
UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ.

EXTENSIÓN MULTIDISCIPLINARIA DE CIUDAD
UNIVERSITARIA

DISEÑO MECATRÓNICO

IMPLEMENTACIÓN DE SLAM CON ROS 2 EN UN ROBOT
MÓVIL DIFERENCIAL UTILIZANDO SENSOR LiDAR

Autores:

Muñoz Silva Brenda Lizeth 201820
Rincón Loaeza Niza Yamilet 201203
Baca Salas Viri Danitzi 201284
Duran Chacón Jazmín Andrea 192738
Ramos Rivera Kevin 180070

Docente:

Dr.García Luna Francesco José

Fecha de entrega: 21 DE MARZO DE 2025

Ciudad Juárez, Chih

Índice

1. INTRODUCCIÓN	3
2. OBJETIVO	4
2.1. Objetivo Principal	4
2.2. Objetivos Particulares	4
3. FUNDAMENTOS TEÓRICOS	5
3.1. URDF	5
3.2. Gazebo	6
3.2.1. Gazebo Ignition	6
3.3. RViz	7
3.4. ROS2	7
4. METODOLOGÍA	8
4.1. Materiales y herramientas utilizadas	8
4.2. PROCEDIMIENTO DEL DESARROLLO	9
4.2.1. Modelo de Movimiento del Robot	9
4.2.2. Velocidad Angular de las Ruedas	9
4.2.3. Relación entre Velocidad Angular y Velocidad Lineal	9
4.2.4. Velocidad Lineal y Angular del Robot	9
4.2.5. Cálculo de la Posición del Robot	10
4.2.6. Control de los Motores	10
4.3. DISEÑO MECÁNICO Y ESTRUCTURAL	11
4.4. DISEÑO ELECTRÓNICO	14
4.5. IMPLEMENTACIÓN Y ENSAMBLAJE	15
5. PRUEBAS Y RESULTADOS	16
5.1. Pruebas	16
5.1.1. URDF	16
5.1.2. RViz2	17
5.1.3. Gazebo	18
5.1.4. LiDAR	19
5.1.5. Micro-ROS	20
5.1.6. Nodos	21
5.1.7. Tópicos	21
5.2. Resultados	22
6. BIBLIOGRAFÍA	23
A. ANEXOS	24
A.1. Código URDF	29

1. INTRODUCCIÓN

En este proyecto se busca implementar ROS para un robot móvil diferencial con SLAM.

- ROS (Robot Operating System) es un marco de código abierto que facilita el desarrollo de aplicaciones robóticas, proporcionando herramientas y bibliotecas para la creación de software para robots, incluyendo abstracción de hardware, comunicación entre procesos y más.
Facilita la creación de software para robots, permitiendo la comunicación entre diferentes componentes (nodos) en un sistema robótico, abstracción de hardware, control de dispositivos y más.

Los nodos (procesos) en ROS se comunican entre sí a través de "tópicos" (temas) que son canales que transportan mensajes.

Un robot móvil diferencial es un robot que se mueve gracias a dos ruedas motrices independientes, una a cada lado del cuerpo. Este tipo de robot se puede mover hacia adelante y hacia atrás, pero no hacia los lados.

SLAM (localización y mapeo simultáneos) es un método utilizado en vehículos autónomos que permite crear un mapa y localizar el vehículo en el mapa al mismo tiempo. Los algoritmos de SLAM permiten que el vehículo cree mapas de entornos desconocidos.

2. OBJETIVO

2.1. Objetivo Principal

Implementar SLAM con ROS 2 en un robot móvil diferencial utilizando un sensor LiDAR.

2.2. Objetivos Particulares

- Integrar sensores (LiDAR y encoders) en el sistema del robot.
- Configurar nodos y tópicos en ROS 2 para el control del movimiento y la adquisición de datos.
- Implementar SLAM para generación de mapas y localización simultánea.
- Validar el sistema en entornos virtuales y reales mediante simulación en Gazebo y visualización en RViz.

3. FUNDAMENTOS TEÓRICOS

3.1. URDF

Unified Robot Description Format por sus siglas en inglés, fue implementado en 2009 por parte de los desarrolladores del Sistema Operativo de Robot (ROS), como un formato para realizar la descripción de la cinemática, la dinámica y las geometrías de los robots, siendo independiente de los programas de software. El formato URDF es una especificación XML que permite definir las características de un robot, incluyendo su estructura cinemática y propiedades dinámicas, la representación visual utilizada en simulaciones o modelos gráficos, y el modelo de colisión, que sirve para detectar interacciones como su entorno.

La descripción del robot consiste en un conjunto de elementos de enlace y un conjunto de elementos de articulación que conectan los enlaces entre sí. URDF cuenta con diversas restricciones como lo es el no poder describir robots paralelos, por ende no se puede manejar enlaces elásticos transformables, por ende solo un cuerpo rígido. La estructura con la que cuenta tiene un patrón como:

```
<robot name="robot_name">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

URDF cuenta con una herramienta Python y scripts, las cuales permiten a los usuarios analizar de una sencilla manera nuevos archivos URDF. El archivo puede incluir referencias de otros archivos que contienen geometrías 3D de los componentes del robot. En URDF se puede desarrollar todos los detalles relevantes del modelado de un robot en un solo archivo. Tras el uso de XML se permite a las herramientas implementarse de manera sencilla la compatibilidad con el formato, empleando analizadores maduros incluidos en gran parte de los lenguajes de programación. Pese a que se implementó inicialmente para el sistema ROS, su carácter independiente ha sido posible su adopción por distintas herramientas, tanto dentro como fuera del ecosistema ROS. Remiendo el punto de como es su estructura, los requisitos mínimos para crear un archivo de URDF son el nombre del robot y un enlace.

3.2. Gazebo

Gazebo es un entorno de simulación de robots de código abierto. El proyecto está gestionado por Open Robotics, el mismo grupo que gestiona ROS. Con Gazebo, podemos crear un "mundo" virtual y cargar en él versiones simuladas de nuestros robots. Los sensores simulados pueden detectar el entorno y publicar los datos en los mismos temas de ROS que los sensores reales, lo que facilita la prueba de algoritmos. Posteriormente, se pueden aplicar fuerzas a los actuadores simulados del robot, teniendo en cuenta factores como la fricción.

Se caracteriza por su capacidad para modelar tanto la cinemática como la dinámica de sistemas, incorporando parámetros físicos que permiten generar entornos realistas. Este software está diseñado para el desarrollo de plataformas robóticas, ya que facilita la ejecución de paquetes y nodos relacionados con sensores, actuadores y el diseño de robots, además de otras funciones clave para la gestión y creación de sistemas robóticos.[\[RMD20\]](#)

El uso de esta herramienta nos permitirá obtener una simulación precisa del comportamiento del dron en un entorno tridimensional, asegurando que su desempeño en el mundo virtual de Gazebo sea lo más similar posible al del mundo real.

3.2.1. Gazebo Ignition

Gazebo Ignition, ahora conocido como Ignition Gazebo (y recientemente parte de la suite Gazebo Sim), es la evolución moderna del clásico simulador Gazebo. También es de código abierto y ha sido desarrollado por Open Robotics con el objetivo de mejorar y extender las capacidades del simulador original.

Lo que hace especial a Ignition es que fue rediseñado desde cero para ser más modular, más eficiente y más fácil de mantener a largo plazo. Incorpora mejoras significativas en el renderizado gráfico, los modelos de sensores, la simulación física y la interacción con el entorno. Gracias a sus nuevos motores de física y gráficos, los robots y los mundos virtuales se ven y se comportan de una forma mucho más realista.

Ignition también permite simular escenarios más complejos, como robots trabajando en entornos industriales, colaborativos o incluso situaciones con múltiples robots interactuando entre sí. A través de sus plugins y herramientas gráficas, puedes construir mundos, colocar modelos, editar parámetros físicos y controlar el comportamiento del robot, todo desde una interfaz más moderna y amigable.

Además, Ignition mantiene compatibilidad con ROS 2, lo que lo hace una opción ideal para quienes están migrando sus proyectos hacia esta versión más reciente del sistema operativo robótico.

3.3. RViz

RViz es una herramienta de simulación y visualización 3D para robots, el ambiente en el que estos se desempeñan y la información de sensores que estos generan dentro de este. Es altamente configurable y posee distintos tipos de plugins y formatos de visualización. Algunas de las aplicaciones comunes de RViz incluyen la visualización de datos de sensores como cámaras, LiDAR (Light Detection and Ranging), IMU (Inertial Measurement Unit) y otros sensores utilizados en robótica. También se utiliza para mostrar información sobre la posición y orientación de los robots, planificación de trayectorias, mapeo en 2D y 3D, y la visualización de datos de simulaciones robóticas.

Además de su funcionalidad técnica, una de las grandes ventajas de RViz es que permite entender de forma visual lo que está ocurriendo en tiempo real dentro de un sistema robótico. Esto resulta especialmente útil cuando se trabaja con múltiples sensores o cuando se desarrollan algoritmos de navegación, percepción o manipulación. Ver cómo el robot interpreta su entorno facilita muchísimo la tarea de depuración (debugging), ya que permite identificar problemas de calibración, errores en los datos de sensores o fallos en la lógica del software.

Otra cosa que hace muy valioso a RViz es su capacidad para integrarse con ROS (Robot Operating System), lo que le permite recibir y mostrar información de distintos tópicos publicados por los nodos del sistema. Por ejemplo, si un nodo está publicando la nube de puntos de un escáner LiDAR, RViz puede mostrarla en tiempo real, dándote una idea muy clara de cómo el robot "ve" su entorno. También se pueden visualizar frames de referencia, trayectorias planificadas, obstáculos detectados, entre otros elementos, lo que hace que RViz sea una herramienta clave en el desarrollo e investigación en robótica.

Además, RViz no solo sirve para observar lo que el robot está haciendo; también permite interactuar con él. Por ejemplo, se pueden definir metas de navegación directamente desde la interfaz gráfica o mover los brazos de un robot manipulador en una simulación para verificar cómo reaccionaría en un entorno real.

3.4. ROS2

ROS2 (Robot Operating System 2) es una plataforma de software compuesta por bibliotecas y herramientas diseñadas para el desarrollo de aplicaciones robóticas. Abarca desde controladores y algoritmos avanzados hasta utilidades de programación, brindando así los recursos de código abierto necesarios para la creación de sistemas robóticos. Desde su surgimiento en 2007, la comunidad ROS ha experimentado un crecimiento considerable. ROS2 surge como una evolución de ROS1, buscando conservar sus fortalezas y mejorar aquellos aspectos que presentaban limitaciones.

RViz2 es la versión adaptada de RViz para ROS2. Se trata de una herramienta de visualización tridimensional que permite observar robots, sensores, mapas y otros elementos mediante una interfaz gráfica. Esta herramienta se instala automáticamente con ROS2 y requiere un entorno de escritorio en Linux, siendo Ubuntu la opción más recomendada, aunque también es compatible con distribuciones como Fedora o Arch Linux.

En ROS2, el desarrollo de software se organiza en paquetes, los cuales agrupan código fuente, configuraciones y demás recursos necesarios para su funcionamiento. La estructura básica de un paquete se puede generar con el comando `ros2 pkg create`. Estos paquetes pueden incluir nodos, bibliotecas, archivos de configuración y otros elementos necesarios para el sistema.

Existe una gran variedad de paquetes disponibles para distintas funciones. Uno de ellos es `ros_gz_sim`, que permite integrar ROS2 con el simulador Gazebo. Este paquete incluye scripts y archivos de lanzamiento que simplifican la conexión entre ambos entornos, facilitando así la simulación de robots y escenarios virtuales.

4. METODOLOGÍA

4.1. Materiales y herramientas utilizadas

Materiales:

Para la fabricación y ensamblaje del robot móvil se utilizaron los siguientes materiales:

PETG/PLA: Material utilizado en la impresión 3D de componentes como las ruedas locas, ruedas motrices, Tanque, soporte de sensores y estructura del chasis.

Motores DC con encoder: Proveen la tracción necesaria para el movimiento diferencial del robot.

Puente H (L298N): Control de dirección y velocidad de los motores DC.

Ruedas oruga: Diseñadas para mejorar la tracción y estabilidad en diferentes superficies.

Rodamientos y tornillería: Para asegurar el correcto movimiento y ensamblaje de las piezas móviles.

Batería LiPo / Power Bank: Fuente de alimentación para el sistema.

ESP32/ Raspberry Pi: Microcontroladores/microcomputadoras para el control y comunicación.

ESP32: Controlador principal para el manejo de motores mediante PWM.

Raspberry Pi: Procesamiento de alto nivel y manejo de ROS 2.

Sensores

LiDAR: Sensor principal para detección de obstáculos y mapeo.

Herramientas

Impresora 3D: Para la fabricación de piezas personalizadas en PETG / PLA.

Software CAD (SolidWorks): Diseño de las piezas mecánicas del robot.

ROS (Robot Operating System): Framework principal para el desarrollo e integración del robot.

Gazebo: Simulador 3D utilizado para probar el comportamiento del robot en entornos virtuales.

Rviz: Visualizador de ROS para la interpretación de datos de sensores y mapeo.

Python: Lenguaje de programación usados para los nodos de ROS y scripts de control.

Arduino IDE / VSCode: Entornos de desarrollo para la programación de microcontroladores.

4.2. PROCEDIMIENTO DEL DESARROLLO

4.2.1. Modelo de Movimiento del Robot

El robot utiliza tracción diferencial, lo que significa que las velocidades de las ruedas izquierda y derecha controlan el movimiento y la orientación del robot. El movimiento del robot depende de la diferencia en la velocidad de las ruedas, lo que se puede modelar matemáticamente.

4.2.2. Velocidad Angular de las Ruedas

La velocidad angular ω de cada rueda se calcula en función de los pulsos generados por los encoders. La velocidad angular de la rueda derecha y la rueda izquierda se modelan como:

$$\begin{aligned}\omega_R &= \frac{\text{encoderPositionR}}{PPR_R} \times 2\pi \\ \omega_L &= \frac{\text{encoderPositionL}}{PPR_L} \times 2\pi\end{aligned}$$

Donde:

- ω_R y ω_L son las velocidades angulares de las ruedas derecha e izquierda, respectivamente, en radianes por segundo.
- PPR_R y PPR_L son los pulsos por revolución de las ruedas derecha e izquierda (configurados en 990 pulsos).

4.2.3. Relación entre Velocidad Angular y Velocidad Lineal

Cada rueda tiene un radio R y la velocidad lineal de la rueda se obtiene multiplicando la velocidad angular por el radio:

$$v_R = \omega_R \times R$$

$$v_L = \omega_L \times R$$

Donde:

- v_R y v_L son las velocidades lineales de las ruedas derecha e izquierda, respectivamente, en metros por segundo.

4.2.4. Velocidad Lineal y Angular del Robot

La velocidad lineal del robot v y la velocidad angular ω_{robot} se calculan utilizando las velocidades lineales de las ruedas. Las ecuaciones son las siguientes:

$$v = \frac{v_R + v_L}{2}$$

$$\omega_{\text{robot}} = \frac{v_R - v_L}{d}$$

Donde:

- v es la velocidad lineal del robot.
- ω_{robot} es la velocidad angular del robot.
- d es la distancia entre las dos ruedas del robot.

4.2.5. Cálculo de la Posición del Robot

El cálculo de la posición del robot en un espacio 2D se puede realizar utilizando un modelo de odometría. La posición (x, y) del robot y su orientación θ se actualizan utilizando las siguientes ecuaciones:

$$x_{\text{new}} = x_{\text{old}} + v \cdot \cos(\theta) \cdot \Delta t$$

$$y_{\text{new}} = y_{\text{old}} + v \cdot \sin(\theta) \cdot \Delta t$$

$$\theta_{\text{new}} = \theta_{\text{old}} + \omega_{\text{robot}} \cdot \Delta t$$

Donde:

- x_{new} y y_{new} son las nuevas coordenadas del robot en el plano 2D.
- θ_{new} es la nueva orientación del robot.
- Δt es el intervalo de tiempo entre las actualizaciones.

4.2.6. Control de los Motores

El control de los motores se realiza mediante señales PWM generadas por el joystick de la interfaz Blynk. Las señales PWM se mapean desde los valores del joystick (entre 0 y 100) a valores de 0 a 255 para controlar la velocidad de los motores.

La fórmula para generar la señal PWM es:

$$\text{PWM} = \text{map}(y, 0, 100, 0, 255)$$

Donde y es la entrada del joystick, y la señal PWM se utiliza para controlar la velocidad de los motores.

4.3. DISEÑO MECÁNICO Y ESTRUCTURAL

El diseño mecánico de nuestro proyecto fue realizado en el software de diseño Solidworks, el cual consta de 19 piezas, las cuales conforman el ensamblaje del en físico, dentro de este ensamblaje se hizo presente la participación del equipo.

El chasis del robot móvil diferencial está fabricado en metal con acabado en color negro, lo que le proporciona resistencia estructural y una buena base para la instalación de los componentes. Sus dimensiones son de 140 mm de largo por 75 mm de ancho, con una extrucción de 2 mm lo cual ofrece un equilibrio entre compactación y espacio suficiente para la integración de sensores, motores y placas de control.



Figura 1: chasis

El diseño incluye cuatro extensiones en las esquinas, con 37 mm de largo y 42.15 mm de ancho, de las cuales dos están destinadas al montaje de los motores, permitiendo una conexión firme y alineada. La estructura cuenta además con múltiples ranuras y orificios, los cuales están estratégicamente distribuidos para facilitar la instalación de componentes electrónicos, sensores, y sistemas de fijación, proporcionando así versatilidad en el montaje y posibles ajustes futuros.

El diseño de la rueda loca está compuesto por dos partes principales: la parte A y la parte B. Ambas piezas se ensamblan al final para formar una sola unidad funcional. En particular, la parte B presenta las siguientes características dimensionales, Altura total: 27.5 mm, Diámetro exterior: 49.6 mm, Redondeos: 2 mm en los bordes para mejorar la estética y reducir puntos de tensión. Diámetro interior (para el ensamblaje con la parte A): 10 mm.

Esta pieza está diseñada para acoplarse con precisión a la parte A, permitiendo un ensamblaje firme y estable.



Figura 2: Rueda Parte B

La parte A del diseño comparte una geometría general similar con la parte B; sin embargo, se diferencia principalmente en las dimensiones de ensamble, ya que esta pieza debe acoplarse a la parte B. Las especificaciones dimensionales de la parte A son las siguientes, Altura total: 25.15 mm, Diámetro exterior: 49.65 mm, Diámetro de ensamble (con la parte B): más pequeño que el de la parte B para lograr un ajuste firme y preciso. Extrusión para ensamblar con la parte B: 6.85 mm de largo.



Figura 3: Rueda Parte A

Esta pieza corresponde al piñón de transmisión principal, el cual va montado directamente al eje del motor y se encarga de transmitir el movimiento hacia las ruedas oruga del sistema. Características del diseño: Diámetro exterior con dientes: 46 mm, Diámetro del núcleo central: 41 mm, Diámetro del orificio central de acoplamiento al eje del motor: 4 mm, Altura total de la pieza: 31 mm, Altura del cubo central: 13 mm, Geometría del diente: cuenta con dientes uniformemente distribuidos (ángulo entre dientes de 12°), lo cual permite una tracción eficiente y alineada con la oruga.



Figura 4: Piñón

El diseño de nuestras llantas orugas esta compuesto por 47 instancias (eslabones) con una separación uniforme de 8.13 mm entre cada uno.

Características del diseño: Longitud total del tramo recto horizontal: 109.80 mm, proporcionando la distancia entre los dos extremos del sistema de tracción. Extremos semicirculares: con un radio de 25.68 mm, permiten el cambio de dirección de la banda de oruga con un movimiento suave, lo que evita fricción excesiva y facilita el desplazamiento continuo.



Figura 5: Diseño Oruga

Esta pieza corresponde a la carcasa superior del robot móvil diferencial. Tiene como función principal alojar y proteger los componentes electrónicos internos, tales como la tarjeta controladora, módulos de comunicación, cableado y sensores.

Características del diseño: Dimensiones generales: 179 mm de largo, 95 mm de ancho y 65 mm de alto. Abertura superior central: orificio de 30 mm de diámetro, destinado al paso de cables o montaje de componentes como

sensores o actuadores. Perforaciones: tiene 4 perforaciones rectangulares de 10 mm de largo y 5.42 de ancho para el montaje de los sensores.

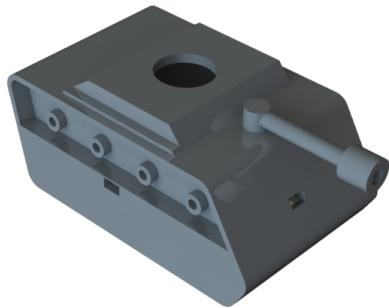


Figura 6: Diseño del Tanque

4.4. DISEÑO ELECTRÓNICO

Un ESP32 se utiliza para controlar dos motores con encoder de 12V, gestionados a través de un puente H L298N, permitiendo el movimiento preciso del sistema. Para el procesamiento central, se incorpora una Raspberry Pi, la cual se conecta a un sensor LiDAR, encargado de la detección y mapeo del entorno en tiempo real. Los datos recopilados por el LiDAR son procesados mediante un script en Python, que permite identificar obstáculos y construir un mapa 2D del área explorada.

En las conexiones del puente H se encuentra en salida uno la salida de voltaje positivo del motor, en salida dos la salida negativa del voltaje del motor, en las salidas 3 y 4 es el mismo caso, pero en esta ocasión con respecto al segundo motor.

En la salida ENA del puente H se encuentra conectada el pin 12 del ESP32, seguido de la salida IN1 al pin 14, IN2 al pin 27, IN3 al pin 16, IN4 al pin 17, y por último, la salida ENB al pin 13 del ESP32.

4.5. IMPLEMENTACIÓN Y ENSAMBLAJE

Ubicación de los sensores

Sensor LiDAR: Se encuentra ubicado en la parte superior central del robot, montado sobre una plataforma elevada. Esta posición le permite tener un campo de visión de 360° para mapeo del entorno y detección de obstáculos.

Ensamble de motores y puente H (L298N)

El robot móvil diferencial utiliza dos motores DC, acoplados a los ejes de las orugas izquierda y derecha. Estos motores están conectados a un módulo puente H L298N, el cual permite controlar la dirección y velocidad de cada oruga de manera independiente. El módulo está fijado dentro del chasis, protegido de posibles impactos, y conectado a la fuente de alimentación principal.

Posición de los módulos ESP32 y Raspberry Pi

ESP32: Está montado dentro del compartimento central del robot, conectado al puente H y otros elementos. Su función principal es el control de bajo nivel (motores).

Raspberry Pi: Ubicada también en el interior, está dedicada al procesamiento de alto nivel, como la interpretación de datos del LiDAR, navegación y comunicación con interfaces externas.

Alimentación del sistema

Todo el sistema se alimenta a través de una batería LiPo de 7.4V o 11.1V, conectada a un módulo reductor de voltaje que distribuye la energía en dos niveles:

- **5V** para la Raspberry Pi, sensores y algunos periféricos.
- **6–12V** para los motores a través del puente H L298N.



Figura 7: Robot móvil diferencial

5. PRUEBAS Y RESULTADOS

5.1. Pruebas

5.1.1. URDF

Primero realizamos el URDF el cual exportamos directamente desde SolidWorks con una extensión. Con nuestro modelo ya creado y ensamblado, verificamos las partes que se van a mover y una vez que lo exportamos a URDF seleccionamos los Links y Joints asociados con nuestro modelo, en este caso al ser un robot diferencial solo 2 ruedas serán con movimiento continuo y están conectadas con los motores con encoder.

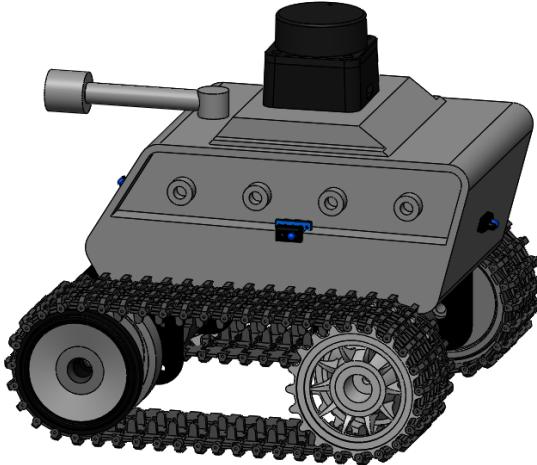


Figura 8: Modelo ensamblado en SolidWorks

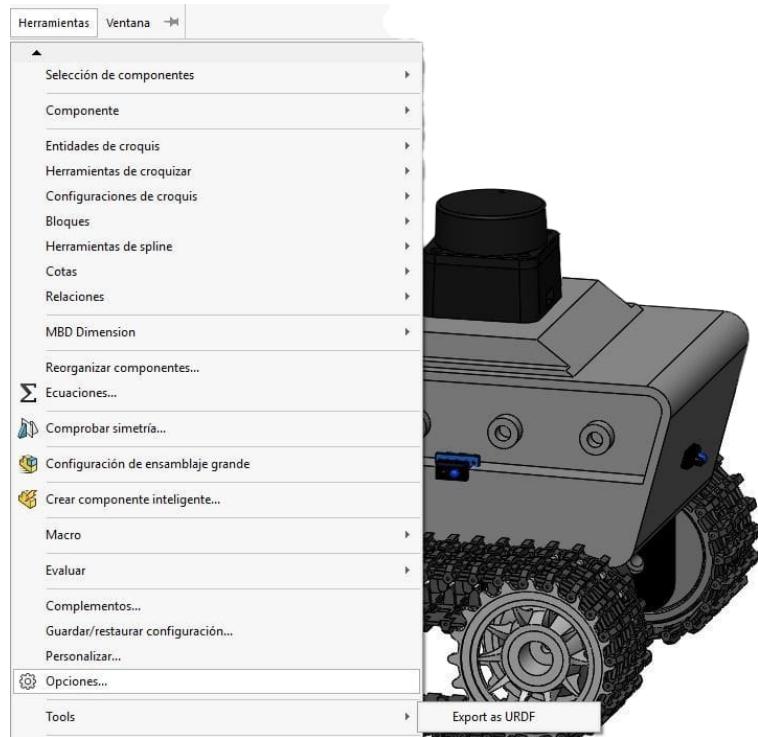


Figura 9: Exportar a URDF

5.1.2. RViz2

Una vez teniendo nuestro archivo URDF que se nos genero, podemos visualizarlo en RViz2 que también nos permitirá ver las transformaciones y mover las articulaciones con movimiento y verificar si el robot fue creado correctamente. Para visualizarlo, configuramos nuestro paquete en nuestro workspace para que puede leer el URDF y las meshes, ademas de crear una carpeta para agregar los códigos de lanzamiento.

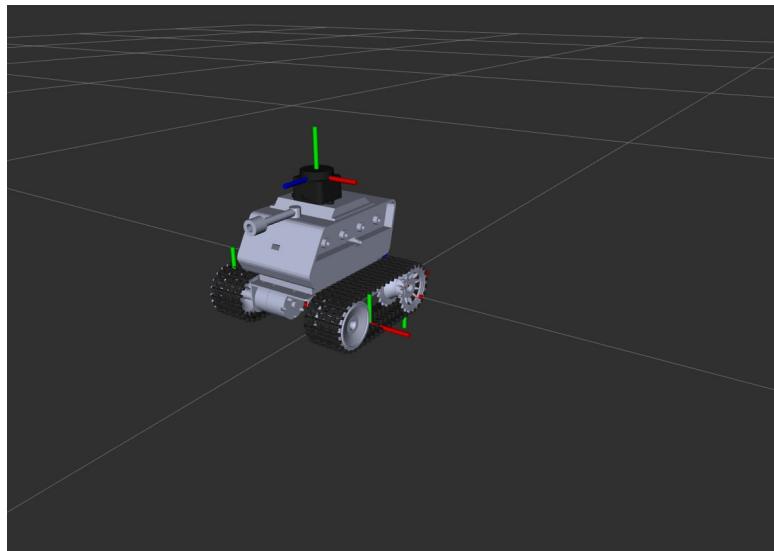


Figura 10: Modelo visualizado en Rviz2

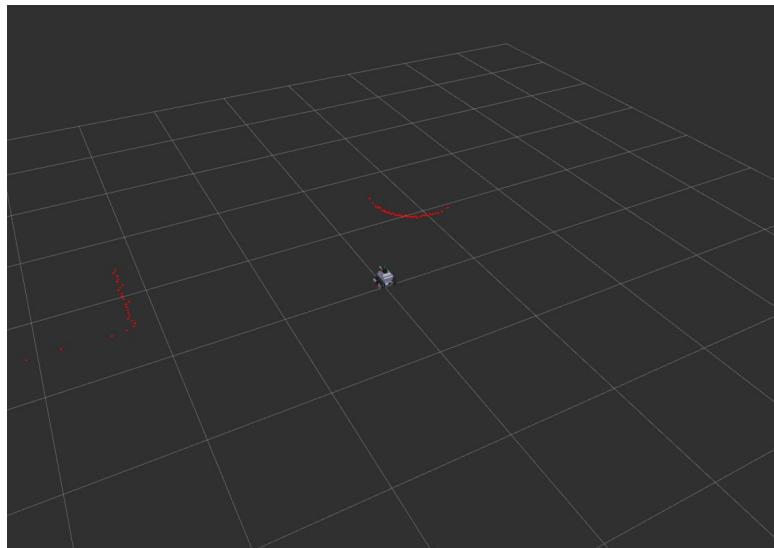


Figura 11: Modelo visualizado en Rviz2

5.1.3. Gazebo

Una vez visualizado el modelo del robot en Rviz2 y verificar que todo esta correcto podemos simular directamente el modelo en Gazebo, con Gazebo se puede ver el movimiento real del robot agregando los tópicos y plugins para obtener los datos tanto del movimiento como de los sensores.

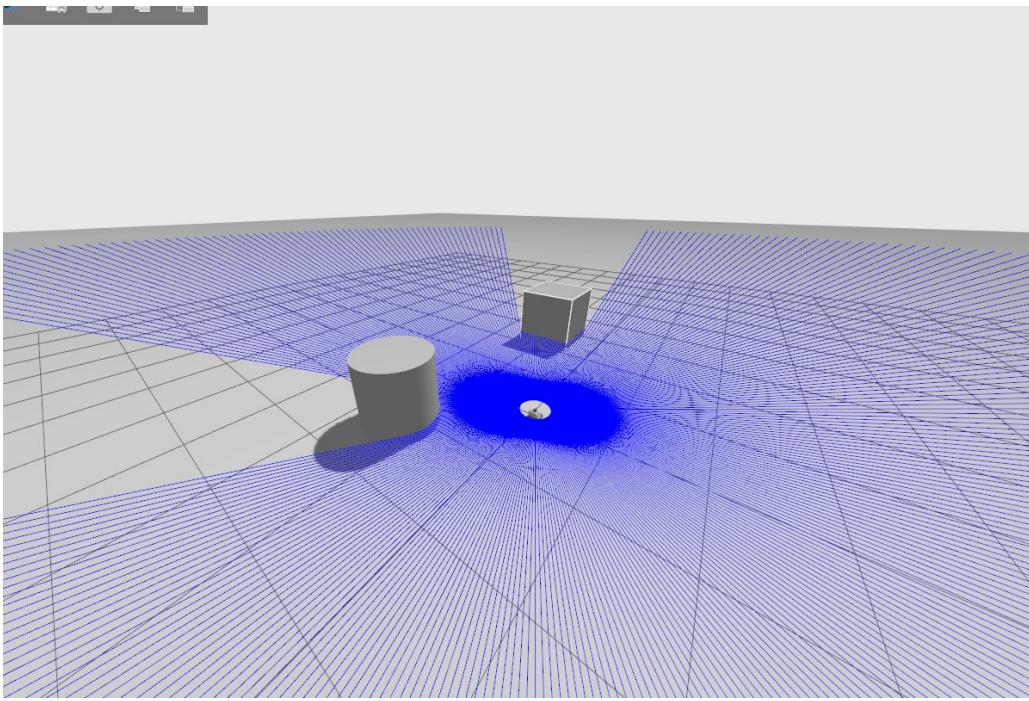


Figura 12: Simulación en Gazebo

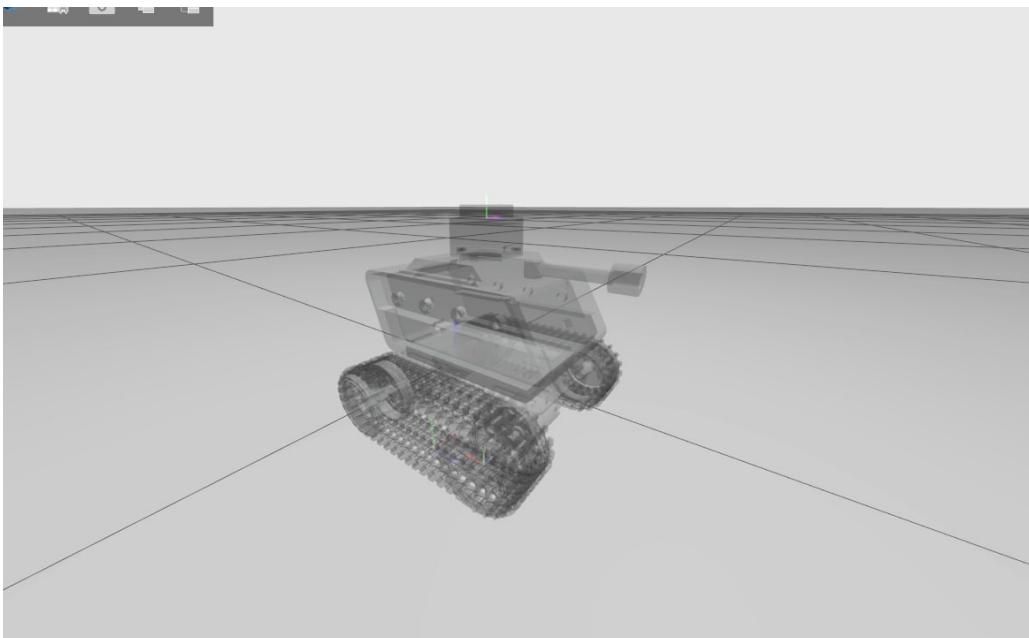


Figura 13: Simulación en Gazebo

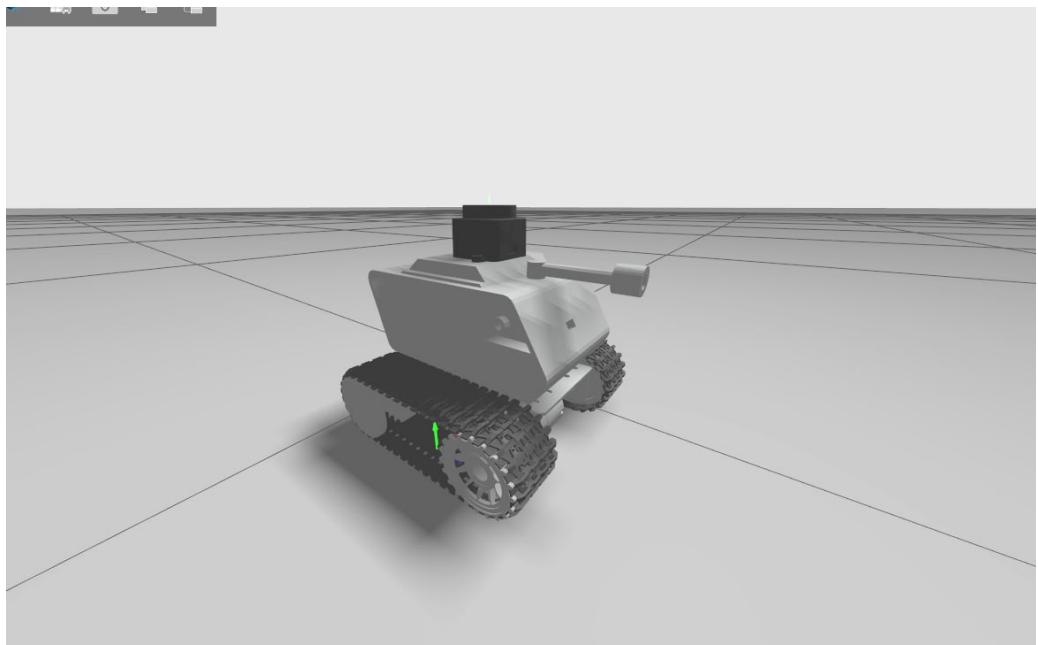


Figura 14: Simulación en Gazebo

5.1.4. LiDAR

Prueba del sensor LiDAR en RViz2, este sensor va a hacer el principal para poder mapear el entorno, este sensor ya esta diseñado para este tipo de proyectos.

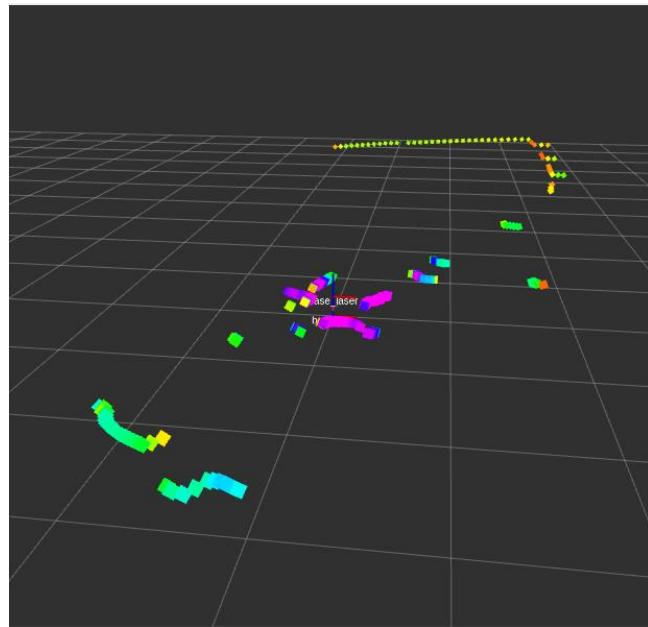


Figura 15: Lectura de sensor LiDAR.

5.1.5. Micro-ROS

1. Acceder al espacio de trabajo de ROS 2

```
cd ros_ws
```

Este comando cambia la directorio del espacio de trabajo de ROS 2, donde se encuentra las paquetes y configuraciones necesarias para compilar y ejecutar los nodos del sistema robótico.

2. Compilar los paquetes con colcon

```
colcon build
```

Se compilan todos los paquetes dentro del espacio de trabajo. Este paso es necesario cada vez que se modifican o agregan archivos fuente, para que ROS 2 los reconozca correctamente.

3. Configurar el entorno

```
source install/setup.bash
```

Este comando carga el entorno de trabajo, estableciendo las variables necesarias para que los paquetes recién compilados puedan ser ejecutados por ROS 2.

4. Ejecutar el agente de micro-ROS

```
ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888
```

Se lanza el agente de micro-ROS, que permite establecer la comunicación entre el ESP32 (que ejecuta el cliente micro-ROS) y la computadora que ejecuta ROS 2.

Aquí se utiliza el protocolo UDP en el puerto 8888.

La IP del ESP32 en nuestro caso 192.168.0.28 debe coincidir con el destino del agente, que está esperando recibir mensajes.

5. Verificar los tópicos activos

```
ros2 topic list
```

Este comando muestra todos los tópicos actualmente publicados en la red de ROS 2. Es útil para confirmar que el ESP32 está enviando correctamente mensajes y que están siendo reconocidos por el agente.

6. Controlar el robot desde el teclado

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Con este comando se ejecuta una interfaz de control por teclado (teleop).

Al presionar teclas como W, A, S, D, se publican mensajes del tipo "geometry_msgs/Twist" en un tópico, que luego el robot interpreta para moverse hacia adelante, atrás, girar, entre otras cosas.

5.1.6. Nodos

- **lineturtle_esp32**

Nodo ejecutado en el microcontrolador ESP32 a través de micro-ROS. Se encarga de publicar los datos provenientes de los encoders de las ruedas y de suscribirse al tópico `/cmd_vel` para ejecutar comandos de movimiento. Este nodo sirve como puente entre el hardware del robot y el ecosistema ROS 2.

- **teleop_twist_keyboard**

Nodo ejecutado en la computadora host que permite controlar el movimiento del robot mediante el teclado. Publica mensajes del tipo `geometry_msgs/Twist` en el tópico `/cmd_vel`, los cuales son interpretados por el nodo `linetuttle_esp32` para mover el robot hacia adelante, atrás o girar.

5.1.7. Tópicos

- **/cmd_vel**

Tópico de tipo `geometry_msgs/Twist` utilizado para enviar comandos de velocidad lineal y angular al robot. Es publicado por interfaces de control como `teleop_twist_keyboard./derecho_motor_ticks`.
Publica el número de pulsos generados por el encoder del motor derecho. Esto es esencial para el cálculo de la odometría.

- **/izquierdo_motor_ticks**

Publica el número de pulsos del encoder del motor izquierdo, igualmente utilizado para estimaciones de velocidad y distancia recorrida.

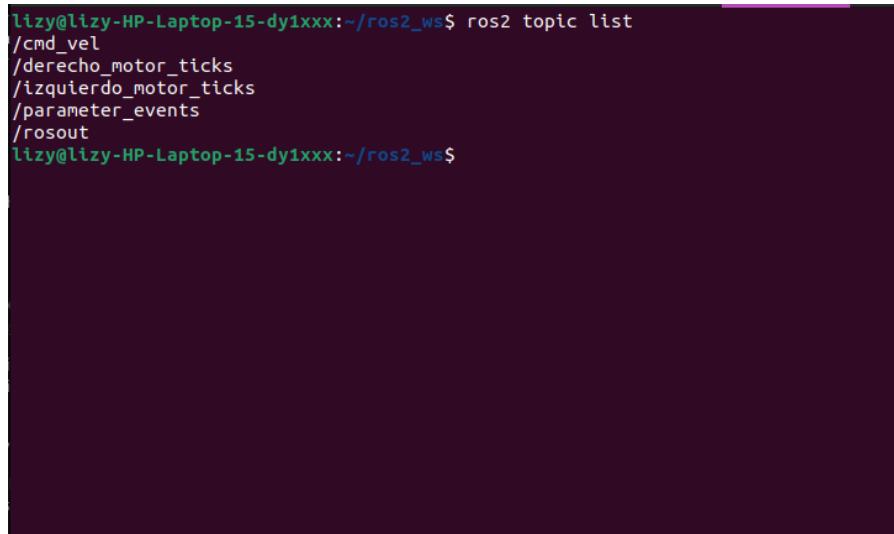
- **/parameter_events**

Tópico interno del sistema ROS 2 que notifica eventos relacionados con los parámetros de los nodos (creación, modificación o eliminación de parámetros).

- **/rosout**

Canal estándar donde los nodos publican mensajes de salida del sistema como advertencias, errores o información útil para depuración (*logging*).

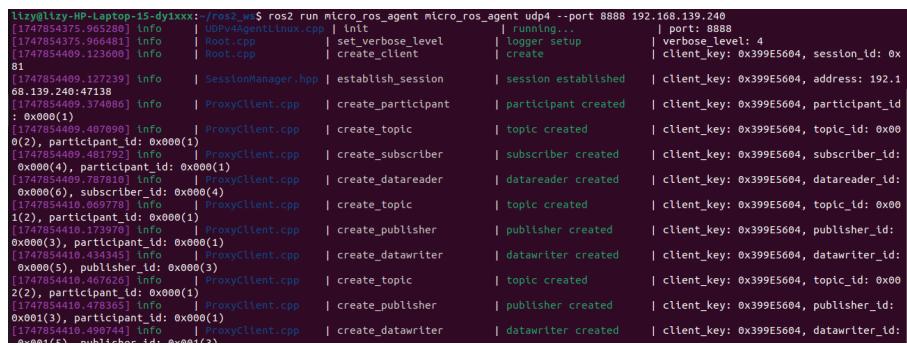
5.2. Resultados



A terminal window showing the output of the command `ros2 topic list`. The output lists several topics: `/cmd_vel`, `/derecho_motor_ticks`, `/izquierdo_motor_ticks`, `/parameter_events`, and `/rosout`.

```
lizy@lizy-HP-Laptop-15-dy1xxx:~/ros2_ws$ ros2 topic list
/cmd_vel
/derecho_motor_ticks
/izquierdo_motor_ticks
/parameter_events
/rosout
lizy@lizy-HP-Laptop-15-dy1xxx:~/ros2_ws$
```

Figura 16: Manejo del robot móvil diferencial con el teclado



A terminal window showing the output of the command `ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888 192.168.139.240`. The log shows the creation of a session, participants, topics, subscribers, datareaders, publishers, and datawriters.

```
lizy@lizy-HP-Laptop-15-dy1xxx:~/ros2_ws$ ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888 192.168.139.240
[1747854375.965288] Info  | UDPv4AgentLinux.cpp | init          | running...           | port: 8888
[1747854375.966481] Info  | Root.cpp        | set_verbose_level | logger setup       | verbose_level: 4
[1747854409.123608] Info  | Root.cpp        | create_client     | create             | client_key: 0x399E5604, session_id: 0x
81
[1747854409.127739] Info  | SessionManager.hpp| establish_session | session established | client_key: 0x399E5604, address: 192.1
68.139.240:47138
[1747854409.374086] Info  | ProxyClient.cpp | create_participant | participant created | client_key: 0x399E5604, participant_id:
0x000(1)
[1747854409.407698] Info  | ProxyClient.cpp | create_topic      | topic created      | client_key: 0x399E5604, topic_id: 0x00
0(2), participant_id: 0x000(1)
[1747854409.481792] Info  | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x399E5604, subscriber_id:
0x000(4), participant_id: 0x000(1)
[1747854409.787810] Info  | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x399E5604, datareader_id:
0x000(6), subscriber_id: 0x000(4)
[1747854410.089778] Info  | ProxyClient.cpp | create_topic      | topic created      | client_key: 0x399E5604, topic_id: 0x00
0(3), participant_id: 0x000(1)
[1747854410.179778] Info  | ProxyClient.cpp | create_publisher   | publisher created  | client_key: 0x399E5604, publisher_id:
0x000(3), participant_id: 0x000(1)
[1747854410.434345] Info  | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x399E5604, datawriter_id:
0x000(5), publisher_id: 0x000(3)
[1747854410.467626] Info  | ProxyClient.cpp | create_topic      | topic created      | client_key: 0x399E5604, topic_id: 0x00
2(2), participant_id: 0x000(1)
[1747854410.478365] Info  | ProxyClient.cpp | create_publisher   | publisher created  | client_key: 0x399E5604, publisher_id:
0x001(3), participant_id: 0x000(1)
[1747854410.496744] Info  | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x399E5604, datawriter_id:
0x000(4), publisher_id: 0x000(3)
```

Figura 17: Comprobación del funcionamiento de Micro-ROS

6. BIBLIOGRAFÍA

Referencias

- [Edi23] Edisonsasig. Modelo cinemático y simulación de un robot móvil diferencial, 7 2023.
 - [Lim24] Henry Lim. Introduction to SLAM (Simultaneous Localization and Mapping), 4 2024.
 - [Mar25] Elena Martínez. Sistema ROS para el movimiento de Robots - ATRIA Innovation, 3 2025.
 - [RMD20] Maria Ibáñez Rubio, Rosario Aragüés Munoz, and Enrique Teruel Donate. Control de vuelo para seguimiento de trayectorias de un cuadricóptero simulado mediante ros/gazebo autopilot for trajectory tracking of a quadcopter simulated in ros/gazebo. 2020.
 - [TC23] Daniella Tola and Peter Corke. Understanding urdf: A dataset and analysis, 08 2023.
 - [unk] Simulating with Gazebo — Articulated Robotics.
- [\[TC23\]](#) [\[Edi23\]](#) [\[Lim24\]](#) [\[Mar25\]](#) [\[unk\]](#)

A. ANEXOS

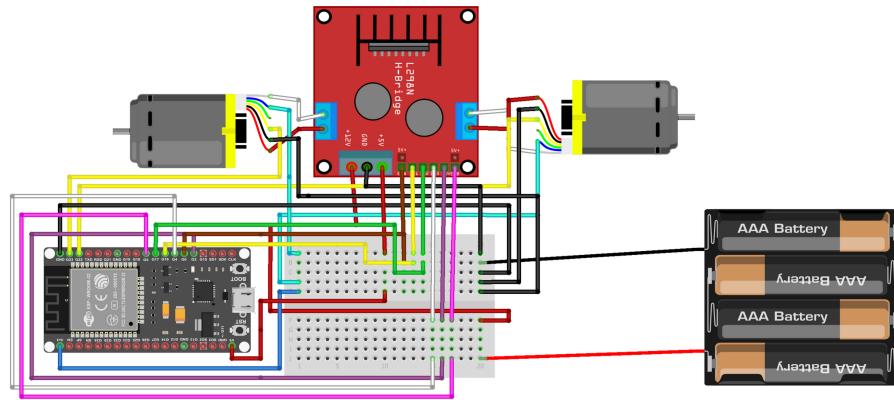


Figura 18: Diseño eléctrico de motores

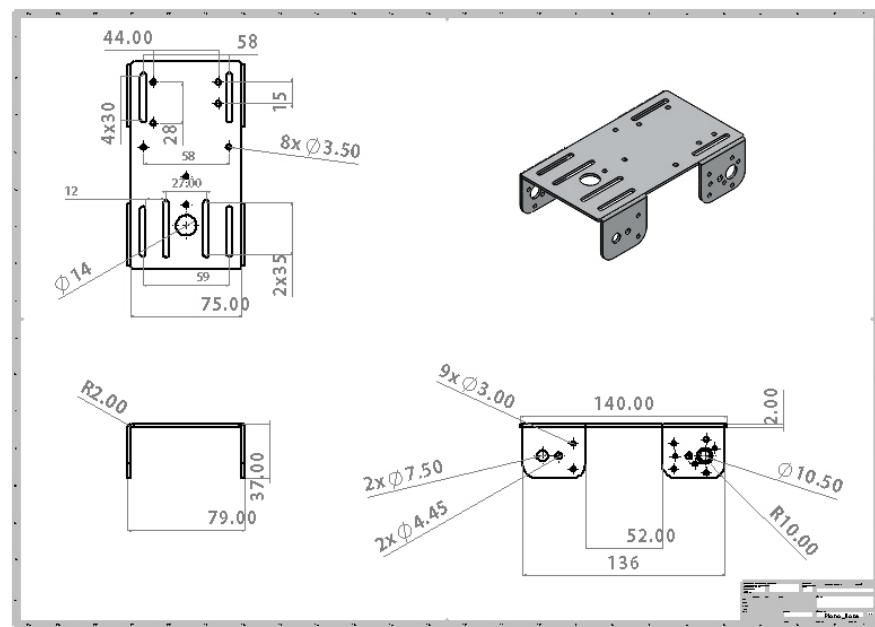


Figura 19: Plano Chasis

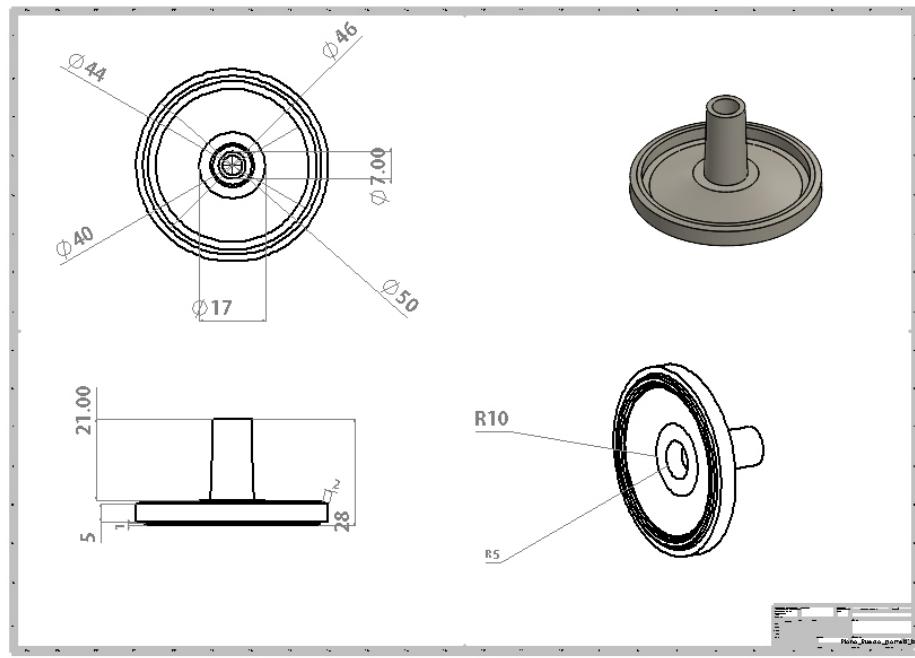


Figura 20: Plano rueda parte B

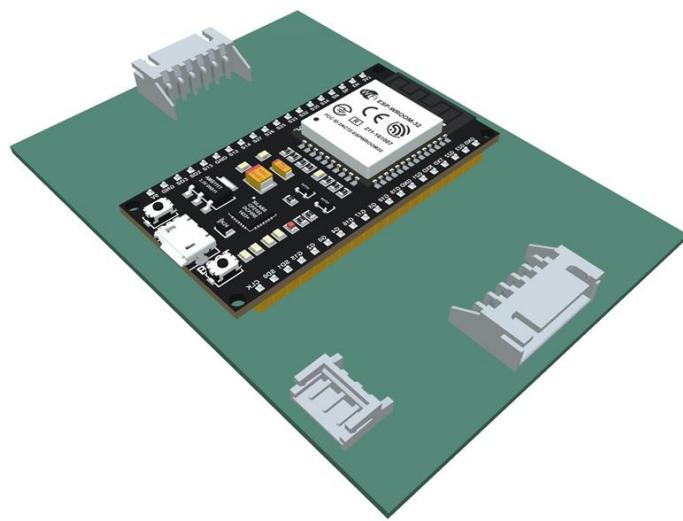


Figura 21: Diseño eléctrico de motores

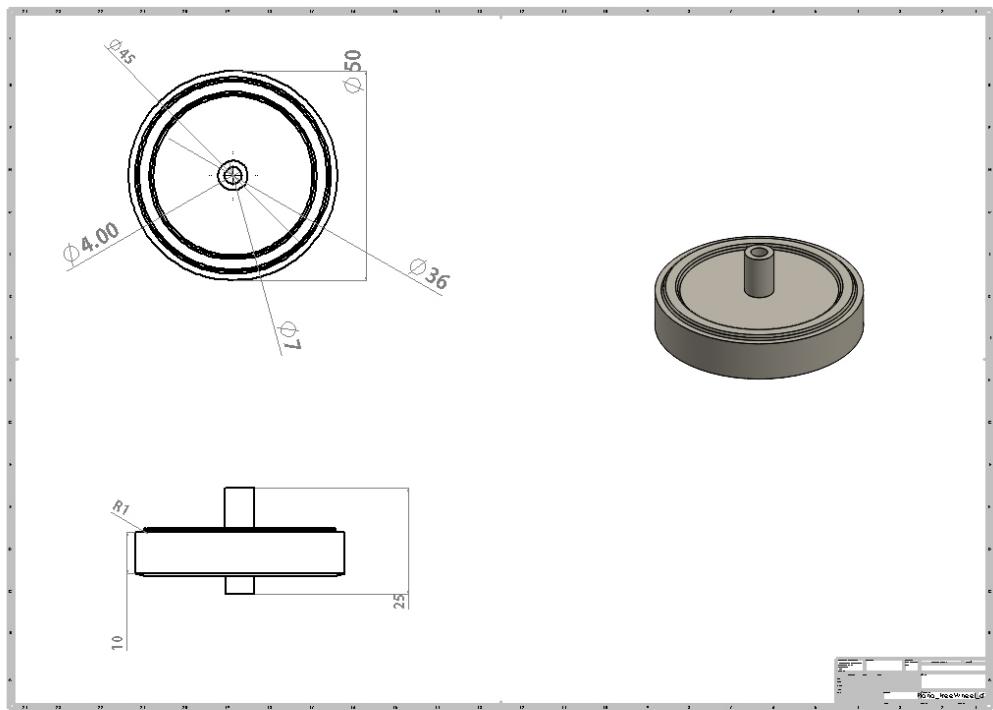


Figura 22: Plano rueda parte A

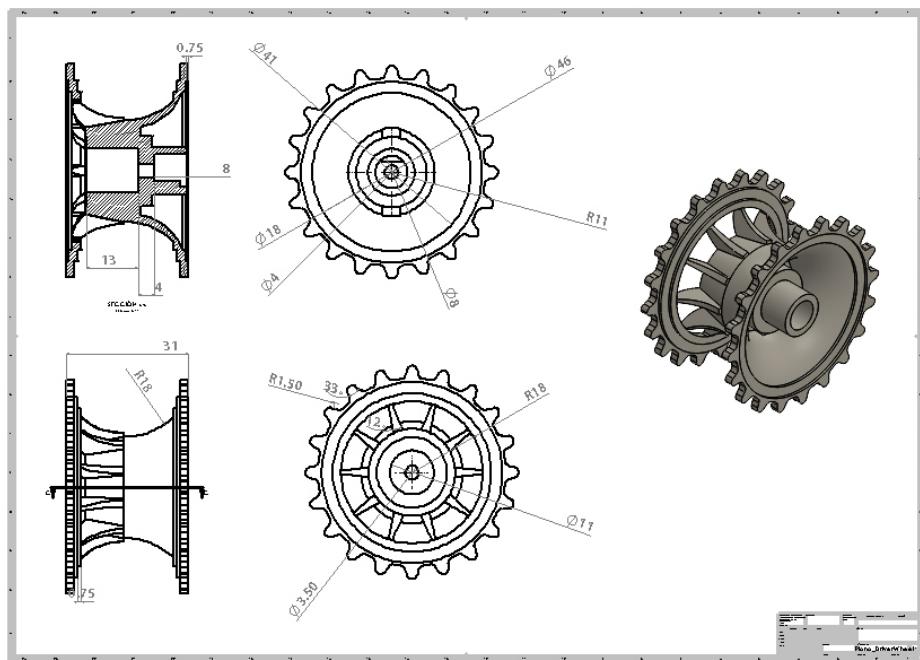


Figura 23: Plano piñón

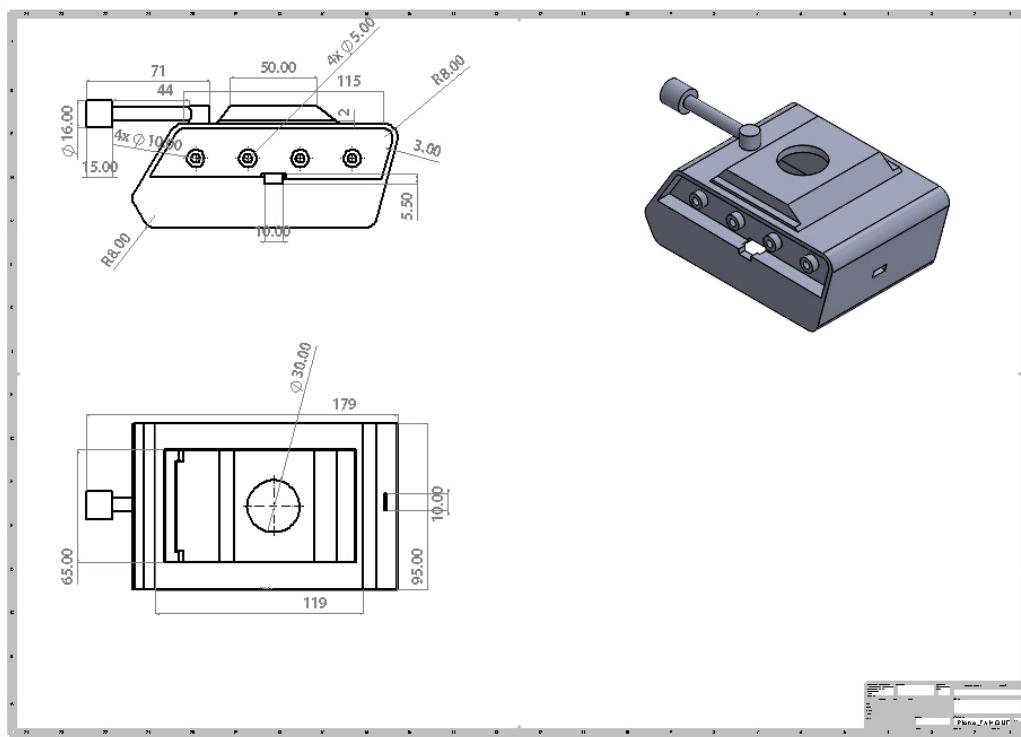


Figura 24: Plano tanque

A.1. Código URDF

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!-- This URDF was automatically created by SolidWorks to URDF Exporter!
3   Originally created by Stephen Brawner (brawner@gmail.com)
4   Commit Version: 1.6.0-4-g7f85cfe Build Version: 1.6.7995.38578
5   For more information, please see http://wiki.ros.org/sw_urdf_exporter-->
6 <robot
7   name="Carrito_Completo">
8     <link
9       name="base_link">
10      <inertial>
11        <origin
12          xyz="0.0606652029744953 -0.110561115570321 0.0715143463649022"
13          rpy="0 0 0" />
14        <mass
15          value="0.235396400581228" />
16        <inertia
17          ixx="0.000465846044179388"
18          ixy="5.13733784902491E-06"
19          ixz="-1.23366751234216E-09"
20          iyy="0.000223485618625359"
21          iyz="-1.68465191912738E-06"
22          izz="0.000637995033688768" />
23      </inertial>
24      <visual>
25        <origin
26          xyz="0 0 0"
27          rpy="0 0 0" />
28        <geometry>
29          <mesh
30            filename="package://modelo_robot/meshes/base_link.STL" />
31        </geometry>
32        <material
33          name="">
34          <color
35            rgba="0.650980392156863 0.619607843137255 0.588235294117647 1" />
36        </material>
37      </visual>
38      <collision>
39        <origin
40          xyz="0 0 0"
41          rpy="0 0 0" />
42        <geometry>
43          <mesh
44            filename="package://modelo_robot/meshes/base_link.STL" />
45        </geometry>
46      </collision>
47    </link>
48    <link
49      name="Motor1_Link">
50      <inertial>
51        <origin
52          xyz="-0.000128088993923248 2.01329385063204E-05 0.0309834580656092"
53          rpy="0 0 0" />
```

```

54     <mass
55         value="0.00169360706016677" />
56     <inertia
57         ixx="7.58897323779908E-08"
58         ixy="-9.30659196917006E-11"
59         ixz="-1.49688937569123E-09"
60         iyy="7.53122628973331E-08"
61         iyz="2.3528051929544E-10"
62         izz="3.09996788763902E-08" />
63 </inertial>
64     <visual>
65         <origin
66             xyz="0 0 0"
67             rpy="0 0 0" />
68         <geometry>
69             <mesh
70                 filename="package://modelo_robot/meshes/Motor1_Link.STL" />
71         </geometry>
72         <material
73             name="">
74             <color
75                 rgba="0.792156862745098 0.819607843137255 0.9333333333333333 1" />
76         </material>
77     </visual>
78     <collision>
79         <origin
80             xyz="0 0 0"
81             rpy="0 0 0" />
82         <geometry>
83             <mesh
84                 filename="package://modelo_robot/meshes/Motor1_Link.STL" />
85         </geometry>
86     </collision>
87 </link>
88 <joint
89     name="Motor1_Joint"
90     type="continuous">
91     <origin
92         xyz="0.14131 -0.056802 0.05704"
93         rpy="1.5708 0 -1.5708" />
94     <parent
95         link="base_link" />
96     <child
97         link="Motor1_Link" />
98     <axis
99         xyz="0 0 -1" />
100 </joint>
101 <link
102     name="RuedaDentada1_Link">
103     <inertial>
104         <origin
105             xyz="1.06470451024451E-05 -1.6734962355533E-06 0.0103653997911204"
106             rpy="0 0 0" />
107         <mass
108             value="0.0131897296692292" />

```

```

109     <inertia
110         ixx="2.79914053327356E-06"
111         ixy="9.07093808848256E-10"
112         ixz="-1.4618372985992E-09"
113         iyy="2.80476902966058E-06"
114         iyz="2.29770719796608E-10"
115         izz="3.13286860001254E-06" />
116     </inertial>
117     <visual>
118         <origin
119             xyz="0 0 0"
120             rpy="0 0 0" />
121         <geometry>
122             <mesh
123                 filename="package://modelo_robot/meshes/RuedaDentada1_Link.STL" />
124         </geometry>
125         <material
126             name="">
127             <color
128                 rgba="0.843137254901961 0.815686274509804 0.752941176470588 1" />
129         </material>
130     </visual>
131     <collision>
132         <origin
133             xyz="0 0 0"
134             rpy="0 0 0" />
135         <geometry>
136             <mesh
137                 filename="package://modelo_robot/meshes/RuedaDentada1_Link.STL" />
138         </geometry>
139     </collision>
140 </link>
141 <joint
142     name="RuedaDentada1_Joint"
143     type="continuous">
144     <origin
145         xyz="0 0 0"
146         rpy="0 0 0" />
147     <parent
148         link="Motor1_Link" />
149     <child
150         link="RuedaDentada1_Link" />
151     <axis
152         xyz="0 0 -1" />
153 </joint>
154 <link
155     name="Rueda1_Link">
156     <inertial>
157         <origin
158             xyz="1.60253021574963E-07 1.36233197917113E-08 0.0166225640052029"
159             rpy="0 0 0" />
160         <mass
161             value="0.0146353128046997" />
162         <inertia
163             ixx="2.86431359854419E-06"

```

```

164     ixy="3.78094173425755E-11"
165     ixz="-5.13382875047584E-12"
166     iyy="2.86420056008203E-06"
167     iyz="-2.91956633530046E-12"
168     izz="5.12921905017638E-06" />
169 </inertial>
170 <visual>
171   <origin
172     xyz="0 0 0"
173     rpy="0 0 0" />
174   <geometry>
175     <mesh
176       filename="package://modelo_robot/meshes/Rueda1_Link.STL" />
177   </geometry>
178   <material
179     name="">
180     <color
181       rgba="0.843137254901961 0.815686274509804 0.752941176470588 1" />
182   </material>
183 </visual>
184 <collision>
185   <origin
186     xyz="0 0 0"
187     rpy="0 0 0" />
188   <geometry>
189     <mesh
190       filename="package://modelo_robot/meshes/Rueda1_Link.STL" />
191   </geometry>
192 </collision>
193 </link>
194 <joint
195   name="Rueda1_Joint"
196   type="continuous">
197   <origin
198     xyz="0.14356 -0.1666 0.05719"
199     rpy="1.5708 0 -1.5708" />
200   <parent
201     link="base_link" />
202   <child
203     link="Rueda1_Link" />
204   <axis
205     xyz="0 0 1" />
206 </joint>
207 <link
208   name="Motor2_Link">
209   <inertial>
210     <origin
211       xyz="0.000129543636545021 -5.52917786558055E-06 0.0309834580656092"
212       rpy="0 0 0" />
213     <mass
214       value="0.00169360706016677" />
215     <inertia
216       ixx="7.59032571524561E-08"
217       ixy="-2.58495918205669E-11"
218       ixz="1.51388885525118E-09"

```

```

219     iyy="7.52987381228679E-08"
220     iyz="-6.46162499955292E-11"
221     izz="3.09996788763902E-08" />
222 </inertial>
223 <visual>
224   <origin
225     xyz="0 0 0"
226     rpy="0 0 0" />
227 <geometry>
228   <mesh
229     filename="package://modelo_robot/meshes/Motor2_Link.STL" />
230 </geometry>
231 <material
232   name="">
233   <color
234     rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
235 </material>
236 </visual>
237 <collision>
238   <origin
239     xyz="0 0 0"
240     rpy="0 0 0" />
241   <geometry>
242     <mesh
243       filename="package://modelo_robot/meshes/Motor2_Link.STL" />
244     </geometry>
245   </collision>
246 </link>
247 <joint
248   name="Motor2_Joint"
249   type="continuous">
250   <origin
251     xyz="-0.019986 -0.1666 0.05704"
252     rpy="1.5708 0 1.5708" />
253   <parent
254     link="base_link" />
255   <child
256     link="Motor2_Link" />
257   <axis
258     xyz="0 0 -1" />
259 </joint>
260 <link
261   name="RuedaDentada2_Link">
262   <inertial>
263     <origin
264       xyz="-1.07679583947473E-05 4.59598816601736E-07 0.0103653997911205"
265       rpy="0 0 0" />
266     <mass
267       value="0.0131897296692291" />
268     <inertia
269       ixx="2.79900871046341E-06"
270       ixy="2.51947840096162E-10"
271       ixz="1.47843867100036E-09"
272       iyy="2.80490085247073E-06"
273       iyz="-6.31028314963949E-11"

```

```

274     izz="3.13286860001255E-06" />
275 </inertial>
276 <visual>
277   <origin
278     xyz="0 0 0"
279     rpy="0 0 0" />
280   <geometry>
281     <mesh
282       filename="package://modelo_robot/meshes/RuedaDentada2_Link.STL" />
283     </geometry>
284   <material
285     name="">
286     <color
287       rgba="0.843137254901961 0.815686274509804 0.752941176470588 1" />
288     </color>
289   </material>
290 </visual>
291 <collision>
292   <origin
293     xyz="0 0 0"
294     rpy="0 0 0" />
295   <geometry>
296     <mesh
297       filename="package://modelo_robot/meshes/RuedaDentada2_Link.STL" />
298     </geometry>
299   </collision>
300 </link>
301 <joint
302   name="RuedaDentada2_Joint"
303   type="continuous">
304   <origin
305     xyz="0 0 0"
306     rpy="0 0 0" />
307   <parent
308     link="Motor2_Link" />
309   <child
310     link="RuedaDentada2_Link" />
311   <axis
312     xyz="0 0 -1" />
313 </joint>
314 <link
315   name="Rueda2_Link">
316   <inertial>
317     <origin
318       xyz="-1.50895418847552E-07 -5.56524782438506E-08 0.0166225640052028"
319     <mass
320       value="0.0146353128046998" />
321     <inertia
322       ixx="2.86428629785049E-06"
323       ixy="6.14023826530502E-11"
324       ixz="4.17524053691283E-12"
325       iyy="2.86422786077575E-06"
326       iyz="4.17701229222821E-12"
327       izz="5.1292190501764E-06" />
328   </inertial>

```

```

329 <visual>
330   <origin
331     xyz="0 0 0"
332     rpy="0 0 0" />
333   <geometry>
334     <mesh
335       filename="package://modelo_robot/meshes/Rueda2_Link.STL" />
336   </geometry>
337   <material
338     name="">
339     <color
340       rgba="0.843137254901961 0.815686274509804 0.752941176470588 1" />
341   </material>
342 </visual>
343 <collision>
344   <origin
345     xyz="0 0 0"
346     rpy="0 0 0" />
347   <geometry>
348     <mesh
349       filename="package://modelo_robot/meshes/Rueda2_Link.STL" />
350   </geometry>
351 </collision>
352 </link>
353 <joint
354   name="Rueda2_Joint"
355   type="continuous">
356   <origin
357     xyz="-0.022236 -0.056801 0.05719"
358     rpy="1.5708 0 1.5708" />
359   <parent
360     link="base_link" />
361   <child
362     link="Rueda2_Link" />
363   <axis
364     xyz="0 0 1" />
365 </joint>
366 <link
367   name="Zapata1_Link">
368   <inertial>
369     <origin
370       xyz="0.000771572019376526 0.0256030633706692 2.04689703471594E-06"
371       rpy="0 0 0" />
372     <mass
373       value="0.0346502561990607" />
374     <inertia
375       ixx="3.9841426067531E-07"
376       ixy="-1.92796041641482E-10"
377       ixz="-5.12055819625237E-11"
378       iyy="5.26939178848389E-06"
379       iyz="4.31007731232225E-08"
380       izz="5.17897148167474E-06" />
381   </inertial>
382   <visual>
383     <origin

```

```

384     xyz="0 0 0"
385     rpy="0 0 0" />
386 <geometry>
387     <mesh
388         filename="package://modelo_robot/meshes/Zapata1_Link.STL" />
389 </geometry>
390 <material
391     name="">
392     <color
393         rgba="0.294117647058824 0.294117647058824 0.294117647058824 1" />
394 </material>
395 </visual>
396 <collision>
397     <origin
398         xyz="0 0 0"
399         rpy="0 0 0" />
400 <geometry>
401     <mesh
402         filename="package://modelo_robot/meshes/Zapata1_Link.STL" />
403 </geometry>
404 </collision>
405 </link>
406 <joint
407     name="Zapata1_Joint"
408     type="fixed">
409     <origin
410         xyz="0.12511 -0.11174 0.031435"
411         rpy="1.5694 0 0" />
412     <parent
413         link="base_link" />
414     <child
415         link="Zapata1_Link" />
416     <axis
417         xyz="0 0 0" />
418 </joint>
419 <link
420     name="Zapata2_Link">
421     <inertial>
422     <origin
423         xyz="0.000771572019376514 0.0256030633706692 2.04689703463268E-06"
424         rpy="0 0 0" />
425     <mass
426         value="0.0346502561990607" />
427     <inertia
428         ixx="3.98414260675311E-07"
429         ixy="-1.92796041641026E-10"
430         ixz="-5.12055819626158E-11"
431         iyy="5.26939178848389E-06"
432         iyz="4.31007731232226E-08"
433         izz="5.17897148167474E-06" />
434     </inertial>
435     <visual>
436     <origin
437         xyz="0 0 0"
438         rpy="0 0 0" />

```

```

439 <geometry>
440   <mesh
441     filename="package://modelo_robot/meshes/Zapata2_Link.STL" />
442   </geometry>
443   <material
444     name="">
445     <color
446       rgba="0.294117647058824 0.294117647058824 0.294117647058824 1" />
447   </material>
448 </visual>
449 <collision>
450   <origin
451     xyz="0 0 0"
452     rpy="0 0 0" />
453   <geometry>
454     <mesh
455       filename="package://modelo_robot/meshes/Zapata2_Link.STL" />
456     </geometry>
457   </collision>
458 </link>
459 <joint
460   name="Zapata2_Joint"
461   type="fixed">
462   <origin
463     xyz="-0.0037856 -0.11167 0.031435"
464     rpy="1.5694 0 -3.1416" />
465   <parent
466     link="base_link" />
467   <child
468     link="Zapata2_Link" />
469   <axis
470     xyz="0 0 0" />
471 </joint>
472 <link
473   name="MotorCuerpo1_Link">
474   <inertial>
475     <origin
476       xyz="6.93889390390723E-18 -5.91203580460609E-05 0.00403102506950906"
477       rpy="0 0 0" />
478     <mass
479       value="0.0340361492669301" />
480     <inertia
481       ixx="6.99359535201545E-06"
482       ixy="-1.6459069285762E-09"
483       ixz="6.78805090176328E-23"
484       iyy="6.96795761840755E-06"
485       iyz="-4.75658152548743E-08"
486       izz="4.48388636210458E-06" />
487   </inertial>
488   <visual>
489     <origin
490       xyz="0 0 0"
491       rpy="0 0 0" />
492     <geometry>
493       <mesh

```

```

494     filename="package://modelo_robot/meshes/MotorCuerpo1_Link.STL" />
495 </geometry>
496 <material
497   name="">
498   <color
499     rgba="0.792156862745098 0.819607843137255 0.9333333333333333 1" />
500 </material>
501 </visual>
502 <collision>
503   <origin
504     xyz="0 0 0"
505     rpy="0 0 0" />
506   <geometry>
507     <mesh
508       filename="package://modelo_robot/meshes/MotorCuerpo1_Link.STL" />
509     </geometry>
510   </collision>
511 </link>
512 <joint
513   name="MotorCuerpo1_Joint"
514   type="fixed">
515   <origin
516     xyz="0.082164 -0.062802 0.05701"
517     rpy="-1.5708 1.5658 1.5708" />
518   <parent
519     link="base_link" />
520   <child
521     link="MotorCuerpo1_Link" />
522   <axis
523     xyz="0 0 0" />
524 </joint>
525 <link
526   name="MotorCuerpo2_Link">
527   <inertial>
528     <origin
529       xyz="1.38777878078145E-17 -5.91203580461164E-05 0.00403102506950907"
530       rpy="0 0 0" />
531     <mass
532       value="0.0340361492669301" />
533     <inertia
534       ixx="6.99359535201545E-06"
535       ixy="-1.64590692857673E-09"
536       ixz="2.7048806030484E-22"
537       iyy="6.96795761840755E-06"
538       iyz="-4.75658152548731E-08"
539       izz="4.48388636210457E-06" />
540   </inertial>
541   <visual>
542     <origin
543       xyz="0 0 0"
544       rpy="0 0 0" />
545     <geometry>
546       <mesh
547         filename="package://modelo_robot/meshes/MotorCuerpo2_Link.STL" />
548     </geometry>

```

```

549      <material
550        name="">
551          <color
552            rgba="0.792156862745098 0.819607843137255 0.9333333333333333 1" />
553        </material>
554      </visual>
555      <collision>
556        <origin
557          xyz="0 0 0"
558          rpy="0 0 0" />
559        <geometry>
560          <mesh
561            filename="package://modelo_robot/meshes/MotorCuerpo2_Link.STL" />
562          </geometry>
563        </collision>
564      </link>
565      <joint
566        name="MotorCuerpo2_Joint"
567        type="fixed">
568        <origin
569          xyz="0.039164 -0.1606 0.05701"
570          rpy="-1.5708 1.5658 -1.5708" />
571        <parent
572          link="base_link" />
573        <child
574          link="MotorCuerpo2_Link" />
575        <axis
576          xyz="0 0 0" />
577      </joint>
578    </robot>

```