ROS Lab 8

Vipul Dinesh, 220929024, MTE-A-09

1. Get the following required packages

- `sudo apt install ros-humble-navigation2 rod-humble-nav2-bringup “ros-humble-turtlebot3*” ros-humble-rmw-cyclonedds-cpp ros-humble-slam-toolbox`

2. Set turtlebot3 model to waffle. Then reopen a new terminal for the configuration changes to take effect.

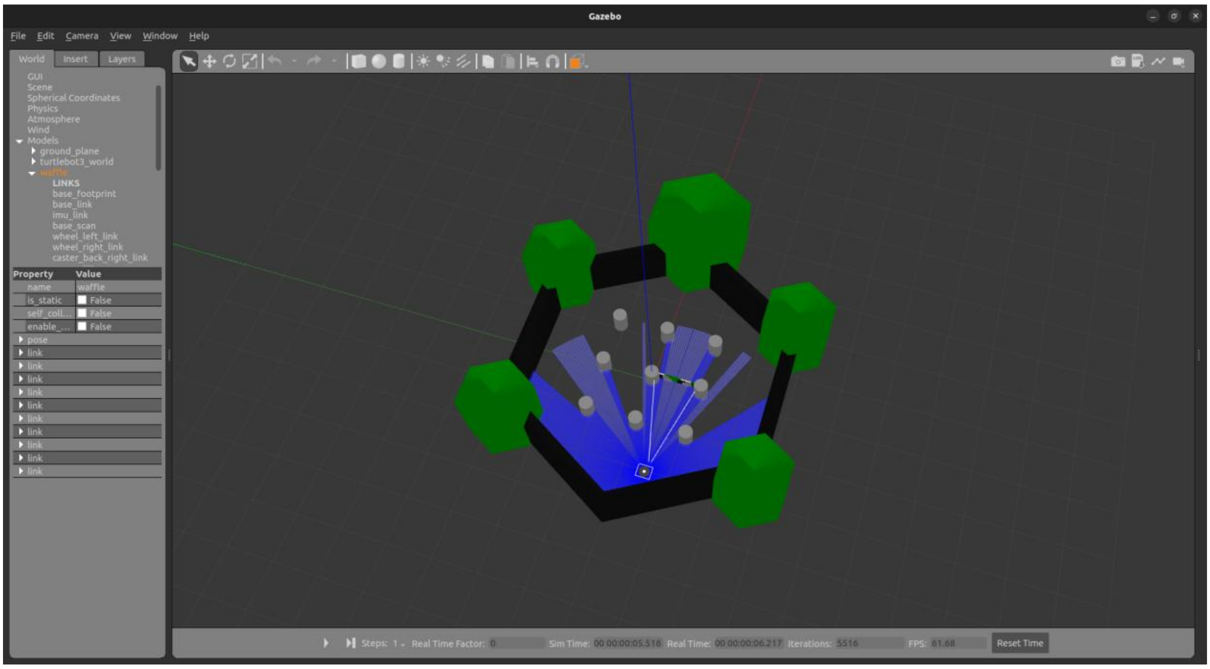
- `echo “export TURTLEBOT3_MODEL=waffle” >> ~/.zshrc`
- `echo “export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp” >> ~/.zshrc`

3. Generate a map for SLAM using cartographer

a. Launch turtlebot3 in gazebo with the world scene

`Terminal 1`

- `ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py`



Property	Value
name	waffle
is_static	□ False
self_coll...	□ False
enable_...	□ False

Steps: 1 Real Time Factor: 0 Sim Time: 00 00:00:05.516 Real Time: 00 00:00:06.217 Iterations: 5516 FPS: 61.68

b. Use cartographer to create a map for SLAM using laser data and odometry of the robot

`Terminal 2`

- `ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=true`

c. Use teleop to roam around the scene and generate data from the scanner which can be fed to cartographer to make a map

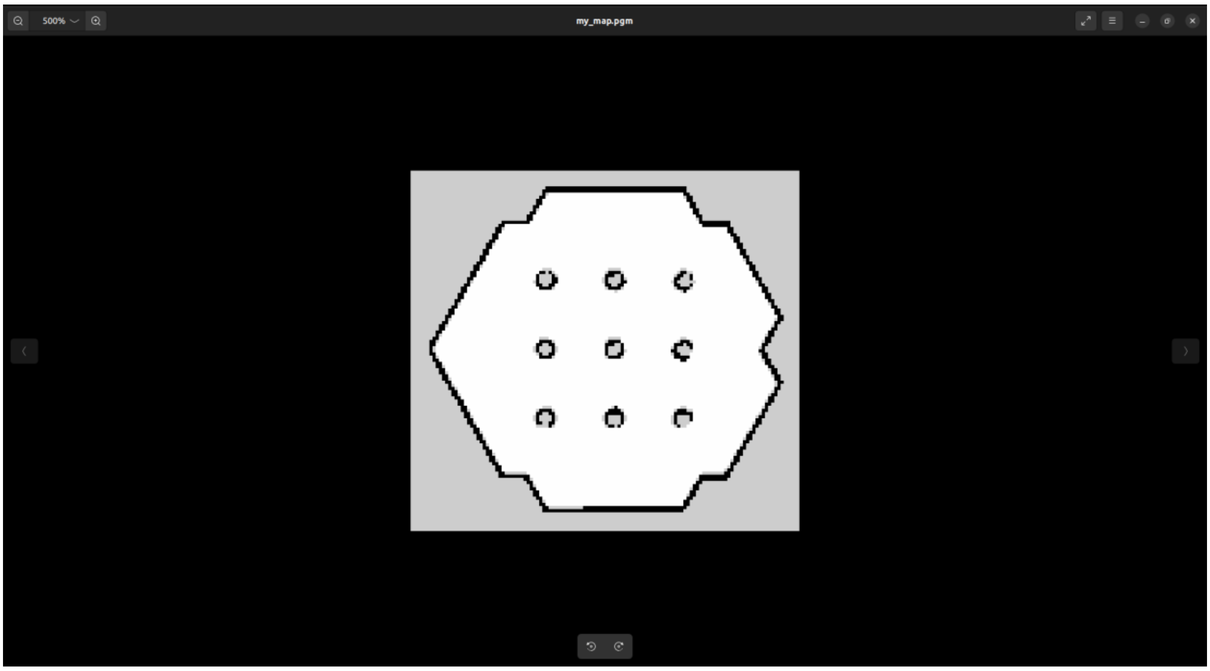
Terminal 3

- `ros2 run turtlebot3_teleop teleop_keyboard`

d. After the data has been generated as seen in rviz, save the map

Terminal 3

- `cd ~`
- `mkdir maps`
- `ros2 run nav2_map_server map_saver_cli -f maps/my_map`



e. Close all terminals using Ctrl+C

4. Use the generated map to navigate the bot from point A to point B

a. Make the following changes in waffle.yaml

- `sudo vim /opt/ros/humble/share/turtlebot3_navigation2/param/waffle.yaml`

```
#robot_model_type: "differential"
robot_model_type: "nav2_amcl::DifferentialMotionModel"
```

b. Launch turtlebot3 in gazebo with the world scene

Terminal 1

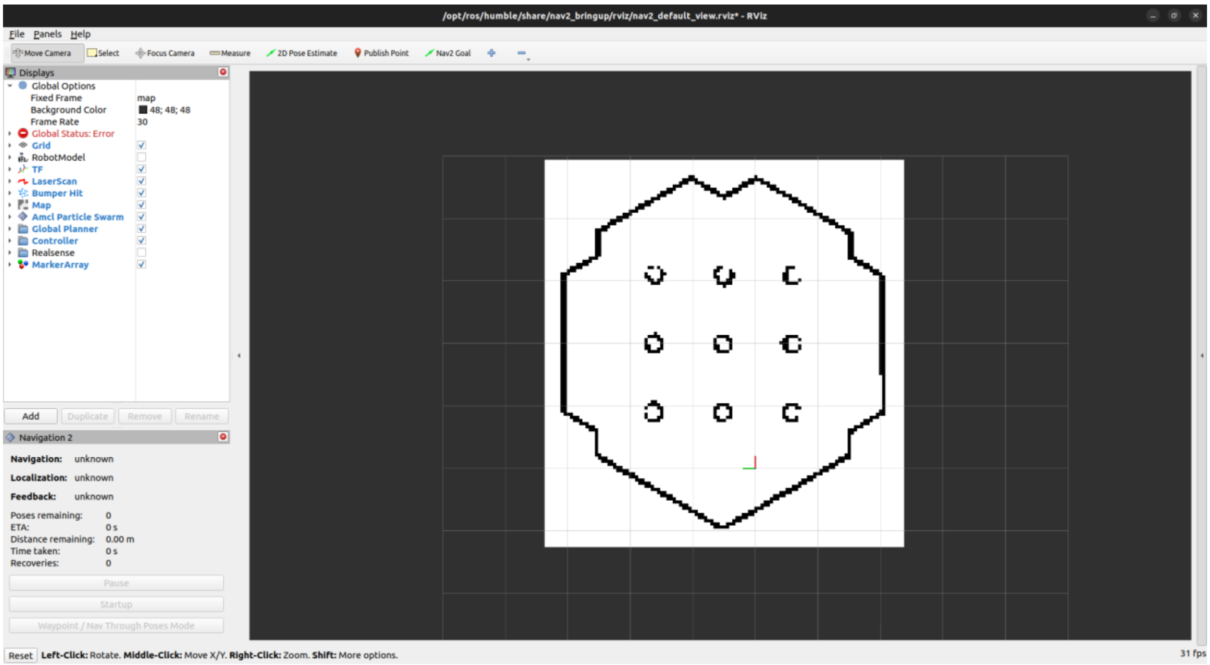
- `ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py`

c. Run navigation with my_map map

Terminal 2

- `cd ~`

- `ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=maps/my_map.yaml`



/opt/ros/humble/share/nav2_bringup/rviz/nav2_default_view.rviz* - RViz

Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Global Status: Error
Grid ☑
RobotModel □
TF ☑
LaserScan ☑
Bumper Hit ☑
Map ☑
Amcl Particle Swarm ☑
Global Planner ☑
Controller ☑
Realsense □
MarkerArray ☑

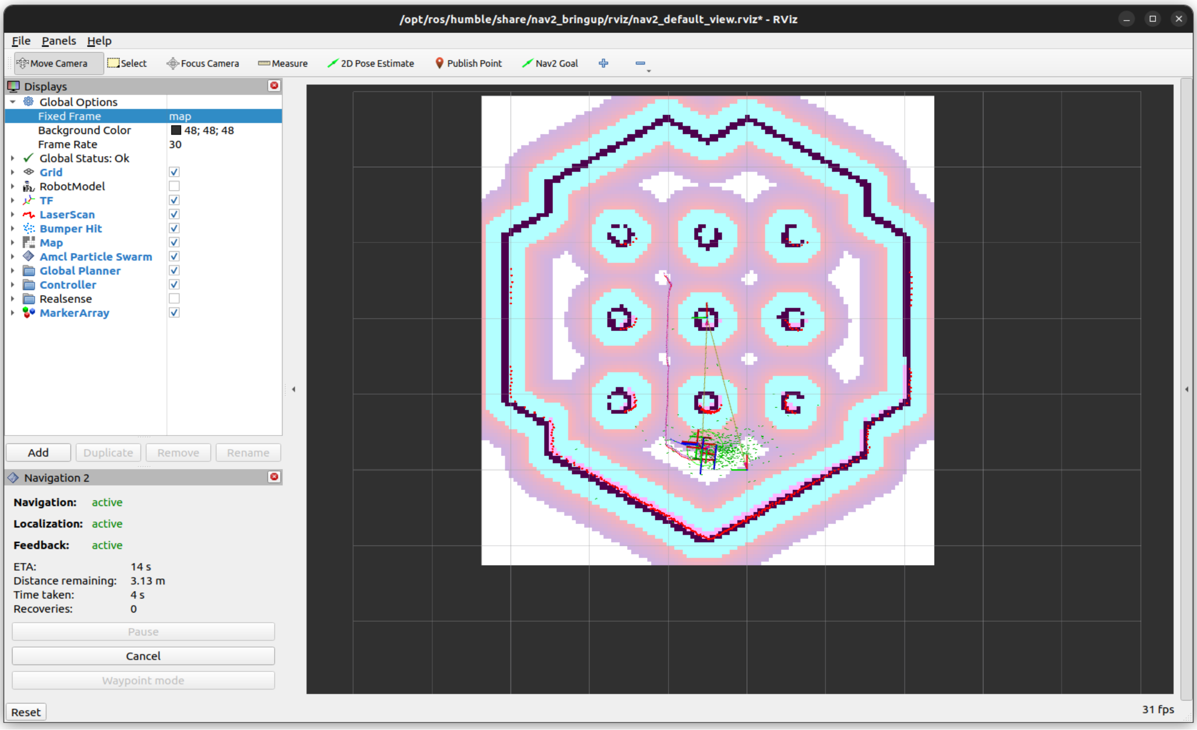
Navigation: unknown
Localization: unknown
Feedback: unknown
Poses remaining: 0
ETA: 0 s
Distance remaining: 0.00 m
Time taken: 0 s
Recoveries: 0

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

31 fps

d. Click on 2D Pose estimate and click on current position of robot

e. Click on Next Goal and choose destination to perform autonomous navigation



/opt/ros/humble/share/nav2_bringup/rviz/nav2_default_view.rviz* - RViz

Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Grid ☑
RobotModel □
TF ☑
LaserScan ☑
Bumper Hit ☑
Map ☑
Amcl Particle Swarm ☑
Global Planner ☑
Controller ☑
Realsense □
MarkerArray ☑

Navigation: active
Localization: active
Feedback: active
ETA: 14 s
Distance remaining: 3.13 m
Time taken: 4 s
Recoveries: 0

31 fps

5. Generate a map for SLAM using slam toolbox
   a. Launch turtlebot3 in gazebo with the world scene

Terminal 1

- `ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py`

b. Terminal 2

- `ros2 launch nav2_bringup navigation_launch.py use_sim_time:=True`

c. Terminal 3

- `ros2 launch slam_toolbox online_async_launch.py use_sim_time:=True`

d. Terminal 4

- `rviz2`

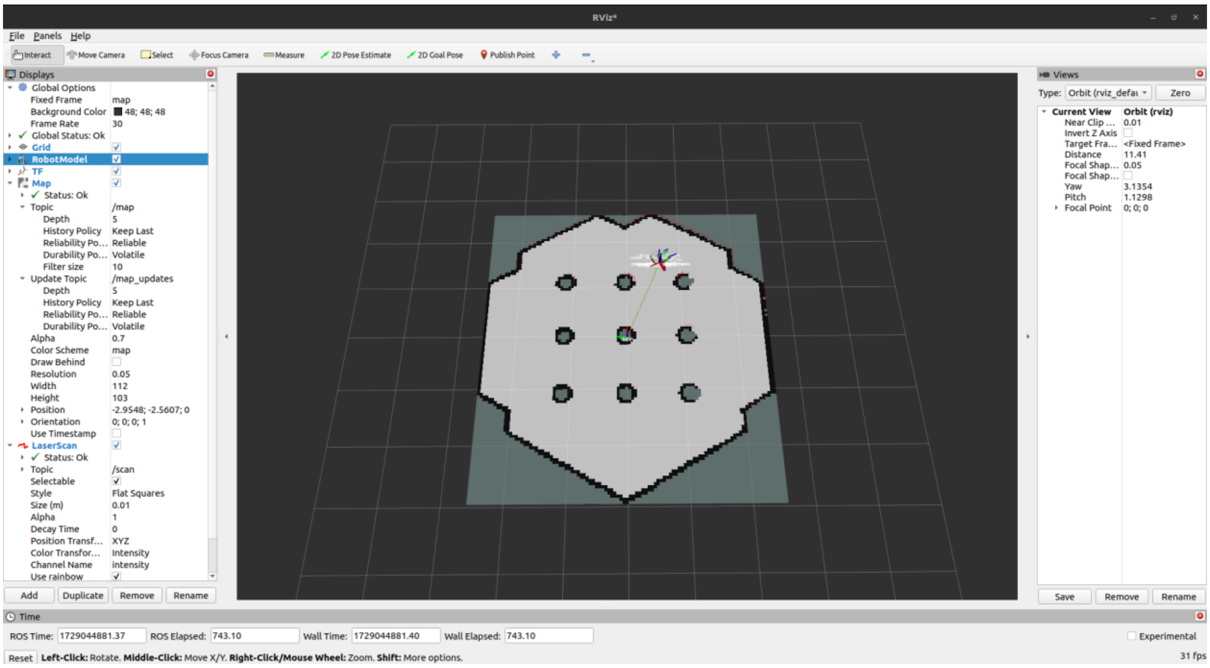
In RViz Window, Add:

- TF
- Laserscan
  - topic: /scan
- RobotModel
  - Description: /robot_description
- Map
  - topic: /map

e. Use teleop to roam around the scene and generate data from the scanner which can be fed to cartographer to make a map

Terminal 5

- `ros2 run turtlebot3_teleop teleop_keyboard`



Displays	
Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
Grid	☑
RobotModel	☑
TF	☑
Map	☑
✓ Status: Ok	
Topic	/map
Depth	5
History Policy	Keep Last
Reliability Po...	Reliable
Durability Po...	Volatile
Filter size	10
Update Topic	/map_updates
Depth	5
History Policy	Keep Last
Reliability Po...	Reliable
Durability Po...	Volatile
Alpha	0.7
Color Scheme	map
Draw Behind	□
Resolution	0.05
Width	112
Height	103
Position	-2.9548; -2.5607; 0
Orientation	0; 0; 0; 1
Use Timestamp	□
LaserScan	☑
✓ Status: Ok	
Topic	/scan
Selectable	☑
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transf...	XYZ
Color Transfor...	Intensity
Channel Name	intensity
Use rainbow	☑

Type: Orbit (rviz_defau Zero

Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	□
Target Fra...	<Fixed Frame>
Distance	11.41
Focal Shap...	0.05
Focal Shap...	□
Yaw	3.1354
Pitch	1.1298
Focal Point	0; 0; 0

ROS Time: 1729044881.37 ROS Elapsed: 743.10 Wall Time: 1729044881.40 Wall Elapsed: 743.10

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

f. After the data has been generated as seen in rviz, save the map

Terminal 6

- `cd ~`

- `ros2 run nav2_map_server map_saver_cli -f maps/my_world`

g. Save this rviz config as `map.rviz` in the ~/maps folder

h. Close all terminals using Ctrl+C

6. Use the generated map to navigate the bot from point A to point B

a. Launch turtlebot3 in gazebo with the world scene

Terminal 1

- `ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py`

b. Run navigation with my_world map

Terminal 2

- `cd ~`
- `ros2 launch nav2_bringup bringup_launch.py use_sim_time:=True`
- `map:=maps/my_world.yaml`

c. Terminal 4

```
ros2 run rviz2 rviz2
```

In RViz Window, Add:

- TF
- Laserscan
  - topic: /scan
- RobotModel
  - Description: /robot_description
- Map
  - topic - /map
- Map - Rename to GlobalCostmap
  - topic : global_costmap
  - color scheme: costmap
- Map - Rename to LocalCostmap
  - topic : local_costmap
  - color scheme: costmap

d. Save this rviz config as `map2.rviz` in the ~/maps folder

e. Following the steps from Step 4, use RViz's 2D Pose Estimate and 2D Goal Point for autonomous navigation