4. Voice-controlled robotic arm to pick up objects

4.1. Function description

Through interaction with the voice module, the rosmaster can be used to navigate to any of the five positions 1/2/3/4/origin. After reaching the destination, the robotic arm on the rosmaster can also be used to pick up the passed object, and then The command lets it navigate to any of the five positions 1/2/3/4/origin, and put the object down and wait for the next command.

4.2. Start

```
#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
```

4.2.1. Function package location

```
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/
```

4.2.2. Start

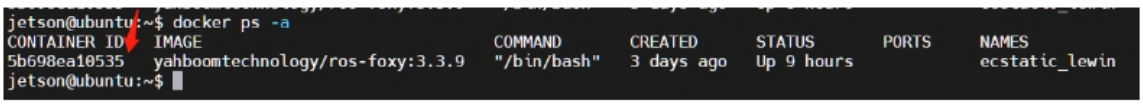
Note: **Multi-machine communication configuration** needs to be implemented between rosmaster X3Plus and the virtual machine. You can refer to the content of "**yahboomcar Course\06Linux Operating System\04. Multi-machine Communication Configuration** for configuration

```
#rosmaster X3Plus launch
roslaunch yahboomcar_voice_ctrl voice_transport_base.launch
```

<PI5 needs to open another terminal to enter the same docker container

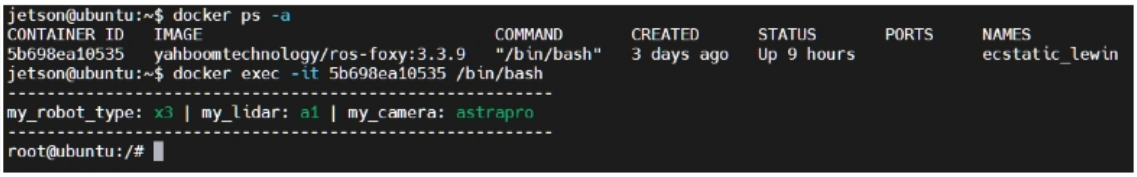
1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```



2. Now enter the docker container in the newly opened terminal:

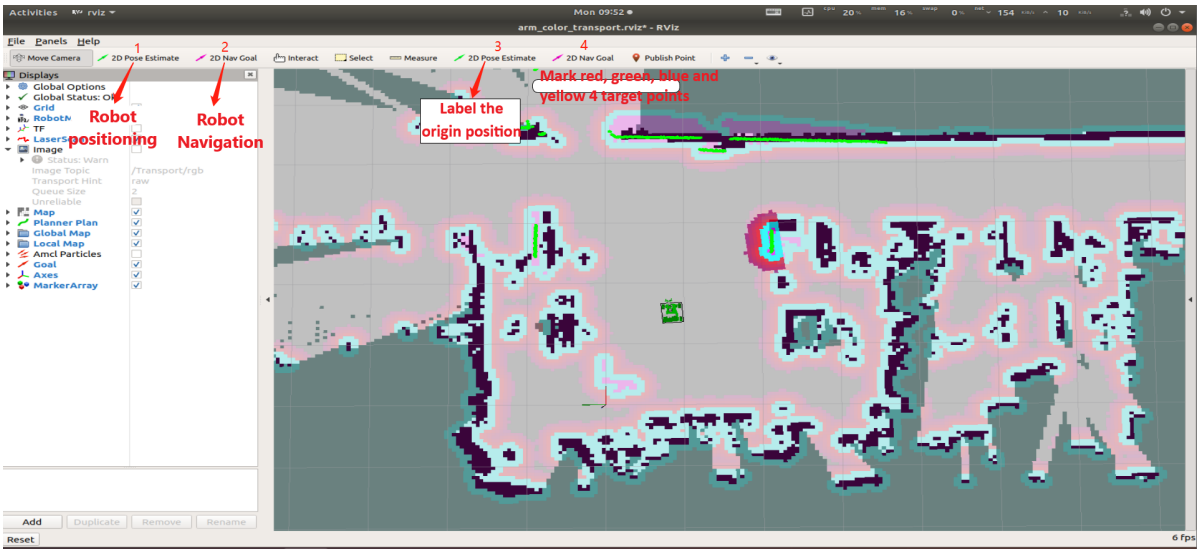
```
docker exec -it 5b698ea10535 /bin/bash
```



After successfully entering the container, you can open countless terminals to enter the container.

```
python3
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/scripts/voice_ctrl_takethings.py
#Virtual machine startup
roslaunch arm_color_transport transport_rviz.launch
```

Each tool annotation on the virtual machine rviz is as shown in the figure below,



- step 1

  In rviz, use tool 1 to calibrate and position the rosmaster

- Step 2

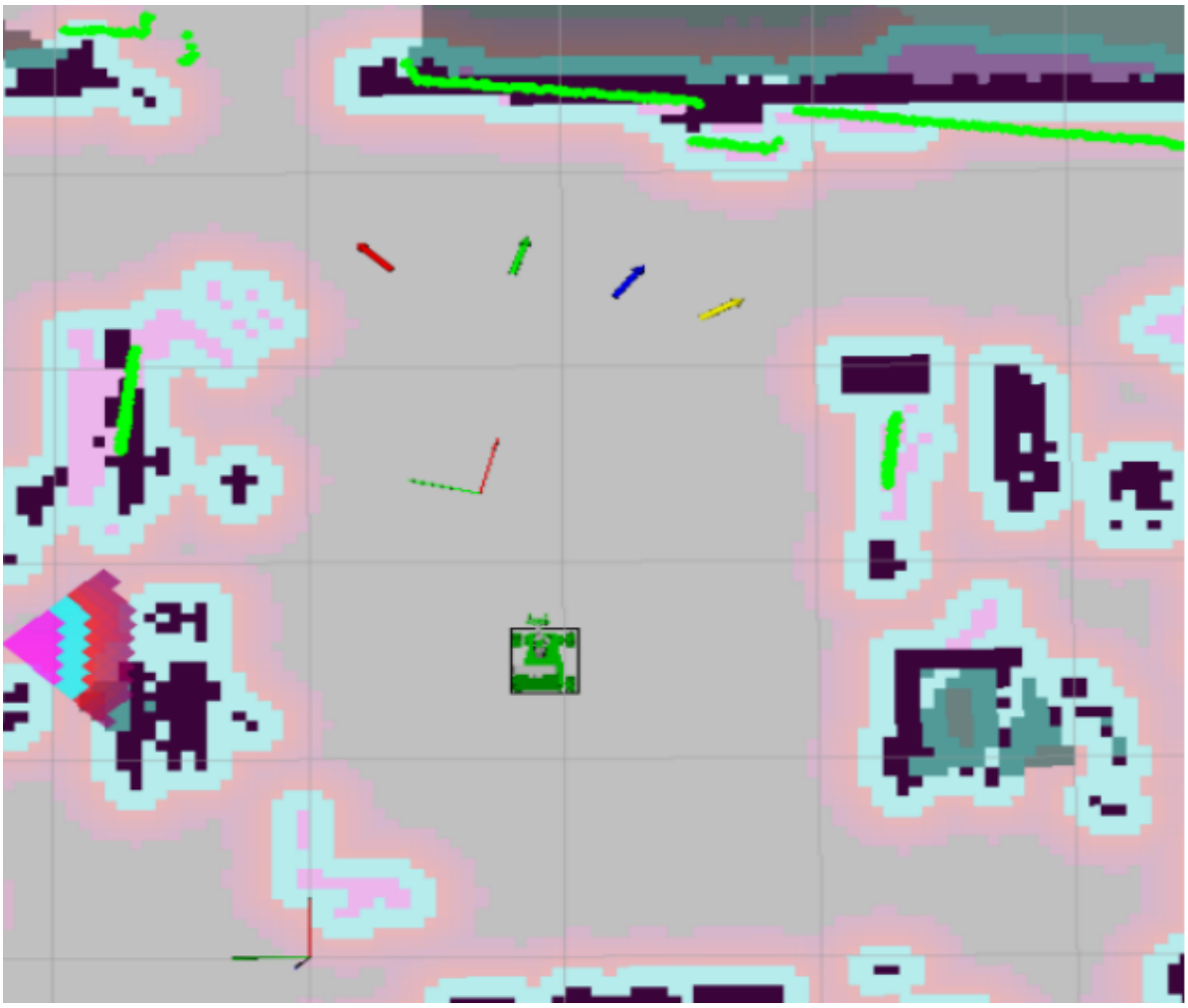
  In rviz, use 3 tools to mark the origin of rosmaster. This origin is the position where the car automatically returns after putting down the color block.

- Step 3

  In rviz, use 4 tools to mark red, green, blue, and yellow destinations on the map in sequence. After the rosmaster's robotic arm gripper picks up the color blocks,

  It will navigate to the destination corresponding to the color of the color block. It is best to mark the four points in a row, and the distance between the front and back should not be too far apart.

  Of course, when planning a path, you may hit color blocks.



Note: **No. 1** corresponds to the **red** marked point; **No. 2** corresponds to the **green** marked point; **No. 3** corresponds to ** The blue ** mark point; the **No. 4** position corresponds to the **yellow** mark point;

- Step 4 (Take going to No. 1 first and then taking the items and placing them in No. 2 as an example)

  Say "Hello, Xiaoya" to rosmaster, wake up the voice module, and then say to it "Navigate to position 1". After backing up and adjusting the position, the voice module will say "Okay, going to position 1" and navigate independently. to position 1; after arriving at the destination, the buzzer will beep once, and then say "Clip (home, here is the first tone, the same as home) square"; after the module hears it, the buzzer will The device will beep once to indicate that it has received the command, and then pass the object to the gripper. After the second beep, the gripper will pick up the object and announce "I just picked it up"; then, say to it " "Navigate to position 2". After the car adjusts its position backwards, it will announce "OK, going to position 2" and autonomously navigate to position 2; after arriving at position 2, the buzzer will beep once, indicating that it has reached the destination. to the ground, then grab down, put the object down and announce "Place completed".

4.2.3. Core code analysis voice_ctrl_takethings.py

- code path

  ```
  ~/yahboomcar/src/yahboomcar_voice_ctrl/scripts
  ```

- Core code analysis

  1), several important mode states

```
self.model== "Init": initialization state
self.model == "Grip_Target": ready to grip state
self.model == "next_points": Navigate to the next target point state
self.model == "Grip_down": grab and release state
```

2), several important flag flags

```
self.clip determines whether the object is clipped. True means the object is
clipped. Fasle means the object is not clipped.
self.come_back_flag Whether to return to the origin flag, True means going
back, False means not going back
self.ros_nav.goal_result Whether the navigation target point has been
reached, 0 means not arrived, 3 means arrived
```

3), several important functions

```
next_points: Recognize speech and navigate to the next target point
comeback: Return to origin function
Grip_down: grab and release objects
Grip_Target: Recognize speech, grip objects or navigate to the next target
point
```

- Program Description

  In the main function, directly enter the process function. In the process function, determine the mode status and flag value, and then execute the corresponding function or program.

4.3. Program flow chart