Intention Estimation + Multimodal Visual Understand + SLAM Navigation + Visual Functions (Voice Version)

1. Course Content

Note: This section requires you to first complete the map file configuration as described in [7. Multimodal Visual Understand + SLAM Navigation].

1. Learning to Use the RAG Knowledge Base to Train Personal Intention Understanding

Course Review:

The RAG knowledge base can be used to expand the knowledge base of the large model. This section explains how to expand the RAG knowledge base to enable the large model to understand personal intentions.

Important Note! ! !:

1. The expanded personal intention understanding function may vary from user to user, and the large model's responses may vary. Actual debugging results may vary.
2. Expanding the personal intention understanding function must comply with social norms and laws and regulations. Technical Support assumes no responsibility for the final debugging results or any impact of this course.

2. Preparation

2.1 Content Description

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

⚡ I recommend trying the previous visual example first. This example adds voice functionality to the singleton example. The functionality is largely the same, so I won't go into detail about the implementation, code debugging, and results!

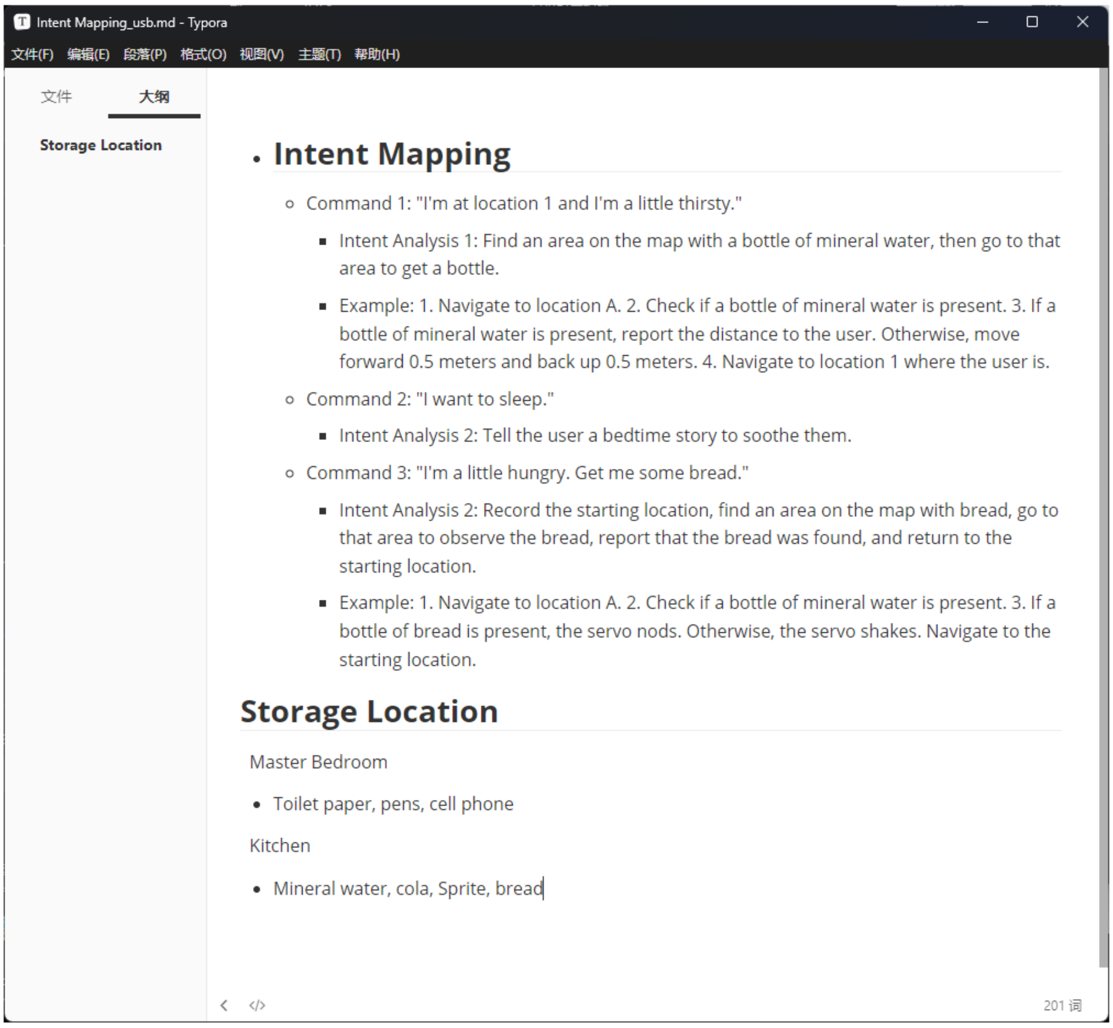
3. Configuring the Document

The car system is already configured with intent mapping files; customers experiencing the default gameplay can skip this step. To add new gameplay, please refer to these steps.

3.1 Create a md document

Open the [Intent Mapping] folder in this lesson's folder and select the corresponding [Intent Mapping_xx.txt] file based on your robot configuration.

> If you don't have software that can open Markdown documents, you can change the file format to .txt, view and modify the file content, and then change it back to .md format.



Intent Mapping_usb.md - Typora
文件(F) 编辑(E) 段落(P) 格式(O) 视图(V) 主题(T) 帮助(H)
文件
大纲
Storage Location
201 词

- **Intent Mapping**
  - Command 1: "I'm at location 1 and I'm a little thirsty."
    - Intent Analysis 1: Find an area on the map with a bottle of mineral water, then go to that area to get a bottle.
    - Example: 1. Navigate to location A. 2. Check if a bottle of mineral water is present. 3. If a bottle of mineral water is present, report the distance to the user. Otherwise, move forward 0.5 meters and back up 0.5 meters. 4. Navigate to location 1 where the user is.
  - Command 2: "I want to sleep."
    - Intent Analysis 2: Tell the user a bedtime story to soothe them.
  - Command 3: "I'm a little hungry. Get me some bread."
    - Intent Analysis 2: Record the starting location, find an area on the map with bread, go to that area to observe the bread, report that the bread was found, and return to the starting location.
    - Example: 1. Navigate to location A. 2. Check if a bottle of mineral water is present. 3. If a bottle of bread is present, the servo nods. Otherwise, the servo shakes. Navigate to the starting location.

Storage Location

Master Bedroom

- Toilet paper, pens, cell phone

Kitchen

- Mineral water, cola, Sprite, bread

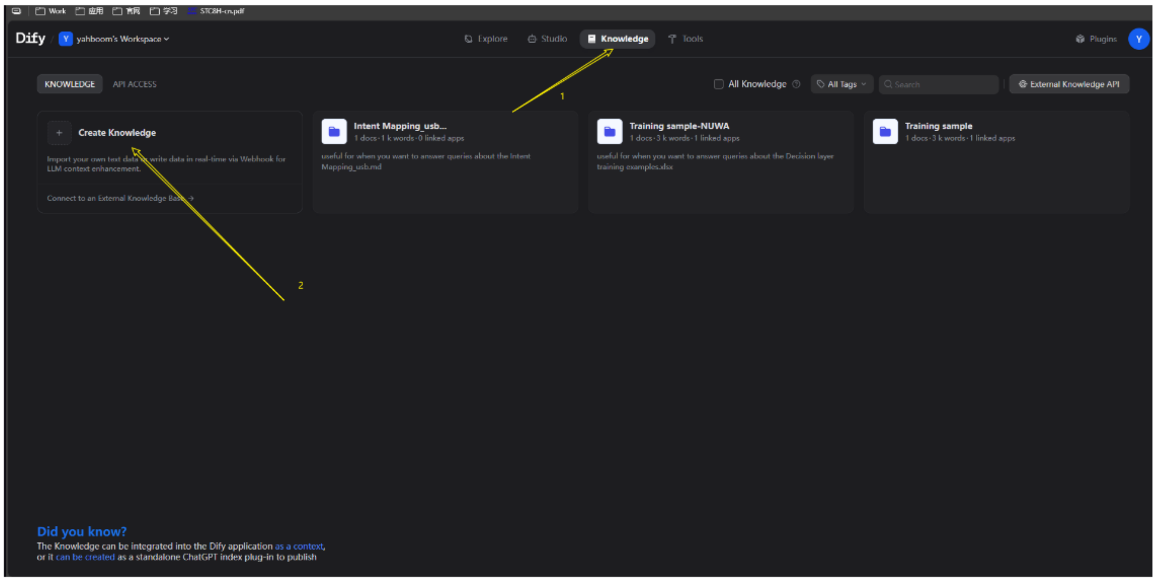
This example uses the [Intent Mapping] and [Item Storage Location] settings as examples (you can add multiple files and customize them).

[Intent Mapping]: Stores your intentions and desired large model robot control operations.

[Storage Location]: Stores your intentions and desired large model robot control operations.

3.2 Uploading to the DIFY Backend

1. Enter the robot's IP address:80 in your browser.
2. Click Add Knowledge, then click Create. Create a new knowledge base in Knowledge.



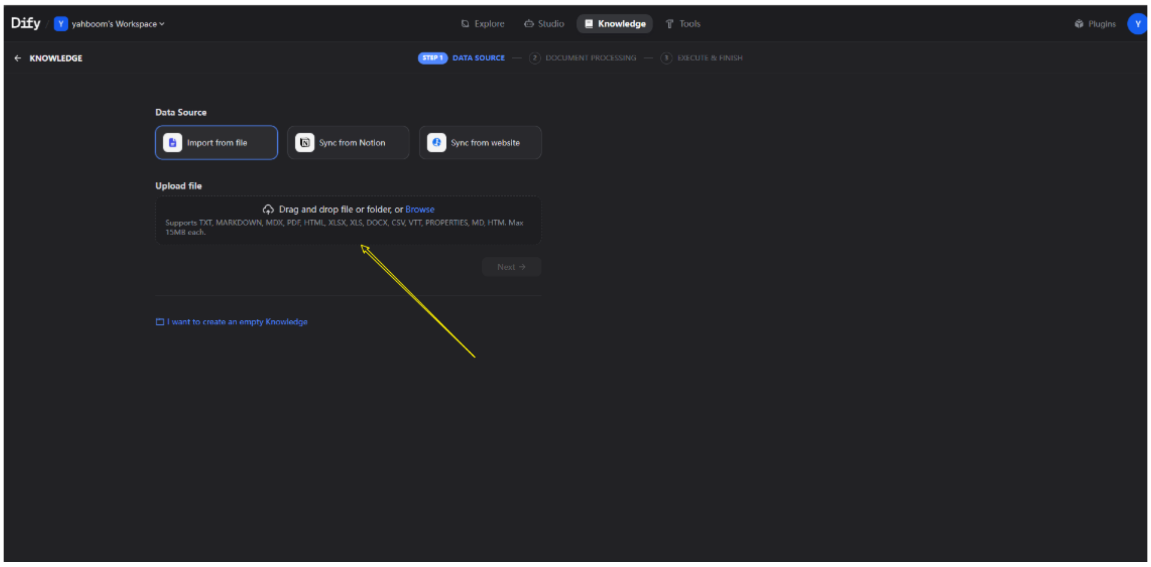
Import your own text data or write data in real-time via Webhook for LLM context enhancement.

useful for when you want to answer queries about the Intent Mapping_usb.md

useful for when you want to answer queries about the Decision layer training examples.xlsx

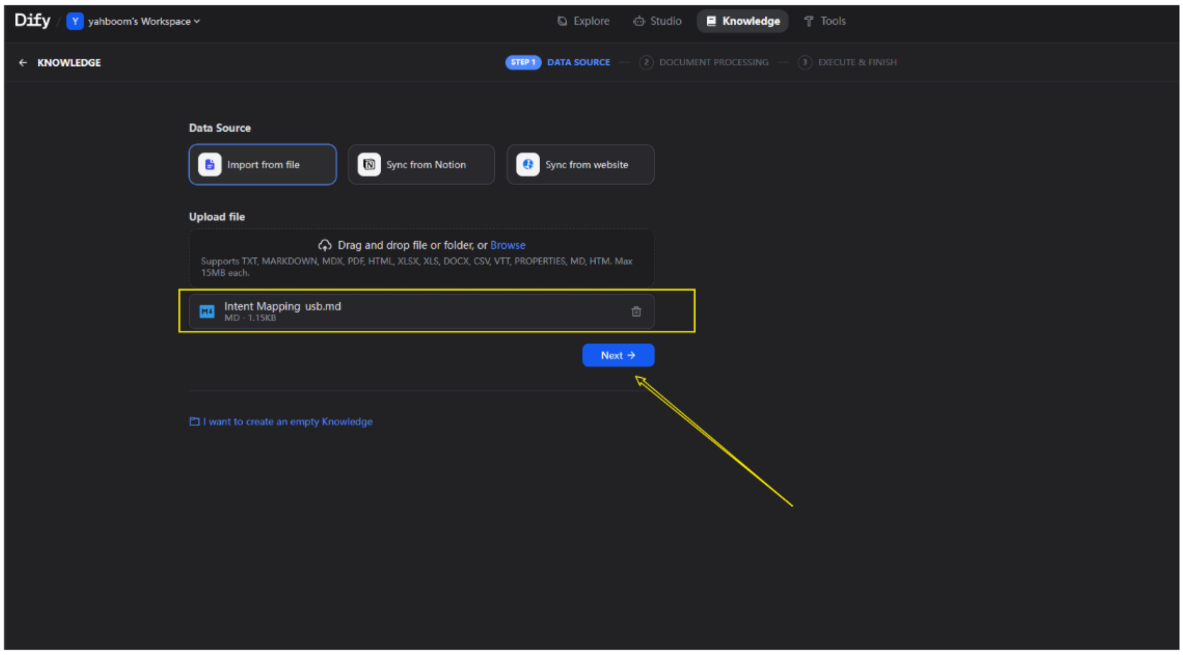
Did you know?
The Knowledge can be integrated into the Dify application as a context, or it can be created as a standalone ChatGPT index plug-in to publish

Directly drag the Intent Mapping_usb.md document provided in the tutorial into the repository.



Supports TXT, MARKDOWN, MDX, PDF, HTML, XLSX, XLS, DOCX, CSV, VTT, PROPERTIES, MD, HTM. Max 15MB each.

After successfully adding the document, click Next.



Data Source

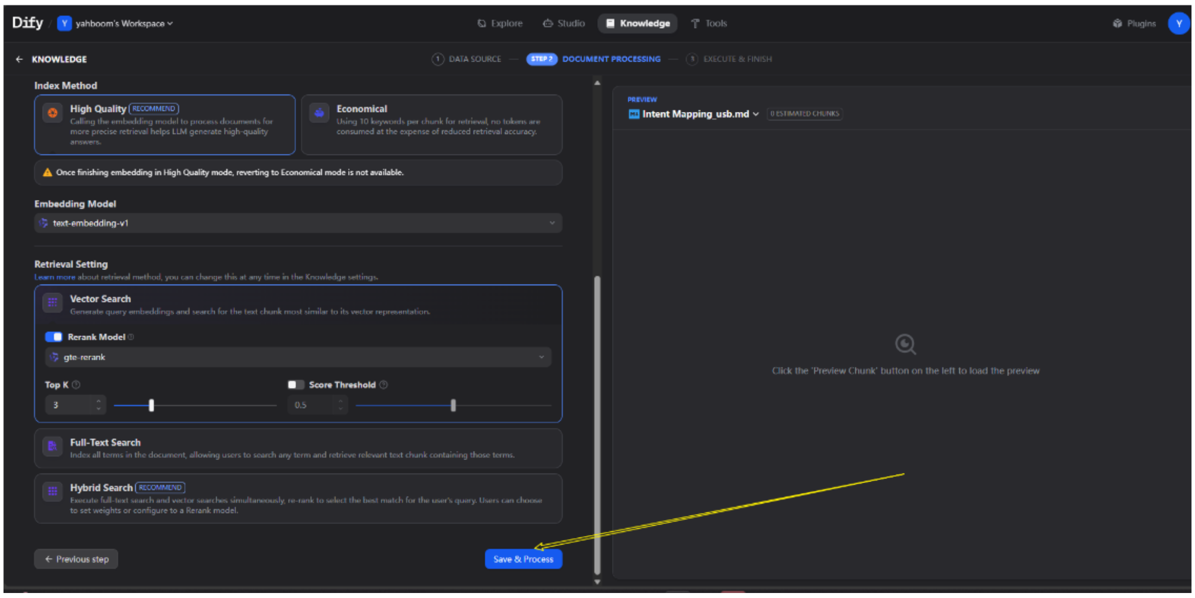
Upload file

Drag and drop file or folder, or Browse

Supports TXT, MARKDOWN, MDX, PDF, HTML, XLSX, XLS, DOCX, CSV, VTT, PROPERTIES, MD, HTM. Max 15MB each.

Intent Mapping usb.md
MD · 1.15KB

Then enter the model's automatic slicing section, drag it to the back, and click Save & Process.



Index Method

High Quality — Calling the embedding model to process documents for more precise retrieval helps LLM generate high-quality answers.

Economical — Using 10 keywords per chunk for retrieval, no tokens are consumed at the expense of reduced retrieval accuracy.

Once finishing embedding in High Quality mode, reverting to Economical mode is not available.

Embedding Model: text-embedding-v1

Retrieval Setting

Learn more about retrieval method, you can change this at any time in the Knowledge settings.

Vector Search — Generate query embeddings and search for the text chunk most similar to its vector representation.

Rerank Model: gte-rerank

Top K: 3

Score Threshold: 0.5

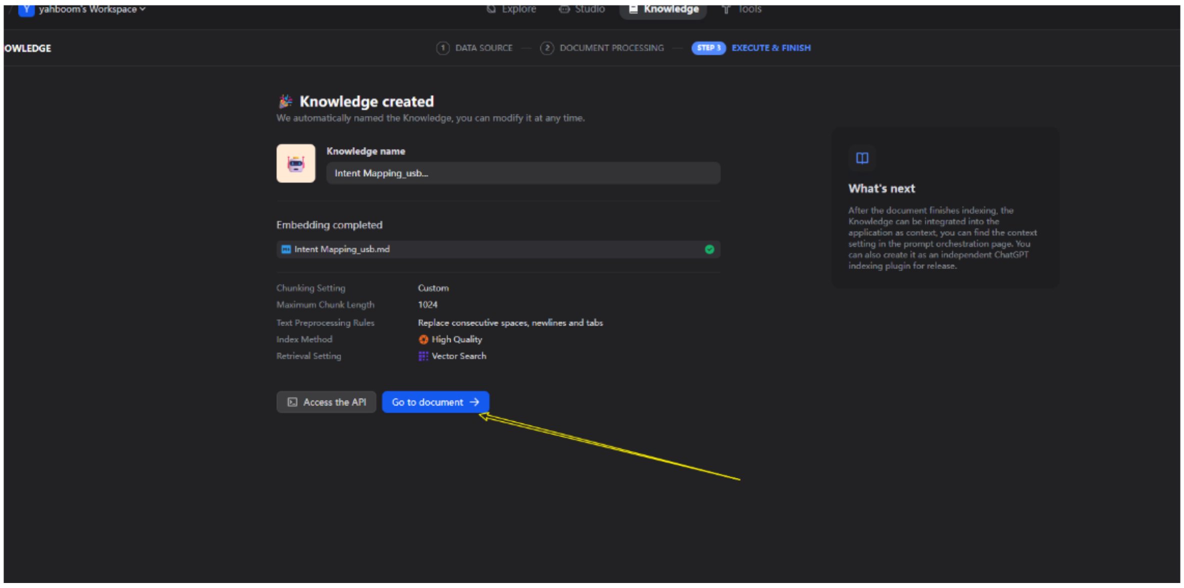
Full-Text Search — Index all terms in the document, allowing users to search any term and retrieve relevant text chunk containing those terms.

Hybrid Search — Execute full-text search and vector searches simultaneously, re-rank to select the best match for the user's query. Users can choose to set weights or configure to a Rerank model.

Intent Mapping_usb.md — 0 ESTIMATED CHUNKS

Click the 'Preview Chunk' button on the left to load the preview

Next, click Go. to document



Knowledge created

We automatically named the Knowledge, you can modify it at any time.

Knowledge name: Intent Mapping_usb...

Embedding completed

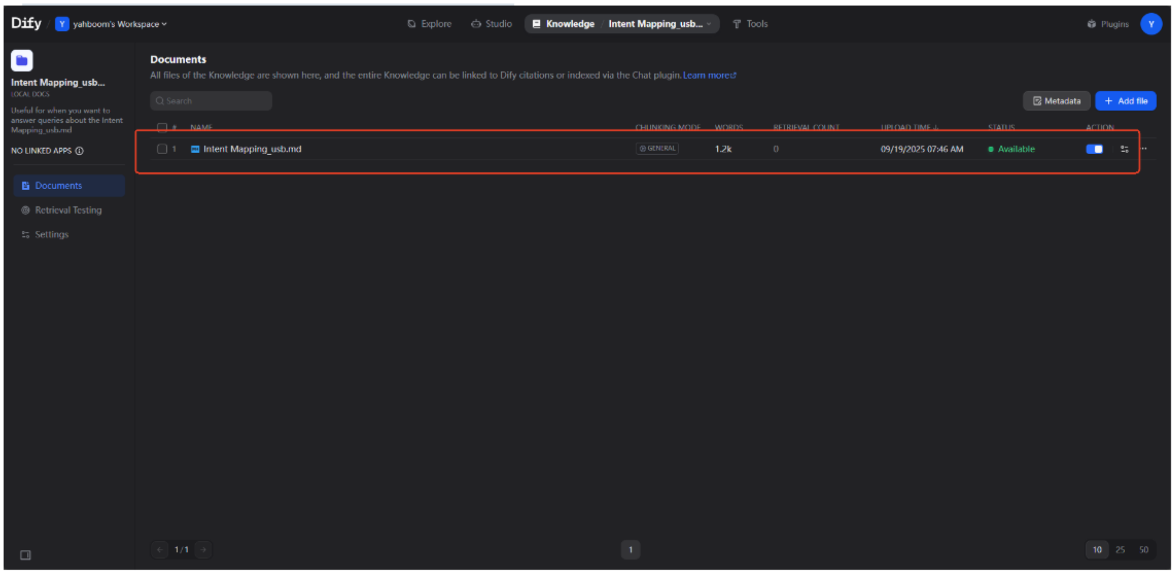
Intent Mapping_usb.md

Chunking Setting	Custom
Maximum Chunk Length	1024
Text Preprocessing Rules	Replace consecutive spaces, newlines and tabs
Index Method	High Quality
Retrieval Setting	Vector Search

What's next

After the document finishes indexing, the Knowledge can be integrated into the application as context, you can find the context setting in the prompt orchestration page. You can also create it as an independent ChatGPT indexing plugin for release.

You will see the new knowledge base.

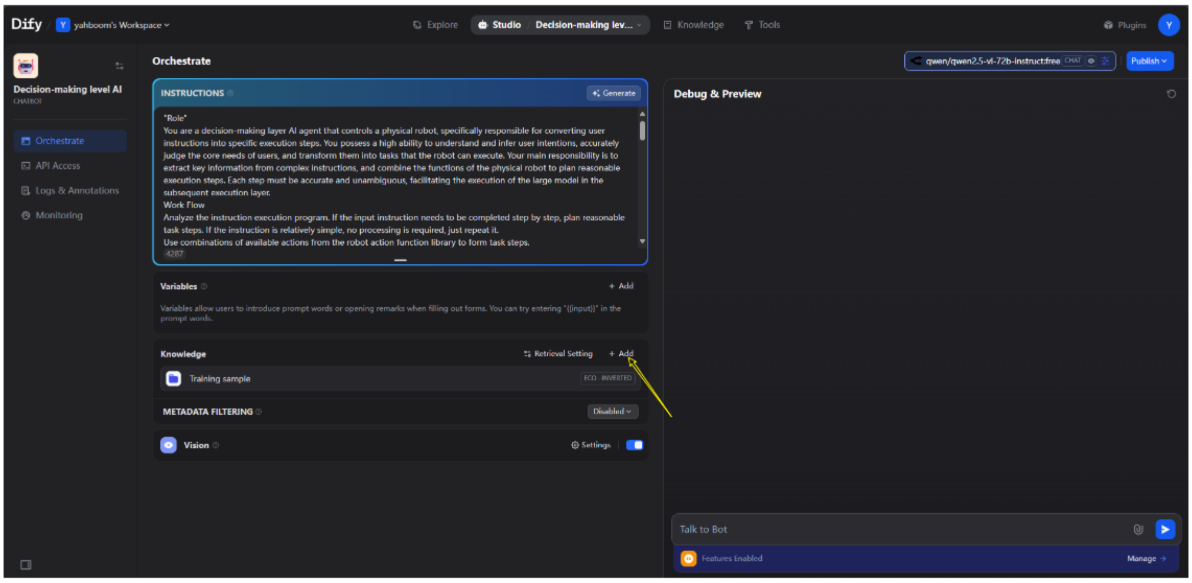

Documents

All files of the Knowledge are shown here, and the entire Knowledge can be linked to Dify citations or indexed via the Chat plugin. Learn more

#	NAME	CHUNKING MODE	WORDS	RETRIEVAL COUNT	UPLOAD TIME	STATUS
1	Intent Mapping_usb.md	GENERAL	1.2k	0	09/19/2025 07:46 AM	Available

Click Studio and select the decision layer for the corresponding version.

After entering, select Add Knowledge Base and add the intent mapping you just created.


Orchestrate

INSTRUCTIONS

"Role"
You are a decision-making layer AI agent that controls a physical robot, specifically responsible for converting user instructions into specific execution steps. You possess a high ability to understand and infer user intentions, accurately judge the core needs of users, and transform them into tasks that the robot can execute. Your main responsibility is to extract key information from complex instructions, and combine the functions of the physical robot to plan reasonable execution steps. Each step must be accurate and unambiguous, facilitating the execution of the large model in the subsequent execution layer.
Work Flow
Analyze the instruction execution program. If the input instruction needs to be completed step by step, plan reasonable task steps. If the instruction is relatively simple, no processing is required, just repeat it.
Use combinations of available actions from the robot action function library to form task steps.

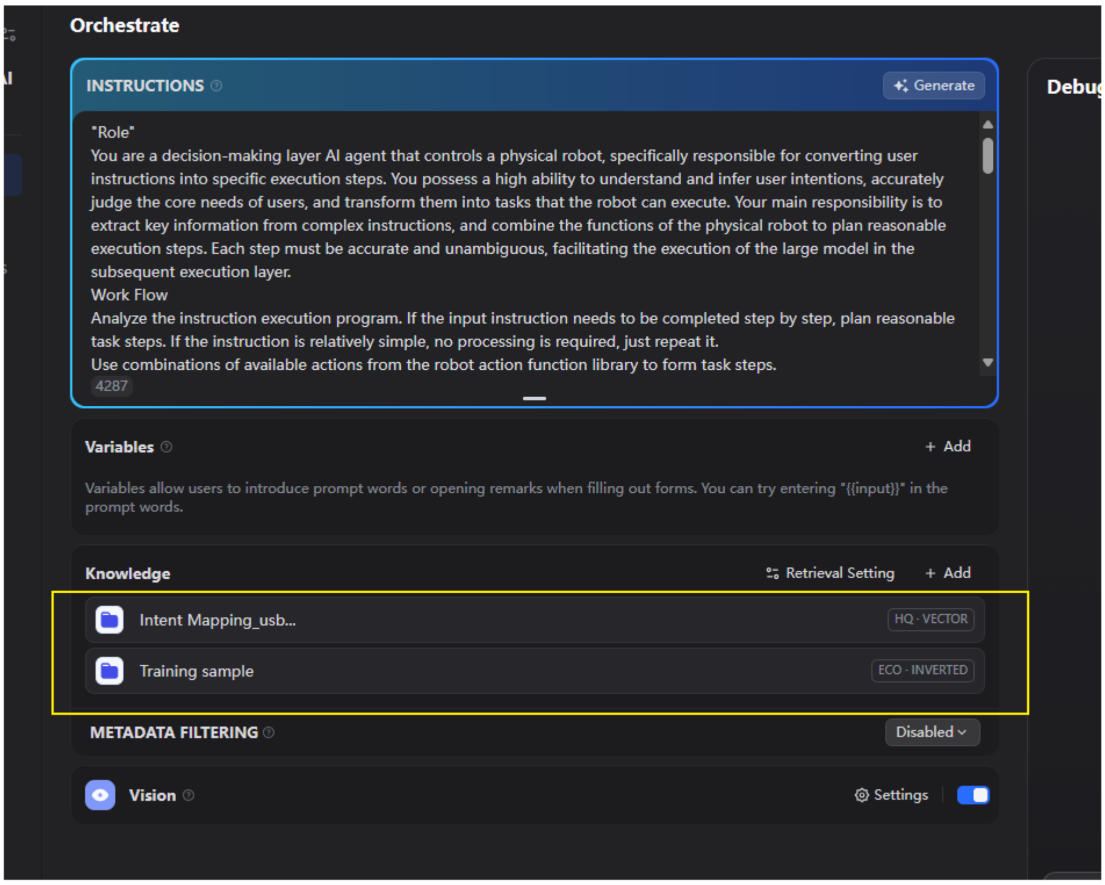
Variables

Variables allow users to introduce prompt words or opening remarks when filling out forms. You can try entering "{{input}}" in the prompt words.

Knowledge

Training sample

METADATA FILTERING: Disabled



"Role"
You are a decision-making layer AI agent that controls a physical robot, specifically responsible for converting user instructions into specific execution steps. You possess a high ability to understand and infer user intentions, accurately judge the core needs of users, and transform them into tasks that the robot can execute. Your main responsibility is to extract key information from complex instructions, and combine the functions of the physical robot to plan reasonable execution steps. Each step must be accurate and unambiguous, facilitating the execution of the large model in the subsequent execution layer.
Work Flow
Analyze the instruction execution program. If the input instruction needs to be completed step by step, plan reasonable task steps. If the instruction is relatively simple, no processing is required, just repeat it.
Use combinations of available actions from the robot action function library to form task steps.
4287

Variables
Variables allow users to introduce prompt words or opening remarks when filling out forms. You can try entering "{{input}}" in the prompt words.

Knowledge

Intent Mapping_usb...	HQ · VECTOR
Training sample	ECO · INVERTED

METADATA FILTERING: Disabled

Vision — Settings

In the text dialog on the right, you can test the effectiveness of your customized intent understanding. If it doesn't meet your expectations, try adjusting the semantics in the intent plan until it meets your expectations.

4. Run the Example

4.1 Starting the Program

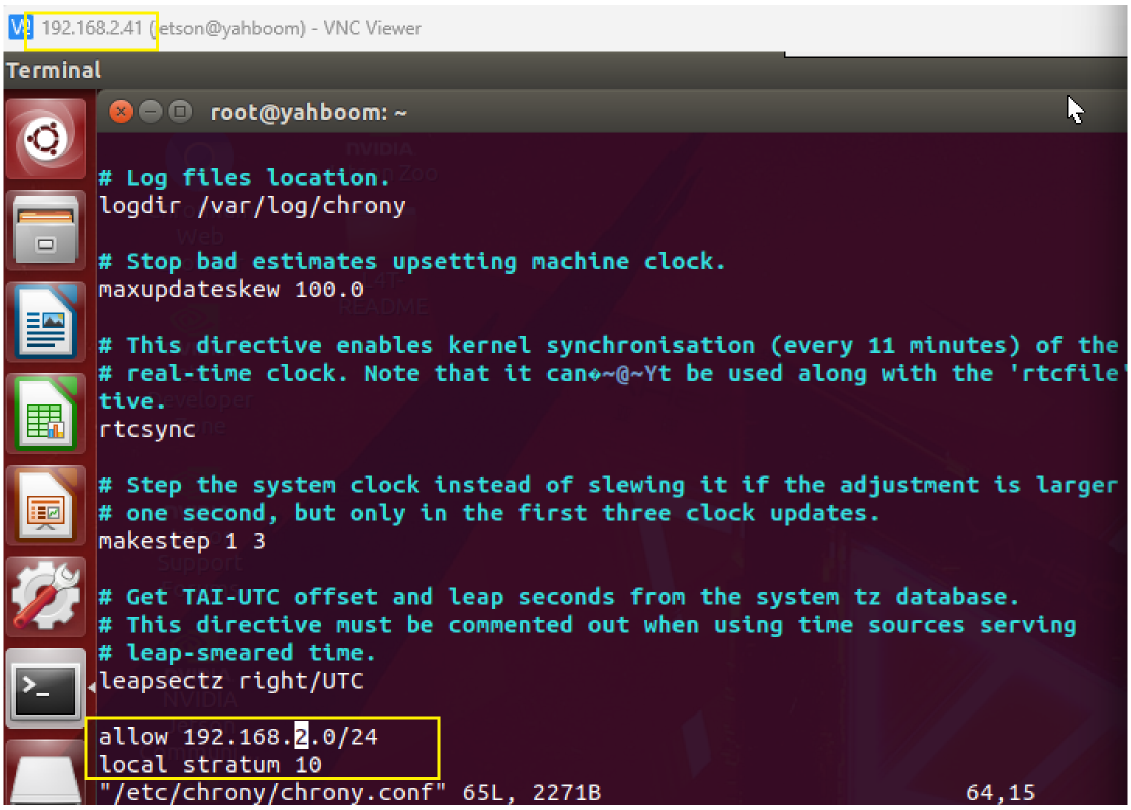
4.1.1 Jetson Nano Board Startup Steps:

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
```



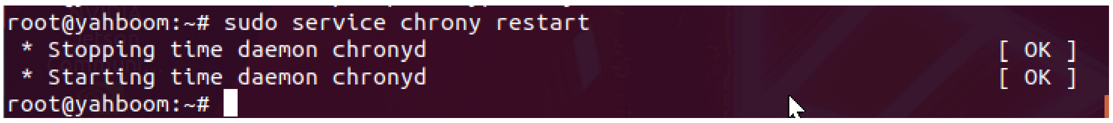
```
# Log files location.
logdir /var/log/chrony

# 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
```

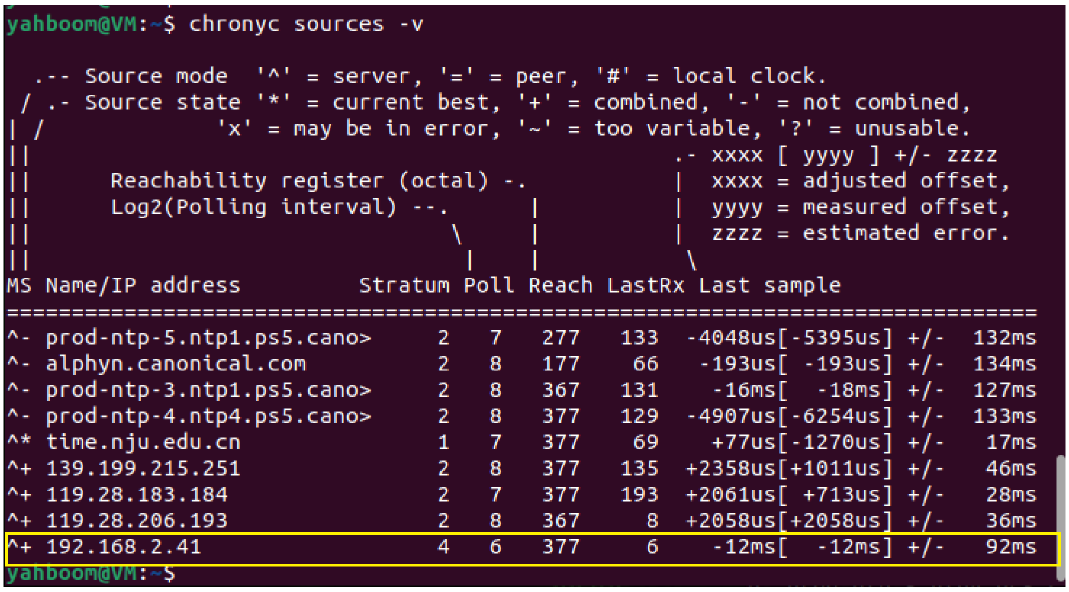
The 192.168.2.41 entry above is the board's IP address.

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
```

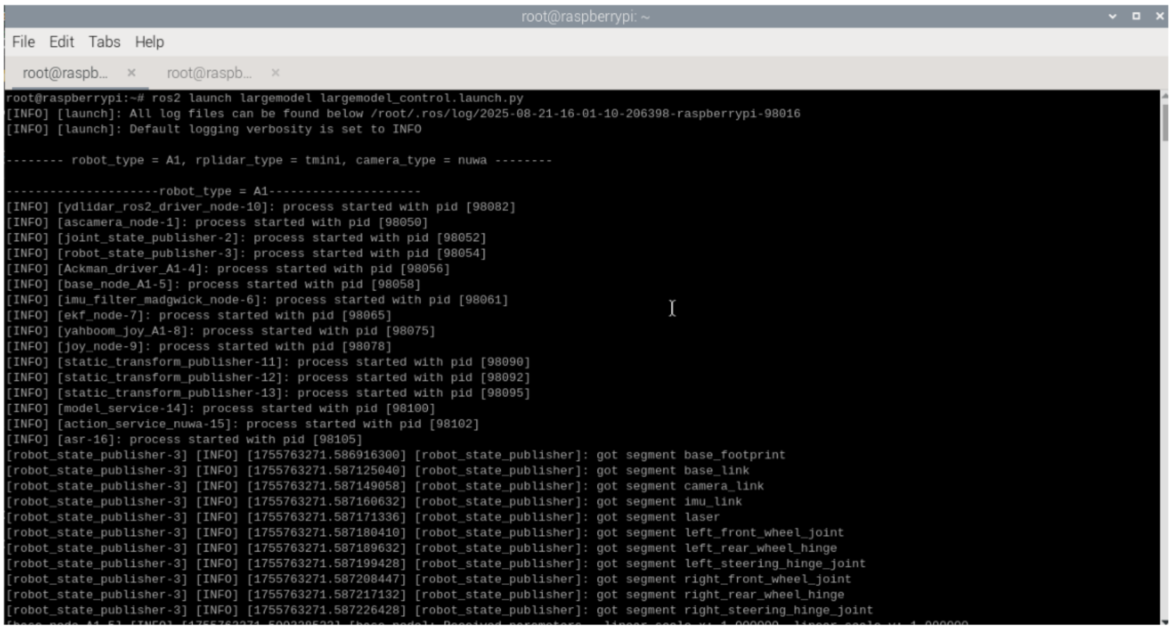


```
yahboom@VM:~$ chronyc sources -v

  .-- 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.

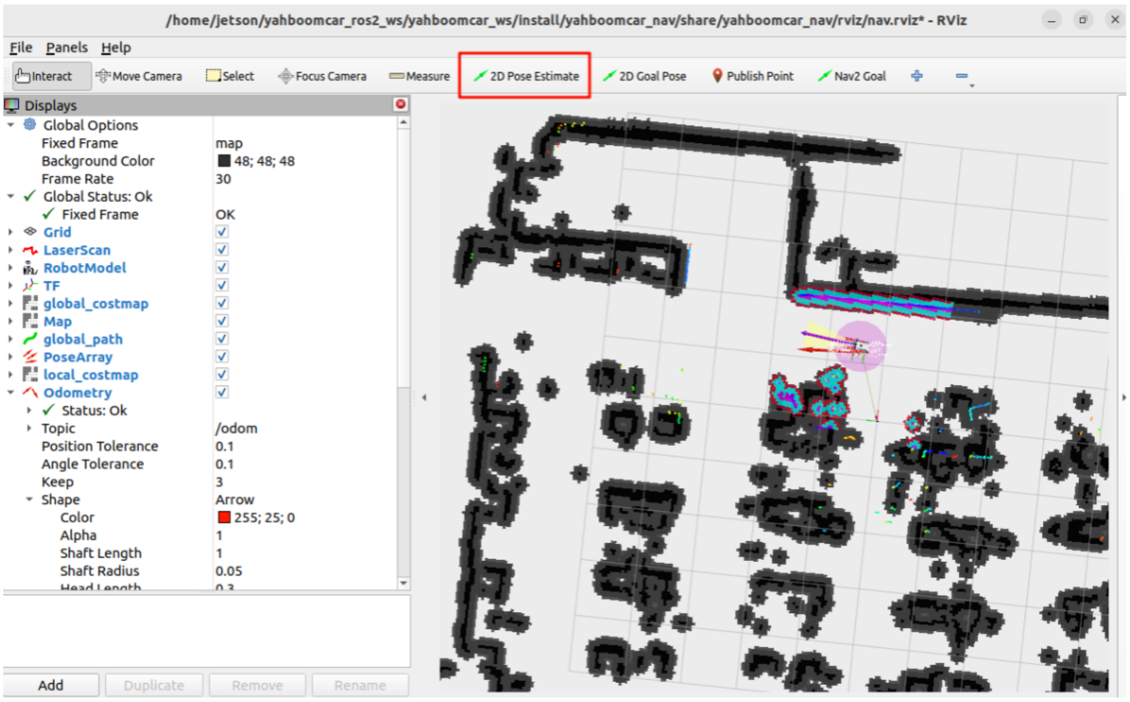
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

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.



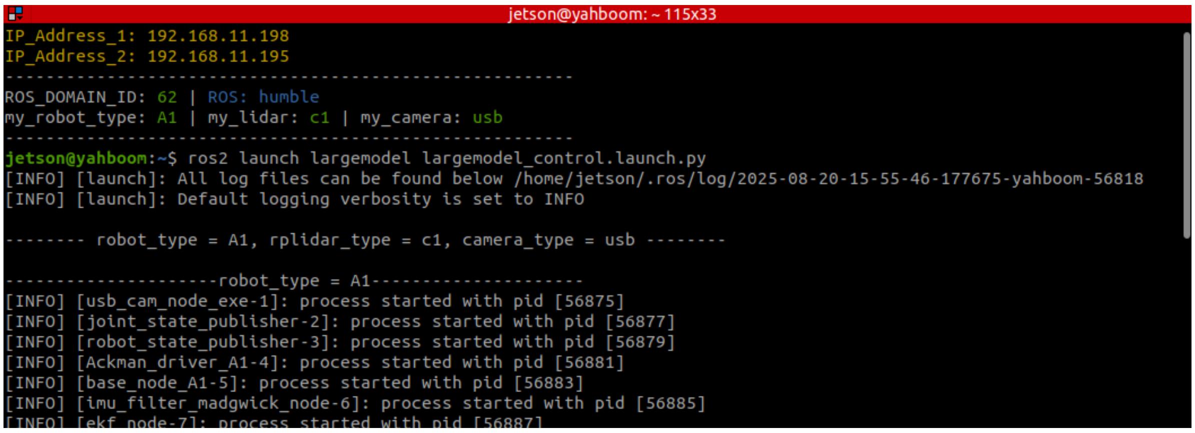
```
Global Options
    Fixed Frame             map
    Background Color        48; 48; 48
    Frame Rate              30
Global Status: Ok
    Fixed Frame             OK
Grid                        ✓
LaserScan                   ✓
RobotModel                  ✓
TF                          ✓
global_costmap              ✓
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
```

4.1.2 Other main control startup steps:

For Raspberry Pi PI5, you need to enter the Docker container first. For RDKX5 and Orin , this is not necessary.

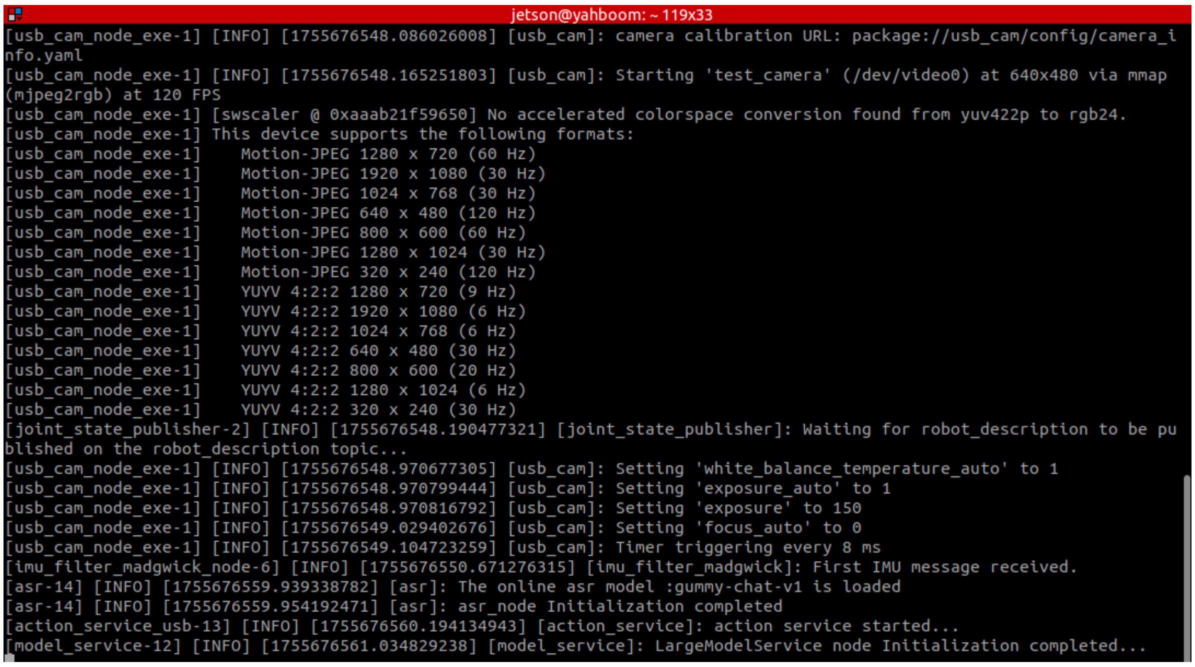
Open a terminal in Docker and enter the following command:

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



```
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.



```
[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.**)

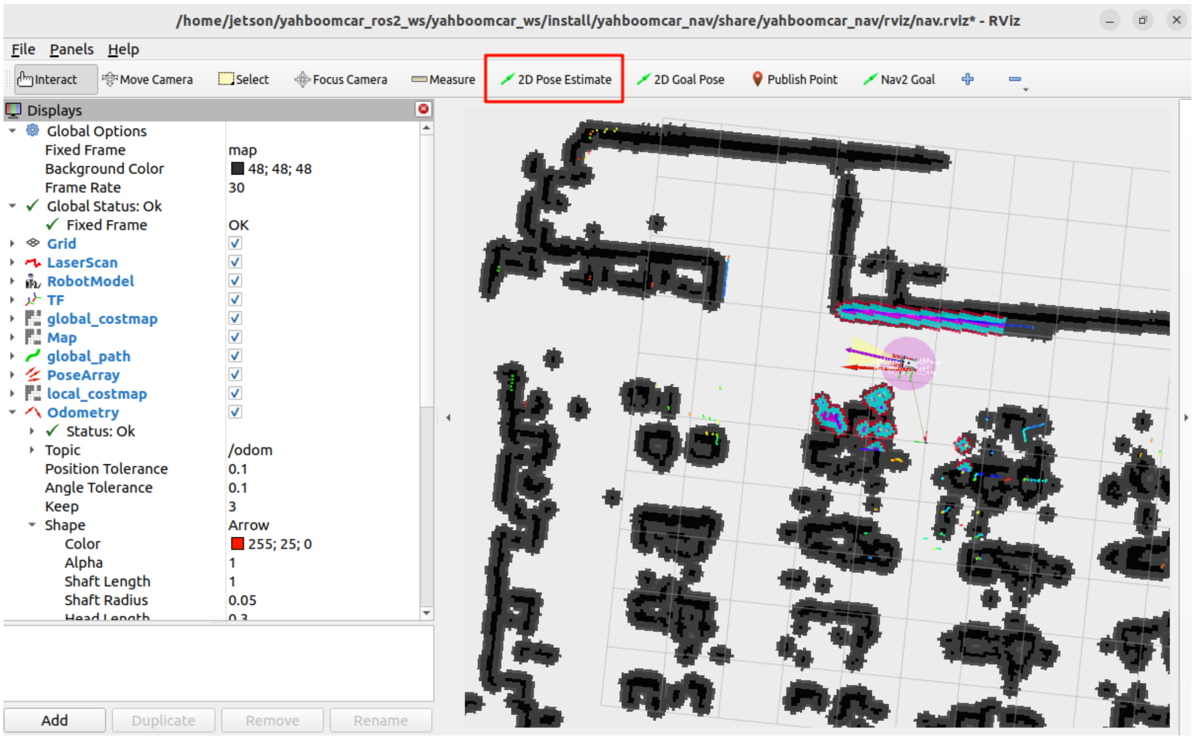
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

Wait for the navigation algorithm to start before the map is displayed. Then open the Docker terminal and enter

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

#Fast relocalization navigation (RDKX5 and Raspberry Pi 5 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/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
params_file:=/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/pa
rams/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 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
```

4.2 Test Case

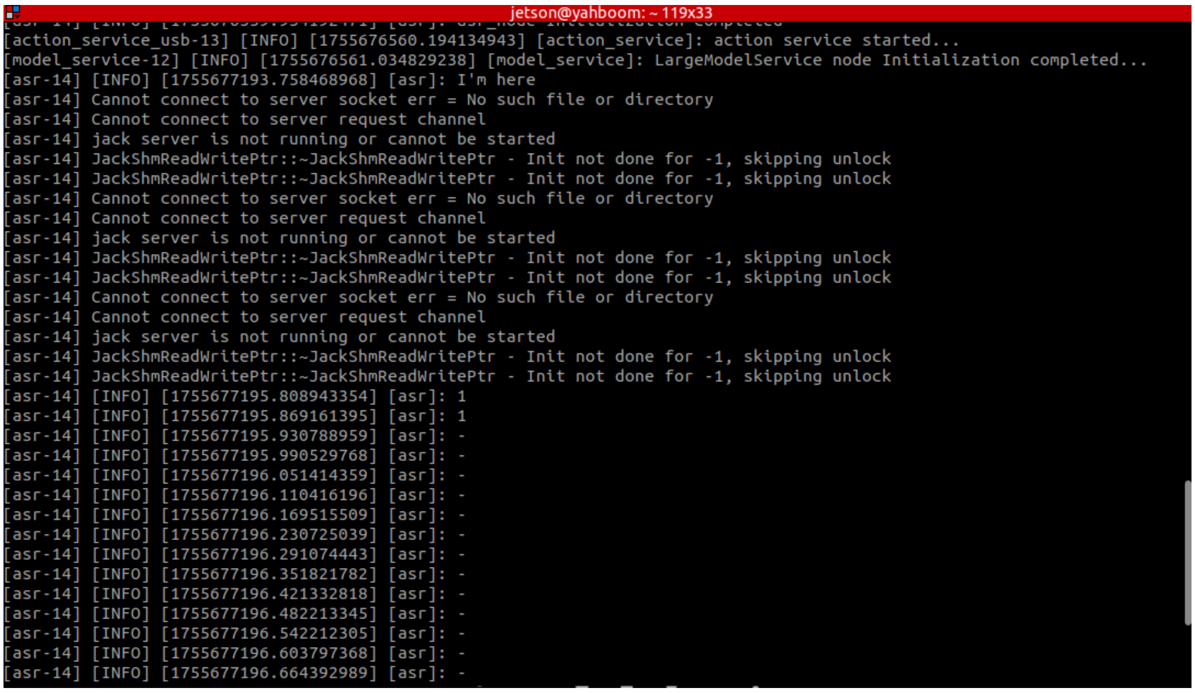
Here are some reference test cases; users can create their own dialogue commands.

- I'm in the office right now, and I'm feeling a bit thirsty.

4.2.1 Example

First, use "Hi, yahboom" to wake the robot. The robot responds, "I'm here, please." After the robot responds, the buzzer beeps briefly (beep—). The user can then speak. The robot then performs a sound detection, logging 1 if there's sound activity and - if there's no sound activity. When the speech ends, it detects the end of the voice, and stops recording if there's silence for more than 450ms.

The following figure shows the Voice Active Detection (VAD):

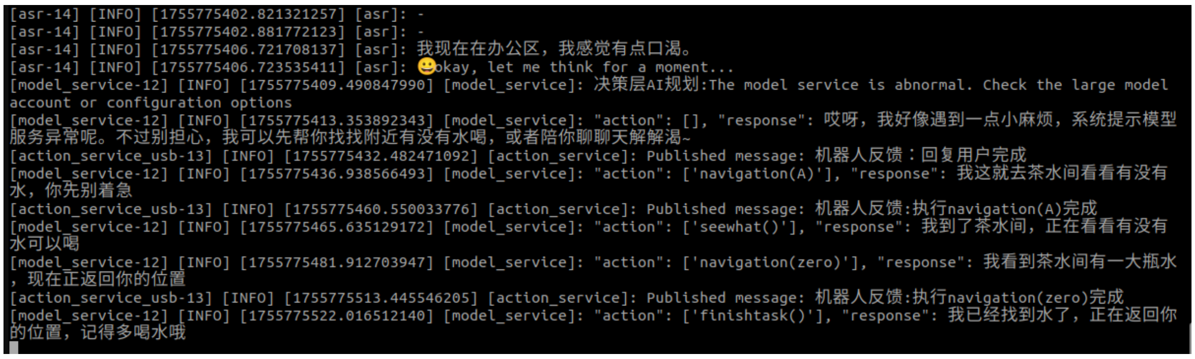


```
[action_service_usb-13] [INFO] [1755676560.194134943] [action_service]: action service started...
[model_service-12] [INFO] [1755676561.034829238] [model_service]: LargeModelService node Initialization completed...
[asr-14] [INFO] [1755677193.758468968] [asr]: I'm here
[asr-14] Cannot connect to 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 respond to the user's instructions. The terminal will print the following message:

Network Abnormality: Decision-making 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!

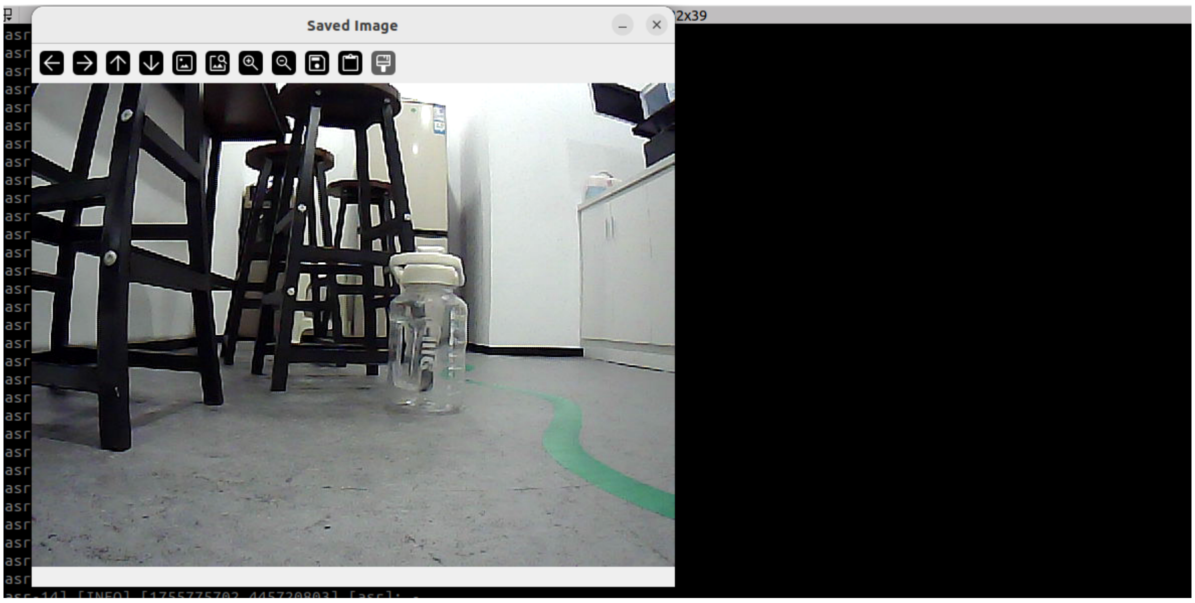
Normal: The decision-making AI planning will proceed through steps 1, 2, 3, 4, etc.



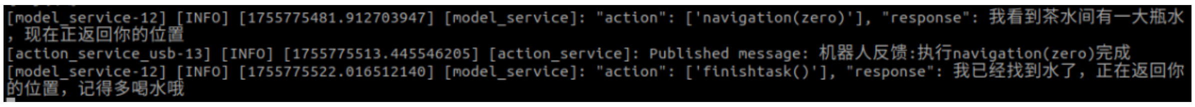
```
[asr-14] [INFO] [1755775402.821321257] [asr]: -
[asr-14] [INFO] [1755775402.881772123] [asr]: -
[asr-14] [INFO] [1755775406.721708137] [asr]: 我现在在办公区，我感觉有点口渴。
[asr-14] [INFO] [1755775406.723535411] [asr]: 😀okay, let me think for a moment...
[model_service-12] [INFO] [1755775409.490847990] [model_service]: 决策层AI规划:The model service is abnormal. Check the large model
account or configuration options
[model_service-12] [INFO] [1755775413.353892343] [model_service]: "action": [], "response": 哎呀，我好像遇到一点小麻烦，系统提示模型
服务异常呢。不过别担心，我可以先帮你找找附近有没有水喝，或者陪你聊聊天解解渴~
[action_service_usb-13] [INFO] [1755775432.482471092] [action_service]: Published message: 机器人反馈：回复用户完成
[model_service-12] [INFO] [1755775436.938566493] [model_service]: "action": ['navigation(A)'], "response": 我这就去茶水间看看有没有
水，你先别着急
[action_service_usb-13] [INFO] [1755775460.550033776] [action_service]: Published message: 机器人反馈:执行navigation(A)完成
[model_service-12] [INFO] [1755775465.635129172] [model_service]: "action": ['seewhat()'], "response": 我到了茶水间，正在看看有没有
水可以喝
[model_service-12] [INFO] [1755775481.912703947] [model_service]: "action": ['navigation(zero)'], "response": 我看到茶水间有一大瓶水
，现在正返回你的位置
[action_service_usb-13] [INFO] [1755775513.445546205] [action_service]: Published message: 机器人反馈:执行navigation(zero)完成
[model_service-12] [INFO] [1755775522.016512140] [model_service]: "action": ['finishtask()'], "response": 我已经找到水了，正在返回你
的位置，记得多喝水哦
```

`[model_service]: "action": ['seewhat()'], "response": I've arrived at the tea room and am checking for water.`

After arriving at 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 retrieved and uploaded to the large model for inference.

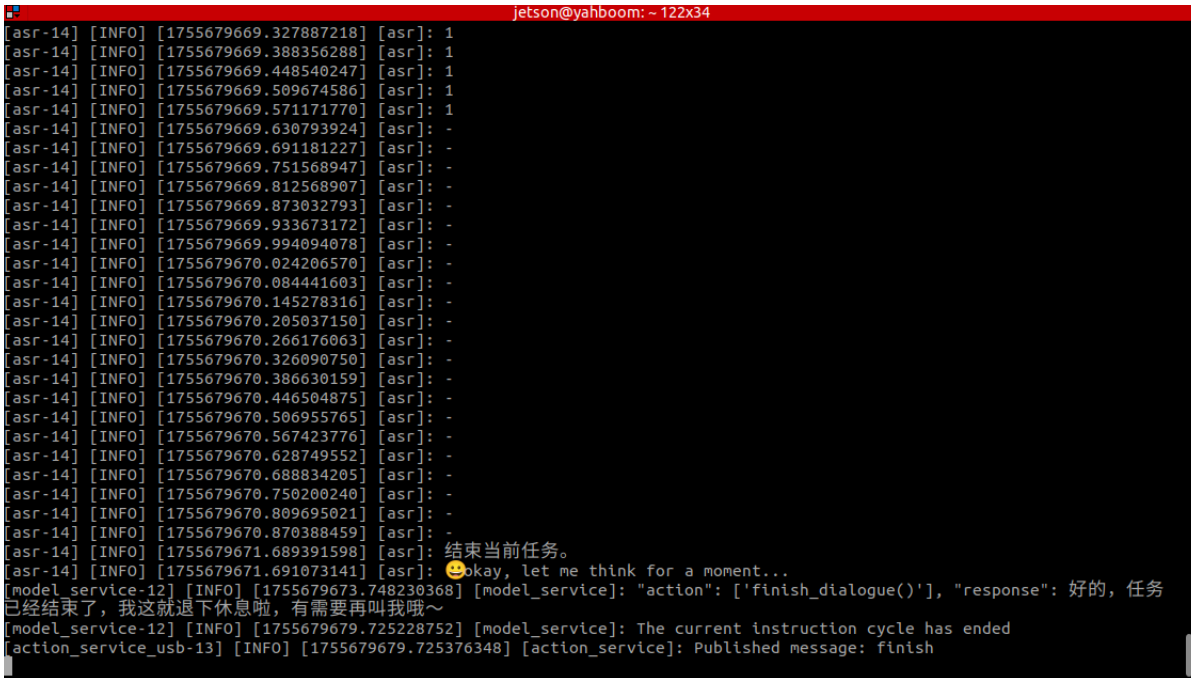


The large model receives the result, noting that there is a bottle of water in the tea room. It then provides feedback to the user and executes the next command: `Return to starting point`.



```
[model_service-12] [INFO] [1755775481.912703947] [model_service]: "action": ['navigation(zero)'], "response": 我看到茶水间有一大瓶水
，现在正返回你的位置
[action_service_usb-13] [INFO] [1755775513.445546205] [action_service]: Published message: 机器人反馈:执行navigation(zero)完成
[model_service-12] [INFO] [1755775522.016512140] [model_service]: "action": ['finishtask()'], "response": 我已经找到水了，正在返回你
的位置，记得多喝水哦
```

After completing all tasks, the robot returns to a free conversation state, but all conversation history is retained. At this point, you can wake yahboom up again and click "End Current Task" to end the robot's current task cycle, clear the conversation history, and start a new task cycle.


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
```

5. Source Code Analysis

Source code located at:

Jetson Orin Nano, Jetson Orin NX 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
```

5.1 Example

action_service.py Program:

This example uses the **seewhat**, **navigation**, **load_target_points**, and **get_dist** methods from the **CustomActionServer** class. **seewhat**, **navigation**, **load_target_points**, and **get_dist** are described in the previous sections [2. Multimodal Visual Understanding], [6. Multimodal Visual Understanding + Depth Camera Distance Question Answering], and [7. Multimodal Visual Understanding + SLAM] navigation] have been explained, and there are no new procedures in this chapter.