

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE
ESCUELA POLITÉCNICA SUPERIOR DE ELCHE
GRADO EN INGENIERÍA ELECTRÓNICA Y
AUTOMÁTICA INDUSTRIAL



Diseño e implementación de una
arquitectura de control para navegación y
teleoperación del robot móvil
RB-THERON

TRABAJO FIN DE GRADO

Junio - 2025

AUTOR: Jorge Juan Fernández Calero

DIRECTORES: José María Catalán Orts

David Martínez Pascual

“Más personas alcanzan la excelencia por la constancia y la práctica que por sus dotes naturales.”

- Demócrito



AGRADECIMIENTOS

En primer lugar, quiero dar las gracias a mi familia por todo el apoyo que me han dado durante estos cuatro años y por ser mi pilar en los momentos más duros.

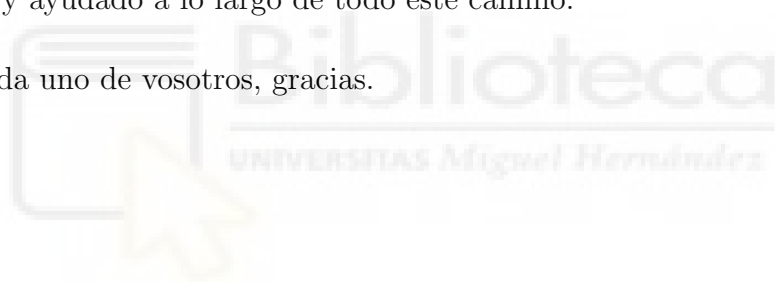
A mi padre, por su trabajo y esfuerzo diario para que cada uno de sus hijos pueda alcanzar sus metas.

A mi hermano, por ser un gran ejemplo para mí sin ni siquiera saberlo. Verle todas las noches acostarse tan tarde para sacar adelante un examen me ha demostrado que, con esfuerzo y dedicación, uno es capaz de lograr cualquier cosa.

Quiero agradecer de manera especial a mi madre por darme la mano durante toda esta etapa. Gracias por no soltarme nunca, por confiar en mí más que nadie y por darme fuerzas de donde parecía no haberlas.

Por último, a mis compañeros de clase, y en especial a Jonny y Sonia, que me han acompañado y ayudado a lo largo de todo este camino.

A todos y cada uno de vosotros, gracias.



RESUMEN

Este proyecto tiene por objetivo la adaptación de la base móvil RB-THERON como sistema de control inteligente de una silla de ruedas, con el fin de proporcionar mayor independencia a personas que padezcan algún tipo de discapacidad motora severa, como la tetraplejia o la ELA.

Para alcanzar este propósito, el presente trabajo se ha centrado en el desarrollo de una arquitectura de control basada en ROS Noetic y en la integración de los diferentes dispositivos que permiten el control de la base móvil RB-THERON. Para el manejo del robot, se ha empleado el protocolo de comunicación UDP (*User Datagram Protocol*), el cual permite la comunicación entre dispositivos que no comparten la misma arquitectura de software.

A su vez, como medida de seguridad, el sistema es capaz de abortar la misión en modo autónomo si el robot no logra planificar una trayectoria viable hacia su objetivo. Por otro lado, en el modo manual, el vehículo se detiene de manera instantánea ante la ausencia de comandos desde el teclado. Finalmente, a través de los resultados obtenidos en los ensayos, se puede concluir que el sistema es robusto y fiable, presentando un error en régimen permanente del 0,39 %, lo que valida la arquitectura propuesta como una solución precisa y segura para entornos de asistencia.

ÍNDICE

AGRADECIMIENTOS	III
RESUMEN EJECUTIVO	V
ÍNDICE DE TABLAS	XI
ÍNDICE DE FIGURAS	XIV
1. Introducción	1
1.1. Historia de la robótica Móvil	2
1.2. Clasificación de los Robots Móviles	4
1.2.1. Robots Aéreos	4
1.2.2. Robots Acuáticos	6
1.2.3. Robots Terrestres	6
1.3. Principales Ámbitos de Aplicación de los Robots Terrestres	8
1.4. Antecedentes	10
1.5. Problemática	12
1.6. Objetivos	13
2. Materiales y Métodos	14
2.1. Plataforma móvil: Robot RB-THERON	14
2.1.1. Hardware y Componentes de seguridad	15
2.1.2. Sistema Sensorial	16
2.1.3. Cinemática y Locomoción	17
2.1.4. Interfaces de Control	17

2.2. Entorno de Desarrollo y Middleware	18
2.2.1. Sistema Operativo (Ubuntu 20.04 LTS)	18
2.2.2. Robot Operating System (ROS)	18
2.2.2.1. Justificación de la elección ROS y la distribución Noetic	19
2.2.2.2. Estructura de ROS	20
2.3. Generación del Entorno de Navegación mediante la interfaz HMI . . .	21
2.3.1. Mapeo 2D del entorno (<i>Gmapping</i>)	22
2.3.2. Localización y obtención de coordenadas (<i>localization_2d</i>) . . .	23
2.4. Arquitectura de control en ROS	24
2.4.1. Diagrama Global del Sistema	25
2.4.1.1. PC del Robot	25
2.4.1.2. PC de Control	27
2.4.2. Nodo Principal	28
2.4.2.1. Entradas y Salidas del Nodo	28
2.4.2.2. Procesos Internos del Programa	29
2.4.2.3. Procesos de Ejecución por Comandos	30
2.4.2.4. Método de Supervisión	36
2.4.2.5. Transmisión de Telemetría	37
2.5. Software de Comprobación	39
2.5.1. Módulo Emisor	39
2.5.2. Módulo Visualizador	40
2.6. Adquisición de Datos y Herramientas de Análisis	42
2.6.1. Registro de datos mediante rosbag	42

2.6.2. Herramienta de Cálculo y Visualización Gráfica (MATLAB)	43
3. Resultados y Discusión	44
3.1. Navegación autónoma	44
3.1.1. Validación de la trayectoria de forma autónoma	44
3.1.2. Comprobación de las restricciones de velocidad	46
3.2. Seguridad en entornos dinámicos	48
3.2.1. Evasión de obstáculos	48
3.2.2. Robustez y repetibilidad del sistema	49
3.3. Validación del modo manual	51
3.3.1. Respuesta del robot frente a la consigna enviadas	51
3.3.2. Análisis del error en régimen permanente	52
4. Conclusiones y Trabajos Futuros	56
4.1. Conclusiones	56
4.2. Trabajos Futuros	57
BIBLIOGRAFÍA	58

ÍNDICE DE TABLAS

3.1. Tiempos de ejecución y cálculo de desviaciones.	50
3.2. Muestra de datos y cálculo del error instantáneo.	54



ÍNDICE DE FIGURAS

1.1. El robot Shakey. Fuente: [9]	3
1.2. Robot y su interacción con el entorno. Fuente: [6, p. 4]	4
1.3. Dron de ala fija.	5
1.4. Dron de ala rotatoria.	5
1.5. Dron de ala híbrida.	6
1.6. Robot EMILY para Salvamento Marítimo. Fuente: [11]	6
1.7. Robot móvil RB-VOGUE. Fuente: [13]	7
1.8. Robot cuadrúpedo Spot. Fuente: [14]	7
1.9. Robot desactivador de explosivos tEODor. Fuente: [15]	8
1.10. Plataforma móvil RB-THERON. Fuente: [17]	9
1.11. Robot agrícola del proyecto BACCHUS. Fuente: [18]	9
1.12. Robot bombero para tareas de rescate. Fuente: [19]	9
1.13. Robot móvil UVD de desinfección sanitaria. Fuente: [21]	10
1.14. El rover Perseverance en la superficie de Marte. Fuente: [22]	10
2.1. Componentes principales RB-THERON. Fuente: [28]	15
2.2. Botonera de control RB-THERON. Fuente: [28]	15
2.3. Controlador Dualshock. Fuente: [28]	16
2.4. Rango de visión de la cámara del RB-THERON. Fuente: [28]	16
2.5. Rango de visión del láser LiDAR del RB-THERON. Fuente: [28]	16
2.6. Ruedas del RB-THERON. Fuente: [28]	17
2.7. Esquemático de relaciones con el ROS Master.	20
2.8. Ejemplo de comunicación mediante un tópico.	21

2.9. Ejemplo de comunicación mediante un servicio.	21
2.10. Mapa 2D generado mediante el algoritmo Gmapping.	22
2.11. Interfaz gráfica HMI para la visualización y obtención de coordenadas espaciales.	23
2.12. Esquema de comunicación inalámbrica entre el PC y el robot.	24
2.13. Esquema general de la arquitectura ROS	25
2.14. Flujo general de programación.	30
2.15. Fragmento de código para el procesamiento del comando UDP y ejecución de la navegación.	31
2.16. Diagrama de flujo del comando 1/destino.	32
2.17. Diagrama de flujo del comando 2/stop.	33
2.18. Diagrama de flujo del comando 3/acción.	34
2.19. Diagrama de flujo del comando 4/manual.	35
2.20. Máquina de estados de la rutina de supervisión y control de navegación.	37
2.21. Cálculo mediante el Teorema de Pitágoras.	38
2.22. Diagrama de flujo del módulo emisor (Sender).	40
2.23. Diagrama de flujo del módulo visualizador (Receiver).	41
2.24. Flujo de adquisición y procesamiento de datos del proyecto.	42
3.1. Mapa del pasillo del Instituto de Bioingeniería con los puntos de interés marcados.	45
3.2. Trayectoria real estimada frente a la ruta planificada.	45
3.3. Evolución temporal de las coordenadas X e Y durante la misión.	46
3.4. Validación de la restricción de velocidad en modo autónomo.	47
3.5. Foto de la disposición de los obstáculos en el pasillo.	48
3.6. Validación de la trayectoria ante obstáculos.	49

3.7. Respuesta del sistema ante una entrada de tipo escalón.	51
3.8. Evolución temporal del error de velocidad.	53



1. Introducción

La disfunción en la actividad motora condiciona la autonomía de las personas, limitando por completo la capacidad de desplazarse por sus propios medios. A pesar de la existencia de numerosos vehículos robóticos, la gran mayoría cuentan con métodos de manejo manual (por ejemplo, joystick), siendo inadecuados para pacientes que presentan una pérdida total de la movilidad [1].

Algunas de las condiciones médicas relevantes para este proyecto pueden ser las siguientes:

1. **Tetraplejia:** pacientes con lesiones cervicales graves, producidas por algún tipo de traumatismo o enfermedades del sistema nervioso, lo que deriva en una pérdida de la capacidad motora y sensitiva a partir del cuello. Esto les impide hacer uso de las sillas eléctricas convencionales [2].
2. **Esclerosis Lateral Amiotrófica (ELA):** se trata de una enfermedad neurodegenerativa donde los síntomas empeoran con el paso del tiempo. Esto conlleva que, en sus fases más avanzadas, el paciente pierda la movilidad completa de su cuerpo aun estando su capacidad cognitiva intacta. Por ello los pacientes necesitan de total asistencia para moverse [3].
3. **Distrofia Muscular de Duchenne (DMD):** este trastorno afecta a 1 de cada 3500 niños a nivel mundial, condicionándolos a llevar silla de ruedas el resto de sus vidas. En las fases más avanzadas de la enfermedad, los pacientes son incapaces de moverse o interactuar con objetos, por lo que un joystick tradicional sería inadecuado para estos usuarios [4].

El presente proyecto trata de aportar una mayor independencia a personas con discapacidad motora. Este Trabajo de Fin de Grado tiene el objetivo de desarrollar un vehículo para personas con movilidad reducida mediante una base móvil industrial RB-THERON (Robotnik Automation).

El pilar fundamental de este proyecto se basa en el diseño e implementación de una arquitectura de control en ROS (*Robot Operating System*), enfocada en la navegación autónoma y teleoperada del robot RB-THERON. Esta arquitectura está encargada de gestionar los diferentes nodos y de procesar la información del entorno obtenida mediante los sensores que incorpora el robot. Gracias a este diseño, permite planificar rutas fiables y seguras, garantizando un correcto transporte de los usuarios.

Para asegurar la viabilidad del proyecto, se han llevado a cabo diversas pruebas experimentales en entornos reales, lo que demuestra que el funcionamiento es el adecuado. Analizando los datos, se ha verificado el correcto funcionamiento de la base móvil.

Este documento se estructura de la siguiente manera:

- Capítulo 2: se describen los materiales y métodos empleados, enfocándose en las características del robot y la arquitectura en ROS utilizada.
- Capítulo 3: expone los resultados obtenidos ante las pruebas experimentales realizadas, demostrando la viabilidad de la solución propuesta.
- Capítulo 4: se expone la conclusión del proyecto y los posibles trabajos futuros.

1.1. Historia de la robótica Móvil

Los robots móviles son equipos capaces de percibir su entorno, planificar rutas y desplazarse por sí mismos [5], gracias a una serie de mecanismos equipados en su interior, como son sensores y módulos de desplazamiento (ruedas, patas, orugas o reptación) [6]. Estos dispositivos se pueden controlar de manera remota o autónoma según su función.

El objetivo de los robots consiste en hacer la vida más fácil al ser humano, evitando que las personas realicen tareas repetitivas, peligrosas o problemáticas [7]. Los primeros robots móviles autónomos fueron desarrollados en 1948 por, W. Grey Walter en Inglaterra bajo el nombre de ELSIE y ELMER, popularmente denominadas “tortugas”. Esto fue un gran avance en el mundo de la robótica, ya que permitió dar a conocer una nueva forma de entender la electrónica de la época [8]. Pocos años después entre 1966 y 1972, Charles Rosen, Nils John Nilsson y Peter E. Hart concibieron un proyecto muy innovador llamado SHAKEY [5]. Fue el primer robot móvil de propósito general, este podía realizar una gran variedad de tareas distintas en entornos cambiantes, lo que dio lugar a que se le considere el “abuelo” de los vehículos autónomos y robots modernos, obligando a sus creadores a desarrollar algoritmos que se siguen usando hoy en día, se puede apreciar una imagen de este en la Figura 1.1.

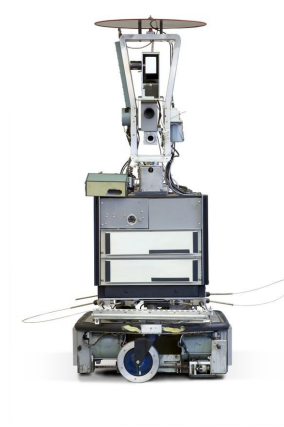


Figura 1.1: El robot Shakey. Fuente: [9]

Aunque el objetivo fundamental de los robots móviles sigue siendo el mismo respecto a sus antecesores históricos, se puede hablar del gran salto tecnológico que hay en la actualidad. Esto ha permitido que sean capaces de planificar trayectorias en función de la idoneidad del momento [6]. Esta capacidad de adaptación aporta gran valor tanto a la industria como en entornos de servicio.

Al margen de la evolución histórica, el valor de la robótica móvil actual reside en la capacidad de operar en escenarios críticos. Esto les diferencia de los robots industriales cuyo objetivo es el de realizar tareas repetitivas en entornos controlados, en cambio los robots móviles se enfrentan a problemas que tienen diversas soluciones, por lo que el margen de error es mayor, como por ejemplo obstáculos imprevistos, personas en movimiento o suelos irregulares [6].

Para controlar esta problemática, todo robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales [7]. Siendo los más importantes el bucle generado por la parte de Sistema Sensorial (Percepción), Sistema de Control (Decisión) y Sistema de Accionamiento (Acción), apreciándose un ejemplo en la Figura 1.2. Todas estas características permiten al robot interactuar con el medio [6].

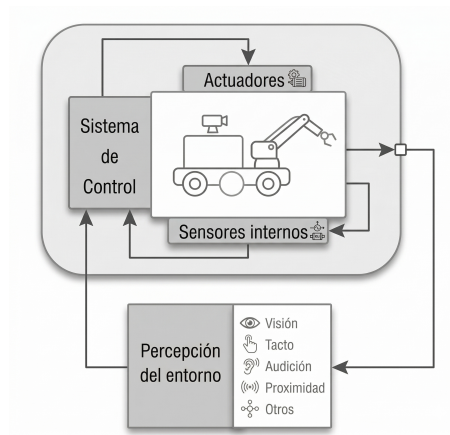


Figura 1.2: Robot y su interacción con el entorno. Fuente: [6, p. 4]

1. **Percepción:** se define como la visión del robot, es la parte en la que se debe captar información del entorno, mediante el uso de sensores externos como pueden ser: cámaras, láseres, ultrasonidos, encoders. Esto permite situarse en el medio físico.
2. **Decisión:** es considerado el cerebro del robot, el procesado de datos debe ser lo suficientemente rápido para anticiparse a situaciones de peligro, su objetivo es el de generar las órdenes necesarias para seguir la trayectoria correcta.
3. **Acción:** representa las extremidades del robot móvil, es la parte encargada de ejecutar las decisiones que tienen repercusión en el exterior. Estas acciones son efectuadas mediante los actuadores o mecanismos de movimiento, como son los motores, brazos robóticos o las ruedas.

1.2. Clasificación de los Robots Móviles

La robótica móvil actual es conocida por estar presente en todos los medios posibles (terrestre, aéreo y acuático). Según su ámbito de aplicación, la morfología del robot debe adaptarse para cumplir con las necesidades del entorno [6].

1.2.1. Robots Aéreos

Se trata de aeronaves pilotadas de manera remota o autónoma, generalmente conocidos como drones o UAV (*Unmanned Aerial Vehicle*). Estos dispositivos están equipados con sensores de alta precisión, fundamentales para la estabilización y control de vuelo [10].

Se pueden clasificar según el tipo de ala que presenten (ala fija, ala rotatoria o híbridos).

- **Ala fija (*Fixed-wings*)**: los drones de ala fija son los menos comunes, debido a la necesidad de infraestructuras de despegue y aterrizaje. Su estructura, como se puede apreciar en el ejemplo de la Figura 1.3, es muy similar a la de los aviones convencionales, por lo que presentan una serie de ventajas a diferencia de los de ala rotatoria. Son capaces de alcanzar mayores velocidades gracias a la sustentación aerodinámica, además presentan mayor autonomía, lo que permite volar grandes distancias sin necesidad de recargar la batería, siendo ideales para tareas de vigilancia o mapeo del terreno.



Figura 1.3: Dron de ala fija.

- **Ala rotatoria (*Rotary-wing*)**: este tipo de drones es el más utilizado tanto en el ámbito industrial como en el comercial, se ha popularizado en estos últimos años, permitiendo realizar tareas en alturas elevadas. Estos robots, como se puede apreciar en la Figura 1.4, están equipados con varios sistemas de rotores (hélices) que permiten su sustentación en el aire. A diferencia de los de ala fija, poseen gran maniobrabilidad y capacidad de vuelo estacionario, haciendo posible el despegue y aterrizaje en espacios reducidos.



Figura 1.4: Dron de ala rotatoria.

- **Ala híbrida (*Hybrid*)**: en general cada tipo de robot aéreo tiene una serie de desventajas (los de ala fija necesitan mucho espacio para maniobrar y los de ala rotatoria no son capaces de recorrer grandes distancias), por lo que para contrarrestar estas problemáticas, se optó por simplificarlo en un único

vehículo, desarrollando una configuración híbrida entre los drones de ala fija y los de ala rotatoria, tal como se muestra en la Figura 1.5.



Figura 1.5: Dron de ala híbrida.

1.2.2. Robots Acuáticos

Están destinados a sumergirse a gran profundidad para obtener muestras de micro-organismos o realizar estudios del terreno. Suelen estar equipados con cámaras de alta resolución, o brazos articulados para la manipulación del entorno. Pueden ser controlados de manera autónoma o dirigida, dependiendo de la misión asignada [6]. Recientemente se ha presentado un robot acuático capaz de asistir a personas en peligro dentro del mar, lo que abre una puerta a la robótica de asistencia marítima. Un ejemplo es el robot EMILY, mostrado en la Figura 1.6.



Figura 1.6: Robot EMILY para Salvamento Marítimo. Fuente: [11]

1.2.3. Robots Terrestres

Los robots terrestres son los más populares, están presentes prácticamente en todos los ámbitos industriales, permiten realizar tareas peligrosas o repetitivas debido a las características físicas que presentan. Estos pueden desplazarse en superficies muy irregulares gracias a los sistemas de locomoción que presentan (ruedas, patas u orugas). Al igual que el resto de robots pueden ser controlados de manera autónoma o teleoperada [6].

- **Ruedas:** diseñados para terrenos planos o regulares, son los más utilizados en

la industria ya que trabajan perfectamente en entornos dinámicos. Son usados para el desplazamiento de objetos pesados o para la manipulación de sistemas de manera continua. En ocasiones pueden llevar brazos articulados que permiten una interacción física con el entorno. Todos estos robots son capaces de desplazarse gracias a los sensores de su sistema, que les permiten ubicarse en el mapa mediante navegación autónoma. Están encargados de realizar junto a los operarios una labor de coordinación y de ejecución de tareas sincronizadas facilitando la maniobrabilidad de la carga [6][12]. Un ejemplo de este tipo de robots es el RB-VOGUI mostrado en la Figura 1.7.



Figura 1.7: Robot móvil RB-VOGUI. Fuente: [13]

- **Patas:** estos robots son capaces de simular el comportamiento de insectos o mamíferos, gracias a su estructura física inspirada en la biomimética. Están diseñados para poder desplazarse en entornos de alta complejidad, donde es necesario el uso de articulaciones. Asimismo, mediante patas, es posible conseguir la omnidireccionalidad y el deslizamiento en la locomoción es mucho menor [6][12]. El cuadrúpedo Spot es un ejemplo de los robot con patas, apreciándose en la Figura 1.8.



Figura 1.8: Robot cuadrúpedo Spot. Fuente: [14]

- **Orugas:** Su uso se ha estandarizado en la industria militar y espacial, ya que su objetivo principal es el de desplazarse a través de terrenos irregulares, donde es necesaria mayor estabilidad, tracción o capacidad de carga. Permiten

sortear obstáculos y atravesar largas distancias en terrenos peligrosos, donde a cualquier otro robot le resultaría imposible acceder. Son robots de exploración, por lo que a menudo llevan incorporado un brazo articulado para la manipulación de objetos, tal como se ve en la Figura 1.9 [6].



Figura 1.9: Robot desactivador de explosivos tEODor. Fuente: [15]

1.3. Principales Ámbitos de Aplicación de los Robots Terrestres

La adaptabilidad que han experimentado en la última década los robots terrestres les ha permitido integrarse en múltiples sectores. Esto ha cambiado la manera en la que se desempeñan los trabajos, facilitando y mejorando su eficiencia y su seguridad laboral [16]. Se van a explicar a continuación los ámbitos en los que la robótica ha tenido mayor impacto.

1. **Industria y Logística:** este sector es uno de los principales demandantes de robots de todos los ámbitos de actuación. En este campo los vehículos terrestres no tripulados son los encargados de la automatización de procesos y de realizar actividades donde se demande grandes esfuerzos físicos, desempeñando funciones como el de transporte de cargas o misiones de reconocimiento. Un ejemplo de estos robots es la base móvil usada para este proyecto, que se aprecia en la Figura 1.10 [16].



Figura 1.10: Plataforma móvil RB-THERON. Fuente: [17]

2. **Agricultura:** este sector ha crecido mucho en los últimos años. Gracias a la agricultura de precisión, permite automatizar procesos repetitivos, como la siembra o la supervisión de cultivos, sin necesidad de la interacción humana [16]. La marca Robotnik cuenta con uno de alta precisión, visualizándose en la Figura 1.11.



Figura 1.11: Robot agrícola del proyecto BACCHUS. Fuente: [18]

3. **Seguridad y Rescate:** comúnmente utilizados por los cuerpos de seguridad, permiten hacer búsquedas en entornos naturales de difícil acceso. Su uso minimiza el riesgo para los especialistas humanos y agiliza la localización de víctimas [16]. En la Figura 1.12 se puede observar un robot bombero utilizado en las fuerzas británicas.



Figura 1.12: Robot bombero para tareas de rescate. Fuente: [19]

4. **Medicina y Salud:** en el ámbito de la salud, estos robots se han convertido en dispositivos de gran ayuda. Son utilizados para el transporte de suministros, especialmente de residuos peligrosos, ya que necesitan unas condiciones especiales. A su vez son usados para la desinfección de áreas contaminadas,

evitando así la exposición de los sanitarios, pudiéndose apreciar un ejemplo en la Figura 1.13 [20].



Figura 1.13: Robot móvil UVD de desinfección sanitaria. Fuente: [21]

5. **Entornos Críticos:** se puede definir los entornos críticos como aquellos a los que el ser humano no puede acceder de forma segura. Destacando el ámbito militar y espacial, son dos sectores en los que un robot es altamente necesario, ya que estos son capaces de desplazarse por terrenos peligrosos [16]. El rover Perseverance de la Figura 1.14, es una muestra del avance tecnológico que se ha conseguido.

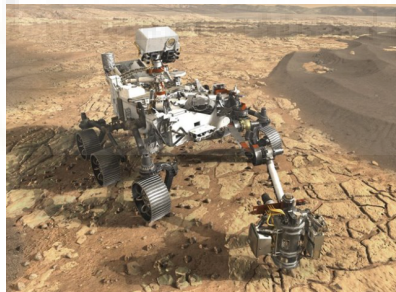


Figura 1.14: El rover Perseverance en la superficie de Marte. Fuente: [22]

1.4. Antecedentes

Existen varios estudios científicos que proponen soluciones para proporcionar autonomía a personas con discapacidad motora, haciendo uso de vehículos autónomos. Gran parte de los estudios se enfocan en la eliminación del joystick tradicional, debido a que presenta muchas incompatibilidades en pacientes con enfermedades en estadios avanzados.

Trabajos como el de Marcelo Arévalo Luzuriaga o J.M. Alcubierre se centran en diseñar un método de control basado en una arquitectura de reconocimiento de

voz, liberando al paciente de hacer uso de sus extremidades[23][24]. Sin embargo, presentan limitaciones en su interfaz, imposibilitando el uso del vehículo en entornos muy ruidosos o excluyendo por completo a pacientes con patologías severas en el habla.

Por otro lado, Julio González propone el desarrollo de una silla móvil denominada SENA[25], controlada también por voz, pero centrada en la colaboración con el usuario. Presenta una comunicación bidireccional asemejándose a una conversación humana entre el sistema y el usuario, de manera que el robot emite las dificultades físicas que es incapaz de superar y necesita la asistencia de un usuario externo. De manera análoga a los proyectos anteriores, este sistema mantiene las mismas limitaciones en cuanto a su interfaz principal, siendo inadecuadas para personas con discapacidad vocal.

Desde otra perspectiva, Fernando A. Auat Cheein propone un proyecto basado en algoritmos de SLAM (Mapeo y Localización Simultánea)[26]. Este enfoque demuestra la viabilidad y robustez del transporte mediante un robot, gracias al trazado de trayectorias y la evasión de obstáculos. A pesar de este avance, cuenta con un inconveniente principal, este vehículo autónomo necesita que le seleccionen el destino mediante una pantalla táctil, lo que resulta inviable en pacientes con movilidad reducida en el tren superior.

Para dar respuesta a estas necesidades, Iñaki Iturrate junto a su equipo de la universidad de Zaragoza, proponen el proyecto “Dispositivos robóticos de rehabilitación basados en Interfaces Cerebro-Ordenador: silla de ruedas y robot para teleoperación”[27], donde se centran en crear una arquitectura de control basada en el uso de la actividad cerebral del paciente. Permite al usuario desplazarse por un entorno controlado por encefalografía que son medidas mediante electrodos superficiales y a una pantalla como método de visualización, esta le muestra los posibles destinos. Como consecuencia, este tipo de control generan en el paciente fatiga mental, debido a la carga intelectual necesaria para la selección de la ubicación deseada.

Como se puede observar, se han presentado algunos proyectos destinados a la eliminación de los dispositivos externos manuales, encargados del manejo del vehículo. Pero todos ellos cuentan con algún inconveniente que justifica el desarrollo del presente proyecto.

1.5. Problemática

A pesar de la tecnología existente en el mundo de la robótica móvil y de los vehículos para personas con movilidad reducida, nos enfrentamos a 4 problemas que destacan en el presente proyecto.

El primero es el método de control del vehículo por el usuario. La mayoría dependen de mandos manuales, excluyendo de su uso a las personas sin ninguna capacidad de movimiento, por ejemplo, pacientes con tetraplejia.

En segundo lugar, la necesidad de integración entre dispositivos que no comparten la misma arquitectura. Para poder integrar ROS (*Robot Operating System*) con elementos externos, es necesario desarrollar un puente de enlace entre ambos dispositivos, haciendo uso de protocolos de comunicación. En este proyecto se ha optado por el protocolo de comunicación UDP (*User Datagram Protocol*).

En tercer lugar, surge la obligación de desarrollar una arquitectura de control propia y adaptada. Aunque el robot RB-THERON ya cuenta con un sistema de navegación de fábrica controlado por el mando Dualshock o la interfaz web HMI (Human-Machine Interface), estos métodos no están pensados para ser operados por pacientes con problemas motores severos. Dada esta situación, es necesario diseñar e implementar una arquitectura en ROS que permita gestionar los diferentes nodos del robot, asegurando un desplazamiento óptimo y seguro, sin la intervención física de un usuario externo.

Finalmente, destaca el factor de seguridad. Las plataformas móviles industriales son dispositivos diseñados para el transporte de cargas pesadas en el menor tiempo posible, siendo inadecuadas para el transporte humano. De este modo, debemos corregir la aceleración y la velocidad máxima a la que es capaz de transportar el robot al paciente, para generar un entorno estable.

1.6. Objetivos

El objetivo principal de este proyecto se enfoca en dotar de autonomía de desplazamiento a personas con discapacidad motora severa, desarrollando una solución viable para aquellos pacientes incapaces de controlar la interfaz estándar (joystick).

Para alcanzar este propósito se definen los siguientes objetivos específicos:

1. **Implementación en ROS:** configurar y controlar los nodos de la base móvil RB-THERON, gobernados mediante ROS Noetic, permitiendo la gestión de los diferentes sensores.
2. **Interoperabilidad (Comunicación UDP):** crear un método de comunicación basado en UDP, que permita controlar el robot RB-THERON mediante sistemas externos a ROS.
3. **Mapeo del Entorno:** implementar un sistema de desplazamiento autónomo de forma inteligente, usando la herramienta gráfica (HMI) que incorpora el robot.
4. **Varios Modos de Control:** crear un método de control manual para una interacción humano-máquina más precisa, que permita mayor aproximación a objetivos del mapa.
5. **Feedback de Estado:** integrar un sistema capaz de enviar de forma continua la información necesaria sobre el estado del robot.

2. Materiales y Métodos

En el presente capítulo se detallan los componentes físicos, el entorno de desarrollo y la arquitectura de software en ROS necesaria para la correcta comunicación con la plataforma móvil. Inicialmente, se exponen las características técnicas del robot RB-THERON junto a la justificación del entorno de trabajo utilizado para dar soporte a la arquitectura ROS. A continuación, se describe la herramienta gráfica que incorpora la base para poder realizar el mapeo del área de trabajo deseada y hallar las coordenadas de los puntos de interés. Como bloque central, se muestra el diseño y el flujo de ejecución de los nodos que hacen posible la navegación autónoma de manera segura y eficiente. Finalmente, se presentan las aplicaciones utilizadas para la adquisición y procesamiento de los datos para su posterior análisis.

2.1. Plataforma móvil: Robot RB-THERON

RB-THERON se trata de una plataforma móvil autónoma, diseñada principalmente para entornos industriales de interior, donde requieren flexibilidad y modularidad. Permitiendo transportar cargas de hasta 200 Kg [28].

Este robot ha sido seleccionado para este proyecto por su versatilidad y características que presenta, es capaz de navegar por entornos industriales con eficiencia, evitando objetos y seleccionando la ruta óptima a su destino. Mediante la plataforma elevadora, es capaz de transportar mercancías pesadas con gran facilidad, incluyendo elementos de tamaños variados. A su vez presenta varios sensores, encargados del control y posicionamiento de la plataforma. Se muestra un esquema de los componentes principales del robot RB-THERON en la Figura 2.1.

Este robot presenta un diseño modular que permite integrar sensores y componentes adicionales según las necesidades del proyecto.

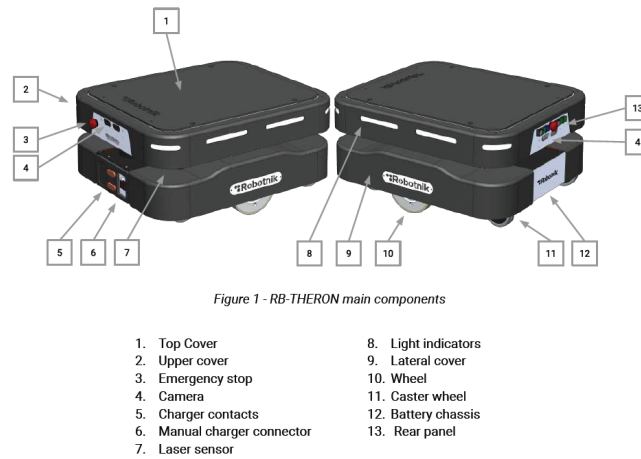


Figura 2.1: Componentes principales RB-THERON. Fuente: [28]

2.1.1. Hardware y Componentes de seguridad

El chasis cuenta con un panel trasero orientado a poner en marcha el robot de manera manual. Integra un selector donde permite elegir la posición de encendido o apagado. Como medida de seguridad adicional, dispone de una seta de paro de emergencia, destinado a corta la potencia ante cualquier evento crítico.

Como método de seguridad adicional, el robot exige la pulsación del botón de reinicio, para garantizar un arranque seguro, es debido a posibles acumulaciones de comandos en su sistema. Esto obliga la interacción de un operario para confirmar la primera orden de “Robot Seguro”, reiniciando los posibles comandos enviados. Para visualizar el estado del robot cuenta con dos métodos, el primero se trata de unas tiras LED en los costados, y el segundo método es mediante un diodo LED que se encuentra en la botonera, como se puede apreciar en la imagen 2.2.

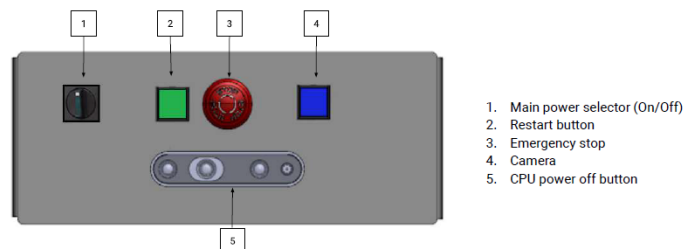


Figura 2.2: Botonera de control RB-THERON. Fuente: [28]

Para el manejo manual del robot, se dispone de un Dualshock controller, pudiendo ser teleoperado por los operarios, siendo este el método de control de fábrica, se puede apreciar el mando utilizado en la Figura 2.3.



Figura 2.3: Controlador Dualshock. Fuente: [28]

2.1.2. Sistema Sensorial

Para tener una visión general del entorno, ejecutar algoritmos de mapeo y localización (SLAM) y planificar rutas donde sea capaz de evitar obstáculos, el robot cuenta con dos sensores principales.

1. **Cámaras de profundidad (3D):** presenta dos sensores de cámara 3D, siendo estas las encargadas de detección volumétrica de obstáculos, ubicación del robot o lectura de marcadores QR, como se ilustra en la Figura 2.4 el rango de visión del robot.



Figura 2.4: Rango de visión de la cámara del RB-THERON. Fuente: [28]

2. **Sensores LiDAR 2D:** además de las cámaras, dispone de dos sensores LiDAR 2D ubicados en sus esquinas opuestas. Estos sensores se utilizan para la navegación y la detección de obstáculos. Es capaz de realizar un barrido en un plano horizontal a una altura de 180 mm, siendo los encargados de la detección principal de obstáculos. Se puede ver una representación de la medida del sensor LiDAR en la Figura 2.5.



Figura 2.5: Rango de visión del laser LiDAR del RB-THERON. Fuente: [28]

2.1.3. Cinemática y Locomoción

El vehículo presenta una configuración cinemática en modo diferencial compuesto por 6 ruedas, de las cuáles 2 de ellas van motorizadas, siendo estas las encargadas de generar el par suficiente para desplazar la base móvil junto a la carga. Las ruedas restantes se les conoce como ruedas de apoyo o ruedas giratorias, proporcionan estabilidad a la plataforma y permite girar sobre sí mismo. La disposición de estas ruedas se pueden observar en la Figura 2.6.

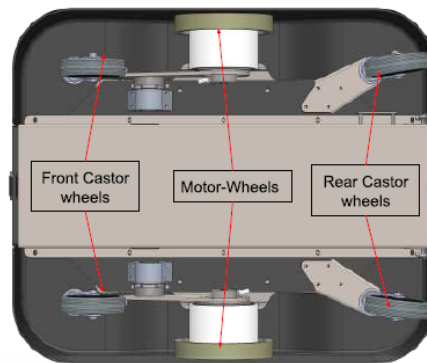


Figura 2.6: Ruedas del RB-THERON. Fuente: [28]

2.1.4. Interfaces de Control

El robot RB-THERON está destinado a ser controlado y monitorizado mediante dos alternativas principales de operación:

1. **Interfaz HMI (*Human Machine Interface*)**: el robot puede ser controlado mediante una aplicación web integrada en su sistema local. Destinada a ser usada por cualquier usuario sin conocimientos previos. Se ejecuta en el navegador web, pudiendo ser teleoperado y monitorizado de forma sencilla e intuitiva, esto es gracias a la programación en bloques que ofrece el apartado de secuencias.
2. **Programación ROS**: otra opción más potente destinada a la investigación o al desarrollo de nuevas funcionalidades, se realiza mediante la programación bajo la arquitectura ROS (*Robot Operating System*), que permite el control directo de la base. Ofrece un acceso más profundo al sistema, permitiendo programar comportamientos más complejos. Este método está reservado para personas con conocimientos avanzados de robótica.

2.2. Entorno de Desarrollo y Middleware

Para la realización de este proyecto, ha sido necesaria la elección de un entorno de trabajo fiable, estandarizado y robusto, donde las diferentes herramientas utilizadas presenten total compatibilidad entre sí. Esto ha permitido que la comunicación con el robot RB-THERON sea posible mediante unos estándares universales. En concreto, la arquitectura desarrollada se apoya en el sistema operativo Ubuntu 20.04 LTS, utilizado para proporcionar soporte a la programación pertinente. A su vez, se ha utilizado el middleware ROS (Robot Operating System) en su distribución Noetic. A continuación, se describen los sistemas que han sido necesarios para el correcto desarrollo de la arquitectura de control en ROS.

2.2.1. Sistema Operativo (Ubuntu 20.04 LTS)

El medio de trabajo sobre el que se desarrolla la arquitectura de control de este proyecto es Ubuntu 20.04 LTS (Focal Fossa). Este sistema operativo, basado en el núcleo Linux, tiene la ventaja de ser código abierto (software libre). Esto permite su adaptación a las necesidades del programador, ofreciendo un entorno de desarrollo accesible, versátil y altamente personalizable, características que resultan fundamentales en el ámbito de la robótica.

El motivo por el que se ha llevado a cabo el proyecto mediante este sistema operativo radica en su compatibilidad directa con ROS. Especialmente, la versión de Ubuntu 20.04 es el soporte oficial que se recomienda para poder instalar y ejecutar la distribución ROS Noetic. Este sistema gestiona directamente el hardware de red y los puertos de comunicación del ordenador. De este modo, Ubuntu permite la ejecución de los controladores oficiales de Robotnik a través del middleware (ROS), lo que garantiza la lectura de los sensores y el envío de comandos.

2.2.2. Robot Operating System (ROS)

A pesar de su denominación, ROS (Robot Operating System) no es reconocido como un sistema operativo, se trata de un middleware o framework, lo que significa que es una capa intermedia que hace de puente entre el sistema operativo (Ubuntu) y los programas del usuario (nodos), facilitando la comunicación y evitando la programación directa del hardware del robot [29, pp. 3 y 54]. Este middleware comenzó en el año 2007 en el departamento de inteligencia artificial de la Universidad de Stanford, siendo posteriormente consolidado por la empresa Willow Garage.

Su objetivo principal ha sido la estandarización de interfaces y mecanismos de comunicación, promoviendo la modularidad y reutilización de componentes, así como la creación de un sistema de código abierto que facilite la integración entre las diferentes arquitecturas. Todo esto garantiza mayor facilidad, compatibilidad y robustez en el desarrollo de softwares robóticos.

2.2.2.1 Justificación de la elección ROS y la distribución Noetic

ROS en su distribución Noetic presenta gran modularidad, lo que es fundamental para este proyecto. Esto supone que, en vez de controlar el robot en un único programa, ROS permite la segmentación de sistemas mediante procesos aislados, donde otorga el control del hardware a diferentes subprocesos. Para el presente proyecto, esta característica es fundamental, ya que permite acoplar de manera eficiente los controladores oficiales proporcionados por Robotnik (gestión de motores y sensores LiDAR) con el nodo central de desarrollo propio. Esto garantiza que todos los nodos, tanto los nativos (desarrollados por Robotnik) como los programados para este TFG, interactúen entre sí de manera adecuada y fiable.

Las tres características técnicas principales que ofrece la distribución de ROS Noetic son las siguientes:

1. **Homologado por el fabricante:** se trata de la distribución original exigida por el fabricante Robotnik, lo que garantiza un correcto funcionamiento de la base móvil RB-THERON sobre el sistema operativo Ubuntu 20.04.
2. **Repositorio de paquetes oficiales (*Navigation Stack*):** la Navigation Stack se define como el conjunto de paquetes estandarizados de ROS, encargados de procesar los datos de los sensores (láser LiDAR y odometría) para calcular y planificar la trayectoria de manera segura y eficiente, enviando los comandos adecuados a los motores. ROS Noetic ofrece un entorno fiable y estable para poder ejecutar las herramientas necesarias para el control autónomo del robot RB-THERON.
3. **Herramientas de verificación y registro de ensayos:** la distribución Noetic incorpora una serie de herramientas que permiten al usuario visualizar y analizar el comportamiento de su programa en ROS. Estas acciones son necesarias para hacer un estudio previo del sistema y validar su correcto funcionamiento. Algunas de las utilidades son las siguientes: *RViz* (visualización y análisis de sensores), *rqt_graph* (análisis del grafo de computación, muestra los

nodos en ejecución y sus conexiones) y *rosvbag* (grabación y reproducción de datos).

2.2.2.2 Estructura de ROS

El funcionamiento interno de ROS se basa en una red de procesos punto a punto, comúnmente conocida como grafo de computación (*Computation Graph*). Se trata de una red de nodos interconectados entre sí, los cuales intercambian información mediante tópicos o servicios. La resolución de nombres y la gestión de parámetros se realiza mediante el ROS Master junto al *Parameter Server*. Este middleware actúa como puente universal, independientemente del lenguaje de programación empleado (C++ o Python). En los puntos siguientes se explican los elementos clave de la red de comunicación ROS:

- **El Maestro (*ROS Master*)**: es el núcleo del sistema. Se trata de un servidor de nombres o nomenclaturas cuyo objetivo es permitir que los nodos individuales de ROS se localicen entre sí. Actúa como el registro central que permite la comunicación entre nodos, realizando seguimientos de los publicadores o suscriptores. Se puede observar un ejemplo de la conexión interna en la Figura 2.7.

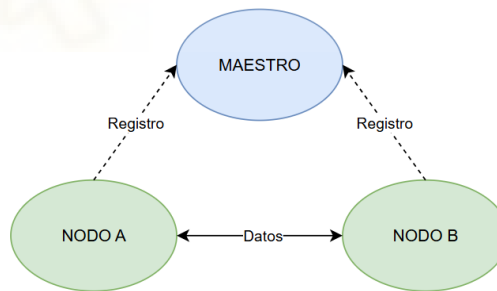


Figura 2.7: Esquemático de relaciones con el ROS Master.

- **Nodos (*Nodes*)**: son las unidades básicas de ROS encargadas del procesamiento de los datos. Son principalmente programas que realizan tareas específicas y modulares, como leer el movimiento de las ruedas, calcular la trayectoria o enviar comandos a los motores.
- **Mensajes (*Messages*)**: son estructuras de datos fuertemente tipadas que se utilizan para intercambiar información entre nodos. Permiten enviar tanto variables simples (números enteros o texto) como estructuras más complejas (matrices de odometría).

- **Tópicos (*Topics*)**: son los canales asíncronos de comunicación entre nodos, basados en el modelo de Publicación-Suscripción. Un publicador (*Publisher*) es el encargado de enviar los mensajes a un tópico específico sin conocer quién lo recibe, y un suscriptor (*Subscriber*) es aquel que se enlaza a ese tópico para recibir la información enviada por el publicador, pudiéndose ver un ejemplo en la Figura 2.8. De esta manera se genera una relación donde un nodo envía datos y otros los reciben.

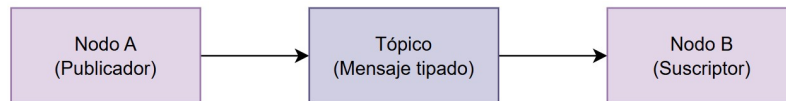


Figura 2.8: Ejemplo de comunicación mediante un tópico.

- **Servicios (*Services*)**: es el canal de comunicación síncrono basado en el modelo Petición-Respuesta. Está orientado a operaciones puntuales donde el nodo cliente se bloquea hasta recibir la respuesta del servidor. A diferencia de los tópicos los cuales envían información de manera constante, los servicios están destinados a acciones aisladas donde es necesaria la confirmación de la respuesta. Se ha generado un ejemplo ilustrativo que permite entender mejor la comunicación, visualizándose en la Figura 2.9.

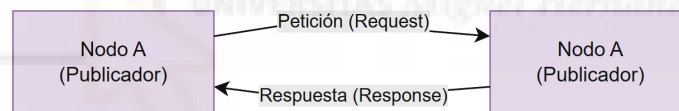


Figura 2.9: Ejemplo de comunicación mediante un servicio.

- **Servidor de parámetros (*Parameter Server*)**: es un diccionario compartido dentro del ROS Master bajo el modelo clave-valor, el cual permite configurar parámetros globales en tiempo real, evitando reiniciar el sistema en cada configuración. Una de las herramientas más importantes es *dynamic_reconfigure*, la cual permite alterar parámetros como la velocidad del robot durante el tiempo de ejecución.

2.3. Generación del Entorno de Navegación mediante la interfaz HMI

Con el fin de que la base móvil RB-THERON sea capaz de ubicarse y navegar de forma autónoma hacia el destino marcado, es necesario generar un mapa bidimensional del entorno físico y obtener las coordenadas espaciales de los puntos a los que

se desea que llegue el robot. Para poder realizar el mapeo del espacio de trabajo, se hace uso de la *Human Machine Interface* (HMI) que proporciona el fabricante Robotnik.

Esta interfaz gráfica consiste en una aplicación web interna que se ejecuta de manera local en el robot. Esto permite monitorizar su estado, gestionar las acciones y generar el mapeo de manera más accesible para el usuario. El acceso a esta aplicación se realiza conectando un equipo (PC) a la red local que proporciona el RB-THERON e introduciendo la IP del robot (http://192.168.0.200/robotnik_hmi) en un navegador web.

2.3.1. Mapeo 2D del entorno (*Gmapping*)

El objetivo de este apartado consiste en digitalizar en 2D el área de trabajo por donde se desea mover el robot de forma autónoma. Para acceder a esta opción, se debe seleccionar el modo *Gmapping* desde la interfaz principal (HMI). Este algoritmo procesa la lectura de datos obtenida mediante el escáner láser LiDAR para crear el mapa del recinto en tiempo real.

El procedimiento para el mapeo consiste en teleoperar el robot RB-THERON de manera manual por todas las estancias en las que se desee operar con la base móvil.

Una vez completado el recorrido, la interfaz permite asignar un nombre al mapa generado y guardarlo en su memoria. Esto posibilita la generación de varias áreas de trabajo y almacenarlas en la base de datos del sistema para su posterior uso. El mapa generado mediante este procedimiento es el mostrado en la Figura 2.10.

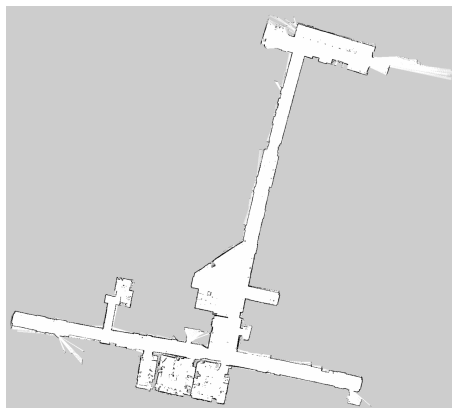


Figura 2.10: Mapa 2D generado mediante el algoritmo Gmapping.

2.3.2. Localización y obtención de coordenadas (*localization_2d*)

Una vez se haya cargado el mapa en la base de datos del robot, este es capaz de seleccionar diferentes entornos gráficos en los que realizar sus operaciones. Para ello se debe asegurar que el módulo de selección conste en modo *localization_2d*, dando la opción de definir el *Environments* que más se adecue a la ocasión.

En este punto, el sistema presenta una particularidad en su funcionamiento. La base móvil cuenta con un sistema de localización inicial marcado por el punto de finalización del proceso de mapeo, lo que quiere decir que en el momento de iniciar el robot, este supondrá que se encuentra ubicado en el mismo punto de partida donde finalizó el proceso de cartografiado. Esto supone una limitación en su diseño, ya que, si el robot no se encuentra dentro de los puntos indicados por el sensor LiDAR al finalizar el mapeo, este no realizará ninguna acción predefinida, indicando que no se ubica dentro de los márgenes establecidos. En la imagen 2.11 se aprecia la interfaz que pone a disposición Robotnik para hallar las coordenadas espaciales.

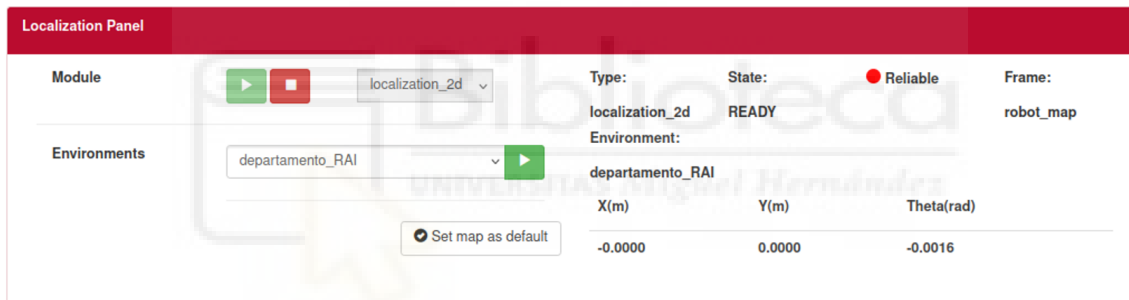


Figura 2.11: Interfaz gráfica HMI para la visualización y obtención de coordenadas espaciales.

Para solucionar esta problemática, el robot presenta una herramienta denominada *Init Pose* que alinea la estimación del robot con la realidad física en la que se encuentra.

Una vez el sistema confirma que el estado del robot es adecuado para su desplazamiento (*Reliable*), se procede a llevar de manera manual al robot hacia los puntos de interés en los que se desea operar. La interfaz gráfica proporciona en tiempo real una serie de características, destacando las coordenadas espaciales exactas (x , y , θ) de dicha ubicación. Estos valores numéricos son los parámetros fundamentales para posteriormente enviar el robot a la ubicación deseada mediante programación.

2.4. Arquitectura de control en ROS

El fabricante ofrece una herramienta que permite mapear el entorno y obtener las coordenadas espaciales. Esto posibilita el uso de la plataforma RB-THERON a través de unas secuencias predefinidas que se encargan de enviar el robot a puntos específicos. Sin embargo, esta interfaz impide la integración con dispositivos externos y la automatización de la plataforma. Por lo que este proyecto se ha centrado en el diseño, programación e implementación de una arquitectura de software basada en ROS, bajo su distribución Noetic.

La misión de este proyecto reside en crear un sistema de control a distancia seguro, capaz de enviar y recibir datos de forma simultánea. Este conjunto de programas, dan al robot la capacidad de ser operado de manera autónoma o manual, junto con la actuación del elevador, recibiendo de manera continua un feedback de su estado en tiempo real.

Se ilustra en la Figura 2.12 un esquema representativo de la comunicación bidireccional entre el robot y el PC de control.

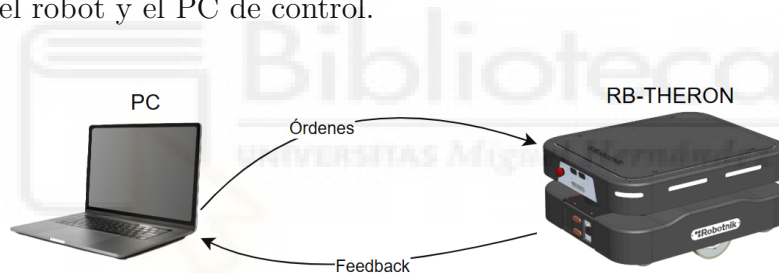


Figura 2.12: Esquema de comunicación inalámbrica entre el PC y el robot.

Para lograrlo, el diseño de la arquitectura se ha centrado en su división en dos bloques. El primero se encarga del control externo, desde el cual se maneja el robot junto a la recepción de su estado. El segundo bloque está destinado al dominio interno, que es el responsable de enviar las instrucciones necesarias a los actuadores del propio vehículo, controlados por el nodo central (`robot_navigator`), el cual se ejecuta en el ordenador que lleva integrado el robot.

Gracias a este método de división, podemos ordenar acciones desde el exterior mientras el nodo del robot las procesa de forma independiente en su interior. Todos estos intercambios se realizan en tiempo real, garantizando un control seguro de la máquina.

2.4.1. Diagrama Global del Sistema

Para comprender la arquitectura diseñada en ROS, se ha creado un esquema general del sistema, donde se explica el flujo de información y los procesos utilizados en este proyecto. Se divide en dos bloques: el primero (PC de Control) es el encargado de enviar las instrucciones al robot, junto con la recepción del estado de este. El segundo bloque (PC del Robot) es el encargado de interactuar con los nodos internos del robot, para cumplir con las acciones que se le han destinado.

En la Figura 2.13 se aprecia una imagen de toda la arquitectura ROS diseñada para el correcto funcionamiento de la base RB-THERON.

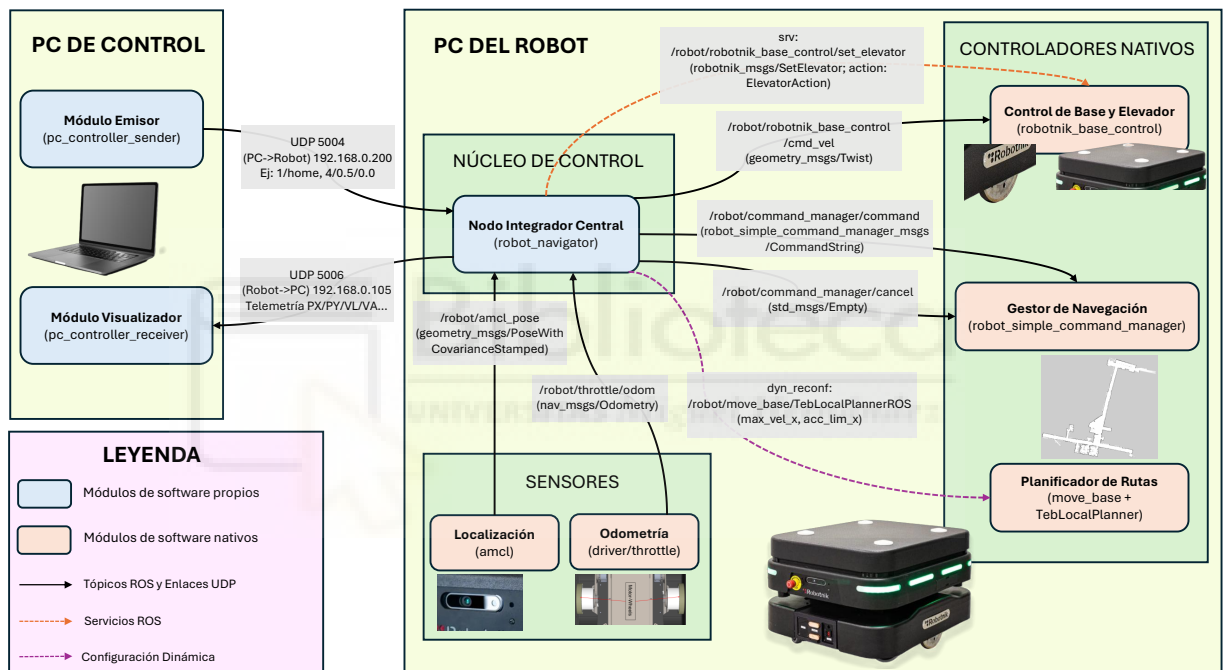


Figura 2.13: Esquema general de la arquitectura ROS

2.4.1.1 PC del Robot

Este bloque representa todo el procesamiento interno que alberga la CPU del robot RB-THERON. Es el encargado de controlar los actuadores y sensores integrados en el robot, que hacen posible la interacción con la base móvil.

Dentro de este bloque se diferencian tres unidades funcionales que interactúan entre sí para controlar la base de manera correcta:

- **Núcleo de Control (*robot_navigator*)**: es el programa central de este pro-

yecto. Actúa como traductor de los comandos recibidos por el módulo emisor, junto con su envío de vuelta de la telemetría del estado del robot. Además, se encarga de interactuar con los nodos y llamar a los servicios internos para cumplir con las acciones necesarias.

Este nodo se ha programado usando hilos independientes (*threading*), los cuales trabajan de manera concurrente para no interrumpir el funcionamiento de la base móvil. El primero de ellos es el encargado de recibir por UDP todos los comandos, este los traduce y hace las llamadas a las funciones pertinentes. El segundo hilo se encarga de obtener la información de los sensores para posteriormente enviarla por telemetría. La explicación de este nodo se detallará en el punto 2.4.2.

- **Controladores Nativos:** este bloque está compuesto por los nodos desarrollados por el fabricante Robotnik, los cuales permiten el control de los actuadores de la base RB-THERON.

La comunicación con estos nodos se realiza a través de tópicos o servicios. Los nodos que se han usado para este proyecto son los siguientes:

1. ***robot_simple_command_manager*:** gestiona la navegación de manera autónoma a través del tópico `/robot/command_manager/command`, enviando la base móvil a un destino predefinido. También es capaz de detener al robot de manera inmediata mediante el tópico `/robot/command_manager/cancel`.
 2. ***robotnik_base_control*:** gobierna de manera directa los motores de las ruedas y la plataforma elevadora. El movimiento de la base es controlado mediante el tópico `/robot/robotnik_base_control/cmd_vel`, inyectando la velocidad a la que debe moverse. El elevador se controla a través de un servicio propio (`/robot/robotnik_base_control/set_elevator`), que se encarga de subir o bajar la plataforma.
 3. ***move_base + TebLocalPlannerROS*:** este nodo se utiliza para restringir la velocidad y la aceleración del robot. Se controla a través de una configuración dinámica (*dynamic reconfigure*), la cual permite ajustar de manera temporal los parámetros de cinemática predefinidos por el fabricante.
- **Sensores:** se trata de los tópicos a los que el programa principal se suscribe para recibir información. Cuenta con dos suscriptores principales para monitorizar el estado del robot. Se suscribe al tópico `/robot/throttle/odom`, quien se encarga de enviar la velocidad real de las ruedas de manera constante. Por

otro lado, recibe información del tópico `/robot/amcl_pose`, el cual proporciona la localización espacial de la base en todo momento. Estos sensores permiten al nodo central conocer la pose y el estado real del dispositivo.

2.4.1.2 PC de Control

Este equipo es controlado por un usuario, quien se encarga de decidir las acciones que debe ejecutar el robot. Además, recibe información real del estado del robot en todo momento, como método de comprobación.

En este ordenador operan los módulos externos desarrollados de manera propia para este proyecto, los cuales han sido utilizados para validar el correcto funcionamiento y la comprobación de la navegación del robot:

- **Módulo Emisor (*pc_controller_sender*)**: este módulo es el encargado de capturar y enviar por teclado los comandos para el control del robot. El programa permite enviar cuatro tipos de acciones, siendo estas las siguientes:
 - **1/**: marca un punto de destino para que el robot se traslade de manera autónoma.
 - **2/**: detiene el robot de cualquier acción que tuviera en curso.
 - **3/**: eleva o baja la plataforma destinada para el transporte de mercancías.
 - **4/**: habilita el modo manual, permitiendo manejar de manera directa al robot, haciendo uso de las flechas del teclado.
- **Módulo Visualizador (*pc_controller_receiver*)**: este programa se ha destinado a la comprobación del estado del robot. Muestra en pantalla una cadena de valores que permite conocer la información más relevante en tiempo real: posición (x, y), velocidad (lineal, angular), distancia al objetivo, tiempo de llegada, estado general y posición del elevador.

Estos comandos son enviados y recibidos mediante el protocolo de comunicación UDP. Esto es debido a que el envío de señales está destinado a ser usado por un dispositivo externo ajeno al estándar ROS. De este modo, se garantiza una correcta integración con elementos que no comparten el mismo protocolo de comunicación.

2.4.2. Nodo Principal

La función de este programa consiste en integrar los dispositivos externos con la base móvil. Sirve de traductor entre el ordenador que controla el usuario y los programas internos del robot. El nodo se encarga de recibir por UDP las instrucciones y convertirlas en comandos de ROS para que el robot entienda la tarea que debe realizar. Al mismo tiempo, envía por telemetría el estado del robot para comprobar su funcionamiento, gracias a los sensores internos.

A continuación, se explica cómo está estructurado y cómo se procesa cada comando.

2.4.2.1 Entradas y Salidas del Nodo

a) **Entradas** Se trata de la información que recibe el nodo. El robot necesita saber qué acción quiere que realice el usuario y cuál es su propio estado en todo momento. Por ello utiliza dos vías de comunicación:

1. **Sensores de ROS:** se trata de los nodos nativos del robot, que dan información del estado de la base móvil de manera continua. Por ello el nodo principal se suscribe a los tópicos para leer los datos. El tópico */robot/amcl_pose* se encarga de enviar la posición exacta del robot en el mapa. Por otro lado, el tópico */robot/throttle/odom* lee la odometría para saber la velocidad de las ruedas.
2. **Comandos del usuario:** al mismo tiempo que capta la información de los sensores, el programa recibe las órdenes del PC a través del puerto UDP que son enviadas por el usuario. La recepción por UDP permite la interacción entre dispositivos que no comparten el mismo sistema de comunicación, lo que hace más versátil el programa.

b) **Salidas**

Una vez que se arranca el nodo, este es capaz de realizar cuatro acciones diferentes. Para ello, interactúa con los distintos nodos nativos, llamando a servicios o publicando en tópicos.

1. **Enviar órdenes de movimiento:** el programa está pensado para trabajar mediante dos modos distintos de control, uno de operación manual y otro de manera autónoma. El control manual se realiza publicando la velocidad en el tópico */robot/robotnik_base_control/cmd_vel*, a través de un mensaje tipo Twist. El siguiente modo consiste en la navegación autónoma del robot, donde se publican las coordenadas en un tópico llamado

/robot/command_manager/command, el cual ha sido desarrollado por el propio fabricante.

2. **Llamar a servicios:** se ha desarrollado una función que llama al servicio del elevador a través de */robot/robotnik_base_control/set_elevator*. Este servicio es único de la base móvil RB-THERON y la interacción con esta plataforma lo especifica el fabricante a través de su manual. Este comando se detallará en el apartado 2.4.2.2.
3. **Configurar los parámetros internos:** la base móvil se utiliza para el transporte de personas con algún tipo de discapacidad física, por lo que el control de la velocidad y la aceleración máxima es necesario. Al iniciar el programa, el nodo limita las consignas de velocidad y aceleración lineales máximas, a través de la configuración dinámica del robot. Este cambio se realiza en el nodo *move_base* a través del módulo *TebLocalPlannerROS*, obligando al robot a moverse con los parámetros impuestos.
4. **Enviar el estado por telemetría:** para comprobar el funcionamiento del robot, este empaqueta toda la información que se comentó en el apartado anterior y la envía de vuelta al usuario a través de un puerto UDP. Es un método de visualización para verificar si se cumplen con las características deseadas.

2.4.2.2 Procesos Internos del Programa

Para evitar que el sistema se bloquee cuando espera la recepción de comandos externos, mientras al mismo tiempo debe recibir los datos de los sensores, es necesario crear hilos de ejecución de manera concurrente (*threading*). Esto divide el trabajo en dos procesos que operan simultáneamente sin interponerse entre ellos. Se necesita crear hilos debido a que la función que escucha por UDP bloquea el sistema esperando únicamente la recepción de datos por Wi-Fi. Es decir, congela el programa principal impidiendo leer los tópicos de los sensores.

1. **Hilo de recepción UDP:** este subproceso se mantiene escuchando de manera continua los comandos enviados por el puerto UDP, cuando llega un comando por la red, lo procesa y ejecuta la acción correspondiente. De este modo, se evita bloquear al robot si se necesita información de otras fuentes.
2. **Lectura de sensores:** por otro lado, se utiliza un temporizador (*timer*) que se dedica a comprobar los tópicos cada 0.3 segundos. Esta función calcula la

distancia al objetivo, comprueba si la máquina está en movimiento y envía de vuelta por telemetría la información del estado del robot.

En la siguiente imagen se puede visualizar el flujo de programación:

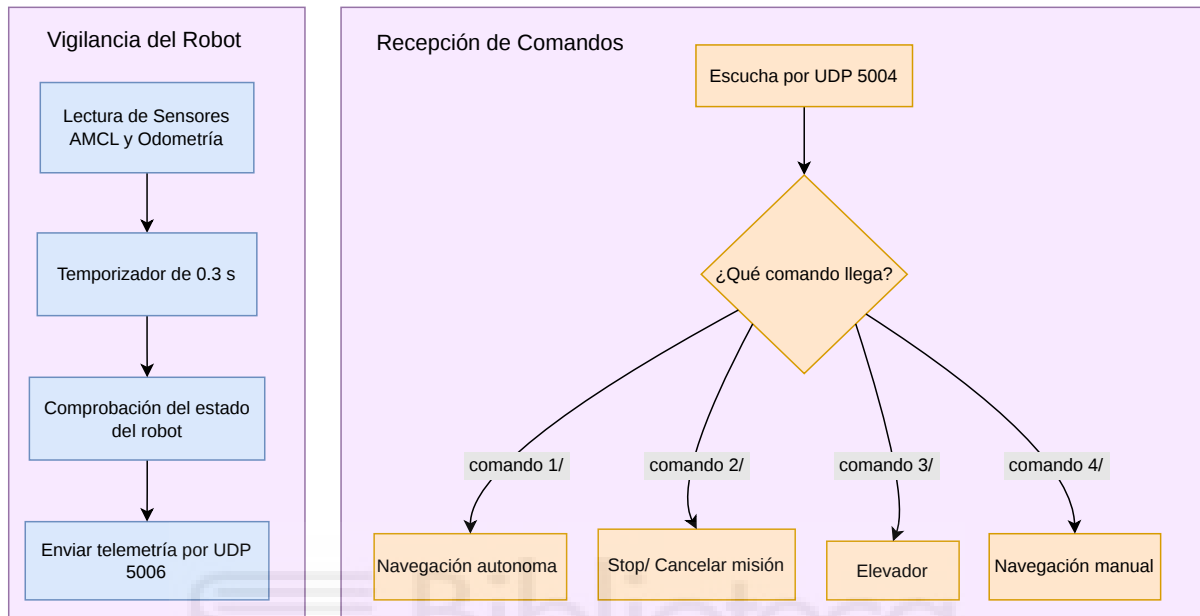


Figura 2.14: Flujo general de programación.

2.4.2.3 Procesos de Ejecución por Comandos

Dependiendo del comando que se recibe por el puerto UDP, el programa ejecuta unas funciones u otras. Como se ha comentado en el apartado anterior, el usuario (PC Externo), envía cuatro posibles comandos, los cuales seleccionan la acción que debe realizar el robot.

A continuación, se explican los cuatro posibles casos a través de diagramas de flujo:

- **Comando 1/(destino):** este comando se encarga de enviar la base móvil a las coordenadas guardadas en su memoria. Si el nodo recibe, por ejemplo, el comando 1/mechanics, este busca en el diccionario si esa palabra se encuentra dentro de los posibles destinos junto a las coordenadas predefinidas. Si existe este lugar, el programa guarda las coordenadas y publica el mensaje GOTO x y theta en el tópico /robot/command_manager/command. El Command Manager lanza la navegación interna move_base + TEB, que se encargan de planificar la ruta y esquivar los obstáculos.

En la siguiente figura se puede observar la parte del código encargada de procesar el comando y llamar a la función de navegación.



```
1  # Recepción comando 1/
2      if command_id == "1":
3          if action in self.target_cmds: # Comprueba si existe
4              self.navigate_to(action) # Llama a la función que
5              envía el GOTO
6              # action es el destino recibido (ej: mechanics)
7          else:
8              rospy.logwarn("Destino no reconocido")

1 # 2) Envío del destino: guarda objetivo y publica (GOTO x y theta)
2 def navigate_to(self, location):
3     if location in self.target_cmds:
4         cmd = self.target_cmds[location] # (GOTO x y theta)
5         self.current_target = self.target_coords[location]
6         self.has_arrived = False
7         # Publica el comando en el Command Manager
8         self.command_pub.publish(CommandString(command=cmd)) # tópico /robot/
9         command_manager/command
10        return True
11        return False
```

Figura 2.15: Fragmento de código para el procesamiento del comando UDP y ejecución de la navegación.

Al mismo tiempo, el temporizador comprueba la posición y velocidad del robot, calcula el tiempo y distancia al objetivo, actualiza el estado del robot y por último lo envía por telemetría de vuelta al PC.

En el siguiente esquema se puede ver el diagrama de flujo del comando 1/destino.

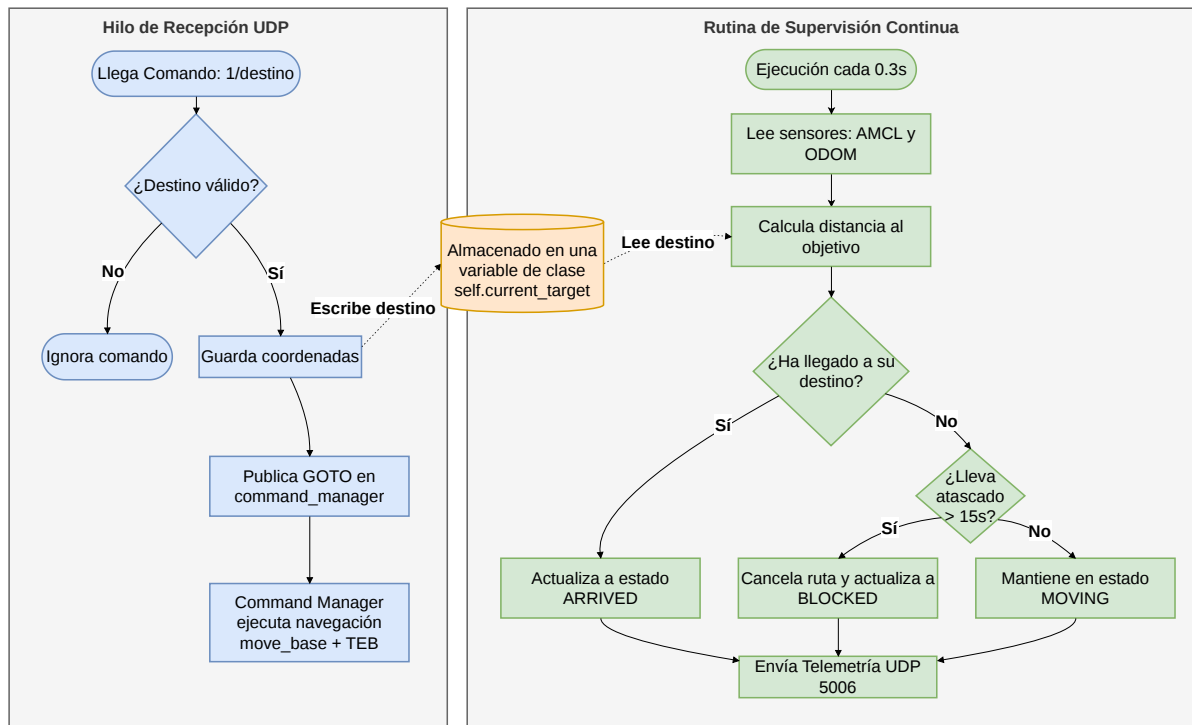


Figura 2.16: Diagrama de flujo del comando 1/destino.

- **Comando 2/stop:** para detener el robot de forma instantánea, se ha implementado un comando llamado 2/stop por motivos de seguridad. El programa realiza dos acciones para forzar la parada. En la primera, publica un mensaje vacío en el tópico `/robot/command_manager/cancel`, que se encarga de cancelar cualquier ruta que tuviera en su memoria de manera automática. Como método de protección, el programa publica un mensaje de tipo Twist con valores nulos (velocidad lineal, velocidad angular) en el tópico `/robot/robotnik_base_control/cmd_vel`. Esto fuerza la detención de la plataforma instantáneamente, ya que se actúa directamente sobre la velocidad de las ruedas. Por último, borra de la memoria el objetivo marcado, cambia a modo espera (*READY*) y envía por telemetría su estado.

En el esquema siguiente se visualiza el recorrido del comando 2/stop.

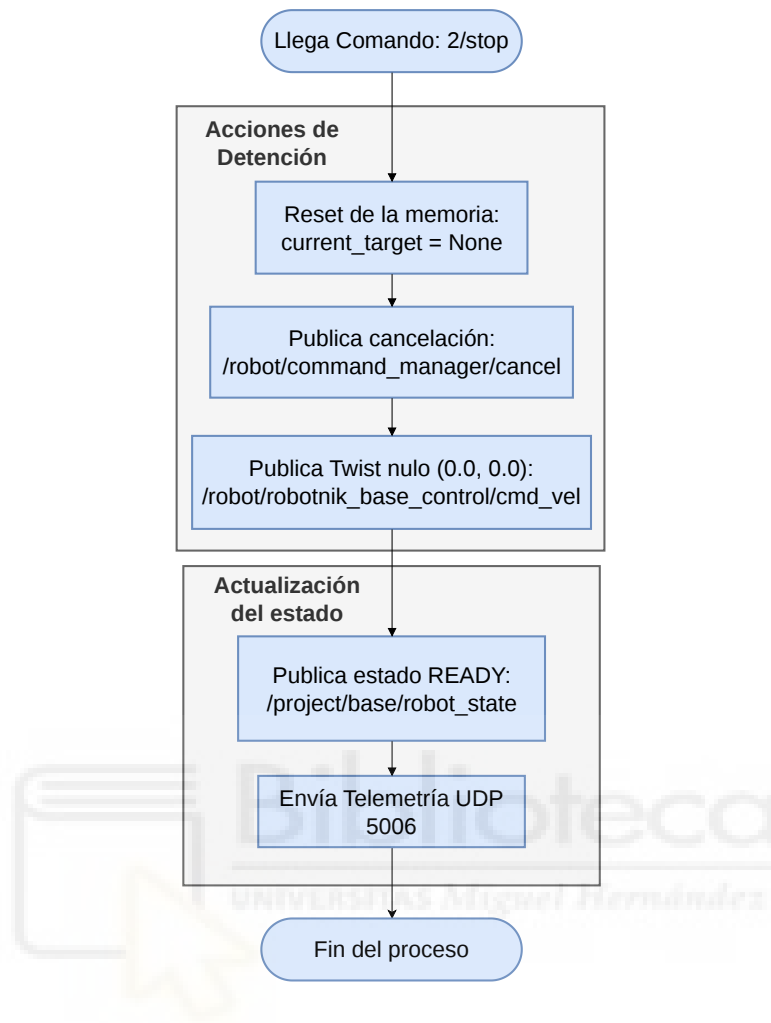


Figura 2.17: Diagrama de flujo del comando 2/stop.

- **Comando 3/acción:** mediante este comando, el robot eleva o baja la plataforma que lleva integrada. Para controlar el elevador, se hace una llamada al servicio `/robot/robotnik_base_control/set_elevator` indicando en qué posición se debe colocar la plataforma (*RAISE*, *LOWER*). Esta llamada se hace a través del comando 3/elevar o 3/bajar que se encarga de traducir a la acción deseada. Cuando el servicio es correcto, se actualiza la variable del elevador y lo envía por telemetría.

En la siguiente imagen se aprecia el diagrama de flujo del comando 3/(posición del elevador).

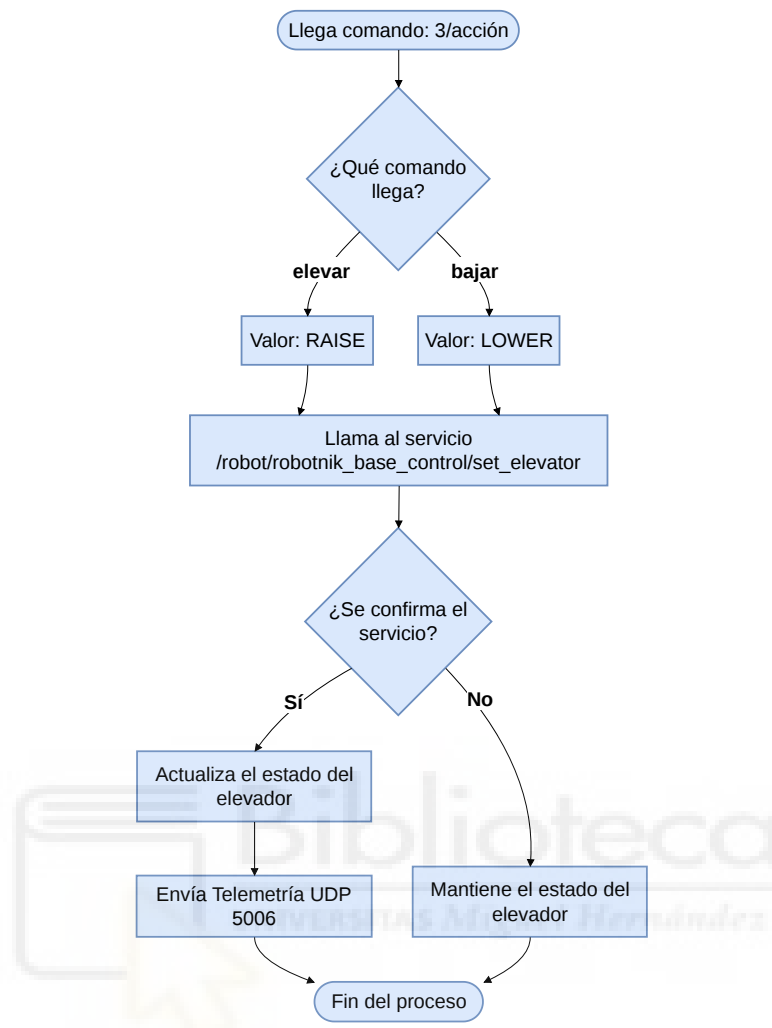


Figura 2.18: Diagrama de flujo del comando 3/acción.

- Comando 4/manual:** este proceso sirve para controlar de manera manual al robot a través de las flechas del teclado. El comando 4/manual no se envía al robot directamente, sino que entra en un bucle en el PC externo que se encarga de leer y enviar las teclas de entrada. Cuando se pulsan las flechas del teclado, el PC externo envía por UDP las velocidades lineales y angulares predefinidas (por ejemplo 4/0.4/0.0), que el nodo *robot_navigator* se encarga de traducir y publicar un mensaje de tipo Twist en el tópico */robot/robotnik_base_control/cmd_vel*. Se puede seguir el flujo del proceso a través del esquema de la imagen 2.19.

Como método de seguridad se han implementado dos acciones:

- Si existe algún objetivo activo, este lo cancela, publicando un mensaje vacío en el tópico */robot/command_manager/cancel*. Esto evita errores

con el planificador de rutas.

2. El robot se detiene de manera instantánea si pasado 0.2 s no se recibe pulsaciones del teclado, ya que el PC emisor envía directamente 4/0.0/0.0 por UDP, lo que provoca que se publique en el tópico *cmd_vel* velocidades nulas.

El modo manual tiene prioridad sobre la navegación autónoma, ya que al recibir las velocidades de manera manual se cancela cualquier objetivo que estuviera en su memoria, y actúa directamente sobre el tópico *cmd_vel*.

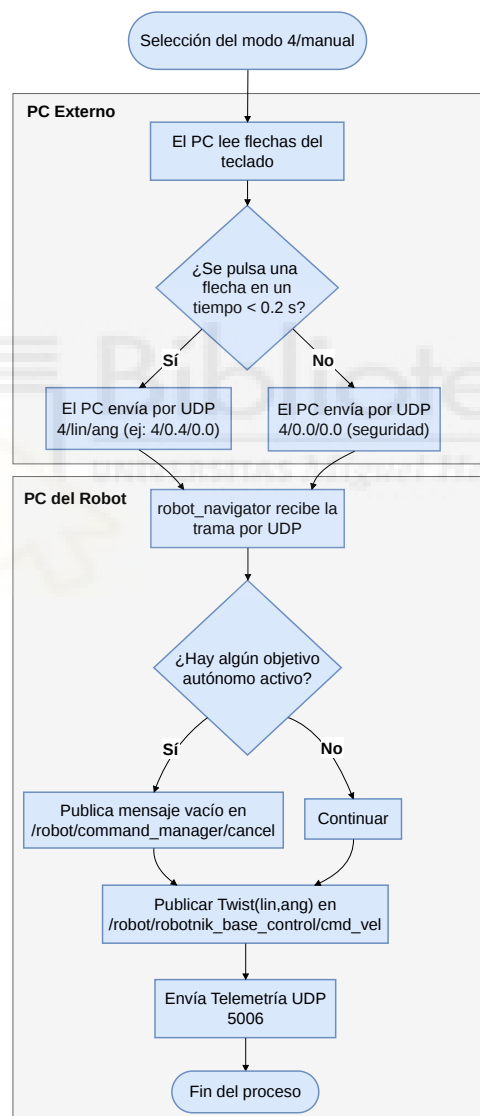


Figura 2.19: Diagrama de flujo del comando 4/manual.

2.4.2.4 Método de Supervisión

La lógica de la supervisión se puede observar en la parte derecha de la Figura 2.16.

El nodo *robot_navigator* permite conocer el estado del robot en todo momento y detectar si este se ha quedado bloqueado en alguna estancia. Esta supervisión se realiza a través de la función *publish_state*, la cual se actualiza mediante un temporizador cada 0,3 segundos.

Esta función realiza dos acciones:

1. **Envía por UDP** la telemetría del robot al PC externo en tiempo real.
2. **Publica en el tópico** */project/base/robot_state* el estado del robot. Esta acción se ha implementado para poder registrar los datos a través de la herramienta *rosvag*.

El estado de la plataforma móvil se evalúa siguiendo los siguientes criterios:

- **ARRIVED (2)**: indica cuando el robot ha llegado a su objetivo marcado de forma autónoma. Se han establecido unos márgenes de llegada debido a la dificultad de posicionarse en las coordenadas exactas. Se confirma la llegada cuando el destino se encuentra a una distancia inferior a 0,25 m y las velocidades lineal y angular son menores a 0,05 m/s. El robot confirma durante 2 segundos que ha llegado marcando el estado ARRIVED, pasado ese tiempo, su estado pasa a READY.
- **MOVING (1)**: marca el estado de movimiento del robot durante el transporte autónomo.
- **BLOCKED (-1)**: este estado indica que el robot se encuentra bloqueado. Esto ocurre cuando tiene un destino marcado en su memoria, pero la velocidad a la que se desplaza es inferior a 0,01 m/s, lo que señala alguna anomalía en el transporte. Si pasados 15 segundos el robot sigue parado, el sistema aborta la misión y limpia la memoria.
- **READY (0)**: el estado 0 garantiza que el robot está listo para que se le asigne una nueva misión, ya que la variable *current_target* se encuentra vacía.

Se ha generado un esquema donde se puede visualizar la máquina de estados implementada:

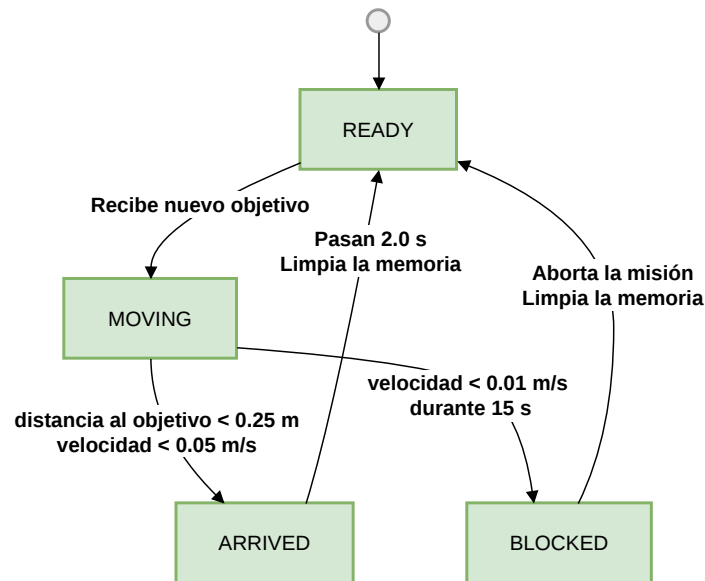


Figura 2.20: Máquina de estados de la rutina de supervisión y control de navegación.

2.4.2.5 Transmisión de Telemetría

Cuando el programa central realiza las acciones mandadas por el usuario, este debe enviar de vuelta al propio usuario el estado del robot. Se crean unas variables que se actualizan constantemente en función de cómo se encuentre el robot, más tarde se empaquetan en una cadena de texto de 8 bloques y se envían por UDP.

Antes de transmitir la telemetría al PC externo, el programa realiza unos cálculos para obtener la distancia y el tiempo estimado de llegada:

- **Estimación simple de la distancia hasta el objetivo:** para saber la distancia que le falta por recorrer al robot hasta llegar a su objetivo, el programa compara las coordenadas actuales del robot con las coordenadas del destino marcado. La diferencia entre ambas marca la distancia en metros que falta tanto en el eje X (dx) como en el eje Y (dy). De este modo, se conoce la distancia en línea recta que falta para llegar al punto marcado, como se aprecia en a figura siguiente.

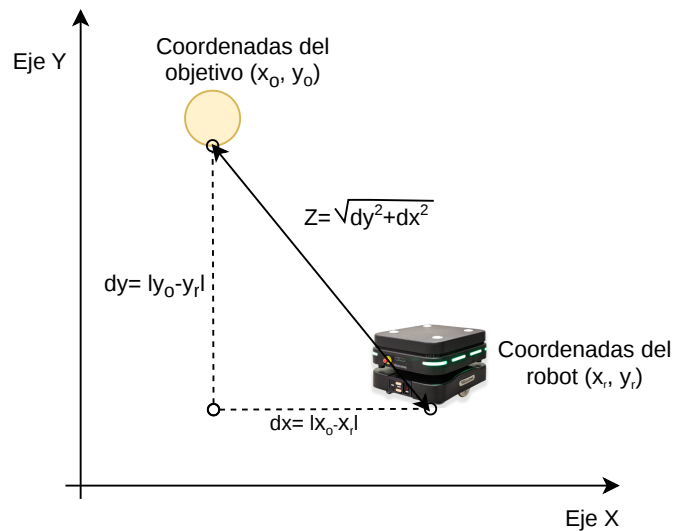


Figura 2.21: Cálculo mediante el Teorema de Pitágoras.

- **Cálculo del tiempo de llegada:** una vez que se ha calculado la distancia hasta el objetivo, mediante la fórmula de $\text{Tiempo} = \text{Distancia} / \text{Velocidad}$, se puede hallar una estimación del tiempo en segundos que le falta al robot para llegar a las coordenadas marcadas. Como método de seguridad en el cálculo, se ha implementado el sistema asuma una velocidad de 0,05 m/s en el momento que el robot se encuentra parado. Así se evita que el sistema divida por cero. Además, se usa siempre la velocidad lineal en valor absoluto, evitando dar tiempos en negativo.
- **Limpieza de decimales:** para evitar visualizar un gran número de decimales, se ha optado por redondear a tres decimales en la posición y la velocidad, y el tiempo restante a un decimal.

Una vez calculados los parámetros, el programa junta los datos en una sola línea y los separa mediante barras (/). Se obtiene como resultado una cadena de texto de 8 bloques (*string*): PX / PY / VL / VA / DIST / TIME / STATUS / ELEVATOR.

Donde cada valor significa lo siguiente:

- **PX/PY:** coordenadas actuales del robot en el eje X e Y.
- **VI/VA:** indica la velocidad lineal y angular a la que se desplaza el robot.
- **DIST:** distancia que le falta al robot para llegar al objetivo en metros.

- **TIME**: tiempo estimado de llegada en segundos.
- **STATUS**: estado en el que se encuentra el robot (READY, MOVING, ARRIVED o BLOCKED)
- **ELEVATOR**: indica la posición del elevador.

Una vez se haya empaquetado todo en la cadena de texto, se envía por UDP para que lo reciba el PC de control.

2.5. Software de Comprobación

El nodo de control *robot_navigator*, está diseñado para ser controlado a través de unas gafas de realidad aumentada, por lo que ha sido necesario desarrollar unos módulos externos que permitan comprobar el funcionamiento del programa central, sin la necesidad de hacer uso de este equipo.

Se han desarrollado dos módulos que se implementan en el PC de control, estos simulan el envío de comandos a través de las gafas.

2.5.1. Módulo Emisor

El programa *pc_controller_sender* se encarga de enviar las órdenes externas del usuario. Su objetivo es simular el envío y recepción de comandos a través del puerto UDP y comprobar que el robot los recibe correctamente realizando las acciones deseadas. Se aprecia en la Figura 2.22 el flujo que sigue este programa.

El módulo realiza tres acciones principales:

1. **Comunicación UDP**: el programa crea un puerto UDP (*socket*) por el que envía la cadena de comandos al robot, para que este los procese. Esto evita que el sistema dependa de ROS para el envío y recepción de información, ya que las gafas trabajan bajo el entorno de Windows.
2. **Menú desplegable de comandos**: al iniciar el programa, se muestra en la terminal un menú con los posibles comandos de envío. Además, es un menú interactivo, lo que permite enviar de forma directa las acciones en vez de cargar el programa cada vez que se quiera enviar un comando nuevo.

3. **Control manual por teclado:** como se comentó en el punto anterior, el comando 4/manual, no se envía directamente al robot, sino que entra en un bucle en el PC que cambia la terminal a modo RAW (envía directamente el comando sin la necesidad de pulsar la tecla Enter). Este bucle lee las flechas del teclado y las convierte en velocidades predefinidas, enviando el comando 4/vel lin/vel ang. Para conseguir que el método de control funcione como un mando de control donde si dejas de mover el joystick el robot se para, se ha implementado un método de parada forzada. Este método consiste en que, si el sistema no detecta ninguna pulsación de las flechas del teclado, espera 0,2 segundo y envía el comando 4/0.0/0.0, lo que fuerza la parada del robot. Con esto se consigue controlar el movimiento del robot como se desea.

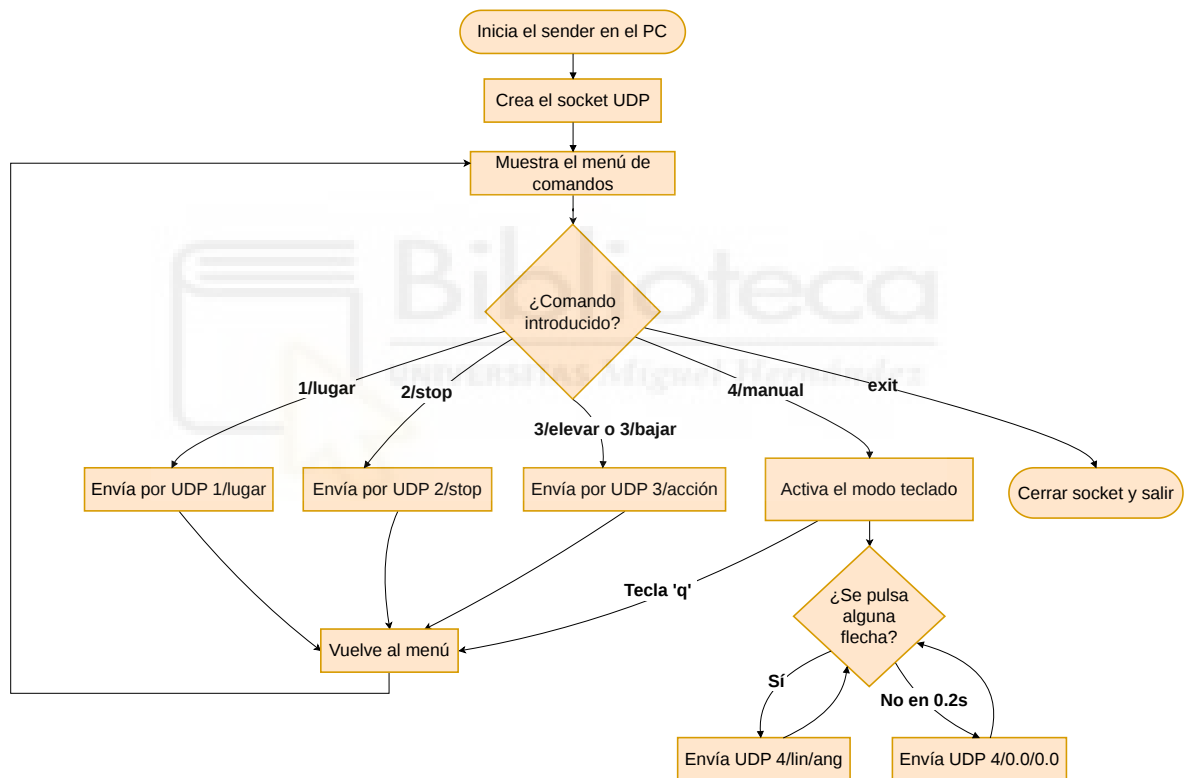


Figura 2.22: Diagrama de flujo del módulo emisor (Sender).

2.5.2. Módulo Visualizador

El programa *pc_controller_receiver* se ha desarrollado para comprobar que el funcionamiento del robot es el correcto. Este recibe en tiempo real una cadena de texto que envía el nodo central *robot_navigator*, con las características importantes del robot. El objetivo es verificar si cumple con las funciones programadas. En la imagen 2.23 se aprecia el flujo del módulo visualizador.

Las características de este nodo son las siguientes:

1. **Puerto UDP de entrada:** crea una puerta de recepción de datos a través de un socket UDP. El programa escucha los paquetes que envía el robot a través de la red Wi-Fi, lo que permite recibir la información sin depender de ROS.
2. **Decodificación de la trama:** el programa recibe una cadena de texto de 8 bloques divididos por barras (/). El sistema se asegura de que lleguen los 8 elementos y luego los separa para poder tratar con ellos. Los campos son los siguientes: PX / PY / VL / VA / DIST / TIME / STATUS / ELEVATOR.
3. **Visualización por consola:** el programa imprime por consola el estado del robot para que el usuario pueda comprobar si la información recibida es la adecuada.

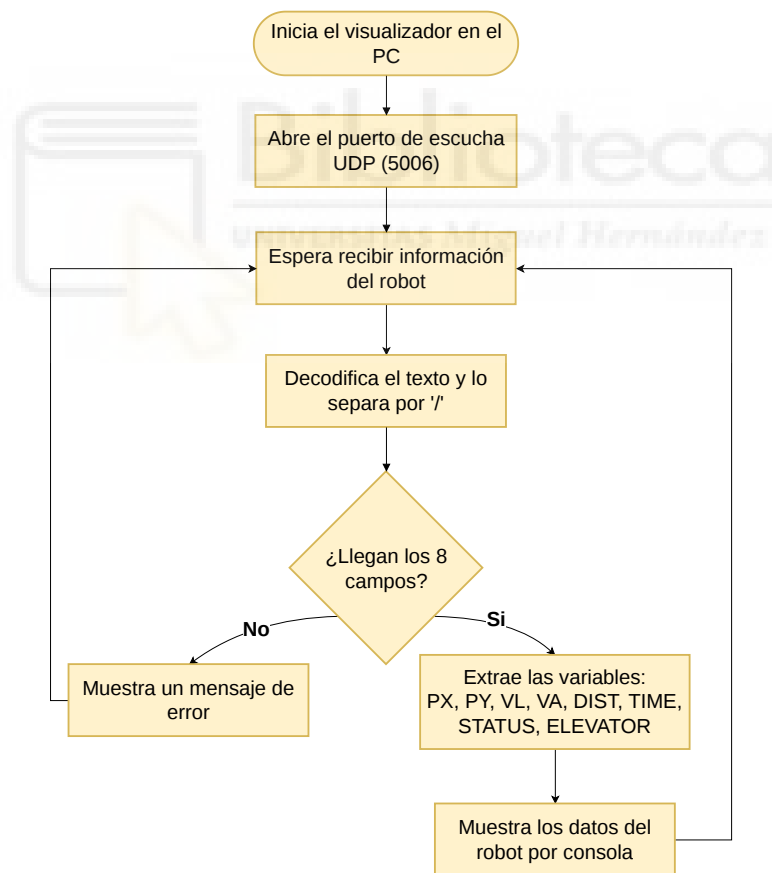


Figura 2.23: Diagrama de flujo del módulo visualizador (Receiver).

2.6. Adquisición de Datos y Herramientas de Análisis

Para poder comprobar que el funcionamiento del sistema es correcto, es necesario realizar estudios donde se registren los datos y posteriormente evaluarlos para asegurar que el robot cumple con los parámetros previstos.

En este proyecto, se han utilizado dos herramientas para esta comprobación, la primera se trata de una herramienta que ofrece ROS para poder grabar las señales mientras el robot realiza las acciones ordenadas. La segunda herramienta que se ha utiliza es el software MATLAB para interpretar los datos mediante graficas.

Se puede observar un esquema del flujo que se ha seguido para obtener las graficas de los resultados.

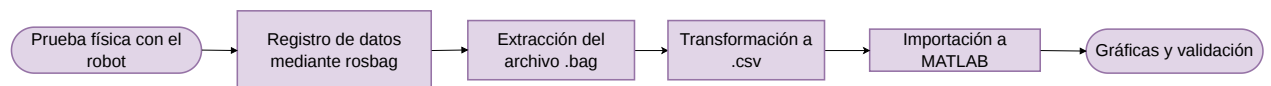


Figura 2.24: Flujo de adquisición y procesamiento de datos del proyecto.

2.6.1. Registro de datos mediante rosbag

Para registrar las pruebas a tiempo real, se ha utilizado la extensión rosbag que implementa ROS. Esta herramienta permite grabar los datos que circulan por los tópicos de ROS mientras el robot está en funcionamiento. La herramienta rosbag guarda los datos que se publican en el tópico seleccionado y los exporta en un archivo .bag para que puedan ser interpretados posteriormente mediante software gráfico.

Los elementos utilizados para la comprobación del funcionamiento son los siguientes:

1. **Odometría y Velocidad:** el rosbag se suscribe al tópico de `/robot/throttle/odom` para capturar la velocidad lineal y angular a la que se desplaza el robot.
2. **Posicionamiento:** se graban las coordenadas actuales del robot mediante el tópico `/robot/amcl_pose`.
3. **Estado del Sistema:** gracias a la creación del tópico `/project/base/robot.state`, la herramienta rosbag es capaz de capturar el estado en el que se encuentra el robot en cada momento (READY, MOVING, ARRIVED, BLOCKED).

2.6.2. Herramienta de Cálculo y Visualización Gráfica (MATLAB)

Una vez se han obtenido las grabaciones del rosbag, es necesario transformar esos datos en gráficas donde se pueda evaluar el comportamiento del robot ante diferentes situaciones. Para poder generar estas gráficas, se ha utilizado el software MATLAB.

El proceso consiste en transformar el archivo .bag, previamente generado durante los ensayos en ROS, a un formato .csv, el cual facilita la importación de los datos a MATLAB. Mediante esta conversión se generan los cálculos y las gráficas necesarias para comprobar el funcionamiento del robot.



3. Resultados y Discusión

Este capítulo presenta la validación experimental de la arquitectura diseñada en ROS para el control del robot. El objetivo de este apartado es analizar los resultados obtenidos en las diferentes pruebas experimentales para poder confirmar que la plataforma cumple con los requisitos de diseño. A través de las herramientas de análisis expuestas en el apartado 2.6 (el entorno MATLAB junto con la herramienta de registro rosbag), se aportan gráficas y tablas que demuestran la precisión de la navegación, el cumplimiento de los límites de seguridad y la fiabilidad del control manual.

Todas las pruebas que se exponen a continuación se han realizado de manera real sobre la plataforma móvil, en un entorno controlado para poder crear los escenarios adecuados para cada experimento. Se ha prescindido de estudios simulados ya que se tenía la posibilidad de realizar las pruebas de manera real, aumentando la fiabilidad de los resultados en entornos reales. Para evaluar el sistema de manera correcta, se han planteado ensayos donde se obliga al robot a navegar sobre los diferentes puntos de interés, de esta manera verificamos el funcionamiento y calculamos los errores durante el transporte.

3.1. Navegación autónoma

En este punto se valida el funcionamiento del robot cuando opera de manera autónoma. Los objetivos de este apartado son los siguientes:

1. Comprobar con datos reales que el robot se desplaza de manera autónoma al punto marcado.
2. Verifica que se respeta la velocidad máxima impuesta para mayor seguridad.

3.1.1. Validación de la trayectoria de forma autónoma

Para evaluar el sistema de navegación, se ha realizado una prueba que consiste en recorrer una secuencia de destinos predefinidos de forma autónoma. La ruta programada es la siguiente: Home - Mechanics - Electronics - Mechanics - Home. El ensayo se ha realizado en el Instituto de Bioingeniería, donde cada destino corresponde a un punto real del laboratorio. En la Figura 3.1 se aprecia el plano del pasillo del instituto junto a la ubicación de los puntos de interés.

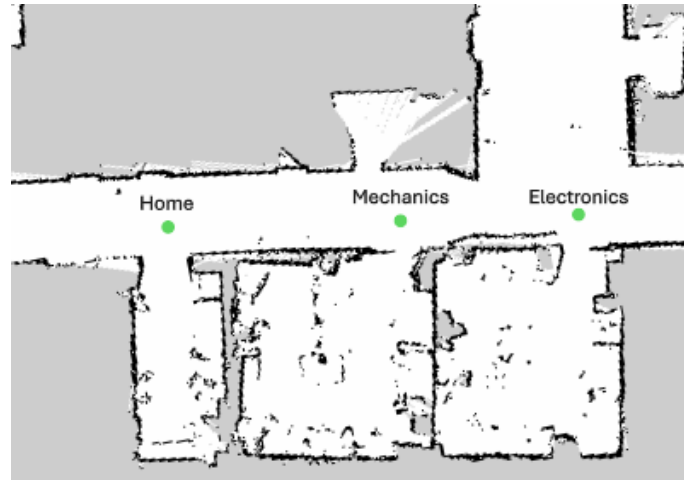


Figura 3.1: Mapa del pasillo del Instituto de Bioingeniería con los puntos de interés marcados.

Para realizar el ensayo, se han extraído los datos del tópico `/robot/amcl_pose` de ROS. Mediante esta información, que indica la localización espacial del robot, se ha generado una gráfica que representa las coordenadas del robot en el mapa con respecto a los ejes X e Y.

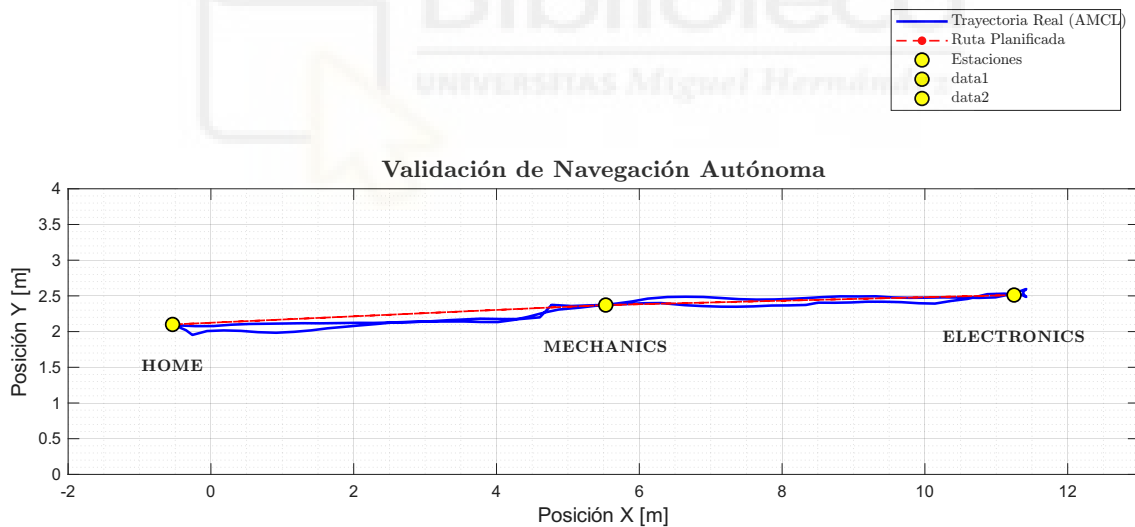


Figura 3.2: Trayectoria real estimada frente a la ruta planificada.

Como se puede observar en la Figura 3.2, la línea roja discontinua representa la ruta óptima que debe seguir el robot, mientras que la línea azul muestra la trayectoria real que ha ejecutado la plataforma. La ruta realizada sigue correctamente a la planificada, presentando pequeñas desviaciones debido al intento de mantenerse dentro de los límites marcados por el sensor.

Para comprobar que el funcionamiento de la plataforma es el adecuado, se ha ge-

nerado otra gráfica que muestra la evolución de las coordenadas por separado en función del tiempo, de manera que se aprecia si el robot pasa por las coordenadas marcadas (Figura 3.3).

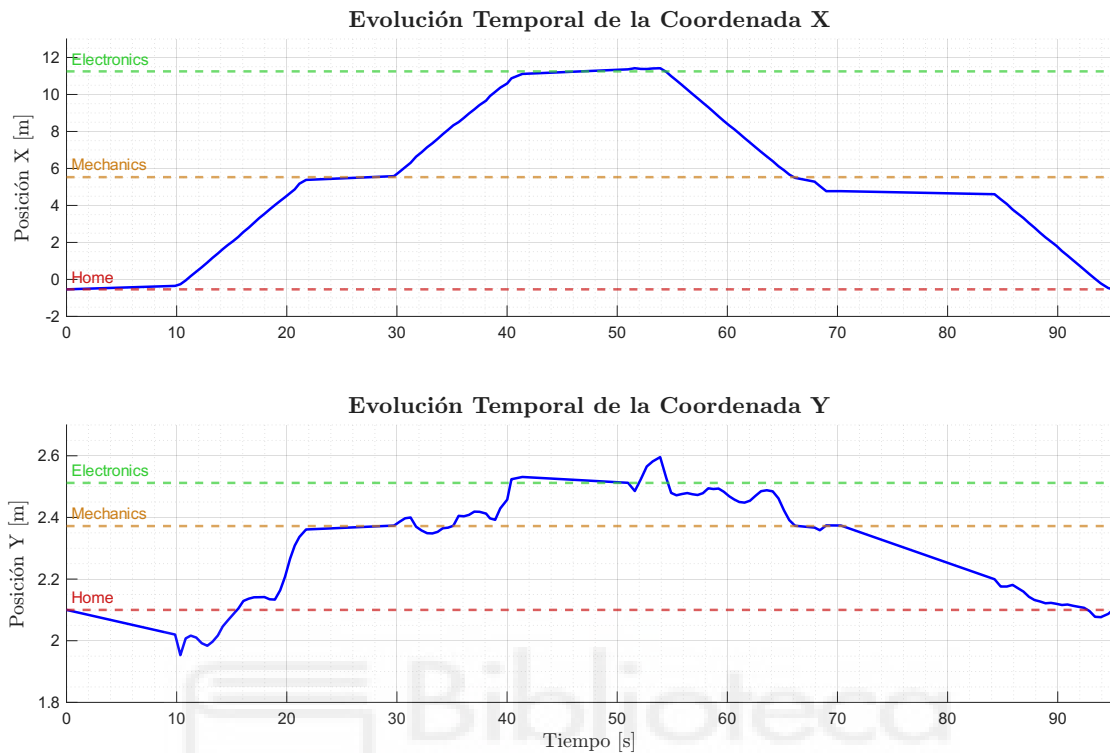


Figura 3.3: Evolución temporal de las coordenadas X e Y durante la misión.

La Figura 3.3 muestra la evolución temporal del desplazamiento en los ejes X e Y en función del tiempo. La gráfica superior muestra cómo la trayectoria pasa por los puntos marcados del eje X de manera precisa, ya que la ruta real del robot frena exactamente en los destinos marcados. Siguiendo las líneas discontinuas se aprecia cómo el error de llegada es pequeño por lo que se puede concluir que la navegación autónoma es precisa.

3.1.2. Comprobación de las restricciones de velocidad

Para comprobar que el robot no supera la velocidad límite preestablecida y que cumple con las normas especificadas de seguridad, se ha generado una gráfica con la velocidad a la que se desplaza el robot durante su tarea de forma autónoma. En la Figura 3.4 se representa con una línea azul la velocidad lineal a la que se desplaza el robot, obtenida a través del tópico de odometría `/robot/throttle/odom`. Por otro lado, se marca con una línea roja discontinua la velocidad límite programada en 0.5 m/s.

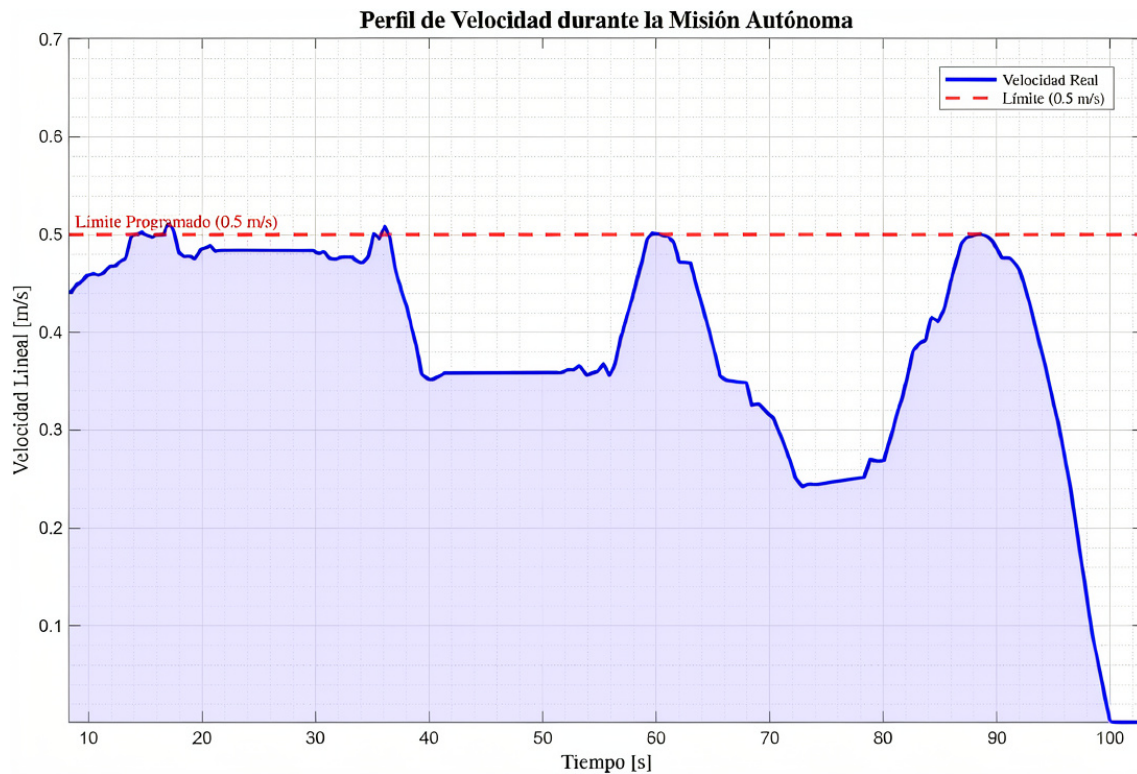


Figura 3.4: Validación de la restricción de velocidad en modo autónomo.

En esta gráfica se pueden observar dos comportamientos que garantizan un transporte de forma segura:

1. **Límite de velocidad respetado:** en la gráfica se aprecia cómo la línea azul llega al tope máximo de los 0.5 m/s programados. Se visualizan unos sobreimpulsos en la señal causados por la inercia del robot. Para comprobar el sobrepaso exacto, se ha usado la función *max* de MATLAB para hallar el valor máximo de la función, siendo este valor de 0.51 m/s, lo que indica un 2% de sobreimpulso con respecto a la consigna máxima. Esto demuestra que el error de velocidad que sobrepasa los límites es bajo, por lo que se concluye que se respetan los límites de velocidad.
2. **Aceleración y frenado suave:** se puede observar cómo las pendientes de aceleración y frenado son suaves, lo que permite mayor estabilidad a la plataforma, evitando cambios bruscos. Para demostrar que la aceleración de subida y de bajada cumple con los límites preestablecidos (0.1 m/s^2), se ha calculado la aceleración de la plataforma en cada instante de tiempo mediante la fórmula de la derivada de la velocidad respecto del tiempo ($a = \frac{dv}{dt}$), obteniendo una aceleración máxima positiva de 0.058 m/s^2 , y una deceleración máxima negativa de -0.067 m/s^2 . Ambos valores se encuentran por debajo del límite

de 0.1 m/s^2 . Este límite se ha establecido de forma empírica tras realizar numerosas pruebas en el laboratorio. Se ha concluido que la aceleración de 0.1 m/s^2 desplaza de manera segura al usuario.

3.2. Seguridad en entornos dinámicos

En este apartado se evalúa la capacidad del robot para operar de manera segura y fiable en entornos con obstáculos. Se valida la respuesta ante obstáculos inesperados, comprobando que los imprevistos no afectan a la actividad del sistema.

3.2.1. Evasión de obstáculos

Para comprobar la seguridad de la plataforma y del entorno, se ha realizado la misma prueba que en el apartado 3.1.1, pero posicionando varios obstáculos en la trayectoria que debe seguir el robot. De esta manera, se puede comprobar si la base móvil es capaz de evitar los impedimentos y continuar con la trayectoria marcada de manera estable sin perder la referencia. En la Figura 3.5 se muestra la posición de los obstáculos en la zona de pruebas.



Figura 3.5: Foto de la disposición de los obstáculos en el pasillo.

De la misma manera que en la prueba 3.1.1, se han guardado nuevamente las posiciones X e Y extraídas del tópico `/robot/amcl_pose`. La ruta seguida es la siguiente: Home - Mechanics - Electronics - Home. En este caso, se ha obviado el paso por

Mechanics en la fase de vuelta para comprobar si el robot es capaz de realizar una trayectoria de mayor distancia esquivando los obstáculos sin perder la referencia (Figura 3.4).

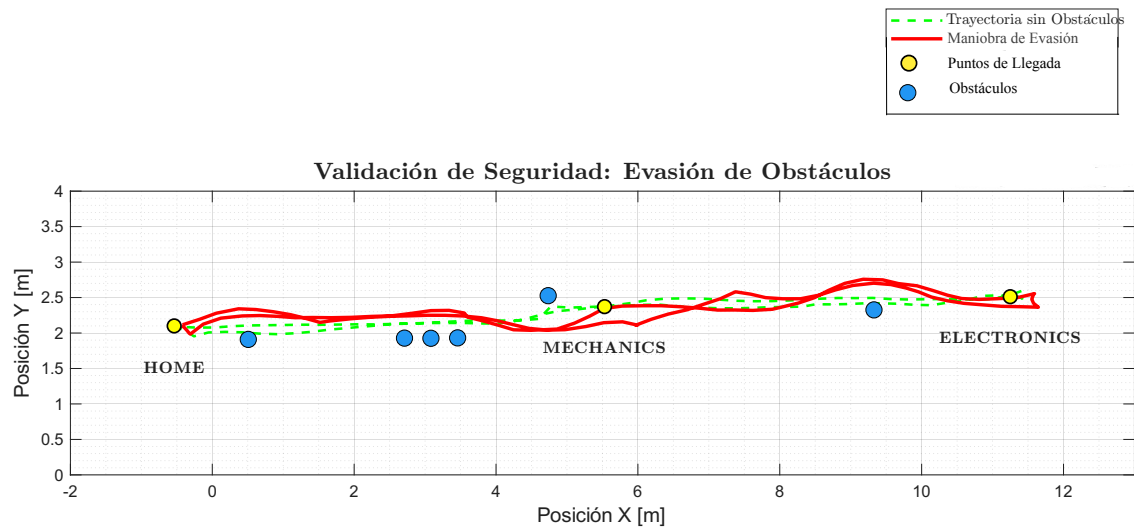


Figura 3.6: Validación de la trayectoria ante obstáculos.

Al analizar la Figura 3.6, se aprecia cómo el sistema detecta los obstáculos (marcados con puntos azules en la gráfica) y modifica la ruta original (línea verde), ejecutando la trayectoria real de evasión (línea roja) para evitar la colisión. Cabe destacar en esta imagen cómo el robot es capaz de llegar a su destino mediante diferentes maniobras, independientemente de los obstáculos que se encuentre en el camino. El sistema trata de seguir la ruta óptima en todo momento.

3.2.2. Robustez y repetibilidad del sistema

Para evaluar la repetibilidad y estabilidad del sistema de navegación, se ha cronometrado en cinco ocasiones distintas el trayecto Home - Mechanics, registrando además las incidencias ocurridas durante la trayectoria. A partir de estos tiempos, se ha calculado la variabilidad relativa para comprobar si el rendimiento de la plataforma es correcto o varía mucho si se encuentra con obstáculos.

Tabla 3.1: Tiempos de ejecución y cálculo de desviaciones.

Pruebas	Tiempo (x_i)	Desviación ($x_i - \bar{x}$)	Cuadrado ($(x_i - \bar{x})^2$)	Incidenias
1	14,96 s	+0,29 s	0,0864	Evasión de 1 persona
2	14,36 s	-0,31 s	0,0936	Trayectoria libre
3	14,55 s	-0,12 s	0,0135	Trayectoria libre
4	15,03 s	+0,36 s	0,1325	Evasión de 2 personas
5	14,43 s	-0,24 s	0,0557	Trayectoria libre
SUMA	73,33 s	-	0,3817	-

Una vez calculados estos valores, se obtienen los siguientes resultados:

1. **Media aritmética ($\bar{x}$):**

$$\frac{73,33}{5} = 14,67 \text{ s}$$

2. **Varianza muestral (S^2):**

$$\frac{\sum (x_i - \bar{x})^2}{N - 1} = \frac{0,3817}{4} = 0,0954$$

3. **Desviación típica (S):**

$$\sqrt{S^2} = 0,31 \text{ s}$$

4. **Variabilidad relativa (Coeficiente de variación):**

$$CV = \frac{S}{\bar{x}} \cdot 100 = 2,1 \%$$

Tras estos resultados, se comprueba que los tiempos de ejecución son similares entre sí a pesar de las perturbaciones externas. Se observa cómo la diferencia entre una trayectoria libre (14,36 s) y una donde debe esquivar a dos personas (15,03 s) es inferior a un segundo, por lo que la adaptación de la trayectoria es óptima. Se ha obtenido una variabilidad relativa de 2,1%, lo que indica que el robot es capaz de esquivar obstáculos y adaptar su trayectoria de forma eficiente, sin detener el vehículo de manera innecesaria.

3.3. Validación del modo manual

En este último bloque se comprueba el funcionamiento del sistema bajo el control manual. El objetivo es confirmar que el control manual responde correctamente y en tiempo real a las órdenes enviadas por el usuario de manera externa, garantizando una conducción suave y precisa.

3.3.1. Respuesta del robot frente a la consigna enviadas

Para validar el control del sistema, se ha analizado la respuesta temporal del robot frente a una entrada de tipo escalón. En la Figura 3.7 se comparan dos señales, la de color azul hace referencia a la consigna enviada (es decir, la velocidad que el usuario le indica que debe mantener), mientras que la señal de color rojo muestra la respuesta real de las ruedas.

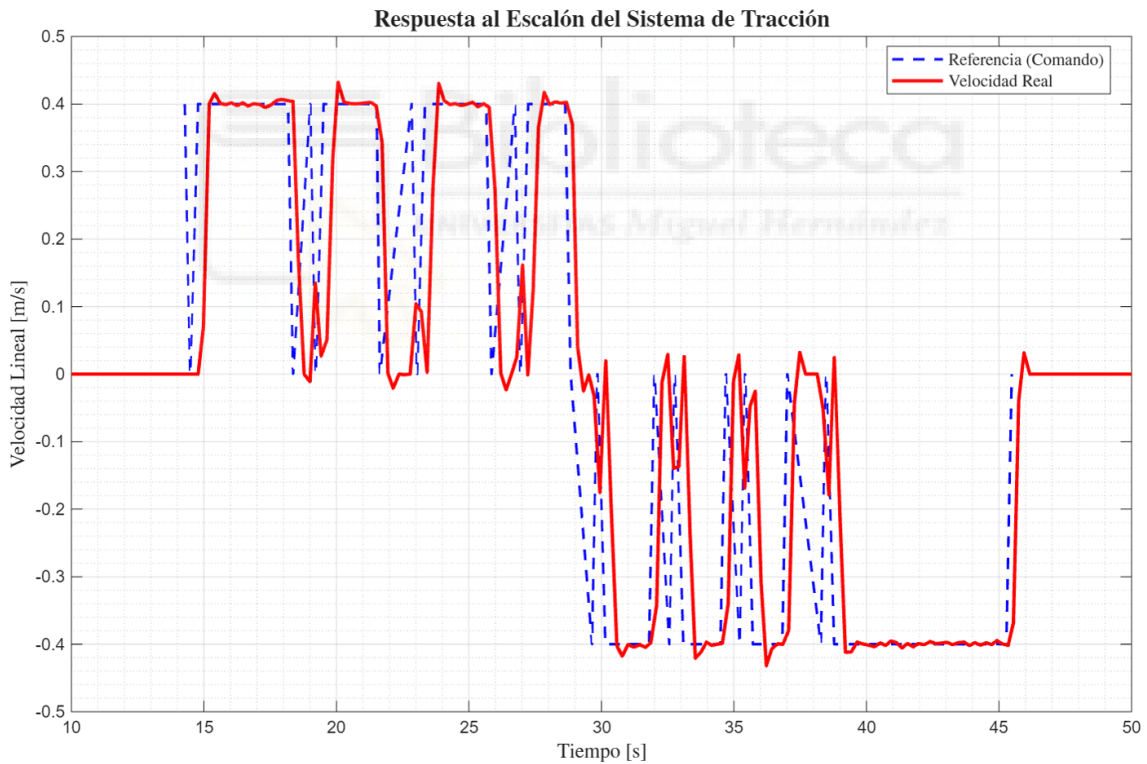


Figura 3.7: Respuesta del sistema ante una entrada de tipo escalón.

La línea azul representa la señal enviada por el teleoperador a través del tópico `/robot/robotnik_base_control/cmd_vel`, que se encarga de imponer 0,4 m/s al sistema. La línea roja hace referencia a la respuesta real a la que se desplazan las ruedas, extraída del tópico `/robot/throttle/odom`.

Se obtienen las siguientes conclusiones de la gráfica:

1. **Seguimiento de la trayectoria:** el sistema es capaz de seguir la referencia enviada por el usuario de manera rápida y sin sobreimpulsos significativos. Para comprobar numéricamente si el rendimiento es correcto, se ha calculado la diferencia de tiempos desde que el usuario emite la orden, hasta que el robot llega a la consigna de 0,4 m/s. Se ha obtenido un tiempo de respuesta de 0,424 s, lo que supone un tiempo de reacción notable. Además, se ha calculado el error de seguimiento en régimen permanente restando la velocidad real frente a la consigna enviada y calculando la media de los tramos más estables. Se ha obtenido una desviación media de 0,0003 m/s, lo que demuestra que el control de velocidad es preciso.
2. **Velocidad máxima de control:** se observa cómo el robot se desplaza a una velocidad de 0,4 m/s, siendo esta la consigna de velocidad que se le ha ordenado alcanzar. Esta información se obtiene del tópico `/robot/robotnik_base_control/cmd_vel`.
3. **Estabilidad ante perturbaciones:** existen pequeñas oscilaciones en la velocidad lineal al tratar de seguir la señal ideal. Esto representa el comportamiento normal de un sistema físico, donde el control PID trata de mantener la velocidad lo más cercana a la consigna, compensando en tiempo real las irregularidades del terreno, la fricción del suelo y la propia inercia del robot.
4. **Seguridad en la parada:** a diferencia de las señales teóricas, el robot no puede detenerse de forma instantánea debido a su propia inercia. De la misma forma que en el punto 1, se ha calculado el tiempo que tarda en detenerse el robot desde que se envía la consigna de 0 m/s hasta que las ruedas realmente se detienen, siendo este valor de 0,417 s. Esta respuesta garantiza una parada segura y rápida.

3.3.2. Análisis del error en régimen permanente

Para comprobar la precisión del robot, se ha calculado el error en régimen permanente de la velocidad, que se define como la diferencia entre la velocidad de referencia (consigna) y la velocidad real a la que se desplazan las ruedas.

Se ha generado una gráfica que muestra la evolución temporal de este error durante toda la prueba en los puntos más estables (Figura 3.8).

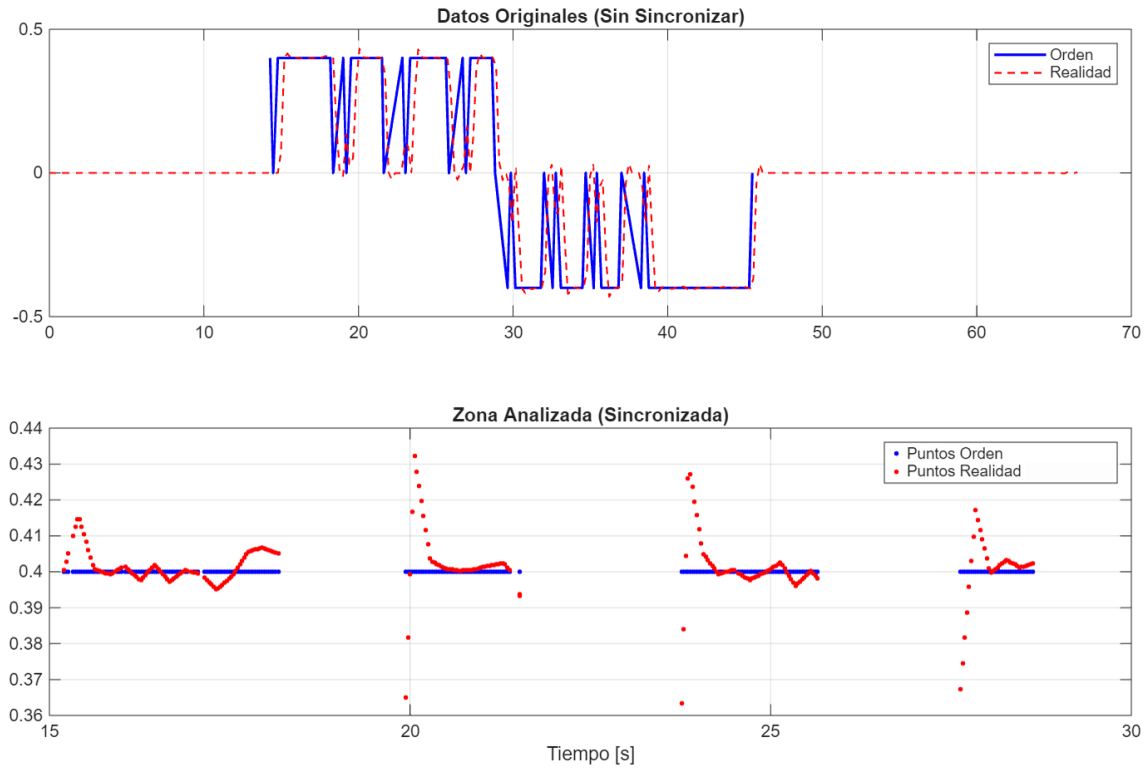


Figura 3.8: Evolución temporal del error de velocidad.

En la Figura 3.6 se aprecia el análisis de velocidad dividido en dos partes. En la gráfica superior se muestran los datos originales sin sincronizar obtenidos durante todo el ensayo. En la gráfica inferior se han aislado y sincronizado los datos de una zona estable para analizarla con más detalle. En esta segunda gráfica se puede observar cómo existen dos fases diferentes. La primera corresponde a los picos de sobreoscilación producidos durante el régimen transitorio. Estos son causados por la inercia del robot y el tiempo de respuesta de los motores. La segunda fase corresponde al régimen permanente, donde el robot estabiliza su velocidad en los 0,4 m/s marcados.

Para garantizar unos datos fiables, se han descartado los instantes de arranque y parada, siendo estos la fase transitoria del sistema. Se han tomado únicamente los datos cuando el robot se encuentra en régimen permanente, a una velocidad constante de 0,4 m/s. Se han registrado 239 muestras, calculando el error en cada instante (k) mediante la siguiente expresión:

$$e[k] = \text{Velocidad de Referencia}[k] - \text{Velocidad Real}[k]$$

La siguiente tabla muestra los cálculos de los primeros datos de la prueba:

Tabla 3.2: Muestra de datos y cálculo del error instantáneo.

Muestra (k)	Tiempo (s)	V_{ref} (Pedida)	V_{real} (Medida)	Error ($e[k]$)	Error Cuadrático ($e^2[k]$)
1	20,10	0,4000	0,4015	-0,0015	$2,25 \cdot 10^{-6}$
2	20,20	0,4000	0,4021	-0,0021	$4,41 \cdot 10^{-6}$
3	20,30	0,4000	0,3998	0,0002	$0,04 \cdot 10^{-6}$
...	...	...	...	...	...
Suma Total ($N = 239$)	-	-	-	-0,3824	0,014916

Con los valores calculados en esta tabla, se pueden hallar los tres indicadores que definen el rendimiento del sistema.

1. **Error Medio:**

$$\bar{e} = \frac{\sum e[k]}{N} = \frac{-0,3824}{239} = -0,0016 \text{ m/s}$$

Este cálculo indica que el error medio de velocidad es pequeño, prácticamente un error de 1,6 mm/s. El signo negativo señala que el robot va más rápido de lo que realmente debe ir. Este tipo de error es normal en sistemas físicos, siendo consecuencia de la inercia de la plataforma móvil durante su desplazamiento.

2. **Raíz del error cuadrático medio (RMSE):**

$$RMSE = \sqrt{\frac{\sum e^2[k]}{N}} = \sqrt{\frac{0,014916}{239}} = 0,0079 \text{ m/s}$$

El cálculo del RMSE evalúa la dispersión del error, penalizando las variaciones grandes. Al obtener un valor de 0,0079 m/s, significa que no existen variaciones bruscas y que el movimiento del robot es suave y estable.

3. **Error relativo en régimen permanente (E %):**

$$\epsilon \% = \left| \frac{\bar{e}}{v_{\text{ref}}} \right| \times 100 = \left| \frac{-0,0016}{0,4000} \right| \times 100 = 0,39 \%$$

El sistema de control presenta un error relativo en régimen permanente de 0,39 %. Esto confirma que el sistema sigue la consigna enviada de manera robusta, con un error inferior al 1 %. Teniendo en cuenta que los márgenes de

error permitidos en la industria rondan el 5 %, se demuestra que este sistema se encuentra por debajo de los límites estándares.



4. Conclusiones y Trabajos Futuros

En este último capítulo se exponen las conclusiones del proyecto tras realizar varios ensayos al sistema, donde se evalúan los objetivos propuestos inicialmente. En segundo lugar, se presentan las posibles líneas de trabajo futuras, las cuales se centran en mejorar la autonomía y las capacidades del sistema.

4.1. Conclusiones

Tras el desarrollo y el estudio de este Trabajo de Fin de Grado, se concluye que el proyecto ha cumplido con el objetivo principal de facilitar el desplazamiento autónomo a personas con discapacidades motoras.

El proyecto se ha centrado en el diseño de una arquitectura de control basada en ROS Noetic para la plataforma industrial RB-THERON. La comunicación entre dispositivos se ha realizado mediante el protocolo UDP. Esto ha permitido que el robot sea independiente de un sistema operativo específico, lo que hace posible su control desde cualquier dispositivo externo de manera flexible.

Los ensayos realizados en entornos reales han verificado la fiabilidad y robustez del sistema, destacando los siguientes resultados:

1. Se ha demostrado que el robot es capaz de ejecutar una navegación autónoma segura, completando las rutas asignadas y esquivando obstáculos sin interrumpir la trayectoria. A su vez, el sistema controla de manera óptima la velocidad máxima impuesta, sin sobrepasar los límites.
2. Mediante ensayos empíricos, se ha probado que el sistema tiene una repetibilidad adecuada, ya que mantiene tiempos de ejecución similares en tareas repetitivas a pesar de los obstáculos del entorno.
3. El control manual presenta un error en régimen permanente del 0,39 %, lo que prueba que la señal real sigue de manera fiel la consigna enviada.

Como aspecto adicional, durante el desarrollo del proyecto, se han identificado una serie de limitaciones. En primer lugar, el sistema requiere establecer manualmente la posición inicial del robot (*initial pose*), lo que obliga a encender la base móvil siempre en el mismo punto al no contar con un método de autolocalización inicial. En segundo lugar, el robot carece de una rutina de retorno automático hacia la base

de carga una vez finalizado su propósito o al detectar batería baja, siendo necesaria la intervención humana para la recarga del vehículo.

4.2. Trabajos Futuros

En las líneas futuras de este proyecto se pretende investigar cuatro puntos clave que no se han abordado durante la realización de este TFG:

1. **Retorno automático a la estación de carga:** con el fin de desarrollar un sistema completamente autónomo, se plantea la implementación de un algoritmo capaz de guiar la base móvil a su estación de carga cuando detecte niveles bajos de batería o haya finalizado su misión. Esto asegurará una mayor autonomía del robot y eliminará la necesidad de intervención humana para cargarlo.
2. **Implementación de un algoritmo de autolocalización:** como se ha comentado en el apartado anterior, el sistema cuenta con una limitación de autolocalización inicial. Por ello, en una línea futura se pretende integrar un algoritmo que permita al dispositivo determinar sus coordenadas exactas de manera autónoma al encenderse, sin la necesidad de establecer manualmente la posición inicial en el arranque.
3. **Implementación de un SLAM dinámico:** como mejora en el sistema de navegación, se propone implementar un algoritmo capaz de realizar un SLAM dinámico de manera continua. El objetivo es que el robot actualice su mapa interno en tiempo real durante cada tarea, permitiendo que el sistema se adapte a cambios estructurales del entorno sin la necesidad de volver a generar un mapa estático mediante *gmapping*.
4. **Mejora de la estimación del tiempo y distancia:** en la versión actual del proyecto, el cálculo de estimación del tiempo y distancia de llegada al objetivo es simple, debido a que solo calcula el desplazamiento en línea recta, sin tener en cuenta las limitaciones físicas del entorno como paredes o puertas. En un futuro, se pretende integrar un algoritmo que base su estimación en la ruta real generada por el planificador global, de este modo, las estimaciones serán más precisas.

BIBLIOGRAFÍA

- [1] A. Gupta, K. R. Chowdhury, N. Farheen y M. P. Schoen, «Evolution and Emerging Trends in Intelligent Wheelchair Control: A Comprehensive Review,» *Machines*, vol. 14, n.º 1, pág. 33, 2026.
- [2] C. S. Ahuja, V. Nori, L. Tetreault et al., «Traumatic spinal cord injury,» *Nature Reviews Disease Primers*, vol. 3, pág. 17018, 2017.
- [3] M. C. Kiernan, S. Vucic, B. C. Cheah, M. R. Turner, A. Eisen, O. Hardiman, J. R. Burrell y M. C. Zoing, «Amyotrophic lateral sclerosis,» *The Lancet*, vol. 377, n.º 9769, págs. 942-955, 2011.
- [4] K. Bushby, R. Finkel, D. J. Birnkrant et al., «Diagnosis and management of Duchenne muscular dystrophy, part 1: diagnosis, and pharmacological and psychosocial management,» *The Lancet Neurology*, vol. 9, n.º 1, págs. 77-93, 2010.
- [5] N. J. Nilsson, «Shakey the robot,» SRI International, Menlo Park, CA, Tech. Rep. 323, 1984.
- [6] A. Ollero, *Robótica: Manipuladores y robots móviles*. Barcelona: Marcombo, 2001.
- [7] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, *Fundamentos de Robótica*, 2.ª ed. Madrid: McGraw-Hill, 2007.
- [8] W. G. Walter, «An imitation of life,» *Scientific American*, vol. 182, n.º 5, págs. 42-45, 1950.
- [9] SRI International, *Shakey the robot - Timeline of Innovation*, Recuperado de: <https://www.sri.com/hoi/shakey-the-robot/>, Accedido: 17-02-2026, 2024.
- [10] P. G. Fahlstrom y T. J. Gleason, *Introduction to UAV Systems*, 5.ª ed. John Wiley & Sons, 2022.

- [11] AIT SRL. «Emily, el robot salvavidas.» Blog de automatización industrial e ingeniería, visitado 31 de mar. de 2026. dirección: <https://www.ait.com.ar/es/blog/emily-el-robot-salvavidas>
- [12] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2.^a ed. MIT Press, 2011.
- [13] Robotnik Automation. «RB-VOGUI - Robot Móvil Autónomo,» visitado 17 de feb. de 2026. dirección: <https://robotnik.eu/es/productos/robots-moviles/rb-vogui/>
- [14] Boston Dynamics. «Boston Dynamics Expands Global Sales of Spot Robot,» visitado 17 de feb. de 2026. dirección: <https://bostondynamics.com/news/boston-dynamics-expands-global-sales-of-spot-robot/>
- [15] Wikimedia Commons. «Robot desactivador de explosivos tEODor - Ejército de Tierra,» visitado 17 de feb. de 2026. dirección: https://es.wikipedia.org/wiki/Archivo:Telerob_tEODor_Ej%C3%A9rcito_espa%C3%B1ol.jpg
- [16] J. E. Gutiérrez Lopera, J. A. Toloza Rangel, Á. J. Soto Vergel, O. López-Bustamante y D. Guevara Ibarra, «Vehículos terrestres no tripulados, sus aplicaciones y tecnologías de implementación,» *Ingeniare*, vol. 17, n.º 30, págs. 47-71, 2021.
- [17] Robotnik Automation. «RB-THERON - Manipulador móvil,» visitado 17 de feb. de 2026. dirección: <https://robotnik.eu/es/productos/robots-moviles/rb-theron/>
- [18] Robotnik Automation. «BACCHUS - Robots móviles para agricultura,» visitado 17 de feb. de 2026. dirección: <https://robotnik.eu/es/projects/bacchus/>
- [19] Futuro Eléctrico. «Robot bombero británico: Tecnología para salvar vidas,» visitado 17 de feb. de 2026. dirección: <https://futuroelectrico.com/robot-bombero-britanico/>
- [20] J. Holland, L. Kingston, C. McCarthy, E. Armstrong, P. O'Dwyer, F. Merz y M. McConnell, «Service Robots in the Healthcare Sector,» *Robotics*, vol. 10, n.º 1, pág. 47, 2021.

- [21] Hospitecnia. «Robot móvil UVD para desinfección por radiación ultravioleta,» visitado 17 de feb. de 2026. dirección: <https://hospitecnia.com/productos/matachana-robot-movil-uvd-desinfeccion-radiacion-ultravioleta/>
- [22] National Geographic España. «Objetivos del rover Perseverance en Marte,» visitado 17 de feb. de 2026. dirección: https://www.nationalgeographic.com.es/ciencia/objetivos-rover-perseverance-marte_16363
- [23] M. Arévalo Luzuriaga, «Sistema de Navegación Automática Controlada por Voz para una Silla de Ruedas,» *Revista Enfoque UTE*, vol. 1, n.º 1, 2010.
- [24] J. Alcubierre, J. Minguez, L. Montesano, L. Montano, O. Saz y E. Lleida, «Silla de Ruedas Inteligente Controlada por Voz,» en *Congreso Internacional de Domótica, Robótica y Teleasistencia para Todos (DRT4all)*, 2005.
- [25] J. González, C. Galindo, J. Fernández, J. Blanco, A. Muñoz y V. Arévalo, «La Silla Robótica SENA: un Enfoque Basado en la Interacción Hombre-Máquina,» *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 2008.
- [26] F. A. Auat Cheein, C. De la Cruz, R. Carelli y T. F. Bastos Filho, «Navegación Autónoma Asistida Basada en SLAM para una Silla de Ruedas Robotizada en Entornos Restringidos,» *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 2011.
- [27] I. Iturrate, C. Escolano, J. Antelis y J. Minguez, «Dispositivos robóticos de rehabilitación basados en Interfaces Cerebro-Ordenador: silla de ruedas y robot para teleoperación,» en *Actas de las Jornadas de Automática*, Universidad de Zaragoza, 2009.
- [28] *RB-THERON User Manual (Version STOCK 0.1)*, Manual técnico oficial del fabricante para la plataforma móvil, Robotnik Automation, 2024.
- [29] A. S. Tanenbaum y M. van Steen, *Distributed Systems: Principles and Paradigms*, 3.^a ed. Pearson Education, 2017.