

Intention understand planning+Multimodal visual understand+robotic arm grab+SLAM Navigation

1. Course Content

Note: This chapter requires you to first complete the configuration of the map mapping file according to the [Multimodal Visual Understanding + SLAM Navigation] chapter course.

Learn to use the RAG knowledge base to train personal intent understanding

Course Review:

The concept and function of the RAG knowledge base are explained in the chapter [2. Basic Knowledge of AI Big Models - 2. RAG Retrieval Enhancement and Model Training Samples]. In the chapter [2. Basic Knowledge of AI Big Models - 5. Configuring AI Big Models], the section [5. RAG Knowledge Base and Training Samples] explains how to configure and expand the RAG knowledge base, and also configures the necessary [Robot Action Function Library].

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

Special attention!!! :

1. The ability to expand personal intent understanding varies from user to user, and the large model's responses are divergent, so the actual debugging results may not be exactly the same.
2. Expanding the personal intent understanding function must comply with social norms and laws and regulations. Technical services are not responsible for the final debugging results and impacts of this course.

2. Preparation

2.1 Content Description

This course uses the Jetson Orin NX 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 course in the terminal. For instructions on entering the Docker container from the host computer, refer to the [Configuration and Operation Guide] -- [Enter the Docker (Jetson Nano and Raspberry Pi 5 users see here)] section of this product tutorial. For Orin and NX boards, simply open a terminal and enter the commands mentioned in this course.

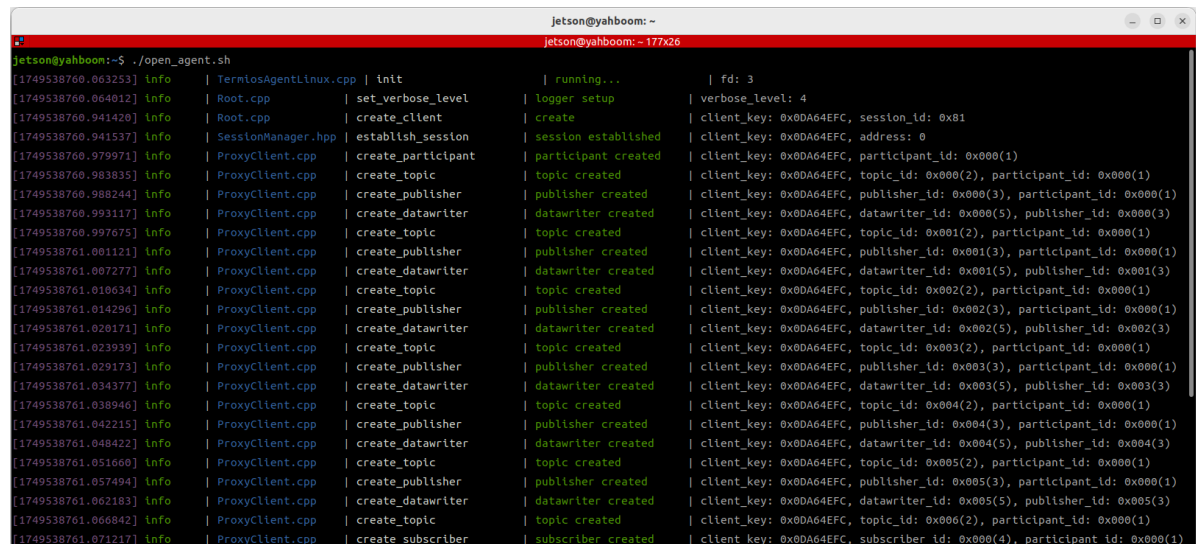
2.2 Start the Agent

Note: To test all cases, you must start the docker agent first. If it has already been started, you do not need to start it again.

Enter the command in the vehicle terminal:

```
sh start_agent.sh
```

The terminal prints the following information, indicating that the connection is successful

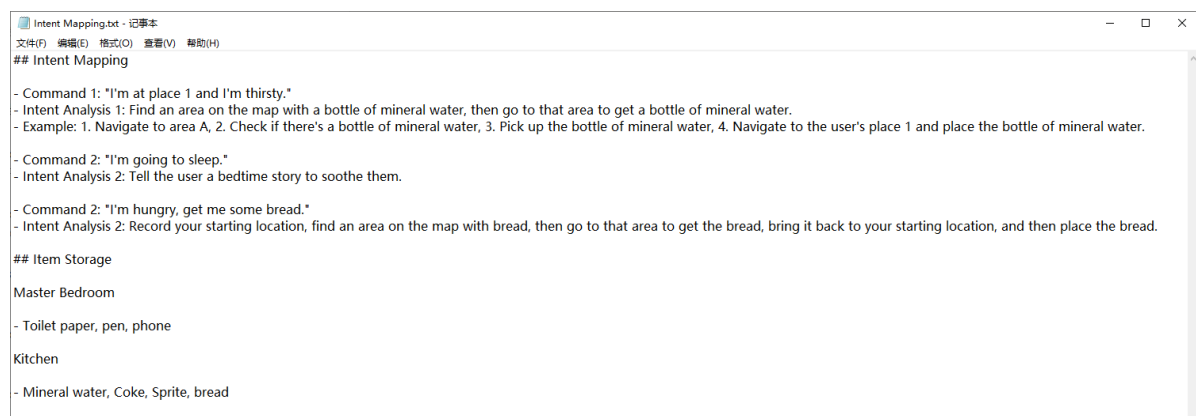


```
jetson@yahboom:~$ ./open_agent.sh
[1749538760.063253] Info | TermiosAgentLinux.cpp | Init | running... | fd: 3
[1749538760.064012] Info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4
[1749538760.941420] Info | Root.cpp | create_client | create | client_key: 0x0DA64EFC, session_id: 0x81
[1749538760.941537] Info | SessionManager.hpp | establish_session | session established | client_key: 0x0DA64EFC, address: 0
[1749538760.979971] Info | ProxyClient.cpp | create_participant | participant created | client_key: 0x0DA64EFC, participant_id: 0x000(1)
[1749538760.983835] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x000(2), participant_id: 0x000(1)
[1749538760.988244] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x000(3), participant_id: 0x000(1)
[1749538760.993117] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1749538760.997675] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x001(2), participant_id: 0x000(1)
[1749538761.001121] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x001(3), participant_id: 0x000(1)
[1749538761.007277] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x001(5), publisher_id: 0x001(3)
[1749538761.010634] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x002(2), participant_id: 0x000(1)
[1749538761.014296] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x002(3), participant_id: 0x000(1)
[1749538761.020171] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x002(5), publisher_id: 0x002(3)
[1749538761.023939] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x003(2), participant_id: 0x000(1)
[1749538761.029173] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x003(3), participant_id: 0x000(1)
[1749538761.034377] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x003(5), publisher_id: 0x003(3)
[1749538761.038946] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x004(2), participant_id: 0x000(1)
[1749538761.042215] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x004(3), participant_id: 0x000(1)
[1749538761.048422] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x004(5), publisher_id: 0x004(3)
[1749538761.051660] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x005(2), participant_id: 0x000(1)
[1749538761.057494] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x005(3), participant_id: 0x000(1)
[1749538761.062183] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x005(5), publisher_id: 0x005(3)
[1749538761.066842] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x006(2), participant_id: 0x000(1)
[1749538761.071217] Info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x0DA64EFC, subscriber_id: 0x000(4), participant_id: 0x000(1)
```

3. Configuration Document

3.1 Create a txt document

Open the [Intent Mapping.txt] in the folder of this course



```
Intent Mapping.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
## Intent Mapping

- Command 1: 'I'm at place 1 and I'm 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 of mineral water.
- Example: 1. Navigate to area A, 2. Check if there's a bottle of mineral water, 3. Pick up the bottle of mineral water, 4. Navigate to the user's place 1 and place the bottle of mineral water.

- Command 2: 'I'm going to sleep.'
- Intent Analysis 2: Tell the user a bedtime story to soothe them.

- Command 2: 'I'm hungry, get me some bread.'
- Intent Analysis 2: Record your starting location, find an area on the map with bread, then go to that area to get the bread, bring it back to your starting location, and then place the bread.

## Item Storage

Master Bedroom

- Toilet paper, pen, phone

Kitchen

- Mineral water, Coke, Sprite, bread
```

This case takes the setting of [Intent Mapping] and [Item Storage Location] as an example (you can add multiple files and more customized information by yourself)

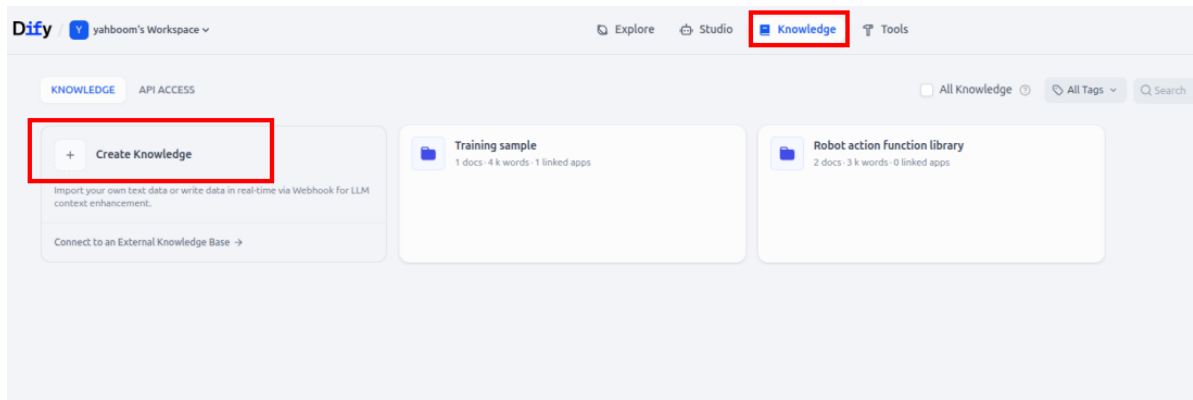
[Intent Mapping] : Stores personal intentions and expectations to control the robot to operate

[Item Storage Location] : Store personal intentions and expectations. Large models control the robot to operate.

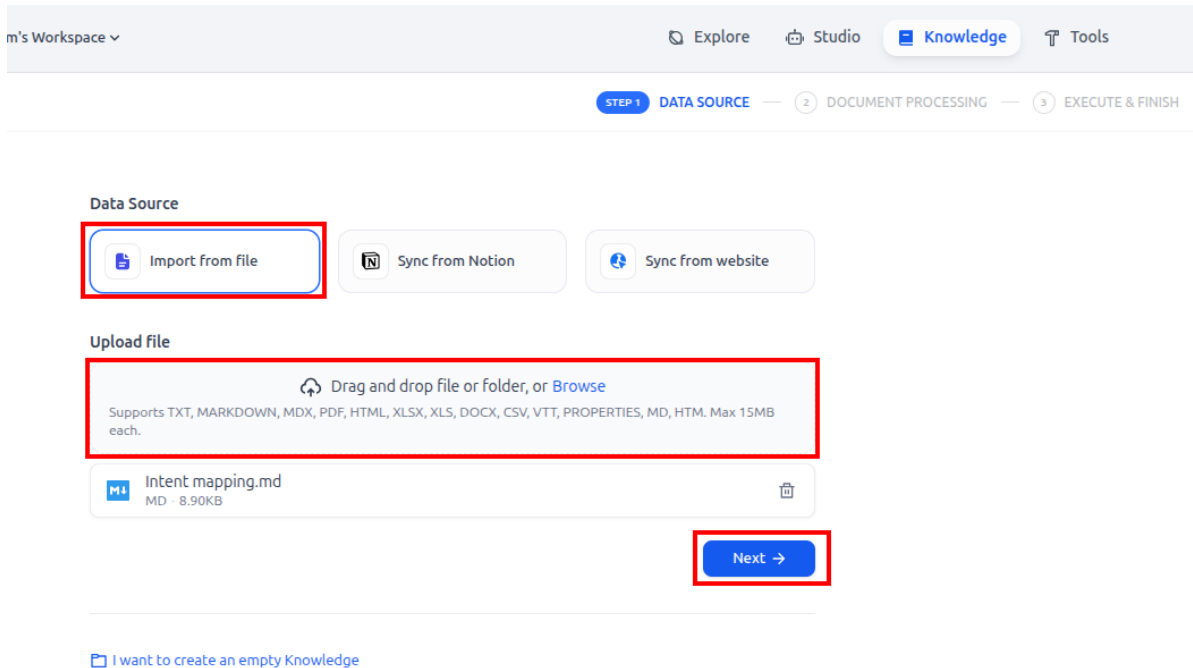
3.2 Upload to RAG Knowledge Base

Then follow the steps in [2. AI Big Model Basics - 5. Configuring AI Big Model - 7. International Version Configuration - 7.5 Expanding the Knowledge Base] to upload the .txt format document.

Click on the knowledge in the main interface of the diify interface, and then click Create knowledge to create a new knowledge base



Select import from file, then click browser to browse to browse local files and upload, and click Next after uploading



Select economic mode, keep the others by default

Dify

Y

yahboom's Workspace

Explore

Studio

←

KNOWLEDGE

1 DATA SOURCE

STEP 2 DOCU

Chunk Settings

General

General text chunking mode, the chunks retrieved and recalled are the same.

Delimiter

Maximum chunk length

Chunk overlap

\n\n

1024

characters

50

characters

Text Pre-processing Rules

☒ Replace consecutive spaces, newlines and tabs

☐ Delete all URLs and email addresses

☐ Chunk using Q&A format in

English

Preview Chunk

Reset

Parent-child

When using the parent-child mode, the child-chunk is used for retrieval and the parent-chunk is used for recall as context.

Index Method

High Quality

RECOMMEND

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.

Retrieval Setting

Wait for the document embedding to complete, and the intent mapping knowledge base creation can be completed

Knowledge created

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



Knowledge name

Intent mapping.md...

Embedding completed

Intent mapping.md

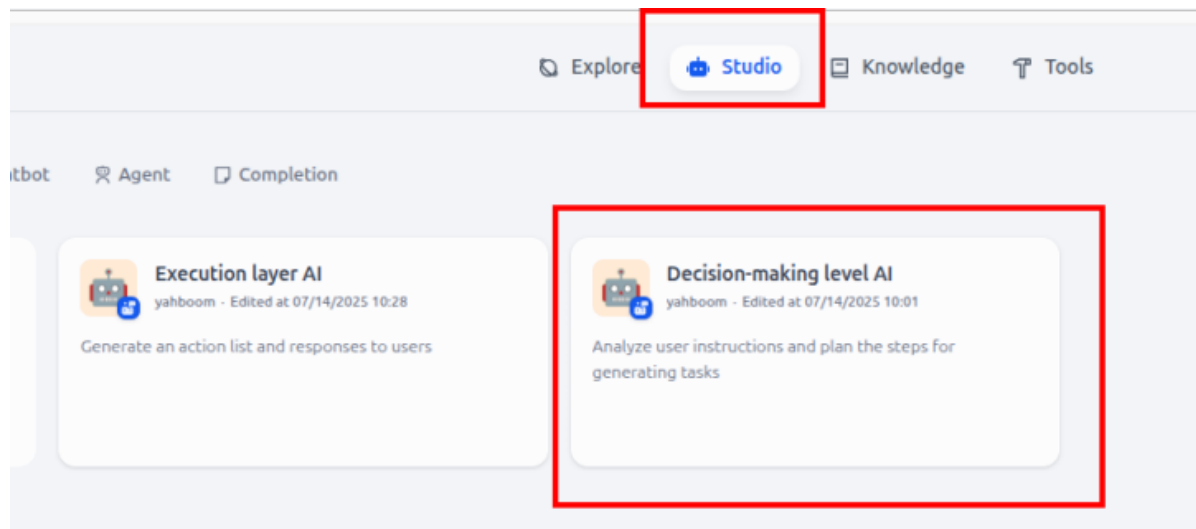


Chunking Setting	Custom
Maximum Chunk Length	1024
Text Preprocessing Rules	Replace consecutive spaces, newlines and tabs
Index Method	Economical
Retrieval Setting	Inverted Index

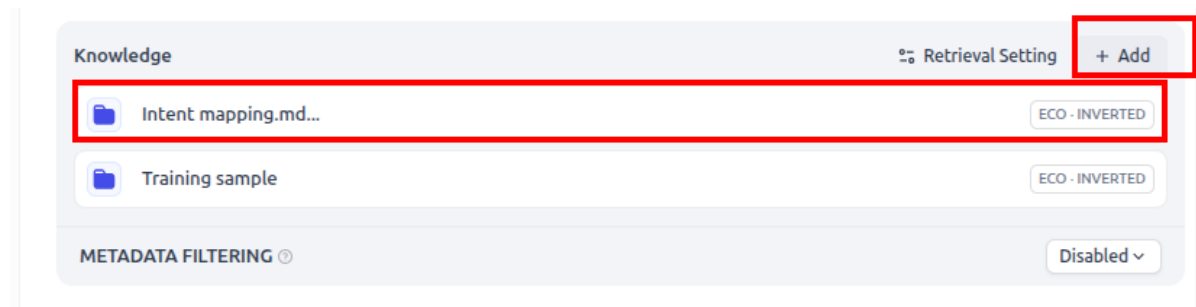
Access the API

Go to document →

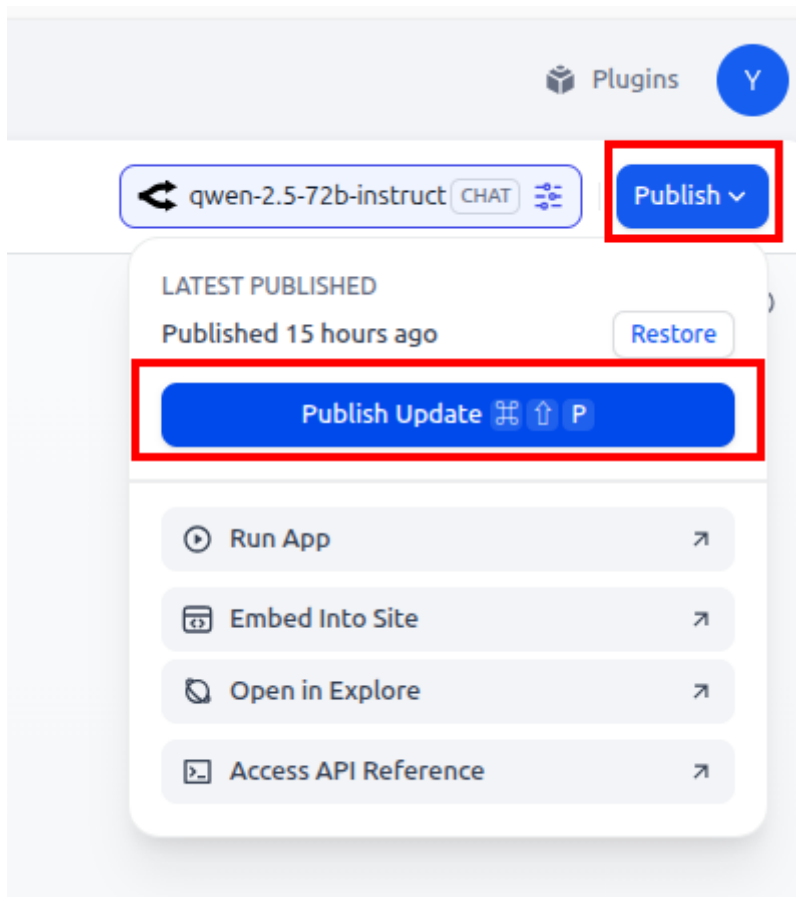
Switch back to the home page and click Configure the decision-making layer model



Click add to add the knowledge base for the intent mapping just created



Click Publish to save the effective configuration

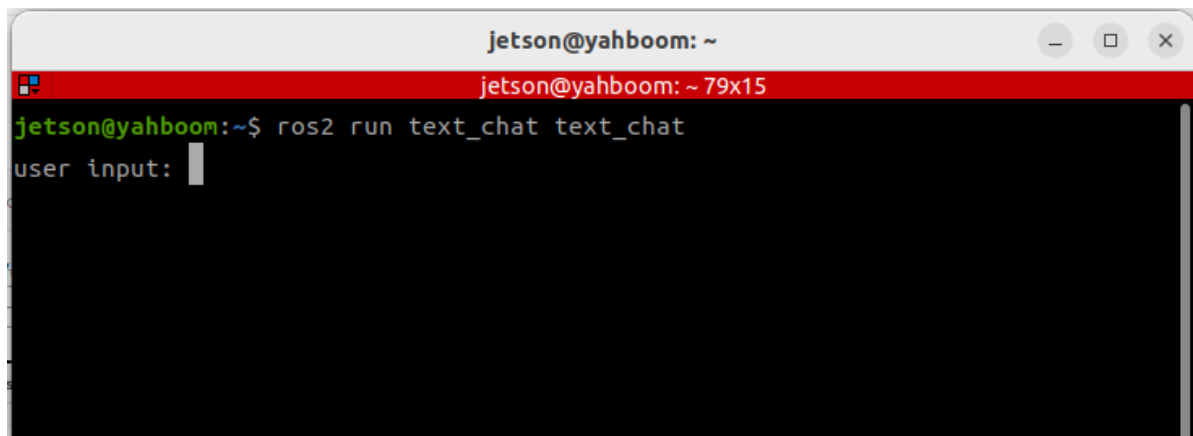


4. Run the case

4.1 Startup Program

Take the virtual machine as an example, open a terminal and start

```
ros2 run text_chat text_chat
```



The vehicle computer starts the large model control node:

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

Open two terminals on the vehicle and enter commands to start the navigation function:

```
ros2 launch M3Pro_navigation base_bringup.launch.py
ros2 launch M3Pro_navigation navigation2.launch.py
```

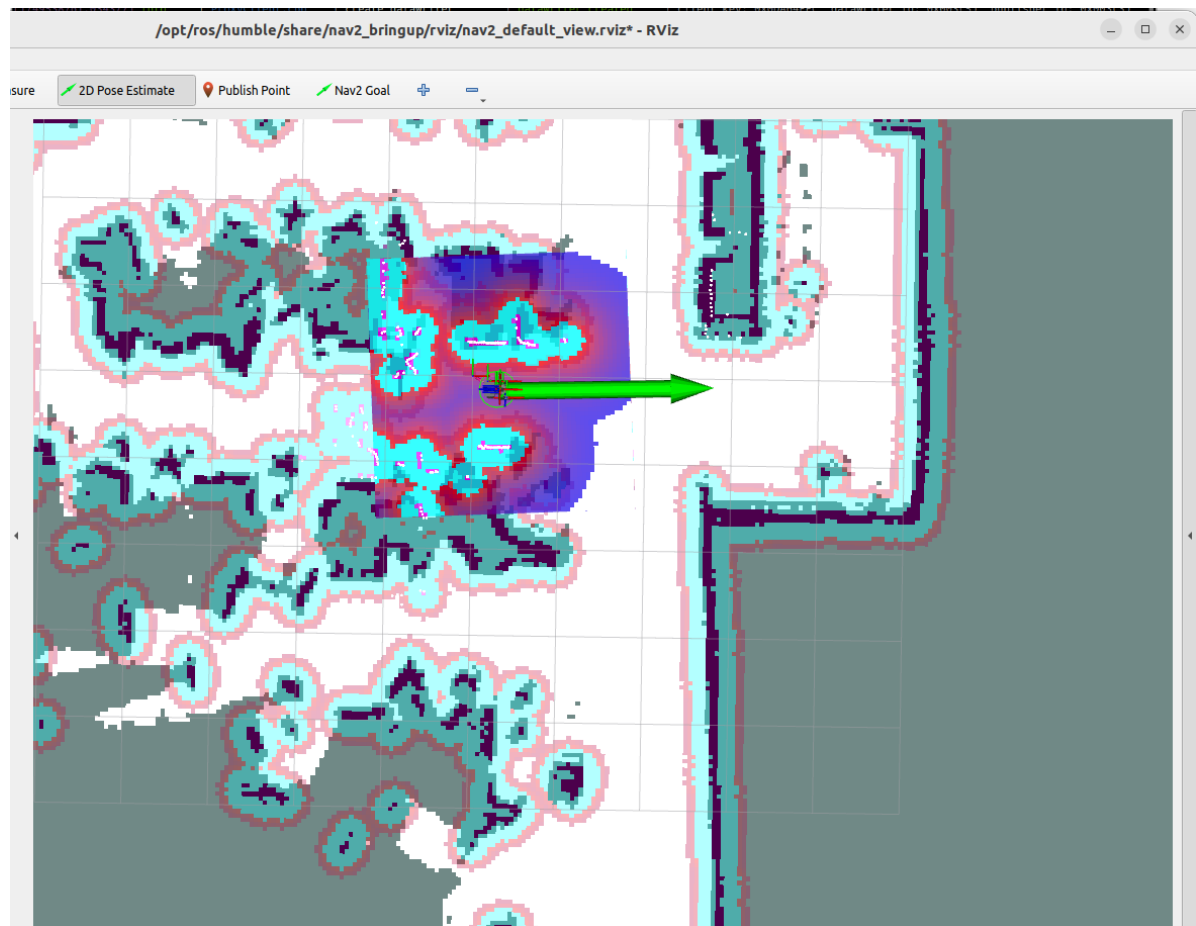
Create a new terminal on the virtual terminal to start the rviz visualization interface:

```
ros2 launch slam_view nav_rviz.launch.py
```

Or start the rviz visualization page on the car side:

```
ros2 launch M3Pro_navigation nav_rviz.launch.py
```

Then follow the process of starting the navigation function to initialize the positioning. The rviz2 visualization interface will open. Click **2D Pose Estimate** in the upper toolbar to enter the selection state. Roughly mark the position and orientation of the robot on the map. After initialization positioning, the preparation work is completed.



4.2 Test Cases

Here are some test cases for reference. Users can write their own dialogue commands.

- I'm in the master bedroom now and I'm feeling a little thirsty

Enter the test case in the text interactive terminal:

```
jetson@yahboom: ~  
jetson@yahboom: ~ 79x15  
jetson@yahboom:~$ ros2 run text_chat text_chat  
user input: 我现在在主卧室，我感觉有点口渴了
```

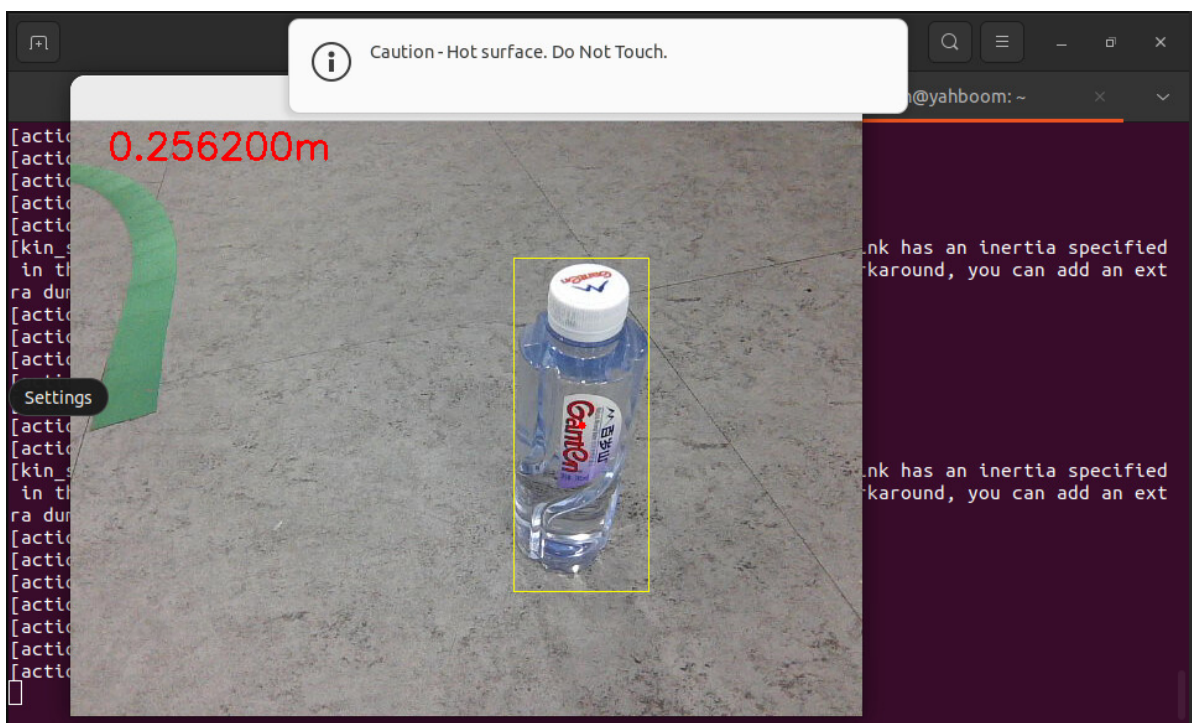
The decision-making model planning task steps are:

```
user input: 我现在在主卧室，我感觉有点口渴了  
okay😊, let me think for a moment... \[INFO] [1750496517.715551973] [text_chat_  
node]: robot response: The upcoming task to be carried out: 导航前往厨房区域，观  
察夹取矿泉水，导航到主卧室，放下矿泉水  
user input: \[INFO] [1750496520.302491350] [text_chat_node]: robot response: "act
```

Follow the task steps planned by the decision-making model in sequence:

```
user input: \[INFO] [1750496520.302491350] [text_chat_node]: robot response: "act  
ion": ['navigation(G)'], "response": 好的，我这就去厨房帮你拿矿泉水。就像一个忠  
诚的骑士一样，为你的需求而行动！  
\[INFO] [1750496539.481789151] [text_chat_node]: robot response: "action": ['seew  
hat()'], "response": 我已经到达厨房了，现在开始寻找矿泉水。让我用我的火眼金睛来  
定位它！  
\[INFO] [1750496542.712828693] [text_chat_node]: robot response: "action": ['gras  
p_obj(350,65,450,310)'], "response": 找到了！矿泉水就在那里，我马上把它夹取过来  
。就像魔术师一样，变~
```

When the robot arm is grasping, a visualization window will be displayed as shown below: Note: The preset opening and gripping size of the robot arm when it leaves the factory is to grip a 3 cm square. Here, we are gripping mineral water only for process demonstration. If you want to grip objects of other sizes, you need to modify the gripper opening angle yourself.



After arriving at the "master bedroom", the robot will use its robotic arm to put down the red block in its hand and prompt the user that the task is completed.

```
[INFO] [1750496576.744473877] [text_chat_node]: robot response: "action": ['navigation(A)'], "response": 矿泉水已经稳稳地在我手里了，现在我将它带回主卧室。就像一个快递小哥一样，使命必达！  
[INFO] [1750496595.137145707] [text_chat_node]: robot response: "action": ['putdown()'], "response": 我已经顺利到达主卧室，现在把矿泉水放在这里。请慢用，希望这能解你的渴！  
[INFO] [1750496605.264521960] [text_chat_node]: robot response: "action": ['finishtask()'], "response": 任务完成啦！如果你还需要什么帮助，随时告诉我哦。祝你有个美好的一天！
```

5. Source code analysis

The source code is located at:

jetson orin nano, jetson orin NX host:

```
/home/jetson/M3Pro_ws/src/largemodel/largemodel/src/largemodel/largemodel/action_service.py
```

Jetson Nano, Raspberry Pi host, you need to enter Docker first:

```
root/M3Pro_ws/src/largemodel/largemodel/src/largemodel/largemodel/action_service.py
```

action_service.py program:

This example uses **the seewhat , navigation , load_target_points , putdown , and grasp_obj** methods in **the CustomActionServer class**. **Seewhat , navigation, load_target_points , and grasp_obj have** been covered in the previous chapters [2. Multimodal Visual Understanding, 3. Multimodal Visual Understanding + Robotic Arm Grabbing, 4. Multimodal Visual Understanding + SLAM Navigation, and 5. Robotic Arm Grabbing + Multimodal Visual Understanding + SLAM Navigation]. No new procedures are used in this chapter.