Escuela Superior Politécnica Del Litoral

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

Simulación y control de un robot cuadrúpedo multiservicio en entornos regulares utilizando Robot Operating System

INGE-2789

Proyecto Integrador

Previo a la obtención del Título de:

Ingenieros en Mecatrónica

Presentado por:

Andrius Jeremías Jaya Muñoz

Diego Adriel Rodriguez Flores

Guayaquil - Ecuador

Año: 2025

Dedicatoria

El presente proyecto lo dedico a mis padres, Julio Jaya y Zobedia Muñoz, a mis hermanos, Julio y Gregorio, a mi novia Mabel con quienes he compartido grandes momentos de mi vida universitaria y he contado con su apoyo en cualquier adversidad, además de valores que me han inculcado y he podido demostrar a lo largo de toda mi carrera universitaria.

Andrius Jaya M.

Dedicatoria

Este proyecto lo dedico a Dios, mi familia y amigos quienes me apoyaron a lo largo de mi carrera universitaria.

Diego Rodriguez F.

Agradecimientos

Mi más sincero agradecimiento a Dios, por guiarme en toda mi vida, y lograr hacer buenas amistades en mi vida universitaria. A mis Padres, Julio Jaya y Zobeida Muñoz que gracias a su apoyo en cada momento ha sido uno de mis mejores fortalezas a lo largo de mi vida. A mi novia Mabel, agradezco de todo corazón el apoyo y amor dentro de mi vida universitaria.Adicionalemte agradecer a mis hermanos por formar parte de esta etapa.

Andrius Jaya M.

Agradecimientos

Agradezco a Dios y a mis padres por siempre apoyarme. A nuestro tutor Dennys Paillacho y nuestro profesor Marcelo Fajardo por su guia en este proyecto.

Diego Rodriguez F.

Declaración Expresa

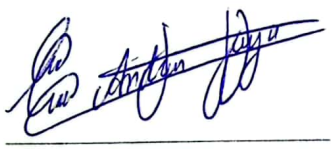
Nosotros Andrius Jeremías Jaya Muñoz y Diego Adriel Rodríguez Flores acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

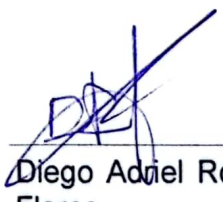
La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 18 de octubre del 2024.



Andrius Jeremías Jaya Muñoz



Diego Adriel Rodríguez Flores

Evaluadores

Firmado electrónicamente por:
MARCELO RODOLFO
FAJARDO PRUNA

Marcelo Fajardo, Ph.D.

Profesor de la Materia

Firmado electrónicamente por:
DENNYS FABIAN
PAILLACHO CHILUIZA

Dennys Paillacho, Ph.D.

Tutor del proyecto

Resumen

Este proyecto presenta el desarrollo y simulación de un sistema de control para un robot cuadrúpedo multiservicio, diseñado para patrullaje autónomo en entornos regulares. Se plantea como solución a las limitaciones de la vigilancia humana, como fatiga y distracción, mediante un sistema basado en ROS 2 y Gazebo.El desarrollo incluye la configuración de un modelo de simulación con sensores avanzados, principalmente LiDAR, para la percepción del entorno. Se implementa un control jerárquico basado en esfuerzo, con un controlador PID por articulación, garantizando estabilidad en la locomoción. Además, se integran técnicas de SLAM para mapeo y navegación autónoma, junto con estrategias de planificación de rutas y evasión de obstáculos.Los resultados de la simulación validan la efectividad del sistema en escenarios de vigilancia, demostrando precisión en la detección de obstáculos y estabilidad en la locomoción. Se concluye que la metodología aplicada permite una implementación eficiente en entornos simulados, con posibilidad de optimización para futuras aplicaciones en vigilancia autónoma.

Palabras Clave: Robot cuadrúpedo, ROS 2, SLAM, control PID, simulación.

Abstract

This project presents the development and simulation of a control system for a quadruped multi-service robot, designed for autonomous patrolling in regular environments. It addresses the limitations of human surveillance, such as fatigue and distraction, by implementing a system based on ROS 2 and Gazebo. The development includes configuring a simulation model with advanced sensors, mainly LiDAR, for environmental perception. A hierarchical effort-based control system is implemented, featuring a PID controller for each joint to ensure locomotion stability. Additionally, SLAM techniques are integrated for mapping and autonomous navigation, along with route planning and obstacle avoidance strategies. The simulation results validate the effectiveness of the system in surveillance scenarios, demonstrating accurate obstacle detection and stable locomotion. It is concluded that the applied methodology allows for an efficient implementation in simulated environments, with potential optimization for future autonomous surveillance applications.

Keywords: *Quadruped robot, ROS 2, SLAM, PID control, simulation.*

Índice general

Abreviaturas

ROS 2	Robot Operating System 2
URDF	Unified Robot Description Format
LiDAR	Light Detection and Ranging
SLAM	Simultaneous Localization and Mapping
EKF	Extended Kalman Filter
SLIP	Spring-Loaded Inverted Pendulum
CPG	Central Pattern Generator
ZMP	Zero Moment Point
CHAMP	Control Hierarchy for Autonomous Multi-legged Platforms
RRT	Rapidly-exploring Random Tree
DWA	Dynamic Window Approach
TEB	Timed Elastic Band
IMU	Inertial Measurement Unit
ESPOL	Escuela Superior Politécnica del Litoral
PID	Control Proporcional-Integral-Derivativo
TF	Transformaciones en ROS
FL	Front Left (Frontal Izquierda)
FR	Front Right (Frontal Derecha)
RL	Rear Left (Trasera Izquierda)
RR	Rear Right (Trasera Derecha)

Simbología

J	Matriz Jacobiana
τ	Vector de torques
q	Vector de coordenadas generalizadas
$\dot{q}$	Velocidad generalizada
$\ddot{q}$	Aceleración generalizada
$M(q)$	Matriz de inercia
$C(q, \dot{q})$	Matriz de Coriolis y fuerzas centrífugas
$G(q)$	Vector de fuerzas gravitacionales
F	Fuerza aplicada externa
θ	Ángulos articulares
x	Posición cartesiana
$\dot{x}$	Velocidad cartesiana
$\ddot{x}$	Aceleración cartesiana
$u(t)$	Señal de control aplicada a la articulación
$e(t)$	Error entre la posición deseada y la posición actual
K_p, K_i, K_d	Ganancias proporcional, integral y derivativa del controlador PID

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

En los últimos años, el interés por el desarrollo de robots autónomos ha crecido considerablemente debido a su capacidad para realizar tareas que requieren precisión, resistencia y autonomía [1; 2]. Los robots cuadrúpedos, en particular, han ganado popularidad en aplicaciones que demandan movilidad en terrenos regulares y desafiantes [3]. Inspirados en la locomoción animal, estos robots son capaces de adaptarse a diversas superficies, lo que los hace idóneos para tareas de vigilancia y patrullaje en entornos donde la seguridad y la capacidad de reacción son primordiales [4].

La vigilancia humana en estos entornos presenta limitaciones significativas debido a factores como la fatiga, la distracción y la reducción de la concentración en turnos largos o nocturnos. Estos factores pueden llevar a un incremento en la tasa de incidentes, ya que los guardias pueden no detectar amenazas a tiempo. En este contexto, el uso de un robot cuadrúpedo multiservicio controlado por el sistema operativo de robots (ROS, por sus siglas en inglés) surge como una solución viable y eficiente. Este tipo de robot tiene el potencial de realizar patrullajes de forma autónoma, detectando anomalías, intrusiones o movimientos en el entorno sin sufrir los efectos de la fatiga, y proporcionando así una vigilancia constante y precisa [5].

La propuesta de este proyecto se centra en el desarrollo y simulación de un sistema de control para un robot cuadrúpedo que permita validar su capacidad de patrullaje en entornos regulares. Utilizando sensores avanzados como LiDAR , y controlado mediante algoritmos optimizados en ROS, se busca que el robot pueda mapear y navegar de manera autónoma [6], ajustándose dinámicamente a cambios en el entorno y superando obstáculos. A través de una simulación en el entorno de Gazebo, se pretende evaluar la eficacia del robot antes de su implementación física, garantizando que cumpla con los requisitos necesarios para operar en

un entorno real.

1.1 Descripción del problema

La vigilancia en entornos críticos, tales como instalaciones industriales y de infraestructura estratégica, enfrenta importantes limitaciones cuando se confía exclusivamente en supervisión humana. Factores como la fatiga, la distracción y la reducción de concentración, especialmente en turnos largos o nocturnos, comprometen la seguridad de los espacios monitoreados. Estas limitaciones incrementan el riesgo de incidentes, ya que los guardias pueden no detectar amenazas o anomalías a tiempo, generando brechas en la protección de estos entornos y exponiéndolos a posibles fallas de seguridad.

En este contexto, el uso de un robot cuadrúpedo multiservicio, diseñado para realizar patrullajes autónomos mediante sensores avanzados como LiDAR y cámaras, se presenta como una solución efectiva. Equipado con algoritmos de navegación y detección de objetos en ROS, el robot puede realizar vigilancia constante, manteniendo un monitoreo preciso sin verse afectado por distracciones o fatiga, y garantizando una capacidad de reacción rápida ante cambios en el entorno o la presencia de obstáculos.

1.1.1 Requerimientos del sistema

Para garantizar un rendimiento óptimo en tareas de patrullaje, el robot cuadrúpedo debe cumplir con los siguientes requerimientos:

- **Precisión de detección de sensores**: El sistema de sensores, que incluye LiDAR y cámaras, debe detectar objetos y anomalías en un radio mínimo de 5 metros. La resolución de los sensores debe permitir la identificación precisa de amenazas potenciales en el entorno.
- **Capacidad de respuesta en tiempo real**: El robot debe ser capaz de responder ante

cualquier cambio en el entorno en un tiempo de reacción inferior a 5 segundos, asegurando que pueda maniobrar y evitar obstáculos sin retrasos.

- **Navegación autónoma en ROS**: El sistema debe incluir un algoritmo de navegación optimizado en ROS, que permita el mapeo del entorno en tiempo real y la adaptación dinámica ante obstáculos.

1.1.2 Restricciones del sistema

El diseño y la implementación del sistema deben tener en cuenta las siguientes restricciones:

- **Condiciones ambientales**: El robot está diseñado para operar en entornos con iluminación moderada y en superficies regulares. No se espera que funcione en condiciones extremas de temperatura o humedad, ni en terrenos accidentados.
- **Capacidad de procesamiento**: La capacidad de procesamiento del robot está limitada, por lo que los algoritmos de detección y navegación deben estar optimizados para ejecutarse en un sistema con recursos computacionales moderados.

1.1.3 Variables de interés

Para evaluar el rendimiento del sistema y su efectividad en tareas de patrullaje, se identificarán y monitorearán las siguientes variables de interés:

- **Precisión en el mapeo y navegación del entorno**: Exactitud del sistema al crear y actualizar el mapa del área patrullada, y su capacidad para identificar obstáculos y rutas de navegación.
- **Tiempo de respuesta ante eventos**: Tiempo necesario para que el robot detecte y reaccione ante una anomalía o un obstáculo en su camino.

1.2 Justificación del problema

La vigilancia en instalaciones críticas, como infraestructuras estratégicas e industrias, es una tarea fundamental para garantizar la seguridad y continuidad de las operaciones. Sin embargo, las tareas de vigilancia realizadas exclusivamente por personal humano presentan limitaciones importantes, especialmente en condiciones de trabajo prolongado o nocturno, donde la fatiga, la distracción y la pérdida de concentración reducen la efectividad de la supervisión. Estas limitaciones aumentan el riesgo de incidentes que pueden tener consecuencias significativas, como la pérdida de activos valiosos, riesgos de seguridad para los empleados o interrupciones en las operaciones.

La implementación de un robot cuadrúpedo autónomo, equipado con un sistema de control optimizado en ROS y sensores avanzados como LiDAR y cámaras, ofrece una solución innovadora que supera las limitaciones humanas en la vigilancia. Este robot tiene la capacidad de realizar patrullajes continuos sin verse afectado por la fatiga, proporcionando una vigilancia precisa y constante en áreas críticas. Además, su capacidad de detectar y reaccionar ante anomalías o movimientos en el entorno lo convierte en una herramienta eficaz para mitigar brechas de seguridad en estos espacios.

El uso de un sistema de control en ROS permite no solo una navegación autónoma eficiente, sino también una fácil integración y adaptación del robot a diferentes entornos y requerimientos de vigilancia. Esta flexibilidad es especialmente relevante en un mundo donde las amenazas y los riesgos de seguridad evolucionan constantemente. La simulación del sistema de control en un entorno virtual, como Gazebo, también facilita la optimización de su rendimiento antes de su implementación física, asegurando que el robot esté completamente preparado para las demandas de vigilancia en condiciones reales. Esto no solo ahorra tiempo y recursos en pruebas físicas, sino que también permite realizar ajustes

previos en el comportamiento del robot, aumentando su confiabilidad en escenarios prácticos.

En conclusión, este proyecto aporta una solución tecnológica innovadora que responde a los desafíos actuales de la vigilancia en entornos industriales y de infraestructura crítica, proporcionando una alternativa autónoma y optimizada para reforzar la seguridad y la efectividad en tareas de monitoreo constante.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema de simulación y control para un robot cuadrúpedo multiservicio, capaz de realizar tareas de vigilancia en entornos regulares, empleando el Robot Operating System (ROS).

1.3.2 Objetivos Específicos

- Implementar un algoritmo de control de navegación y detección de obstáculos en un entorno simulado.
- Integrar sensores avanzados para el monitoreo y detección en tiempo real de intrusiones o anomalías.
- Evaluar los algoritmos de control para mejorar la estabilidad del robot cuadrúpedo durante la vigilancia en entornos regulares.

1.4 Marco teórico

1.4.1 Introducción a los Robots Cuadrúpedos

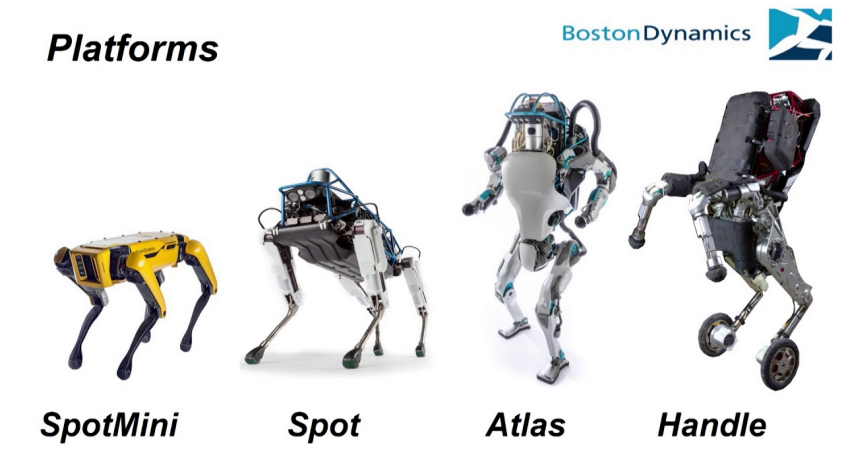
Definición y Tipos de Robots Cuadrúpedos

Los robots cuadrúpedos, una categoría dentro de los robots móviles con patas, se diseñan para replicar la locomoción de animales de cuatro patas, como perros y caballos, y se destacan por su capacidad para navegar en terrenos complejos. A diferencia de los robots con ruedas o orugas, que dependen de superficies regulares para moverse con eficacia, los robots cuadrúpedos son capaces de superar obstáculos, escalar terrenos irregulares y mantener la estabilidad en superficies inestables. Estas características hacen que los robots cuadrúpedos sean ideales para aplicaciones en entornos naturales y difíciles, donde los métodos convencionales de movilidad son ineficaces[7].

Los robots cuadrúpedos pueden clasificarse según diferentes criterios, como el sistema de accionamiento (hidráulico, eléctrico, neumático) y el tipo de estructura corporal (torso rígido o flexible). Robots como BigDog de Boston Dynamics utilizan accionamientos hidráulicos, que les permiten una gran capacidad de carga y adaptación a terrenos difíciles[3], mientras que modelos como MIT MiniCheetah, spot [8], A1[9], Anymal[10; 11], emplean actuadores eléctricos, que ofrecen mayor control y son más compactos[12]. La clasificación de estos robots también puede basarse en su diseño biomimético, replicando las características de animales específicos para mejorar sus capacidades de locomoción en entornos específicos, como robots inspirados en perros, felinos o antílopes.

Figura 1.1.

Robots cuadrúpedos



Comparativa con Otros Tipos de Robots

Los robots cuadrúpedos presentan varias ventajas frente a los robots con ruedas o con orugas. En primer lugar, su capacidad de adaptación a diversos tipos de terrenos los hace adecuados para tareas de exploración en áreas irregulares, donde los robots con ruedas tienden a quedarse atascados o a experimentar una pérdida significativa de tracción. Esto es debido a que los cuadrúpedos tienen múltiples puntos de contacto con el suelo, permitiéndoles distribuir su peso y ajustar cada pata de forma independiente para mantener la estabilidad [7].

Además, los cuadrúpedos tienen una movilidad omnidireccional, lo que les permite moverse lateralmente y maniobrar en espacios reducidos[13; 14], una ventaja significativa sobre los robots con ruedas en entornos confinados o de acceso complejo. Esta habilidad es particularmente relevante en aplicaciones de rescate, patrullaje y exploración, donde la capacidad de moverse entre escombros o dentro de estructuras estrechas es crucial para su efectividad. Robots como BigDog han demostrado la efectividad de esta configuración en aplicaciones militares y de rescate, debido a su capacidad para transportar cargas pesadas y moverse en terrenos que serían imposibles para vehículos convencionales.

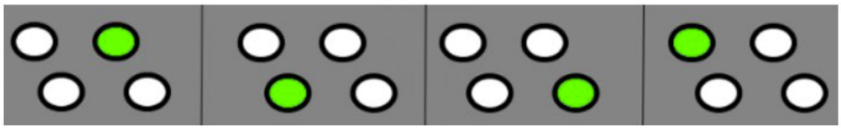
Métodos de locomoción

Luego de explorar las ventajas de los robots cuadrúpedos frente a otros tipos de robots, es importante describir los diferentes métodos de locomoción que emplean, ya que cada uno afecta su estabilidad y adaptabilidad a distintos terrenos. Los métodos de locomoción en robots cuadrúpedos incluyen la marcha, el trote y el galope, cada uno con características específicas que permiten ajustar el movimiento según las condiciones del entorno.

- **Marcha** : En este modo, al menos tres patas están en contacto con el suelo en todo momento, lo que proporciona alta estabilidad. Las patas avanzan una a una en un ciclo constante, lo cual hace que este método sea ideal para terrenos irregulares y situaciones que requieren precisión, reduciendo el riesgo de caídas

Figura 1.2.

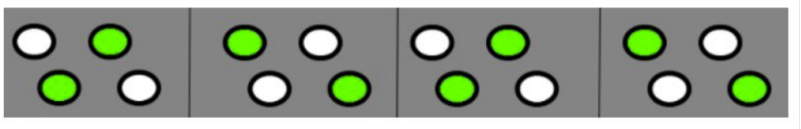
PATRÓN DE MARCHA



- **Trote**: Aquí, el robot mueve simultáneamente las patas diagonales opuestas (por ejemplo, pata delantera derecha con pata trasera izquierda), lo que asegura que siempre haya dos puntos de apoyo en el suelo. Esto incrementa la velocidad respecto a la marcha sin comprometer demasiado la estabilidad .

Figura 1.3.

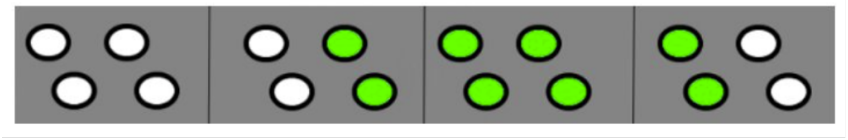
PATRÓN DE TROTE



- **Galope**: Este método es el más veloz, con las patas traseras y delanteras moviéndose en conjunto. Durante una fase del ciclo, todas las patas están en el aire, lo que reduce la estabilidad, haciéndolo más adecuado para terrenos lisos y donde se prioriza la rapidez sobre el equilibrio [15].

Figura 1.4.

PATRÓN DE GALOPE



Estos modelos de marcha ofrecen distintos grados de velocidad y adaptabilidad, permitiendo ajustar la locomoción según las condiciones del entorno.

1.4.2 Control Dinámico de Robots Cuadrúpedos

Cinematica

La cinemática de un robot cuadrúpedo describe cómo se mueven sus patas y articula sus eslabones para alcanzar diferentes posiciones en el espacio tridimensional, lo cual es esencial para controlar su desplazamiento y postura en terrenos variados. Este análisis incluye tanto la cinemática directa como la inversa y se fundamenta en el método de Denavit-Hartenberg (D-H), que facilita la representación del movimiento y las relaciones de posición en las patas del robot [11]. Comprender la cinemática permite optimizar la estabilidad y adaptabilidad del robot al moverse, características que se destacaron previamente en la comparación con otros tipos de robots móviles. Además, esta base teórica es fundamental para la metodología de simulación, donde el control cinemático precisa ajustarse dinámicamente según el terreno, asegurando que el robot mantenga un equilibrio adecuado durante el patrullaje autónomo. [16].

Cinematica directa

La cinemática directa permite calcular la posición y orientación del extremo de una pata a partir de los ángulos de sus articulaciones. En el caso de un robot cuadrúpedo, cada pata suele modelarse con tres grados de libertad (DOF), permitiendo el movimiento tridimensional. El método D-H se usa para definir una serie de sistemas de referencia para cada articulación y eslabón, mediante matrices de transformación que describen la posición de cada eslabón respecto al anterior. La matriz de transformación T_i para cada eslabón está dada por:

$$T_i = \begin{bmatrix} \cos\theta_i & -\theta_i \cos\alpha_i & \theta_i\alpha_i & a_i \cos\theta_i \\ \theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i\alpha_i & a_i\theta_i \\ 0 & \alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

donde θ_i, d_i, a_i y α_i son los parámetros de cada articulación y eslabón, representando el ángulo de rotación, desplazamiento, longitud del eslabón y ángulo entre ejes, respectivamente. La posición final del extremo de la pata, P_{extremo}, se calcula multiplicando las matrices de transformación de cada eslabón:

$$T_{\text{total}} = T_1 \cdot T_2 \cdot T_3 \tag{1.2}$$

Esta matriz total T_{total} permite determinar la posición y orientación de la pata en el marco de referencia del cuerpo del robot, un paso crucial para controlar la postura y el equilibrio durante la marcha [17; 18].

Cinematica inversa

La cinemática inversa, por otro lado, es el proceso de calcular los ángulos articulares que deben adoptar las articulaciones para que el extremo de la pata alcance una posición y orientación específicas. En la cinemática inversa se busca resolver un sistema de ecuaciones no lineales que puede tener múltiples o ninguna solución física, dependiendo de las limitaciones de diseño y las posiciones deseadas.

Para resolver este problema de cinemática inversa, se utiliza la *matriz Jacobiana* J, que relaciona los cambios en los ángulos articulares $\Delta\theta$ con los cambios en la posición del extremo de la pata ΔP:

$$\Delta P = J \cdot \Delta\theta \tag{1.3}$$

donde la Jacobiana se define como:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \end{bmatrix} \tag{1.4}$$

Este sistema de ecuaciones se resuelve iterativamente, por ejemplo, con el método de Newton-Raphson, ajustando los ángulos articulares según la diferencia entre la posición deseada y la actual del extremo de la pata. El ajuste se realiza con la fórmula:

$$\theta_{n+1} = \theta_n + J^{-1} \cdot (P_{\text{deseado}} - P_{\text{actual}}) \tag{1.5}$$

donde J^{-1} es la inversa de la Jacobiana. Este proceso continúa hasta que la diferencia entre P_{deseado} y P_{actual} sea mínima, asegurando que el extremo de la pata se acerque con precisión a la posición objetivo [19; 18; 17]

La combinación de la cinemática directa e inversa es fundamental para el control del movimiento de robots cuadrúpedos. La cinemática directa proporciona la posición actual de cada pata en el espacio, mientras que la inversa permite determinar los ángulos de las articulaciones para alcanzar posiciones específicas. Esto es esencial para la planificación de trayectorias y el control de estabilidad, facilitando así la locomoción segura y estable del robot en entornos complejos.

Dinamica

Para el modelado dinámico del robot cuadrúpedo, se emplean las formulaciones de Newton-Euler y de Lagrange, siendo esta última preferida para configuraciones complejas de manipuladores. La ecuación de movimiento general en el espacio articular se expresa como:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G_g(\theta) = \tau \tag{1.6}$$

donde: - $M(\theta)$: matriz inercial, - $C(\theta, \dot{\theta})\dot{\theta}$: vector de fuerzas de Coriolis y centrífugas, - $G_g(\theta)$: vector de fuerzas gravitacionales, - τ: vector de torques en las articulaciones.

La formulación de Lagrange, utilizada aquí, define la energía del sistema mediante:

$$L = T - U \tag{1.7}$$

donde T es la energía cinética y U es la energía potencial. La ecuación dinámica resultante es:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i \tag{1.8}$$

Estas ecuaciones permiten el control preciso de cada articulación del robot cuadrúpedo durante el movimiento de trote. La dinámica del sistema, junto con la cinemática descrita

previamente, proporciona una base integral para coordinar el control del robot y responder adecuadamente a fuerzas externas, como las que enfrentaría en terrenos irregulares. Este enfoque de modelado dinámico complementa la precisión que ofrece la cinemática, mejorando aún más la estabilidad y adaptabilidad del robot [19].

1.4.3 Control de Torque y Equilibrio en Robots Cuadrúpedos

Tras definir el modelado dinámico del robot cuadrúpedo, el control de torque y equilibrio se convierte en un aspecto esencial para mantener estabilidad y precisión en movimientos complejos. Los actuadores cuasi-directos (QDD) permiten aplicar fuerzas controladas de manera directa, mejorando la respuesta dinámica del robot frente a variaciones en el terreno. Estos actuadores facilitan el control preciso del torque en cada articulación, lo cual es crucial para la estabilidad, ya que minimizan los efectos de fuerzas externas sobre el robot [20]

El control PID (Proporcional-Integral-Derivativo) es ampliamente utilizado para gestionar el equilibrio en robots cuadrúpedos. Al regular el torque en función del error de posición, los controladores PID ajustan el movimiento de cada pata para alcanzar la posición deseada, logrando así una marcha estable. Para mejorar la adaptabilidad, el PID puede ser implementado con ajustes automáticos que permiten cambiar los parámetros en función de las condiciones del entorno, reduciendo errores de balance y optimizando la respuesta a perturbaciones externas [21].

Tipos de controladores

Los robots cuadrúpedos emplean una variedad de controladores para garantizar estabilidad y adaptabilidad en terrenos complejos. Entre los controladores más comunes están los Controladores de Patrón Central Generador (CPG), que imitan patrones neuronales

para generar movimientos rítmicos [22]. Estos controladores son eficaces en la generación de gaits básicos como el trote y el galope, proporcionando estabilidad sin necesidad de modelos dinámicos complejos. Osciladores no lineales que imitan los CPG biológicos incluyen el modelo de Fukuoka [23; 24], el oscilador de Van der Pol [25] y el oscilador de Hopf [26].

Otra aproximación es el Modelo de Pendulum Invertido con Resorte (SLIP), que simplifica el control de locomoción al modelar cada pata como un resorte, permitiendo al robot cuadrúpedo realizar saltos y mantener el equilibrio en superficies irregulares. Este modelo es común en robots como el RHex [27], que dependen de un balance pasivo complementado con controladores activos para entornos difíciles. La estabilidad del modelo SLIP también inspiró controladores híbridos que combinan la dinámica pasiva predicha por el modelo SLIP con un controlador de nivel superior [28; 29].

Por último, los Controladores Basados en Punto de Momento Cero (ZMP), como los empleados en robots como StarlETH [30] o KOLT [31], garantizan estabilidad al mantener el punto de contacto en el área de soporte del robot, aunque pueden ser limitantes en terrenos no planos. Sin embargo, combinaciones de ZMP con técnicas de optimización ofrecen un control robusto para diferentes tipos de gaits y saltos, como se observa en robots desarrollados por Boston Dynamics [32; 3].

1.4.4 Navegación Autónoma y Técnicas SLAM

Introduccion a SLAM

SLAM, que significa Simultaneous Localization and Mapping (localización y mapeo simultáneos), es una técnica fundamental en robótica que permite a un robot crear un mapa de su entorno mientras determina su posición dentro de él. Este proceso es crucial para la navegación autónoma en espacios desconocidos o en situaciones donde no hay acceso a

GPS, como en interiores o entornos subterráneos. SLAM combina sensores de percepción, como LiDAR o cámaras, con algoritmos de localización probabilística para estimar la posición del robot y actualizar el mapa en tiempo real. Su importancia radica en la capacidad de permitir que el robot navegue y evite obstáculos de forma autónoma en entornos complejos, mejorando así su autonomía y adaptabilidad a diferentes escenarios.[33; 34]

Algoritmos de SLAM en ROS

En el entorno de ROS (Robot Operating System), los algoritmos de SLAM más populares incluyen Gmapping, HectorSLAM y Cartographer. Gmapping utiliza filtros de partículas para estimar la posición del robot y es ideal cuando hay datos de odometría precisos, como en robots con ruedas. HectorSLAM, por otro lado, emplea el filtro de Kalman extendido (EKF) y es útil en sistemas con sensores LiDAR de alta frecuencia, aunque su rendimiento disminuye cuando la resolución del LiDAR es baja o el entorno es extenso. Cartographer, desarrollado por Google, implementa un enfoque basado en grafos que combina SLAM local y global, ofreciendo alta precisión y eficiencia en el mapeo. Aunque Cartographer demanda más recursos de procesamiento, su rendimiento en entornos cerrados y la precisión de los mapas generados suelen superar a los otros algoritmos en términos de calidad de mapeo y estabilidad en trayectorias complejas[34].

Sensores para SLAM en Simulación

Para realizar SLAM en simulación, se emplean sensores como LiDAR y cámaras RGB-D (profundidad y color). LiDAR es uno de los sensores más precisos para capturar detalles del entorno, ya que emite láseres que miden distancias, generando nubes de puntos que se convierten en mapas de alta resolución. Las cámaras RGB-D, que combinan color y profundidad, complementan el LiDAR, especialmente en entornos con baja iluminación o

donde se requiere una mayor percepción de profundidad. Estos sensores permiten una integración eficaz en entornos simulados de ROS, proporcionando datos críticos que los algoritmos de SLAM procesan para generar mapas detallados y precisos, lo que facilita la localización y la navegación segura del robot en su entorno [33; 34].

1.4.5 Planificación de Rutas y Evitación de Obstáculos

Algoritmos de Planificación de Rutas

La planificación de rutas es fundamental para la navegación de robots en entornos complejos. Existen varios algoritmos comunes, cada uno con sus fortalezas y aplicaciones específicas. El algoritmo A* es conocido por ser heurístico y eficiente, utilizando una función de coste para buscar la ruta óptima en un gráfico o mapa, siendo ideal para entornos estructurados. Por otro lado, Dijkstra garantiza la ruta más corta, explorando todos los caminos posibles, aunque puede ser más costoso en términos de tiempo computacional. Para entornos más complejos o en tiempo real, el árbol de exploración aleatoria rápida (RRT) y su variante optimizada RRT* son altamente efectivos. Estos algoritmos generan rutas mediante puntos aleatorios en el espacio libre, conectándolos en un gráfico, lo que permite una solución subóptima rápida con la ventaja de ajustarse bien a configuraciones dinámicas. Estos métodos, especialmente RRT*, han sido implementados en simulaciones de robots de servicio y cuadrúpedos para proporcionar trayectorias adaptativas y óptimas[35; 33].

Estrategias de Evitación de Obstáculos

La evitación de obstáculos es esencial para la seguridad y adaptabilidad en la navegación robótica. La aproximación DWA (Dynamic Window Approach) es una estrategia reactiva que calcula velocidades seguras para evitar colisiones en tiempo real, considerando la dinámica y las limitaciones del robot. Esta técnica evalúa las velocidades posibles dentro

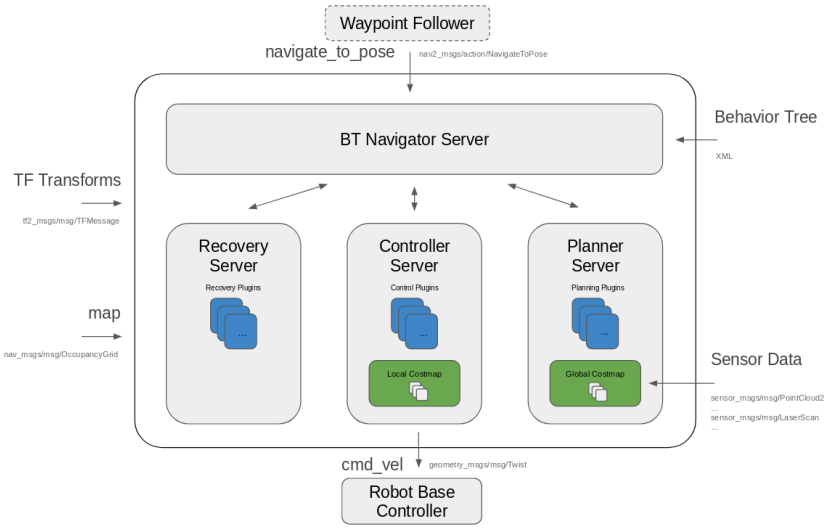
de una ”ventana” basada en la posición actual y el destino inmediato del robot, generando comandos de velocidad que evitan obstáculos sin desviarse significativamente de la trayectoria óptima. Otra técnica destacada es el uso de campos de potencial artificial, que genera fuerzas atractivas y repulsivas para mantener la trayectoria del robot lejos de los obstáculos, siendo útil en ambientes con obstáculos dinámicos. La implementación de estas técnicas, especialmente en robots cuadrúpedos, permite una mayor capacidad de respuesta y adaptabilidad, asegurando una navegación eficiente en entornos impredecibles[35; 33].

1.4.6 Nav2 Structure

El sistema de navegación Nav2 en ROS 2 es una evolución del nodo `move_base` de ROS 1 y está diseñado para proporcionar una navegación autónoma robusta en robots móviles. Nav2 está compuesto por varios servidores y nodos que trabajan en conjunto para realizar tareas de planificación de rutas, control de movimiento y recuperación en caso de fallas en la navegación [36].

Figura 1.5.

Navegación en Ros2



1. **BT Navigator Server** El servidor de BT Navigator carga un árbol de comportamiento

(Behavior Tree) en formato XML, donde cada nodo del árbol representa un paso en la lógica de navegación, como planificación de rutas, control de movimiento o recuperación. Este árbol permite al robot seleccionar y ejecutar comportamientos específicos basados en su estado actual y en los requerimientos de la misión. La estructura modular del árbol de comportamiento facilita la personalización y escalabilidad del sistema de navegación, permitiendo adaptar el flujo de trabajo del robot según sus necesidades operativas.

2. **Planner Server** El servidor de planificación, o Planner Server, emplea plugins de planificación para generar rutas óptimas en el entorno del robot. Este componente utiliza algoritmos de planificación como A* o Dijkstra para determinar rutas seguras desde la posición actual hasta el objetivo, considerando el mapa global generado por sensores como LiDAR o cámaras. Además, Nav2 soporta `NavFn_planner`, que es una versión mejorada del planificador de ROS 1. Este servidor utiliza el `Global Costmap`, un mapa de costos que asigna valores a cada celda del entorno, destacando obstáculos y áreas de riesgo.

3. **Controller Server** El servidor controlador o Controller Server es responsable de guiar al robot a lo largo de la ruta planificada, ajustando su posición y velocidad en tiempo real. En Nav2, existen varias opciones de control, incluyendo `dwb_controller` y `TEB Controller`, que permiten al robot evitar obstáculos dinámicos mientras sigue la trayectoria. Este servidor utiliza un `Local Costmap` para obtener información actualizada del entorno inmediato del robot, permitiéndole responder rápidamente a cambios en su proximidad y realizar ajustes de velocidad y dirección.

4. **Recovery Server** El Recovery Server actúa como un sistema de emergencia para situaciones en las que el robot no puede continuar su ruta debido a obstáculos imprevistos o errores en la planificación. Este servidor incluye plugins de recuperación predefinidos, como girar en su lugar, retroceder o esperar hasta que el camino esté despejado. Estas funciones

aseguran que el robot pueda intentar nuevas maniobras en caso de bloqueos o rutas fallidas, aumentando así su capacidad de completar misiones complejas sin intervención humana.

5. **Waypoint Follower** El módulo de Waypoint Follower permite al robot seguir una serie de puntos de referencia (waypoints) que definen su trayectoria. Este componente es especialmente útil para misiones de patrullaje o vigilancia, donde el robot debe visitar múltiples ubicaciones en un entorno. El Waypoint Follower se comunica con el BT Navigator para ejecutar los puntos de referencia de manera secuencial, asegurando que cada etapa de la misión se cumpla de forma ordenada.

Flujo de Información El sistema Nav2 emplea transformaciones de TF (`TF Transforms`) para mantener la relación entre el robot y su entorno en tiempo real, y utiliza mensajes de `OccupancyGrid` para representar el mapa, que es actualizado constantemente con datos de sensores. Además, los comandos de velocidad (`cmd_vel`) generados por el Controller Server son enviados al controlador base del robot para ejecutar el movimiento. Esta arquitectura modular permite al robot realizar tareas de navegación complejas y reaccionar ante cambios en el entorno de manera rápida y eficiente.

1.4.7 ROS

El Robot Operating System (ROS) es un marco de software ampliamente utilizado en robótica que facilita el desarrollo de aplicaciones complejas. ROS ofrece una estructura modular que incluye nodos, mensajes y servicios, permitiendo la comunicación eficiente entre diferentes componentes del robot [5]. Su popularidad en proyectos robóticos se debe a su flexibilidad y a la extensa comunidad de desarrolladores, quienes aportan paquetes especializados. Algunos paquetes útiles incluyen SLAM Toolbox [37] para localización y mapeo e Nav2 para navegación autónoma [36]. ROS es compatible con simuladores avanzados como Gazebo [38], PyBullet [39] y NVIDIA Isaac Sim [40], que permiten probar

algoritmos en entornos virtuales antes de implementarlos físicamente, reduciendo costos y riesgos. Esta capacidad de simular y luego implementar, junto con el soporte de una comunidad activa, hace de ROS una herramienta esencial en el diseño, control y simulación de robots.

Capítulo 2

2. Metodología

La metodología aborda desde la definición de requerimientos iniciales hasta la validación del sistema en un entorno de simulación, empleando herramientas tecnológicas como el Robot Operating System (ROS) y el simulador Gazebo.

2.1 Identificación del problema

El problema identificado radica en las limitaciones inherentes de la vigilancia realizada exclusivamente por humanos en instalaciones críticas. Factores como la fatiga y la distracción en turnos largos o nocturnos comprometen la seguridad. Por ello, se propuso un sistema autónomo basado en un robot cuadrúpedo capaz de realizar patrullajes constantes, adaptándose dinámicamente a cambios en el entorno.

2.2 Análisis de requerimientos

Para el diseño del sistema, fue fundamental realizar un análisis exhaustivo de requerimientos técnicos y operativos, los cuales se detallan en la Tabla 2.1.

Tabla 2.1.

Requerimientos del sistema

Concepto	**Descripción**
Detección precisa	Sensor LiDAR capaz de detectar objetos en un radio mínimo de 5 metros con alta resolución.
Capacidad de respuesta	Reacción ante cambios en el entorno.
Navegación autónoma	Algoritmos en ROS optimizados para mapeo en tiempo real y adaptación a obstáculos.
Condiciones ambientales	Operación en entornos de iluminación moderada y superficies regulares.
Procesamiento eficiente	Algoritmos optimizados para hardware de capacidad moderada.
Estabilidad en locomoción	Antes de implementar navegación autónoma, el robot debe ser capaz de mantener estabilidad durante la marcha, caminando de forma controlada y segura.

Los requerimientos funcionales se enfocaron en asegurar la capacidad básica del robot para moverse de manera estable y responder eficientemente en tareas teleoperadas y autónomas, mientras que las restricciones consideraron las limitaciones ambientales y computacionales del sistema.

2.3 Soluciones

2.3.1 Alternativas para el controlador

Basándonos en los requerimientos previamente definidos y el marco teórico desarrollado, se identificaron las siguientes alternativas para implementar el sistema de control:

- **Alternativa 1:**Locomoción mediante Aprendizaje por Refuerzo con SLAM. Utiliza redes neuronales para optimizar la locomoción mediante prueba y error, mejorando estabilidad y eficiencia. SLAM permite generar mapas y planificar rutas en tiempo real. Es flexible y adaptable, pero requiere entrenamiento intensivo y alta capacidad computacional.
- **Alternativa 2:** Locomoción mediante Controlador de Patrón Central Generador (CPG) con SLAM. Los CPG generan patrones rítmicos de movimiento, logrando locomoción fluida y estable. SLAM permite navegación autónoma en entornos desconocidos. Es eficiente para movimientos coordinados, aunque requiere ajuste preciso de parámetros.
- **Alternativa 3:** Sistema de control modular basado en Controlador Jerárquico con Modulación de Patrones y Control PID, que permita la integración gradual de funcionalidades. Comenzaría con un control manual estable y evolucionaría hacia la autonomía, asegurando compatibilidad con herramientas de simulación y algoritmos de control avanzados como planificación de rutas y SLAM.

2.3.2 Criterios de selección

Para determinar la alternativa más adecuada para la simulación y control del robot cuadrúpedo utilizando ROS, se definieron los siguientes:

Eficiencia de la locomoción: Este criterio evalúa la estabilidad del sistema de locomoción del robot en diversos entornos. Es fundamental garantizar que el robot pueda moverse de manera óptima en escenarios simulados o reales.

Facilidad de implementación: Se valora la complejidad del desarrollo e integración del sistema, priorizando alternativas compatibles con las herramientas existentes en ROS y que no requieran recursos computacionales excesivos.

Flexibilidad y escalabilidad: Este criterio mide qué tan adaptable es la solución para incorporar funcionalidades adicionales, como nuevos algoritmos de planificación o sensores avanzados, asegurando una evolución continua del sistema.

Robustez y confiabilidad: Analiza la capacidad del sistema para responder ante fallos de hardware o software, garantizando el correcto funcionamiento del robot en condiciones adversas.

Curva de aprendizaje y mantenimiento: Evalúa qué tan accesible es la alternativa para desarrolladores y usuarios, así como el esfuerzo requerido para mantener el sistema actualizado y funcional.

Riesgos y viabilidad técnica: Se examinan los riesgos asociados con cada alternativa, incluyendo el nivel de madurez de las tecnologías propuestas y las limitaciones técnicas que puedan surgir durante su implementación.

Tabla 2.2.

Ponderación de criterios de selección

Criterio	Relevancia	Ponderación
Eficiencia de la locomoción	Alta	5
Facilidad de implementación	Media	4
Flexibilidad y escalabilidad	Media	4
Robustez y confiabilidad	Alta	5
Curva de aprendizaje y mantenimiento	Baja	3
Riesgos y viabilidad técnica	Alta	5

La importancia de cada criterio se estableció según su impacto en los objetivos del proyecto, dividiéndolos en tres grupos. Los criterios críticos incluyen la eficiencia de la locomoción, la robustez y confiabilidad, así como los riesgos y la viabilidad técnica, ya que estos aspectos son fundamentales para el éxito del proyecto al influir directamente en la funcionalidad y seguridad del sistema. Los criterios intermedios, como la facilidad de implementación y la flexibilidad y escalabilidad, aportan ventajas prácticas y garantizan la sostenibilidad del desarrollo. Finalmente, los criterios complementarios, como la curva de aprendizaje y mantenimiento, aunque relevantes, tienen un impacto menor en la viabilidad técnica y funcional del sistema.

2.3.3 Matriz de decisión

Tabla 2.3.

Evaluación de alternativas para el controlador

Criterio	Alt. 1: Aprendizaje por Refuerzo	Alt. 2: CPG + SLAM	Alt. 3: Control Jerárquico + PID
Eficiencia de la locomoción (5)	3 × 5 = 15	4 × 5 = 20	5 × 5 = 25
Facilidad de implementación (4)	2 × 4 = 8	3 × 4 = 12	4 × 4 = 16
Flexibilidad y escalabilidad (4)	3 × 4 = 12	3 × 4 = 12	5 × 4 = 20
Robustez y confiabilidad (5)	3 × 5 = 15	4 × 5 = 20	5 × 5 = 25
Curva de aprendizaje y mantenimiento (3)	2 × 3 = 6	3 × 3 = 9	4 × 3 = 12
Riesgos y viabilidad técnica (5)	3 × 5 = 15	4 × 5 = 20	5 × 5 = 25
Puntaje Total	**71**	**93**	**123**

2.4 Selección de alternativas para el URDF del robot cuadrúpedo

Para seleccionar el modelo de robot más adecuado para la simulación y control, se definieron criterios que tomaron en cuenta aspectos técnicos, disponibilidad de recursos y compatibilidad con el proyecto. Las alternativas evaluadas fueron: *Hyperdog*, *Unitree Go1* y *MIT Mini Cheetah* debido a que tienen un desarrollo de muchos años y soporte de la comunidad de ROS.

2.4.1 Criterios de selección del robot cuadrúpedo

Se establecieron los siguientes criterios para realizar la evaluación:

Disponibilidad de documentación y soporte técnico: Este criterio evaluó la cantidad y calidad de información disponible, como artículos científicos, tesis y manuales técnicos, que facilitaran el desarrollo e integración.

Madurez tecnológica: Analizó el tiempo que cada robot había estado en desarrollo, considerando la experiencia acumulada y la confiabilidad de sus componentes en simulación y aplicaciones reales.

Compatibilidad con simulación: Valoró qué tan bien el URDF de cada robot se comportaba en entornos de simulación como Gazebo, incluyendo estabilidad física y realismo de los modelos declarados.

Acceso a recursos abiertos: Este criterio midió la accesibilidad del código, repositorios públicos y datos asociados al robot, favoreciendo alternativas con mayor apertura y transparencia.

2.4.2 Evaluación de alternativas del cuadrúpedo.

Se utilizó un sistema de ponderación en el que cada criterio recibió un puntaje basado en su relevancia para el proyecto. Posteriormente, se evaluaron las alternativas de acuerdo con estos criterios que se oberva en la Tabla 2.4.

Tabla 2.4.

Evaluación de alternativas para el URDF del robot cuadrúpedo.

Criterio	**Hyperdog**	**Unitree Go1**	**MIT Mini Cheetah**
Disponibilidad de documentación (5)	2 × 5 = 10	3 × 5 = 15	5 × 5 = 25
Madurez tecnológica (4)	2 × 4 = 8	3 × 4 = 12	5 × 4 = 20
Compatibilidad con simulación (5)	2 × 5 = 10	4 × 5 = 20	5 × 5 = 25
Acceso a recursos abiertos (4)	3 × 4 = 12	2 × 4 = 8	5 × 4 = 20
Puntaje Total	**40**	**55**	**90**

El análisis mostró que el MIT Mini Cheetah fue la mejor opción, obteniendo el puntaje más alto gracias a las siguientes razones:

El **Mini Cheetah** ha sido ampliamente estudiado y desarrollado por investigadores del MIT y otros laboratorios, contando con abundante documentación en artículos científicos, repositorios y tesis, lo que facilita su implementación. Su **madurez tecnológica** se refleja en los años de desarrollo, permitiendo la identificación y resolución de problemas de diseño para garantizar estabilidad y confiabilidad en simulaciones y aplicaciones reales. En cuanto a su **compatibilidad con simulación**, su URDF está bien optimizado para entornos como Gazebo y PyBullet, con físicas bien definidas que evitan problemas como resbalones o comportamientos inestables. Además, su **acceso a recursos abiertos** es una ventaja significativa frente al Unitree Go1, cuya documentación es limitada y mayormente en chino, mientras que el Mini Cheetah cuenta con repositorios accesibles, facilitando su integración y personalización.

2.5 Etapas del diseño

Figura 2.1.

Proceso de diseño para la elaboración de la solución

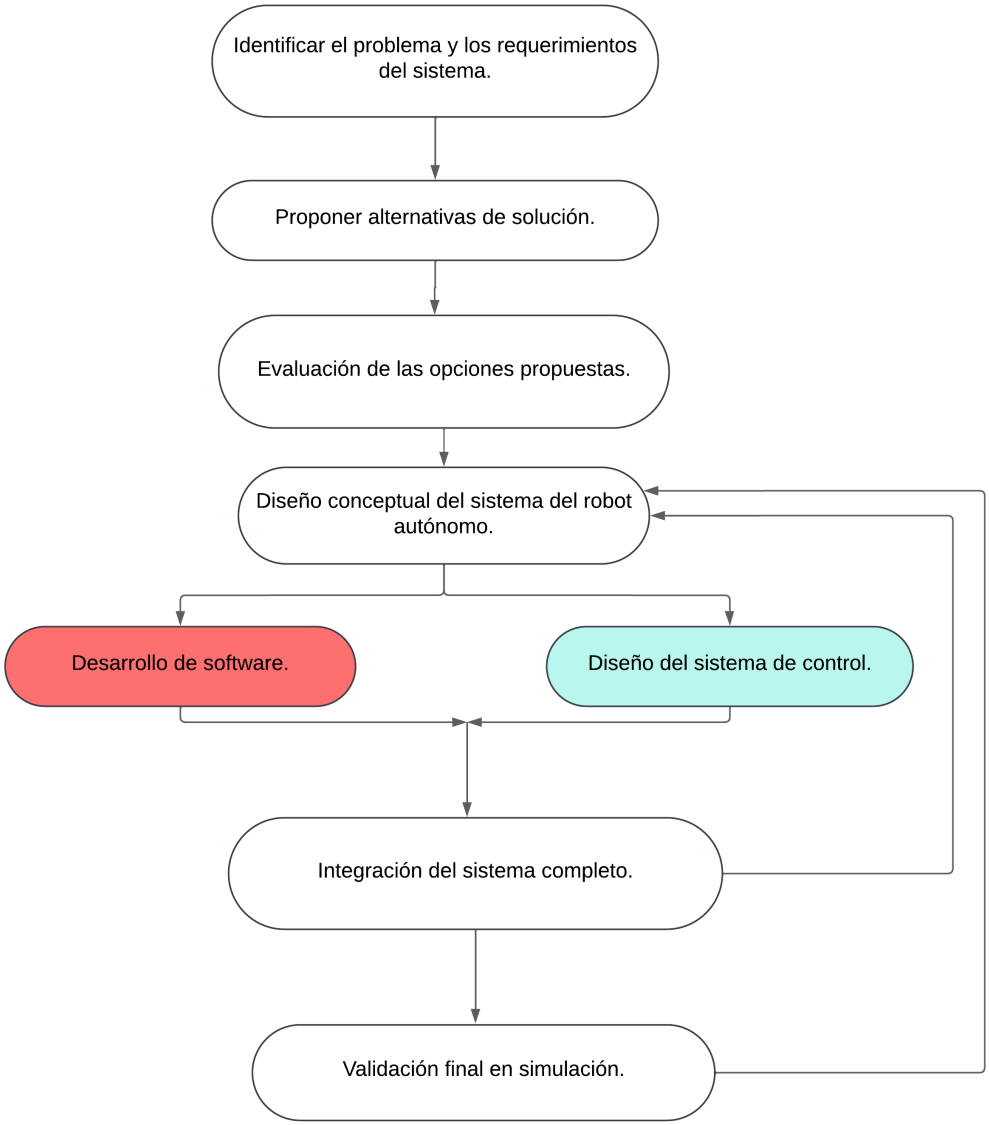


Figura 2.2.

Proceso de diseño del desarrollo del software

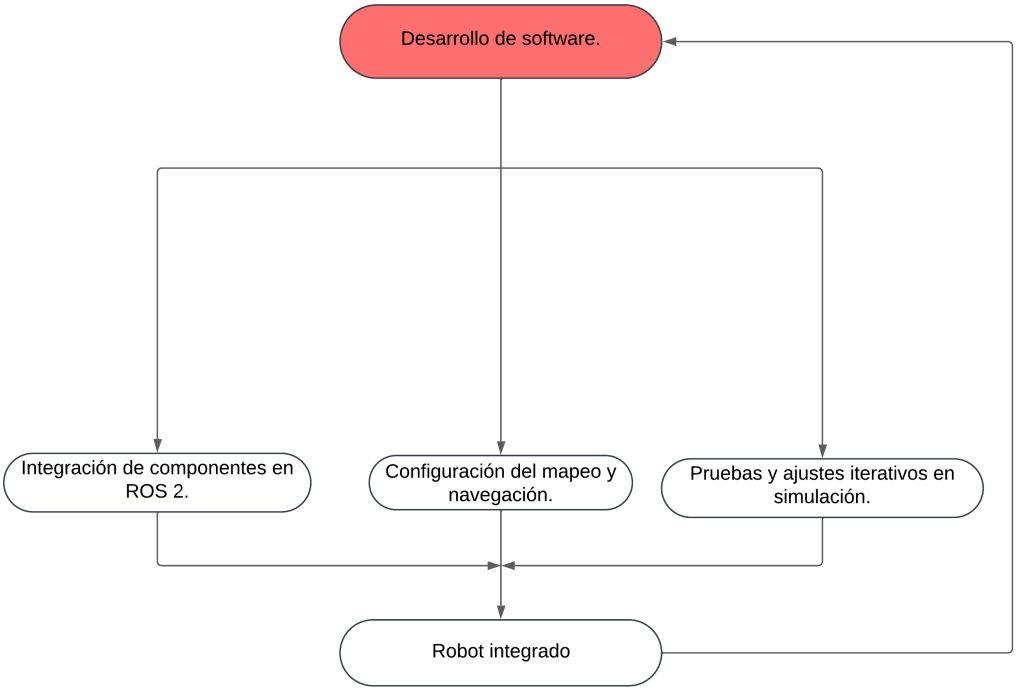
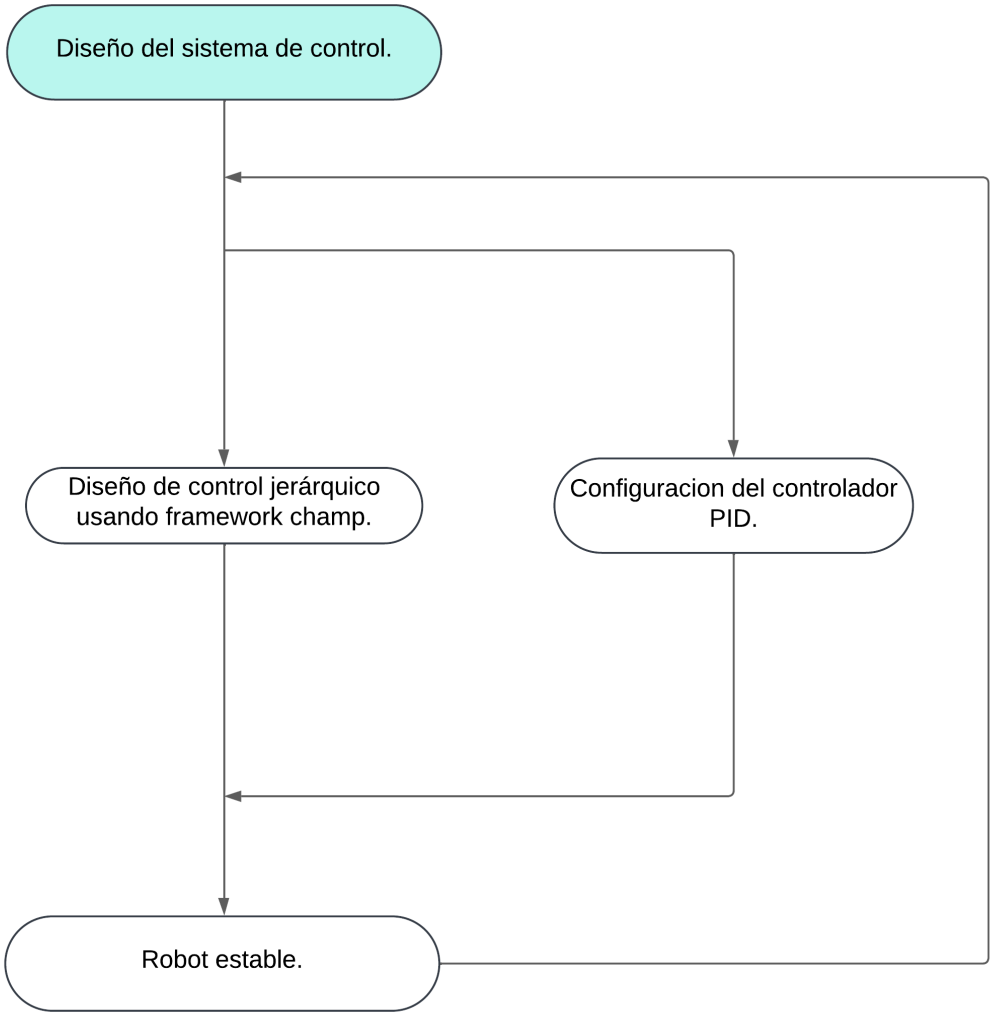


Figura 2.3.

Proceso de diseño del sistema de control



2.6 Diseño Conceptual

El diseño conceptual del sistema se centra en la simulación de un robot cuadrúpedo multiservicio utilizando ROS 2 y Gazebo. Este diseño busca validar algoritmos de control y navegación autónoma en un entorno virtual seguro y modular, optimizando el desarrollo y análisis de funcionalidades clave del robot.

2.6.1 Propósito del Diseño

El propósito es establecer una arquitectura preliminar que permita evaluar el rendimiento del robot en tareas de patrullaje y evasión de obstáculos, integrando percepción, planificación y control en un entorno simulado.

2.6.2 Componentes Principales

El sistema cuenta con varios componentes fundamentales que permiten la simulación y el control del robot cuadrúpedo. El modelo del robot, simulado en URDF/Xacro, representa un cuadrúpedo con articulaciones funcionales y configuraciones dinámicas que replican su comportamiento realista. Para la percepción del entorno, se utiliza un único sensor LiDAR, encargado de proporcionar datos para la generación de mapas ocupacionales. El entorno simulado en Gazebo incluye escenarios con terrenos planos y obstáculos configurables, diseñados específicamente para pruebas de estabilidad y navegación. Finalmente, la arquitectura en ROS 2 organiza el sistema en nodos que gestionan la percepción, la navegación y el control, asegurando una estructura modular y una comunicación eficiente entre los distintos componentes.

2.6.3 Interacción General

El sensor LiDAR captura datos del entorno que son procesados para generar un mapa ocupacional. Este mapa se utiliza para planificar rutas y calcular comandos de control, que se ejecutan en las articulaciones del robot, permitiendo la locomoción y la reacción a cambios en el entorno.

2.6.4 Ventajas del Diseño

El diseño del sistema ofrece múltiples beneficios que facilitan su desarrollo y evaluación. Proporciona una base flexible y modular que permite analizar el rendimiento del robot en distintos escenarios. Al operar exclusivamente en simulación, posibilita iteraciones rápidas y seguras sin riesgo de daños físicos. Además, optimiza el desarrollo de algoritmos al permitir pruebas controladas en un entorno virtual.

2.7 Diseño de Control

El diseño de control del robot cuadrúpedo se basa en un sistema jerárquico que utiliza controladores basados en esfuerzo para cada articulación. Este enfoque permite garantizar movimientos estables y precisos, optimizando el rendimiento del robot en entornos simulados.

2.7.1 Propósito del Control

El propósito del diseño de control es proporcionar estabilidad, precisión y adaptabilidad en los movimientos del robot cuadrúpedo, permitiendo una locomoción fluida y una rápida respuesta ante cambios en el entorno.

2.7.2 Estructura del Control Jerárquico

El sistema de control se organiza en tres niveles principales que trabajan de manera coordinada para garantizar el movimiento preciso del robot cuadrúpedo. En primer lugar, el

control de trayectorias genera las posiciones y orientaciones deseadas para las patas, considerando los objetivos de locomoción establecidos. Luego, el motor de cinematica inversa convierte esas coordenadas en las posicianes angulares necesarias para cada articulación. Finalmente, el control local de articulaciones emplea controladores PID en cada articulación, incluyendo la cadera (hip), el muslo (thigh) y la pantorrilla (calf), con el objetivo de minimizar el error entre las posiciones deseadas y las actuales, asegurando un seguimiento preciso y estable del movimiento.

2.7.3 Control PID por Articulación

Cada articulación del robot utiliza un controlador PID ajustado específicamente para garantizar un desempeño estable y eficiente. La ecuación utilizada es:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{2.1}$$

Donde:

- $u(t)$ es la señal de control, que se traduce en esfuerzo aplicado a la articulación.
- $e(t)$ es el error, definido como la diferencia entre la posición deseada y la posición actual.
- K_p, K_i, K_d son las ganancias proporcional, integral y derivativa, respectivamente, que fueron ajustadas durante la simulación.

El ajuste de los parámetros PID (K_p, K_i, K_d) se realizó de forma iterativa en simulación mediante un proceso de prueba y error, evaluando métricas como la estabilidad del robot, la minimización del error de seguimiento y la respuesta ante perturbaciones en el entorno. Para ello, se analizaron los datos del **IMU**, específicamente las variaciones en los ángulos de **roll** y **pitch**, con el objetivo de optimizar la estabilidad y reducir oscilaciones en la locomoción del robot.

2.7.4 Arquitectura del Controlador de locomocion utilizando el framework CHAMP

CHAMP implementa un controlador modular que permite gestionar las trayectorias y movimientos de un robot cuadrúpedo de manera eficiente y adaptable. La arquitectura mostrada en la imagen puede dividirse en varios bloques funcionales, cada uno de los cuales desempeña un rol crucial en el proceso de locomoción. A continuación, se describe cada componente y su función:

1. Body Controller (Controlador del Cuerpo)

La entrada de este bloque es la pose corporal del robot, que incluye la posición y orientación deseadas del cuerpo. Su función es generar las posiciones de referencia iniciales de los pies con respecto al marco base del robot, garantizando el equilibrio y asegurando que los pies comiencen las trayectorias desde posiciones adecuadas para cumplir los objetivos del movimiento. A partir de estas referencias, el sistema planifica los pasos necesarios para alcanzar el movimiento deseado. Como salida, se obtienen los puntos de inicio para las trayectorias de las patas.

2. Phase Generator (Generador de Fases)

Las entradas de este bloque incluyen los retardos de fase, que representan el desfase entre las patas para garantizar movimientos suaves y coordinados, y la velocidad deseada del robot, tanto lineal como angular. Su función es dividir el ciclo de locomoción en fases para cada pata, evitando que todas se muevan simultáneamente y generando un patrón de marcha estable. Además, define el momento exacto en el que cada pata debe estar en fase de soporte, en contacto con el suelo, o en fase de balanceo, en el aire. Como salida, se generan indicaciones temporales para el planeador de trayectorias (Trajectory Planner).

3. Leg Transformer (Transformador de Patas)

La entrada de este bloque consiste en las velocidades deseadas, tanto lineales como angulares. Su función es calcular la longitud y rotación de las trayectorias para cada pata en función de la velocidad, lo que permite adaptar dichas trayectorias al movimiento global del robot y asegurar que las patas tracen un camino adecuado para alcanzar la velocidad deseada. Como salida, se obtienen las longitudes y rotaciones ajustadas para las trayectorias.

4. Trajectory Planner (Planeador de Trayectorias)

Los componentes de este bloque incluyen la fase de balanceo (Swing), en la que la pata se encuentra en el aire y sigue una trayectoria definida por una curva Bézier de 12 puntos para elevarse y descender de forma eficiente, y la fase de soporte (Stance), donde la pata permanece en contacto con el suelo, proporcionando estabilidad y fuerza de empuje al robot mediante una trayectoria con un patrón de onda sinusoidal. La función de este módulo es alternar entre estas fases para cada pata según las indicaciones del generador de fases. Como salida, se obtienen las posiciones exactas de los pies en el espacio tridimensional en cada instante del ciclo de marcha.

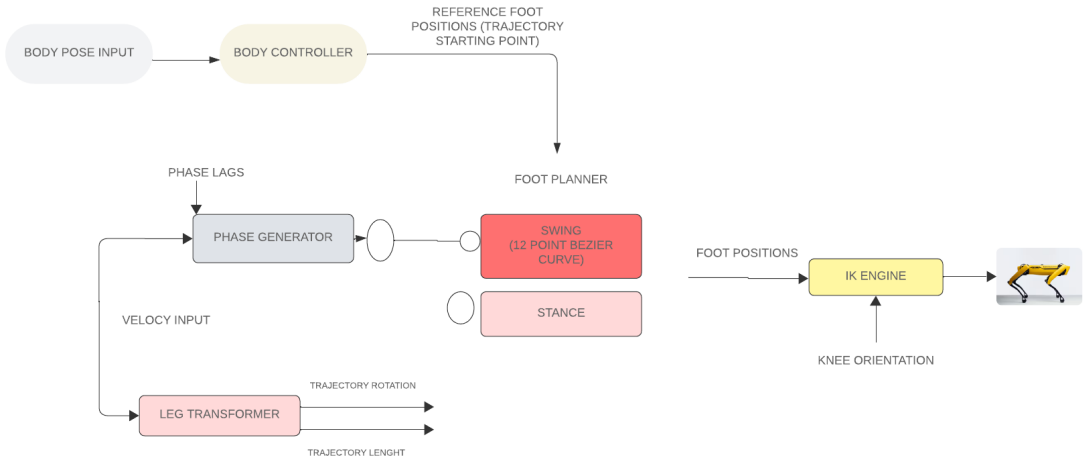
5. IK Engine (Motor de Cinemática Inversa)

La entrada de este bloque consiste en las posiciones finales de las patas generadas por el planeador de trayectorias. Su función es resolver las ecuaciones de cinemática inversa para calcular los ángulos articulares necesarios en las patas del robot, teniendo en cuenta la orientación de las rodillas y las longitudes de los eslabones definidos en la configuración del robot. Además, diferentes robots cuadrúpedos pueden requerir adaptaciones en este bloque, ya que la cinemática varía según su diseño. Finalmente, la salida son los ángulos articulares

que se envían a los actuadores para ejecutar el movimiento.

Figura 2.4.

Sistema de control



2.7.5 Configuración del controlador

Para controlar de manera eficiente el robot cuadrúpedo elegido utilizando el framework CHAMP, fue necesario configurar parámetros específicos como las **ganancias del controlador**, los **límites de esfuerzo**, las **constantes PID** y la **cinemática de las articulaciones**. Estos parámetros conectaron el software con el hardware o simulador, permitiendo un control preciso del movimiento.

Estas configuraciones definieron aspectos críticos como la estructura física del robot (articulaciones y enlaces), los controladores responsables de su movimiento y las ganancias que aseguraron un comportamiento dinámico estable. A través de estos archivos de configuración, se establecieron las bases para que el sistema ROS 2 pudiera interpretar y gestionar el modelo del robot en la simulación. A continuación, se describe cada uno de los archivos que se utilizaron para configurar el controlador, junto con su propósito y estructura específica.

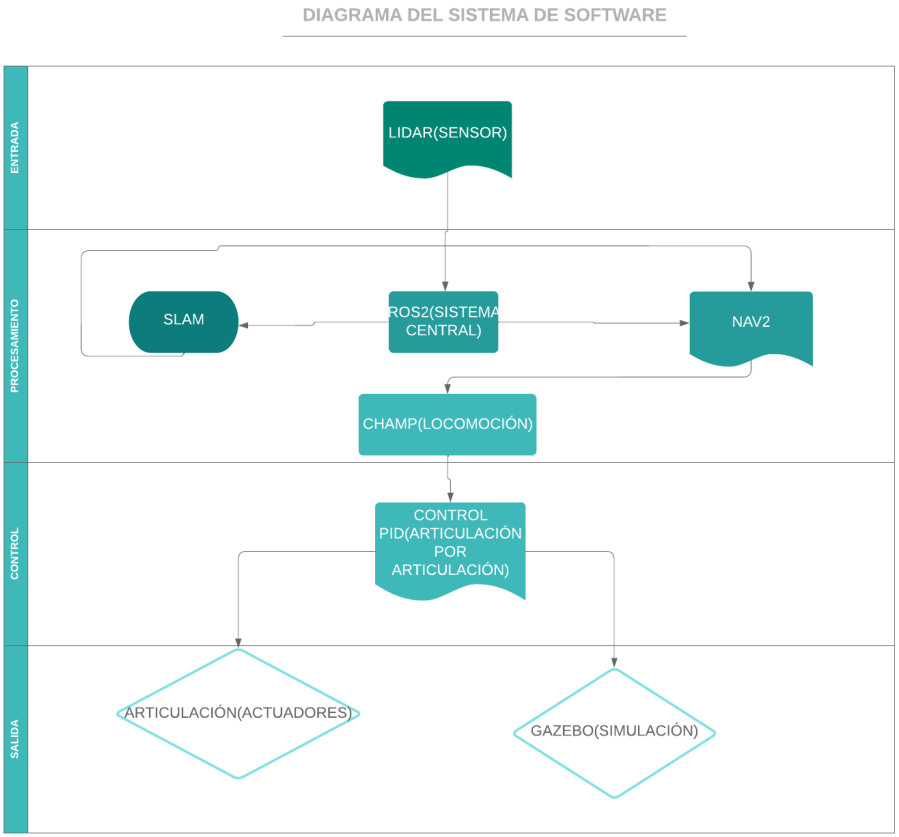
2.8 Desarrollo del Software

El desarrollo del software en este proyecto integra distintos módulos que trabajan de manera conjunta para garantizar la funcionalidad y precisión del robot cuadrúpedo en simulación. La Figura 2.5 presenta la arquitectura general del sistema, destacando cómo las entradas (sensores) se procesan en el marco de ROS 2, pasando por módulos de navegación, control y locomoción, hasta alcanzar la simulación y los actuadores. Esta estructura jerárquica organiza los componentes del sistema en capas funcionales: entrada, donde sensores como el LiDAR proporcionan datos del entorno; procesamiento, con módulos de navegación autónoma y SLAM para localización y planificación; control, que incluye controladores PID y el framework CHAMP para la locomoción; y salida, representada por los

actuadores y la simulación en Gazebo para validar los movimientos del robot.

Figura 2.5.

Arquitectura del sistema. El diagrama muestra cómo las entradas, el procesamiento, el control y la salida interactúan en el sistema.



En las secciones siguientes, se describe cómo cada componente fue configurado y adaptado para cumplir con los requisitos del proyecto.

2.8.1 Configuración Geométrica del Robot

El archivo `quadruped_description.h` se utilizó para definir una descripción estática de la geometría del robot cuadrúpedo. Este archivo proporcionó una forma eficiente de inicializar la configuración geométrica del robot mediante valores codificados directamente en el código fuente, eliminando la necesidad de utilizar un archivo URDF externo.

El propósito principal de este archivo fue establecer las posiciones y orientaciones relativas de las diferentes partes del robot, como la cadera, pierna superior, pierna inferior y pie, en el espacio tridimensional. Para ello, se especificaron tanto las coordenadas relativas x, y, z en metros como las orientaciones relativas $roll, pitch, yaw$ en radianes.

La función `loadFromHeader` fue utilizada para asignar manualmente estas posiciones y orientaciones a las estructuras definidas en la clase `champ::QuadrupedBase`. Este método permitió configurar la geometría del robot de forma estática, asegurando que estuviera preparada para ser empleada por el controlador sin depender de configuraciones externas.

El código organizó las patas del robot en cuatro grupos principales: `lf` (pata delantera izquierda), `rf` (pata delantera derecha), `lh` (pata trasera izquierda) y `rh` (pata trasera derecha). Para cada una de estas patas, se definieron las posiciones relativas de componentes específicos, como `hip` (cadera), `upper_leg` (pierna superior), `lower_leg` (pierna inferior) y `foot` (pie). Estas configuraciones se implementaron mediante el método `setOrigin`, que asignó coordenadas y orientaciones en el espacio 3D utilizando el formato:

```
setOrigin(x, y, z, roll, pitch, yaw);
```

El uso de este enfoque ofreció ventajas significativas. Por un lado, simplificó y aceleró

la configuración al eliminar la necesidad de depender de archivos URDF, haciéndolo ideal para pruebas rápidas o diseños estáticos. Por otro lado, permitió una personalización sencilla de los valores geométricos directamente en el código, facilitando ajustes rápidos en el diseño del robot y su adaptación a diferentes entornos o prototipos.

2.8.2 Configuración del Mapeo de Articulaciones y Links en YAML

En la configuración del controlador, se utilizó un archivo en formato YAML tanto para definir el mapeo de articulaciones como el de links.

Configuración del Mapeo de Articulaciones Cada pata fue representada como un grupo con un orden jerárquico de articulaciones (`FL_hip_joint`, `FL_thigh_joint`, etc.). Los nombres de las articulaciones incluían prefijos descriptivos (`FL` para la delantera izquierda, `FR` para la delantera derecha, etc.), y este mapeo fue crucial para la interoperabilidad entre el simulador Gazebo y el controlador en ROS, permitiendo un control preciso y estable del robot.

Configuración del Mapeo de Links Similar al mapeo de articulaciones, el archivo YAML definió los links del robot (`base_link`, `FL_hip`, `FL_thigh`, etc.). Este mapeo era esencial para que los nodos ROS identificaran las partes físicas del robot y se utilizaba en simuladores como Gazebo y herramientas como RViz para una visualización correcta y para asegurar las transformaciones cinemáticas y dinámicas del robot.

2.8.3 Configuración de los Parámetros de Locomoción del Robot

Para implementar un control de locomoción eficiente en el robot cuadrúpedo, se configuraron los parámetros del controlador en función de los requisitos del proyecto. Esta configuración buscó garantizar estabilidad, precisión en el movimiento y adaptabilidad a diferentes terrenos, priorizando la eficiencia en las fases de soporte y balanceo de las patas.

La orientación de las rodillas del robot se definió apuntando hacia atrás, ya que así está diseñado el robot cuadrúpedo Mini Cheetah. Esto permite que el motor de cinemática inversa de Champ realice los cálculos necesarios para mover las articulaciones del robot de manera eficiente. La opción `"pantograph_leg"` se estableció como `false` para evitar que las patas simulen un sistema pantográfico, ya que la geometría del robot no requería esta característica para lograr un movimiento natural.

El parámetro `"odom_scaler"` se ajustó a 0.9 para compensar posibles errores en la odometría, lo que permitió mantener una correspondencia más precisa entre la posición estimada y la real del robot durante la navegación. Las velocidades máximas se definieron cuidadosamente: la velocidad lineal máxima en el eje X se configuró en 0.5 m/s, la velocidad lateral en el eje Y en 0.25 m/s, y la velocidad angular máxima en 1.0 rad/s. Estos valores fueron seleccionados para garantizar movimientos suaves y controlados, adaptándose a las capacidades mecánicas del robot.

La altura de balanceo (`"swing_height"`) se fijó en 0.05 m, lo que permitió que las patas se elevaran lo suficiente durante la fase de balanceo para evitar colisiones con el suelo sin comprometer la eficiencia energética. La profundidad de la trayectoria en la fase de soporte (`"stance_depth"`) se estableció en 0.0, asegurando que el movimiento en contacto con el suelo fuera firme y estable. La duración de la fase de soporte (`"stance_duration"`) se configuró en 0.3 segundos, optimizando el tiempo de contacto para garantizar estabilidad durante la marcha.

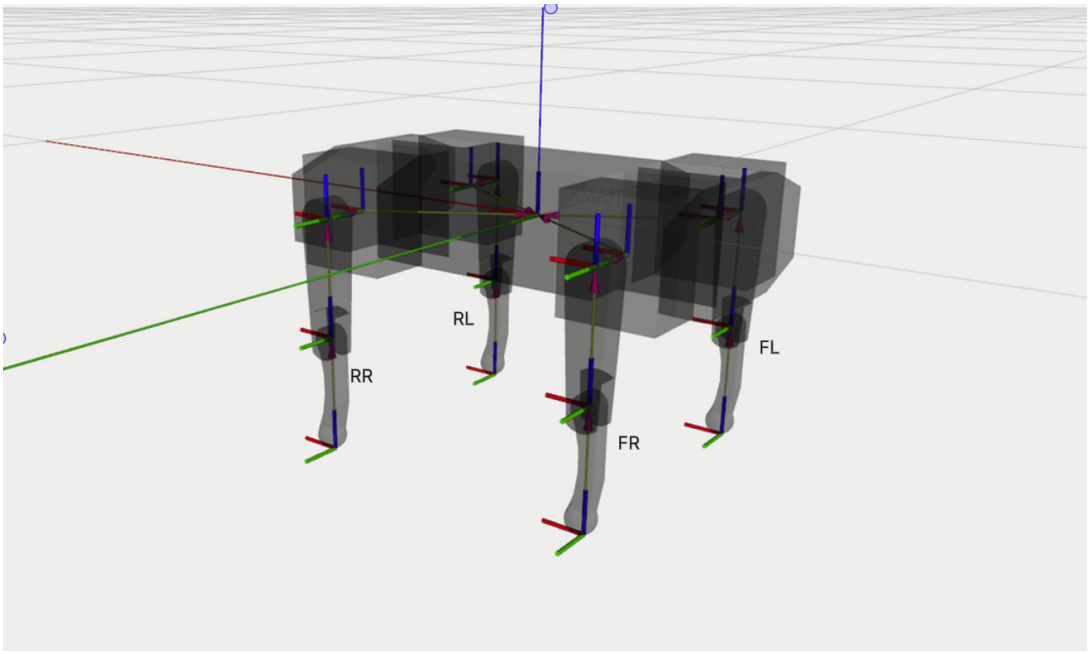
La altura nominal del robot (`"nominal_height"`) se ajustó a 0.3 m, lo que mantuvo un centro de masa adecuado para maximizar la estabilidad en diferentes terrenos. Finalmente, el desplazamiento del centro de masa en el eje X (`"com_x_translation"`) se configuró en 0.0, ya que no fue necesario ajustar el balanceo dinámico del robot debido a un diseño estructural equilibrado.

2.8.4 Configuración del control de bajo nivel

Para el control de bajo nivel del cuadrupedo se utilizo el framework ros2_Control y fue configurado utilizando un archivo YAML. Este archivo define los parámetros del controlador y las interfaces necesarias para manejar las articulaciones del robot. Se estableció la configuración del `controller_manager`, utilizando el tiempo de simulación (`use_sim_time: True`) y una frecuencia de actualización de 250 Hz (`update_rate: 250`). El controlador `joint_states_controller`, tipo `joint_state_broadcaster/JointStateBroadcaster`, difunde el estado actual de las articulaciones (posición, velocidad y esfuerzo) como un mensaje de tipo `sensor_msgs/JointState`, lo cual es esencial para otros componentes del sistema como la visualización en RViz y los nodos de control.El controlador `joint_group_effort_controller`, tipo `joint_trajectory_controller/JointTrajectoryController`, permite enviar trayectorias de esfuerzo utilizando la interfaz de comando de esfuerzo (effort). Las ganancias PID fueron ajustadas para cada articulación del robot (`FL`, `FR`, `RL`, `RR`) para asegurar un control preciso y estable.

Figura 2.6.

Patas del robot



2.8.5 Configuración del URDF

El modelo URDF del robot cuadrúpedo fue configurado cuidadosamente para cumplir con las directrices del controlador del framework CHAMP, asegurando compatibilidad y optimización en el control de la locomoción. Para evitar fragmentación en los diferentes marcos de referencia del robot, se eliminaron rotaciones en todos los marcos conjuntos, estableciendo las propiedades de ”rpy” (roll, pitch, yaw) en cero. Las articulaciones de las caderas se configuraron para rotar únicamente en el eje X, mientras que las articulaciones superiores e inferiores de las patas se ajustaron para rotar en el eje Y, respetando la anatomía funcional del robot.

Además, los orígenes de las mallas de los actuadores se posicionaron en el centro de rotación de las articulaciones, lo cual garantizó un comportamiento dinámico consistente en el modelo. En la posición inicial del robot, todas las articulaciones fueron configuradas para

mantener las patas completamente extendidas hacia el suelo, asegurando que desde una vista frontal y sagital las patas permanecieran perpendiculares al suelo. Esto proporcionó un punto de referencia claro para evaluar las posiciones relativas del robot durante la locomoción.

En el URDF también se integraron los sensores necesarios para habilitar capacidades de mapeo y navegación, incluyendo un sensor LiDAR y un sensor IMU. El LiDAR fue incorporado para proporcionar datos de percepción del entorno y permitir la construcción de mapas en tiempo real, mientras que el sensor IMU se agregó para medir la orientación y aceleración del robot, mejorando la estabilidad y la precisión en la locomoción. Estas modificaciones e integraciones garantizaron que el robot estuviera completamente equipado para navegar y mapear de manera eficiente.

2.8.6 Configuración del Sistema de Navegación Autónoma

El proyecto de navegación autónoma del robot cuadrúpedo se configuró utilizando el framework *Navigation2* (Nav2) dentro de ROS 2, adaptado para satisfacer los requisitos cinemáticos y dinámicos de la plataforma. Esta configuración permitió escalar las capacidades de locomoción existentes hacia un sistema autónomo, integrando algoritmos avanzados para la planificación global y local, localización, generación de trayectorias y evitación de obstáculos.

Localización con AMCL En el nodo de localización AMCL (Adaptive Monte Carlo Localization), se configuraron parámetros específicos para optimizar el comportamiento del robot en entornos complejos. Se utilizaron modelos probabilísticos con parámetros ajustados, como un rango máximo del láser de 100.0 metros, un rango mínimo de -1.0 metros y un modelo basado en `likelihood_field`. Además, se establecieron configuraciones para un número de partículas entre 500 y 2000, incluyendo la capacidad de

transformar entre marcos de referencia como `map`, `odom` y `base_footprint`, ajustando tolerancias de actualización de posición y rotación. La inicialización de la posición utilizó coordenadas predeterminadas para `x`, `y` y `yaw`, con un modelo de movimiento diferencial (`nav2_amcl::DifferentialMotionModel`).

Planificación Global La planificación global utilizó el algoritmo NavFn implementado en el servidor de planeación, configurado para manejar mapas con una resolución de 0.05 metros y una tolerancia de 0.5 metros. Este algoritmo priorizó rutas costo-conscientes al integrar valores provenientes de las *costmaps* locales y globales. Las *costmaps* se actualizaron a frecuencias de 5 Hz y 2 Hz, respectivamente, y se definieron capas como `voxel_layer` e `inflation_layer` para gestionar obstáculos en 2D y 3D. Los sensores empleados incluyeron escáneres láser y cámaras 3D, integrados con temas como `/scan` y `/camera/depth/color/points`.

Control de Trayectorias Locales El control de trayectorias locales se realizó con el controlador DWB (Dynamic Window Approach), configurado con críticos personalizados para evaluar y seleccionar las mejores velocidades, evitando colisiones y maximizando la eficiencia. Los críticos incluyeron evaluaciones de alineación con la meta, distancia al objetivo y penalización por oscilaciones. El sistema se ajustó para manejar velocidades máximas de 0.4 m/s en el eje lineal y 0.75 rad/s en rotación, con aceleraciones limitadas a 2.5 m/s^2 y deceleraciones a -2.5 m/s^2. Además, el servidor de recuperación implementó comportamientos como giros en el lugar, retrocesos y esperas para resolver situaciones donde el robot pudiera quedar bloqueado.

2.8.7 Implementar la lógica de patrullaje

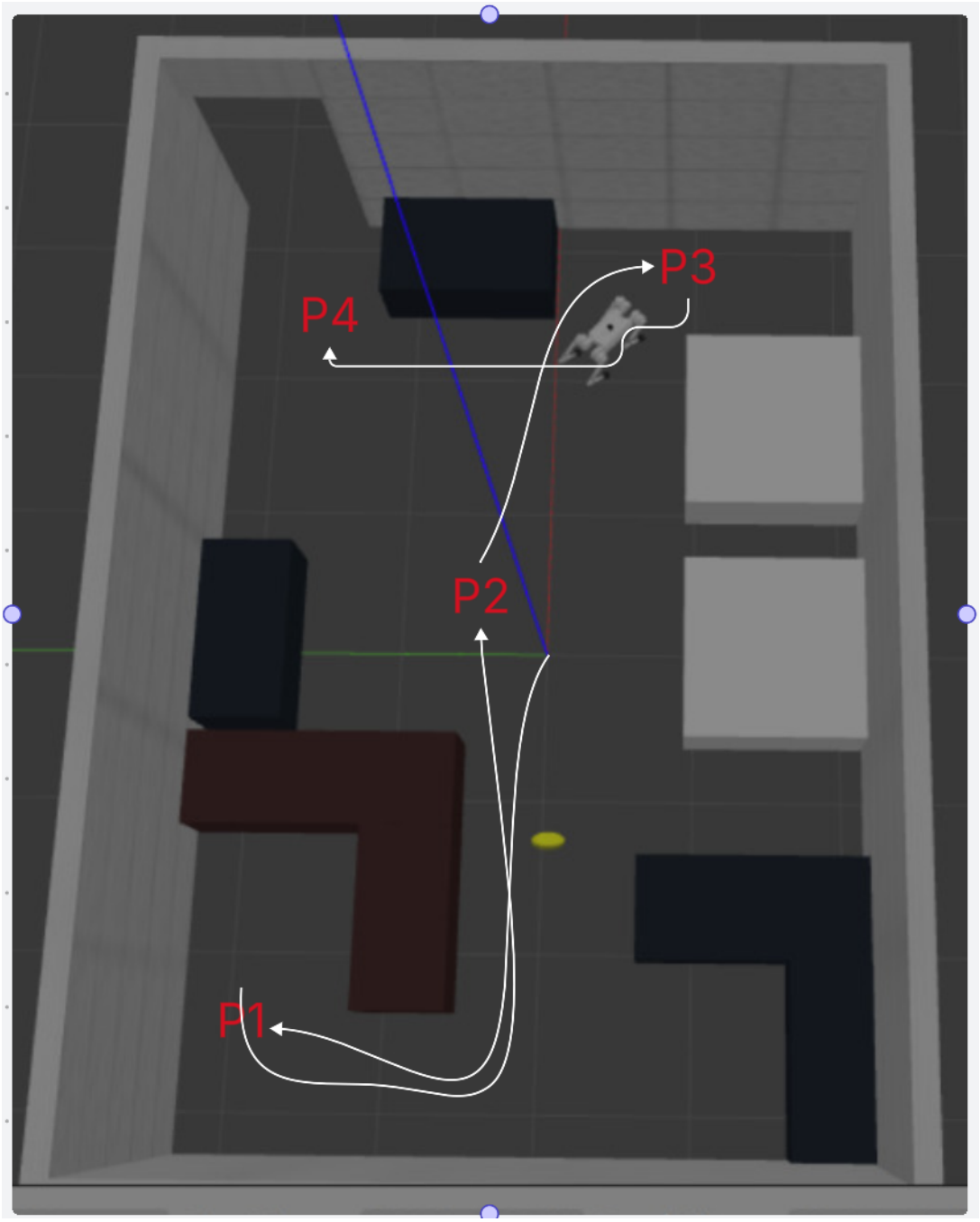
La funcionalidad de patrullaje se implementó definiendo una serie de puntos clave en el mapa, los cuales el robot debía visitar secuencialmente. Estas coordenadas se

determinaron en función de las necesidades del escenario de patrullaje y se almacenaron como una lista ordenada. Posteriormente, se desarrolló un nodo específico en ROS2 que permitió enviar comandos al robot para navegar hacia estos puntos. Este nodo utilizó la acción ”NavigateToPose” de Nav2, la cual facilitó el envío de metas secuenciales. Para cada punto, se especificó la posición y orientación del robot en el entorno, asegurando una transición suave entre las metas.

El procedimiento incluyó el cálculo de las orientaciones necesarias para cada punto, transformando los ángulos en representaciones de cuaterniones, adecuados para el control de la orientación del robot. La lógica del nodo permitió que, una vez alcanzado un punto, el sistema enviara automáticamente la siguiente meta de la lista, repitiendo el ciclo indefinidamente. Finalmente, el sistema se validó observando el comportamiento del robot en la simulación y asegurando que cumpliera con la trayectoria de patrullaje especificada de manera continua y eficiente.

Figura 2.7.

Lógica de patrullaje



2.9 Análisis de costos

2.9.1 Software

El desarrollo e implementación del sistema de navegación autónoma y control del robot cuadrúpedo se basó en software de código abierto, lo que permitió reducir significativamente los costos de desarrollo. ROS2 fue la plataforma central utilizada para la integración y comunicación de todos los bloques del sistema de control, proporcionando una arquitectura modular y flexible para la ejecución de las distintas tareas del robot. Dentro del ecosistema de ROS2, se emplearon paquetes específicos que desempeñaron roles clave en la funcionalidad del robot: Nav2, utilizado para la navegación autónoma, permitiendo la planificación de trayectorias, evasión de obstáculos y localización del robot dentro del entorno; CHAMP, responsable del control de locomoción del robot cuadrúpedo, generando los patrones de marcha y ejecutando los movimientos necesarios para desplazarse de manera estable; y ros2_control, implementado para gestionar el control de bajo nivel de las articulaciones, asegurando que las trayectorias generadas por CHAMP sean ejecutadas con precisión a través de controladores PID. Al tratarse de software de código abierto, la implementación del sistema no requirió licencias comerciales, lo que representa una ventaja significativa en términos de costos.

2.9.2 Hardware

Para una implementación en hardware real, se requiere una serie de sensores que permitan la percepción del entorno y la retroalimentación necesaria para el control de locomoción. Entre los componentes esenciales se incluyen un sensor LiDAR para la detección de obstáculos y generación de mapas del entorno, una IMU para proporcionar información sobre la orientación y aceleración del robot, y sensores de presión en las patas

para detectar el contacto con el suelo. La siguiente tabla 2.5 resume los sensores necesarios para la implementación del sistema en un robot físico:

Tabla 2.5.

Costos de hardware para la implementación en robot real

Componente	Costo (USD)
Sensor LiDAR RPLIDAR C1	80
Sensor IMU BMI160	20
Force-Sensitive Resistors (FSR 402) (4 unidades)	40

2.9.3 Costo Total

Para evaluar el costo total de la implementación del sistema de navegación y control del robot cuadrúpedo, se consideran tanto los costos de hardware como los costos de desarrollo de software en términos de horas de programación. La siguiente tabla 2.6 resume todos los costos hasta el momento:

Tabla 2.6.

Resumen de costos totales

Concepto	Costo (USD)
Sensor LiDAR RPLIDAR C1	80
Sensor IMU BMI160	20
Force-Sensitive Resistors (FSR 402) (4 unidades)	40
Horas de programación para locomoción (100 horas x 2 programadores)	840
Horas de programación para navegación (60 horas x 2 programadores)	504
Total	**1484**

Capítulo 3

3. Resultados y análisis

En este capítulo se presentan los resultados obtenidos del sistema de simulación y control diseñado para el robot cuadrúpedo multiservicio. Se realizaron experimentos en un entorno simulado que evaluaron el desempeño del robot en tareas de patrullaje y navegación autónoma en escenarios regulares. Estos experimentos tuvieron como objetivo validar la eficacia y robustez del sistema bajo diversas condiciones controladas. Se realizaron análisis cualitativos y cuantitativos para medir la estabilidad, precisión en la navegación y tiempo de respuesta del robot, permitiendo así identificar las fortalezas y áreas de mejora del diseño propuesto.

3.1 Configuración Experimental

La fase de pruebas del sistema de simulación y control para el robot cuadrúpedo multiservicio se llevó a cabo en un entorno controlado de simulación utilizando Gazebo, integrado con ROS2. Este entorno proporcionó una plataforma integral para analizar detalladamente las capacidades del robot, incluyendo la locomocion, navegación autónoma basada en SLAM y localización mediante AMCL. Las interacciones y respuestas del robot dentro del entorno simulado fueron visualizadas de manera detallada a través de la herramienta RViz, permitiendo un monitoreo preciso del desempeño en tiempo real.

3.2 Control de Bajo Nivel de las Articulaciones

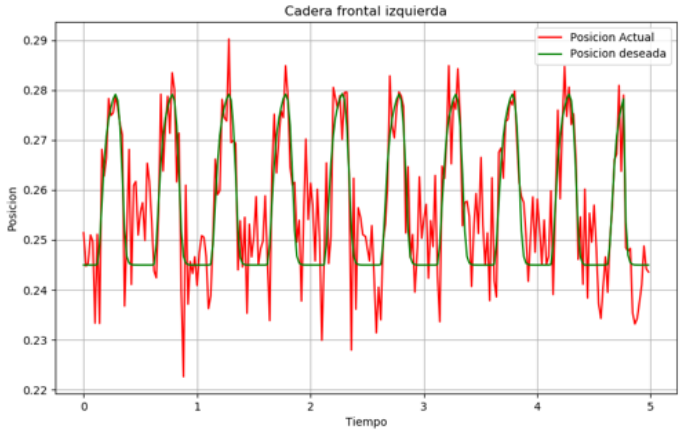
El control de bajo nivel de las articulaciones del robot cuadrúpedo se implementó utilizando el framework ros2_control, permitiendo la ejecución de comandos de movimiento generados por el controlador de alto nivel CHAMP. Este enfoque facilitó la comunicación fluida entre la planificación de trayectoria y la ejecución precisa de los movimientos en el hardware simulado.

Inicialmente, se empleó un control proporcional (P) para regular la posición de cada articulación, asegurando que estas siguieran las trayectorias planeadas por CHAMP. Sin embargo, al evaluar el desempeño del sistema, se identificaron oscilaciones y errores residuales que afectaban la estabilidad y precisión de los movimientos del robot, como se muestra en la Figura 3.1 . Para mitigar estas deficiencias, se implementó un controlador PID (Proporcional-Integral-Derivativo) en cada articulación, ajustando sus ganancias de manera iterativa para optimizar el rendimiento.

El controlador PID mejoró significativamente la respuesta dinámica del sistema, reduciendo el error de seguimiento y aumentando la estabilidad en cada articulación como se muestra en la Figura 3.2. Gracias a este ajuste, el robot logró ejecutar las trayectorias con mayor precisión y fluidez, minimizando desviaciones y mejorando la coordinación de sus movimientos. Este enfoque permitió garantizar que las acciones generadas por CHAMP fueran ejecutadas de manera efectiva, proporcionando un control robusto y adaptable para la locomoción cuadrúpeda a comparacion del control proporcional.

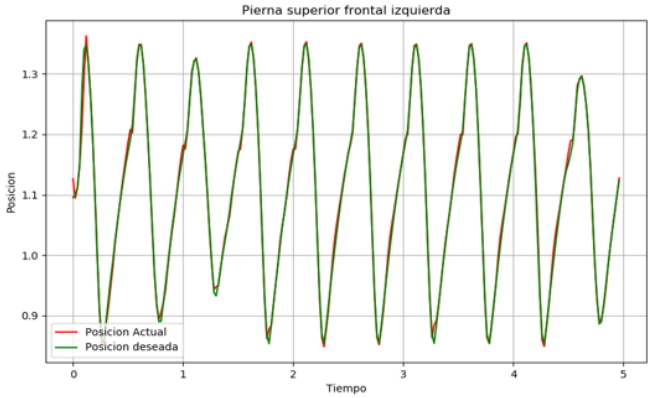
Figura 3.1.

Posición actual vs deseada de las articulaciones con control P. $k_P = 180$



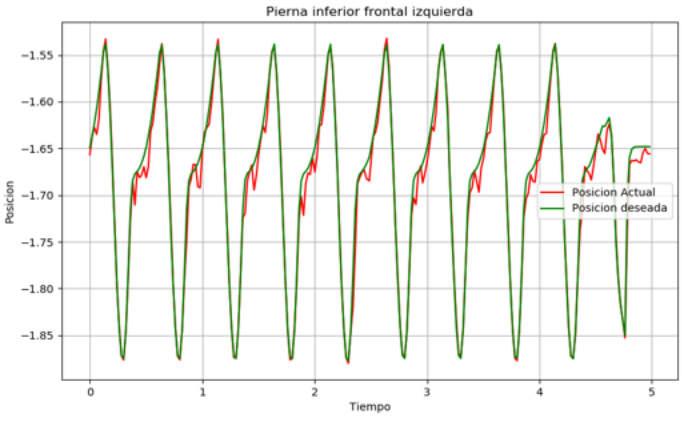
(a)

Cadera frontal izquierda



(b)

Pierna superior frontal izquierda



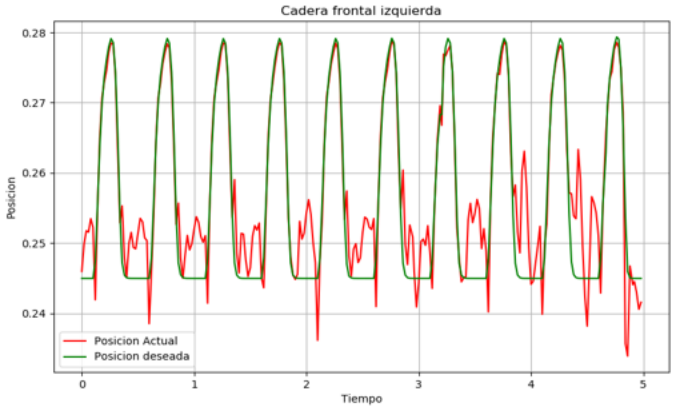
(c)

Pierna inferior frontal izquierda

Las gráficas muestran el comportamiento de las articulaciones de la pierna del robot cuadrúpedo utilizando control proporcional (kP=180). Aunque el control proporcional reduce el error entre la posición actual (línea roja) y la posición deseada (línea verde), se observa que las posiciones siguen oscilando significativamente, lo que indica que el error no se ha eliminado por completo. La oscilación y el error considerable sugieren que el valor de kP es insuficiente para estabilizar el sistema, lo que podría mejorar con la implementación de controladores más complejos, como PID.

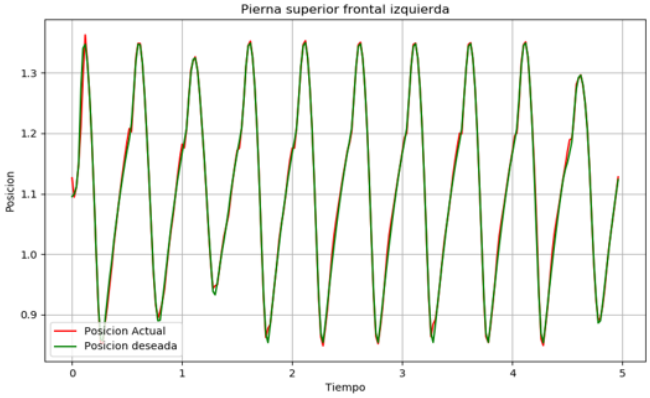
Figura 3.2.

Posición actual vs deseada de las articulaciones con control PID. $k_P = 180, k_D = 0.09, k_I = 20$



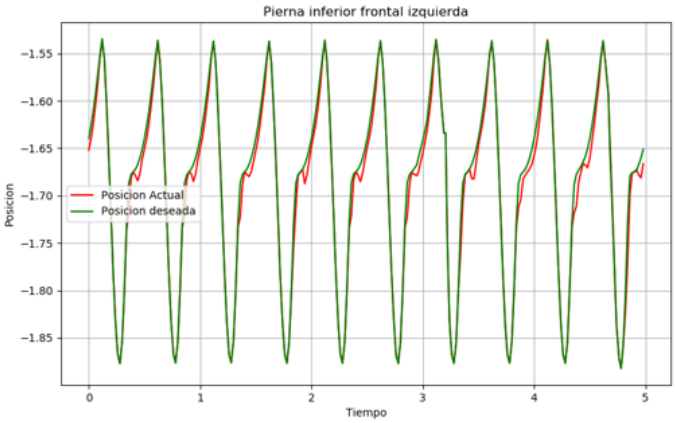
(a)

Cadera frontal izquierda



(b)

Pierna superior frontal izquierda



(c)

Pierna inferior frontal izquierda

Las gráficas muestran el comportamiento de las articulaciones de la pierna del robot cuadrúpedo bajo control PID (kP=180, kI=20, kD=0.09). En la cadera frontal izquierda, el control PID mejoró el seguimiento de la posición deseada (línea verde), pero la posicion actual (línea roja) aún presenta oscilaciones que impiden un seguimiento preciso. Esto se debe a que la cadera, al estar conectada al torso, experimenta un mayor numero de fuerzas de reacción en comparación con las demás articulaciones. En la pierna superior e inferior frontal izquierda, el seguimiento es más preciso, indicando un mejor ajuste para estas articulaciones.

3.3 Estabilad de la locomoción

Para evaluar la estabilidad de la locomoción del robot cuadrúpedo multiservicio, se analizaron los datos proporcionados por el sensor IMU, específicamente la orientación en términos de roll (inclinación respecto al eje longitudinal del robot) y pitch (inclinación respecto al eje transversal del robot). Ambos parámetros son fundamentales para determinar la estabilidad del movimiento, ya que permiten observar cómo se comporta el cuerpo del robot durante el proceso de locomoción. Durante las pruebas, se registró una media de 0.0057 radianes y una desviación estándar de 0.0351 radianes con respecto al roll. Si bien se presentaron variaciones u oscilaciones en la orientación del robot, estás se mantienen en un rango pequeño y son esperadas debido a la dinámica propia de la locomoción cuadrúpeda. Cuando el robot se desplaza, su cuerpo tiende a inclinarse ligeramente hacia el lado de la pierna que se encuentra en apoyo durante cada paso. Este comportamiento es natural en sistemas de locomoción cuadrúpeda, ya que la distribución del peso varía conforme se transfiere el soporte entre las diferentes extremidades, generando pequeñas oscilaciones laterales. La baja desviación estándar obtenida indica que estas oscilaciones son de pequeña magnitud y consistentes, lo que sugiere que el sistema logra mantener el equilibrio

de manera eficiente a pesar de los cambios temporales en la orientación. Además, la media cercana a cero refleja que, aunque existen inclinaciones durante el movimiento, el robot no presenta desviaciones significativas a largo plazo, lo cual refuerza la estabilidad general de la locomoción. Las representaciones gráficas obtenidas muestran estos resultados de forma clara: el histograma del roll revela que la mayoría de los valores se concentran cerca de cero, lo que confirma la estabilidad del sistema a pesar de las oscilaciones naturales, mientras que la gráfica temporal del roll permite observar cómo estas variaciones se comportan de manera periódica y controlada a lo largo del tiempo debido al cambio constante de apoyo entre las extremidades.

Por otro lado, el análisis del pitch aporta información adicional sobre la estabilidad en el eje transversal del robot, permitiendo evaluar cómo se comporta la inclinación hacia adelante o hacia atrás durante el desplazamiento. Durante las pruebas realizadas, se registró una media de 0.0238 radianes y una desviación estándar de 0.0200 radianes. Estas cifras reflejan que, aunque existen ligeras inclinaciones en el eje de pitch, estas son muy pequeñas y consistentes, indicando que el robot logra mantener el equilibrio de manera eficiente en este eje. Las oscilaciones observadas están relacionadas con el ciclo de locomoción, donde el cuerpo del robot tiende a inclinarse ligeramente hacia adelante al avanzar y regresa a su posición inicial con cada cambio de soporte entre las extremidades. Este patrón cíclico es inherente a sistemas de locomoción cuadrúpeda y no afecta negativamente la estabilidad general del sistema. El histograma del pitch refuerza estos hallazgos, mostrando que la mayoría de los valores están concentrados alrededor de la media, con una distribución simétrica y bien definida. Esto confirma que las oscilaciones en el eje de pitch son controladas y de baja magnitud. Asimismo, la baja desviación estándar indica que estas inclinaciones se mantienen dentro de un rango muy limitado

Figura 3.3.

Gráfica temporal de la orientación roll (radianes)

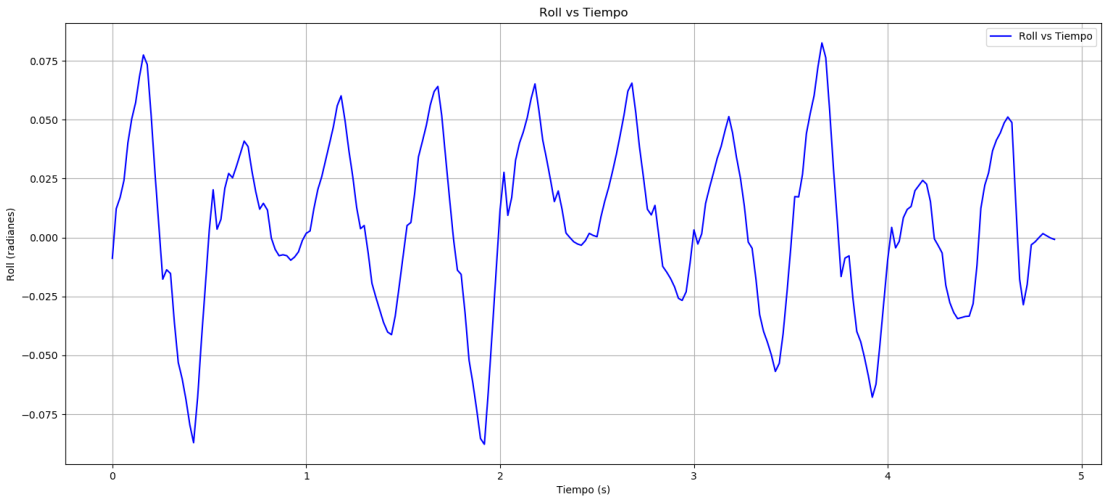


Figura 3.4.

Histograma de la orientación roll (grados)

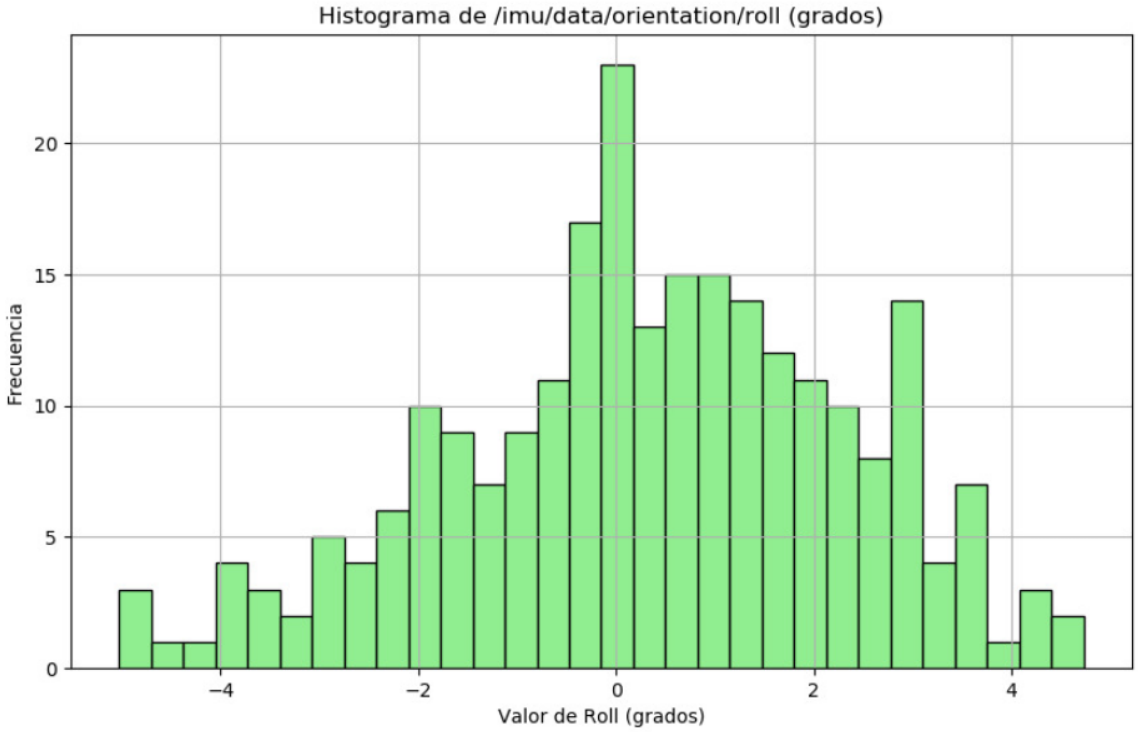


Figura 3.5.

Gráfica temporal de la orientación Pitch (radianes)

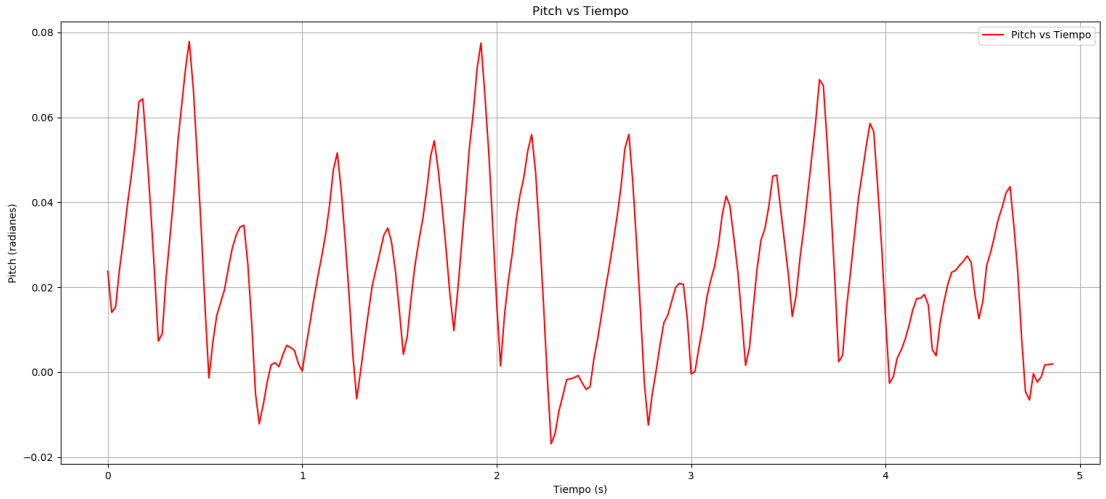
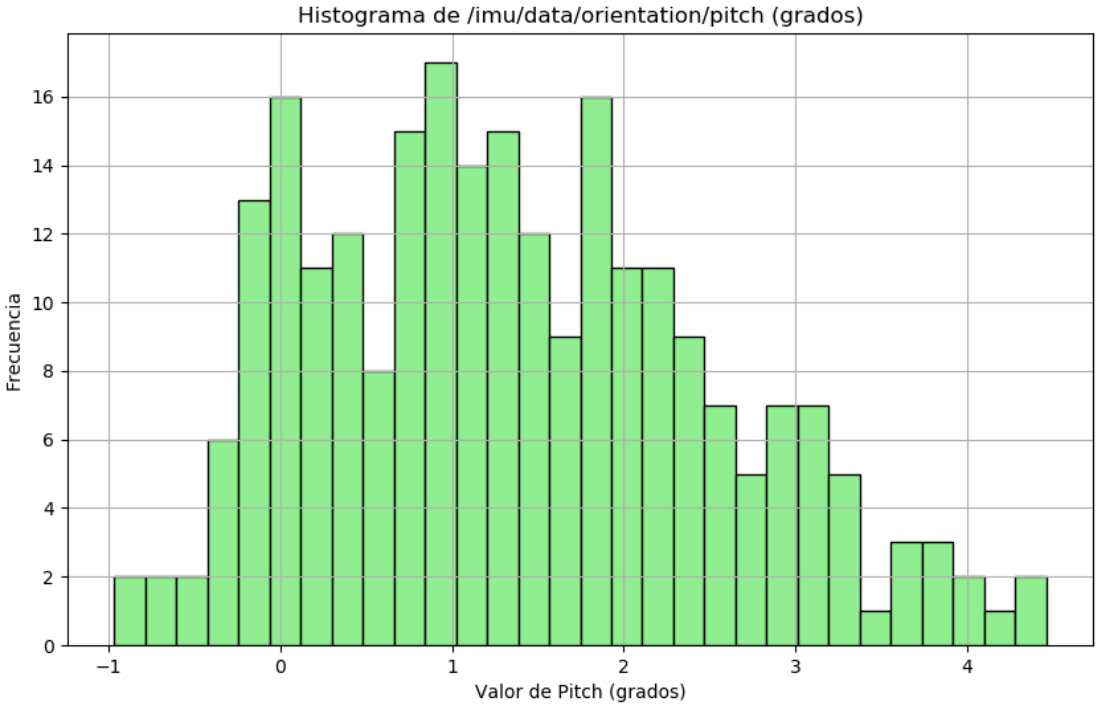


Figura 3.6.

Histograma de la orientación Pitch (grados)

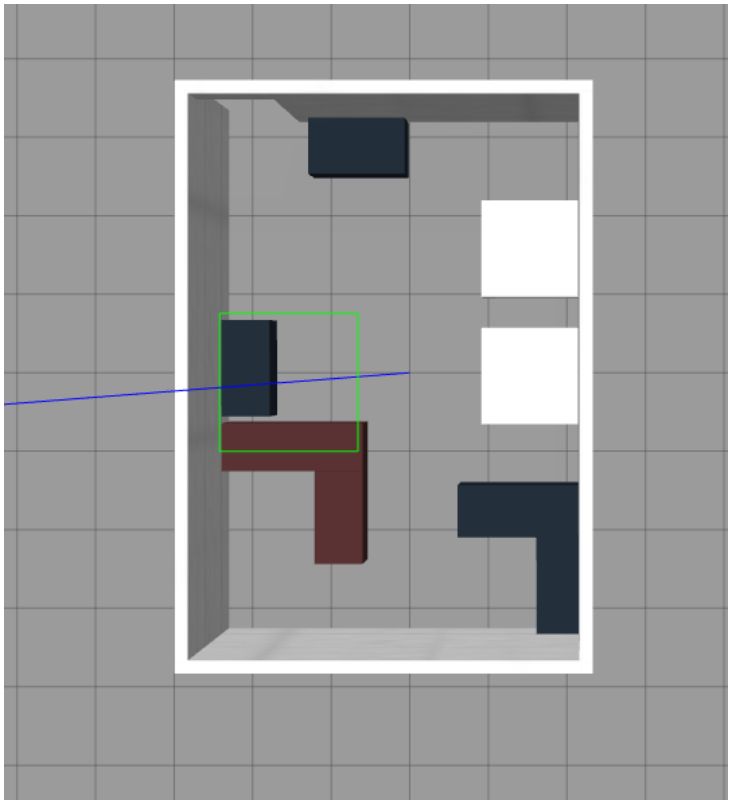


3.4 Escenario de pruebas

La simulación del robot cuadrúpedo multiservicio se realizó utilizando el simulador Gazebo, el cual permitió evaluar el desempeño del sistema en un entorno virtual controlado. El escenario seleccionado consistió en un mapa delimitado por paredes, creando un área cerrada ideal para analizar la navegación autónoma y la estabilidad de la locomoción del robot sin interferencias externas. Este entorno presentó un terreno regular, sin variaciones en la altura, lo que garantizó condiciones homogéneas para las pruebas.

Figura 3.7.

Mundo donde se realizaron las pruebas



3.5 Navegación autónoma

3.5.1 Localización durante la navegación autonoma

El robot cuadrúpedo fue ubicado en una posición específica dentro de un entorno simulado previamente mapeado por el propio robot. Para evaluar su capacidad de localización, se inició el proceso de AMCL (Adaptive Monte Carlo Localization) desde el terminal de ROS2, permitiendo analizar la precisión con la que el robot determinaba su pose relativa al mapa existente. Los resultados obtenidos destacaron una notable precisión en la localización del robot, con un error promedio inferior al 2%. Esto evidencia que el sistema AMCL es altamente eficiente en la estimación de la posición del robot dentro de un entorno conocido. Además, el robot demostró una rápida convergencia en la localización, logrando estimar su pose de manera precisa en menos de 5 segundos tras la activación del proceso de AMCL. Esta capacidad de convergencia rápida es crucial para tareas de navegación en tiempo real, ya que asegura una respuesta confiable del sistema en entornos previamente mapeados. Estos resultados confirman que el sistema de localización basado en AMCL proporciona información posicional precisa, lo que contribuye significativamente a la efectividad general de la navegación autónoma del robot en escenarios simulados.

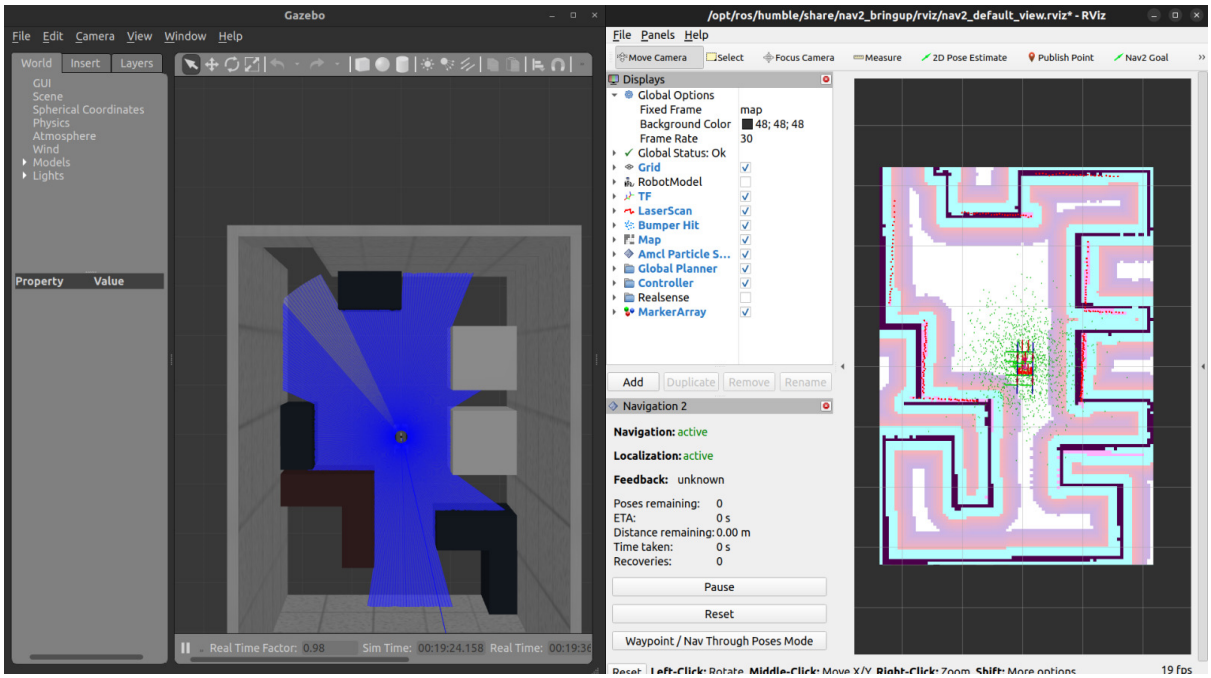
3.5.2 Esquiva de Obstáculos Estáticos - Mapa Predefinido

En este escenario, el robot cuadrúpedo demostró con eficacia su capacidad para navegar alrededor de un obstáculo estático. Durante la simulación, el robot maniobró hábilmente en un entorno previamente mapeado, utilizando la información almacenada en el mapa para planificar una trayectoria eficiente. Al aprovechar el conocimiento de la posición exacta del obstáculo, el robot calculó de manera precisa una ruta que evitó completamente la colisión, destacando la precisión de los algoritmos de mapeo y navegación implementados.

Esta maniobra, realizada con fluidez y sin interrupciones, resalta la confiabilidad del sistema de control del robot, asegurando un desplazamiento seguro y efectivo dentro del entorno simulado. La capacidad del robot para evitar obstáculos estáticos reafirma su idoneidad para navegar de manera autónoma en escenarios estructurados y controlados.

Figura 3.8.

Robot evadiendo obstáculo estático en el mapa



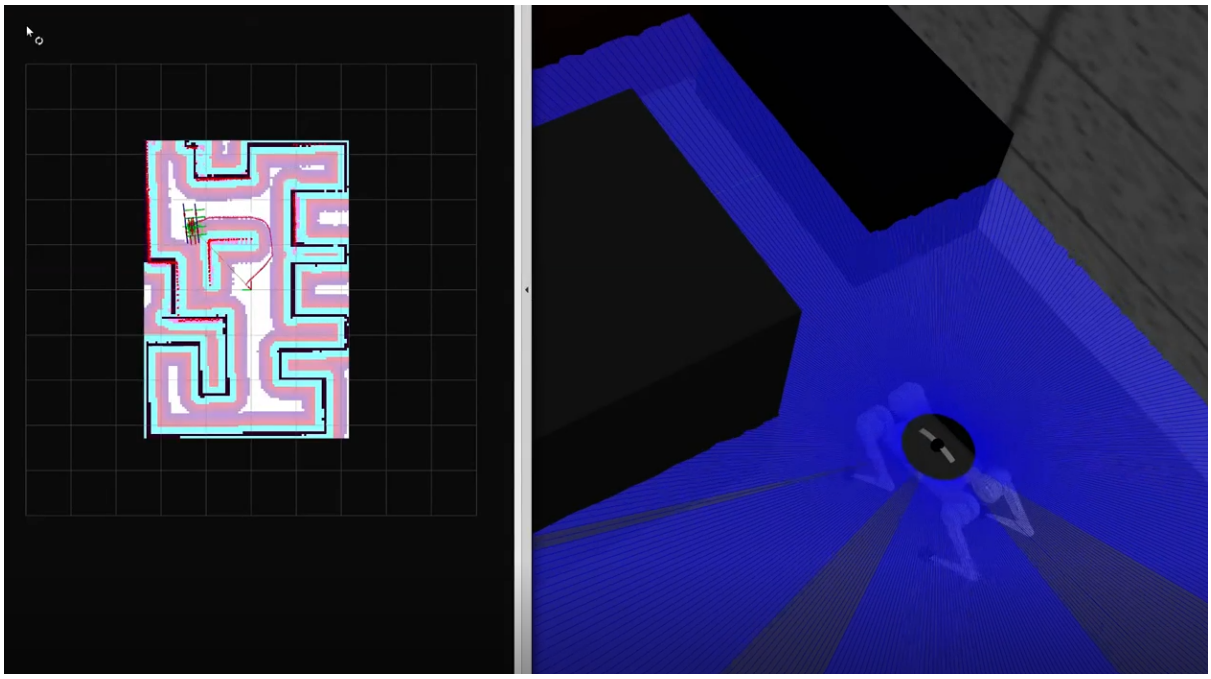
3.5.3 Esquiva de Obstáculos No Mapeados - Datos en Tiempo Real de LiDAR

En este escenario, se evaluó la capacidad del robot cuadrúpedo para esquivar un obstáculo estático introducido en el entorno durante la navegación, pero que no había sido previamente mapeado. Utilizando los datos en tiempo real capturados por el sensor LiDAR, el sistema detectó de manera autónoma la presencia del obstáculo y calculó una nueva trayectoria que permitió evitarlo con precisión. Este enfoque contrasta con el caso anterior, en el cual el robot se apoyaba en información preexistente del mapa. Aquí, la detección y reacción se basaron exclusivamente en datos sensoriales en vivo, destacando la

adaptabilidad del sistema ante cambios inesperados en el entorno. La habilidad del robot para identificar y esquivar obstáculos no mapeados demuestra la eficacia de los algoritmos de percepción y planificación.

Figura 3.9.

Robot evadiento obstáculo no mapeado



3.5.4 Navegación Basada en Metas

El robot cuadrúpedo fue configurado para navegar de manera autónoma hacia cuatro metas seleccionadas dentro del entorno simulado. Estas metas fueron ubicadas estratégicamente para evaluar la capacidad del robot para planificar y seguir trayectorias utilizando el framework `Nav2`, con parámetros adaptados en el archivo de configuración `nav2_params.yaml`. Durante las pruebas, se registraron los tiempos necesarios para alcanzar cada meta, así como las velocidades lineales y angulares promedio durante los trayectos.

La configuración del sistema incluyó el uso de `AMCL` para la localización, con un modelo de movimiento `nav2_amcl::OmniMotionModel` y un rango máximo de 10 metros para

el sensor `LiDAR`. El servidor de control fue configurado para una frecuencia de 20 Hz, utilizando el planificador `dwb_core::DWBLocalPlanner` con tolerancias de meta de 0.25 m para posición y orientación. Los mapas global y local se generaron con resoluciones de 0.06 m y 0.05 m, respectivamente, y se utilizó un enfoque de rolling window para el *costmap* local, asegurando que el robot respondiera dinámicamente a su entorno inmediato.

Tabla 3.1.

Tiempos de navegación hacia puntos de meta

Meta (Coordenadas)	Tiempo (s)	Velocidad Lineal (m/s)	Velocidad Angular (rad/s)
(-2.592, 1.629)	85.31	0.07	-0.05
(0.435, 0.178)	86.67	0.07	-0.01
(2.655, -0.910)	60.44	0.07	-0.03
(3.024, 1.998)	78.73	0.07	0.02

La planificación y ejecución de trayectorias se realizaron con el plugin NavfnPlanner, configurado para tolerar áreas desconocidas en el mapa, mientras que la evasión de obstáculos se manejó mediante capas estáticas, de obstáculos e inflado en los costmaps. Los resultados obtenidos reflejan un desempeño consistente en la navegación hacia metas, destacando la eficacia de los parámetros configurados en el sistema Nav2 para entornos simulados.

3.5.5 Analisis de datos

El análisis de los datos recopilados permitió obtener una comprensión detallada del desempeño del sistema de navegación, proporcionando una base sólida para conclusiones fundamentadas sobre sus capacidades. Se calcularon métricas estadísticas clave, como la covarianza entre la velocidad lineal y el tiempo, que fue de 0.137103, y la covarianza entre la

velocidad angular y el tiempo, que alcanzó un valor de -1.294495. La covarianza positiva en la velocidad lineal indica que el sistema mantiene una aceleración constante durante los trayectos, reflejando una adecuada coordinación en los algoritmos de control y un equilibrio entre eficiencia y estabilidad en los desplazamientos. Por otro lado, la covarianza negativa en la velocidad angular resalta la capacidad del sistema para regular cuidadosamente las desaceleraciones durante las maniobras angulares, lo que demuestra su eficacia para ejecutar giros precisos y suaves. Estos resultados subrayan la robustez del sistema de navegación, evidenciando su potencial para garantizar desplazamientos eficientes, estables y precisos en entornos simulados.

Figura 3.10.

Distribucion de la velocidad lineal

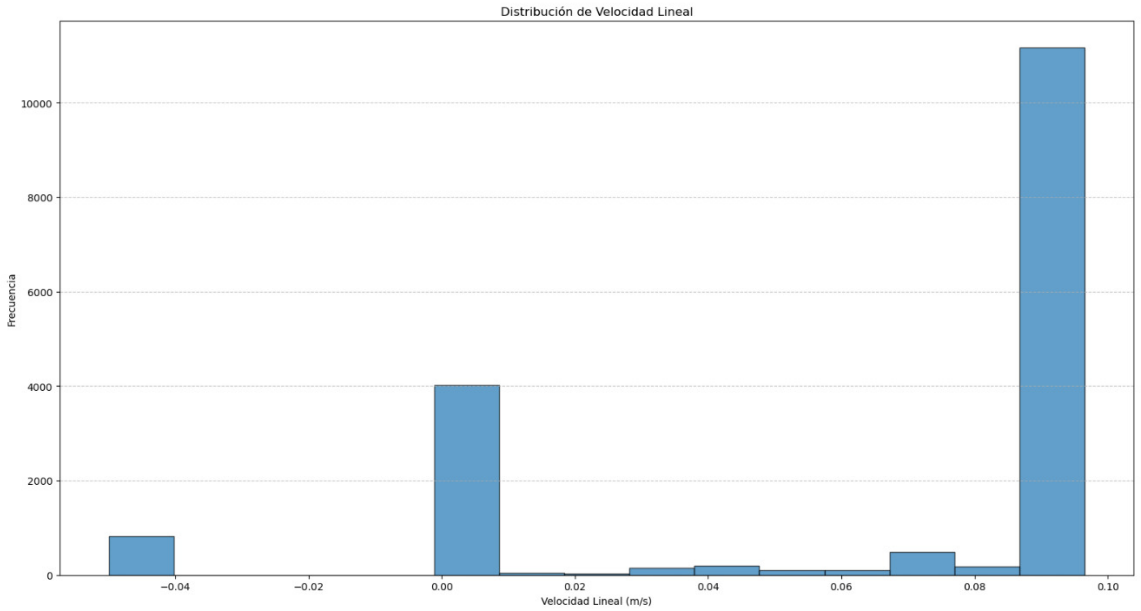
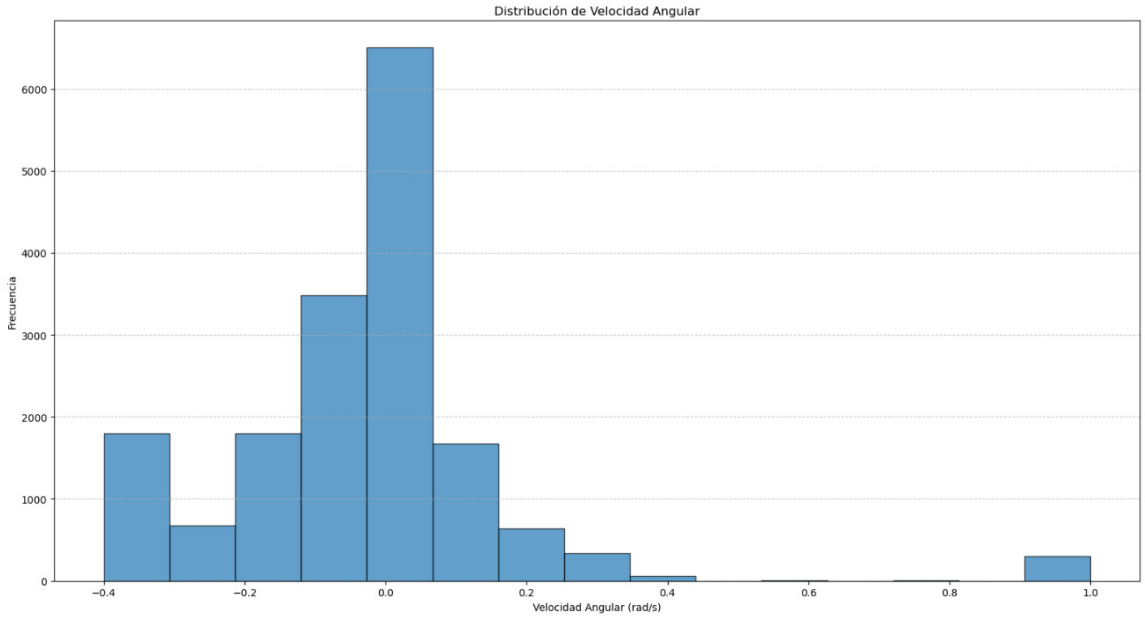


Figura 3.11.

Distribucion de la velocidad angular



Capítulo 4

4. Conclusiones y recomendaciones

El desarrollo del sistema de navegación autónoma para el robot cuadrúpedo en un entorno simulado permitió evaluar la eficacia de los algoritmos de control, localización y planificación de trayectorias. A lo largo del proyecto, se integraron herramientas como ROS2, Nav2, CHAMP y ros2_control, logrando que el robot pudiera desplazarse de manera autónoma, evitando obstáculos y alcanzando metas predefinidas con una alta precisión. La validación del desempeño del sistema se llevó a cabo mediante métricas de tiempo, estabilidad y error en el seguimiento de trayectorias, lo que permitió realizar ajustes y mejoras en el control de locomoción. Además, el análisis de datos proporcionó información clave sobre la relación entre la velocidad y la respuesta del sistema, permitiendo optimizar el control de bajo nivel. A continuación, se presentan las principales conclusiones y recomendaciones obtenidas en este trabajo.

4.1 Conclusiones

- Se desarrolló exitosamente un sistema de simulación y control para un robot cuadrúpedo multiservicio, utilizando ROS como herramienta principal. Este sistema permite realizar tareas de vigilancia en entornos regulares, demostrando la viabilidad de los robots como una alternativa eficiente y continua para la vigilancia autónoma.

- Los algoritmos implementados permitieron que el robot navegue de manera autónoma en el entorno simulado, identificando y evitando obstáculos con precisión, validando la funcionalidad del sistema en escenarios reales simulados.

- Los sensores integrados proporcionaron datos confiables para el monitoreo en tiempo real, permitiendo detectar intrusiones o anomalías de forma eficiente, destacando la importancia de estos componentes en tareas de vigilancia.

- Las pruebas realizadas demostraron que los ajustes progresivos en los algoritmos de control, comenzando con un control P y finalmente a un PID, mejoraron notablemente la estabilidad del robot cuadrúpedo.

4.2 Recomendaciones

Tras culminar lo planificado en la propuesta y evaluar el desempeño del sistema de navegación y control del robot cuadrúpedo, se identificaron aspectos que pueden optimizarse para mejorar su estabilidad, precisión y adaptabilidad en entornos más complejos. A continuación, se presentan las recomendaciones primordiales derivadas de este trabajo:

- Para trabajos futuros, se recomienda utilizar el Reinforcement Learning Toolbox junto con el ROS Toolbox de MATLAB, que permiten entrenar al robot cuadrúpedo en tareas complejas, como subir escaleras o navegar en terrenos irregulares.
- Se sugiere integrar cámaras RGB-D o sensores de profundidad en el modelo del robot para mejorar sus capacidades de percepción y mapeo. Esto permitiría al robot generar mapas en 3D y navegar en entornos complejos con mayor precisión, complementando los datos del LiDAR.

Bibliografía

[1] M. Gupta, S. Kumar, L. Behera, and V. K. Subramanian, “A novel vision-based tracking algorithm for a human-following mobile robot,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, pp. 1415–1427, 7 2017.

[2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Springer, 2009.

[3] M. Raibert, “Bigdog, the rough-terrain quadruped robot,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, 2008.

[4] A. Majithia, D. Shah, J. Dave, A. Kumar, S. Rathee, N. Dogra, V. H. M, D. S. Chiniwar, and S. Hiremath, “Design, motions, capabilities, and applications of quadruped robots: a comprehensive review,” *Frontiers in Mechanical Engineering*, vol. 10, p. 1448681, 8 2024.

[5] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: An open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[7] Y. Fan, Z. Pei, C. Wang, M. Li, Z. Tang, and Q. Liu, “A review of quadruped robots: Structure, control, and autonomous motion,” *Advanced Intelligent Systems*, vol. 6, 6 2024.

[8] “Spot | boston dynamics.” [Online]. Available: https://bostondynamics.com/products/spot/

[9] “Unitree a1 highly integrated, pushing limits - unitree robotics.” [Online]. Available: https://www.unitree.com/a1

[10] M. Hutter, “Anymal-toward legged robots for harsh environments.” [Online]. Available: https://doi.org/10.3929/ethz-b-000194231

[11] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” 1 2022. [Online]. Available: http://arxiv.org/abs/2201.08117http://dx.doi.org/10.1126/scirobotics.abk2822

[12] G. Bledt, M. J. Powell, B. Katz, J. D. Carlo, P. M. Wensing, and S. Kim, “Mit cheetah 3: Design and control of a robust, dynamic quadruped robot,” in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 12 2018, pp. 2245–2252.

[13] J. M. Yang, “Tripod gaits for fault tolerance of hexapod walking machines with a locked joint failure,” *Robotics and Autonomous Systems*, vol. 52, pp. 180–189, 8 2005.

[14] Z. Yu, X. Chen, Q. Huang, W. Zhang, L. Meng, W. Zhang, and J. Gao, “Gait planning of omnidirectional walk on inclined ground for biped robots,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 888–897, 7 2016.

[15] J. Pomares, B. Álvaro, and B. Baeza, “Escuela politécnica superior control de locomoción de robots cuadrúpedos mediante reinforcement learning.”

[16] A. E. R. d. Jonge, “Discussion: “a kinematic notation for lower-pair mechanisms based on matrices” (denavit, j., and hartenberg, r. s., 1955, asme j. appl. mech., 22, pp. 215–221),” Mar 1956.

[17] M. Kalyoncu, M. A. Sen, and V. Bakircioglu, “Inverse kinematic analysis of a quadruped robot,” *International Journal of Scientific Technology Research*, vol. 6, 2017. [Online]. Available: www.ijstr.org

[18] L. A. O. Moreno, C. Rodríguez, and E. Valverde, “Control cinemático para un robot cuadrúpedo usando el método de newton-raphson,” *Elektron*, vol. 5, 5 2021.

[19] P. Biswal and P. K. Mohanty, “Kinematic and dynamic modeling of a quadruped robot,” in *Lecture Notes in Mechanical Engineering*. Springer Science and Business Media Deutschland GmbH, 2022, pp. 369–378.

[20] O. K. Adak, B. Bahceci, and K. Erbatur, “Modeling of a quadruped robot with spine joints and full-dynamics simulation environment construction,” 3 2022. [Online]. Available: http://arxiv.org/abs/2203.09622

[21] M. H. Bui, T. P. Pham, T. M. N. Nguyen, and X. B. Dang, “Development of a direct adaptive pid controller for a quadruped robot,” *Journal of Technical Education Science*, vol. 17, pp. 1–9, 8 2022. [Online]. Available: https://jte.edu.vn/index.php/jte/article/view/1130

[22] S. i. M. E. M. I. o. T. L. Jongwoo, “Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the mit cheetah robot,” 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/85490

[23] Y. Fukuoka, H. Kimura, and A. H. Cohen, “Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts,” *http://dx.doi.org/10.1177/0278364903022003004*, vol. 22, pp. 187–202, 3 2003. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/0278364903022003004

[24] S. S. Kotha, N. Akter, S. H. Abhi, S. K. Das, M. R. Islam, M. F. Ali, M. H. Ahamed, M. M. Islam, S. K. Sarker, M. F. R. Badal, P. Das, Z. Tasneem, and M. M. Hasan, “Next generation legged robot locomotion: A review on control techniques,” *Heliyon*, vol. 10, 9 2024.

[25] D. Zhang, D. Hu, L. C. Shen, and H. Xie, “Design of a central pattern generator for bionic-robot joint with angular frequency modulation,” *2006 IEEE International Conference on Robotics and Biomimetics, ROBIO 2006*, pp. 1664–1669, 2006.

[26] L. Righetti and A. J. Ijspeert, “Pattern generators with sensory feedback for the control of quadruped locomotion,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 819–824, 2008. [Online]. Available: https://www.researchgate.net/publication/4341200_Pattern_generators_with_sensory_feedback_for_the_control_of_quadruped_locomotion

[27] J. Seipel and P. Holmes, “A simple model for clock-actuated legged locomotion,” *Regular and Chaotic Dynamics*, vol. 12, pp. 502–520, 2007. [Online]. Available: https://link.springer.com/article/10.1134/S1560354707050048

[28] I. Poulakakis, E. Papadopoulos, and M. Buehler, “On the stability of the passive dynamics of quadrupedal running with a bounding gait,” *International Journal of Robotics Research*, vol. 25, pp. 669–687, 7 2006. [Online]. Available: https://www.researchgate.net/publication/220122413_On_the_Stability_of_the_Passive_Dynamics_of_Quadrupedal_Running_with_a_Bounding_Gait

[29] L. R. Palmer and D. E. Orin, “Force redistribution in a quadruped running trot,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4343–4348, 2007.

[30] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, “Control of dynamic gaits for a quadrupedal robot,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3287–3292, 2013. [Online]. Available: https://www.researchgate.net/publication/243971891_Control_of_Dynamic_Gaits_for_a_Quadrupedal_Robot

[31] J. Estremera and K. J. Waldron, “Thrust control, stabilization and energetics of a quadruped running robot,” *International Journal of Robotics Research*, vol. 27, pp. 1135–1151, 10 2008. [Online]. Available: https://dl.acm.org/doi/10.1177/0278364908097063

[32] M. H. Raibert, “Legged robots that balance,” p. 250, 2000.

[33] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part i history of the slam problem,” 2006.

[34] Q. H. Nguyen, P. Johnson, and D. Latham, “Performance evaluation of ros-based slam algorithms for handheld indoor mapping and tracking systems,” *IEEE Sensors Journal*, 2022.

[35] M. D. Rose, “Politecnico di torino lidar-based dynamic path planning of a mobile robot adopting a costmap layer approach in ros2,” 2021.

[36] “Nav2 — nav2 1.0.0 documentation.” [Online]. Available: https://docs.nav2.org/

[37] S. Macenski and I. Jambrecic, “Slam toolbox: Slam for the dynamic world,” *Journal of Open Source Software*, vol. 6, p. 2783, 5 2021.

[38] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, 2004.

[39] C. E. Mower, T. Stouraitis, J. Moura, C. Rauch, L. Yan, N. Z. Behabadi, M. Gienger, T. Vercauteren, C. Bergeles, and S. Vijayakumar, “Ros-pybullet interface: A framework for reliable contact simulation and human-robot interaction,” *Proceedings of Machine Learning Research*, vol. 205, pp. 1411–1423, 10 2022. [Online]. Available: https://arxiv.org/abs/2210.06887v1

[40] M. Zwingel, C. May, M. Kalenberg, and J. Franke, “Robotics simulation – a comparison of two state-of-the-art solutions.” ARGESIM Arbeitsgemeinschaft Simulation News, 7 2022, pp. 171–178.