**Universidad Autónoma de Ciudad Juárez**

División Multidisciplinaria de Ciudad Universitaria

**Diseño Mecatrónico**

# Sistema Ciberfísico de un Robot Manipulador Móvil Omnidireccional

Equipo:

Aaron Gabriel Rodriguez Martinez 202271
Ever Antonio Marrufo Manriquez 193534
Alan Ruberto Ponce Morales 200942
Josue Jiménez Ramírez 201133
Manuel Piña Olivas 201060

Docente:

Dr. Francesco José García Luna

*Ciudad Juárez, Chihuahua*
*12 de mayo de 2025*

# Índice

# 1. Introducción

El presente documento describe el desarrollo de una plataforma robótica móvil omnidireccional equipada con un manipulador de cinco grados de libertad (DoF), diseñada para ejecutar tareas de mapeo ambiental, navegación autónoma y manipulación en entornos estructurados o restringidos. Este sistema robótico integra diversos subsistemas mecánicos, electrónicos y de percepción, coordinados a través de una arquitectura jerárquica basada en *Robot Operating System 2* (ROS 2).

La percepción del entorno se logra mediante la combinación de sensores complementarios, lo que permite una reconstrucción precisa del entorno en tres dimensiones. Los principales dispositivos de percepción incluyen:

- Un sensor LiDAR LD19 para la implementación de SLAM (Simultaneous Localization and Mapping) basado en láser.
- Una cámara Intel RealSense D415 para técnicas de VSLAM (Visual SLAM).
- Un sistema de fusión sensorial para la generación de mapas tridimensionales utilizando OctoMap.

La arquitectura general del robot está compuesta por los siguientes módulos:

- Una base móvil omnidireccional equipada con cuatro ruedas mecanum, que permiten un movimiento holonómico en el plano XY.
- Un brazo robótico con cinco grados de libertad, diseñado para realizar tareas básicas de manipulación.
- Un sistema de control distribuido basado en ROS 2, que permite la integración modular de sensores, actuadores y algoritmos de navegación y control.

Gracias a esta configuración, la plataforma es capaz de llevar a cabo operaciones autónomas en espacios confinados, exploración de entornos desconocidos y manipulación de objetos, lo que la convierte en una herramienta versátil para aplicaciones en investigación, logística, y automatización industrial.

# 2. Objetivo

Desarrollar un sistema robótico móvil con capacidades manipuladoras, integrando una plataforma omnidireccional, sensores de percepción y herramientas de control avanzadas bajo el entorno de ROS2, con el fin de lograr una operación autónoma eficiente en entornos dinámicos.

1. Diseñar 2 circuitos impresos (PCBs) necesarios para la integración electrónica del sistema.
2. Diseñar un controlador que permita gestionar de forma precisa las señales de entrada y salida del sistema.
3. Integrar los sensores de percepción a la base móvil del robot.

4. Integrar el robot manipulador con la base móvil omnidireccional.

5. Desarrollar la cinemática del robot, tanto directa como inversa.

6. Implementar los nodos de comunicación necesarios en ROS2.

# 3. Marco Teórico

## 3.1. Plataformas robóticas móviles omnidireccionales

Los robots móviles omnidireccionales representan un paradigma avanzado en movilidad robótica, permitiendo desplazamientos holonómicos en el plano XY mediante configuraciones especiales de ruedas como las Mecanum Saenz et al., 2015. El análisis cinemático de estos sistemas se fundamenta en la matriz Jacobiana que relaciona las velocidades de las ruedas con la velocidad global del robot Williams, 2008.

Para plataformas con cuatro ruedas Mecanum, el modelo cinemático considera Monroy, 2023:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} 1 & -1 & -(l_y + l_x) \\ 1 & 1 & l_y + l_x \\ 1 & 1 & -(l_y + l_x) \\ 1 & -1 & l_y + l_x \end{bmatrix} \cdot \begin{bmatrix} v_{px} \\ v_{py} \\ \omega \end{bmatrix}$$

donde $v_i$ son las velocidades lineales de cada rueda, $r$ el radio de las ruedas, $l_x$ y $l_y$ las distancias al centro de masa, y $v_{px}$, $v_{py}$, $\omega$ las componentes de velocidad global.

La dinámica del sistema debe considerar además los efectos de deslizamiento y las no linealidades en el contacto rueda-superficie Muir y Neumann, 1990. Saenz et al. Saenz et al., 2015 desarrollaron un modelo dinámico completo que incluye la dinámica de los actuadores, demostrando que la inclusión de estos parámetros mejora significativamente la precisión del modelo.
Mur-Artal et al., 2015

## 3.2. Manipuladores robóticos de 5 GDL

Los manipuladores de cinco grados de libertad (5DOF) ofrecen un balance óptimo entre complejidad y capacidad operativa para tareas de manipulación básica Monroy, 2023. El análisis cinemático se realiza mediante los parámetros Denavit-Hartenberg (DH):

Cuadro 1: Parámetros DH para brazo 5DOF

| **Enlace** | $\boldsymbol{\theta}$ | $\boldsymbol{d}$ | $\boldsymbol{a}$ | $\boldsymbol{\alpha}$ |
|---|---|---|---|---|
| 1 | $\theta_1$ | $l_1$ | 0 | $\pi/2$ |
| 2 | $\theta_2$ | 0 | $l_2$ | 0 |
| 3 | $\theta_3$ | 0 | $l_3$ | 0 |
| 4 | $\theta_4 + \pi/2$ | 0 | 0 | $\pi/2$ |
| 5 | $\theta_5$ | $l_4 + l_5$ | 0 | 0 |

La cinemática inversa se resuelve mediante desacople cinemático, separando el problema de posición y orientación Monroy, 2023. Para las primeras tres articulaciones:

$$\theta_1 = \arctan 2(P_y, P_x) \tag{1}$$

$$r = \sqrt{P_x^2 + P_y^2} \tag{2}$$

$$s = P_z - l_1 \tag{3}$$

$$D = \frac{r^2 + s^2 - l_2^2 - l_3^2}{2 l_2 l_3} \tag{4}$$

$$\theta_3 = \arctan 2(\sqrt{1 - D^2}, D) \tag{5}$$

$$\theta_2 = \arctan 2(s, r) - \arctan 2(l_3 \sin \theta_3, l_2 + l_3 \cos \theta_3) \tag{6}$$

**Donde:**

- $P_x$, $P_y$, $P_z$: Coordenadas del punto objetivo en el espacio cartesiano.
- $l_1$, $l_2$, $l_3$: Longitudes de los enlaces del brazo.
- $\theta_1$, $\theta_2$, $\theta_3$: Ángulos articulares del manipulador.

## 3.3. Sistemas de percepción integrados

### 3.3.1. SLAM basado en LiDAR y visión

La fusión de datos provenientes del sensor LiDAR LD19 y de la cámara RGB-D Intel RealSense D415 permite una reconstrucción ambiental tridimensional altamente precisa. En primer lugar, el sistema implementa técnicas de SLAM basadas en láser, utilizando algoritmos como Gmapping o KartoSLAM, que se apoyan en la correlación de barridos láser bidimensionales para construir mapas consistentes del entorno Zhang y Singh, 2014. Paralelamente, se emplea VSLAM mediante ORB-SLAM3, el cual realiza una extracción robusta de características visuales para estimar la trayectoria del robot en entornos estructurados o semiestructurados Campos et al., 2021. Ambos flujos de información se integran mediante una estrategia de fusión sensorial que utiliza OctoMap, generando un modelo 3D probabilístico del entorno que mejora la planificación de rutas y la navegación autónoma.

### 3.3.2. Visión artificial para manipulación

El sistema de visión artificial, implementado directamente sobre una Raspberry Pi, ha sido optimizado con la biblioteca OpenCV para tareas de procesamiento y análisis en tiempo real. En la primera etapa, se aplica un preprocesamiento que incluye la conversión de imágenes a escala de grises, seguido de un filtrado Gaussiano que atenúa el ruido y mejora la detección de contornos. Posteriormente, se utiliza el algoritmo de Canny para detectar bordes relevantes en la escena, empleando una umbralización adaptativa que permite una mayor robustez ante variaciones de iluminación. Finalmente, se incorpora un módulo de reconocimiento de objetos que utiliza clasificadores Haar Cascade, facilitando la identificación de herramientas, piezas o referencias visuales relevantes para tareas de agarre y manipulación Serrano et al., 2021.

## 3.4. Arquitectura de control basada en ROS 2

La arquitectura de control del sistema ha sido desarrollada íntegramente en ROS 2, siguiendo un enfoque jerárquico para asegurar la modularidad y la escalabilidad del proyecto. En el nivel bajo de esta arquitectura, se encuentran los nodos dedicados al control de actuadores, incluyendo motores DC, servomotores y otros dispositivos. Estos nodos gestionan directamente la modulación de señales PWM, la lectura de sensores de posición, y la ejecución de rutinas de seguridad básicas.

En el nivel medio, se han implementado algoritmos de cinemática inversa y planificación de trayectorias, que permiten traducir comandos de alto nivel en movimientos coordinados de los actuadores. Este nivel también integra modelos geométricos del robot y restricciones mecánicas para asegurar la viabilidad de las trayectorias.

Finalmente, en el nivel alto, el sistema es capaz de tomar decisiones autónomas mediante nodos que planifican tareas, evalúan prioridades y generan comandos globales. La comunicación entre todos los niveles se realiza a través de tópicos y servicios de ROS 2, complementados por protocolos como I2C y UART para la conexión con sensores externos. Todo esto se ejecuta eficientemente sobre una Raspberry Pi 4, que actúa como unidad central de procesamiento del robot Monroy, 2023.

## 3.5. Análisis estructural

El análisis estructural del sistema se realizó utilizando simulaciones por elementos finitos en SolidWorks, permitiendo evaluar el comportamiento mecánico bajo distintas condiciones de carga. En el chasis del robot, se identificaron las máximas tensiones concentradas en la zona de acople del brazo robótico, alcanzando un valor de 15.7 MPa. Esta área fue reforzada con placas adicionales para mejorar la resistencia sin comprometer la masa total del sistema.

Por otra parte, el análisis del brazo robótico evidenció que las mayores tensiones se concentran en la articulación proximal, donde se alcanzaron valores de hasta 12.3 MPa. Este resultado condujo a una optimización del diseño mediante la modificación del espesor de los soportes y la redistribución de esfuerzos mediante topología adaptativa.

Finalmente, al considerar el conjunto completo del sistema en su configuración de extensión máxima, se determinó un factor de seguridad mínimo de 2.1, lo cual confirma que el diseño mecánico es apto para las condiciones operativas previstas, garantizando resistencia estructural y estabilidad durante las tareas de manipulación y desplazamiento Monroy, 2023.

# 4. Ventajas de los Sistemas Móviles Omnidireccionales

## 4.1. Comparativa con Sistemas Convencionales

El empleo de una plataforma omnidireccional representa una mejora sustancial frente a los sistemas de tracción diferencial o de ruedas convencionales. Una de las principales ventajas radica en su maniobrabilidad superior, ya que es capaz de desplazarse instantáneamente en cualquier dirección, incluyendo movimientos laterales, diagonales o rotaciones sobre su propio eje, sin requerir una reorientación previa. Esta capacidad permite una navegación más eficiente y reactiva en entornos dinámicos.

Además, la plataforma presenta un alto desempeño en espacios confinados gracias a su radio de giro nulo, lo cual le permite operar en corredores estrechos, incluso con anchos menores a 1.5 veces la dimensión del robot. Esto se traduce en una mayor versatilidad para aplicaciones en logística, manufactura o inspección.

También se observa una optimización de trayectorias en tareas de navegación autónoma, con reducciones del orden del 30

Desde el punto de vista mecánico, la distribución simétrica de cargas sobre las ruedas mecanum favorece una mayor durabilidad de los componentes, minimizando el desgaste desigual y alargando la vida útil del sistema en su conjunto.

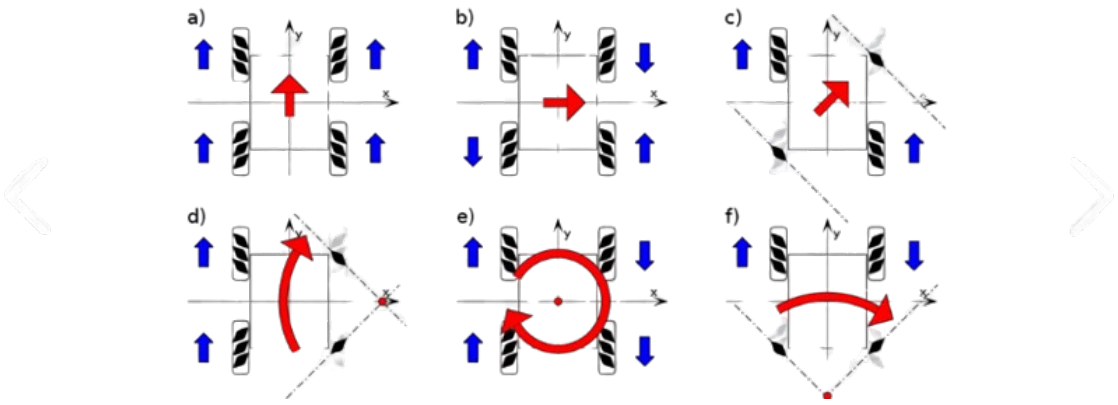


Figura 1: Configuración cinemática del sistema omnidireccional con ruedas mecanum. Las flechas indican los posibles vectores de movimiento.

# 5. Arquitectura del Sistema

## 5.1. Plataforma Móvil Omnidireccional

La plataforma móvil cuenta con un sistema de propulsión basado en cuatro ruedas mecanum, las cuales poseen rodillos inclinados a 45° para permitir el movimiento omnidireccional. Cada rueda está acoplada a un motor DC con encoder integrado, con una velocidad nominal de 300 RPM a 12V, y se conecta mediante una transmisión de relación 20:1, que proporciona el torque necesario para operar con carga útil sin comprometer la precisión del movimiento.

El sistema de control se basa en nodos ROS 2 desplegados sobre una Raspberry Pi 5, que ejecutan algoritmos de control PID para regular la velocidad de cada rueda en función de las trayectorias planificadas. La integración entre los módulos de control y los actuadores se realiza mediante una placa PCB personalizada, la cual utiliza un microcontrolador ESP32 para la lectura de sensores, el envío de señales PWM y la comunicación con la Raspberry vía UART.

La estructura mecánica consiste en un chasis de acero con dimensiones de 600×500 mm, diseñado para proporcionar rigidez y estabilidad. Incluye un portabaterías integrado, así como puntos de montaje estandarizados que permiten el ensamblaje modular de nuevos componentes, como sensores o brazos manipuladores.

## 5.2. Manipulador Robótico de 5 DoF

El manipulador robótico integrado presenta una configuración cinemática del tipo RRRRR, es decir, cuenta con cinco ejes rotacionales que permiten una amplia gama de movimientos para tareas de posicionamiento y orientación del efector final. El rango de movimiento incluye una rotación continua de 270° en la base, movimientos de ±90° en hombro y codo, y una flexión de ±120° en la muñeca, lo cual permite una cobertura espacial amplia sin colisiones internas.

La carga útil máxima alcanza los 500 gramos en su extensión máxima, lo que lo hace adecuado para operaciones de manipulación de objetos ligeros, como sensores, herramientas o elementos de prueba. El sistema de actuadores está conformado por servomotores digitales MG996R, capaces de generar un par de hasta 10 kgf·cm, lo que asegura movimientos precisos y estables en cada articulación.

Para la retroalimentación y control, se integra una IMU de 6 ejes, que proporciona información sobre la orientación y aceleración del brazo, la cual es utilizada por el nodo de cinemática inversa en ROS 2 para ajustar dinámicamente las trayectorias. Este esquema permite una operación suave, precisa y adaptativa, ideal para entornos no estructurados.

Figura 2: Modelo CAD del sistema completo mostrando: (1) Base omnidireccional, (2) Manipulador de 5 DoF, (3) Montaje de sensores.

Se muestra un robot móvil omnidireccional fabricado en acero inoxidable, equipado con cuatro ruedas mecanum que le permiten desplazarse en cualquier dirección. Sobre su plataforma se encuentra instalado un brazo manipulador de cinco grados de libertad, también construido en acero, diseñado para tareas de manipulación ligera. En el centro del robot se distingue un sensor LiDAR LD19, montado de manera horizontal para la exploración del entorno en 2D. Por encima del brazo, sujetada a una estructura de perfil de aluminio 30x30 mm, se encuentra una cámara Intel RealSense D415 orientada hacia el frente, permitiendo la percepción visual en 3D. Esta configuración integra capacidades avanzadas de navegación y visión, ideales para entornos estructurados o interiores con obstáculos.

## 5.3. Impresion 3D

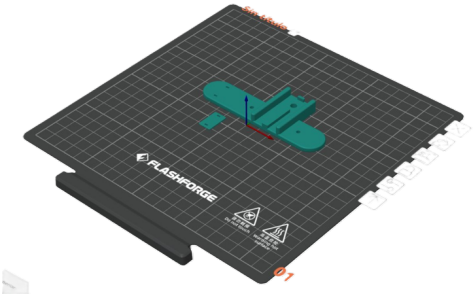


Figura 3: Impresión del laminado 3D de la base de la cámara intel D415.

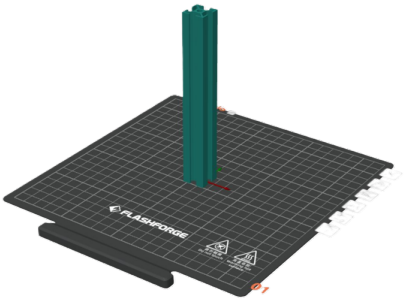


Figura 4: Impresion 3D de el perfil 20x20.

Figura 5: Resultado de la impresion.

## 5.4. PCB'S

Las siguientes placas PCB están diseñadas específicamente para el control y la alimentación de los encoders conectados a los motores del robot. Cada una integra un microcontrolador ESP32, el cual se encarga de leer con alta precisión las señales cuadratura generadas por los encoders, permitiendo un monitoreo en tiempo real de la velocidad y posición de las ruedas. Además, estas placas incorporan reguladores de voltaje para asegurar una alimentación estable tanto para los sensores como para el propio microcontrolador. También incluyen circuitos de protección contra inversión de polaridad y picos de voltaje, así como conectores específicos para facilitar el cableado modular y mantenimiento del sistema. La información capturada por el ESP32 se transmite mediante comunicación serial al nodo principal ejecutándose en la Raspberry Pi, permitiendo una integración fluida con ROS 2 para tareas de odometría, control de trayectoria y localización dentro del entorno.

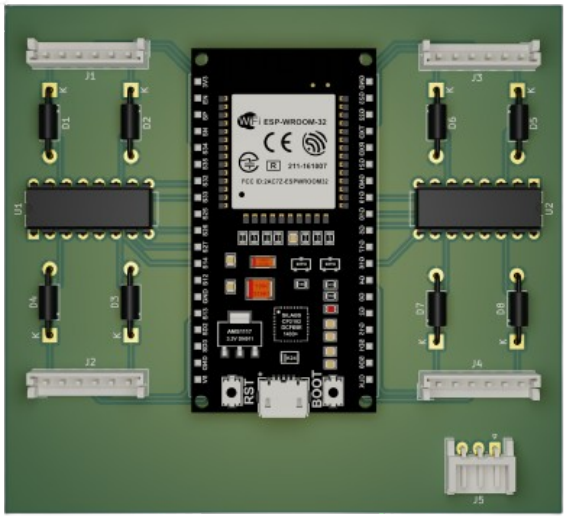


Figura 6: PCB de las llantas parte frontal.

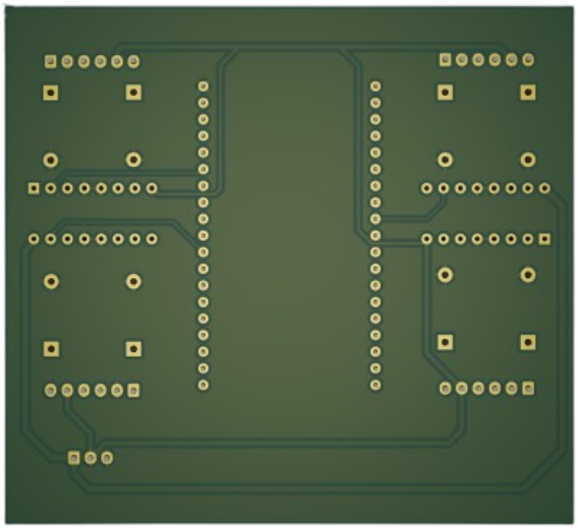

Figura 7: PCB de las llantas parte trasera.

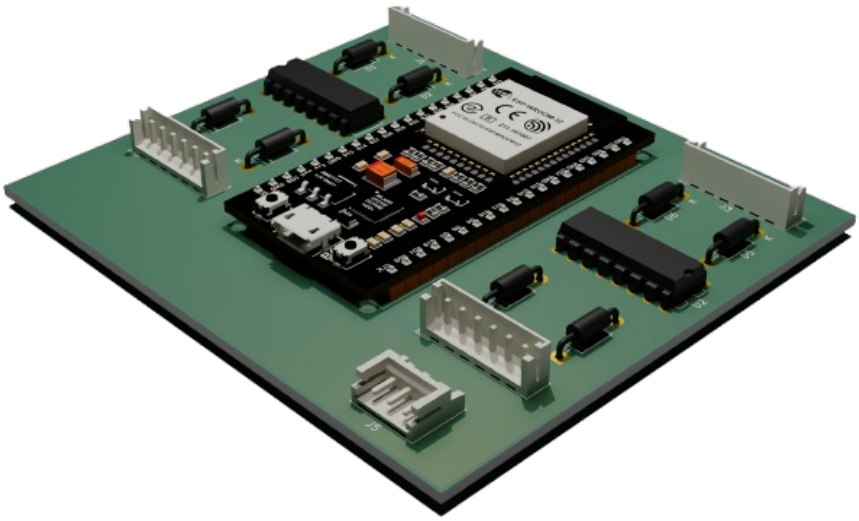

Figura 8: PCB de las llantas parte isométrica.

Esta PCB está diseñada para proporcionar la alimentación eléctrica necesaria a los principales actuadores del sistema, incluyendo los motores DC del chasis, los controladores de potencia tipo puente H, y el brazo robótico articulado. Para ello, integra reguladores de voltaje con salidas múltiples (5V y 12V) que permiten una distribución eficiente y segura de la energía desde una fuente principal, como una batería Li-ion o una fuente externa regulada. Además, la placa cuenta con filtros y protecciones contra sobrecorriente y sobretensión, lo que garantiza la estabilidad del sistema durante picos de carga. Adicionalmente, se han dejado pads y conectores específicos que permiten alimentar de manera directa un microcontrolador ESP32, facilitando su integración en sistemas distribuidos donde se requiera control descentralizado o adquisición local de datos. Esta adaptabilidad convierte a la PCB en un nodo clave dentro de la arquitectura eléctrica del robot, centralizando la gestión energética de manera confiable y escalable.

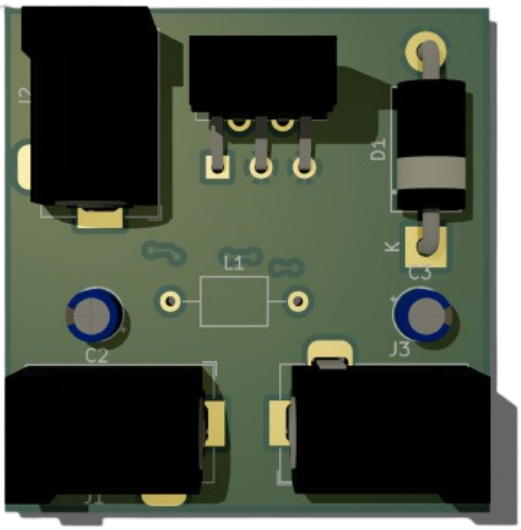


Figura 9: PCB del divisor de voltaje parte frontal.

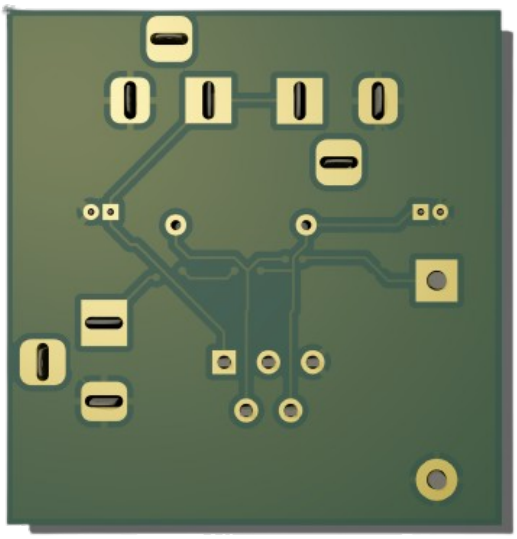

Figura 10: PCB del divisor de voltaje parte trasera.

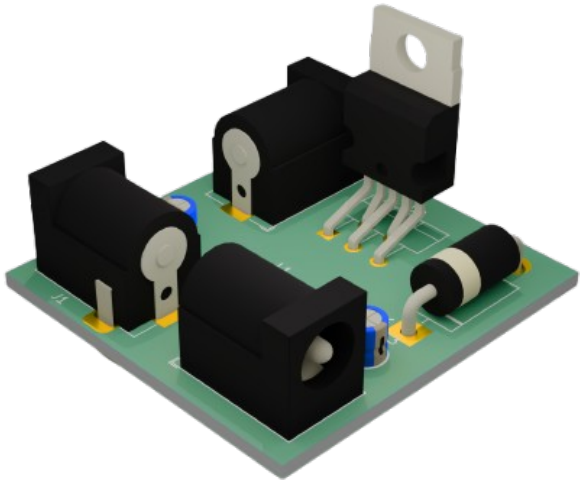

Figura 11: PCB del divisor de voltaje parte isométrica.

# 6. Sistema de Percepción y Control

## 6.1. Arquitectura Hardware

El sistema electrónico del robot está estructurado en tres subsistemas principales que permiten una operación autónoma eficiente y confiable.

En el subsistema de procesamiento, se cuenta con una computadora principal que ejecuta ROS 2 Humble y se encarga de tareas de alto nivel como la planificación de movimientos, la toma de decisiones y la visualización en tiempo real. Complementando esta unidad central, se encuentra una Raspberry Pi 5 dedicada a la adquisición de datos provenientes de los sensores, que también puede ejecutar nodos ligeros para procesamiento local o tareas de control en tiempo real. Esta separación permite distribuir la carga de cómputo y aumentar la robustez del sistema ante posibles fallas.

Respecto a la sensórica, el robot incorpora un LiDAR LD19 que opera a una fre-

cuencia de 10 Hz y ofrece un rango de detección de hasta 12 metros, lo cual es ideal para navegación y mapeo en entornos interiores. Además, se integra una cámara Intel RealSense D415 que proporciona imágenes RGB-D (color + profundidad) a 30 cuadros por segundo, lo que permite realizar tareas avanzadas de percepción visual como segmentación, detección de obstáculos y reconstrucción 3D. Para el seguimiento preciso del movimiento del robot, se utilizan cuatro encoders ópticos de 1000 pulsos por revolución, los cuales están conectados a placas PCB específicas que permiten la lectura confiable de sus señales mediante microcontroladores ESP32.

En cuanto al subsistema de alimentación, el robot está equipado con una batería LiPo de 12V y 10Ah, capaz de alimentar los diferentes componentes durante períodos prolongados de operación. Además, se dispone de una PCB reguladora que convierte el voltaje de entrada en salidas estables de 12V y 5V, con una capacidad máxima de corriente de hasta 15 amperios. Esta configuración asegura la alimentación simultánea de motores, sensores y placas de control sin caídas de tensión ni sobrecalentamientos.

## 6.2. Integración Software

Desde el punto de vista del software, la arquitectura de integración se basa completamente en ROS 2, utilizando una red inalámbrica con tecnología WiFi 6 (802.11ax) para garantizar una comunicación estable y de alta velocidad entre los distintos nodos distribuidos en el sistema.

Los nodos ROS 2 implementados permiten la ejecución de diversas funcionalidades críticas. Uno de los principales procesos es la fusión sensorial, donde los datos provenientes del LiDAR y la cámara RGB-D se combinan para generar una representación más completa y precisa del entorno. A partir de esta información, se utiliza OctoMap para construir mapas tridimensionales probabilísticos, fundamentales para la planificación de trayectorias seguras.

En paralelo, se ejecuta el nodo de control cinemático, que traduce las órdenes de movimiento en comandos específicos para las ruedas mecanum, considerando las restricciones y capacidades del sistema omnidireccional.

La interfaz de usuario se implementa en RViz, lo que permite al operador visualizar en tiempo real el estado del robot, los datos sensoriales, el mapa generado y los planes de trayectoria. Esta herramienta también sirve como consola de monitoreo y depuración durante las fases de desarrollo y pruebas.

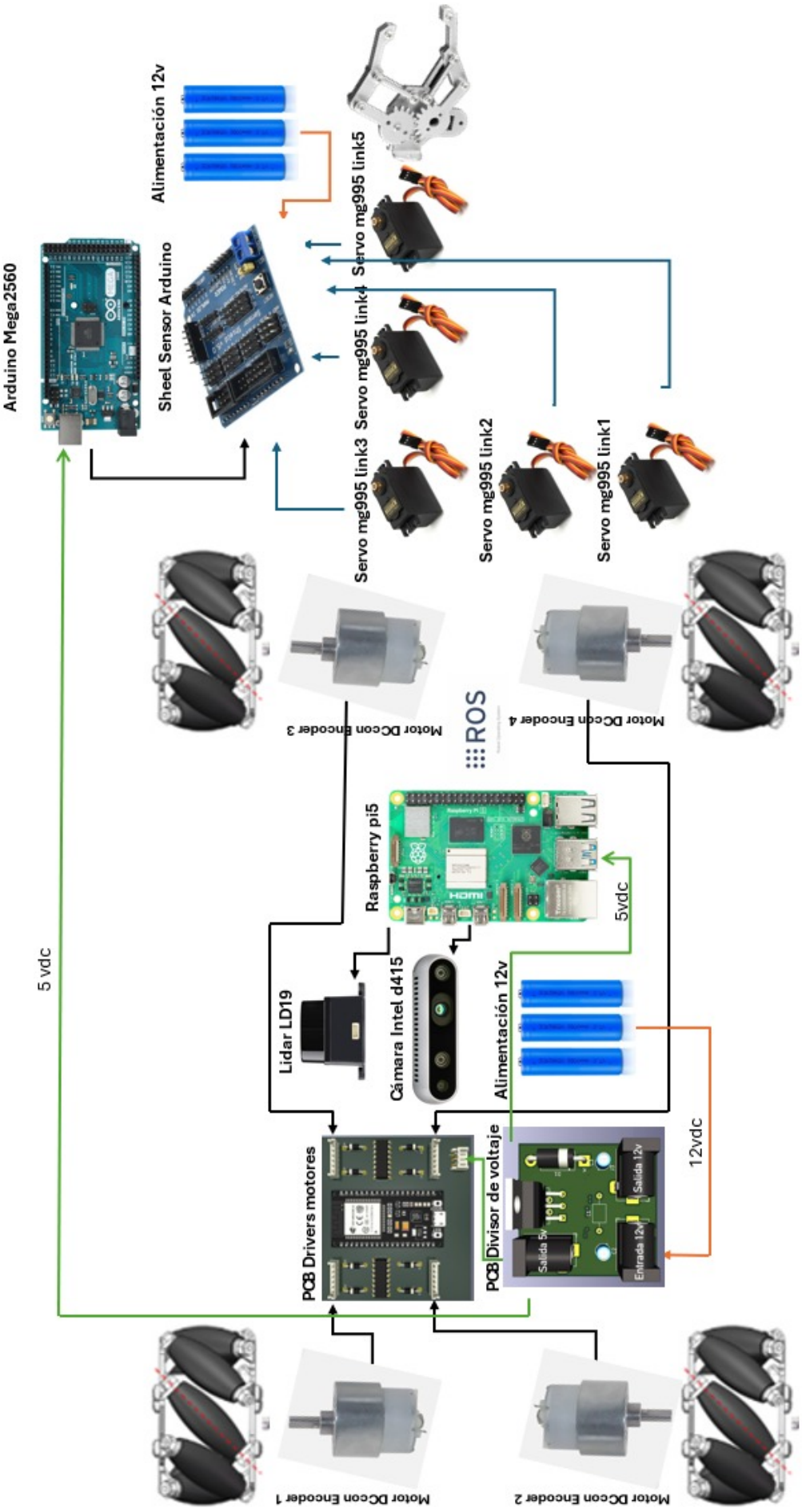




Figura 12: Diagrama de bloques del sistema de control

### 6.2.1. Tópicos Definidos para el Robot Omnidireccional

A continuación, se listan los tópicos utilizados para la comunicación entre los nodos del sistema en ROS 2. Estos tópicos permiten el control, monitoreo y navegación del robot omnidireccional, así como la operación del brazo manipulador y la adquisición de datos de sensores.

- **Encoders de Ruedas (Velocidad Angular y Distancia):** Los encoders son esenciales para la medición precisa del movimiento del robot. Estos sensores ópticos permiten calcular tanto la velocidad angular como la distancia recorrida por cada rueda, proporcionando datos críticos para la odometría del robot. Cada uno de los cuatro encoders se ubica en una rueda, proporcionando información en tiempo real sobre el movimiento del robot en sus ejes respectivos:

  - `/robot/omni/sensor/encoder_fleft/vel_angular`: Velocidad angular de la rueda frontal izquierda.
  - `/robot/omni/sensor/encoder_fright/vel_angular`: Velocidad angular de la rueda frontal derecha.
  - `/robot/omni/sensor/encoder_bleft/vel_angular`: Velocidad angular de la rueda trasera izquierda.
  - `/robot/omni/sensor/encoder_bright/vel_angular`: Velocidad angular de la rueda trasera derecha.
  - `/robot/omni/sensor/encoder_fleft/distance`: Distancia recorrida por la rueda frontal izquierda.
  - `/robot/omni/sensor/encoder_fright/distance`: Distancia recorrida por la rueda frontal derecha.
  - `/robot/omni/sensor/encoder_bleft/distance`: Distancia recorrida por la rueda trasera izquierda.
  - `/robot/omni/sensor/encoder_bright/distance`: Distancia recorrida por la rueda trasera derecha.

  Estos datos se utilizan para calcular la posición y orientación del robot en su entorno, lo cual es esencial para la navegación y control preciso, especialmente en un sistema omnidireccional donde las ruedas pueden moverse en cualquier dirección.

- **Sensores de Percepción:** Los sensores de percepción son fundamentales para que el robot pueda interpretar su entorno, evitando obstáculos y realizando tareas de mapeo y localización:

  - `/robot/omni/sensor/lidar/scan`: Datos del escaneo LiDAR, que ofrecen información sobre la distancia de los objetos cercanos en un rango de 360 grados. Estos datos son cruciales para generar un mapa del entorno y para detectar obstáculos de manera precisa.
  - `/robot/omni/sensor/camera/imag_depth`: Imágenes de profundidad de la cámara RGB-D, que proporcionan información sobre las distancias relativas a los objetos en el campo visual del robot. Son especialmente útiles para tareas como la manipulación de objetos o la navegación en espacios complejos.

- `/robot/omni/sensor/camera/imag_cloudpoints`: Nube de puntos generada a partir de las imágenes de profundidad de la cámara, utilizada para reconstruir el entorno 3D y facilitar la toma de decisiones en tiempo real.

Estos sensores permiten al robot "ver"su entorno de manera tridimensional, lo que es esencial para una navegación autónoma efectiva, así como para realizar tareas complejas como la manipulación de objetos.

- **Manipulador de 5 Grados de Libertad:** El manipulador robótico de 5 grados de libertad (DoF) permite al robot realizar movimientos precisos y complejos para interactuar con su entorno. Los datos proporcionados por los sensores de las juntas son fundamentales para controlar la cinemática inversa y la planificación de trayectorias del manipulador:

  - `/robot/omni/manipulator/junta1/deg`: Ángulo de la primera junta del manipulador.
  - `/robot/omni/manipulator/junta2/deg`: Ángulo de la segunda junta del manipulador.
  - `/robot/omni/manipulator/junta3/deg`: Ángulo de la tercera junta del manipulador.
  - `/robot/omni/manipulator/junta4/deg`: Ángulo de la cuarta junta del manipulador.
  - `/robot/omni/manipulator/junta5/deg`: Ángulo de la quinta junta del manipulador.

  Estos datos son esenciales para la ejecución precisa de tareas como el levantamiento, movimiento y colocación de objetos, así como para la interacción del robot con su entorno.

- **Navegación y Seguridad:** La navegación y seguridad del robot se gestionan mediante sensores y algoritmos de planificación de rutas que permiten al robot moverse de manera autónoma mientras evita obstáculos y mantiene la seguridad en su entorno:

  - `/robot/navigation/path`: El camino de navegación calculado por el sistema de planificación de rutas. Este dato representa la trayectoria a seguir por el robot para alcanzar un objetivo determinado, evitando obstáculos en el proceso.
  - `/robot/navigation/alert_obstacles`: Alerta generada por el sistema de navegación cuando se detecta un obstáculo en la trayectoria del robot. Esta alerta puede activar mecanismos de evasión o reajuste de la ruta.

  Estos parámetros de navegación aseguran que el robot pueda operar de manera autónoma y segura, ajustando su ruta en tiempo real y reaccionando ante situaciones imprevistas.

| Nodo de Ros2 | Tópicos Publicados | Tópicos suscritos |
|---|---|---|
| rplidar_node | /robot/omni/sensor/lidar/scan | — |
| realsense2_camera_node | /robot/omni/sensor/camera/image_depth<br>/robot/omni/sensor/camera/image_cloudpoints | — |
| micro_ros_agent (ESP32) | /robot/omni/sensor/encoder_fleft/vel_angular<br>/robot/omni/sensor/encoder_fright/vel_angular<br>/robot/omni/sensor/encoder_bleft/vel_angular<br>/robot/omni/sensor/encoder_bright/vel_angular<br>/robot/omni/sensor/encoder_*/distance | — |
| odometry_node (Raspberry Pi) | /odom<br>/tf | Todos los tópicos vel_angular de los encoders |
| slam_toolbox | /map<br>/tf, /tf_static | /odom<br>/robot/omni/sensor/lidar/scan |
| navigation_node | /robot/navigation/path<br>/robot/navigation/alert_obstacles | /odom<br>/map<br>/robot/omni/sensor/lidar/scan |
| manipulator_controller_node | /robot/omni/manipulator/junta1/deg<br>... hasta junta5/deg | (pendiente de definir nodo de control o interfaz) |

Cuadro 2: Resumen de nodos y sus tópicos correspondientes

# 7. Implementación del gemelo digital

Para nuestro proyecto, la base principal es implementar un sistema ciberfísico de un robot manipulador móvil omnidireccional. Para esto, utilizaremos una serie de herramientas para realizar diferentes acciones y simulaciones de nuestro sistema, los cuales son rviz, gazebo y moveit2.

## 7.1. Rviz2

Ros visualization es una herramienta gráfica de ROS, se utiliza para visualizar los datos del robot, es realmente útil para nuestro proyecto ya que podemos tener la visualización del robot y sus sensores para crear un octomap de un área determinada.

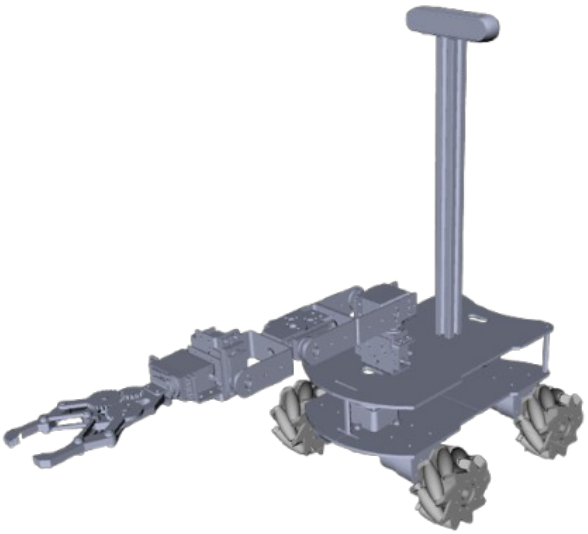

Figura 13: Visualización del robot en Rviz

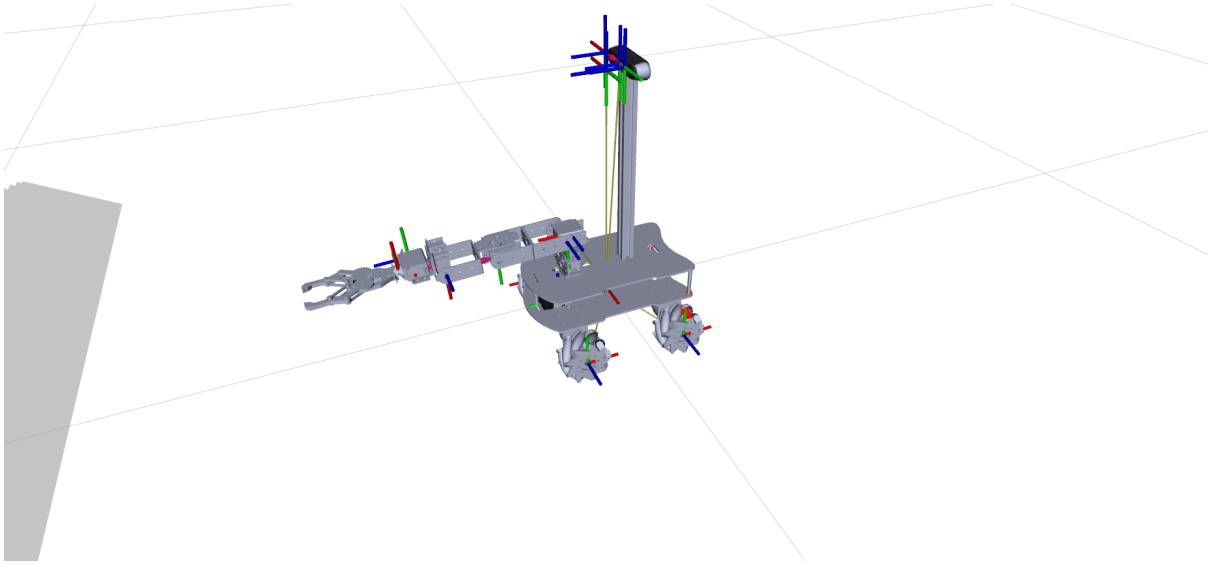

Figura 14: Marcos coordenados del robot

## 7.2. Gazebo

Gazebo es un simulador 3D de robótica que permite probar y desarrollar nuestro robot en un entorno virtual y al mismo tiempo llevarlo en el mundo real, aquí realizaremos las simulaciones físicas de nuestro robot en el entorno en el que se encuentre así como la evasión de obstáculos que se encuentra mediante los sensores anteriormente mencionados, cámara intelrealsense d415 y el sensor lidar, para implementar un control de posición en el mapa ya generado.

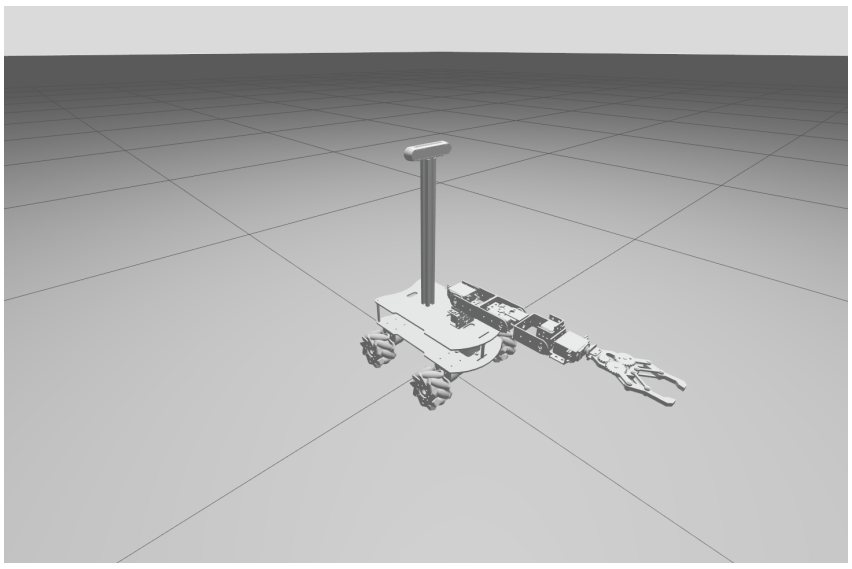

Figura 15: Modelo del robot en simulador gazebo

## 7.3. Moveit2

Moveit2 es una herramienta para la planificación de movimiento de robots, podemos realizar varias cosas con el como calcular las cinemáticas inversas y directas, planificar trayectorias y control de movimiento, en nuestro caso lo estamos utilizando para el calculo de cinemáticas directa e inversa.

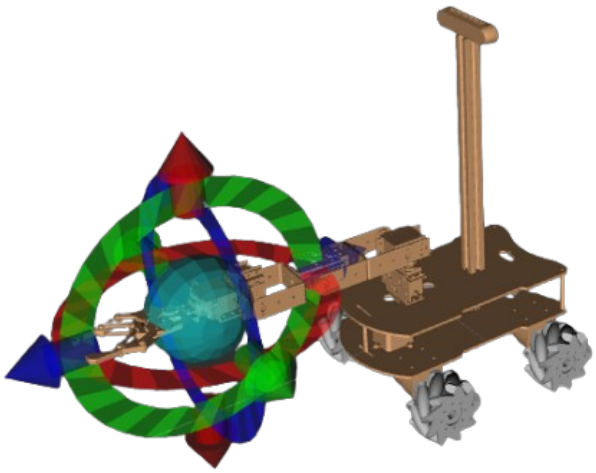

Figura 16: Modelo del robot en el framework Moveit2

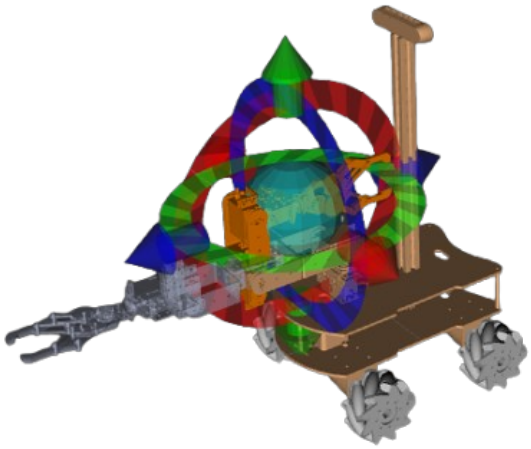

Figura 17: Pose de home del robot en Moveit2

# 8. Percepción del robot

# 9. Sensores de Percepción: Intel RealSense D415 y LiDAR LD19

En este proyecto, se integraron dos sensores principales para la percepción del entorno: la cámara Intel RealSense D415 y el sensor LiDAR LD19. Ambos dispositivos se utilizaron en simulaciones realizadas en Gazebo y RViz2, aprovechando herramientas como `slam_toolbox`, `depthcloud`, y un puente de comunicación entre `gz_sim` y `rviz2` en ROS 2.

## 9.1. Cámara Intel RealSense D415

La cámara Intel RealSense D415 es un sensor RGB-D que proporciona imágenes en color y mapas de profundidad. En este proyecto, se utilizó para generar nubes de puntos (*pointclouds*) y para la percepción tridimensional del entorno. Las principales características y aplicaciones de la cámara son:

- **Resolución y profundidad:** La cámara genera mapas de profundidad con una resolución de hasta 1280x720 píxeles y una frecuencia de 30 FPS, lo que permite una percepción precisa del entorno.
- **Generación de nubes de puntos:** Utilizando el paquete `depthcloud`, se generaron nubes de puntos en tiempo real, las cuales fueron visualizadas en RViz2 para analizar la geometría del entorno.
- **Integración con ROS 2:** La cámara se configuró mediante el paquete `realsense2_camera`, que publica los datos de profundidad en el tópico `/robot/omni/sensor/camera/image_cloudpoin`
- **Aplicaciones:** Los datos de la cámara se utilizaron para la detección de obstáculos, generación de mapas tridimensionales y como entrada para algoritmos de navegación y manipulación.

## 9.2. LiDAR LD19

El sensor LiDAR LD19 es un dispositivo de escaneo láser que proporciona datos de distancia en un rango de 360 grados. Este sensor fue utilizado para la implementación de SLAM (*Simultaneous Localization and Mapping*) y para la navegación autónoma del robot. Sus principales características y aplicaciones son:

- **Rango y frecuencia:** El LiDAR LD19 tiene un rango de detección de hasta 12 metros y una frecuencia de escaneo de 10 Hz, lo que permite una percepción rápida y precisa del entorno.
- **SLAM Toolbox:** Los datos del LiDAR se procesaron mediante el paquete `slam_toolbox`, que generó mapas bidimensionales del entorno y proporcionó estimaciones de la posición del robot en tiempo real.

- **Publicación de datos:** Los datos del LiDAR se publicaron en el tópico `/robot/omni/sensor/lid` y se integraron con la odometría del robot para mejorar la precisión de la localización.
- **Visualización:** Los datos del LiDAR se visualizaron en RViz2, permitiendo observar el mapa generado y la posición del robot en el entorno.

## 9.3. Integración y Simulación en Gazebo y RViz2

Para simular el comportamiento de los sensores en un entorno virtual, se utilizó Gazebo como simulador físico y RViz2 como herramienta de visualización. La integración entre ambos se logró mediante un puente de comunicación en ROS 2 (`ros_gz_bridge`). Las principales funcionalidades implementadas fueron:

- **Teleoperación:** Se controló el robot de manera remota utilizando nodos de teleoperación, lo que permitió probar la navegación y la percepción en tiempo real.
- **Odometría:** Los datos de los encoders y el LiDAR se combinaron para calcular la odometría del robot, mejorando la precisión de la localización.
- **Fusión de datos:** Los datos de la cámara y el LiDAR se integraron para generar mapas tridimensionales utilizando OctoMap, lo que permitió una representación más completa del entorno.
- **Visualización en RViz2:** Se visualizaron las nubes de puntos generadas por la cámara, los mapas bidimensionales del SLAM Toolbox y los datos de odometría, proporcionando una vista completa del estado del robot y su entorno.

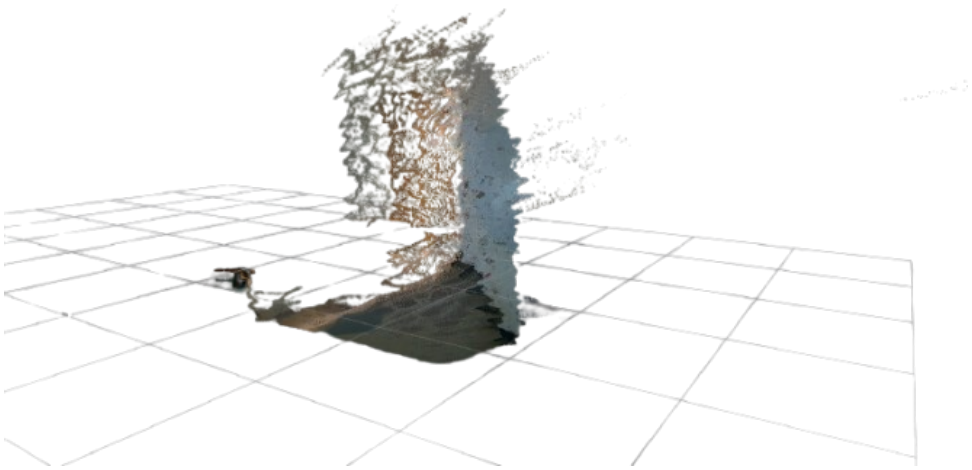

Figura 18: Visualización de la nube de puntos generada por la cámara Intel RealSense D415 en RViz2.

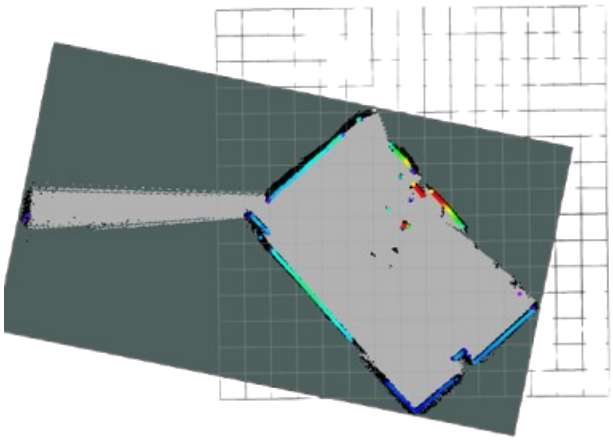

Figura 19: Mapa generado por el LiDAR LD19 utilizando SLAM Toolbox.

# 10. Cinemática Directa del Manipulador

Para modelar la cinemática directa del manipulador robótico de cinco grados de libertad (DoF), se empleó el método de Denavit-Hartenberg (DH), el cual permite representar matemáticamente la relación espacial entre los eslabones consecutivos del robot mediante matrices de transformación homogénea. Cada una de estas matrices define una transformación en el espacio tridimensional basada en cuatro parámetros: $a_i$ (longitud del eslabón), $\alpha_i$ (ángulo de torsión), $d_i$ (desplazamiento a lo largo del eje z), y $\theta_i$ (ángulo de rotación).

La siguiente tabla resume los parámetros DH utilizados para describir la geometría del manipulador:

| **Link** | $a_i$ (mm) | $\alpha_i$ (°) | $d_i$ (mm) | $\theta_i$ (°) |
|---|---|---|---|---|
| 1 | 0 | 90 | 13 | $\theta_1$ |
| 2 | 105 | 0 | 0 | $\theta_2$ |
| 3 | 100 | 0 | 0 | $-\theta_3$ |
| 4 | 60 | 90 | 0 | $\theta_4$ |
| 5 | 0 | 0 | 35 | $-\theta_5$ |

Cuadro 3: Parámetros DH del manipulador de 5 DoF

La cinemática directa consiste en calcular la posición y orientación del efector final a partir de los ángulos articulares. Para ello, se construyen sucesivamente las matrices de transformación homogénea $T_i^{i-1}$ para cada eslabón, de acuerdo con la convención DH. La matriz de transformación general desde el sistema base hasta el efector final se obtiene

como el producto de las matrices individuales:

$$T_0^5 = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5$$

Cada una de estas matrices $T_i^{i-1}$ se calcula mediante la siguiente ecuación estándar:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El resultado de esta multiplicación proporciona la posición y orientación del efector final en el sistema de coordenadas base. Adicionalmente, la visualización tridimensional del manipulador se genera mediante el cálculo iterativo de las posiciones relativas de cada eslabón, utilizando las transformaciones acumuladas. Esta representación facilita el análisis espacial del robot durante la ejecución de tareas de manipulación.

# 11. Cinemática Directa Del Robot Omnidireccional

La cinemática directa de un robot móvil omnidireccional nos permite obtener las velocidades lineales y angulares del cuerpo del robot a partir de las velocidades angulares de sus ruedas. En este caso, consideramos un robot con tres ruedas omnidireccionales ubicadas simétricamente a 120° entre sí.

## Parámetros del sistema

- $r$: radio de las ruedas.
- $L$: distancia desde el centro del robot hasta cada rueda.
- $\omega_i$: velocidad angular de la rueda $i$ (con $i = 1, 2, 3$).
- $(v_x, v_y, \omega_z)$: velocidades lineales y angular del robot en el marco de referencia global.

La relación entre las velocidades de las ruedas y las velocidades del robot está dada por la ecuación:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{r}{3} \begin{bmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

Donde los ángulos de las ruedas respecto al eje $x$ son:

$$\theta_1 = 0^\circ$$
$$\theta_2 = 120^\circ$$
$$\theta_3 = 240^\circ$$

Utilizando identidades trigonométricas:

$$\sin(0^\circ) = 0 \qquad \cos(0^\circ) = 1$$
$$\sin(120^\circ) = \frac{\sqrt{3}}{2} \qquad \cos(120^\circ) = -\frac{1}{2}$$
$$\sin(240^\circ) = -\frac{\sqrt{3}}{2} \qquad \cos(240^\circ) = -\frac{1}{2}$$

Sustituyendo en la matriz, tenemos:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{r}{3} \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

Resultado: La ecuación final de la cinemática directa queda expresada como:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{r}{3} \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

Ejemplo Numérico: Supongamos los siguientes valores:

- $r = 0{,}05$ m
- $L = 0{,}15$ m
- $\omega_1 = 10$ rad/s, $\omega_2 = 5$ rad/s, $\omega_3 = -5$ rad/s

Sustituimos:

$$\frac{r}{3} = \frac{0{,}05}{3} \approx 0{,}0167$$

Multiplicamos:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \approx 0{,}0167 \cdot \begin{bmatrix} 0 & -0{,}866 & 0{,}866 \\ 1 & -0{,}5 & -0{,}5 \\ \frac{1}{0{,}15} & \frac{1}{0{,}15} & \frac{1}{0{,}15} \end{bmatrix} \begin{bmatrix} 10 \\ 5 \\ -5 \end{bmatrix}$$

# Referencias


Campos, C., Elvira, R., Gómez Rodríguez, J. J., Montiel, J. M. M., & Tardós, J. D. (2021). ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*.

Monroy, D. F. (2023). *Desarrollo de un prototipo de un robot móvil autónomo de bajo costo con Raspberry* [Tesis de maestría, Universidad ECCI].

Muir, P., & Neumann, C. (1990). Kinematic Modeling of Wheeled Mobile Robots. *Journal of Robotic Systems*.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, *31*(5), 1147-1163. https://doi.org/10.1109/TRO.2015.2463671

Saenz, J. A., Bugarin, E., & Santibáñez, V. (2015). Modelado Cinemático y Dinámico de un Robot Móvil Omnidireccional de 4 Ruedas. *Congreso Internacional de Robótica*.

Serrano, T., Rincon, N. L., Nava, A. M., & Flores, Y. S. (2021). Artificial vision system for object classification using Raspberry Pi. *Revista Tecnologías de la Información*.

Williams, R. L. (2008). *Robot Dynamics and Control*. Ohio University.

Zhang, J., & Singh, S. (2014). LOAM: Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems*.


## 11.1. Anexos

### 11.1.1. Código cinemática directa omnidireccional

```python
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import JointState
from .direct_kinematics import MecanumDirectKinematics

class KinematicsNode(Node):
    def __init__(self):
        super().__init__('kinematics_node')
        self.dk = MecanumDirectKinematics()
        self.subscription = self.create_subscription(
            JointState,
            '/joint_states',
            self.listener_callback,
            10)
        self.subscription

    def listener_callback(self, msg):
        joint_names = ['fl_wheel_joint', 'fr_wheel_joint',
                       'rl_wheel_joint', 'rr_wheel_joint']
        wheel_velocities = [0.0, 0.0, 0.0, 0.0]

        for i, joint in enumerate(joint_names):
            try:
                index = msg.name.index(joint)
                if index < len(msg.velocity):
```

```
                    wheel_velocities[i] = msg.velocity[index]
                else:
                    self.get_logger().warn(
                        f"Joint '{joint}' no tiene información de velocidad.")
                    return
            except ValueError:
                self.get_logger().warn(
                    f"Joint '{joint}' no encontrado en el mensaje de joint_states.")
                return

        vel = self.dk.compute(wheel_velocities)
        self.get_logger().info(f'Velocidad del robot [vx, vy, ]: {vel}')

def main(args=None):
    rclpy.init(args=args)
    node = KinematicsNode()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()
```

## 11.2. Código cinemática inversa omnidireccional

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from sensor_msgs.msg import JointState
from .inverse_kinematics import MecanumInverseKinematics

class InverseKinematicsNode(Node):
    def __init__(self):
        super().__init__('inverse_kinematics_node')
        self.ik = MecanumInverseKinematics()

        self.subscription = self.create_subscription(
            Twist,
            '/cmd_vel',
            self.cmd_vel_callback,
            10)

        self.publisher = self.create_publisher(
            JointState,
            '/wheel_commands',
            10)

        self.joint_names = ['fl_wheel_joint', 'fr_wheel_joint',
                            'rl_wheel_joint', 'rr_wheel_joint']

    def cmd_vel_callback(self, msg):
        vx = msg.linear.x
        vy = msg.linear.y
        omega = msg.angular.z
```

```
30
31          wheel_velocities = self.ik.compute(vx, vy, omega)
32
33          joint_msg = JointState()
34          joint_msg.name = self.joint_names
35          joint_msg.velocity = wheel_velocities.tolist()
36          self.publisher.publish(joint_msg)
37
38          self.get_logger().info(f'Entradas vx={vx:.2f}, vy={vy:.2f}, ={omega:.2f} -> Ruedas:
39
40  def main(args=None):
41      rclpy.init(args=args)
42      node = InverseKinematicsNode()
43      rclpy.spin(node)
44      node.destroy_node()
45      rclpy.shutdown()
```

### 11.2.1. Código cinemática brazo

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import Int32MultiArray
from pymata4 import pymata4
import time

class RobotManipulador(Node):
    def __init__(self):
        super().__init__('robot_manipulador')

        self.board = pymata4.Pymata4()
        self.servo_pins = [3, 5, 6, 9, 10, 11]
        self.current_angles = [90] * 6

        for pin in self.servo_pins:
            self.board.set_pin_mode_servo(pin)

        self.subscription = self.create_subscription(
            Int32MultiArray,
            '/servo_commands',
            self.int_callback,
            10
        )

        self.publisher = self.create_publisher(
            Int32MultiArray,
            '/servo_positions',
            10
        )

        self.timer = self.create_timer(1.0, self.publish_positions)
        self.get_logger().info('Nodo del manipulador iniciado correctamente')

    def int_callback(self, msg: Int32MultiArray):
        if len(msg.data) != 6:
            self.get_logger().warn(f'Se esperaban 6 valores, se recibieron: {len(msg.data)}
            return

        for i in range(6):
            angle = max(0, min(180, msg.data[i]))
            self.board.servo_write(self.servo_pins[i], angle)
            self.current_angles[i] = angle
            time.sleep(0.05)

        self.get_logger().info(f'Servos movidos a: {self.current_angles}')

    def publish_positions(self):
        msg = Int32MultiArray()
        msg.data = self.current_angles
        self.publisher.publish(msg)
```

```
51
52  def main(args=None):
53      rclpy.init(args=args)
54      node = RobotManipulador()
55      try:
56          rclpy.spin(node)
57      except KeyboardInterrupt:
58          pass
59      finally:
60          node.board.shutdown()
61          node.destroy_node()
62          rclpy.shutdown()
63
64  if __name__ == '__main__':
65      main()
```