**Content**

[**Description** 1](#_Toc164427892)

[**Important information** 1](#_Toc164427893)

[**Characteristics** 1](#_Toc164427894)

[**Work cycle** 2](#_Toc164427895)

[**Operations for simulation execution** 2](#_Toc164427896)

[**Operation for 1 follower ROVER and 1 main ROVER (teleoperation control)** 2](#_Toc164427897)

[**Operation for 1 follower ROVER and 1 main ROVER (autonomous navigation)** 4](#_Toc164427898)

[**Operation for 2 follower ROVERs and 1 main ROVER (teleoperation control)** 6](#_Toc164427899)

[**Operation for 2 ROVER followers and 1 main ROVER (autonomous navigation)** 8](#_Toc164427900)

[**Operation for real life execution** 9](#_Toc164427901)

# **Description**

The project developed in ROS Melodic consists of the development of an algorithm that allows a ROVER to follow another ROVER that is in front. The algorithm does not focus on a particular robot, that is, the algorithm can be run on any ROVER, thus creating a chain of robot followers.

# **Important information**

* Subscriber and Publisher is a type of communication in ROS.
* The cmd\_vel topic receives linear and angular velocity.
* The LIDAR topic provides a list of 180 data, 1 data per angle.
* It only subscribes to the LIDAR topic and publishes to the cmd\_vel topic.
* It does not use Machine Learning or any computer vision cameras.
* The algorithm separates the data into 5 regions, it always tries to have the main ROVER be in the front region (region 3).

Diagrama

Descripción generada automáticamente

Figure 1 Region separations.

# **Characteristics**

1. It is independent whether the main ROVER executes autonomous navigation or navigation through teleoperation.
2. It can be run on N ROVER at the same time.
3. Implements speed control by distance to the main ROVER.
4. The follower ROVER always tries to position itself behind the main ROVER.
5. The follower ROVER is located at a maximum distance of 0.5 meters from the main ROVER.

# **Work cycle**

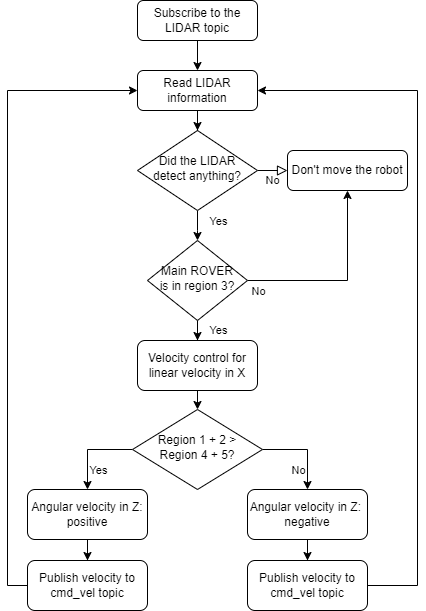


Figure 2 Work cycle.

# **Operations for simulation execution**

## **Operation for 1 follower ROVER and 1 main ROVER (teleoperation control)**

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

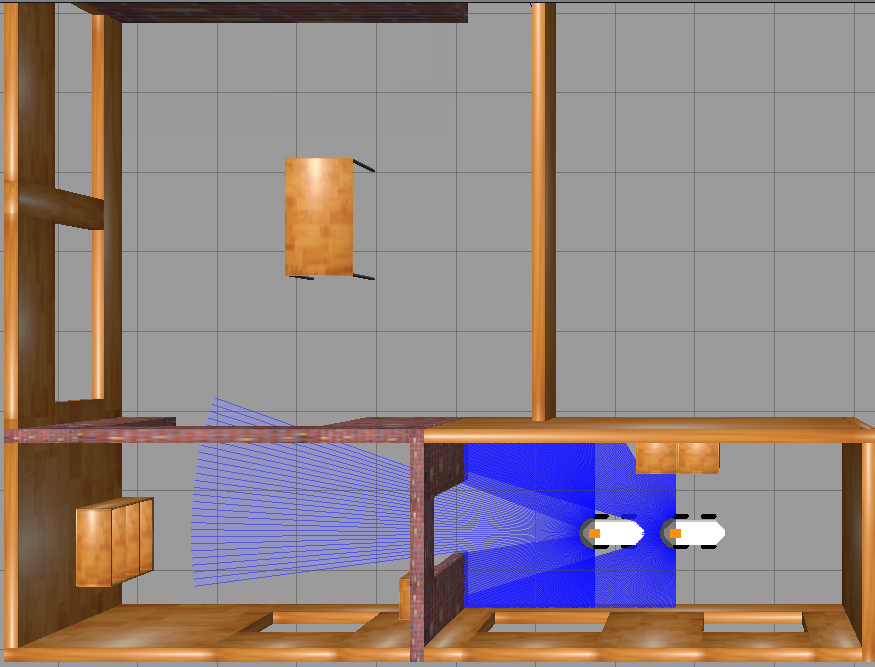


Figure 3 Multirobot simulation without autonomous navigation.

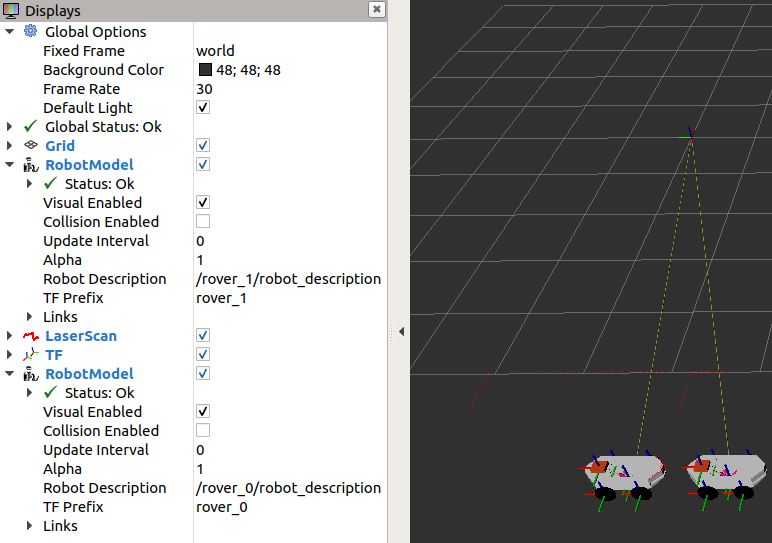


Figure 4 Multirobot RViz without autonomous navigation.

1. Run the follower algorithm (example with the robot named rover\_1): rosrun follower\_rover follower\_2.py NOMBRE\_ROVER

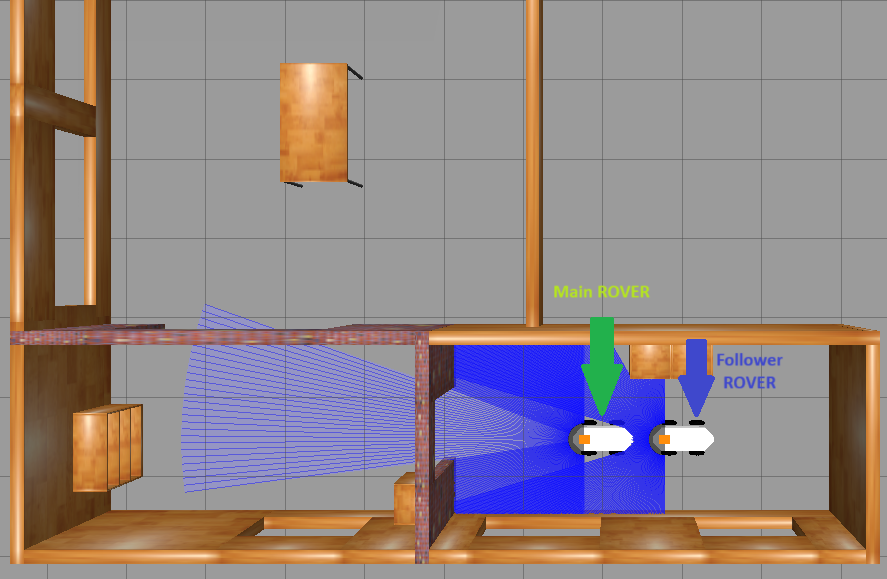


Figure 5 Follower ROVER

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

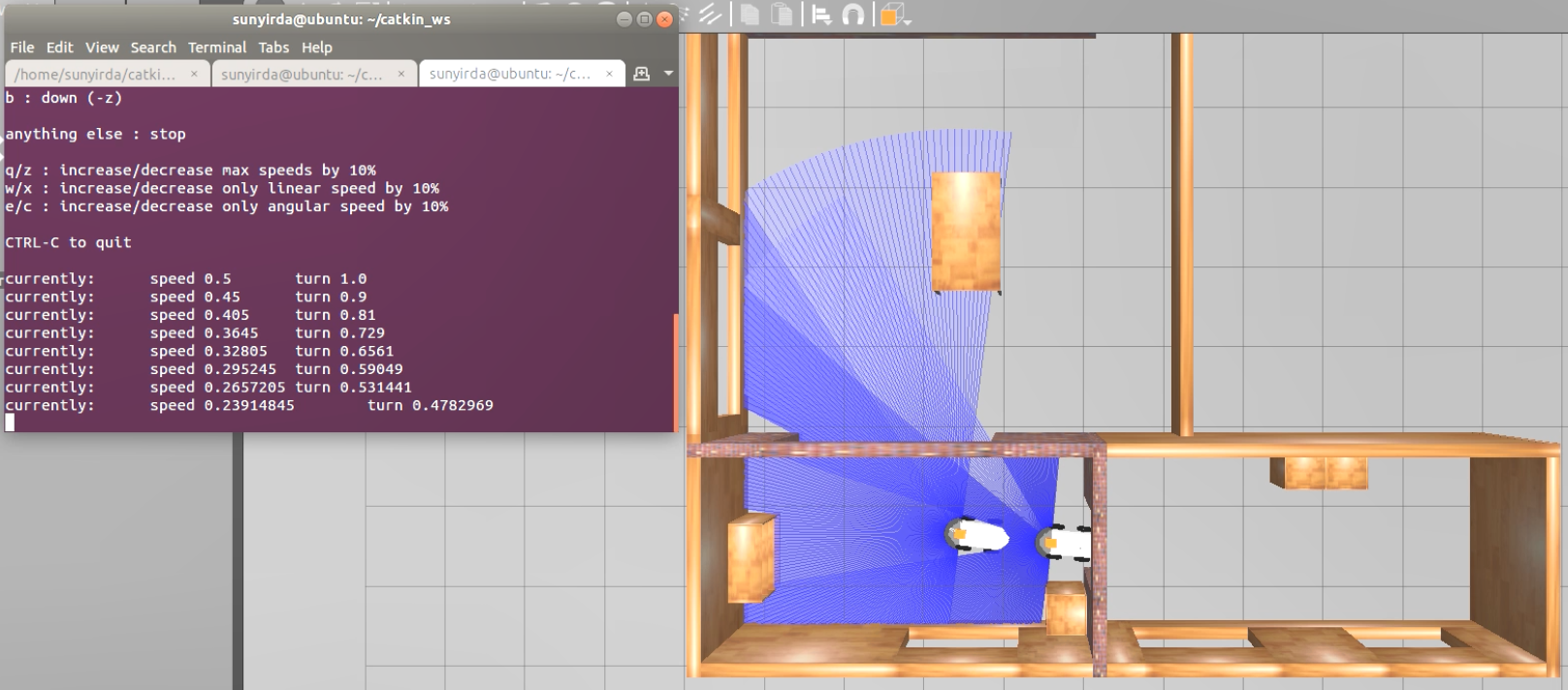


Figure 6 rover\_0 teleoperation

## **Operation for 1 follower ROVER and 1 main ROVER (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\_world.launch navigation:=true

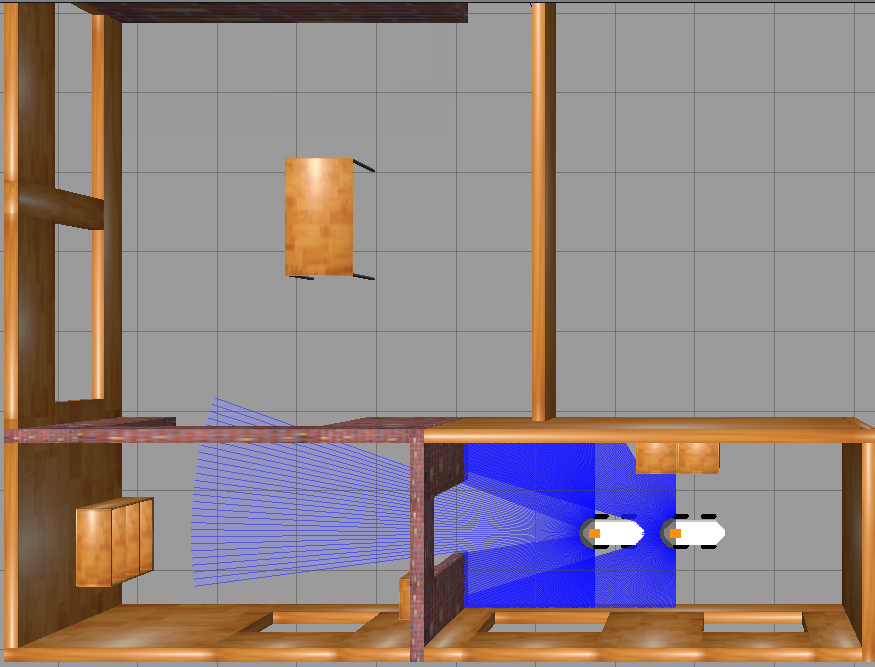


Figure 7 Multi-robot simulation with autonomous navigation.

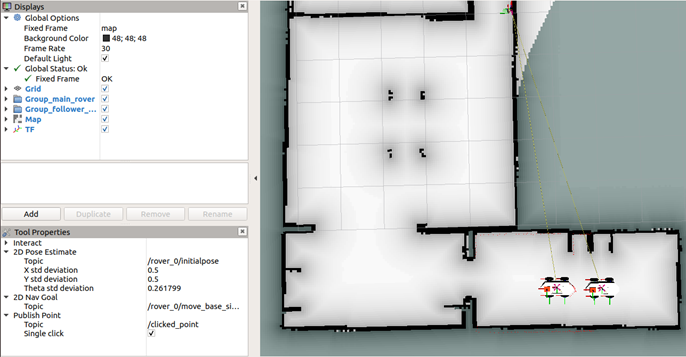


Figure 8 RViz multirobot with autonomous navigation.

1. Run the follower algorithm (example with the robot named rover\_1): rosrun follower\_rover follower\_2.py NOMBRE\_ROVER

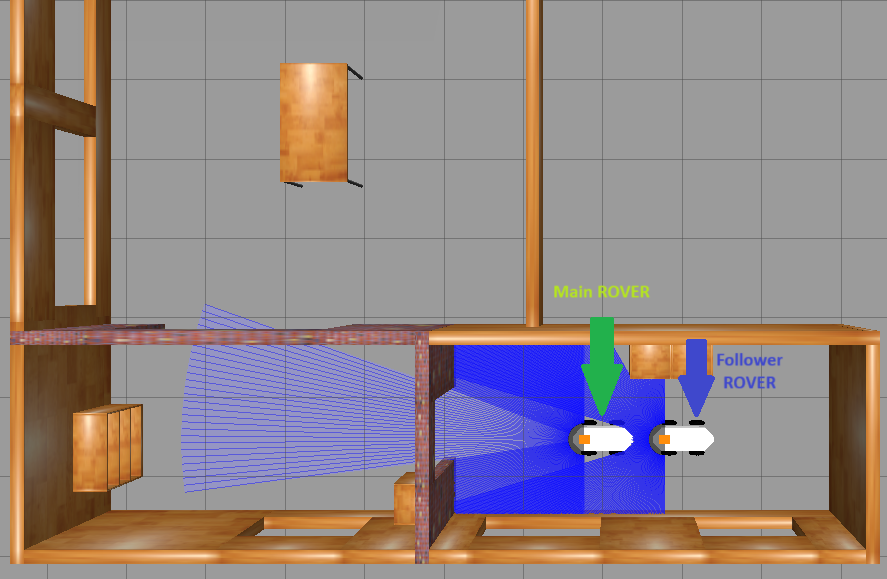


Figure 9 Follower ROVER.

## **Operation for 2 follower ROVERs and 1 main ROVER (teleoperation control)**

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

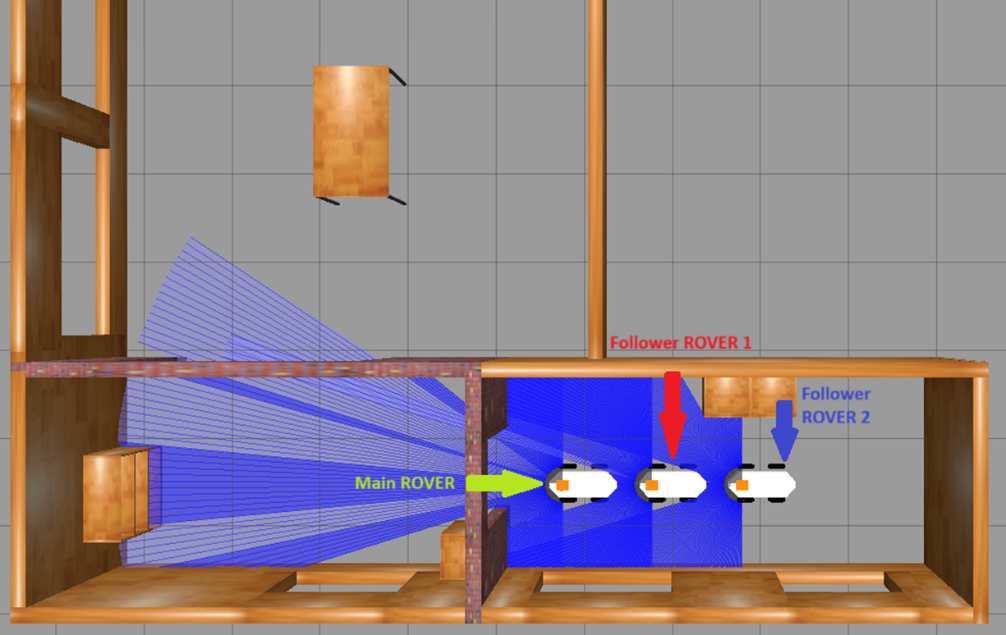


Figure 10 A main rover without autonomous navigation and 2 followers.

1. Run the follower algorithm for rover\_1: rosrun follower\_rover follower\_2.py rover\_1

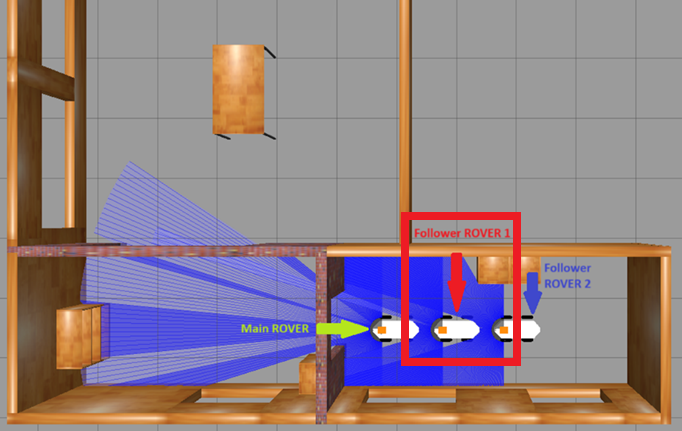


Figure 11 Follower ROVER 1.

1. Run the follower algorithm for rover\_2: rosrun follower\_rover follower\_2.py rover\_2

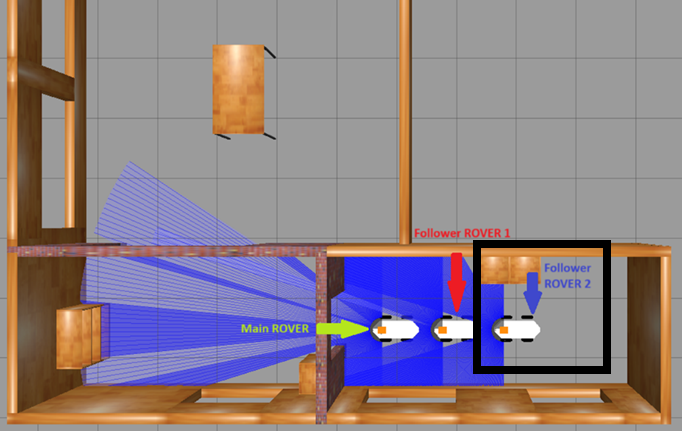


Figure 12 Follower ROVER 2.

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

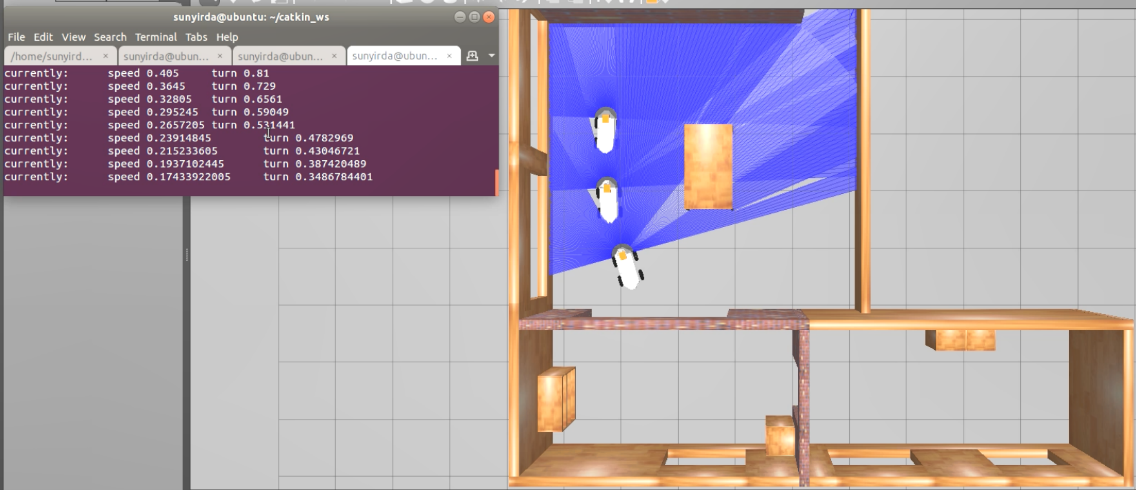


Figure 13 rover\_0 teleoperation.

## **Operation for 2 ROVER followers and 1 main ROVER (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\_world.launch navigation:=true

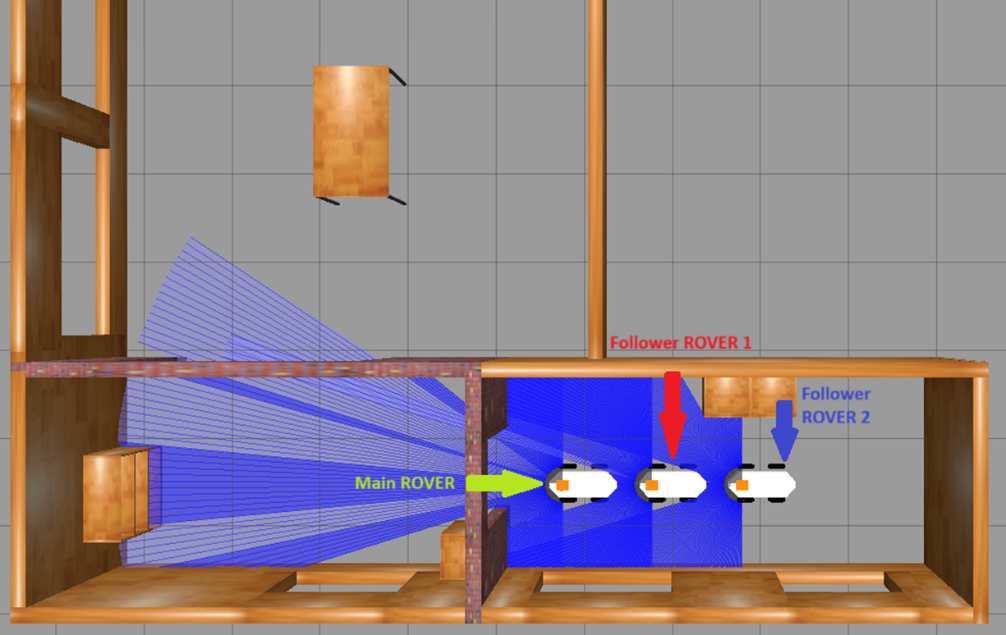


Figure 14 A main rover with autonomous navigation and 2 followers.

1. 2. Run the follower algorithm for rover\_1: rosrun follower\_rover follower\_2.py rover\_1

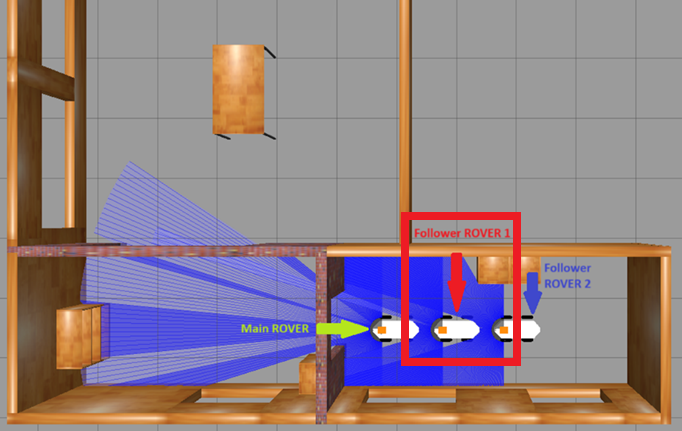


Figure 15 Follower ROVER 1.

1. Run the follower algorithm for rover\_2: rosrun follower\_rover follower\_2.py rover\_2

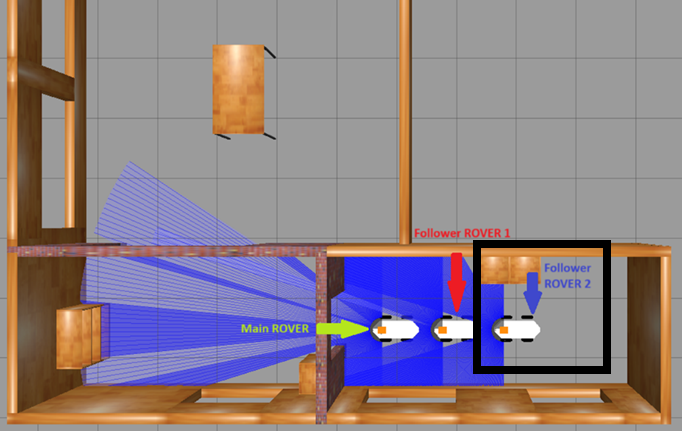


Figure 16 Follower ROVER 2.

# **Operation for real life execution**

1. Run everything necessary for the main ROVER. The algorithm is independent of what the main ROVER executes and how it executes it.
2. Run everything necessary for the ROVER follower.
3. Run the follower algorithm for the corresponding ROVER: rosrun follower\_rover follower\_2.py ROVER\_NAME
4. If you want to run the code on more than 1 robot, repeat steps 2 and 3 for the other followers.