1. Multi-robot control

1.1 Achievement effect

When you have two or more transbot se, after running the multi-robot control function, you can control all transbot se at the same time.

1.2 Preparations before realization

Before realizing this function, the handle receiver needs to be connected to the virtual machine of the computer, and the virtual machine and transbot realize multi-machine communication. Let the transbot be the slave and the virtual machine be the master. Therefore, the program controlled by the handle is on the virtual machine, which is equivalent to that the virtual machine is the central command. Let's take two transbot se as an example to demonstrate this function.

1.3 Start

1.3.1 virtual machine

1. Start roscore

```
roscore
```

2. Start the handle control node

Connect the handle receiver to the USB port of the virtual machine and start the command

```
roslaunch transbot_ctrl transbot_joy_multi.launch
```

1.3.2 transbot se

1. transbot se1 starts the chassis control program

```
roslaunch transbot_mulity transbot_mulity_control.launch namespace:=robot1
```

2. transbot se2 starts the chassis control program

```
roslaunch transbot_mulity transbot_mulity_control.launch namespace:=robot2
```

More bots and so on.

1.3.3 Control

On the virtual machine side, the handle controls the direction, and the two cars will move at the same time.

1.3.4 launch file analysis

launch file location: /home/jetson/transbot_ws/src/transbot_ctrl/launch/transbot_joy_multi.launch

```
<launch>
    <arg name="first_robot1" default="robot1"/>
    <arg name="second_robot2" default="robot2"/>
    <arg name="third_robot3" default="robot3"/>

    <param name="use_sim_time" value="false"/>
    <!-- ############################## first_robot1 ##############################
-->

   <include file="$(find transbot_ctrl)/launch/transbot_joy_multi _base.launch">
    <arg name="namespace" default="$(arg first_robot1)"/>
</include>

<!-- ############################## second_robot2 ############################## --
>

<include file="$(find transbot_ctrl)/launch/transbot_joy_multi _base.launch">
    <arg name="namespace" default="$(arg second_robot2)"/>
</include>

<!-- ############################## third_robot3 ############################## --
>

<include file="$(find transbot_ctrl)/launch/transbot_joy_multi _base.launch">
    <arg name="namespace" default="$(arg third_robot3)"/>
</include>

</launch>
```

If there are 2 robots, just comment out the part of the third one, and it will not affect if you don't comment it out; if there are 4 or more robots, you can increase according to the first 3 samples.

The biggest difference of the chassis startup file here is to use the namespace namespace. With the namespace, there will be no conflict of the same node name when starting the node. For example, in the above example, if the respective namespace "namespace:=robot1 namespace:=robot2" is not added. Then after starting a node, it will cause a conflict caused by starting the transbot_node node at the same time, and the program will report an error and exit.