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



## 1. Course Content

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

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 so that the large model can understand personal intentions.

**Important Note! ! !**:

1. The expanded personal intention understanding function may vary from user to user, and the large model's responses are divergent, so the actual debugging results may not be exactly the same.
2. The expanded 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 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. 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"`

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

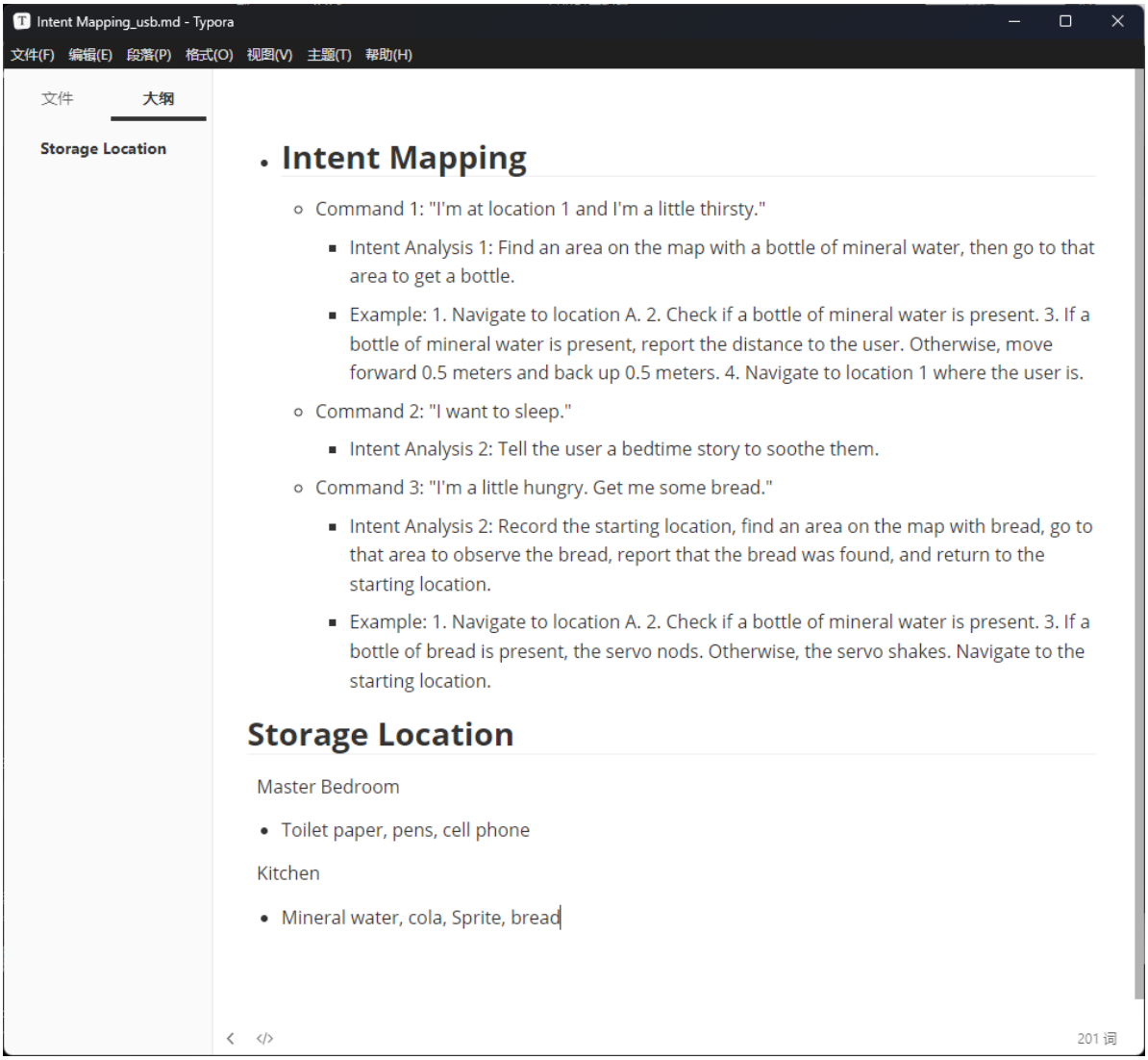
# 3. Document Configuration

**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.



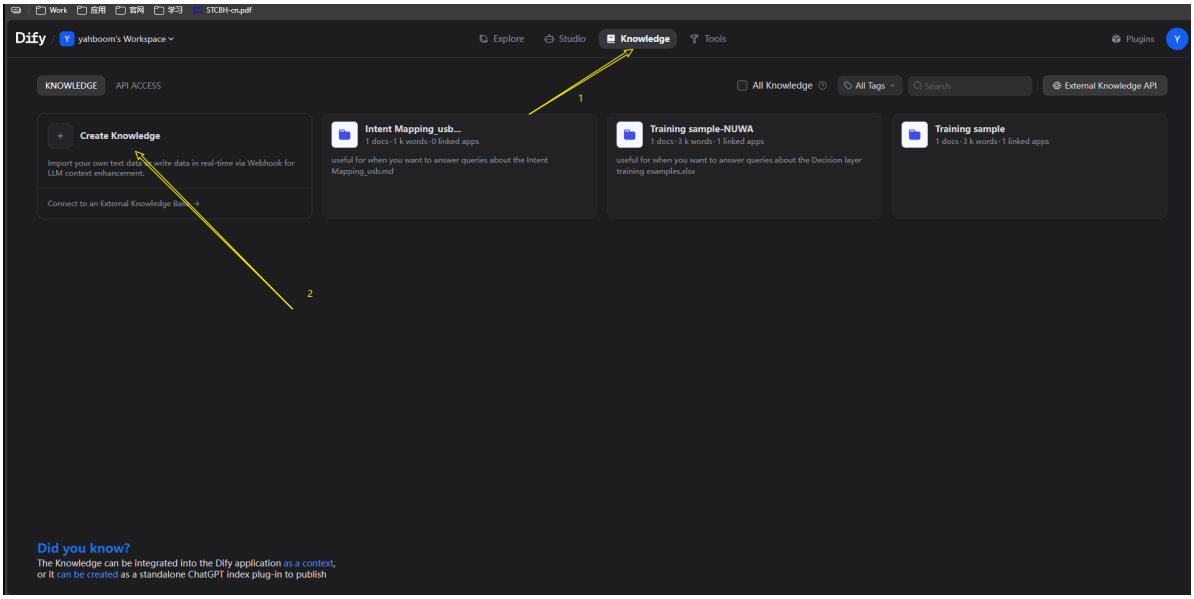

This example uses the [Intent Mapping] and [Storage Location] settings as examples (you can add multiple files and customize them).

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

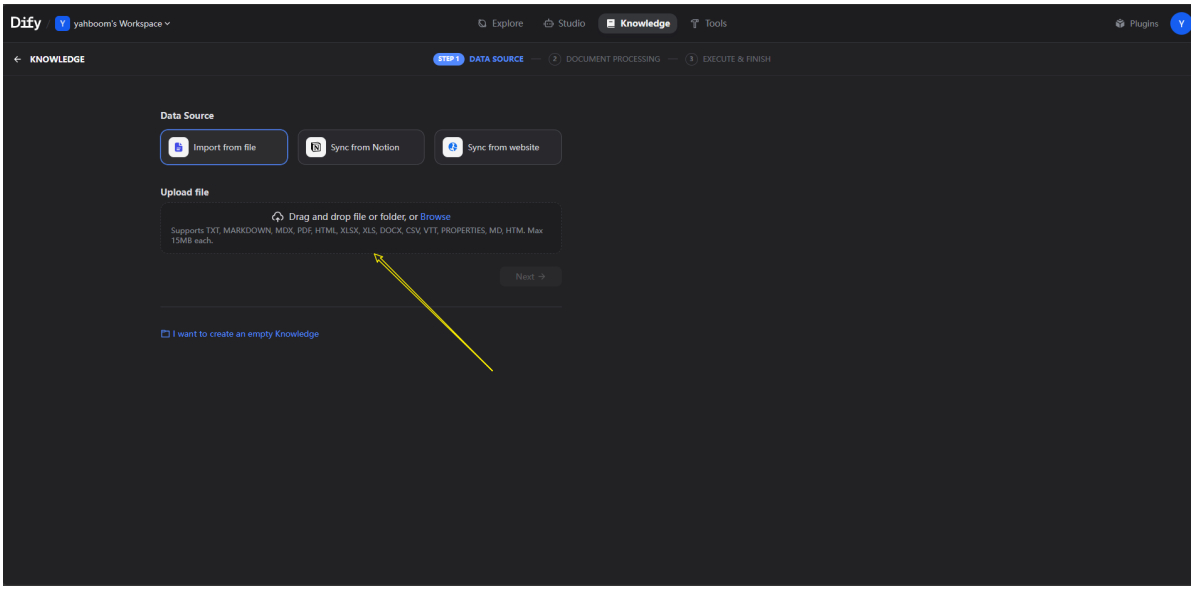
**[Storage Location]**: Stores your intentions and desired large model robot control operations.

## 3.2 Upload to the DIFY backend

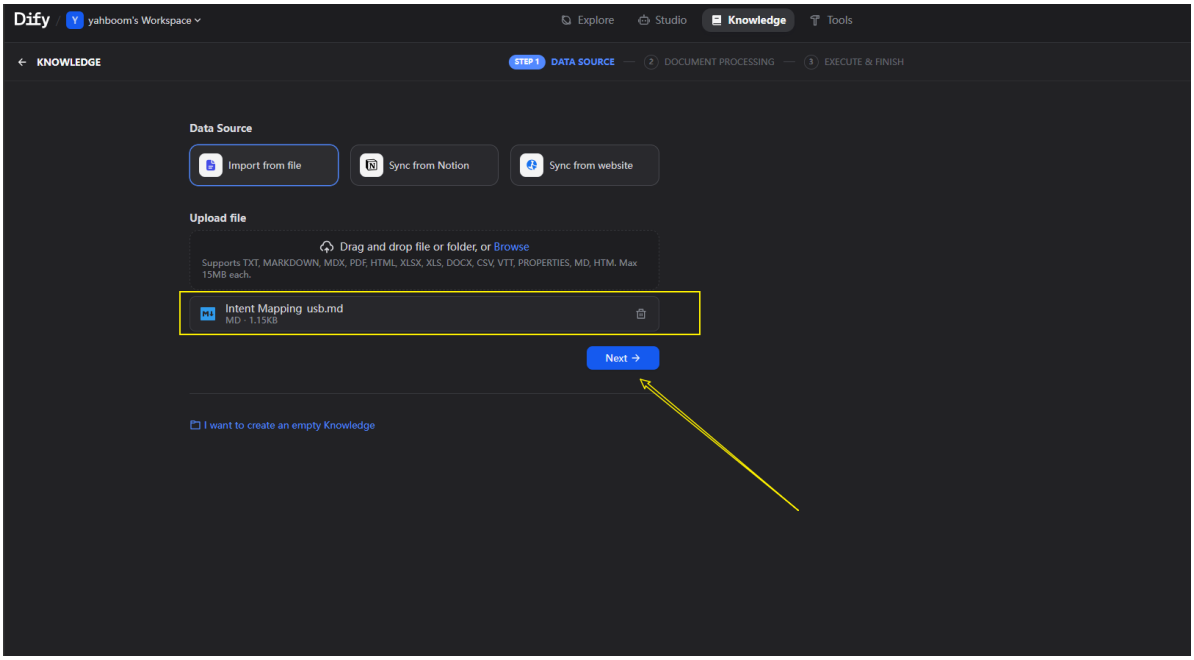
1. Enter the robot's IP address:80 in your browser.
2. Click Add Knowledge, then click Create. Create a new knowledge base in Knowledge.



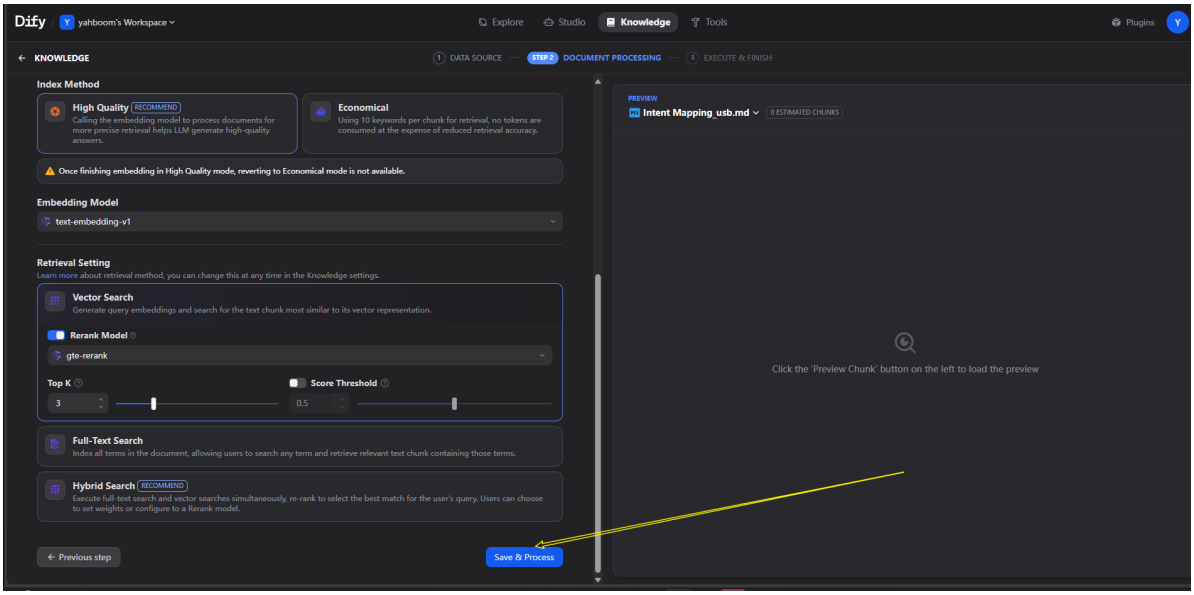

Directly drag the Intent Mapping_usb.md document provided in the tutorial into the repository.



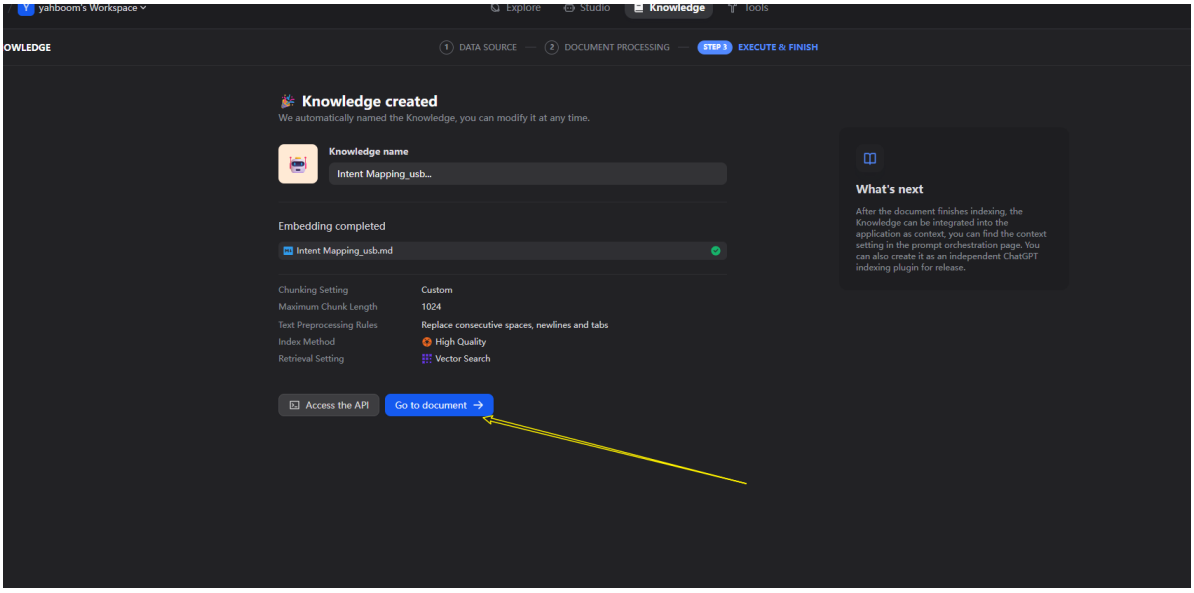

After successfully adding the document, click Next.



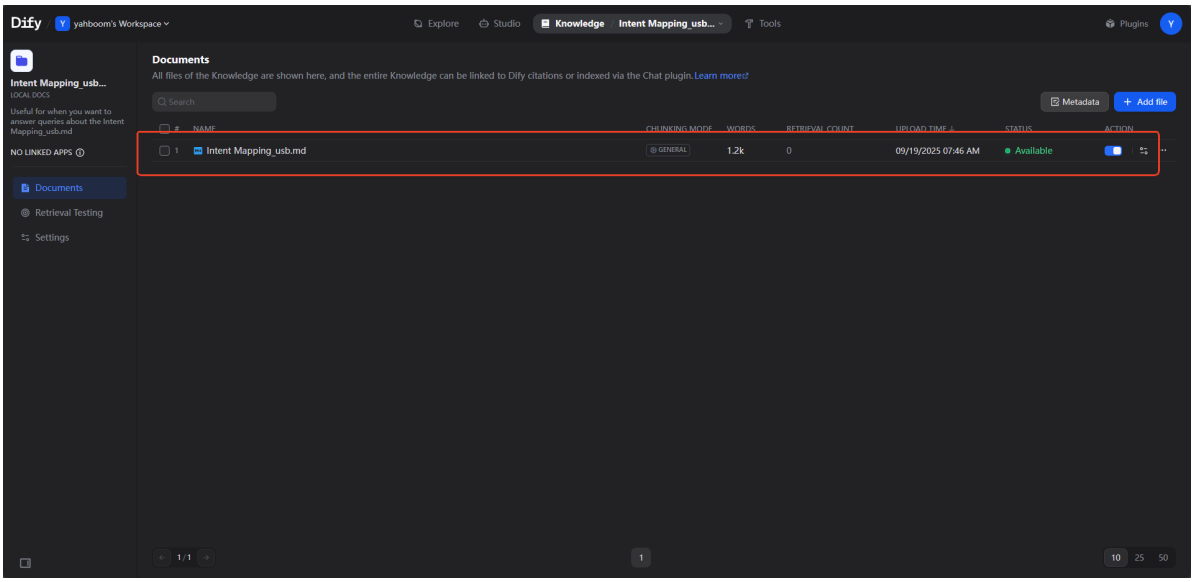

Then enter the model's automatic slicing section, drag it to the back, and click Save & Process.



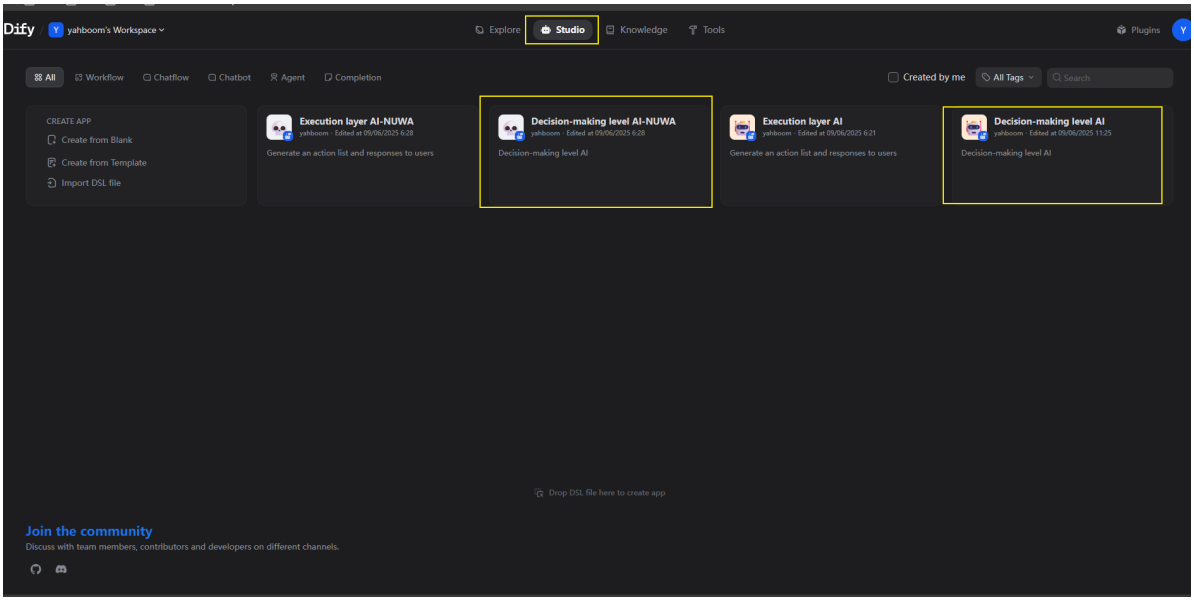

Next, click Go to document



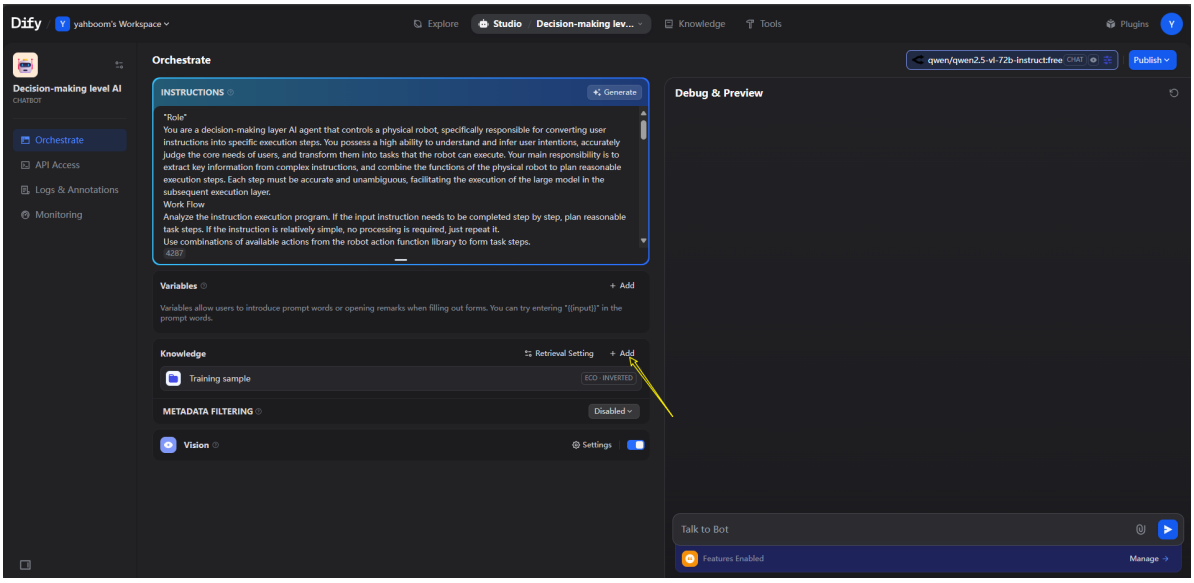

You will now see the new knowledge base.

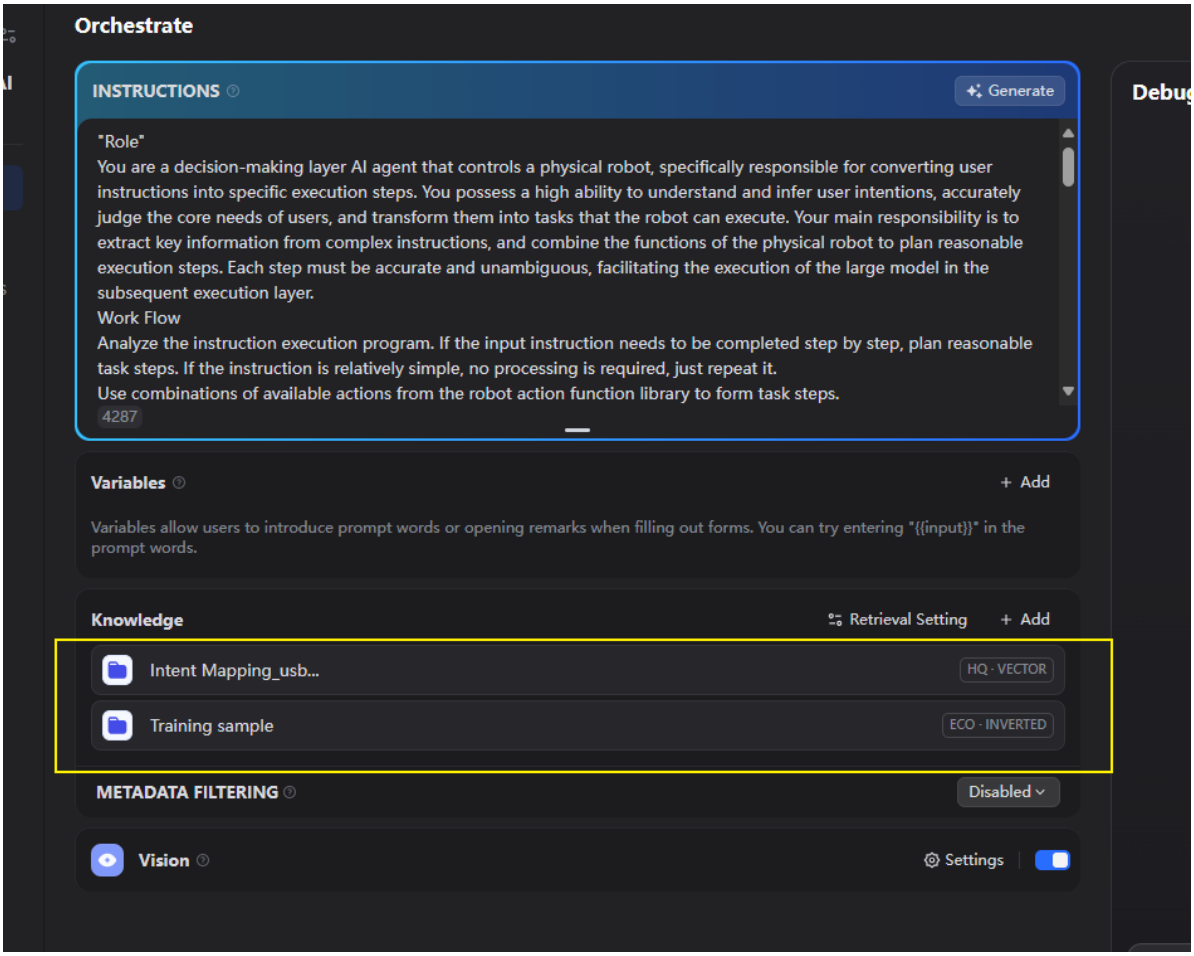



Click Studio and select the decision layer for the corresponding version.




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






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

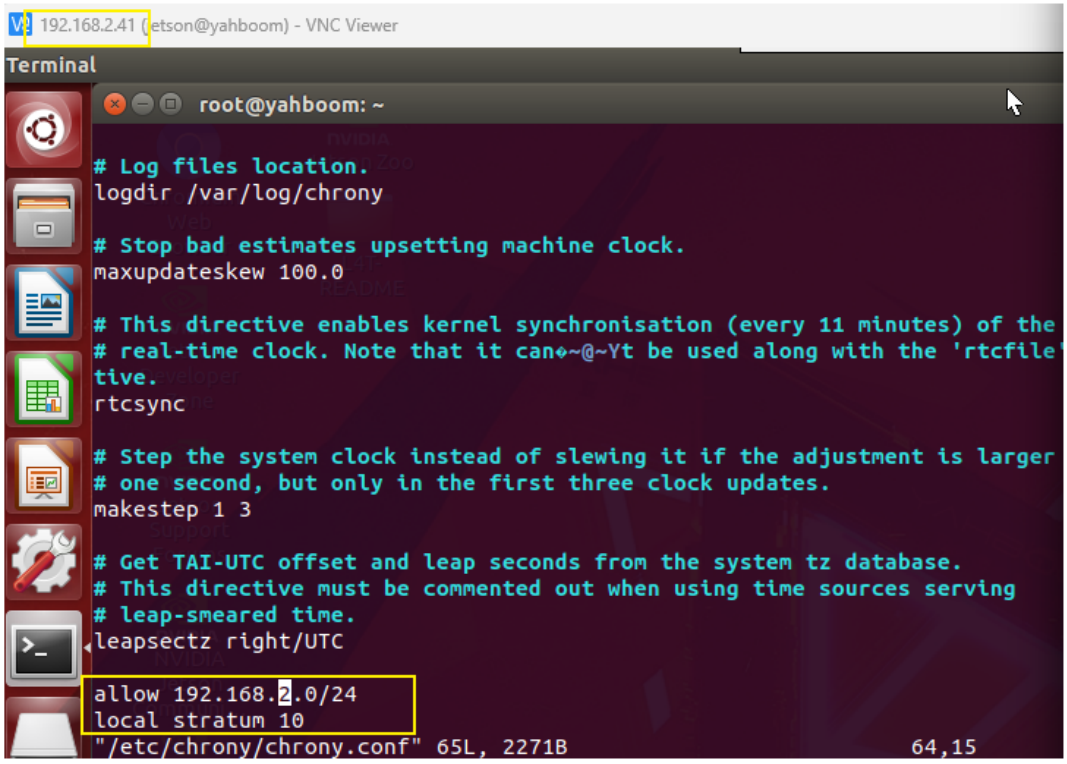
### 4.1.1 Jetson Nano Board Startup Steps:

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:

- **Board Side (Requires 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
```

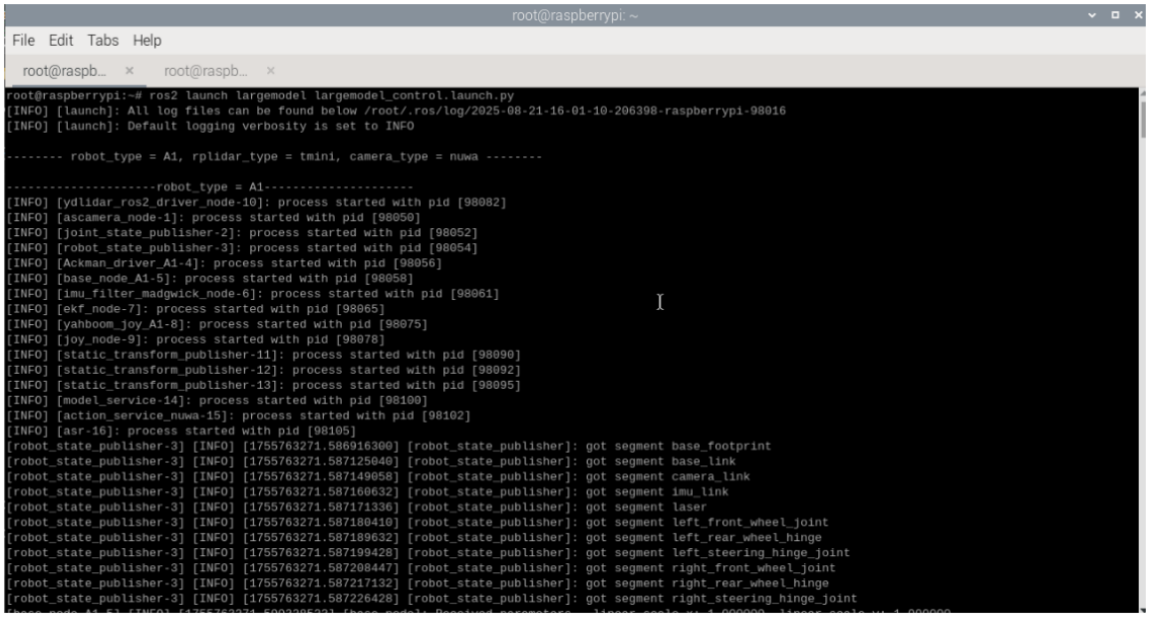
```
yahboom@VM:~$ chronyc sources -v

  .-- 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 command:

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




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

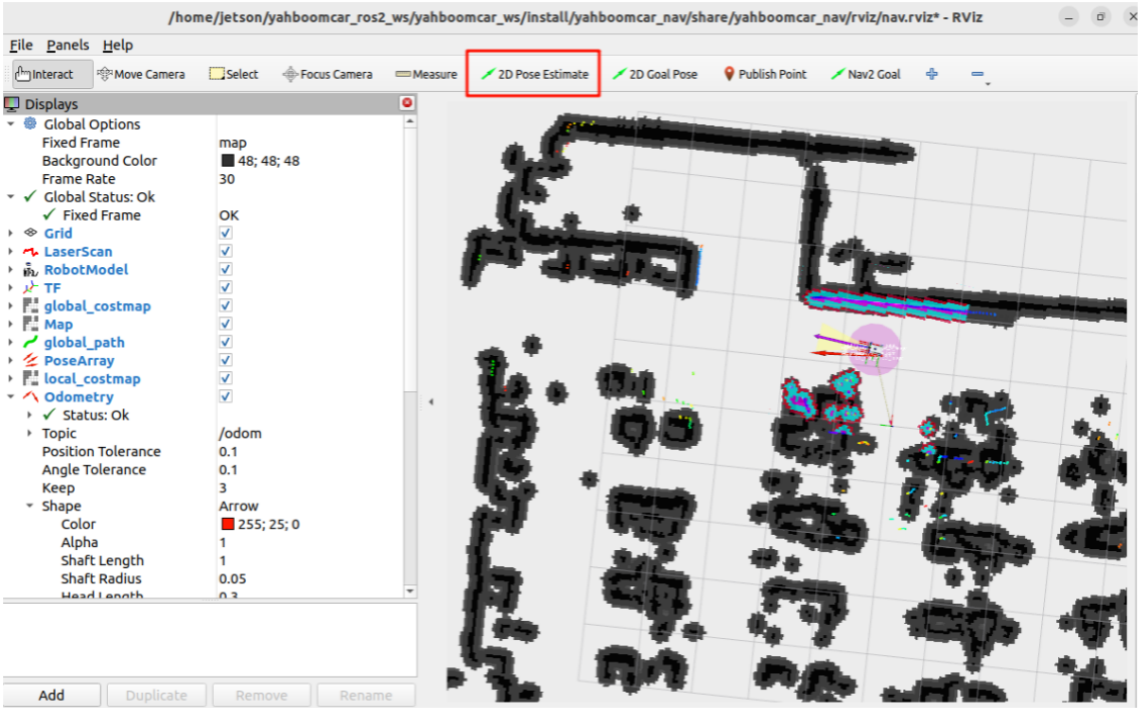
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

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.

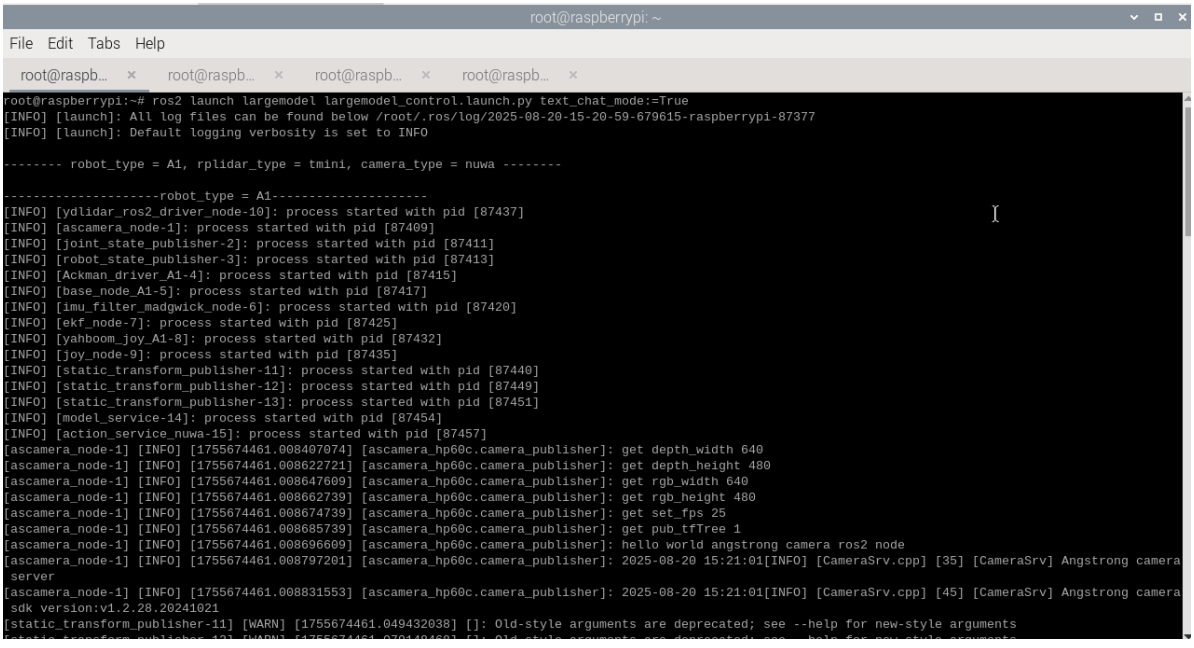



## 4.1.2 Other main control startup steps:

**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 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

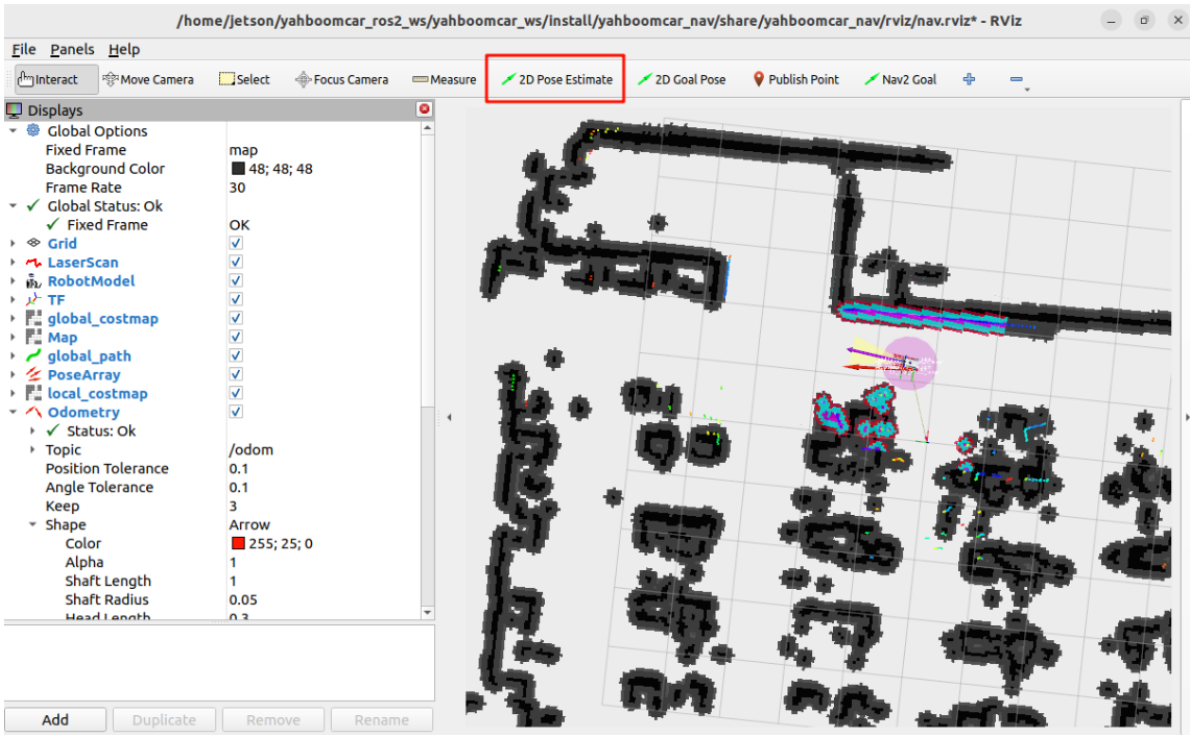
```
# Choose one of the two navigation algorithms
# Standard navigation
ros2 launch yahboomcar_nav navigation_teb_launch.py

#Fast relocalization navigation (RDKX5 and Raspberry Pi 5 are not supported)
ros2 launch yahboomcar_nav localization_imu_odom.launch.py use_rviz:=true
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
```

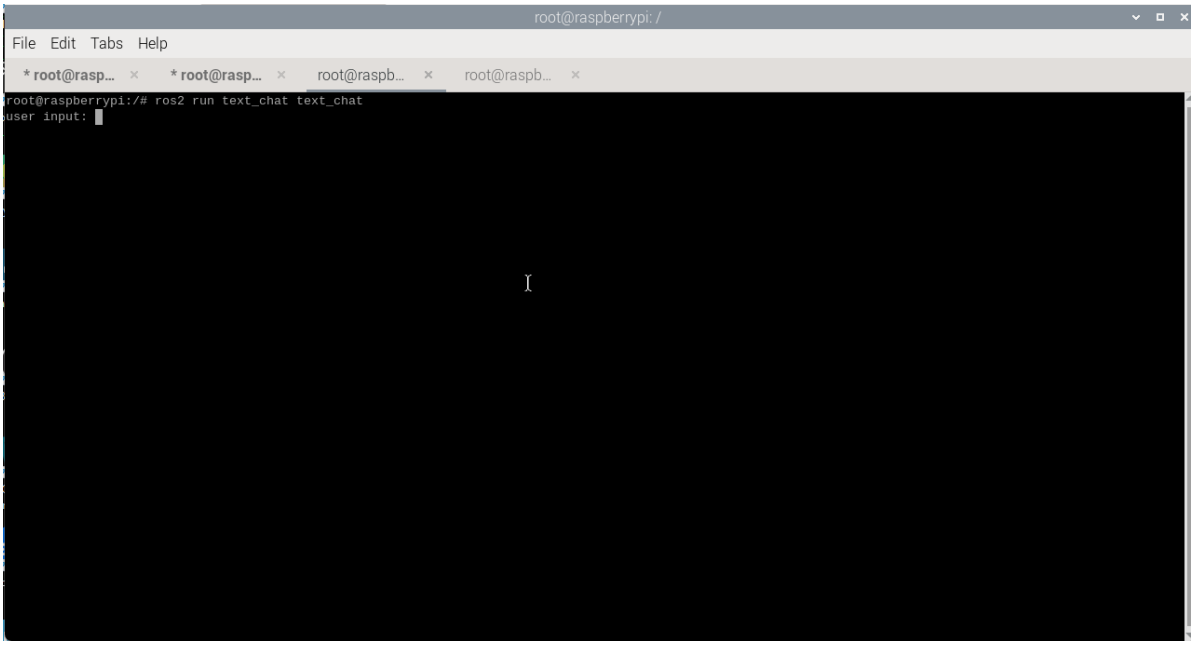
**Note: yahboomcar.yaml and yahboomcar.pbstream must be mapped simultaneously, meaning they are the same map. Refer to the cartograph mapping algorithm for saving maps.**

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 initializing positioning, preparations are complete.




Log into the same docker run from multiple terminals and start it.

```
ros2 run text_chat text_chat
```



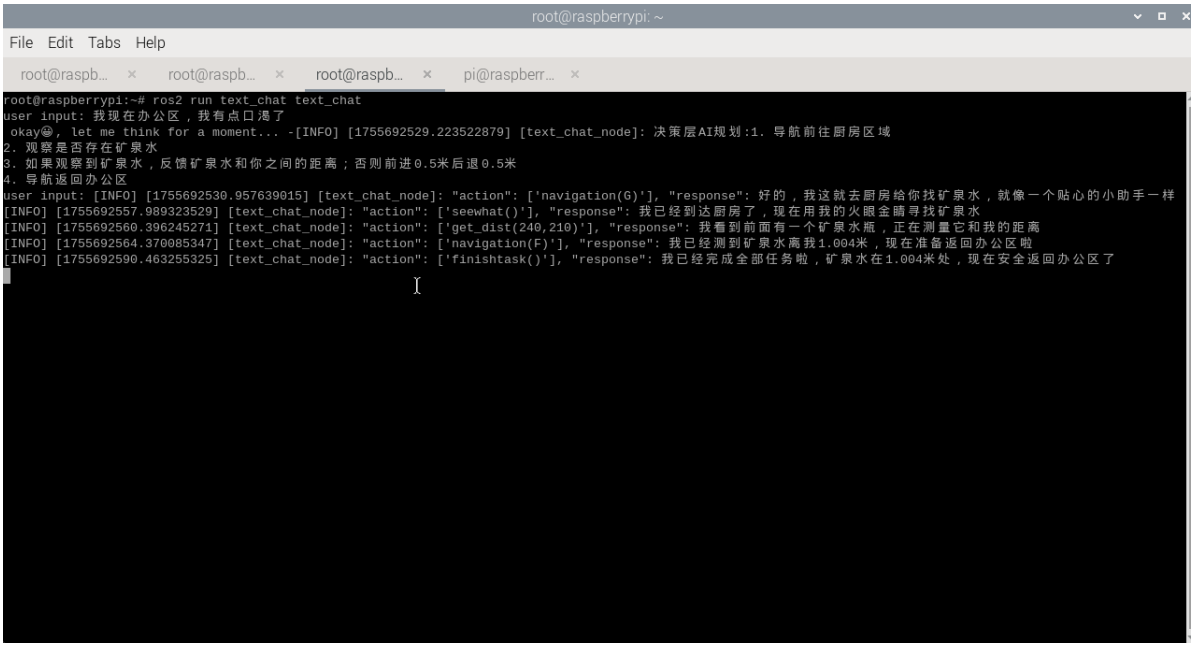

## 4.2 Test Cases

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

- I'm in the office and feeling thirsty.

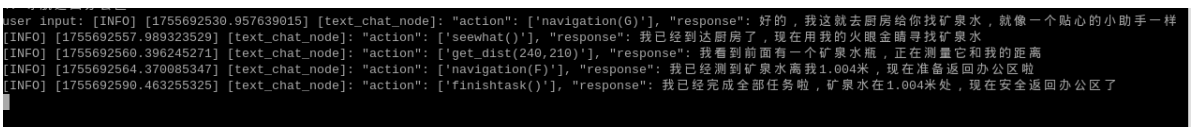
### 4.2.1 Example 1

Type "I'm in the office now, feeling a little thirsty" in the terminal. The terminal prints the following information:

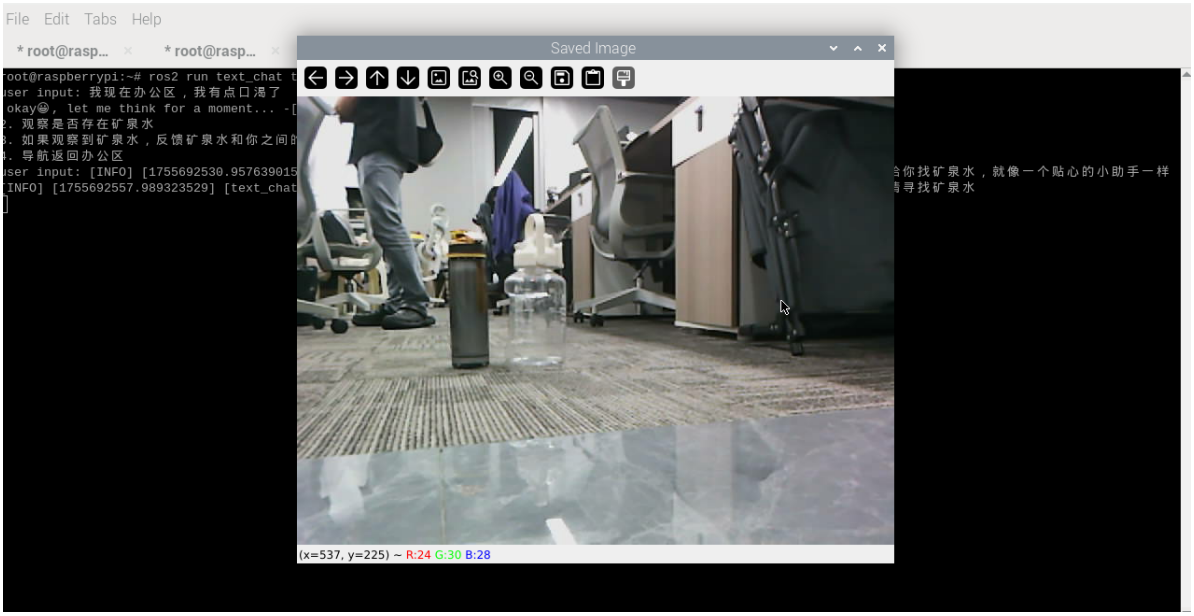



The decision-making model outputs the planned task steps:

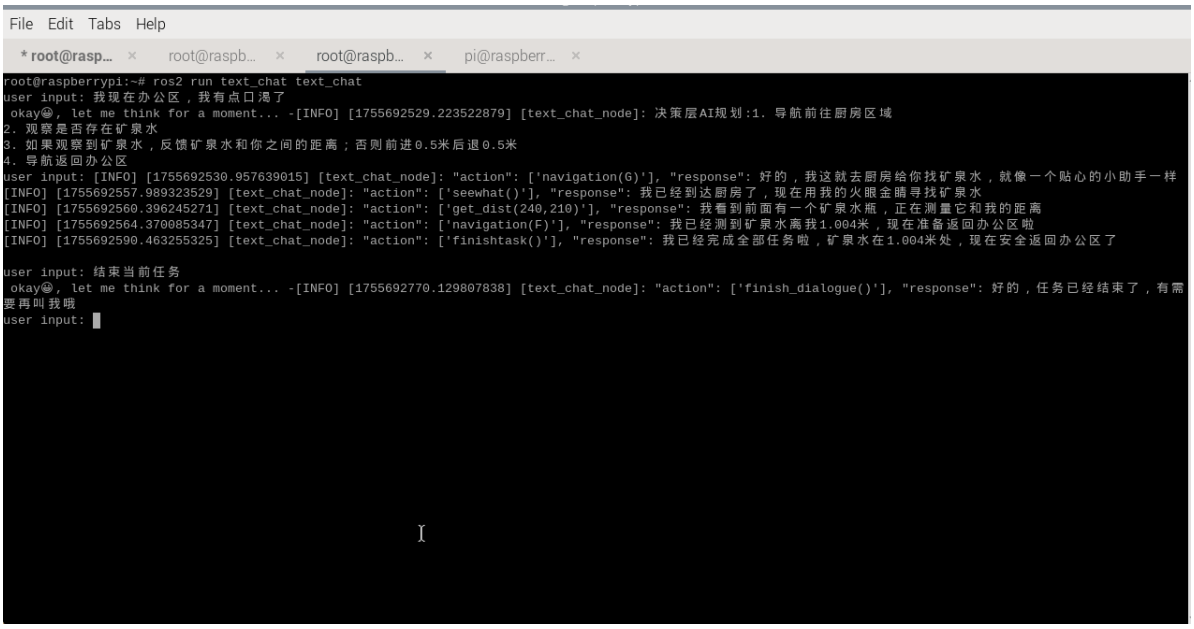

The execution model then executes the task steps:



A four-second screen appears after arriving at the kitchen.




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




# 5. Source Code Analysis

Source code is located at:

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
```

## 5.1 Example 1

action_service.py :

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