Intention understanding and planning + robotic arm gripping + multimodal visual understanding + 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.

1. 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.

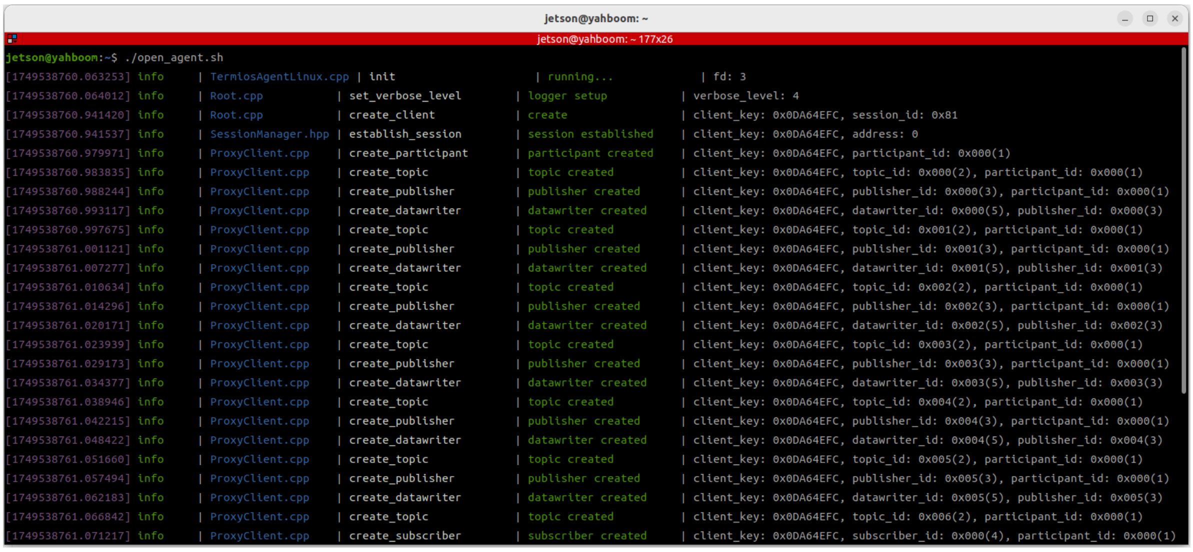
2.2 Start the Agent

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



3. Configuration Document

3.1 Create a txt document

Open the [Intent Mapping File] in the folder of this course

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

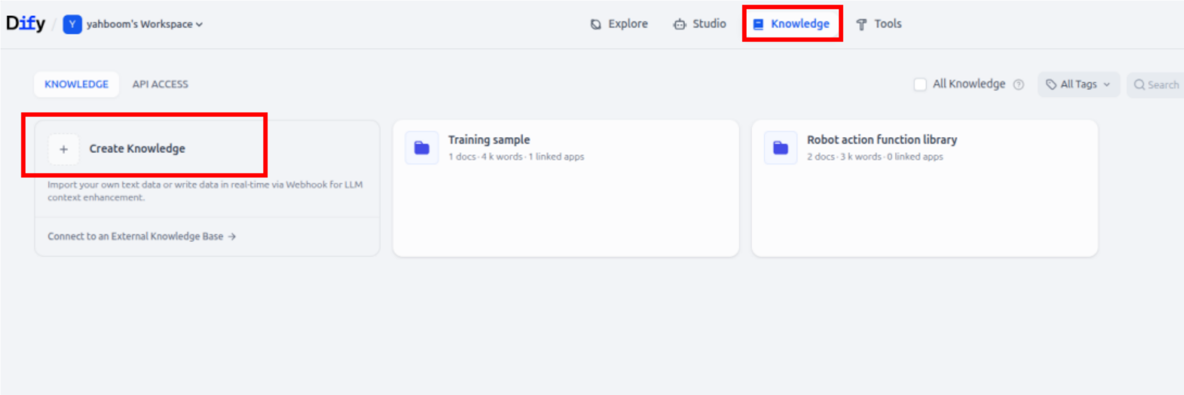
[Intent Mapping] : Stores personal intentions and expectations to control the robot to operate

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

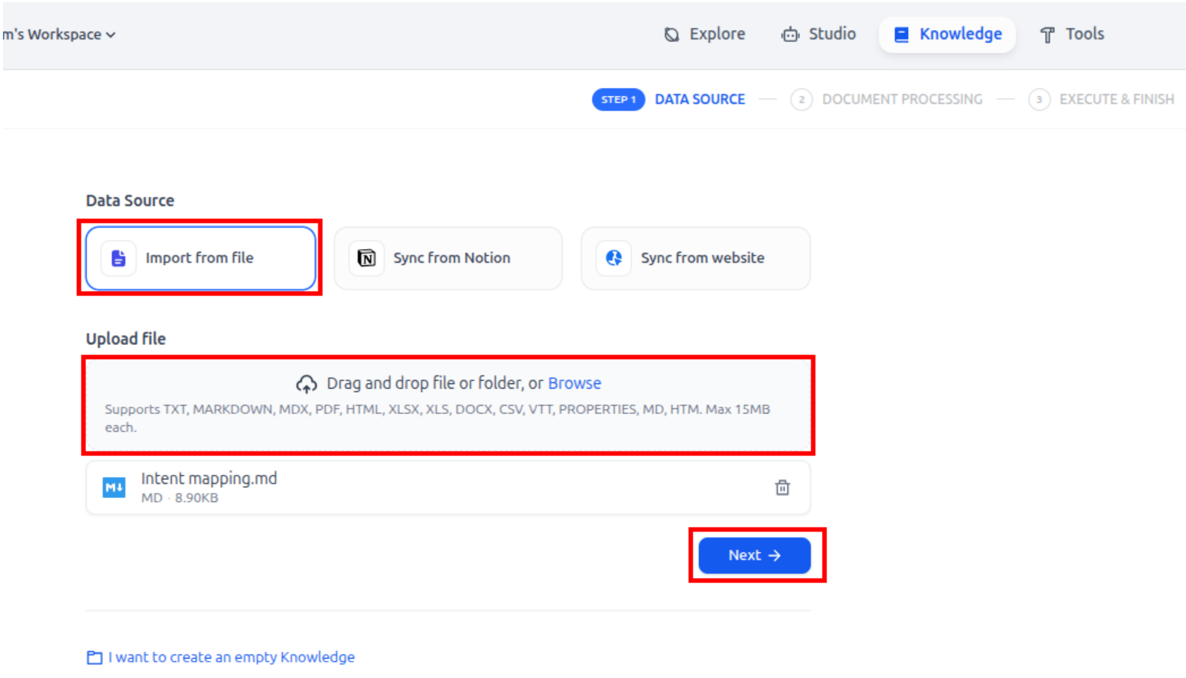
3.2 Upload to RAG Knowledge Base

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.

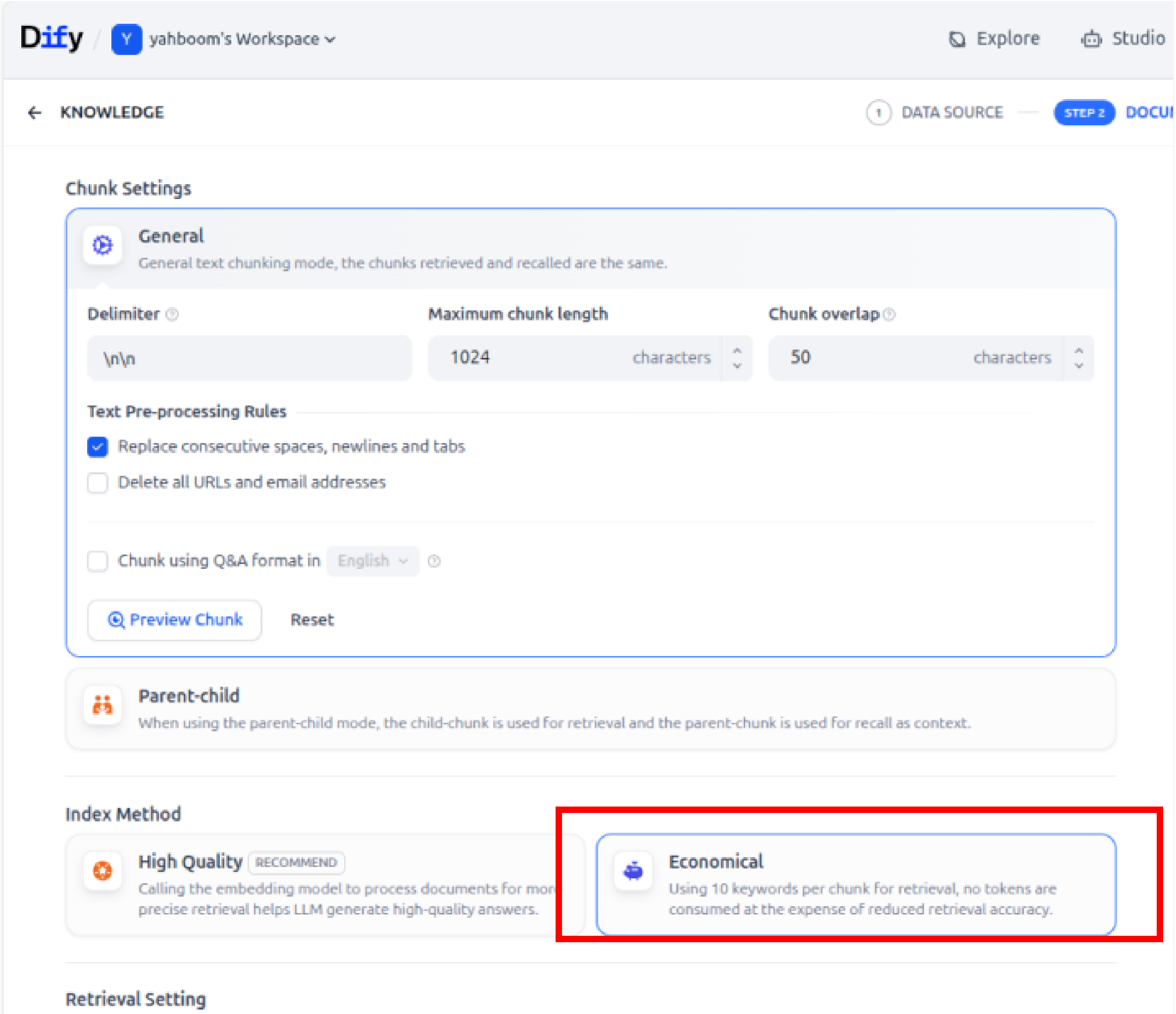
Click on the knowledge in the main interface of the diify interface, and then click Create knowledge to create a new knowledge base



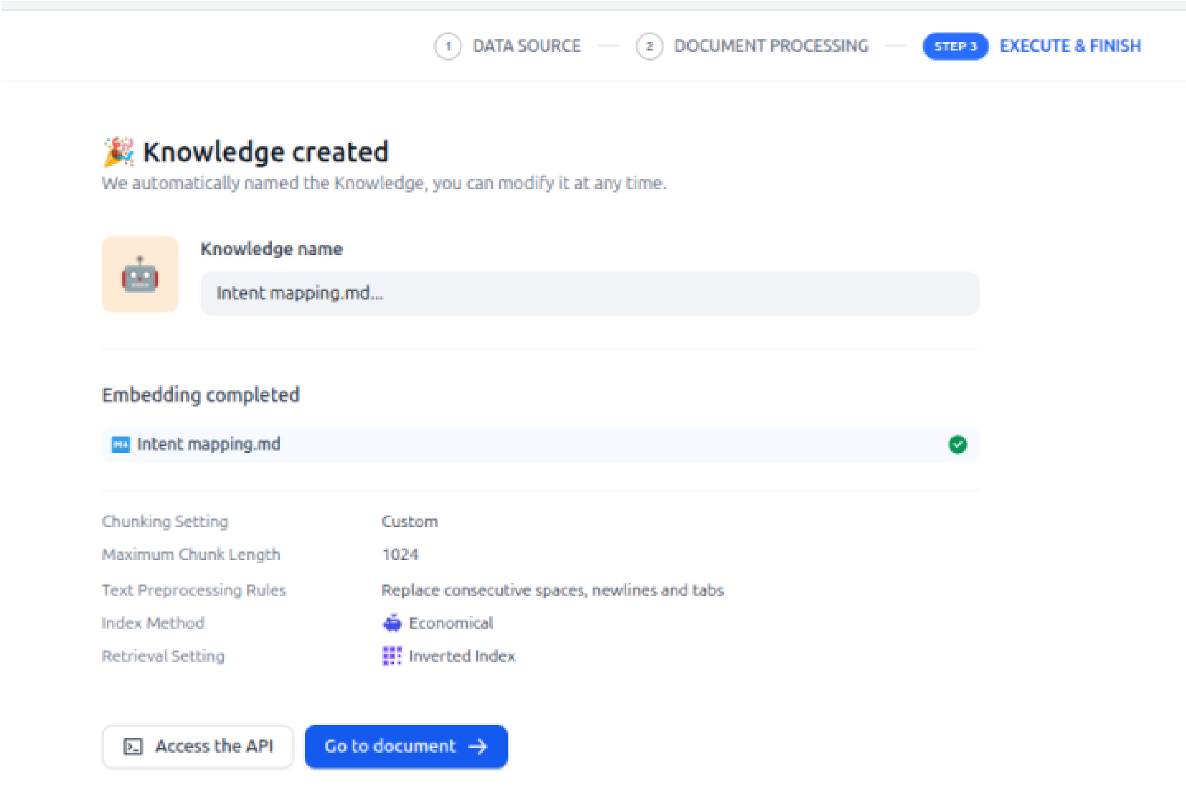
Select import from file, then click browser to browse to browse local files and upload, and click Next after uploading



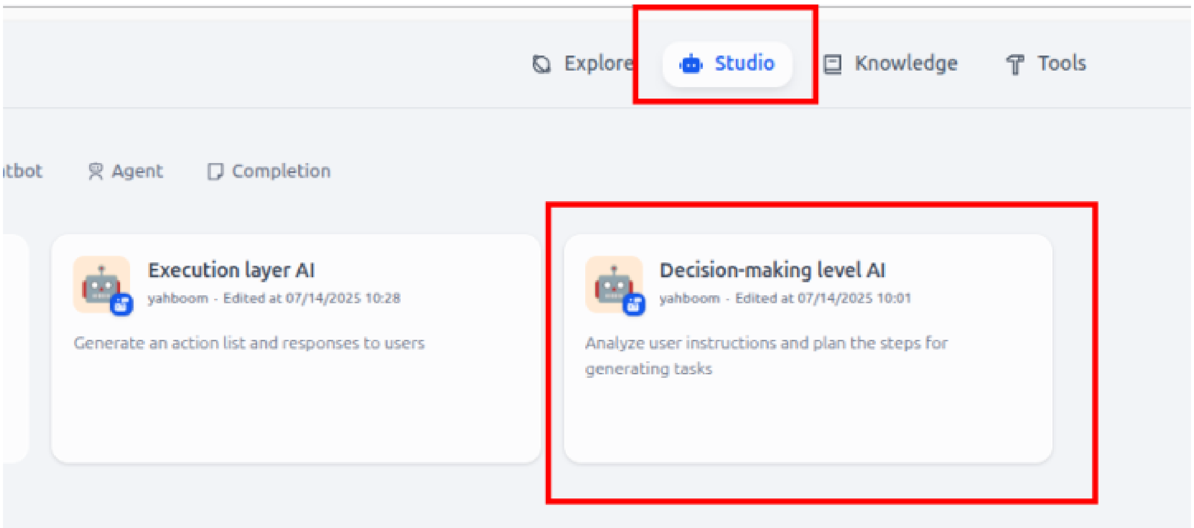
Select economic mode, keep the others by default



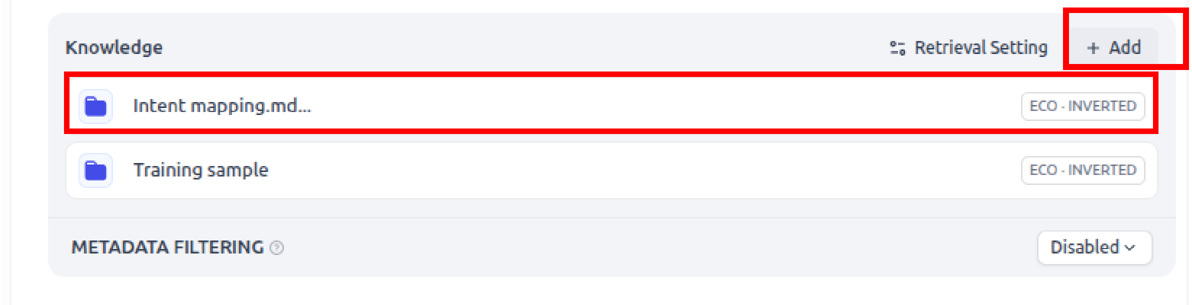
Wait for the document embedding to complete, and the intent mapping knowledge base creation can be completed



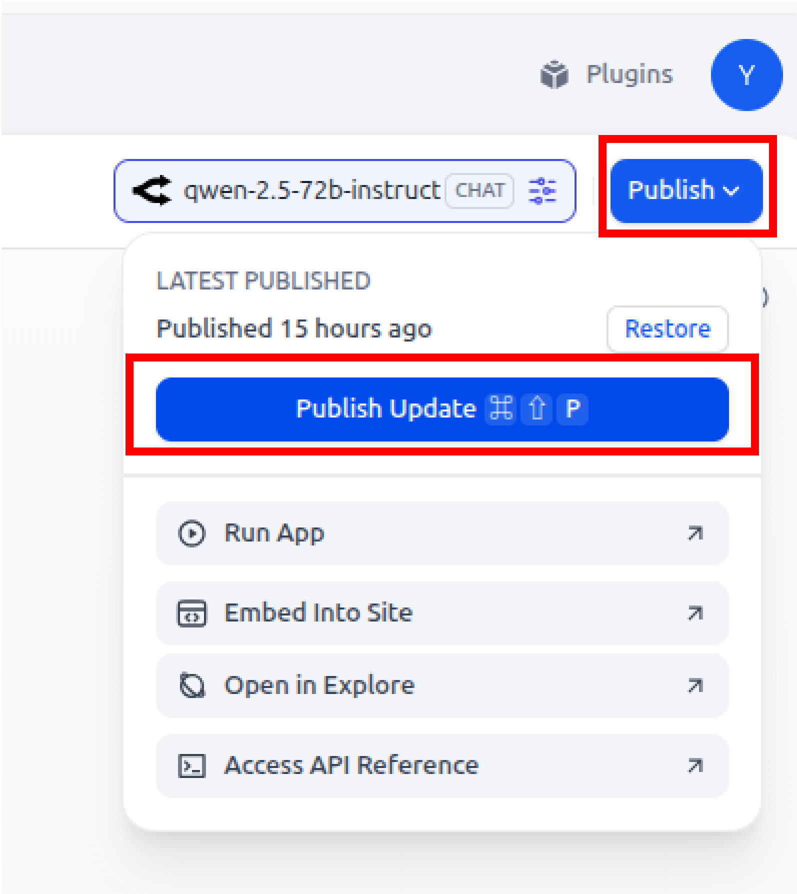
Switch back to the home page and click Configure the decision-making layer model



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



Click Publish to save the effective configuration



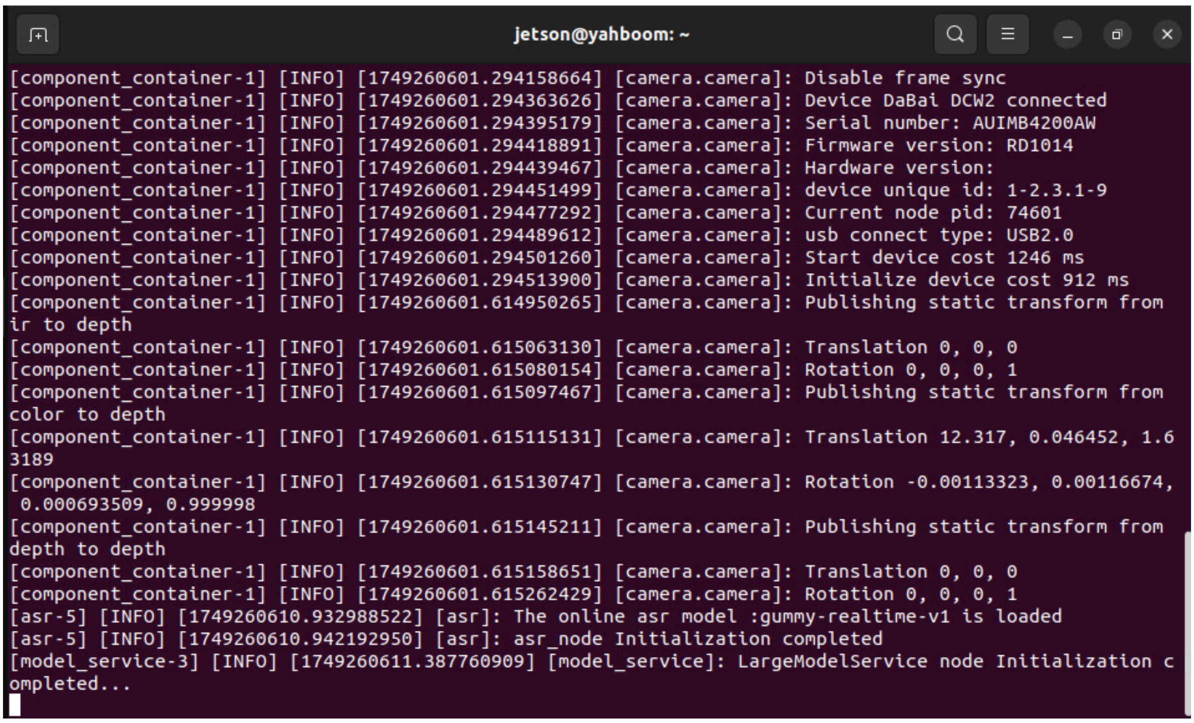
4. Run the case

4.1 Startup Program

Connect to the robot desktop via VNC, open a terminal, and start the command

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

After initialization is complete, the following content will be displayed



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
```

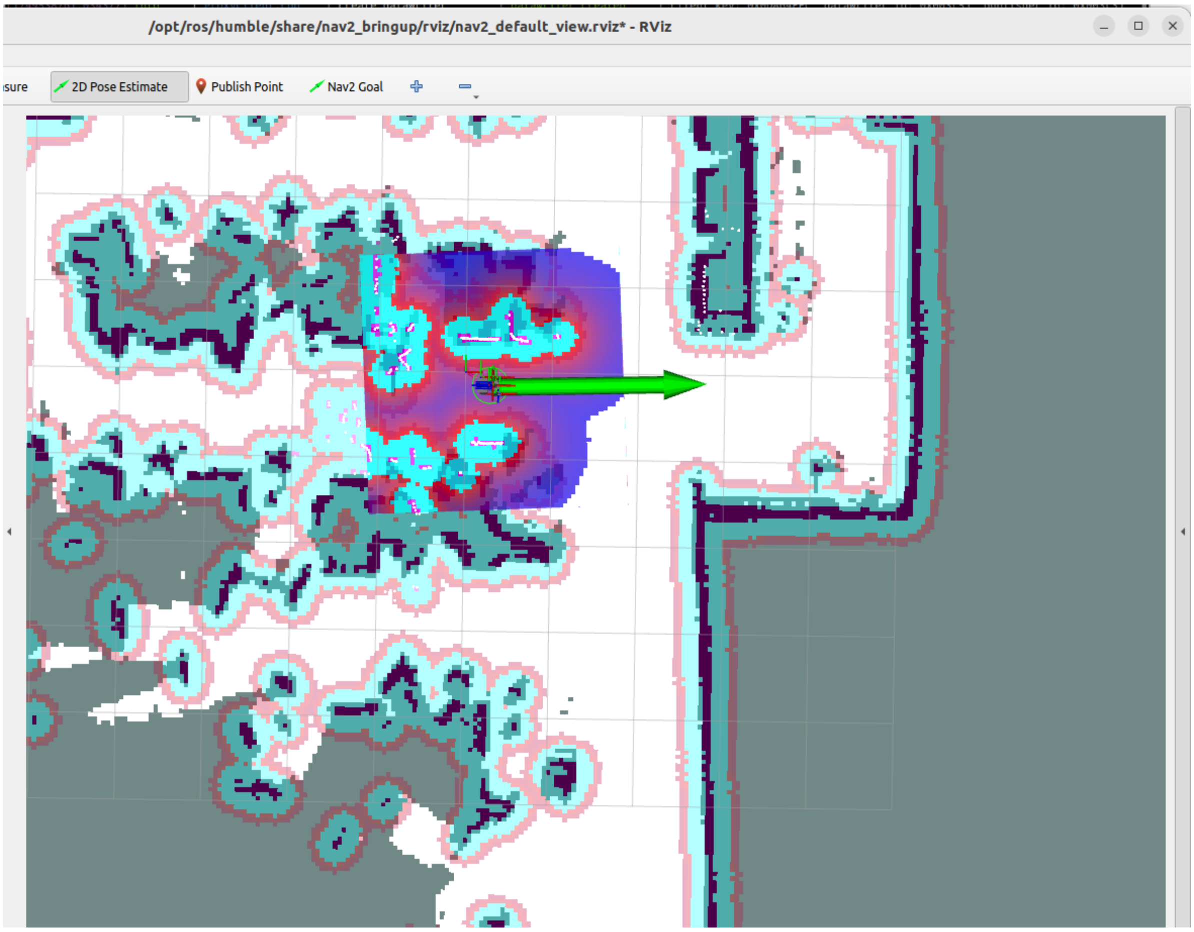
Create a new terminal on the virtual terminal to start the rviz visualization interface:

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

Or start rviz on the car:

```
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

First, use "Hi, yahboom" to wake up the robot. The robot responds: "I'm here, please tell me what to do." After the robot answers, the buzzer beeps briefly (beep—), and the user can speak. After the robot replies, the terminal prints the following information:

```
asr-5] [INFO] [1750491006.444004139] [asr]: 0
asr-5] [INFO] [1750491006.534965377] [asr]: 0
asr-5] [INFO] [1750491006.624816840] [asr]: 0
asr-5] [INFO] [1750491006.715541268] [asr]: 0
asr-5] [INFO] [1750491006.805879567] [asr]: 0
asr-5] [INFO] [1750491006.897671209] [asr]: 0
asr-5] [INFO] [1750491006.989015568] [asr]: 0
asr-5] [INFO] [1750491007.079612532] [asr]: 0
asr-5] [INFO] [1750491007.171301831] [asr]: 0
asr-5] [INFO] [1750491007.263230564] [asr]: 0
asr-5] [INFO] [1750491007.354806162] [asr]: 0
asr-5] [INFO] [1750491007.445503515] [asr]: 0
asr-5] [INFO] [1750491007.536006330] [asr]: 0
asr-5] serial /dev/ttyUSB1 open
asr-5] funasr version: 1.2.6.
'load_data': '0.030', 'extract_feat': '0.021', 'forward': '1.870', 'batch_size': '1', 'rtf': '0.405'},
tf_avg: 0.405: 100%|          | 1/1 [00:01<00:00,  1.87s/it]
tf_avg: 0.405: 100%|          | 1/1 [00:01<00:00,  1.87s/it]
asr-5] [INFO] [1750491009.522065243] [asr]: 我现在在主卧室我感觉我有点口渴了
asr-5] [INFO] [1750491009.522748441] [asr]: okay😀, let me think for a moment...
model_service-3] [INFO] [1750491011.870241207] [model_service]: The upcoming task to be carried out：1.
导航前往厨房
model_service-3] 2. 观察夹取矿泉水
model_service-3] 3. 导航返回主卧室
model_service-3] 4. 放下矿泉水
```

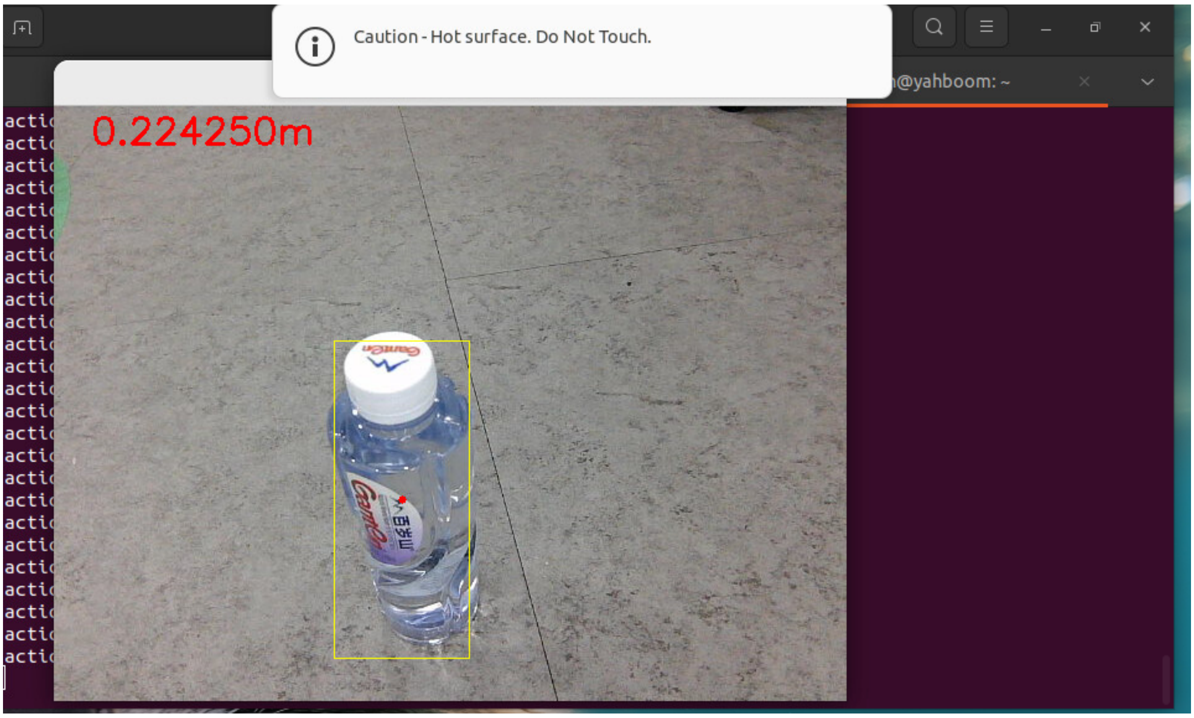
The decision-making model planning task steps are:

```
model_service-3] [INFO] [1750491011.870241207] [model_service]: The upcoming task to be carried out：1.
导航前往厨房
model_service-3] 2. 观察夹取矿泉水
model_service-3] 3. 导航返回主卧室
model_service-3] 4. 放下矿泉水
```

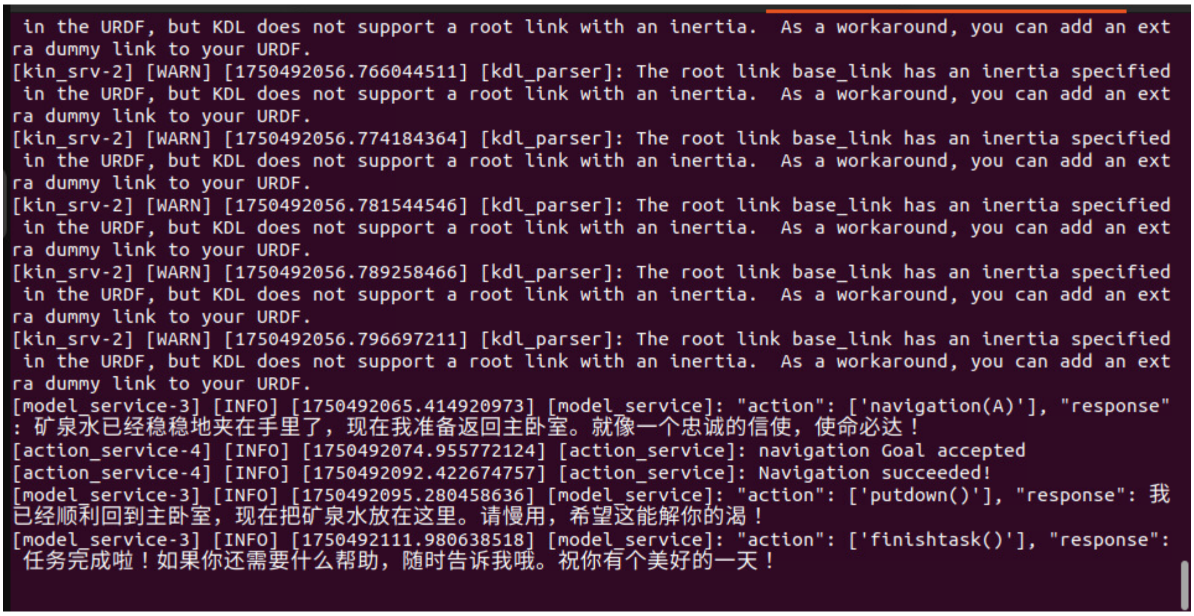
Follow the task steps planned by the decision-making model in sequence:

```
[model_service-3] [INFO] [1750491974.359941765] [model_service]: The upcoming task to be carried out：导
航前往厨房区域，观察夹取矿泉水，导航返回主卧室，放下矿泉水。
[model_service-3] [INFO] [1750491977.376557382] [model_service]: "action": ['navigation(G)'], "response"
: 好的，我这就去厨房帮你拿矿泉水。就像一个勇敢的骑士一样，出发！
[action_service-4] [INFO] [1750491985.052591699] [action_service]: navigation Goal accepted
[action_service-4] [INFO] [1750492007.324887601] [action_service]: Navigation succeeded!
[model_service-3] [INFO] [1750492010.560577885] [model_service]: "action": ['seewhat()'], "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看看它在哪里！
[model_service-3] [INFO] [1750492023.278365389] [model_service]: "action": ['grasp_obj(305, 48, 392, 276
)'], "response": 找到了！矿泉水就在那里，我这就去把它夹起来。看我的动作是不是很熟练？
[kin_srv-2] [WARN] [1750492032.926281990] [kdl_parser]: The root link base_link has an inertia specified
 in the URDF, but KDL does not support a root link with an inertia.  As a workaround, you can add an ext
ra dummy link to your URDF.
[action_service-4] start
[action_service-4] [INFO] [1750492036.744005351] [image_converter]: Received: [305, 48, 392, 276]
```

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.



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.