iron-X's simulation on GAZEBO

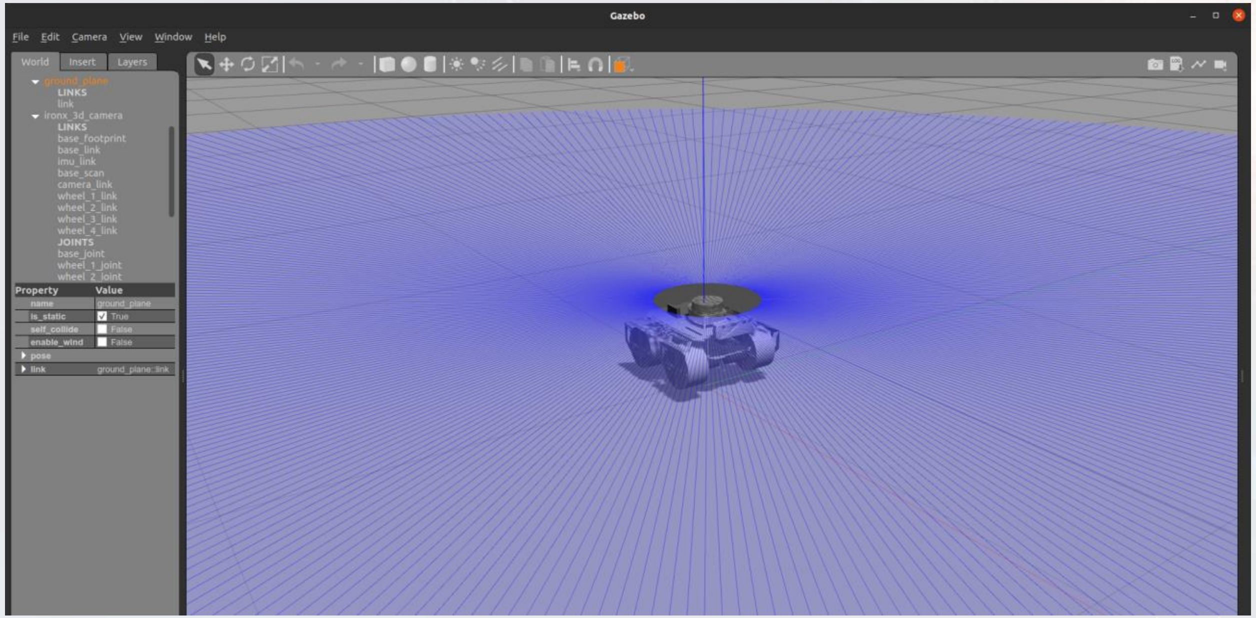
By TESR

ROS2

Thai Embedded

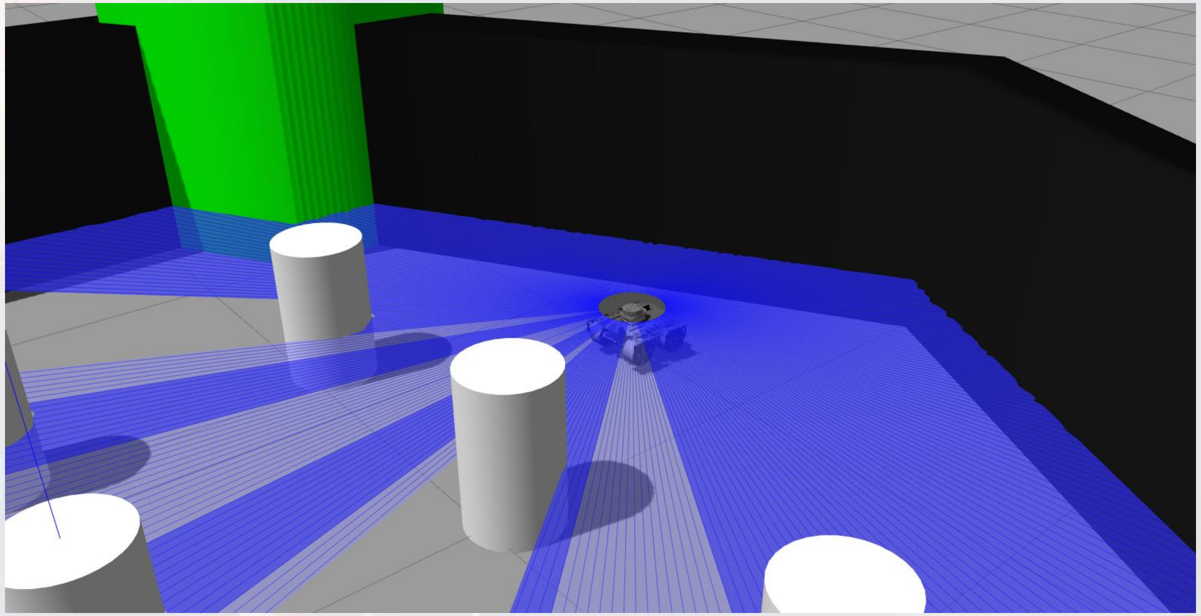
Simulation using GAZEBO

- **Gazebo** is a 3D simulator, while **ROS** serves as the interface for the robot. Combining both results in a powerful robot simulator.
- With **Gazebo** you are able to create a 3D scenario on your computer with robots, obstacles and many other objects. Gazebo also uses a physical engine for **illumination, gravity, inertia, etc.**



ground_plane
LINKS
link
ironx_3d_camera
LINKS
base_footprint
base_link
imu_link
base_scan
camera_link
wheel_1_link
wheel_2_link
wheel_3_link
wheel_4_link
JOINTS
base_joint
wheel_1_joint
wheel_2_joint

Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

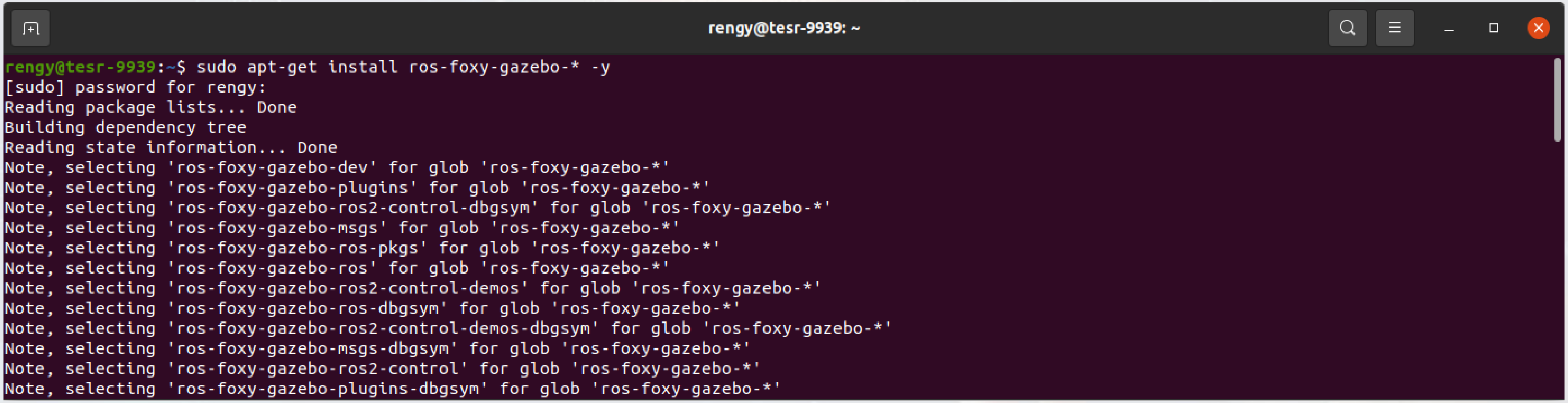


Install GAZEBO

- First, you must install GAZEBO using:

```
sudo apt-get install ros-foxy-gazebo-* -y
```

- Output on terminal should be like below:

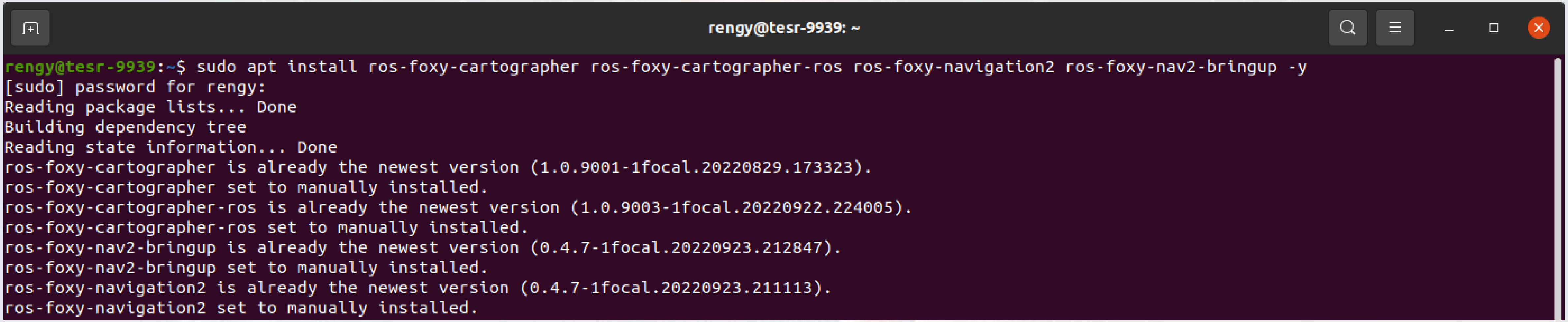


```
rengy@tesr-9939:~$ sudo apt-get install ros-foxy-gazebo-* -y
[sudo] password for rengy:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'ros-foxy-gazebo-dev' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-plugins' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros2-control-dbgsym' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-msgs' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros-pkgs' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros2-control-demos' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros-dbgsym' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros2-control-demos-dbgsym' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-msgs-dbgsym' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-ros2-control' for glob 'ros-foxy-gazebo-*'
Note, selecting 'ros-foxy-gazebo-plugins-dbgsym' for glob 'ros-foxy-gazebo-*'
```

Install Cartographer and Navigation2

- First, we must install package use following commands:
  - Install Cartographer and Navigation2

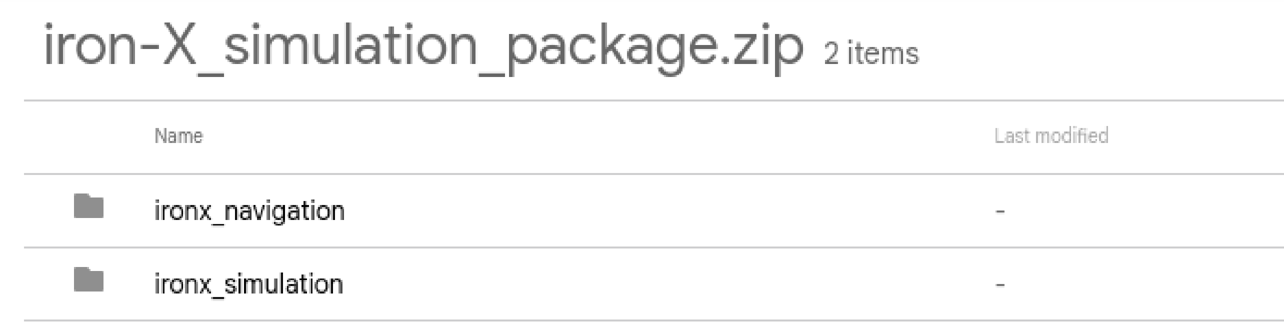
```
sudo apt install ros-foxy-cartographer ros-foxy-cartographer-ros ros-foxy-navigation2 ros-foxy-nav2-bringup -y
```



```
rengy@tesr-9939:~$ sudo apt install ros-foxy-cartographer ros-foxy-cartographer-ros ros-foxy-navigation2 ros-foxy-nav2-bringup -y
[sudo] password for rengy:
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-foxy-cartographer is already the newest version (1.0.9001-1focal.20220829.173323).
ros-foxy-cartographer set to manually installed.
ros-foxy-cartographer-ros is already the newest version (1.0.9003-1focal.20220922.224005).
ros-foxy-cartographer-ros set to manually installed.
ros-foxy-nav2-bringup is already the newest version (0.4.7-1focal.20220923.212847).
ros-foxy-nav2-bringup set to manually installed.
ros-foxy-navigation2 is already the newest version (0.4.7-1focal.20220923.211113).
ros-foxy-navigation2 set to manually installed.
```

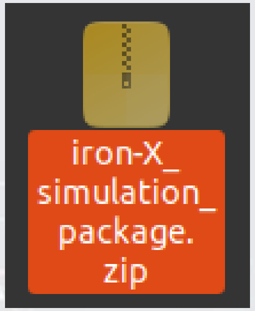
Example package for iron-X's simulation

- Download example package of iron-X's simulation. Click Link

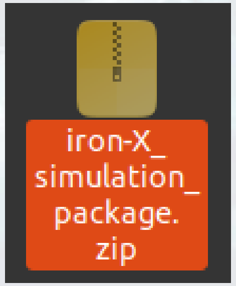


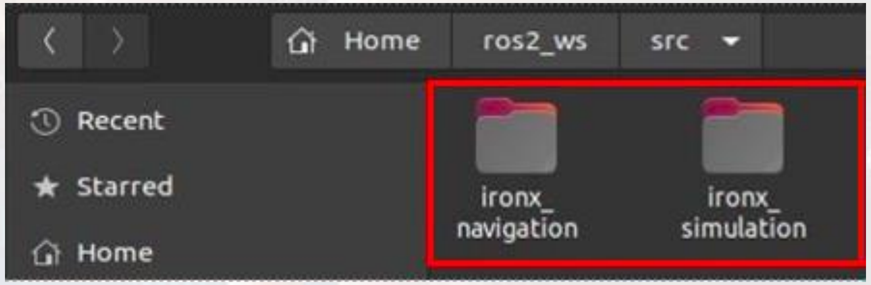
iron-X_simulation_package.zip 2 items

	Name	Last modified
🗀	ironx_navigation	-
🗀	ironx_simulation	-



- Extract file and move to src of **ROS2** workspace(In this case is **ros2_ws**)

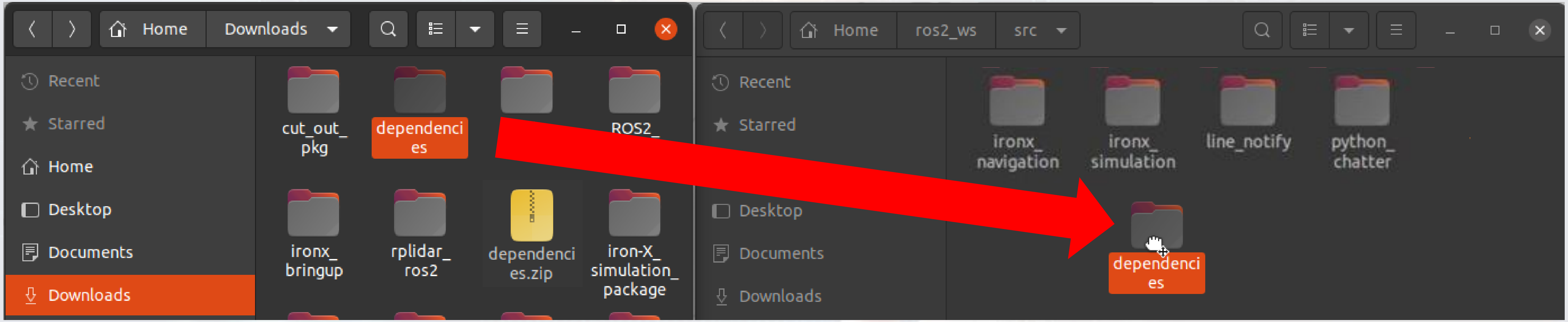




Prepare the dependencies packages

- And then, download the dependencies from Link
- Extract zip file and move to src of **ROS2** workspace.

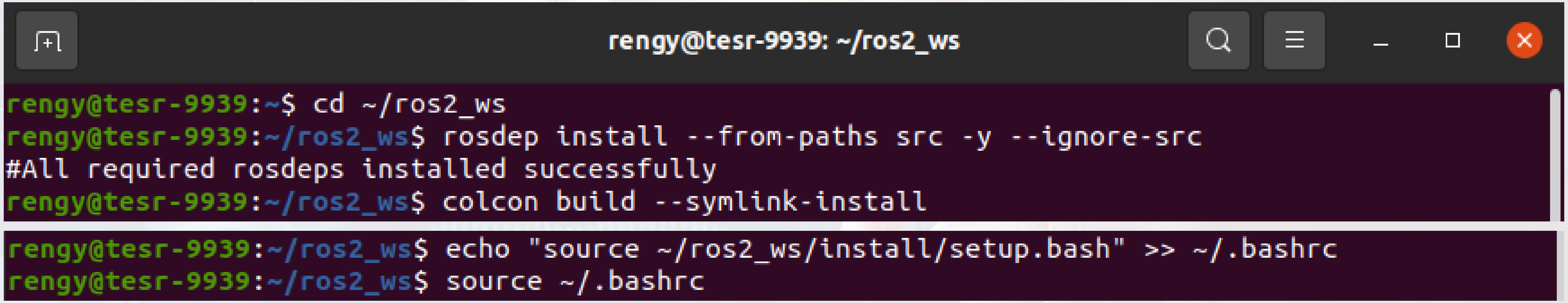
(In this case workspace is ~/ros2_ws)



Build and source the ROS workspace

- After that, run command following below to complete the preparation:

```
cd ~/ros2_ws
rosdep install --from-paths src -y --ignore-src
colcon build --symlink-install

echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
source ~/.bashrc
```



```
rengy@tesr-9939:~$ cd ~/ros2_ws
rengy@tesr-9939:~/ros2_ws$ rosdep install --from-paths src -y --ignore-src
#All required rosdeps installed successfully
rengy@tesr-9939:~/ros2_ws$ colcon build --symlink-install
```

```
rengy@tesr-9939:~/ros2_ws$ echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
rengy@tesr-9939:~/ros2_ws$ source ~/.bashrc
```

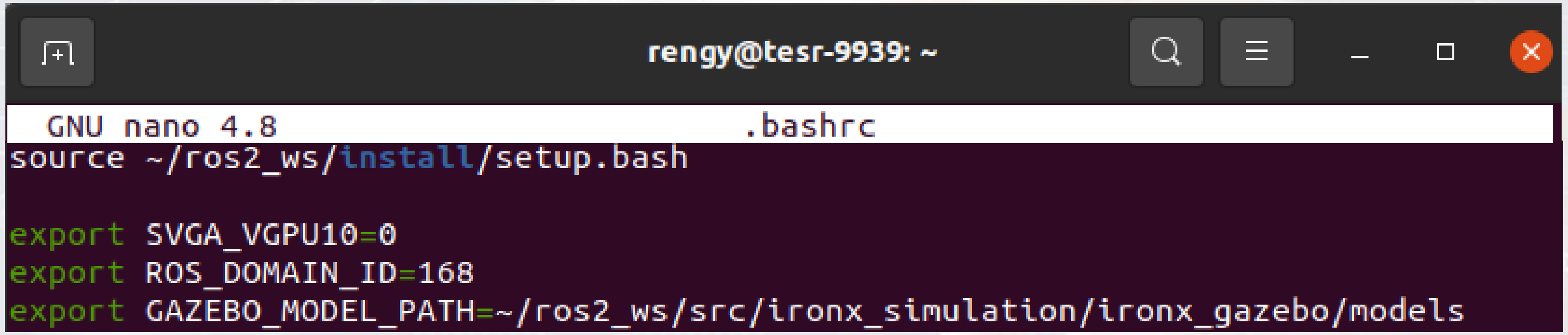
iron-X's simulation on GAZEBO

- First, we must setup a dependencies configuration in .bashrc following as:

```
export SVGA_VGPU10=0
export ROS_DOMAIN_ID=168
export GAZEBO_MODEL_PATH=~/ros2_ws/src/ironx_simulation/ironx_gazebo/models
```

- Edit the .bashrc using nano by type:

```
sudo nano .bashrc
```



```
  GNU nano 4.8                             .bashrc
source ~/ros2_ws/install/setup.bash

export SVGA_VGPU10=0
export ROS_DOMAIN_ID=168
export GAZEBO_MODEL_PATH=~/ros2_ws/src/ironx_simulation/ironx_gazebo/models
```

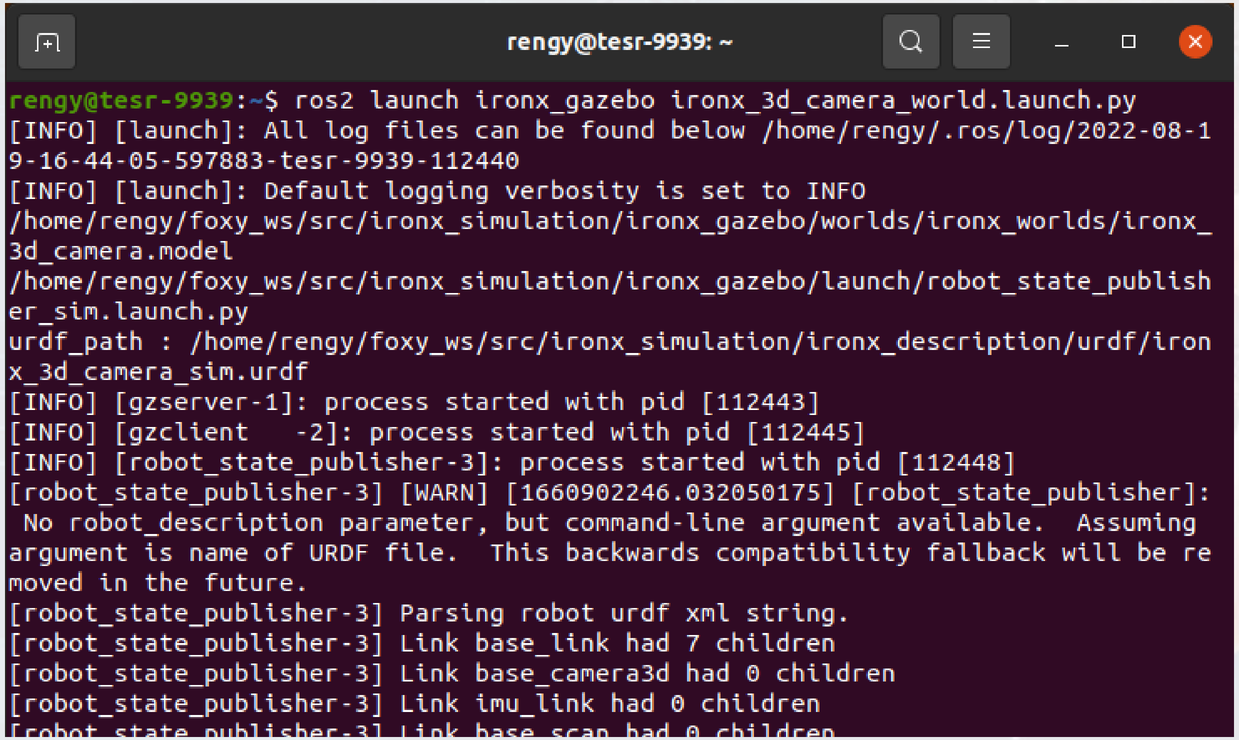
- Then, press **"Ctrl+O" > "Enter"** to **save** and **"Ctrl+X" > "Enter"** to **quit** from nano and type:

```
source ~/.bashrc
```

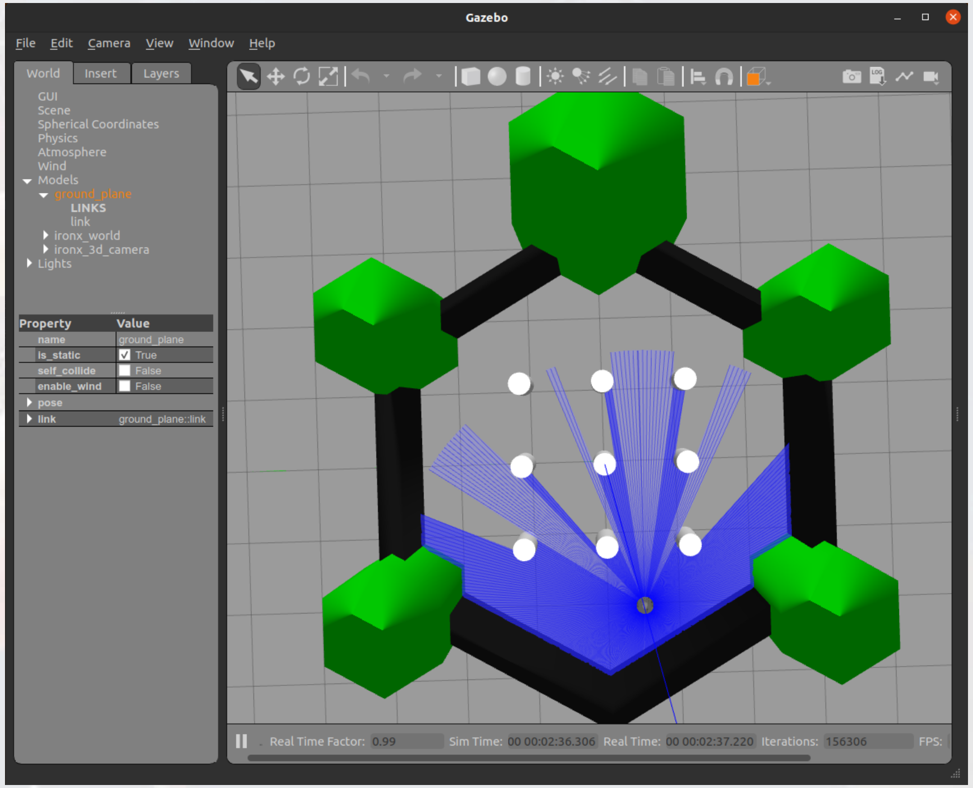
iron-X's simulation on GAZEBO

- You can run example of iron-X's simulation on GAZEBO using:

```
ros2 launch ironx_gazebo ironx_3d_camera_world.launch.py
```



```
rengy@tesr-9939:~$ ros2 launch ironx_gazebo ironx_3d_camera_world.launch.py
[INFO] [launch]: All log files can be found below /home/rengy/.ros/log/2022-08-1
9-16-44-05-597883-tesr-9939-112440
[INFO] [launch]: Default logging verbosity is set to INFO
/home/rengy/foxy_ws/src/ironx_simulation/ironx_gazebo/worlds/ironx_worlds/ironx_
3d_camera.model
/home/rengy/foxy_ws/src/ironx_simulation/ironx_gazebo/launch/robot_state_publish
er_sim.launch.py
urdf_path : /home/rengy/foxy_ws/src/ironx_simulation/ironx_description/urdf/iron
x_3d_camera_sim.urdf
[INFO] [gzserver-1]: process started with pid [112443]
[INFO] [gzclient   -2]: process started with pid [112445]
[INFO] [robot_state_publisher-3]: process started with pid [112448]
[robot_state_publisher-3] [WARN] [1660902246.032050175] [robot_state_publisher]:
 No robot_description parameter, but command-line argument available.  Assuming
argument is name of URDF file.  This backwards compatibility fallback will be re
moved in the future.
[robot_state_publisher-3] Parsing robot urdf xml string.
[robot_state_publisher-3] Link base_link had 7 children
[robot_state_publisher-3] Link base_camera3d had 0 children
[robot_state_publisher-3] Link imu_link had 0 children
[robot_state_publisher-3] Link base_scan had 0 children
```



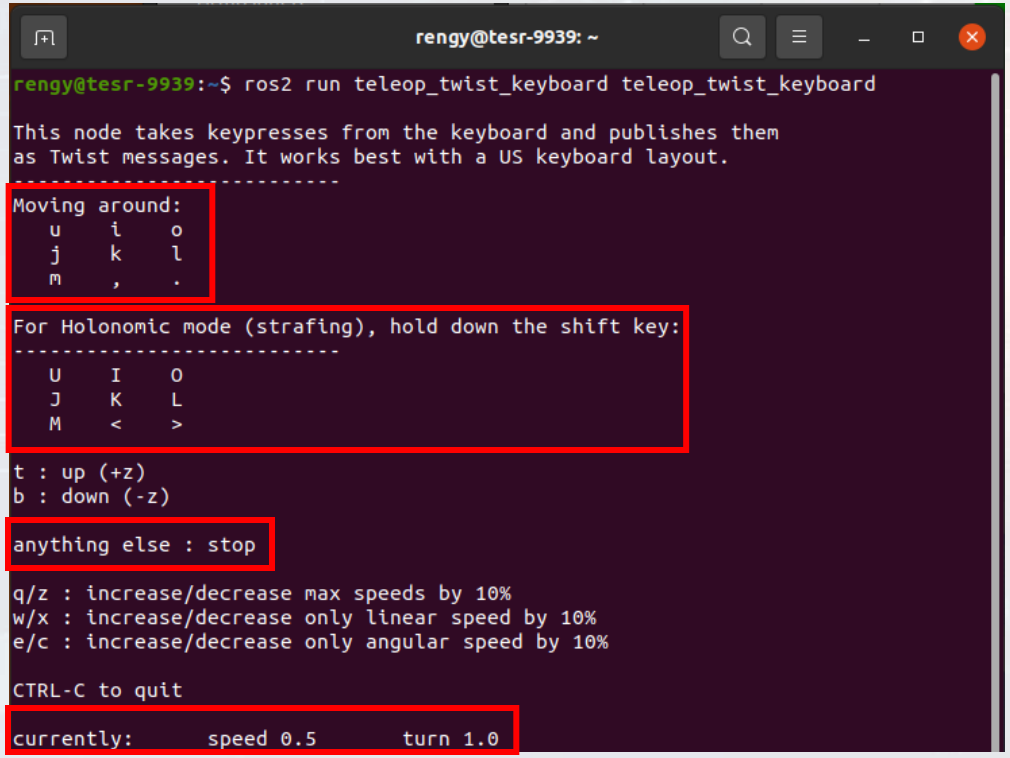
Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Real Time Factor: 0.99 Sim Time: 00 00:02:36.306 Real Time: 00 00:02:37.220 Iterations: 156306 FPS:

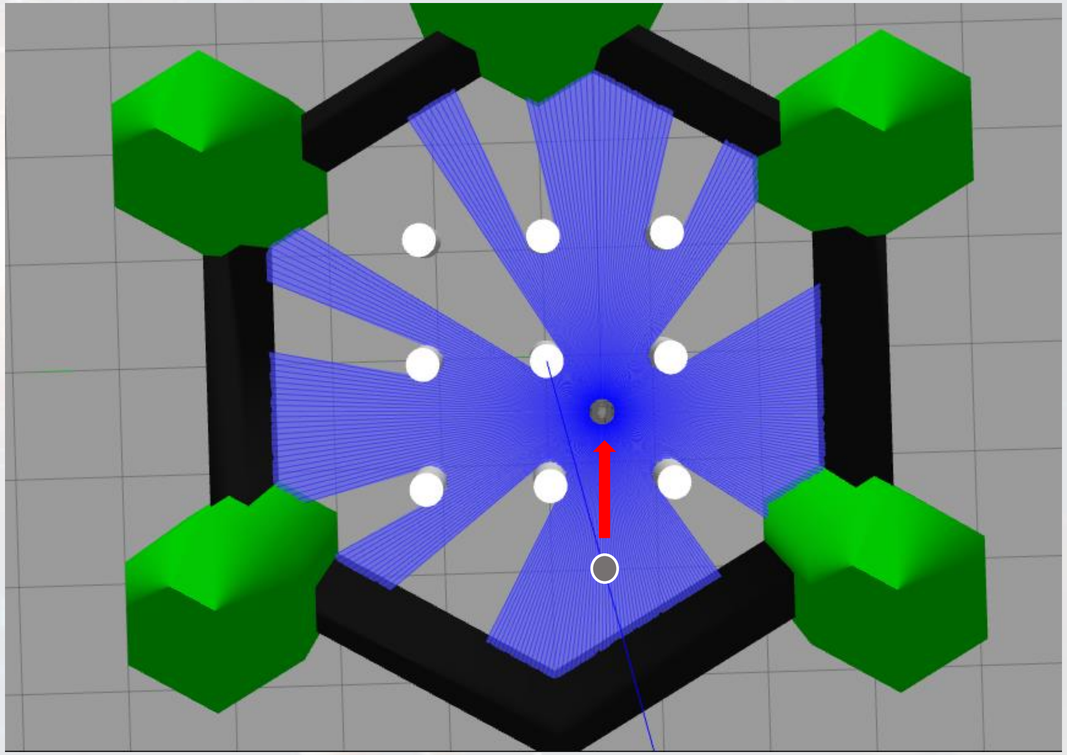
iron-X's keyboard control

- In simulation, you can control iron-X by teleop_twist_keyboard using:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```



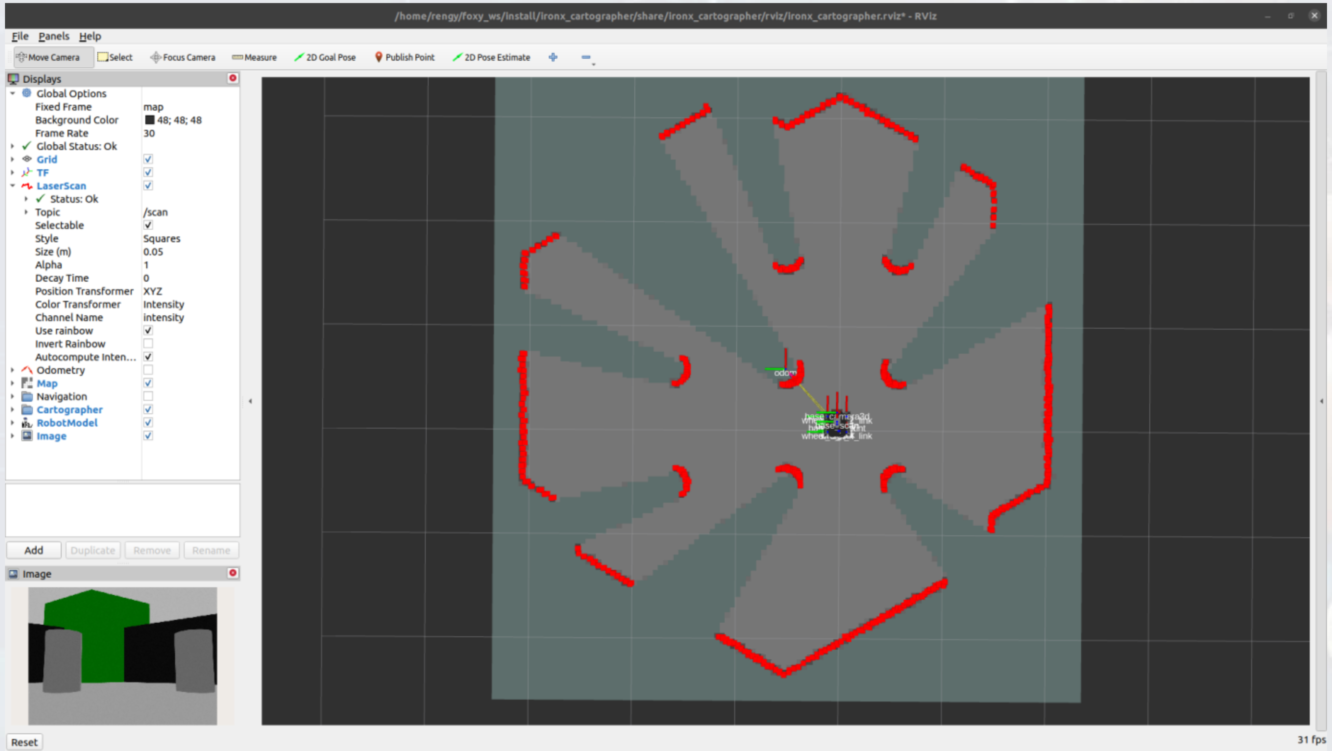
```
rengy@tesr-9939:~$ ros2 run teleop_twist_keyboard teleop_twist_keyboard

This node takes keypresses from the keyboard and publishes them
as Twist messages. It works best with a US keyboard layout.
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```



SLAM Cartographer

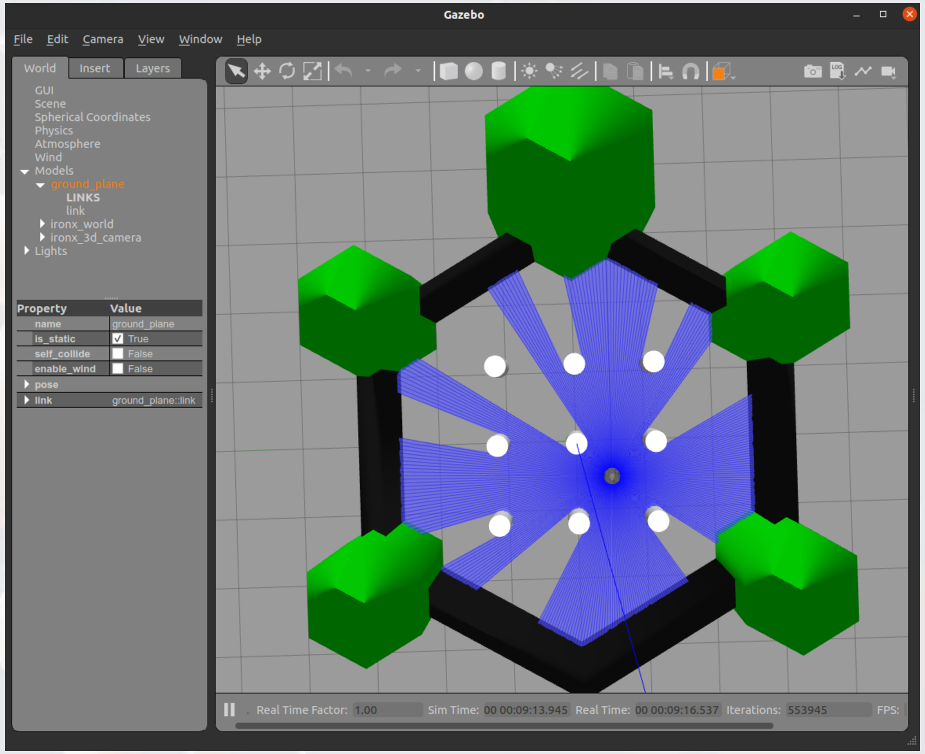
- Then, you can run **SLAM cartographer** to draw a map for navigation using:

```
ros2 launch ironx_navigation view_cartographer.launch.py use_sim_time:=true
```



/home/rengy/foxy_ws/install/ironx_cartographer/share/ironx_cartographer/rviz/ironx_cartographer.rviz* - RViz

Displays	
Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
Grid	☑
TF	☑
LaserScan	☑
✓ Status: Ok	
Topic	/scan
Selectable	☑
Style	Squares
Size (m)	0.05
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	Intensity
Channel Name	intensity
Use rainbow	☑
Invert Rainbow	□
Autocompute Inten...	☑
Odometry	□
Map	☑
Navigation	□
Cartographer	☑
RobotModel	☑
Image	☑



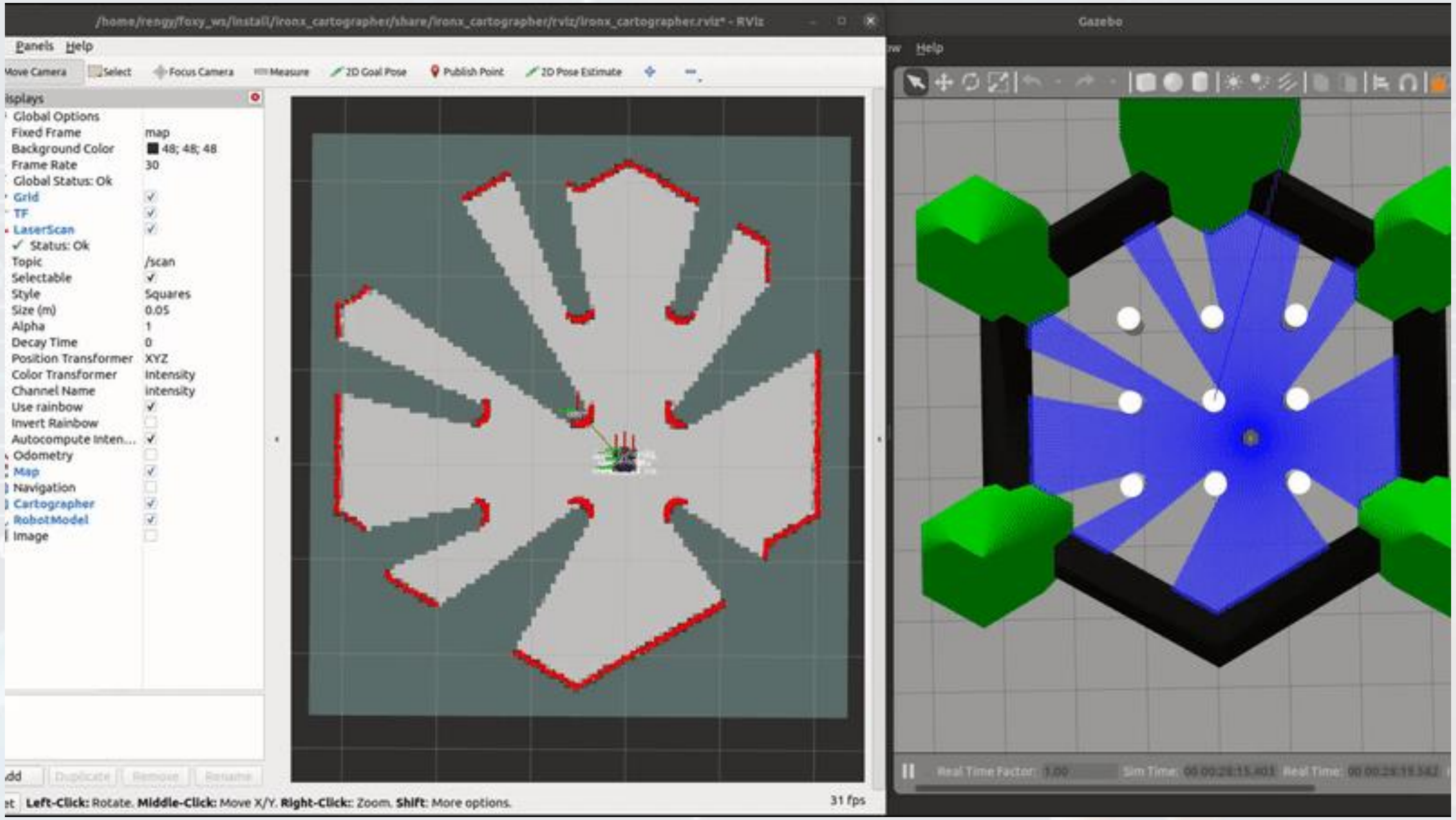
Gazebo

Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Real Time Factor: 1.00 Sim Time: 00 00:09:13.945 Real Time: 00 00:09:16.537 Iterations: 553945 FPS:

SLAM Cartographer

- Let's draw a map using **teleop_twist_keyboard** control the **iron-X.**



/home/rengy/foxy_ws/install/ironx_cartographer/share/ironx_cartographer/rviz/ironx_cartographer.rviz* - RViz

```
isplays
 Global Options
 Fixed Frame                 map
 Background Color            ■ 48; 48; 48
 Frame Rate                  30
 Global Status: Ok
 Grid                        ✓
 TF                          ✓
 LaserScan                   ✓
  ✓ Status: Ok
 Topic                       /scan
 Selectable                  ✓
 Style                       Squares
 Size (m)                    0.05
 Alpha                       1
 Decay Time                  0
 Position Transformer        XYZ
 Color Transformer           Intensity
 Channel Name                intensity
 Use rainbow                 ✓
 Invert Rainbow              □
 Autocompute Inten...        ✓
 Odometry                    □
 Map                         ✓
 Navigation                  □
 Cartographer                ✓
 RobotModel                  ✓
 Image                       □
```

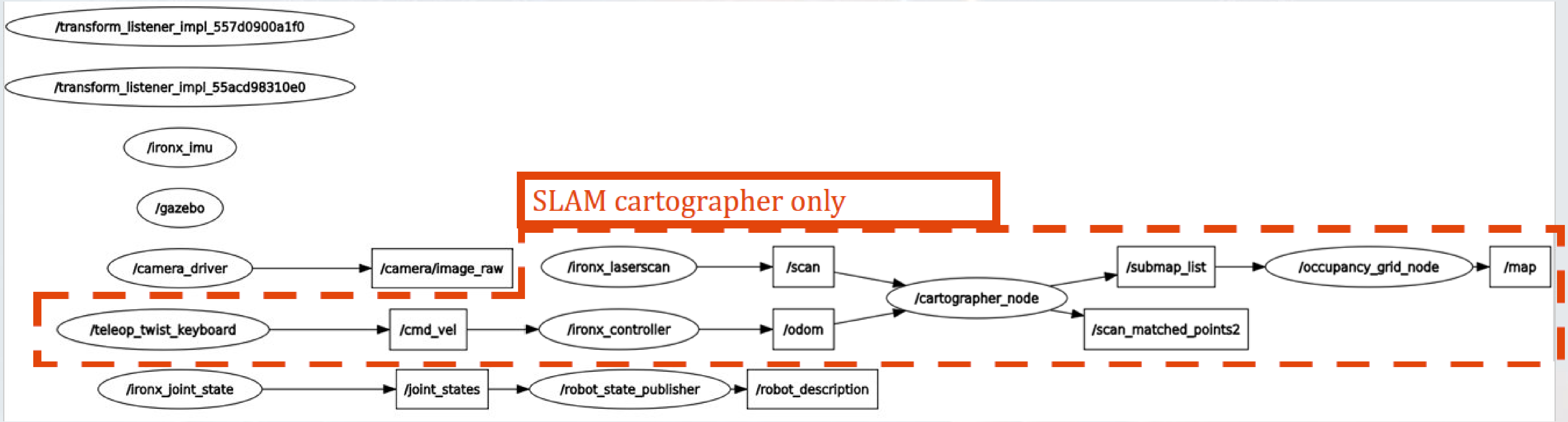
et **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click:** Zoom. **Shift:** More options.

*Speed x3.7

SLAM Cartographer's RosGraph

- You can see RosGraph of SLAM Cartographer using:

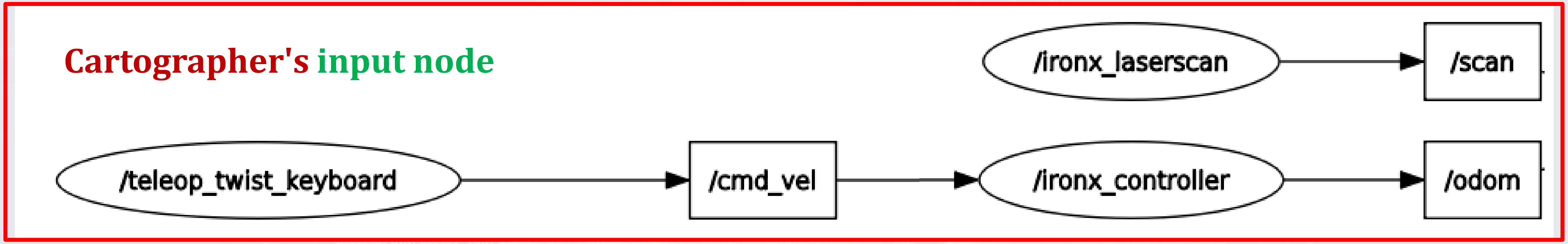
```
rqt_graph
```



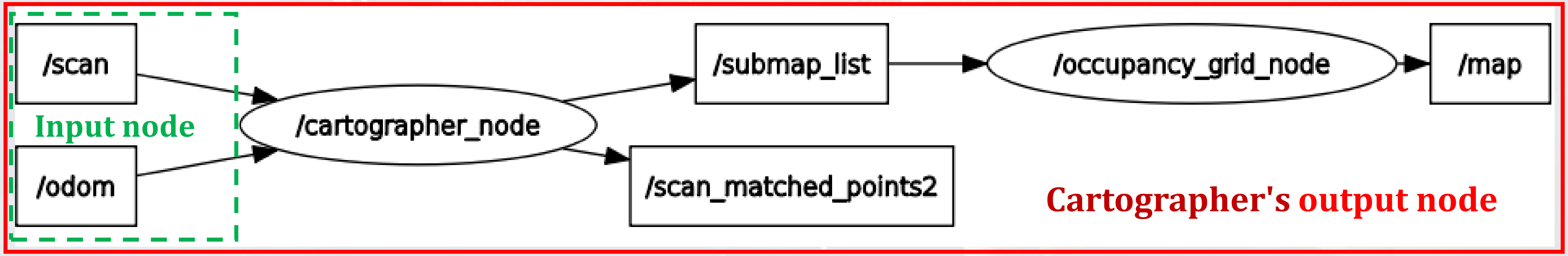
SLAM cartographer only

SLAM Cartographer's RosGraph

- RosGraph of the **SLAM Cartographer** only:



Cartographer's input node



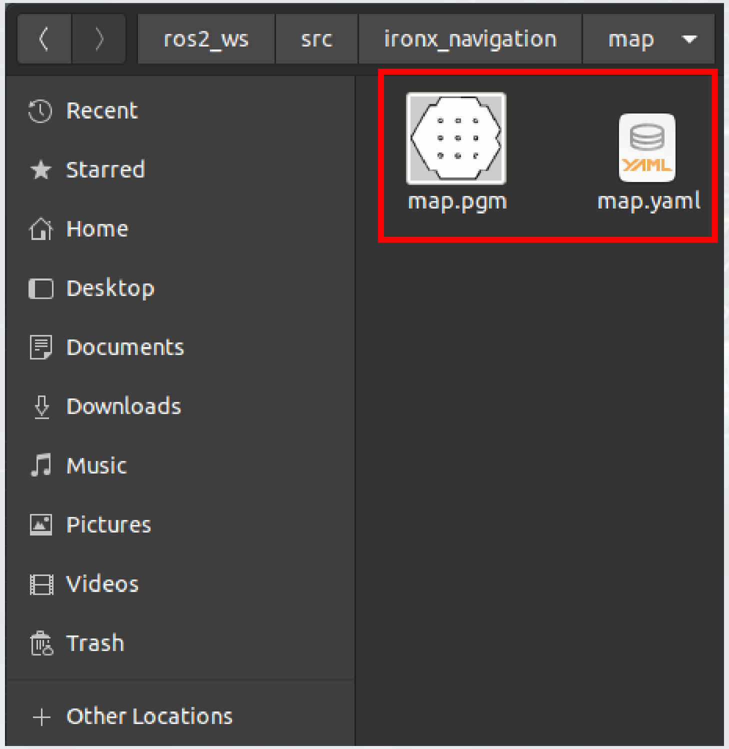
Input node

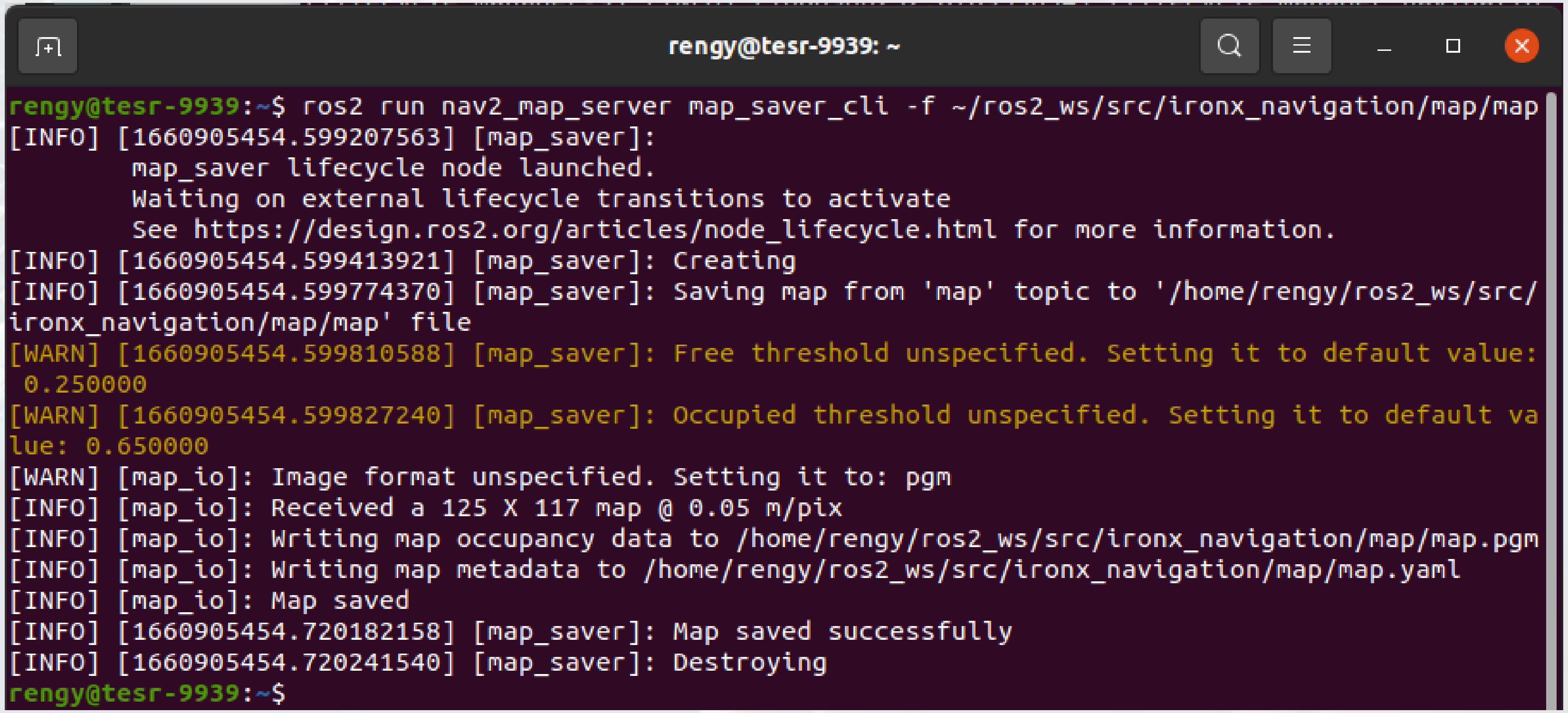
Cartographer's output node

Save a map for Navigation

- After draw a map as much as you're prefer, You can save a map using:

```
ros2 run nav2_map_server map_saver_cli -f ~/ros2_ws/src/ironx_navigation/map/map
```





```
rengy@tesr-9939:~$ ros2 run nav2_map_server map_saver_cli -f ~/ros2_ws/src/ironx_navigation/map/map
[INFO] [1660905454.599207563] [map_saver]:
        map_saver lifecycle node launched.
        Waiting on external lifecycle transitions to activate
        See https://design.ros2.org/articles/node_lifecycle.html for more information.
[INFO] [1660905454.599413921] [map_saver]: Creating
[INFO] [1660905454.599774370] [map_saver]: Saving map from 'map' topic to '/home/rengy/ros2_ws/src/
ironx_navigation/map/map' file
[WARN] [1660905454.599810588] [map_saver]: Free threshold unspecified. Setting it to default value:
 0.250000
[WARN] [1660905454.599827240] [map_saver]: Occupied threshold unspecified. Setting it to default va
lue: 0.650000
[WARN] [map_io]: Image format unspecified. Setting it to: pgm
[INFO] [map_io]: Received a 125 X 117 map @ 0.05 m/pix
[INFO] [map_io]: Writing map occupancy data to /home/rengy/ros2_ws/src/ironx_navigation/map/map.pgm
[INFO] [map_io]: Writing map metadata to /home/rengy/ros2_ws/src/ironx_navigation/map/map.yaml
[INFO] [map_io]: Map saved
[INFO] [1660905454.720182158] [map_saver]: Map saved successfully
[INFO] [1660905454.720241540] [map_saver]: Destroying
rengy@tesr-9939:~$
```

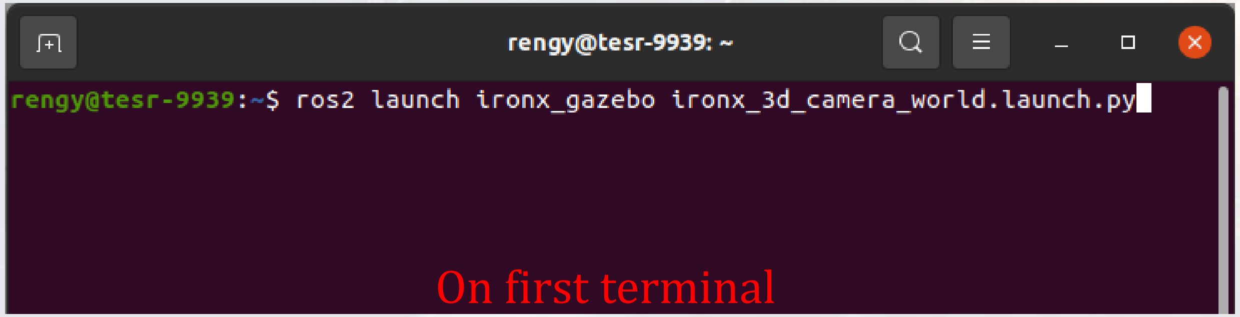
Launch the Navigation

- Now, you have a map. So, you may close all other old terminal and open two new terminal.
  - On the first terminal, launch the gazebo world using:

```
ros2 launch ironx_gazebo ironx_3d_camera_world.launch.py
```

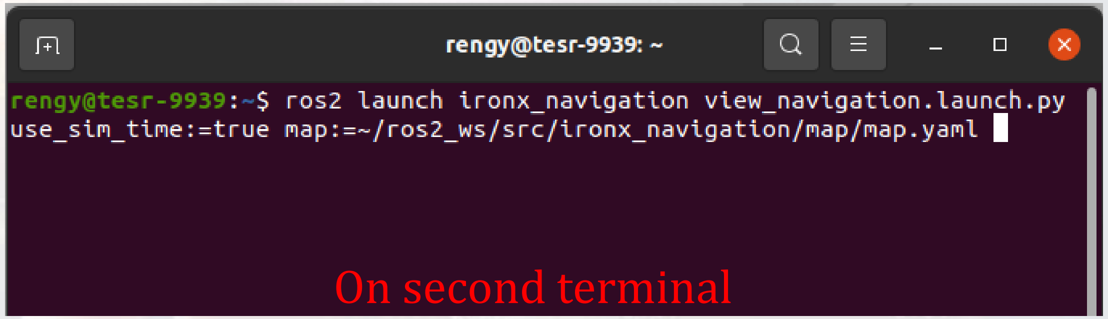
  - On the second terminal, launch the navigation with the map we drawn using:

```
ros2 launch ironx_navigation view_navigation.launch.py use_sim_time:=true map:=~/ros2_ws/src/ironx_navigation/map/map.yaml
```



```
rengy@tesr-9939:~$ ros2 launch ironx_gazebo ironx_3d_camera_world.launch.py
```

On first terminal

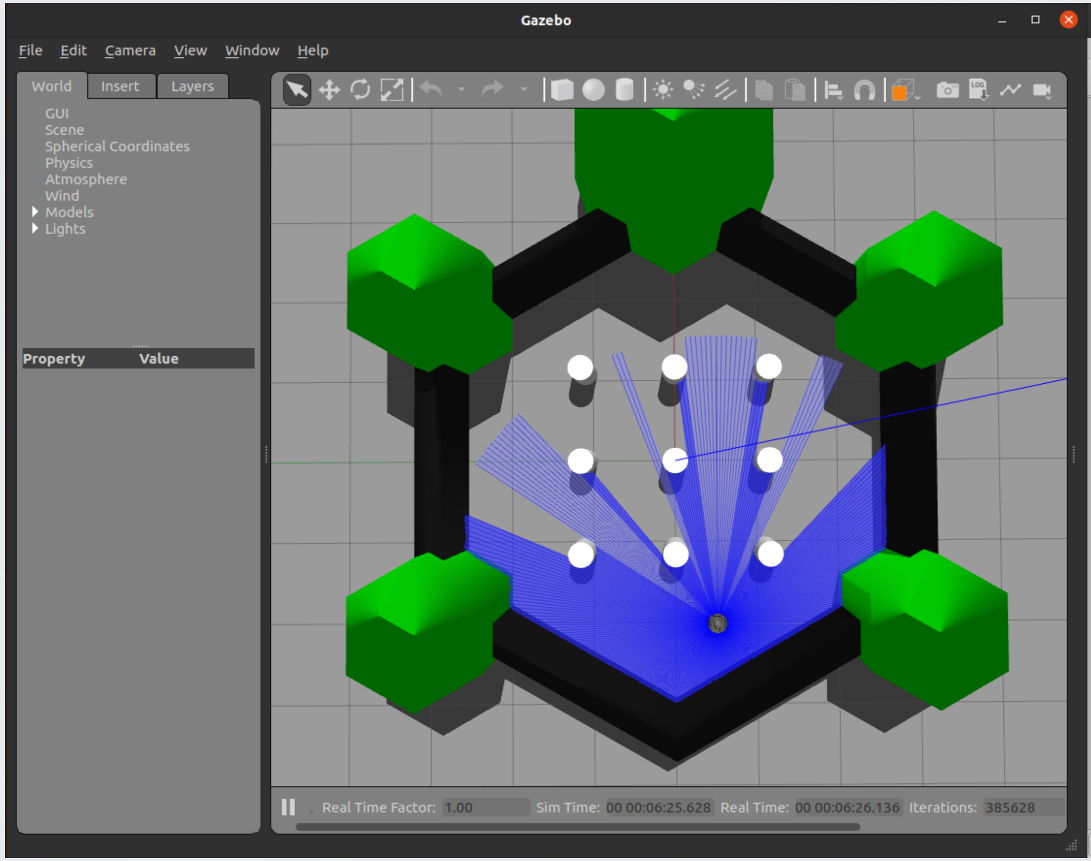


```
rengy@tesr-9939:~$ ros2 launch ironx_navigation view_navigation.launch.py
use_sim_time:=true map:=~/ros2_ws/src/ironx_navigation/map/map.yaml
```

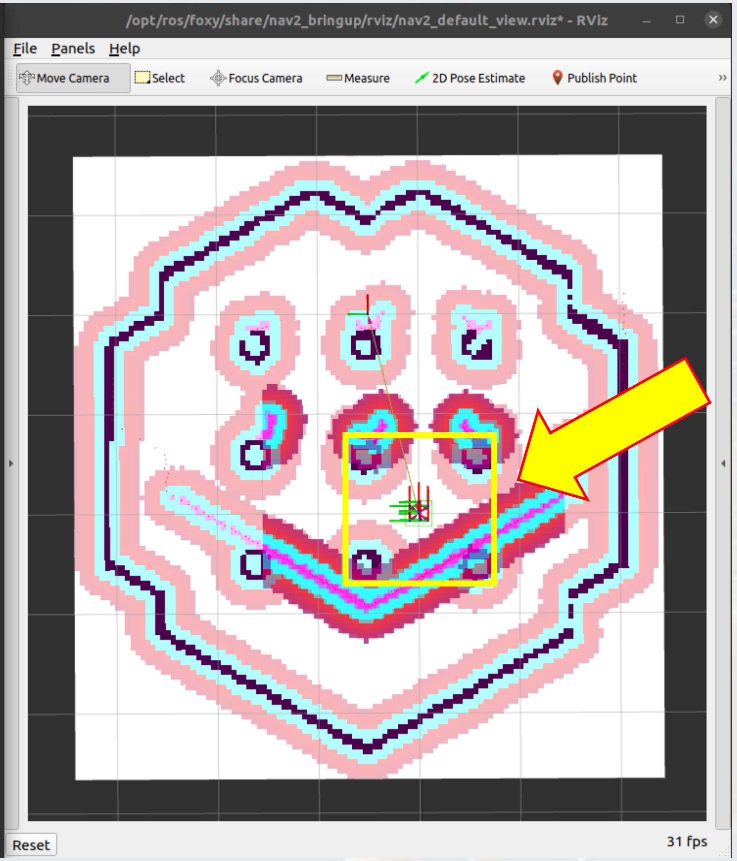
On second terminal

Launch the Navigation

- After that, the result should be show the Gazebo world and Rviz as below:



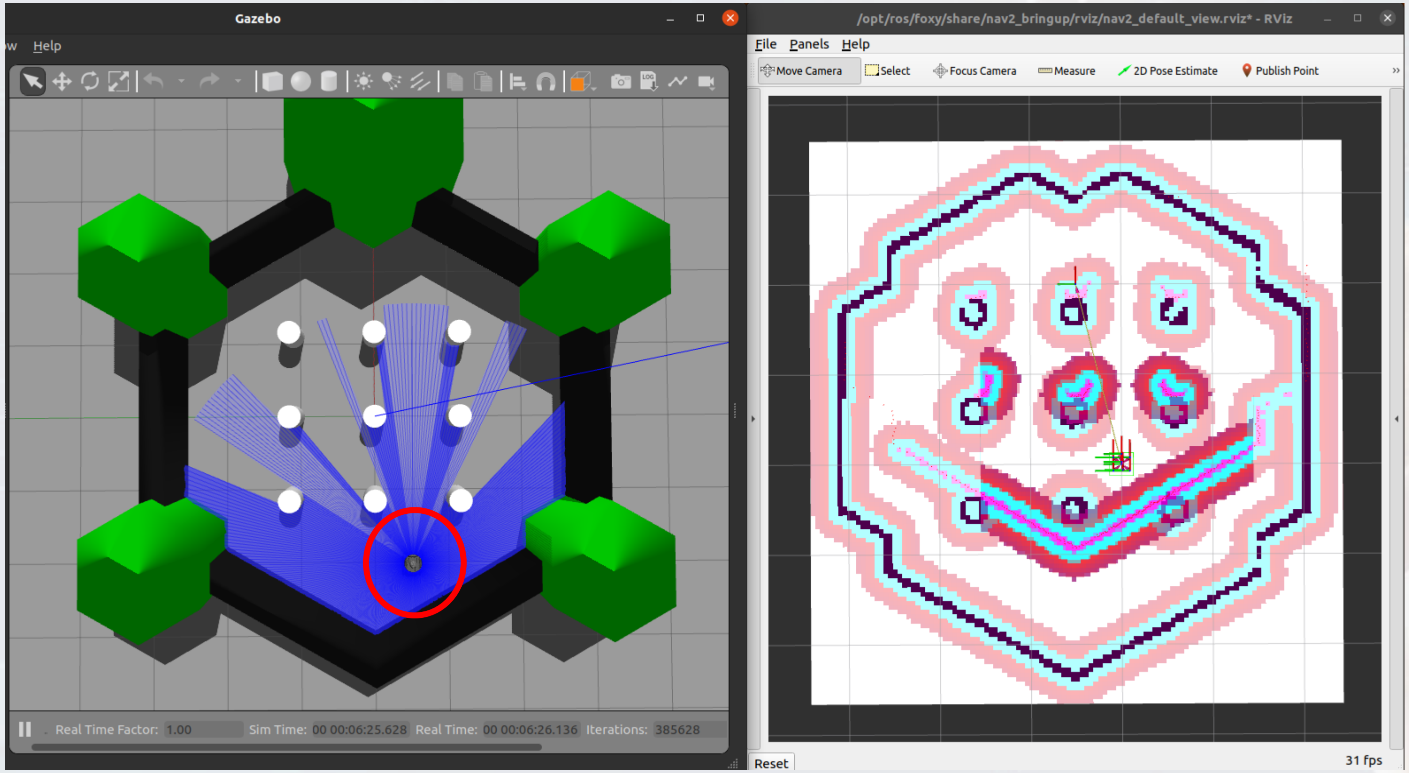
Real Time Factor: 1.00 Sim Time: 00 00:06:25.628 Real Time: 00 00:06:26.136 Iterations: 385628



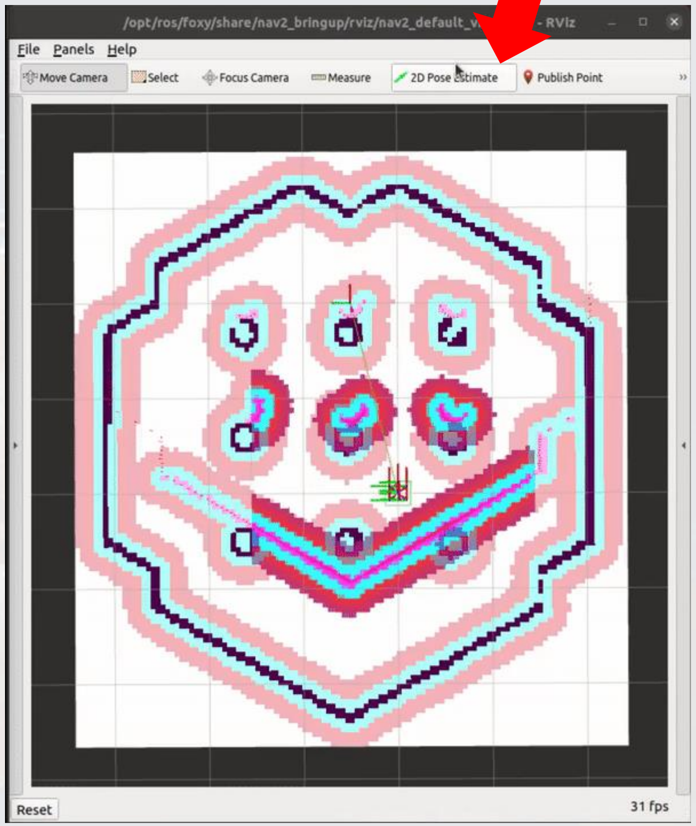
Start point will be as same as where you start the SLAM node.

Re-position using 2D Pose Estimate

- You can see that if the start position of iron-X's model not the same as gazebo. So, we can use **"2D Pose Estimate"** to re-position it:

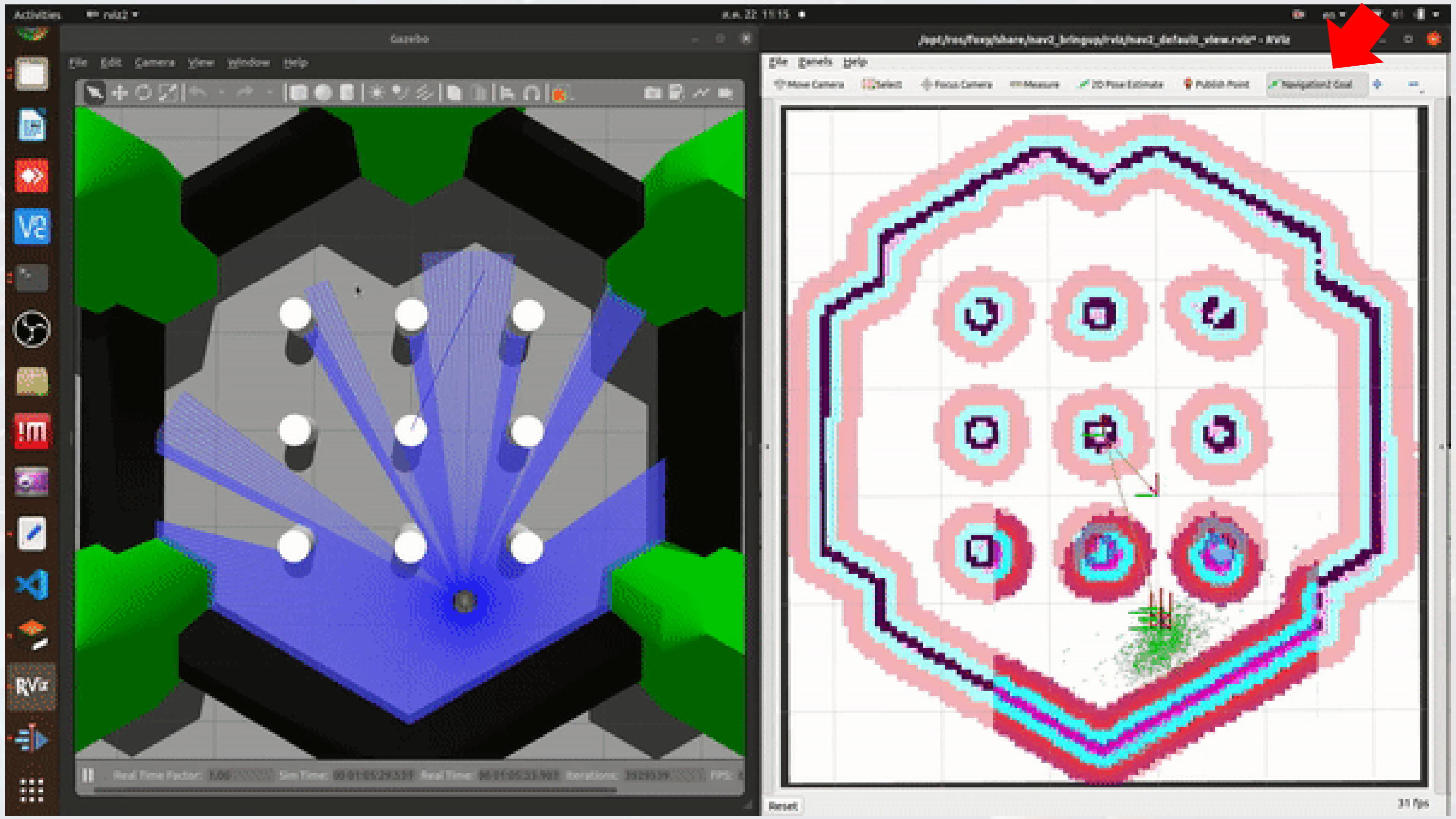


Real Time Factor: 1.00 Sim Time: 00 00:06:25.628 Real Time: 00 00:06:26.136 Iterations: 385628



Navigation2 Goal

- And then, you can publish the goal for navigation using **"Navigatoin2 Goal"**:

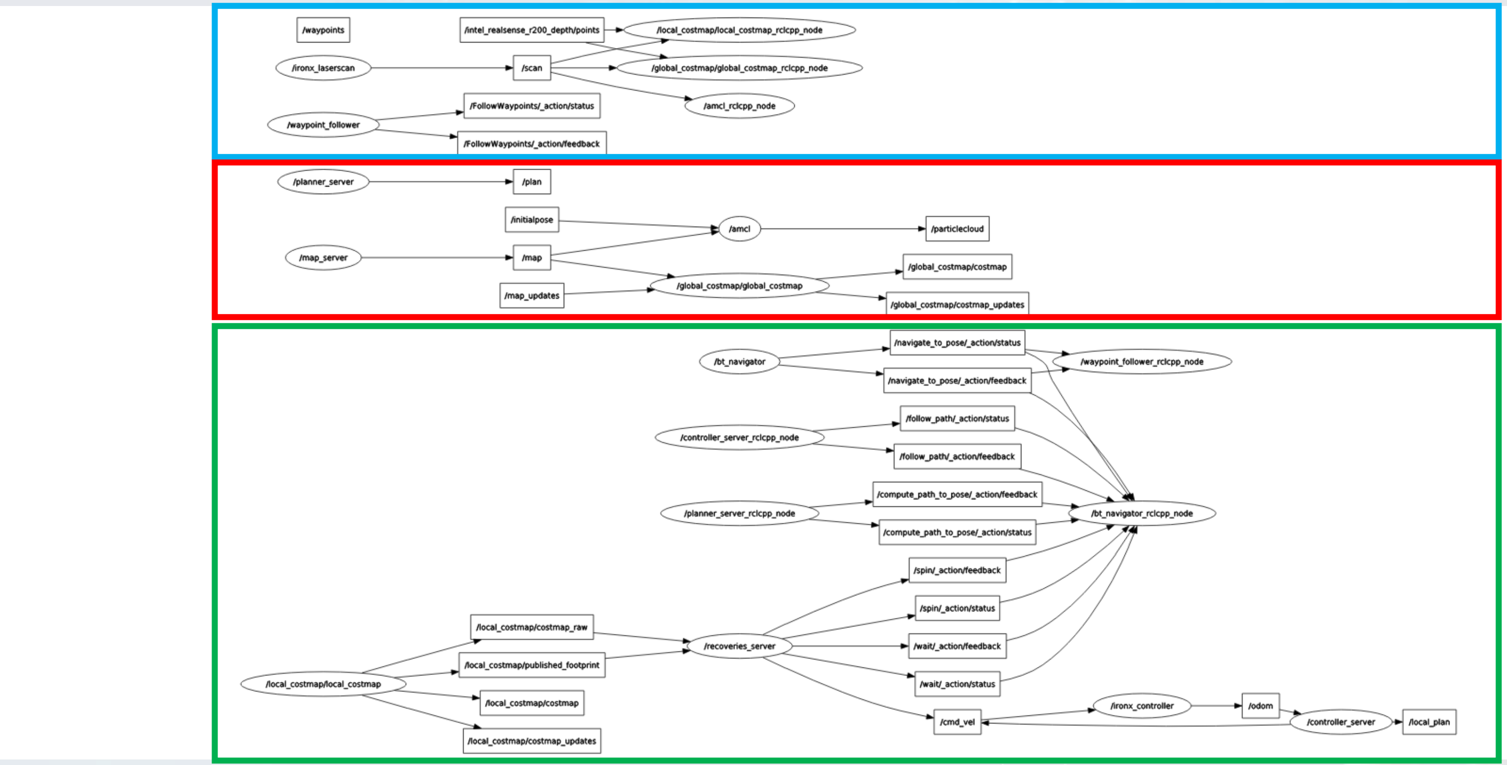


***Speed x4**

Navigation's RosGraph

- You can see RosGraph of Navigation using:

```
rqt_graph
```


/waypoints
/intel_realsense_r200_depth/points
/local_costmap/local_costmap_rclcpp_node
/ironx_laserscan
/scan
/global_costmap/global_costmap_rclcpp_node
/FollowWaypoints/_action/status
/amcl_rclcpp_node
/waypoint_follower
/FollowWaypoints/_action/feedback
/planner_server
/plan
/initialpose
/amcl
/particlecloud
/map_server
/map
/global_costmap/costmap
/map_updates
/global_costmap/global_costmap
/global_costmap/costmap_updates
/navigate_to_pose/_action/status
/bt_navigator
/waypoint_follower_rclcpp_node
/navigate_to_pose/_action/feedback
/follow_path/_action/status
/controller_server_rclcpp_node
/follow_path/_action/feedback
/compute_path_to_pose/_action/feedback
/planner_server_rclcpp_node
/bt_navigator_rclcpp_node
/compute_path_to_pose/_action/status
/spin/_action/feedback
/spin/_action/status
/local_costmap/costmap_raw
/recoveries_server
/local_costmap/published_footprint
/wait/_action/feedback
/local_costmap/local_costmap
/wait/_action/status
/local_costmap/costmap
/ironx_controller
/odom
/cmd_vel
/controller_server
/local_plan
/local_costmap/costmap_updates

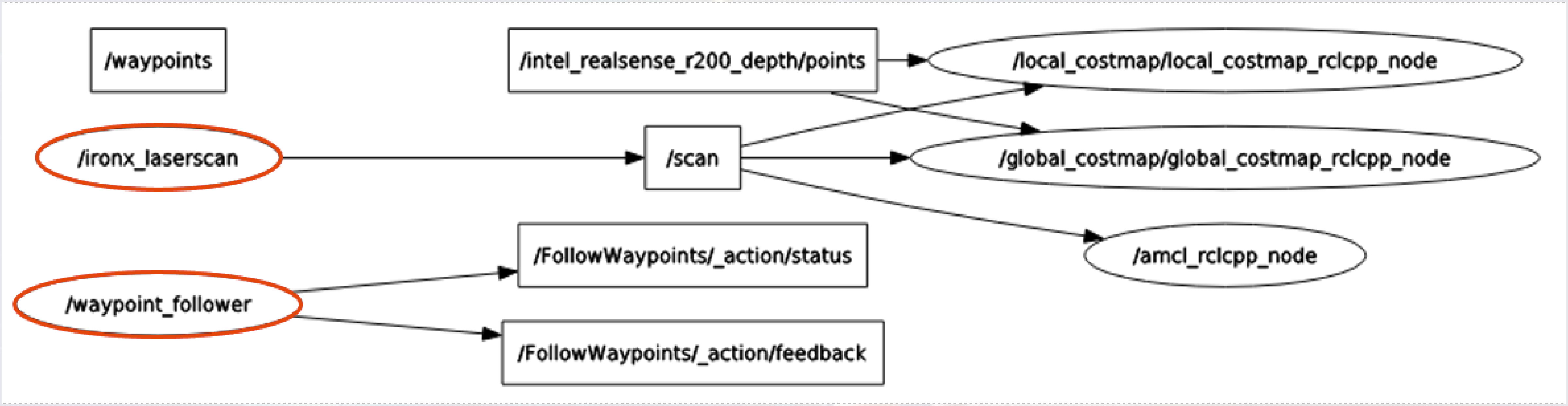
Scan and waypoint_follower

Map_server, amcl, planner_server and global costmap

Local costmap and navigation node server and local_plan

Navigation's RosGraph

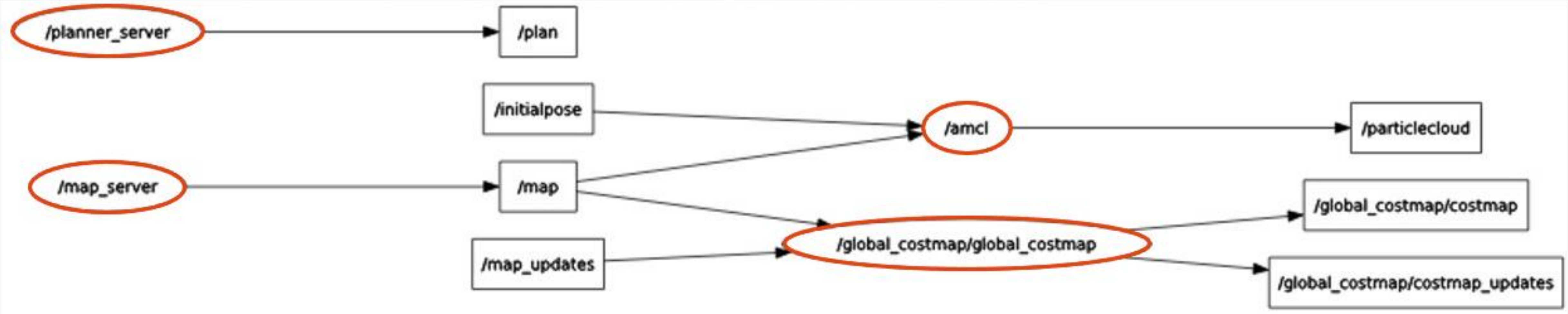
- **LaserScan and waypoint_follower**



/waypoints
/intel_realsense_r200_depth/points
/local_costmap/local_costmap_rclcpp_node
/ironx_laserscan
/scan
/global_costmap/global_costmap_rclcpp_node
/FollowWaypoints/_action/status
/amcl_rclcpp_node
/waypoint_follower
/FollowWaypoints/_action/feedback

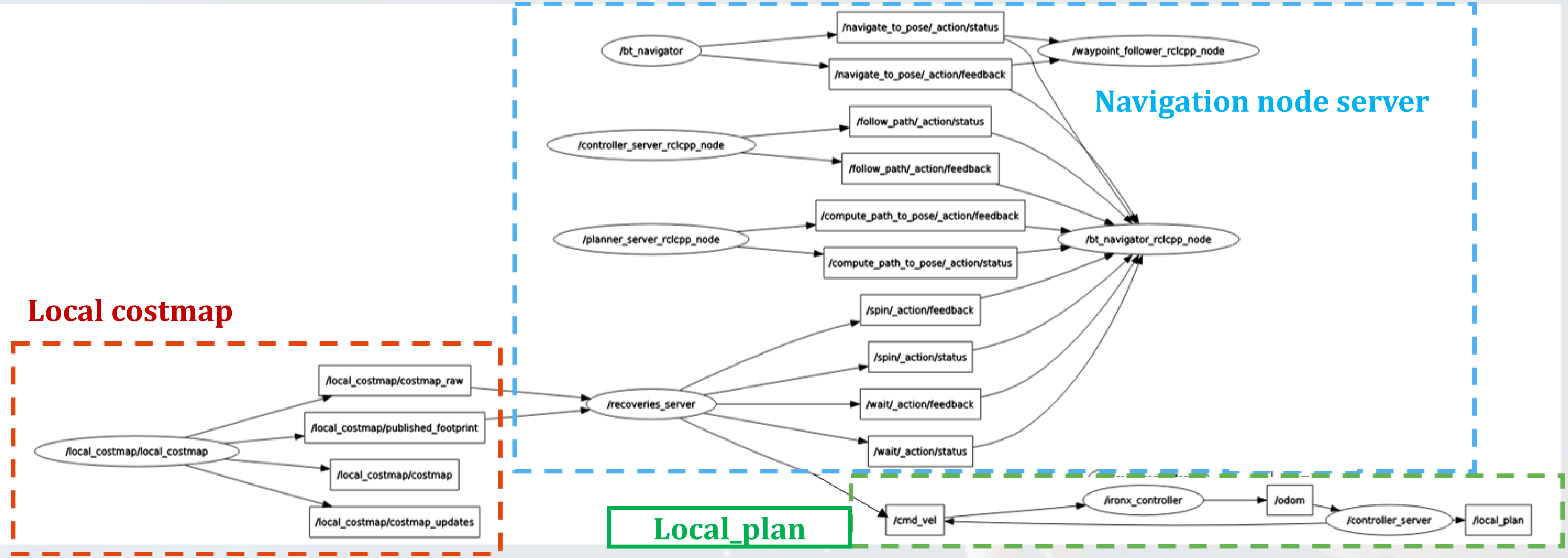
Navigation's RosGraph

- **Map_server, amcl, planner_server and global_costmap**



Navigation's RosGraph

- **Local costmap and navigation node server and local_plan**


/bt_navigator
/navigate_to_pose/_action/status
/navigate_to_pose/_action/feedback
/waypoint_follower_rclcpp_node
/controller_server_rclcpp_node
/follow_path/_action/status
/follow_path/_action/feedback
/planner_server_rclcpp_node
/compute_path_to_pose/_action/feedback
/compute_path_to_pose/_action/status
/bt_navigator_rclcpp_node
/spin/_action/feedback
/spin/_action/status
/wait/_action/feedback
/wait/_action/status
/recoveries_server
/local_costmap/costmap_raw
/local_costmap/published_footprint
/local_costmap/local_costmap
/local_costmap/costmap
/local_costmap/costmap_updates
/cmd_vel
/ironx_controller
/odom
/controller_server
/local_plan

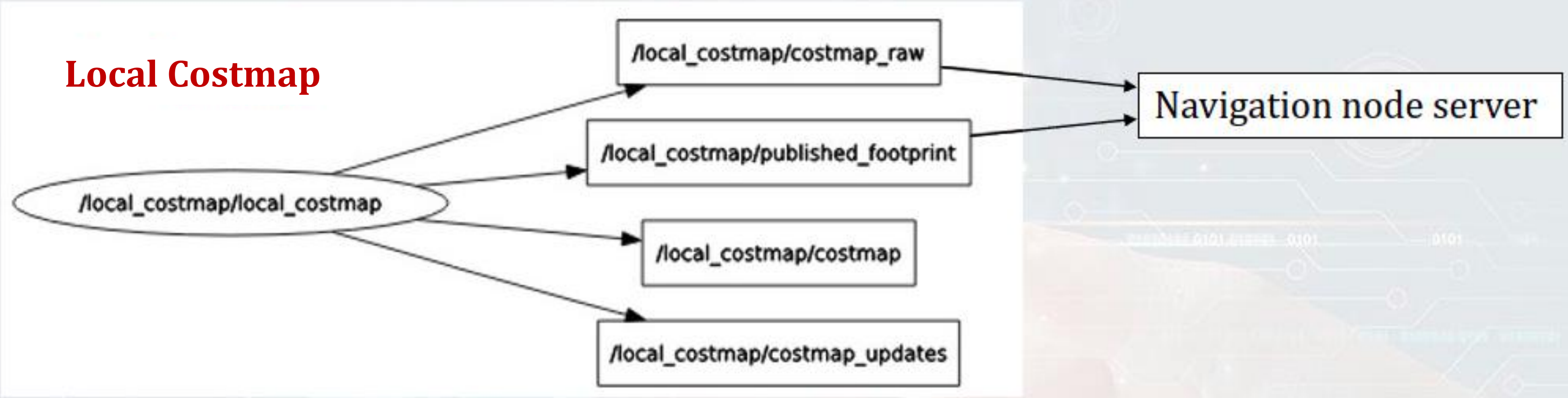
Navigation node server

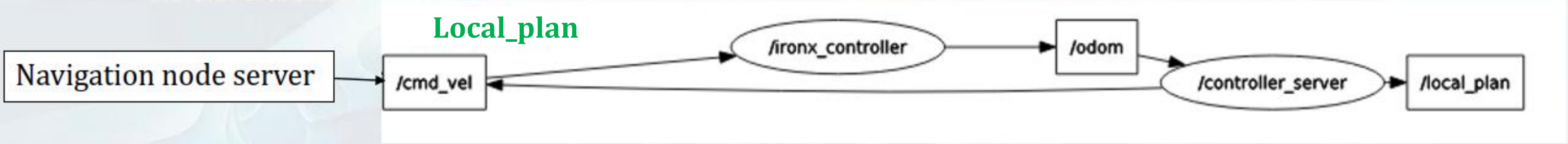
Local costmap

Local_plan

Navigation's RosGraph

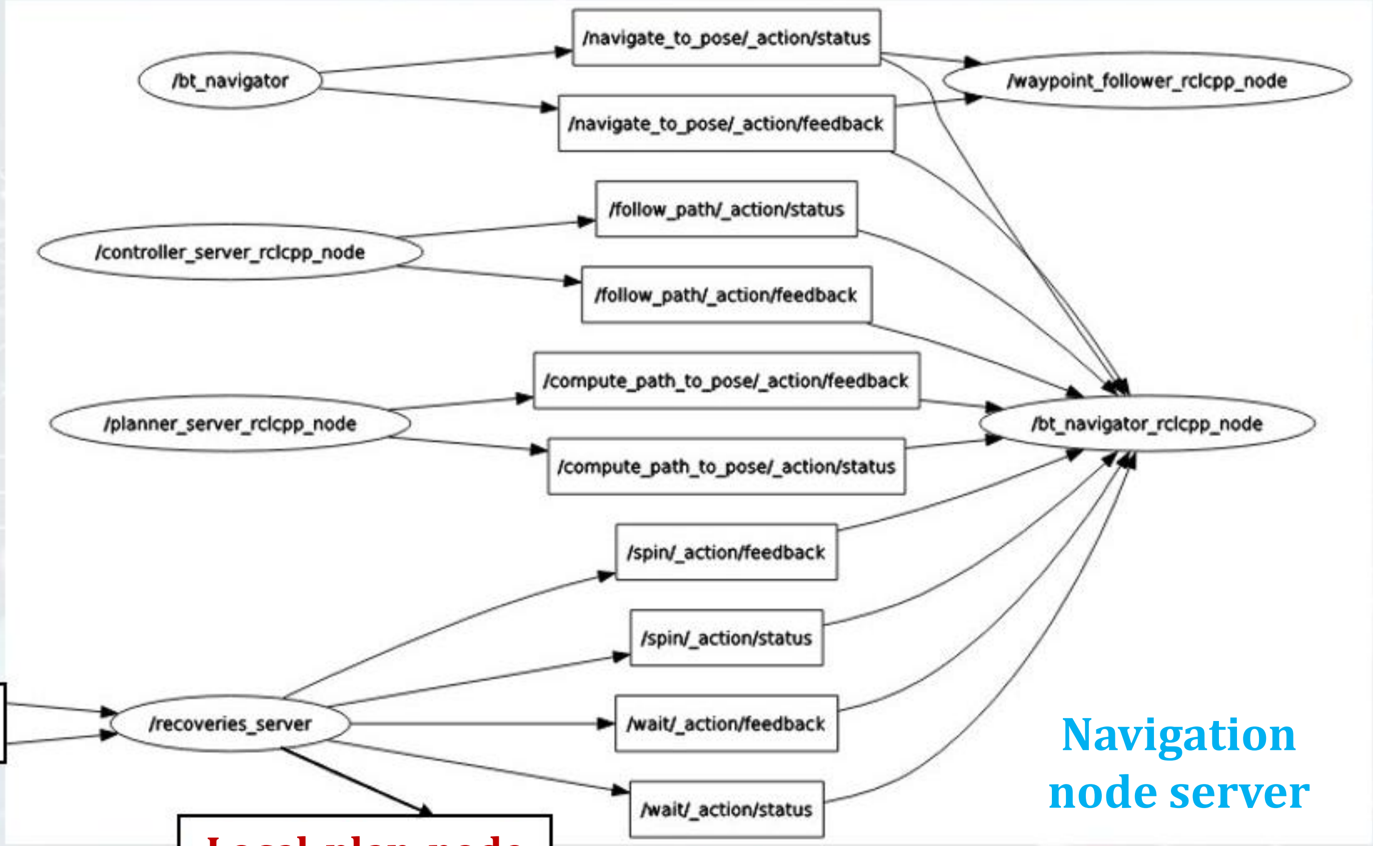
- **Local Costmap and local_plan**


Local Costmap
/local_costmap/local_costmap
/local_costmap/costmap_raw
/local_costmap/published_footprint
/local_costmap/costmap
/local_costmap/costmap_updates
Navigation node server



Navigation's RosGraph

- **Navigation node server**


/bt_navigator
/navigate_to_pose/_action/status
/navigate_to_pose/_action/feedback
/waypoint_follower_rclcpp_node
/follow_path/_action/status
/controller_server_rclcpp_node
/follow_path/_action/feedback
/compute_path_to_pose/_action/feedback
/planner_server_rclcpp_node
/bt_navigator_rclcpp_node
/compute_path_to_pose/_action/status
/spin/_action/feedback
/spin/_action/status
/recoveries_server
/wait/_action/feedback
/wait/_action/status
Navigation node server

Local Costmap node

Local_plan_node

Contact Us

Email: tesrshop@gmail.com
Line official Account: @ion1900z
Facebook fanpage: TESR
Tel. 082-983-7768

Scan here

More

Thai Embedded

TESR Co., LTD

112/296 หมู่บ้าน เพอร์เฟค มาสเตอร์พีซ
หมู่ที่ 2 ตำบลไทรม้า อำเภอเมืองนนทบุรี
จังหวัดนนทบุรี 11000