**Content**

[**Description** 1](#_Toc164426799)

[**General operations** 1](#_Toc164426800)

[**Operation for 1 robot (without autonomous navigation)** 1](#_Toc164426801)

[**Operation for 1 robot (autonomous navigation)** 3](#_Toc164426802)

[**Operation for multirobots (without autonomous navigation)** 5](#_Toc164426803)

[**Operation for multirobots (autonomous navigation)** 7](#_Toc164426804)

[**Add a new world or more robots** 10](#_Toc164426805)

[**Add new world** 10](#_Toc164426806)

[**Add a new ROVER** 12](#_Toc164426807)

# **Description**

The multirobot system developed at ROS Melodic has a stable and functional platform designed for autonomous or assisted exploration of environments that require navigation for different research tasks where the human hand does not intervene. All robots within the system are of the ROVER type where each mobile robot has autonomous navigation independently, which allows more special navigation for very large environments where a single robot may be of little use.

The system meets the feasibility of being able to easily incorporate more robots without the need to restructure all the developed code. In addition, it meets the detail that the system can support from 1 robot to N robots where the only limitation would be power. computational to incorporate N ROVERs.

# **General operations**

## **Operation for 1 robot (without autonomous navigation)**

1. Run the simulation: roslaunch main gazebo\_house.launch

There are more worlds where to execute them are as follows:

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

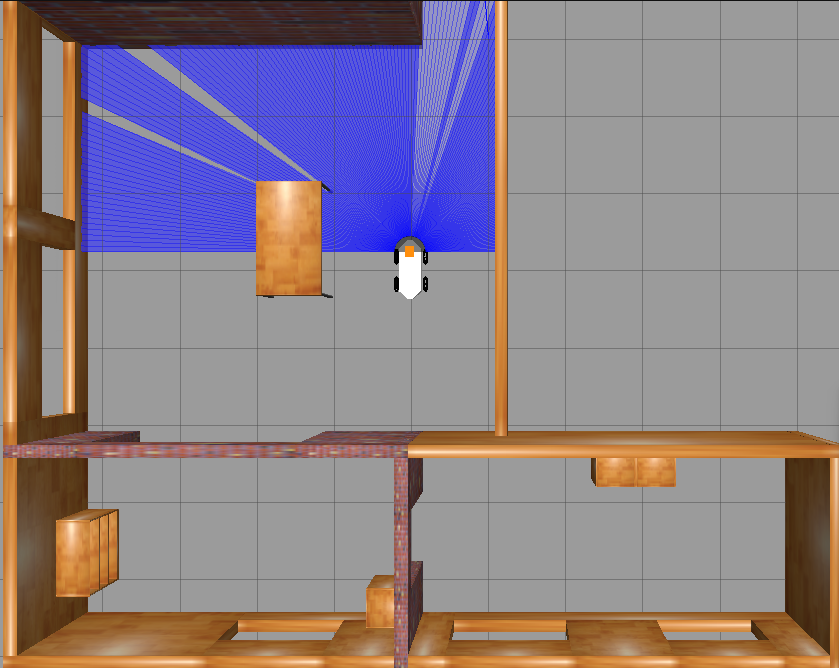


Figure 1 Simulation without autonomous navigation - 1 robot

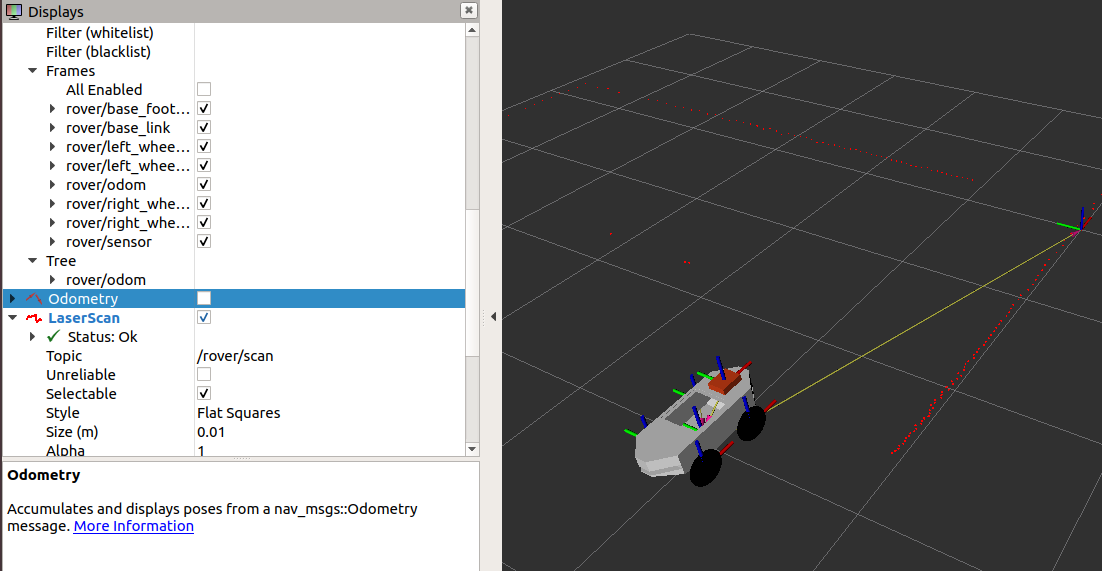


Figure 2 RViz without autonomous navigation - 1 robot

1. Execute teleoperation: rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=rover/cmd\_vel

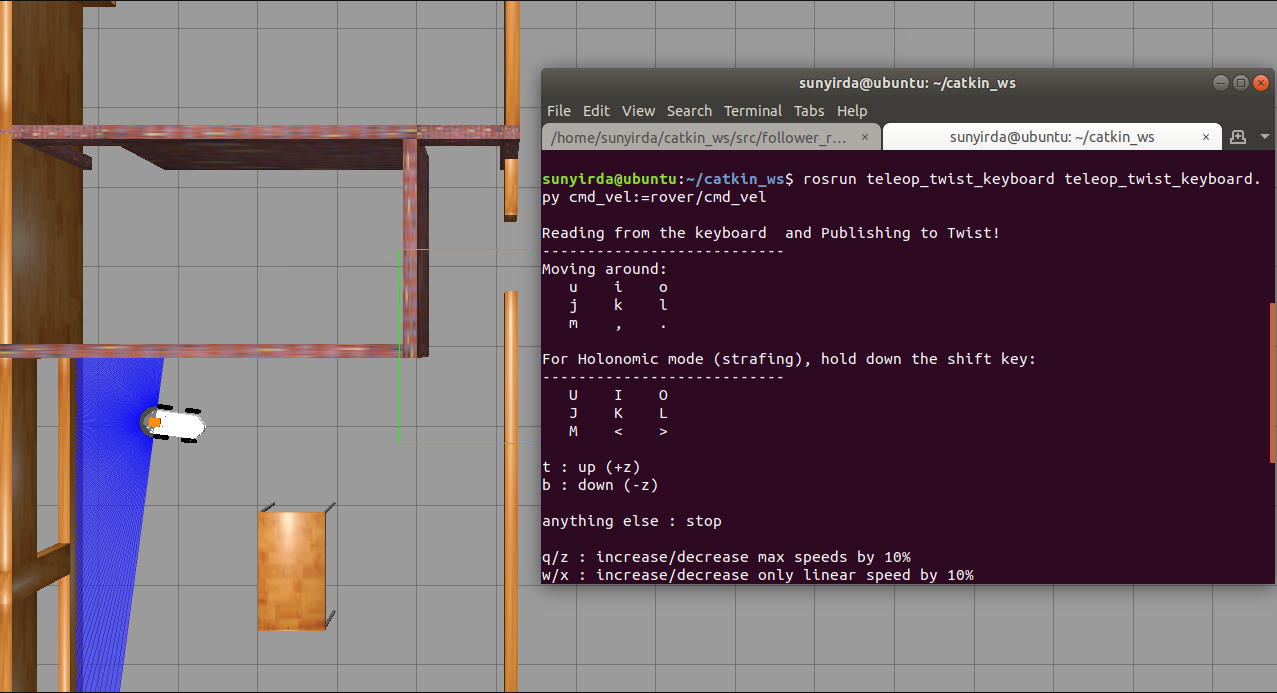


Figure 3 Teleoperación de ROVER

## **Operation for 1 robot (autonomous navigation)**

1. Run the simulation: roslaunch main gazebo\_house.launch navigation:=true

There are more worlds where to execute them are as follows:

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

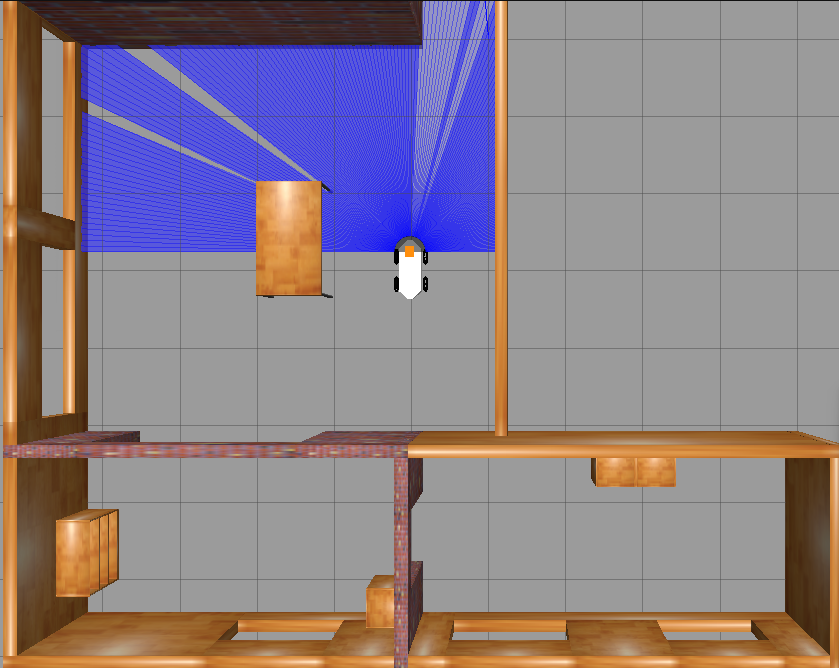


Figure 4 Simulation with autonomous navigation - 1 robot

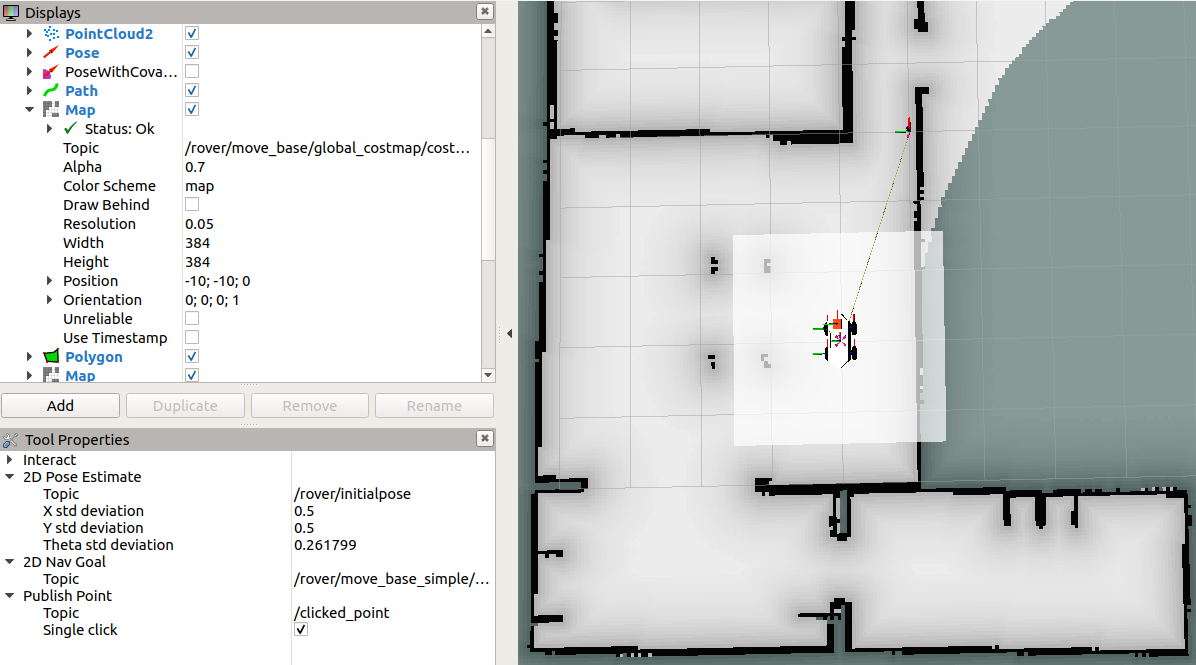


Figure 5 RViz with autonomous navigation - 1 robot

1. Send a goal point using RViz: Select the “2D Nav Goal” option (red arrow) and select the point where you want to move the robot (blue arrow).

All autonomous navigation using maps requires establishing an initial position of movement, in this case the system when executed is responsible for defining said initial position.

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.

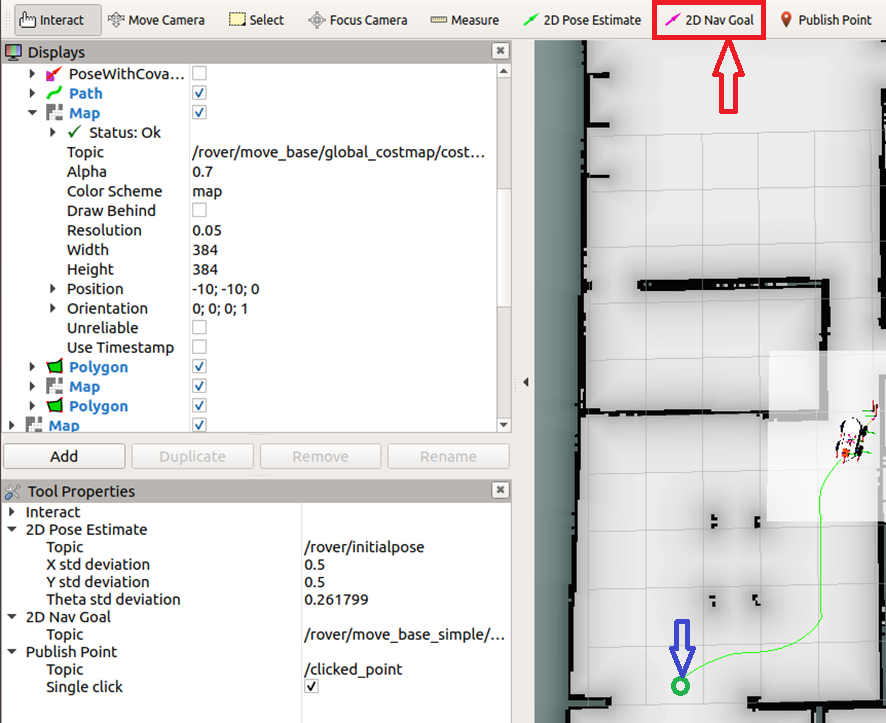


Figure 6 Send a goal point.

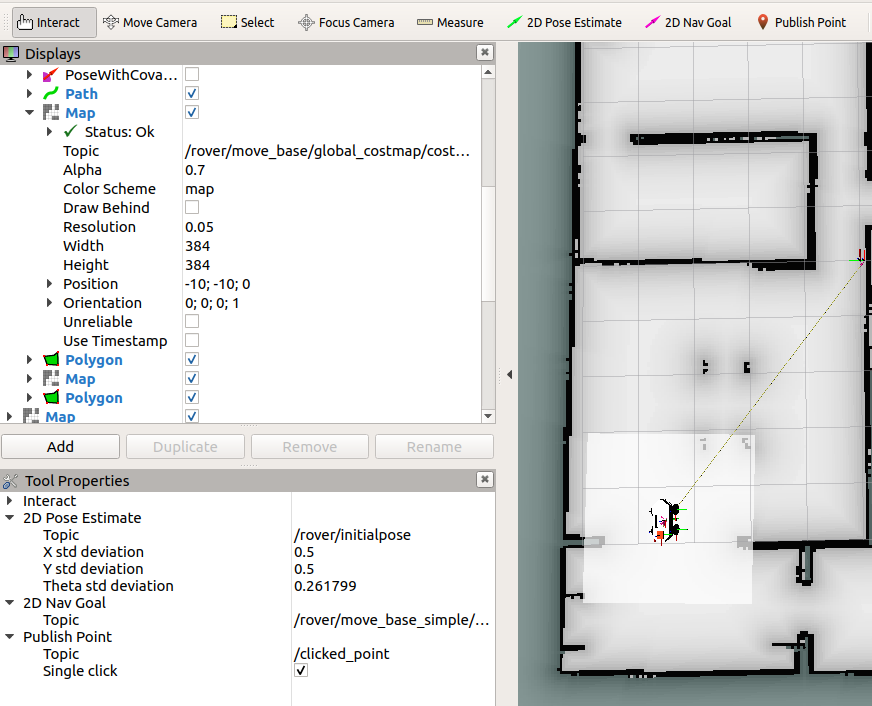


Figure 7 Robot at the finish point.

1. Execution of teleoperation (if necessary): rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=rover/cmd\_vel



Figure 8 Teleoperation with activation of autonomous navigation.

## **Operation for multirobots (without autonomous navigation)**

1. Run the simulation: roslaunch main multi\_rover\_house.launch

There are more worlds where to execute them are as follows:

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

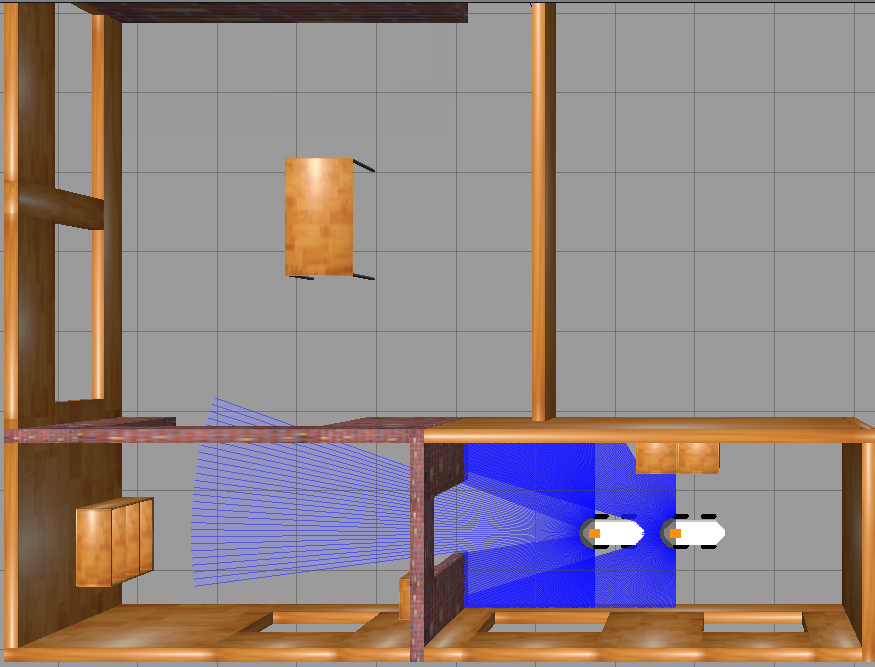


Figure 9 Multirobot simulation without autonomous navigation

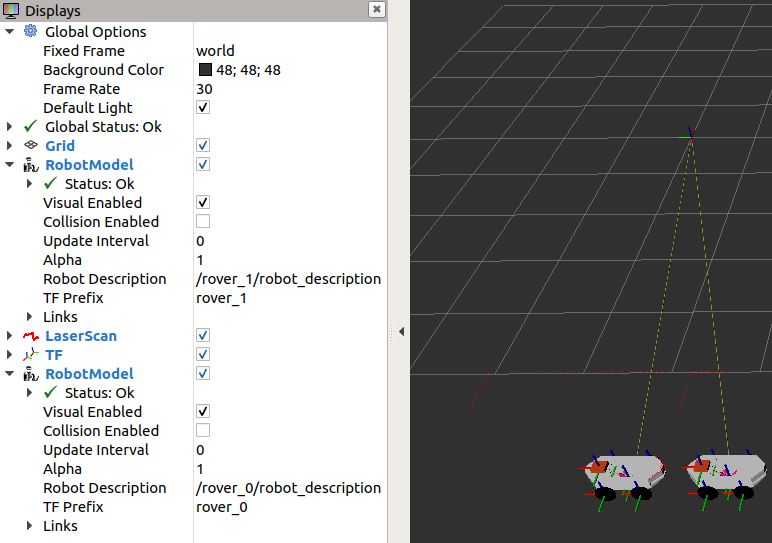


Figure 10 Multirobot RViz without autonomous navigation.

1. Execute the teleoperation (example with the robot named rover\_0): rosrun teleop\_twist\_keyboar teleop\_twist\_keyboard.py cmd\_vel:=NAME\_ROBOT/cmd\_vel

In the multirobot system each topic depends on the robot, so if we have N robots then we will have N topics cmd\_vel with the prefix of the name of its robot. In this case, by having two robots we will have the following topics:

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

**Important note:** The same happens for the other topics such as robot\_description, odom, etc.

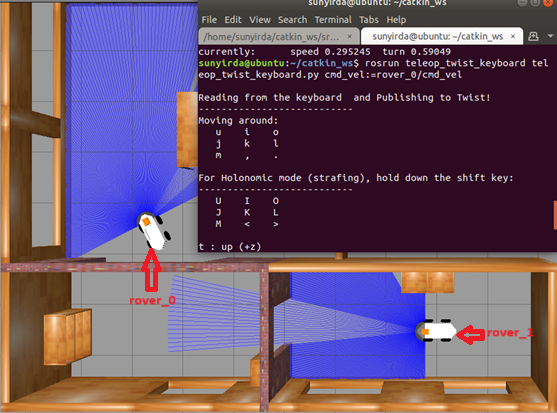


Figure 11 rover\_0 teleoperation.

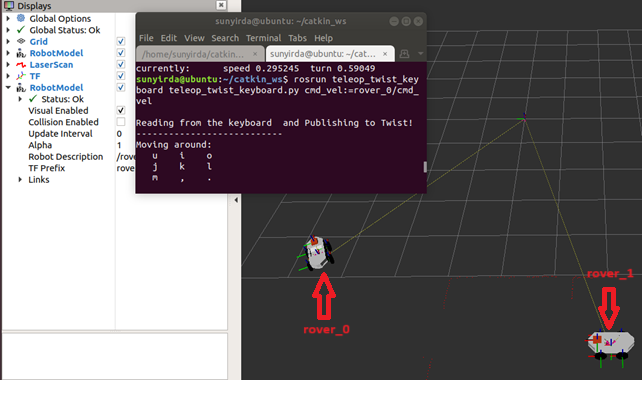


Figure 12 Rover\_0 teleoperation - Visualization in RViz.

## **Operation for multirobots (autonomous navigation)**

1. Run the simulation: roslaunch main multi\_rover\_house.launch navigation:=true

There are more worlds where to execute them are as follows:

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

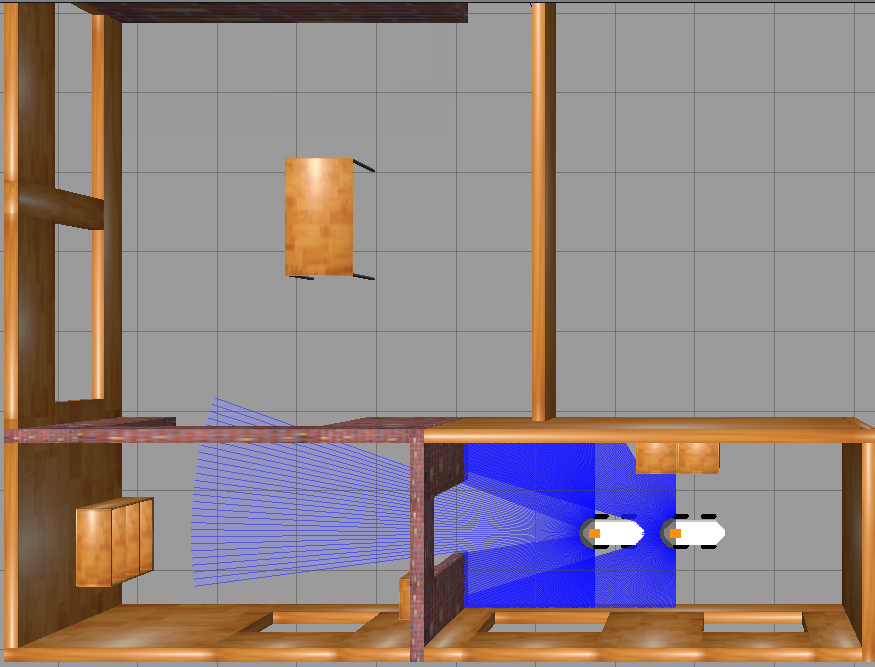


Figure 13 Multi-robot simulation with autonomous navigation.

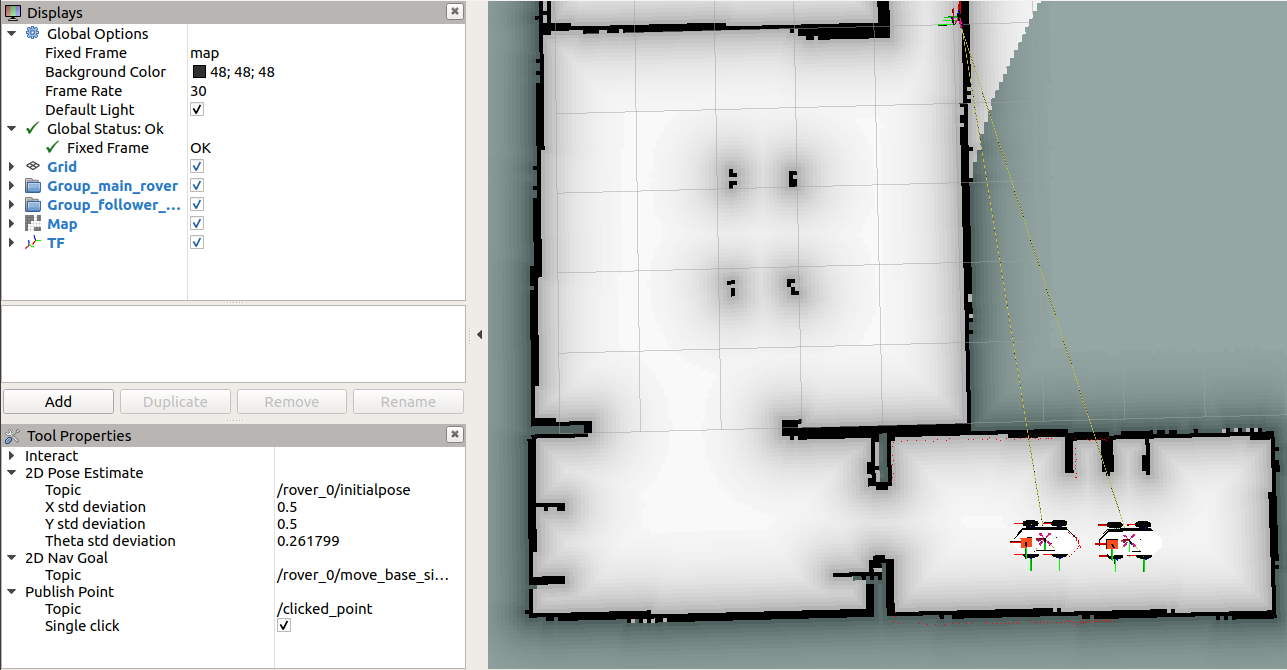


Figure 14 RViz multirobot with autonomous navigation.

1. Send a goal point using RViz (example with the robot called rover\_0): Select the “2D Nav Goal” option (red arrow) and select the point where you want to move the robot (blue arrow).

All autonomous navigation using maps requires establishing an initial position of movement, in this case the system when executed is responsible for defining said initial position.

The goal point can be sent using the actions offered by ROS, using RViz is a more intuitive view to show correct operation.

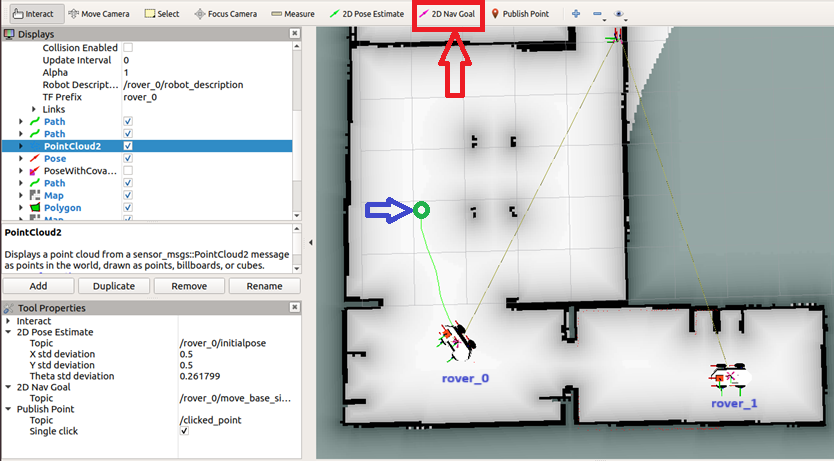


Figure 15 Send a goal point to rover\_0.

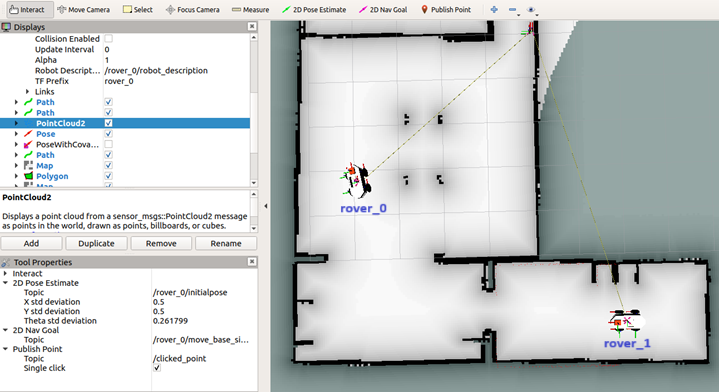


Figure 16 Robot rover\_0 at the finish point.

1. Send a goal point using RViz to any robot.

By default, RViz is set to use the “2D Nav Goal” button only for rover\_0, but if it is required to pass a goal point to another robot using RViz, the following process must be carried out.

* 1. Change the topic of the robot's initial position (example with the robot named rover\_1): /NAME\_ROBOT/initialpose

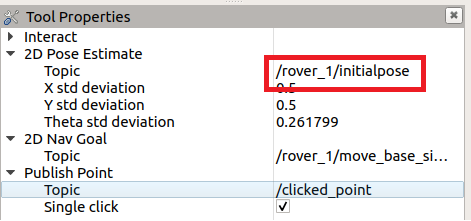


Figure 17 Update the 2D Pose Estimate topic.

* 1. Change the topic of “2D Nav Goal” (example with the robot named rover\_1): /NAME\_ROBOT/ move\_base\_simple/goal

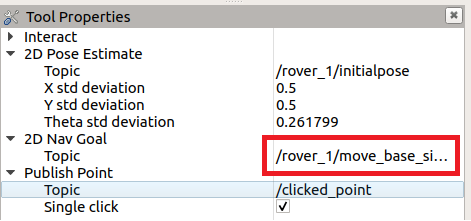


Figure 18 Update the 2D Nav Goal topic.

* 1. Press the “2D Nav Goal” button as explained in step 2

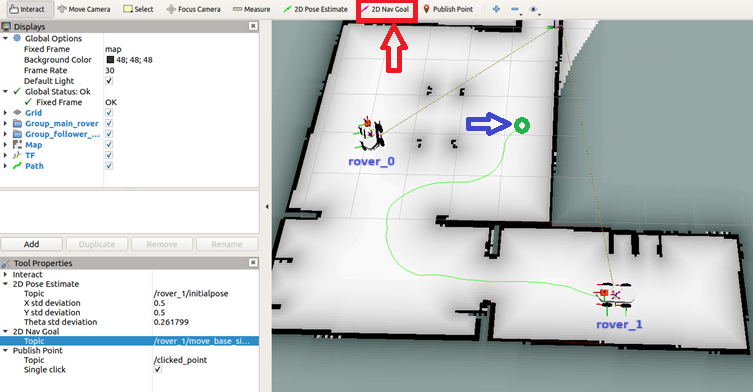


Figure 19 Send a goal point to rover\_1.

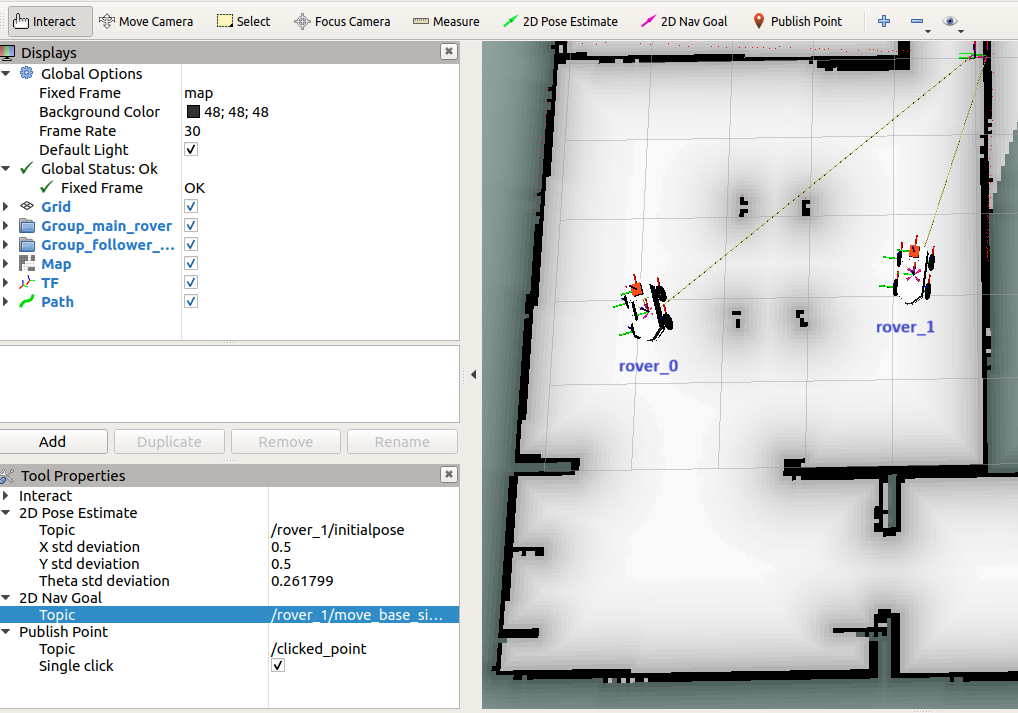


Figure 20 Robot rover\_1 at the finish point.

# **Add a new world or more robots**

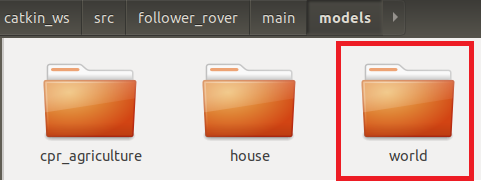
## **Add new world**

For practical purposes a world designed by other people will be used, this procedure is to add a new world, not to design a new world.

All these steps are within the package called “main”.

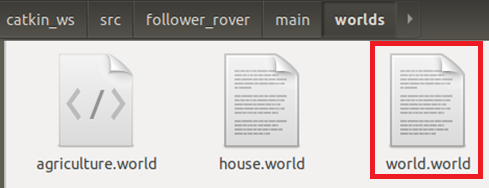
1. Inside the model’s folder, create a folder with the name of what our world model will be called.

Here the design will be placed in SDF, the model configuration file and the design folder.



1. Inside the worlds folder, import the file with the world extension that we have designed or imported from external people.

**Recommendation:** The world model and the world as such are suggested to contain the same name.



1. If you require a world for a single robot, go to step 4, if you want a world for multiple robots, go to step 5.
2. In the launch folder create a new launch: gazebo\_world.launch

Copy and paste the launch of gazebo\_house.launch and change the following lines:

|  |  |
| --- | --- |
| Previous line | New line |
| <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" /> | Origin X position. |
| <arg name="rover\_y\_pos" value=" 1.0" /> | Origin Y position. |
| <arg name="rover\_z\_pos" value=" 0.0" /> | Origin Z position. |
| <arg name="rover\_yaw" value=" 0.0" /> | Origin YAW orientation. |
| <arg name="initial\_pose\_x" value="-3.0" /> | Origin X position. (It is used for the initialpose topic of autonomous navigation) |
| <arg name="initial\_pose\_y" value=" 1.0" /> | Origin Y position. (It is used for the initialpose topic of autonomous navigation) |

Table 1 Launch for new world - 1 robot

1. In the launch folder create a new launch: multi\_robot\_world.launch

Copy and paste the launch of gazebo\_house.launch and change the following lines:

|  |  |
| --- | --- |
| Previous line | New line |
| <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"/> | Name of the first ROVER. |
| <arg name="second\_rover" default="rover\_1"/> | Name of the second ROVER. |
| <arg name="first\_rover\_x\_pos" default="-0.5"/> | Origin X position for the first ROVER. |
| <arg name="first\_rover\_y\_pos" default=" 0.5"/> | Origin Y position for the first ROVER. |
| <arg name="first\_rover\_z\_pos" default=" 0.0"/> | Origin Z position for the first ROVER. |
| <arg name="first\_rover\_yaw" default=" 0.0"/> | Origin YAW orientation for the first ROVER |
| <arg name="second\_rover\_x\_pos" default="-1.5"/> | Origin X position for the second ROVER. |
| <arg name="second\_rover\_y\_pos" default=" 0.5"/> | Origin Y position for the second ROVER. |
| <arg name="second\_rover\_z\_pos" default=" 0.0"/> | Origin Z position for the second ROVER. |
| <arg name="second\_rover\_yaw" default=" 0.0"/> | Origin YAW orientation for the second ROVER. |

Table 2 Launch for new world – multirobot

1. Go to your workspace and execute: catkin\_make

## **Add a new ROVER**

All these steps are within the package called “main”.

1. Go to the launch called spawn\_multi\_rover.launch and add the following lines:

|  |  |
| --- | --- |
| Línea nueva | Explicación |
| <arg name="third\_rover" default="rover\_2"/> | Name of the new robot |
| <arg name="third\_rover\_x\_pos" default="-6.0"/> | Origin X position. |
| <arg name="third\_rover\_y\_pos" default=" 1.0"/> | Origin Y position. |
| <arg name="third\_rover\_z\_pos" default=" 0.0"/> | Origin Z position. |
| <arg name="third\_rover\_yaw" default=" 0.0"/> | Origin YAW orientation. |

Table 3 Multi ROVER spawn modification.

1. In the same launch file add a new group:

These tags called <group> allow for a better structure when working with multirobots and a correct definition of transformations, topics, services, etc. is required.

<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. Create a launch similar to multi\_rover\_house.launch, but instead of placing 2 robots, do it to place 3 robots.
2. Go to your workspace and execute: catkin\_make