Multi-vehicle Navigation2 navigation

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check **[Must read before use]** to set the IP and ROS_DOMAIN_ID on the board. in addition. The two cars also need to set up additional command spaces. For the setting method, please refer to the section "**Setting the Namespace**" in "**Read Before Use**".

1、Program function description

After the program is started, the target points of the two cars can be given in rviz. After receiving the instruction, the two cars calculate the path based on their own posture and move to their destinations.

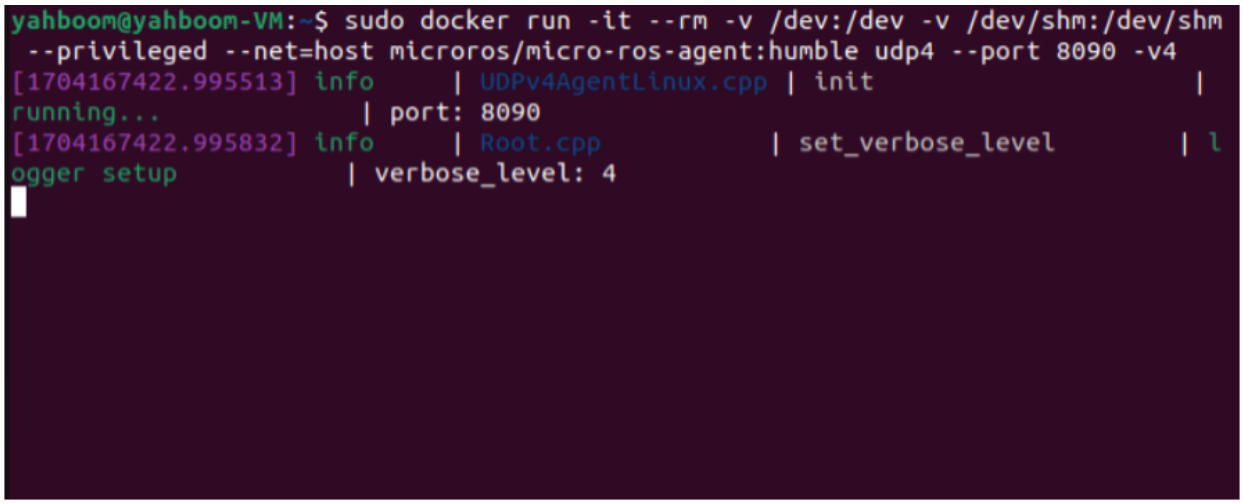
2、Multi-machine function basic settings

Taking two cars as an example, it is recommended to use two computers with matching virtual machines, change the config_robot.py files respectively, and set robot.set_ros_namespace() to robot1 and robot2 respectively. And **the ROS_DOMAIN_ID of the two cars and the ROS_DOMAIN_ID of the virtual machine need to be set to the same**. Then open the terminal in the /home/yahboom directory and enter `sudo python3 config_robot.py` to run this program (you need to change it back and re-run this program to run other programs except multi-car).

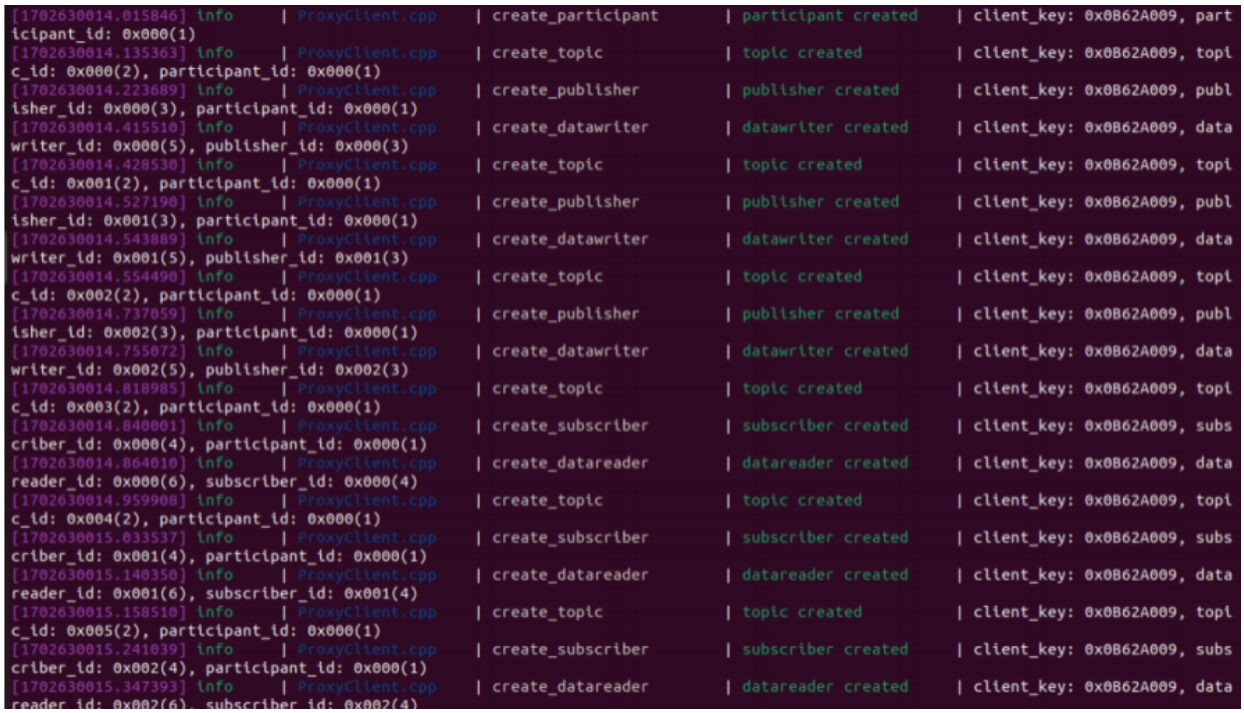
3、Start and connect to the agent

Taking the supporting virtual machine as an example, under the two virtual machines, enter the following commands to start the agents of the respective cars:

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host
microros/micro-ros-agent:humble udp4 --port 8090 -v4
```



Then, turn on the switches of the two cars and wait for the two cars to connect to their respective agents. The connection is successful and the terminal display is as shown in the figure below.



Check the currently started node. Select one of the two virtual machines, open the terminal and enter the following command:

```
ros2 node list
```

```
yahboom@yahboom-VM:~$ ros2 node list
/robot1/YB_Car_Node
/robot2/YB_Car_Node
yahboom@yahboom-VM:~$
```

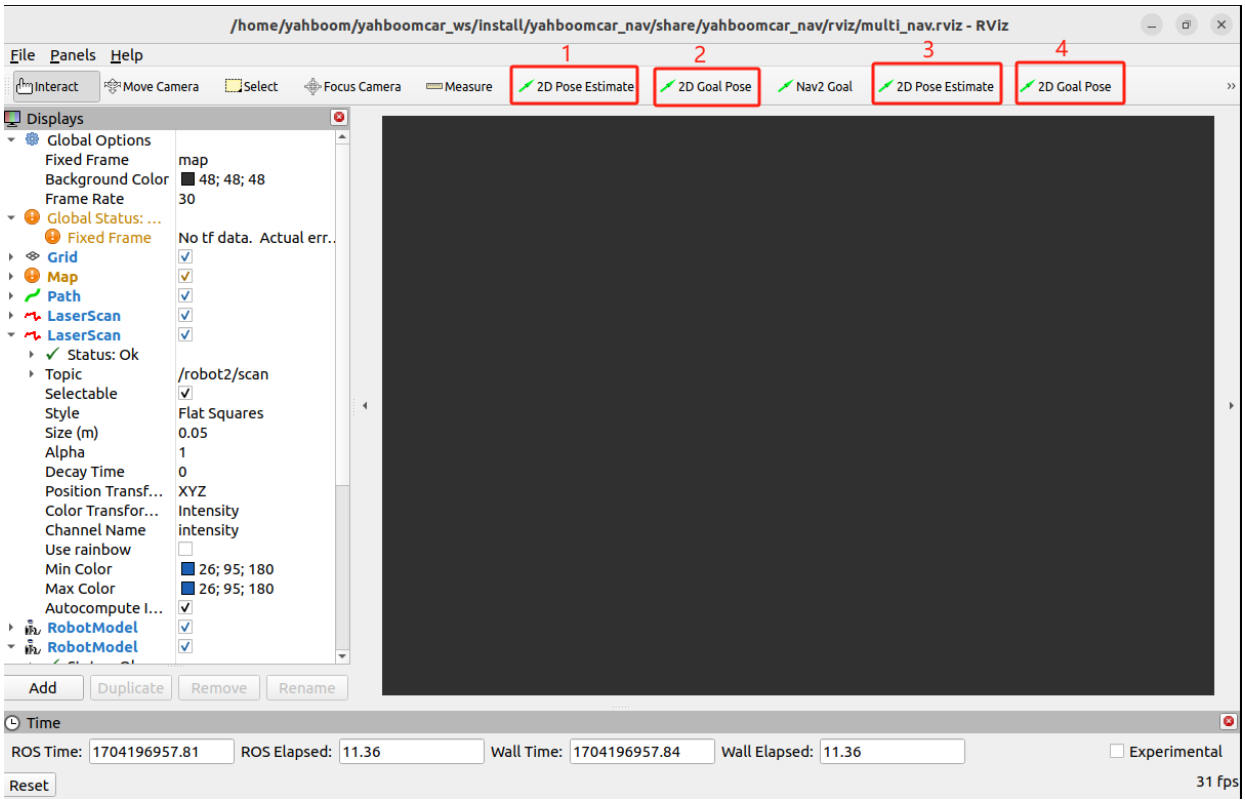
As shown in the picture above, the nodes of both cars have been started.

4、Start rviz and load the map

4.1、Start rviz display

Randomly select one of the two virtual machines, open the terminal and enter the following command:

```
ros2 launch yahboomcar_nav multi_nav_display_launch.py
```



The functions of the signs in Figure 1-4 above are as follows:

【1】: robot1 calibrate initial pose

【2】：robot1 gives a target point

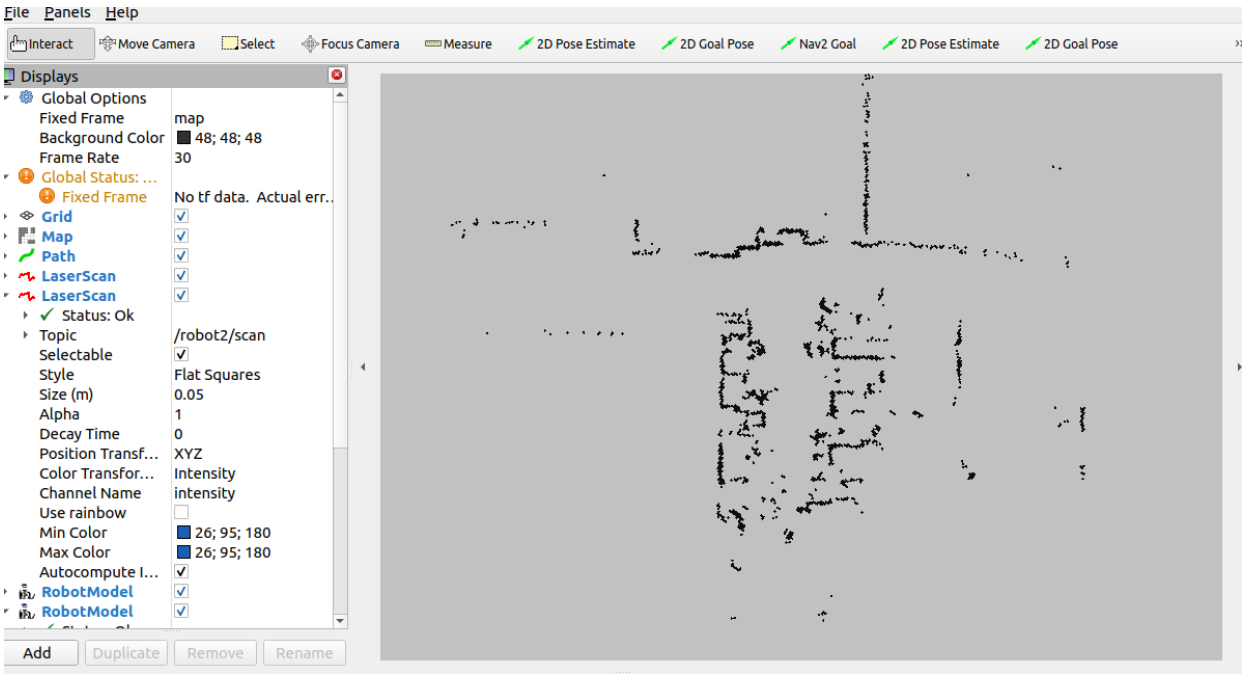
【3】：robot2 calibrate initial pose

【4】：robot2 given target point

4.2、Load map

Among the two virtual machines, select any one, open the terminal and enter

```
ros2 launch yahboomcar_multi map_server_launch.py
```



Note: The map may fail to load here. If the map is not loaded, just ctrl c to close and rerun the program. The map loaded here is.

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/maps/yahboom_map.yaml
```

If you need to modify the default loading of other maps, copy the map's yaml file and pgm file to

`/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/maps/` directory, and then modify the map_server_launch.py program.If you need to modify the default loading of other maps, copy the map's yaml file and pgm file to

`/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/maps/` directory, and then modify the map_server_launch.py program

Replace the area in the red box with the name of your own map, save and exit, and then enter the following instructions to compile,

```
cd ~/yahboomcar_ws
colcon build
```

Then enter the following command to resource the environment variables,

```
source ~/.bashrc
```

5、Start the car's underlying data processing program

Enter the following command in the virtual machine terminal that starts robot1

```
ros2 launch yahboomcar_multi yahboomcar_bringup_multi.launch.xml robot_name:=robot1
```

Enter the following command in the virtual machine terminal that starts robot2

```
ros2 launch yahboomcar_multi yahboomcar_bringup_multi.launch.xml robot_name:=robot2
```

The source code path of yahboomcar_bringup_multi.launch.xml (take the supporting virtual machine as an example).

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/yahboomcar_bringup_multi.lau
nch.xml
```

```
<launch>
    <arg name="robot_name" default="robot1"/>
    <group>
        <push-ros-namespace namespace="$(var robot_name)"/>
        <!--imu_filter_node-->
```

```
        <node name="imu_filter" pkg="imu_filter_madgwick"
exec="imu_filter_madgwick_node" output="screen">
            <param name="fixed_frame" value="$(var robot_name)/base_link"/>
            <param name="use_mag" value="false"/>
            <param name="publish_tf" value="false"/>
            <param name="world_frame" value="$(var robot_name)/enu"/>
            <param name="orientation_stddev" value="0.00"/>
            <remap from="imu/data_raw" to="imu"/>

        </node>
        <!--ekf_node-->
        <node name="ekf_filter_node" pkg="robot_localization" exec="ekf_node">
            <param name="odom_frame" value="$(var robot_name)/odom"/>
            <param name="base_link_frame" value="$(var robot_name)/base_footprint"/>
            <param name="world_frame" value="$(var robot_name)/odom"/>
            <param from="$(find-pkg-share yahboomcar_multi)/param/ekf_$(var
robot_name).yaml"/>
            <remap from="odometry/filtered" to="odom"/>
            <remap from="/odom_raw" to="odom_raw"/>
        </node>
        <node pkg="tf2_ros" exec="static_transform_publisher"
name="base_link_to_base_imu"
          args="-0.002999 -0.0030001 0.031701 0 0 0  $(var robot_name)/base_link
$(var robot_name)/imu_frame " />
    </group>
    <include file="$(find-pkg-share
yahboomcar_description)/launch/description_multi_$(var robot_name).launch.py"/>

</launch>
```

A pair of tags are used here. The command space of all programs within this tag will be robot_name, which is the robot1 or robot2 we defined. Among them, there are also some parameter files or topic names that are automatically selected and loaded through this robot_name. You can view the content in the code for details.

6、Start the AMCL positioning program of the car

Enter the following command in the virtual machine terminal that starts robot1:

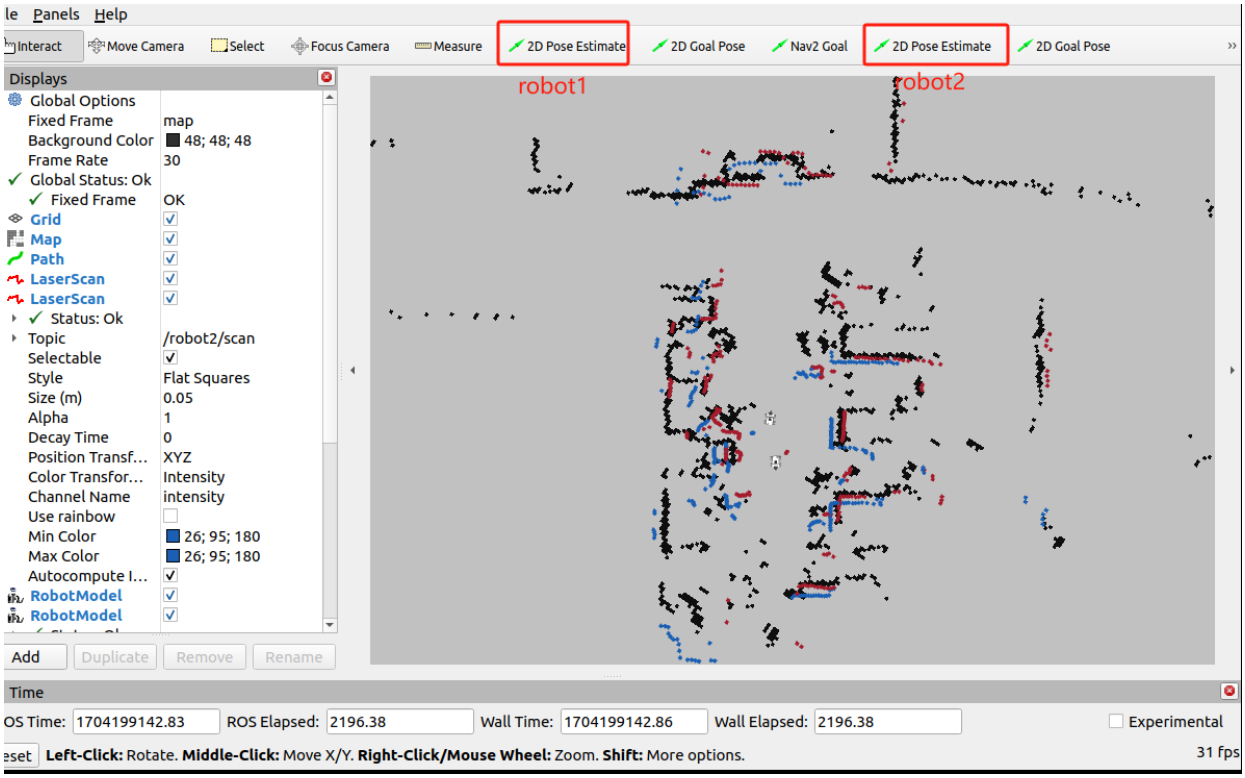
```
ros2 launch yahboomcar_multi robot1_amcl_launch.py
```

Enter the following command in the virtual machine terminal that starts robot2:

```
ros2 launch yahboomcar_multi robot2_amcl_launch.py
```

```
[INFO] [static_transform_publisher-3]: process started with pid [8261]
[static_transform_publisher-3] [WARN] [1704198745.682226809] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-3] [INFO] [1704198745.760571636] [robot2.base_link_to_base_laser]: Spinning until stopped - publishing t
ransform
[static_transform_publisher-3] translation: ('-0.004641', '0.000000', '0.094079')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'robot2/base_link' to 'robot2/laser_frame'
[lifecycle_manager-2] [INFO] [1704198745.764837773] [robot2_amcl_lifecycle_manager]: Creating
[lifecycle_manager-2] [INFO] [1704198745.769386287] [robot2_amcl_lifecycle_manager]: Creating and initializing lifecycle service cli
ents
[lifecycle_manager-2] [INFO] [1704198745.774042422] [robot2_amcl_lifecycle_manager]: Starting managed nodes bringup...
[lifecycle_manager-2] [INFO] [1704198745.774638030] [robot2_amcl_lifecycle_manager]: Configuring robot2_amcl
[amcl-1] [INFO] [1704198745.775423568] [robot2_amcl]:
[amcl-1]        robot2_amcl lifecycle node launched.
[amcl-1]        Waiting on external lifecycle transitions to activate
[amcl-1]        See https://design.ros2.org/articles/node_lifecycle.html for more information.
[amcl-1] [INFO] [1704198745.776357910] [robot2_amcl]: Creating
[amcl-1] [INFO] [1704198745.954733505] [robot2_amcl]: Configuring
[amcl-1] [INFO] [1704198745.954864171] [robot2_amcl]: initTransforms
[amcl-1] [INFO] [1704198745.960068087] [robot2_amcl]: initPubSub
[amcl-1] [INFO] [1704198745.963159185] [robot2_amcl]: Subscribed to map topic.
[lifecycle_manager-2] [INFO] [1704198745.965181911] [robot2_amcl_lifecycle_manager]: Activating robot2_amcl
[amcl-1] [INFO] [1704198745.965384016] [robot2_amcl]: Activating
[amcl-1] [INFO] [1704198745.965403773] [robot2_amcl]: Creating bond (robot2_amcl) to lifecycle manager.
[lifecycle_manager-2] [INFO] [1704198746.069306402] [robot2_amcl_lifecycle_manager]: Server robot2_amcl connected with bond.
[lifecycle_manager-2] [INFO] [1704198746.069355845] [robot2_amcl_lifecycle_manager]: Managed nodes are active
[lifecycle_manager-2] [INFO] [1704198746.069364357] [robot2_amcl_lifecycle_manager]: Creating bond timer...
[amcl-1] [WARN] [1704198748.123555121] [robot2_amcl]: Waiting for map....
[amcl-1] [INFO] [1704198748.742479904] [robot2_amcl]: Received a 960 X 800 map @ 0.050 m/pix
[amcl-1] [INFO] [1704198748.779290919] [robot2_amcl]: createLaserObject
[amcl-1] [WARN] [1704198750.161458790] [robot2_amcl]: AMCL cannot publish a pose or update the transform. Please set the initial pos
e...
[amcl-1] [WARN] [1704198752.179990112] [robot2_amcl]: AMCL cannot publish a pose or update the transform. Please set the initial pos
e...
```

As shown in the picture above, "Please set the initial pose..." appears. You can use the corresponding [2D Pose] tool to set the initial poses for the two cars respectively. According to the position of the car in the actual environment, click and drag with the mouse in rviz, and the car model moves to the position we set. As shown in the figure below, if the area scanned by the radar roughly coincides with the actual obstacle, it means that the pose is accurate.



Red represents robot1 and blue represents robot2.

Note: If "Please set the initial pose..." cannot be printed on the terminal, it may be due to the data timestamp being out of sync. Press the reset button of the car and let the car reconnect to the agent to ensure that the data timestamp is correct. , and try again several times until "Please set the initial pose..." appears.

Source code path location：/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch

robot1_amcl_launch.py

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node

def generate_launch_description():
    #package_path = get_package_share_directory('yahboomcar_multi')
    #nav2_bringup_dir = get_package_share_directory('nav2_bringup')
    lifecycle_nodes = ['map_server']
    param_file =
os.path.join(get_package_share_directory('yahboomcar_multi'),'param','robot1_amcl_pa
rams.yaml')
    amcl_node = Node(
        name="robot1_amcl",
        package='nav2_amcl',
        executable='amcl',
        parameters=
[os.path.join(get_package_share_directory('yahboomcar_multi'),'param','robot1_amcl_p
arams.yaml')],
        remappings=[('/initialpose', '/robot1/initialpose')],
        output = "screen"
    )

    life_node = Node(
        name="robot1_amcl_lifecycle_manager",
        package='nav2_lifecycle_manager',
        executable='lifecycle_manager',
        output='screen',
        parameters=[{'use_sim_time': False},{'autostart': True},{'node_names':
['robot1_amcl']}]
        )

    base_link_to_laser_tf_node = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        name='base_link_to_base_laser',
        namespace = 'robot1',
        arguments=['-0.0046412', '0' ,
'0.094079','0','0','0','robot1/base_link','robot1/laser_frame']
    )

    return LaunchDescription([
        #lifecycle_nodes,
        #use_sim_time,\
```

```
        amcl_node,
        life_node,
        base_link_to_laser_tf_node
    ])
```

amcl_node：Start the amcl node program, which is used to estimate pose and achieve positioning.

life_node：amcl node lifecycle manager

base_link_to_laser_tf_node：Static transformation of chassis and radar data

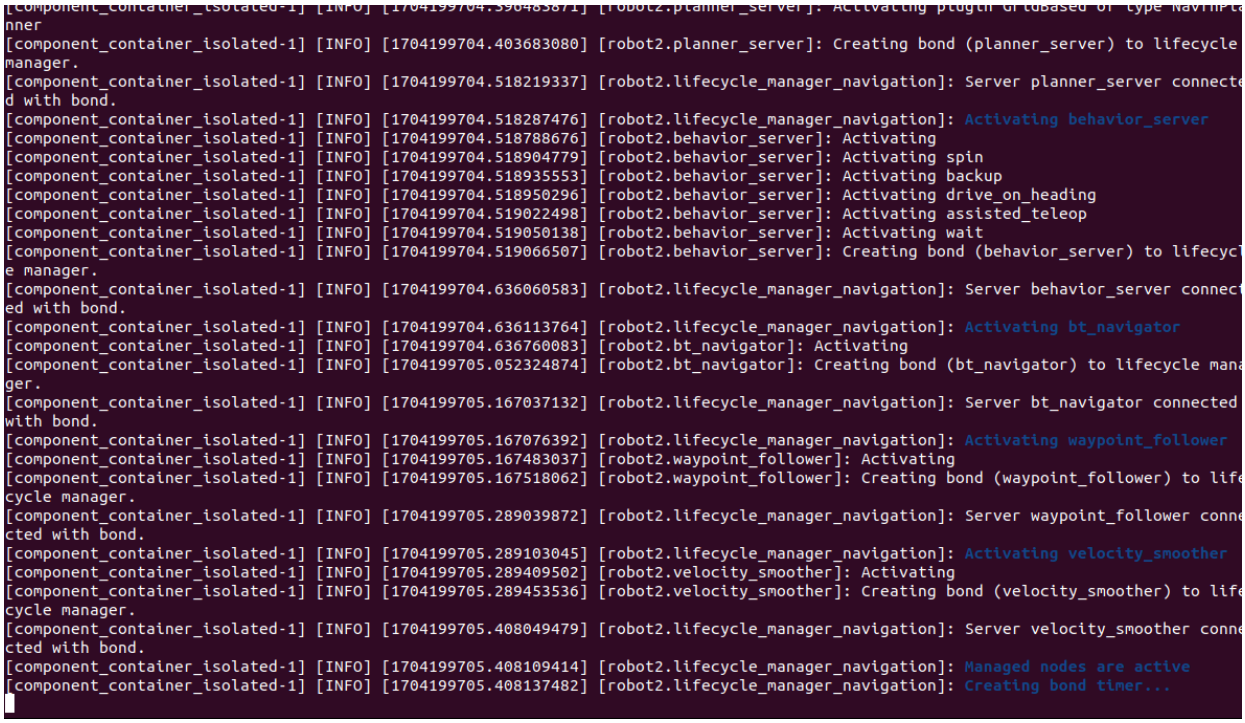
7、Start the car navigation program

Enter the following command in the virtual machine terminal that starts robot1.

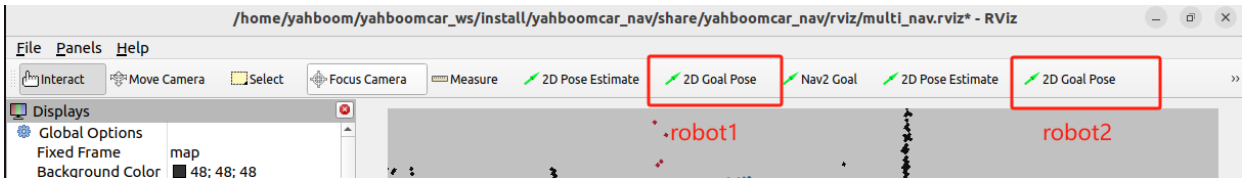
```
ros2 launch yahboomcar_multi robot1_navigation_dwb_launch.py
```

Enter the following command in the virtual machine terminal that starts robot1.

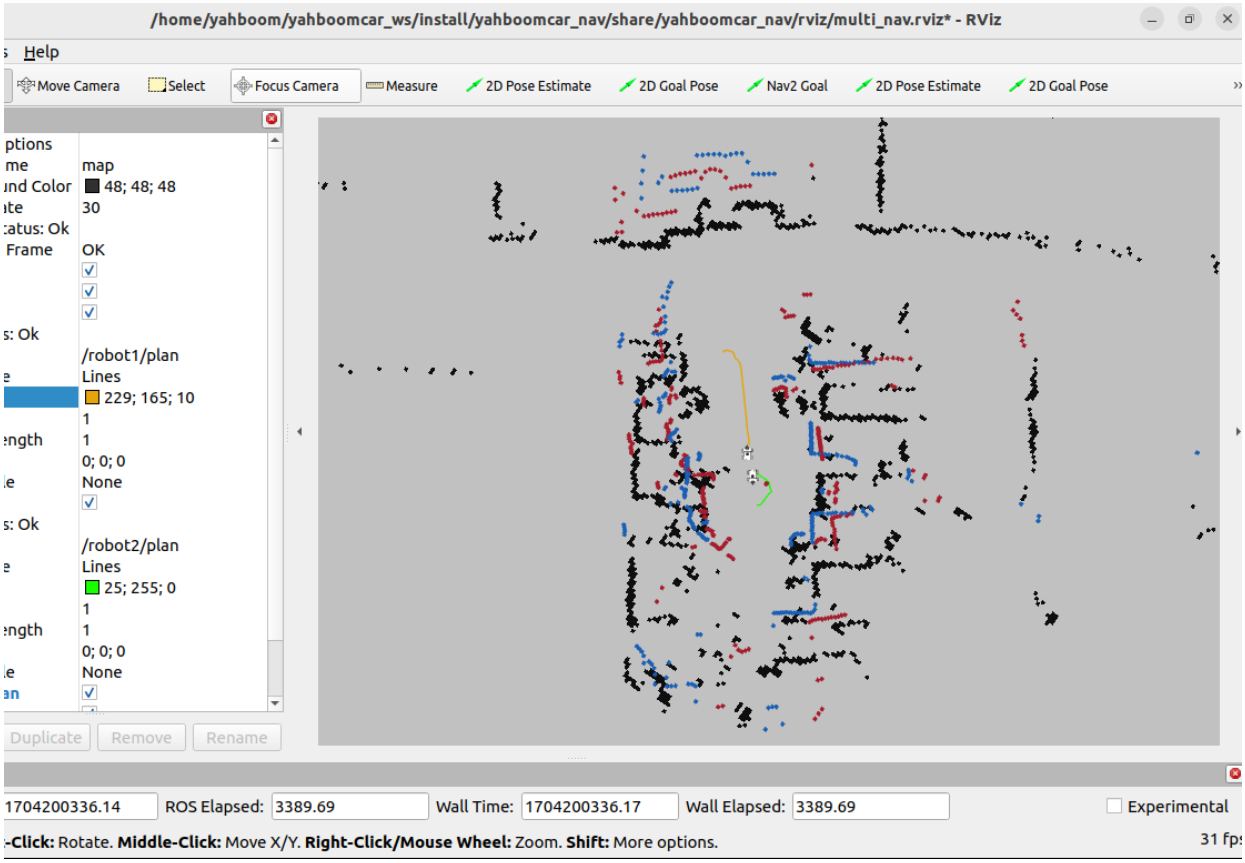
```
ros2 launch yahboomcar_multi robot2_navigation_dwb_launch.py
```



As shown in the picture above, if **"Creating bond timer..."** appears, it means that the program loading is complete. Then you can give the target points of the two cars through the corresponding **[2D Goal Pose]** on riviz. The cars combine their respective poses and poses. If there are surrounding obstacles, a path will be generated to autonomously navigate to their respective destinations.



The yellow route is the route planned by robot1, and the green line is the route planned by robot2.



8、Multi-vehicle navigation expansion

The tutorial takes two cars as an example. If you want to add other cars, you need to make the following modifications:

8.1、Add the URDF model of the car and add the urdf model loader

- Added car model

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_description/urdf/MicroROS_robot1.urdf, and change the name and robot1 that appear in the urdf file to other car names, such as robot3.

- Added urdf model loader

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_description/launch/description_multi_robot1.launch.py to change the name and robot1 that appears in the launch file to other car names. The name needs to be consistent with the new urdf.

8.2、Added car ekf parameter table

You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/ekf_robot1.yaml and change the name and robot1 that appear in the file to other car names. The name needs to be consistent with the new urdf.

8.3、Added car amcl parameter table and launch file to start amcl

- Added car amcl parameter table

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/robot1_amcl_params.yaml and change the name and robot1 that appears in the file to the name of other cars. The name needs to be consistent with the new urdf.

- Add the launch file of amcl that is started

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_amcl_launch.py, and change the name and robot1 that appears in the file to the name of other cars. The name needs to be consistent with the new urdf.

8.4、Added car nav2 parameter table and launch file to start nav2

- Added car nav2 parameter table

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_nav/params/robot1_nav_params.yaml and change the name and robot1 that appears in the file to the name of other cars. The name needs to be consistent with the new urdf.

- Added new launch file of nav2

  You can refer to /home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_navigation_dwb_launch.py, and change the name and robot1 that appears in the file to the name of other cars. The name needs to be consistent with the new urdf.

8.5、Added [2D Pose Estimate] and [2D Goal Pose] in the rviz toolbar

Modify the multi_nav.rviz file. The directory of the file is `/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/rviz`.Find the following section

```
- Class: rviz_default_plugins/SetInitialPose
  Covariance x: 0.25
  Covariance y: 0.25
  Covariance yaw: 0.06853891909122467
  Topic:
    Depth: 5
    Durability Policy: Volatile
    History Policy: Keep Last
    Reliability Policy: Reliable
    Value: /robot1/initialpose
- Class: rviz_default_plugins/SetGoal
```

```
  Topic:
    Depth: 5
    Durability Policy: Volatile
    History Policy: Keep Last
    Reliability Policy: Reliable
    Value: /robot1/goal_pose
```

The above are the two tools of robot1. You can make a copy and put it behind. Change the robot1 that appears to the name of other cars. The name needs to be consistent with the new urdf.

After completing the above 5 steps, return to the yahboomcar_ws workspace, use colcon build to compile, and then run the test according to the tutorial. After running successfully, you can add the car model and radar data to display in rviz.