Multi-car Navigation2 Navigation

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be the same. You can check [Must-read before use] to set the IP and ROS_DOMAIN_ID on the board.

1. Program function description

After the program is started, you can give the target points of the two cars in rviz. After receiving the command, the two cars will calculate the path based on their own posture and move to their destinations.

```
Note: Before running the program, the car needs to be restarted in a standing
position to ensure that all sensors are reset
```

2. Basic settings for multi-machine functions

Take two cars as an example. It is recommended to use two computers with matching virtual machines, change the config_robot.py files respectively, set robot.set_ros_namespace() to robot1 and robot2 respectively; set robot.set_udp_config() to the IP addresses of the two virtual machines 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, enter `sudo python3 config_Balance_Car.py` to run this program (the rest of the programs other than running multiple cars need to be changed back and re-run this program).

Open ▾ | config_Balance_Car.py ~/ | Save

```
        car_type = self.read_car_type()
        print("car_type:", car_type)

        domain_id = self.read_ros_domain_id()
        print("domain_id:", domain_id)

        baudrate = self.read_ros_serial_baudrate()
        print("ros_serial_baudrate:", baudrate)

        ros_namespace = self.read_ros_namespace()
        print("ros_namespace:", ros_namespace)



if __name__ == '__main__':
    robot = MicroROS_Robot(port='/dev/ttyUSB0', debug=False)
    print("Rebooting Device, Please wait.")
    robot.reboot_device()

    robot.set_wifi_config("Yahboom2", "yahboom890729")
    robot.set_udp_config([192, 168, 2, 99], 8899)
    robot.set_car_type(robot.CAR_TYPE_COMPUTER)
    #robot.set_car_type(robot.CAR_TYPE_UASRT_CAR)

    robot.set_ros_domain_id(20)
    robot.set_ros_serial_baudrate(921600)
    robot.set_ros_namespace("robot2")


    time.sleep(.1)
    robot.print_all_firmware_parm()
    print("Please reboot the device to take effect, if you change some device config.")

    try:
        while False:
            # robot.beep(100)
            time.sleep(1)
    except:
```

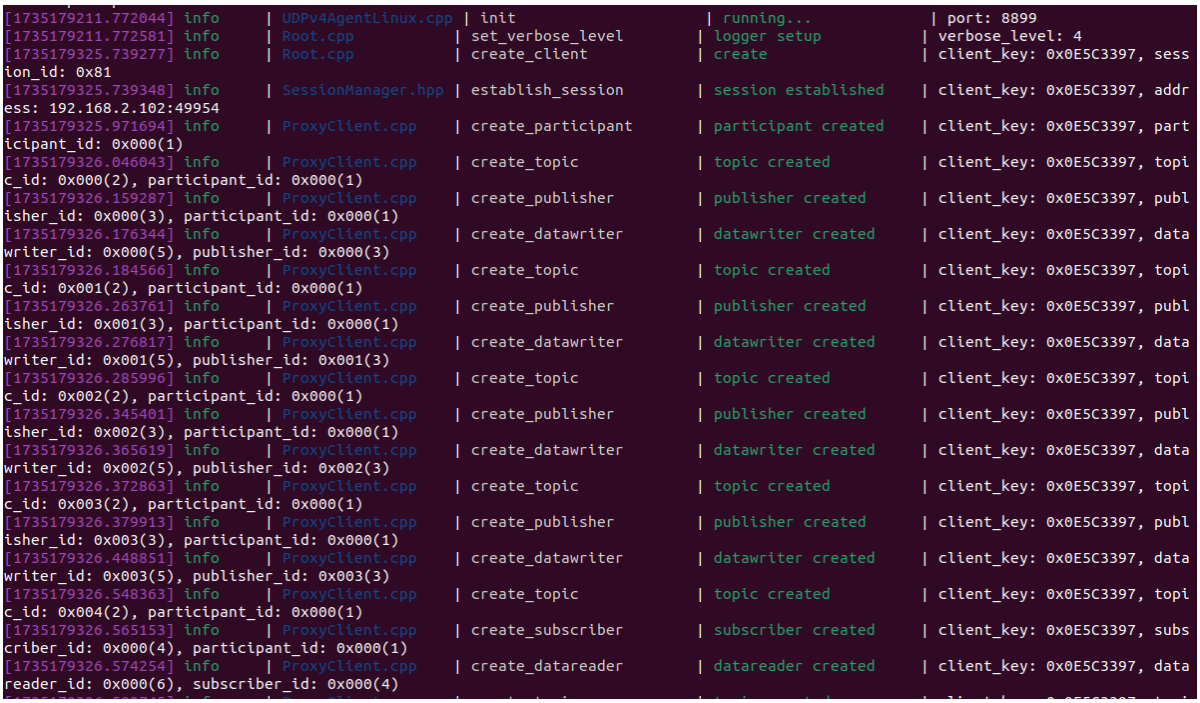
3. Start and connect the agent

Take the matching virtual machine as an example. In the two virtual machines, enter the following command to start the agent of each car.

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
```

```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     |
running...              | port: 8899
[1735179211.772581] info     | Root.cpp           | set_verbose_level        | l
ogger setup             | verbose_level: 4
```

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



Check the currently started node. In the two virtual machines, randomly select one and open the terminal to enter,

```
ros2 node list
```

```
yahboom@yahboom-VM:~$ ros2 node list
WARNING: Be aware that are nodes in the graph that share an exact name, this can
 have unintended side effects.
/robot2/YB_BalanceCar_Node
/robot2/YB_BalanceCar_Node
```

As shown in the figure above, the nodes of both cars have been started. Query the current topic information, enter in the terminal,

```
ros2 topic list
```

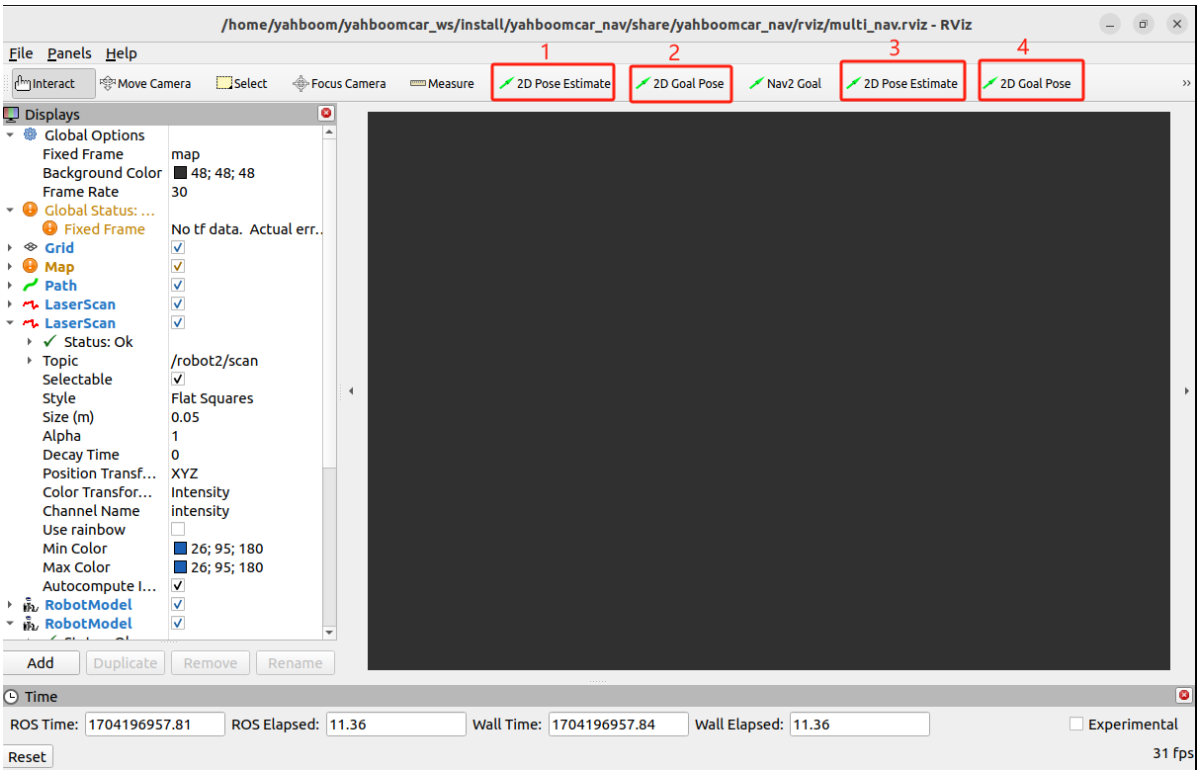
```
yahboom@yahboom-VM:~$ ros2 topic list
/parameter_events
/robot1/beep
/robot1/cmd_vel_bl
/robot1/imu
/robot1/mpuimu
/robot1/odom_raw
/robot1/scan
/robot2/beep
/robot2/cmd_vel_bl
/robot2/imu
/robot2/mpuimu
/robot2/odom_raw
/robot2/scan
/rosout
```

4. Start rviz and load the map program

4.1. Start rviz display

Select one of the two virtual machines at random, open the terminal and enter,

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



The functions of the symbols in Figures 1-4 above are as follows:

[1]: robot1 calibrates the initial position

[2]: robot1 gives the target point

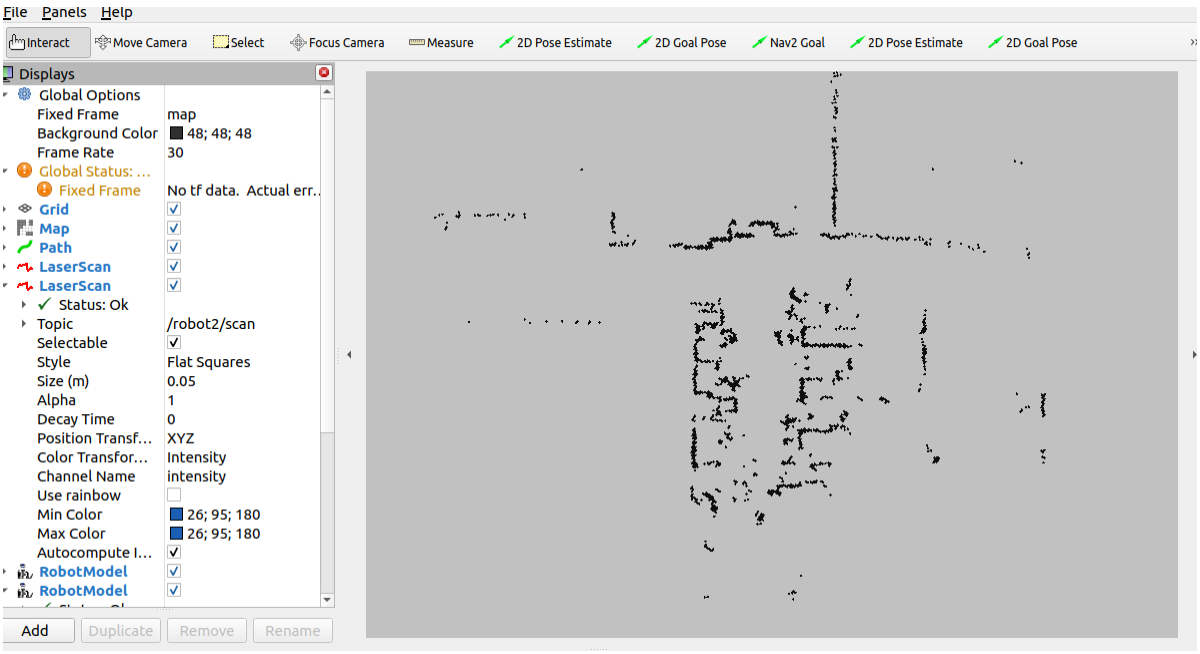
[3]: robot2 calibrates the initial position

[4]: robot2 gives the target point

4.2. Load the map

Select one of the two virtual machines at random, open the terminal and enter,

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



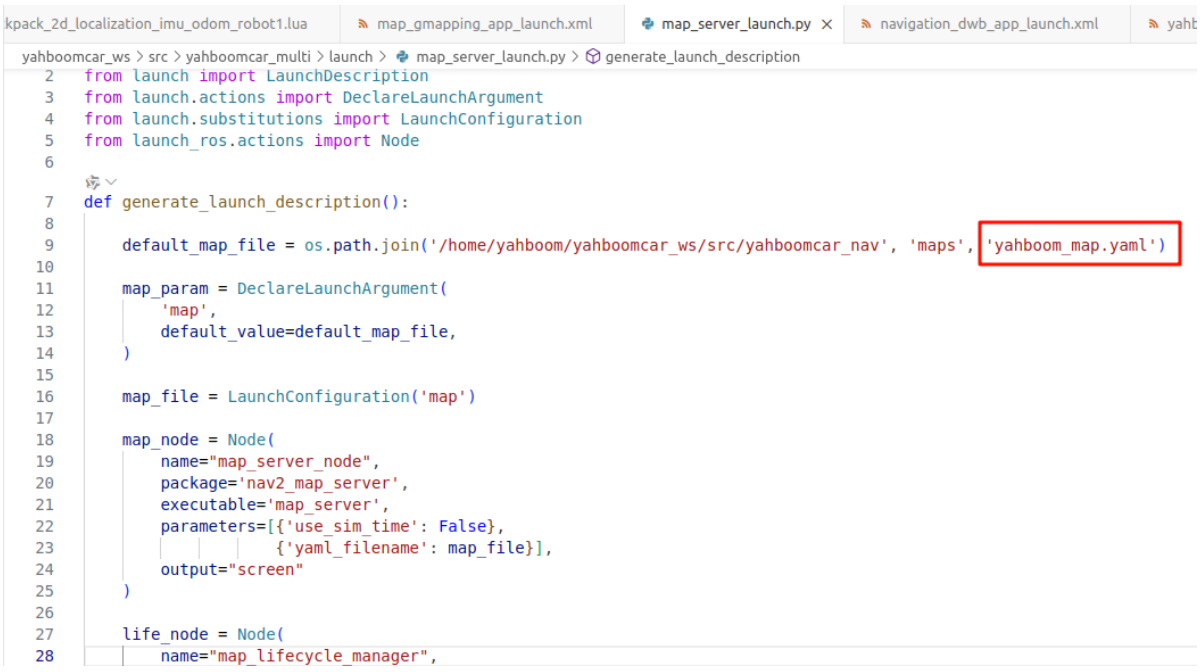
Note: The map may fail to load here. If the map is not loaded, press ctrl c to close it and rerun the program.

Take the virtual machine as an example. The map loaded here is,

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/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 the directory

`/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/`, and then modify Modify the map_server_launch.py program, which is located in `/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch`, and modify the following places,



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

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

Then enter the following command to re-source the environment variable,

```
source ~/.bashrc
```

5. Start the car's underlying data processing program

In the virtual machine terminal that starts robot1, enter,

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

In the virtual machine terminal that starts robot2, enter,

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

yahboomcar_bringup_multi.launch.xml source code path (take the matching virtual machine as an example),

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

```
<launch>
    <arg name="robot_name" default="robot1"/>
    <group>
        <push-ros-namespace namespace="$(var robot_name)"/>
        <!--base_node-->
        <node name="base" pkg="yahboomcar_bringup" exec="cmdvel2bl"
output="screen">
            <param name="mode" value="nav"/>
            <remap from="/cmd_vel" to="cmd_vel"/>
            <remap from="/cmd_vel_bl" to="cmd_vel_bl"/>
        </node>
        <!--imu_filter_node-->
        <node name="imu_filter" pkg="imu_filter_madgwick"
exec="imu_filter_madgwick_node" output="screen">
            <param name="fixed_frame" value="$(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 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 0.016325 0.080691 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>
```

Here, a pair of tags are used. 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, which are also automatically selected and loaded through this robot_name. You can view the content in the code for details.

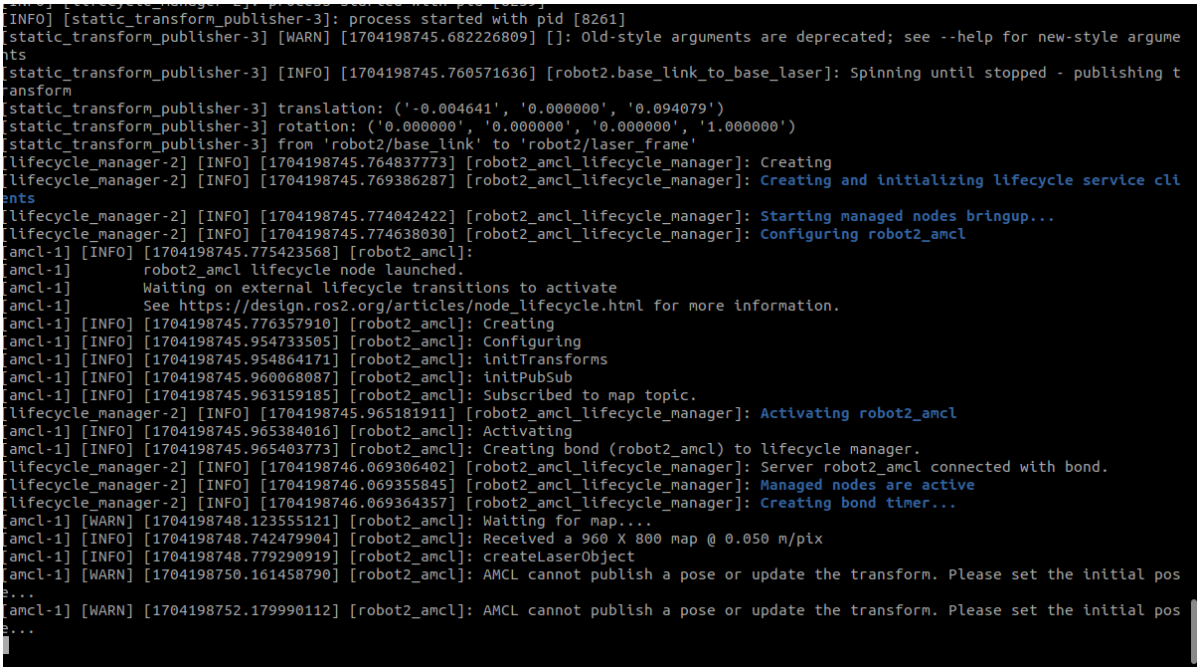
6. Start the AMCL positioning program of the car

In the virtual machine terminal that starts robot1, enter,

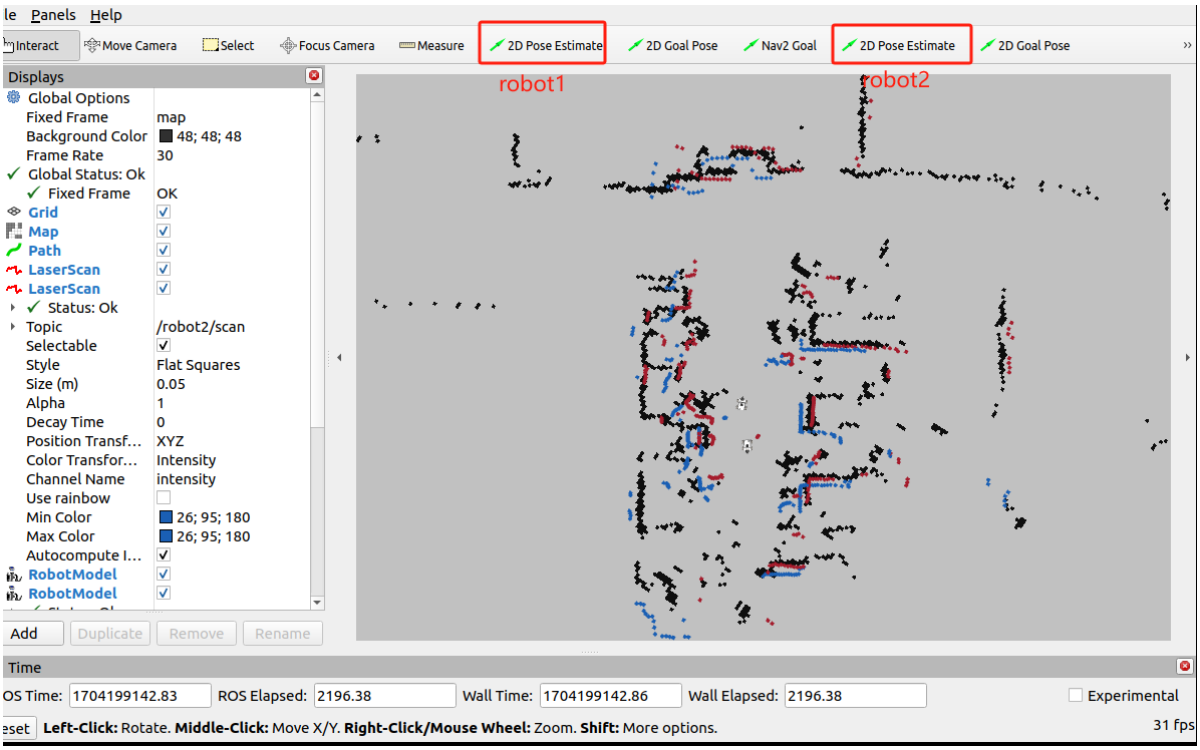
```
ros2 launch yahboomcar_multi robot1_amcl_launch.py
```

In the virtual machine terminal that starts robot2, enter,

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



As shown in the figure above, "Please set the initial pose..." appears, and the corresponding [2D Pose] tool can be used to give the initial poses to 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 posture is accurate.



Red indicates robot1 and blue indicates robot2.

Note: If the terminal cannot print "Please set the initial pose...", it may be a problem of data timestamp asynchrony. Press the reset button of the car to let the car reconnect to the proxy to ensure that the data timestamp is correct. Repeat several times until "Please set the initial pose..." appears.

Take the virtual machine as an example, the source code 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
_params.yaml')
    amcl_node = Node(
        name="robot1_amcl",
        package='nav2_amcl',
        executable='amcl',
        parameters=
[os.path.join(get_package_share_directory('yahboomcar_multi'),'param','robot1_amc
l_params.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', '0' ,
'0.138','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, used to estimate the pose and realize positioning

life_node: amcl node life cycle manager

base_link_to_laser_tf_node: static transformation of chassis and radar data

7. Start the car navigation program

In the virtual machine terminal that starts robot1, enter,

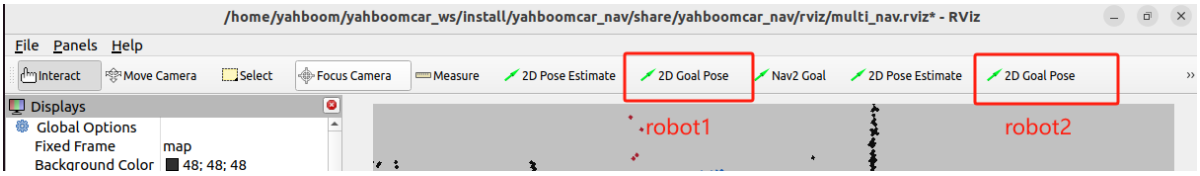
```
ros2 launch yahboomcar_multi robot1_navigation_dwb_launch.py
```

In the virtual machine terminal that starts robot2, enter,

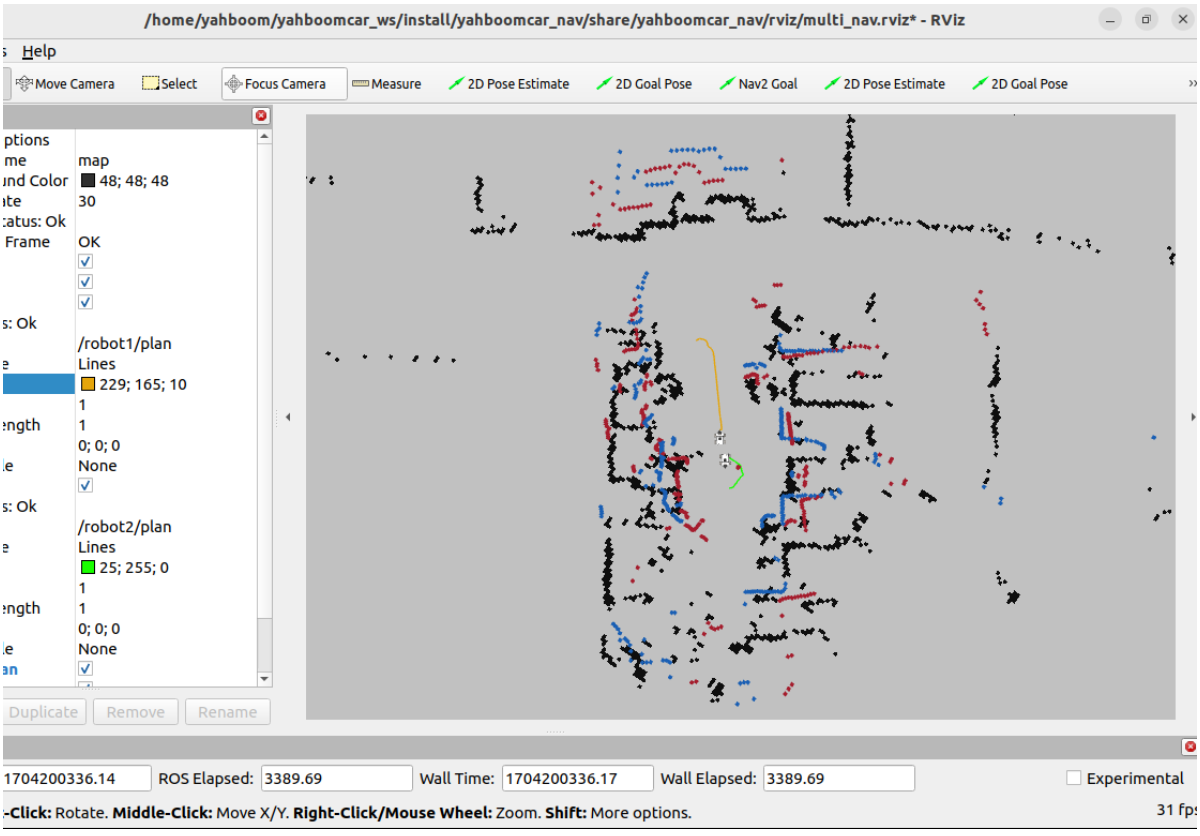
```
ros2 launch yahboomcar_multi robot2_navigation_dwb_launch.py
```

```
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 figure above, the "Creating bond timer..." appears, indicating that the program has been loaded, then you can use the corresponding [2D Goal Pose】Given the target points of the two cars, the cars will generate a path based on their respective positions and surrounding obstacles and 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-car 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 a new car URDF model and add a urdf model loader

Note: The path is,

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_description/launch/description_multi_robot1.launch.py
/home/yahboom/yahboomcar_ws/src/yahboomcar_description/urdf/STM32-V2-V1_robot1.urdf
```

- Add a new car model

For reference, `/home/yahboom/yahboomcar_ws/src/yahboomcar_description/urdf/STM32-V2-V1_robot1.urdf`

Change the name and robot1 in the urdf file to other car names, such as robot3.

- Added urdf model loader

For reference, `/home/yahboom/yahboomcar_ws/src/yahboomcar_description/launch/description_multi_robot1.launch.py` Change the name and robot1 in the launch file to other car names. The name needs to be consistent with the newly added urdf.

8.2, Added car ekf parameter table

Note: The path is,

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/ekf_robot1.yaml
```

You can refer to `/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/ekf_robot1.yaml`, change the name and robot1 that appears in the file to other car names, and the name needs to be consistent with the newly added urdf.

8.3. Add a car amcl parameter table and launch file for amcl

Note: The path is:

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/robot1_amcl_params.yaml
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_amcl_launch.py
```

- Add a car amcl parameter table

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 newly added urdf.

- Added launch file of amcl

Refer to `/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_amcl_launch.py`, change the name and robot1 in the file to the name of other cars, and the name needs to be consistent with the newly added urdf.

8.4, Added car nav2 parameter table and launch file for starting nav2

Note: The path is:

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/robot1_nav_params.yaml
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_navigation_dwb_lau
nch.py
```

- Added car nav2 parameter table

Refer to `/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/param/robot1_nav_params.yaml`, change the name and robot1 in the file to the name of other cars, and the name needs to be consistent with the newly added urdf.

- Added launch file for nav2

Refer to `/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/launch/robot1_navigation_dwb_launch.py`, change the name and robot1 in the file to the name of other cars, and the name needs to be consistent with the newly added urdf.

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

Note: The path is:

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/rviz
```

Modify the multi_nav.rviz file, the directory of the file is `/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/rviz/multi_nav.rviz`, find the following part,

```
- 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 two tools for robot1. You can copy one and put it behind. Change the robot1 that appears to the name of other cars. The name needs to be consistent with the newly added urdf.

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