Contenido

[**Descripción** 1](#_Toc162848974)

[**Funcionamientos generales** 1](#_Toc162848975)

[**Funcionamiento para 1 robot (sin navegación autónoma)** 1](#_Toc162848976)

[**Funcionamiento para 1 robot (navegación autónoma)** 3](#_Toc162848977)

[**Funcionamiento para multirobots (sin navegación autónoma)** 5](#_Toc162848978)

[**Funcionamiento para multirobots (navegación autónoma)** 7](#_Toc162848979)

[**Añadir un mundo nuevo o mas robots** 10](#_Toc162848980)

[**Añadir nuevo mundo** 10](#_Toc162848981)

[**Añadir un nuevo ROVER** 12](#_Toc162848982)

# **Descripción**

El sistema multirobot desarrollado en ROS Melodic posee una plataforma estable y funcional diseñada para la exploración autónoma o asistida de entornos que se requieran navegar para diferentes tareas de investigación donde no intervenga la mano humana. Todos los robots dentro sistema son de tipo ROVER donde cada robot móvil posee una navegación autónoma de manera independiente por lo que permite una navegación más especial para entornos muy grandes donde un solo robot puede ser de poca utilidad.

El sistema cumple la viabilidad de poder incorporar de manera muy sencilla más robots sin la necesidad de reestructurar todo el código desarrollado, además, se cumple con el detalle de que el sistema puede soportar desde 1 robot hasta N robots donde la única limitante sería el poder computacional para incorporar N ROVERs.

# **Funcionamientos generales**

## **Funcionamiento para 1 robot (sin navegación autónoma)**

1. Ejecutar la simulación: roslaunch main gazebo\_house.launch

Hay más mundos donde para ejecutarlos son de la siguiente manera:

* roslaunch main gazebo\_empty.launch
* roslaunch main gazebo\_world.launch
* roslaunch main gazebo\_agriculture.launch

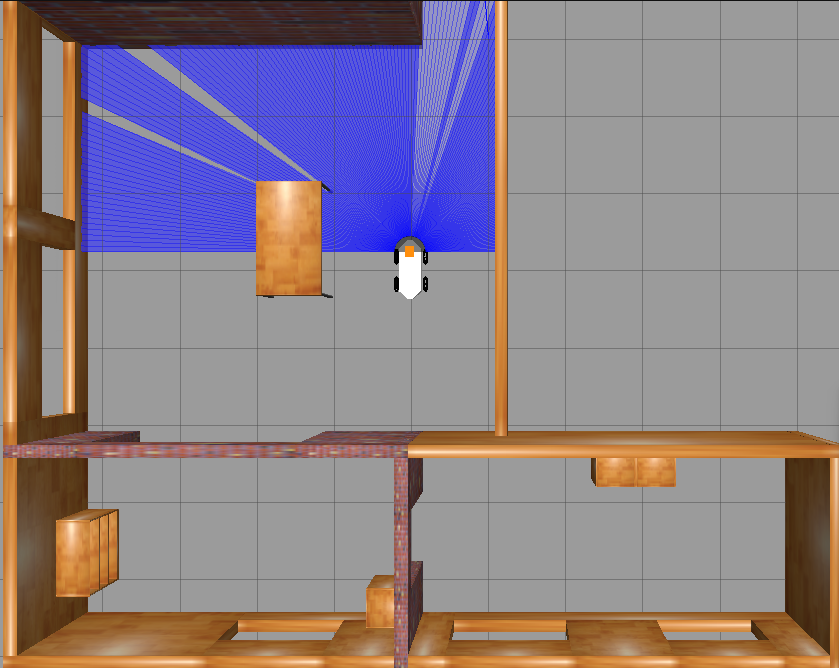


Figure 1 Simulación sin navegación autónoma - 1 robot

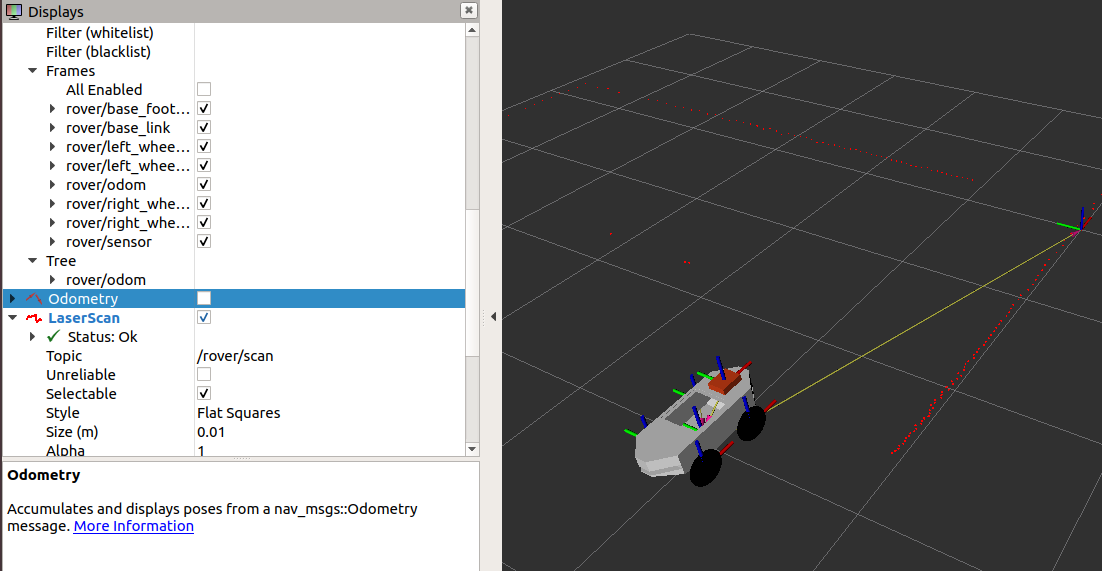


Figure 2 RViz sin navegación autónoma - 1 robot

1. Ejecutar la teleoperación: rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=rover/cmd\_vel

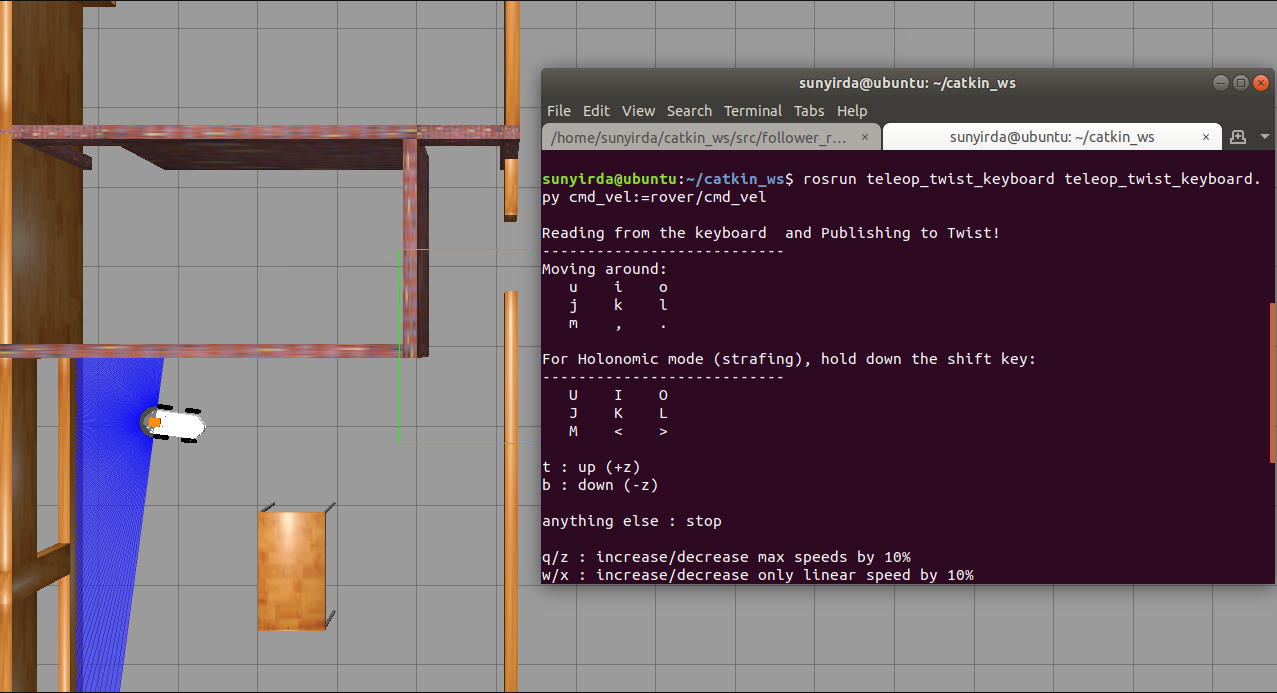


Figure 3 Teleoperación de ROVER

## **Funcionamiento para 1 robot (navegación autónoma)**

1. Ejecutar la simulación: roslaunch main gazebo\_house.launch navigation:=true

Hay más mundos donde para ejecutarlos son de la siguiente manera:

* roslaunch main gazebo\_empty.launch navigation:=true
* roslaunch main gazebo\_world.launch navigation:=true
* roslaunch main gazebo\_agriculture.launch navigation:=true

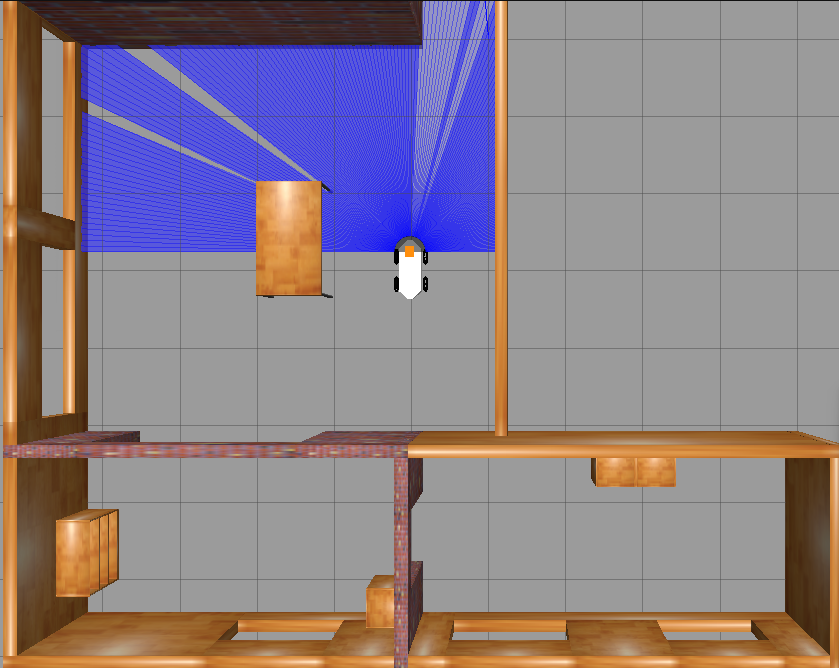


Figure 4 Simulación con navegación autónoma - 1 robot

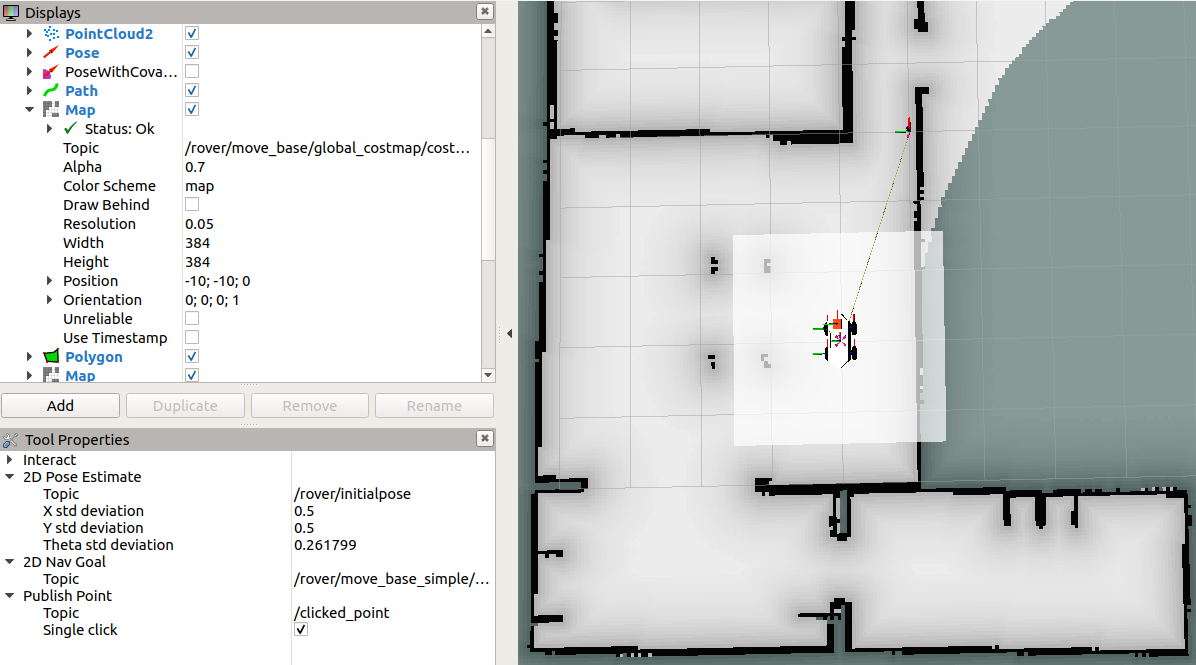


Figure 5 RViz con navegación autónoma - 1 robot

1. Enviar un punto de meta mediante RViz: Seleccionar la opción “2D Nav Goal” (flecha roja) y seleccionar el punto hacia donde se desee mover el robot (flecha azul).

Toda navegación autónoma mediante mapas requiere establecer una posicion inicial de movimiento, en este caso el sistema al ejecutarse se encarga de definir dicha posicion inicial.

Se puede enviar el punto de meta mediante los actions que ofrece ROS, el uso de RViz es una vista más intuitiva para mostrar el correcto funcionamiento.

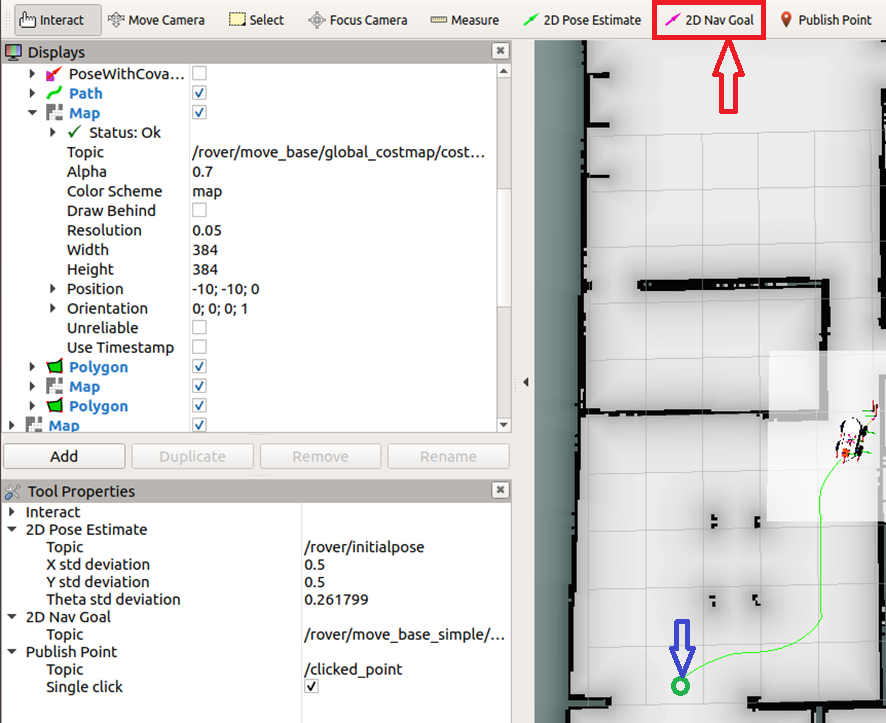


Figure 6 Enviar un punto de meta.

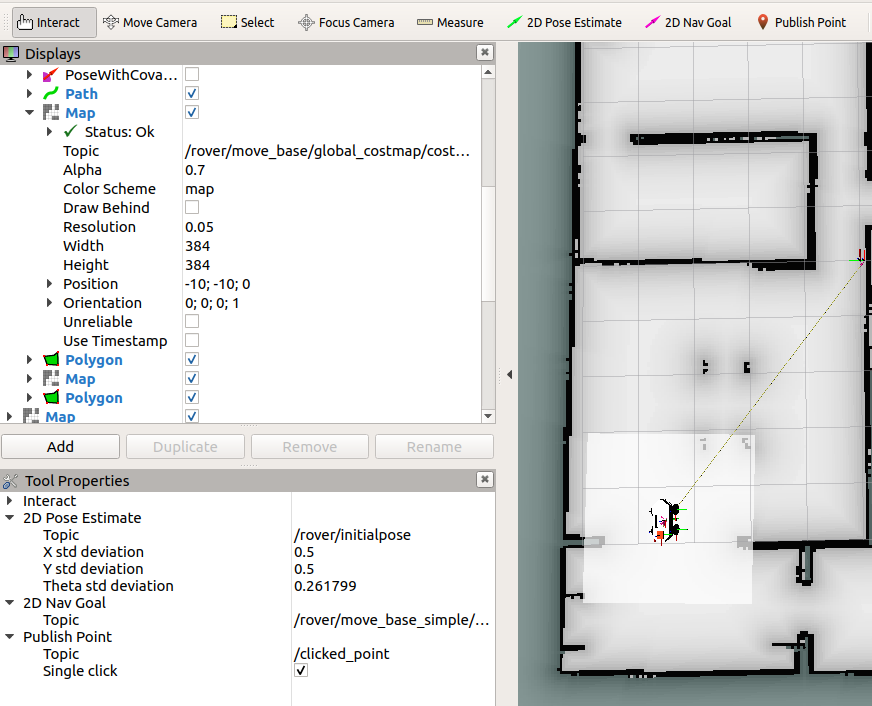


Figure 7 Robot en el punto de meta.

1. Ejecución de teleoperación (en caso de necesitar): rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=rover/cmd\_vel



Figure 8 Teleoperación con activación de navegación autonoma.

## **Funcionamiento para multirobots (sin navegación autónoma)**

1. Ejecutar la simulación: roslaunch main multi\_rover\_house.launch

Hay más mundos donde para ejecutarlos son de la siguiente manera:

* roslaunch main multi\_rover\_empty.launch
* roslaunch main multi\_rover\_world.launch

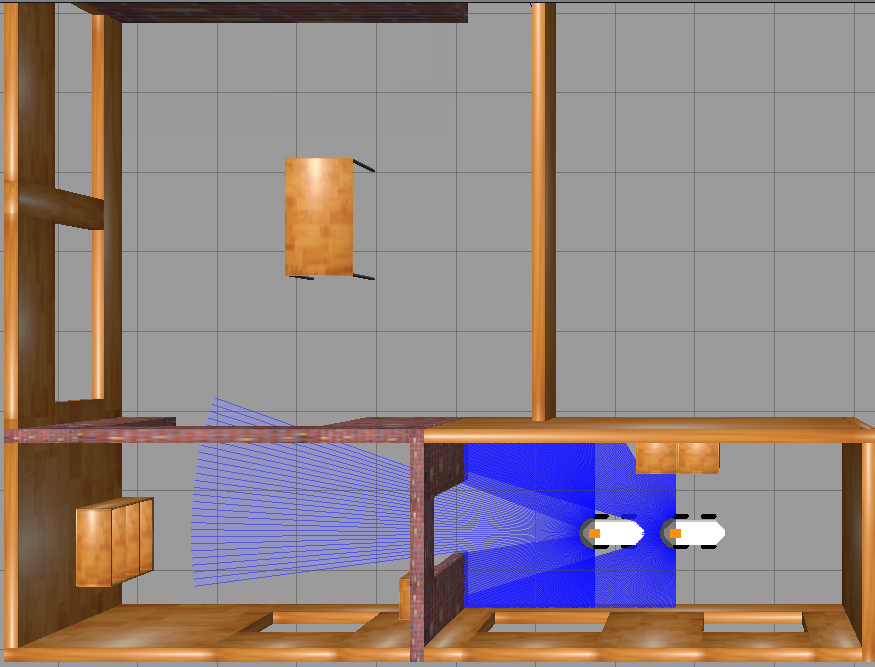


Figure 9 Simulación multirobot sin navegación autónoma

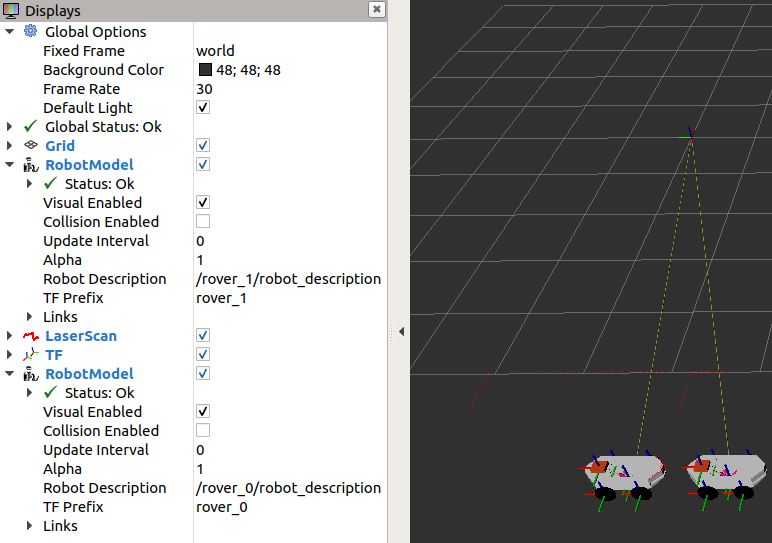


Figure 10 RViz multirobot sin navegación autónoma

1. Ejecutar la teleoperación (ejemplo con el robot llamado rover\_0): rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=NAME\_ROBOT/cmd\_vel

En el sistema multirobot cada topic depende del robot, por lo que si tenemos N robots entonces tendremos N topics cmd\_vel con el prefijo del nombre de su robot. En este caso al tener dos robots tendremos los siguientes topics:

* /rover\_0/cmd\_vel
* /rover\_1/cmd\_vel

**Nota importante:** Lo mismo sucede para los otros topics como para el robot\_description, odom, etc.

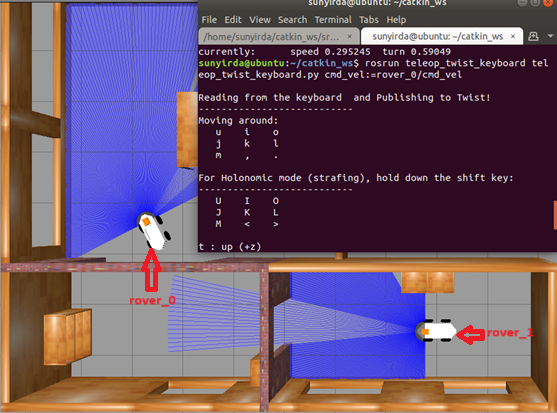


Figure 11 Teleoperación de rover\_0

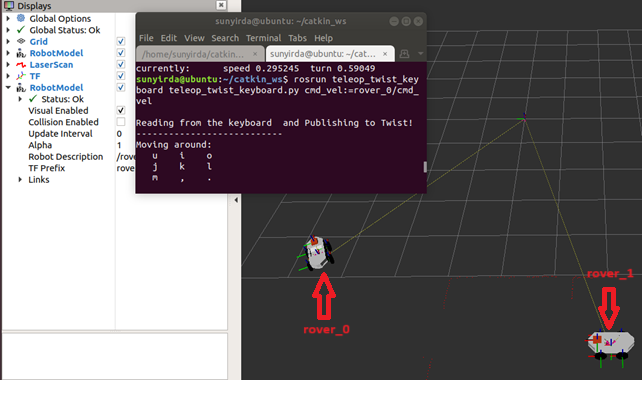


Figure 12 Teleoperación de rover\_0 - Visualizacion en RViz

## **Funcionamiento para multirobots (navegación autónoma)**

1. Ejecutar la simulación: roslaunch main multi\_rover\_house.launch navigation:=true

Hay más mundos donde para ejecutarlos son de la siguiente manera:

* roslaunch main multi\_rover empty.launch navigation:=true
* roslaunch main multi\_rover launch navigation:=true

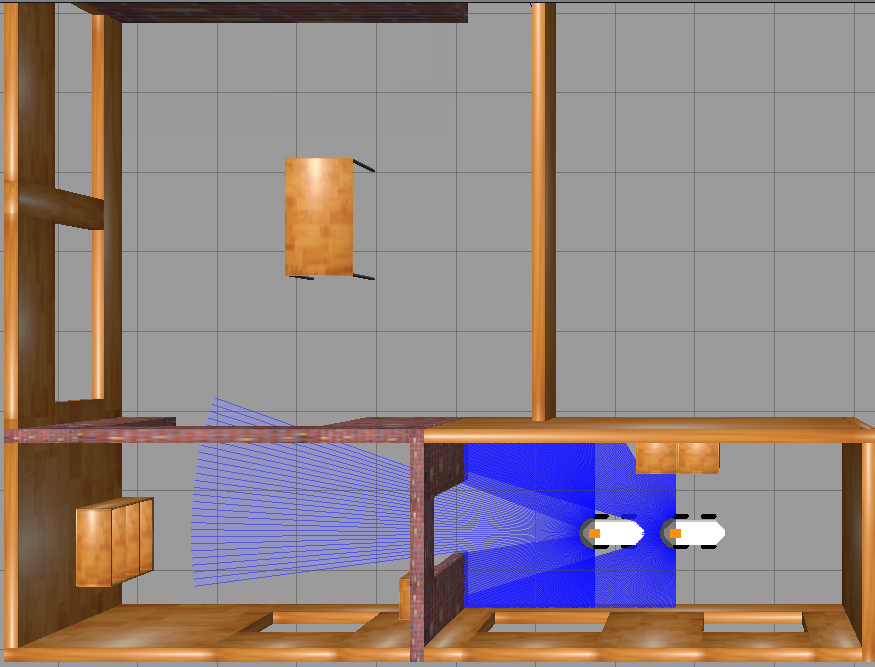


Figure 13 Simulación multirobot con navegación autónoma

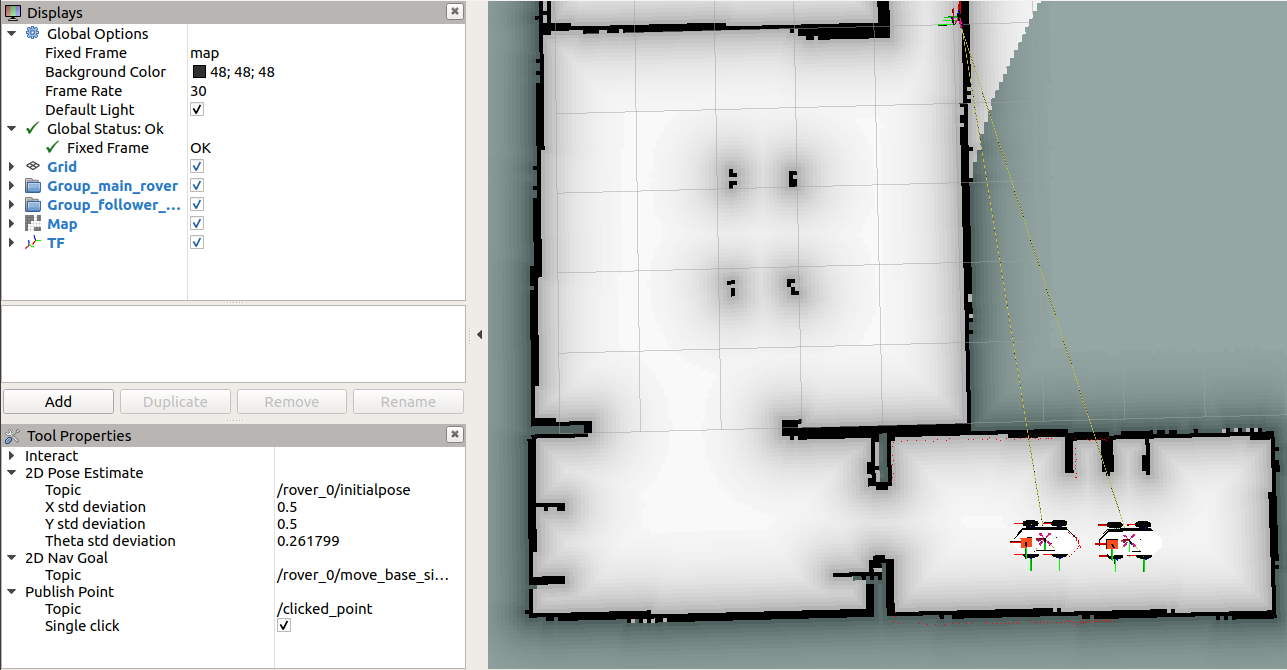


Figure 14 RViz multirobot con navegación autónoma

1. Enviar un punto de meta mediante RViz (ejemplo con el robot llamado rover\_0): Seleccionar la opción “2D Nav Goal” (flecha roja) y seleccionar el punto hacia donde se desee mover el robot (flecha azul).

Toda navegación autónoma mediante mapas requiere establecer una posicion inicial de movimiento, en este caso el sistema al ejecutarse se encarga de definir dicha posicion inicial.

Se puede enviar el punto de meta mediante los actions que ofrece ROS, el uso de RViz es una vista mas intuitiva para mostrar el correcto funcionamiento.

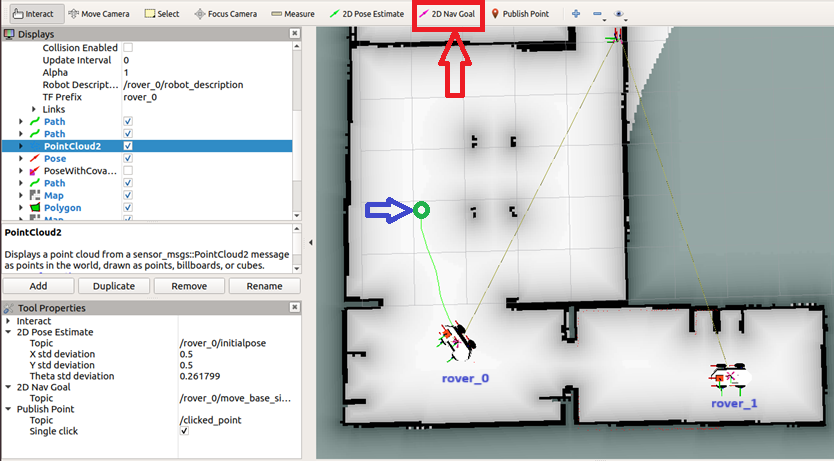


Figure 15 Enviar un punto de meta a rover\_0.

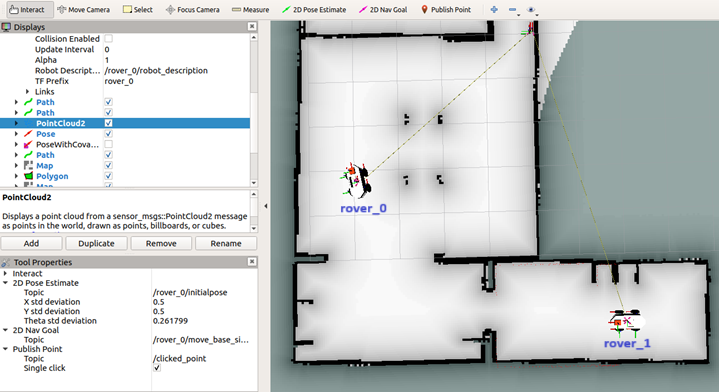


Figure 16 Robot rover\_0 en el punto de meta.

1. Enviar un punto de meta mediante RViz a cualquier robot.

Por defecto se tiene fijado en RViz para usar el boton de “2D Nav Goal” solo para rover\_0, pero en caso de que se requiera pasar un punto de meta a otro robot haciendo uso de RViz se tienen que realizar el siguiente proceso.

* 1. Cambiar el topic de la posicion inicial del robot (ejemplo con el robot llamado rover\_1): /NAME\_ROBOT/initialpose

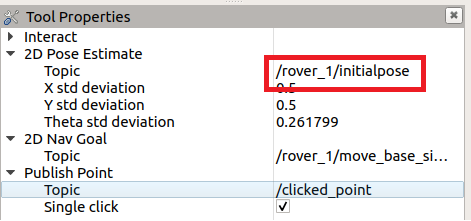


Figure 17 Actualizar el topic de 2D Pose Estimate.

* 1. Cambiar el topic de “2D Nav Goal” (ejemplo con el robot llamado rover\_1): /NAME\_ROBOT/ move\_base\_simple/goal

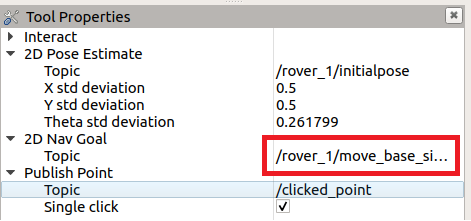


Figure 18 Actualizar el topic de 2D Nav Goal.

* 1. Pulsar el boton de “2D Nav Goal” tal como se explica en el paso 2

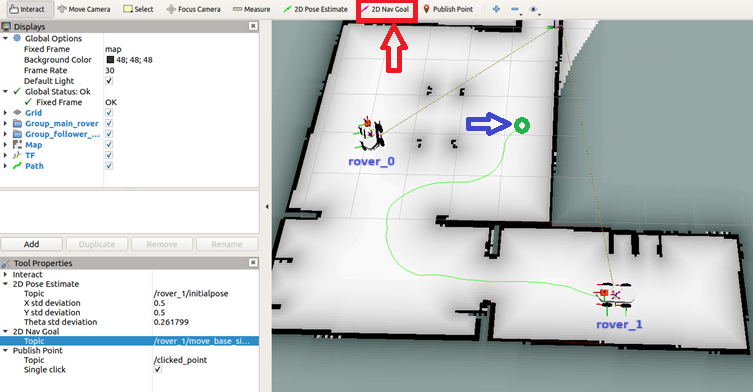


Figure 19 Enviar un punto de meta a rover\_1.

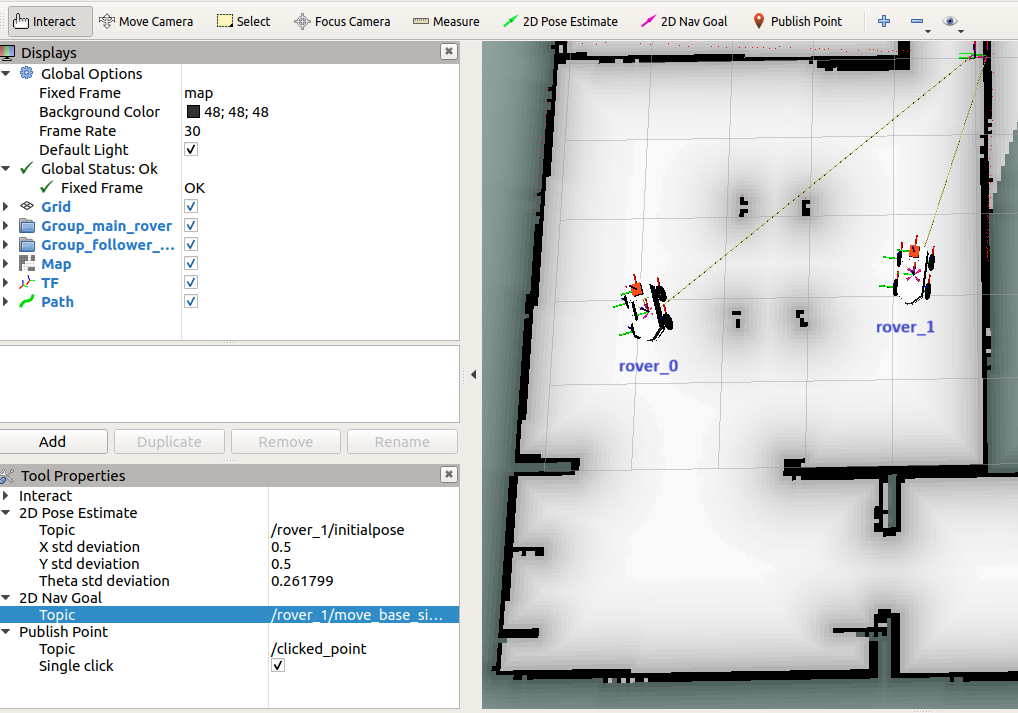


Figure 20 Robot rover\_1 en el punto de meta.

# **Añadir un mundo nuevo o más robots**

## 

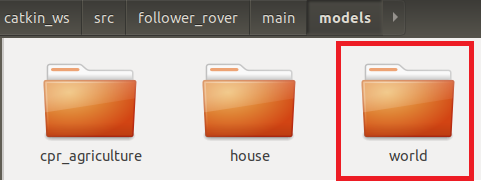
## **Añadir nuevo mundo**

Para fines prácticos se utilizará un mundo diseñado por otras personas, este procedimiento es para añadir un nuevo mundo, no para diseñar un nuevo mundo.

Todos estos pasos son dentro del paquete llamado “main”.

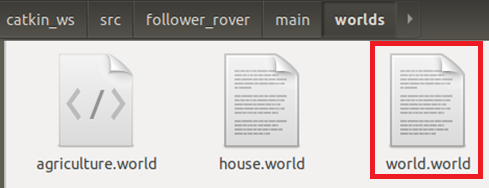
1. Dentro de la carpeta models crear una carpeta con el nombre de cómo se llamará nuestro modelo del mundo.

Aquí se colocará el diseño en SDF, el archivo de configuración del modelo y la carpeta de diseño.



1. Dentro de la carpeta worlds importar el archivo con extensión world que hemos diseñado o importado de personas externas.

**Recomendación:** El modelo del mundo y el mundo como tal se sugiere que contengan el mismo nombre.



1. En caso de que requiera un mundo para un robot simple dirigirse al paso 4, si desea un mundo para múltiples robots dirigirse al paso 5.
2. En la carpeta launch crear un nuevo launch: gazebo\_world.launch

Copiar y pegar el launch de gazebo\_house.launch y cambiar las siguientes líneas:

|  |  |
| --- | --- |
| Línea anterior | Línea nueva |
| <arg name="world\_name" value="$(find main)/worlds/house.world"/> | <arg name="world\_name" value="$(find main)/worlds/world.world"/> |
| <arg name="rover\_x\_pos" value="-3.0" /> | Posición X de origen. |
| <arg name="rover\_y\_pos" value=" 1.0" /> | Posición Y de origen. |
| <arg name="rover\_z\_pos" value=" 0.0" /> | Posición Z de origen. |
| <arg name="rover\_yaw" value=" 0.0" /> | Orientación YAW de origen. |
| <arg name="initial\_pose\_x" value="-3.0" /> | Posición X de origen. (Sirve para el topic de initialpose de la navegación autonoma) |
| <arg name="initial\_pose\_y" value=" 1.0" /> | Posición Y de origen. (Sirve para el topic de initialpose de la navegación autonoma) |

Tabla 1 Launch para mundo nuevo - 1 robot

1. En la carpeta launch crear un nuevo launch: multi\_robot\_world.launch

Copiar y pegar el launch de gazebo\_house.launch y cambiar las siguientes líneas:

|  |  |
| --- | --- |
| Línea anterior | Línea nueva |
| <arg name="world\_name" value="$(find main)/worlds/house.world"/> | <arg name="world\_name" value="$(find main)/worlds/world.world"/> |
| <arg name="first\_rover" default="rover\_0"/> | Nombre del primer ROVER. |
| <arg name="second\_rover" default="rover\_1"/> | Nombre del segundo ROVER. |
| <arg name="first\_rover\_x\_pos" default="-0.5"/> | Posición X de origen para el primer ROVER. |
| <arg name="first\_rover\_y\_pos" default=" 0.5"/> | Posición Y de origen para el primer ROVER. |
| <arg name="first\_rover\_z\_pos" default=" 0.0"/> | Posición Z de origen para el primer ROVER. |
| <arg name="first\_rover\_yaw" default=" 0.0"/> | Orientación YAW de origen para el primer ROVER. |
| <arg name="second\_rover\_x\_pos" default="-1.5"/> | Posición X de origen para el segundo ROVER. |
| <arg name="second\_rover\_y\_pos" default=" 0.5"/> | Posición Y de origen para el segundo ROVER. |
| <arg name="second\_rover\_z\_pos" default=" 0.0"/> | Posición Z de origen para el segundo ROVER. |
| <arg name="second\_rover\_yaw" default=" 0.0"/> | Orientación YAW de origen para el segundo ROVER. |

Tabla 2 Launch para mundo nuevo – multirobot

1. Dirigirse a su workspace y ejecutar: catkin\_make

## **Añadir un nuevo ROVER**

Todos estos pasos son dentro del paquete llamado “main”.

1. Dirigirse al launch llamado spawn\_multi\_rover.launch y agregar las siguientes líneas:

|  |  |
| --- | --- |
| Línea nueva | Explicación |
| <arg name="third\_rover" default="rover\_2"/> | Nombre del nuevo robot |
| <arg name="third\_rover\_x\_pos" default="-6.0"/> | Posición X de origen. |
| <arg name="third\_rover\_y\_pos" default=" 1.0"/> | Posición Y de origen. |
| <arg name="third\_rover\_z\_pos" default=" 0.0"/> | Posición Z de origen. |
| <arg name="third\_rover\_yaw" default=" 0.0"/> | Orientación YAW de origen. |

Tabla 3 Modificación de spawn multi ROVER.

1. En el mismo archivo archivo launch agregar un grupo nuevo:

Están etiquetas llamadas <group> permiten llevar una mejor estructura cuando se trabajan con multirobots y se requiere una correcta definición de las transformaciones, topics, services, etc.

<group ns = "$(arg third\_rover)">

<include file="$(find main)/launch/spawn\_rover.launch">

<arg name="name\_rover" value="$(arg third\_rover)" />

<arg name="rover\_x\_pos" value="$(arg third\_rover\_x\_pos)" />

<arg name="rover\_y\_pos" value="$(arg third\_rover\_y\_pos)" />

<arg name="rover\_z\_pos" value="$(arg third\_rover\_z\_pos)" />

<arg name="rover\_yaw" value="$(arg third\_rover\_yaw)" />

</include>

<param if="$(arg navigation)" name="$(arg third\_rover\_x\_pos)/amcl/initial\_pose\_x" value="$(arg third\_rover\_x\_pos)" />g

<param if="$(arg navigation)" name="$(arg third\_rover\_x\_pos)/amcl/initial\_pose\_y" value="$(arg third\_rover\_y\_pos)" />

<include if="$(arg navigation)" file="$(find lidar\_navigation)/launch/lidar\_navigation.launch">

<arg name="initial\_pose\_x" value="$(arg third\_rover\_x\_pos)" />

<arg name="initial\_pose\_y" value="$(arg third\_rover\_y\_pos)" />

<arg name="initial\_pose\_a" value="$(arg third\_rover\_yaw)"/>

<arg name="robot\_namespace" value="$(arg third\_rover)"/>

<arg name="map\_file" value="$(arg map\_file)"/>

</include>

<node unless="$(arg navigation)" pkg="tf" type="static\_transform\_publisher" name="static\_transform\_publisher" args="0 0 0 0 0 0 world $(arg third\_rover)/odom 1" />

</group>

1. Crear un launch parecido al multi\_rover\_house.launch, pero en lugar de colocar 2 robots hacerlo para colocar 3 robots.
2. Dirigirse a su workspace y ejecutar: catkin\_make