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

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

- 1. Course Content
- 2. Preparation
  - 2.1 Content Description
- 3. Run Case
  - 3.1 Starting the Program
    - 3.1.1 Jetson Nano board Startup Steps:
    - 3.1.1 Raspberry Pi PI5 and ORIN board startup steps:
  - 3.2 Test Case
    - 3.2.1 Example
- 4. Source Code Analysis

1. Course Content

1. Learn to use the robot's visual understanding, line tracking, 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 have completed the map file configuration as described in the previous section on [7.Multimodal visual understand+SLAM navigation]**

2. Preparation

2.1 Content Description

> This section uses the Jetson Orin Nano as an example. For users of the gimbal servo USB camera version, this is used 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 section in the terminal. For instructions on entering the Docker container on the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [5.Enter Docker (For JETSON Nano and RPi 5)]**. For Orin boards, simply open a terminal and enter the commands mentioned in this section. >

📝 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. If you need 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 the **[03.AI Model Basics] -- [5.Configure AI large model]**.

⚡ It is recommended that you first try the previous visual example. This example adds voice functionality to the singleton, and while the functionality is largely the same, I will not elaborate on the program implementation, code debugging, or results in detail! ! !

3. Run Case

3.1 Starting the Program

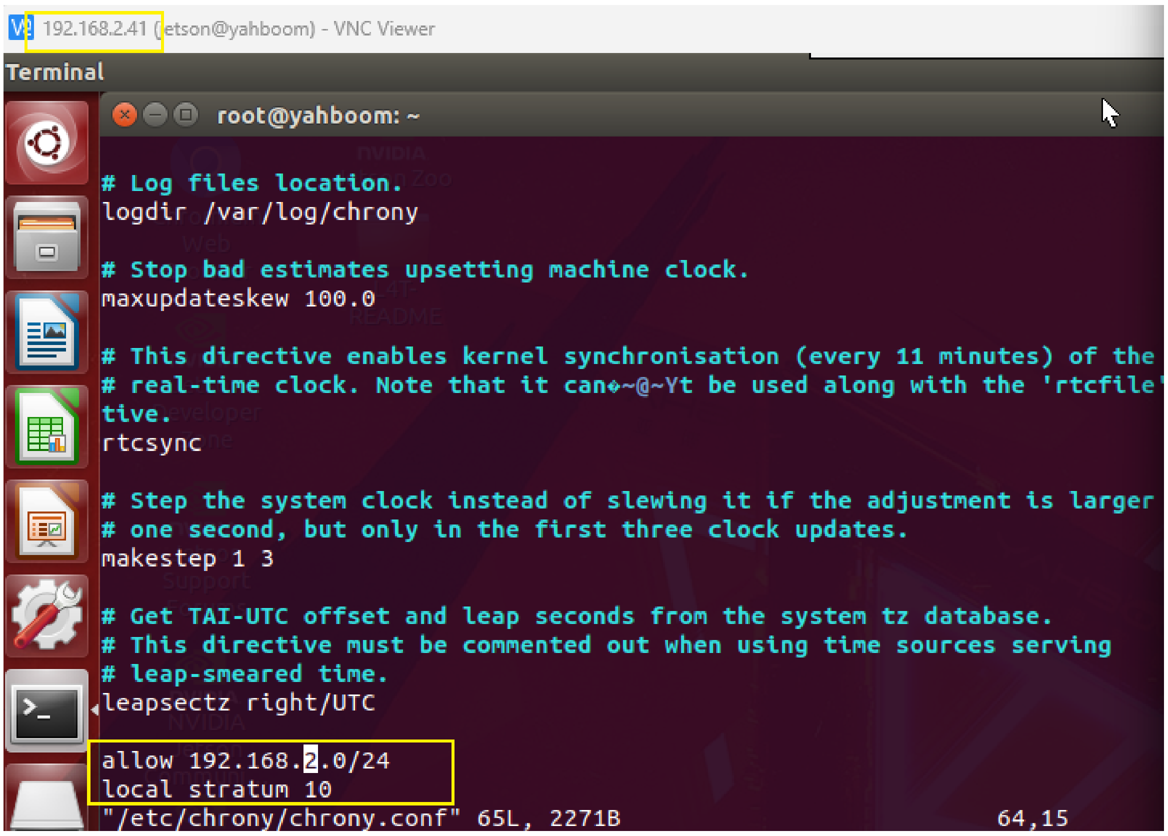
3.1.1 Jetson Nano board Startup Steps:

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:

- **Board Side (Requires Docker)**

Open a terminal and enter

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



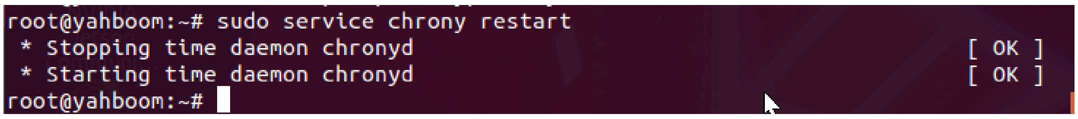
```
# Log files location.
logdir /var/log/chrony

# Stop bad estimates upsetting machine clock.
maxupdateskew 100.0

# This directive enables kernel synchronisation (every 11 minutes) of the
# real-time clock. Note that it can�~@~Yt be used along with the 'rtcfile'
tive.
rtcsync

# Step the system clock instead of slewing it if the adjustment is larger
# one second, but only in the first three clock updates.
makestep 1 3

# Get TAI-UTC offset and leap seconds from the system tz database.
# This directive must be commented out when using time sources serving
# leap-smeared time.
leapsectz right/UTC

allow 192.168.2.0/24
local stratum 10
"/etc/chrony/chrony.conf" 65L, 2271B                              64,15
```

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
```



```
root@yahboom:~# sudo service chrony restart
 * Stopping time daemon chronyd                                    [ OK ]
 * Starting time daemon chronyd                                    [ OK ]
root@yahboom:~#
```

- **Virtual Machine**

Open a terminal and enter

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

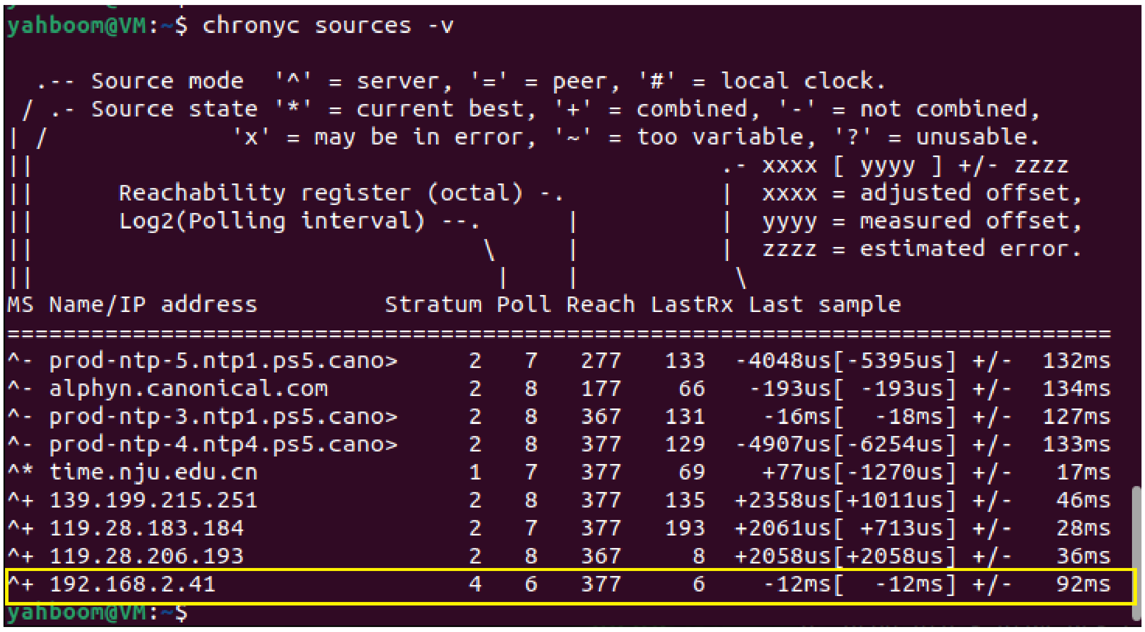
The 192.168.2.41 entry above is the board's IP address.

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
```

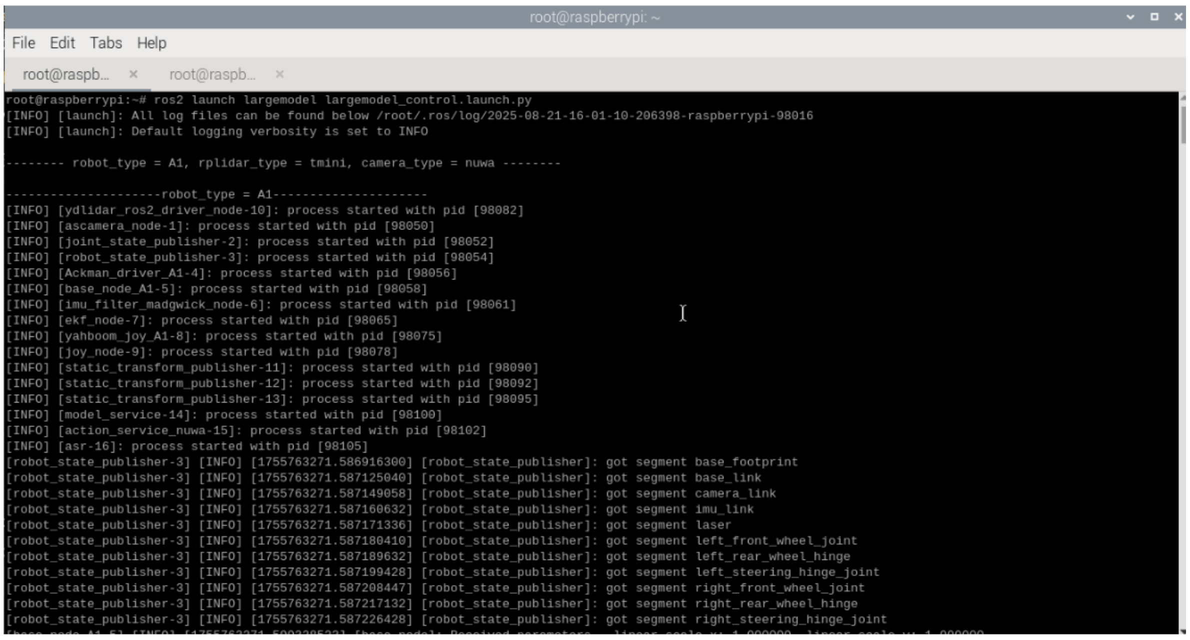


```
yahboom@VM:~$ chronyc sources -v

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /             'x' = may be in error, '~' = too variable, '?' = unusable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- prod-ntp-5.ntp1.ps5.cano>     2   7   277   133  -4048us[-5395us] +/-  132ms
^- alphyn.canonical.com          2   8   177    66   -193us[ -193us] +/-  134ms
^- prod-ntp-3.ntp1.ps5.cano>     2   8   367   131    -16ms[  -18ms] +/-  127ms
^- prod-ntp-4.ntp4.ps5.cano>     2   8   377   129  -4907us[-6254us] +/-  133ms
^* time.nju.edu.cn               1   7   377    69    +77us[-1270us] +/-   17ms
^+ 139.199.215.251               2   8   377   135  +2358us[+1011us] +/-   46ms
^+ 119.28.183.184                2   7   377   193  +2061us[ +713us] +/-   28ms
^+ 119.28.206.193                2   8   367     8  +2058us[+2058us] +/-   36ms
^+ 192.168.2.41                  4   6   377     6    -12ms[  -12ms] +/-   92ms
yahboom@VM:~$
```

- **Start the program**

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

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



```
root@raspberrypi:~# ros2 launch largemodel largemodel_control.launch.py
[INFO] [launch]: All log files can be found below /root/.ros/log/2025-08-21-16-01-10-206398-raspberrypi-98016
[INFO] [launch]: Default logging verbosity is set to INFO

-------- robot_type = A1, rplidar_type = tmini, camera_type = nuwa --------

---------------------robot_type = A1---------------------
[INFO] [ydlidar_ros2_driver_node-10]: process started with pid [98082]
[INFO] [ascamera_node-1]: process started with pid [98050]
[INFO] [joint_state_publisher-2]: process started with pid [98052]
[INFO] [robot_state_publisher-3]: process started with pid [98054]
[INFO] [Ackman_driver_A1-4]: process started with pid [98056]
[INFO] [base_node_A1-5]: process started with pid [98058]
[INFO] [imu_filter_madgwick_node-6]: process started with pid [98061]
[INFO] [ekf_node-7]: process started with pid [98065]
[INFO] [yahboom_joy_A1-8]: process started with pid [98075]
[INFO] [joy_node-9]: process started with pid [98078]
[INFO] [static_transform_publisher-11]: process started with pid [98090]
[INFO] [static_transform_publisher-12]: process started with pid [98092]
[INFO] [static_transform_publisher-13]: process started with pid [98095]
[INFO] [model_service-14]: process started with pid [98100]
[INFO] [action_service_nuwa-15]: process started with pid [98102]
[INFO] [asr-16]: process started with pid [98105]
[robot_state_publisher-3] [INFO] [1755763271.586916300] [robot_state_publisher]: got segment base_footprint
[robot_state_publisher-3] [INFO] [1755763271.587125040] [robot_state_publisher]: got segment base_link
[robot_state_publisher-3] [INFO] [1755763271.587149058] [robot_state_publisher]: got segment camera_link
[robot_state_publisher-3] [INFO] [1755763271.587160632] [robot_state_publisher]: got segment imu_link
[robot_state_publisher-3] [INFO] [1755763271.587171336] [robot_state_publisher]: got segment laser
[robot_state_publisher-3] [INFO] [1755763271.587180410] [robot_state_publisher]: got segment left_front_wheel_joint
[robot_state_publisher-3] [INFO] [1755763271.587189632] [robot_state_publisher]: got segment left_rear_wheel_hinge
[robot_state_publisher-3] [INFO] [1755763271.587199428] [robot_state_publisher]: got segment left_steering_hinge_joint
[robot_state_publisher-3] [INFO] [1755763271.587208447] [robot_state_publisher]: got segment right_front_wheel_joint
[robot_state_publisher-3] [INFO] [1755763271.587217132] [robot_state_publisher]: got segment right_rear_wheel_hinge
[robot_state_publisher-3] [INFO] [1755763271.587226428] [robot_state_publisher]: got segment right_steering_hinge_joint
```

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

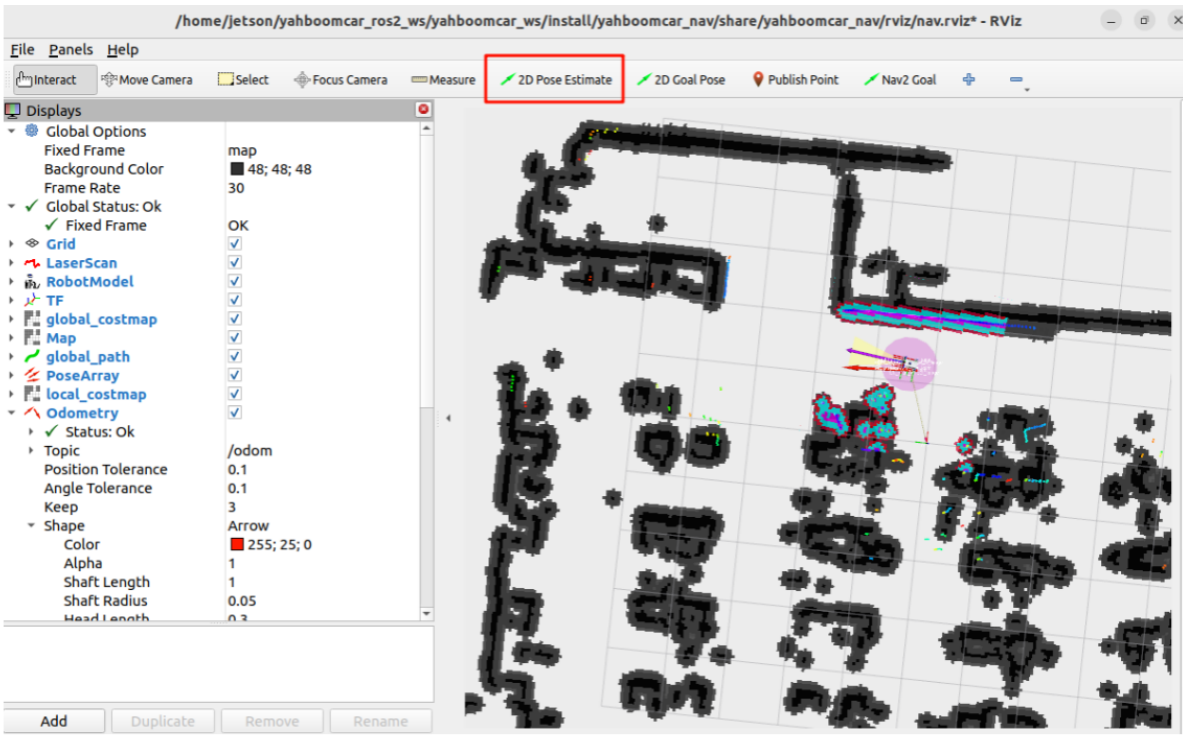
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

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.



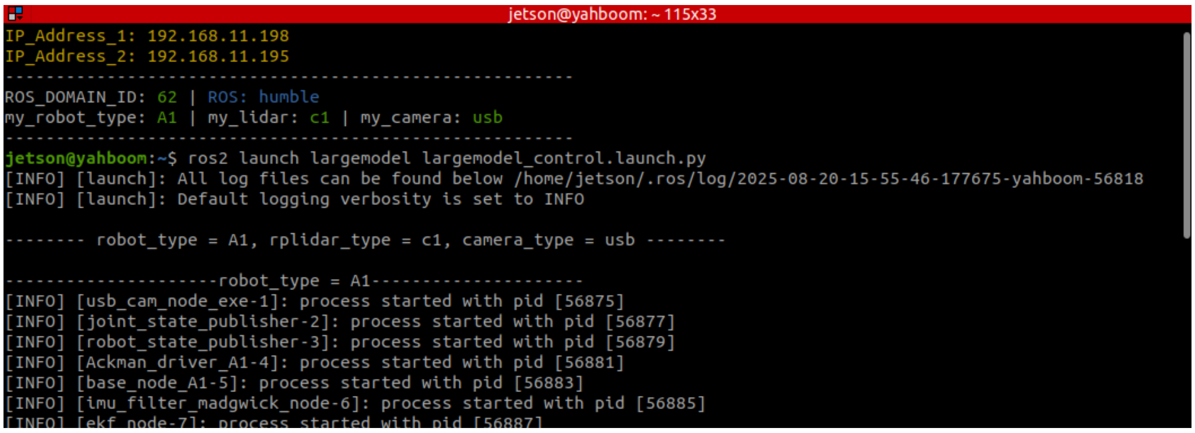
```
Global Options
    Fixed Frame              map
    Background Color         48; 48; 48
    Frame Rate               30
✓ Global Status: Ok
    ✓ Fixed Frame            OK
Grid                         ✓
LaserScan                    ✓
RobotModel                   ✓
TF                           ✓
global_costmap               ✓
Map                          ✓
global_path                  ✓
PoseArray                    ✓
local_costmap                ✓
Odometry                     ✓
    ✓ Status: Ok
    Topic                    /odom
    Position Tolerance       0.1
    Angle Tolerance          0.1
    Keep                     3
    Shape                    Arrow
        Color                255; 25; 0
        Alpha                1
        Shaft Length         1
        Shaft Radius         0.05
        Head Length          0.3
```

3.1.1 Raspberry Pi PI5 and ORIN board startup steps:

For the Raspberry Pi PI5, you need to first enter the Docker container; the ORIN board does not require this.

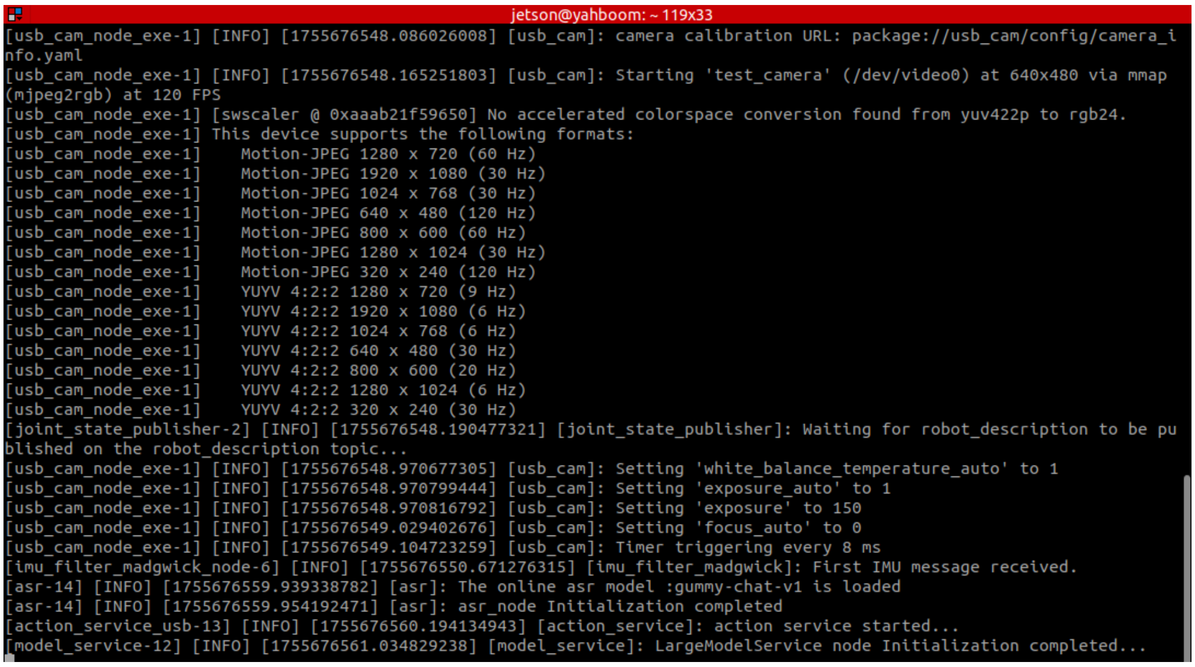
Open a terminal in Docker and enter the following command:

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



```
jetson@yahboom: ~ 115x33
IP_Address_1: 192.168.11.198
IP_Address_2: 192.168.11.195
-------------------------------------------------------
ROS_DOMAIN_ID: 62 | ROS: humble
my_robot_type: A1 | my_lidar: c1 | my_camera: usb
-------------------------------------------------------
jetson@yahboom:~$ ros2 launch largemodel largemodel_control.launch.py
[INFO] [launch]: All log files can be found below /home/jetson/.ros/log/2025-08-20-15-55-46-177675-yahboom-56818
[INFO] [launch]: Default logging verbosity is set to INFO

-------- robot_type = A1, rplidar_type = c1, camera_type = usb --------

--------------------robot_type = A1--------------------
[INFO] [usb_cam_node_exe-1]: process started with pid [56875]
[INFO] [joint_state_publisher-2]: process started with pid [56877]
[INFO] [robot_state_publisher-3]: process started with pid [56879]
[INFO] [Ackman_driver_A1-4]: process started with pid [56881]
[INFO] [base_node_A1-5]: process started with pid [56883]
[INFO] [imu_filter_madgwick_node-6]: process started with pid [56885]
[INFO] [ekf_node-7]: process started with pid [56887]
```

After initialization is complete, the following content will be displayed.



```
jetson@yahboom: ~ 119x33
[usb_cam_node_exe-1] [INFO] [1755676548.086026008] [usb_cam]: camera calibration URL: package://usb_cam/config/camera_i
nfo.yaml
[usb_cam_node_exe-1] [INFO] [1755676548.165251803] [usb_cam]: Starting 'test_camera' (/dev/video0) at 640x480 via mmap
(mjpeg2rgb) at 120 FPS
[usb_cam_node_exe-1] [swscaler @ 0xaaab21f59650] No accelerated colorspace conversion found from yuv422p to rgb24.
[usb_cam_node_exe-1] This device supports the following formats:
[usb_cam_node_exe-1]    Motion-JPEG 1280 x 720 (60 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 1920 x 1080 (30 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 1024 x 768 (30 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 640 x 480 (120 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 800 x 600 (60 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 1280 x 1024 (30 Hz)
[usb_cam_node_exe-1]    Motion-JPEG 320 x 240 (120 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 1280 x 720 (9 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 1920 x 1080 (6 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 1024 x 768 (6 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 640 x 480 (30 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 800 x 600 (20 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 1280 x 1024 (6 Hz)
[usb_cam_node_exe-1]    YUYV 4:2:2 320 x 240 (30 Hz)
[joint_state_publisher-2] [INFO] [1755676548.190477321] [joint_state_publisher]: Waiting for robot_description to be pu
blished on the robot_description topic...
[usb_cam_node_exe-1] [INFO] [1755676548.970677305] [usb_cam]: Setting 'white_balance_temperature_auto' to 1
[usb_cam_node_exe-1] [INFO] [1755676548.970799444] [usb_cam]: Setting 'exposure_auto' to 1
[usb_cam_node_exe-1] [INFO] [1755676548.970816792] [usb_cam]: Setting 'exposure' to 150
[usb_cam_node_exe-1] [INFO] [1755676549.029402676] [usb_cam]: Setting 'focus_auto' to 0
[usb_cam_node_exe-1] [INFO] [1755676549.104723259] [usb_cam]: Timer triggering every 8 ms
[imu_filter_madgwick_node-6] [INFO] [1755676550.671276315] [imu_filter_madgwick]: First IMU message received.
[asr-14] [INFO] [1755676559.939338782] [asr]: The online asr model :gummy-chat-v1 is loaded
[asr-14] [INFO] [1755676559.954192471] [asr]: asr_node Initialization completed
[action_service_usb-13] [INFO] [1755676560.194134943] [action_service]: action service started...
[model_service-12] [INFO] [1755676561.034829238] [model_service]: LargeModelService node Initialization completed...
```

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

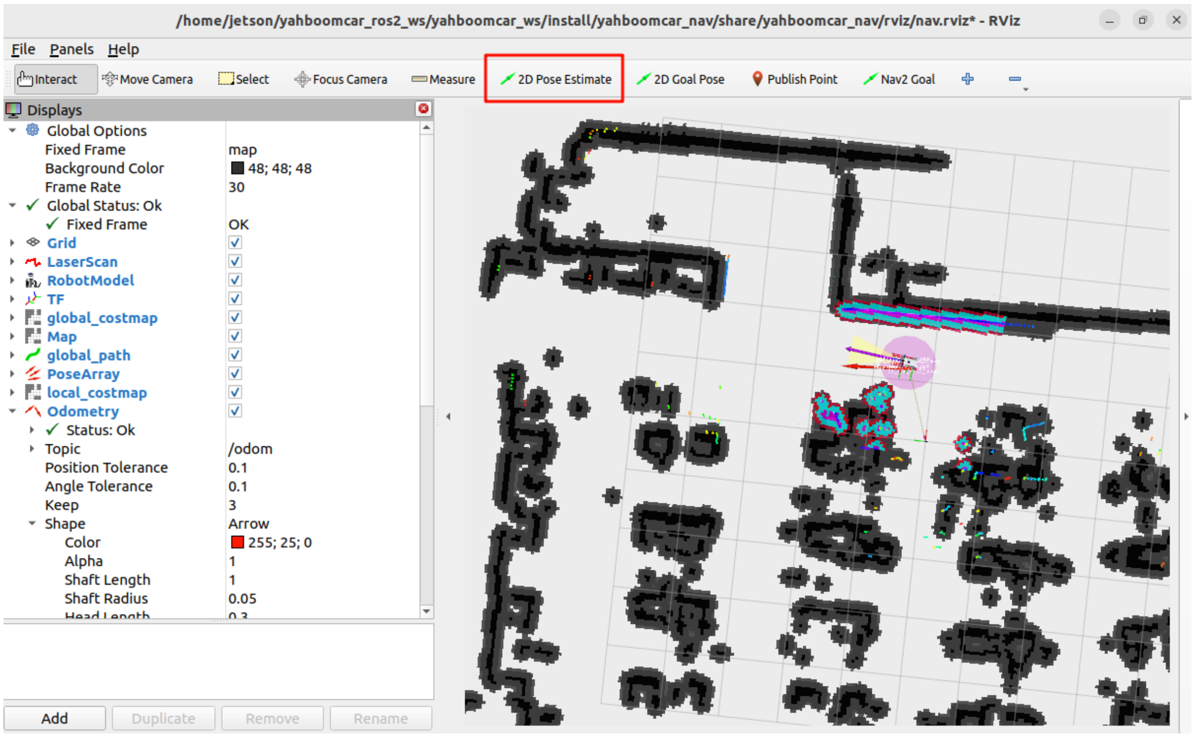
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

Wait for the navigation algorithm to start before displaying the map. Then open the Docker terminal and type

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

#Fast relocalization navigation (not supported on Raspberry Pi 5 or Jetson Nano
controllers)
ros2 launch yahboomcar_nav localization_imu_odom.launch.py ••use_rviz:=false
load_state_filename:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomca
r_nav/maps/yahboomcar.pbstream

ros2 launch yahboomcar_nav navigation_cartodwb_launch.py
maps:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/yahb
oomcar.yaml
```

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

After that, follow the navigation 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 initial positioning, preparations are complete.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
✓ Status: Ok	
Topic	/odom
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	3
Shape	Arrow
Color	🟥 255; 25; 0
Alpha	1
Shaft Length	1
Shaft Radius	0.05
Head Length	0.3

3.2 Test Case

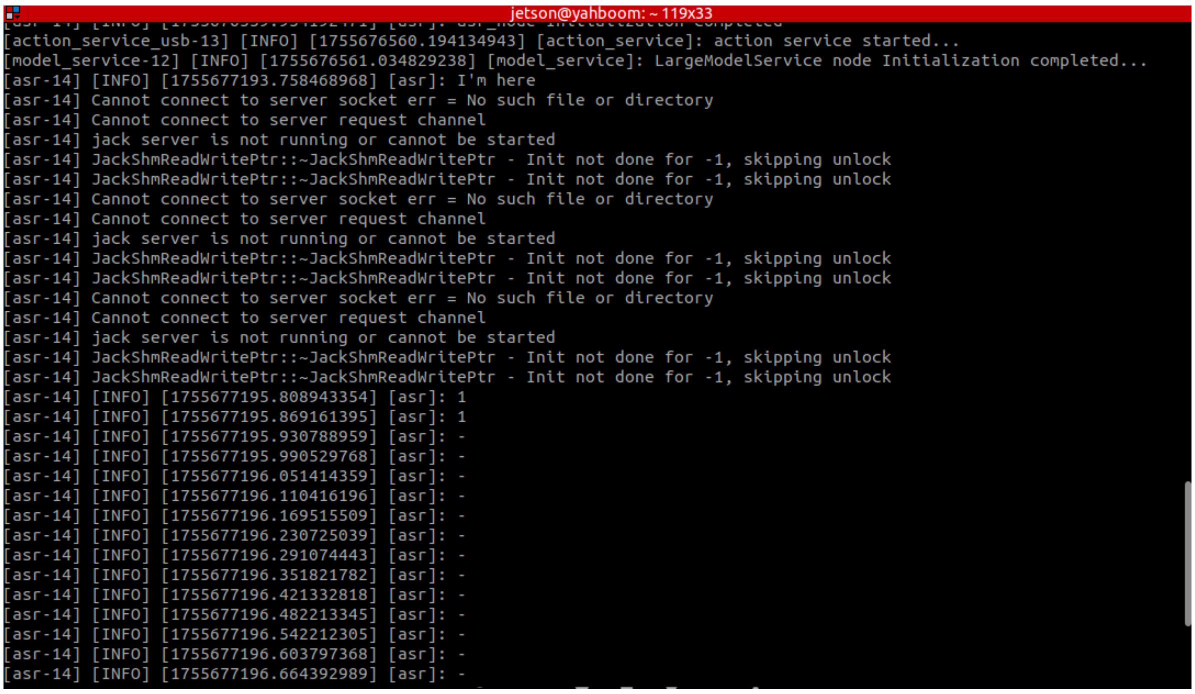
Here are some reference test cases; users can create their own dialogue commands.

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

3.2.1 Example

First, use "Hi, yahboom" to wake the robot. The robot responds: "I'm here, please tell me." After the robot responds, the buzzer beeps briefly (beep-). The user can speak, and the robot will perform dynamic sound detection. If there is sound activity, it will print 1, and if there is no sound activity, it will print -. When the speech ends, it will perform tail tone detection. If there is silence for more than 450ms, the recording will stop.

The following figure shows the dynamic voice detection (VAD):

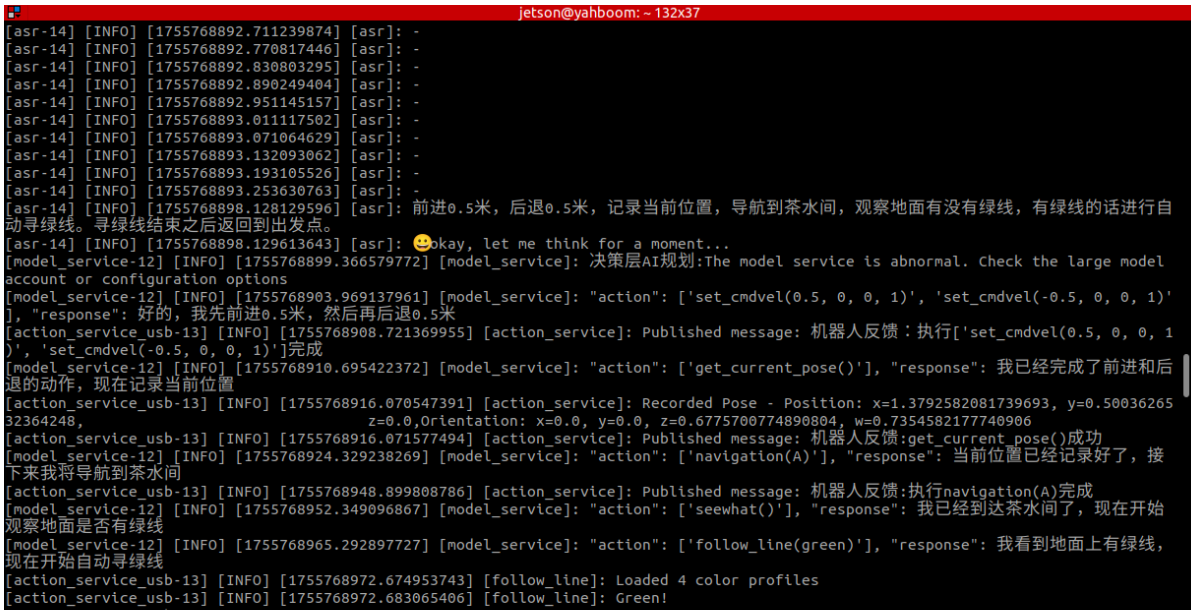


```
[action_service_usb-13] [INFO] [1755676560.194134943] [action_service]: action service started...
[model_service-12] [INFO] [1755676561.034829238] [model_service]: LargeModelService node Initialization completed...
[asr-14] [INFO] [1755677193.758468968] [asr]: I'm here
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] [INFO] [1755677195.808943354] [asr]: 1
[asr-14] [INFO] [1755677195.869161395] [asr]: 1
[asr-14] [INFO] [1755677195.930788959] [asr]: -
[asr-14] [INFO] [1755677195.990529768] [asr]: -
[asr-14] [INFO] [1755677196.051414359] [asr]: -
[asr-14] [INFO] [1755677196.110416196] [asr]: -
[asr-14] [INFO] [1755677196.169515509] [asr]: -
[asr-14] [INFO] [1755677196.230725039] [asr]: -
[asr-14] [INFO] [1755677196.291074443] [asr]: -
[asr-14] [INFO] [1755677196.351821782] [asr]: -
[asr-14] [INFO] [1755677196.421332818] [asr]: -
[asr-14] [INFO] [1755677196.482213345] [asr]: -
[asr-14] [INFO] [1755677196.542212305] [asr]: -
[asr-14] [INFO] [1755677196.603797368] [asr]: -
[asr-14] [INFO] [1755677196.664392989] [asr]: -
```

The robot will first communicate with the user, then follow the instructions. The terminal will print the following message:

Network abnormality: Decision-making layer AI planning: The model service is abnormal. Check the large model account or configuration options. This does not affect the AI model's decision-making function, but it is recommended to try again under smooth network conditions!!

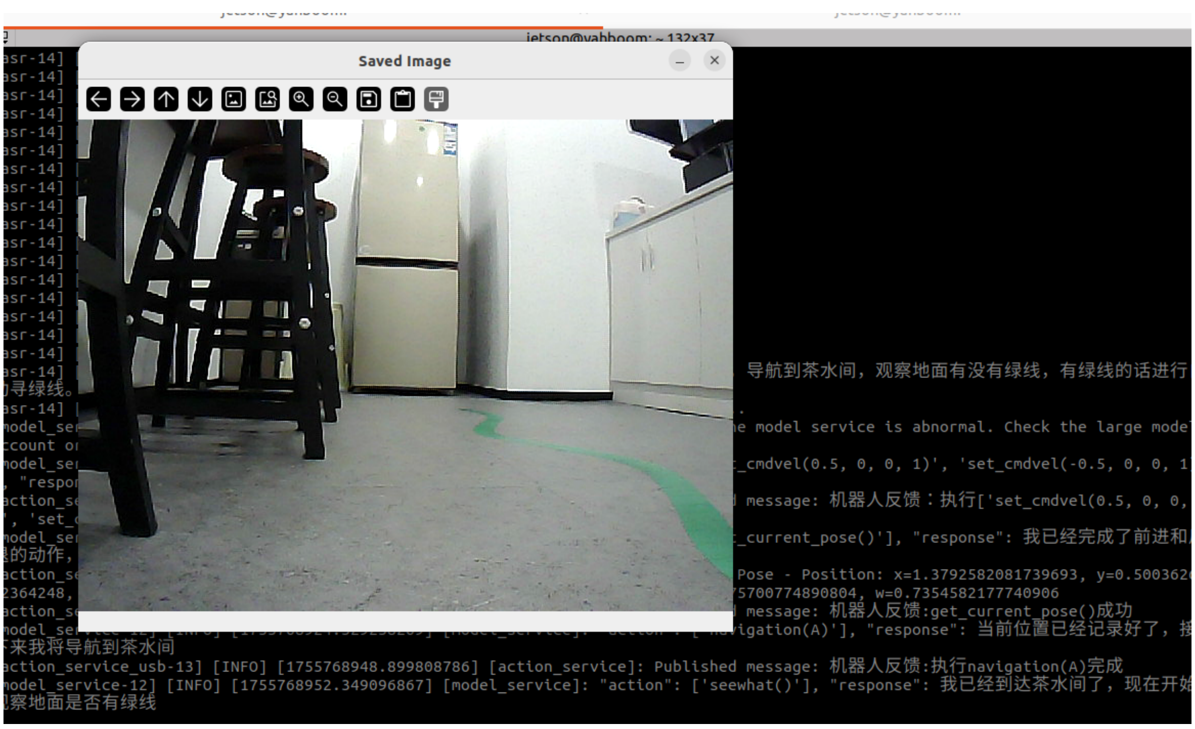
Normal: The decision-making layer AI planning will divide the steps 1, 2, 3, 4, etc.



```
[asr-14] [INFO] [1755768892.711239874] [asr]: -
[asr-14] [INFO] [1755768892.770817446] [asr]: -
[asr-14] [INFO] [1755768892.830803295] [asr]: -
[asr-14] [INFO] [1755768892.890249404] [asr]: -
[asr-14] [INFO] [1755768892.951145157] [asr]: -
[asr-14] [INFO] [1755768893.011117502] [asr]: -
[asr-14] [INFO] [1755768893.071064629] [asr]: -
[asr-14] [INFO] [1755768893.132093062] [asr]: -
[asr-14] [INFO] [1755768893.193105526] [asr]: -
[asr-14] [INFO] [1755768893.253630763] [asr]: -
[asr-14] [INFO] [1755768898.128129596] [asr]: 前进0.5米，后退0.5米，记录当前位置，导航到茶水间，观察地面有没有绿线，有绿线的话进行自
动寻绿线。寻绿线结束之后返回到出发点。
[asr-14] [INFO] [1755768898.129613643] [asr]: 😀okay, let me think for a moment...
[model_service-12] [INFO] [1755768899.366579772] [model_service]: 决策层AI规划:The model service is abnormal. Check the large model
account or configuration options
[model_service-12] [INFO] [1755768903.969137961] [model_service]: "action": ['set_cmdvel(0.5, 0, 0, 1)', 'set_cmdvel(-0.5, 0, 0, 1)'
], "response": 好的，我先前进0.5米，然后再后退0.5米
[action_service_usb-13] [INFO] [1755768908.721369955] [action_service]: Published message: 机器人反馈：执行['set_cmdvel(0.5, 0, 0, 1
)', 'set_cmdvel(-0.5, 0, 0, 1)']完成
[model_service-12] [INFO] [1755768910.695422372] [model_service]: "action": ['get_current_pose()'], "response": 我已经完成了前进和后
退的动作，现在记录当前位置
[action_service_usb-13] [INFO] [1755768916.070547391] [action_service]: Recorded Pose - Position: x=1.3792582081739693, y=0.50036265
32364248,                             z=0.0,Orientation: x=0.0, y=0.0, z=0.6775700774890804, w=0.7354582177740906
[action_service_usb-13] [INFO] [1755768916.071577494] [action_service]: Published message: 机器人反馈:get_current_pose()成功
[model_service-12] [INFO] [1755768924.329238269] [model_service]: "action": ['navigation(A)'], "response": 当前位置已经记录好了，接
下来我将导航到茶水间
[action_service_usb-13] [INFO] [1755768948.899808786] [action_service]: Published message: 机器人反馈:执行navigation(A)完成
[model_service-12] [INFO] [1755768952.349096867] [model_service]: "action": ['seewhat()'], "response": 我已经到达茶水间了，现在开始
观察地面是否有绿线
[model_service-12] [INFO] [1755768965.292897727] [model_service]: "action": ['follow_line(green)'], "response": 我看到地面上有绿线，
现在开始自动寻绿线
[action_service_usb-13] [INFO] [1755768972.674953743] [follow_line]: Loaded 4 color profiles
[action_service_usb-13] [INFO] [1755768972.683065406] [follow_line]: Green!
```

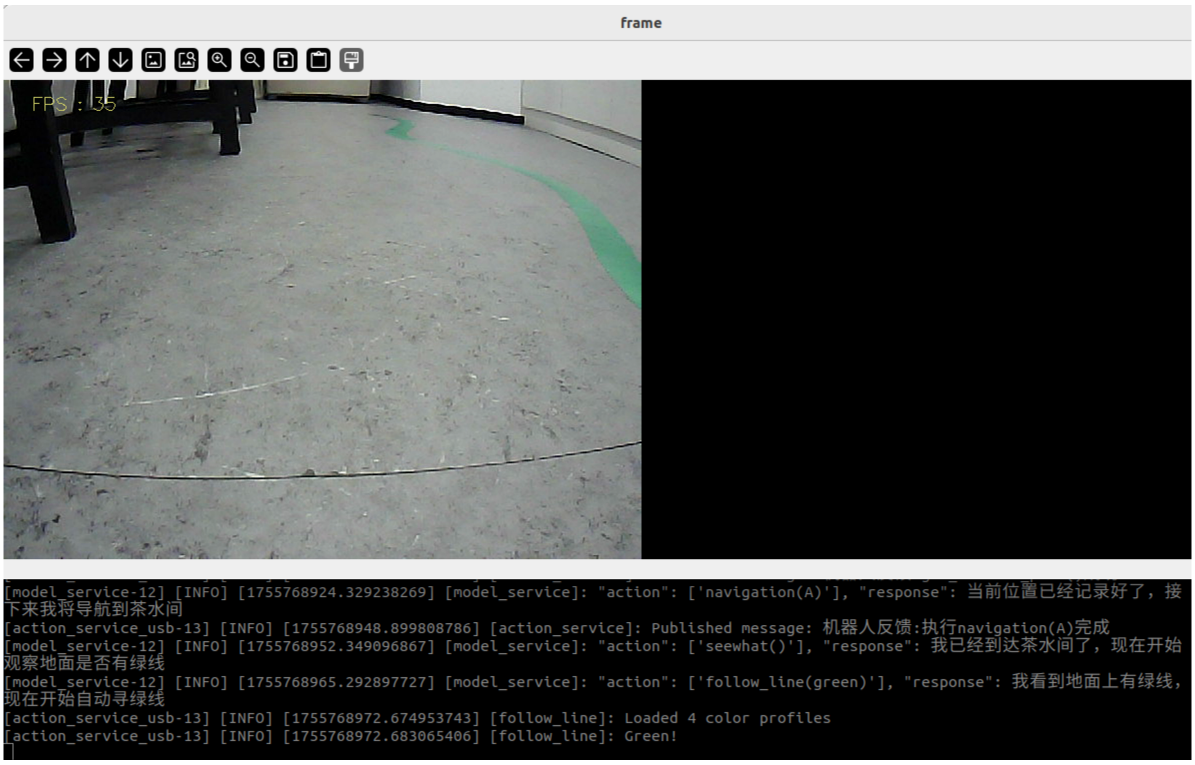
[model_service]: "action": ['seewhat()'], "response": I've arrived at the tea room. Now, I'm checking to see if there's a green line on the ground.

After reaching the navigation destination, this step calls the **seewhat** method. A window will open on the VNC screen and display for 5 seconds. An image is captured and uploaded to the large model for inference.



```
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]
asr-14]                                                                 导航到茶水间，观察地面有没有绿线，有绿线的话进行
寻绿线。
asr-14]                                                                .
model_se                                                               e model service is abnormal. Check the large mode
ccount o
model_se                                                               _cmdvel(0.5, 0, 0, 1)', 'set_cmdvel(-0.5, 0, 0, 1
, "respo
action_s                                                               message: 机器人反馈：执行['set_cmdvel(0.5, 0, 0,
', 'set_
model_se                                                               _current_pose()'], "response": 我已经完成了前进和
的动作,
action_s                                                               Pose - Position: x=1.3792582081739693, y=0.500362
2364248,                                                               5700774890804, w=0.7354582177740906
action_s                                                               message: 机器人反馈:get_current_pose()成功
model_se                                                               vigation(A)'], "response": 当前位置已经记录好了，接
来我将导航到茶水间
action_service_usb-13] [INFO] [1755768948.899808786] [action_service]: Published message: 机器人反馈:执行navigation(A)完成
model_service-12] [INFO] [1755768952.349096867] [model_service]: "action": ['seewhat()'], "response": 我已经到达茶水间了，现在开始
察地面是否有绿线
```

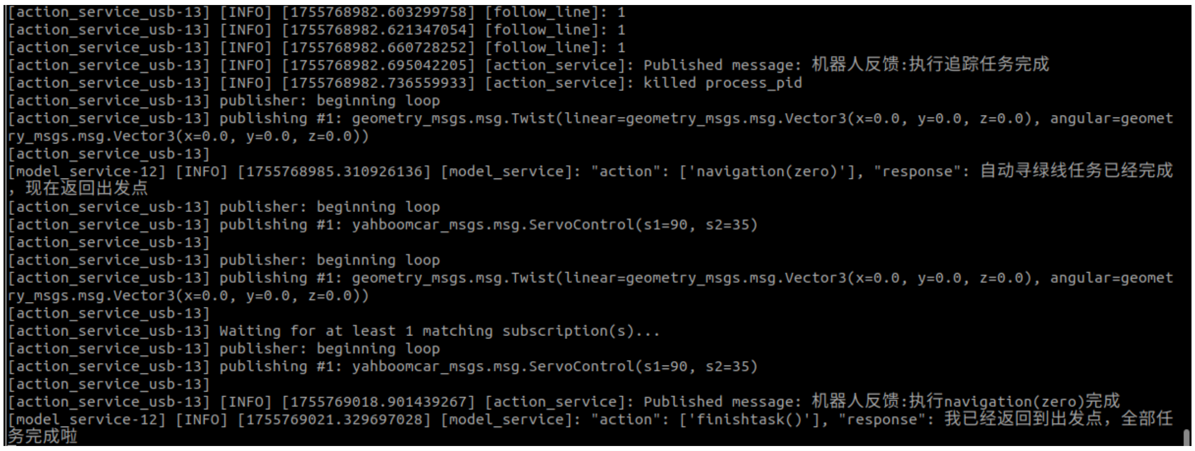
The large model receives the result, indicating that a green line has been observed, so the automatic green line detection program begins.



```
[model_service-12] [INFO] [1755768924.329238269] [model_service]: "action": ['navigation(A)'], "response": 当前位置已经记录好了，接
下来我将导航到茶水间
[action_service_usb-13] [INFO] [1755768948.899808786] [action_service]: Published message: 机器人反馈:执行navigation(A)完成
[model_service-12] [INFO] [1755768952.349096867] [model_service]: "action": ['seewhat()'], "response": 我已经到达茶水间了，现在开始
观察地面是否有绿线
[model_service-12] [INFO] [1755768965.292897727] [model_service]: "action": ['follow_line(green)'], "response": 我看到地面上有绿线,
现在开始自动寻绿线
[action_service_usb-13] [INFO] [1755768972.674953743] [follow_line]: Loaded 4 color profiles
[action_service_usb-13] [INFO] [1755768972.683065406] [follow_line]: Green!
```

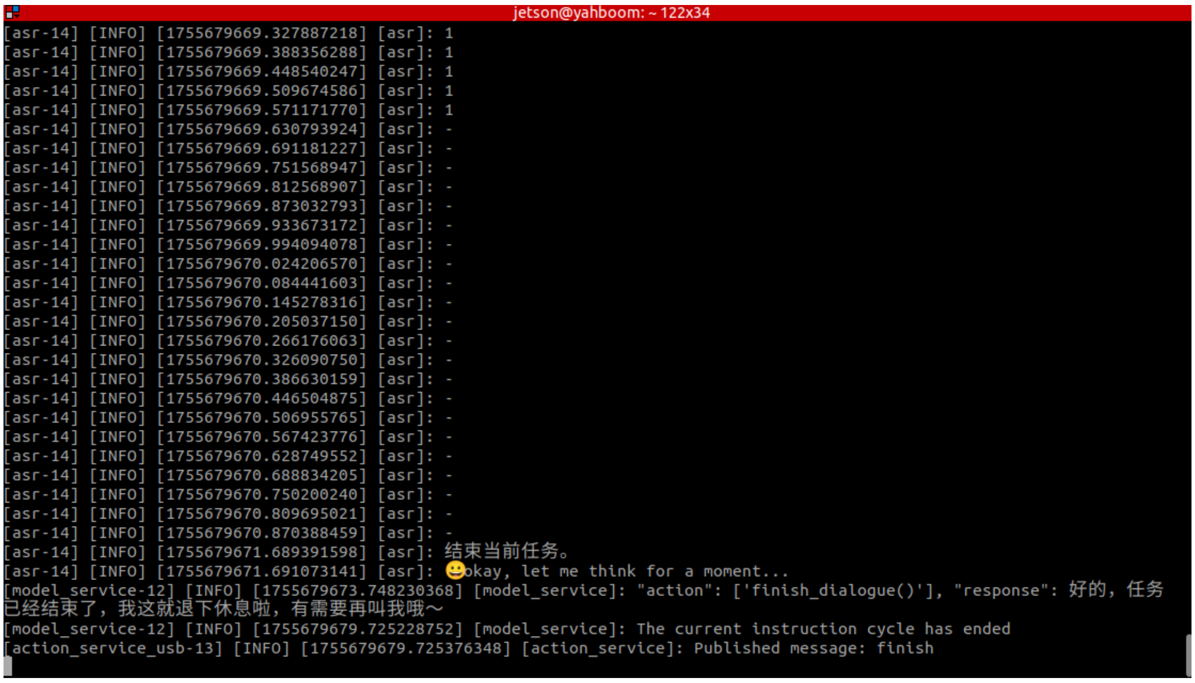
The logic is the same as in the previous example. If there is no target to track on the screen, the program will count down for 10 seconds, the terminal will print a 5-second countdown, and the process will automatically end, indicating that the task is complete.

Then the next command, `Return to Starting Point`, is executed.



```
[action_service_usb-13] [INFO] [1755768982.603299758] [follow_line]: 1
[action_service_usb-13] [INFO] [1755768982.621347054] [follow_line]: 1
[action_service_usb-13] [INFO] [1755768982.660728252] [follow_line]: 1
[action_service_usb-13] [INFO] [1755768982.695042205] [action_service]: Published message: 机器人反馈:执行追踪任务完成
[action_service_usb-13] [INFO] [1755768982.736559933] [action_service]: killed process_pid
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geomet
ry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[action_service_usb-13]
[model_service-12] [INFO] [1755768985.310926136] [model_service]: "action": ['navigation(zero)'], "response": 自动寻绿线任务已经完成
，现在返回出发点
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: yahboomcar_msgs.msg.ServoControl(s1=90, s2=35)
[action_service_usb-13]
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geomet
ry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[action_service_usb-13]
[action_service_usb-13] Waiting for at least 1 matching subscription(s)...
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: yahboomcar_msgs.msg.ServoControl(s1=90, s2=35)
[action_service_usb-13]
[action_service_usb-13] [INFO] [1755769018.901439267] [action_service]: Published message: 机器人反馈:执行navigation(zero)完成
[model_service-12] [INFO] [1755769021.329697028] [model_service]: "action": ['finishtask()'], "response": 我已经返回到出发点，全部任
务完成啦
```

After completing all tasks, the robot will enter the free conversation state again, but all conversation history will be retained. At this point, you can wakeyahboom up again and click "End Current Task" to end the current task cycle, clear the conversation history, and start a new one.



```
jetson@yahboom: ~ 122x34
[asr-14] [INFO] [1755679669.327887218] [asr]: 1
[asr-14] [INFO] [1755679669.388356288] [asr]: 1
[asr-14] [INFO] [1755679669.448540247] [asr]: 1
[asr-14] [INFO] [1755679669.509674586] [asr]: 1
[asr-14] [INFO] [1755679669.571171770] [asr]: 1
[asr-14] [INFO] [1755679669.630793924] [asr]: -
[asr-14] [INFO] [1755679669.691181227] [asr]: -
[asr-14] [INFO] [1755679669.751568947] [asr]: -
[asr-14] [INFO] [1755679669.812568907] [asr]: -
[asr-14] [INFO] [1755679669.873032793] [asr]: -
[asr-14] [INFO] [1755679669.933673172] [asr]: -
[asr-14] [INFO] [1755679669.994094078] [asr]: -
[asr-14] [INFO] [1755679670.024206570] [asr]: -
[asr-14] [INFO] [1755679670.084441603] [asr]: -
[asr-14] [INFO] [1755679670.145278316] [asr]: -
[asr-14] [INFO] [1755679670.205037150] [asr]: -
[asr-14] [INFO] [1755679670.266176063] [asr]: -
[asr-14] [INFO] [1755679670.326090750] [asr]: -
[asr-14] [INFO] [1755679670.386630159] [asr]: -
[asr-14] [INFO] [1755679670.446504875] [asr]: -
[asr-14] [INFO] [1755679670.506955765] [asr]: -
[asr-14] [INFO] [1755679670.567423776] [asr]: -
[asr-14] [INFO] [1755679670.628749552] [asr]: -
[asr-14] [INFO] [1755679670.688834205] [asr]: -
[asr-14] [INFO] [1755679670.750200240] [asr]: -
[asr-14] [INFO] [1755679670.809695021] [asr]: -
[asr-14] [INFO] [1755679670.870388459] [asr]: -
[asr-14] [INFO] [1755679671.689391598] [asr]: 结束当前任务。
[asr-14] [INFO] [1755679671.691073141] [asr]: 😀okay, let me think for a moment...
[model_service-12] [INFO] [1755679673.748230368] [model_service]: "action": ['finish_dialogue()'], "response": 好的，任务
已经结束了，我这就退下休息啦，有需要再叫我哦～
[model_service-12] [INFO] [1755679679.725228752] [model_service]: The current instruction cycle has ended
[action_service_usb-13] [INFO] [1755679679.725376348] [action_service]: Published message: finish
```

4. Source Code Analysis

Source code location:

Jetson Orin Nano, Jetson Orin NX host:

```
#NUWA camera users
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_s
ervice_nuwa.py
#USB camera users
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_s
ervice_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 **[7.Multimodal visual understand+SLAM navigation]** . For a detailed explanation of the **follow_line(self, color)** function, refer to the **[5.Multimodal visual understand + visual line patroling]**.