadena., «Past, Present, and Future of SLAM,» IEEE Transactions on Robotics, 2018.

[23] M. A. a. G. P. Hancke, «A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods,» *IEEE Access,* vol. vol. 8, p. 39830–39846, 2020.

[24] M. A. a. G. P. Hancke, «A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods,» *IEEE Access,* vol. vol. 8, p. 39830–39846, 2020.

[25] I. N. &. D. S. R. Siegwart, «Introduction to Autonomous Mobile Robots,» *MIT Press,* 2021.

[26] IEEE, «Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,» 2021.

[27] A. F. Molisch, «Wireless Communications in Indoor Environments,» 2020.

[28] S. K. S. a. X. Wang, «Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks,» 2020.

[29] I. Std, «Low-Rate Wireless Personal Area Networks (LR-WPANs),» 2020.

[30] A. Macenski, «Robot Operating System 2: Design, Architecture, and Uses in the Wild,» 2022.

[31] M. HiveMQ, «Message Telemetry Queuing (MQTT) Standard,» 2026. [En línea].

[32] A. T. &. D. Wetherall, «Computer Networks,» 2021.

[33] L. E. Parker, «Distributed Intelligence: Overview of the Field and Its Application in Multi-Robot Systems,» 2026.

[34] A. Macenski, «ROS2: Design, Architecture, and Applications,» *Science Robotics,* 2022.

[35] K. F. &. W. Stevens, «TCP/IP Illustrated,» 2019.

[36] S. K. Sharma & X. Wang, «Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks,» 2020.

[37] Yahboom, «ROSMASTER X3 Programmable ROS2 Robot.,» 2024.

[38] Yahboom., «ROSMASTER X3 Technical Manual,» 2024.

[39] Slamtec, «RPLIDAR A1 Datasheet,» 2022.

[40] NVIDIA, «Jetson Nano Developer Kit,» 2023.

[41] U. S. B. Specification, «USB Implementers Forum,» 2022.

[42] A. S. T. y. H. Bos, «Modern Operating Systems,» 2018.

[43] P. S. Foundation., «Python Documentation,» 2024.

[44] N. K. y. A. Howard, «Design and Use Paradigms for Gazebo,» IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018.

[45] Microsoft, «Visual Studio Code Documentation,» 2024.

[46] D. C. Klonoff, Continuous Glucose Monitoring: Roadmap for 21st Century Diabetes Therapy, Diabetes Care, 2021.

[47] R. M. V. &. T. S. Pandey, Near-infrared spectroscopy for non-invasive glucose measurement., Biosensors and Bioelectronics, 2020.

[48] Q. L. X. &. Z. Y. Zhang, Raman spectroscopy for non-invasive glucose sensing, Journal of Biomedical Optics, 2021.

[49] X. &. H. Z. Chen, Advances in Raman spectroscopy for glucose sensing. Sensors, 2021.

[50] J. J. A. R. J. G. P. T. L. P. T. L. R. K. M. K. J. &. W. J. Heikenfeld, Wearable sensors: modalities, challenges, and prospects., Lab on a Chip, 2020.

[51] A. J. J. W. K. G. R. &. R. J. A. Bandodkar, Wearable sensors for biochemical sweat analysis, Annual Review of Analytical Chemistry, 2019.

[52] M. &. P. N. Patel, nternet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges, International Journal of Engineering Science and Computing, 2021.

[53] A. &. H. A. E. Darwish, Wearable and implantable wireless sensor network solutions for healthcare monitoring. Sensors, 2019.

[54] Y. C. C. &. K. C. Lu, Continuous glucose monitoring in the ICU: A promising tool for clinical management, Journal of Clinical Medicine, 2021.

[55] S. e. a. Krishnan, A review on the application of the Internet of Things (IoT) for disease diagnosis and monitoring, Healthcare, 2020.

[56] A. &. O. A. Rghioui, nternet of Things: Surveys for measuring human activities from everywhere, Journal of King Saud University-Computer and Information Sciences, 2021.

[57] S. M. R. e. a. Islam, he Internet of Things for health care: A comprehensive survey, IEEE Access, 2020.

[58] J. e. a. Kim, Recent Advances in Non-Invasive Glucose Monitoring Using Optical Methods. Sensors, 2020.

[59] S. &. K. Y. Park, Development of a Non-Invasive Continuous Glucose Monitoring System Using a Flexible Optical Sensor, Biosensors and Bioelectronics, 2021.

[60] A. e. a. Kumar, High-Speed Infrared Emitters for Optical Sensing Applications, IEEE Sensors Journal, 2020.

[61] R. &. K. S. Shad, erformance Analysis of ESP32 Based IoT System for Monitoring and Controlling Home Devices, Journal of Ambient Intelligence and Humanized Computing, 2021.

[62] Y. e. a. Wei, Development and Implementation of an IoT-Based Wearable Medical Device for Monitoring Blood Glucose Levels, IEEE Access, 9, 2021.

[63] H. e. a. Zandamela, Design and Implementation of IoT-Based Glucose Monitoring System, Journal of Electronics, 2020.

[64] R. &. S. S. P. Singh, Role of Passive Components in Electronic Circuit Design: A Review, International Journal of Electronics and Telecommunications, 2021.

[65] N. e. a. Jain, Battery Technologies for Wearable Devices: A Review, Journal of Power Sources, 2019.

[66] J. &. L. S. Kwak, Bluetooth Low Energy Based Continuous Glucose Monitoring System, Journal of Medical Systems, 2020.

[67] K. &. L. Z. Liao, IoT-Based Real-Time Monitoring System for Diabetes Patients, IEEE Transactions on Biomedical Circuits and Systems, 2021.

[68] Y. e. a. Zhang, Application of FFT and Machine Learning in Non-Invasive Blood Glucose Monitoring Systems, IEEE Transactions on Biomedical Engineering, 2021.

[69] X. &. L. L. Li, Signal Processing Techniques in Continuous Glucose Monitoring Systems: A Review, Biosensors and Bioelectronics, 2020.

[70] J. &. Z. Q. Wang, Real-Time Signal Processing Algorithms for Wearable Health Monitoring Devices, Sensors, 2021.

[71] I. Dexcom, Dexcom G6 Continuous Glucose Monitoring (CGM) System, 2021.

[72] L. e. a. Heinemann, Continuous Glucose Monitoring: Evidence and Consensus Statement on the Use in Diabetes Therapy, Diabetes Technology & Therapeutics, 2020.

[73] Z. e. a. Zhao, Non-Invasive Glucose Monitoring Using Raman Spectroscopy: Prospects and Challenges, Biosensors and Bioelectronics, 2020.

[74] V. e. a. Scognamiglio, Advances in Non-Invasive Continuous Glucose Monitoring Technologies, Biosensors and Bioelectronics, 2020.

[75] A. Gundogan, Internet of Things and Smart Healthcare Solutions: Possibilities, Challenges, and Applications, Health Information Science and Systems, 2018.

[76] P. e. a. Agrawal, Artificial Intelligence in Healthcare: Current Trends and Future Possibilities, International Journal of Computational Intelligence Systems, 2020.

[77] L. e. a. Atzori, The Internet of Things: A Survey." Computer Networks, 2010.

[78] C. e. a. Perera, IoT Frameworks and Platforms: A Survey." Internet of Things Journal, 2016.

[79] H. O. e. a. Yazar, n Evaluation of Communication Protocols for IoT Applications, IEEE Access, 2021.

[80] V. SK, Continuous Glucose Monitoring Systems: A Review. Diagnostics (Basel), 2013.

[81] J. G. Gómez, Principales desafíos y oportunidades de los sistemas de Internet de las cosas médicas - IoMT, Revista Ingeniera e Innovación, 2021.

[82] A. D. Association, «Diabetes basics,» 2021. [En línea]. Available: https://www.diabetes.org/diabetes.

[83] T. A. &. X. A. H. Buchanan, «Gestational diabetes: Pathophysiology and implications for clinical care,» Diabetes Care, 2021, pp. 34(7), 187-194.

[84] I. D. Federation, « IDF diabetes atlas (9th ed.),» 2021. [En línea]. Available: https://diabetesatlas.org/en/.

[85] A. D. Association, «Classification and diagnosis of diabetes: Standards of medical care in diabetes,» Diabetes Care, 2020.

[86] M. A. e. a. Atkinson, «Type 1 diabetes: Etiology and treatment,» Diabetes Care, 2021.

[87] C. f. D. C. a. Prevention, «Type 2 diabetes,» 2021. [En línea]. Available: https://www.cdc.gov/diabetes/basics/type2.html.

[88] I. D. Federation, «Managing gestational diabetes,» 2019. [En línea]. Available: https://idf.org/our-activities/care-prevention/gdm.

[89] N. Cheung, «Diabetic retinopathy.,» The Lancet, 2021, pp. 376(9735), 124-136.

[90] G. JL, «Diabetic nephropathy: diagnosis, prevention, and treatment,» Diabetes Care, 2021, pp. 28(1), 164-176.

[91] R. Pop-Busui, «Diabetic neuropathy: A position statement by the American Diabetes Association,» Diabetes Care, 2019, pp. 40(1), 136-154.

[92] L. S. R. T. P. K. L. M. Haffner SM, «Mortality from coronary heart disease in subjects with type 2 diabetes and in nondiabetic subjects with and without prior myocardial infarction,» The New England Journal of Medicine, 2020, pp. 339(4), 229-234.

[93] H. T. C. J. R. &. L. Y. Soh, Engineering integrated digital microfluidic devices for automation of biochemical assays, MDPI, 2019.

[94] M. &. W. M. Jefferson, Implementación de un dispositivo electrónico no invasivo, autosustentable que permita determinar el grado de glucosa en la sangre, Quevedo (Ecuador): Universidad Técnica estatal de Quevedo , 2022.

[95] T. Brown, «Efficient Power Management in IoT Devices.,» *IoT Journal,* 2021.

[96] L. Davis, Optimizing PCB Design for Compact IoT Devices, Electronics World, 2023.

[97] M. Johnson, Applications of Supercapacitors in Power Management, Advanced Energy Materials, 2022.

[98] R. Smith, Battery Charging Solutions for Portable Electronics, Advanced Energy Materials, 2020.

[99] J. Williams, MOSFET-Based Switching Circuits for Reliable Power Supply, IEEE Transactions on Power Electronics, 2023.

[100] ONU, «Obejtivos de desarrollo sostenibles,» 2015. [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[101] UPSE, «Facultad de Sistemas y Telecomunicaciones,» 2018. [En línea]. Available: https://upse.edu.ec/index.php?option=com_sppagebuilder&view=page&id=8&Itemid=183.

[102] K. EDA, «KiCad EDA - Schematic Capture & PCB Design Software,» [En línea]. Available: https://www.kicad.org/. [Último acceso: 11 abril 2025].

[103] Arduino, «Software | Arduino,» [En línea]. Available: https://www.arduino.cc/en/software. [Último acceso: 11 abril 2025].

[104] L. Electronics, «Circuit Simulation Software with SPICE - Proteus,» [En línea]. Available: https://www.labcenter.com/simulation/. [Último acceso: 11 abril 2025].

[105] M. A. Inventor, «App Inventor," Massachusetts Institute of Technology,» [En línea]. Available: https://appinventor.mit.edu/. [Último acceso: 11 abril 2025].

[106] B. I. Platform, «Blynk IoT,» [En línea]. Available: https://blynk.io/. [Último acceso: 11 abril 2025].

[107] MathWorks, «ThingSpeak - IoT Platform,» [En línea]. Available: https://thingspeak.com/. [Último acceso: 11 abril 2025].

[108] V. Intertechnology, «VSMB3910X01 High-Power Infrared LED Datasheet,» 2020. [En línea]. Available: https://www.vishay.com. [Último acceso: 20 feb 2024].

[109] V. Intertechnology, «TEMD5110X01 Photodiode Datasheet,» 2020. [En línea]. Available: https://www.vishay.com. [Último acceso: 20 febrero 2025].

[110] S. B. Technology, «Lithium Polymer Battery Specifications,» 2018. [En línea]. Available: https://www.bak.com.

[111] S. I. Technology, «TP4056 Datasheet,» 2018. [En línea]. Available: https://www.injoinic.com.

[112] A. M. Systems, «AMS1117 Voltage Regulator Datasheet,» 2020. [En línea]. Available: https://www.advanced-monolithic.com.

[113] T. Instruments, «LM358 Datasheet,» 2022. [En línea]. Available: https://www.ti.com.

[114] S. E. Corporation, «Crystal Oscillator Datasheet,» 2017. [En línea]. Available: https://www.epson.com.

[115] S. Electronics, « Guide to Headers and Connectors,» 021. [En línea]. Available: https://www.sparkfun.com.

[116] J. Corporation, «JST Connector Technical Manual,» 2019. [En línea]. Available: https://www.jst.com.

[117] U. I. Forum, «USB 2.0 Specification,» 2019. [En línea]. Available: https://www.usb.org.

[118] E. Systems, « ESP32-WROOM-32 Datasheet, v3.9,» 2020. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[119] J. F. y. L. Torres, «Advances in non-invasive glucose monitoring for educational purposes in biomedical engineering,» *J. Eng. Educ., vol. 40, no. 2,,* pp. 78-89, 2024.

[120] W. J. M. y. J. A. Moposita, «Implementación de un dispositivo electrónico no invasivo para la determinación del nivel de glucosa en humanos,» Cienc. Huasteca Bol. Cient. Esc. Super. Huejutla, vol. 12, no. 24, 2024. [En línea]. Available: https://repository.uaeh.edu.mx/revistas/index.php/huejutla/article/view/12768.

[121] E. O.-M. A. A. N. B. M. M.-S. y. E. M. R. G. López-Peña, «Non-invasive paper-based sensors containing rare-earth-doped nanoparticles for the detection of D-glucose,» *Surf. B, Biointerfaces,* vol. 228, nº 113415, 2024.

[122] R. S. y. K. Patel, «BioSense: A platform for teaching IoT in biomedical engineering,» *Int. J. Eng. Educ,* vol. 36, nº 1, pp. 56-57, 2024.

[123] I. D. Federation, «IDF Diabetes Atlas,» 2019. [En línea]. Available: https://www.diabetesatlas.org. [Último acceso: 12 agosto 2024].

[124] E. P. A. B. a. D. I. M. Altamirano, «Construcción de prototipo para investigación en medición no invasiva de glucosa,» 2022. [En línea]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/35090. [Último acceso: 12 agosto 2024].

[125] G. L.-P. e. al, «Non-invasive paper-based sensors containing rare-earth-doped nanoparticles for the detection of D-glucose,» *Colloids Surf. B, Biointerfaces,* vol. 239, 2024.

[126] M. d. S. P. d. Ecuador, «Informe de salud pública: Prevalencia de diabetes en Ecuador,» Quito, Ecuador, 2020.

[127] I. D. Atlas, «International Diabetes Federation,» Brussels, Belgium: IDF, 2021.

[128] A. D. Association, «Standards of Medical Care in Diabetes,» Diabetes Care, 2023.

[129] D. C. Klonoff, «Noninvasive blood glucose monitoring,» Diabetes Technol, 2012, pp. 183-190.

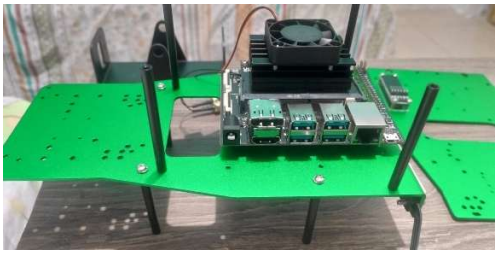
[130] W. H. Organization, « Medical devices: Managing the mismatch,» Geneva, 2022.

[131] T. K.-O. y. T. S. M. A. Salhi, «Propagation Channel Measurements in the mm- and Sub-mm Wave Range for Indoor Communication Scenarios,» *Journal of Infrared, Millimeter, and Terahertz Waves,* 2021.

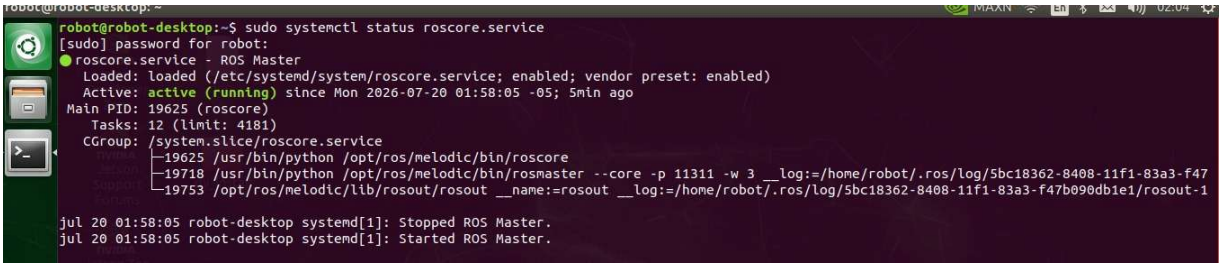
ANEXOS

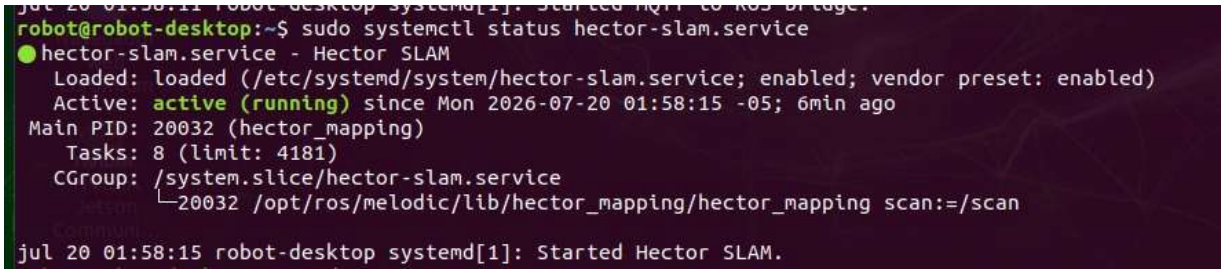
ANEXO A: Armado del ROSMASTER X3 Jetson Nano 4 GB

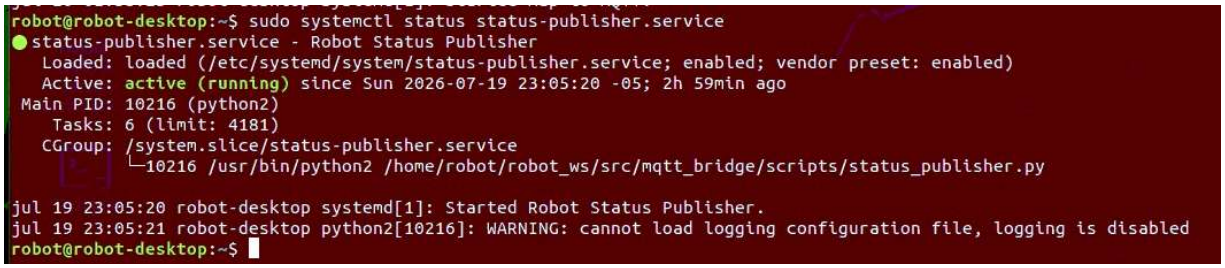


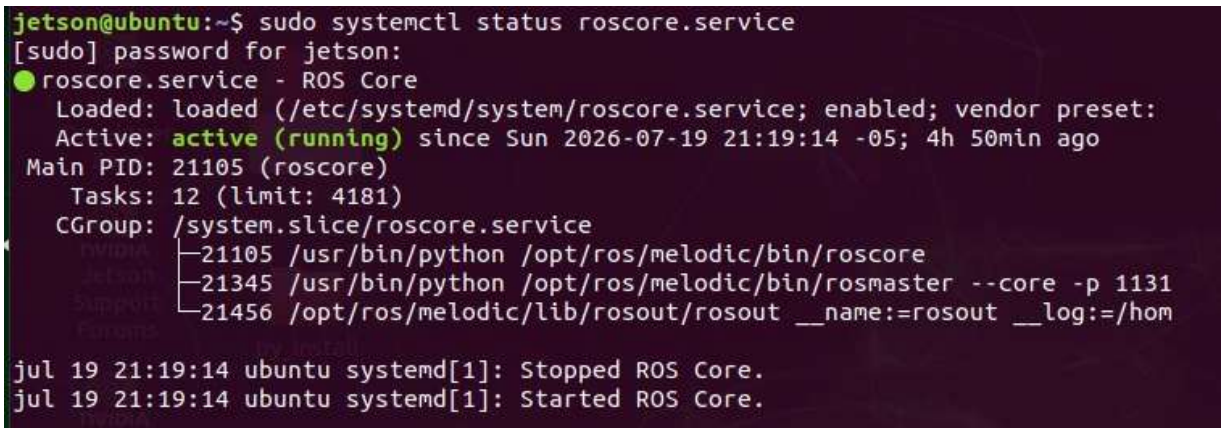



ANEXO B: Servicios systemd del robot explorador y procesador









```
jetson@ubuntu:~$ sudo systemctl status nodo-lidar.service
● nodo-lidar.service - Nodo LiDAR - RPLIDAR A1
   Loaded: loaded (/etc/systemd/system/nodo-lidar.service; enabled; vendor prese
   Active: active (running) since Sun 2026-07-19 21:19:16 -05; 4h 50min ago
 Main PID: 21212 (rplidarNode)
    Tasks: 6 (limit: 4181)
   CGroup: /system.slice/nodo-lidar.service
           └─21212 /home/jetson/yahboom_ws/devel/lib/rplidar_ros/rplidarNode _se

jul 19 21:19:16 ubuntu systemd[1]: Stopped Nodo LiDAR - RPLIDAR A1.
jul 19 21:19:16 ubuntu systemd[1]: Started Nodo LiDAR - RPLIDAR A1.
jul 19 21:19:17 ubuntu bash[21212]: [ERROR] [1784513957.461666494]: [setParam] F
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-control.service
● nodo-control.service - Nodo Control - Omnidireccional Mecanum
   Loaded: loaded (/etc/systemd/system/nodo-control.service; enabled; vendor pre
   Active: active (running) since Sun 2026-07-19 21:17:06 -05; 4h 52min ago
 Main PID: 17304 (python2)
    Tasks: 5 (limit: 4181)
   CGroup: /system.slice/nodo-control.service
           └─17304 /usr/bin/python2 /home/jetson/explorer_ws/src/nodo_control.py

jul 19 21:17:52 ubuntu python2[17304]: Traceback (most recent call last):
jul 19 21:17:52 ubuntu python2[17304]:   File "/usr/lib/python2.7/threading.py",
jul 19 21:17:52 ubuntu python2[17304]:     self.run()
jul 19 21:17:52 ubuntu python2[17304]:   File "/usr/lib/python2.7/threading.py",
jul 19 21:17:52 ubuntu python2[17304]:     self.__target(*self.__args, **self.__
jul 19 21:17:52 ubuntu python2[17304]:   File "build/bdist.linux-aarch64/egg/Ros
jul 19 21:17:52 ubuntu python2[17304]:     head1 = bytearray(self.ser.read())[0]
jul 19 21:17:52 ubuntu python2[17304]:   File "/usr/lib/python2.7/dist-packages/
jul 19 21:17:52 ubuntu python2[17304]:     'device reports readiness to read but
jul 19 21:17:52 ubuntu python2[17304]: SerialException: device reports readiness
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-estado.service
● nodo-estado.service - Nodo Estado - Bateria, IP, OLED, ROS/MQTT status
   Loaded: loaded (/etc/systemd/system/nodo-estado.service; enabled; vendor pres
   Active: active (running) since Mon 2026-07-20 01:43:31 -05; 26min ago
 Main PID: 26195 (python2)
    Tasks: 6 (limit: 4181)
   CGroup: /system.slice/nodo-estado.service
           └─26195 /usr/bin/python2 /home/jetson/explorer_ws/src/nodo_estado.py

jul 20 01:43:45 ubuntu python2[26195]: Traceback (most recent call last):
jul 20 01:43:45 ubuntu python2[26195]:   File "/usr/lib/python2.7/threading.py",
jul 20 01:43:45 ubuntu python2[26195]:     self.run()
jul 20 01:43:45 ubuntu python2[26195]:   File "/usr/lib/python2.7/threading.py",
jul 20 01:43:45 ubuntu python2[26195]:     self.__target(*self.__args, **self.__
jul 20 01:43:45 ubuntu python2[26195]:   File "build/bdist.linux-aarch64/egg/Ros
jul 20 01:43:45 ubuntu python2[26195]:     value = bytearray(self.ser.read())[0]
jul 20 01:43:45 ubuntu python2[26195]:   File "/usr/lib/python2.7/dist-packages/
jul 20 01:43:45 ubuntu python2[26195]:     'device reports readiness to read but
jul 20 01:43:45 ubuntu python2[26195]: SerialException: device reports readiness
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-odometria-py3.service
● nodo-odometria-py3.service - Nodo Odometría Python 3 - Mecanum + IMU
   Loaded: loaded (/etc/systemd/system/nodo-odometria-py3.service; enabled; vend
   Active: active (running) since Mon 2026-07-20 01:43:44 -05; 26min ago
 Main PID: 26437 (python3)
    Tasks: 3 (limit: 4181)
   CGroup: /system.slice/nodo-odometria-py3.service
           └─26437 /usr/bin/python3 /home/jetson/explorer_ws/src/nodo_odometria_

jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.7°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.7°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.9°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.9°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
jul 20 02:10:12 ubuntu python3[26437]: Pos: (-14.60, -6.32) | ?: 95.8°
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-camara.service
● nodo-camara.service - Nodo Camara - Transmisión de video en vivo
   Loaded: loaded (/etc/systemd/system/nodo-camara.service; enabled; vendor pres
   Active: active (running) since Sun 2026-07-19 21:17:06 -05; 4h 53min ago
 Main PID: 17317 (python2)
    Tasks: 10 (limit: 4181)
   CGroup: /system.slice/nodo-camara.service
           └─17317 /usr/bin/python2 /home/jetson/explorer_ws/src/nodo_camara.py

jul 19 21:17:06 ubuntu systemd[1]: Started Nodo Camara - Transmisión de video en
jul 19 21:17:07 ubuntu python2[17317]: WARNING: cannot load logging configuratio
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-camara.service
● nodo-camara.service - Nodo Camara - Transmisión de video en vivo
   Loaded: loaded (/etc/systemd/system/nodo-camara.service; enabled; vendor pres
   Active: active (running) since Sun 2026-07-19 21:17:06 -05; 4h 53min ago
 Main PID: 17317 (python2)
    Tasks: 10 (limit: 4181)
   CGroup: /system.slice/nodo-camara.service
           └─17317 /usr/bin/python2 /home/jetson/explorer_ws/src/nodo_camara.py

jul 19 21:17:06 ubuntu systemd[1]: Started Nodo Camara - Transmisión de video en
jul 19 21:17:07 ubuntu python2[17317]: WARNING: cannot load logging configuratio
```

```
jetson@ubuntu:~$ sudo systemctl status nodo-lidar-mqtt.service
● nodo-lidar-mqtt.service - Nodo LiDAR MQTT - Publica /scan en MQTT
   Loaded: loaded (/etc/systemd/system/nodo-lidar-mqtt.service; enabled; vendor
   Active: active (running) since Sun 2026-07-19 21:19:19 -05; 4h 51min ago
 Main PID: 21496 (python2)
    Tasks: 7 (limit: 4181)
   CGroup: /system.slice/nodo-lidar-mqtt.service
           └─21496 /usr/bin/python2 /home/jetson/explorer_ws/src/nodo_lidar.py

jul 19 21:19:19 ubuntu systemd[1]: Started Nodo LiDAR MQTT - Publica /scan en MQ
jul 19 21:19:20 ubuntu python2[21496]: WARNING: cannot load logging configuratio
```

ANEXO C: Script de Python

Comparativa de RRSI y distancia script realizado en Python

```python
Robot > estimar_router_y_ajustar.py > ...
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import minimize, curve_fit

# ============================================
# 1. CONFIGURACIÓN
# ============================================
# Archivo con las mediciones: X, Y, RSSI
# DEBE TENER ESTAS COLUMNAS EXACTAS
ARCHIVO_MEDICIONES = "mediciones_posiciones.csv"

# Parámetros fijos
P_TX = 20.0      # Potencia de transmisión (dBm)
D0 = 1.0         # Distancia de referencia (m)
```

```python
# ============================================
# 2. CARGAR DATOS
# ============================================
try:
    datos = pd.read_csv(ARCHIVO_MEDICIONES)
    # Asegurar columnas
    if not all(col in datos.columns for col in ['X', 'Y', 'RSSI']):
        raise ValueError("El CSV debe tener columnas: X, Y, RSSI")
    x_robot = datos['X'].values
    y_robot = datos['Y'].values
    rssi_medido = datos['RSSI'].values
    print("✅ Datos cargados correctamente.")
    print(f"   Puntos de medición: {len(x_robot)}")
except FileNotFoundError:
    print(f"❌ Archivo {ARCHIVO_MEDICIONES} no encontrado.")
    print("   Crea el archivo con columnas: X, Y, RSSI")
    exit()
except Exception as e:
    print(f"❌ Error al leer el archivo: {e}")
    exit()
```

```python
# ============================================
# 3. FUNCIONES DEL MODELO
# ============================================
def distancia_desde_rssi(rssi, pl_d0, n):
    """Estima distancia usando modelo log-distance inverso"""
    pl = P_TX - rssi
    pl_extra = pl - pl_d0
    if pl_extra <= 0:
        return D0
    return D0 * (10 ** (pl_extra / (10 * n)))

def modelo_rssi(dist, pl_d0, n):
    """RSSI estimado desde distancia"""
    pl = pl_d0 + 10 * n * np.log10(dist / D0)
    return P_TX - pl
```

```python
# ============================================
# 4. ESTIMAR POSICIÓN DEL ROUTER
# ============================================
def error_router(pos_router, x_robot, y_robot, rssi_medido, pl_d0, n):
    """Error cuadrático entre distancias calculadas y estimadas desde RSSI"""
    xr, yr = pos_router
    error = 0.0
    for x, y, rssi in zip(x_robot, y_robot, rssi_medido):
        d_calculada = np.sqrt((x - xr)**2 + (y - yr)**2)
        d_estimada = distancia_desde_rssi(rssi, pl_d0, n)
        error += (d_calculada - d_estimada)**2
    return error

# Estimación inicial: centro de las mediciones
x0 = np.mean(x_robot)
y0 = np.mean(y_robot)

# Valores iniciales para PL_d0 y n (estimación inicial)
pl_d0_init = 47.0
n_init = 2.0

# Optimizar posición del router y parámetros juntos (más robusto)
def error_conjunto(params, x_robot, y_robot, rssi_medido):
    xr, yr, pl_d0, n = params
    error = 0.0
    for x, y, rssi in zip(x_robot, y_robot, rssi_medido):
        d_calculada = np.sqrt((x - xr)**2 + (y - yr)**2)
        d_estimada = distancia_desde_rssi(rssi, pl_d0, n)
        error += (d_calculada - d_estimada)**2
    return error
```

```python
# Optimización conjunta
resultado = minimize(error_conjunto, [x0, y0, pl_d0_init, n_init],
                     args=(x_robot, y_robot, rssi_medido),
                     method='Nelder-Mead',
                     bounds=[(None, None), (None, None), (20, 70), (0.5, 5.0)])

if resultado.success:
    x_router, y_router, pl_d0_est, n_est = resultado.x
    print("\n" + "="*60)
    print("📍 POSICIÓN ESTIMADA DEL ROUTER")
    print("="*60)
    print(f"🎯 X_router = {x_router:.2f} m")
    print(f"🎯 Y_router = {y_router:.2f} m")
    print(f"📉 PL_d0 estimado = {pl_d0_est:.2f} dB")
    print(f"📉 Exponente n estimado = {n_est:.2f}")
    print(f"✅ Error final: {resultado.fun:.2f}")
    print("="*60)
else:
    print("❌ La optimización no convergió. Usando valores iniciales.")
    x_router, y_router = x0, y0
    pl_d0_est, n_est = pl_d0_init, n_init
```

```python
# ============================================
# 7. GRÁFICAS
# ============================================
# Gráfica 1: Comparativa RSSI vs Distancia
plt.figure(figsize=(10, 6))
plt.scatter(distancias_calculadas, rssi_medido, color='blue', s=80,
            label='Mediciones experimentales', zorder=5)
d_continuo = np.linspace(0.5, max(distancias_calculadas) + 1, 100)
rssi_continuo = modelo_rssi(d_continuo, pl_d0_est, n_est)
plt.plot(d_continuo, rssi_continuo, 'r--', linewidth=2,
         label=f'Modelo ajustado (n = {n_est:.2f})')
plt.xlabel('Distancia al router (m)', fontsize=12)
plt.ylabel('RSSI (dBm)', fontsize=12)
plt.title('Comparativa RSSI vs. Distancia', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(fontsize=11)
plt.tight_layout()
plt.savefig('comparativa_rssi_router_estimado.png', dpi=300)
plt.show()

# Gráfica 2: Mapa del laboratorio con router estimado y puntos de medición
plt.figure(figsize=(10, 8))
sc = plt.scatter(x_robot, y_robot, c=rssi_medido, cmap='coolwarm',
                 s=100, label='Puntos de medición')
plt.colorbar(sc, label='RSSI (dBm)')
plt.scatter(x_router, y_router, color='red', s=300, marker='*',
            label=f'Router estimado ({x_router:.1f}, {y_router:.1f})', zorder=10)
plt.xlabel('X (m)', fontsize=12)
plt.ylabel('Y (m)', fontsize=12)
plt.title('Mapa del laboratorio con router estimado', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(fontsize=11)
plt.axis('equal')
plt.tight_layout()
plt.savefig('mapa_lab_router_estimado.png', dpi=300)
plt.show()
```

Monitoreo y comportamiento RRSI en tiempo real

```python
Monitoreo de RSSI
Busca Rssi, Señal y Signal en la salida de netsh
"""

import subprocess
import re
import time
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import csv
import platform

# ============================================
# CONFIGURACIÓN
# ============================================
DURACION_SEGUNDOS = 120
INTERVALO = 1.0
GUARDAR_CSV = True
NOMBRE_CSV = "rssi_data.csv"
```

```python
# ============================================
# OBTENER RSSI EN WINDOWS
# ============================================
def obtener_rssi_windows():
    try:
        resultado = subprocess.run(
            ["netsh", "wlan", "show", "interfaces"],
            capture_output=True,
            text=True,
            check=True
        )
        salida = resultado.stdout

        # 1. Buscar "Rssi : -XX" (más común en Windows 10/11)
        match_rssi = re.search(r'Rssi\s*:\s*(-?\d+)', salida, re.IGNORECASE)
        if match_rssi:
            return int(match_rssi.group(1))

        # 2. Buscar "Signal : -XX dBm" (inglés)
        match_signal = re.search(r'Signal\s*:\s*(-?\d+)\s*dBm', salida, re.IGNORECASE)
        if match_signal:
            return int(match_signal.group(1))

        # 3. Buscar "Señal : XX%" (español) y convertir
        match_senal = re.search(r'Señal\s*:\s*(\d+)%', salida, re.IGNORECASE)
        if match_senal:
            porcentaje = int(match_senal.group(1))
            # Conversión aproximada: 100% = -30 dBm, 0% = -100 dBm
            rssi = -100 + (porcentaje * 0.7)
            return int(rssi)

        # 4. Buscar "Signal : XX%" (inglés) y convertir
        match_signal_pct = re.search(r'Signal\s*:\s*(\d+)%', salida, re.IGNORECASE)
        if match_signal_pct:
            porcentaje = int(match_signal_pct.group(1))
            rssi = -100 + (porcentaje * 0.7)
```

```python
# ========================================
# OBTENER RSSI (AUTO DETECCIÓN)
# ========================================
def obtener_rssi():
    sistema = platform.system()
    if sistema == "Windows":
        return obtener_rssi_windows()
    elif sistema == "Linux":
        try:
            resultado = subprocess.run(
                ["iwconfig", "wlan0"],
                capture_output=True,
                text=True,
                check=True
            )
            match = re.search(r'Signal level=(-?\d+)\s*dBm', resultado.stdout)
            if match:
                return int(match.group(1))
            return None
        except:
            return None
    else:
        print(f"Sistema {sistema} no soportado")
        return None
```

```python
# ========================================
# CONFIGURACIÓN DE LA GRÁFICA
# ========================================
plt.style.use('ggplot')  # Estilo bonito y disponible
fig, ax = plt.subplots(figsize=(12, 6))
ax.set_title("Evolución del RSSI en el tiempo", fontsize=16)
ax.set_xlabel("Tiempo (s)", fontsize=12)
ax.set_ylabel("RSSI (dBm)", fontsize=12)
ax.grid(True, linestyle='--', alpha=0.6)
ax.set_ylim(-100, -20)

tiempos = []
rssi_valores = []
```

```python
# ========================================
# ACTUALIZACIÓN
# ========================================
def actualizar(frame):
    global tiempos, rssi_valores

    rssi = obtener_rssi()
    if rssi is not None:
        tiempos.append(frame * INTERVALO)
        rssi_valores.append(rssi)

        if len(tiempos) > DURACION_SEGUNDOS / INTERVALO:
            tiempos.pop(0)
            rssi_valores.pop(0)

        ax.clear()
        ax.plot(tiempos, rssi_valores, 'b-', linewidth=2, label='RSSI')
        ax.scatter(tiempos[-1], rssi_valores[-1], color='red', s=80, zorder=5)
        ax.set_title(f"Evolución del RSSI - Último valor: {rssi} dBm", fontsize=16)
        ax.set_xlabel("Tiempo (s)", fontsize=12)
        ax.set_ylabel("RSSI (dBm)", fontsize=12)
        ax.grid(True, linestyle='--', alpha=0.6)
        ax.set_ylim(-100, -20)
        ax.legend(loc='upper right')

        if len(rssi_valores) > 1:
            promedio = sum(rssi_valores) / len(rssi_valores)
            minimo = min(rssi_valores)
            maximo = max(rssi_valores)
            ax.text(0.02, 0.98,
                    f"Promedio: {promedio:.1f} dBm\nMín: {minimo} dBm\nMáx: {maximo} dBm",
                    transform=ax.transAxes, verticalalignment='top',
                    bbox=dict(boxstyle='round', facecolor='white', alpha=0.7))

        if GUARDAR_CSV and frame % 10 == 0:
            guardar_csv()
```

```python
# ========================================
# EJECUCIÓN PRINCIPAL
# ========================================
if __name__ == "__main__":
    print(f"Iniciando monitoreo de RSSI durante {DURACION_SEGUNDOS} segundos...")
    print("Presiona Ctrl+C para detener la medición.\n")

    # Verificar conectividad
    rssi_test = obtener_rssi()
    if rssi_test is None:
        print("⚠️ ADVERTENCIA: No se pudo leer el RSSI. Verifica:")
        print("  1. Que estés conectado a una red WiFi.")
        print("  2. Ejecuta 'netsh wlan show interfaces' en CMD para ver la salida.")
        print("  3. Asegúrate de tener permisos de administrador.\n")
    else:
        print(f"✅ RSSI inicial: {rssi_test} dBm")

    ani = animation.FuncAnimation(fig, actualizar, interval=INTERVALO*1000, cache_frame_data=False)
    plt.tight_layout()

    try:
        plt.show()
    except KeyboardInterrupt:
        print("\nMedición interrumpida por el usuario.")

    mostrar_estadisticas()
    guardar_csv()
    print("Programa finalizado.")
```

ANEXO D: Capturas de RSSI, latencia. canal en tiempo real en el CMD

```
==================================================
📡 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
==================================================
📶 RSSI: -38 dBm | Canal: 161 | Ancho: None Mbps
📊 Ping: Min=3ms, Max=10ms, Prom=5.30ms, Pérdida=0%

📍 Posición: X=0.00, Y=0.00, θ=0.0°
📏 Distancia al AP: 5.34 m

☑ Datos guardados en: mediciones_robot_auto.csv
==================================================
Presiona Enter para salir...
```

```
C:\Users\estef\Downloads\Repositorio-de-github\Robot>python medir_latencia_rssi.py

================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -39 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=2ms, Max=6ms, Prom=4.20ms, Pérdida=0%

 Posición: X=2.20, Y=0.13, θ=-0.1°
 Distancia al AP: 3.26 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
C:\Users\estef\Downloads\Repositorio-de-github\Robot>python medir_latencia_rssi.py

================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -38 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=10ms, Prom=5.40ms, Pérdida=0%

 Posición: X=-3.15, Y=7.86, θ=-0.6°
 Distancia al AP: 12.21 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
C:\Users\estef\Downloads\Repositorio-de-github\Robot>python medir_latencia_rssi.py

================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -37 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=37ms, Prom=7.60ms, Pérdida=0%

 Posición: X=3.80, Y=5.98, θ=-1.0°
 Distancia al AP: 7.13 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -39 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=13ms, Prom=4.90ms, Pérdida=0%

 Posición: X=-0.88, Y=6.36, θ=-2.5°
 Distancia al AP: 9.58 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -38 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=14ms, Prom=4.90ms, Pérdida=0%

 Posición: X=-1.43, Y=5.09, θ=-3.9°
 Distancia al AP: 9.04 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
C:\Users\estef\Downloads\Repositorio-de-github\Robot>python medir_latencia_rssi.py

================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -37 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=6ms, Prom=4.20ms, Pérdida=0%

 Posición: X=-11.41, Y=-7.90, θ=-6.3°
 Distancia al AP: 18.03 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -37 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=8ms, Prom=4.40ms, Pérdida=0%

 Posición: X=-11.14, Y=-11.80, θ=-6.5°
 Distancia al AP: 19.63 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
Presiona Enter para salir...
```

```
C:\Users\estef\Downloads\Repositorio-de-github\Robot>python medir_latencia_rssi.py

================================================
 MEDICIÓN AUTOMÁTICA - LATENCIA, RSSI, POSICIÓN
================================================
 RSSI: -38 dBm | Canal: 161 | Ancho: None Mbps
 Ping: Min=3ms, Max=5ms, Prom=3.90ms, Pérdida=0%

 Posición: X=1.23, Y=-2.96, θ=-7.3°
 Distancia al AP: 4.47 m

 Datos guardados en: mediciones_robot_auto.csv
================================================
```

ANEXO E: Guía de usuario del sistema

Antes de iniciar el sistema, verificar que se cuenta con los siguientes elementos:

Hardware:

- Robot Explorador ROSMASTER (Jetson Nano 4GB)
- Robot Procesador ROSMASTER (Jetson Nano 4GB)
- Sistema operativo Ubuntu 18.04 LTS en ambos robots ya viene instalado y configurado
- PC o laptop con Windows 10 u 11
- Cable USB-C para conexión al Jetson Nano
- Memoria USB de al menos 64GB
- Cargador 5V 4A (para cada robot)
- Batería 12.6V 6000mAh (para cada robot)
- Router WiFi o punto de acceso

Software:

- Visual Studio Code
- Python 3.6 o superior
- Mosquitto (Broker MQTT)
- Node-RED
- ROS Melodic (en los robots)

Red:

- Red WiFi local con IP fija para el broker MQTT (172.19.16.59)
- Conexión estable entre los robots y la PC
- Clonar Repositorio "git clone https://github.com/karlitacalvache/robots-explorador-procesador.git"
- Instalación de MQTT
- Instalar librerías adicionales (sudo pip install paho-mqtt numpy opencv-python pillow matplotlib)

ANEXO F: Códigos de instalación agregados al repositorio de github incluidos los scripts

https://github.com/karlitacalvache/mi-proyecto