UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA ELECTRÓNICA

Control de Trayectoria de un Robot Móvil Omnidireccional en un Entorno Controlado

Trabajo de graduación previo a la obtención del título de:

INGENIERO ELECTRÓNICO

Autor:

BRAYAN JOSEPH CASTILLO OTOYA

Director:

ANDRÉS PATRICIO CABRERA FLOR

CUENCA, ECUADOR

2023

DEDICATORIA

A mi amada familia, quienes siempre han sido mi fuente de inspiración y apoyo incondicional. Esta tesis está dedicada a ustedes, por su amor, paciencia y aliento constante durante todo este arduo proceso. Cada logro alcanzado es gracias a su confianza en mí y su infinita motivación. Gracias por ser mi apoyo constante y creer en mis sueños.

AGRADECIMIENTO

Quiero dedicar un sincero agradecimiento a la familia Ocampo, por su incondicional apoyo y confianza, los cuales han sido pilares fundamentales para seguir mis estudios y alcanzar este significativo logro académico. Estoy profundamente agradecido por su constante respaldo y por creer en mi potencial. Su invaluable contribución ha dejado una huella imborrable en mi vida, y siempre los llevaré en mi corazón con gratitud y aprecio.

Resumen:

Para diversas aplicaciones, los robots móviles incorporan diferentes componentes estructurales, electrónicos y mecánicos, con el objetivo de maniobrar de mejor manera dentro los distintos espacios operativos. Esta investigación consiste en la implementación y análisis de algoritmos de control de trayectorias basado en el modelo cinemático de una plataforma robótica omnidireccional con cuatro ruedas tipo Mecanum en un entorno controlado, que permite movimientos instantáneos en cualquier dirección sin necesidad maniobras extras. La metodología incluye el desarrollo del modelo cinemático del robot dentro del simulador Gazebo y la implementación del control de un prototipo utilizando Raspberry Pi y OpenCR a través del entorno ROS. La comunicación entre los componentes del sistema se ha establecido mediante protocolos TCPROS y UDPROS. Los resultados obtenidos muestran el correcto funcionamiento tanto del prototipo como del modelo simulado, así como la capacidad de seguir trayectorias preestablecidas.

Palabras clave: Mecanum, ROS, cinemática, control de trayectoria, omnidireccional, robot móvil

Abstract:

For several applications, mobile robots incorporate different structural, electronic, and mechanical components with the aim of maneuvering better within different operating spaces. This research consists of the implementation and analysis of trajectory control algorithms based on the kinematic model of an omnidirectional robotic platform with four Mecanum-type wheels in a controlled environment, which allows instant movements in any direction without the need for extra maneuvers. The methodology includes the development of the kinematic model of the robot within the Gazebo simulator and the implementation of a control algorithm for a prototype using Raspberry Pi and OpenCR through the ROS environment. Communication between system components has been established through TCPROS and UDPROS protocols. The results obtained show the correct functioning of both the prototype and the simulated model, as well as the ability to follow pre-established trajectories.

Keywords: Mecanum, ROS, kinematics, mobile robot, omnidirectional, trajectory control

Este certificado se encuentra en el repositorio digital de la Universidad del Azuay, para verificar su autenticidad escanee el código QR

Este certificado consta de: 1 página

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Control de Trayectoria de un Robot Móvil Omnidireccional en un Entorno Controlado

Brayan Joseph Castillo Otoya
Escuela de Ingeniería Electrónica
Universidad del Azuay
Cuenca, Ecuador
jcasot@es.uazuay.edu.ec

***Resumen*— Para diversas aplicaciones, los robots móviles incorporan diferentes componentes estructurales, electrónicos y mecánicos, con el objetivo de maniobrar de mejor manera dentro los distintos espacios operativos. Esta investigación consiste en la implementación y análisis de algoritmos de control de trayectorias basado en el modelo cinemático de una plataforma robótica omnidireccional con cuatro ruedas tipo Mecanum en un entorno controlado, que permite movimientos instantáneos en cualquier dirección sin necesidad maniobras extras. La metodología incluye el desarrollo del modelo cinemático del robot dentro del simulador Gazebo y la implementación del control de un prototipo utilizando Raspberry Pi y OpenCR a través del entorno ROS. La comunicación entre los componentes del sistema se ha establecido mediante protocolos TCPROS y UDPROS. Los resultados obtenidos muestran el correcto funcionamiento tanto del prototipo como del modelo simulado, así como la capacidad de seguir trayectorias preestablecidas.**

Palabras clave—Mecanum, ROS, cinemática, control de trayectoria, omnidireccional, robot móvil.

I. INTRODUCCIÓN

El concepto de movilidad omnidireccional es una situación común en la ficción, como en Star Wars donde los pilotos de las naves se mueven libremente y sin ningún límite. En la realidad, alcanzar un nivel de libertad de movimiento para los vehículos no es fácil, debido a que las ruedas tradicionales se limitan a moverse en una sola dirección y requieren mecanismos adicionales para poder realizar movimientos que les permita tener un mayor rango de maniobra [1]. Es por esto que la movilidad omnidireccional se ha convertido en una necesidad para distintas áreas industriales, médicas, militares e incluso en el hogar. Debido a la constante demanda de mejoras en las distintas áreas de producción y automatización, las aplicaciones robóticas móviles se están volviendo cada vez más extensas [2].

Para los distintos escenarios, los vehículos robóticos incorporan diferentes componentes estructurales, elementos electrónicos y mecánicos, los cuales le otorgan características y funcionalidades únicas para desempeñar roles específicos requeridos por los usuarios. Por lo tanto, para poder maniobrar de mejor manera en los distintos espacios operativos existen un sinfín de configuraciones y tipos de robots móviles [3].

Diferentes escenarios requieren diferentes requisitos, y para saber a dónde ir, el robot móvil debe tener un conocimiento de su ubicación actual o su lugar de destino utilizando referencias externas y algoritmos programados en su controlador que faciliten moverse a una posición designada de manera segura [2]-[3]. Además, la reducción del espacio de operación impulsa el desarrollo y la investigación del control de equipos robóticos móviles, para cumplir tareas en los lugares que son inhóspitos para el ser humano, pero que pueden ser de fácil acceso para un robot móvil controlado [4].

El robot móvil omnidireccional (OMR) con ruedas Mecanum tiene características que lo hacen llamativo: posee cuatro ruedas, cada una de estas cuentan con rodillos insertados en una estructura circular exterior fijada a la rueda, y cada rodillo está dispuesto en un ángulo de 45 grados con respecto a la circunferencia exterior que los soporta (Fig. 1), con cada rueda impulsada por un motor individual [5]-[6]. Esto le permite desplazarse hacia cualquier punto del plano, haciendo que el robot cambie de dirección de forma inmediata sin necesidad de girar el ángulo de la rueda antes de iniciar el movimiento en cualquiera de sus componentes de traslación [7].

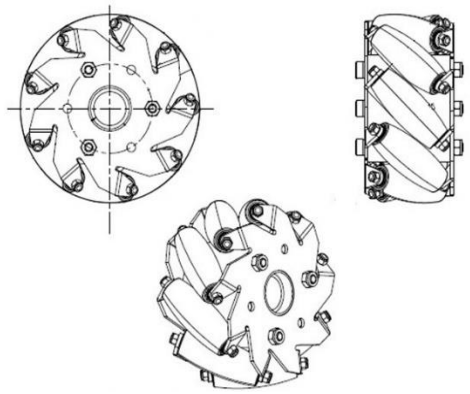

Fig. 1. Diseño de la rueda Mecanum.

En la actualidad existen muchas investigaciones centradas en el control y monitorización de trayectorias. En [7], se menciona la implementación y evaluación de algoritmos de control para la monitorización de trayectorias basadas en el modelo cinemático y dinámico de una plataforma robótica omnidireccional con el objetivo de utilizar la simulación hardware-in-the-loop que permita comparar los datos obtenidos.

Piemngam, Nilkhamhang y Bunnun presentan un proyecto de un robot móvil autónomo con capacidades de movimiento omnidireccional, el cual puede encontrar la mejor ruta para llegar al área de destino utilizando un mapa precargado. El robot móvil autónomo utiliza datos de cámaras para detectar el entorno y crear un mapa virtual, lo cual sirve en la planificación de rutas [8].

Los autores Mulun-Wu, Shi Lu Dai y Yang proponen una ruta interactiva para el usuario, mejorada con realidad mixta HoloLens (dispositivo holográfico). La interfaz de dicho dispositivo puede mostrar el mapa, la ruta, y el comando de control para lograr una interacción en tiempo real entre el robot móvil omnidireccional y los objetos virtuales [9].

Inspirado en los estudios antes mencionados, este artículo tiene como objetivo la implementación de un sistema de control para un robot móvil con ruedas Mecanum utilizando los controladores Raspberry Pi 4 y OpenCR. La comunicación con el robot se realiza a través del entorno ROS (Robot Operating System), el cual también permite realizar simulaciones para hacer un seguimiento de las trayectorias. Este documento está organizado de la siguiente manera: en la Sección 2, se describe la metodología, presentando el modelo cinemático del robot omnidireccional con cuatro ruedas Mecanum, el diseño del robot en Gazebo y el desarrollo tanto de la arquitectura del hardware como la del software. Los resultados basados en el análisis comparativo de la trayectoria se presentan y discuten en la Sección 3. Finalmente, se dan las conclusiones en la Sección 4.

II. Metodología

A. *Modelo cinemático del robot omnidireccional*

El modelo cinemático de un robot móvil con ruedas Mecanum se basa en las relaciones existentes entre las velocidades de las ruedas, y la velocidad y dirección del robot en el plano. Estas relaciones se pueden expresar mediante ecuaciones matemáticas, lo que permite controlar la velocidad y dirección del robot de manera precisa.

1) Cinemática directa: Para obtener el modelo se analiza el modo en el que el robot móvil se comporta en el marco de coordenadas global $\{G\} = [X_G, Y_G, Z_G]^T$ [10]-[11]. En Fig. 2 se encuentra el sistema de referencia para este modelo.

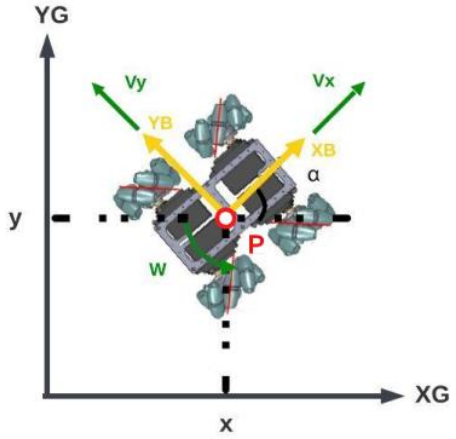

Fig. 2. Sistema de referencia para el movimiento omnidireccional de la plataforma robótica.

Donde:

$P\ (x, y)$ = es el centroide del robot.

α = es el ángulo entre el rumbo del robot y el eje X_G.

V_x = es la velocidad lineal en X_B.

V_y = es la velocidad lineal en Y_B.

W = es la velocidad angular, con respecto al marco del robot $\{B\} = [X_B, Y_B, Z_B]^T$.

A partir del sistema de referencia inercial del robot {B}, se representan las velocidades que regirán al robot móvil mediante las siguientes ecuaciones:

$$x = V_x cos(\alpha) - V_y sin(\alpha) \quad (1)$$

$$y = V_x sin(\alpha) + V_y cos(\alpha) \quad (2)$$

$$\alpha = W \quad (3)$$

Se escriben las ecuaciones de forma matricial.

$$\begin{bmatrix} x \\ y \\ q \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} \quad (4)$$

Se aplica la cinemática inversa.

$$\begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ q \end{bmatrix} \quad (5)$$

Luego, se obtiene la relación entre las velocidades tanto de la plataforma robótica como de las ruedas Mecanum, por lo que se toma en cuenta el sistema de referencia global $\{G\}$. En la Fig. 3, se presentan los diferentes componentes vectoriales de cada rueda, donde ω_i ($i = 1, 2, 3, 4$), son las velocidades de rotación de las ruedas, mientras que R es el radio y $L = L_x + L_y$ es la suma de la mitad de la distancia entre las ruedas [12].

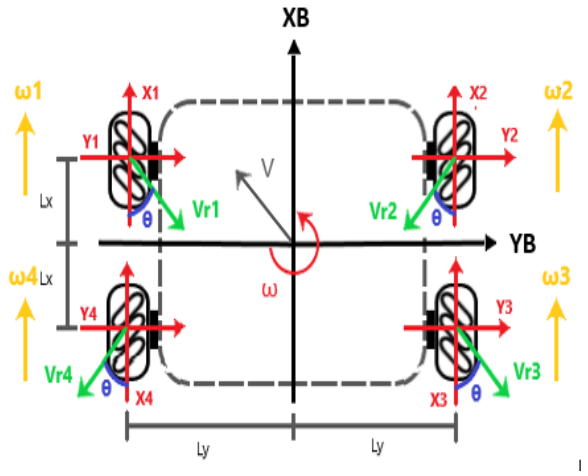

Fig. 3. Configuración de ruedas y definición de postura de la plataforma robótica.

Al final, se obtienen las ecuaciones correspondientes a cada uno de los motores, las cuales permiten calcular las velocidades de referencia a las que estos deben girar.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & -1 & -L \\ 1 & 1 & L \\ 1 & 1 & -L \\ 1 & -1 & L \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} \quad (6)$$

$$\omega_1 = \frac{1}{R}(V_x - V_y - (L)(W)) \quad (7)$$

$$\omega_2 = \frac{1}{R}(V_x + V_y + (L)(W)) \quad (8)$$

$$\omega_3 = \frac{1}{R}(V_x + V_y - (L)(W)) \quad (9)$$

$$\omega_4 = \frac{1}{R}(V_x - V_y + (L)(W)) \quad (10)$$

2) Cinemática inversa: La aplicación de la cinemática inversa permite obtener las velocidades de cada motor en un robot. A través del uso de encoders, se pueden medir las velocidades de cada motor, lo que permite determinar la velocidad con la que se produce el movimiento del robot. Estos parámetros son esenciales para aplicar al modelo cinemático del robot, permitiendo así controlar su movimiento de manera precisa y eficiente [11].

$$\begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} = \frac{R}{4}\begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{1}{L} & \frac{-1}{L} & \frac{-1}{L} & \frac{1}{L} \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (11)$$

$$V_x = \frac{R}{4}(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \quad (12)$$

$$V_y = \frac{R}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4) \quad (13)$$

$$W = \frac{R}{4}(\frac{\omega_1}{L} - \frac{\omega_2}{L} - \frac{\omega_3}{L} + \frac{\omega_4}{L}) \quad (14)$$

3) Movimientos de la plataforma omnidireccional: Las ruedas omnidireccionales, que en la mayoría de los casos están conectadas al cuerpo del robot, no se giran en la dirección deseada, sino que la dirección se logra mediante una combinación de las velocidades de rotación de las ruedas [4]. Cuando las ruedas giran, solo se ejerce una componente de fuerza sobre el suelo a lo largo del eje de los rodillos, mientras que el segundo componente de fuerza se utiliza para hacer girar los rodillos pasivos. De este modo, la fuerza que actúa sobre el vehículo desde la rueda en cuestión se orienta en un ángulo de 45° con respecto al eje de la rueda, pero no afecta el movimiento del vehículo. Al controlar la rotación de cada rueda individualmente (y, por tanto, de cada fuerza individual), el vehículo puede desplazarse en cualquier dirección deseada, tal y como se ilustra en la Fig. 4 [13].

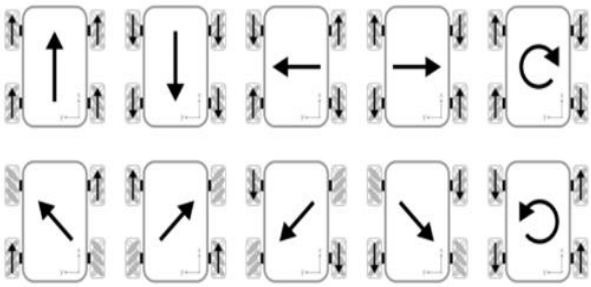

Fig. 4. Las principales direcciones de locomoción del robot móvil.

B. *Diseño y construcción de la arquitectura del hardware*

El diseño y construcción de un robot implica consideraciones de sus componentes tanto mecánicas como eléctricas. Este enfoque es fundamental para garantizar que el robot funcione de manera óptima y eficiente en el cumplimiento de las tareas previstas. El robot desarrollado y sus componentes mecánicos y eléctricos se describen detalladamente en la siguiente sección de este documento.

1) Controladores:

a) La Raspberry Pi 4, es una mini computadora de placa única que cuenta con un procesador Broadcom BCM2711 de cuatro núcleos ARM Cortex-A72 de 64 bits, 4 GB de RAM y conectividad Bluetooth 5.0, Wi-Fi de doble banda 802.11ac y Gigabit Ethernet. Debido a su potencia de procesamiento y memoria, la Raspberry Pi 4 es compatible con una variedad de sistemas operativos y es capaz de ejecutar aplicaciones de ROS, incluyendo la simulación y control de robots. Además, su tamaño compacto la convierte en una excelente opción para su uso en robots móviles y otros proyectos de robótica donde el espacio es limitado [14].

b) OpenCR 1.0, es una placa de control de robot de código abierto con un microcontrolador ARM Cortex-M7 de 32 bits. Cuenta con múltiples opciones de conectividad, sensores integrados y capacidad de expansión, además de opciones de programación y compatibilidad con Arduino lo que la hace una buena opción para encargarse del control de los servomotores del robot móvil omnidireccional [15].

2) Actuadores: El Dynamixel XL430-W250-T es un servo motor de alta gama diseñado para aplicaciones de robótica avanzada. Sus características destacadas incluyen alto rendimiento con un torque máximo de 2.5 Nm y velocidad de rotación de hasta 61 RPM, comunicación bidireccional, diseño compacto y ligero, sensor de posición de alta precisión y software de control avanzado. Es ideal para aplicaciones de robótica que requieren alta potencia, velocidad y precisión en el control del motor [16].

3) Batería: Para la fuente de energía del robot se utilizó una batería de polímero de litio de 11.1 V y con capacidad de carga de 5500 mAh. Esta batería es la encargada de ser la fuente de energía de ambos controladores, alimenta directamente a la placa OpenCR y esta a su vez actúa como distribuidor de energía alimentando a los servomotores y a la Raspberry Pi.

4) Chasis: El robot omnidireccional tiene un cuerpo de plástico liviano de dos pisos, en el primer piso se encuentran ubicados los controladores y actuadores, mientras que en el segundo piso se encuentra la batería. Los acoples entre las ruedas y los servomotores se crearon utilizando un software de diseño y fueron fabricados en una impresora 3D. La Fig. 5 muestra la apariencia física real del robot. En la Fig. 6 se observa la conexión de los distintos elementos del robot.

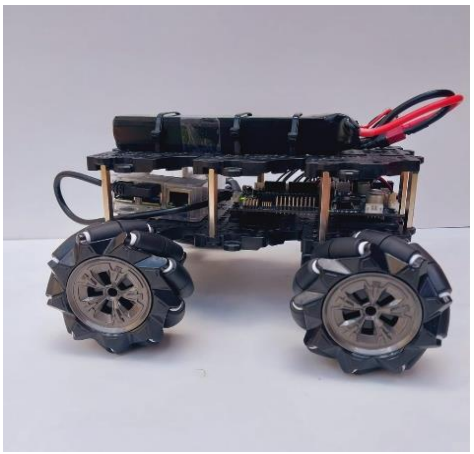

Fig. 5. Montaje final del robot móvil omnidireccional.

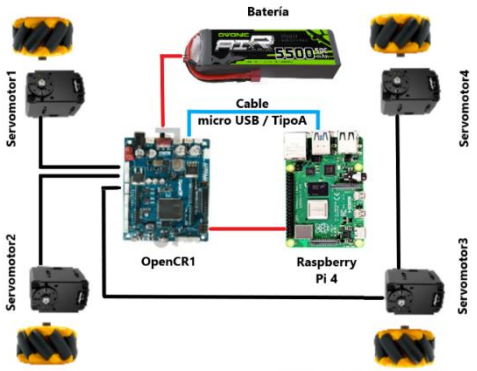

Fig. 6. Diagrama de conexión de diferentes componentes eléctricos.

C. *Software Implementado en el sistema*

El software es una de las partes más importantes del robot debido a que los programas y algoritmos son los encargados de controlar y coordinar el funcionamiento del robot. En esta sección, se presenta el conjunto de programas que se emplean en el control del robot y las diferentes tecnologías que se utilizan para lograrlo. Además, se explica muchos de los conceptos relacionados con ellas.

1) ROS (Robot Operating System): El robot móvil está construido alrededor de ROS, que es un marco de trabajo de robótica que facilita el desarrollo de software para robots. ROS proporciona herramientas y librerías para la comunicación entre procesos, la gestión de software y la simulación, y se utiliza en la investigación y desarrollo de robótica. Además, es compatible con diferentes plataformas de hardware y software, ya que cuenta con una comunidad activa de usuarios y desarrolladores que contribuyen a su mejora continua y al desarrollo de nuevos paquetes y aplicaciones [17]. A continuación, se describen algunos conceptos básicos tomados de [18], estos describen la estructura interna de ROS y permiten tener una mejor comprensión del tema.

a) ROS Master: El funcionamiento de ROS comienza con el ROS Master, el cual es responsable de permitir que todas las piezas de software de ROS (Nodos) se encuentren y se comuniquen entre sí. Gracias a esto, no es necesario indicar específicamente a qué computadora enviar los datos de un sensor, sino que simplemente se puede instruir al Nodo 1 que envíe mensajes al Nodo 2, y el ROS Master se encargará de dirigirlos adecuadamente. De esta forma, se simplifica y se optimiza la comunicación entre los distintos nodos que conforman el sistema de control del robot.

b) Nodos: En ROS, un "nodo" se refiere a la unidad más pequeña de procesamiento, como un programa ejecutable. Se recomienda crear un nodo específico para cada función del robot, como la adquisición de datos de un sensor, la conversión de datos del mismo, el reconocimiento de obstáculos, la activación de motores y la navegación. Cada nodo registra su información con nombre, tipo de mensaje, dirección URI y número de puerto al iniciarse.

Los nodos pueden actuar como publicadores, suscriptores, servidores o clientes de servicios, y pueden intercambiar mensajes mediante tópicos. Para comunicarse con el maestro, el nodo utiliza XMLRPC (XML para llamada de procedimientos remotos), y para comunicarse con otros nodos, utiliza XMLRPC o TCPROS de los protocolos TCP/IP. El nodo también utiliza una variable llamada "ROS_HOSTNAME" para establecer su dirección URI, y un puerto único arbitrario para la comunicación.

c) Paquete: Un paquete es la unidad básica de ROS. La aplicación de ROS se desarrolla en base a paquetes, y cada paquete contiene archivos de configuración para lanzar otros paquetes o nodos. También incluye todos los archivos necesarios para ejecutar el paquete, incluyendo librerías de dependencia de ROS para ejecutar varios procesos, conjuntos de datos y archivos de configuración.

d) Tópico: Es literalmente un tema de una conversación. El nodo publicador primero registra su tópico con el maestro y luego comienza a publicar mensajes sobre el tópico. Los nodos suscriptores que desean recibir el tema solicitan información al nodo publicador correspondiente al nombre del tópico registrado en el maestro. Con base en esta información, el nodo suscriptor se conecta directamente al nodo publicador para intercambiar mensajes.

e) Message: Un nodo en ROS envía o recibe datos entre nodos a través de un mensaje. Los mensajes son variables como enteros, punto flotante y booleanos. Se puede utilizar una estructura de mensaje anidado que contenga otros mensajes o una matriz de mensajes dentro de un mensaje.

Se utiliza el protocolo de comunicación TCPROS y UDPROS para la entrega de mensajes. Los tópicos se utilizan en la entrega de mensajes unidireccionales, mientras que el servicio se utiliza en la entrega de mensajes bidireccionales que implican solicitud y respuesta.

2) Simulador Gazebo: Gazebo es un simulador que provee modelos de robots, sensores y entornos para la simulación 3D requeridos en el desarrollo de plataformas robóticas. Gazebo es desarrollado y distribuido por Open Robotics, que está a cargo de ROS y su comunidad, por lo que está integrado con ROS [19].

Estas son algunas características de Gazebo:

a) Simulación de dinámica

b) Simulación de sensores: Admite el rango del láser, la cámara 2D/3D, la cámara de profundidad, el sensor de contacto, el sensor de fuerza y torque y mucho más. Además, se puede aplicar ruido a los datos del sensor similar al entorno real.

c) Plug-ins: Proporcionan APIs para permitir a los usuarios crear robots, sensores y control de entornos como complemento.

d) Modelo de robot: Los usuarios pueden agregar sus propios robots con un archivo especial.

e) Transmisión de datos TCP/IP: La simulación se puede ejecutar en un servidor remoto y se utiliza la transmisión de mensajes basada en sockets de Google Protobufs.

f) Herramienta de línea de comandos: Se admiten herramientas GUI y CUI para verificar y controlar el estado de la simulación.

D. Simulación del robot en Gazebo

Para crear el diseño del robot, se utiliza el formato URDF (Universal Robot Description Format), el cual es un documento XML en el que se pueden especificar las características y parámetros físicos del robot, los cuales van desde la longitud del vehículo robótico y el tamaño de las ruedas, hasta la ubicación de los sensores sobre el robot. Una vez definidos estos parámetros, mediante librerías se puede controlar de manera sencilla el robot en simuladores [20].

1) Creación de paquetes de la simulación: Dentro de la carpeta "scr" en la carpeta designada como el espacio de trabajo del proyecto se crean tres paquetes.

a) mecanum_description, especifica toda la estructura del robot, enlaces, articulaciones, complementos y transmisiones. Las propiedades físicas y los parámetros de la dinámica deben estar bien definidos. Además, se configuran las características visuales.

b) mecanum_tools, contiene algunos nodos utilizados para controlar el robot.

c) mecanum_control, especifica el controlador de position/velocity/effort utilizado para mover los servomotores. Además, contiene el archivo de lanzamiento que carga los controladores que se encuentran en el paquete.

2) Modelo del robot omnidireccional utilizando URDF: El robot móvil omnidireccional con cuatro ruedas Mecanum se simula en Gazebo usando el paquete "gazebo_ros". Para ello, se utiliza el archivo "empty_world.launch" del paquete "gazebo_ros" para cargar un entorno vacío en Gazebo. El lanzamiento de la simulación se realiza a través del archivo ubicado en "mecanum _description/launch", donde se crea y define el comportamiento del robot, por lo cual se deben crear dos archivos "xacro". El archivo principal "mecanum_robot_urdf.xacro" carga otro archivo y contiene únicamente elementos URDF, como articulaciones y enlaces. Por otra parte, el archivo "mecanum_robot_gazebo.xacro" contiene especificaciones de Gazebo, tales como las propiedades físicas, enlaces, colores y los complementos de Gazebo que se pueden observar en la Fig. 7.

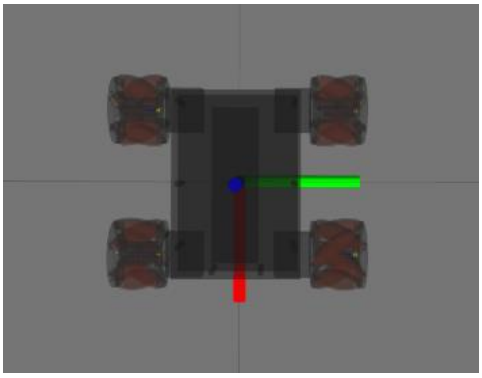

Fig. 7. Vista del modelo de simulación del robot móvil en el plano XY.

Una vez definida la parte visual, se configuraron cuatro controladores del tipo "joint_velocity_controller" del plugin "effort_controller", ya que el robot real tiene un motor para cada rueda (Fig. 8). El paquete "joint_state_publisher" se puede usar para publicar la posición y la transformación de las articulaciones móviles. Luego, el "controller_manager" publica la posición y la transformación de las articulaciones de los rodillos y esas articulaciones pueden girar libremente (Fig. 8) [21].

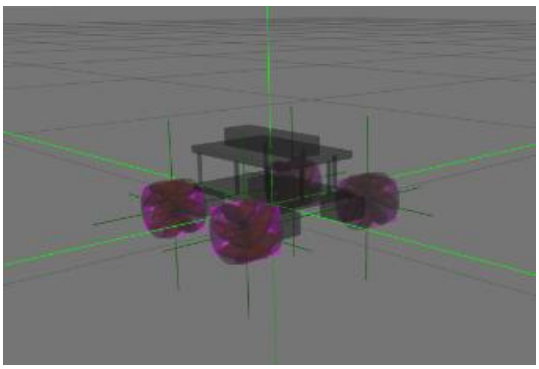

Fig. 8. Modelado 3D del robot omnidireccional.

3) Gazebo Plugin para robots omniwheel: El plugin de Gazebo para controlar robots omnidireccionales de cuatro ruedas Mecanum permite publicar mensajes en el tópico "cmd_vel" con velocidades lineales a lo largo de los ejes X y el eje Y, mientras que para la velocidad angular se lo realiza a lo largo del eje Z. A partir de los datos obtenidos, se calculan las velocidades de cada una de las ruedas de acuerdo con las ecuaciones (7) - (10). Después, se publica la velocidad objetivo de la rueda en el tópico "/command" del modelo del robot omnidireccional y de cada controlador.

El plugin permite seleccionar dos tipos de odometría para publicar: la odometría de la posición real del robot, que publica la posición actual del robot en Gazebo, y la odometría de las ruedas, en donde el plugin busca las articulaciones necesarias en el tópico "joint_states" y su velocidad de rotación actual. Luego, se calcula la velocidad del robot usando las ecuaciones (12) - (14) y finalmente, se calcula la posición actual del robot y se publican los datos obtenidos, lo que permite elegir entre la odometría precisa y la odometría real.

4) Visualización con RViz: Para visualizar el robot en RViz (Fig. 9), se utilizan los nodos "joint_state_controller", que publica los estados de las articulaciones del robot y "robot_state_publisher", que convierte los estados de las articulaciones en transformaciones TF para su visualización en RViz .

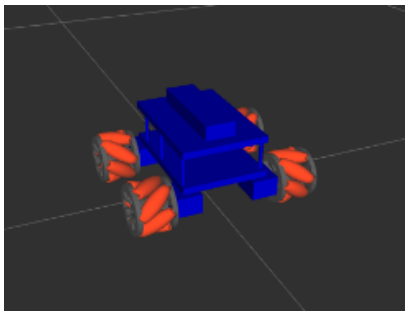

Fig. 9. Simulación del robot móvil en RViz.

E. Diseño de la arquitectura del software

El diseño de la arquitectura del software se concentra en la creación de mecanismos de control de velocidad para los motores. La arquitectura de control se basa en [22], por lo que se ha dividido esta en dos secciones para una aclaración completa. El primer componente se ocupa del control de velocidad, que se denomina control de nivel inferior (OpenCR - servomotores) y es responsable de ejecutar directivas sin ninguna capacidad de decisión. Por el contrario, el control de nivel superior (PC – Raspberry PI - OpenCR) dota al robot de la capacidad de generar trayectorias hacia un lugar específico.

1) Diseño de software de control de nivel inferior: El software OpenCR utiliza el entorno de desarrollo Arduino, lo que permite programar el robot en lenguaje de programación similar a C++ e implementar distintas librerías de Arduino que otorgan acceso a funciones específicas sin la necesidad de escribir todo el código desde cero.

El código fuente del firmware del OpenCR, llamado "core.ino", implementa la lógica de control de los servomotores. En este código, se definen las configuraciones de comunicación, los puertos de los servomotores, los valores de velocidad y posición, y los comandos para enviar y recibir datos. El OpenCR se suscribe y publica varios mensajes para controlar los servomotores. Entre los mensajes más relevantes, están [18]:

- motor_power: Se utiliza para habilitar o deshabilitar la energía de los servomotores. El OpenCR se suscribe a este mensaje para determinar cuándo debe encender o apagar los motores
- cmd_vel: Ejecuta el nodo de control de movimiento para enviar comandos de velocidad lineal y angular al robot. El OpenCR se suscribe a este mensaje para recibir los comandos de movimiento que debe ejecutar.
- joint_states: Publica las posiciones actuales y las velocidades de los servomotores. El OpenCR se suscribe a este mensaje para obtener información sobre la posición y velocidad de cada servomotor.
- battery_state: Publica información sobre el nivel de batería. El OpenCR se suscribe a este mensaje para obtener información sobre el nivel de batería y ajustar su comportamiento en consecuencia.

En cuanto a los servomotores, el OpenCR recibe información de los sensores de posición y velocidad de los mismos, y utiliza la librería "DynamixelSDK" para controlarlos. El OpenCR publica comandos de velocidad y posición a los servomotores Dynamixel a través de la comunicación serial de 115200 baudios.

2) Diseño de software de control de nivel superior: El PC Master es el sistema central que se utiliza para enviar comandos y recibir datos del robot. La PC se encarga de enviar los comandos de movimiento a través del mensaje "cmd_vel" y de registrar los nombres de los tópicos y nodos para establecer la comunicación entre los diferentes nodos y controlar las acciones del robot. Al registrar los nombres de los tópicos y nodos, la PC Master puede asegurarse de que los mensajes se envíen y reciban correctamente entre los diferentes nodos.

La Raspberry Pi se utiliza como una computadora auxiliar, puesto que recibe los comandos de movimiento de la PC Master a través de la red Wi-Fi. Estos comandos son enviados a través de mensajes en el tópico "cmd_vel" y contienen información sobre la velocidad lineal y angular que se desea que el robot tenga. Luego, la Raspberry Pi utiliza los datos de los sensores del robot (como el giroscopio y los encoders de los motores) para estimar la posición y la orientación del robot móvil en el espacio [8]. El OpencCR utiliza esta información junto con el objetivo de trayectoria para calcular los comandos de velocidad que se deben enviar a los servomotores. Este proceso se puede observar en la Fig. 10.

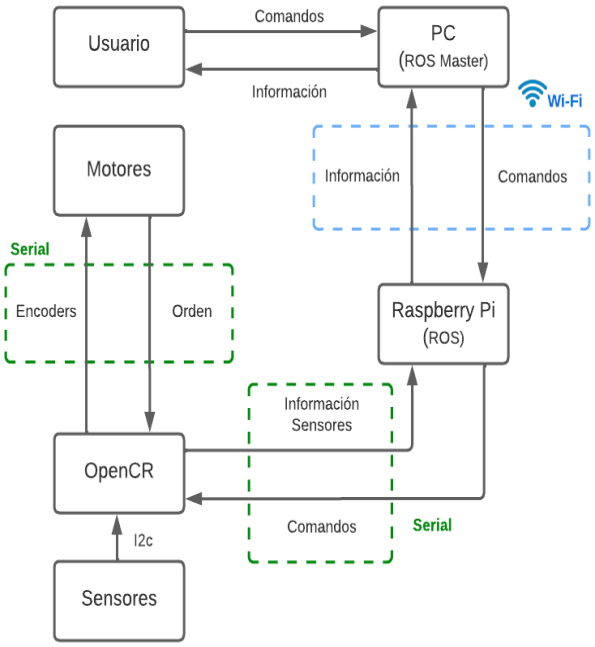

Fig. 10. Esquema de funcionamiento de la plataforma robótica Mecanum.

3) Comunicación: La comunicación entre la PC Master y la Raspberry Pi se realiza a través de Wi-Fi, en donde la Raspberry Pi actúa como un punto de acceso Wi-Fi y la PC Master se conecta a ella como un cliente.

Para establecer la conexión, se utiliza el protocolo de red TCP/IP para enviar paquetes de datos de una máquina a otra a través de la red. La Raspberry Pi actúa como un servidor y escucha las solicitudes entrantes de la PC Master. Una vez que se ha establecido la conexión, la PC Master registra los nombres de los nodos y tópicos de la Raspberry Pi. Esto permite que la PC Master y la Raspberry Pi se comuniquen a través del ROS. La PC Master publica mensajes en los tópicos relevantes y la Raspberry Pi los recibe y los procesa (Fig. 11).

En cambio, la comunicación entre la Raspberry Pi y el OpenCR se realiza a través del puerto serial, en donde la Raspberry Pi envía comandos a través del puerto serial al OpenCR, y este último envía información de retroalimentación y datos de sensor a la Raspberry Pi.

Para lograr esto, se utiliza un nodo de comunicación serial del paquete "rosserial_python", que ejecuta el archivo "serial_node.py" que utiliza los parámetros del puerto conectado "/dev/ttyACM1" y la tasa de baudios 115200, los cuales permiten una comunicación más rápida (Fig. 11).

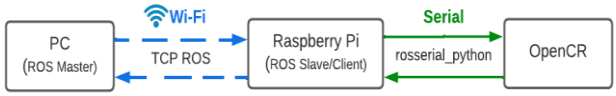

Fig. 11. Sistema de comunicación de la computadora y controlador.

4) Control de trayectoria: El desarrollo del control de trayectoria para un robot móvil omnidireccional se basó en [23]-[24] y se implementaron los siguientes pasos. En primer lugar, se definieron los parámetros del robot y se establecieron los objetivos del control de trayectoria, basados en figuras geométricas descritas por ecuaciones paramétricas:

$$x = f(t) \tag{15}$$
$$y = g(t) \tag{16}$$

Para obtener las velocidades deseadas a lo largo de la trayectoria, se necesita calcular las derivadas temporales de las ecuaciones paramétricas:

$$\frac{dx}{dt} = f'(t) \tag{17}$$
$$\frac{dy}{dt} = g'(t) \tag{18}$$

Luego, con la obtención de los parámetros, se genera la trayectoria deseada utilizando funciones trigonométricas. Además, se utiliza un control cinemático para generar las velocidades lineales y angular del robot, teniendo en cuenta los errores de seguimiento y la aplicación de límites a las velocidades generadas para garantizar la estabilidad del robot. Estas velocidades calculadas se publican en el sistema utilizando mensajes y se envía la información al modelo cinemático del robot en el controlador de nivel bajo, el cual es el encargado de controlar las ruedas y hacer que sigan la trayectoria comandada.

Tanto las velocidades, tiempos y posiciones, son guardadas en un archivo ".txt" en forma de caracteres con la finalidad de procesar la información y verificar los resultados. Finalmente, se realizan pruebas y ajustes del programa, observando el comportamiento del robot. El funcionamiento del script en Python se detalla en el diagrama de la Fig. 12.

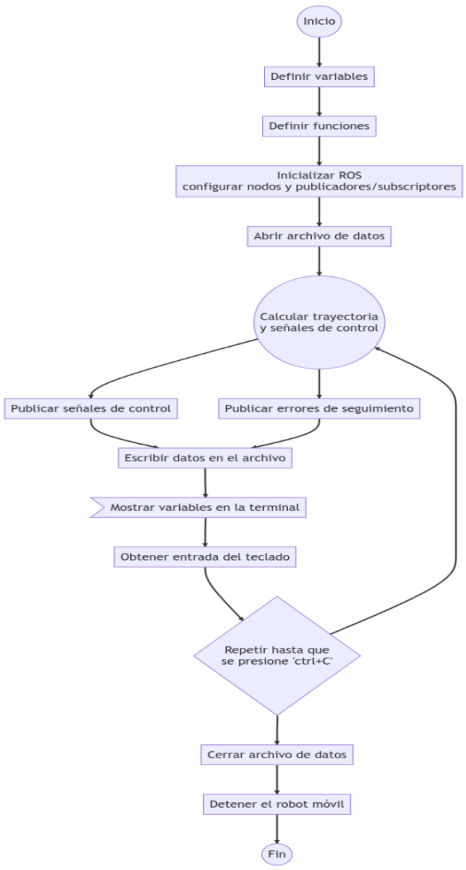

Fig. 12. Diagrama de flujo del script de control de trayectoria.

Una vez generado el archivo ".txt", se procede a cargar los datos del archivo en un script de MATLAB, en donde son realizados los cálculos y visualizaciones de las gráficas relacionadas con trayectorias y señales de control, además de proporcionar métricas de error para evaluar la precisión de las trayectorias realizadas en comparación con las trayectorias deseadas. Esto se detalla en el diagrama de la Fig. 13.

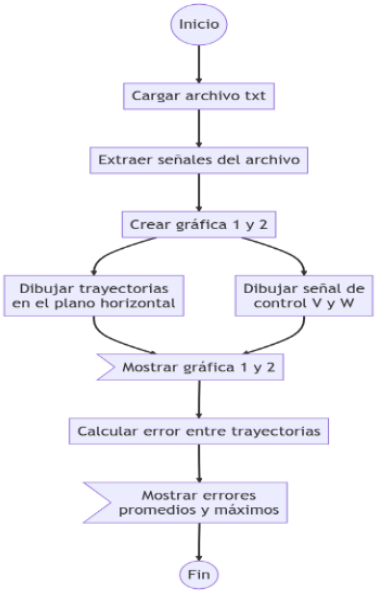

Fig. 13. Diagrama de flujo del script para la graficación de trayectorias, velocidades y errores.

III. Resultados

En esta sección se presentan las pruebas y resultados obtenidos del control de trayectoria de un robot móvil omnidireccional utilizando ROS. El objetivo de estas pruebas fue validar la funcionalidad del sistema de control de trayectoria preestablecida, tanto en la simulación como en la plataforma robótica.

Para llevar a cabo estas pruebas, primero, se realizan las simulaciones para verificar el correcto funcionamiento del modelo del robot en Gazebo. Luego se simularon diferentes trayectorias y se comprobó su funcionamiento en el prototipo real. Las velocidades de las trayectorias y los parámetros para la simulación y el prototipo real son las mismas, y se envían a través del mismo nodo de ROS.

Para la obtención de los errores, primero se calcula el error medio, que es la distancia media entre la posición real del robot y la posición deseada, mientras que el error máximo es la mayor distancia entre la posición real del robot y la posición deseada.

Todas las simulaciones se realizaron con los parámetros de la TABLA I:

TABLA I. Velocidades lineal y angular máximas.

V_max	0.22 m/s
W_max	2.84 rad/s

A. *Lemniscata.*

Las ecuaciones paramétricas de esta figura son:

$$x = X_0 + a * sin(w * t) \quad (19)$$
$$y = Y_0 + b * sin(2w * t) \quad (20)$$

Donde:

X_0 e Y_0 son las coordenadas del centro de la lemniscata,
a y b son los parámetros de escala que determinan el tamaño de la lemniscata,
w es la frecuencia angular y t es el tiempo.

Las derivadas de (19) y (20) (velocidades) corresponden a las siguientes ecuaciones:

$$\frac{dx}{dt} = a * w * cos(w * t) \quad (21)$$
$$\frac{dy}{dt} = 2b * w * cos(2w * t) \quad (22)$$

1) Prueba en el simulador: Las pruebas se iniciaron en el simulador Gazebo, en donde la trayectoria fue comandada desde un nodo de control de trayectoria (Fig. 12). El robot pudo seguir con éxito la trayectoria de la lemniscata con los parámetros de la TABLA I. Para la lemniscata, el semieje mayor (a) se fijó en 1.0 m y el semieje menor (b) en 0.5 m. La simulación de la trayectoria se puede evidenciar en la Fig.14 mientras su trayectoria en el plano se observa en la Fig. 15.

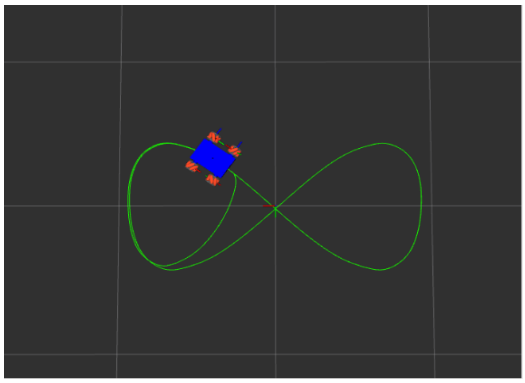

Fig. 14. Simulación de trayectoria del robot móvil sobre lemniscata.

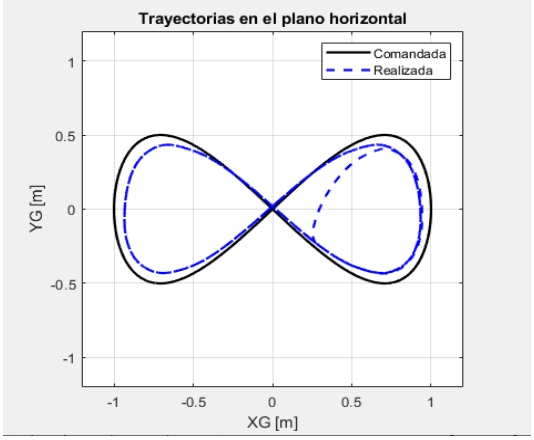

Fig. 15. Gráfica de trayectoria del robot móvil sobre lemniscata.

La velocidad lineal del robot aumenta a medida que se aleja del centro de la lemniscata y es mínima en el centro. La velocidad angular del robot aumenta a medida que se aleja del centro de la lemniscata, y alcanza su máximo en el punto más lejano (Fig. 16). Los errores se muestran en la TABLA II.

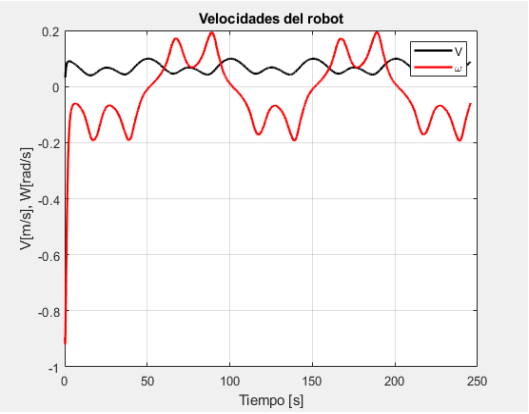

Fig. 16. Gráfica de las velocidades lineales y angulares sobre la lemniscata.

TABLA II. Error de simulación en lemniscata.

Error promedio	0.53427cm
Error máximo	1.2019 cm

2) Prueba en el prototipo: Se realizaron pruebas con el prototipo real del robot para evaluar su capacidad de seguir una trayectoria de lemniscata. El prototipo demostró un seguimiento estable de la trayectoria, validando así la efectividad del control implementado en un entorno físico.

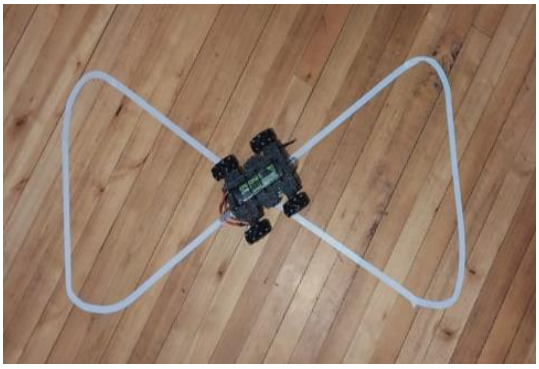

Fig. 17. Trayectoria de prototipo real sobre lemniscata.

B. *Cuadrado.*

1) Prueba en el simulador: Para la trayectoria del cuadrado en el simulador, se establecieron los mismos parámetros de velocidad anteriores sobre un cuadrado de 1.0 m de lado. En la Fig. 18 y la Fig. 19, se logra evidenciar como se completa la trayectoria con la figura deseada.

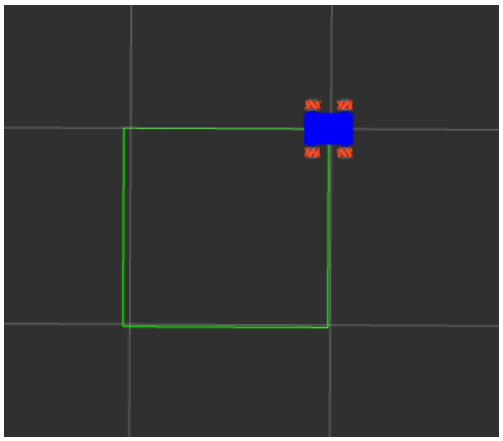

Fig. 18. Simulación de trayectoria del robot móvil sobre cuadrado.

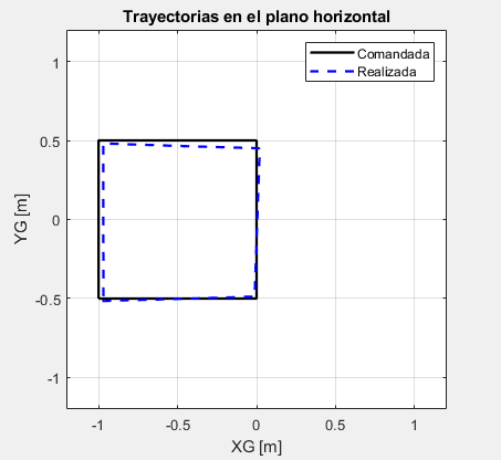

Fig. 19. Gráfica de trayectoria del robot móvil sobre cuadrado.

La velocidad lineal del robot aumenta a medida que se aleja del vértice del cuadrado y disminuye al acercase al siguiente vértice. La velocidad angular del robot aumenta en el vértice debido a que el robot realiza un giro de 90 grados. Esto se evidencia en la Fig. 20. Los errores se muestran en la TABLA III.

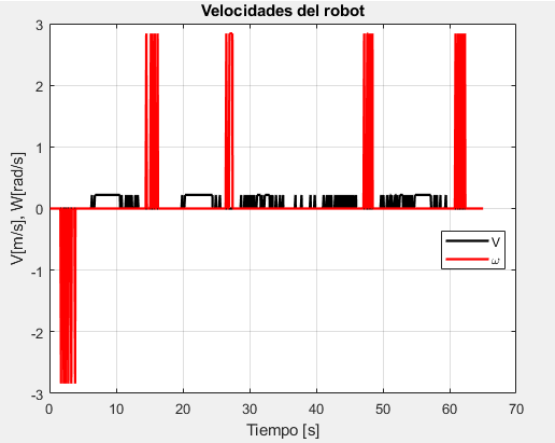

Fig. 20. Gráfica de las velocidades lineales y angulares sobre cuadrado.

TABLA III. ERROR DE SIMULACIÓN EN CUADRADO.

Error promedio	2.5511 cm
Error máximo	4.5986 cm

2) Prueba en el prototipo: En la prueba de la Fig. 21, se evidencia que el robot completa la trayectoria propuesta.

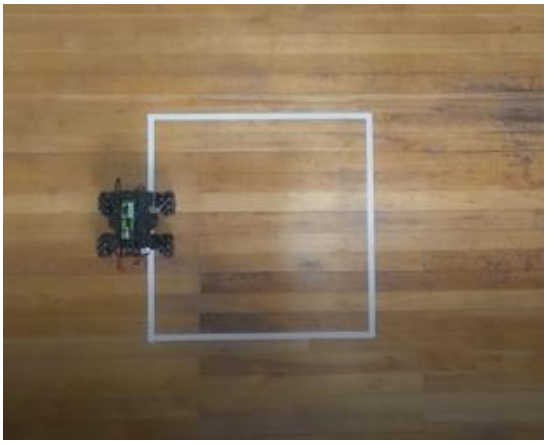

Fig. 21. Trayectoria de prototipo real sobre cuadrado.

C. *Circunferencia.*

Las ecuaciones paramétricas de esta figura son:

$$x = X_0 + a * sin(w * t) \quad (23)$$
$$y = Y_0 + a * cos(w * t) \quad (24)$$

Donde:

X_0 e Y_0 son las coordenadas del centro de la circunferencia,
a es el radio de la circunferencia,
w es la frecuencia angular y t es el tiempo.

Las derivadas de (23) y (24) (velocidades) corresponden a las siguientes ecuaciones:

$$\frac{dx}{dt} = a * w * cos(w * t) \quad (25)$$

$$\frac{dy}{dt} = -a * w * cos(w * t) \quad (26)$$

1) Prueba en el simulador: Para la trayectoria de la circunferencia en el simulador, se utilizaron las mismas configuraciones de velocidad que en las anteriores figuras, estableciendo un radio de un 0.5 m. En la Fig. 22 y la Fig. 23, se logra evidenciar como se completa la trayectoria con la figura deseada.

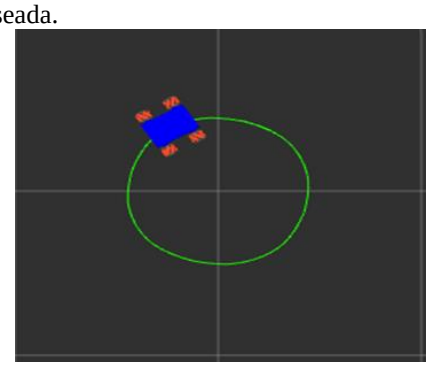

Fig. 22. Simulación de trayectoria del robot móvil sobre la circunferencia.

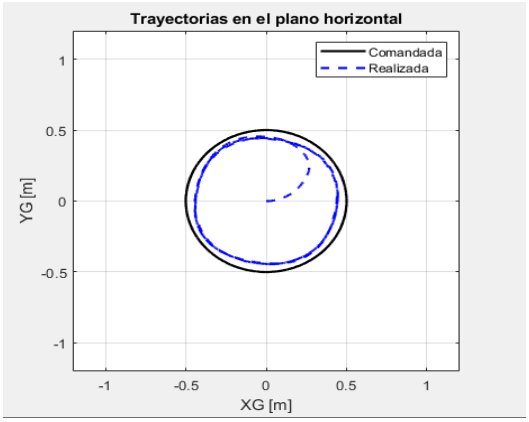

Fig. 23. Gráfica de trayectoria del robot móvil sobre circunferencia.

La velocidad lineal del robot se mantiene constante a lo largo de la circunferencia. La velocidad angular del robot oscila dentro de un rango muy pequeño debido a que se ajusta al radio de la circunferencia. Esto se evidencia en la Fig. 24. Los errores se muestran en la TABLA IV.

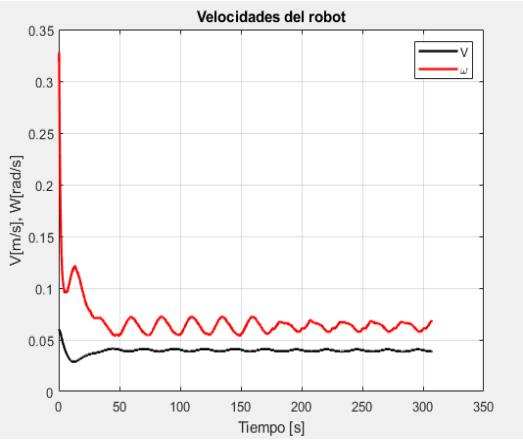

Fig. 24. Gráfica de las velocidades lineales y angulares sobre circunferencia.

TABLA IV. ERROR DE SIMULACIÓN EN CIRCUNFERENCIA.

Error promedio	0.57503cm
Error máximo	1.6669

2) Prueba en el prototipo: En la siguiente prueba (Fig. 25), se evidencia que el robot completa la trayectoria de manera satisfactoria.

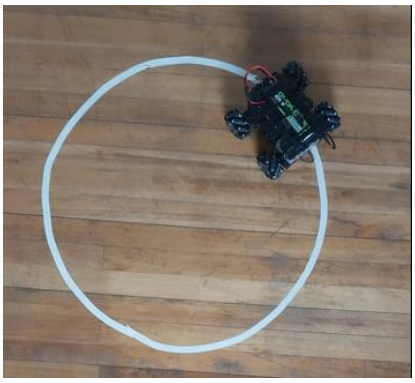

Fig. 25. Trayectoria de prototipo real sobre circunferencia.

IV. CONCLUSIONES

En conclusión, este estudio ha validado y evaluado de manera satisfactoria el sistema de control de trayectoria de un robot móvil omnidireccional utilizando ROS. Los resultados obtenidos respaldan la efectividad y viabilidad del sistema implementado tanto en simulación como en un prototipo real.

Se pudo observar que el sistema de control de trayectoria fue capaz de seguir de manera eficaz las trayectorias preestablecidas, tanto en el entorno simulado como en el entorno físico. Los errores promedio y máximo calculados para las diferentes trayectorias fueron relativamente bajos, lo que indica una buena capacidad del modelo para predecir y simular el comportamiento del robot en diversas situaciones y trayectorias. Además, el prototipo real del robot demostró un seguimiento estable de las trayectorias propuestas, validando así la efectividad del control implementado en un entorno físico.

Para añadir una navegación autónoma, se recomienda el uso adicional de sensores que ayuden a la odometría y posicionamiento del robot en el entorno. Estos podrían ser LIDARs (Laser Imaging Detection and Ranging) o cámaras con visión artificial. Se debe mencionar que el prototipo ya cuenta con encoders en los motores y una IMU (Inertial Measurement Unit), las cuales podrían ser combinadas para hacer navegación por estimación.

V. REFERENCIAS

[1] S. Mishra, M. Sharma, y S. Mohan, “Behavioural Fault tolerant control of an Omni directional Mobile Robot with Four mecanum Wheels,” Def. Sci. J., vol. 69, no. 4, pp. 353–360, 2019.

[2] V. Alakshendra y S. S. Chiddarwar, “A robust adaptive control of mecanum wheel mobile robot: simulation and experimental validation,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 5606–5611.

[3] J. Ortega y L. Yapo, “Construcción de un Robot Móvil Híbrido Omnidireccional”, 2017.

[4] M. Tatar, C. Popovici, D. Mandru, I. Ardelean, y A. Plesa, “Design and development of an autonomous omni-directional mobile robot with Mecanum wheels”, en *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, 2014, pp. 1–6.

[5] P. Yadav, V. Agrawal, J. Mohanta, y MD. Faiyaz, “A theoretical review of mobile robot locomotion based on mecanum wheels”, Evergreen, vol. 9, núm. 2, pp. 396–403, 2022.

[6] M. Hijikata, R. Miyagusuku, y K. Ozaki, “Wheel arrangement of four Omni wheel mobile robot for compactness,” Appl. Sci. (Basel), vol. 12, no. 12, p. 5798, 2022.

[7] L. Gallo, B. Paste, J. Ortiz, y V. Andaluz, “Control of an omnidirectional robot based on the kinematic and dynamic model”, en Communications in Computer and Information Science, Cham: Springer International Publishing, 2021, pp. 444–457.

[8] K. Piemngam, I. Nilkhamhang, y P. Bunnun, “Development of autonomous mobile robot platform with mecanum wheels”, en *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, 2019, pp. 90–93.

[9] M. Wu, S.-L. Dai, y C. Yang, “Mixed reality enhanced user interactive path planning for omnidirectional mobile robot”, *Appl. Sci. (Basel)*, vol. 10, núm. 3, p. 1135, 2020.

[10] A. Cabrera y J. Reinozo, “Diseño y construcción de un robot móvil omnidireccional basado en ROS para la enseñanza de conceptos matemáticos a estudiantes de bachillerato”, 2022.

[11] H. Taheri, B. Qiao, y N. Ghaeminezhad, “Kinematic model of a four mecanum wheeled mobile robot”, *International journal of computer applications*, vol. 113, pp. 6–9, 2015.

[12] S. Tzafestas, Introduction to Mobile Robot Control. Philadelphia, PA: Elsevier Science Publishing, 2017

[13] B. Woods, “Omni-Directional Wheelchair”, The University of Western Australia, 2006.

[14] Raspberry Pi Foundation, "Raspberry Pi 4 Model B," Raspberry Pi, [En línea]. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[15] ROBOTIS, "Dynamixel XL430-W250-T Servo," ROBOTIS e-Manual, [En línea]. Disponible en: http://emanual.robotis.com/docs/en/dxl/x/xl430-w250/.

[16] ROBOTIS, "OpenCR 1.0," ROBOTIS e-Manual, [En línea]. Disponible en: http://emanual.robotis.com/docs/en/parts/controller/opencr10/. [Consultado el 11 de junio de 2023].

[17] C. Fairchild y T. L. Harman, *ROS Robotics By Example*, 2a ed. Birmingham, England: Packt Publishing, 2017.

[18] P. Yoonseok, C. Hancheol, J. Leon, y L. Darby, ROS *Robot Programming*. ROBOTIS, 2017.

[19] W. Newman, *A systematic approach to learning robot programming with ROS*, 1st Edition. Boca Raton: CRC Press, [2017]: Chapman and Hall/CRC, 2017.

[20] P. Mascaró, “Modelado virtual y simulación de vehículos autónomos en Gazebo”, Universidad Carlos III de Madrid, 2018.

[21] A. Apurin, B. Abbyasov, A. Dobrokvashina, Y. Bai, M. Svinin, y E. Magid, “Omniwheel chassis model and plugin for gazebo simulator”, Alife-robotics, 2023.

[22] A. Neaz, S. Lee, y K. Nam, “Design and implementation of an integrated control system for omnidirectional mobile robots in industrial logistics”, Sensors (Basel), vol. 23, núm. 6, p. 3184, 2023.

[23] Y. Li et al., “Kinematic modeling of a combined system of multiple Mecanum-wheeled robots with velocity compensation,” Sensors (Basel), vol. 20, no. 1, p. 75, 2019.

[24] P. Manzl, *Realtime movement of a mecanum wheeled robot using the robot operating system ROS*, University Innsbruck, 2020.