Follower ROVER
Name: Iesus René Dávila Aguilar

Contenido

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).

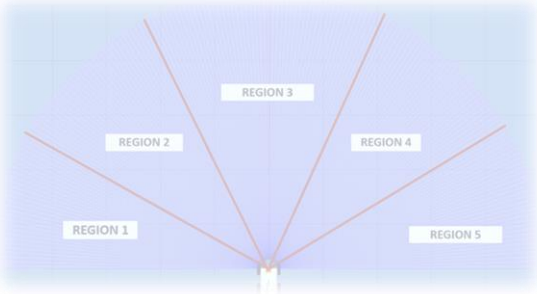

Figure 1 Separaciones de regiones.

Follower ROVER
Name: Iesus René Dávila Aguilar

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

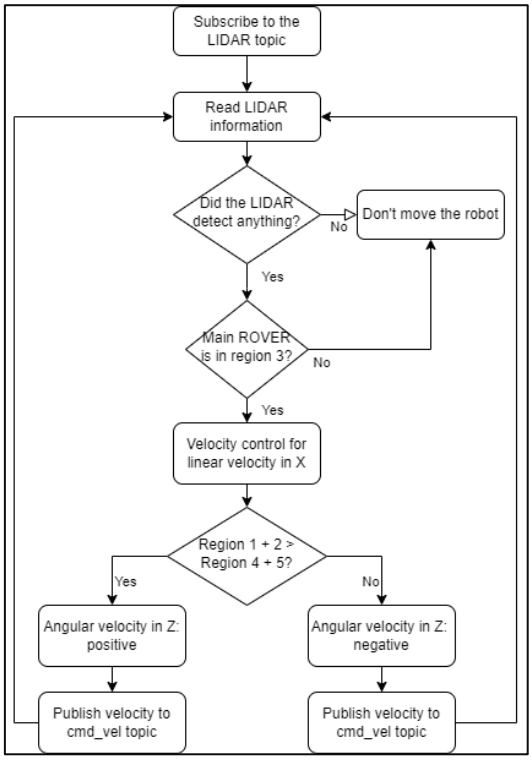

Figure 2 Ciclo de trabajo.

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

Follower ROVER
Name: Iesus René Dávila Aguilar

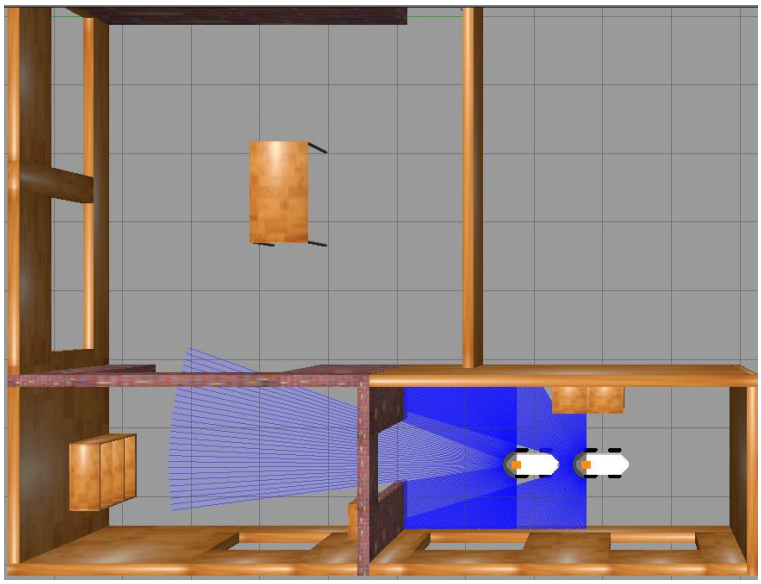

Figure 3 Simulación multirobot sin navegación autónoma

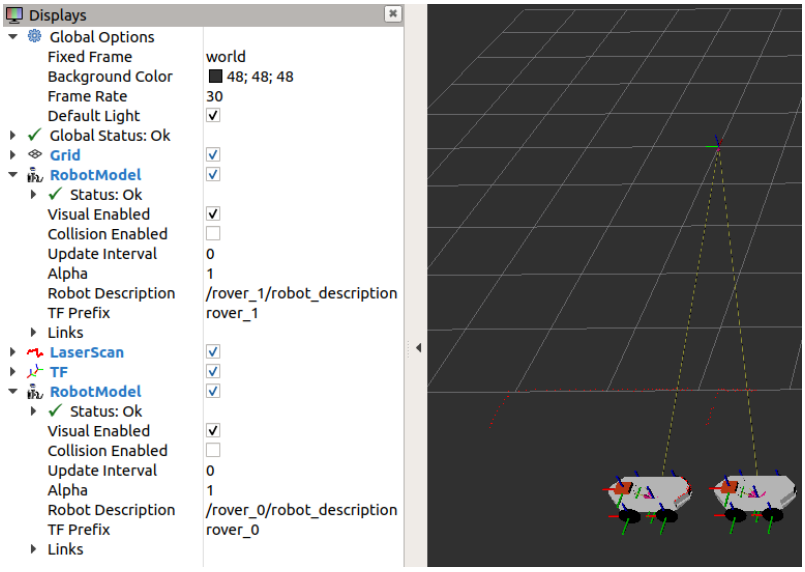

Figure 4 RViz multirobot sin navegación autónoma

2. Ejecutar el algoritmo seguidor (ejemplo con el robot llamado rover_1): rosrun follower_rover follower_2.py NOMBRE_ROVER

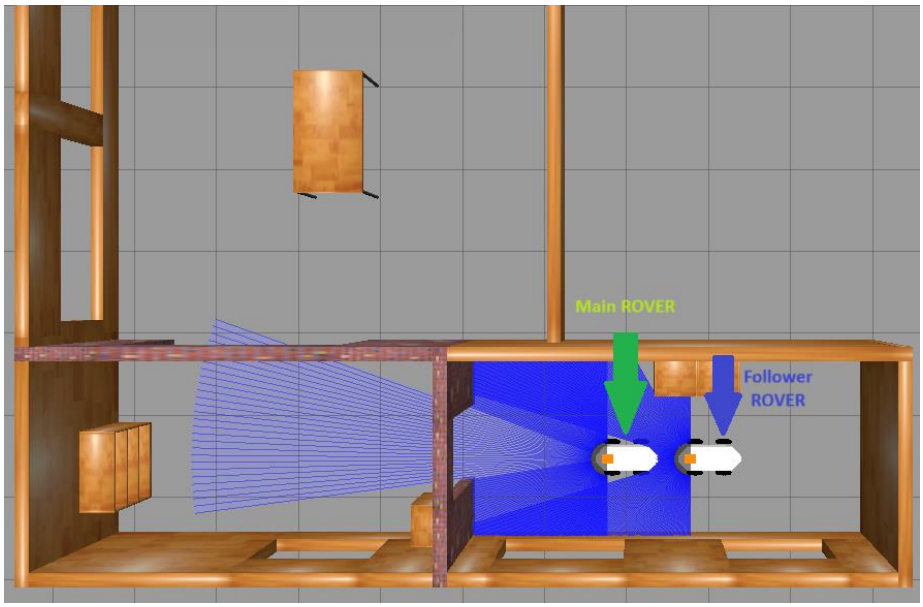

Figure 5 Follower ROVER

3. Ejecutar la teleoperación (ejemplo con el robot llamado rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel

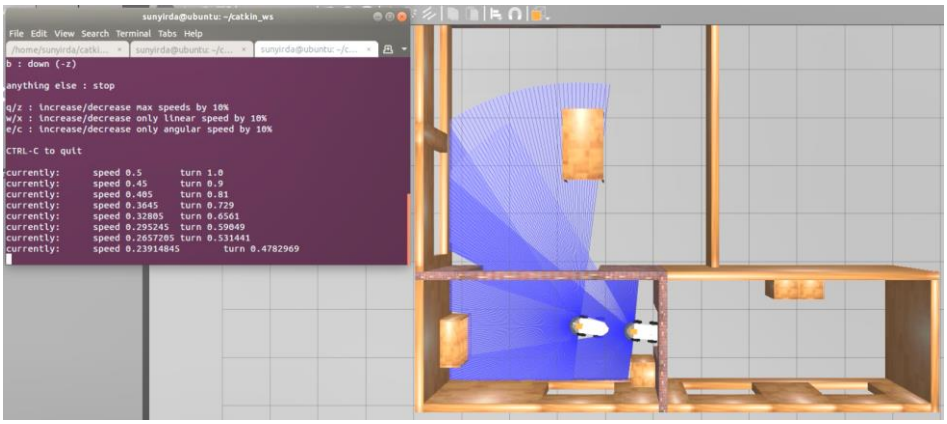

Figure 6 Teleoperación de rover_0

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

Follower ROVER
Name: Iesus René Dávila Aguilar

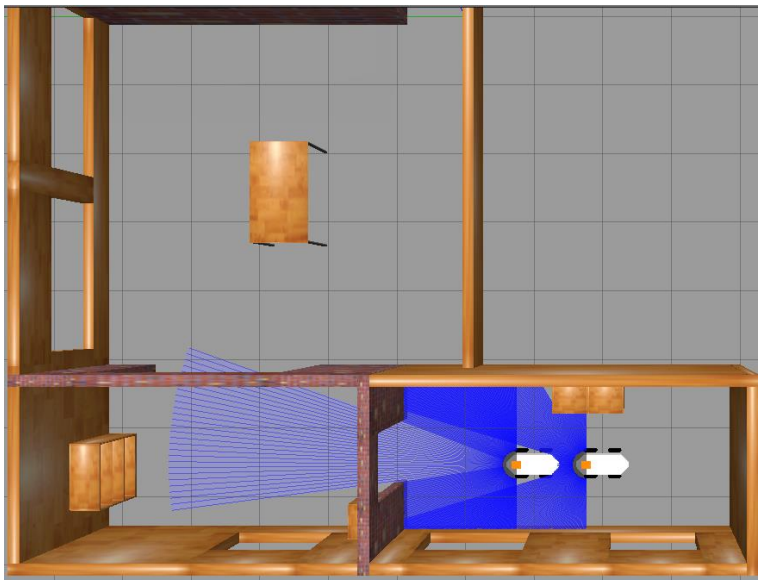

Figure 7 Simulación multirobot con navegación autónoma

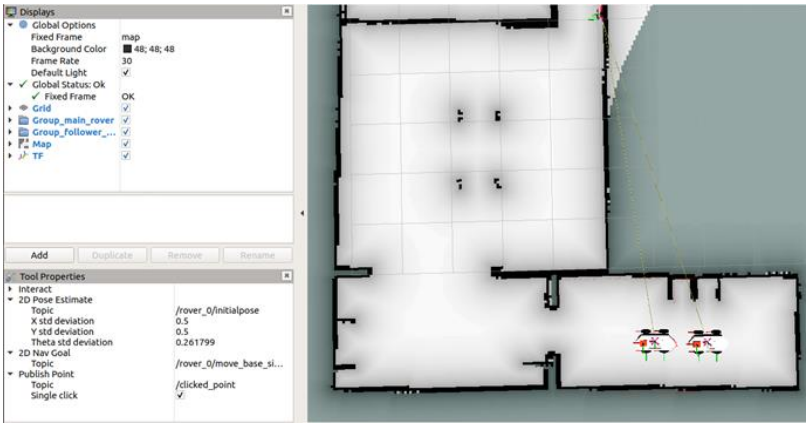

Figure 8 RViz multirobot con navegación autónoma

2. Ejecutar el algoritmo seguidor (ejemplo con el robot llamado rover_1): rosrun follower_rover follower_2.py NOMBRE_ROVER

Follower ROVER
Name: Iesus René Dávila Aguilar

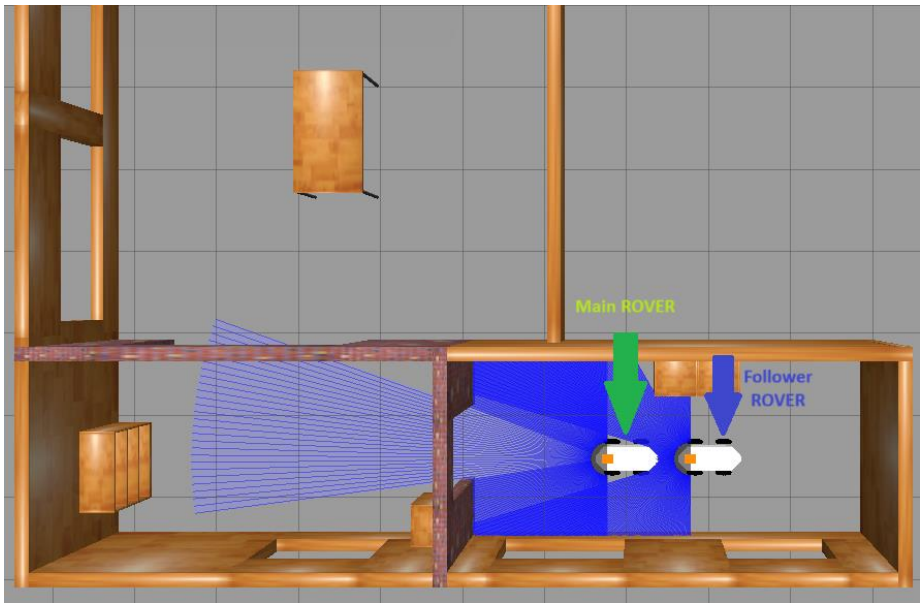

Figure 9 Follower ROVER

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

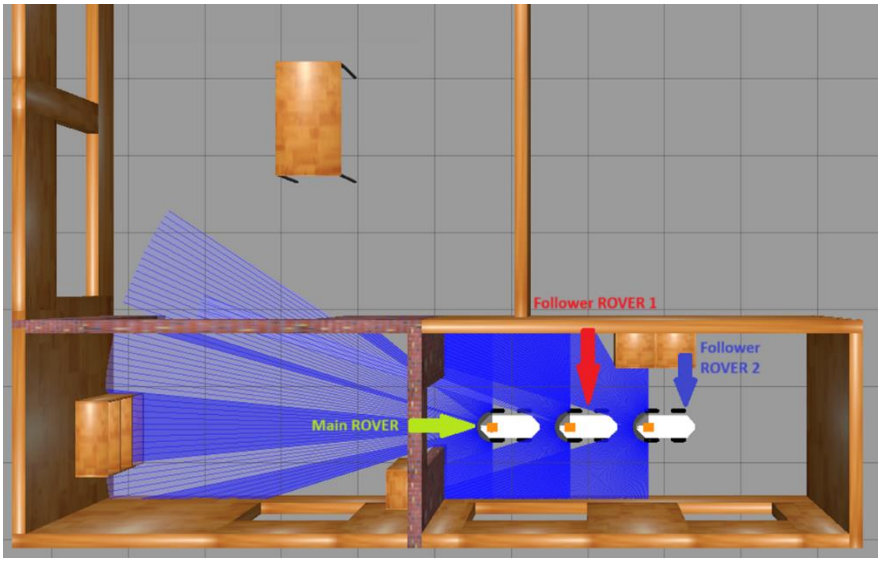

Figure 10 Un rover principal sin navegación autónoma y 2 seguidores

2. Ejecutar el algoritmo seguidor para rover_1: rosrun follower_rover follower_2.py rover_1

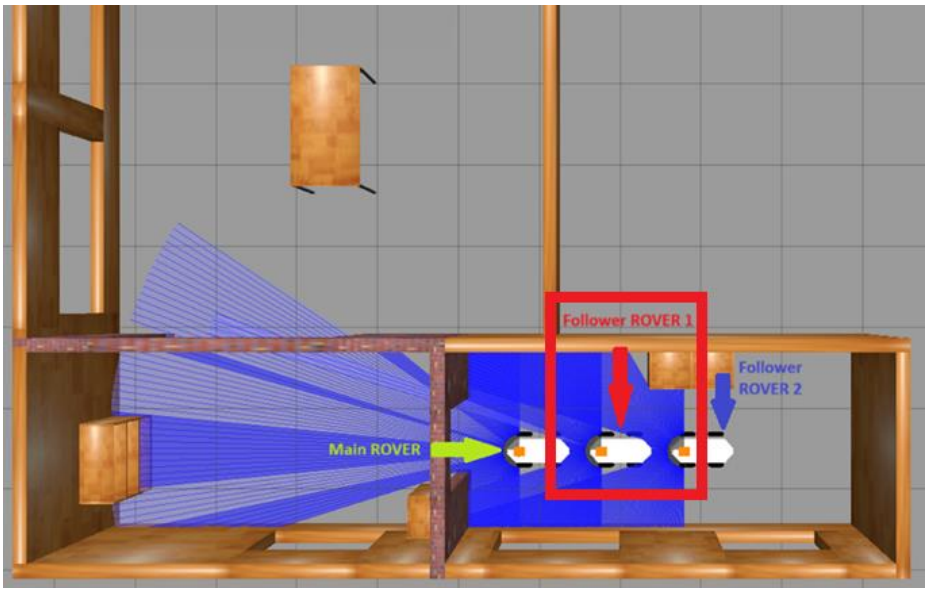

Figure 11 Follower ROVER 1

3. Ejecutar el algoritmo seguidor para rover_2: rosrun follower_rover follower_2.py rover_2

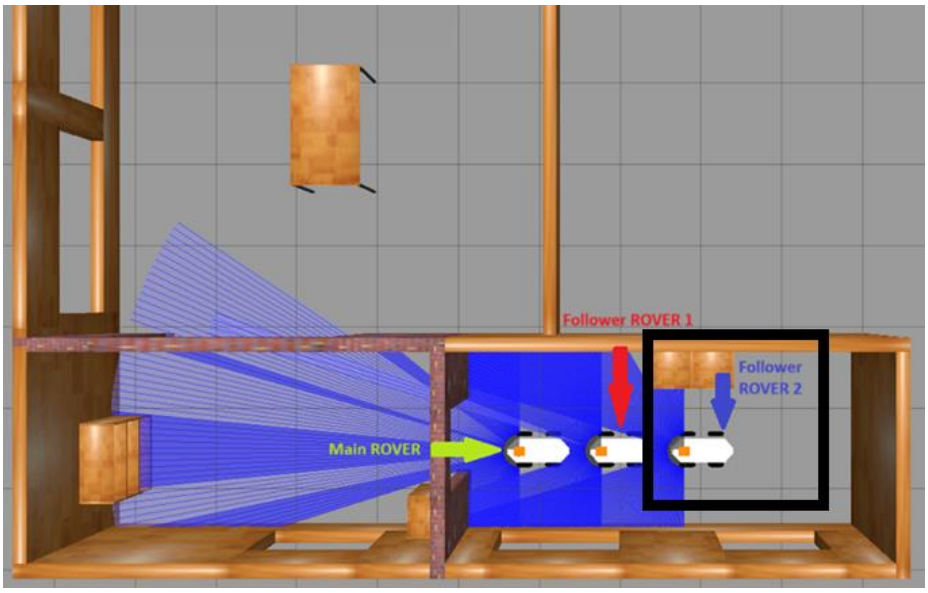

Figure 12 Follower ROVER 2

4. Ejecutar la teleoperación (ejemplo con el robot llamado rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel

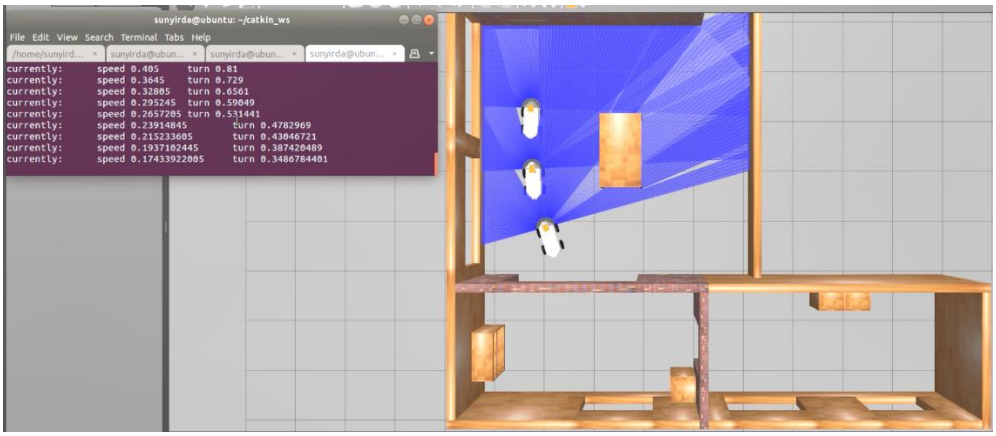

Figure 13 Teleoperación de rover_0

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

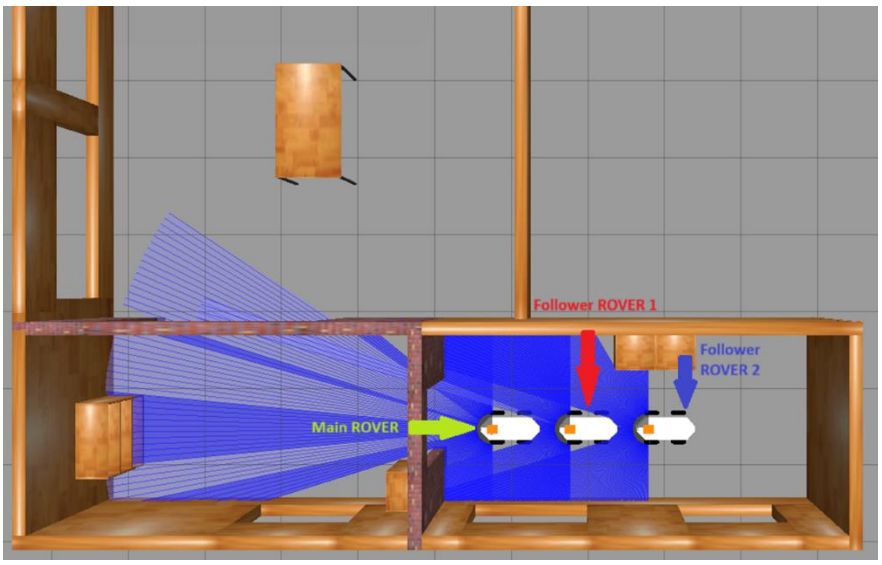

Figure 14 Un rover principal con navegación autónoma y 2 seguidores

2. Ejecutar el algoritmo seguidor para rover_1: rosrun follower_rover follower_2.py rover_1

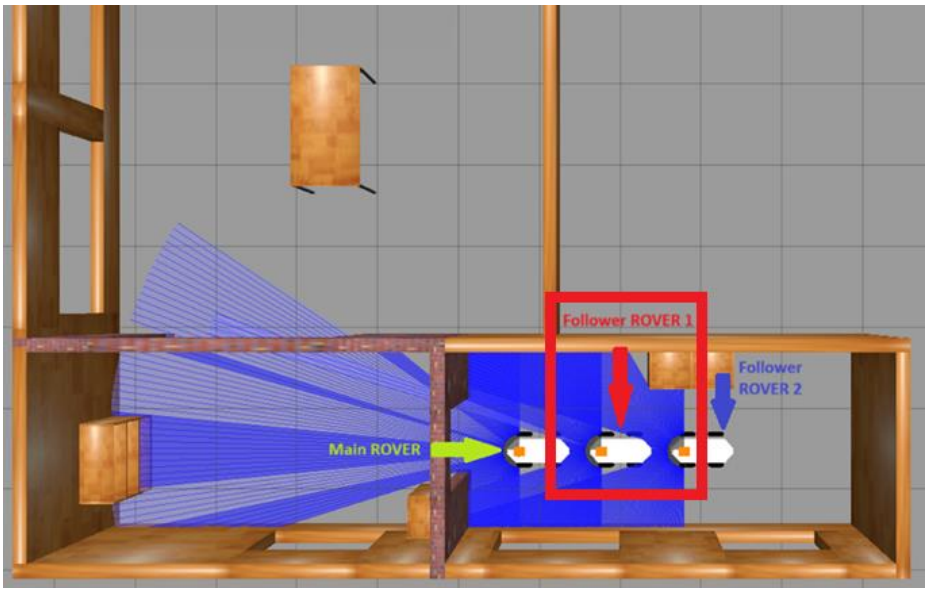

Figure 15 Follower ROVER 1

3. Ejecutar el algoritmo seguidor para rover_2: rosrun follower_rover follower_2.py rover_2

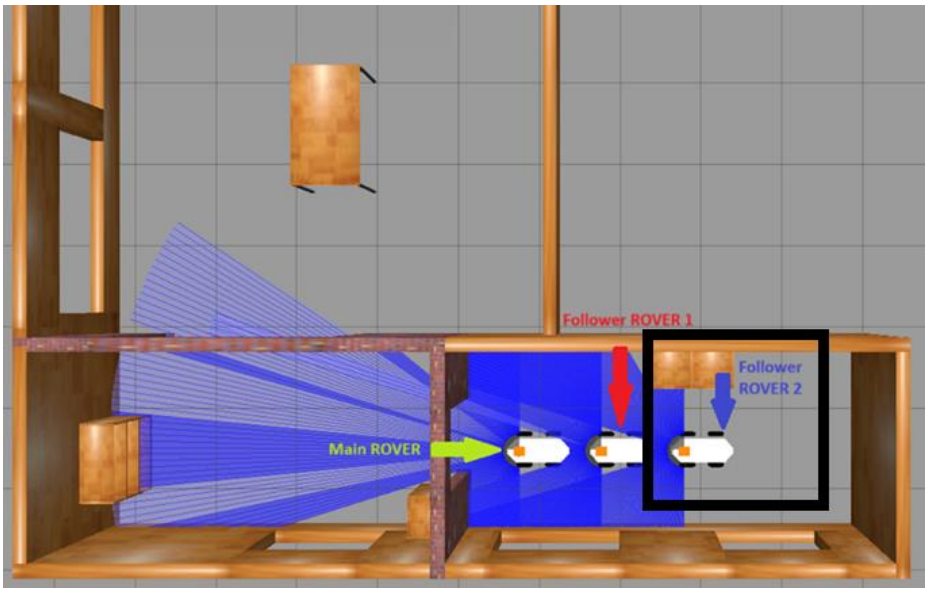

Figure 16 Follower ROVER 2

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.