The Navigation Stack : Nav2

Create the base layer super fast
Provide a standard for robotics applications
Use on any robots
Allows you to avoid reinventing the wheel
Open source community
Plug and Play packages

ROS Stack:

Move a robot to reach the goal location without colliding the obstacles

Step 1: Create a map (SLAM)
Step 2: Make the robot navigate from Point A to Point B

sudo apt update
sudo apt install ros-humble-navigation2 ros-humble-nav2-bringup ros-humble-turtlebot3*
sudo apt install git
sudo apt install terminator
sudo snap install code --classic

Extensions: ROS (microsoft)
Cmake (twxs)

Generate a map and save the map

SLAM – Simultaneous Localization And Mapping

gedit ~/.bashrc
export TURTLEBOT3_MODEL=waffle

Terminal 1:

printenv | grep TURTLE

gazebo
check the FPS

Terminal 1:
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
Terminal 2:
ros2 run turtlebot3_teleop teleop_keyboard
Terminal 3:
ros2 topic list
/camera/camera_info
/camera/image_raw

/camera/image_raw/compressed
/camera/image_raw/compressedDepth
/camera/image_raw/theora
/clock
/cmd_vel
/imu
/joint_states
/odom
/parameter_events
/performance_metrics
/robot_description
/rosout
/scan
/tf
/tf_static

rqt_graph

Kill all the windows using Ctrll+C

Step 1: Generate map using SLAM

Terminal 1:
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

Terminal 2:
ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=true

Terminal 3:
ros2 run turtlebot3_teleop teleop_keyboard

Terminal 4:
(in the home location)
mkdir maps
ros2 run nav2_map_server map_saver_cli -f maps/my_map

Stop all using Ctrl + C

cd maps
nano my_map.pgm

Step 2. Navigate the robot from point A to point B

sudo apt install ros-humble-rmw-cyclonedds-cpp
gedit ~/.bashrc
export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
save and exit

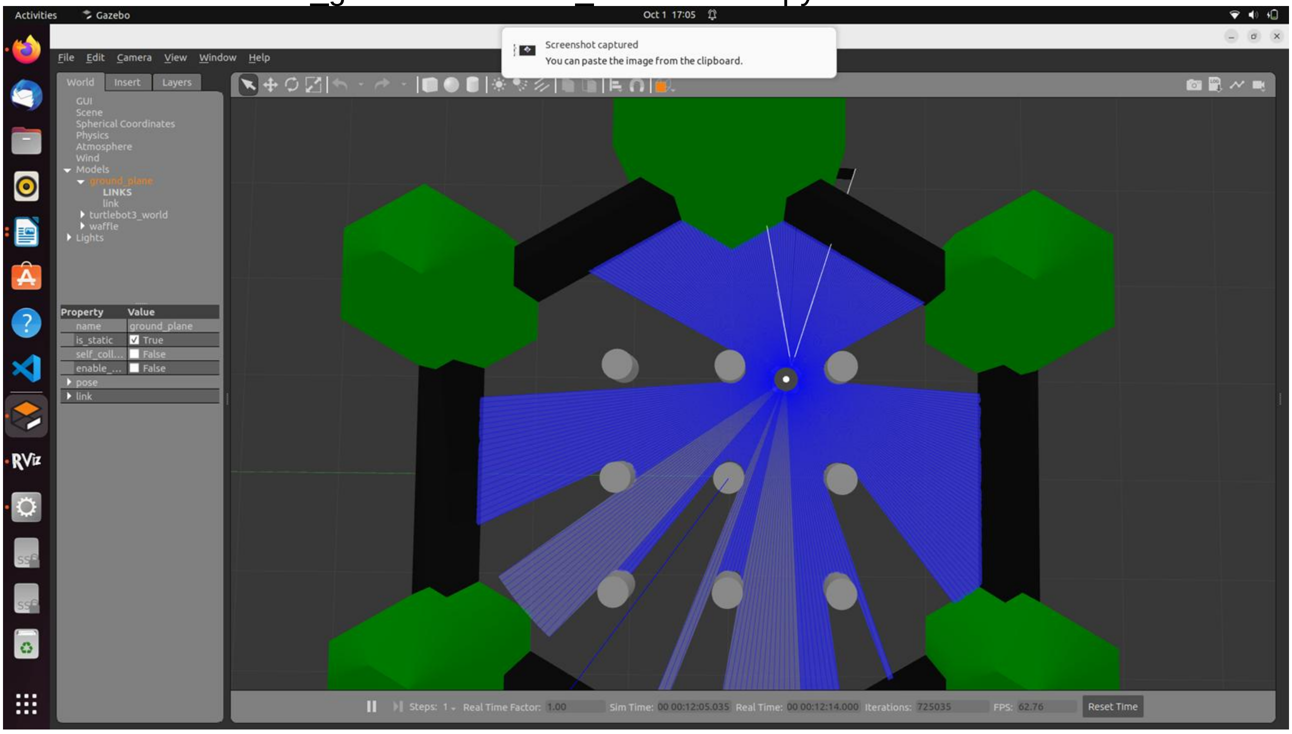
Terminal 1:

cd /opt/ros/humble/share/turtlebot3_navigation2/param
sudo gedit waffle.yaml

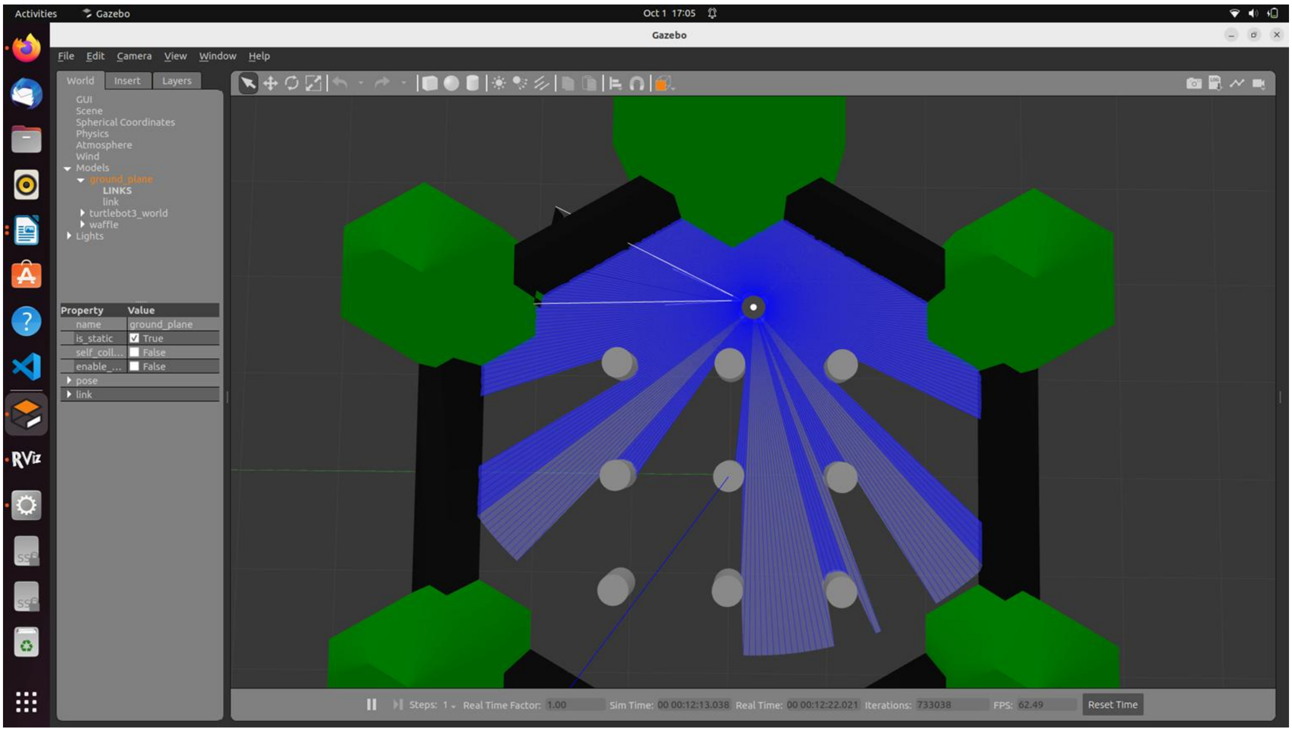
robot_model_type:"nav2_amcl: DifferentialMotionModel"
save and exit

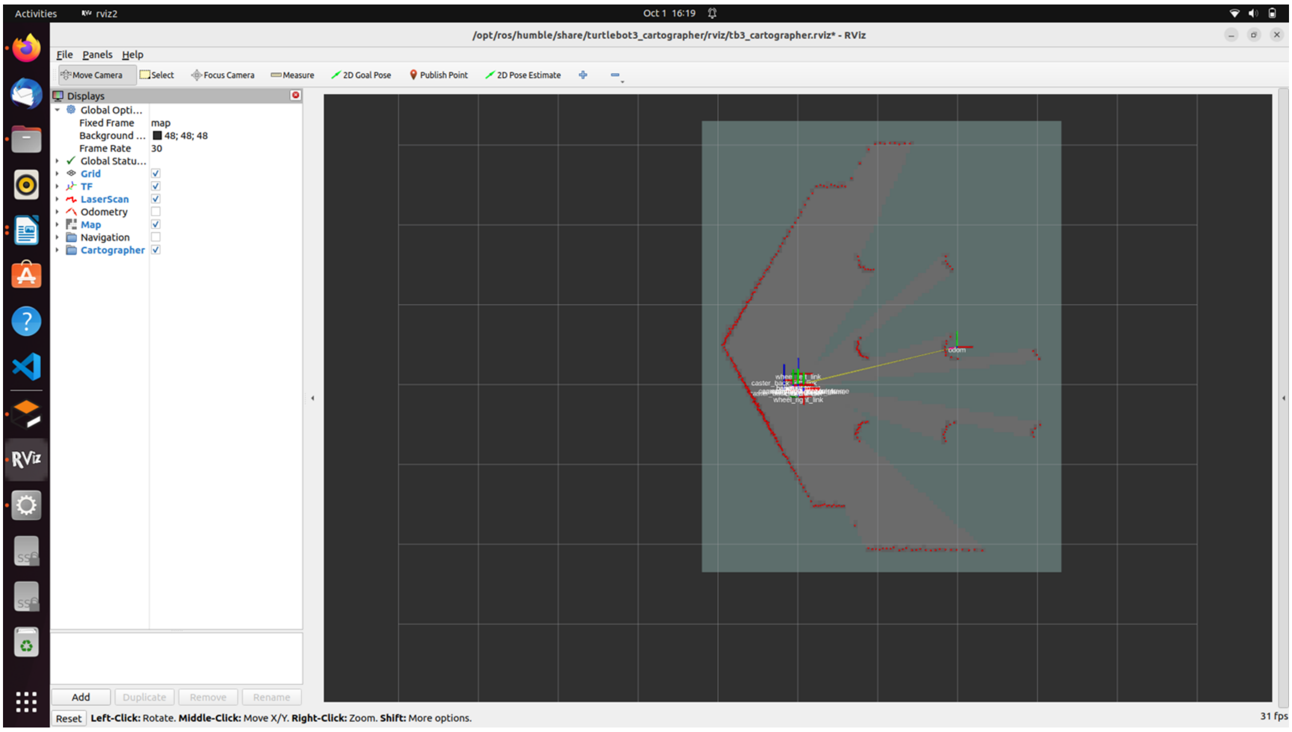
Terminal 1:

ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py



Terminal 2:
(run in the home location)
ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=maps/my_map.yaml


Activities
Gazebo
Oct 1 17:05
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
LINKS
link
turtlebot3_world
waffle
Lights
Property Value
name ground_plane
is_static True
self_coll... False
enable_... False
pose
link
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:12:13.038 Real Time: 00 00:12:22.021 Iterations: 733038 FPS: 62.49 Reset Time

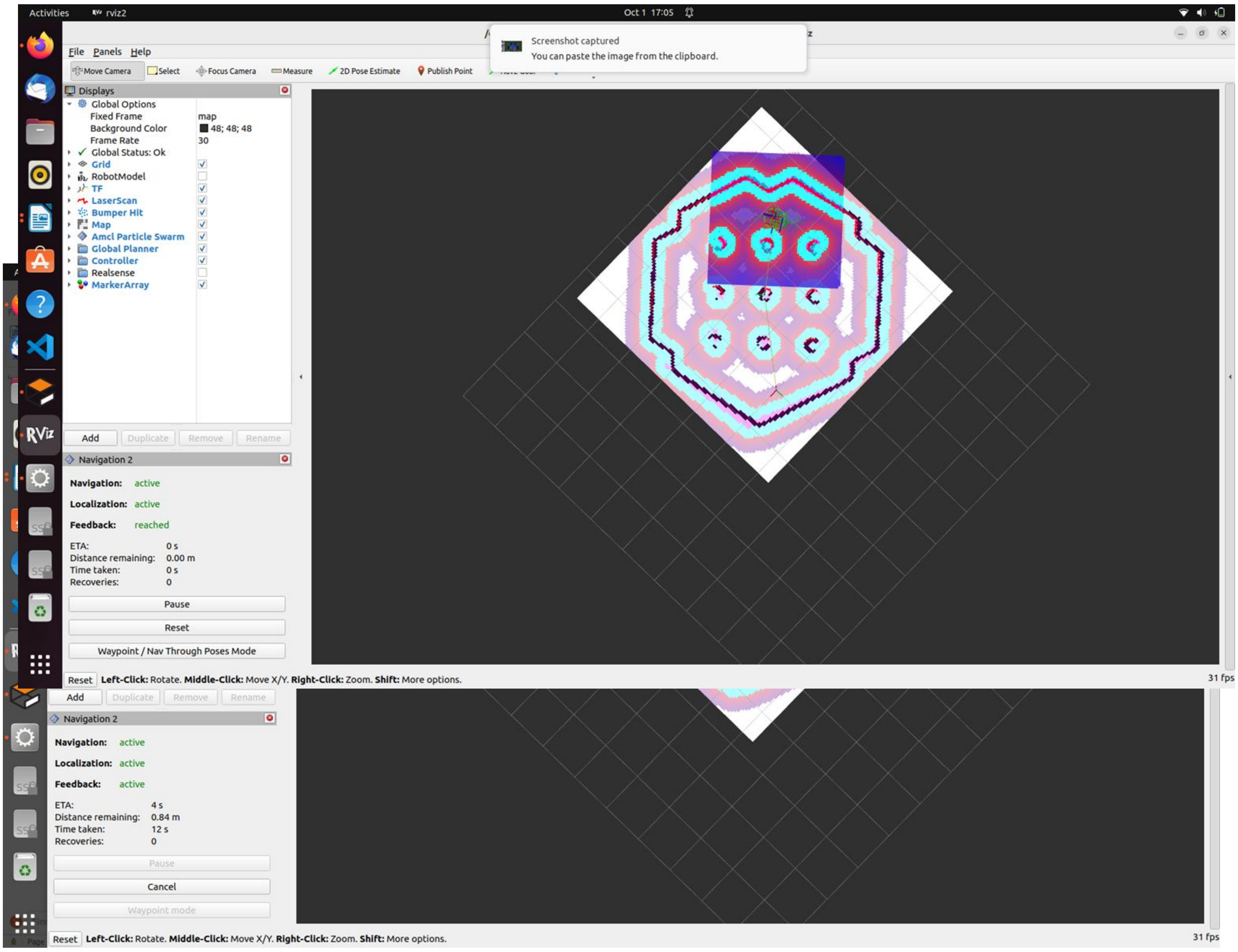


sudo apt install ros-humble-slam-toolbox

Terminal 1:
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
Terminal 2:
ros2 topic list
see the topic /scan

Terminal 3:
ros2 launch nav2_bringup navigation_launch.py use_sim_time:=True
Terminal 4:
ros2 launch slam_toolbox online_async_launch.py use_sim_time:=True

Terminal 5:
rviz2



Add TF
- Map
  - topic : /map
- Lazerscan
  - topic: /scan
- RobotModel
  - Description: /robot_description

Terminal 6:
ros2 run turtlebot3_teleop teleop_keyboard
Terminal 7:
ros2 run nav2_map_server map_saver_cli -f maps/my_world

cd maps
save config as map.rviz

stop everything by pressing ctrl + c

Terminal 1:

ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

Terminal 2:
ros2 launch nav2_bringup bringup_launch.py use_sim_time:=True map:=maps/my_world.yaml

Terminal 3:
ros2 run rviz2 rviz2
add
- LazerScan
  - topic: /scan
- TF
- RobotModel
  - Description: /robot_description
- Map
  - topic: /map
    - change volatile to Transient_Local
- Map: rename as GlobalCostmap
  - topic : global_costmap
  - color scheme: costmap

- Map: rename as LocalCostmap
  - topic : local_costmap
  - color scheme: costmap

Save config as map2.rviz

Exercise 1

SLAM and Navigation Steps for Any Robot

Those commands are the general commands to run for SLAM and Navigation, when using any robot that is configured for the Nav2 stack.
Steps – SLAM

You will need to install the slam_toolbox package:

$ sudo apt install ros-humble-slam-toolbox
1. Start your robot
This will be specific to your own robot.
Example with simulation:
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

2. Start a Navigation launch file
$ ros2 launch nav2_bringup navigation_launch.py
(add use_sim_time:=True if using Gazebo)

3. Start SLAM with slam_toolbox
$ ros2 launch slam_toolbox online_async_launch.py
(add use_sim_time:=True if using Gazebo)
4. Start Rviz
$ ros2 run rviz2 rviz2
(you will need to configure RViz, follow the instructions in the video)

5. Generate and save your map
Make the robot move in the environment (specific to your own robot).
Example with simulation:
$ ros2 run turtlebot3_teleop teleop_keyboard
Save the map:
$ ros2 run nav2_map_server map_saver_cli -f ~/my_map

Steps – Navigation

1. Start your robot
This will be specific to your own robot.
Example with simulation:
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

2. Start the main Navigation2 launch file
$ ros2 launch nav2_bringup bringup_launch.py map:=path/to/map.yaml
(add use_sim_time:=True if using Gazebo)

3. Start RViz
$ ros2 run rviz2 rviz2
(you will need to configure RViz)

4. Send navigation commands
Use the “2D Pose Estimate” button to set the initial pose, and the “Nav2 Goal” button to send navigation goals.

Note: instead of using RViz to send commands, you can directly interact with the Nav2 interfaces in your own code, for example using the Simple Commander API