ROS Robot App mapping

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.
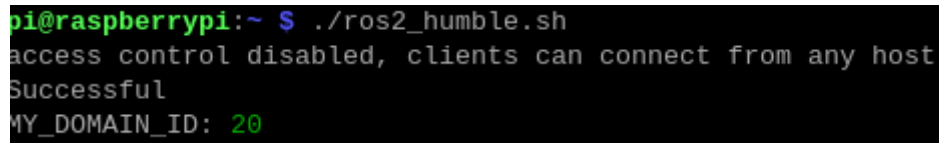
2.2、 Enter the car docker

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.

3、 starting program

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

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

```
[INFO] [imu_filter_madgwick_node-1]: process started with pid [6263]
[INFO] [ekf_node-2]: process started with pid [6265]
[INFO] [static_transform_publisher-3]: process started with pid [6267]
[INFO] [joint_state_publisher-4]: process started with pid [6269]
[INFO] [robot_state_publisher-5]: process started with pid [6271]
[INFO] [static_transform_publisher-6]: process started with pid [6286]
[static_transform_publisher-3] [WARN] [1706181650.342105372] []: Old-style arguments are depreca
ted; see --help for new-style arguments
[static_transform_publisher-3] [INFO] [1706181650.459314055] [base_link_to_base_imu]: Spinning u
ntil 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'
[imu_filter_madgwick_node-1] [INFO] [1706181650.478942143] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1706181650.480114862] [imu_filter]: Using dt computed from
message headers
[imu_filter_madgwick_node-1] [INFO] [1706181650.480197121] [imu_filter]: The gravity vector is k
ept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1706181650.480631749] [imu_filter]: Imu filter gain set to
0.100000
[imu_filter_madgwick_node-1] [INFO] [1706181650.480707508] [imu_filter]: Gyro drift bias set to
0.000000
[imu_filter_madgwick_node-1] [INFO] [1706181650.480720249] [imu_filter]: Magnetometer bias value
s: 0.000000 0.000000 0.000000
[static_transform_publisher-6] [WARN] [1706181650.493533858] []: Old-style arguments are depreca
ted; see --help for new-style arguments
[robot_state_publisher-5] [WARN] [1706181650.639387954] [kdl_parser]: The root link base_link ha
s 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] [1706181650.640061915] [robot_state_publisher]: got segment bas
e_link
[robot_state_publisher-5] [INFO] [1706181650.640174267] [robot_state_publisher]: got segment imu
_Link
[robot_state_publisher-5] [INFO] [1706181650.640191229] [robot_state_publisher]: got segment jq1
_Link
[robot_state_publisher-5] [INFO] [1706181650.640201822] [robot_state_publisher]: got segment jq2
_Link
[robot_state_publisher-5] [INFO] [1706181650.640211952] [robot_state_publisher]: got segment rad
ar_Link
[robot_state_publisher-5] [INFO] [1706181650.640221211] [robot_state_publisher]: got segment yh_
Link
[robot_state_publisher-5] [INFO] [1706181650.640229452] [robot_state_publisher]: got segment yq_
Link
[robot_state_publisher-5] [INFO] [1706181650.640238470] [robot_state_publisher]: got segment zh_
Link
[robot_state_publisher-5] [INFO] [1706181650.640246655] [robot_state_publisher]: got segment zq_
Link
[imu_filter_madgwick_node-1] [INFO] [1706181650.655098332] [imu_filter]: First IMU message recei
ved.
[static_transform_publisher-6] [INFO] [1706181650.681177050] [static_transform_publisher_JarNTEa
1OrW2k0Zb]: 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'
[joint_state_publisher-4] [INFO] [1706181650.989117137] [joint_state_publisher]: Waiting for rob
ot_description to be published on the robot_description topic...
```

Start the APP mapping command and enter the following command in the terminal

```
#Choose one of the following drawings
ros2 launch yahboomcar_nav map_gmapping_app_launch.xml
ros2 launch yahboomcar_nav map_cartographer_app_launch.xml
```

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-rasp
berrypi-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/video
0) 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 the first Video Tpoic below, and finally click [Connect]

After successful connection, the following is displayed:
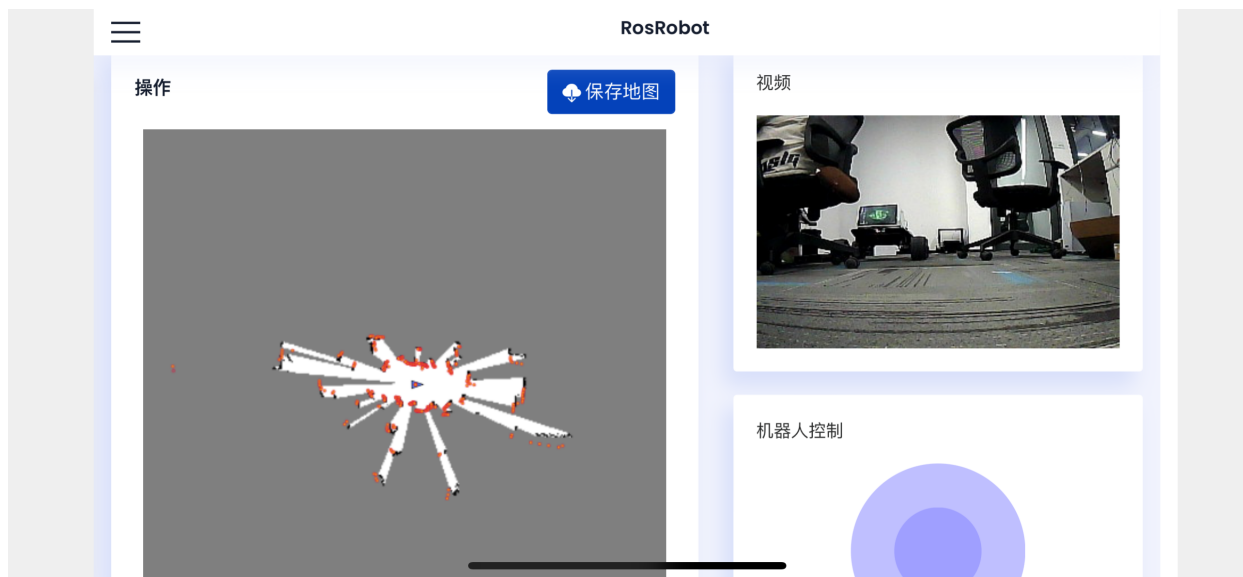
Use the sliding wheel to control the car to move slowly through the area that needs to be mapped, then click to save the map, enter the map name and click submit to save the map.

The location where the map is saved is as follows.

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

4、Code analysis

Here is an explanation of the launch file that opens the APP mapping, taking gmapping mapping as an example.

map_gmapping_app_launch.xml

```
<launch>
    <include file="$(find-pkg-share
rosbridge_server)/launch/rosbridge_websocket_launch.xml"/>
    <node name="laserscan_to_point_publisher" pkg="laserscan_to_point_publisher"
exec="laserscan_to_point_publisher"/>
    <include file="$(find-pkg-share yahboomcar_nav)/launch/map_gmapping_launch.py"/>
    <include file="$(find-pkg-share
robot_pose_publisher_ros2)/launch/robot_pose_publisher_launch.py"/>
    <include file="$(find-pkg-share
yahboom_app_save_map)/yahboom_app_save_map.launch.py"/>
</launch>
```

The following launch files and nodes are run here:

- rosbridge_websocket_launch.xml：Open the rosbridge service-related nodes. After starting, you can connect to ROS through the network.

- laserscan_to_point_publisher：Publish the radar point cloud conversion to the APP for visualization

- map_gmapping_launch.py：gmapping mapping program

- robot_pose_publisher_launch.py：Car pose publishing program, the car pose is visualized in the APP

- yahboom_app_save_map.launch.py：Program to save maps.