Contenido

[**Descripción** 1](#_Toc164425385)

[**Información Importante** 1](#_Toc164425386)

[**Características** 2](#_Toc164425387)

[**Ciclo de trabajo** 2](#_Toc164425388)

[**Funcionamientos para ejecución en simulación** 2](#_Toc164425389)

[**Funcionamiento para 1 ROVER seguidor y 1 ROVER principal (control por teleoperacion)** 2](#_Toc164425390)

[**Funcionamiento para 1 ROVER seguidor y 1 ROVER principal (navegación autónoma)** 4](#_Toc164425391)

[**Funcionamiento para 2 ROVER seguidores y 1 ROVER principal (control por teleoperacion)** 6](#_Toc164425392)

[**Funcionamiento para 2 ROVER seguidores y 1 ROVER principal (navegación autónoma)** 8](#_Toc164425393)

[**Funcionamiento para ejecución en vida real** 9](#_Toc164425394)

# **Descripción**

El proyecto desarrollado en ROS Melodic consiste en el desarrollo de un algoritmo que permita a un ROVER seguir a otro ROVER que se encuentre delante. El algoritmo no se centra en un robot en particular, es decir, el algoritmo se puede ejecutar en cualquier ROVER, creando así una cadena de seguidores de robots.

# **Información Importante**

* Suscriptor y Publicador es un tipo de comunicación en ROS.
* El topic cmd\_vel recibe velocidad lineal y angular.
* El topic LIDAR proporciona una lista de 180 datos, 1 dato por ángulo.
* Sólo se suscribe al topic LIDAR y publica en el topic cmd\_vel.
* No utiliza Machine Learning ni ninguna cámara con visión por computadora.
* El algoritmo separa los datos en 5 regiones, siempre intenta que el ROVER principal esté en la región frontal (región 3).

Diagrama

Descripción generada automáticamente

Figure 1 Separaciones de regiones.

# **Características**

1. Es independiente si el ROVER principal ejecuta navegación autónoma o navegación mediante teleoperación.
2. Se puede ejecutar en N ROVER al mismo tiempo.
3. Implementa control de velocidad por distancia al ROVER principal.
4. El ROVER seguidor siempre intenta posicionarse detrás del ROVER principal.
5. El ROVER seguidor se ubica a una distancia máxima de 0,5 metros del ROVER principal.

# **Ciclo de trabajo**

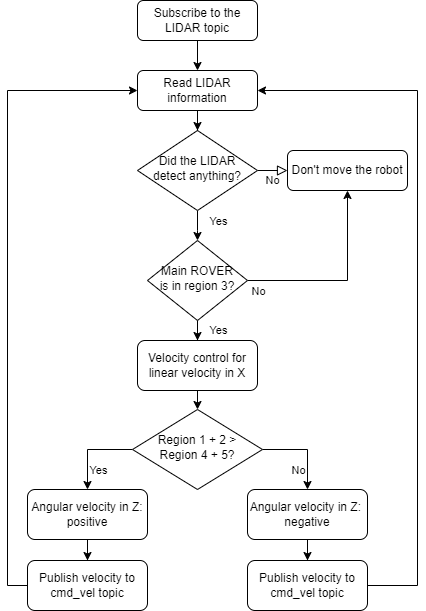


Figure 2 Ciclo de trabajo.

# **Funcionamientos para ejecución en simulación**

## **Funcionamiento para 1 ROVER seguidor y 1 ROVER principal (control por teleoperacion)**

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

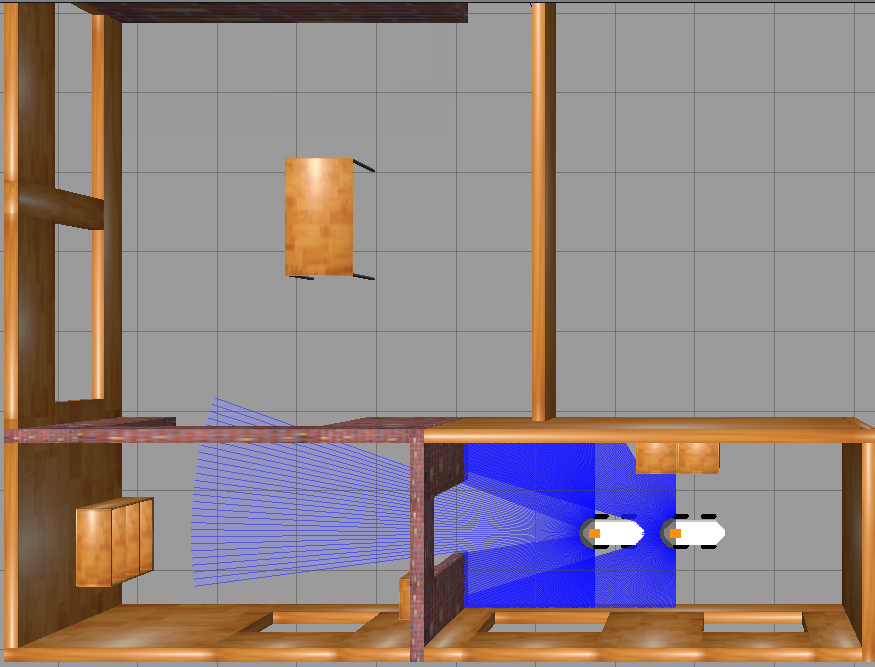


Figure 3 Simulación multirobot sin navegación autónoma

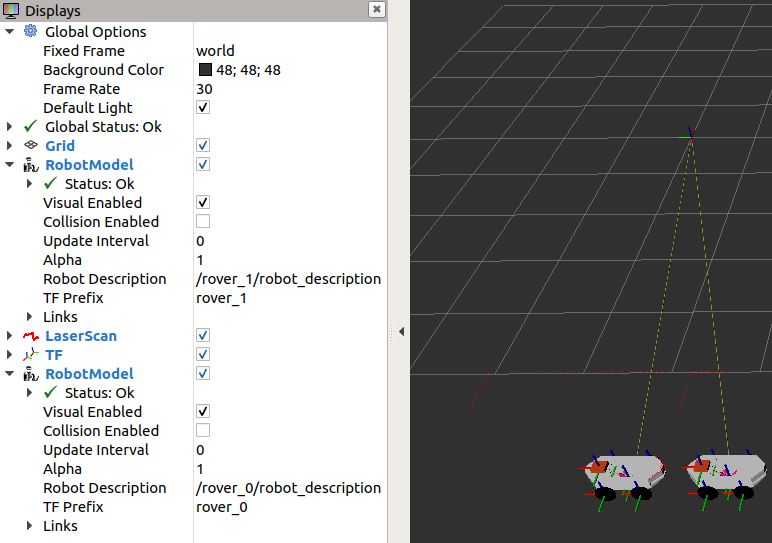


Figure 4 RViz multirobot sin navegación autónoma

1. Ejecutar el algoritmo seguidor (ejemplo con el robot llamado rover\_1): rosrun follower\_rover follower\_2.py NOMBRE\_ROVER

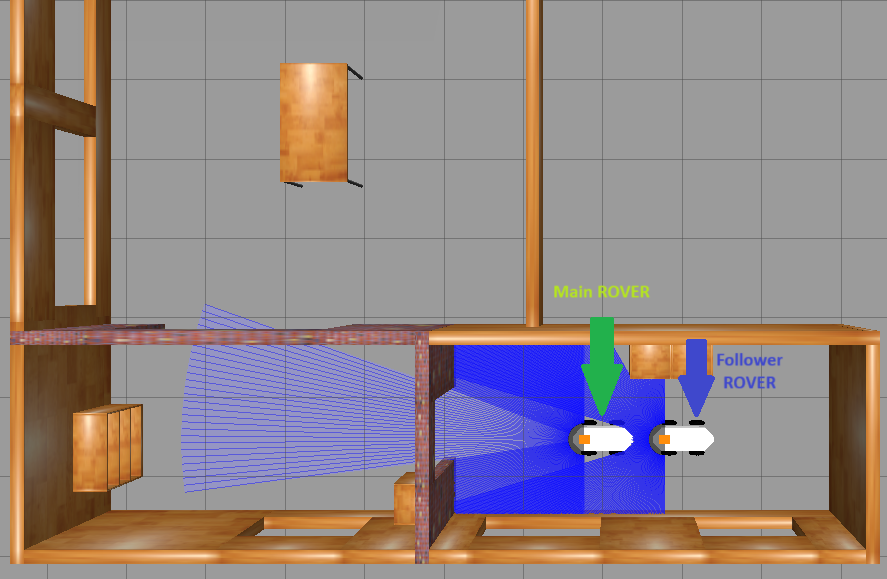


Figure 5 Follower ROVER

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

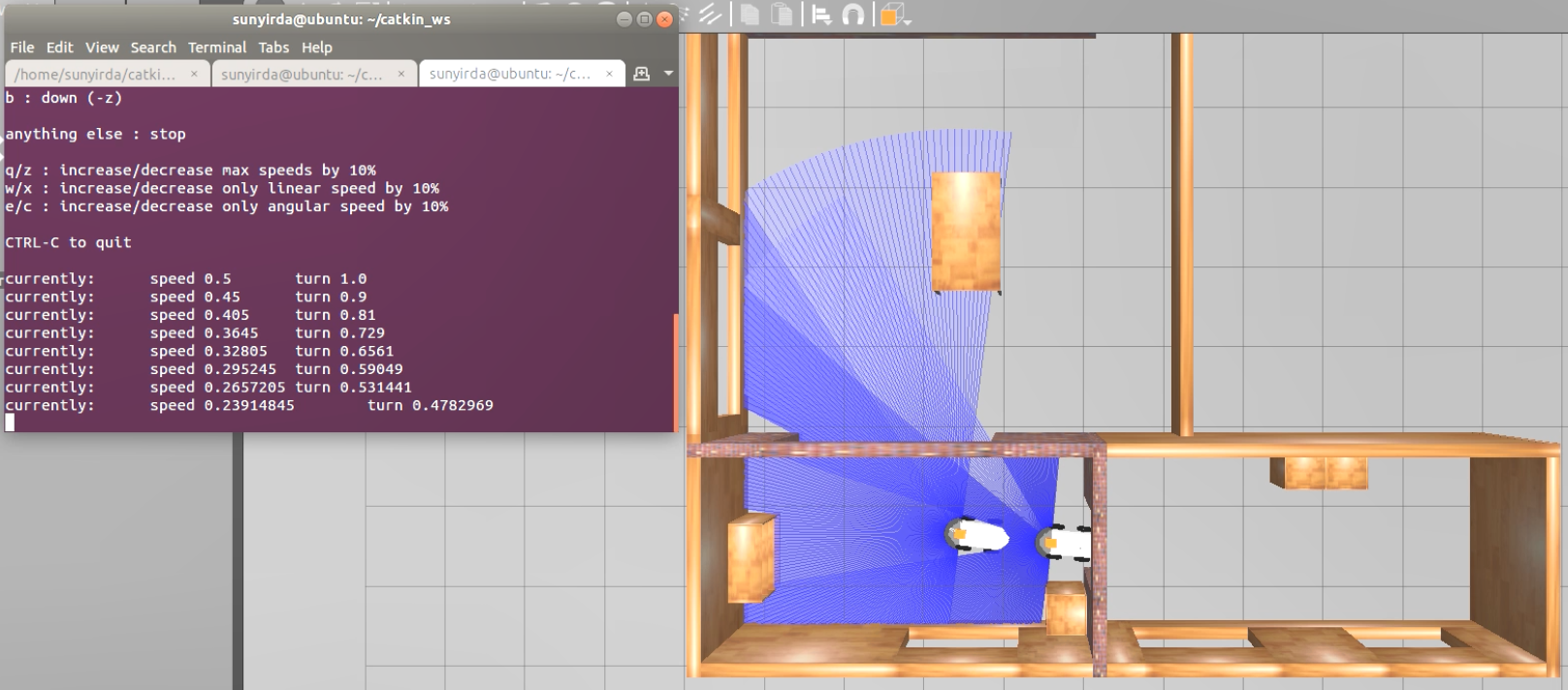


Figure 6 Teleoperación de rover\_0

## **Funcionamiento para 1 ROVER seguidor y 1 ROVER principal (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\_world.launch navigation:=true

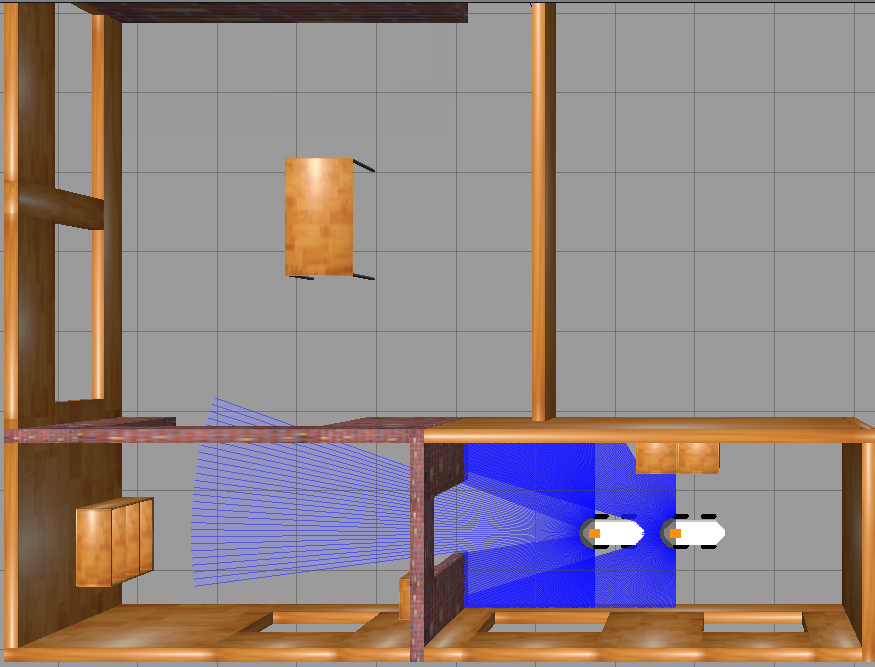


Figure 7 Simulación multirobot con navegación autónoma

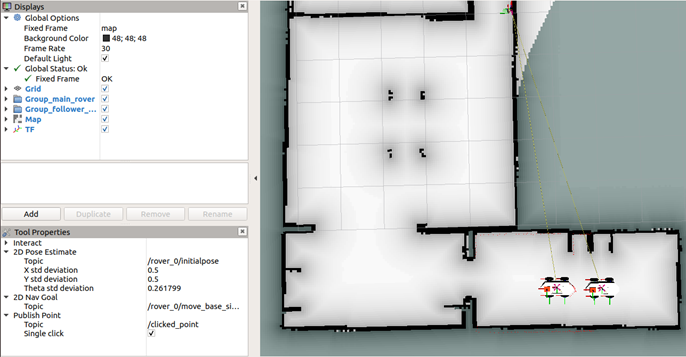


Figure 8 RViz multirobot con navegación autónoma

1. Ejecutar el algoritmo seguidor (ejemplo con el robot llamado rover\_1): rosrun follower\_rover follower\_2.py NOMBRE\_ROVER

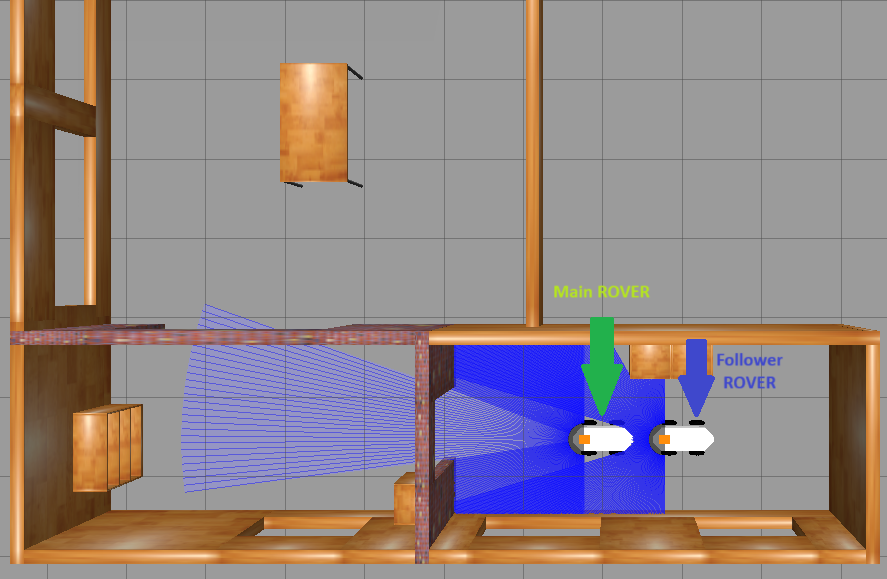


Figure 9 Follower ROVER

## **Funcionamiento para 2 ROVER seguidores y 1 ROVER principal (control por teleoperacion)**

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

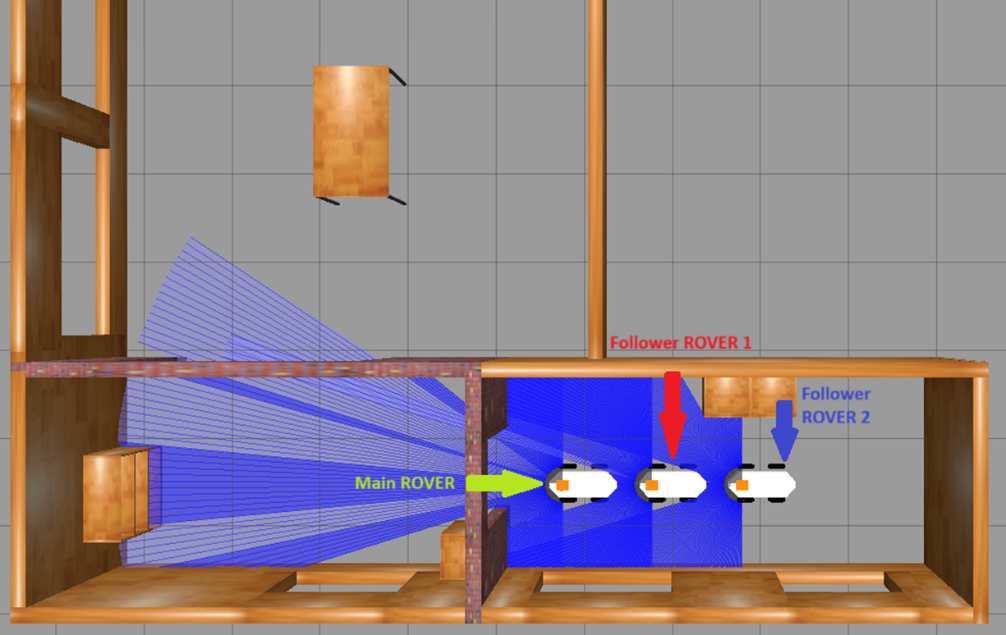


Figure 10 Un rover principal sin navegación autónoma y 2 seguidores

1. Ejecutar el algoritmo seguidor para rover\_1: rosrun follower\_rover follower\_2.py rover\_1

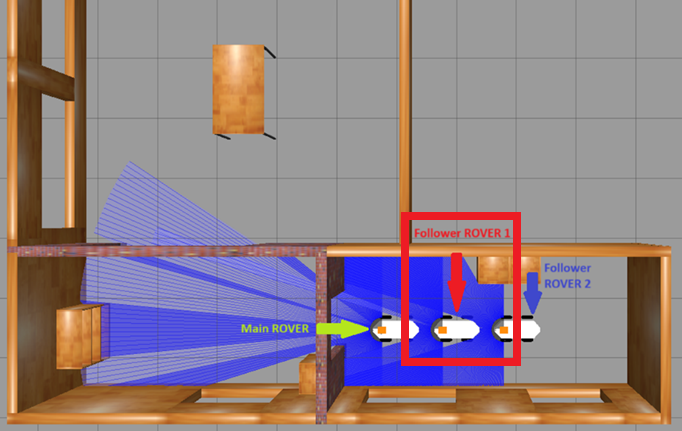


Figure 11 Follower ROVER 1

1. Ejecutar el algoritmo seguidor para rover\_2: rosrun follower\_rover follower\_2.py rover\_2

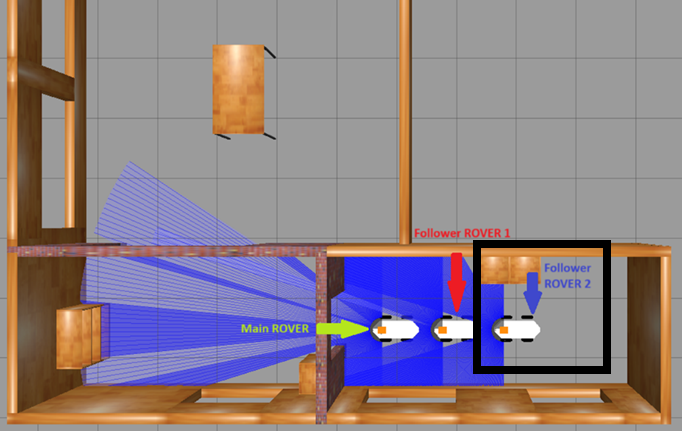


Figure 12 Follower ROVER 2

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

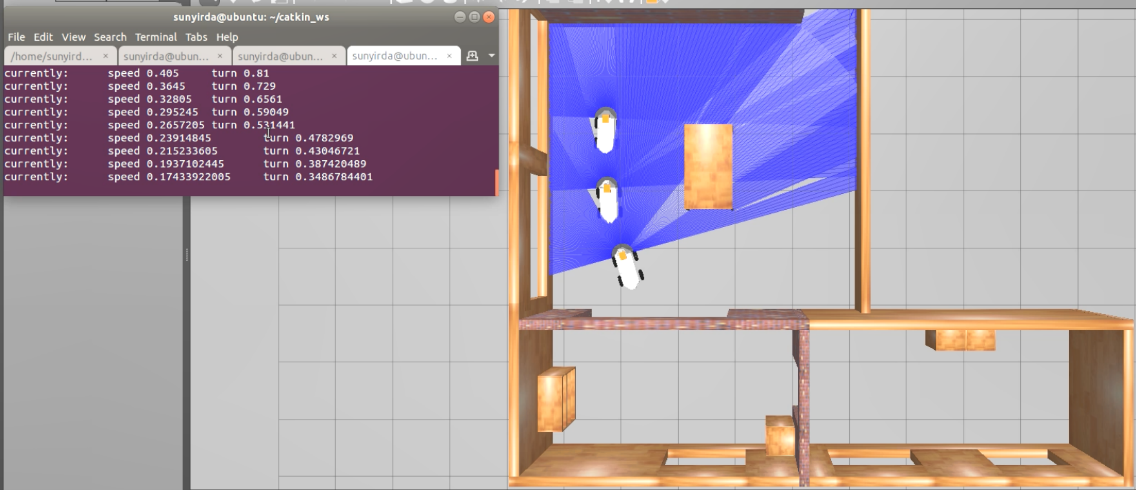


Figure 13 Teleoperación de rover\_0

## **Funcionamiento para 2 ROVER seguidores y 1 ROVER principal (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\_world.launch navigation:=true

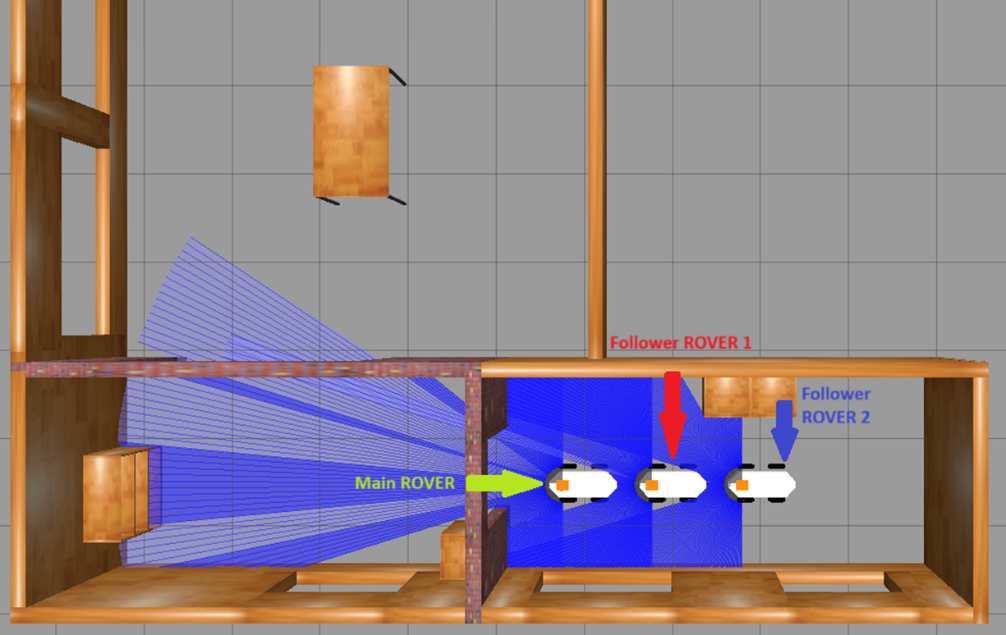


Figure 14 Un rover principal con navegación autónoma y 2 seguidores

1. Ejecutar el algoritmo seguidor para rover\_1: rosrun follower\_rover follower\_2.py rover\_1

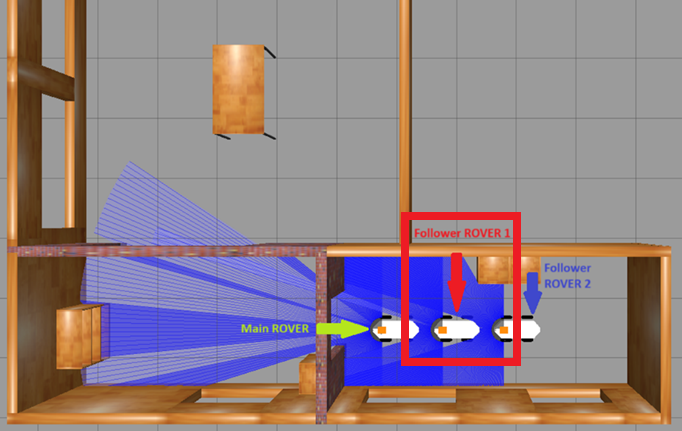


Figure 15 Follower ROVER 1

1. Ejecutar el algoritmo seguidor para rover\_2: rosrun follower\_rover follower\_2.py rover\_2

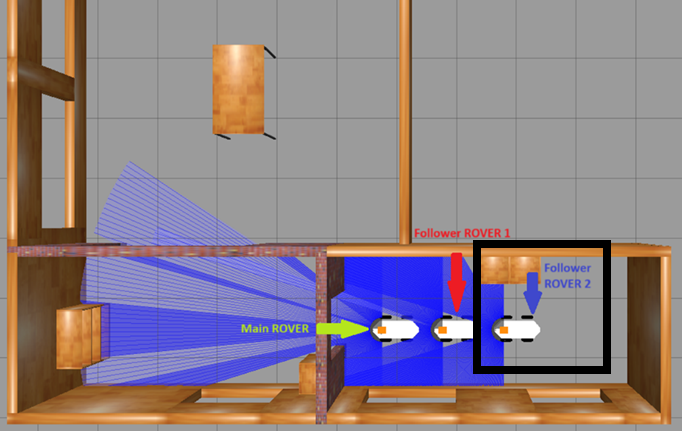


Figure 16 Follower ROVER 2

# **Funcionamiento para ejecución en vida real**

1. Ejecutar todo lo necesario para el ROVER principal. El algoritmo es independiente de lo que ejecute y como lo ejecute el ROVER principal.
2. Ejecutar todo lo necesario para el ROVER seguidor.
3. Ejecutar el algoritmo seguidor para el ROVER correspondiente: rosrun follower\_rover follower\_2.py NOMBRE\_ROVER
4. En caso de querer ejecutar el código en mas de 1 robot repetir el paso 2 y 3 para los otros seguidores.