PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Diseño de un Sistema de Control Avanzado No Lineal para la Navegación de un Robot Móvil Articulado con Orugas

Tesis para obtener el grado académico de Maestro en Ingeniería de Control y Automatización que presenta:

Martín Giancarlo Leiva Ottone

Asesor:

Dr. Carlos Gustavo Pérez Zúñiga

Lima, 2024

Informe de Similitud

Yo, Carlos Gustavo Pérez Zuñiga docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesor(a) de la tesis/el trabajo de investigación titulado Diseño de un Sistema de Control Avanzado No Lineal para la Navegación de un Robot Móvil Articulado con Orugas, del autor Martín Giancarlo Leiva Ottone, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 9%. Así lo consigna el reporte de similitud emitido por el software Turnitin el 02/12/2024.
- He revisado con detalle dicho reporte y confirmo que cada una de las coincidencias detectadas no constituyen plagio alguno.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:

Lima, 2024

<table>
<tr><td colspan="2">Apellidos y nombres del asesor: Pérez Zuñiga, Carlos Gustavo</td></tr>
<tr><td>DNI: 41864666</td><td rowspan="2">Firma:
 </td></tr>
<tr><td>ORCID: 0000-0001-5946-1395</td></tr>
</table>

AGRADECIMIENTOS

Agradezco a mis padres Fernando Leiva y Teresa Ottone por haberme acompañado y apoyado a lo largo de esta etapa universitaria. Al profesor Gustavo Pérez por guiarme en la realización de mi tesis. A mi mejor amigo André Lazo por haberme brindado su apoyo incondicional a lo largo de estos años. Finalmente, agradezco Tumi Robotics por brindar los recursos necesarios para la implementación y simulación del robot.

Se agradece el apoyo del proyecto titulado: “Sistema robótico móvil para inspección y digitalización remota de plantas industriales con infraestructura critica” a través del contrato: PE501081571-2023 del Concurso de Proyectos de Desarrollo Tecnológico 2023-03 de PROCIENCIA-CONCYTEC. La presente tesis se ha desarrollado en el marco de este proyecto.

RESUMEN

Este documento de tesis, se tiene como objetivo diseñar un sistema de control avanzado no lineal para la navegación de un robot móvil articulado con orugas que incorpore un controlador de seguimiento de trayectorias para la plataforma móvil del robot y otro para el sistema de articulaciones.

Actualmente, la aplicación de los robots móviles con orugas de manera autónoma se mantiene como un desafío. Este trabajo propone abordar dicho desafío mediante el diseño de un sistema de control avanzado no lineal para un robot articulado con orugas.

Para alcanzar este objetivo, se realiza el modelo cinemático de tanto la plataforma móvil como de su sistema de articulaciones. Seguidamente, se presenta la simulación del modelo, siendo Gazebo el software seleccionado. Dicha simulación se valida comparando sus resultados con los obtenidos a partir del modelo matemático. Posteriormente, se diseñan y comparan dos controladores para la plataforma móvil, uno del tipo Backstepping y otro del tipo Feedback Linearization. Para el sistema de articulaciones, se diseñan y comparan un controlador del tipo Gain Scheduling y otro basado en lógica difusa. Los resultados los controladores se evalúan considerando se precisión, tiempo de respuesta y respuesta ante perturbaciones. Se elige el controlador Backstepping para la plataforma móvil, y el controlador Gain Scheduling para el sistema de articulaciones. Finalmente, se concluye con una propuesta de implementación del sistema de control en una computadora embebida, con las consideraciones para integrarlo a un robot.

Palabras clave: Robots móviles articulados con orugas, Control No Lineal, Gazebo, Backstepping, Gain Scheduling.

Índice

AGRADECIMIENTOS i
RESUMEN iii
Índice de Figuras vii
Índice de Tablas xi
INTRODUCCIÓN 1
1. CAPÍTULO 1: ESTUDIO DEL ESTADO DEL ARTE 3
1.1 Introducción 3
1.2 Motivación 3
1.3 Problemática 6
1.4 Estado del arte de robots articulados con orugas 7
1.4.1 Instrumentación de robots articulados con orugas 9
1.4.1.1 Sensores de rango 9
1.4.1.2 Sensores de posición 11
1.4.1.3 Sensores visuales 12
1.4.2 Sistemas de control 13
1.4.2.1 Algoritmos de control para flippers 13
1.4.2.2 Algoritmos de control de plataforma móvil 17
1.5 Planteamiento de la propuesta y justificación 19
1.6 Objetivos de la tesis 21
1.6.1 Objetivo general 21
1.6.2 Objetivos específicos 21
2. CAPÍTULO 2: MODELADO DEL ROBOT 22
2.1 Introducción 22
2.2 Modelado Matemático 22
2.2.1 Modelado de la plataforma móvil 23
2.2.2 Modelado de las articulaciones 27
2.3 Pruebas de los modelos matemático 35
2.3.1 Pruebas del modelo de la plataforma móvil 35
2.3.2 Pruebas del modelo del sistema de flippers. 37
2.4 Conclusiones del modelado matemático 39

3. CAPÍTULO 3: SIMULACIÓN DEL ROBOT 40
3.1 Introducción 40
3.2 Softwares de simulación para robótica 40
3.2.1 Conceptos teóricos sobre softwares de simulación 41
3.2.2 Comparativa de softwares de simulación 42
3.2.3 Selección de software de simulación 43
3.3 Implementación de la simulación del modelo en Gazebo 44
3.3.1 Modelado del robot y entorno de simulación Gazebo 46
3.3.2 Integración del modelo con ROS 47
3.3.3 Implementación de realimentación de estado 48
3.3.4 Pruebas iniciales del modelo en Gazebo 49
3.4 Validación de la simulación 53
3.4.1 Pruebas de validación 53
3.4.2 Conclusiones de la validación 56
4 CAPÍTULO 4: DISEÑO DEL CONTROLADOR 57
4.1 Introducción 57
4.2 Control no Lineal en sistemas de robótica 58
4.2.1 Backstepping 58
4.2.2 Feedback Linearization 59
4.2.3 Control por Gain Scheduling 60
4.2.4 Control por lógica difusa 60
4.2.5 Comparativa de características de técnicas de control 62
4.3 Diseño del controlador para plataforma móvil. 63
4.3.1 Controlador de plataforma móvil basado en Backstepping 63
4.3.2 Controlador de plataforma móvil basado en Feedback Linearization 66
4.4 Diseño del controlador para sistema de articulaciones. 67
4.5 Comparativa de controladores y selección para plataforma móvil. 77
4.6 Comparativa de controladores y selección para sistema de articulaciones. 87
4.7 Evaluación de estabilidad de los controladores 97
4.7.1 Evaluación de estabilidad del controlador de la plataforma móvil 97
4.72 Evaluación de estabilidad del controlador del sistema de articulaciones 97

4.8 Implementación de los controladores en Gazebo. 98
5. CAPÍTULO 5: PROPUESTA DE IMPLEMENTACIÓN 101
5.1 Introducción 101
5.2 Diagrama general del sistema 101
5.3 Selección de sensores y unidades de procesamiento 102
5.3.1 Sistema de Referencia de Actitud y Rumbo 102
5.3.2 Sensor de distancia 103
5.3.3 Microcontrolador 104
5.3.4 Computadora embebida 106
5.3.5 Presupuesto del hardware 107
5.4 Instalación y configuración de hardware y software 108
5.4.1 Conexiones del sistema 108
5.4.2 Instalación y configuración del AHRS 110
5.4.3 Instalación y configuración de Sensor de distancia 111
5.4.4 Instalación y configuración de Microcontrolador 112
5.4.5 Instalación y configuración de Computadora embebida 114
5.5 Integración con ROS 114
5.5.1 Publicación de la información del AHRS en tópicos de ROS 115
5.5.2 Publicación de la información de los LiDAR en tópicos de ROS 115
5.5.3 Publicación de la información de motores en tópicos de ROS 116
5.6 Conclusiones de la propuesta de implementación 117
CONCLUSIONES 119
RECOMENDACIONES 121
BIBLIOGRAFÍA 122

Índice de Figuras

Figura 1.1. Fuente de energía del parque generador peruano (Banco Central de Reserva del Perú, 2021)..... 4
Figura 1.2. Esquema de control utilizado en robots móviles (Gonzáles, Pérez, Milanés, & Nashashibi, 2016)..... 8
Figura 1.3. Estructura de red neuronal de máquina de estados. (Azayev & Zimmermann, 2022)..... 17
Figura 2.1. Partes de robot articulado con orugas. Adaptado de: (Mario & Pirri, 2019)..... 23
Figura 2.2. Esquema para modelo cinemático de plataforma con orugas. (Al-Jarrah, Salah, & Almomani, 2019)..... 23
Figura 2.3. Esquema para modelo dinámico de plataforma con orugas. (Al-Jarrah, Salah, & Almomani, 2019)..... 26
Figura 2.4. Coordenadas para el modelo cinemático de las articulaciones. (Wang, y otros, 2018)..... 27
Figura 2.5. Diagrama simplificado 2D del robot articulado con orugas (vista lateral)..... 28
Figura 2.6. Diagrama simplificado 2D del robot articulado con orugas (vista frontal)..... 31
Figura 2.7. Notación utilizada para los Flippers. Elaboración propia..... 33
Figura 2.8. Variables de control utilizadas en la plataforma móvil. Elaboración propia..... 36
Figura 2.9. Trayectoria descrita por modelo de plataforma móvil. Elaboración propia..... 37
Figura 2.10. Variables de control utilizadas en el sistema de flippers. Elaboración propia..... 38
Figura 2.11. Trayectoria descrita por modelo de sistema de flippers. Elaboración propia..... 39
Figura 3.1 Coordenadas para el modelo cinemático de las articulaciones (Silva, Teichrieb, & Teixeira, 2019)..... 42
Figura 3.2 Modelo de ejemplo de un robot con orugas. Adaptado de: (Gazebo, 2024)..... 45
Figura 3.3. Articulaciones y eslabones de robot con orugas. Adaptado de: (Gazebo, 2024)..... 45
Figura 3.4. Campo de pruebas y modelo de robot. Elaboración propia..... 47
Figura 3.5. Pasos utilizados para lanzar un SDF en Gazebo utilizando ROS. Elaboración propia..... 48
Figura 3.6. Prueba de desplazamiento en terreno plano. Elaboración propia..... 50
Figura 3.7. Prueba de subida de escaleras. Elaboración propia..... 50
Figura 3.8. Prueba de bajada de escaleras. Elaboración propia..... 51
Figura 3.9. Prueba 1 de atravesar obstáculos. Elaboración propia..... 51
Figura 3.10. Prueba 2 de atravesar obstáculos. Elaboración propia..... 52

Figura 3.11. Utilización plataforma móvil y flippers simultáneamente. Elaboración propia. 52
Figura 3.12. Posición del robot moviéndose mediante su plataforma móvil. Elaboración propia. 53
Figura 3.13. Error de posición del robot moviéndose mediante su plataforma móvil. Elaboración propia. 54
Figura 3.14. Posición del robot moviéndose mediante sus articulaciones. Elaboración propia. 55
Figura 3.15. Posición del robot moviéndose mediante sus articulaciones. Elaboración propia. 55
Figura 4.1. Error de posición de plataforma móvil. (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016) 63
Figura 4.2. Estructura del controlador Backstepping. (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016) 64
Figura 4.3. Esquema utilizado para controlador por Feedback Linearization. (Kowalczyk & Kozlowski, 2014) 66
Figura 4.4. Esquema de control para sistema de articulaciones. Elaboración propia 68
Figura 4.5. Ley utilizada para determinar los parámetros de control. Elaboración propia 71
Figura 4.6. Función de membresía de Error en Z. Elaboración propia 72
Figura 4.7. Función de membresía de Error en Phi. Elaboración propia 73
Figura 4.8. Función de membresía de Theta1. Elaboración propia 73
Figura 4.9. Función de membresía de Error en Z. Elaboración propia 74
Figura 4.10. Función de membresía de Error en Omega 1. Elaboración propia 74
Figura 4.11. Función de membresía de Error en Omega 1. Elaboración propia 75
Figura 4.12. Lógica detrás de las reglas difusas. Elaboración propia 76
Figura 4.13. Trayectoria obtenida por controlador Backstepping. Elaboración propia 79
Figura 4.14. Trayectoria obtenida por controlador Feedback Linearization vs Trayectoria deseada. Elaboración propia 79
Figura 4.15. Error con respecto a la trayectoria – Controlador Backstepping. Elaboración propia 80
Figura 4.16. Error con respecto a la trayectoria – Controlador Feedback Linearization. Elaboración propia 80
Figura 4.17 Señales de control – Controlador Backstepping. Elaboración propia 81
Figura 4.18. Señales de control – Controlador Feedback Linearization. Elaboración propia 81
Figura 4.19. Trayectoria obtenida por controlador Backstepping afectado por ruido Gaussiano. Elaboración propia 82

Figura 4.20. Trayectoria obtenida por controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia..... 83
Figura 4.21. Error con respecto a la trayectoria – Controlador Backstepping afectado por ruido Gaussiano. Elaboración propia..... 83
Figura 4.22. Error con respecto a la trayectoria – Controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia 84
Figura 4.23 Señales de control – Controlador Backstepping afectado por ruido Gaussiano. Elaboración propia 84
Figura 4.24. Señales de control – Controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia 85
Figura 4.25. Respuesta al Controlador basado en Gain Scheduling. Elaboración propia. 88
Figura 4.26. Respuesta al Controlador basado en Lógica Difusa. Elaboración propia. 89
Figura 4.27. Salidas de control del Controlador basado en Gain Scheduling. Elaboración propia. 90
Figura 4.28. Salidas de control del Controlador basado en Lógica Difusa. Elaboración propia. 90
Figura 4.29. Respuesta al Controlador basado en Gain Scheduling afectado por ruido Gaussiano. Elaboración propia. 91
Figura 4.30. Respuesta al Controlador basado en Lógica Difusa afectado por ruido Gaussiano. Elaboración propia. 92
Figura 4.31. Salidas de control del Controlador basado en Gain Scheduling afectado por ruido Gaussiano. Elaboración propia..... 93
Figura 4.32. Salidas de control del Controlador basado en Lógica Difusa. Elaboración propia. 93
Figura 4.33. Respuesta del controlador de estabilización para valores iniciales positivos. Elaboración propia..... 94
Figura 4.34. Respuesta del controlador de estabilización para valores iniciales negativos. Elaboración propia. 94
Figura 4.35. Señales de control del controlador de estabilización para valores iniciales positivos. Elaboración propia. 95
Figura 4.36. Salidas de control del controlador de estabilización para valores iniciales negativos. Elaboración propia. 95
Figura 4.37. Trayectoria obtenida Backstepping dentro de Gazebo. Elaboración propia 98
Figura 4.38. Prueba de seguimiento de trayectoria. Elaboración propia 99
Figura 5.1. Robot articulado con orugas propuesto. Elaboración propia 101
Figura 5.2 Diagrama de conexiones del robot. Elaboración propia 108
Figura 5.3. Esquemático de conexión microcontrolador a motor. Elaboración propia 109
Figura 5.4. Instalación de hardware del robot. Elaboración propia..... 109
Figura 5.5 Conexión con el AHRS mediante EvalTool. Elaboración propia ... 110
Figura 5.6. Configuración de envío de datos del AHRS. Elaboración propia . 110

Figura 5.7. Elección de microcontrolador en STM32 Cube IDE. Elaboración propia 112
Figura 5.8. Configuración de pines del microcontrolador. Elaboración propia 112
Figura 5.9. Configuración de reloj del periférico CAN. Elaboración propia..... 113
Figura 5.10. Configuración del periférico CAN. Elaboración propia 113
Figura 5.11. Publicación de información de orientación en ROS. Elaboración propia 115
Figura 5.12. Visualización de nube de puntos en RViz. Elaboración propia .. 116
Figura 5.13. Pasos utilizados integrar motores con ROS. Elaboración propia 117

Índice de Tablas

Tabla 1.1. Panorama general de trabajos relacionados. 14
Tabla 3.1. Comparativa de softwares de simulación. 43
Tabla 3.2. Características de la computadora utilizada para la simulación. Elaboración propia 44
Tabla 3.3. Leyenda de "Articulaciones y eslabones de robot con orugas". Elaboración propia 46
Tabla 3.4. Cambios del modelo del robot realizados en el SDF. Elaboración propia 46
Tabla 3.5. Parámetros usados para lanzar archivo SDF en Gazebo. Elaboración propia 47
Tabla 3.6. Parámetros usados para lanzar archivo SDF en Gazebo. Elaboración propia 48
Tabla 3.7. Relación de pruebas realizadas. Elaboración propia..... 49
Tabla 4.1. Características de las técnicas de control no lineal. Elaboración propia 62
Tabla 4.2. Parámetros para los controladores de la plataforma móvil. Elaboración propia 78
Tabla 4.3. Métricas de los controladores de la plataforma móvil. Elaboración propia 86
Tabla 4.4. Parámetros para los controladores de la plataforma móvil. Elaboración propia 87
Tabla 4.5. Parámetros para los controladores de seguimiento para el sistema de Flippers. Elaboración propia...... 96
Tabla 5.1. Comparativa de AHRS. Elaboración propia 102
Tabla 5.2. Comparativa de LiDARs 3D. Elaboración propia..... 104
Tabla 5.3. Características de los motores utilizado. Elaboración propia 105
Tabla 5.4. Comparativa de microcontroladores. Elaboración propia 106
Tabla 5.5. Comparativa de computadoras embebidas. Elaboración propia ... 107
Tabla 5.6. Presupuesto de la propuesta de implementación. Elaboración propia 108
Tabla 5.7. Requerimientos de software para uso de Livox MID-360. (Livox, 2024) 111
Tabla 5.8. Configuración de Livox MID-360. Elaboración propia..... 111

INTRODUCCIÓN

La industria del futuro se va a caracterizar por la adopción de tecnologías de sistemas autónomos. Muchas industrias como la minera, energética, agrícola entre otras, presentan distintas oportunidades de aplicación de dichas tecnologías, como la inspección automatizada de infraestructura crítica, que requeriría el uso de vehículos con la capacidad de maniobrar en terrenos irregulares de manera robusta.

Los robots móviles con orugas presentan distintas ventajas en comparación a otros robots móviles con la capacidad de navegar en terrenos irregulares, debido a que el uso de articulaciones provee al robot de mayor movilidad sin incrementar considerablemente el consumo energético ni los grados de libertad. En este documento se presenta la propuesta para diseñar y simular un sistema de control avanzado no lineal de un robot móvil articulado con orugas.

Estas tecnologías van a permitir incrementar la productividad y eficiencia de las operaciones, así como garantizar la seguridad y salud a los trabajadores. Al desarrollar el conocimiento y tecnología propietaria se tiene la posibilidad de atender nuevas aplicaciones a la medida de la industria minera y energética que cuenta con un mercado relevante.

En el Capítulo 1, se presenta a manera de motivación las aplicaciones de los robots móviles articulados con orugas en distintas industrias. Asimismo, se presenta y justifica el problema de control abierto que existe en este tipo de robots. Se realiza un estudio del estado del arte en las estrategias de control e instrumentación utilizada en los robots móviles articulados con orugas. El capítulo culmina con la presentación del objetivo general y los objetivos específicos de la tesis.

En el Capítulo 2 se realiza el modelamiento matemático de un robot móvil articulado con orugas, presentando ecuaciones cinemáticas y dinámicas. Luego se evaluará el comportamiento del modelo obtenido en distintas condiciones de operación.

En el Capítulo 3 se documenta y describe la implementación de la simulación de un robot móvil articulado con orugas en un ambiente móvil. Asimismo, se hace una breve introducción a los softwares de simulación en robótica.

En el Capítulo 4 se realiza el diseño y simulación del controlador de seguimiento de trayectorias, se hace una introducción al control no lineal. Luego se describen las etapas de la metodología y su implementación. Además, se presentan los resultados de la evaluación del algoritmo.

Finalmente, en el Capítulo 5, se ofrece una propuesta de implementación, la cual incluye la descripción de los sensores que serán utilizados, así como de la tarjeta de acondicionamiento y el controlador. También se especifica la instalación de estos componentes en el robot móvil articulado con orugas.

CAPÍTULO 1: ESTUDIO DEL ESTADO DEL ARTE

1.1 Introducción

El presente capítulo inicia exponiendo el estado actual los robots móviles articulados con orugas, introduciendo el problema de control a resolver en la investigación. A continuación, se presenta una revisión del estado del arte de los sistemas de control e instrumentación de los robots articulados con orugas orientados a aplicaciones en terrenos irregulares. Luego, se justifica y plantea la propuesta de solución. Finalmente se expone el objetivo principal y los objetivos específicos de la investigación.

1.2 Motivación

La aplicación de la robótica en terrenos irregulares es una rama que ha comenzado a ser explorada con mayor frecuencia estos últimos años debido a que la complejidad del control de robots que puedan navegar eficientemente este tipo de terreno era computacionalmente prohibitiva previamente. Los robots considerados dentro de esta rama tienen la capacidad de maniobrar en terrenos irregulares de manera robusta, subir escaleras, así como transitar sobre obstáculos (Mitriakov A. , Papadakis, Kerdreux, & Garlatti, 2021).

Este tipo de robots presentan distintas ventajas en comparación a otros robots móviles. Por un lado, el uso de orugas en comparación a ruedas, provee al robot de mayor área de contacto con la superficie, ayudando en su movilización en ambientes flexibles; mientras que el bajo centro de gravedad característico de este tipo de robots, le da mayor estabilidad que robots bípedos o cuadrúpedos. Asimismo, el uso de articulaciones, las cuales consisten mayormente de cuatro (04) sub-orugas montadas al frente y atrás del robot unidas al cuerpo principal por articulaciones rotatorias, proveen al robot de mayor movilidad y de la capacidad de maniobrar en terrenos irregulares sin incrementar la complejidad del control, consumo energético ni los grados de libertad tanto como otras opciones, por ejemplo robots cuadrúpedos o robots cuadrúpedos de locomoción híbrida (Yuan, Wang, & Schwertfeger, 2019).

Una de las mayores aplicaciones de estos robots es la de realizar inspecciones y hacer diagnósticos precisos en las instalaciones de plantas industriales con infraestructura crítica, siendo objeto de estudio para los principales centros de investigación enfocados en inspección de ambiente complejos, especialmente los orientados a la industria minera y energética (Szrek, Jakubiak, & Zimroz, 2022).

En este contexto, es necesario profundizar en la dinámica de las industrias mencionadas previamente. La industria energética peruana, reconocida por el Banco Central de Reserva del Perú por su suma importancia, ya que un suministro eléctrico seguro y continuo es una condición necesaria para mitigar posibles limitaciones para el crecimiento económico sostenido en todos los niveles. Esta industria cuenta con un parque generador con características resumidas en la *Figura 1.1.*

La energía hidráulica se posiciona como la más predominante, componiendo el 40% de la oferta de generación, mientras que la segunda más común es la energía térmica proveniente del gas de Camisea, siendo el 30% de la oferta de generación. En cuanto, a ubicación, el 65% de la oferta de generación está concentrada en el centro del país (Banco Central de Reserva del Perú, 2021).

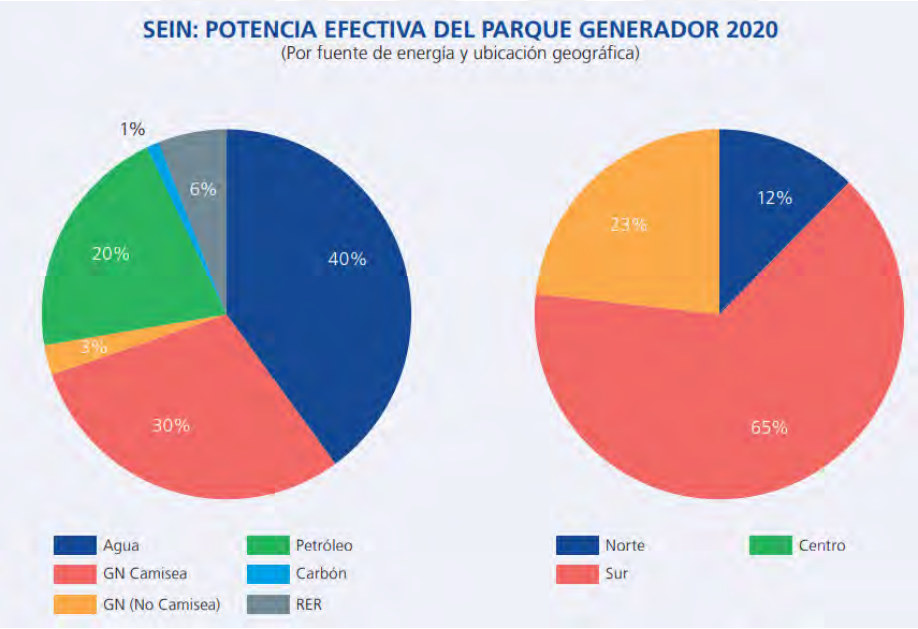

Figura 1.1. Fuente de energía del parque generador peruano (Banco Central de Reserva del Perú, 2021).

En el caso de la industria de hidrocarburos, hay 18 cuencas con potencial de sedimentos, la mayoría en la selva (más del 50% del total nacional (Centro de Investigación y Tecnología del Agua, 2021). Mientras que, en el caso de la minería, los principales yacimientos mineros se encuentran en su mayoría en el centro del país (sobre los 4000 metros sobre el nivel del mar), y cuyo uso está regulado por el Ministerio de Energía y Minas (Ministerio de Energía y Minas, 2017). Esto implica que los sitios de producción y/o explotación están alejados de zonas de alta demanda (como ciudades, industrias y refinerías), siendo de difícil acceso. Los trabajos de inspección en estos lugares remotos requieren la movilización de personal altamente capacitado para detectar fallas estructurales, descargas superficiales, grietas, puntos de corrosión, etc.

Según los estándares internacionales, el costo de los trabajos de inspección en relación con el costo total de las operaciones puede llegar a representar del 30% al 35%, sin incluir los gastos de movilización del personal ni el cierre de la producción en caso de un fallo crítico (Stellman, 1998). Además, aunque toda la información recabada en las inspecciones es obtenida por personal calificado, no se tiene un registro histórico que con el cual se pueda comparar las mediciones periódicas, ni un gemelo digital que permita una identificación espacial de cada elemento de la infraestructura inspeccionada en una base de datos sistematizada y segura. Si se usaran métodos tradicionales, esto implicaría una mayor frecuencia de inspección por parte del personal y, por lo tanto, una mayor exposición a los riesgos propios de trabajar en lugares remotos o con una geografía agreste (acceso limitado a servicios médicos, condiciones meteorológicas adversas, acceso limitado a suministros y problemas de comunicación) (Gehring, Fankhauser, & Diethelm, 2021).

Debido a esto, se ha identificado que la utilización de robots que puedan navegar en ambientes con terrenos irregulares, como los robots móviles articulados con orugas, resultan ventajosos, ya que no exponen al personal y pueden ejecutar este proceso de manera más frecuente que los operarios humanos, así como recolectar información de manera automatizada (Gehring, Fankhauser, & Diethelm, 2021).

1.3 Problemática

Teniendo en cuenta las condiciones del terreno en donde se operarían los robots de inspección, se requieren robots con la capacidad de maniobrar en terrenos irregulares de manera robusta, subir escaleras, así como transitar sobre obstáculos. Los robots articulados con orugas han sido propuestos en investigaciones relacionadas debido a que son capaces de cumplir con los retos propuestos previamente (Chen, Wu, Wang, Yang, & Zhang, Motion Planning for Autonomous Climbing Stairs for Flipper Robot, 2020).

Modelos comerciales de robots articulados con orugas teleoperados ya se han desarrollado y son utilizados en distintos campos como agricultura, exploración de terrenos irregulares, reconocimiento de lugares radiactivos o contaminados, minería y vigilancia (Bruzzone, Nodehi, & Fanghella, 2022). Entre los modelos comerciales se destacan la plataforma móvil “Tready” desarrollado por HEBI Robotics (HEBI Robotics, 2023), y el modelo “Jaguar V6” de Dr. Robot (Dr Robot, 2018), pero dichos modelos son teleoperados. Por el lado de desarrollo de autonomía, actualmente existen robots con orugas disponibles comercialmente, los cuales son guiados automáticamente, pero estos se desplazan en terrenos planos y compactos y son usados para mover componentes y productos dentro de edificios industriales, la aplicación extensiva de este tipo de robots de manera autónoma en entornos no estructurados o estructurados para humanos se mantiene como un desafío en estos años (Markets And Markets, 2023).

Entre las razones para que el desarrollo de un robot móvil articulado con orugas comercial que cuente con navegación autónoma se mantenga como un desafío son diversas. Entre estas se encuentran la falta de reproducibilidad de las investigaciones relacionadas debido al costo de los equipos y experimentos, así como la tendencia a desarrollar el sistema de control del robot de manera modular debido a su complejidad y no llegar a integrar los submódulos independientes. Estas condiciones impactan en el desempeño de los robots en situaciones reales (Rouček, y otros, 2021).

Asimismo, cabe resaltar que el control de los flippers presenta un problema importante en lograr la autonomía de estos robots, ya que sus grados de libertad adicionales imponen una carga cognitiva significativa en caso de operadores

humanos, o una complejidad computacional adicional en el caso de operación autónoma (Chen, Wu, Wang, Yang, & Zhang, Motion Planning for Autonomous Climbing Stairs for Flipper Robot, 2020). En parte debido a que es necesario considerar en el control la traversabilidad adaptativa, la cual consiste en adaptar la morfología del robot para atravesar terrenos de forma óptima (Zimmermann, Zuzanek, Reinstein, & Hlavac, 2014).

Existe un problema de control abierto, tanto del lado del control de la plataforma móvil como del control de las articulaciones del robot. Los controladores que se muestran en la literatura son novedosos y avanzados, siendo lo más común el uso de técnicas de *Reinforcement Learning* o *Random Search*. Este enfoque típicamente asume poco o nada de la tarea en cuestión y por lo tanto llevan a un controlador agnóstico al robot y puede causar comportamientos poco deseados, por lo que se requiere una cuidadosa ingeniería de recompensas para aliviar este problema. Asimismo, este método conlleva a un algoritmo de control poco interpretable y que requiere una alta complejidad de muestras y simulaciones rápidas de millones de pasos para obtener un entrenamiento apropiado (Azayev & Zimmermann, 2022).

1.4 Estado del arte de robots articulados con orugas

Al momento de desarrollar la autonomía de robots articulados con orugas, una revisión de la literatura muestra distintas arquitecturas de control. La universidad de Stanford y la Universidad de Carnegie Mellon (CMU) son pioneras en la implementación de vehículos autónomos en los desafíos DARPA Grand y Urban. Por otro lado, en Europa, los proyectos VIAC y Daimler describieron las distintas etapas de procesamiento necesarias para la arquitectura de control de vehículos automatizados, dándole mayor énfasis a la percepción, toma de decisiones y control. En la *Figura 1.2* se sintetizan de manera organizada todas las etapas necesarias para controlar este tipo de robots. En este esquema el proceso de control se divide en 4 bloques los cuales no necesariamente tienen los mismos nombres que los presentes en la literatura relacionada, pero presentan nombres y funciones similares en la mayoría de los casos (Gonzáles, Pérez, Milanés, & Nashashibi, 2016).

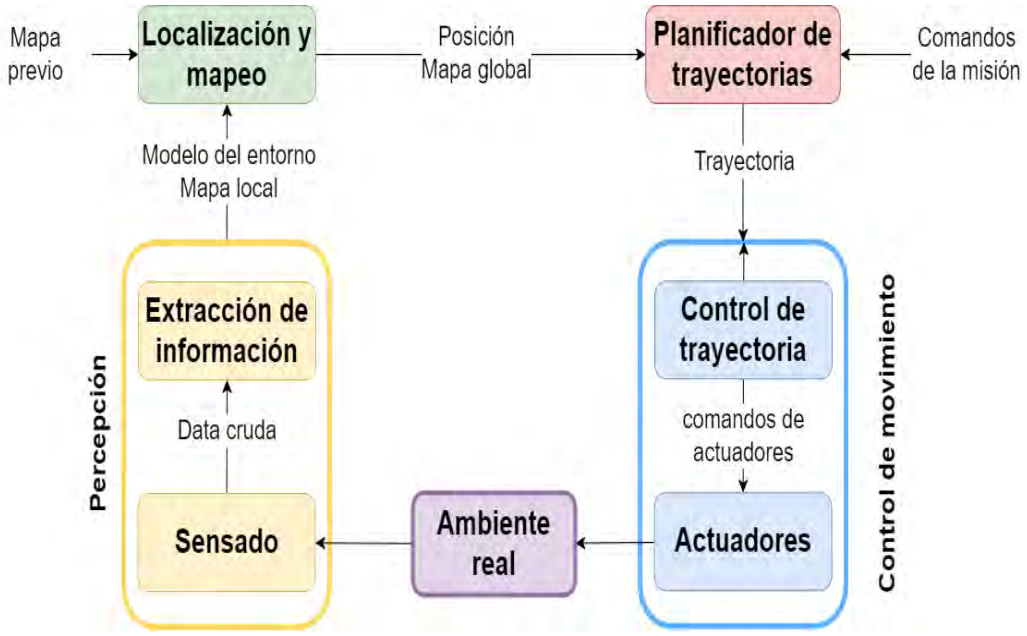

Figura 1.2. Esquema de control utilizado en robots móviles (Gonzáles, Pérez, Milanés, & Nashashibi, 2016).

Un esquema similar, mencionado en (Fankhauser P. , 2018) propone que los distintos módulos están interconectados pero que funcionan en paralelo a diferentes frecuencias de actualización. Se comienza en el módulo de percepción, el cual obtiene información del robot como su inclinación, velocidad, aceleración y distancia con respecto al terreno, utilizando sensores de posición, sensores de rango, sensores visuales, entre otros. Esta información va al módulo de localización y mapeo el cual utiliza la información recaudada por el módulo de percepción, y en ciertos casos es ayudado por mapas previos del ambiente para crear un mapa global del entorno y obtener la posición del robot dentro de dicho mapa. El módulo de planificación de trayectorias utiliza la información del anterior módulo, así como los comandos de la misión (los cuales consisten en la posición y orientación final del robot, así como opcionalmente puntos intermedios) para generar una ruta que alcance los objetivos previamente indicados así poder cambiar de trayectoria para esquivar los obstáculos físicos que se presenten en el recorrido. Dicha trayectoria es enviada al módulo de control de movimiento el cual genera los comandos de actuadores necesarios para que el robot pueda seguir la trayectoria deseada.

En esta sección se presentará el estado del arte del controlador de seguimiento y la instrumentación requerida para la percepción.

1.4.1 Instrumentación de robots articulados con orugas

La instrumentación utilizada en robots articulados con orugas tiene como fin percibir los alrededores del robot, permitiendo su control y navegación. Los tipos de sensores más usados son: sensores de rango, sensores de posición y sensores visuales.

1.4.1.1 Sensores de rango

Los sensores de rango son usados para medir la distancia que tiene el robot con diferentes puntos en el ambiente donde este se encuentra, luego se toman dichos puntos para generar una reconstrucción digital del ambiente (Fankhauser P. B., 2015). En el campo de robots móviles, se utilizan distintas tecnologías como sensores de rango ya sea de manera individual o combinando múltiples tecnologías. Las más comunes son las cámaras de luz estructurada, cámaras estéreo, cámaras de tiempo de vuelo y sensores láser, ya que son sensores de corto alcance, pero cuentan con una alta densidad de puntos, ideales para escanear los terrenos más próximos al robot (Siciliano, 2016).

Entre los sensores de rango disponibles, todos cuentan con distintas ventajas y desventajas, ninguno de ellos es universalmente mejor en todos los aspectos.

Cámaras de luz estructurada

Estas son un tipo de cámaras de profundidad que se basan en proyectar un patrón de luz conocido en el ambiente y reconstruir la geometría del ambiente analizando la distorsión de la luz en toda la escena (Zanuttigh, 2016). Las ventajas de estos sensores son su reducido peso, costo y consumo en comparación a los demás, así como la alta calidad y densidad de las imágenes de profundidad que estas proveen. Por otro lado, sus mediciones son sensibles a la luz solar por lo cual solo son recomendables para su uso en ambientes cerrados.

Cámaras estéreo

Este es otro tipo de cámaras de profundidad en la cual la estimación de profundidad se basa en la triangulación de un objetivo desde dos diferentes puntos de vista, siendo la distancia calculada conociendo la posición relativa

entre las dos cámaras e identificando un punto común entre ambas. Las ventajas de estas cámaras son similares a las cámaras de luz estructurada, siendo de un tamaño y consumo similar, pero en este caso las cámaras estéreo no se ven afectadas por la luz solar, por lo que pueden ser usadas en exteriores. Por el lado de las desventajas, estás cámaras suelen ser ligeramente más afectadas por el ruido, especialmente en distancias mayores a 30 metros (Keselman, 2017).

Cámaras de tiempo de vuelo

Las cámaras de tiempo de vuelo determinan la distancia a los objetos calculando el tiempo de tránsito de pulsos de luz. Este proceso no se realiza de manera secuencial, sino que estas cámaras utilizan un chip que contiene una matriz de circuitos temporizadores que les permite registrar el tiempo de vuelo para cada píxel de la cámara infrarroja, de esta manera al momento de iluminar la escena, la luz es reflejada por los obstáculos, capturando toda la escena al mismo tiempo (Keselman, 2017). Este tipo de cámara ofrece una resolución a la par con el resto de las alternativas a un costo reducido, pero con la desventaja de un mayor peso, consumo de corriente y sensibilidad al ruido (Fankhauser P. B., 2015).

Sensores láser

Los sensores de rango láser o LIDAR miden la distancia estimando el tiempo que la luz viaja del sensor al objeto y regresa. El impulso láser que es transmitido desde el emisor al objeto es parcialmente reflejado al detector. La alteración de la fase y/o módulo de esta señal es utilizada para determinar la distancia del LIDAR al objetivo medido. Este tipo de sensores son los más comunes entre los sensores de rango debido a que pueden proveer información a largas distancias y son bastantes robustos ante distintas iluminaciones (J. Nubert, 2022). Los LIDAR más populares en robot articulados con orugas son LIDAR 3D como VLP-16 PUCK u Ouster OS1-32, los cuales presentan un arreglo rotativo de 32 y 16 distribuciones de láseres ordenados de manera vertical, y un campo de visión vertical de 45° y 30° respectivamente. Este tipo de sensores se pueden utilizar para poder localizar y evitar obstáculos, pero debido a su campo de visión, no son ideales para reconstrucción de terreno si no están equipados con un mecanismo rotatorio (Fankhauser P. D., 2016). Debido a eso, en los últimos años

se ha visto la aplicación de LIDARs con un campo de visión de 90°, como el RS-BPearl. Este tipo de LIDAR si bien pueden escanear mayores volúmenes, no tienen una densidad tan alta de puntos como los demás sensores láser (Tranzatto M. &., 2022).

1.4.1.2 Sensores de posición

El uso de sensores de posición tiene como finalidad medir en tiempo real la cinemática interna del robot, así como la aceleración inercial, ratio de giro e información precisa de la posición del robot con el fin de estimar correctamente los estados del robot. Los sensores frecuentemente utilizados en estas aplicaciones son encoders incrementales y sensores inerciales, los cuales son usados en conjunto (Bloesch, Hutter, & Leutenegger, 2013).

Los encoders incrementales proveen información de la posición angular de todas las articulaciones. Con esta información, y conociendo las distancias de cada elemento del robot, se puede determinar a través de cinemática directa, la posición de cada articulación con respecto al cuerpo (Bloesch M. , 2017).

Los Sistemas de Referencia de Actitud Rumbo o *Attitude Heading Reference System* (AHRS) es un tipo de sensor que mide la aceleración propia y la velocidad angular del cuerpo principal del robot. Esta información puede ser usada de manera individual para realizar estimaciones de alta calidad y ancho de banda de la inclinación del robot. Asimismo, la información obtenida suele ser fusionada con distintos sensores como encoders, LIDAR o cámaras. La fusión con encoders puede ser utilizada en un modelo predictivo de los estados del robot (A. Chilian, 2011), mientras que fusionando la información de las mediciones de LIDAR e AHRS se puede reconstruir ambientes 3D utilizando métodos de Localización y Mapeo Simultáneo (SLAM) (Murray, 2007). Mientras que las reconstrucciones 3D utilizando SLAM pueden obtener buenos resultados en términos de precisión, pueden tener problemas en escenarios donde se presenta un desplazamiento rápido o faltan texturas. Una manera común de mejorar la robustez de las reconstrucciones es integrar la data de las cámaras al proceso de estimación, lo cual puede mitigar el efecto del ruido de medición del AHRS (Uhlmann, 2001).

1.4.1.3 Sensores visuales

Esta clasificación se refiere a las cámaras que no pueden ser clasificadas como sensores de rango. Existen distintos tipos de cámara que recaen en esta clasificación, entre los más usados en robots articulados con orugas se encuentran: cámaras RGB estáticas, cámaras PTZ y cámaras térmicas. Asimismo, el uso que se le da a este tipo de sensores bajo el contexto de robots articulados con orugas se puede resumir en detección de objetos, localización y teleoperación.

En el caso de detección de objetos, se pueden utilizar todos los tipos de cámara previamente mencionados con el fin de captar información del ambiente y utilizarla para detectar y clasificar distintos objetos o personas relacionadas con la aplicación del robot con la ayuda de redes neuronales. Las cámaras más comunes para esta tarea son las cámaras RGB, usando un arreglo de cámaras para tener una distribución que permita una visualización de 360° (Tranzatto M. &., 2022). Asimismo, las cámaras PTZ pueden ser usadas en caso se quiera enfocar un objeto alejado previamente detectado por un LIDAR y que las cámaras RGB no pudieron captar debido a distancia (M. Patel, 2022). En el caso de las cámaras térmicas, estas son usadas en situaciones específicas, por ejemplo, detección de personas y ciertos equipos utilizando cámaras térmicas.

El uso de cámaras en SLAM se describió en la sección 1.3.1.2 aunque cabe resaltar que las cámaras más frecuentemente usadas para esta función son las cámaras RGB con lente de pez, con el fin de captar escaleras o rampas. Otras cámaras usadas con esta aplicación son cámaras térmicas como se puede ver en (Tranzatto M. &., 2021).

Finalmente, se tiene el caso de la teleoperación en el cual se usa normalmente una cámara RGB con un amplio campo de visión para permitir al operario del robot conocer el terreno en el que se encuentra y decidir las metas de la navegación e identificar objetos relevantes en la inspección (Arm, 2022).

1.4.2 Sistemas de control

La navegación del robot es dependiente del ambiente donde opera el mismo. Debido a la numerosa cantidad de obstáculos en tanto ambientes al aire libre como en ambientes cerrados, se han añadido articulaciones adicionales (*flippers*) a los robots con orugas (Mitriakov A. , Papadakis, Kerdreux, & Garlatti, 2021). Si bien la incorporación de *flippers* en la parte delantera y trasera de un robot aumentan la capacidad de navegar a través obstáculos complejos, estos aumentan la complejidad del control, cálculo de la cinemática y estimación de la dinámica del robot (Yuan, Wang, & Schwertfeger, 2019). Se desarrollan distintas técnicas de control para abordar la utilización de estas articulaciones en la navegación del robot de manera eficiente y robusta.

1.4.2.1 Algoritmos de control para flippers

Entre las principales estrategias de control presentes en la literatura actual, se apreciar que los robots articulados con orugas presentan controladores separados para el control de la plataforma con orugas (o plataforma móvil) y para el control de sus articulaciones o flippers. En cuanto a controladores para los flippers, se pueden encontrar controladores neuro difusos y basados en redes neuronales que utilizan Reinforcement Learning para su entrenamiento en la mayoría de los casos. La ventaja de estos controladores en contraste con los métodos que no utilizan entrenamiento es su capacidad de utilizar data de grandes dimensiones y toma de decisiones desde data cruda, como píxeles (Mitriakov A. , Papadakis, Kerdreux, & Garlatti, 2021). En la *Tabla 1.1* se muestra panorama general a grandes rasgos. Se divide la clasificación de los controladores de flippers entre control standard o control sin entrenamiento, y control de flippers basado en entrenamiento.

Tabla 1.1. Panorama general de trabajos relacionados.

Paper	Método	DoF	Sensores utilizados
Mourikis et al.	Linealización por retroalimentación de estados	3	Cámara RGB y giroscopio
Nagatani et al.	Difuso + visión por computadora	4	Sensores láser
Ben-Tzvi et al.	Fuzzy	3	Encoders y compas
Zhang et al.	Fuzzy + controlador de velocidad de movimiento resuelto	3	Giroscopio
Pecka et al.	Neurocontrolador RL	2	AHRS, LiDAR
Endo et al.	PI	4	AHRS, LiDAR
Ejaz et al.	Neurocontrador RL, End-to-end	2	Cámara de profundidad

Adaptado de: (Mitriakov A. , Papadakis, Kerdreux, & Garlatti, 2021)

Control convencional

Trabajos preliminares para navegación de obstáculos de manera autónoma utilizaron controladores sin entrenamiento los cuales eran altamente específicos al robot utilizado y no podían ser generalizados. Entre los métodos utilizados, se encuentran algoritmos basados en la estimación de bordes de obstáculos y data del giroscopio (Mourikis, Trawny, Roumeliotis, & Matthies, 2007)

Asimismo, también se presentan algoritmos de control para los flippers que perciben la geometría de los obstáculos usando sensores laser y utilizan dicha información para adaptar sus flippers de tal manera que se encuentren tangentes a la superficie. Los autores afirman que el sistema es capaz de navegar en terreno desigual basándose en data exhaustiva de los sensores. Aun así, el robot no es completamente autónomo, y el método es poco generalizable para plataformas similares (Nagatani, Yamasaki, Yoshida, Yoshida, & Koyanagi, 2008).

Otro método utilizado se basa en la estimación precisa de la configuración de los obstáculos, en donde el robot puede navegar de manera autónoma utilizando la estimación de estados propioceptivo. Si bien esta estrategia presentó buenos

resultados, solo aplica a escaleras conocidas de manera previa (Endo, Watanabe, & Nagatani, 2016).

En los últimos años se han visto también métodos de control similares al uso de máquinas de estado, en donde el proceso de subir escaleras se divide en pasos dependiendo de la distancia del robot a las escaleras, y su distancia con el piso, de esta manera, cada paso es un estado, y cada estado tiene una acción de control distinta, las cuales están preprogramadas (Yuan, Wang, & Schwertfeger, 2019).

Control basado en entrenamiento

Los métodos basados en entrenamiento se perfilan como los más adecuados al momento de desarrollar controladores para robots cuya cinemática y dinámica son difíciles de modelar. En la mayoría de los casos, uno de los requerimientos principales es la escalabilidad a espacios de gran dimensionalidad de estados y acciones. Asimismo, se buscan métodos de entrenamiento generalizables y robustos que le permitan al robot aprender diferentes tareas bajo distintas condiciones (Mitriakov A. , Papadakis, Mai Nguyen, & Garlatti, 2020) .

Debido a la dificultad de modelar la dinámica de la interacción física de atravesar obstáculos, sobre todo en el momento de subir y bajar escaleras, es necesario la utilización de políticas de control que consideren tanto la estabilidad del sistema como la irregularidad del terreno, esta tarea se puede realizar de distintas maneras. Por ejemplo, se pueden incluir restricciones en el problema de optimización de RL (Reinforcement Learning), como en en (Pecka, Šalanský, Zimmermann, & Svoboda, 2016), mientras que la seguridad del sistema y la recompensa se puede asociar en la misma función (Zimmermann, Zuzanek, Reinstein, & Hlavac, 2014). Otra manera más sofisticada de considerar la estabilidad del sistema en el controlador es incorporar restricciones en el problema de optimización de la búsqueda de políticas de entropía relativa utilizando RL (Pecka, Šalanský, Zimmermann, & Svoboda, 2016).

El uso de imágenes en la navegación también es contemplado como una posible entrada al momento de diseñar un Neurocontrolador para este tipo robots. En el trabajo de (Ejaz, Tang, & Lu, 2021) se puede ver un controlador basado en una

doble Deep Q-network (DQN) con normalización de capas e inyección de ruido. Aun así, este controlador fue usado sólo para navegación en superficies 2D. A lo largo de los años los algoritmos de RL se han vuelto de los más comunes en cuanto a redes neuronales para el control de este tipo de robots, así como han mostrado una significativa evolución. Asimismo, también se utilizan algoritmos del tipo actor-crítico, los cuales se benefician de la estimación de estado-acción y optimizan los parámetros de sus políticas (Mitriakov A. , Papadakis, Kerdreux, & Garlatti, 2021).

Un método adicional que combina técnicas basadas en control por entrenamiento, y control standard, es el método aplicado en el trabajo de Teymur Azayev y Karel Zimmermann. Esta publicación propone una máquina de estados similar a la propuesta por el control standard, con la diferencia que las transiciones de los estados son decididas por redes neuronales entrenadas por demostraciones, las cuales tienen como entrada información de distancia de los objetos al frente de cada Flipper, la orientación del robot y el estado anterior de las probabilidades de cada estado; mientras que su salida son las probabilidades de que el robot se encuentre en cada uno de los estados. Existen 7 estados y una red neuronal por cada estado, las cuales comparten la misma estructura, la red neuronal utilizada para definir la transición del estado en un tiempo “t” es la red neuronal correspondiente al estado identificado en el tiempo “t-1”. En la *Figura 1.3* se muestra la estructura de la red neuronal, por simplicidad, reducida a 3 estados y 3 periodos de tiempo (Azayev & Zimmermann, 2022).

Una vez es obtenido el estado más probable en la máquina de estado, se ejecuta la acción de control correspondiente a dicho estado, esta acción de control es sumada con una acción de control de estabilización y una acción de control de escape. La acción de control de estabilización proviene de un controlador PD que tiene como función estabilizar el ángulo de roll del robot. La acción de control de escape es una acción de control proporcional al “estancamiento”, que es definido como la diferencia entre la velocidad estimada del robot y la velocidad obtenida por el planeador de movimiento. Entre más “estancado” se encuentre el robot, más bajará el ángulo de los Flipper, de tal manera que intenten tocar el suelo.

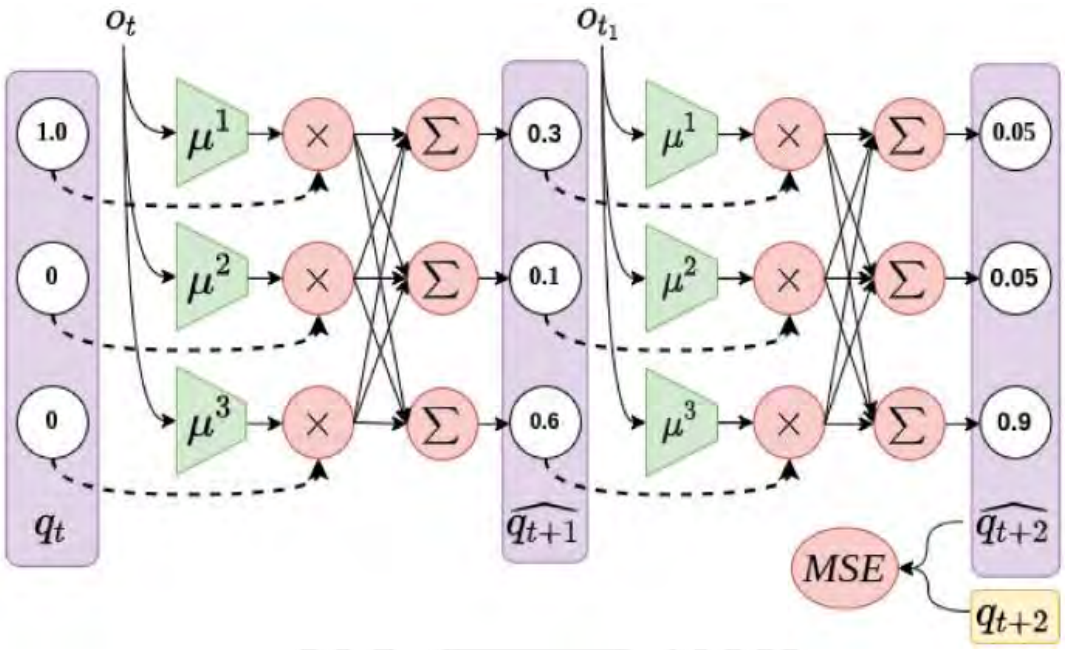

Figura 1.3. Estructura de red neuronal de máquina de estados. (Azayev & Zimmermann, 2022).

1.4.2.2 Algoritmos de control de plataforma móvil

El rol del controlador de la plataforma móvil es el de estabilizar la trayectoria de referencia teniendo en cuenta el error de modelamiento u otras formas de incertidumbre. Esta trayectoria de referencia es obtenida del planeador de movimiento.

El problema de estabilización de trayectoria se puede describir de la siguiente manera. Dada una ecuación diferencial controlada, y una trayectoria de referencia, se debe encontrar una señal de control que, al ser usada como entrada de la ecuación diferencial, permita a la misma converger a la trayectoria de referencia. En muchos casos, analizar la estabilidad de las trayectorias puede ser reducido a determinar la estabilidad del origen en un sistema variable en el tiempo. Algunos calificativos útiles para diversos tipos de estabilidad incluyen:

1. *Estabilidad asintótica uniforme*: Es la capacidad de un sistema dinámico para regresar a un estado de equilibrio estable independientemente de las condiciones iniciales.

2. *La estabilidad exponencial:* asegura que la tasa de convergencia está limitada por una decaída exponencial. Un asunto delicado para tener en

cuenta es que las especificaciones del controlador suelen expresarse en términos del error de seguimiento asintótico a medida que el tiempo tiende a infinito. En la práctica, las trayectorias de referencia son finitas, por lo que también debería considerarse la respuesta transitoria del sistema (Paden, ˇC´ap, Zheng Yong, Yershov, & Frazzoli, 2016).

Teniendo en cuenta los criterios previamente mencionados, se explican distintos algoritmos de control utilizados en plataformas móviles, los cuales son usados para estabilización de trayectorias.

Diseño de control basado en Lyapunov: Se define el error de seguimiento en un marco de coordenadas fijas a la plataforma móvil. El error de seguimiento se puede expresar mediante un cambio de base del marco de coordenadas inerciales usando la trayectoria de referencia. El análisis local de la ley de control provee estabilidad exponencial local. Sin embargo, para que el sistema pueda ser invariante en el tiempo, la velocidad linear y angular de referencia deben ser invariables. También se proponen utilizar un diseño que aplique back-stepping para lograr una estabilidad exponencial local uniforme para un dominio de tiempo y referencias variables (Jiang & Nijmeijer, 1997).

Linealización de Retroalimentación de Salida: Con el fin de utilizar la plataforma móvil con mayores velocidades, es apropiado limitar el ángulo de giro para que tenga un movimiento continuo. De esta manera, se vuelve más difícil diseñar un controlador a partir de dichas consideraciones geométricas. Una opción adecuada en este caso es linealizar la salida del sistema. Esto no se logra fácilmente utilizando las posiciones de las ruedas delanteras o traseras. Se propone una salida que simplifica la linealización de la retroalimentación, donde se selecciona un punto delante del vehículo por cualquier distancia mayor a 0, alineado con el ángulo de dirección (Novel, Campion, & Bastin, 1995).

Control Predictivo sin Restricciones con Modelos Cinemáticos: El control predictivo por modelo es un diseño de control que suele ser bastante efectivo en la conducción autónoma de vehículos. Conceptualmente, este se basa en resolver un problema de planeamiento de movimiento en un horizonte de tiempo reducido, tomar un pequeño intervalo de la solución y aplicarlo al sistema. Mientras se ejecuta esta solución, el problema de planeamiento de movimiento

es resuelto nuevamente para encontrar una solución de control para el siguiente intervalo de tiempo. Este tipo de control puede ser aplicado sin restricciones en sus entradas y salidas utilizando un modelo cinemático linealizado e implementado utilizando programación cuadrática linear variante en el tiempo (Raffo, Gomes, Normey-Rico, & Kelber, 2009).

Controladores de Seguimiento de Trayectorias: En la literatura se muestra la implementación de un controlador predictivo utilizando un modelo dinámico con comportamiento no lineal de los neumáticos. La estrategia completa de control predictivo no lineal se llevó a cabo en simulación y se demostró que estabilizaba una maniobra simulada de emergencia en condiciones de hielo con una frecuencia de control de 20 Hz. Sin embargo, con un horizonte de control de solo dos pasos de tiempo, el tiempo de cálculo fue tres veces el tiempo de muestreo del controlador, lo que no hizo posible la validación experimental (Falcone, Borrelli, Asgari, Tseng, & Hrova, 2007). También se investigó un enfoque basado en una única linealización sobre el estado de la plataforma móvil en el paso de tiempo actual. La reducida complejidad de resolver el programa cuadrático resultó en un tiempo de cálculo aceptable, y se informaron resultados experimentales exitosos para la conducción en condiciones de hielo a velocidades de hasta 21 m/s (Falcone, Borrelli, Tseng, Asgari, & Hrovat, 2008).

1.4.5 Análisis del estado del arte

Luego de realizar un estudio a la literatura actual acerca del estado del arte de los robots articulados con orugas se puede concluir que existe un problema de control abierto debido a que la mayoría de los métodos explorados no consideran completamente la morfología de este tipo de robots, tratando el problema de control como el de un robot diferencial, que, si bien este acercamiento ha generado resultados, no aprovecha completamente las potencialidades de la estructura del robot.

1.5 Planteamiento de la propuesta y justificación

En base a lo anterior se propone diseñar y simular un sistema de control avanzado no lineal para un robot móvil articulado con orugas. Dicho sistema de

control consta de dos (02) controladores de seguimiento de trayectorias. El primero para la plataforma móvil del robot móvil articulado con orugas. El segundo para el sistema de articulaciones. La selección de cada controlador debe ser realizada comparando los resultados de distintos controladores no lineales, eligiendo aquel que ofrezca la mejor respuesta temporal y mayor robustez frente a perturbaciones.

La aplicación de este tipo de robots va a permitir incrementar la productividad y eficiencia de las operaciones, así como garantizar la seguridad y salud a los trabajadores. Al desarrollar el conocimiento y tecnología propietaria se tiene la posibilidad de atender nuevas aplicaciones a la medida de la gran minería, y también de la mediana minería que cuenta con un mercado relevante. Es de gran demanda en la industria minera el reducir los accidentes durante la operación.

La ventaja de la inspección robótica semiautónoma o autónoma sobre la inspección manual, es que permite un mayor nivel de garantía y control de calidad con menos errores, lo cual es un factor importante. Los niveles de concentración del personal de inspección suelen disminuir con el tiempo debido a la repetitividad de las tareas, agotamiento mental, y disminución gradual de la productividad. Agiliza los procesos, ya que los robots de inspección están diseñados para tareas repetitivas sin comprometer la precisión. Además, garantiza la seguridad del entorno de trabajo, ya que pueden trabajar en condiciones peligrosas e inestables. El uso de robots de inspección también reduce el tiempo de inactividad, maximiza la capacidad organizativa, y disminuye el impacto económico de las inspecciones. El uso de algoritmos utilizando inteligencia articial, permite detectar defectos y problemas a través de la inspección visual. La integración con *Internet of Things* permite que los sensores en los robots detecten cambios en el ruido, la vibración, o la aspereza. Por lo tanto, ayudan en el análisis predictivo de fallas al proporcionar datos de salud en tiempo real de elementos y máquinas.

La hipótesis es que, mediante el uso de robots móviles equipados con sensores especializados, algoritmos de navegación autónoma, y visión por computadora, se pueden realizar inspecciones y diagnósticos precisos de la infraestructura, minimizando el riesgo para el personal que actualmente realiza esta tarea.

1.6 Objetivos de la tesis

1.6.1 Objetivo general

Diseñar un sistema de control avanzado no lineal para la navegación de un robot móvil articulado con orugas el cual incorpore un controlador de seguimiento de trayectorias para la plataforma móvil del robot, y otro para el sistema de articulaciones, ambos considerando la cinemática de los flippers y orugas del robot.

1.6.2 Objetivos específicos

- Realizar el estudio del estado del arte de los robots articulados con orugas.
- Desarrollar un modelo matemático completo del robot móvil articulado con orugas que describa de manera adecuada el comportamiento dinámico del sistema y pueda ser utilizado para simular el sistema de control, considerando ambos tipos de desplazamientos del robot.
- Simular el robot móvil articulado con orugas en un ambiente virtual, validando la simulación mediante la comparación de su respuesta con el modelo matemático.
- Diseñar, simular y evaluar la estabilidad de los controladores de seguimiento de trayectorias para la plataforma móvil del robot y su sistema de articulaciones.
- Proponer la implementación en hardware del sistema de control, especificando la configuración y selección de los componentes a utilizar.

CAPÍTULO 2: MODELADO DEL ROBOT

2.1 Introducción

En el campo del control, la obtención del modelo de la planta de estudio es fundamental para el desarrollo e implementación de distintos tipos de controladores, ya que estos, en la mayoría de los casos requieren de un modelo matemático para su uso. Asimismo, el modelamiento preciso del sistema estudiado es fundamental para obtener la respuesta deseada en el tiempo y en frecuencia.

El presente capitulo está centrado en el modelamiento matemático de un robot móvil articulado con orugas. En primer lugar, se presentará el modelo matemático del robot, incluyendo ecuaciones cinemáticas y dinámicas; detallando las ecuaciones y principios físicos de su movimiento, y las fuerzas involucradas. A manera de hacer más sencilla la tarea de plantear el modelo, este proceso se desglosará en modelo de la plataforma móvil con orugas y el modelo de las articulaciones

Luego, se evaluará el comportamiento del modelo obtenido en distintas condiciones de operación y se comprobará si el modelo funciona de la manera esperada. Finalmente, se discutirán las conclusiones derivadas de este capítulo, destacando las implicaciones del modelo, así como posibles áreas de mejora.

2.2 Modelado Matemático

Con el fin de desarrollar el modelo matemático de un robot articulado con orugas, es necesario considerar tanto el modelo cinemático y dinámico de la plataforma móvil con orugas como el modelo de sus contrapartes correspondientes a las articulaciones activas con las que cuenta el robot. La plataforma móvil cuenta con dos (02) grados de libertad, los cuales son dos orugas, y puede ser modelada de manera similar a un robot diferencial (Al-Jarrah, Salah, & Almomani, 2019), mientras que las articulaciones activas o flippers son elementos que pueden rotar alrededor de su articulación, añadiendo 4 grados de libertad adicionales, los cuales son usados para controlar el ángulo de los flippers, ayudando al robot a adaptarse a diferentes terrenos. Dichos flippers

también son una continuación de las orugas, por lo cual también cuentan con tracción y previenen que el robot se deslice al momento de escalar obstáculos (Chen, Wu, Wang, Yang, & Zhang, Motion Planning for Autonomous Climbing Stairs for Flipper Robot, 2020) . En la *Figura 2.1*, se puede observar un ejemplo de robot articulado con orugas, el robot TRADR, en el cual se detallan sus partes.

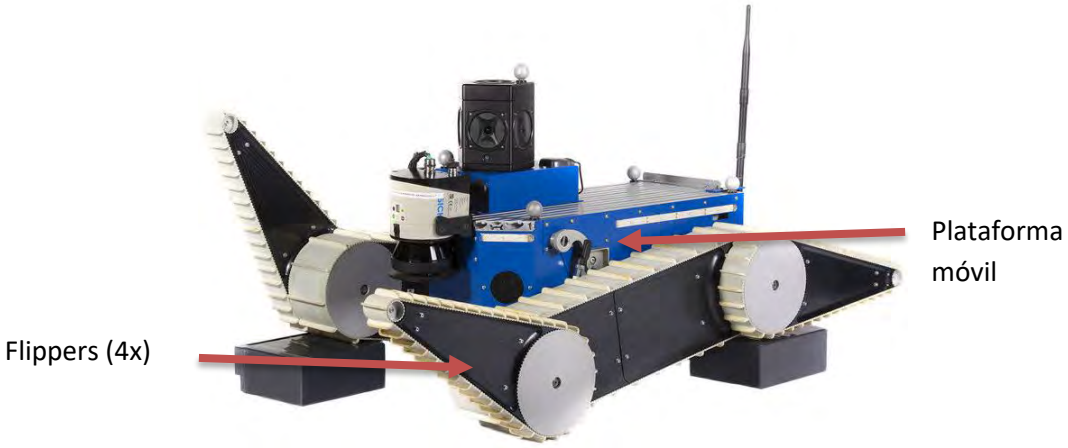

Figura 2.1. Partes de robot articulado con orugas. Adaptado de: (Mario & Pirri, 2019)

2.2.1 Modelado de la plataforma móvil

Considerando el esquema de la plataforma móvil de 2 grados de libertad conformada por 2 orugas mostrado en la *Figura 2.2*. Se deriva el modelo matemático utilizado para simular el movimiento del sistema y su dinámica. A continuación, se muestra tanto el modelo cinemático como dinámico.

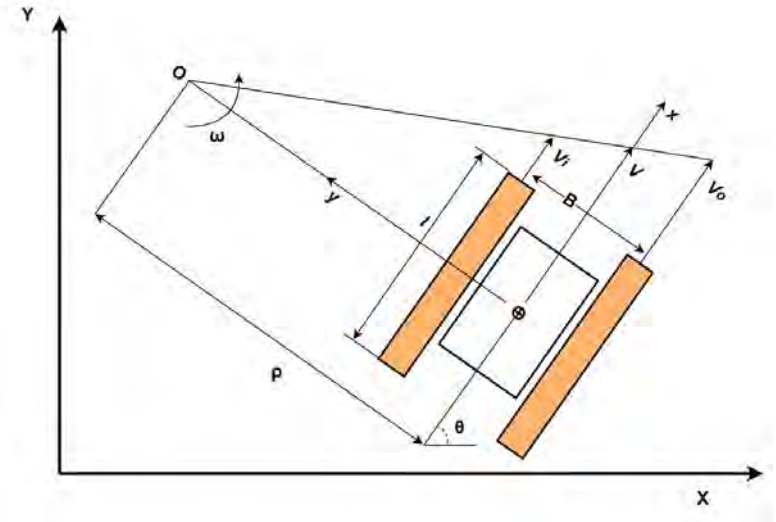

Figura 2.2. Esquema para modelo cinemático de plataforma con orugas. (Al-Jarrah, Salah, & Almomani, 2019)

Modelo cinemático

Las ecuaciones utilizadas en el modelo cinemático de la plataforma móvil comparten variables con la mostradas en la Figura 2.2. Las posiciones actuales del robot se representan como x e y, las cuales se encuentran en el sistema de coordenadas global X-Y. El ángulo de inclinación del robot con respecto al eje X global es θ, mientras que la velocidad de traslación hacia adelante es V. Además, las velocidades de las orugas izquierda y derecha se definen como Vi y Vo, respectivamente. La velocidad angular del robot se define como ω ,mientras que el radio de curvatura del robot se denota como ρ. En relación con la geometría del vehículo se define la distancia entre las orugas como B, y el largo de las orugas como l.

El modelo cinemático de la plataforma móvil del robot puede ser descrito como el de un robot diferencial si es que no se tiene en cuenta el deslizamiento (Vadym & Aleksey, 2012). En este caso las ecuaciones son las siguientes:

$$\begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} Vcos\theta \\ Vsin\theta \\ \omega \end{bmatrix} \tag{2.1}$$

Donde V_x *y* V_y son las velocidades de traslación en el eje x e y, respectivamente. La velocidad angular del robot también es definida como ω. Las velocidades de las orugas Vi y Vo se describen de la siguiente manera:

$$\begin{bmatrix} V_i \\ V_o \end{bmatrix} = \begin{bmatrix} r\omega_i \\ r\omega_o \end{bmatrix} \tag{2.2}$$

En este caso r es el radio de los sprockets de la oruga, mientras que ω_i y ω_o son las velocidades angulares de los motores de tracción del robot. Por lo tanto, la velocidad lineal de avance del robot es descrita con la siguiente ecuación:

$$V = \frac{Vi + Vo}{2} \tag{2.3}$$

Utilizando la anterior ecuación, la velocidad de rotación es descrita como:

$$\dot{\theta} = \frac{V}{\rho} = \frac{Vi + Vo}{2\rho} = \frac{Vo - Vi}{B} \tag{2.4}$$

Donde ρ es el radio de curvatura del robot, y B es la distancia entre orugas del robot. Para añadir deslizamiento a la ecuación, los parámetros i_i e i_o pueden ser utilizados. Estos parámetros representan el deslizamiento longitudinal de ambas orugas, y dependen en las propiedades de las orugas, el terreno, y las fuerzas de empuje. Estos parámetros pueden ser hallados de manera experimental o aproximados mediante ecuaciones (Al-Jarrah, Salah, & Almomani, 2019). Añadiendo estos parámetros, la ecuación (2.2) se vuelve la siguiente:

$$\begin{bmatrix} V_i \\ V_o \end{bmatrix} = \begin{bmatrix} r\omega_i(1-i_i) \\ r\omega_o(1-i_o) \end{bmatrix} \tag{2.5}$$

Donde i_i corresponde a la oruga derecha, e i_o corresponde a la oruga izquierda. Reemplazando en (2.3) y (2.4):

$$\begin{bmatrix} V \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dfrac{r\omega_o(1-i_o) + r\omega_i(1-i_i)}{2} \\ \dfrac{r\omega_o(1-i_o) - r\omega_o(1-i_o)}{B} \end{bmatrix} \tag{2.6}$$

En consecuencia, el radio de curvatura ρ' considerando deslizamiento es el mostrado a continuación:

$$\rho' = \frac{B[r\omega_o(1-i_o) + r\omega_i(1-i_i)]}{2[r\omega_o(1-i_o) - r\omega_o(1-i_o)]} \tag{2.7}$$

Los parámetros de deslizamiento (i_i, i_o) pueden ser aproximados utilizando las ecuaciones descritas en (2.8) (Al-Jarrah, Salah, & Almomani, 2019).

$$\begin{bmatrix} i_i \\ i_o \\ Fmax \end{bmatrix} = \begin{bmatrix} \dfrac{K/l}{\ln(Fi/Fmax)} \\ \dfrac{K/l}{\ln(Fo/Fmax)} \\ 0.5(A\psi + Wtan\theta) \end{bmatrix} \tag{2.8}$$

Donde Fi y Fo son las fuerzas de empuje correspondientes a las orugas, A es el área de contacto de las orugas, ψ depende de la naturaleza del terreno, K es el modulo de deformación del terreno, l es el largo de las orugas, y W es el peso del robot.

Modelo dinámico

El comportamiento de giro de un robot con orugas depende de las fuerzas de empuje de las orugas, Fi y Fo, de la fuerza de resistencia Ftot, del momento de resistencia al giro Mr ejercido en las orugas por el piso, y de los parámetros de vehículo mostrados en la Figura 2.3. Tener en cuenta que las fuerzas de empuje dependen de los torques de los motores de tracción, la ratio de reducción, y el radio del sprocket. Considerando la aplicación del robot, se establece que la velocidad del robot es suficientemente baja como para no considerar la fuerza centrífuga. Bajo estas condiciones, el comportamiento del vehículo se puede describir por las siguientes ecuaciones de movimiento:

$$\begin{bmatrix} m\dfrac{dV}{dt} \\ I\dfrac{d\dot{\theta}}{dt} \end{bmatrix} = \begin{bmatrix} F_i + F_o - F_{tot} \\ M_a - M_r \end{bmatrix} \tag{2.9}$$

Donde m e I son la masa y el momento de inercia del robot, respectivamente, $M_a = B/(F_o - F_i)/2$ es el momento aplicado debido a las fuerzas de empuje en la plataforma móvil, mientras que M_r es el momento generado a manera de reacción del giro del robot.

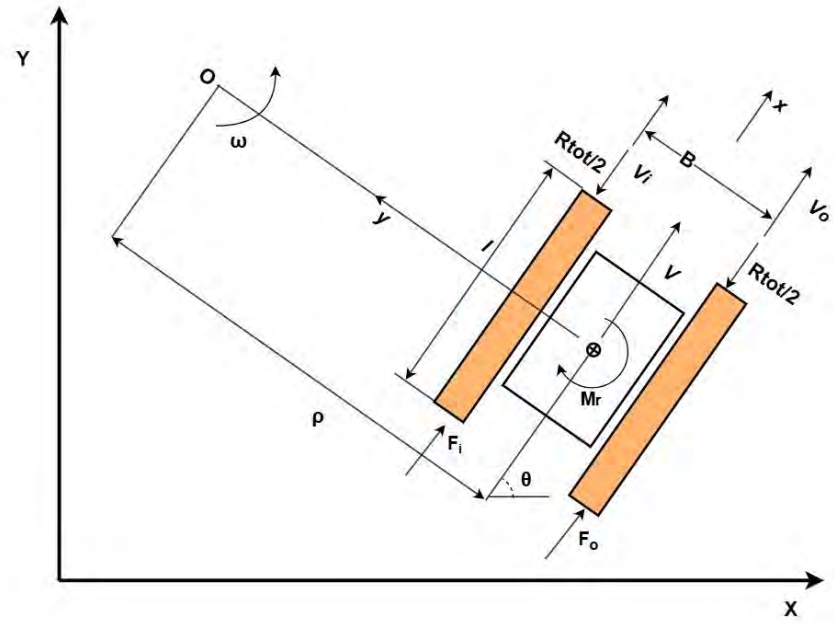

Figura 2.3. Esquema para modelo dinámico de plataforma con orugas. (Al-Jarrah, Salah, & Almomani, 2019)

2.2.2 Modelado de las articulaciones

Considerando la Figura 2.4, que presenta el diseño de las cuatro articulaciones en conjunto con la plataforma móvil. A continuación, se muestra el proceso en el cual se adquiere tanto el modelo cinemático como dinámico.

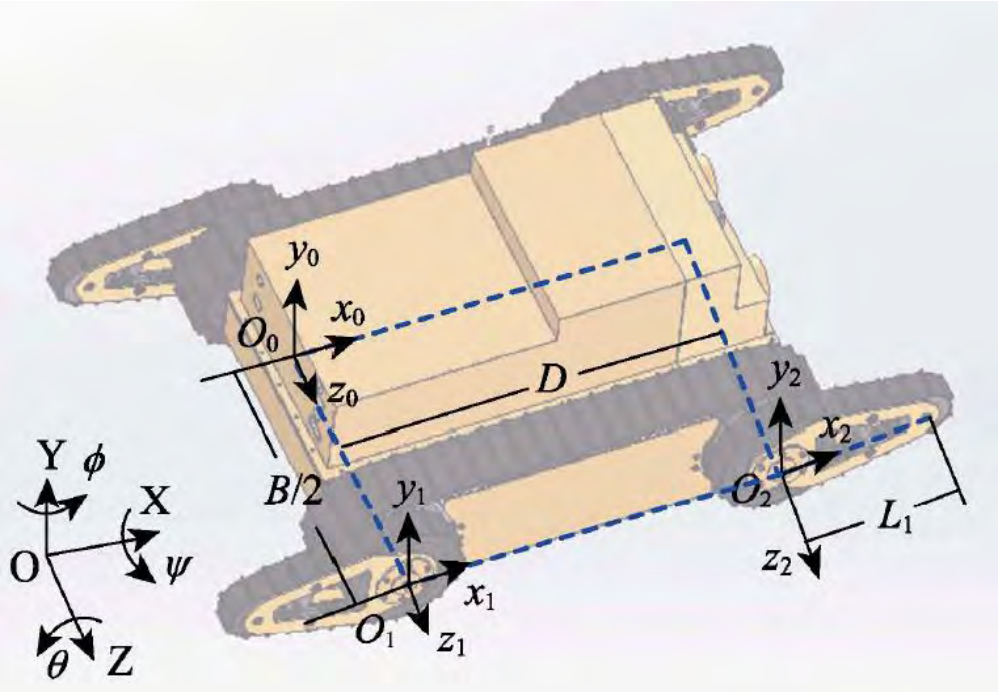

Figura 2.4. Coordenadas para el modelo cinemático de las articulaciones. (Wang, y otros, 2018)

Modelo cinemático

Cada Flipper contiene un grado de libertad, el cual le permite al Flipper girar con respecto a su eje de rotación. Este giro le permite al robot adaptarse a diferentes terrenos, ya que le da la capacidad de modificar su posición en la coordenada Z, así como sus ángulos de pitch y roll. Asimismo, las orugas colocadas en los Flippers previenen que el robot se resbale al momento de atravesar obstáculos. Con la finalidad de controlar estos 4 grados de libertad adicionales (un grado de libertad por Flipper), se establece un modelo cinemático para determinar la postura del robot y la ubicación de su centro de masa de la plataforma móvil en el eje Z.

En la *Figura 2.5* se muestra una simplificación en 2D del robot articulado con orugas, la cual representa la vista lateral, y será utilizada para modelar las ecuaciones de posición en el eje Z y ángulo pitch. En esta figura, P1, P2

representan el punto de contacto de los Flippers con el terreno, siendo P1 correspondiente a los Flippers delanteros y P2 correspondiente a los traseros. El eje de giro de los Flippers son F y B, mientras que el centro de la plataforma móvil se considera como el punto E. L es la distancia entre los ejes de giro de los Flippers y Lf es la longitud de los Flippers. Asimismo, $\theta_1 \, y \, \theta_2$ son las inclinaciones de los Flippers delanteros y traseros con respecto a la plataforma móvil, la cual es definida con la recta $\overrightarrow{FB}$. Finalmente, se representa el ángulo pitch como φ. Es necesario tener en cuenta que esta representación asume que ambos Flippers delanteros tienen el mismo ángulo de inclinación, y que lo mismo se cumple para ambos Flippers traseros.

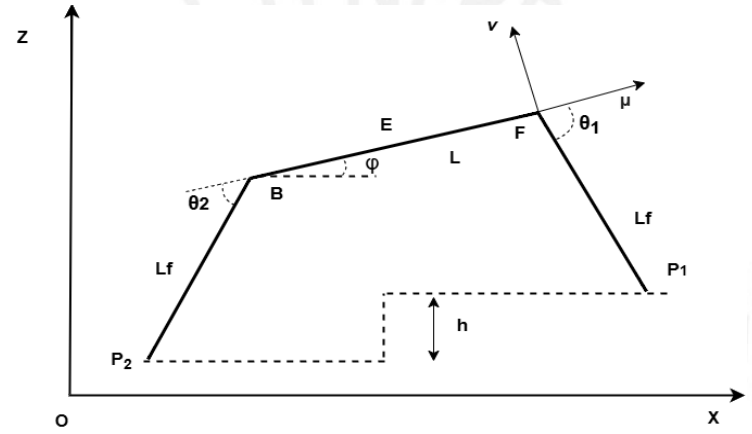

Figura 2.5. Diagrama simplificado 2D del robot articulado con orugas (vista lateral). Elaboración propia

De la geometría descrita en la Figura 2.5, se obtiene:

$$Z = P_{2z} + Lf.\sin(\theta_2 + \varphi) + \frac{L}{2}\sin(\varphi) \tag{2.10}$$

$$\varphi = \arcsin\left(\frac{Lf}{L}.(\sin(\theta_1 - \varphi) - \sin(\theta_2 + \varphi))\right) \tag{2.11}$$

$$\theta_1 = \arcsin\left(\frac{E_z + \frac{L}{2} * \sin(\varphi)}{Lf}\right) + \varphi \tag{2.12}$$

$$\theta_2 = \arcsin\left(\frac{E_z - \frac{L}{2} * \sin(\varphi)}{Lf}\right) - \varphi \tag{2.13}$$

Donde P_{2z} y E_z son las coordenadas en el eje Z del punto P_2 y E, respectivamente.

Definiendo las variables de control ω_1 y ω_2 como la velocidad de giro de los Flippers delanteros y traseros respectivamente, se puede definir los ángulos de θ_1 y θ_2 de la siguiente manera:

$$\theta_1 = \arcsin\left(\frac{Z + \frac{L}{2} * \sin(\varphi)}{Lf}\right) + \varphi \tag{2.14}$$

$$\theta_2 = \arcsin\left(\frac{Z - \frac{L}{2} * \sin(\varphi)}{Lf}\right) - \varphi \tag{2.15}$$

Para obtener el modelo cinemático del sistema, primero derivamos (2.11), y reemplazamos (2.14) y (2.15) donde es pertinente. La ecuación obtenida es:

$$\dot{\varphi} = f_3(\theta_1, \theta_2, \varphi).\omega_1 + f_4(\theta_1, \theta_2, \varphi).\omega_2 \tag{2.16}$$

Donde se define las siguientes subfunciones:

$$f_3(\theta_1, \theta_2, \varphi) = \left(\frac{\frac{Lf}{L}(\cos(\theta_1 - \varphi))}{\sqrt{1 - \left(\frac{Lf}{L}(\sin(\theta_1 - \varphi) - \sin(\theta_2 + \varphi))\right)^2} + \frac{Lf}{L}(\cos(\theta_1 - \varphi) + \cos(\theta_2 + \varphi))}\right) \tag{2.17}$$

$$f_4(\theta_1, \theta_2, \varphi) = \left(\frac{\frac{Lf}{L}(\cos(\theta_2 + \varphi))}{\sqrt{1 - \left(\frac{Lf}{L}(\sin(\theta_1 - \varphi) - \sin(\theta_2 + \varphi))\right)^2} + \frac{Lf}{L}(\cos(\theta_1 - \varphi) + \cos(\theta_2 + \varphi))}\right) \tag{2.18}$$

Luego, se deriva (2.10) y se reemplaza (2.16), obteniendo:

$$\dot{Z} = f_1(\theta_1, \theta_2, \varphi).\omega_1 + f_2(\theta_1, \theta_2, \varphi).\omega_2 \tag{2.19}$$

Donde se define las siguientes subfunciones:

$$f_1(\theta_1, \theta_2, \varphi) =$$

$$\left(\frac{\frac{Lf}{L}(\cos(\theta_1 - \varphi))(\frac{L}{2}\cos(\varphi) + Lf.\cos(\theta_2 + \varphi))}{\sqrt{1 - \left(\frac{Lf}{L}(\sin(\theta_1 - \varphi) - \sin(\theta_2 + \varphi))\right)^2} + \frac{Lf}{L}(\cos(\theta_1 - \varphi) + \cos(\theta_2 + \varphi))} \right) \tag{2.20}$$

$$f_2(\theta_1, \theta_2, \varphi) =$$

$$\left(\frac{\frac{Lf}{L}(\cos(\theta_2 + \varphi))(\frac{L}{2}\cos(\varphi) + Lf.\cos(\theta_2 + \varphi))}{\sqrt{1 - \left(\frac{Lf}{L}(\sin(\theta_1 - \varphi) - \sin(\theta_2 + \varphi))\right)^2} + \frac{Lf}{L}(\cos(\theta_1 - \varphi) + \cos(\theta_2 + \varphi))} + Lf.\cos(\theta_2 + \varphi) \right) \tag{2.21}$$

Finalmente, se obtiene la representación espacio-estado del sistema:

$$\dot{X} = \begin{bmatrix} \dot{Z} \\ \dot{\varphi} \\ \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z \\ \varphi \\ \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} f_1(\theta_1, \theta_2, \varphi) & f_2(\theta_1, \theta_2, \varphi) \\ f_3(\theta_1, \theta_2, \varphi) & f_4(\theta_1, \theta_2, \varphi) \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \tag{2.22}$$

Luego de obtener el modelo de la posición en Z y el pitch, se hace el modelado del ratio de cambio del ángulo de roll.

En la *Figura 2.6* se muestra una simplificación en 2D de la vista frontal del robot articulado con orugas, y será utilizada para modelar las ecuaciones del ángulo de roll. En esta figura, P1, P2 representan el punto de contacto de los Flippers

con el terreno, siendo P1 correspondiente a los Flippers delanteros y P2 correspondiente a los traseros. La distancia h representa la diferencia de elevación entre P1 y P2 en el eje Z. El eje de giro de los Flippers son F y B, mientras que el centro de la plataforma móvil se considera como el punto E. D es la distancia entre los Flippers y Lf es la longitud de los Flippers. Asimismo, $\theta_a \, y \, \theta_b$ son las inclinaciones de ambos Flippers del lado derecho y ambos Flippers del lado izquierdo con respecto a la plataforma móvil, la cual es definida con la recta $\overrightarrow{FB}$. Finalmente, se representa el ángulo de roll como ϕ. Es necesario tener en cuenta que esta representación asume que ambos Flippers ubicados en el lado derecho del robot tienen el mismo ángulo de inclinación, y que lo mismo se cumple para ambos Flippers del lado izquierdo.

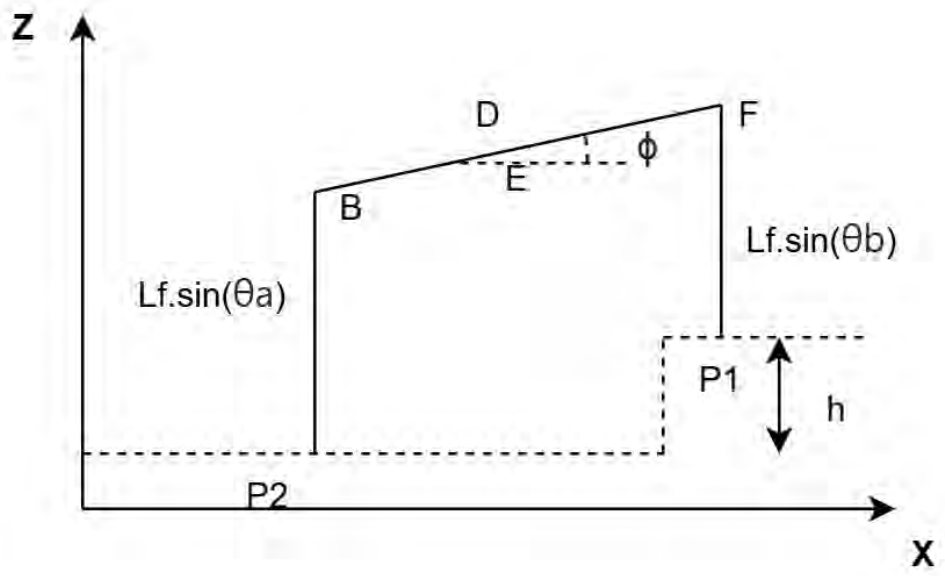

Figura 2.6. Diagrama simplificado 2D del robot articulado con orugas (vista frontal). Elaboración propia

De la geometría descrita en la *Figura 2.6*, se puede obtener la siguiente ecuación:

$$\phi = \arcsin\left(\frac{Lf.(\sin(\theta_a).\cos(\phi) - \sin(\theta_b).\cos(\phi)) - h}{D}\right) \qquad (2.23)$$

Definiendo las variables de control ω_a y ω_b como la velocidad angular de giro de los Flippers del lado derecho e izquierdo respectivamente, se puede definir las derivadas de θ_a y θ_b de la siguiente manera:

$$\dot{\theta_a} = \omega_a \tag{2.24}$$

$$\dot{\theta_b} = \omega_b \tag{2.25}$$

Se deriva la ecuación para obtener la ecuación cinemática del ángulo de roll. La ecuación obtenida es:

$$\dot{\phi} = f_5(\theta_a, \theta_b, \phi).\omega_a + f_6(\theta_a, \theta_b, \phi).\omega_b \tag{2.26}$$

Donde definimos las siguientes subfunciones:

$$f_5(\theta_a, \theta_b, \phi) = \frac{\frac{Lf}{D}(\cos(\theta_a).\cos(\phi))}{\sqrt{1 - \left(\frac{Lf}{D}(\sin(\theta_a).\cos(\phi) - \sin(\theta_b).\cos(\phi))\right)^2} + \frac{Lf}{D}(\sin(\theta_a).\sin(\phi) - \sin(\theta_b).\sin(\phi))} \tag{2.27}$$

$$f_6(\theta_a, \theta_b, \phi) = \frac{-\frac{Lf}{D}(\cos(\theta_b).\cos(\phi))}{\sqrt{1 - \left(\frac{Lf}{D}(\sin(\theta_a).\cos(\phi) - \sin(\theta_b).\cos(\phi))\right)^2} + \frac{Lf}{D}(\sin(\theta_a).\sin(\phi) - \sin(\theta_b).\sin(\phi))} \tag{2.28}$$

Finalmente, se obtiene la representación espacio-estado del sistema:

$$\dot{X} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta_a} \\ \dot{\theta_b} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta_a \\ \theta_b \end{bmatrix} + \begin{bmatrix} f_5(\theta_a, \theta_b, \phi) & f_6(\theta_a, \theta_b, \phi) \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_a \\ \omega_b \end{bmatrix} \tag{2.29}$$

Luego de definir las ecuaciones que describen la cinemática de los ángulos de los Flippers en pares de dos, se define la cinemática de los Flippers individuales por superposición de efectos en la ecuación (2.30). La notación de cada Flipper de muestra en la *Figura 2.7*.

$$\dot{\theta} = \begin{bmatrix} \dot{\theta_{1a}} \\ \dot{\theta_{1b}} \\ \dot{\theta_{2a}} \\ \dot{\theta_{2b}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_a \\ \omega_b \end{bmatrix} \tag{2.30}$$

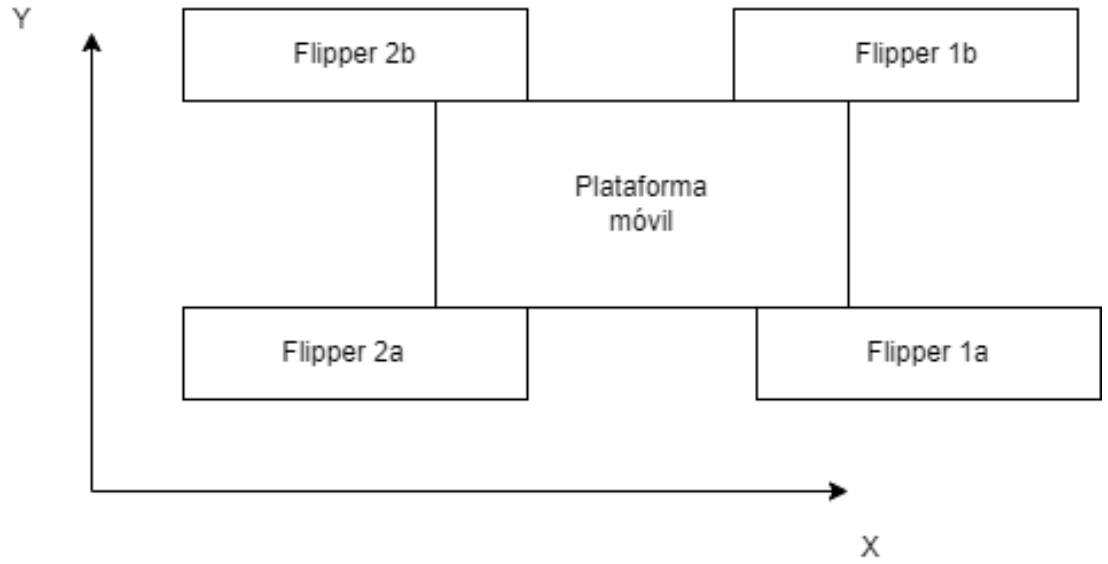

Figura 2.7. Notación utilizada para los Flippers. Elaboración propia

Asimismo, al momento de realimentar los ángulos de los Flippers se tendrán las siguientes consideraciones:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_a \\ \theta_b \end{bmatrix} = \begin{bmatrix} \max(\theta_{1a}, \theta_{1b}) \\ \max(\theta_{2a}, \theta_{2b}) \\ \max(\theta_{1a}, \theta_{2a}) \\ \max(\theta_{1b}, \theta_{2b}) \end{bmatrix} \tag{2.31}$$

Uniendo las ecuaciones (2.22), (2.29) y (2.30), se obtiene la ecuación espacio-estado de las articulaciones.

$$\begin{bmatrix} \dot{Z} \\ \dot{\varphi} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\phi} \\ \dot{\theta}_a \\ \dot{\theta}_b \\ \dot{\theta_{1a}} \\ \dot{\theta_{1b}} \\ \dot{\theta_{2a}} \\ \dot{\theta_{2b}} \end{bmatrix} = \begin{bmatrix} f_1 & f_3 & 0 & 0 \\ f_3 & f_4 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & f_5 & f_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_a \\ \omega_b \end{bmatrix} \quad (2.32)$$

Modelo dinámico

El comportamiento de giro de un Flipper se halla de manera similar al de la plataforma móvil (Al-Jarrah, Salah, & Almomani, 2019). Este depende de la fuerza de empuje generada por su motor F_{fi}, de la fuerza de resistencia F_{ftot}, del momento de resistencia al giro M_{fri} ejercido en el Flipper por el terreno, y de los parámetros de vehículo mostrados en el análisis cinemático. Bajo estas condiciones, el comportamiento del Flipper se puede describir por la siguiente ecuación de movimiento:

$$\begin{bmatrix} m_1 \dfrac{dVi}{dt} \\ I \dfrac{d\dot{\alpha}_\iota}{dt} \end{bmatrix} = \begin{bmatrix} F_{fi} - F_{ftot} \\ M_{fai} - M_{fri} \end{bmatrix} \quad (2.33)$$

Donde Vi, α_i son las velocidades y aceleraciones de cada Flipper, respectivamente. Por otro lado, m_1 e I son la masa y el momento de inercia del Flipper, respectivamente, $M_{fai} = L1/(F_{fi})/2$ es el momento aplicado debido a las fuerzas de empuje en el Flipper.

2.3 Pruebas de los modelos matemático

En la presente sección se comprobará el funcionamiento de los modelos obtenidos en la sección 2.2. Por este motivo, se simularán diferentes valores en las variables de control de los modelos para evaluar si el comportamiento es el esperado.

2.3.1 Pruebas del modelo de la plataforma móvil

El primer modelo en ser evaluado es la plataforma móvil. De acuerdo con la ecuación 2.1, se pueden deducir distintas propiedades. En primer lugar, la plataforma iniciará orientada al eje X. Por otro lado, las entradas del sistema son la velocidad lineal y la velocidad angular. De acuerdo con las ecuaciones de la plataforma móvil, si la velocidad lineal es cero, y la velocidad angular es distinta de cero la plataforma es capaz de girar sobre su propio eje con un radio de giro nulo. Además, si la velocidad lineal es distinta de cero y la velocidad angular es cero, la plataforma se movilizará en línea recta. Finalmente, si la velocidad lineal y angular son constantes, la plataforma seguirá una trayectoria circular. Con el fin de ejemplificar y comprobar las propiedades previamente descritas, se formará la letra griega “phi” (φ), utilizando la trayectoria de la plataforma móvil, variando las variables de control de su modelo matemático.

Con el fin de obtener la trayectoria deseada empleando un control en lazo abierto, se aplican los siguientes pasos. Primero, el robot debe girar 90° en sentido horario para estar alineado con el eje Y negativo, Luego el robot debe mantener una velocidad angular antihoraria constante y una velocidad linear constante para describir una media circunferencia. A continuación, se debe reducir la velocidad lineal a la mitad hasta generar una nueva media circunferencia con la mitad del radio de la anterior. Finalmente, el robot debe avanzar con una velocidad lineal positiva y una velocidad angular nula para terminar de generar la trayectoria deseada.

En la *Figura 2.8* se pueden ver las variables de control a lo largo del tiempo, mientras que en la Figura 2.9 se puede ver la trayectoria obtenida utilizando el modelo matemático, la cual es aproximada a la letra “phi”, por lo cual se comprueba que, utilizando un control en lazo abierto, el modelo matemático de la plataforma móvil se comporta de la manera esperada.

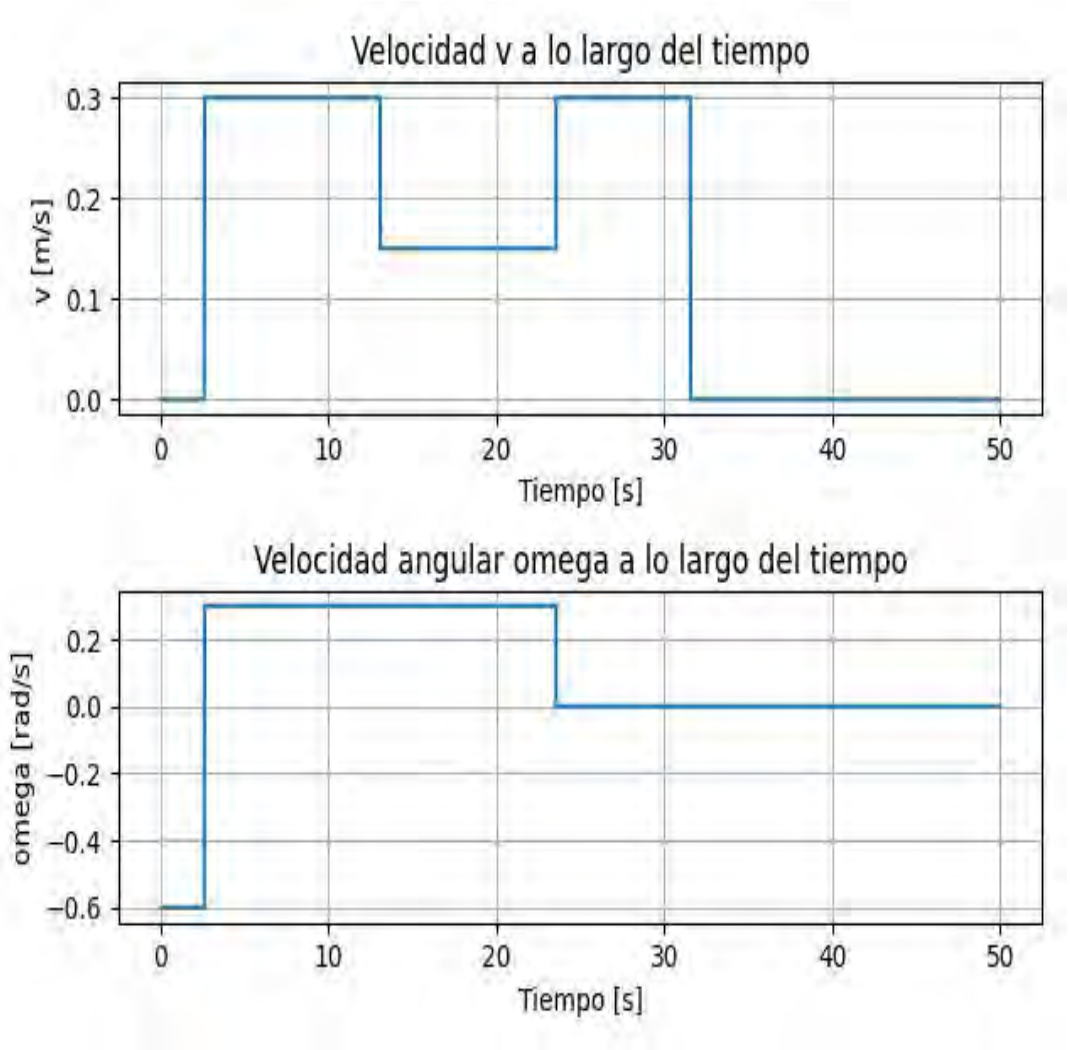

Figura 2.8. Variables de control utilizadas en la plataforma móvil. Elaboración propia

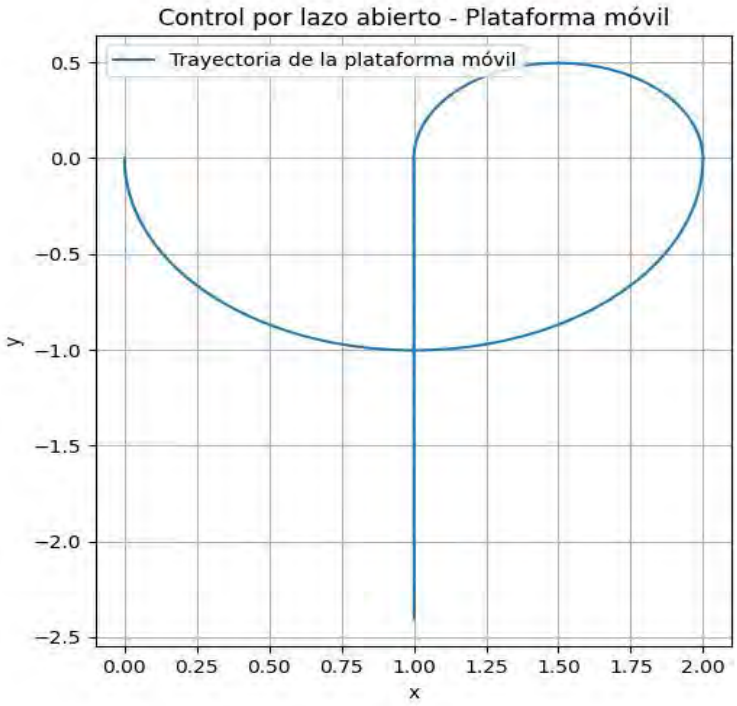

Figura 2.9. Trayectoria descrita por modelo de plataforma móvil. Elaboración propia

2.3.2 Pruebas del modelo del sistema de flippers.

El segundo modelo en ser evaluado el sistema de flippers. De acuerdo con la ecuación 2.22, se pueden deducir distintas propiedades. Si las velocidades angulares de los flipper traseros y delanteros son iguales, la posición en Z aumentará si las velocidades son positivas, o disminuirá si las velocidades son negativas, pero sin cambiar el ángulo de roll en el proceso. Por otro lado, si las velocidades angulares son opuestas, el ángulo de roll aumentará si $\omega 1$ es positivo, o disminuirá sí $\omega 1$ es negativo, pero sin afectar la posición en Z.

Con el fin de ejemplificar y comprobar estas propiedades, se seguirá la siguiente secuencia de pasos. El primer tramo consistirá en un movimiento con la misma velocidad positiva en todos los flippers, la velocidad aumentará con una aceleración constante hasta llegar a 2 rad/s, luego se mantendrá estable por seis segundos. El segundo tramo será aplicar velocidades opuestas en los flippers,

una velocidad positiva en los Flippers delanteros, y negativa en los traseros. Por otro lado, el tercer tramo será invertir las velocidades usadas en el tramo anterior, tanto el segundo como el tercer tramo buscarán demostrar que la posición en Z no varía si se aplican velocidades con signos opuestos en los Flippers. Finalmente, el último tramo será aplicar velocidades inversas que en el primer tramo para devolver el sistema a su posición inicial.

En la *Figura 2.10* se pueden ver la trayectoria obtenida, mientras que en la *Figura 2.11* se muestran las variables de control a lo largo del tiempo, lo cual comprueba las propiedades descritas previamente. De esta manera se muestra que el modelo matemático del sistema de flippers se comporta de la manera esperada.

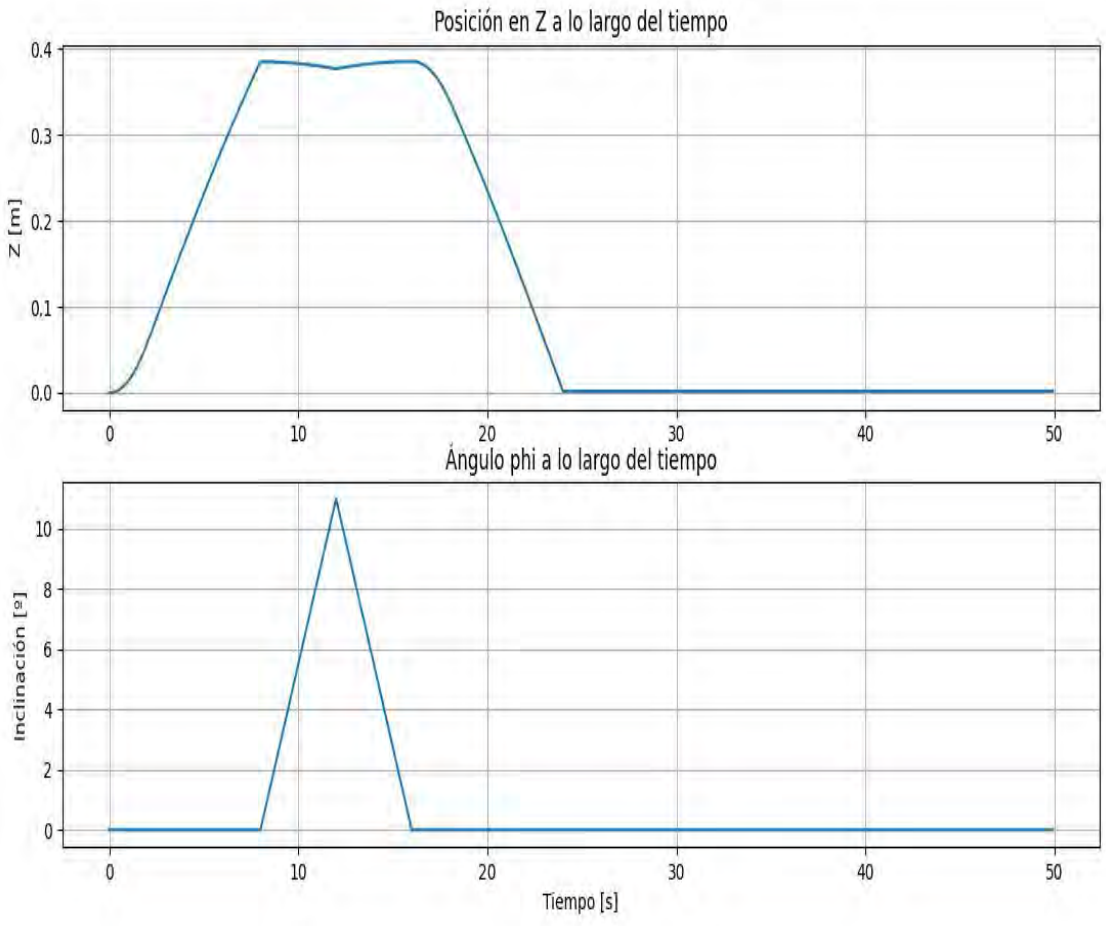

Figura 2.10. Variables de control utilizadas en el sistema de flippers. Elaboración propia

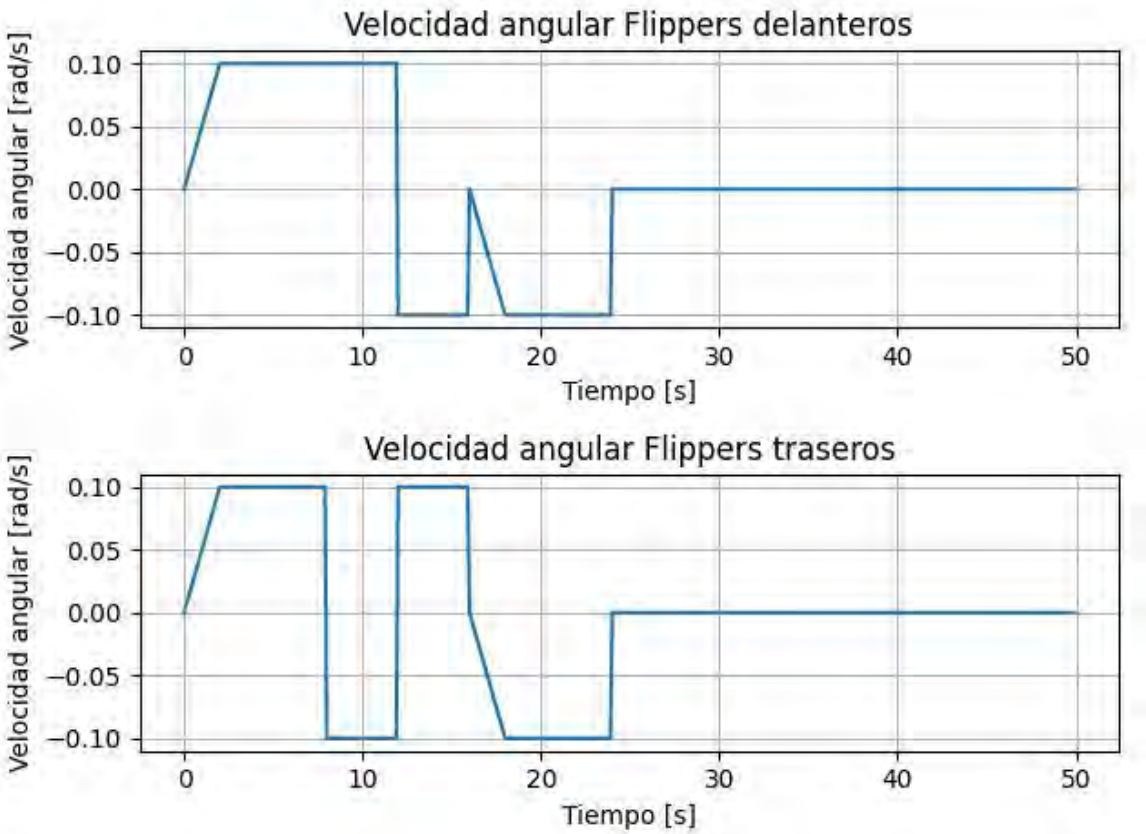

Figura 2.11. Trayectoria descrita por modelo de sistema de flippers. Elaboración propia

2.4 Conclusiones del modelado matemático

En el presente capítulo se ha desarrollado de manera satisfactoria el modelado matemático del robot articulado con orugas, el cual fue dividido en el modelamiento de la plataforma móvil y la del sistema de Flippers, ambos siendo sistemas altamente no lineales con niveles distintos de complejidad. Por un lado, se tiene la plataforma móvil, el cual tiene un modelo cinemático que se asemeja a un robot diferencial, siendo este un modelo ampliamente conocido y estudiado, mientras que por otro lado se tiene el sistema de Flippers, el cual es un sistema mucho más complejo debido a la cantidad de ecuaciones involucradas, y debido a esto se tuvo que hacer una simplificación a un esquema en dos dimensiones. Luego del modelado se comprobó la que el modelo funcionara de la manera esperada a través de aplicar distintas variables de control al sistema y evaluar el comportamiento de las variables del sistema. Con este procedimiento realizado de manera exitosa, el modelo será simulado y posteriormente usado para el diseño de un controlador no lineal apropiado en los siguientes capítulos

CAPÍTULO 3: SIMULACIÓN DEL ROBOT

3.1 Introducción

En el contexto de la robótica, la simulación de distintos modelos permite a los desarrolladores predecir el comportamiento de los robots en distintas condiciones de operación, y permite realizar pruebas de algoritmos de control que de otra manera serían prohibitivamente costosos y potencialmente peligrosos (Rouček, y otros, 2021). Debido a esto, el modelado y simulación del robot en un ambiente virtual es esencial para diseñar y validar los algoritmos de control necesarios para el funcionamiento del robot y, asimismo, evaluar el rendimiento de dichas técnicas (Al-Jarrah, Salah, & Almomani, 2019).

Este capítulo inicia con una breve introducción a los softwares de simulación en robótica, los cuales son las herramientas que permiten simular sistemas de manera virtual. Se discute su importancia y ventajas que presenta, así como menciona los distintos softwares de simulación usados en trabajos de robótica relacionados con este documento.

Luego, se documenta y describe a detalle la implementación de la simulación de un robot articulado con orugas en un ambiente virtual, así se discute su integración con el sistema de control desarrollado en siguientes capítulos. Finalmente, se valida la simulación, presentando los resultados obtenidos, y se comparan con los obtenidos por el modelo matemático. De esta manera comprobando la coherencia entre el modelo matemático y el robot simulado.

3.2 Softwares de simulación para robótica

En los últimos años la robótica y sus campos adyacentes como la informática y la electrónica han progresado de manera significativa, lo cual ha llevado al desarrollo de una amplia gama de robots usados en contexto industriales y domésticos. Estos robots, con la llegada de tecnologías de aprendizaje profundo, han ido adquiriendo la capacidad de realizar sus tareas autónomamente. Sin embargo, estas estrategias de control requieren de grandes cantidades de data para enseñar los robots distintos comportamientos, y dicha cantidad de información es poco práctica de adquirir solo utilizando campos de prueba convencionales (Audonnet, Hamilton, & Aragon-Camarasa, 2022). En un

ambiente simulado, el mundo puede ser controlado, incluyendo características que serían difíciles de incluir en un campo de pruebas físico, haciendo la recolección de data más sencilla. Además, reducen los costos, permiten probar distintos sensores y eliminan los riesgos de dañar al entorno o al operario, de esta manera también facilitando la prueba de sistemas, plataformas y prototipos complejos. En esta sección se compara los tres simuladores 3D más usados en la comunidad de robótica: V-Rep, Gazebo y Unity, analizando sus características y seleccionando el más apropiado para este proyecto (Silva, Teichrieb, & Teixeira, 2019).

3.2.1 Conceptos teóricos sobre softwares de simulación

Al momento de realizar una investigación acerca de este tema, se debe estar familiarizado con los siguientes términos.

ROS: De sus siglas en inglés: “Robot Operating System”, es un framework utilizado para desarrollar proyectos de robótica, facilitando distintas tareas comunes como transformación de sistemas de coordenadas, planificación de movimiento, comunicación e integración de distintos sensores (Silva, Teichrieb, & Teixeira, 2019).

URDF y SDF: De sus siglas en inglés: “Universal Robotic Description Format” y “Simulation Description Format”, respectivamente. Ambos tipos de archivos utilizan el formato de XML (Extensible Markup Language) para describir todos los elementos de un robot o de su ambiente. En la *Figura 3.1* se puede apreciar que un URDF representa la descripción de un solo robot, mientras que un SDF puede representar un solo robot, como toda la escena. Es importante mencionar que ROS es capaz de interpretar ambos tipos de archivos (Silva, Teichrieb, & Teixeira, 2019).

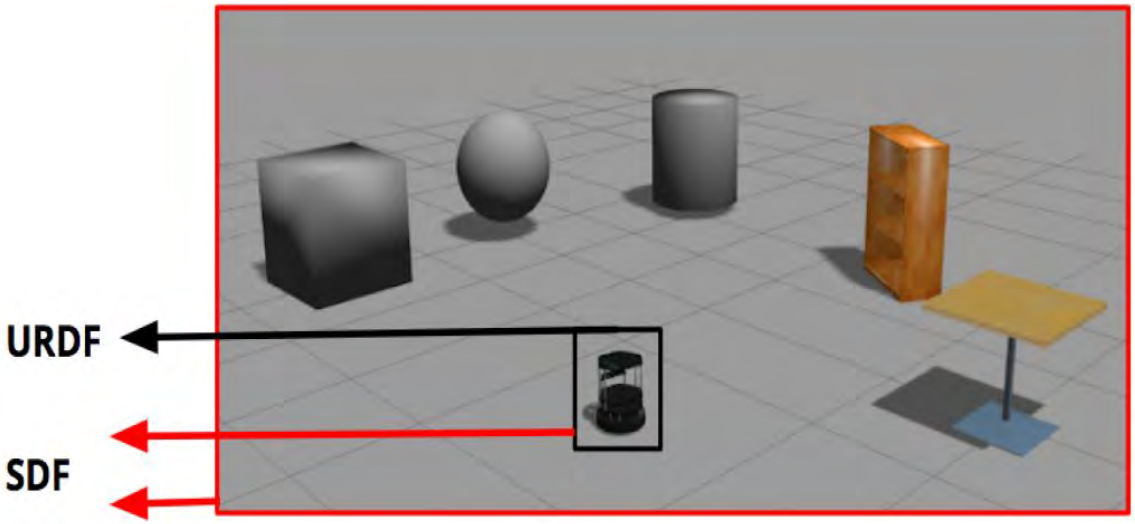

Figura 3.1 Coordenadas para el modelo cinemático de las articulaciones (Silva, Teichrieb, & Teixeira, 2019).

3.2.2 Comparativa de softwares de simulación

De acuerdo con un estudio realizado por (Silva, Teichrieb, & Teixeira, 2019), actualmente los tres simuladores 3D más prominentes en la robótica, según el número de artículos publicados mencionándolos son: V-Rep, Gazebo y Unity. Cada simulador cuenta con sus propias fortalezas y debilidades. V-Rep cuenta con una interfaz intuitiva y cuenta con una amplia variedad de características, aunque su uso requiere una licencia de pago. Por otro lado, Gazebo es una opción de código abierto que se integra fácilmente con ROS y se utiliza ampliamente en competiciones de robótica, aunque requiere mayor tiempo de desarrollo que V-Rep. Unity cuenta con una calidad de gráficas superior pero su integración con ROS es compleja y este software es menos orientada a la robótica. En la *Tabla 3.1* se resumen las principales características de estos simuladores, basándose en el estudio mencionado previamente.

Tabla 3.1. Comparativa de softwares de simulación.

Criterio	V-Rep	Gazebo	Unity
Aplicaciones	Simulación de sistemas de automatización de fábricas, monitoreo remoto, control de hardware, robótica.	Simulación de amplia gama de robots.	Juegos, realidad aumentada y virtual, aplicaciones móviles, gráficos de alta calidad, mapeo robótico.
Integración con ROS	Uso de RosInterfac API, requiere conocimiento de ROS.	Fuerte integración con ROS.	Compleja integración usando un paquete en la tienda de Unity
Lenguajes Soportados	Nodos ROS, nodos BlueZero, complementos, scripts integrados, complementos adicionales.	Complementos desarrollados en C++, API externa en C++.	Scripts en C#, extensa API agrupada por namespaces.
Uso de Archivos URDF/SDF	Soporta URDF, no soporta SDF.	Soporta URDF y SDF.	Soporta URDF, no soporta SDF.
Motores de física	ODE, Bullet, Newton, Vortex.	ODE, Bullet, Dart, Simbody.	Motor propio
Licencia	Requiere licencia para uso comercial	Código abierto	Requiere licencia para uso comercial
Facilidad de uso	Interfaz intuitiva, fácil de usar.	Requiere más desarrollo que V-Rep. Cuenta con una amplia documentación.	Menos orientado a la robótica
Cantidad de uso	10,400 resultados de trabajos científicos	30,900 resultados de trabajos científicos	25,800 resultados de trabajos científicos

Adaptado de: (Silva, Teichrieb, & Teixeira, 2019).

3.2.3 Selección de software de simulación

Luego del análisis realizado en base a la *Tabla 3.1*, se elige Gazebo como el simulador 3D utilizando en este trabajo de investigación. Esto se debe a que cuenta con la mayor variedad de librerías y herramientas, las cuales son utilizadas para la mayoría proyectos de robótica. Sumado al amplio uso del simulador por la comunidad de robótica, implica que la cantidad de soporte comunitario y documentación disponible es mayor comparado a otros softwares de simulación. Además, Gazebo se destaca por su sólida y simple integración con, lo que es crucial para la investigación y el desarrollo continuos. Otro factor a favor es que, Gazebo soporta varios motores de física y permite la importación y exportación de archivos URDF y SDF, proporcionando flexibilidad y precisión en la simulación de entornos complejos. La suma de estos factores hace que la implementación de la simulación en Gazebo sea más sencilla, siendo esta la razón principal de la elección de este simulador.

3.3 Implementación de la simulación del modelo en Gazebo

En la sección anterior se mencionó y explicó la importancia del uso de simulación en el campo de la robótica, así como se seleccionó Gazebo como el software de simulación utilizado para el desarrollo y validación del modelo y de los algoritmos de control del robot. En esta sección se describirá a detalle el proceso de implementación del modelo del robot articulado con orugas en Gazebo.

Para llevar a cabo la simulación, se utilizó un ordenador con sistema operativo Ubuntu 20.04, ROS Noetic y Gazebo 11. La instalación se realizó siguiendo la documentación oficial, asegurando la correcta integración entre ROS y Gazebo. Las características de la computadora utilizada se muestran en la *Tabla 3.2.*

Tabla 3.2. Características de la computadora utilizada para la simulación. Elaboración propia

Característica	**Valor**
Almacenamiento	SSD 1TB
RAM	32 GB
CPU	I9-139000KF
Cantidad de núcleos	32
GPU	RTX 1080 16GB VRAM

Se utilizará como base el modelo de ejemplo de un robot con orugas brindado por Gazebo, llamado "*tracked_vehicle_simple*", y se explicará a detalle las modificaciones hechas para que se asemeje al robot objeto de estudio de esta investigación (Gazebo, 2024). En la Figura 3.2 se puede ver el modelo base sin modificaciones, este es un archivo SDF que cuenta tanto con la descripción del robot, como con el ambiente de pruebas. Además, en la *Figura 3.3.* y *Tabla 3.3* Se muestran los eslabones o *links* del robot, así como sus articulaciones o *joints*.

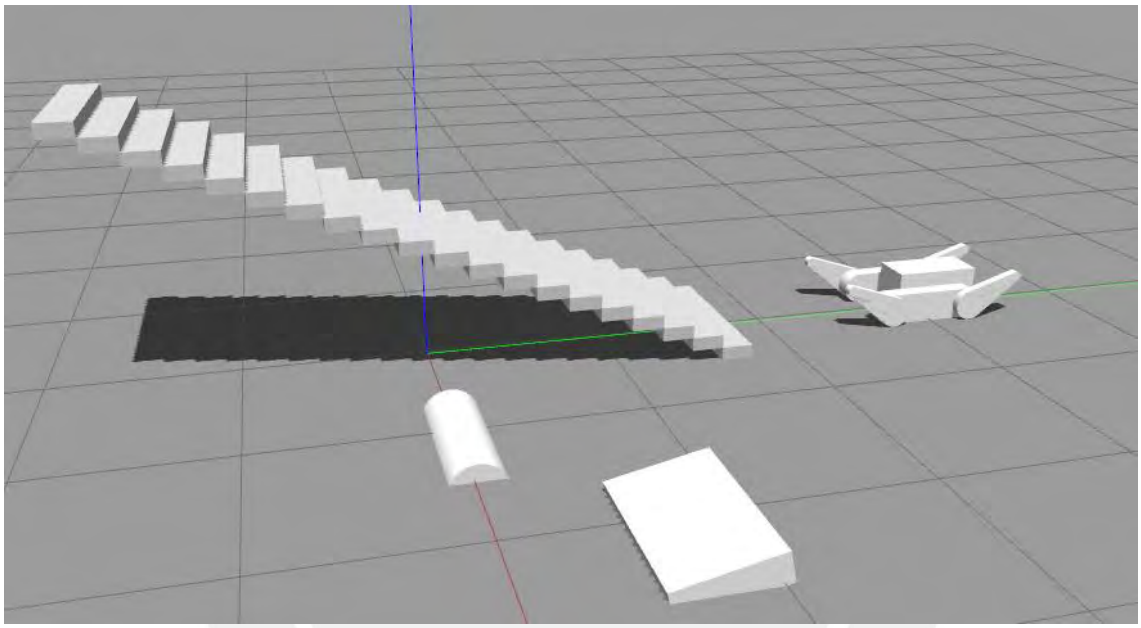

Figura 3.2 Modelo de ejemplo de un robot con orugas. Adaptado de: (Gazebo, 2024).

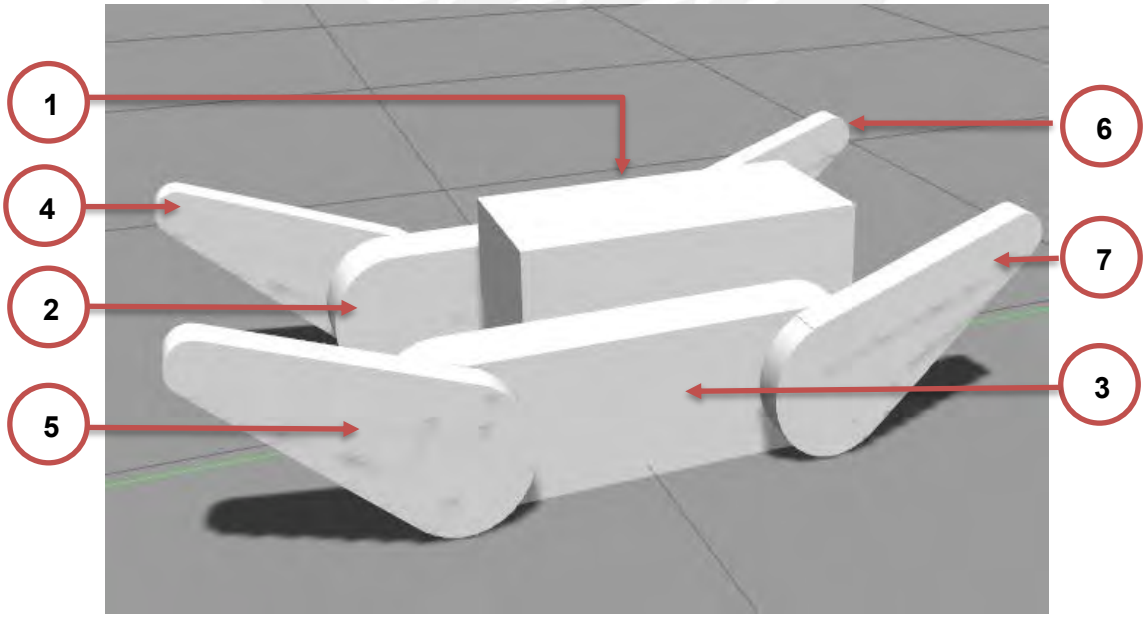

Figura 3.3. Articulaciones y eslabones de robot con orugas. Adaptado de: (Gazebo, 2024).

Tabla 3.3. Leyenda de "Articulaciones y eslabones de robot con orugas". Elaboración propia

Número	Eslabón	Articulación	Tipo de articulación
1	Eslabón principal (base)	-	-
2	Oruga derecha	Base a oruga derecha	Giro continuo
3	Oruga izquierda	Base a oruga izquierda	Giro continuo
4	Flipper derecho delantero	Base a Flipper derecho delantero	Revolución (giro limitado)
5	Flipper izquierdo delantero	Base a Flipper izquierdo delantero	Revolución (giro limitado)
6	Flipper derecho trasero	Base a Flipper derecho trasero	Revolución (giro limitado)
7	Flipper izquierdo trasero	Base a Flipper izquierdo trasero	Revolución (giro limitado)

3.3.1 Modelado del robot y entorno de simulación Gazebo

El modelo del robot presente en el ejemplo de Gazebo si bien se ve estéticamente bastante parecido al de un robot articulado con orugas, como el presente al de la *Figura 2.1*, existen diferentes cambios que se deben realizar para que se comporte en su totalidad como uno, en la *Tabla 3.4* se muestran y explican los cambios realizados en el SDF para lograr esta finalidad.

Tabla 3.4. Cambios del modelo del robot realizados en el SDF. Elaboración propia

Cambio	Explicación
Mover la base de tal manera que el robot sea simétrico en el eje Y	Evita que el robot se desbalancee al momento de subir escaleras
Cada tipo de eslabón tiene un color distinto (base de color blanco, orugas azules y flippers naranjas)	Permite reconocer los elementos de mejor manera
Se cambia el ángulo máximo y mínimo que puede tener la articulación de los flippers.	Anteriormente el mínimo y el máximo eran iguales, por lo cual no se podían mover los flippers. Se cambia ángulo máximo =90°, ángulo mínimo = -90°
Se añade fricción y amortiguamiento a las articulaciones de los flippers	Permite al Flipper mantener su posición mientras no se envía un comando de movimiento

En cuanto al entorno de pruebas, este se mantiene igual excepto por el cambio de color del terreno, de blanco a verde. Este es solamente un cambio estético que le da mayor contraste al robot, haciéndolo más fácil de visualizar. En la *Figura 3.4* se muestra el SDF luego de las modificaciones del robot y entorno de pruebas.

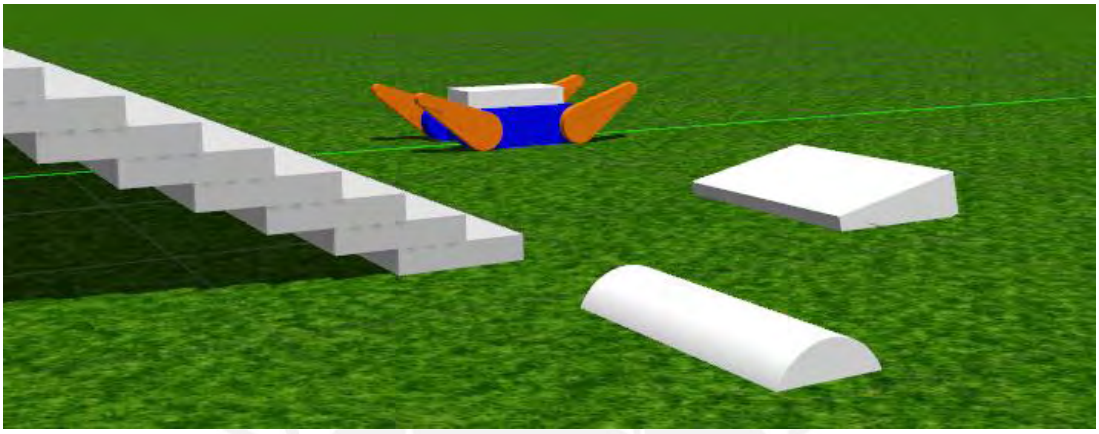

Figura 3.4. Campo de pruebas y modelo de robot. Elaboración propia

3.3.2 Integración del modelo con ROS

La integración inicial en ROS de este proyecto es relativamente sencilla. El objetivo principal de esta integración inicial es lanzar el archivo SDF en Gazebo utilizando ROS mediante un archivo launch principal, el cual es un archivo en formato XML el cual permite la ejecución simultanea de múltiples nodos en ROS, así como la configuración de parámetros necesaria para su funcionamiento. Los pasos necesarios para este proceso se pueden ver en la Figura 3.5, mientras que los parámetros necesarios para configurar el archivo launch usado para ejecuta el SDF se muestran en la *Tabla 3.5*.

Tabla 3.5. Parámetros usados para lanzar archivo SDF en Gazebo. Elaboración propia

Parámetro	Valor	Explicación
World name	$(find my_robot)/world/ tracked_robot_vehicle_simple. world. world	Ubicación del SDF
File	$(find gazebo_ros)/launch/ emplty_world. launch	Ubicación del programa que permite lanzar SDFs en Gazebo
Use_sim_time	True	Habilitar tiempo de simulación propio
gui	True	Habilitar interfaz gráfica

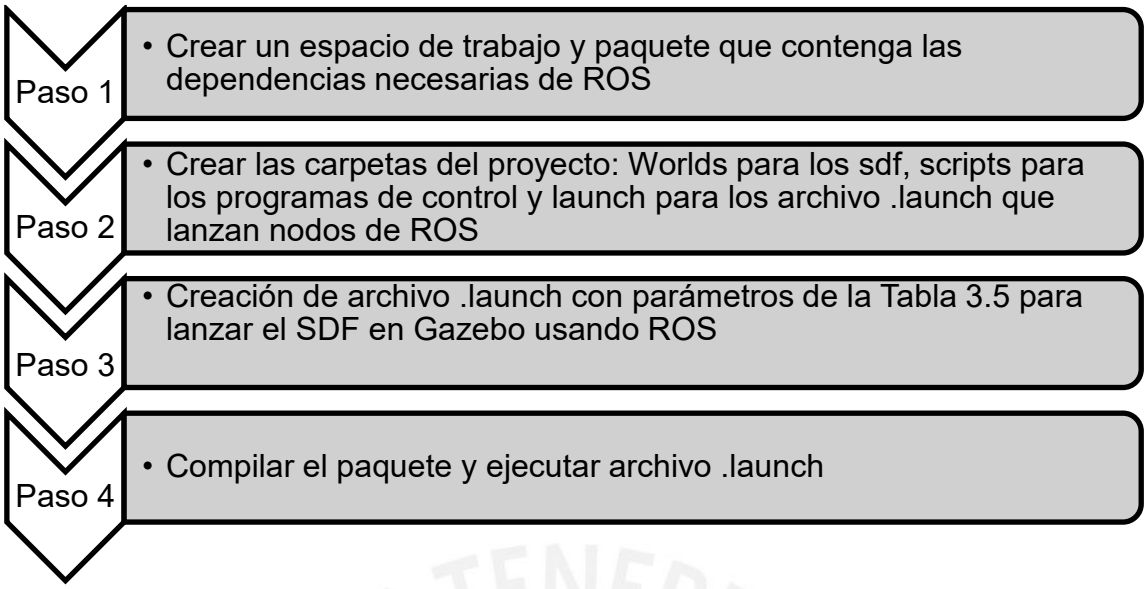

Figura 3.5. Pasos utilizados para lanzar un SDF en Gazebo utilizando ROS. Elaboración propia

3.3.3 Implementación de realimentación de estado

Una vez que se tiene el modelo del robot, es necesario que este se pueda mover y además este pueda realimentar su posición, orientación y velocidades para que estas sean utilizadas en los algoritmos de control que se desarrollarán posteriormente. Con este motivo, se utilizarán distintos plugins de Gazebo para poder mover las articulaciones del robot o para obtener la información sobre el estado de cada articulación del robot. En la *Tabla 3.6* se resume la manera en la que se usaron dichas librerías.

Tabla 3.6. Parámetros usados para lanzar archivo SDF en Gazebo. Elaboración propia

Plugin	**Uso**	**Parámetros**
libSimpleTracked VehiclePlugin.so	Movilizar la plataforma móvil a partir de un setpoint de velocidad.	Nombre de todos los links del robot, fricción de las orugas con el piso
Gazebo_ros_joint _state_publisher.so	Publicar en ROS la orientación de los flippers.	Nombre de los joints correspondiente a cada flipper.
Gazebo_ros_joint _pose_trayectory.so	Controlar el ángulo de cada flipper	Frecuencia de muestreo
Libgazebo_ros_tracked _vehicle_interface.so	Publica en ROS la odometría de la plataforma móvil.	Tópico del setpoint, tópico de odometría, link base, separación entre orugas

3.3.4 Pruebas iniciales del modelo en Gazebo

En esta sección se mostrará al robot navegando alrededor del ambiente de simulación, utilizando tanto la plataforma móvil como los flippers. Estas pruebas tienen como objetivo verificar el funcionamiento de robot simulado, el cual será controlado por lazo abierto. Se utiliza el nodo rqt_robot_steering, añadiéndolo al archivo launch principal, para poder establecer los setpoints de velocidad angular y lineal del robot mediante una interfaz gráfica. Este nodo solo será usado para las pruebas de este capítulo, ya que en capítulos posteriores estos setpoints se variarán de manera automática. En la *Tabla 3.7* se mencionan las pruebas realizadas junto con su figura relacionada, la cual consta de una secuencia de imágenes de la prueba. Cabe resaltar que las pruebas realizadas fueron utilizando un control manual, por lo que no existe una trayectoria matemáticamente definida.

Tabla 3.7. Relación de pruebas realizadas. Elaboración propia

Prueba realizada	**Descripción**	**Figura**
Desplazamiento en terreno plano	Se desplaza el robot, alterando los setpoints de velocidad lineal entre -1 m/s a 1 m/s, y la velocidad angular -3 rad/s a 3 rad/s	Figura 3.6
Subida de escaleras	El robot avanza por las escaleras con una velocidad lineal de 1 m/s y una velocidad angular de 0 rad/s	Figura 3.7
Bajada de escaleras	El robot retrocede por las escaleras con una velocidad lineal de -1 m/s y una velocidad angular de 0 rad/s	Figura 3.8
Atravesar obstáculo 1	El robot atraviesa un obstáculo similar a un rompemuelles con una velocidad lineal de 0.5 m/s	Figura 3.9
Atravesar obstáculo 2	El robot atraviesa un obstáculo similar a una rampa con una velocidad lineal de 0.5 m/s	Figura 3.10
Utilización plataforma móvil y flippers simultáneamente.	El robot coloca inicialmente sus flippers con una inclinación de 90°. Luego, varía su inclinación ±15° mientras avanza con una velocidad lineal de 0.5 m/s.	Figura 3.11

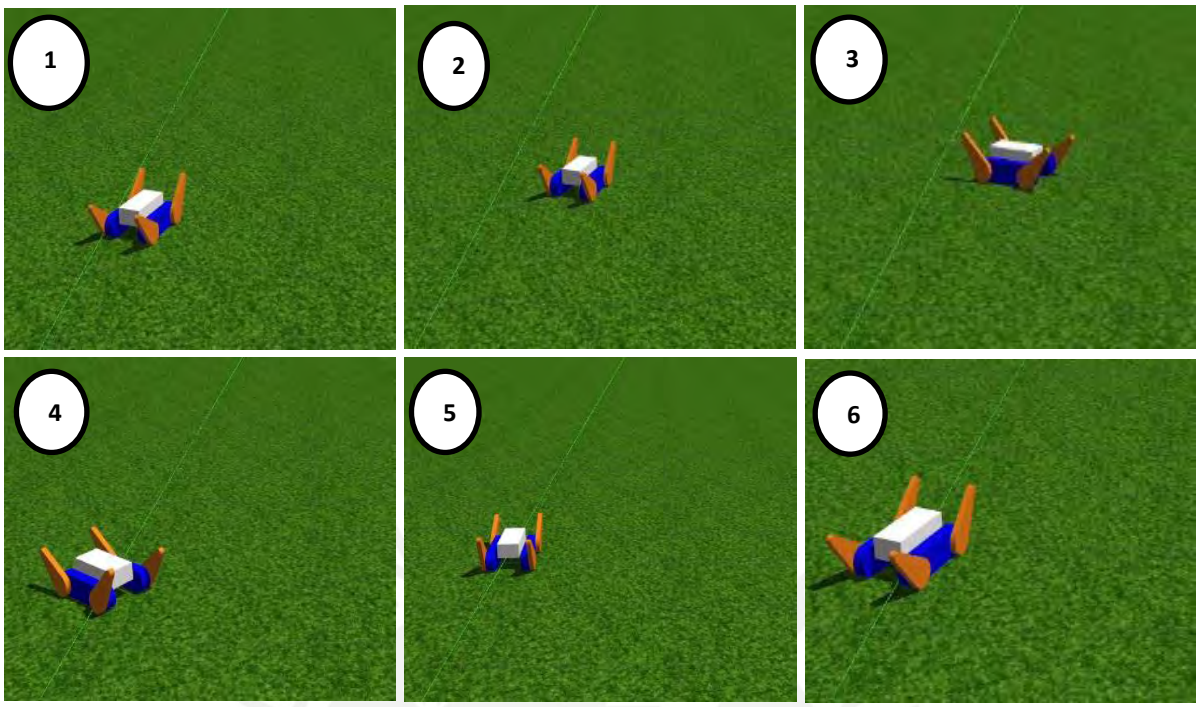

Figura 3.6. Prueba de desplazamiento en terreno plano. Elaboración propia

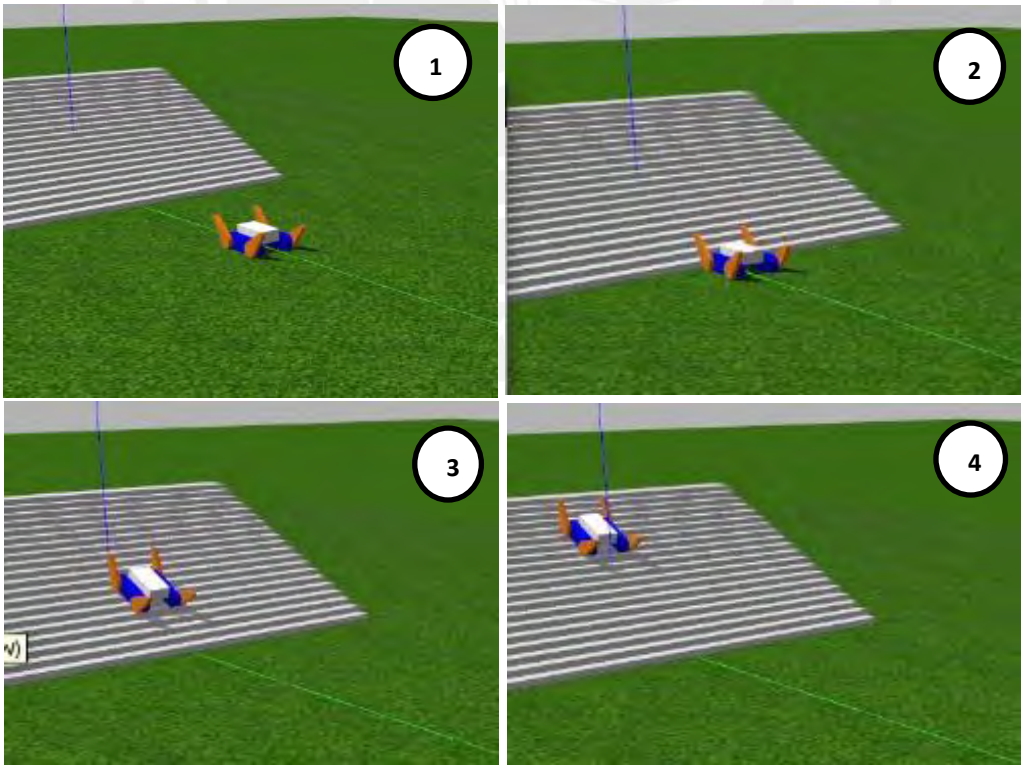

Figura 3.7. Prueba de subida de escaleras. Elaboración propia

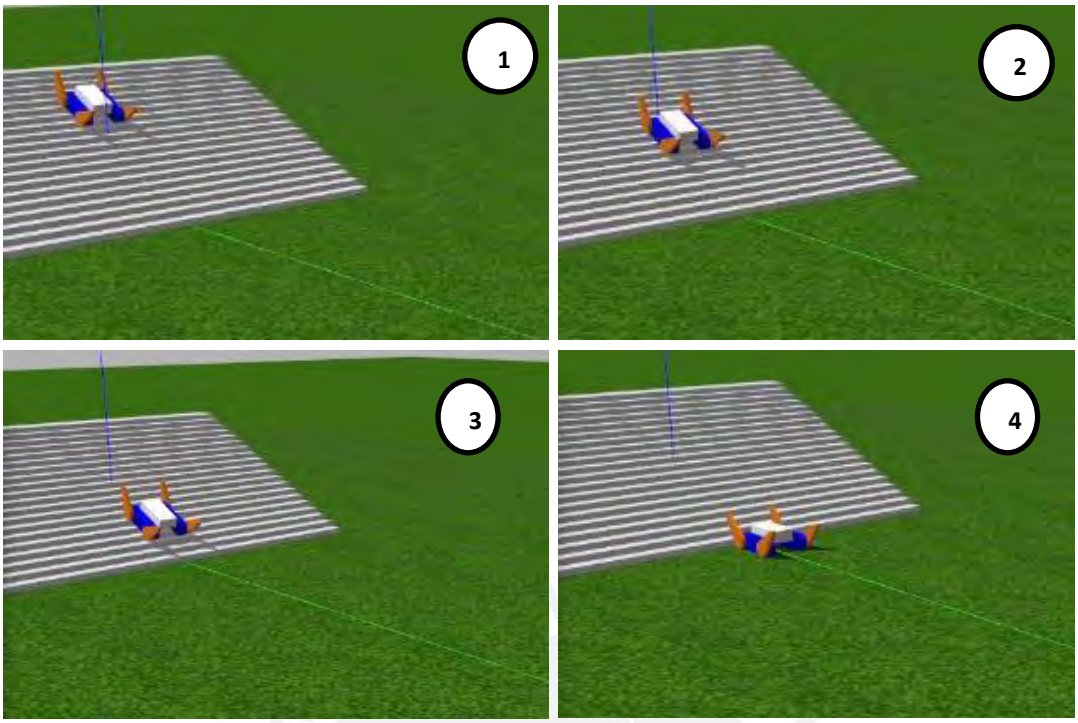

Figura 3.8. Prueba de bajada de escaleras. Elaboración propia

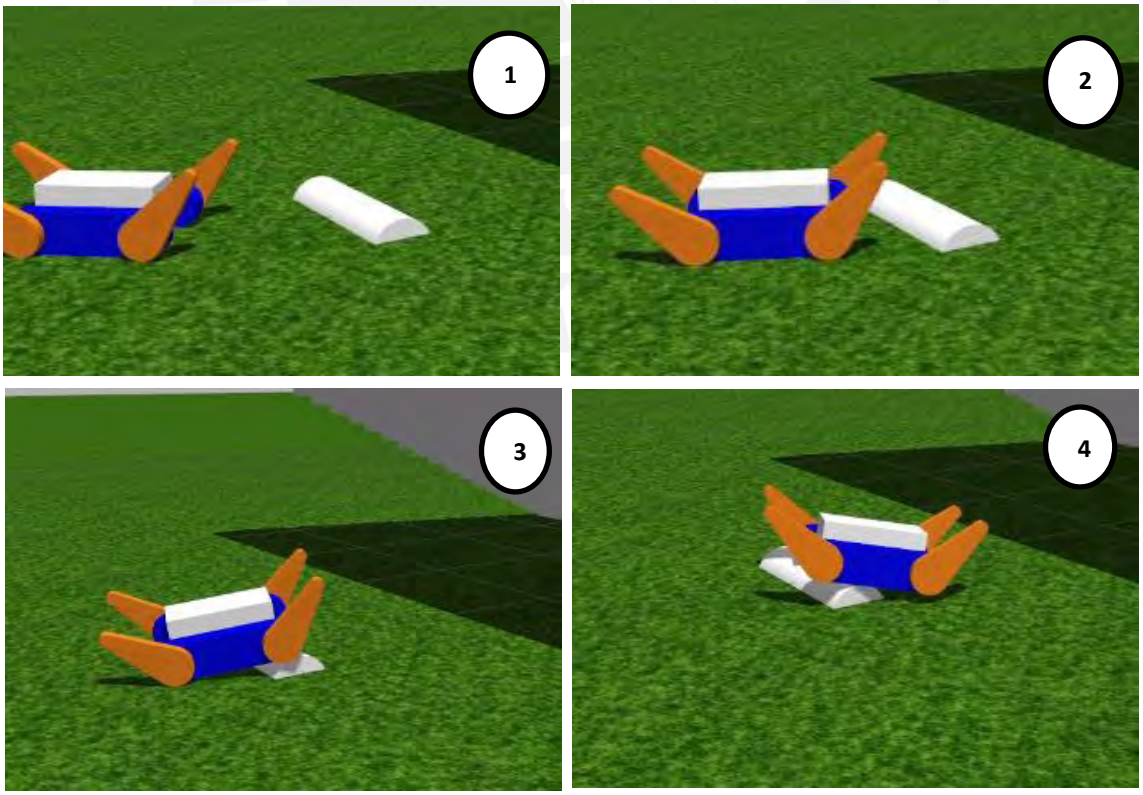

Figura 3.9. Prueba 1 de atravesar obstáculos. Elaboración propia.

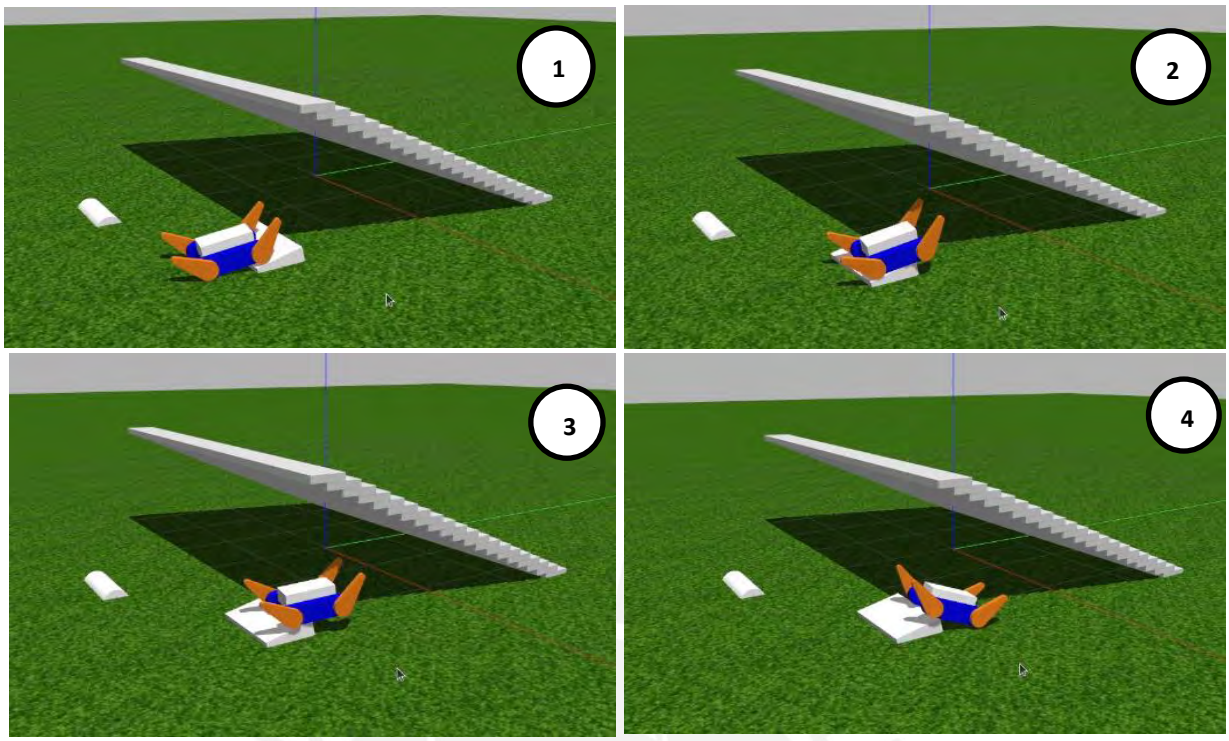

Figura 3.10. Prueba 2 de atravesar obstáculos. Elaboración propia.

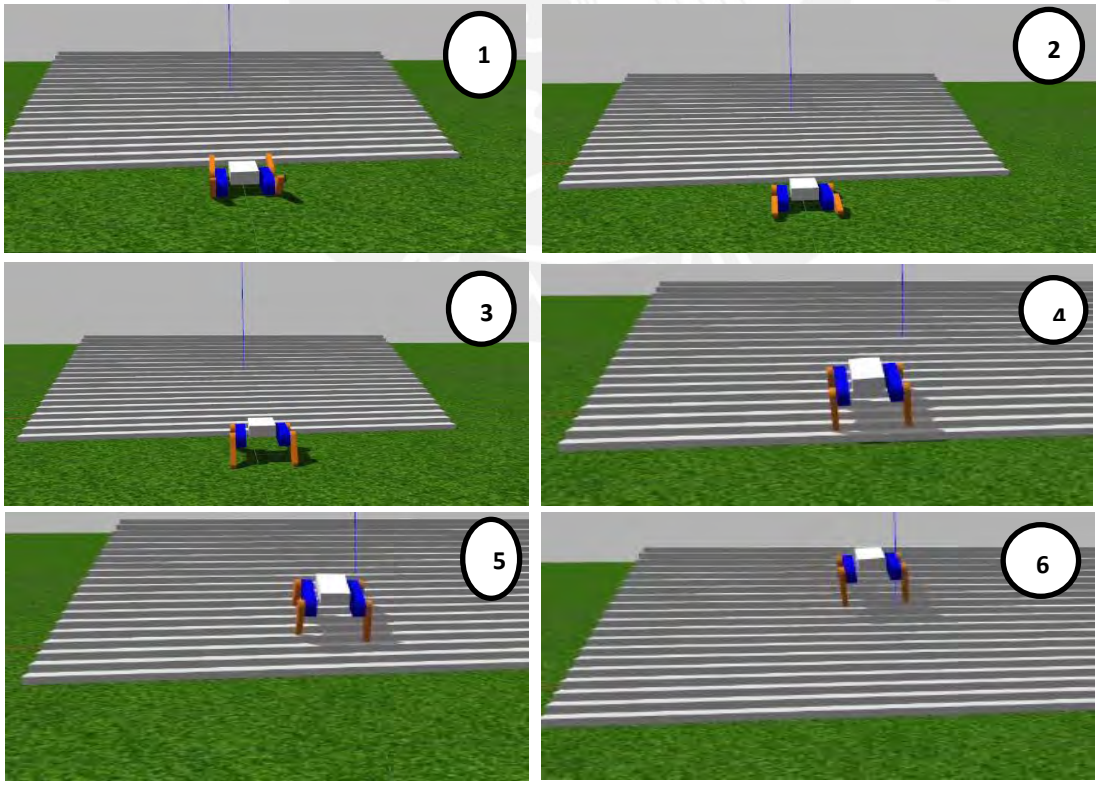

Figura 3.11. Utilización plataforma móvil y flippers simultáneamente. Elaboración propia.

3.4 Validación de la simulación

Luego de obtener el modelo matemático del robot, implementar en Gazebo su simulación de este, y probar su funcionalidad en un ambiente de simulación, en esta sección se validará la simulación implementada contrastándola con el modelo matemático.

3.4.1 Pruebas de validación

Las pruebas realizadas son las siguientes. La primera prueba se centra en la validación de la plataforma móvil, consiste en mover la plataforma móvil del robot con una velocidad lineal constante de 0.5 m/s y una velocidad angular constante de 0.2 rad/s, describiendo una circunferencia. Durante este movimiento, se obtienen los datos de posición del robot, así como las velocidades en la oruga derecha e izquierda desde tópicos de ROS.

El error porcentual de posición se calcula como la diferencia de distancias al centro teórico de los puntos teóricos y los obtenidos en la simulación, dividido entre el radio teórico. En la *Figura 3.12*, utilizando Python, se compara y grafica la posición del robot en la simulación, con la posición del robot obtenida con el modelo matemático de la plataforma móvil a partir de las velocidades en las orugas. En la Figura 3.13 se muestra la gráfica del error y se obtiene un error promedio del 1.48%, validando de esta manera la simulación de la plataforma móvil.

Figura 3.12. Posición del robot moviéndose mediante su plataforma móvil. Elaboración propia.

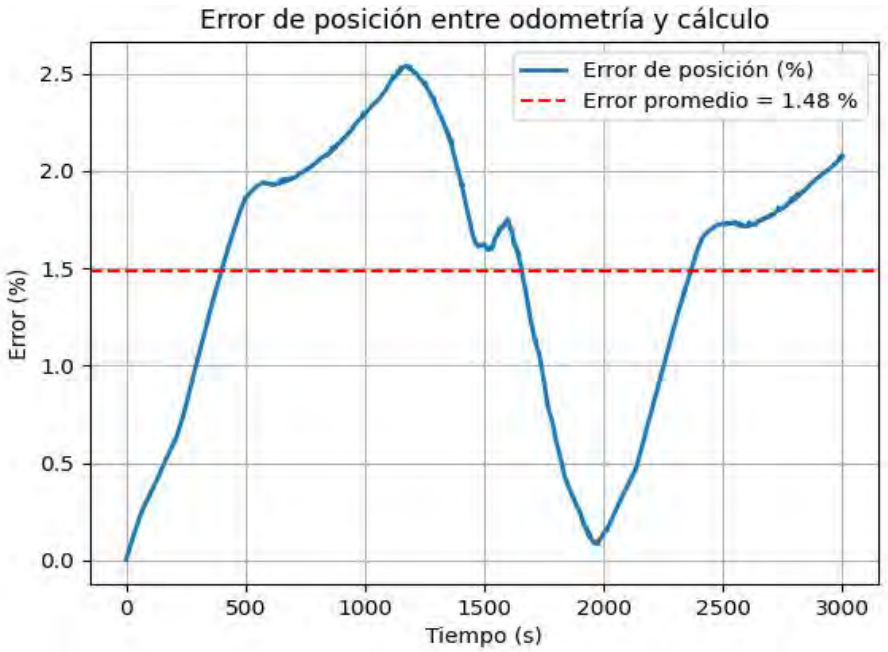

Figura 3.13. Error de posición del robot moviéndose mediante su plataforma móvil. Elaboración propia.

La segunda prueba se enfoca en la validación de las articulaciones del robot. En esta prueba, las articulaciones delanteras del robot se mueven desde un ángulo de 0° con respecto al suelo a -50°, mientras que las articulaciones traseras del robot se mueven de 0° a -70°. Durante este movimiento, se obtienen los datos de posición del robot, así como las inclinaciones de todas las articulaciones desde tópicos de ROS. En la Figura 3.14, utilizando Python, se compara y grafica la trayectoria del del robot en la simulación, mostrada con puntos naranjas y flechas verdes, mientras que la trayectoria obtenida utilizando el modelo matemático, se muestra en flechas rojas y puntos azules con la posición del robot obtenida con el modelo matemático de las articulaciones a partir de sus ángulos. En la *Figura 3.15* se muestra la gráfica del error y se obtiene un error promedio del 3.14%, validando de esta manera la simulación de las articulaciones.

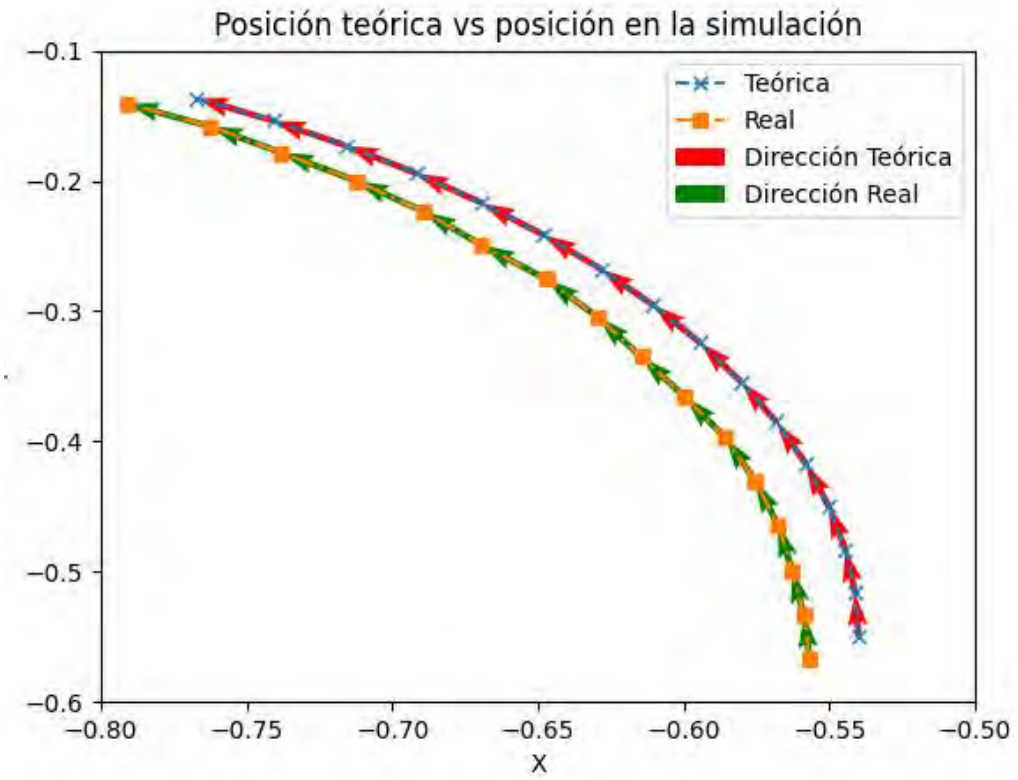

Figura 3.14. Posición del robot moviéndose mediante sus articulaciones. Elaboración propia.

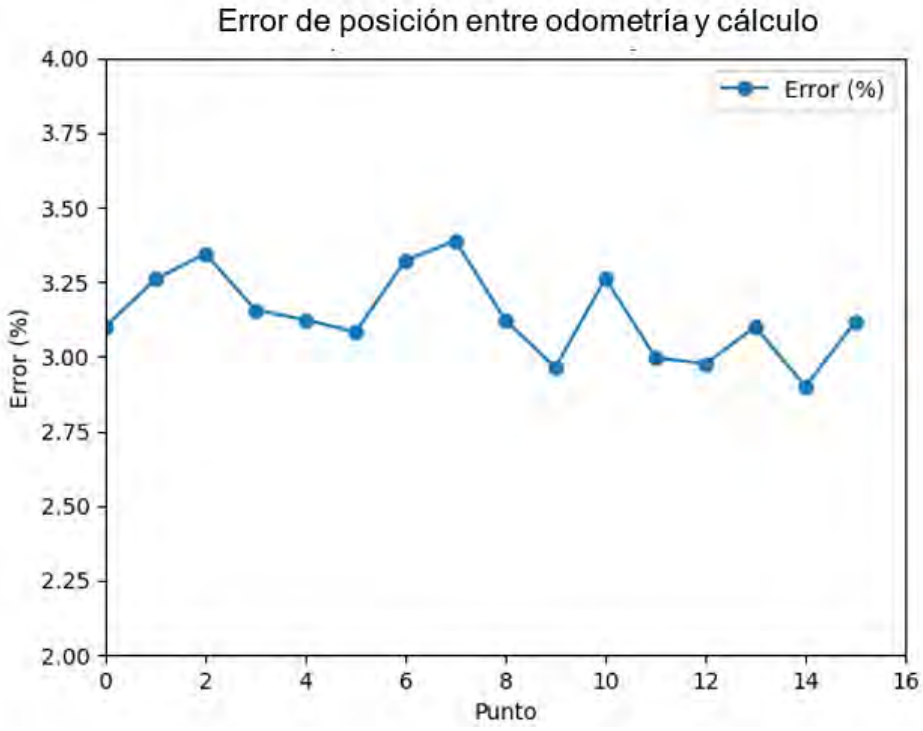

Figura 3.15. Posición del robot moviéndose mediante sus articulaciones. Elaboración propia.

3.4.2 Conclusiones de la validación

En este capítulo, se modela matemáticamente y simula en Gazebo un robot articulado con orugas, utilizando ROS para la integración del modelo con Python y control del sistema. La implementación de la simulación brinda la oportunidad de evaluar el desempeño del robot en diferentes condiciones operativas, asegurando que el modelo refleje con precisión las características dinámicas y cinemáticas del robot.

Las pruebas de validación han mostrado resultados positivos, con un error promedio del 3.16% en la simulación de la plataforma móvil y del 3.14% en las articulaciones. Estos resultados confirman la precisión y confiabilidad del modelo implementado. Estos resultados validan la efectividad del modelo matemático y su implementación en Gazebo. Además, también demuestran la que esta simulación puede replicar el funcionamiento del robot. En resumen, el desarrollo y la validación de este modelo nos dan una base sólida para poder implementar los algoritmos de control en los capítulos posteriores de este documento.

4 CAPÍTULO 4: DISEÑO DEL CONTROLADOR

4.1 Introducción

En este capítulo, se abordará la implementación por separado de controladores avanzados para la plataforma móvil del robot articulado con orugas y para las articulaciones. La correcta implementación de estos controladores es crucial para garantizar la precisión y estabilidad del robot en diversas condiciones de operación, especialmente en entornos industriales con terrenos irregulares.

La plataforma móvil, siendo una parte fundamental del robot, requiere un sistema de control robusto que considere la no linealidad del sistema y las posibles perturbaciones externas. Para ello, se han seleccionado dos controladores: el controlador Backstepping y el controlador Feedback Linearization. Cada uno de estos controladores ofrece ventajas únicas en términos de estabilidad, robustez y facilidad de implementación.

El sistema de flippers, comparado con el modelo de la plataforma móvil, presenta un conjunto de diferencias importantes al momento de realizar el diseño del controlador. En primer lugar, es un sistema que cuenta con una mayor no linealidad. En segundo lugar, las trayectorias a seguir de este sistema son mucho menos complejas que el de una plataforma móvil. Finalmente, a diferencia de la plataforma móvil, su control no se encuentra documentado de manera tan extensa en la literatura. Debido a la complejidad de las ecuaciones, se ha seleccionado 2 controladores: el controlador por Gain Scheduling y el controlador de lógica difusa.

El objetivo es de seleccionar e implementar el controlador adecuado para cada aplicación. Primero, se detallará la teoría detrás de cada controlador. Luego, se realizará una implementación detallada de cada uno de estos controladores, utilizando el modelo cinemático desarrollado en capítulos anteriores. Después, se llevará a cabo una comparativa exhaustiva entre los tres controladores, evaluando su desempeño en términos de precisión de seguimiento de trayectorias, tiempo de respuesta y métricas de error. Una vez seleccionado el controlador con las mejores características, se detallará el proceso de

implementación en Gazebo. Finalmente, se presentarán las conclusiones del capítulo

A través de esta comparación, se pretende identificar el controlador más adecuado para la plataforma móvil del robot. Los resultados obtenidos proporcionarán una base sólida para futuras implementaciones y optimizaciones en el control de robots articulados con orugas, contribuyendo así al desarrollo de soluciones más seguras y eficientes en la industria.

4.2 Control no Lineal en sistemas de robótica

El control no lineal es una rama del control automático que se ocupa de sistemas cuyos comportamientos no pueden ser descritos adecuadamente por modelos lineales. En el caso de los robots articulados con orugas, las no linealidades derivan de las complejas interacciones entre las orugas, los flippers y el terreno. Este capítulo explora cuatro técnicas de control no lineal ampliamente utilizadas: Backstepping, Feedback Linearization, Gain Scheduling y Lógica difusa. Cada una de estas técnicas tiene sus propias ventajas y desafíos, y se utilizan en diferentes contextos para mejorar la estabilidad y el rendimiento del sistema.

4.2.1 Backstepping

Backstepping es una técnica control recursiva basada en Lyapunov, la cual se utiliza descomponiendo un sistema no lineal en subsistemas de menor orden en donde sus entradas de control son las variables del sistema, las cuales se van determinando de manera recursiva hasta llegar al último subsistema en el que aparece la entrada de control del sistema real. Este método es particularmente útil para sistemas con estructuras jerárquicas o en cascada (Aranda-Cetraro, Pérez-Zúñiga, Rivas-Pérez, & Sotomayor-Moriano, 2023).

El método de Backstepping es aplicable a sistemas de realimentación estricta (strict feedback systems) también conocidos como sistemas de bloque triangular inferior. Estos sistemas son los representados de la siguiente forma:

$$\dot{x_1} = f_1(x_1) + g_1(x_1)x_2$$

$$\dot{x_2} = f_2(x_1, x_2) + g_1(x_1, x_2)x_3$$

$$\vdots$$

$$\dot{x_{n-1}} = f_{n-1}(x_1, x_2, \dots, x_{n-1}) + g_{n-1}(x_1, x_2 \dots, x_{n-1})x_n$$

$$\dot{x_n} = f_n(x_1, x_2, \dots, x_n) + g_n(x_1, x_2 \dots, x_n)u \tag{4.1}$$

Donde las variables de estado $x_1, x_2, \dots, x_n \in R$, la entrada de control $u \in R$, f_i y g_i donde i toma valores de 1,2, ..., n, son funciones conocidas de dimensiones adecuadas.

4.2.2 Feedback Linearization

El método de Feedback Linearization o linealización por realimentación diseña un sistema de control basado en la transformación lineal del sistema no lineal a uno lineal a través de una transformación exacta, determinada a partir de las entradas y salidas del sistema. El sistema lineal se define en términos de nuevas variables de estado y variables de control. Este método de linealización por realimentación se aplica a sistemas no lineales con representación afín, es decir, en los que la entrada de control aparece en forma explícita e independiente (Wu & Blaabjerg, 2021).

$$\dot{x} = f(x) + g(x)u \tag{4.2}$$
$$y = h(x)$$

Donde el vector de estado $x \in R^n$, la entrada de control $u \in R^m$, y la entrada $y \in R^p$. Las funciones vectoriales $\boldsymbol{f(x)}: \boldsymbol{R^n} \rightarrow \boldsymbol{R^n}, \boldsymbol{g(x)}: \boldsymbol{R^n} \rightarrow \boldsymbol{R^m}, \boldsymbol{h(x)}: \boldsymbol{R^n} \rightarrow \boldsymbol{R^p}$ son funciones derivables hasta donde sea requerido.

El proceso presentado a continuación es el necesario para linealizar un sistema SISO, para sistemas MIMO, se debe extender el método. Para un sistema no lineal afín como el presente en la ecuación 4.2, donde x es el vector de estados, y u es la entrada de control. El objetivo es encontrar una transformación de estado $z = \phi(x)$ tal que:

$$\dot{z} = Az + Bu \tag{4.3}$$

donde A y B son matrices constantes. El controlador resultante es:

$$u = \phi^{-1}(-A\phi(x) + v) \tag{4.4}$$

donde v es la nueva entrada de control para el sistema linealizado.

4.2.3 Control por Gain Scheduling

El control por gain scheduling, también conocido como control por ganancia programado es un método de control no lineal que se basa en diseñar múltiples controladores lineales los cuales corresponden a un conjunto de linealizaciones del sistema en distintos puntos de operación. La ecuación de control se obtiene interpolando los diferentes controladores, obteniendo un controlador lineal en los cuales sus ganancias o parámetros son ajustados en función de alguna variable del sistema. De esta manera, el diseñador del controlado puede garantizar que, en cada punto de operación, el sistema de control tiene las propiedades de estabilidad y rendimiento necesarias.

También es necesario recalcar que la estabilidad y propiedades del sistema solo se pueden garantizar de manera local y que la ley para determinar los parámetros de control dependiendo de las variables del sistema debe ser inferida a partir de la simulación, por lo cual, en la mayoría de los casos, esta ley debe ser encontrado de manera heurística. (Chen W.-K. , 2005)

4.2.4 Control por lógica difusa

El diseño y modelado por lógica difusa se basa en utilizar reglas sí-entonces para modelar los aspectos cualitativos del conocimiento humano sin necesidad de un análisis cuantitativo complejo, lo que lo hace óptimo para sistemas vagamente definidos o con gran incertidumbre (Fenco, Pérez-Zuñiga, Quiroz, & Cuellar, 2021). Esta técnica, explorada sistemáticamente inicialmente por Takagi y Sugeno, ha encontrado distintas aplicaciones en control. La técnica de control difuso se basa en reglas de control difusas, o enunciados condicionales, que son expresiones de la forma: "SI A, ENTONCES B", donde A y B son categorías pertenecientes a una función de membresía.

El proceso de diseño de un controlador difuso se puede dividir en los siguientes pasos (Jyh & Jang, 1993):

> **Fuzzificación**: Se definen las funciones de membresía para las variables de entrada y salida, dividiendo los valores numéricos en categorías lingüísticas. Por ejemplo, en un sistema de control de nivel de agua, la altura del agua se puede dividir en categorías como "alto", "medio" y "bajo".

Elaboración de las Reglas Difusas: Se construyen reglas que relacionan las variables de entrada con la salida del sistema. Un ejemplo sería: "SI la altura del agua es baja, ENTONCES abrir la válvula al máximo".

Inferencia Difusa: Se evalúan las reglas y se determina el nivel de activación según las entradas actuales, generando un conjunto difuso como resultado.

Defuzzificación: El conjunto difuso obtenido se convierte en un valor numérico preciso que sirve como la salida final del sistema de control, utilizando métodos como el centroide o promedio ponderado.

Estas cuatro técnicas de control no lineal, Backstepping, linealización por realimentación, gain scheduling y control difuso, proporcionan diferentes enfoques para manejar las complejidades y no linealidades de los robots articulados con orugas. Cada técnica ofrece un conjunto único de herramientas y metodologías que, cuando se aplican adecuadamente que pueden mejorar significativamente la estabilidad y el rendimiento del sistema. En las siguientes secciones, se aplicarán estas técnicas para diseñar controladores específicos para la plataforma móvil y las articulaciones. Luego se comparan los controladores entre sí, para definir la mejor alternativa de control en cada caso.

4.2.5 Comparativa de características de técnicas de control

En la *Tabla 4.1* se resumen las características de las técnicas de control descritas previamente

Tabla 4.1. Características de las técnicas de control no lineal. Elaboración propia

Característica	**Backstepping**	**Feedback Linearization**	**Gain Scheduling**	**Control Difuso**
Principio Básico	Descomposición en subsistemas	Transformación a sistema lineal	Ajuste de ganancias basado en el punto de operación	Reglas difusas para modelar el comportamiento del sistema
Metodología	Recursiva	Transformación de entrada-salida	Cambios de parámetros de control en tiempo real	Fuzzificación, inferencia, defuzzificación
Ley de control	$u = -k1x1, u2 = -k2(x2 + k1x1)$	$u = \phi^{-1}(-A\phi(x) + v)$	$u = K(x)$ basado en tablas de ganancias predefinidas	Basada en las reglas: SI A, ENTONCES B
Ventajas	Estructurada, garantiza estabilidad	Utiliza técnicas de control lineal	Simplicidad y flexibilidad	Manejo de incertidumbre en sistemas
Desventajas	Complejidad en sistemas de alta dimensión	Requiere modelo preciso	Requiere conocer todos los modos de operación	Dependencia de las reglas definidas
Aplicabilidad	Sistemas jerárquicos o en cascada	Sistemas donde las no linealidades pueden ser canceladas	Sistemas con diferentes modos de operación	Sistemas con alta incertidumbre o difícil modelado

4.3 Diseño del controlador para plataforma móvil.

En la presente sección se presenta el diseño de dos (02) controladores no lineales para el seguimiento de trayectorias de una plataforma móvil del robot articulado con orugas. Cada uno de los controladores aplicarán una técnica distinta de entre las explicadas en la sección 4.2. Es decir, se implementarán y analizarán dos técnicas de control no lineal: Backstepping y Feedback Linearization.

Cada técnica de control será abordada de manera detallada, basándose en una revisión de la literatura existente. Esto incluye la descripción de la metodología seguida por la derivación de las ecuaciones de control específicas para el modelo del robot.

4.3.1 Controlador de plataforma móvil basado en Backstepping

El planteamiento de este controlador está basado en el diseño presente en (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016). Este utiliza el modelo presente en la ecuación 2.1. La pose de la plataforma móvil se expresa por el vector $p = (x, y, \theta)^T$, el robot cuenta con una velocidad $q = (v, \omega)^T$, con la cual sigue una trayectoria de referencia vector $pr = (xr, yr, \theta r)^T$ y cuenta con una velocidad de referencia $qr = (vr, \omega r)^T$. Estos vectores se visualizan de mejor manera en la *Figura 4.1.*

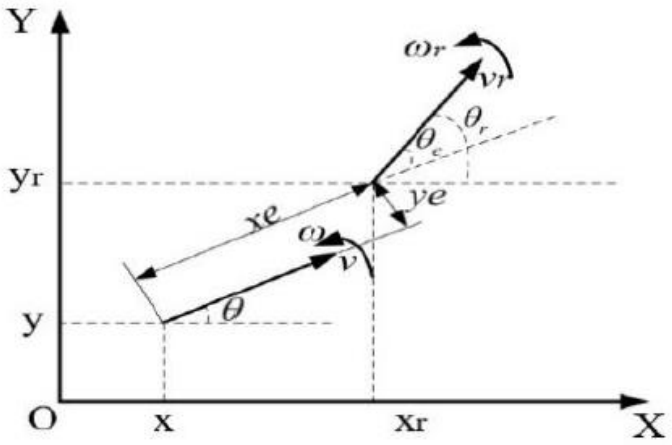

Figura 4.1. Error de posición de plataforma móvil. (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016)

El vector de error se define como $pe = (xe, ye, \theta e)^T$ en las coordenadas locales de la plataforma móvil. Por lo tanto, la ecuación del error se define como:

$$p_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} (p_r - p) = T(p_r - p) \tag{4.5}$$

Donde T_e es la matriz de transformación. Derivando la ecuación 4.5, se obtiene la siguiente ecuación diferencial del error utilizando las variables previamente definidas.

$$\dot{p_e} = \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{bmatrix} = \begin{bmatrix} \omega y_e - v - v_r cos(\theta_e) \\ -\omega y_e + v_r sin(\theta_e) \\ \omega_r - \omega \end{bmatrix} \tag{4.6}$$

El objetivo de este controlador seguidor de trayectoria es basarse en el modelo cinemático de la plataforma móvil y determinar la velocidad $q(v, \omega)^T$ de la plataforma para que esta siga la posición de referencia $pr = (xr, yr, \theta r)^T$ y la velocidad de referencia $qr = (vr, \omega r)^T$ al mismo tiempo que reduce el error inicial de posición a cero. La estructura del controlador es mostrada en la *Figura 4.2*.

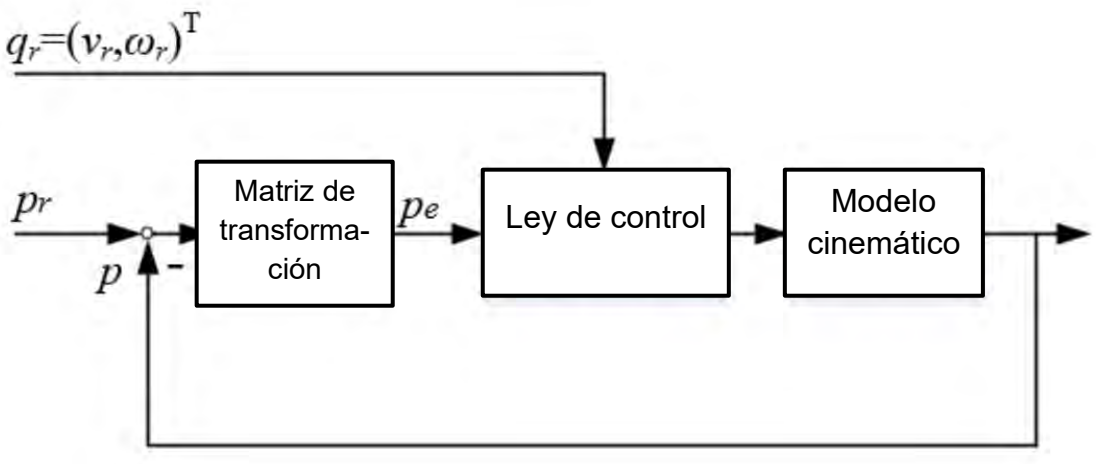

Figura 4.2. Estructura del controlador Backstepping. (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016)

Usando el concepto del Backstepping, el componente de error x_e en el robot, se toma como la variable de control virtual, mientras que $a_1 \sin(\arctan(\omega)) y_e$ es la realimentación virtual idea (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016). Por lo tanto, la nueva variable virtual de error es:

$$\overline{x_e} = x_e - a_1 \sin(\arctan(\omega))\, y_e \tag{4.7}$$

Donde a_1 es una constante mayor a 0. La esencia de esta ley de control es buscar la entrada de control $q(v, \omega)^T$, de tal manera que cuando $t \to \infty$, $x_e \to a_1 \sin(\arctan(\omega))\, y_e$, $\theta_e \to 0$. La función de Lyapunov es construida teniendo en cuenta la anterior idea (Chang, Jun-Hong, Hui, & Jin-Zhong, 2016).

$$V = \frac{1}{2}\overline{x_e^2} + \frac{1}{2} y_e^2 + \frac{2}{a_3}\left(1 - \cos\left(\frac{\theta_e}{2}\right)\right) \tag{4.8}$$

Donde a_3 es una constante mayor a 0. Ya que $a_3 > 0$, está claro que $V \geq 0$, la función de Lyapunov es igual a cero solo si $(xe, ye, \theta e)^T = 0$. La ecuación 4.7 es convertida a la siguiente.

$$\overline{x_e} = x_e - \frac{a_1 \cos(\arctan(\omega))\omega\, y_e}{1 + \omega^2} - a_1 \sin(\arctan(\omega))\, y_e \tag{4.9}$$

Derivamos la Ecuación de Lyapunov para obtener la ley de control

$$\begin{aligned} \dot{V} = \overline{x_e}\Big[& -v + v_r \cos(\theta_e) - a_1 \cos(\arctan(\omega)) \frac{1}{1+\omega^2} \omega\, y_e \\ & - a_1 \sin(\arctan(\omega))(-\omega x_e + v_r \sin(\theta_e))\Big] \\ & - a_1 {y^2}_e \omega \sin(\arctan(\omega)) + \frac{\sin\left(\frac{\theta_e}{2}\right)}{a_3} [-\omega + \omega_r \\ & + 2a_3 y_e v_r \cos\left(\frac{\theta_e}{2}\right) + a_4 \sin\left(\frac{\theta_e}{2}\right)] \end{aligned} \tag{4.10}$$

Las ecuaciones de la ley de control para implementar son las siguientes.

$$v = v_r \cos(\theta_e) + a_1 \sin[(\arctan(\omega)\omega\, x_e) - a_1 v_r \sin(\arctan(\omega)) \sin(\theta_e) - a_1 \cos(\arctan(\omega))] \frac{1}{1+\omega^2}\omega\, y_e + a_2[x_e - a_1 \sin(\arctan(\omega)) y_e] \quad (4.11)$$

$$\omega = \omega_r + 2a_3 y_e v_r \cos\left(\frac{\theta_e}{2}\right) + a_4 \sin\left(\frac{\theta_e}{2}\right) \quad (4.12)$$

Donde a2 y a4 con constantes mayores a 0. Finalmente, se simplifica la derivada de la función de Lyapunov presente en la ecuación (4.10), teniendo en cuenta las ecuaciones (4.11) y (4.12).

$$\dot{V} = -a_2 \overline{x_e^2} - a_1 y_e^2 \omega \sin(\arctan(\omega)) - \frac{a_4}{a_3} \sin^2\left(\frac{\theta_e}{2}\right) \quad (4.13)$$

4.3.2 Controlador de plataforma móvil basado en Feedback Linearization

El planteamiento de este controlador está basado en el diseño presente en (Kowalczyk & Kozlowski, 2014). Este usa el modelo de la ecuación 2.1 y las variables utilizadas se encuentran en el esquema de la *Figura 4.3.*

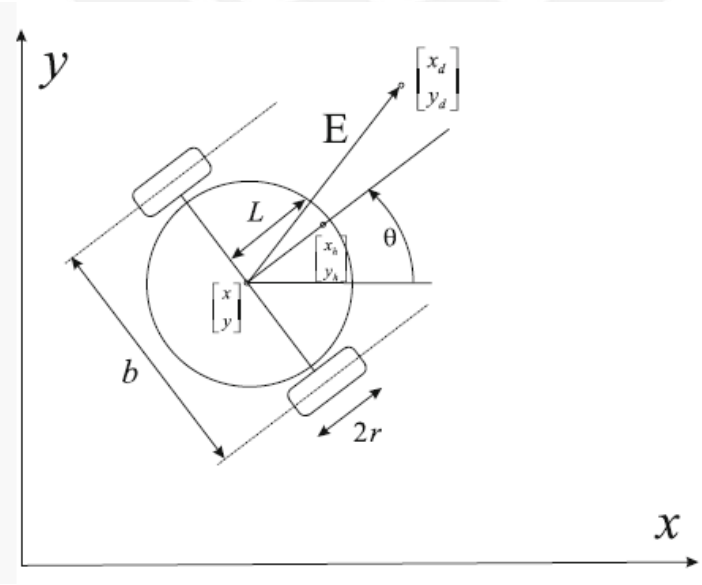

Figura 4.3. Esquema utilizado para controlador por Feedback Linearization. (Kowalczyk & Kozlowski, 2014)

El modelo del robot es linealizado utilizando el método de linealización por retroalimentación para simplificar el movimiento del robot. La posición $h = [x_h, y_h]^T$ es la nueva coordenada de salida del sistema, ubicada a una distancia del eje de las orugas, esta posición está dada por la siguiente ecuación:

$$h = \begin{bmatrix} x + Lcos(\theta) \\ y + Lsin(\theta) \end{bmatrix} \tag{4.14}$$

Derivando la ecuación (4.14) con respecto al tiempo y substituyendo (2.1) se obtiene:

$$\dot{h} = \begin{bmatrix} v - L.\omega.sin(\theta) \\ v + L.\omega.cos(\theta) \end{bmatrix} \tag{4.15}$$

Reemplazando $\dot{h}$ con la nueva entrada $[u_x, u_y]^T$ y transformando la ecuación para obtener las velocidades de la plataforma móvil en el lado izquierdo de la ecuación. Obtenemos la ley de control linealizada.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1}{L} \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{4.16}$$

Sustituyendo (4.15) en (4.14) y simplificando el modelo del sistema linealizado por realimentación, obtenemos:

$$\dot{h} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{4.17}$$

Para el sistema transformado, se aplica un control proporcional, donde Kp es la ganancia de control y $[x_t, y_t]$ es la posición deseada.

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = K_p \left(\begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right) \tag{4.18}$$

4.4 Diseño del controlador para sistema de articulaciones.

Las acciones de control correspondientes al movimiento de cada Flipper, tal como se muestra en la ecuación (2.30), se obtienen al superponer dos acciones por separado. La primera acción es la es obtenido por un controlador de seguimiento de posición en Z y de ángulo pitch; mientras que la segunda acción de control es la encargada de estabilizar el ángulo roll del robot, actuando en el grupo adecuado de Flippers (lado izquierdo o lado derecho). Por lo tanto, se diseñan dos sub-controladores por separado. En la presente sección se presenta

el diseño de dos (02) subcontroladores para el seguimiento de la posición en Z y el ángulo pitch de la plataforma móvil del robot articulado con orugas, y de un sub-controlador de regulación del ángulo de roll de la plataforma móvil del robot articulado con orugas. Para el primer sub-controlador se implementarán y analizarán 2 técnicas de control, gain scheduling y control por lógica difusa, mientras que para el segundo sub-controlador, se implementa un control proporcional para la estabilización del ángulo roll.

Cada técnica de control será abordada de manera detallada, basándose en una revisión de la literatura existente. Esto incluye la descripción de la metodología seguida por la derivación de las ecuaciones de control específicas para el modelo del robot.

El esquema de control, el cual integra a los dos sub-controladores, se muestra en la *Figura 4.4.*

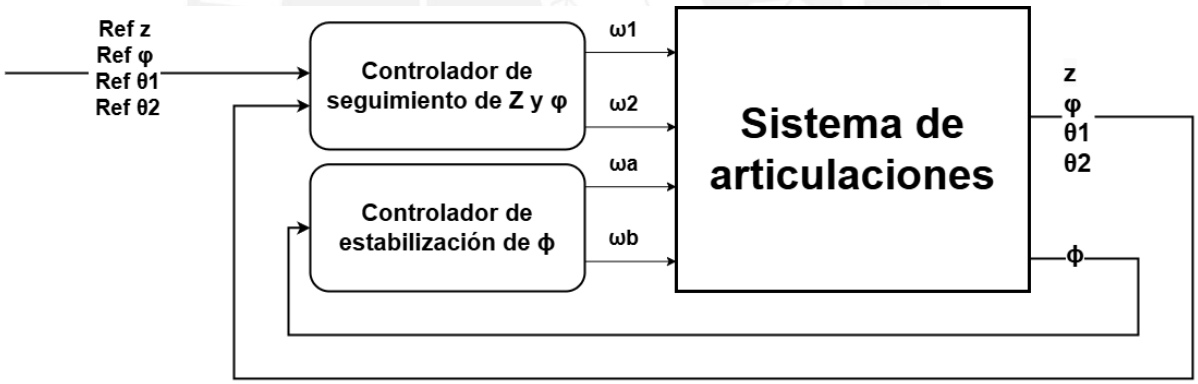

Figura 4.4. Esquema de control para sistema de articulaciones. Elaboración propia

4.4.1 Controlador Gain Scheduling de seguimiento de Z y pitch

El primer controlador se diseña como uno basado en Gain Scheduling. Esto implica linealizar el sistema para distintos puntos de equilibrio, por lo cual el primer paso es obtener una linealización para un punto de equilibrio genérico. El sistema a linealizar es el que se encuentra en la ecuación (2.22).

La linealización del sistema se describe de la siguiente manera:

$$\dot{X}lin = \begin{bmatrix} \dot{Z}lin \\ \dot{\varphi}lin \\ \dot{\theta_1}lin \\ \dot{\theta_2}lin \end{bmatrix} =$$

$$\begin{bmatrix} \left.\frac{d\dot{Z}}{dZ}\right|_{eq}(\bar{Z}) + \left.\frac{d\dot{Z}}{d\varphi}\right|_{eq}(\bar{\varphi}) + \left.\frac{d\dot{Z}}{d\theta_1}\right|_{eq}(\overline{\theta_1}) + \left.\frac{d\dot{Z}}{d\theta_2}\right|_{eq}(\overline{\theta_2}) + \left.\frac{d\dot{Z}}{d\omega_1}\right|_{eq}(\omega_1) + \left.\frac{d\dot{Z}}{\omega_2}\right|_{eq}(\omega_2) \\ \left.\frac{d\dot{\varphi}}{dZ}\right|_{eq}(\bar{Z}) + \left.\frac{d\dot{\varphi}}{d\varphi}\right|_{eq}(\bar{\varphi}) + \left.\frac{d\dot{\varphi}}{d\theta_1}\right|_{eq}(\overline{\theta_1}) + \left.\frac{d\dot{\varphi}}{d\theta_2}\right|_{eq}(\overline{\theta_2}) + \left.\frac{d\dot{\varphi}}{d\omega_1}\right|_{eq}(\omega_1) + \left.\frac{d\dot{\varphi}}{\omega_2}\right|_{eq}(\omega_2) \\ \omega_1 \\ \omega_2 \end{bmatrix} \quad (4.19)$$

Resolviendo las derivadas obtenemos la siguiente función, la cual linealiza el sistema para un punto de equilibrio genérico:

$$\dot{X}lin = \begin{bmatrix} \dot{Z}lin \\ \dot{\varphi}lin \\ \dot{\theta_1}lin \\ \dot{\theta_2}lin \end{bmatrix} = \begin{bmatrix} f1(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}).(\omega_1) + f2(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}).(\omega_2) \\ f3(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}).(\omega_1) + f4(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}).(\omega_2) \\ \omega_1 \\ \omega_2 \end{bmatrix} \quad (4.20)$$

Reescribiendo el sistema de como un sistema espacio-estados lineal:

$$\dot{X}lin = \begin{bmatrix} \dot{Z}lin \\ \dot{\varphi}lin \\ \dot{\theta_1}lin \\ \dot{\theta_2}lin \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z \\ \varphi \\ \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} f1(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}) & f2(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}) \\ f3(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}) & f4(\theta_{1eq}, \theta_{2eq}, \varphi_{eq}) \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}$$

$$\dot{X}lin = \begin{bmatrix} \dot{Z}lin \\ \dot{\varphi}lin \\ \dot{\theta_1}lin \\ \dot{\theta_2}lin \end{bmatrix} = [A][X] + [B][\omega] \quad (4.21)$$

En este caso, la matriz A se mantiene constante mientras que la matriz B depende del punto de equilibrio seleccionado. El punto de equilibrio se define de las ecuaciones de la geometría de las articulaciones, descritas en las ecuaciones (2.10) a (2.13). Estas ecuaciones se acomodan de tal manera que θ_{1eq} y θ_{2eq} dependan de Z_{eq} y φ_{eq}.

$$X_{eq} = \begin{bmatrix} Z_{eq} \\ \varphi_{eq} \\ \left(\frac{Z_{eq} + \frac{L}{2} * \sin(\varphi_{eq})}{Lf}\right) + \varphi_{eq} \\ \left(\frac{Z_{eq} - \frac{L}{2} * \sin(\varphi_{eq})}{Lf}\right) - \varphi_{eq} \end{bmatrix} \tag{4.22}$$

Para el diseño del controlador tipo gain scheduling se definen 10 puntos de equilibrio los cuales los cuales se distribuyen uniformemente desde el punto inicial del robot hasta el punto deseado de seguimiento. De esta manera se genera de manera dinámica las ganancias del controlador dependiendo de la acción a realizar. Las ecuaciones de las acciones de control son las siguientes:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = [K].[X_{ref} - X] = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} \begin{bmatrix} Z_{ref} - Z \\ \varphi_{ref} - \varphi \\ \theta_{1ref} - \theta_1 \\ \theta_{2ref} - \theta_2 \end{bmatrix} \tag{4.23}$$

Donde la matriz de ganancias K se obtiene utilizando la ecuación de Riccati, utilizando las siguientes matrices Q y R.

$$Q = \begin{bmatrix} c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & c \end{bmatrix} \tag{4.24}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.25}$$

Donde c es una constante que se calibra de manera iterativa durante las simulaciones.

Finalmente, se define la ley para determinar los parámetros de control, mediante esta ley se selecciona la matriz K adecuada dependiendo del valor de los estados

de las articulaciones. Debido a que existen 10 puntos de equilibrio, también existen 10 matrices K, de K1 a K10, donde K1 es la matriz calculada con el punto de equilibrio igual al punto inicial, y K10 es la matriz calculada con el punto de equilibrio igual al punto deseado o punto de referencia. En la *Figura 4.5* se muestra el diagrama de flujo de la implementación de la ley utilizada para determinar los parámetros de control. Esta ley es una función con 3 entradas y una salida. Las 3 entradas son la matriz donde se encuentran los puntos de equilibrio, la matriz que contiene K1 a K10, y el valor actual de los estados del robot. La salida de la función es la matriz K de ganancias correspondiente al punto de equilibrio que más se acerque al estado actual del robot.

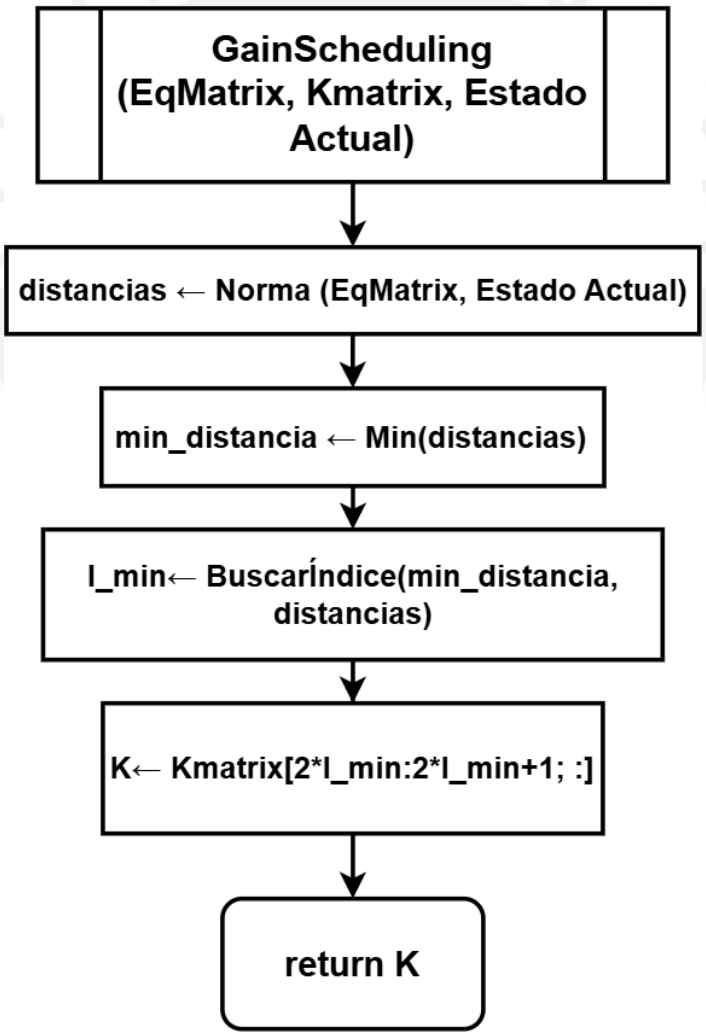

Figura 4.5. Ley utilizada para determinar los parámetros de control. Elaboración propia

4.4.2 Controlador de seguimiento por lógica difusa de Z y pitch

El segundo controlador se diseña como uno basado en control por lógica difusa. Este controlador utiliza 4 variables de entrada y 2 variables de salida. Las variables de entrada son en el error en Z (E_Z), error en pitch (E_φ), el ángulo theta1 (θ_1) y el ángulo theta 2 (θ_1). Cabe resaltar que se define el error como la diferencia entre el valor de referencia y el valor medido. Por otro lado, las variables de salida son las velocidades angulares de los flippers delanteros y traseros, omega 1 (ω_1) y omega 2 (ω_2) respectivamente.

El controlador diseñado utiliza funciones de membresía de que cuentan con perfiles trapezoidales para las regiones extremas, y perfiles triangulares para las regiones centrales. Las variables de entrada están representadas con funciones de membresía de 3 regiones, una elección que busca reducir el tiempo computacional y la complejidad de las reglas difusas. Por su parte, las variables de salida se describen con funciones de membresía de cinco regiones, lo cual permite un control más preciso sobre las salidas del sistema. Estas funciones se pueden ver en las *Figuras 4.6* a *4.11.*

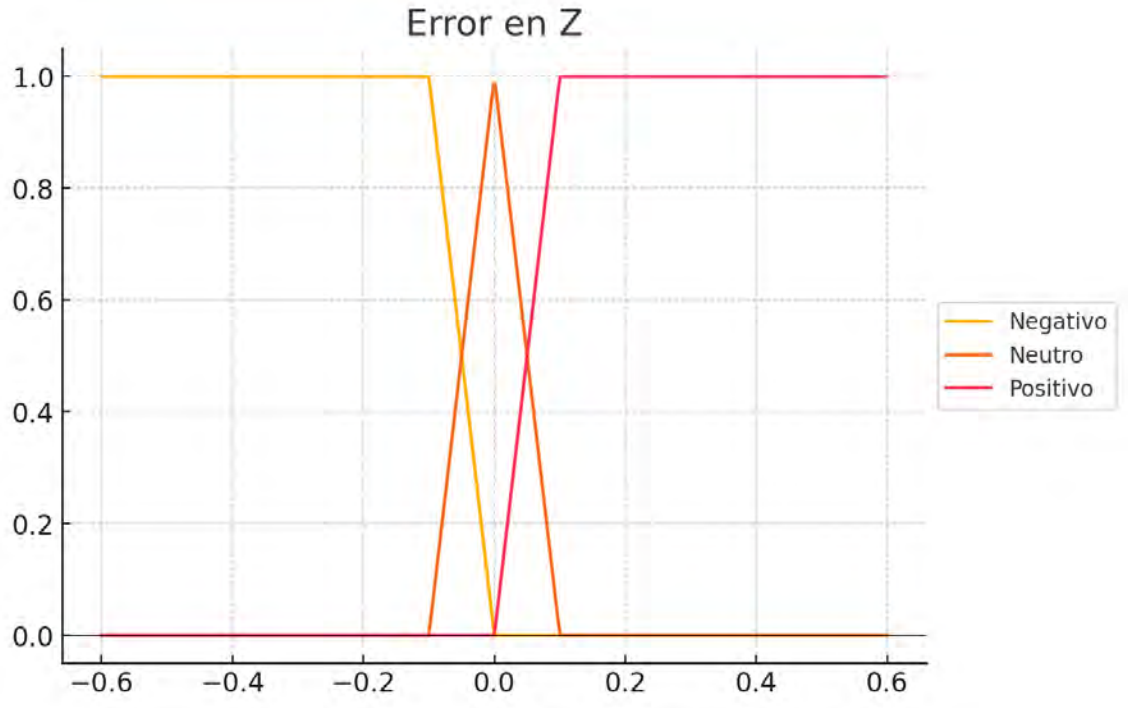

Figura 4.6. Función de membresía de Error en Z. Elaboración propia

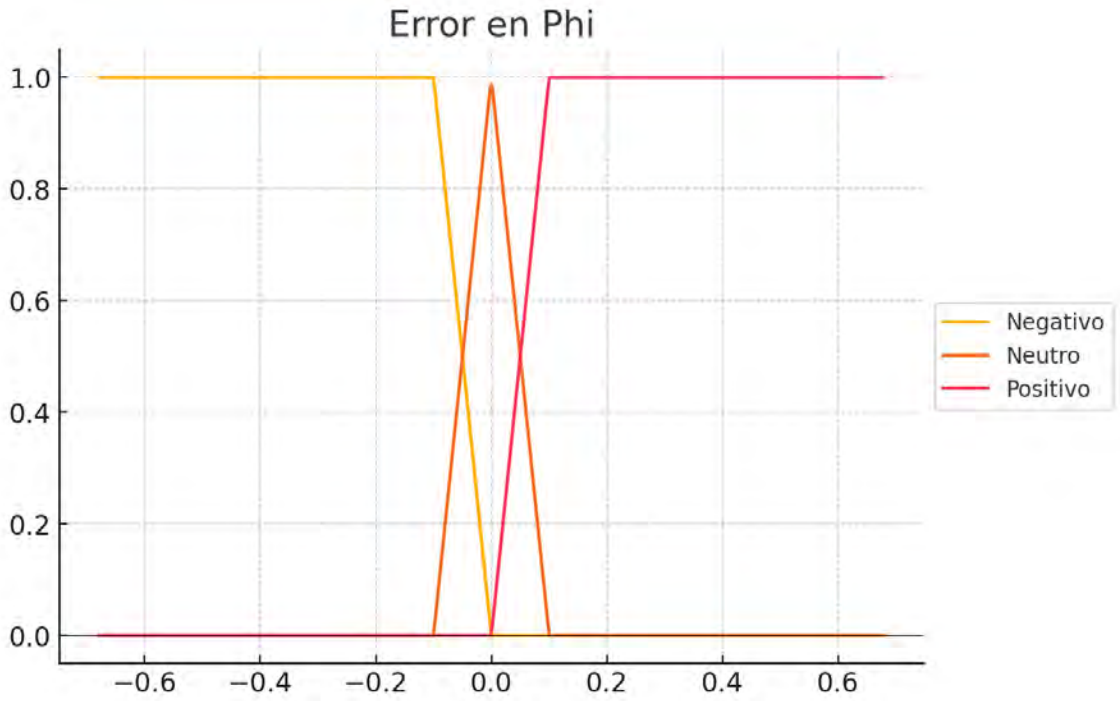

Figura 4.7. Función de membresía de Error en Phi. Elaboración propia

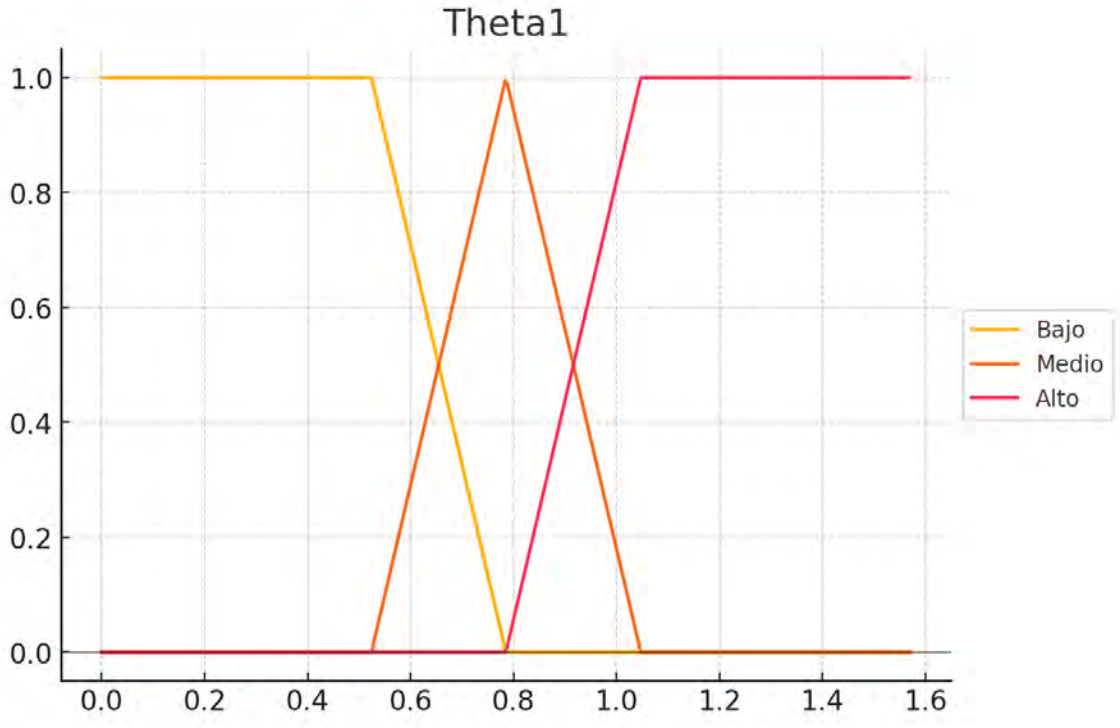

Figura 4.8. Función de membresía de Theta1. Elaboración propia

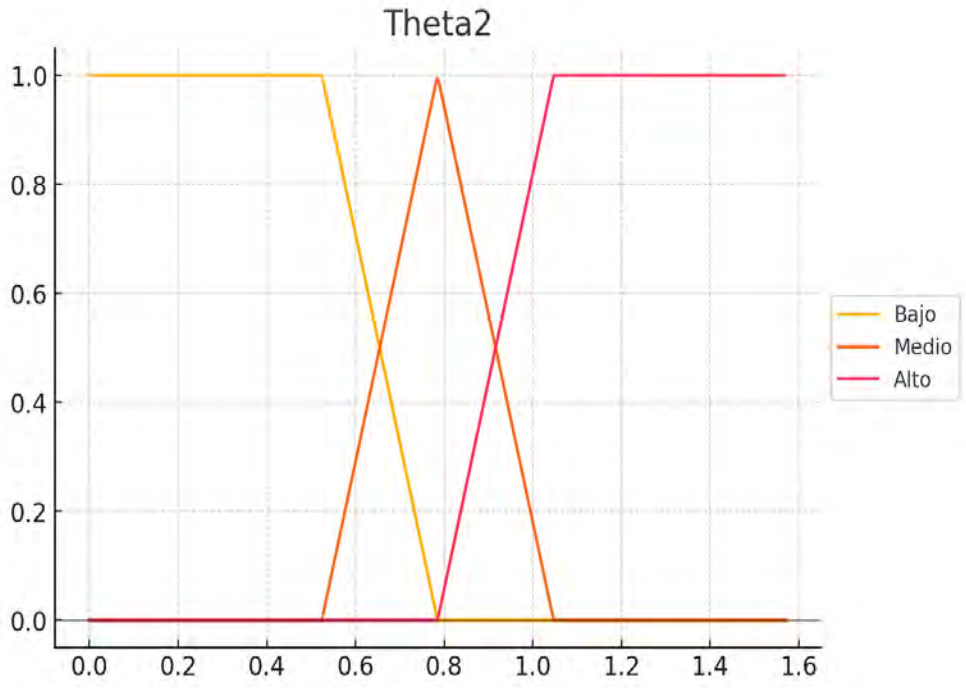

Figura 4.9. Función de membresía de Error en Z. Elaboración propia

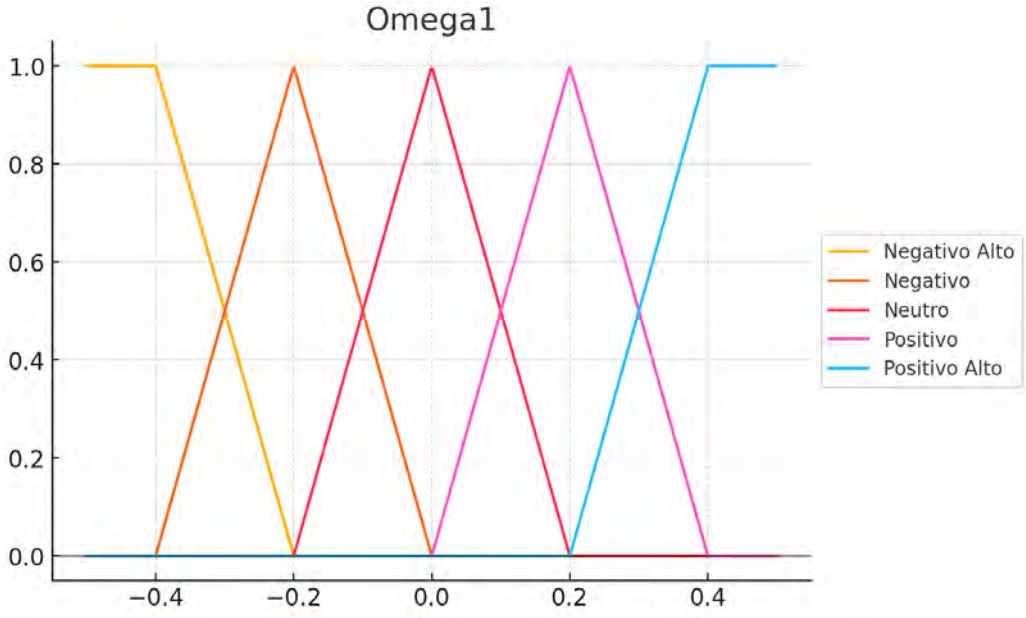

Figura 4.10. Función de membresía de Error en Omega 1. Elaboración propia

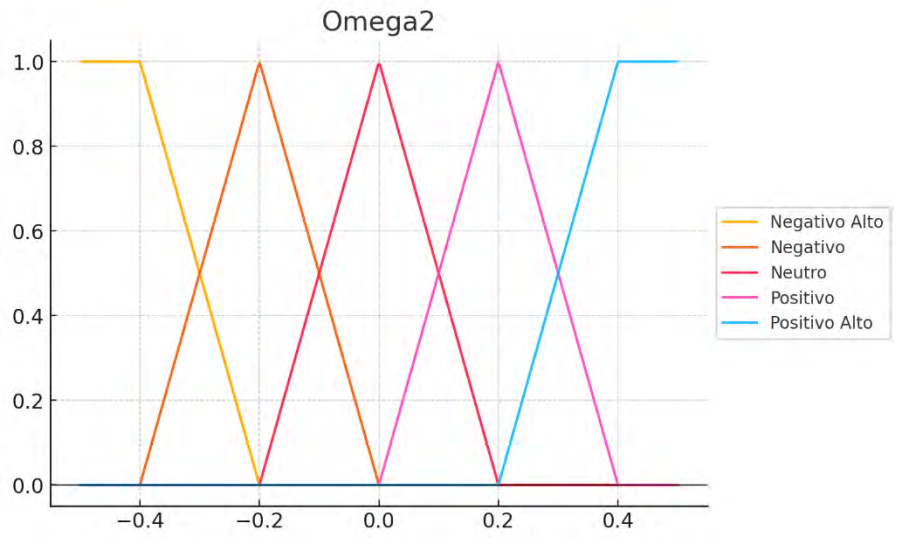

Figura 4.11. Función de membresía de Error en Omega 1. Elaboración propia

Luego de definir las funciones de membresía, se diseñan las reglas difusas, las cuales son obtenidas a base de la observación de diferentes robots articulados con orugas similares. Debido a que se tienen 4 funciones de membresías correspondientes a variables de entrada, y 3 particiones por función, son necesarias 81 reglas difusas. La lógica detrás de las reglas se resume en la *Figura 4.12.* A continuación se muestran cuatro (04) ejemplos de dichas reglas:

- Si “Error en Y” = “Negativo” & “Error en Phi” = “Negativo” & “Theta1” = “Bajo” & “Theta2” = “Bajo” , entonces “Omega1” = “Negativo” & “Omega2” = “Neutro”
- Si “Error en Y” = “Negativo” & “Error en Phi” = “Negativo” & “Theta1” = “Medio” & “Theta2” = “Alto” , entonces “Omega1” = “Negativo Alto” & “Omega2” = “Negativo Alto”
- Si “Error en Y” = “Neutro” & “Error en Phi” = “Neutro” & “Theta1” = “Medio” & “Theta2” = “Alto” , entonces “Omega1” = “Negativo” & “Omega2” = “Positivo”
- Si “Error en Y” = “Positivo” & “Error en Phi” = “Positivo” & “Theta1” = “Alto” & “Theta2” = “Alto” , entonces “Omega1” = “Positivo” & “Omega2” = “Neutro”

Finalmente, el proceso de inferencia difusa y defuzzificación se implementan utilizando el método de Mandami, el cual usa el método MIN-MAX para la inferencia difusa, y el método de centro de gravedad para la defuzzificación.

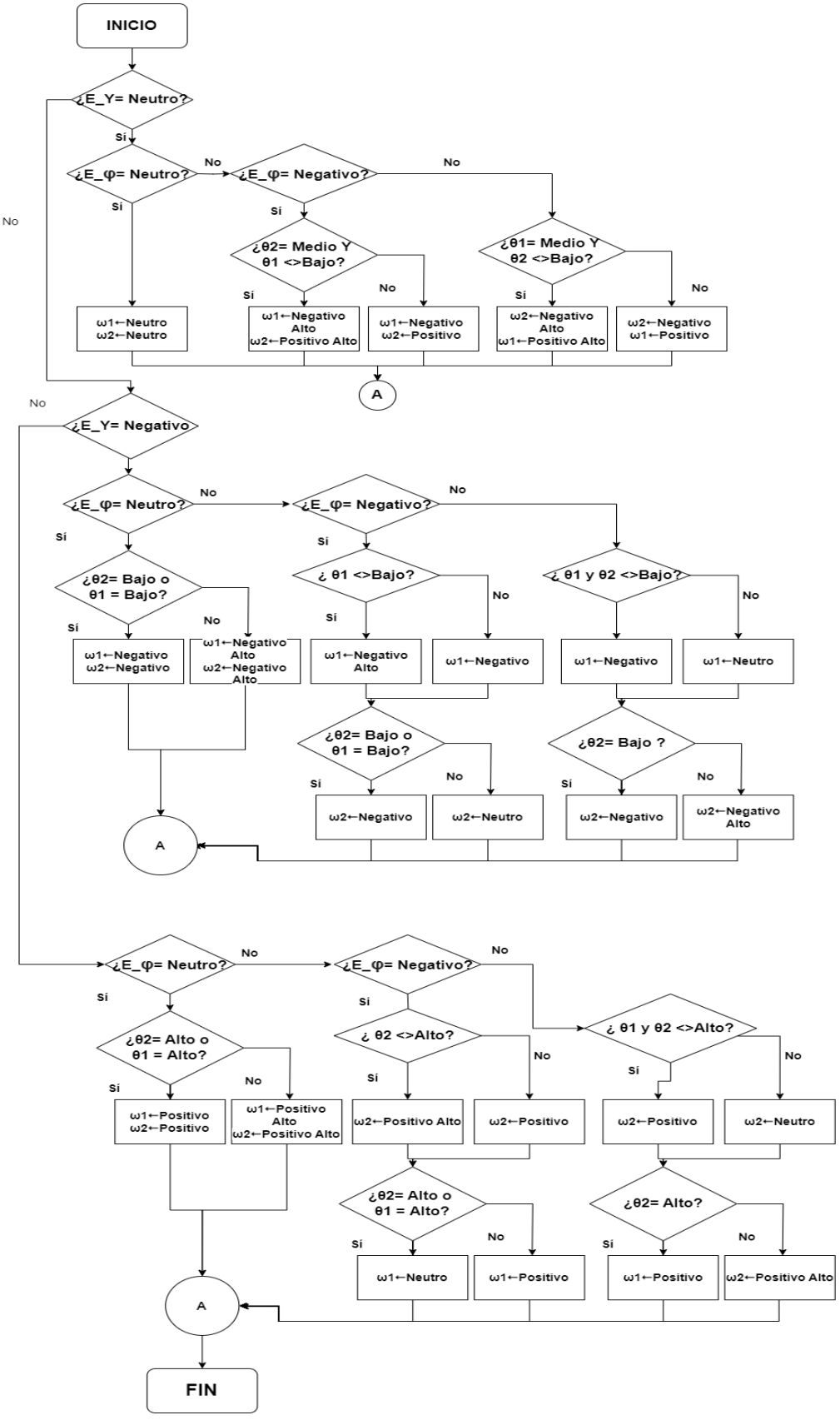

Figura 4.12. Lógica detrás de las reglas difusas. Elaboración propia

4.4.3 Controlador de estabilización de ángulo de roll

La estabilización del ángulo de roll se realiza con una acción de control proporcional dependiente solo del ángulo roll. Esta ley de control está basada en la implementación de (Azayev & Zimmermann, 2022), con la diferencia que la acción de control solo se aplica a un Flipper a la vez dependiendo de la inclinación del robot. Si la inclinación es positiva, la velocidad angular de los Flippers del lado derecho será positiva, mientras que, si la inclinación es negativa, la velocidad angular del lado derecho será positiva. La ley de control se muestra a continuación.

$$\omega_a = \begin{cases} 0, si\ \phi \leq 0 \\ Kp.\phi, si\ \phi > 0 \end{cases} \tag{4.26}$$

$$\omega_b = \begin{cases} 0, si\ \phi \geq 0 \\ -Kp.\phi, si\ \phi < 0 \end{cases} \tag{4.27}$$

Donde ω_a, ω_b son las velocidades de los Flippers del lado derecho e izquierdo, respectivamente, Kp es la ganancia proporcional, ϕ es el ángulo de roll.

4.5 Comparativa de controladores y selección para plataforma móvil.

En esta sección se implementarán los controladores diseñados en la sección 4.3 (Backstepping y Feedback Linearization) y se comparará su desempeño siguiendo una trayectoria en forma de lemniscata. La comparación se basará en parámetros cuantitativos específicos que evaluarán la precisión y la robustez de cada controlador. Estos parámetros incluyen el Error Cuadrático Medio (MSE), el Error Absoluto Medio (MAE), el Máximo Error Absoluto (MaxAE) y el Tiempo de Respuesta. Los resultados de las simulaciones se presentarán mediante gráficos y una tabla comparativa para facilitar la selección del controlador más adecuado para la plataforma móvil del robot.

4.5.1 Implementación de los controladores de la plataforma móvil

Cada uno de los controladores a implementar siguen los diseños detallados en la sección 4.3. Los parámetros utilizados en cada caso, necesarios para obtener los resultados presentados en este capítulo, se muestran en la *Tabla 4.2*.

Tabla 4.2. Parámetros para los controladores de la plataforma móvil. Elaboración propia

Característica	**Backstepping**	**Feedback Linearization**
Ecuaciones de control	(4.11), (4.12)	(4.16), (4.18)
Parámetros	a1, a2, a3, a4	Kp, V
Valores de los parámetros	a1=0.085 a2=0.25 a3=810 a4=2	Kp=8 V=0.4

Con motivos de la implementación, la trayectoria a seguir está descrita por la siguiente ecuación. La trayectoria es seguida por ambos controladores por una duración de 25 segundos.

$$y_d = 3\sin\left(2\pi.\frac{x_d}{10}\right).\cos\left(2\pi.\frac{x_d}{10}\right), x_d = 3\sin\left(2\pi.\frac{x_d}{10}\right), \quad (4.28)$$
$$x_d(0) = 0, y_d(0) = 0$$

En la Figura 4.13 y Figura 4.14, se muestra la comparación de la trayectoria deseada con la trayectoria obtenida por el controlador basado en Backstepping, y la obtenida por el controlador basado en Feedback Linearization, respectivamente. Asimismo, en la Figura 4.15 y Figura 4.16 se muestra el error porcentual con respecto a la trayectoria deseada, obtenida por el controlador basado en Backstepping, y el obtenido por el controlador basado en Feedback Linearization, respectivamente. En la Figura 4.17 y la Figura 4.18 se muestran las señales de control correspondientes a los controladores de Backstepping y Feedback Linearization, respectivamente.

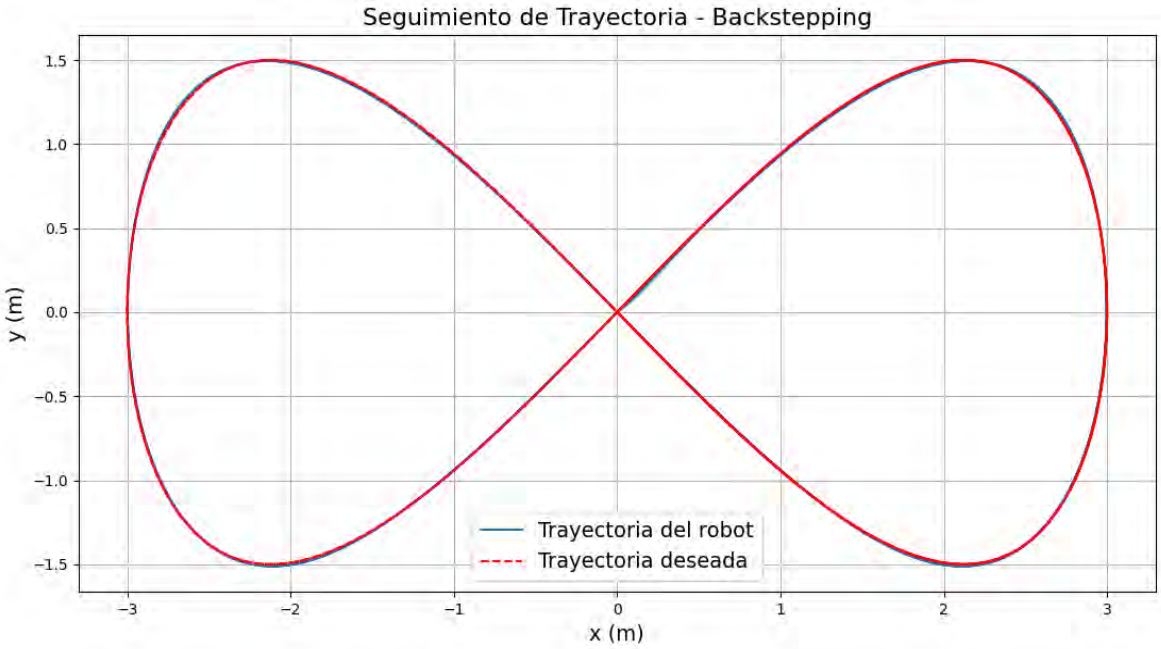

Figura 4.13. Trayectoria obtenida por controlador Backstepping. Elaboración propia

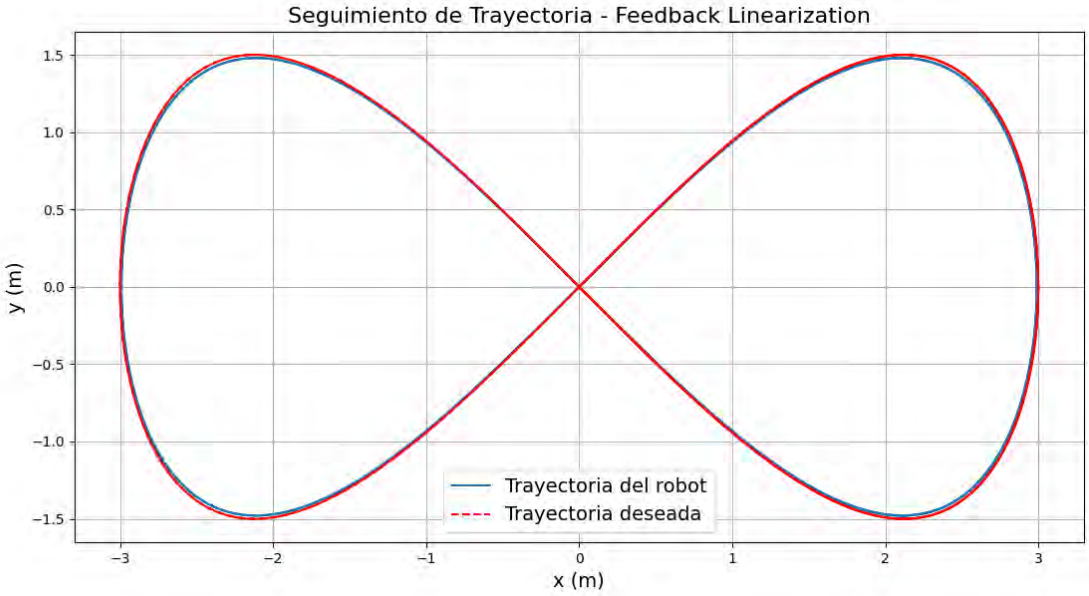

Figura 4.14. Trayectoria obtenida por controlador Feedback Linearization vs Trayectoria deseada. Elaboración propia

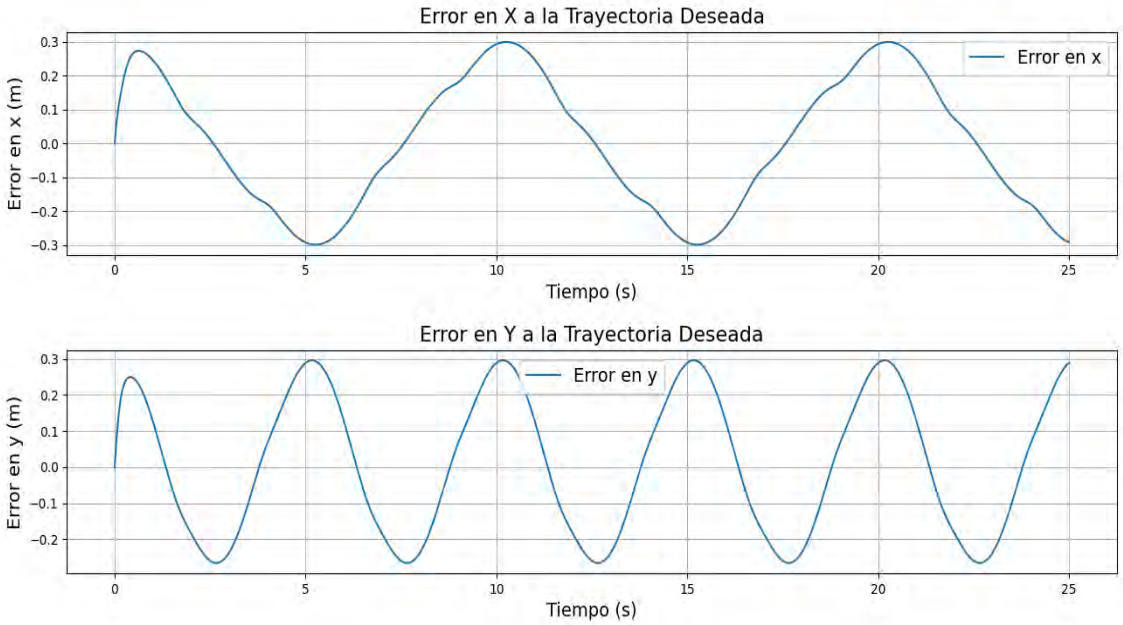

Figura 4.15. Error con respecto a la trayectoria – Controlador Backstepping. Elaboración propia

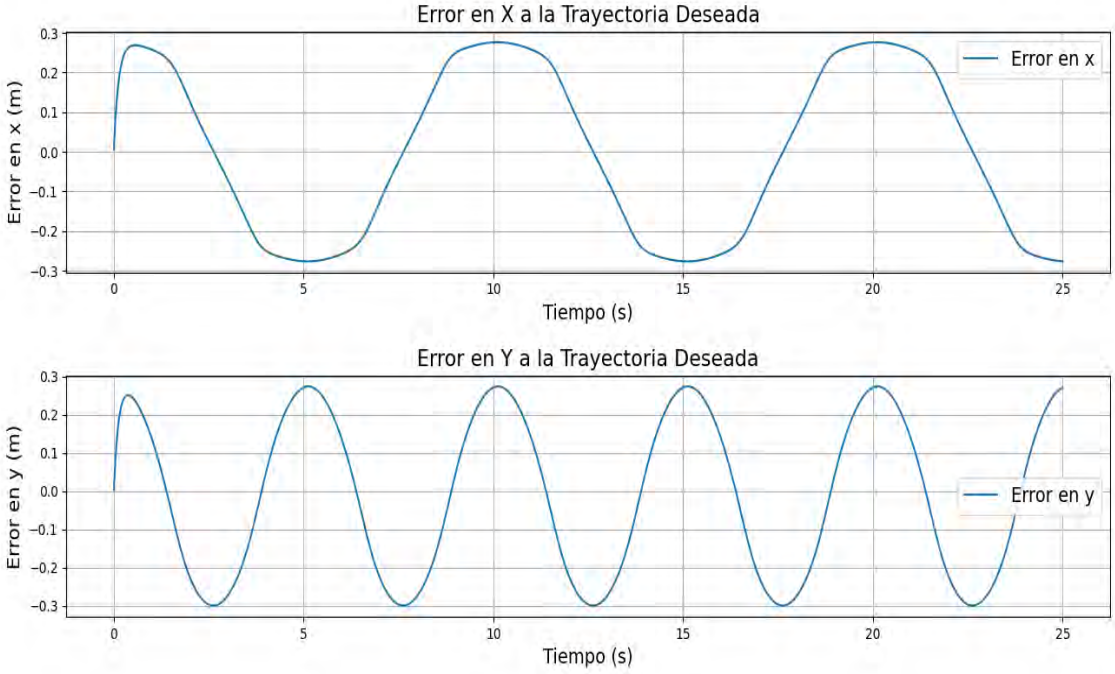

Figura 4.16. Error con respecto a la trayectoria – Controlador Feedback Linearization. Elaboración propia

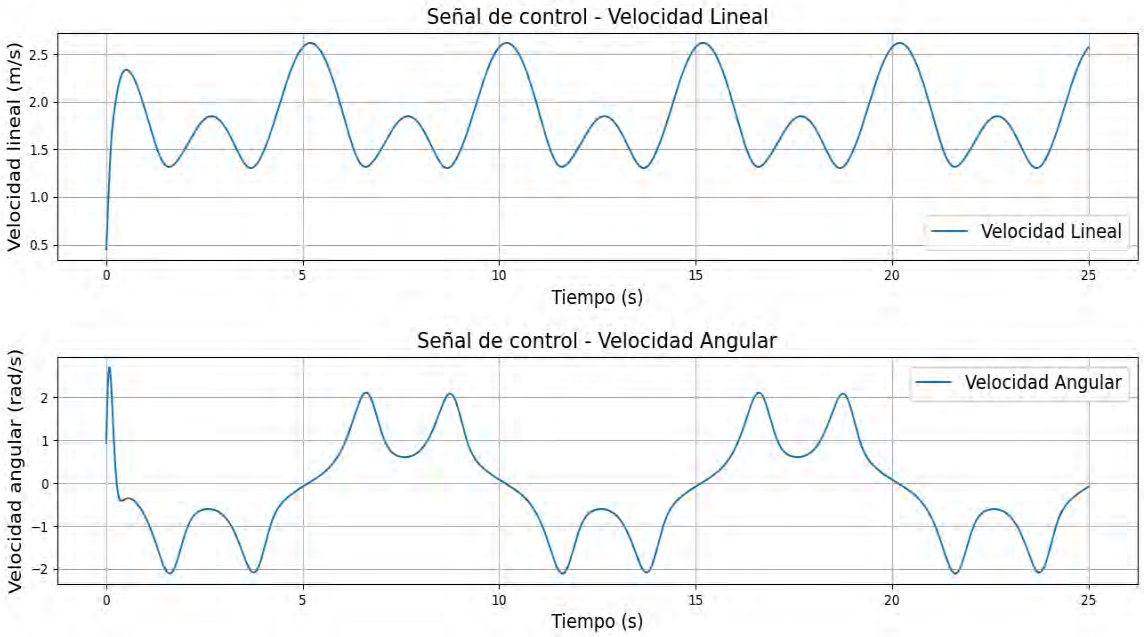

Figura 4.17 Señales de control – Controlador Backstepping. Elaboración propia

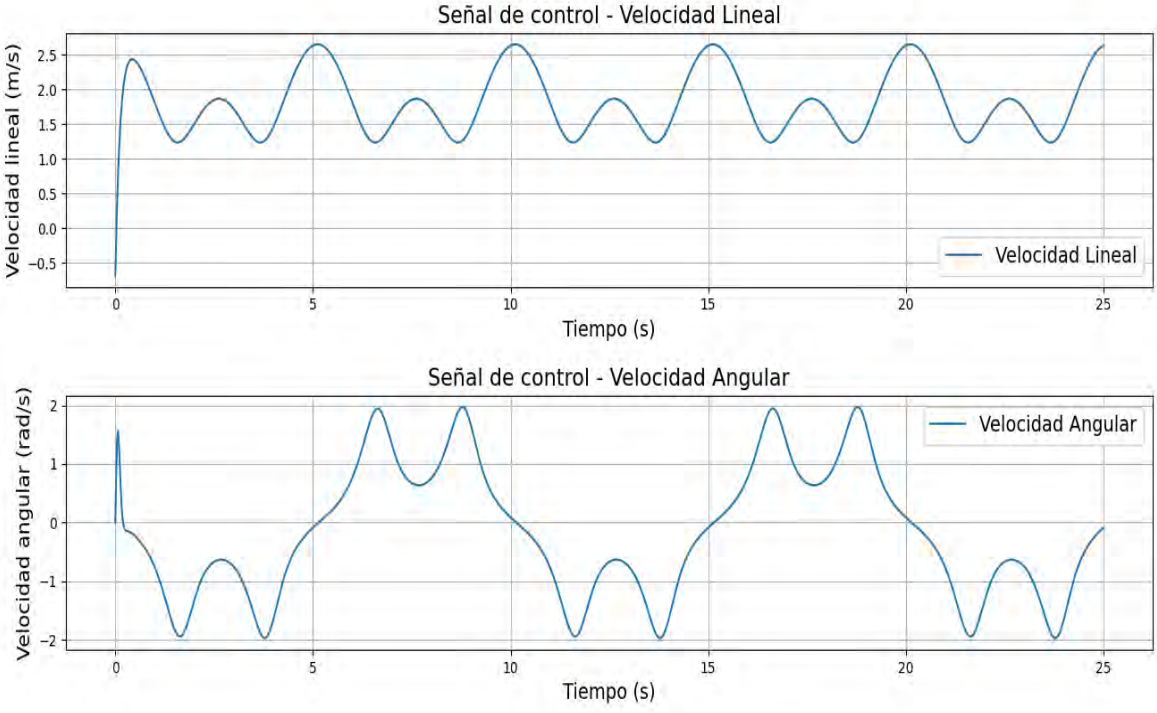

Figura 4.18. Señales de control – Controlador Feedback Linearization. Elaboración propia

Se puede observar en la Figura 4.13 y la Figura 4.14, que, si bien ambos tienen el mismo tiempo de simulación, recorren distancias distintas, esto se debe a que el controlador Backstepping puede alcanzar una mayor velocidad sin comprometer su precisión, y a que en esta prueba los controladores están siguiendo la trayectoria de la ecuación 4.28, no un punto fijo, debido a esto, los puntos finales varían para cada controlador.

A manera de simular el comportamiento del controlador ante ruido en los sensores, se añade ruido Gaussiano con media 0 y desviación estándar de 0.05 en la entrada de los controladores, introduciendo un nivel de perturbación moderado, utilizado para analizar la robustez del controlador. En la Figura 4.19 y Figura 4.20, se muestra la comparación de la trayectoria deseada con la trayectoria obtenida por el controlador basado en Backstepping, y la obtenida por el controlador basado en Feedback Linearization, respectivamente. Asimismo, en la Figura 4.21 y Figura 4.22 se muestra el error porcentual con respecto a la trayectoria deseada, obtenida por el controlador basado en Backstepping, y el obtenido por el controlador basado en Feedback Linearization, respectivamente. En la Figura 4.23 y la Figura 4.24 se muestran las señales de control correspondientes a los controladores de Backstepping y Feedback Linearization, respectivamente.

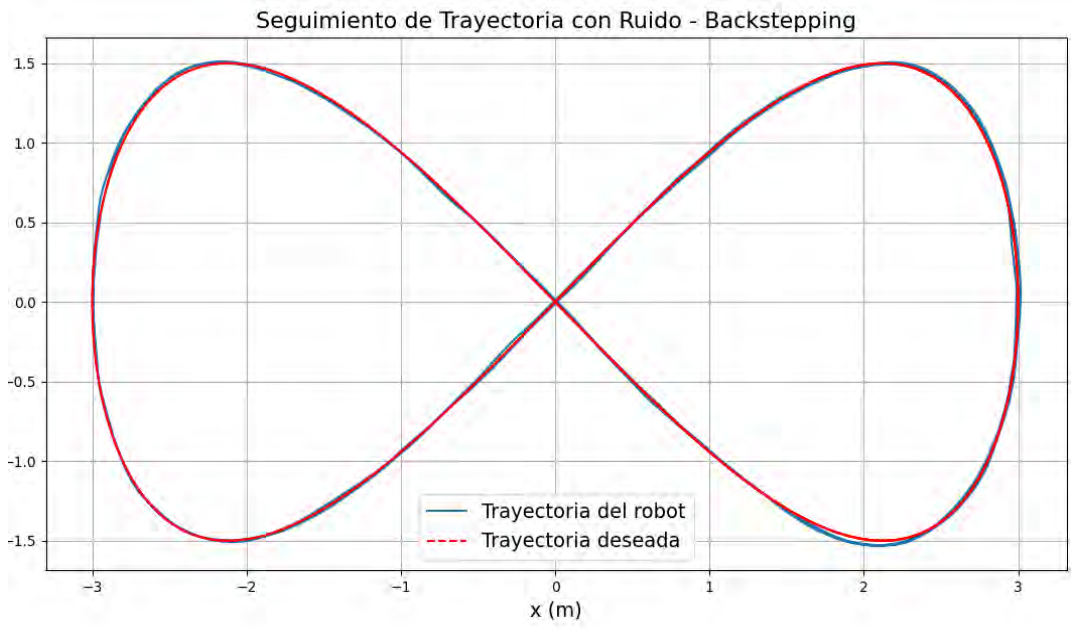

Figura 4.19. Trayectoria obtenida por controlador Backstepping afectado por ruido Gaussiano. Elaboración propia

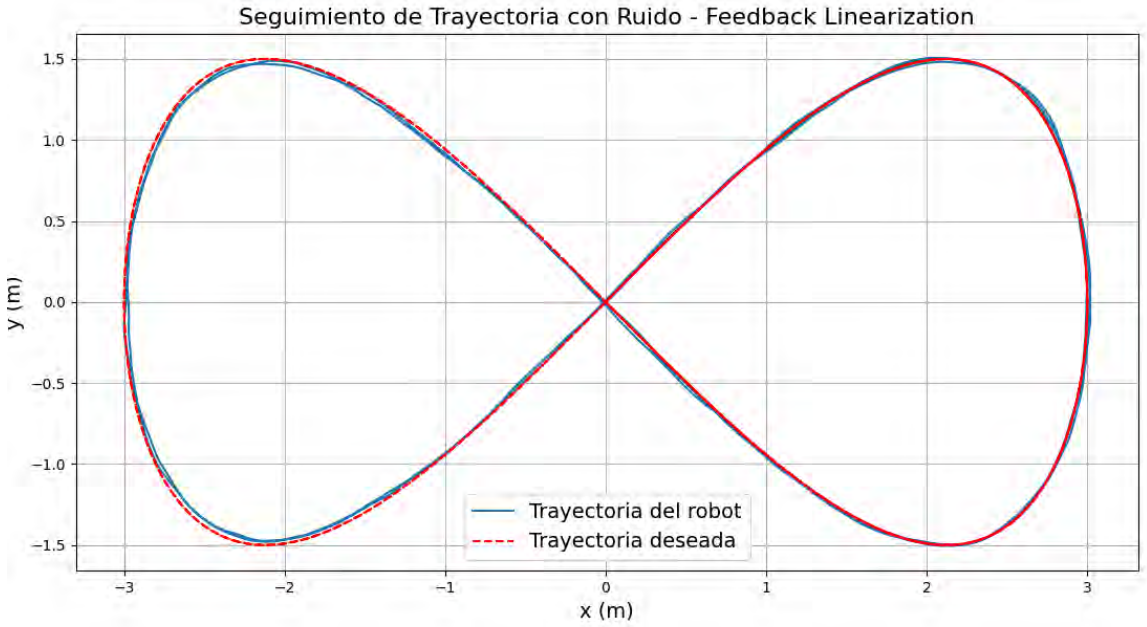

Figura 4.20. Trayectoria obtenida por controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia

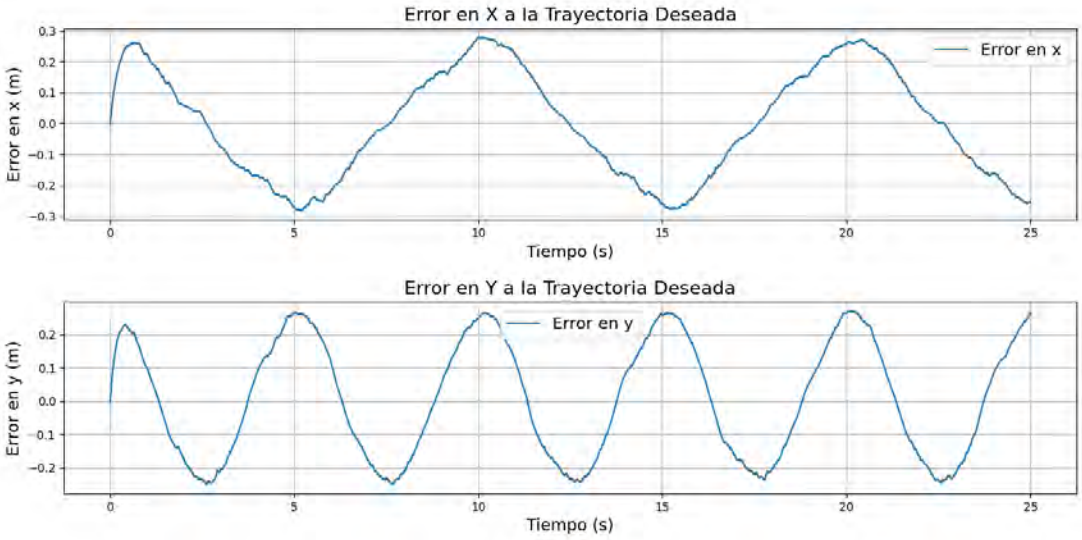

Figura 4.21. Error con respecto a la trayectoria – Controlador Backstepping afectado por ruido Gaussiano. Elaboración propia

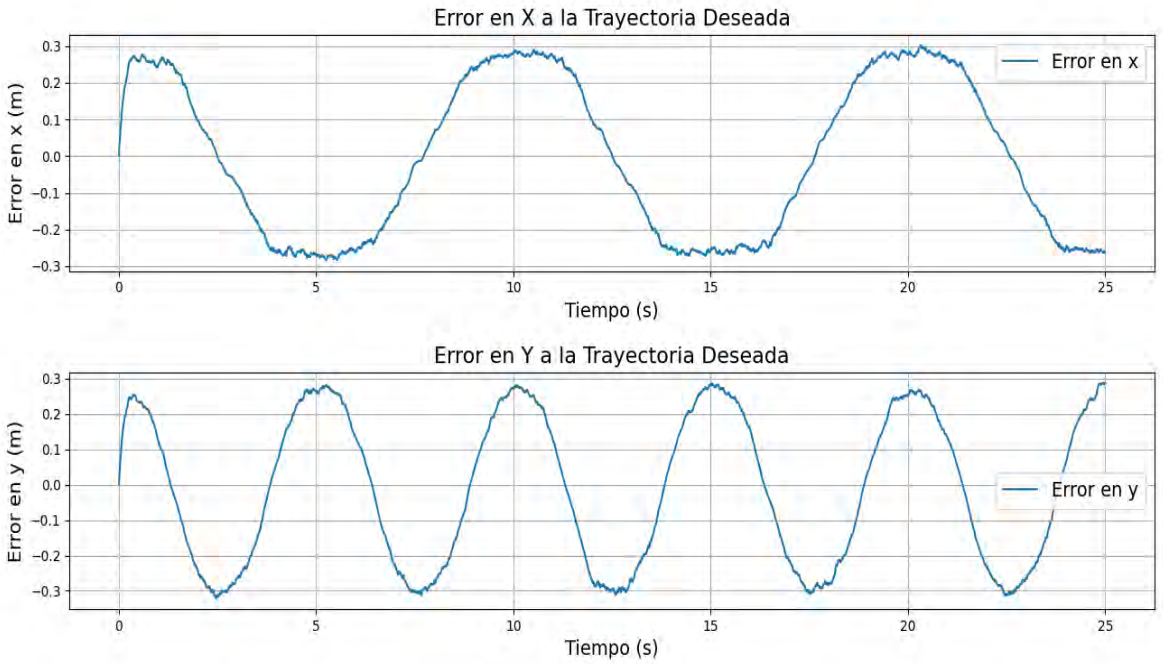

Figura 4.22. Error con respecto a la trayectoria – Controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia

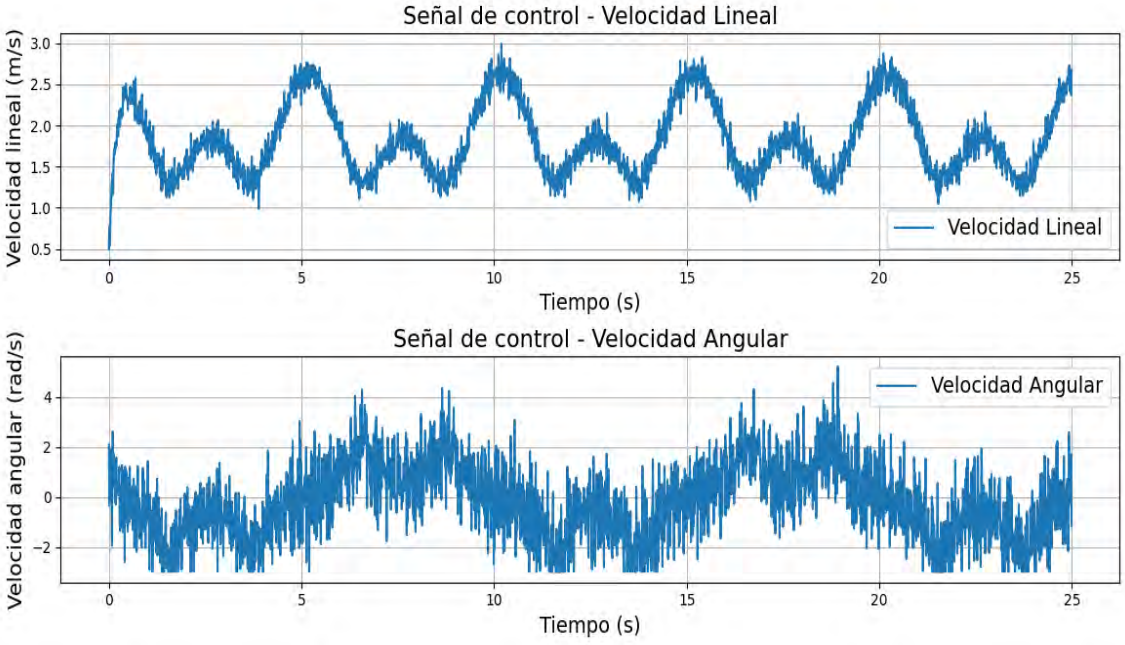

Figura 4.23 Señales de control – Controlador Backstepping afectado por ruido Gaussiano. Elaboración propia

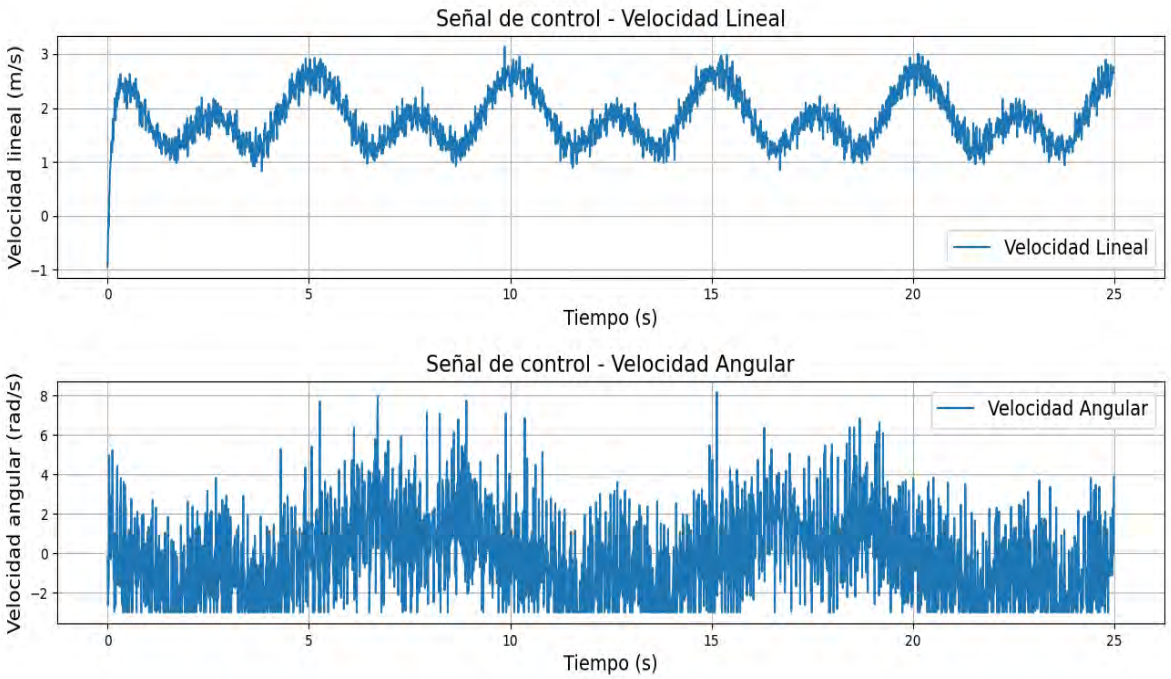

Figura 4.24. Señales de control – Controlador Feedback Linearization afectado por ruido Gaussiano. Elaboración propia

4.5.2 Parámetros de comparación

La comparación de los controladores se realiza utilizando los siguientes parámetros cuantitativos:

Error Cuadrático Medio (MSE): Mide la media del cuadrado de los errores entre la trayectoria deseada (lemniscata) y la trayectoria real del sistema.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \tag{4.29}$$

Donde e_i es el error en el instante i y N es el número total de instantes considerados.

Error Absoluto Medio (MAE): Mide la media de los valores absolutos de los errores entre la trayectoria deseada y la trayectoria real del sistema.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|e_i| \tag{4.30}$$

Máximo Error Absoluto (MaxAE): Mide la media de los valores absolutos de los errores entre la trayectoria deseada y la trayectoria real del sistema.

$$MaxAE = \max\left(|e_1|, |e_2|, \ldots, |e_n|\right) \tag{4.31}$$

Tiempo de Respuesta: El tiempo necesario para que el error de seguimiento permanezca dentro de un rango del ±2% y no vuelva a salir de este rango.

4.5.3 Comparativa de controladores para plataforma móvil y resultados

En la Tabla 4.3, se muestran el valor de los parámetros discutidos en la sección 4.4.2 para cada implementación de controlador. Basado en los resultados de las simulaciones, el controlador diseñado utilizando Backstepping muestra el mejor desempeño en términos de precisión y tiempo de respuesta, con los valores más bajos de MSE, MAE y MaxAE, tanto en los escenarios con y sin ruido en los sensores. Además, proporciona un tiempo de respuesta rápido, lo que indica una capacidad de seguimiento eficiente. Por lo tanto, el controlador a implementar en Gazebo para la plataforma móvil del robot articulado con orugas será el basado en Backstepping

Tabla 4.3. Métricas de los controladores de la plataforma móvil. Elaboración propia

Parámetro	Backstepping	Feedback Linearization	Backstepping + Ruido Gaussiano	Feedback Linearization + Ruido Gaussiano
MSE(x)	0.0370	0.0448	0.0301	0.0459
MSE(y)	0.0376	0.0432	0.0304	0.0448
MAE(x)	0.1691	0.1929	0.1526	0.1949
MAE(y)	0.1740	0.1888	0.1565	0.1924
MaxAE(x)	0.2993	0.2965	0.2875	0.3003
MaxAE(y)	0.2757	0.2994	0.2962	0.3253
Tiempo de respuesta (s)	1.35	1.62	1.35	1.62

4.6 Comparativa de controladores y selección para sistema de articulaciones.

En esta sección se implementarán los sub controladores diseñados en la sección 4.4, control por Gain Scheduling y control por lógica difusa para el sub controlador de seguimiento de Z y φ; así como el sub controlador proporcional para estabilización de ϕ. En cuanto al primer grupo de sub controladores, se comparará su desempeño siguiendo una señal tipo escalón. Por otro, lado, el controlador de estabilización será evaluado por su respuesta a distintas condiciones iniciales. La comparación se basará en parámetros cuantitativos específicos que evaluarán la precisión y la robustez de cada controlador. Estos parámetros incluyen el Tiempo de Respuesta, máximo sobre impulso y error en estado estacionario. Los resultados de las simulaciones se presentarán mediante gráficos y una tabla comparativa para facilitar la selección del controlador más adecuado para el sistema de articulaciones.

4.6.1 Implementación de los controladores del sistema de articulaciones

Cada uno de los controladores fue implementado siguiendo los diseños detallados en la sección 4.4. Los parámetros usados para cada controlador se determinaron mediante simulaciones y ajustes iterativos para mejorar el desempeño de seguimiento. En la *Tabla 4.4* se muestran dichos parámetros.

Tabla 4.4. Parámetros para los controladores de la plataforma móvil. Elaboración propia

Característica	**Controlador por Gain Scheduling**	**Controlador por Lógica Difusa**	**Controlador de estabilización**
Ecuaciones de control	(4.23)	Ver Figura 4.12	(4.26) y (4.27)
Parámetros	Q, R, c	Ver Figuras 4.6 a 4.11	Kp
Valores de los parámetros	(3.47), (3.48) c=2	Ver Figuras 4.6 a 4.11	Kp=1

En la *Figura 4.25* y la *Figura 4.26* se muestran los valores deseados de cada estado del sistema de articulaciones, así como la respuesta en el tiempo de los estados al controlador basado en Gain Scheduling y al controlador basado en Lógica Difusa, respectivamente.

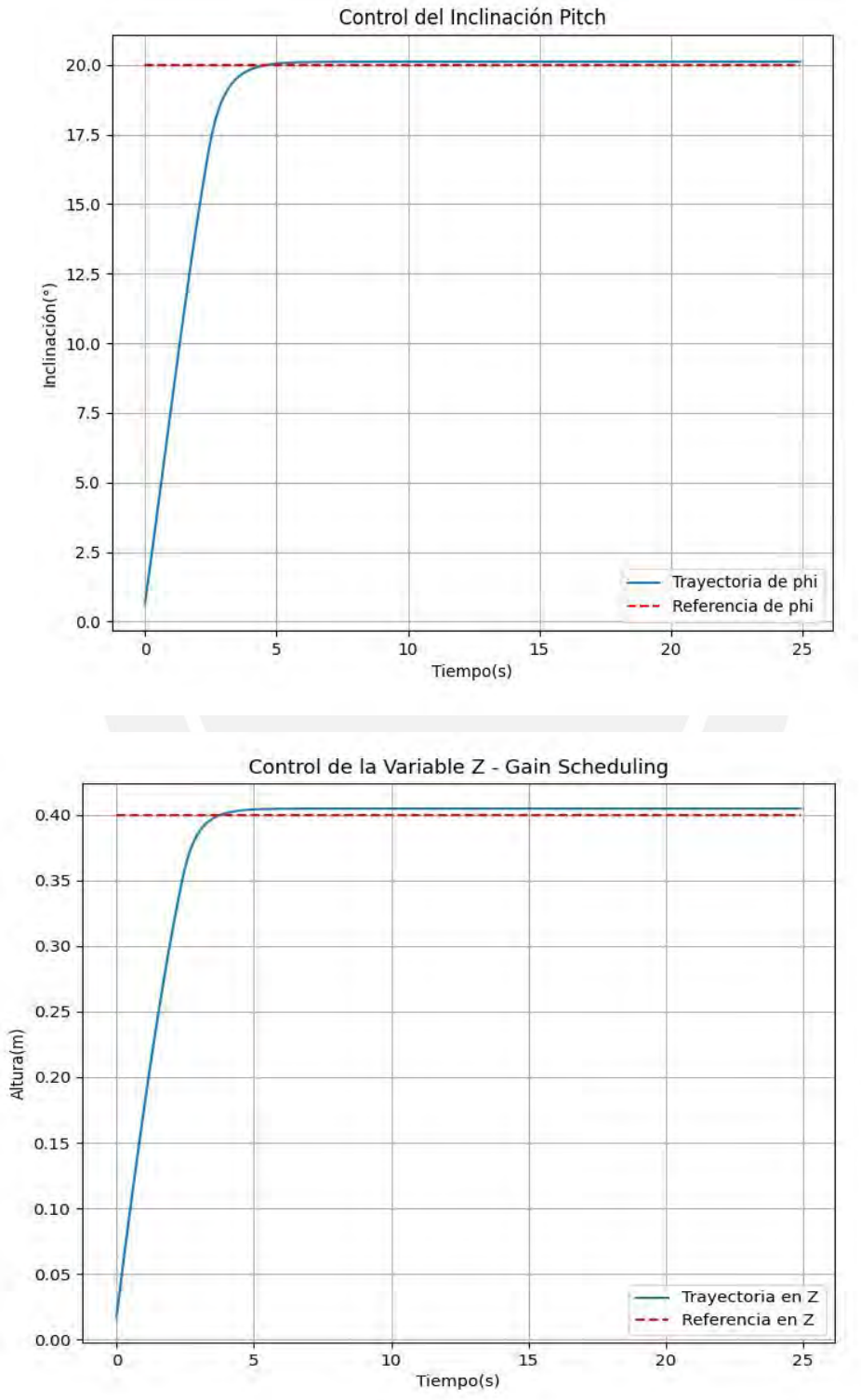

Figura 4.25. Respuesta al Controlador basado en Gain Scheduling. Elaboración propia.

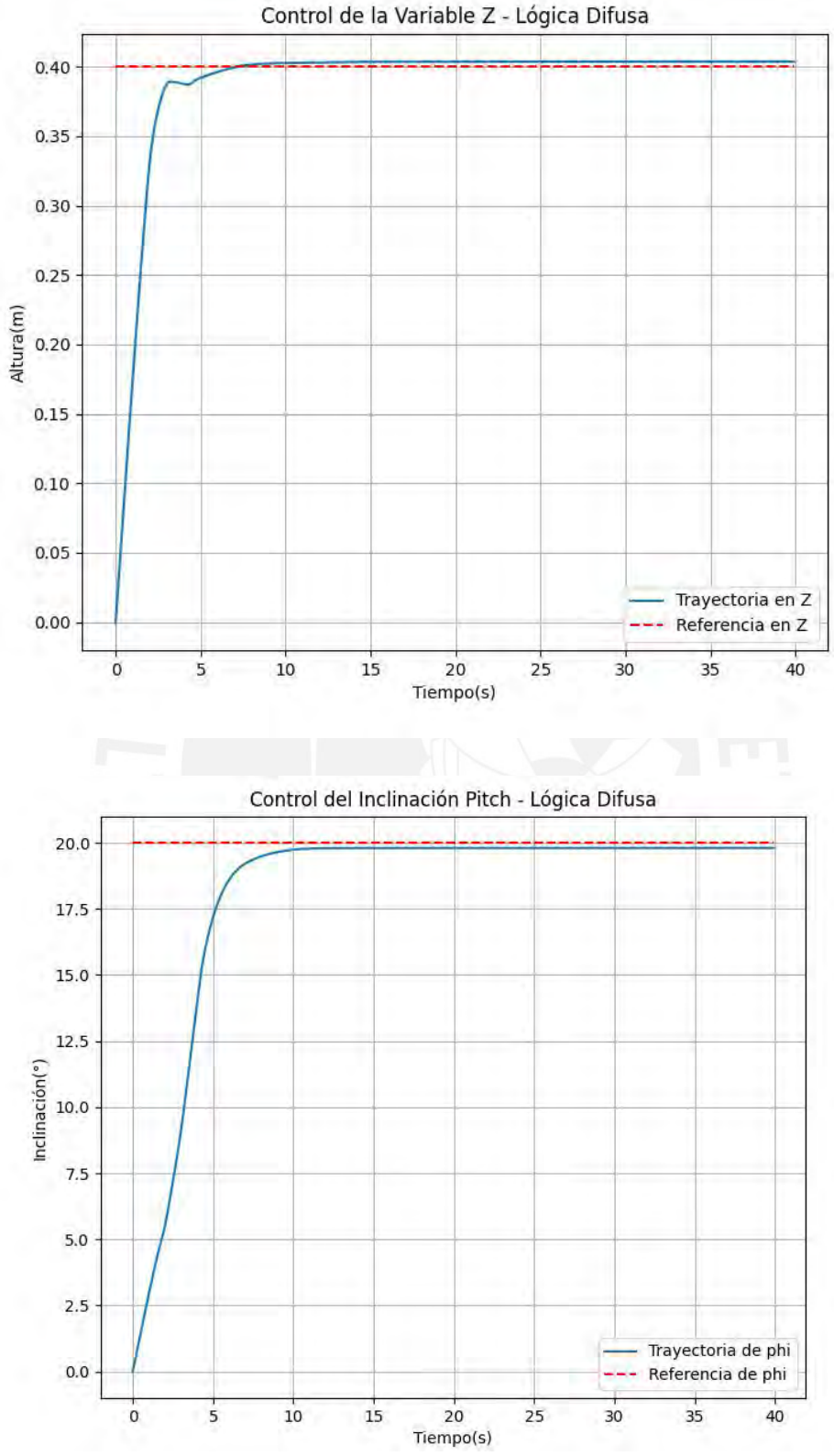

Figura 4.26. Respuesta al Controlador basado en Lógica Difusa. Elaboración propia.

En la *Figura 4.27* y la *Figura 4.28* se muestran las señales de control obtenidas por el controlador basado en Gain Scheduling y las obtenidas por el controlador basado en Lógica Difusa, respectivamente.

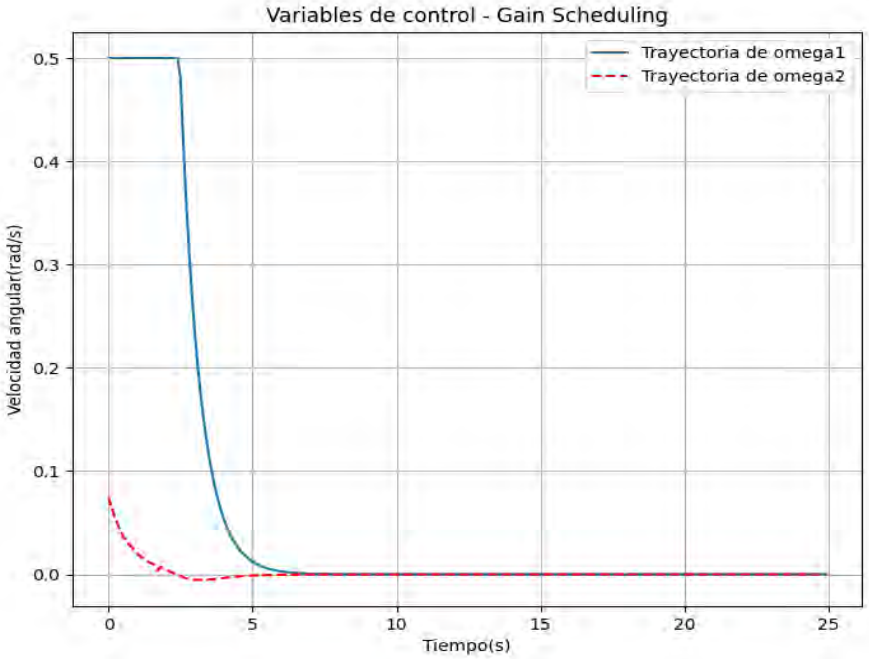

Figura 4.27. Salidas de control del Controlador basado en Gain Scheduling. Elaboración propia.

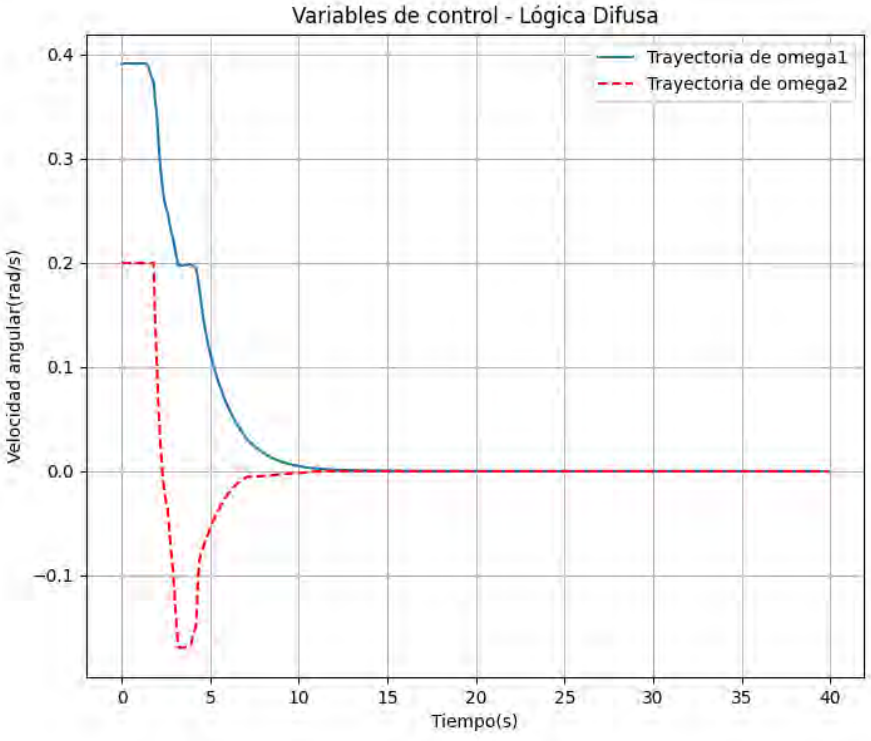

Figura 4.28. Salidas de control del Controlador basado en Lógica Difusa. Elaboración propia.

Al igual que con los controladores de la plataforma móvil, también se simula el comportamiento al ruido en los sensores, añadiendo ruido Gaussiano con media 0 y desviación estándar de 0.02 en la entrada de los controladores del sistema de articulaciones. En la Figura 4.29 y Figura 4.30, se muestran los valores deseados de cada estado del sistema de articulaciones, así como la respuesta en el tiempo de los estados al controlador basado en Gain Scheduling y al controlador basado en Lógica Difusa, respectivamente. Asimismo, en la Figura 4.31 y Figura 4.32 se muestran las señales de control correspondientes a los controladores de Gain Scheduling y Lógica Difusa, respectivamente.

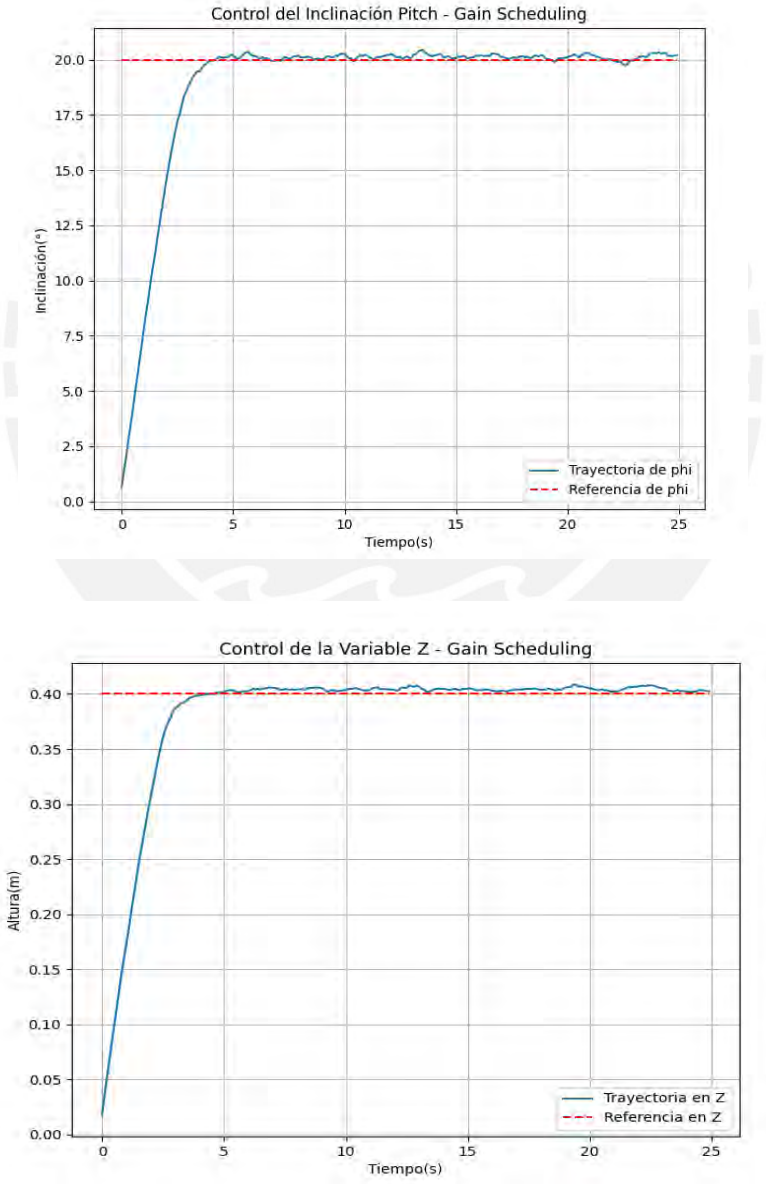

Figura 4.29. Respuesta al Controlador basado en Gain Scheduling afectado por ruido Gaussiano. Elaboración propia.

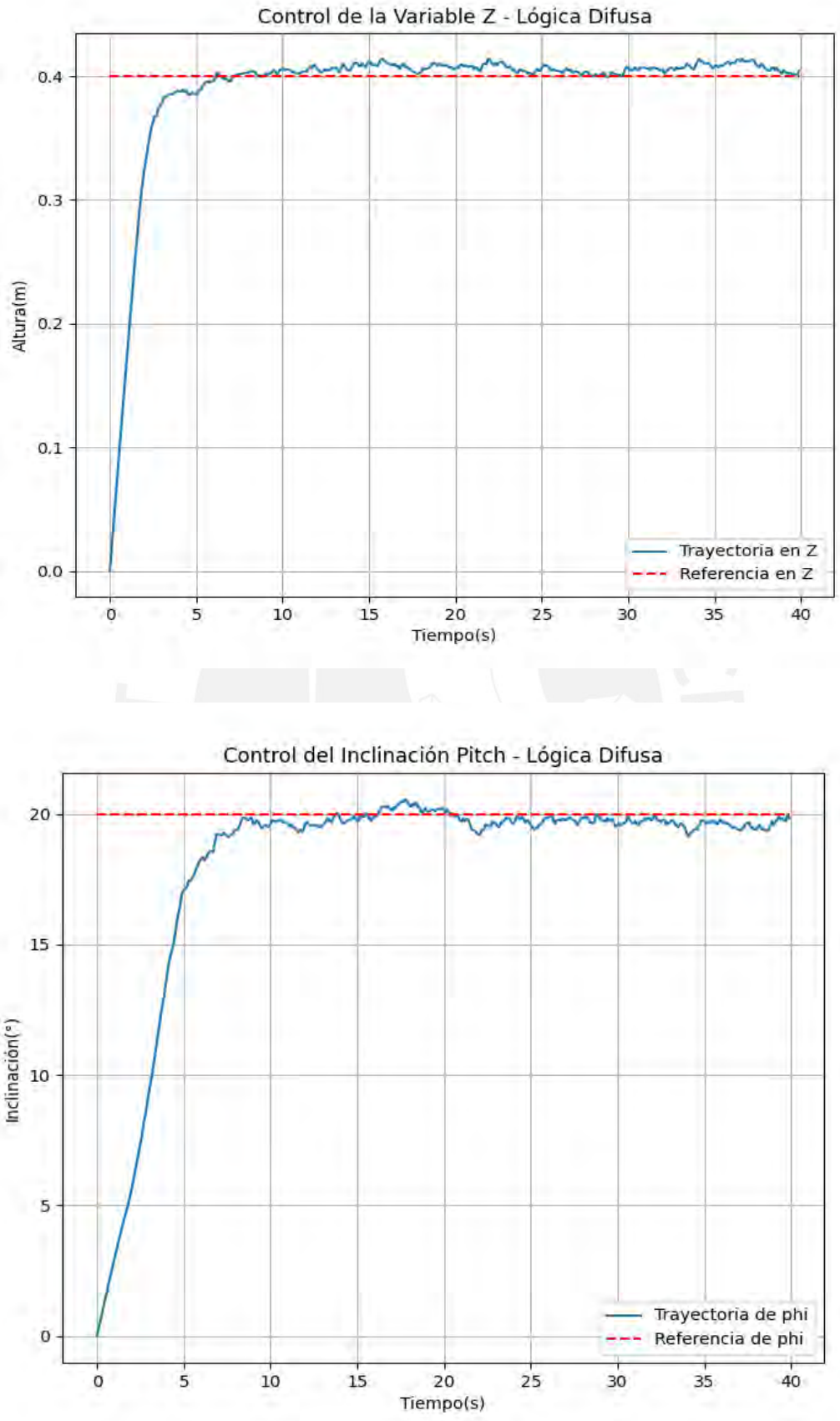

Figura 4.30. Respuesta al Controlador basado en Lógica Difusa afectado por ruido Gaussiano. Elaboración propia.

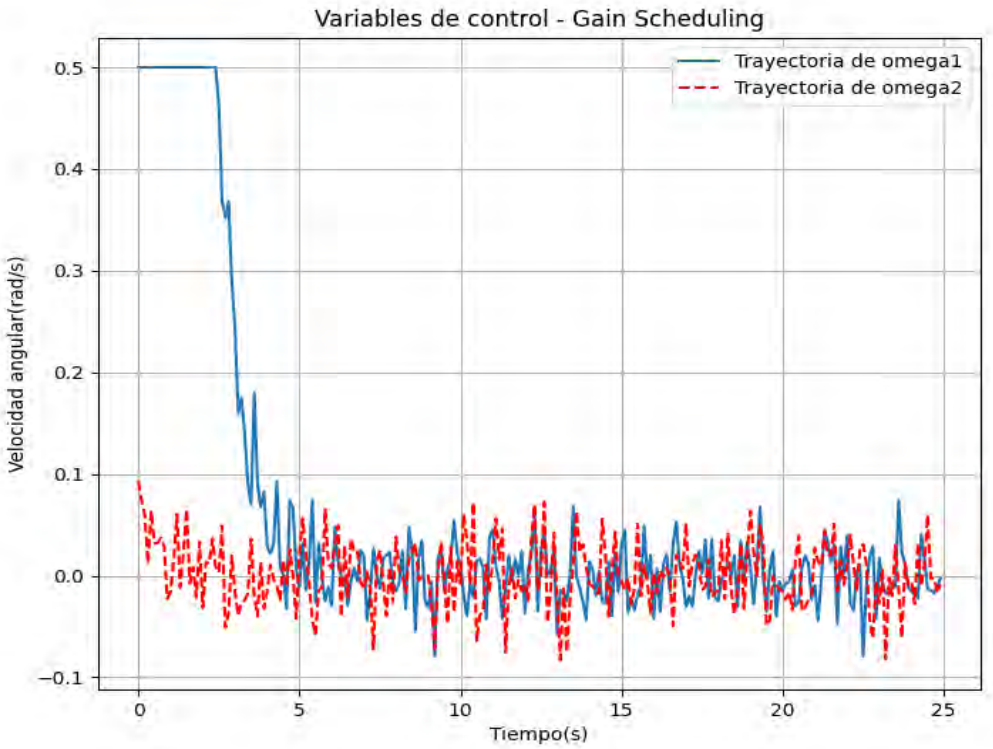

Figura 4.31. Salidas de control del Controlador basado en Gain Scheduling afectado por ruido Gaussiano. Elaboración propia.

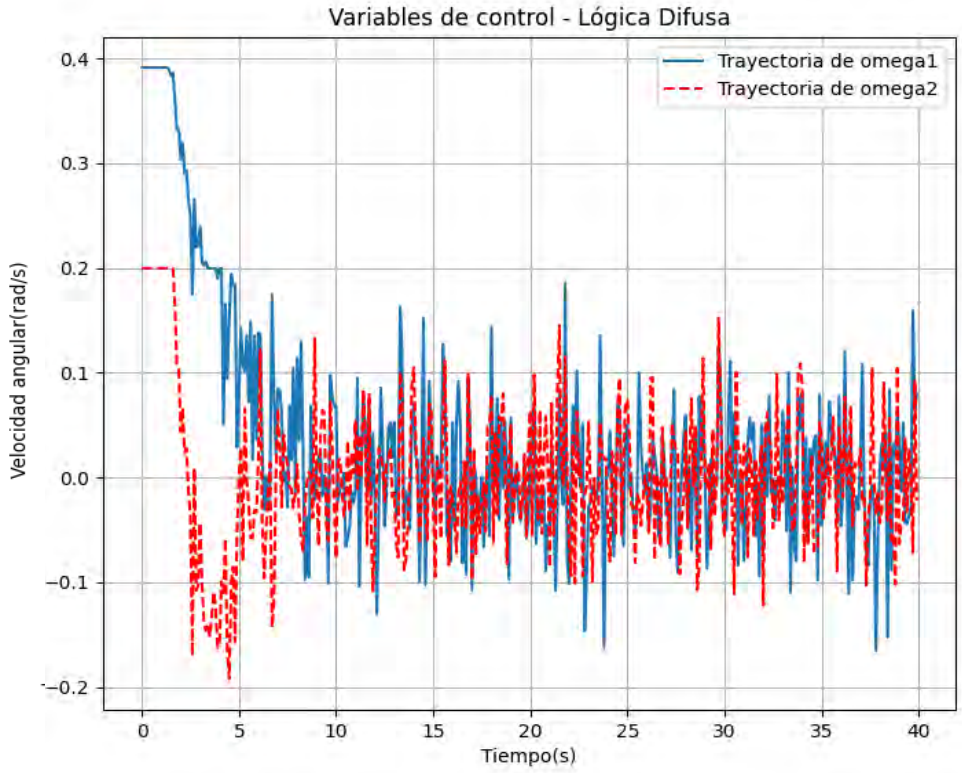

Figura 4.32. Salidas de control del Controlador basado en Lógica Difusa. Elaboración propia.

Por otro lado, en la *Figura 4.33* y *Figura 4.34* se muestra la respuesta del controlador de estabilización de roll ante condiciones iniciales donde el ángulo roll es positivo y negativo respectivamente.

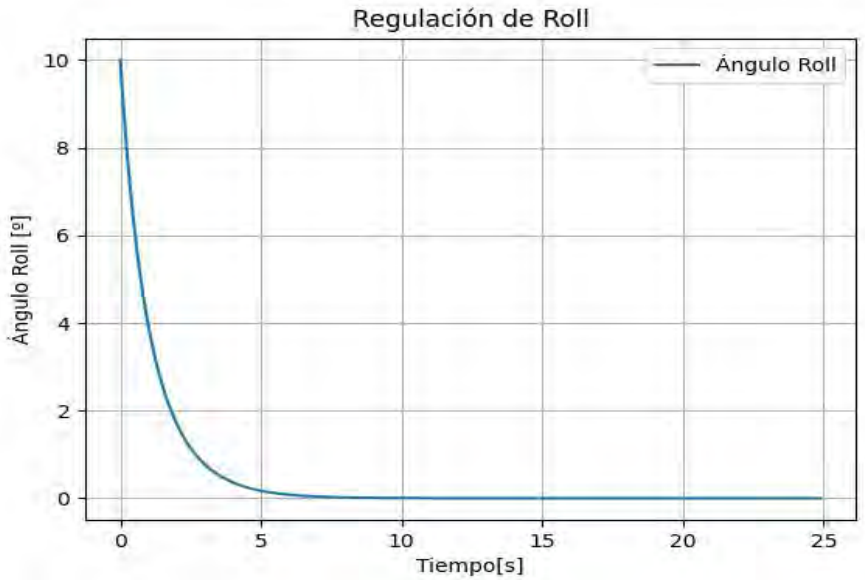

Figura 4.33. Respuesta del controlador de estabilización para valores iniciales positivos. Elaboración propia.

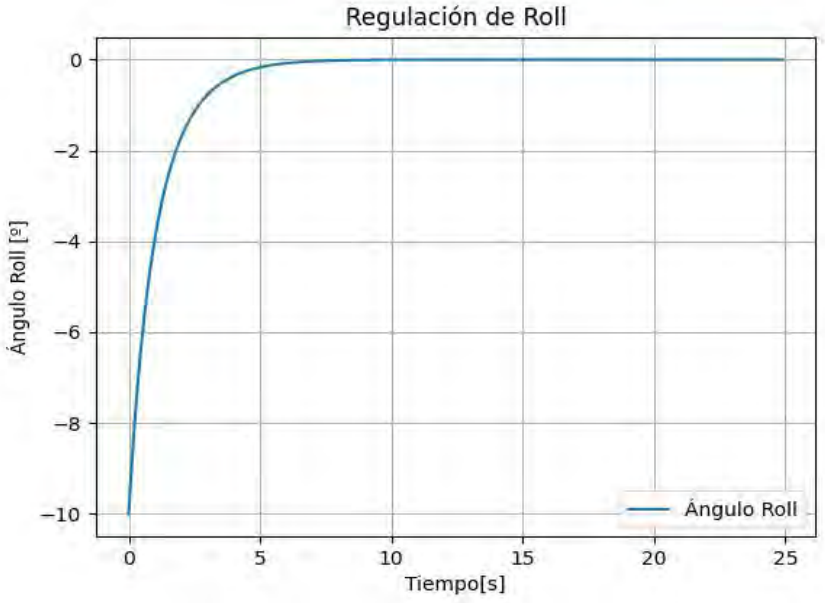

Figura 4.34. Respuesta del controlador de estabilización para valores iniciales negativos. Elaboración propia.

Las señales de control correspondientes se muestran en la *Figura 4.35* y *Figura 4.36,* respectivamente.

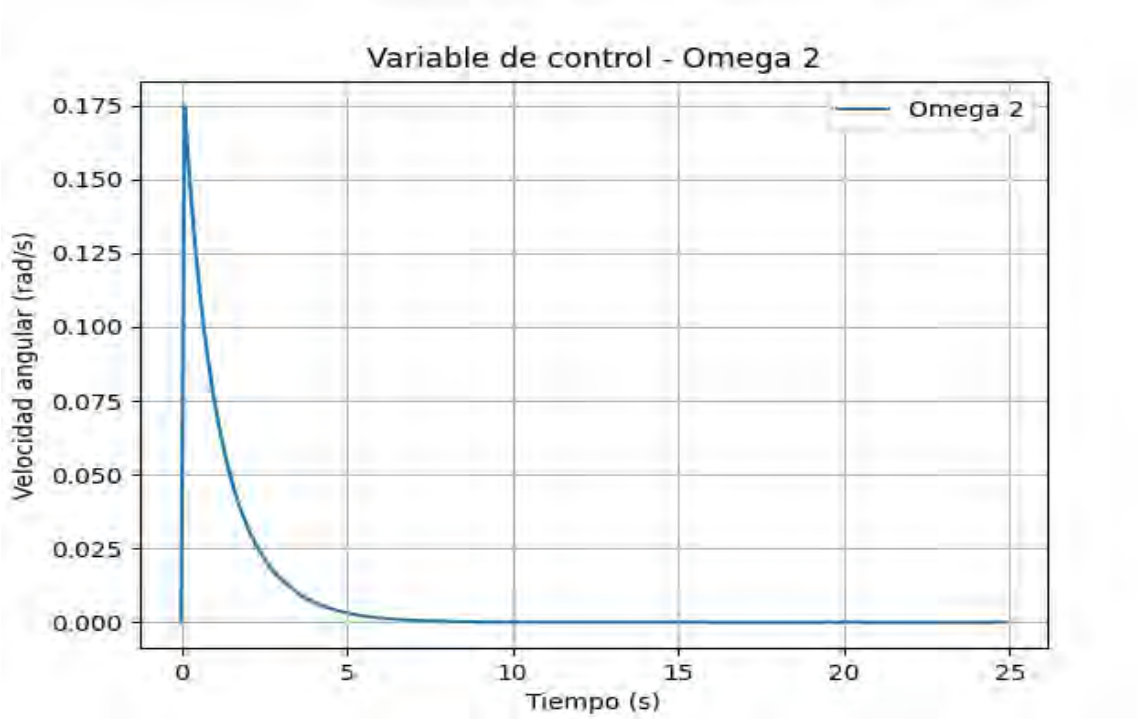

Figura 4.35. Señales de control del controlador de estabilización para valores iniciales positivos. Elaboración propia.

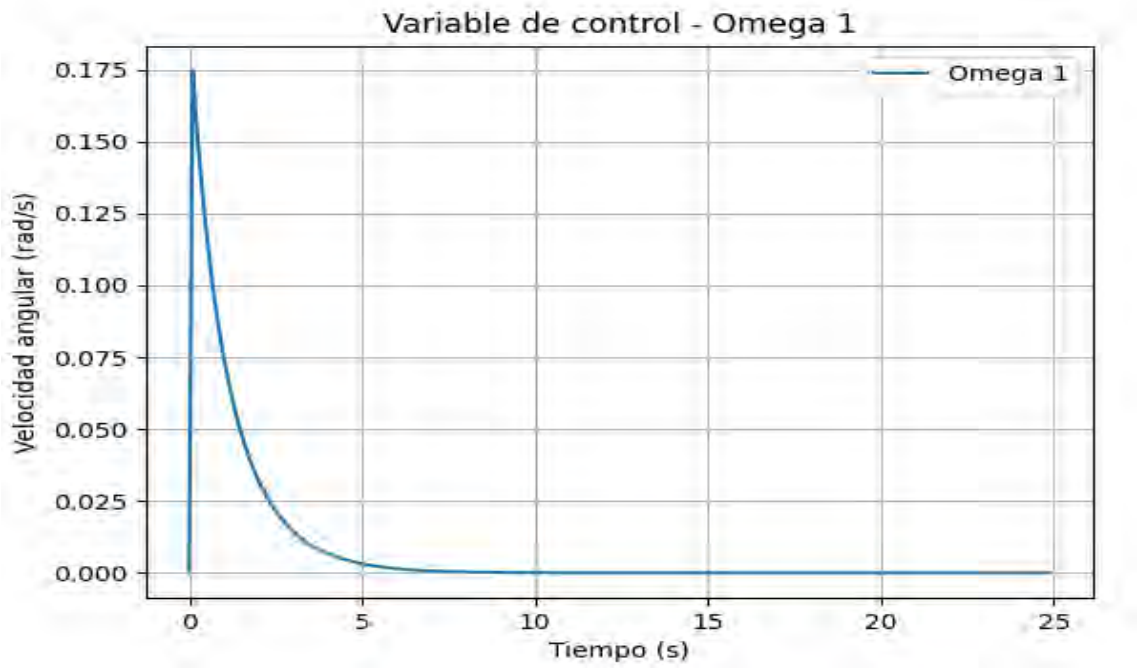

Figura 4.36. Salidas de control del controlador de estabilización para valores iniciales negativos. Elaboración propia.

4.6.2 Comparativa de controladores para sistema de Flippers y resultados

En la *Tabla 4.5* se muestran las distintas métricas utilizadas para comparar los controladores de seguimiento, en la plataforma móvil, ya que a diferencia de los controladores presentes en la sección 4.3, las trayectorias seguidas por los controladores para el sistema de Flippers son valores estáticos en vez de variables, por lo que se opta por evaluar los controladores por máximo sobreimpulso, tiempo de establecimiento y error en estado estacionario en las variables de Z y φ Basado en los resultados de las simulaciones, el controlador diseñado utilizando Gain Scheduling muestra mucho mejor desempeño en términos de tiempo de establecimiento, siendo más del doble de rápido que el otro controlador mientras provee una respuesta sobre amortiguada un error en estado estacionario ligeramente mayor que el obtenido por el controlador de lógica difusa. Por lo tanto, el controlador a implementar en Gazebo para el sistema de Flippers del robot articulado con orugas será el basado en Gain Scheduling.

Tabla 4.5. Parámetros para los controladores de seguimiento para el sistema de Flippers. Elaboración propia.

Parámetro	**Gain Scheduling**	**Lógica Difusa**	**Gain Scheduling + ruido**	**Lógica Difusa + ruido**
Máximo sobreimpulso (Z)	0%	0%	0%	0%
Máximo sobreimpulso (φ)	0%	0%	0%	0%
Tiempo de establecimiento (Z)	2.78 s	2.82 s	2.81 s	3.10 s
Tiempo de establecimiento (φ)	3.072 s	6.95 s	3.08 s	6.92 s
Error en estado estacionario (Z)	1.125%	0.875%	2.45%	4.05%
Error en estado estacionario (φ)	0.550%	1.25%	1.75%	2.8%

Asimismo, de las pruebas con el controlador de estabilización de roll, se obtiene un máximo sobreimpulso de 0%, un tiempo de establecimiento de 4.52 segundos y un error en estado estacionario de 0%.

4.7 Evaluación de estabilidad de los controladores

Una vez elegidos los controladores para la plataforma móvil y el sistema de articulaciones, es necesario determinar la estabilidad de dichos controladores. Se evaluará por separado la estabilidad de cada controlador.

4.7.1 Evaluación de estabilidad del controlador de la plataforma móvil

El controlador seleccionado para la plataforma móvil fue el que utiliza la técnica de Backstepping. En el proceso de diseño de dicho controlador, explicado en la *Sección 4.3.1,* se emplea una función de Lyapunov como parte de la obtención de la ley de control (Huaman & Pérez-Zuñiga, 2019).

En la Ecuación 4.17 se puede ser la derivada de la función de Lyapunov. Debido a que a_1, a_2, a_3, a_4 con constantes mayores a 0 y $\omega \sin(\arctan(\omega)) \geq 0$, se puede ver que $\dot{V} \leq 0$. También se puede observar que *V*, la cual está presente en la Ecuación 4.12 es positiva definida, continuamente diferenciable y acotada. $\dot{V}$ es una función continua negativa semidefinida. De acuerdo con el teorema de Barbalat, bajo las condiciones mostradas previamente, sabemos que cuando *t* converge al infinito, $\dot{V}$ converge a 0. Por lo tanto, $\overline{x_e^2}, y_e^2 \omega \sin(\arctan(\omega)),\ \sin^2\left(\frac{\theta_e}{2}\right)$ también convergen a 0. Basado en el anterior análisis, y aplicando la teoría de estabilidad de Lyapunov, se puede concluir que el error de posición de la plataforma móvil en un sistema de lazo cerrado es globalmente estable.

4.72 Evaluación de estabilidad del controlador del sistema de articulaciones

Al tratarse de un controlador Gain Scheduling, en el cual las ganancias K han sido obtenidas resolviendo la ecuación de Riccati, se garantiza la estabilidad local de esos puntos de operación. Debido a que la interpolación entre ganancias se realiza entre puntos de operación próximos entre sí, se puede considerar que el sistema mantiene su estabilidad.

4.8 Implementación de los controladores en Gazebo.

En esta sección se muestra la implementación en Gazebo de los controladores seleccionados, tanto el controlador basado en Backstepping utilizado en la plataforma móvil, como el controlador basado en Gain Scheduling utilizado en el sistema de articulaciones.

Se mostrará al robot navegando alrededor del ambiente de simulación, utilizando el controlador de Backstepping presentado en la sección 4.3.1. El robot sigue la trayectoria de la ecuación (4.26), la cual fue utilizada para validar los controladores previamente.

La implementación se realiza de manera simular a la sección 3.2.4, la diferencia recae en que en la sección previa se usa el nodo rqt_robot_steering para publicar la velocidad linear y angular del robot de manera manual. En este caso, se cambia este nodo por el controlador, el cual lee los datos de posición y orientación de los tópicos respectivo, y publica automáticamente las velocidad linear y angular al tópico correspondiente. En la *Figura 4.37* se compara la trayectoria de referencia con la trayectoria obtenida en la simulación, la cual se puede observar que es muy similar a la presente en la *Figura 4.17.* Por otro lado, en la *Figura 4.38* se puede ver una secuencia se imágenes del robot siguiendo la trayectoria en Gazebo.

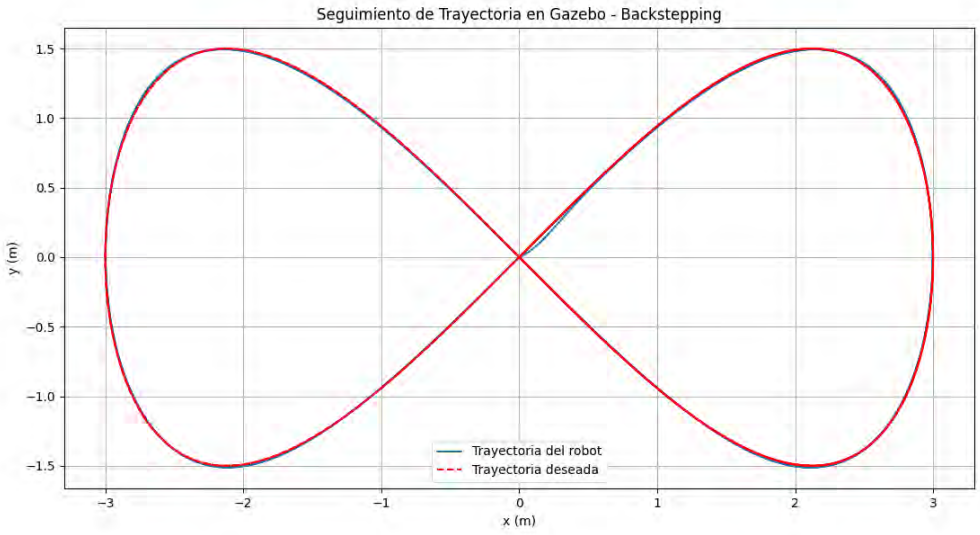

Figura 4.37. Trayectoria obtenida Backstepping dentro de Gazebo. Elaboración propia

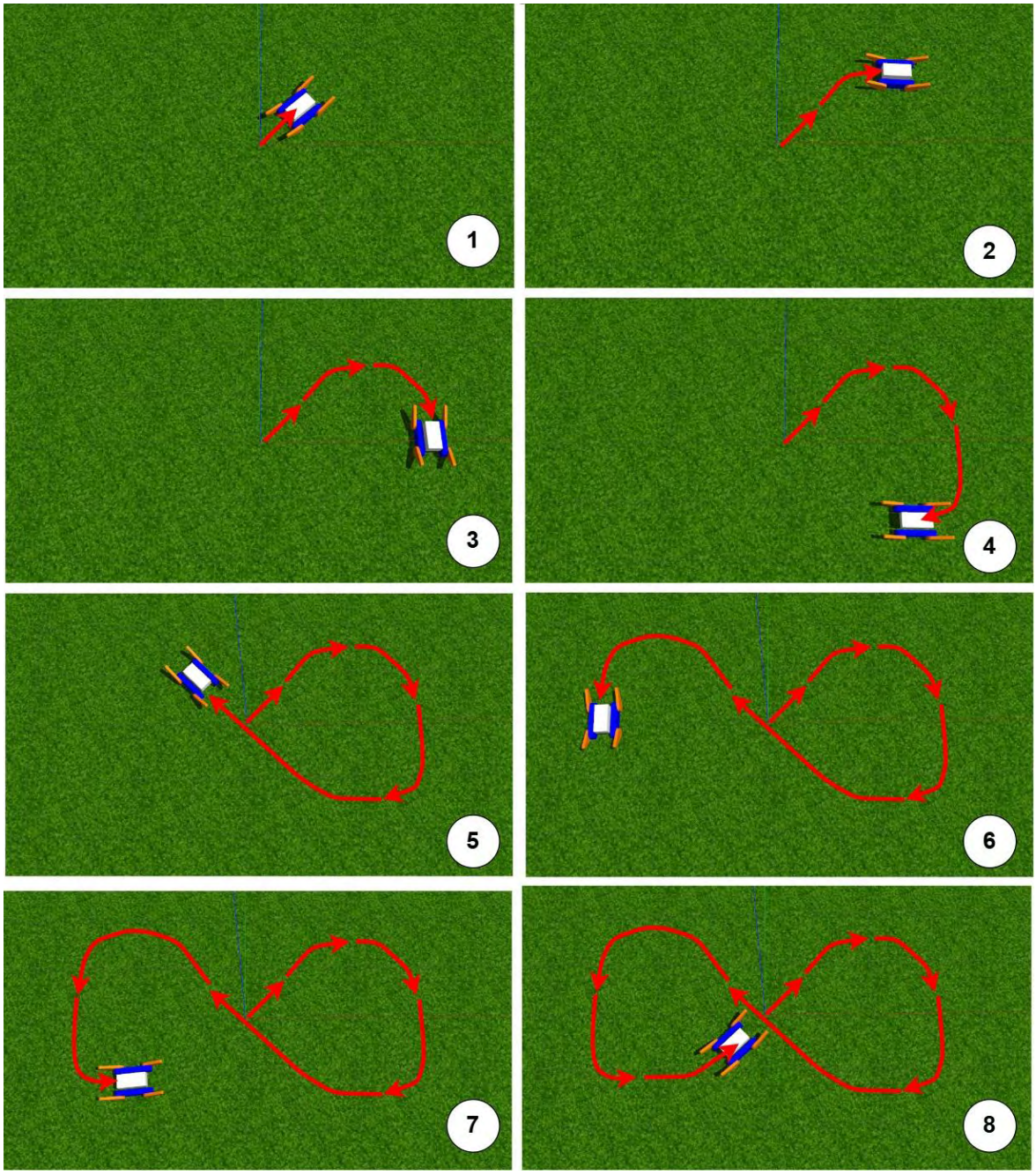

Figura 4.38. Prueba de seguimiento de trayectoria. Elaboración propia

La implementación del controlador ha demostrado ser una tarea fundamental al momento de garantizar el desempeño óptimo del robot articulado con orugas. A través de técnicas de control no lineal, se logró diseñar 2 controladores distintos. De estos dos controladores se seleccionó el controlador de Backstepping ya que este mostró el mejor desempeño en términos de precisión y tiempo de respuesta, con un MSE de 0.0370 m en el eje x y 0.0376 m en el eje y, y un tiempo de respuesta 1.35 segundos. Estos resultados confirman la precisión y confiabilidad

del modelo implementado. Luego, se implementó el controlador en la simulación de Gazebo, obteniendo resultados similares. El próximo capítulo abordará la propuesta de implementación de los controladores en hardware.

CAPÍTULO 5: PROPUESTA DE IMPLEMENTACIÓN

5.1 Introducción

En el presente capítulo es buscar detallar la propuesta de implementación de los algoritmos de control desarrollado en los anteriores capítulos. Se considerará que el diseño mecánico y electrónico se ha desarrollado de manera previa y solo detallará la implementación de los sensores y unidades de procesamiento.

5.2 Diagrama general del sistema

En la *Figura 5.1* se muestra el robot articulado con orugas en el cual se busca implementar los algoritmos de control. Teniendo en cuenta el estado del arte de la instrumentación de robots similares, visto en el capítulo 1, el robot debe contar con un microcontrolador para enviar señales a los actuadores del robot, una computadora embebida para procesar la información proveniente de los sensores. Los sensores considerados para este robot deben ser: un (01) Sistema de Referencia de Actitud y Rumbo (AHRS), dos (02) sensores de distancia laser o LiDAR y seis (06) encoders ubicados en cada uno de los motores.

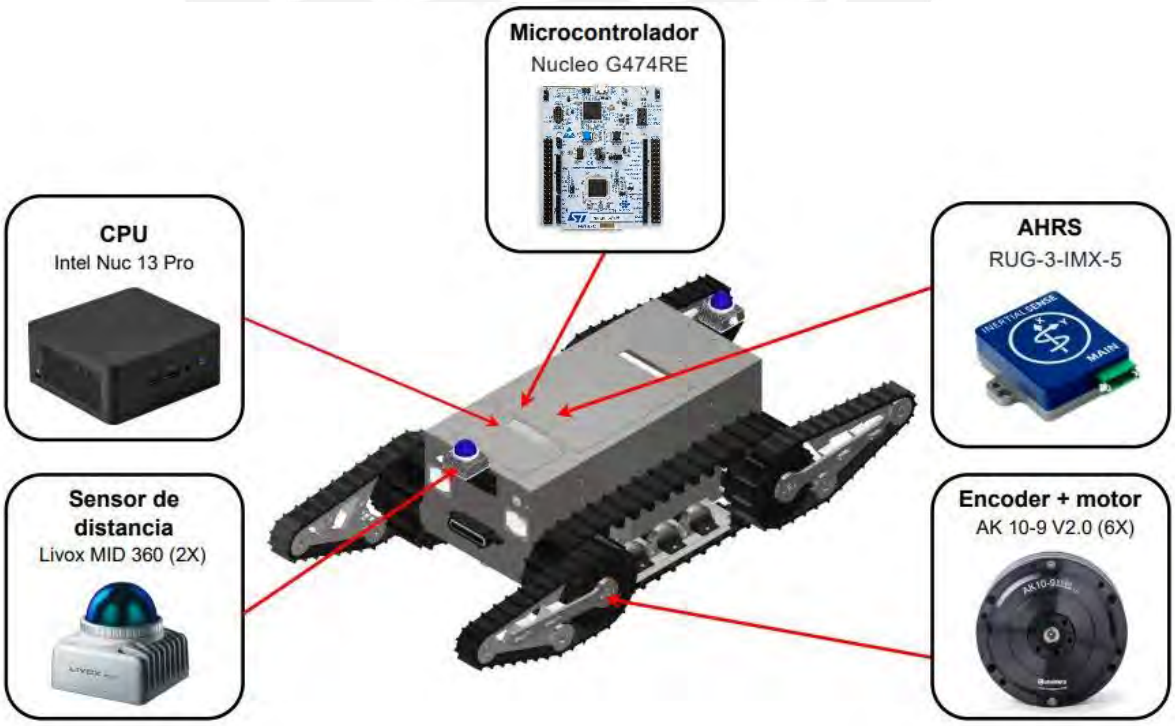

Figura 5.1. Robot articulado con orugas propuesto. Elaboración propia

5.3 Selección de sensores y unidades de procesamiento

De acuerdo con el diagrama general del sistema, se procede a seleccionar los componentes detallados en la *Figura 5.1* con excepción de los encoders, debido a que estos se encuentran incorporados con los motores previamente seleccionados. Al momento de seleccionar dichos componentes se tendrá en cuenta tanto las características técnicas como su precio. Se considerará un presupuesto de aproximadamente 4000 Dólares.

5.3.1 Sistema de Referencia de Actitud y Rumbo

Un Sistema de Referencia de Actitud y Rumbo (AHRS) es un módulo que consiste en una unidad de medición inercial (IMU), un magnetómetro y un barómetro. La información obtenida por este sensor incluye la velocidad angular, aceleración lineal, vector magnético, presión barométrica y altitud. Esta información es procesada dentro de este módulo con el fin de estimar los ángulos roll, pitch y yaw, variables que son necesarias para implementar los algoritmos de control desarrollados previamente. Debido a esto, las características consideradas para la selección de este componente son las relacionadas con la precisión de estimación de estos ángulos, así como el protocolo de comunicación utilizado, dimensiones, peso y el precio del módulo. En la *Tabla 5.1* se muestra la comparativa entre distintos AHRSs.

Tabla 5.1. Comparativa de AHRS. Elaboración propia

Característica	**RUG-3-IMX-5**	**MTi-630**	**3DM-CV5-AHRS**
Foto referencial			
Marca	Inertial Sense	Movella (Xsens)	Parker Hannifin
Dimensiones (mm)	30.25 x 25.4 x 9.9	38 x 24 x 9.7	28x31.5x13
Peso (g)	10.5	11	8.9
Precisión de Roll/Pitch	0.04°	0.5°	0.2°
Precisión de Yaw	0.13°	0.2°	0.1°
Estabilidad de Sesgo (°/h)	1.5	8	8
Frecuencia máxima de Salida de datos (KHz)	1	0.5	2
Protocolos de comunicación	USB, CAN, SPI, RS232, RS485.	USB, TTL	TTL, CAN, RS232
Precio (USD)	800	1070	910

El módulo elegido es el RUG-3-IMX-5, debido a que cuenta con una mejor precisión en términos de roll y pitch, mientras que cuenta con una precisión de yaw comparable a los demás AHRS. Además, tiene una mayor estabilidad de sesgo, lo cual implica que el dispositivo va a desviarse en menor medida a lo largo del tiempo. Si bien la frecuencia máxima de salida de datos no es la mayor, el modelo del robot articulado con orugas no cuenta con una dinámica particularmente rápida, por lo que la velocidad del AHRS escogido es suficiente.

5.3.2 Sensor de distancia

El sensor de distancia escogido para esta aplicación es un LiDAR 3D, debido a que estos son ampliamente utilizados para la aplicación de algoritmos de Mapeo y Localización Simultánea (SLAM), los cuales permiten la generación del mapa donde se encuentra el robot, así como su ubicación espacial dentro del mismo, lo cual nos permitirá obtener las coordenadas tridimensionales del robot con respecto a su ambiente, información que es usada en los algoritmos descritos en capítulos anteriores. Las características consideradas para la selección del componente serán las relacionadas al campo de visión, la densidad y precisión de los puntos, así como el precio del componente. Es necesario aclarar que se seleccionará un LiDAR 3D de mediana o baja gama, debido a que los LIDAR de alta gama, en la mayoría de los casos, tienen precio mayor a 4000 USD. En la *Tabla 5.2* se muestra la comparativa entre 3 distintos LiDARs 3D, Livox MID-360, Unitree 4D LiDAR L1 y C16 LiDAR.

El módulo elegido es el LIVOX MID-360, a su relación calidad-precio, destacando por su alta densidad de puntos en comparación a su costo, siendo esta la principal razón de la elección, puesto que, en la mayoría del resto de los ítems, los 3 LiDARs cuentan con características similares. Además, el Livox MID-360, cuenta con un campo de visión suficientemente amplio como para poder mapear el ambiente de manera eficiente. En cuanto al rango máximo de detección, al ser una aplicación de ambientes cerrados, los tres cuentan con un rango máximo más que adecuado para realizar el mapeo sin inconvenientes. Por otro lado, el rango mínimo y precisión es bastante equiparable entre los competidores.

Tabla 5.2. Comparativa de LiDARs 3D. Elaboración propia

Característica	**Livox MID-360**	**Unitree 4D LiDAR L1**	**C16 LiDAR**
Foto referencial			
Marca	Livox	Unitree	Leishen
Dimensiones (mm)	65x65x60	75x75x65	105x105x81
Peso (g)	265	230	1050
Rango máximo (m)	40	20	70
Rango mínimo (m)	0.1	0.05	0.1
Precisión (cm)	2	2	3
Campo de visión horizontal (°)	360	360	360
Campo de visión vertical (°)	60	90	30
Densidad de puntos (puntos/s)	200000	21600	320000
Frecuencia de muestreo (Hz)	10	11	10
Protocolos de comunicación	Ethernet	TTL	Ethernet
Precio (USD)	900	550	1900

5.3.3 Microcontrolador

El microcontrolador en esta aplicación de control sirve como interfaz entre los motores y la computadora embebida. Por un lado, la computadora embebida es la encargada de procesar los algoritmos de control en base a la data de los sensores y obtener las velocidades requeridas en cada motor. Por otro lado, los motores presentes en este robot, los AK10-9 V2.0 KV60, cuentan con un encoder y controlador integrado, los cuales son usados para regular la velocidad de acuerdo con los comandos recibidos y también brinda la velocidad instantánea del motor. Debido al protocolo de comunicación utilizado por el motor, estos no se pueden conectar directamente a la computadora embebida, por lo que es necesario el uso de un microcontrolador.

Si bien el microcontrolador puede realizar otras funciones dentro del robot como control de luces, activación de relés, entre otras funciones; en esta comparativa se considerará solo las características relacionadas al envío y recepción de datos del motor, así como su comunicación con la computadora embebida. Con el fin

de entender de mejor manera los requerimientos del microcontrolador, en la *Tabla 5.3* se muestran las características relevantes del motor.

Tabla 5.3. Características de los motores utilizado. Elaboración propia

Característica	**Descripción**
Modelo	AK10-9 V2.0 KV60
Foto referencial	
Tipo de controlador	PID
Protocolo de Comunicación	CAN, 1Mbit/s
Tipo de encoder	Magnético
Resolución de encoder	14 bit

Considerando las características del motor, es necesario un microcontrolador que cuente con un periférico CAN y que a su vez cuente con un puerto USB para la comunicación con la computadora embebida. En la selección también se considerará la memoria SRAM, así como la frecuencia del reloj interno, características necesarias para poder procesar de mejor manera los datos obtenidos por los seis (06) motores en simultáneo. En la *Tabla 5.4* se muestra la comparativa entre 3 distintos microcontroladores, Arduino Uno R3, Nucleo G474RE y EK-TM4C123GXL.

El microcontrolador elegido es el Nucleo G474RE, debido a que esta muestra características superiores en tanto memoria flash, SRAM y frecuencia de su reloj, además de ser la opción más barata. La diferencia entre el número de pines no es suficientemente amplia como para poder ser considerada relevante.

Tabla 5.4. Comparativa de microcontroladores. Elaboración propia

Característica	Arduino Uno R3	Nucleo G474RE	EK-TM4C123GXL
Foto referencial			
Marca	Arduino	STM	Texas Instruments
Dimensiones (mm)	80x55x20	85x70x20	50x60x20
Peso (g)	37	43	54
Interfaces disponibles	USB, I2C, SPI, UART, CAN, PWM	USB, I2C, SPI, UART, CAN, PWM	USB, I2C, SPI, UART, CAN, PWM
Pines digitales	14	52	58
Pines analógicos	13	12	14
Memoria Flash (KB)	32	512	256
Memoria SRAM (KB)	2	128	32
Frecuencia de reloj (MHz)	16	170	80
Precio (USD)	27	16	22.6

5.3.4 Computadora embebida

La mayoría de los robots articulados con orugas utilizados en aplicaciones similares utilizan dos computadoras embebidas: una computadora destinada a la localización, mapeo, control y navegación; y otra computadora encargada de detección de objetos y transmisión de video. Con fines de este documento, se seleccionará la primera computadora, puesto a que esta es en la que se ejecutarán los algoritmos de control previamente diseñados, así como se procesará la información de los sensores seleccionados. Debido a que estas tareas utilizan de manera intensiva la CPU, mientras que dejan de lado la GPU, los criterios de selección para la computadora embebida estarán centrados en el rendimiento de la CPU (Rouček, y otros, 2021). En la *Tabla 5.5* se muestra la comparativa entre 3 distintas computadoras embebidas, Intel NUC 13 Pro, Jetson AGX Orin y LattePanda Sigma.

La computadora embebida elegida es la Intel NUC 13 Pro, debido a que esta cuenta con un mejor rendimiento del CPU, contando con un procesador de hasta

5.0 GHz en los núcleos, de los cuales cuenta con 12 en total; lo cual le da un mayor rendimiento en aplicaciones de uso intensivo de CPU, como el procesamiento de las nubes de puntos obtenidas por el LiDAR. Además, cuenta con la mayor capacidad de RAM entre las computadoras comparadas, la cual es requerida para la ejecución eficientes de los procesos relacionados con los algoritmos de control.

Tabla 5.5. Comparativa de computadoras embebidas. Elaboración propia

Característica	Intel® NUC 13 Pro	NVIDIA Jetson AGX Orin	LattePanda Sigma
Foto referencial			
Marca	Intel	Nvidia	LattePanda
Dimensiones (mm)	117x112x37	110x110x72	146x102x24
Peso (g)	538	1580	438
CPU	Intel® 13ª Gen Core™ i7-1360P	Cortex®-A78AE v8.2	Intel® Core™ i5-1340P
Número de núcleos	12	12	12
Frecuencia de operación (GHz)	5	2.2	4.6
RAM (GB)	64	64	32
Almacenamiento (GB)	500	64	500
Puertos USB	3	4	4
Sistema operativo	Windows 11, Ubuntu	Ubuntu	Windows 11, Ubuntu
Precio (USD)	900	2000	650

5.3.5 Presupuesto del hardware

En la *Tabla 5.6,* se el presupuesto de los componentes utilizados en la propuesta de implementación del sistema de control. Se muestra una recopilación de los componentes seleccionados, su precio individual, así como el precio total de todos los componentes.

Tabla 5.6. Presupuesto de la propuesta de implementación. Elaboración propia

Componente	Precio Unitario (USD)	Cantidad	Precio total (USD)
RUG-3-IMX-5	800	1	800
Livox MID 360	900	2	1800
Núcleo G474RE	16	1	16
Intel NUC 13 Pro	900	1	900
Total	-	-	3516

5.4 Instalación y configuración de hardware y software

En la presente sección se detallará las conexiones entre componentes, su instalación en el sistema, así como la configuración necesaria del software para el correcto funcionamiento de los componentes utilizados en los algoritmos de control del robot articulado con orugas.

5.4.1 Conexiones del sistema

En la *Figura 5.2* se muestra un diagrama de conexiones que especifica cómo se deben conectar los componentes seleccionados, así como sus protocolos de comunicación utilizados en su comunicación.

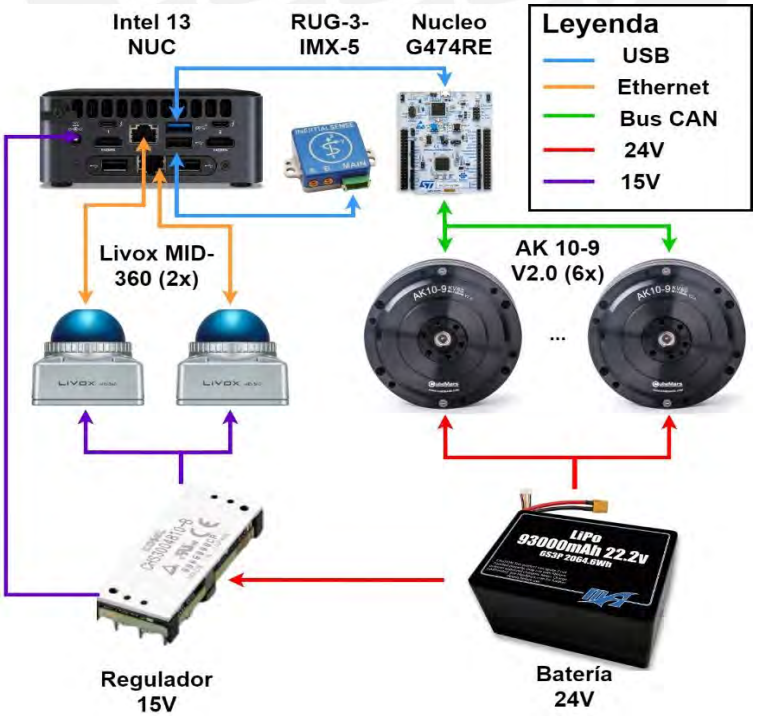

Figura 5.2 Diagrama de conexiones del robot. Elaboración propia

Por otro lado, en la *Figura 5.3* se presenta el esquemático de las conexiones entre el microcontrolador y el motor. Es necesario resaltar que, aunque la Nucleo G474RE cuenta con periféricos CAN, es necesario añadir un transceptor CAN, así como la circuitería requerida para su correcto funcionamiento, la cual también se incluye en el esquemático.

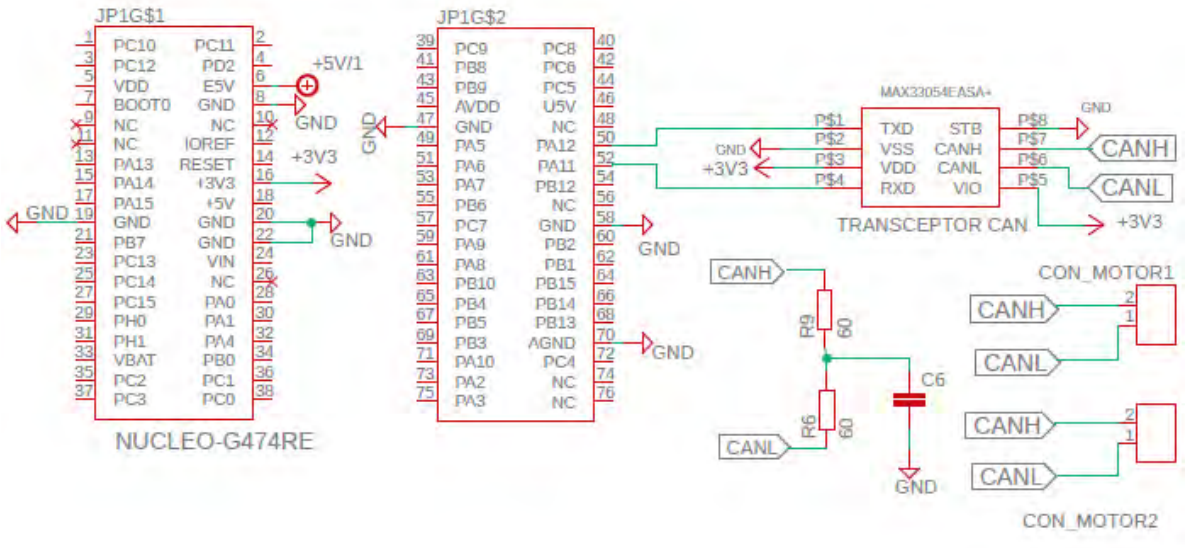

Figura 5.3. Esquemático de conexión microcontrolador a motor. Elaboración propia

Teniendo en cuenta en la *Figura 5.2* y la *Figura 5.3*, en la *Figura 5.4* se muestran las conexiones físicas de entre los componentes, sin contar las conexiones a la alimentación de voltaje.

Figura 5.4. Instalación de hardware del robot. Elaboración propia

5.4.2 Instalación y configuración del AHRS

Se utiliza el software EvalTool proveniente del fabricante del AHRS, Inertial Sense para configurar su envío de datos. El software se obtiene de manera gratuita en la página web de Inertial Sense.

Una vez se instala el software y en ejecuta, se debe conectar con el AHRS, esto se realiza en la pestaña “Settings” de EvalTool, abriendo el puerto de comunicación correspondiente. Este procedimiento se muestra en la *Figura 5.5*.

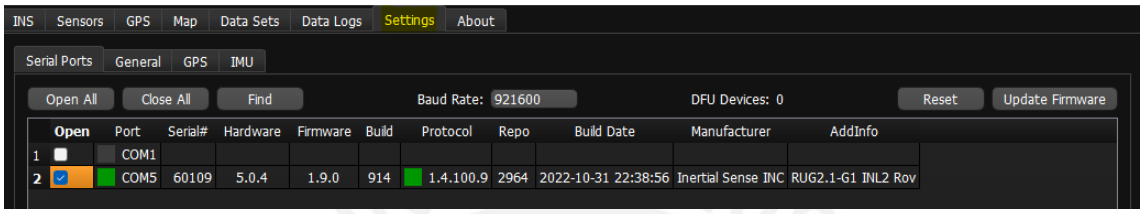

Figura 5.5 Conexión con el AHRS mediante EvalTool. Elaboración propia

Luego, se ingresa a la pestaña “Data Logs”, en la cual se marca el recuadro correspondiente a “DID_INS_1”. Finalmente, se guarda la configuración, de esta manera se habilita al AHRS a que constantemente envié solo los datos de roll, pitch y yaw. En la *Figura 5.6* se muestra la configuración realizada.

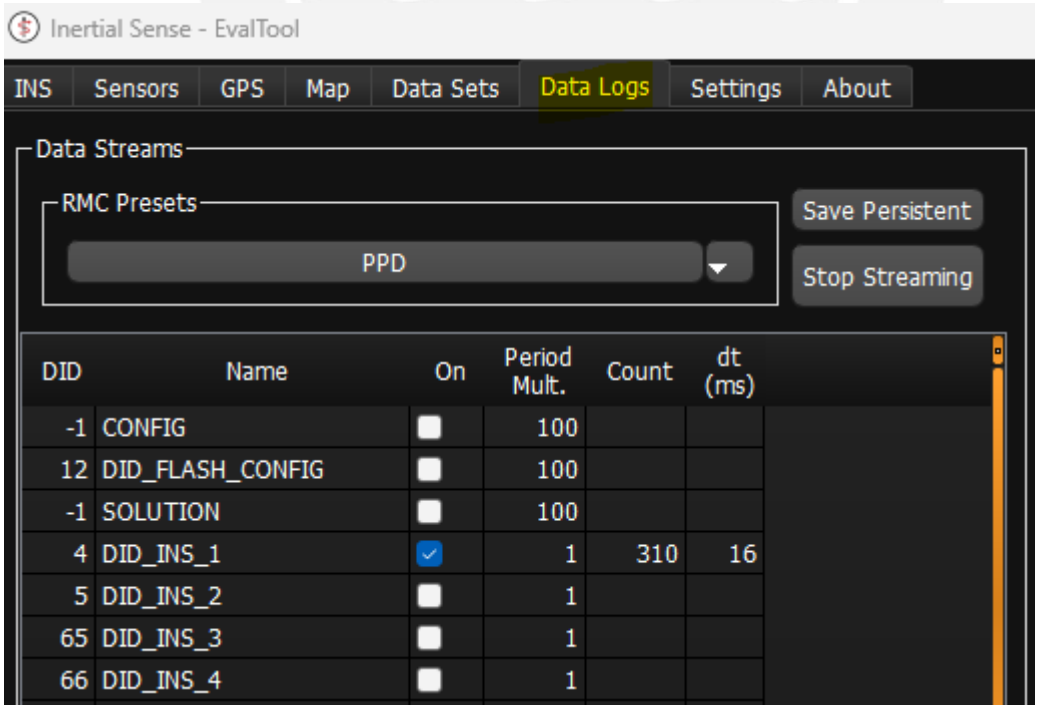

Figura 5.6. Configuración de envío de datos del AHRS. Elaboración propia

5.4.3 Instalación y configuración de Sensor de distancia

El LiDAR Livox MID-360 tiene como requerimiento para su funcionamiento la instalación de los siguientes paquetes: Livox SDK2 y Livox ROS Driver 2, los cuales pueden ser descargados desde el repositorio en GitHub de Livox (Livox, 2024). Los requerimientos para la instalación de ambos paquetes se muestran en la *Tabla 5.7*.

Tabla 5.7. Requerimientos de software para uso de Livox MID-360. (Livox, 2024)

Característica	**Requisitos**
Sistema Operativo	Linux: Ubuntu 18.04 o mayor Windows 10/11
Herramientas	Compiladores que soporten C++ 11 Cmake 3.0+
Arquitectura	x86 ARM

Luego de instalar los paquetes siguiendo las instrucciones encontradas en el repositorio de Livox, el paso final es editar el archivo de configuración "**MID360_config. json**" encontrado en la carpeta "config" del paquete Livox ROS Driver 2. En este archivo se deben cambiar los valores descritos en la *Tabla 5.8.*

Tabla 5.8. Configuración de Livox MID-360. Elaboración propia

Parámetro	**Descripción**	**Valor original**	**Valor nuevo (ejemplo)**
imu_data_ip	Estos parámetros indican la dirección IP de destino de los datos del LiDAR, se recomienda que tengan el mismo valor, y este sea la dirección IP de la computadora embebida.	192.168.1.5	192.168.1.38
push_msg_ip		192.168.1.5	192.168.1.38
point_data_ip		192.168.1.5	192.168.1.38
imu_data_ip		192.168.1.5	192.168.1.38
ip	Dirección IP del LiDAR, este se encuentra indicado en un sticker, también se puede cambiar utilizando el software "Livox Viewer 2"	192.168.1.12	192.168.1.151

5.4.4 Instalación y configuración de Microcontrolador

La configuración del microcontrolador se realiza utilizando el software STM32 Cube IDE, el cual es descargado desde la página web del fabricante. El primer paso de la configuración es crear un nuevo proyecto y seleccionar la tarjeta Nucleo G474RE en la pestaña “Board Selector”, como se muestra en la *Figura 5.7.*

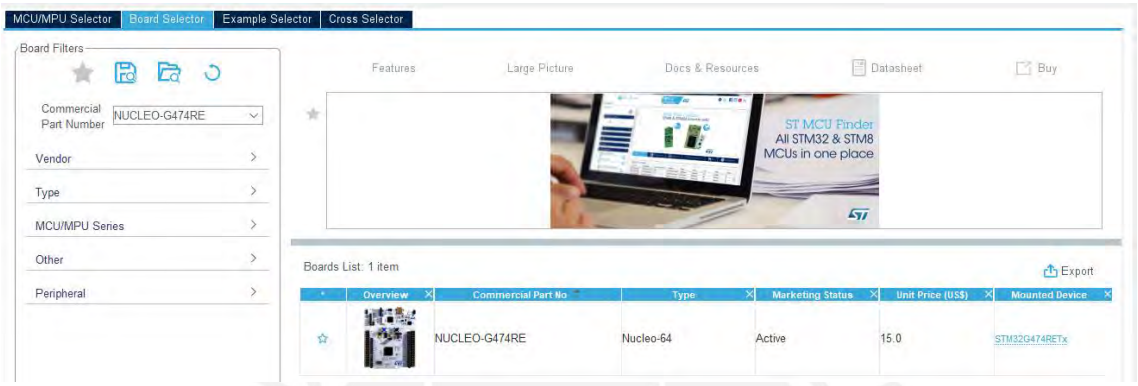

Figura 5.7. Elección de microcontrolador en STM32 Cube IDE. Elaboración propia

Una vez creado el proyecto, se configuran los pines de tal manera que coincidan con el esquemático de la Figura 5.3. Esta configuración se puede ver en la *Figura 5.8.*

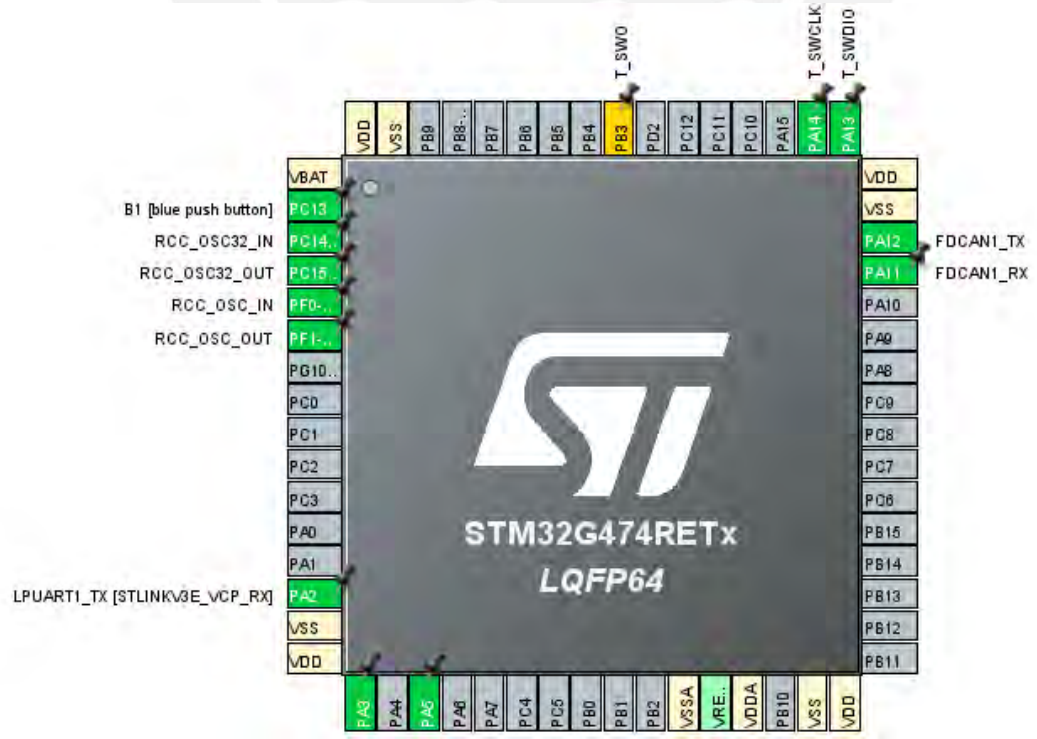

Figura 5.8. Configuración de pines del microcontrolador. Elaboración propia

Luego, en la *Figura 5.9* se configura la frecuencia del reloj del sistema de tal manera que el reloj del periférico CAN tenga una velocidad de 80 MHz, esta configuración facilitará que la frecuencia del bus CAN sea 1 Mbit/s.

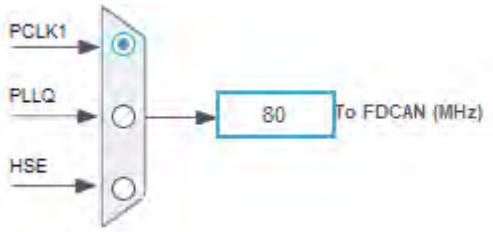

Figura 5.9. Configuración de reloj del periférico CAN. Elaboración propia

Finalmente se configura el periférico CAN con los valores mostrados en la *Figura 5.10.* De esta manera, el microcontrolador puede comunicarse de manera bidireccional con todos los motores del robot a 1 Mbit/s.

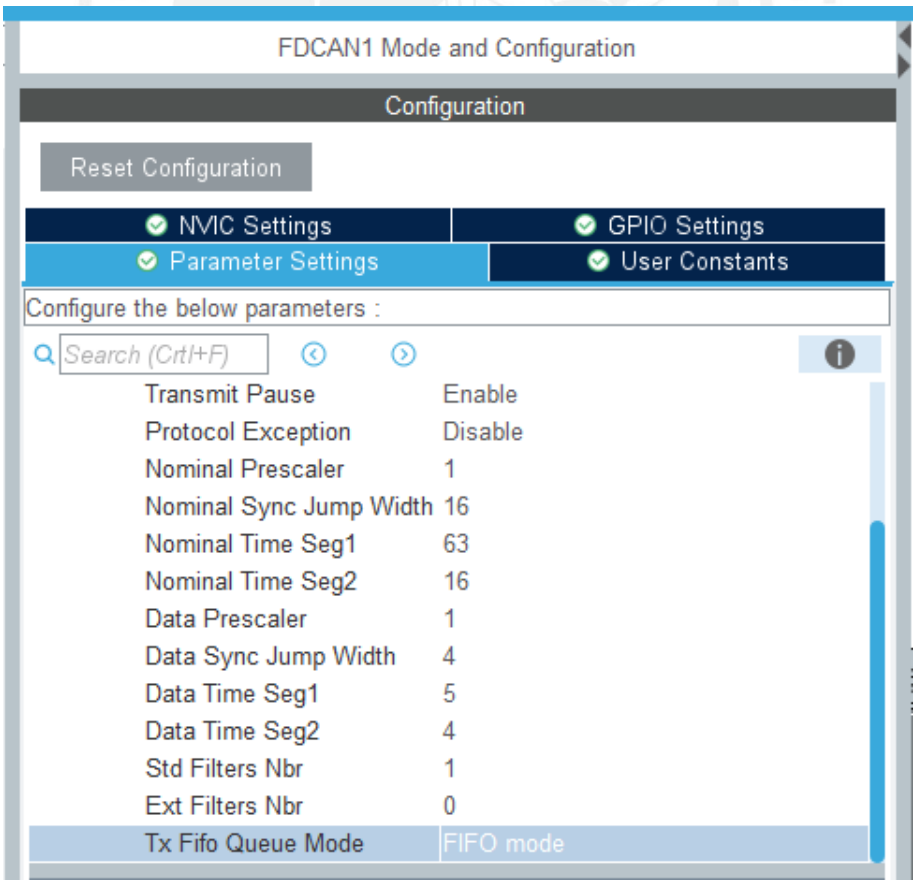

Figura 5.10. Configuración del periférico CAN. Elaboración propia

5.4.5 Instalación y configuración de Computadora embebida

El sistema operativo utilizado por la NUC 13 Pro es Ubuntu 20.04, debido a su facilidad de integración de con ROS y con los distintos drivers utilizados para operar los sensores seleccionados. La instalación se realiza siguiendo los pasos indicados en la página de Ubuntu.

Una vez instalado el sistema operativo, se instala la distribución de ROS correspondiente, la cual es ROS Noetic. Se recomienda optar por la versión completa, la cual incluye visualizadores con Rviz y RQT, esenciales para comprobar el funcionamiento de los algoritmos.

A continuación, en importante verificar la instalación de Python, la versión predeterminada para Ubuntu 20.04 es Python 3.8. Finalmente, se instalan las librerías necesarias para el funcionamiento de los algoritmos de control. Entre las librerías utilizadas se encuentran: matplotlib, numpy, pandas, skfuzzy, rospy, etc. Estas librerías nos permiten, entre otras funcionalidades, generar gráficas, trabajar con matrices, desarrollar controladores difusos, e integrar Python con ROS.

5.5 Integración con ROS

En el capítulo anterior se implementó los controladores en el entorno de simulación Gazebo, en el cual la información de los sensores era simulada y publicada en tópicos de ROS. Estos tópicos eran procesados por el algoritmo de control, que generaba señales de control para los motores simulados en Gazebo.

En esta sección se documentará en la manera en la cual se realiza la transición de la simulación a la integración con el hardware seleccionado en la sección 5.3. Se explicará cómo crear y configurar los tópicos en ROS para recibir información de los sensores físicos, y cómo enviar las señales de control hacia los actuadores reales.

5.5.1 Publicación de la información del AHRS en tópicos de ROS

Con el fin de leer la información proveniente del AHRS utilizando la computadora embebida, se emplea el SDK proporcionado por el fabricante, disponible en su repositorio (Inertial Sense, 2024). En este repositorio, se encuentra un ejemplo de comunicación llamado "*ISCommunicationsExample.cpp*", el cual tiene como función leer distintos mensajes provenientes del sensor, entre estos el mensaje "DID_INS_1", el cual contiene información del roll, pitch y yaw. Los cambios realizados a estos archivos son eliminar la lectura de los otros datos y añadir la publicación de la información de orientación del robot en un tópico de ROS. En la *Figura 5.11* se muestra la información publicada por dicho tópico.

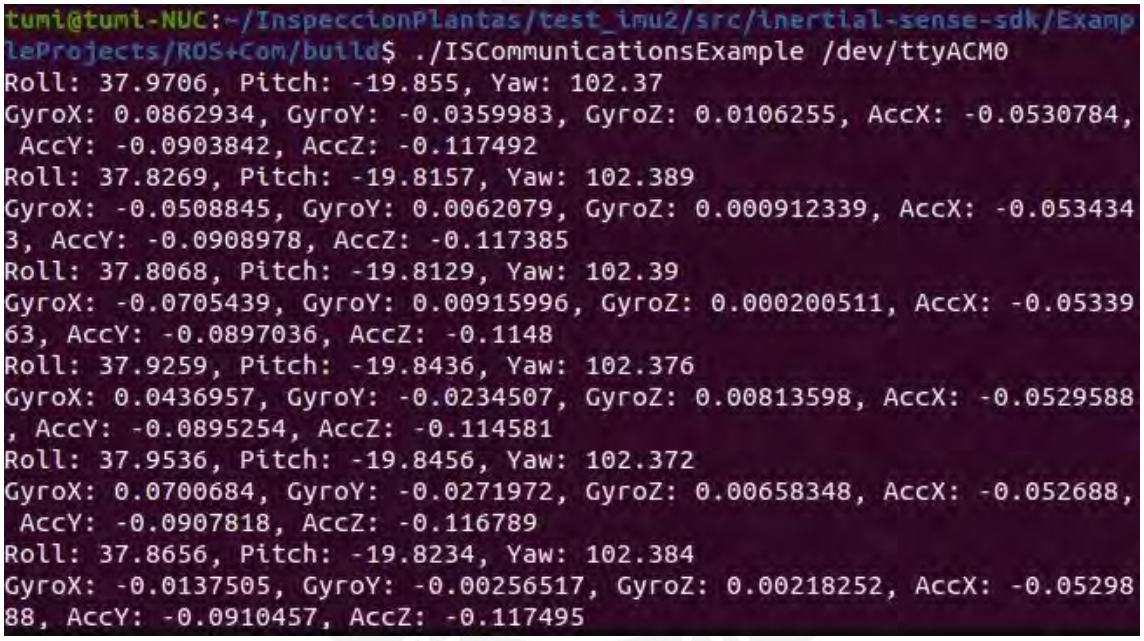

Figura 5.11. Publicación de información de orientación en ROS. Elaboración propia

5.5.2 Publicación de la información de los LiDAR en tópicos de ROS

Debido a su configuración, el Livox MID-360 cuenta de manera predeterminada con una línea de comando *"launch_ROS1/rviz_MID360.launch*" que permite la publicación de un tópico llamado *"/livox/lidar*", donde se encuentra la nube de puntos obtenida por el sensor. En la *Figura 5.12* se puede ver un ejemplo de la nube de puntos, visualizada en RViz.

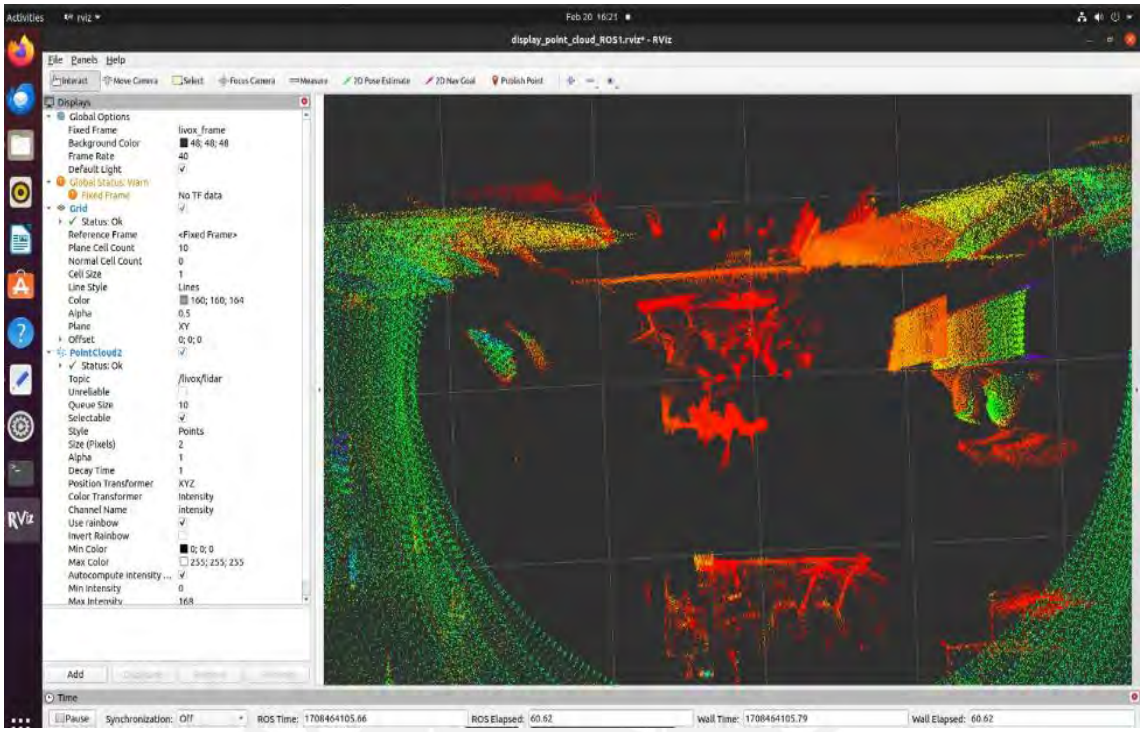

Figura 5.12. Visualización de nube de puntos en RViz. Elaboración propia

5.5.3 Publicación de la información de motores en tópicos de ROS

El proceso para la publicación de la información de los motores es la siguiente. El primer paso es el envío de la velocidad de los motores hacia el microcontrolador. Cada cierto tiempo el microcontrolador solicita esta información de los motores, la cual es enviada mediante el bus CAN. El microcontrolador recepciona la velocidad de los seis motores y la envía a la computadora embebida mediante protocolo USB. La computadora lee estas velocidades y las publica en un tópico de ROS llamado “/joint_states”, este tópico tiene información de la velocidad de los dos motores de tracción y de la velocidad y ángulo de los 4 motores de los flippers. El algoritmo de control crea y publica un tópico llamado “joint_states” donde publica la velocidad deseada de los seis motores, este tópico es leído por otro proceso en la computadora embebida y envía los comandos de velocidad al microcontrolador, el cual luego los envía a los motores. En la *Figura 5.13* se muestra un esquema que resume el proceso de integración en ROS de los motores.

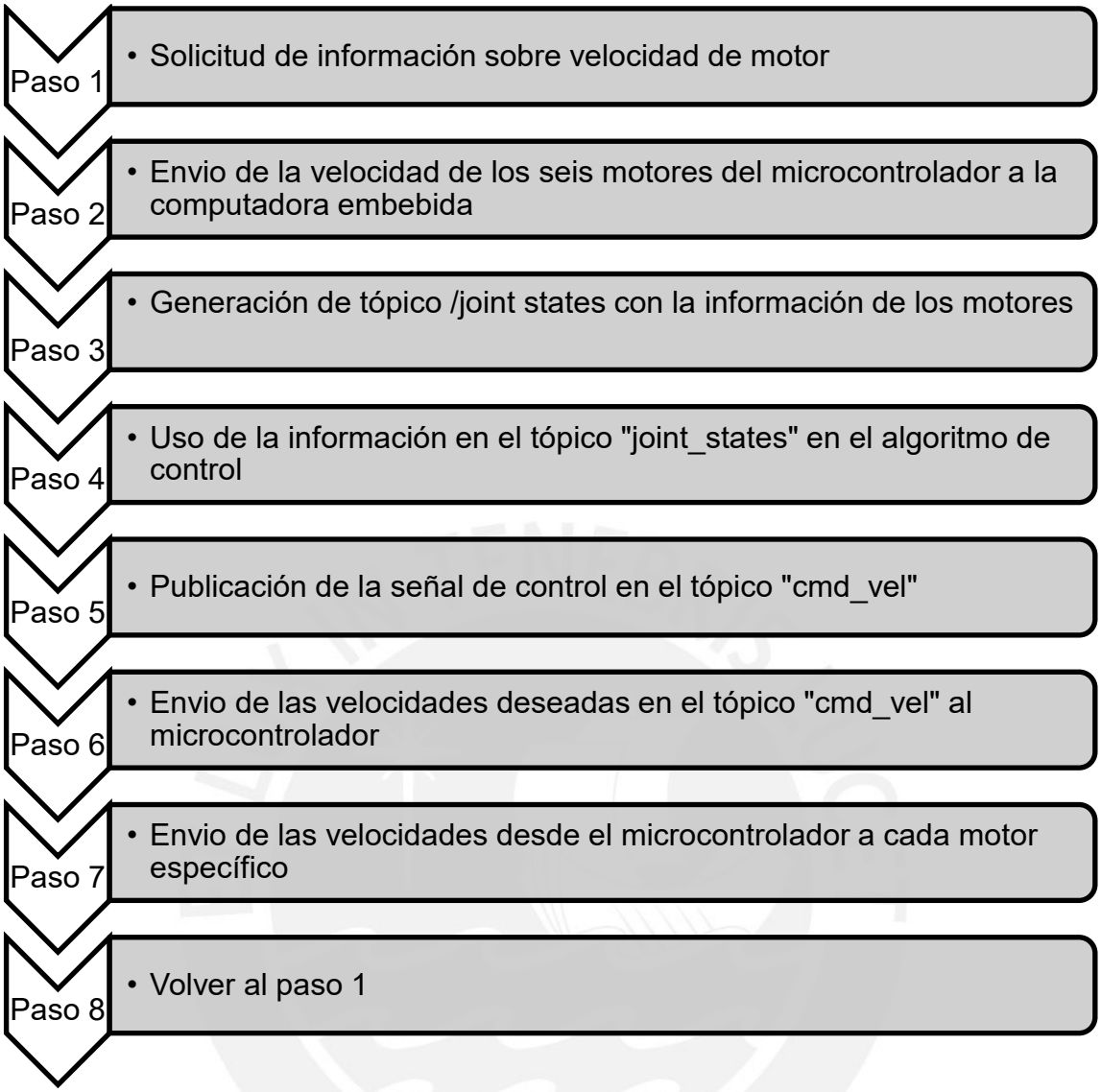

Figura 5.13. Pasos utilizados integrar motores con ROS. Elaboración propia

5.6 Conclusiones de la propuesta de implementación

En este capítulo se presentó la propuesta de implementación en hardware para los algoritmos de control desarrollado a lo largo de este documento. Por un lado, se seleccionó los sensores requeridos para obtener la información necesaria para el control, incluyendo el AHRS, sensor de distancia, computadora embebida y microcontrolador, con un costo aproximado de 4000 USD. Por otro lado, se concluye que, debido a los sensores elegidos, se recomienda que la implementación se lleve a cabo en un entorno con un sistema operativo de distribución Linux, el cual cuente con ROS instalado para una integración sencilla entre los sensores y los algoritmos de control.

Asimismo, se logró configurar y operar con éxito todos los sensores, integrándolos satisfactoriamente mediante ROS, asegurando una comunicación estable entre los distintos componentes del sistema.

CONCLUSIONES

- Se diseñó un sistema de control para un robot articulado con orugas, el cual incorpora un controlador de seguimiento de trayectorias para la plataforma móvil del robot y otro para el sistema de articulaciones, ambos considerando la dinámica de los flippers y orugas del robot.
- Se desarrolló el modelado y simulación del robot móvil articulado con orugas. La precisión y estabilidad del modelo fueron validadas a través de simulaciones en Gazebo, obteniendo un error promedio del 1.48% en la plataforma móvil y del 3.14% en las articulaciones, lo que confirma la similitud de la simulación con el modelo.
- Se seleccionó un controlador de seguimiento basado en Backstepping para la plataforma móvil, luego de demostrar un mejor rendimiento en comparación con un controlador basado en Feedback Linearization. El controlador de Backstepping obtuvo valores de MSE de 0.0370 m en el eje x y 0.0376 m en el eje y, además de un tiempo de respuesta de 1.35 segundos, superando al controlador basado en Feedback Linearization, que obtuvo un MSE de 0.0448 m en el eje x y 0.0432 m en el eje y, con un tiempo de respuesta de 1.62 s. Ambos controladores fueron diseñados y simulados como parte de la evaluación, finalmente, se comprobó la estabilidad del controlador seleccionado.
- Para el sistema de articulaciones se seleccionó un controlador de seguimiento basado en Gain Scheduling, destacando frente a un controlador basado en lógica difusa. Ambos controladores fueron diseñados para este análisis. En las simulaciones, el controlador basado en Gain Scheduling logró un tiempo de establecimiento de 3.072 segundos, un sobreimpulso máximo de 0% y un error en estado estacionario de 1.125%. Por otro lado, el controlador con lógica difusa presentó un tiempo de establecimiento de 6.95 segundos, un sobreimpulso máximo de 0% y un error en estado estacionario de 0.875%. Luego de confirmar la estabilidad del controlador Gain Scheduling, se le seleccionó como la solución adecuada, debido a que este presenta un

tiempo de establecimiento considerablemente menor, mientras que su error en estado estacionario es comparable con la otra alternativa.

- Se presentó la propuesta de implementación en hardware para los algoritmos de control desarrollado a lo largo de este documento. Se seleccionó los componentes necesarios y especificó la configuración necesaria para su uso e integración con ROS. El valor de los componentes seleccionados es de 3516 USD.

RECOMENDACIONES

- Se recomienda explorar técnicas avanzadas de optimización para reducir el tiempo de cálculo y mejorar la precisión del controlador. Esto podría incluir el uso de métodos de optimización, algoritmos genéticos o técnicas de machine learning para ajustar los parámetros del controlador en tiempo real.
- En el modelado del robot es necesario tener en cuenta la existencia de los reductores en los motores, puesto a que al momento de ejecutar la lectura del encoder, la velocidad que se obtiene no es la misma que la de los sprockets motrices de las orugas.
- Es necesario realizar una validación exhaustiva del sistema de control en entornos controlados que simulen condiciones operativas reales. Esto incluye la creación de escenarios de prueba con diferentes tipos de terreno y obstáculos, así como la evaluación del rendimiento del robot en entornos reales de operación industrial para ajustar y perfeccionar los algoritmos de control.
- Como siguiente paso para la implementación de este sistema de control se puede considerar el desarrollo de un planeador de ruta en 3D, se puede plantear este planeador como uno donde el mapa ya es conocido, siendo esta la alternativa más sencilla de implementar. Algoritmos como Dijkstra, o A*, si bien son conocidos por su aplicación en planeadores de ruta en 2D, también existen implementaciones en mapas con 3 dimensiones.

BIBLIOGRAFÍA

A. Chilian, H. H. (2011). Multisensor data fusion for robust pose estimation of a six-legged walking robot. *IEEE International Conference on Intelligent Robots and Systems.*

Al-Jarrah, A., Salah, M., & Almomani, F. (2019). Controlling a Skid-Steered Tracked Mobile Robot with Slippage Using Various Control Schemes. *20th International Conference on Reseach and Eduaction in Mechatronics (REM)* (págs. 1-7). Wels: IEEE.

Aranda-Cetraro, I., Pérez-Zúñiga, G., Rivas-Pérez, R., & Sotomayor-Moriano, J. (2023). Nonlinear Robust Control by a Modulating-Function-Based Backstepping Super-Twisting Controller for a Quadruple Tank System. *Sensors* , 23.

Arm, P. &. (2022). GLIMPSE - Results and Learnings from the First Field Trial of the ESA-ESRIC Space Resources Challenge.

Audonnet, F., Hamilton, A., & Aragon-Camarasa, G. (2022). A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2. 1-7.

Azayev, T., & Zimmermann, K. (2022). Autonomous State-Based Flipper Control for Articulated Tracked Robots in Urban Environments. *IEEE Robotics and Automation Letters*, 7794-7801.

Banco Central de Reserva del Perú. (2021). *Reporte de Inflación. Junio 2021.* Lima: BANCO CENTRAL DE RESERVA DEL PERÚ.

Bhat, A., Kai, N., Susuki, T., Shiroshima, T., & Yoshida, H. (2022). Safe, efficient waypoint manipulation for path planning of non-holonomic robots. *27th International Conference on Automation and Computing (ICAC)* (págs. 1-6). Bristol: IEEE.

Bloesch, M. (2017). *State Estimation for Legged Robots - Kinematics, Inertial Sensing, and Computer Vision.* Zurich: ETH Zurich.

Bloesch, M., Hutter, M., & Leutenegger, S. (2013). State Estimation for Legged Robots: Consistent Fusion of Leg Kinematics and IMU. *MIT Press*, págs. 17-24.

Bruzzone, L., Nodehi, S., & Fanghella, P. (2022). Tracked Locomotion Systems for Ground Mobile Robots: A Review. *Machines*, 648.

Centro de Investigación y Tecnología del Agua. (2021). *Retos y desafíos en la gestión de sedimentos en cuencas andino-amazónicas.* Lima.

Chang, R., Jun-Hong, J., Hui, Z., & Jin-Zhong, Y. (2016). A Backstepping Control Method for Mobile Robot Path Tracking. *3rd Annual International Conference on Mechanics and Mechanical Engineering (MME 2016)* (págs. 680-687). IEEE.

Chen, B., Wu, J., Wang, F., Yang, D., & Zhang, W. (2020). Motion Planning for Autonomous Climbing Stairs for Flipper Robot. *Proceedings of The 2020 IEEE International Conference on Real-time Computing and Robotics* (págs. 531-538). Asahikawa: IEEE.

Chen, B., Wu, J., Wang, F., Yang, D., & Zhang, W. (2020). Motion Planning for Autonomous Climbing Stairs for Flipper Robot. *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, (págs. 531-538).

Chen, W.-K. (2005). *The Electrical Engineering Handbook.* Elseivier.

Claussmann, L., Revilloud, M., Gruyer, D., & Glaser, S. (2020). A Review of Motion Planning for Highway Autonomous Driving. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS vol 21*, 1826-1848.

Dòria-Cerezo, A., Biel, D., Olm, J., & Repecho, V. (2019). Sliding mode control of a differential-drive mobile robot following a path. *18th European Control Conference (ECC)* (págs. 4061-4066). Naples: IEEE.

Dr Robot. (2018). *Jaguar V4 Mobile Platform & Jaguar V4 with Manipulator Arm.* Ontario.

Ejaz, M. M., Tang, T. B., & Lu, C. (2021). Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning. *IEEE Sensors*, (págs. 2230–2240).

Endo, D., Watanabe, A., & Nagatani, K. (2016). Stair climbing control of 4-degrees-of-freedom tracked vehicle based on internal sensors. *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (págs. 112-117).

Falcone, P., Borrelli, F., Asgari, J., Tseng, H., & Hrova, D. (2007). Predictive active steering control for autonomous vehicle systems. *Control Syst. Technol., vol. 15,*, 566-580.

Falcone, P., Borrelli, F., Tseng, H., Asgari, & Hrovat, D. (2008). Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation. *Robust Nonlinear Control, vol. 18,*, 862-875.

Fankhauser, P. (2018). *Perceptive Locomotion for Legged Robots in Rough Terrain.* Zurich: ETH Zurich.

Fankhauser, P. B. (2015). Kinect v2 for Mobile Robot Navigation: Evaluation and Modeling. *International Conference on Advanced Robotics*, (págs. 388-394). Estambul.

Fankhauser, P. D. (2016). Free Gait – An Architecture for the Versatile Control of Legged Robots. *IEEE-RAS International Conference on Humanoid Robots (Humanoids).*

Fenco, L., Pérez-Zuñiga, G., Quiroz, D., & Cuellar, F. (2021). Model Reference Adaptive Fuzzy Controller of a 6-DOF Autonomous Underwater Vehicle. *OCEANS 2021: San Diego – Porto* (págs. 1-7). San Diego: IEEE.

Gazebo. (12 de 07 de 2024). *GitHub - GazeboSim*. Obtenido de tracked_vehicle_simple.world: https://github.com/gazebosim/gazebo-classic/blob/gazebo11/worlds/tracked_vehicle_simple.world

Gehring, C., Fankhauser, P., & Diethelm, R. (2021). ANYmal in the Field: Solving Industrial Inspection of an Offshore HVDC Platform with a Quadrapedal Robot. *Springer Procedings in Advanced Robotics.* Malta.

Gonzáles, D., Pérez, J., Milanés, V., & Nashashibi, F. (2016). A Review of Motion Planning Techniques for Automated Vehicles. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, Vol 17*, 1135 - 1145.

HEBI Robotics. (2023). *Tready - Manual.* Pittsburgh.

Huaman, A., & Pérez-Zuñiga, G. (2019). Design of a Fuzzy Sliding Mode Controller for the Autonomous Path-Following of a Quadrotor. *IEEE Latin America Transactions Vol 17*, 962 - 971.

Inertial Sense. (18 de 10 de 2024). *GitHub*. Obtenido de Inertial Sense SDK: https://github.com/inertialsense/inertial-sense-sdk

J. Nubert, E. W. (2022). Learning-based Localizability Estimation for Robust LiDAR Localization. *EEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Kyoto.

Jiang, Z., & Nijmeijer, H. (1997). Tracking control of mobile robots: A case study in backstepping. *Automatica, vol. 33*, 1393-1399.

Jyh, S., & Jang, R. (1993). ANFIS : Adap tive-Ne twork-Based Fuzzy Inference System. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 23, NO. 3*, 665-685.

Keselman, L. W.-J. (2017). Intel ® RealSense (TM) Stereoscopic Depth Cameras. *IEEE Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1267-1276.

Kowalczyk, W., & Kozlowski, K. (2014). Feedback Linearization of the Differentially Driven Mobile Robot: An Experimental Verification.

Advances in Soft Computing, Intelligent Robotics and Control. (págs. 41-54). Springer.

Leno, V. (2021). Robotic process mining: vision and challenges. *Business & Information Systems Engineering, 63*, 301-314.

Livox. (17 de Octubre de 2024). *GitHub*. Obtenido de Livox-SDK: https://github.com/Livox-SDK

M. Patel, G. W. (2022). LiDAR-guided object search and detection in Subterranean Environments. *IEEE International Symposium on Safety, Security, and Rescue Robotics.* Sevilla.

Mario & Pirri, F. &. (2019). 3D multi-robot patrolling with a two-level coordination strategy. *Autonomous Robots*, 1-33.

Markets And Markets. (12 de Julio de 2023). *Markets And Markets*. Obtenido de Mobile robots market by type (Professional and Personal & Domestic Robts), Application (Agricurltural, Cleaning, Education, Healthcare, Manufacturing, Warehousing & Logistics) and Region - Global Forecast to 2028: https://www.marketsandmarkets.com/Market-Reports/mobile-robots-market-43703276.html

Ministerio de Energía y Minas. (2017). *Actividad económica de las principales regionales mineras del Perú.* Lima.

Mitriakov, A., Papadakis, P., Kerdreux, J., & Garlatti, S. (2021). Reinforcement Learning Based, Staircase Negotiation Learning: Simulation and Transfer to Reality for Articulated Tracked Robots. *IEEE Robotics & Automation Magazine, vol. 28, no. 4*, 10-20.

Mitriakov, A., Papadakis, P., Mai Nguyen, S., & Garlatti, S. (2020). Staircase Negotiation Learning for Articulated Tracked Robots with Varying Degrees of Freedom. *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (págs. 394-400). Abu Dhabi.

Mitriakov, A., Papadakis, P., Mai Nguyen, S., & Garlatti, S. (2020). Staircase Traversal via Reinforcement Learning for Active Reconfiguration of

Assistive Robots. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, (págs. 1-8).

Mourikis, A. I., Trawny, N., Roumeliotis, S. I., & Matthies, D. M. (2007). Autonomous Stair Climbing for Tracked Vehicles. *he International Journal of Robotics Research*, (págs. 737-758).

Murray, G. K. (2007). Parallel Tracking and Mapping for Small AR Workspaces. *IEEE and ACM Int. Symposium on Mixed and Augmented Reality.* Washington DC.

Nagatani, K., Yamasaki, A., Yoshida, K., Yoshida, T., & Koyanagi, E. (2008). Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (págs. 2667-2672).

Novel, B., Campion, G., & Bastin, G. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization. . *J.Robot. Res., vol. 14,*, 543-559.

Organización Internacional del trabajo. (2003). *OIT: El trabajo peligroso mata a millones y cuesta billones*. Obtenido de Organización Internacional del Trabajo (OIT): https://www.ilo.org/global/publications/world-of-work-magazine/articles/WCMS_081389/lang--es/index.htm

Paden, B., ˇC´ap, M., Zheng Yong, S., Yershov, D., & Frazzoli, E. (2016). A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE TRANSACTIONS ON INTELLIGENT VEHICLES*, 33 - 55.

Pecka, M., Šalanský, V., Zimmermann, K., & Svoboda, T. (2016). Autonomous flipper control with safety constraints. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (págs. 2889-2894). Daejeon.

Raffo, G., Gomes, G., Normey-Rico, J., & Kelber, C. (2009). A predictive controller for autonomous vehicle path tracking. *Trans. Intell. Transp. Syst., vol. 10*, 92-102.

Rouček, T., Pecka, M., Čížek, P., Petricek, T., Bayer, J., & Šalanský. (2021). System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge.

Roy, S., & Zhang, Z. (2020). Route Planning for Automatic Indoor Driving of Smart Cars. *2020 IEEE 7th International Conference on Industrial Engineering and Applications* (págs. 1-8). Bangkok: IEEE.

Siciliano, B. a. (2016). *Springer Handbook of Robotics.* Springer.

Silva, P., Teichrieb, V., & Teixeira, J. (2019). Analysis and Comparison of Robotics 3D Simulators. *21st Symposium on Virtual and Augmented Reality (SVR)* (págs. 1-10). Gramado: IEEE.

Silva, S., Verdezoto, N., Paillacho, D. M.-N., & David, H. J. (2023). Online Social Robot Navigation in Indoor, Large and Crowded Environments. *IEEE International Conference on Robotics and Automation (ICRA 2023)* (págs. 1-8). Londres: IEEE.

Stellman, J. (1998). *Enciclopedia de salud y seguridad en el trabajo.* Madrid: Chantal Dufresne.

Szrek, J., Jakubiak, J., & Zimroz, R. (2022). A Mobile Robot-Based System for Automatic Inspection of Belt Conveyors in Mining Industry. *Energies, 15*(1), 327.

Tranzatto, M. &. (2021). CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge .

Tranzatto, M. &. (2022). Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned.

Uhlmann, S. J. (2001). A counter example to the theory of simultaneous localization and map building. *IEEE International Conference on Robotics and Automation.*

Vadym, K., & Aleksey, K. (2012). Modeling and Control of Tracked Mobile Robot. *Third International Scientific Congress, 50th anniversary Technical university of Varna.* Varna: IEEE.

Vaidyanathan, S., & Taher, A. (2020). *Backstepping Control of Nonlinear Dynamical Systems.* Ámsterdam: Elsevier Science.

Wang, C., Ma, K., Yang, L., Ma, H., Xue, X., & Tian, H. (2018). Simulation and experiment on obstacle-surmounting performance of four swing arms and six tracked robot under unilateral step environment. *Transactions of the Chinese Society of Agricultural Engineering*, 46-53.

Wu, C., & Blaabjerg, F. (2021). *Advanced control of power electronic systems—an overview of methods.* Cambridge: Academic Press.

Yuan, Y., Wang, L., & Schwertfeger, S. (2019). Configuration-Space Flipper Planning for Rescue Robots. *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (págs. 37-42). Würzburg: IEEE.

Zanuttigh, P. M. (2016). Operating Principles of Structured Light Depth Cameras". In: Time-of-Flight and Structured Light Depth Cameras: Technology and Applications. *Springer, 2*, 43-79.

Zimmermann, K., Zuzanek, P., Reinstein, M., & Hlavac, V. (2014). Adaptive Traversability of unknown complex terrain with obstacles for mobile robots. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, (págs. 5177-5182). Hong Kong.