# 2. Multi-machine navigation



## 2.1. Introduction

Regarding how to configure multi-machine communication and synchronization time, please refer to the lesson [Multi-machine joystick control] for details; if there is a network, the network system time can be synchronized directly without setting.

When using multi-machine handle control, you first need to ensure that the robots are under the same LAN and configured with the same [ROS_MASTER_URI]; multiple robots can only have one host to control their movements. In the case of this section, the virtual machine is set as the host machine, and other robots are slave machines. There are several slave machines. Of course, you can also set a certain robot as the master machine and the others as slave machines.

According to different models, you only need to set the purchased model in [.bashrc], X1 (normal four-wheel drive) X3 (Mailun) Take X3 as an example

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameters and modify the corresponding car model

```
export ROBOT_TYPE=X3 # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

## 2.2. Use

Take the virtual machine as the host and the three robots as slaves as an example; a map must be available before use.

### 1.2.1. Start the robot

Virtual machine side

```
roscore
```

Start the command (robot1 side). For the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch yahboomcar_multi laser_bringup_multi.launch ns:=robot1 # laser + yahboomcar
roslaunch yahboomcar_multi laser_usb_bringup_multi.launch ns:=robot1 # mono + laser + yahboomcar
roslaunch yahboomcar_multi laser_astrapro_bringup_multi.launch ns:=robot1 # Astra + laser + yahboomcar
```

Start the command (robot2 side). For the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch yahboomcar_multi laser_bringup_multi.launch ns:=robot2 # laser + yahboomcar
roslaunch yahboomcar_multi laser_usb_bringup_multi.launch ns:=robot2 # mono + laser + yahboomcar
roslaunch yahboomcar_multi laser_astrapro_bringup_multi.launch ns:=robot2 # Astra + laser + yahboomcar
```

More robots and so on.

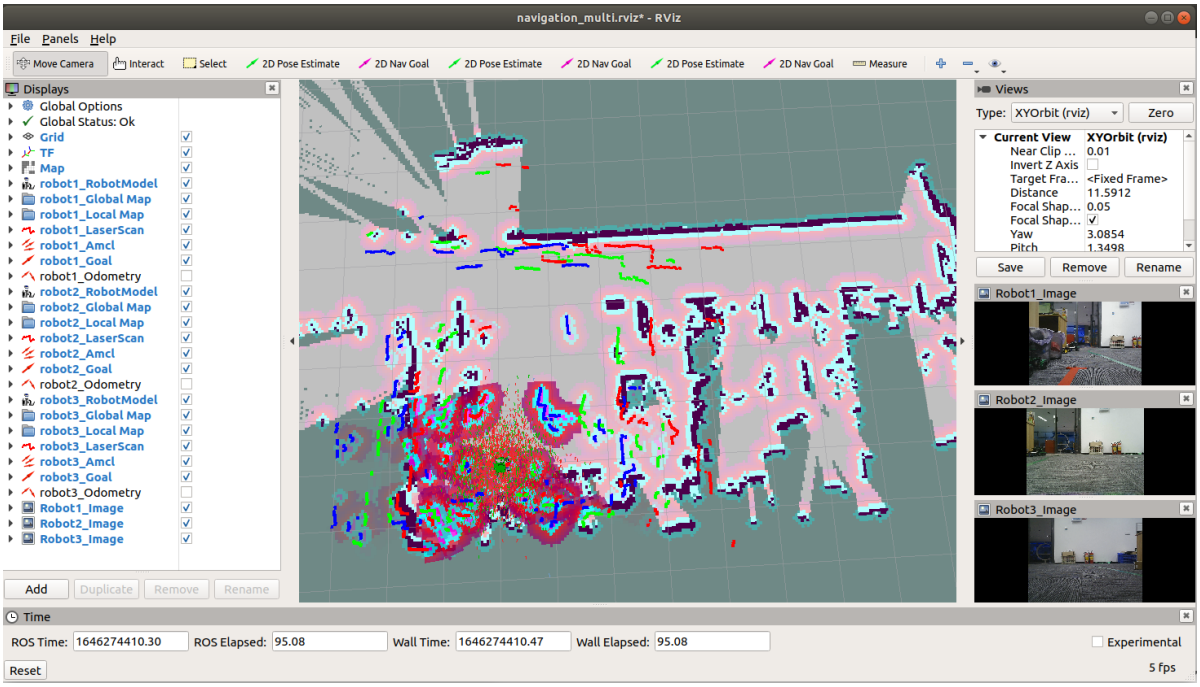
### 1.2.2. Enable multi-machine navigation

For the handle control opening process, please refer to the lesson [Multi-machine handle control].

Virtual machine side

```
roslaunch yahboomcar_multi yahboomcar_nav_multi.launch use_rviz:=true map:=my_map
```

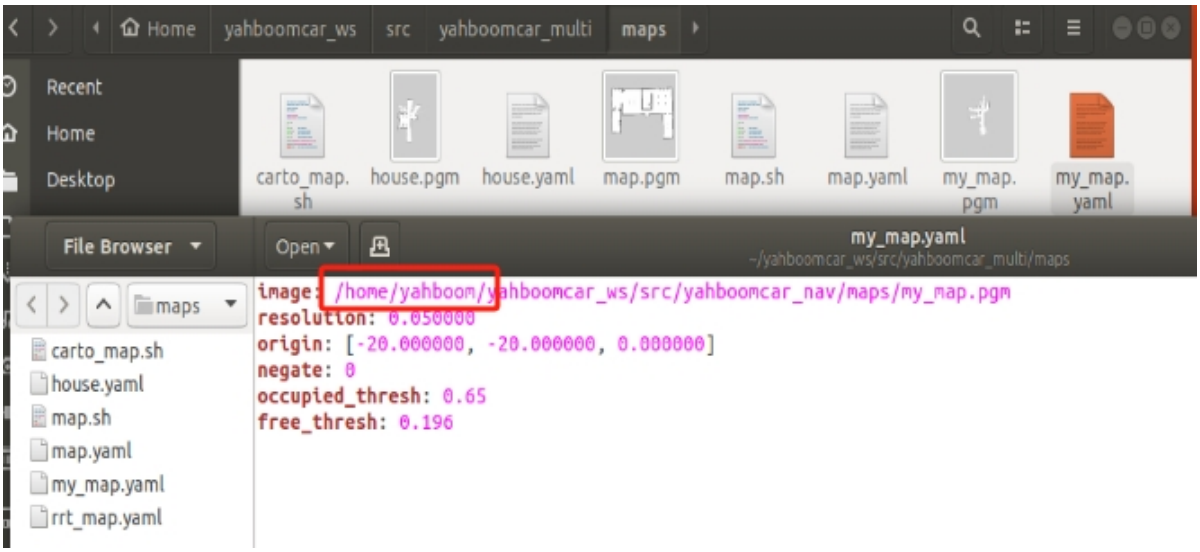
- [use_rviz] parameter: whether to open rviz.
- [map] parameters: map name, map to be loaded.



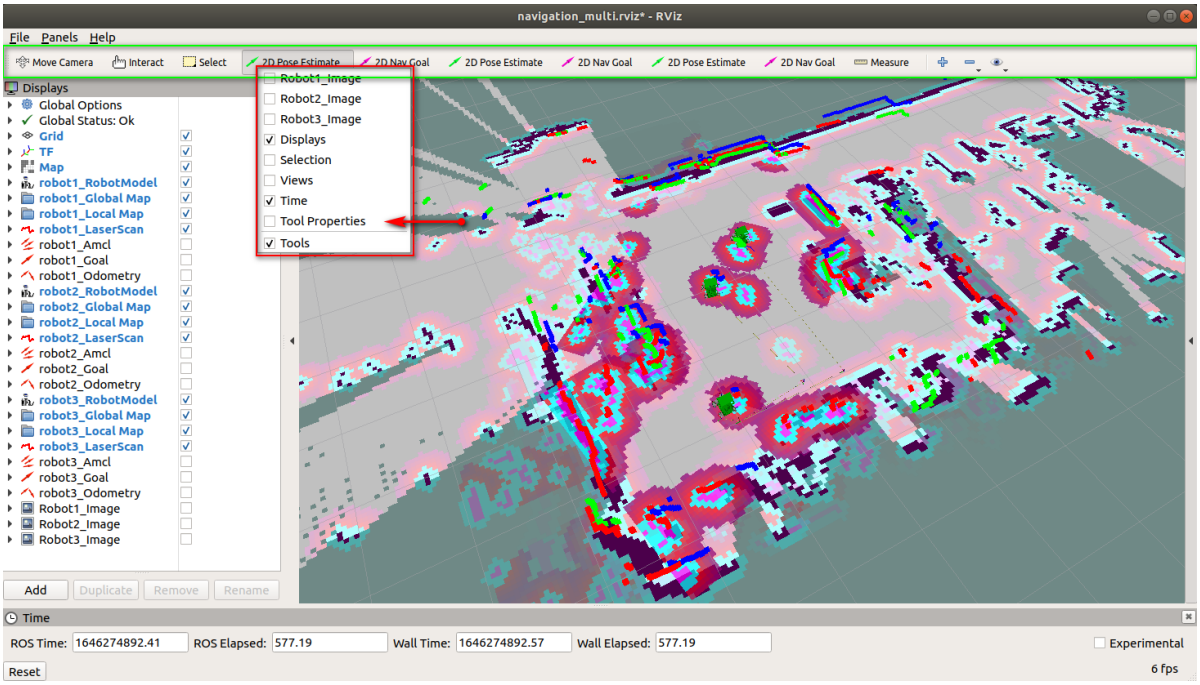

After turning on the multi-machine navigation, there is a dense mess. The pose of each robot is incorrect. First, the initial pose must be set for each robot.

**Tip: Start if it prompts my_map error**

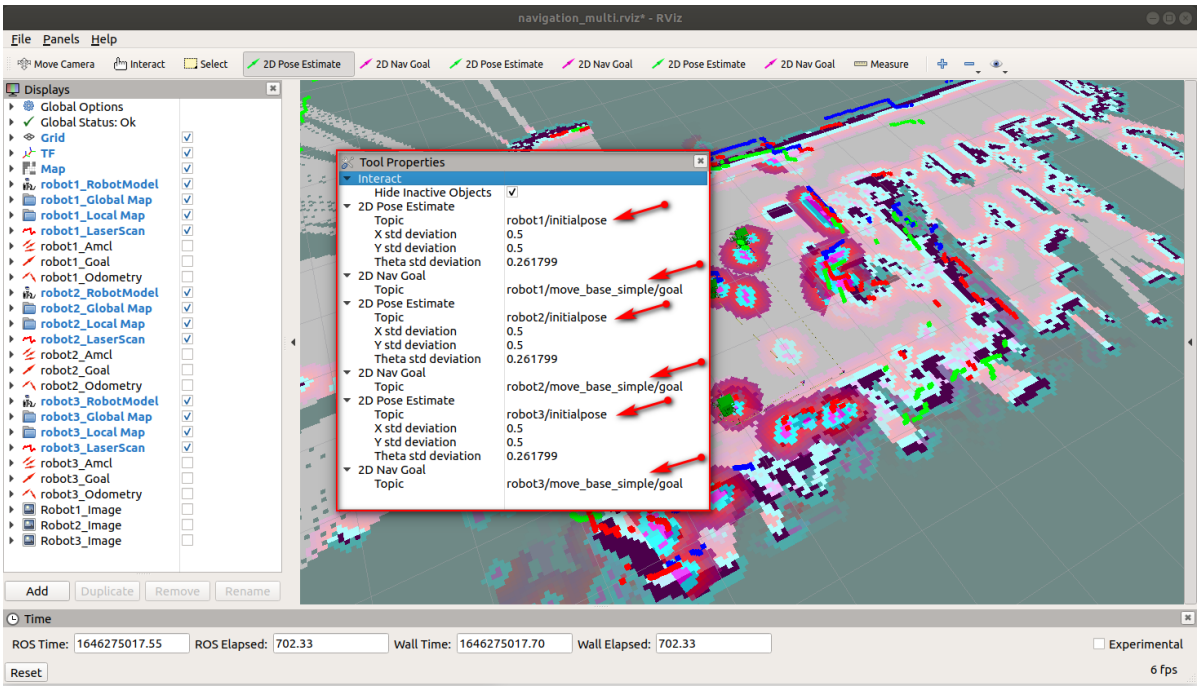
The terminal content path of the .yaml file should be changed correctly




## 1.2.3. Set communication topic




Right-click on the toolbar (green frame) and a dialog box (red frame) will pop up, as shown in the picture above. Right-click on [2D Pose Estimate], and then select [Tool Properties], as shown below, a dialog box will pop up.




From top to bottom [2D Pose Estimate] and [2D Nav Goal] correspond to the [rviz] toolbar from left to right. Set the topic name pointed by the red arrow. This method is quick and convenient.

What should I do if there are 4 or more robots and the initial pose and navigation icons are not enough?

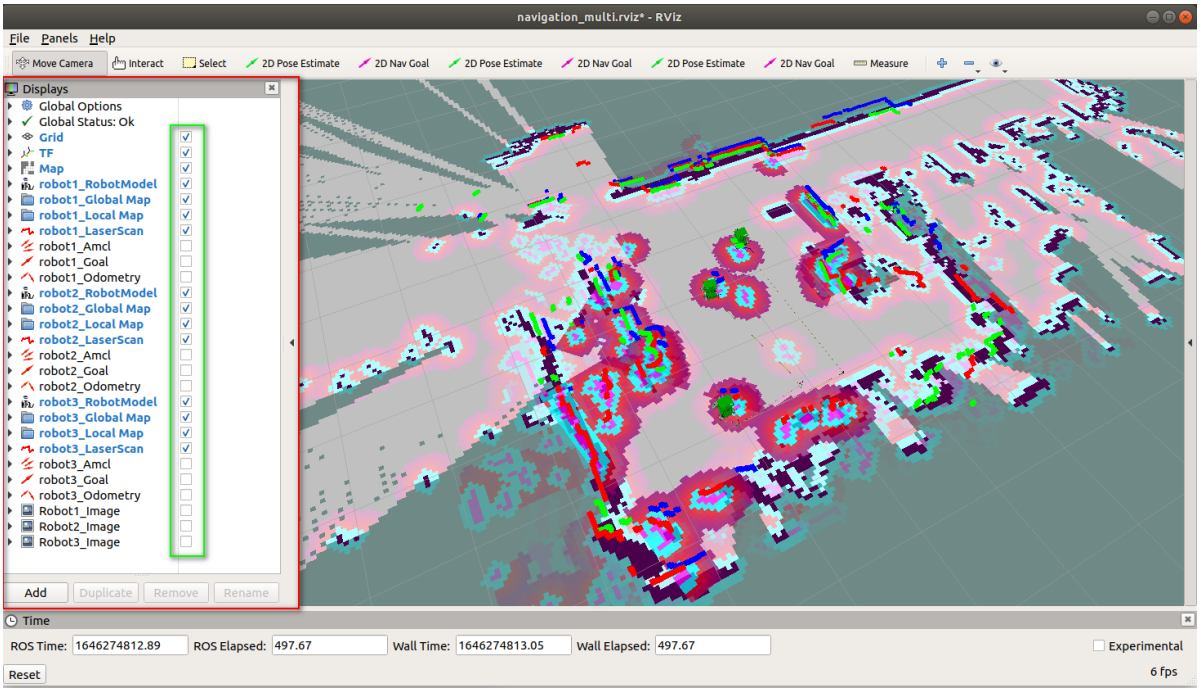
At this time, you need to open the [navigation_multi.rviz] file (open whichever rviz you use, take [navigation_multi.rviz] as an example) and find the following content. These contents correspond to the icons in the rviz toolbar one-to-one. If you want to add an icon, copy the entire [Class] content of that icon. **You must not miss anything, and the format must be correct.**

```
Tools:
      - Class: rviz/MoveCamera
      - Class: rviz/Interact
        Hide Inactive Objects: true
      - Class: rviz/Select
      - Class: rviz/SetInitialPose
        Theta std deviation: 0.2617993950843811
        Topic: robot1/initialpose
        X std deviation: 0.5
        Y std deviation: 0.5
      - Class: rviz/SetGoal
        Topic: robot1/move_base_simple/goal
      - Class: rviz/SetInitialPose
        Theta std deviation: 0.2617993950843811
        Topic: robot2/initialpose
        X std deviation: 0.5
        Y std deviation: 0.5
      - Class: rviz/SetGoal
        Topic: robot2/move_base_simple/goal
      - Class: rviz/SetInitialPose
        Theta std deviation: 0.2617993950843811
        Topic: robot3/initialpose
        X std deviation: 0.5
        Y std deviation: 0.5
      - Class: rviz/SetGoal
        Topic: robot3/move_base_simple/goal
```

```
    - Class: rviz/Measure
```

### 1.2.4. Set initial pose

There are too many robots and the information on the map is very complicated. In this case, you can uncheck the check mark behind the robot in the [Displays] display item list on the left and set the pose of each robot in turn. The effect after setting is as follows. Once set up, you can navigate.




- Use the [2D Pose Estimate] of the [rviz] tool to set the initial pose until the position of the car in the simulation is consistent with the position of the actual car.
- Click [2D Nav Goal] of the [rviz] tool, and then select a target point on the map where there are no obstacles. Release the mouse to start navigation. Only one target point can be selected, and it will stop when it is reached.

## 2.3, launch file

```
<launch>
     <arg name="first_robot1" default="robot1"/>
     <arg name="second_robot2" default="robot2"/>
     <arg name="third_robot3" default="robot3"/>
     <!-- Whether to open rviz || Whether to open rviz -->
     <arg name="use_rviz" default="true"/>
     <!-- Map name || Map name -->
     <arg name="map" default="my_map"/>
     <!-- Load map || Load map -->
     <node name="map_server" pkg="map_server" type="map_server" args="$(find
yahboomcar_nav)/maps/$(arg map).yaml"/>
     <node pkg="rviz" type="rviz" name="rviz" required="true"
           args="-d $(find yahboomcar_multi)/rviz/navigation_multi.rviz"
if="$(arg use_rviz)"/>
     <!-- Multi machine handle control || Multi machine handle control-->
     <include file="$(find yahboomcar_multi)/launch/joy_multi.launch"/>
     <!-- Mobile app node || Mobile app node-->
     <include file="$(find yahboomcar_nav)/launch/library/app.launch"/>
```

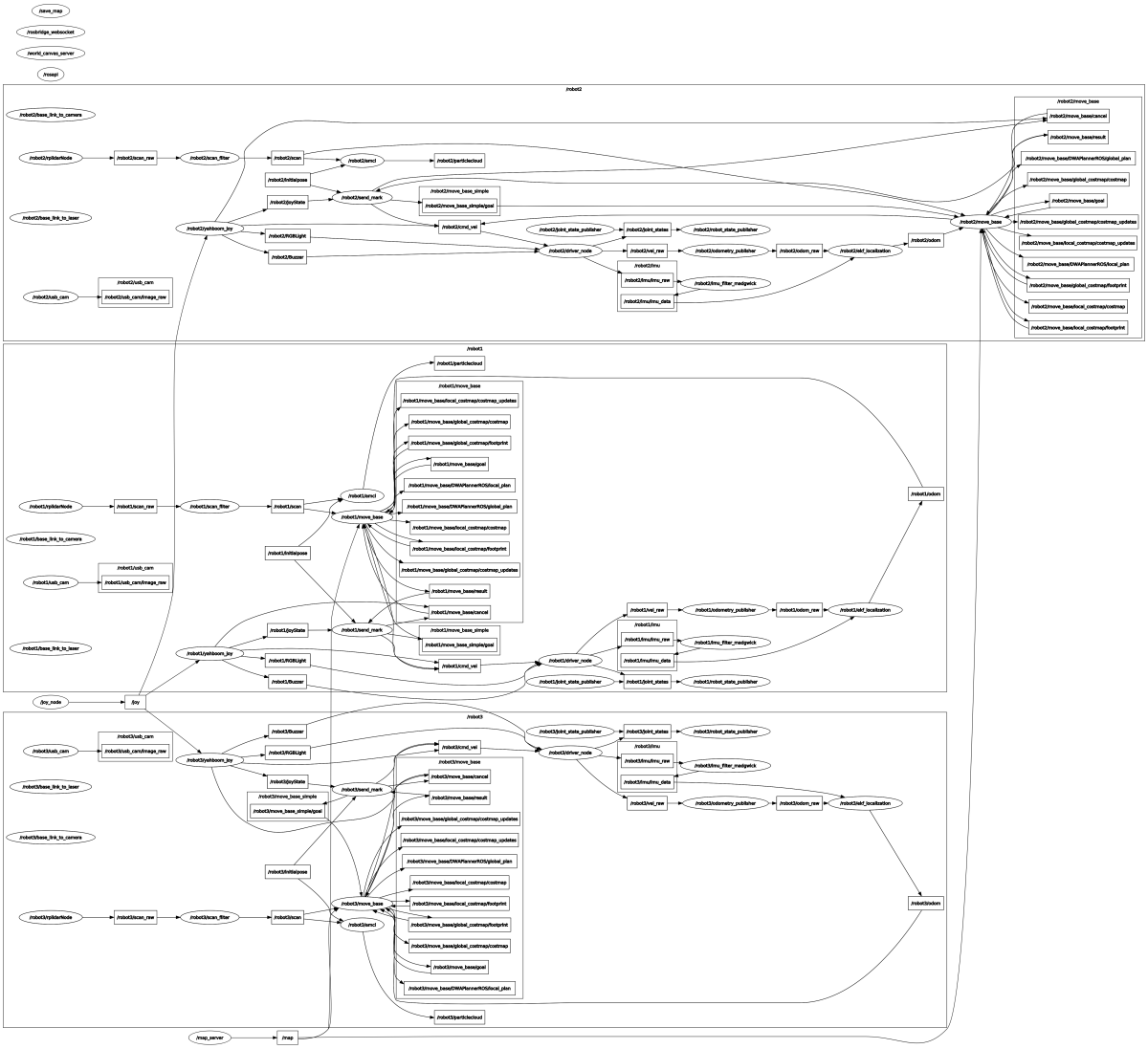
```
    <!-- ############################ first_robot1 ################
############ -->
    <include file="$(find
yahboomcar_multi)/launch/library/move_base_multi.launch">
        <arg name="ns" value="$(arg first_robot1)"/>
    </include>
    <!-- ############################ second_robot2 ###############
############ -->
    <include file="$(find
yahboomcar_multi)/launch/library/move_base_multi.launch">
        <arg name="ns" value="$(arg second_robot2)"/>
    </include>
    <!-- ############################ third_robot3 ################
############ -->
    <include file="$(find
yahboomcar_multi)/launch/library/move_base_multi.launch">
        <arg name="ns" value="$(arg third_robot3)"/>
    </include>
</launch>
```

You can modify it according to your own needs. If there are 4 or more robots, just follow the case of the first 3 robots and add relevant content.

## 2.4. Framework analysis

- Node view

```
rqt_graph
```



There are a lot of node diagrams, which look complicated, but are actually very organized. The internal nodes for each robot are almost the same, and the navigation of a single robot is also the same. It should be noted here that the map only needs to be loaded once, and does not need to be loaded for every robot. At the same time, I also enabled the node controlled by the handle. One handle controls multiple robots at the same time, and only needs to start the [joy_node] node once.

- View tf tree

```
rosrun rqt_tf_tree rqt_tf_tree
```

There are also a lot of tf coordinate systems, so you need to zoom in to see them. There is only one [map] globally, and each [amcl] locates each robot separately. From then on, it is the same as single robot navigation.