Multi-machine navigation

Note: The ROS_DOMAIN_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check **[MicroROS Control Board Parameter Configuration]** to set the microROS control board ROS_DOMAIN_ID. Check the tutorial **[Connect MicroROS Agent]** to determine whether the IDs are consistent.

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.

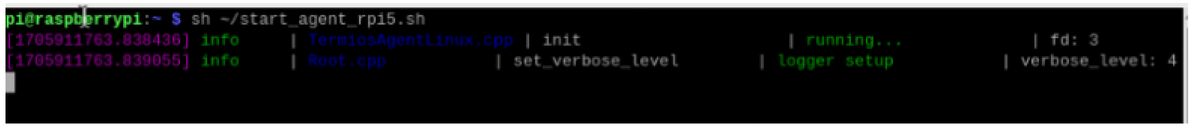
2、Multi-machine function basic settings

Taking two robots as an example, it is recommended to use two robots equipped with Raspberry Pi and change the **config_robot.py** files respectively. 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 needs to be set to the same**.   Then open the terminal in the /home/pi 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

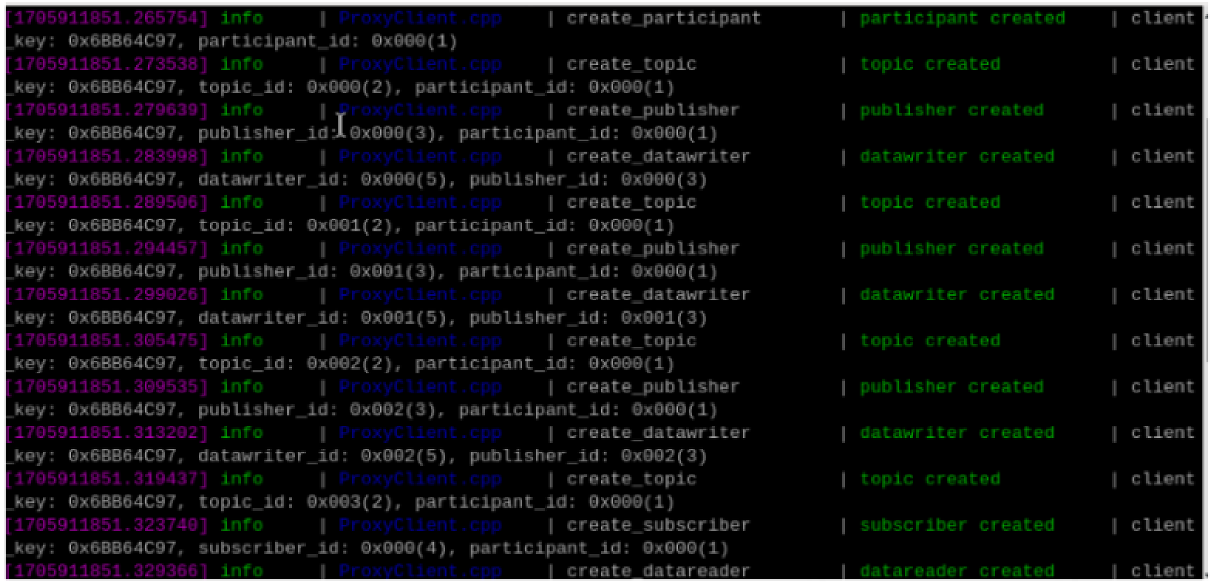
After the Raspberry Pi is successfully powered on, open the terminal and enter the following command to open the agent.

```
sh ~/start_agent_rpi5.sh
```



```
pi@raspberrypi:~ $ sh ~/start_agent_rpi5.sh
[1705911763.838436] info     | TermiosAgentLinux.cpp | init                   | running...             | fd: 3
[1705911763.839055] info     | Root.cpp              | set_verbose_level      | logger setup           | verbose_level: 4
```

Press the reset button on the microROS control board and wait for the car to connect to the agent. The connection is successful as shown in the figure below.



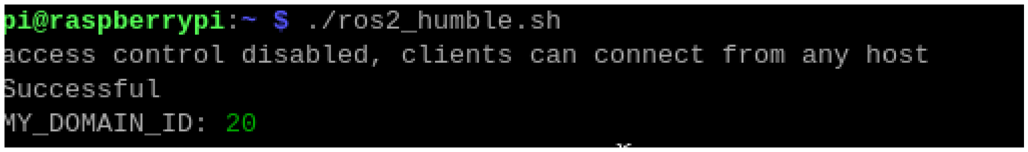
```
[1705911851.265754] info     | ProxyClient.cpp    | create_participant      | participant created     | client
_key: 0x6BB64C97, participant_id: 0x000(1)
[1705911851.273538] info     | ProxyClient.cpp    | create_topic            | topic created           | client
_key: 0x6BB64C97, topic_id: 0x000(2), participant_id: 0x000(1)
[1705911851.279639] info     | ProxyClient.cpp    | create_publisher        | publisher created       | client
_key: 0x6BB64C97, publisher_id: 0x000(3), participant_id: 0x000(1)
[1705911851.283998] info     | ProxyClient.cpp    | create_datawriter       | datawriter created      | client
_key: 0x6BB64C97, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1705911851.289506] info     | ProxyClient.cpp    | create_topic            | topic created           | client
_key: 0x6BB64C97, topic_id: 0x001(2), participant_id: 0x000(1)
[1705911851.294457] info     | ProxyClient.cpp    | create_publisher        | publisher created       | client
_key: 0x6BB64C97, publisher_id: 0x001(3), participant_id: 0x000(1)
[1705911851.299026] info     | ProxyClient.cpp    | create_datawriter       | datawriter created      | client
_key: 0x6BB64C97, datawriter_id: 0x001(5), publisher_id: 0x001(3)
[1705911851.305475] info     | ProxyClient.cpp    | create_topic            | topic created           | client
_key: 0x6BB64C97, topic_id: 0x002(2), participant_id: 0x000(1)
[1705911851.309535] info     | ProxyClient.cpp    | create_publisher        | publisher created       | client
_key: 0x6BB64C97, publisher_id: 0x002(3), participant_id: 0x000(1)
[1705911851.313202] info     | ProxyClient.cpp    | create_datawriter       | datawriter created      | client
_key: 0x6BB64C97, datawriter_id: 0x002(5), publisher_id: 0x002(3)
[1705911851.319437] info     | ProxyClient.cpp    | create_topic            | topic created           | client
_key: 0x6BB64C97, topic_id: 0x003(2), participant_id: 0x000(1)
[1705911851.323740] info     | ProxyClient.cpp    | create_subscriber       | subscriber created      | client
_key: 0x6BB64C97, subscriber_id: 0x000(4), participant_id: 0x000(1)
[1705911851.329366] info     | ProxyClient.cpp    | create_datareader       | datareader created      | client
```

4、Enter the car docker

Open another terminal and enter the following command to enter docker.

```
sh ros2_humble.sh
```

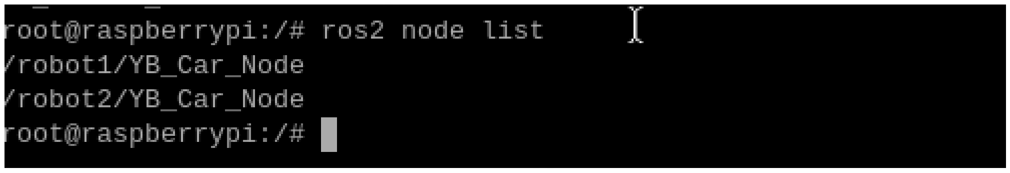
When the following interface appears, you have successfully entered docker. Now you can control the car through commands.



```
pi@raspberrypi:~ $ ./ros2_humble.sh
access control disabled, clients can connect from any host
Successful
MY_DOMAIN_ID: 20
```

Check the currently started node. Choose one of the two Raspberry Pis, open the terminal and enter the following command.

```
ros2 node list
```



```
root@raspberrypi:/# ros2 node list
/robot1/YB_Car_Node
/robot2/YB_Car_Node
root@raspberrypi:/#
```

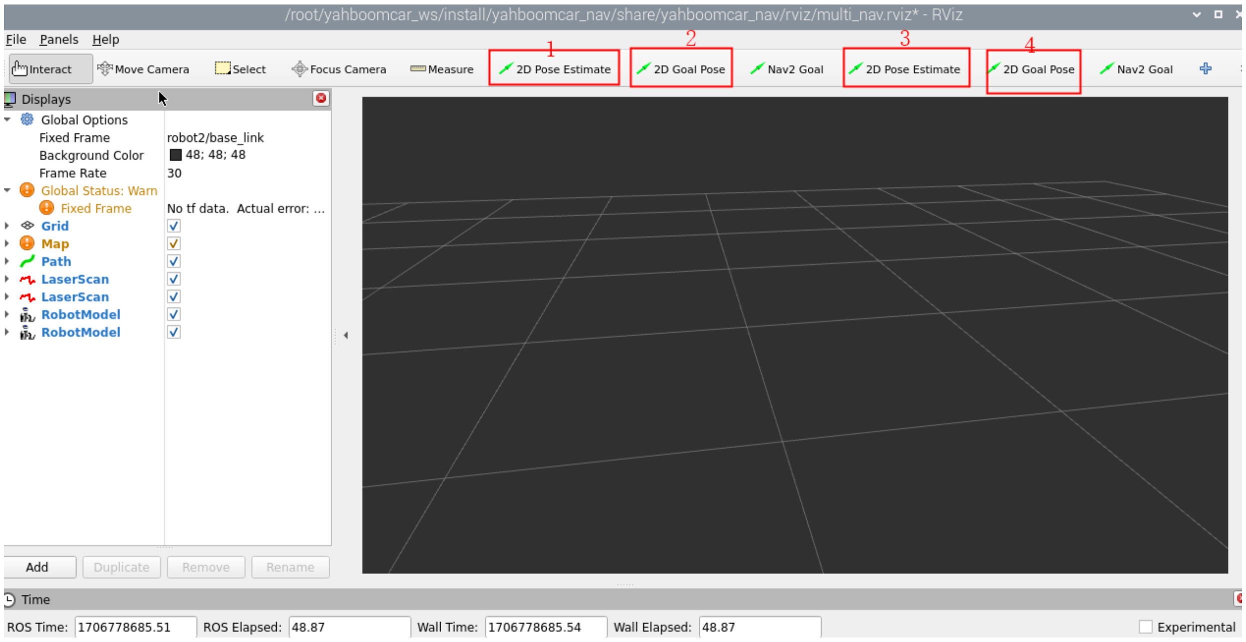
As shown in the picture above, the nodes of both cars have been started.

5、Start rviz and load the map

5.1、Start rviz display

Choose one of the two Raspberry Pis at random, open the terminal and input, (it is recommended to use the matching virtual machine to open the map, opening it on the Raspberry Pi will be very laggy)

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



Global Options
Fixed Frame robot2/base_link
Background Color 48; 48; 48
Frame Rate 30
Global Status: Warn
Fixed Frame No tf data. Actual error: ...

ROS Time: 1706778685.51 ROS Elapsed: 48.87 Wall Time: 1706778685.54 Wall Elapsed: 48.87

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

【1】: robot1 calibrate initial pose

【2】: robot1 gives a target point

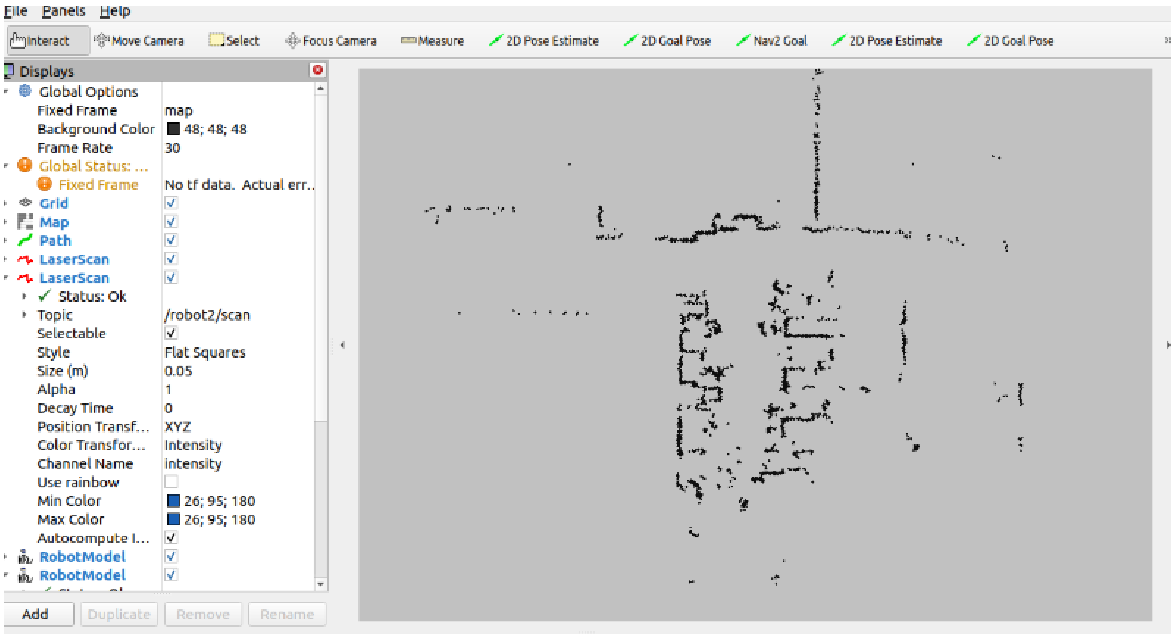
【3】: robot2 calibrate initial pose

【4】: robot2 given target point

4.2、Load map

Choose one of the two Raspberry Pis at random, open the terminal and input, (it is recommended to use the matching virtual machine to open the map, opening it on the Raspberry Pi will be very laggy)

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



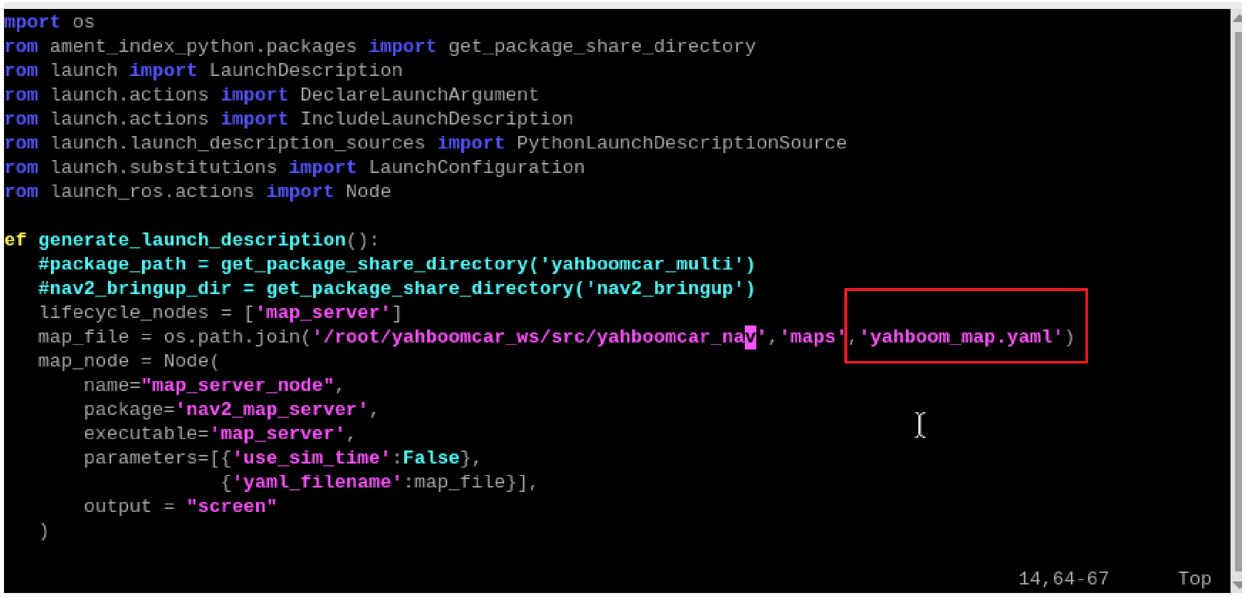
```
Global Options
    Fixed Frame            map
    Background Color       48; 48; 48
    Frame Rate             30
Global Status: ...
    Fixed Frame            No tf data.  Actual err..
Grid                       ✓
Map                        ✓
Path                       ✓
LaserScan                  ✓
LaserScan                  ✓
    Status: Ok
    Topic                  /robot2/scan
    Selectable             ✓
    Style                  Flat Squares
    Size (m)               0.05
    Alpha                  1
    Decay Time             0
    Position Transf...     XYZ
    Color Transfor...      Intensity
    Channel Name           intensity
    Use rainbow
    Min Color              26; 95; 180
    Max Color              26; 95; 180
    Autocompute I...       ✓
RobotModel                 ✓
RobotModel                 ✓
```

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 path loaded here is as follows.

```
/root/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

`/root/yahboomcar_ws/src/yahboomcar_multi/maps/` directory. Then modify the map_server_launch.py program, which is located in `/root/yahboomcar_ws/src/yahboomcar_multi/launch`. The modification places are as follows



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

ef generate_launch_description():
   #package_path = get_package_share_directory('yahboomcar_multi')
   #nav2_bringup_dir = get_package_share_directory('nav2_bringup')
   lifecycle_nodes = ['map_server']
   map_file = os.path.join('/root/yahboomcar_ws/src/yahboomcar_nav','maps','yahboom_map.yaml')
   map_node = Node(
       name="map_server_node",
       package='nav2_map_server',
       executable='map_server',
       parameters=[{'use_sim_time':False},
                   {'yaml_filename':map_file}],
       output = "screen"
   )
                                                          14,64-67      Top
```

Replace 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 Raspberry Pi terminal that starts robot1

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

Enter the following command in the Raspberry Pi terminal that starts robot2

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

yahboomcar_bringup_multi.launch.xml source code path,

```
/root/yahboomcar_ws/src/yahboomcar_multi/launch/yahboomcar_bringup_multi.launch.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 " />
```

```
    <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.

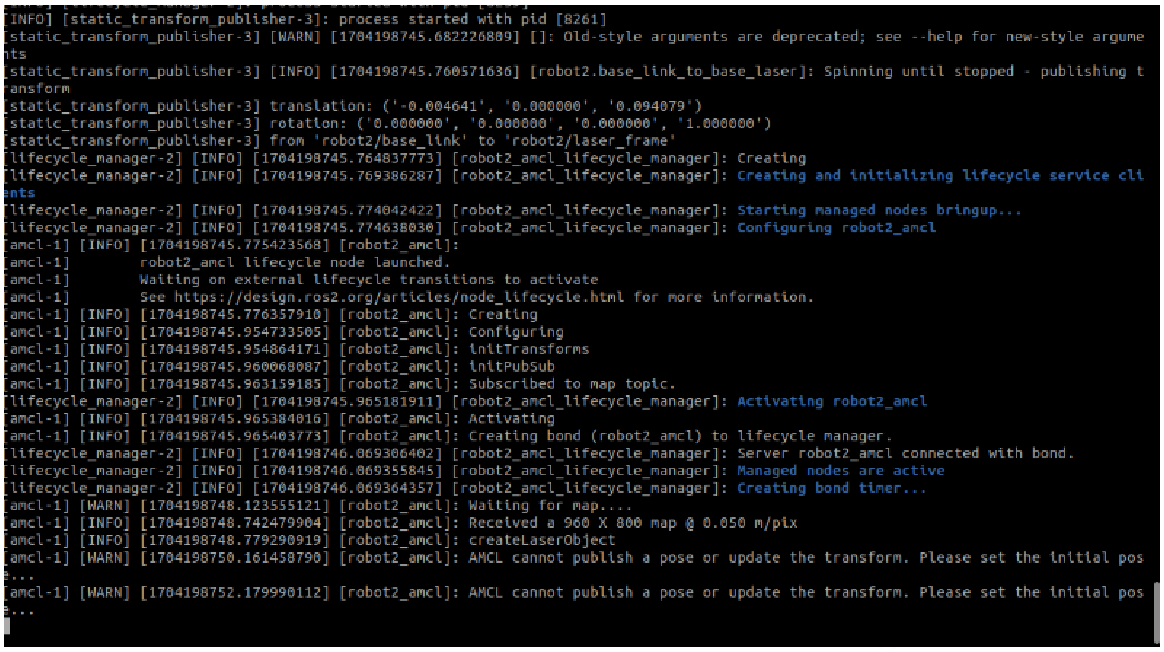
6、Start the AMCL positioning program of the car

Enter the command in the Raspberry Pi terminal that starts robot1

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

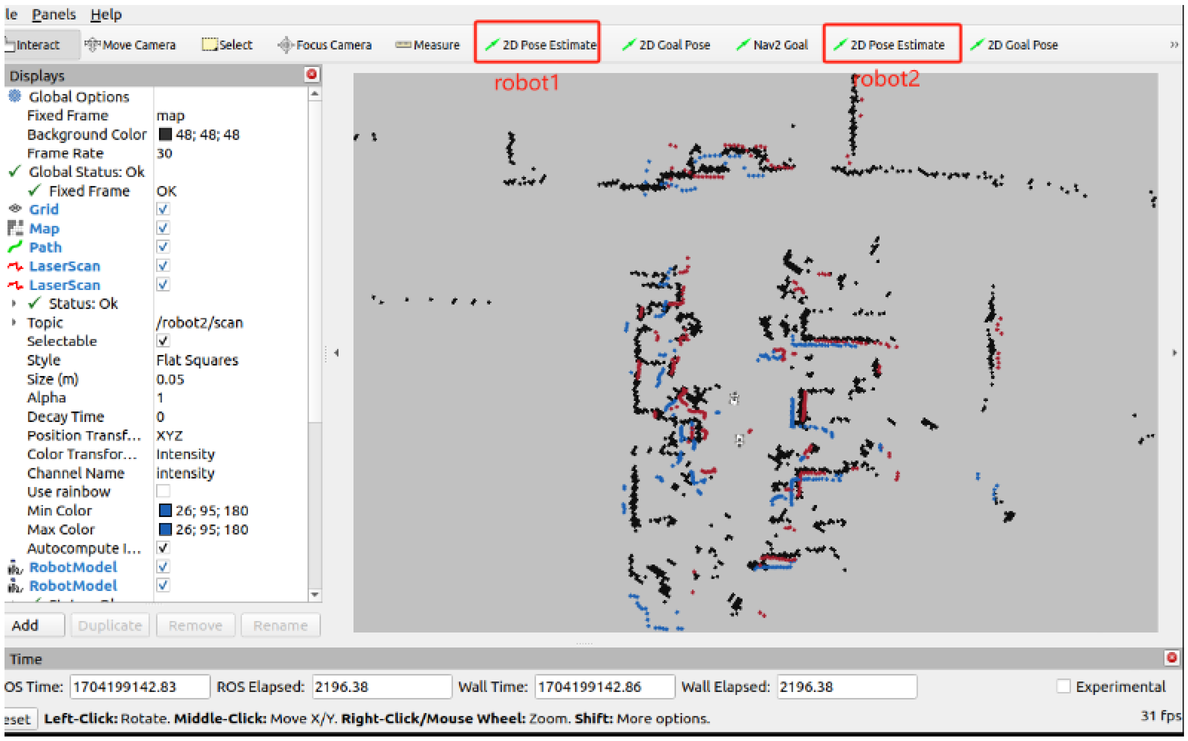
Enter the command in the Raspberry Pi terminal that starts robot2

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



```
[INFO] [static_transform_publisher-3]: process started with pid [8261]
[static_transform_publisher-3] [WARN] [1704198745.682226889] []: 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 figure 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.



```
Displays
Global Options
  Fixed Frame          map
  Background Color     48; 48; 48
  Frame Rate           30
✓ Global Status: Ok
  ✓ Fixed Frame        OK
Grid                   ✓
Map                    ✓
Path                   ✓
LaserScan              ✓
LaserScan              ✓
  ✓ Status: Ok
  Topic                /robot2/scan
  Selectable           ✓
  Style                Flat Squares
  Size (m)             0.05
  Alpha                1
  Decay Time           0
  Position Transf...   XYZ
  Color Transfor...    Intensity
  Channel Name         intensity
  Use rainbow
  Min Color            26; 95; 180
  Max Color            26; 95; 180
  Autocompute I...     ✓
RobotModel             ✓
RobotModel             ✓

Time
OS Time: 1704199142.83   ROS Elapsed: 2196.38   Wall Time: 1704199142.86   Wall Elapsed: 2196.38
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
```

Red represents robot1, 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 location：/root/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

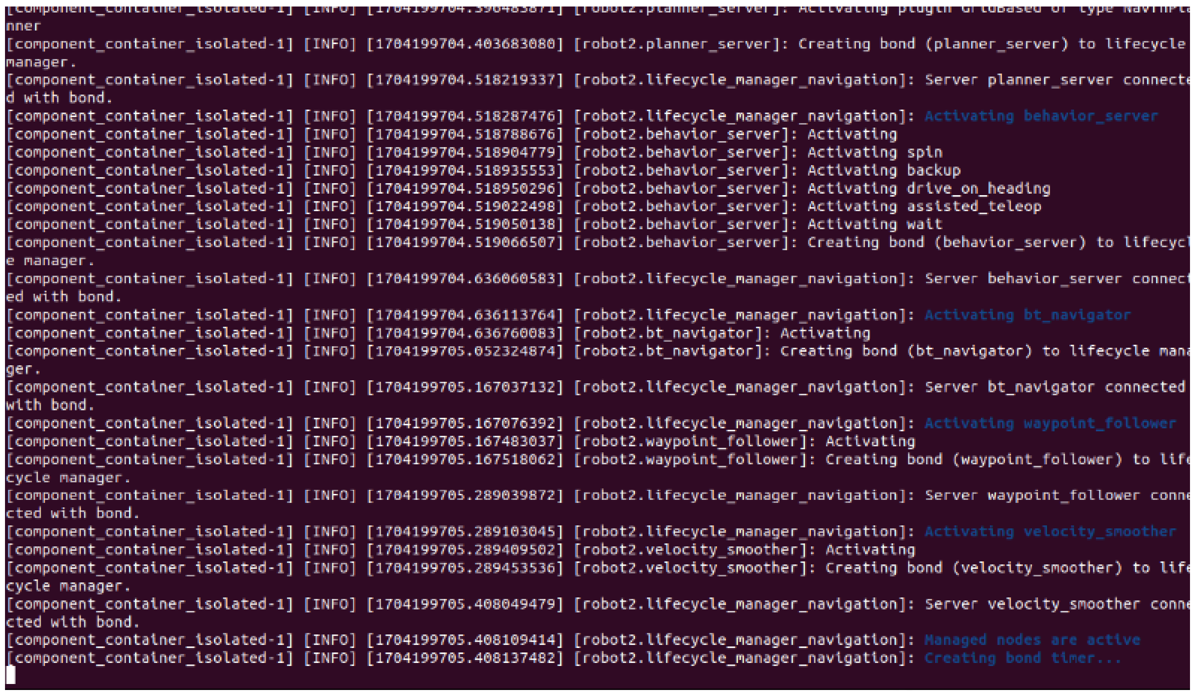
7、Start the car navigation program

Enter the command in the Raspberry Pi terminal that starts robot1

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

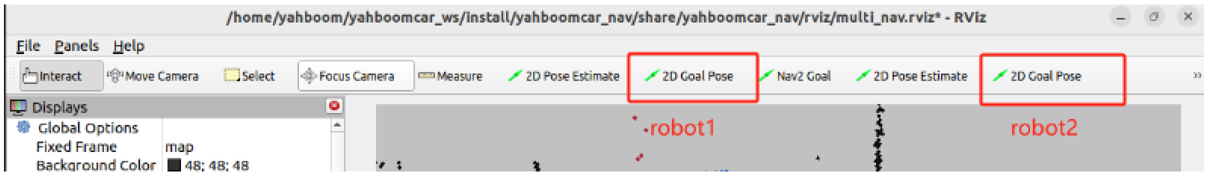
Enter the command in the Raspberry Pi terminal that starts robot2

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

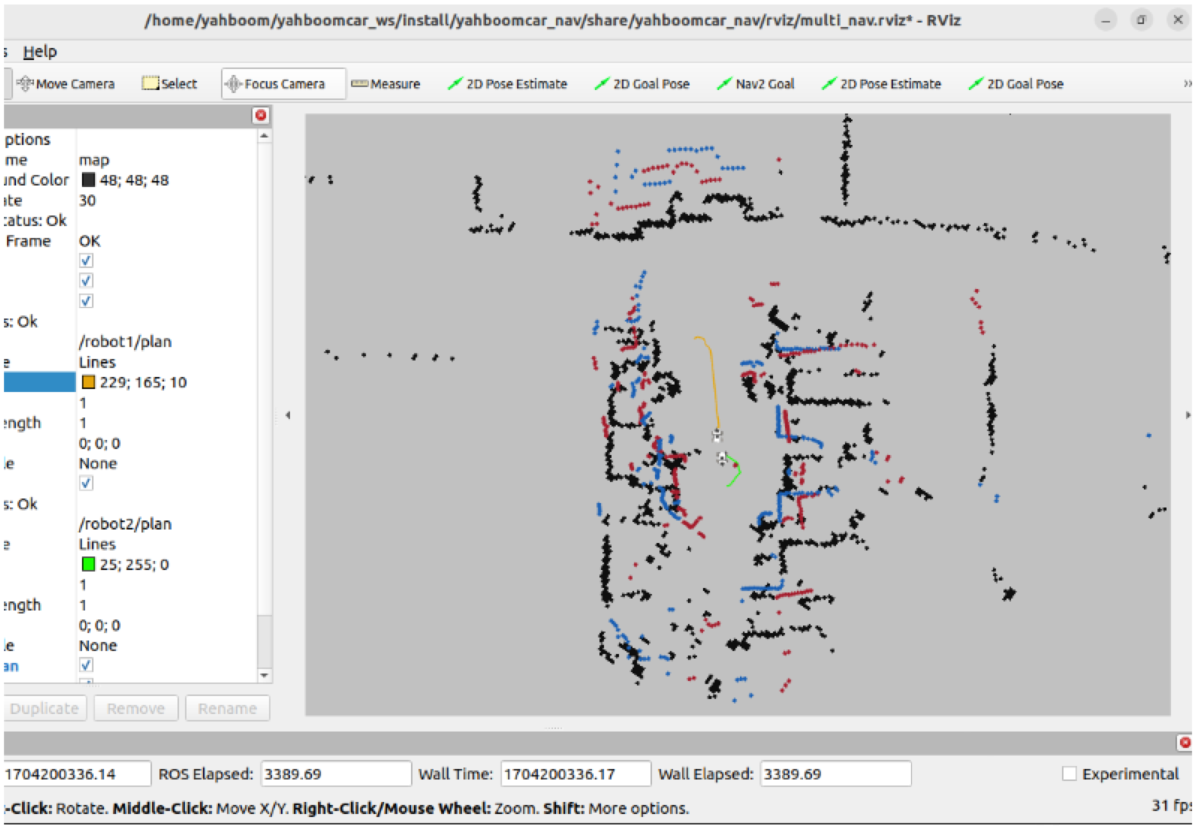


```
[component_container_isolated-1] [INFO] [1704199704.390483871] [robot2.planner_server]: Activating plugin GridBased of type NavfnPla
nner
[component_container_isolated-1] [INFO] [1704199704.403683080] [robot2.planner_server]: Creating bond (planner_server) to lifecycle
manager.
[component_container_isolated-1] [INFO] [1704199704.518219337] [robot2.lifecycle_manager_navigation]: Server planner_server connecte
d with bond.
[component_container_isolated-1] [INFO] [1704199704.518287476] [robot2.lifecycle_manager_navigation]: Activating behavior_server
[component_container_isolated-1] [INFO] [1704199704.518788676] [robot2.behavior_server]: Activating
[component_container_isolated-1] [INFO] [1704199704.518904779] [robot2.behavior_server]: Activating spin
[component_container_isolated-1] [INFO] [1704199704.518935553] [robot2.behavior_server]: Activating backup
[component_container_isolated-1] [INFO] [1704199704.518950296] [robot2.behavior_server]: Activating drive_on_heading
[component_container_isolated-1] [INFO] [1704199704.519022498] [robot2.behavior_server]: Activating assisted_teleop
[component_container_isolated-1] [INFO] [1704199704.519050138] [robot2.behavior_server]: Activating wait
[component_container_isolated-1] [INFO] [1704199704.519066507] [robot2.behavior_server]: Creating bond (behavior_server) to lifecycl
e manager.
[component_container_isolated-1] [INFO] [1704199704.636060583] [robot2.lifecycle_manager_navigation]: Server behavior_server connect
ed with bond.
[component_container_isolated-1] [INFO] [1704199704.636113764] [robot2.lifecycle_manager_navigation]: Activating bt_navigator
[component_container_isolated-1] [INFO] [1704199704.636760083] [robot2.bt_navigator]: Activating
[component_container_isolated-1] [INFO] [1704199705.052324874] [robot2.bt_navigator]: Creating bond (bt_navigator) to lifecycle mana
ger.
[component_container_isolated-1] [INFO] [1704199705.167037132] [robot2.lifecycle_manager_navigation]: Server bt_navigator connected
with bond.
[component_container_isolated-1] [INFO] [1704199705.167076392] [robot2.lifecycle_manager_navigation]: Activating waypoint_follower
[component_container_isolated-1] [INFO] [1704199705.167483037] [robot2.waypoint_follower]: Activating
[component_container_isolated-1] [INFO] [1704199705.167518062] [robot2.waypoint_follower]: Creating bond (waypoint_follower) to life
cycle manager.
[component_container_isolated-1] [INFO] [1704199705.289039872] [robot2.lifecycle_manager_navigation]: Server waypoint_follower conne
cted with bond.
[component_container_isolated-1] [INFO] [1704199705.289103045] [robot2.lifecycle_manager_navigation]: Activating velocity_smoother
[component_container_isolated-1] [INFO] [1704199705.289409502] [robot2.velocity_smoother]: Activating
[component_container_isolated-1] [INFO] [1704199705.289453536] [robot2.velocity_smoother]: Creating bond (velocity_smoother) to life
cycle manager.
[component_container_isolated-1] [INFO] [1704199705.408049479] [robot2.lifecycle_manager_navigation]: Server velocity_smoother conne
cted with bond.
[component_container_isolated-1] [INFO] [1704199705.408109414] [robot2.lifecycle_manager_navigation]: Managed nodes are active
[component_container_isolated-1] [INFO] [1704199705.408137482] [robot2.lifecycle_manager_navigation]: Creating bond timer...
```

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.



/home/yahboom/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/multi_nav.rviz* - RViz

ptions	
me	map
ınd Color	■ 48; 48; 48
ıte	30
:atus: Ok	
Frame	OK
	✓
	✓
	✓
s: Ok	
	/robot1/plan
e	Lines
	🟧 229; 165; 10
	1
ength	1
	0; 0; 0
e	None
	✓
s: Ok	
	/robot2/plan
e	Lines
	🟩 25; 255; 0
	1
ength	1
	0; 0; 0
e	None
an	✓

1704200336.14 ROS Elapsed: 3389.69 Wall Time: 1704200336.17 Wall Elapsed: 3389.69 Experimental

-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options. 31 fp:

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 /root/yahboomcar_ws/src/yahboomcar_description/urdf/MicroROS_robot1.urdf. Change the name and robot1 that appears in the urdf file to another car name, such as robot3.

- Added urdf model loader

  You can refer to /root/yahboomcar_ws/src/yahboomcar_description/launch/description_multi_robot1.launch.py. Change the name and robot1 that appears in the launch file to another car name. The name needs to be consistent with the new urdf.

8.2、Added car ekf parameter table

You can refer to /root/yahboomcar_ws/src/yahboomcar_multi/param/ekf_robot1.yaml. Change the name and robot1 that appears in the file to another car name. 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 /root/yahboomcar_ws/src/yahboomcar_multi/param/robot1_amcl_params.yaml. 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 /root/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_amcl_launch.py. 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 /root/yahboomcar_ws/src/yahboomcar_nav/params/robot1_nav_params.yaml. 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 /root/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_navigation_dwb_launch.py. 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 this file is `/root/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.