

Semantic understand and command follow(Text Version)

1. Course Content

1. Learn the basics of how the large model understands user instructions.
2. After running the large model program, users can interact with the robot through text. The text then generates a large model and uses multimodal visual processing to accurately understand user instructions and speech. Finally, the robot performs the specified actions according to the user's instructions and responds to the user.

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 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.

2.2. Entering the Car Docker

Enter the following command in the host terminal to enter Docker:

```
./run_humble.sh
```

```
pi@raspberrypi:~ $ ./run_humble.sh
access control disabled, clients can connect from any host
-----
ROS_DOMAIN_ID: 61 | ROS:
my_robot_type: A1 | my_lidar: tmini | my_camera: nuwa
-----
```

3. Running the Example

3.1 Launching the Program

For Raspberry Pi P15 and jetson nano, you need to enter the Docker container first. For RDKX5 and Orin main controllers, this is not necessary.

Open a terminal in Docker and enter the command:

```
ros2 launch largemode1 largemode1_control.launch.py text_chat_mode:=True
```

```
root@raspberrypi:/# ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
[INFO] [launch]: All log files can be found below /root/.ros/log/2025-08-18-14-441034-raspberrypi-165
[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 [196]
[INFO] [ascamera_node-1]: process started with pid [177]
[INFO] [joint_state_publisher-2]: process started with pid [179]
[INFO] [robot_state_publisher-3]: process started with pid [181]
[INFO] [Ackman_driver_A1-4]: process started with pid [183]
[INFO] [base_node_A1-5]: process started with pid [185]
[INFO] [imu_filter_madgwick_node-6]: process started with pid [187]
[INFO] [ekf_node-7]: process started with pid [189]
[INFO] [yahboom_joy_A1-8]: process started with pid [192]
[INFO] [joy_node-9]: process started with pid [194]
[INFO] [static_transform_publisher-11]: process started with pid [198]
[INFO] [static_transform_publisher-12]: process started with pid [201]
[INFO] [static_transform_publisher-13]: process started with pid [203]
[INFO] [model_service-14]: process started with pid [205]
[INFO] [action_service_nuwa-15]: process started with pid [216]
[ydlidar_ros2_driver_node-10] [INFO] [1755505216.190453432] [ydlidar_ros2_driver_node]: [YDLIDAR INFO] Current ROS Driver Version: 1.0.1
[ydlidar_ros2_driver_node-10]
[ydlidar_ros2_driver_node-10] [YDLIDAR] SDK initializing
[ydlidar_ros2_driver_node-10] [YDLIDAR] SDK has been initialized
[ydlidar_ros2_driver_node-10] [YDLIDAR] SDK Version: 1.2.3
[static_transform_publisher-12] [WARN] [1755505216.206689783] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-11] [WARN] [1755505216.206729229] []: Old-style arguments are deprecated; see --help for new-style arguments
```

Open another terminal and enter the same Docker container terminal. For detailed steps, see [\[3. Docker Submission and Multi-Terminal Access\]](#) in the Docker course.

```
ros2 run text_chat text_chat
```

Subsequently, enter conversation commands in this terminal.

```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: |
```

3.2 Test Case

Here are some reference test cases; users are welcome to create their own conversation commands.

- Tell me a joke about a kitten and a puppy.
- Please quickly move forward 1 meter, then slowly back 0.5 meters like a turtle, then turn 30 degrees left and 90 degrees right.

3.2.1 Example 1

Open a terminal on the virtual machine and enter a test case. After the model has thought about it, it will respond to the user and execute the action according to the user's instructions.

```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 请你给我讲个关于小猫和小狗的笑话
okay@, let me think for a moment... -[INFO] [1755506486.233778089] [text_chat_node]: 决策层AI规划:1. 给用户讲一个关于小猫和小狗的笑话
user input: [INFO] [1755506490.206578074] [text_chat_node]: "action": [], "response": 好的，这里有一个关于小猫和小狗的笑话：小猫和小狗一起去上学。小狗问小猫：“你会游泳吗？”小猫回答：“不会。”小狗又问：“那你会爬树吗？”小猫说：“当然会啦！”小狗想了想，说：“那我们还是走楼梯吧，这样你就都能上去啦！”
[INFO] [1755506492.624140145] [text_chat_node]: "action": ['finishtask()'], "response": 希望你喜欢这个笑话，有需要再找我哦！
|
```

3.2.2 Example 2

Similar to Example 1, enter Example 2 in the terminal. The model will respond and execute the action according to the instruction.

```
[INFO] [1755506492.624140145] [text_chat_node]: "action": ['finishtask()'], "response": 希望你喜欢这个笑话，有需要再找我哦！
请你快速前进一米，然后像乌龟一样后退0.5米，然后左转30度，再右转90度
okay@, let me think for a moment... -[INFO] [1755506492.815363491] [text_chat_node]: "action": ['set_cmdvel(0.5, 0, 0, 2)', 'set_cmdvel(-0.2, 0, 0, 2.5)', 'move_left(30, 1.5)', 'move_right(90, 1.5)'], "response": 好的，我这就快速前进一米，然后像乌龟一样慢慢后退0.5米，接着左转30度，再右转90度。看我的表演吧！
user input: [INFO] [1755506681.570683395] [text_chat_node]: "action": ['finishtask()'], "response": 我已经完成了所有的动作，是不是很炫呢？如果你还有其他任务需要我帮忙，随时告诉我哦！
| 1.5)'完成
Please quickly move forward 1 meter, then slowly back 0.5 meters like a turtle, then turn 30 degrees left and 90 degrees right.
```

