Simulación de un entorno de trabajo y plataforma robótica en ROS2 y Gazebo

[1]Flor de Liz Martínez García, [1]Dra. Yesenia Eleonor González Navarro, Dr.
[2]Abraham Rodríguez Mota

Departamento de posgrado UPIITA-IPN

[1]Instituto Politécnico Nacional-UPIITA
[2]Instituto Politécnico Nacional-CIC

fmartinezg1600@alumno.ipn.mx
ygonzalezn@ipn.mx
abrodriguezm@ipn.mx

Referencia de este artículo [1].

RESUMEN

El presente artículo aborda la creación de una plataforma robótica móvil en el entorno de simulación GAZEBO y la creación de un escenario de pruebas. Respecto a la plataforma robótica, esta consiste en un móvil omnidireccional de tres ruedas, integrado con una cámara RGBD. Sobre el escenario, se propone una cancha de color verde, sobre el cual se montan bloques azules en distintas posiciones y cantidades. El ecosistema empleado es ROS en su versión 2 Jazzy Jalisco integrado con el simulador Gazebo Harmonic, para facilitar el desarrollo y despliegue del entorno y de la plataforma robótica. Aunado a esto, se describe la estructura del espacio de trabajo desarrollado, un paso clave para el correcto funcionamiento de la simulación. También se describe de manera general el contenido del paquete programado en Python que se encuentra en el espacio de trabajo, así como los elementos necesarios para la generación de los objetos.

Palabras clave: ROS, Gazebo, simulación, robot móvil, cámara RGBD.

ABSTRACT

This article describes the creation of a mobile robotic platform within the GAZEBO simulation environment and the design of a test scenario. The robotic platform consists of a three-wheeled omnidirectional mobile robot integrated with an RGBD camera. The test scenario consists of a green playing field on which blue blocks are placed in various positions and quantities. The ecosystem used is ROS version 2 Jazzy Jalisco integrated with the Gazebo Harmonic simulator, to facilitate the development and deployment of the environment and the robotic platform. In addition to this, the structure of the developed workspace is described, a key step for the simulation to function correctly. A general description is also provided of the contents of the Python package found in the workspace, as well as the elements required for the generation of the objects.

Keywords: ROS, Gazebo, simulation, mobile robot, RGBD camera.

MSC 2020: 35R11

1. Introducción

Este trabajo se centra en el desarrollo de un robot móvil operando bajo el entorno de ROS2 y Gazebo, con el propósito de configurar un escenario de prueba para el robot, así como la visión rgbd del mismo. ROS (Sistema Operativo Robótico) es un conjunto de herramientas y librerías que ayudan a crear robots usando diversas plataformas y lenguajes de programación. ROS tiene dos versiones principales: ROS 1 que ha llegado al final de su desarrollo, sin nuevas versiones previstas y ROS 2 que sigue desarrollándose y lanza nuevas distribuciones cada año. Por otro lado, ROS colabora con otras plataformas de Robótica Abierta para facilitar el desarrollo y despliegue. Una de ellas es Gazebo que ofrece simulación basada en física, para probar robots en un entorno virtual antes de usar hardware real (Foundation, ROS 2 Documentation: Jazzy documentation, 2024). La simulación de un sistema robótico en Gazebo es un paso muy importante en el proceso de implementación de algoritmos en plataformas robóticas ya que permite una reducción de riesgos físicos, la validación de algoritmos de control y la facilidad de generar distintos entornos de prueba, incluso si no se tuvieran al alcance de manera física. Este trabajo se realizó en Ubuntu 24.04 LTS compatible con ROS2 Jazzy Jalisco y Gazebo Harmonic.

2. Metodología

En esta sección se hará referencia a la estructura del espacio de trabajo contenido en el repositorio https://github.com/flor-MCyTIACD/Simulacion-ROS2-Jazzy-con-Gazebo.git, que consiste en un espacio dedicado a la simulación de un robot en un escenario de prueba específico, cuyo paquete fue creado en Python y la estructura se detalla en la siguiente sección.

2.1 Configuración del espacio de trabajo (Workspace)

De acuerdo a la documentación de ROS (Robotics, s.f.), una buena práctica es crear un directorio para cada nuevo *Workspace* y es muy recomendable que el nombre indique el propósito del mismo. En este caso, nuestro *Workspace* se llamará *robot_ros2_BoletinU*. En una estructura estándar la carpeta raíz *robot_ros2_BoletinU/* debe tener una subcarpeta *src/* que contendrá el paquete del *Workspace*. El paquete es una unidad organizativa para nuestro código ROS 2, y es muy útil para poder instalar el código o para compartirlo con otros. La creación de un paquete se puede realizar mediante CMake o Python (Foundation, ROS 2 Documentation: Rolling, s.f.), en este caso se utilizará Python. Cada paquete contiene por default la siguiente estructura de archivos:

- package.xml
- setup.cfg
- setup.py
- test
- resource
- simulacion

Para la simulación es necesario crear los siguientes directorios adentro del paquete:

- launch: automatiza el funcionamiento de muchos nodos en un solo comando.
- materials/meshes: contiene los archivos de malla de un objeto tridimensional, los formatos admitidos son *.stl* y *.dae*.
- urdf: contiene archivos que son un estándar utilizado en ROS para describir las propiedades físicas de un robot.
- worlds: contiene archivos donde se configura el mundo de simulación y se añaden los plugins para interactuar con la simulación en Gazebo.

El árbol de directorios del Workspace debe quedar de la siguiente manera, ver Figura 1.

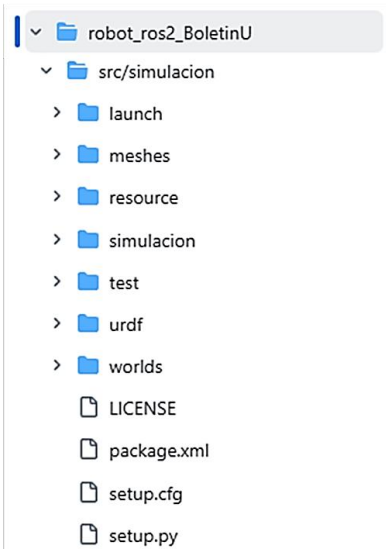

Figura 1. Estructura de carpetas en el Workspace para la integración con Gazebo.

2.2 Modelado del Robot

El modelado del robot se llevó a cabo en dos partes. La primera parte fue el diseño del robot a través del software Fusion 360, ver Figura 2. El diseño se exportó en un formato STL que define la geometría de la base del robot. El archivo STL debe ir en la carpeta meses/. La segunda parte del modelado fue la implementación en ROS 2. Para la configuración de la base del robot se generó un archivo robot.urdf dentro de la carpeta urdf/. Los elementos utilizados fueron los siguientes:

- *<link name = "nombre de la base">*: Aquí se define el nombre del eslabón del robot.
- *<visual>*: Define propiedades visuales como posición, rotación, forma de la pieza, el archivo 3D externo que contiene la forma del eslabón, la escala en que aparecerá en Gazebo, la textura y color del material.
- *<collision>*: Este elemento define las propiedades de colisión del eslabón.
- *<inertial>*: Establece propiedades inerciales, como el peso y tensor de inercia.

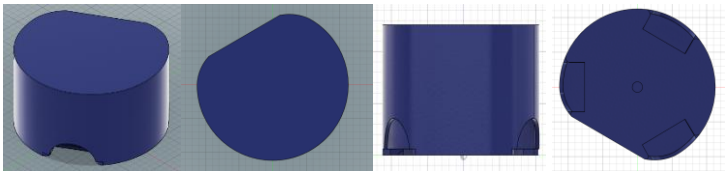

Figura 2. Vista isométrica, superior, lateral e inferior del chasis del robot.

Para las llantas del robot se emplearon modelos precargados en Gazebo. Al igual que en la base del robot, en las llantas se usaron los siguientes elementos:

- *<link name="nombre de llanta">*
- *<joint name="nombre de llanta" type="continuous">* al ser un elemento articular define la cinemática y la dinámica de la unión (Open Robotics, s.f.). Para las llantas se elige un tipo de unión continua que gira alrededor del eje y no tiene límites superiores ni inferiores.
- *<gazebo reference="right_wheel">* la extensión del formato de descripción del robot urdf para la simulación en Gazebo.

2.3 Simulación del Sensor RGB

Se utilizó un sensor tipo rgbd, que entrega una nube de puntos y profundidad. En urdf se usaron los elementos:

- *<link name="camera_link">*
- *<joint>* definió la posición *xyz* de la cámara para lo cual es importante conocer las medidas del robot.
- *<gazebo reference = "camera_link">*

Para ver los datos que publica el sensor, se configuran los tópicos que son el mecanismo por el cual los nodos intercambian mensajes. Los nodos son los responsables de realizar una función específica como publicar datos sobre sensores. Los tópicos de la cámara se configuraron en los argumentos del *main_bridge* del launch (Ver sim.launch.py). Con esto se habilitan los tópicos de imagen, profundidad, nube de puntos y cámara.

2.4 Escenarios de prueba

Se crearon 2 escenarios de prueba en la carpeta worlds/: *escenario_1.world* y *escenario_2.world*. En ambos escenarios se configuró el mismo tipo de iluminación *<light>.*

2.4.1 Escenario 1

Para crear una pista en el mundo Gazebo se usa el elemento *<model>* donde se definen características de posición inicial, colisión, color, tamaño y tipo de material. Todas las líneas delimitadoras se construyeron adentro del *<model>* de la pista para que sean un solo elemento. Las líneas delimitadoras laterales se crearon con elementos *<link>*, ambas con la misma geometría, color y material. Para la línea de meta, se definió también un *<link>,* pero con distinta posición, tamaño y color.

2.4.2 Escenario 2

En este escenario se utilizaron los mismos componentes que en el escenario 1. Para propósitos del proyecto se establecieron otras características de inercia en los bloques que van sobre la pista, se definieron como modelos independientes, con características de posición específicas, inercia, colisión, tamaño, color y material. El primer bloque se definió como <model name=”bloqueAzul_1”>, el segundo como <model name=”bloqueAzul_2”> y así sucesivamente hasta el último.

2.5 Visualización del Ambiente

Antes de hacer el lanzamiento en ROS 2, es necesario verificar las rutas en el **launch.py**. También se deben de registrar las rutas de los nodos de ROS 2 desde el **setup.py** que se encuentra ubicado dentro del paquete creado. Por último, en **package.xml** se deben de añadir las dependencias necesarias para la simulación.

2.6 Integración con Gazebo

Una vez configurado el entorno, se realiza la compilación y ejecución con *colcon build* y la sincronización con *source install/setup.bash*. Luego, se hace el lanzamiento con *ros2 launch*. Para cambiar el escenario a utilizar se debe configurar la ruta al archivo desde el código del *launch.py*.

2.7 Visualización de Imagen RGB

Mientras el *launch* está corriendo, en otra terminal ejecutar *ros2 topic list* para verificar si se están publicando los tópicos de la cámara, ver Figura 3. Si se encuentra */camera/image*, entonces Gazebo si está generando la imagen.

Figura 3. Lista de tópicos que se están publicando, entre ellos */camera/image*.

3. Resultados

3.1 Modelado del sensor y visualización del robot

En la Figura 4 se muestra el lanzamiento del robot y la cámara rgbd en Gazebo.

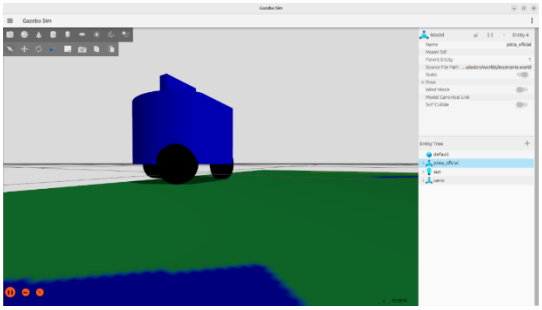

Figura 4. Spawneo del robot en Gazebo.

3.2 Visualización del robot y escenario1

Para lanzar el escenario 1 la ruta se define en el archivo launch.py. En la Figura 5a se muestra el escenario 1, la pista es de color verde tiene características de pasto sintético, dos líneas laterales amarillas delimitadoras y una línea de meta.

3.3 Visualización del robot y escenario2

Para el escenario 2 se utiliza la pista del escenario 1 pero añadiendo bloques azules con características de inercia específicas, ver Figura 5b.

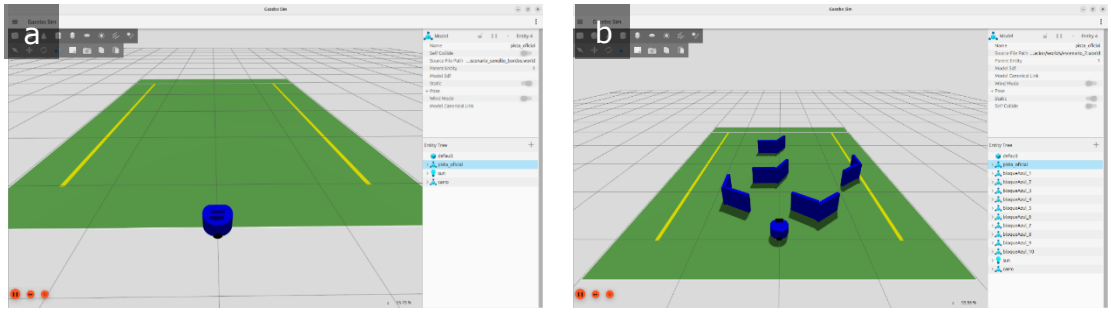

Figura 5. a) Visualización de escenario 2 en Gazebo. b) Visualización de escenario 1 en Gazebo.

3.4 Visualización de imagen RGB, robot y escenario.

Para ver la imagen de cámara en Gazebo, en el menú de opciones se selecciona *Image display*, esto despliega lo que la cámara está capturando en ese instante, ver parte inferior derecha de la Figura 6.

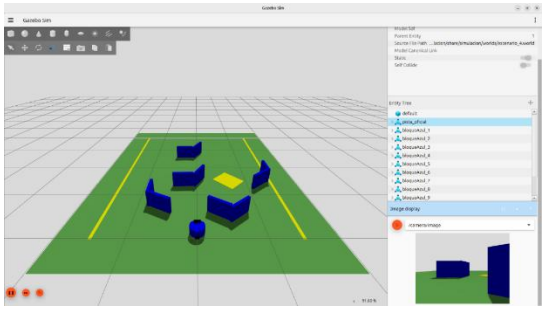

Figura 6. Visualización del escenario y de la imagen (esquina inferior derecha) que está captando el sensor rgbd.

4. Conclusiones

En conclusión, este artículo es una guía introductoria a la integración de ROS con Gazebo. Explicando desde la creación de un Workspace cuya estructura es clave para el correcto funcionamiento e integración de ROS 2 con Gazebo; el diseño de un escenario sencillo, para comprender la estructura en código de cada elemento; la personalización del robot a emplear, y la integración de una cámara con funcionamiento en línea. Una ventaja de hacer esta implementación es la transferencia del modelo desarrollado en la simulación a aplicaciones robóticas autónomas en el mundo real.

Referencias Bibliográficas

Foundation, O. S. **(6 de Mayo de 2024)**. *ROS 2 Documentation: Jazzy documentation.* Obtenido de https://docs.ros.org/en/jazzy/The-ROS2-Project/About-ROS.html

Foundation, O. S. (s.f.). *ROS 2 Documentation: Rolling*. Recuperado el **17 de Junio de 2026**, de https://docs.ros.org/en/rolling/Tutorials/Beginner-Client-Libraries/Creating-Your-First-ROS2-Package.html

Open Robotics. (s.f.). *ROS.org*. Recuperado el **23 de 05 de 2026**, de https://wiki.ros.org/urdf/XML/joint

Robotics, O. (s.f.). *About ROS, ROS 2 Documentation: Rolling.* Recuperado el **17 de Junio de 2026**, de https://docs.ros.org/en/rolling/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html#background