APP mapping control

Note: The ROS_DOMAIN_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check [MicroROS Control Board Parameter Configuration] to set the microROS control board ROS_DOMAIN_ID. Check the tutorial [Connect MicroROS Agent] to determine whether the IDs are consistent.

1. Program function description

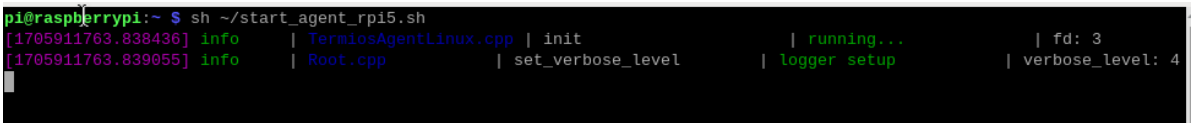
The car connects to the agent, runs the program, opens the [ROS Robot] app downloaded on the mobile phone, enters the IP address of the car, selects ROS2, and clicks connect to connect to the car. You can control the car by sliding the wheel on the interface, and slowly control the car to complete the mapped area. Finally, click Save Map, and the car will save the currently constructed map.

2. Query car information

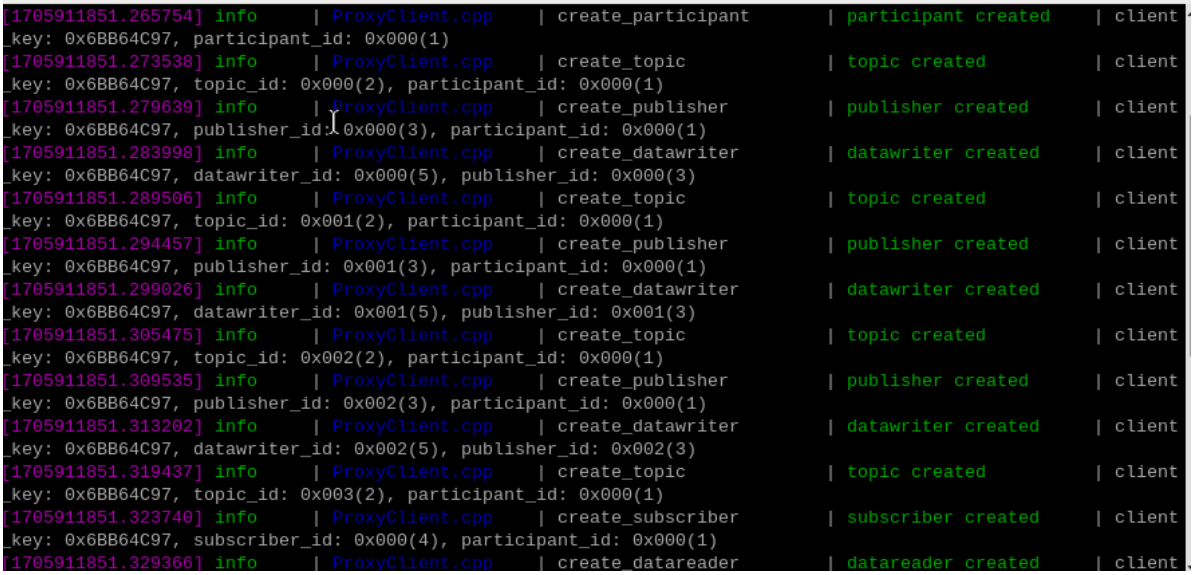
2.1. Start and connect to the agent

After the Raspberry Pi is successfully powered on, open the terminal and enter the following command to open the agent:

```
sh ~/start_agent_rpi5.sh
```



Press the reset button on the microROS control board and wait for the car to connect to the agent. The connection is successful as shown in the figure below.

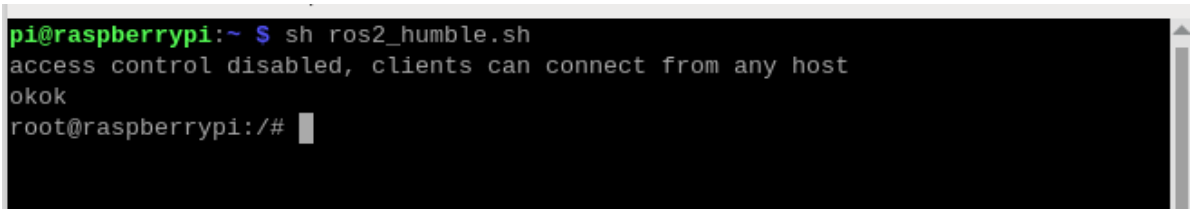


2.2. Enter the car docker

Open another terminal and enter the following command to enter docker:

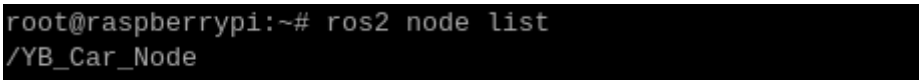
```
sh ros2_humble.sh
```

When the following interface appears, you have successfully entered docker. Now you can control the car through commands.



Enter the following command in the terminal to query the agent node:

```
ros2 node list
```



3. Start the program

First, start the car to process the underlying data program and enter the terminal.

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

```
[INFO] [launch]: Default logging verbosity is set to INFO
----------------------robot_type = x3----------------------
[INFO] [imu_filter_madgwick_node-1]: process started with pid [288]
[INFO] [ekf_node-2]: process started with pid [290]
[INFO] [static_transform_publisher-3]: process started with pid [292]
[INFO] [joint_state_publisher-4]: process started with pid [294]
[INFO] [robot_state_publisher-5]: process started with pid [296]
[INFO] [static_transform_publisher-6]: process started with pid [298]
[static_transform_publisher-6] [WARN] [1706178669.534357284] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1706178669.534366951] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-6] [INFO] [1706178669.608181313] [static_transform_publisher_Y9Btl5eyTEMQpG8K]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.050000')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[static_transform_publisher-3] [INFO] [1706178669.611159254] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('-0.002999', '-0.003000', '0.031701')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'imu_frame'
[robot_state_publisher-5] [WARN] [1706178669.671482195] [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 extra dummy link to your URDF.
[robot_state_publisher-5] [INFO] [1706178669.671695622] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1706178669.671803511] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1706178669.671818696] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1706178669.671830308] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1706178669.671845789] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1706178669.671862623] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1706178669.671877123] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1706178669.671892493] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1706178669.671902827] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1706178669.782244395] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1706178669.783298605] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1706178669.783778570] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1706178669.784159016] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1706178669.784197405] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1706178669.784215702] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[imu_filter_madgwick_node-1] [INFO] [1706178669.853536707] [imu_filter]: First IMU message received.
```

Start the APP mapping command and enter it in the terminal,

```
ros2 launch yahboomcar_nav map_gmapping_app_launch.xml
```

```
[slam_gmapping-4] Laser Pose= 0.00362656 0.00358431 -2.34657
[slam_gmapping-4] update frame 19
[slam_gmapping-4] update ld=0.000324247 ad=0.00966722
[slam_gmapping-4] m_count 1
[slam_gmapping-4] Laser Pose= 0.00385447 0.00362642 -2.33696
[slam_gmapping-4] Average Scan Matching Score=317.913
[slam_gmapping-4] neff= 29.9871
[slam_gmapping-4] Registering Scans:Done
[slam_gmapping-4] update frame 44
[slam_gmapping-4] update ld=1.81496e-05 ad=0.00407955
[slam_gmapping-4] m_count 2
[slam_gmapping-4] Laser Pose= 0.00384172 0.00363931 -2.33288
[slam_gmapping-4] Average Scan Matching Score=316.264
[slam_gmapping-4] neff= 29.9712
[slam_gmapping-4] Registering Scans:Done
[slam_gmapping-4] update frame 69
[slam_gmapping-4] update ld=2.15485e-05 ad=0.00465577
[slam_gmapping-4] m_count 3
[slam_gmapping-4] Laser Pose= 0.00382612 0.00365418 -2.32822
[slam_gmapping-4] Average Scan Matching Score=320.412
[slam_gmapping-4] neff= 29.9622
[slam_gmapping-4] Registering Scans:Done
[slam_gmapping-4] update frame 94
[slam_gmapping-4] update ld=2.97205e-05 ad=0.00640465
```

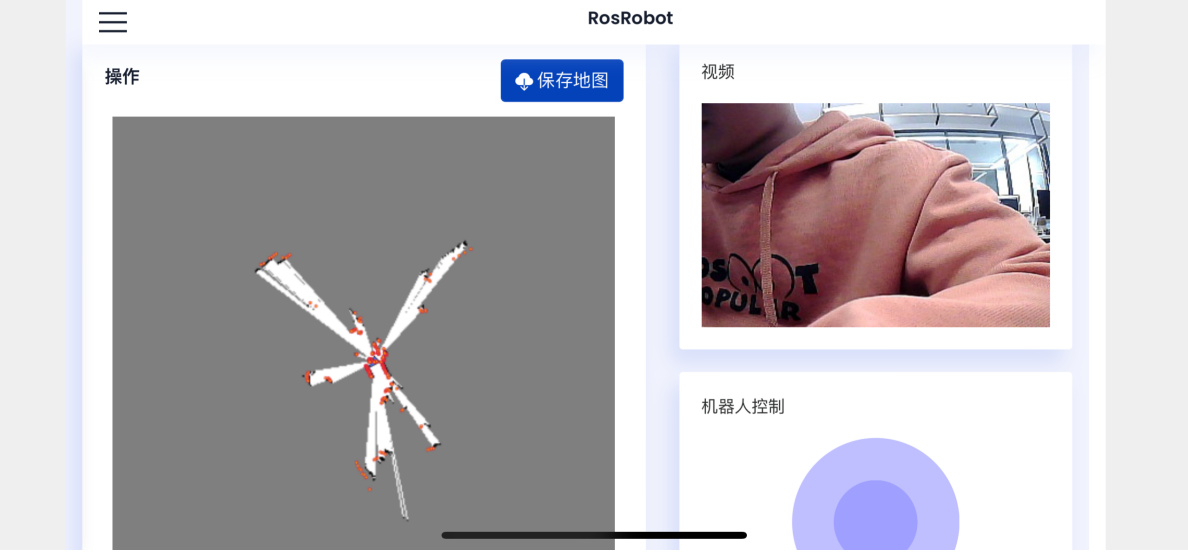
Open another terminal and enter the command to start the camera screen.

```
ros2 launch usb_cam camera.launch.py
```

```
root@raspberrypi:~# ros2 launch usb_cam camera.launch.py
[INFO] [launch]: All log files can be found below /root/.ros/log/2024-01-25-10-33-16-478075-raspberrypi-558
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [usb_cam_node_exe-1]: process started with pid [559]
[usb_cam_node_exe-1] [INFO] [1706178798.300109580] [camera1]: camera_name value: test_camera
[usb_cam_node_exe-1] [WARN] [1706178798.301320740] [camera1]: framerate: 30.000000
[usb_cam_node_exe-1] [INFO] [1706178798.331489959] [camera1]: camera calibration URL: package://usb_cam/config/camera_info.yaml
[usb_cam_node_exe-1] [INFO] [1706178798.333964944] [camera1]: Starting 'test_camera' (/dev/video0) at 320x240 via mmap (mjpeg2rgb) at 30 FPS
[usb_cam_node_exe-1] [swscaler @ 0x55564a91f270] 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 1920 x 1080 (30 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 640 x 480 (120 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 640 x 480 (90 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 640 x 480 (60 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 640 x 480 (30 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 1280 x 720 (60 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 1280 x 720 (30 Hz)
[usb_cam_node_exe-1]     Motion-JPEG 1024 x 768 (30 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 1920 x 1080 (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 1280 x 720 (9 Hz)
[usb_cam_node_exe-1]     YUYV 4:2:2 1024 x 768 (6 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)
[usb_cam_node_exe-1] unknown control 'white_balance_temperature_auto'
[usb_cam_node_exe-1]
[usb_cam_node_exe-1] [INFO] [1706178798.499334881] [camera1]: Setting 'white_balance_temperature_auto' to 1
[usb_cam_node_exe-1] [INFO] [1706178798.499433620] [camera1]: Setting 'exposure_auto' to 3
[usb_cam_node_exe-1] unknown control 'exposure_auto'
[usb_cam_node_exe-1]
[usb_cam_node_exe-1] [INFO] [1706178798.505503067] [camera1]: Setting 'focus_auto' to 0
[usb_cam_node_exe-1] unknown control 'focus_auto'
[usb_cam_node_exe-1]
[usb_cam_node_exe-1] [INFO] [1706178798.648586676] [camera1]: Timer triggering every 33 ms
```

The mobile APP displays as shown below, enter the IP address of the car, **[zh]** means Chinese, **[en]** means English; select ROS2, select /usb_cam/image_raw/compressed below, and finally click **[Connect]**

After successful connection, the following is displayed.



Use the sliding wheel to control the car to move slowly through the area that needs to be mapped, then click to save the map, enter the map name and click submit to save the map.

The location path where the map is saved is as follows

```
/root/yahboomcar_ws/src/yahboomcar_nav/maps
```

This tutorial is mainly used to use the mobile phone APP to control the car, while observing the camera screen and testing the point cloud data of the radar. After the map building is completed, you can refer to the tutorial on mobile APP navigation later.