Multimodal visual understand+SLAM navigation + visual line patroling(Text Version)

1. Course Content

1. Learn to use the robot's visual understanding, line following, SLAM navigation, and other complex functional cases.
2. Study the key source code newly introduced in the tutorial.
3. **Note: This section requires you to complete the map file configuration as described in the previous section [7. Multimodal Visual Understand + SLAM Navigation].**

2. Preparation

2.1 Content Description

> This lesson uses the Raspberry Pi as an example. For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [4.Enter Docker (For JETSON Nano and RPi 5)]**. For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

📝 This example uses `model:"qwen/qwen2.5-vl-72b-instruct:free","qwen-vl-latest"`

⚠ The responses from the big model for the same test command may not be exactly the same and may differ slightly from the screenshots in the tutorial. To increase or decrease the diversity of the big model's responses, refer to the section on configuring the decision-making big model parameters in **[03.AI Model Basics] -- [5.Configure AI large model]** .

3. Run Case

3.1 Starting the Program

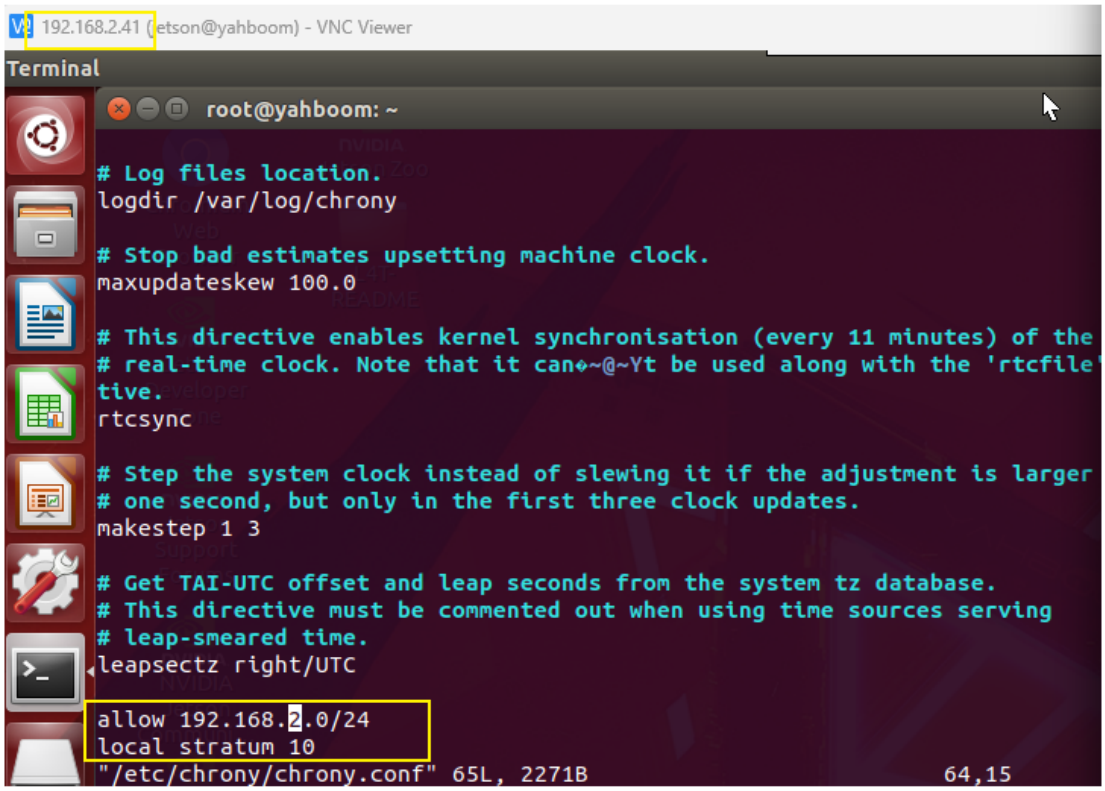
3.1.1 Jetson Nano board Startup Steps:

Due to Nano performance issues, navigation-related nodes must be run on a virtual machine. Therefore, navigation performance is highly dependent on the network. We recommend running in an indoor environment with a good network connection. You must first configure the following:

- **jetson nano (requires entering Docker)**

Open a terminal and enter

```
sudo vi /etc/chrony/chrony.conf
```



Add the following two lines to the end of the file: (Fill in 192.168.2.0/24 based on the actual network segment; this example uses the current board IP address of 192.168.2.41.)

```
allow 192.168.x.0/24 #x indicates the corresponding network segment
local stratum 10
```

After saving and exiting, enter the following command to take effect:

```
sudo service chrony restart
```

- **Virtual Machine**

Open a terminal and enter

```
echo "server 192.168.2.41 iburst" | sudo tee
/etc/chrony/sources.d/jetson.sources
```

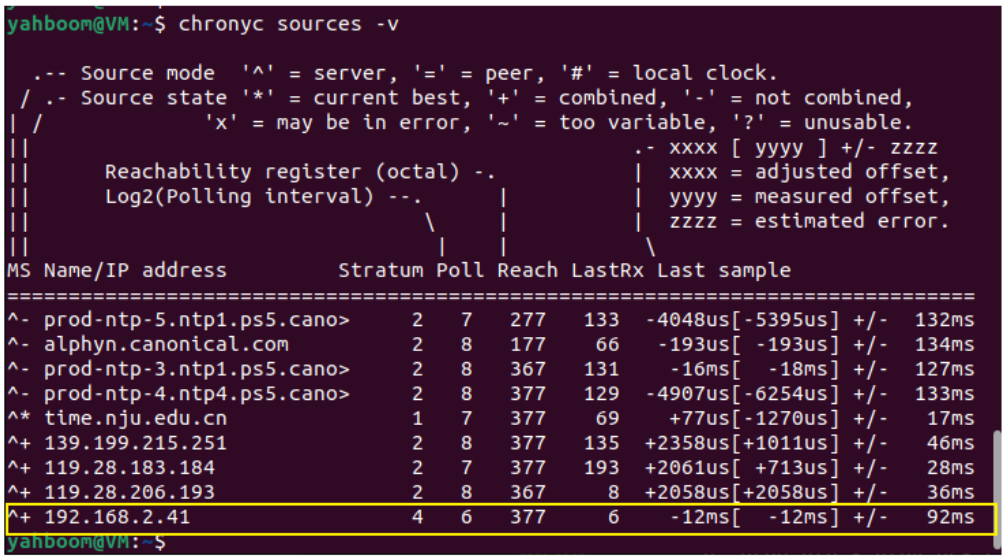
The 192.168.2.41 entry above is the board's IP address.

Enter the following command to take effect.

```
sudo chronyc reload sources
```

Enter the following command again to check the latency. If the IP address appears, it's normal.

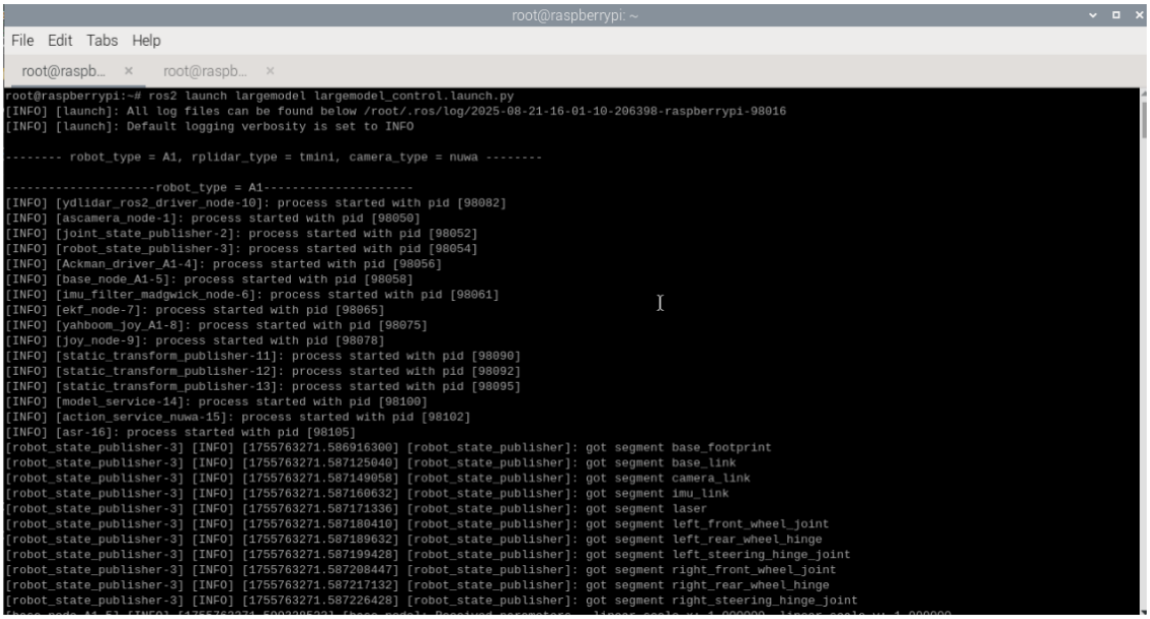
```
chronyc sources -v
```



- **Start the Program**

On the mainboard, open a terminal in Docker and enter the following command:

```
ros2 launch largemodel largemodel_control.launch.py
```



On the virtual machine, create a new terminal and start.

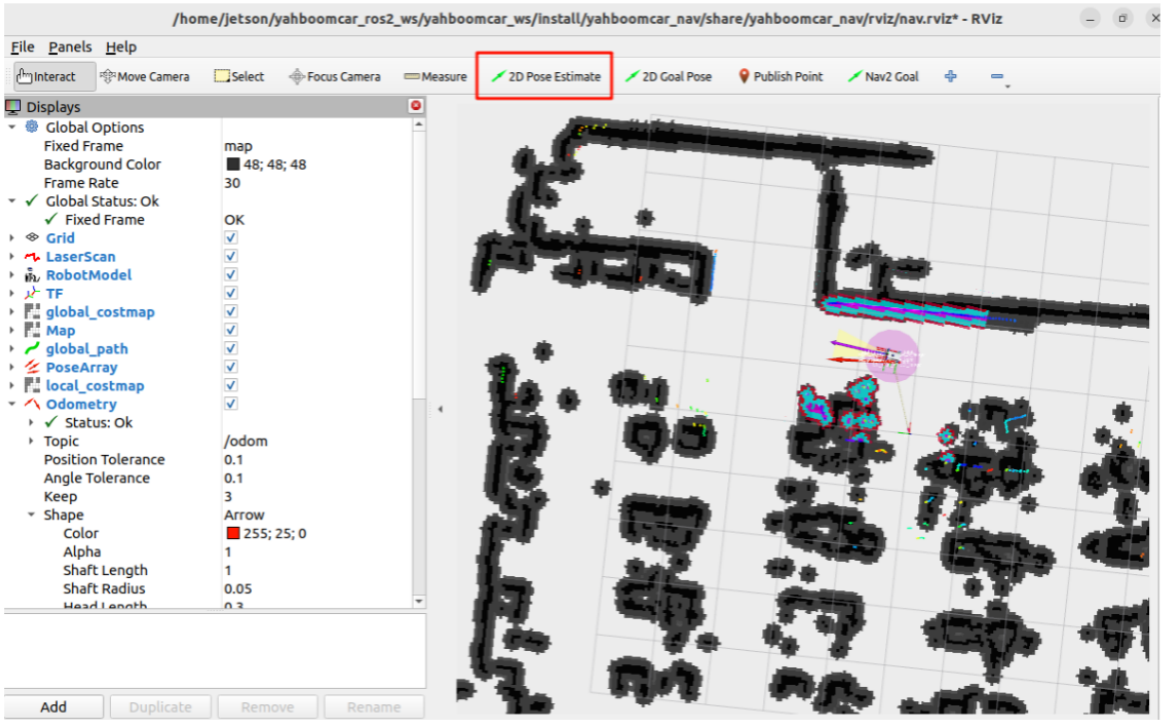
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

Wait for the navigation algorithm to start before displaying the map. Then open a VM terminal and enter:

```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

Note: When running the car's mapping function, you must enter the save map command on the VM to save the navigation map.

After that, follow the navigation function startup process to initialize positioning. The rviz2 visualization interface will open. Click **2D Pose Estimate** in the upper toolbar to select it. Roughly mark the robot's position and orientation on the map. After initializing positioning, preparations are complete.

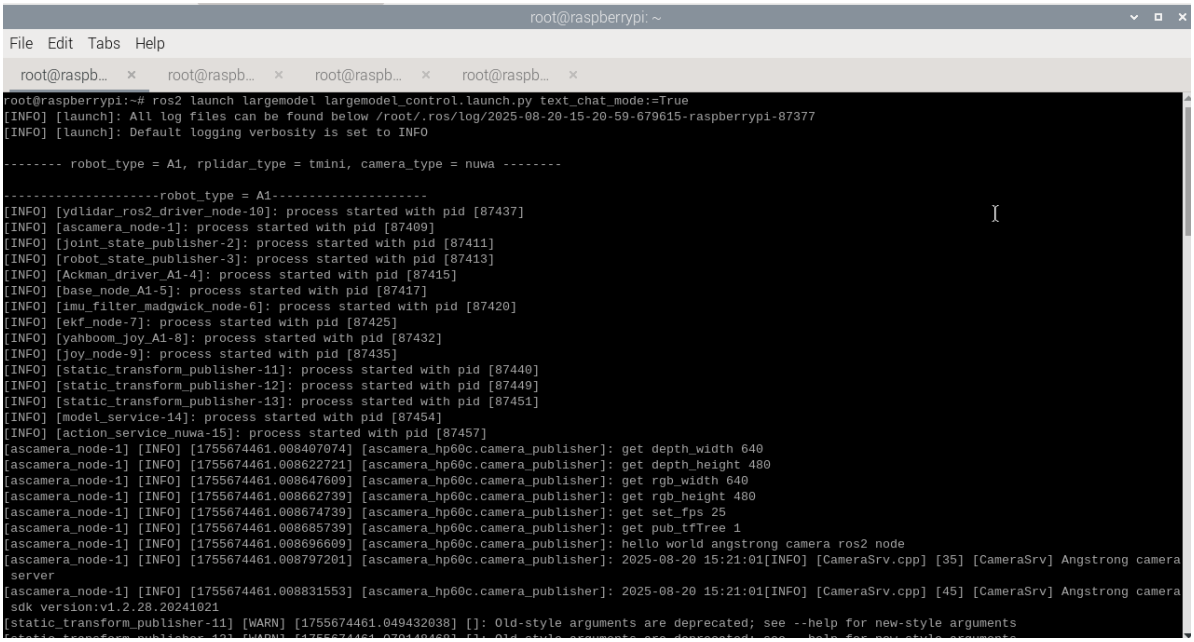


3.1.2 Other main control startup steps:

For the Raspberry Pi 5, you need to first enter the Docker container; the ORIN board does not require this.

Open a terminal in Docker and enter the following command:

```
ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
```



Create a new terminal on the virtual machine and start the command.** (For Raspberry Pi and Jetson Nano, it is recommended to run the visualization in the virtual machine)**

```
ros2 launch yahboomcar_nav display_nav_launch.py
```

Wait for the navigation algorithm to start and the image will be displayed. Then open the Docker terminal and enter the following command:

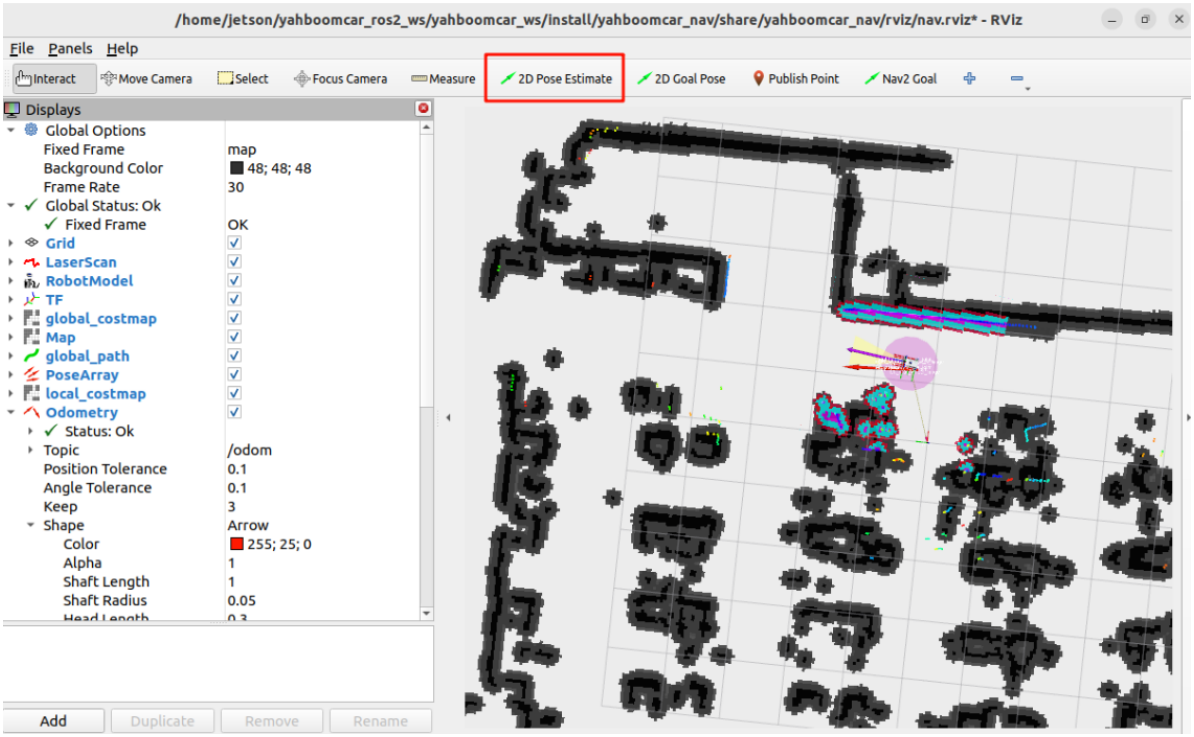
```
#Choose one of the two navigation algorithms
#Standard navigation
ros2 launch yahboomcar_nav navigation_teb_launch.py

#Fast relocalization navigation(RDKX5, Raspberry Pi 5, and Jetson Nano are not supported)
ros2 launch yahboomcar_nav localization_imu_odom.launch.py use_rviz:=false load_state_filename:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/yahboomcar.pbstream

ros2 launch yahboomcar_nav navigation_cartodwb_launch.py maps:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/yahboomcar.yaml params_file:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/params/cartoteb_nav_params.yaml
```

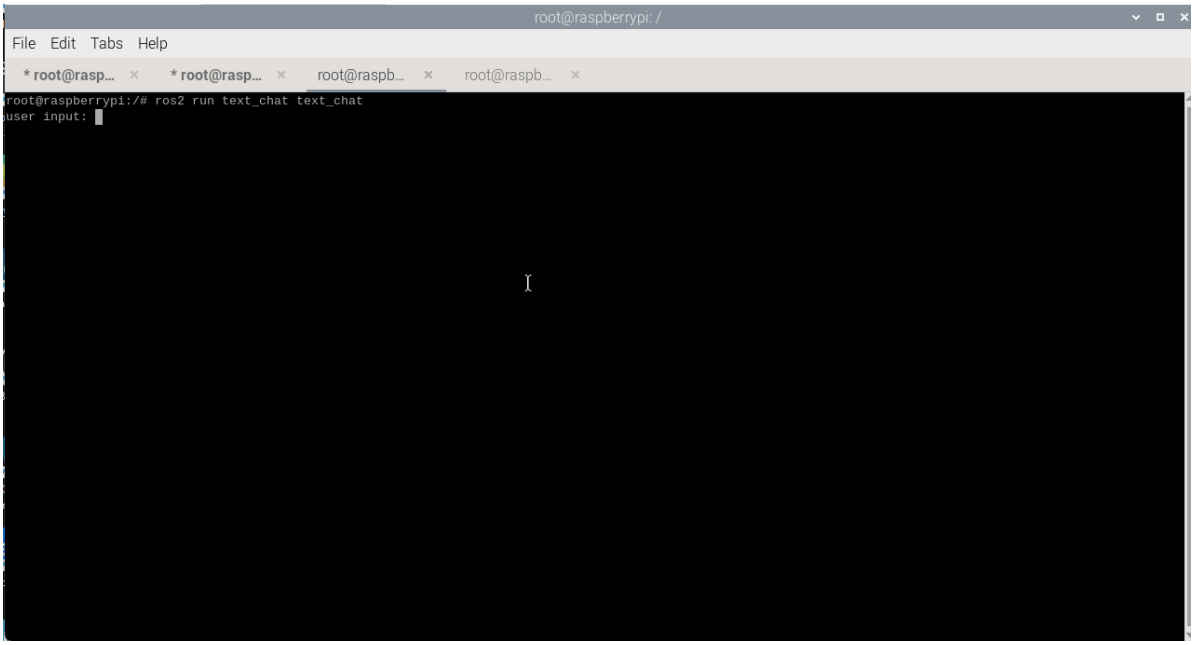
Note: yahboomcar.yaml and yahboomcar.pbstream must be mapped simultaneously, meaning they are the same map. Refer to the cartograph mapping algorithm for saving maps.

After that, follow the navigation function startup process to initialize positioning. The rviz2 visualization interface will open. Click **2D Pose Estimate** in the upper toolbar to enter the selection state. Roughly mark the robot's position and orientation on the map. After initial positioning, preparations are complete.



Open the same docker in multiple terminals and start it.

```
ros2 run text_chat text_chat
```



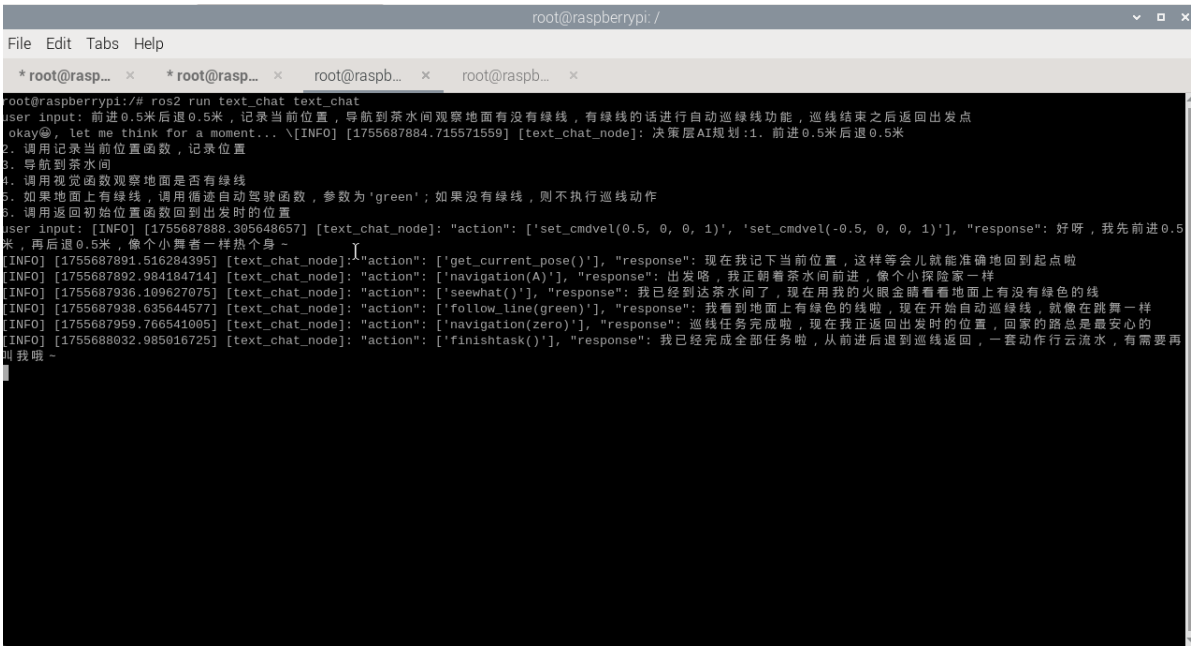
3.2 Test Case

Here are some reference test cases; users can create their own dialogue commands.

- Move forward 0.5 meters and back 0.5 meters, record the current location, navigate to the tea room, and observe if there's a green line on the floor. If there is, automatically patrol the green line. After patrolling, return to the starting point.

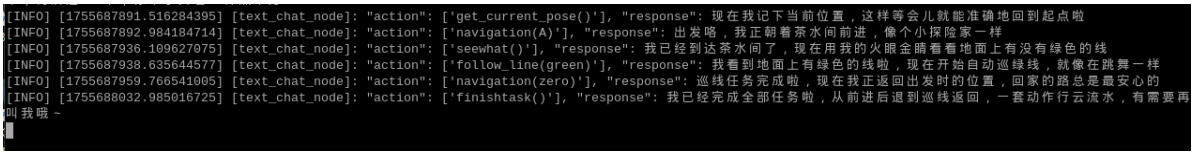
3.2.1 Case 1

In the terminal, enter "Move forward 0.5 meters and back 0.5 meters, record the current location, navigate to the tea room, and observe if there's a green line on the floor. If there is, automatically patrol the green line. After patrolling, return to the starting point." The terminal prints the following information.

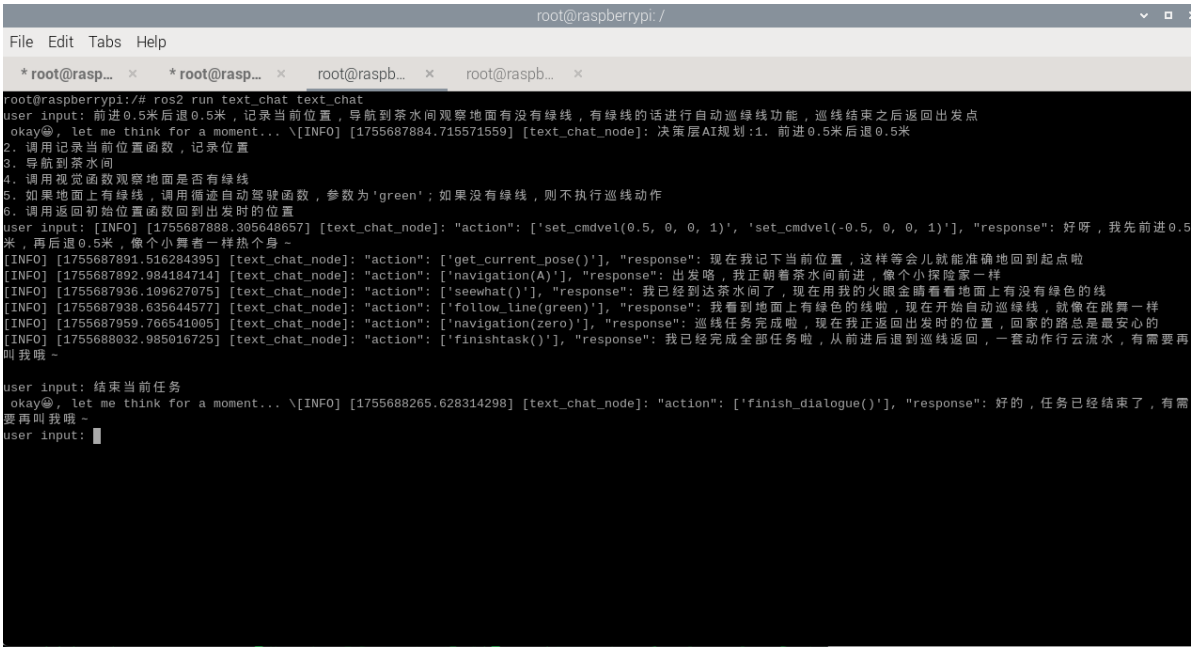


The decision-making layer model outputs the planned task steps:

The execution layer model will then execute the task steps:



After completing the task, the robot will enter a waiting state. During this time, commands are directly passed to the execution layer model, and all conversation history is retained. You can press the `ENTER` key in the terminal to continue the conversation and enter the "End current task" command to terminate the current task cycle and start a new one.



4. Source Code Analysis

Source code location:

Jetson Orin Nano host:

```
#NUWA camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_nuwa.py
#USB camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_usb.py
```

RDK X5：

```
#NUWA camera user
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_nuwa.py
#USB camera user
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_usb.py
```

Jetson Nano, Raspberry Pi host:

You need to first enter Docker.

```
#NUWA Camera User
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_nuwa.py
#USB Camera User
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_usb.py
```

This example uses the **seewhat**, **navigation**, **load_target_points**, **get_current_pose**, and **follow_line(self, color)** methods from the **CustomActionServer** class. Most of these methods have been explained in the **Multimodal Visual Understanding + SLAM Navigation** section. For a detailed explanation of the **follow_line(self, color)** function, refer to the **Multimodal Visual**

Understanding + Visual Line Patrolling section.