ROS Robot APP navigation

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, and the mobile phone is connected to the car through a network. Open the [ROS Robot] app downloaded on your mobile phone, enter the IP address of the car, select ROS2, and click Connect to connect to the car. Select [Navigation], click [Set Initialization Point] on the App interface to set the starting pose of the car, click [Set Navigation Point] on the App interface, and set the target point of the car, and then the car will plan a path to move to that point.

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.
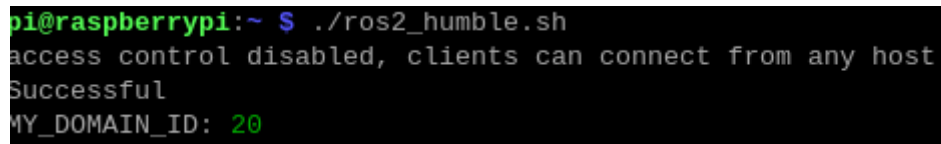
2.2、 Enter the car docker

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, and you can now control the car through commands.

3、 starting program

First, enter the following command in the terminal to start the car to process the underlying data program.

```
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
10rW2k0Zb]: 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...
```

Enter the following command in the terminal to start the APP navigation command.

```
ros2 launch yahboomcar_nav navigation_dwb_app_launch.xml
maps:=/root/yahboomcar_ws/src/yahboomcar_nav/maps/test1219.yaml
```

Load map parameters: maps:=/root/yahboomcar_ws/src/yahboomcar_nav/maps/test1219.yaml
 (Replaceable target map)

Enter the following command in the terminal to start the camera screen.
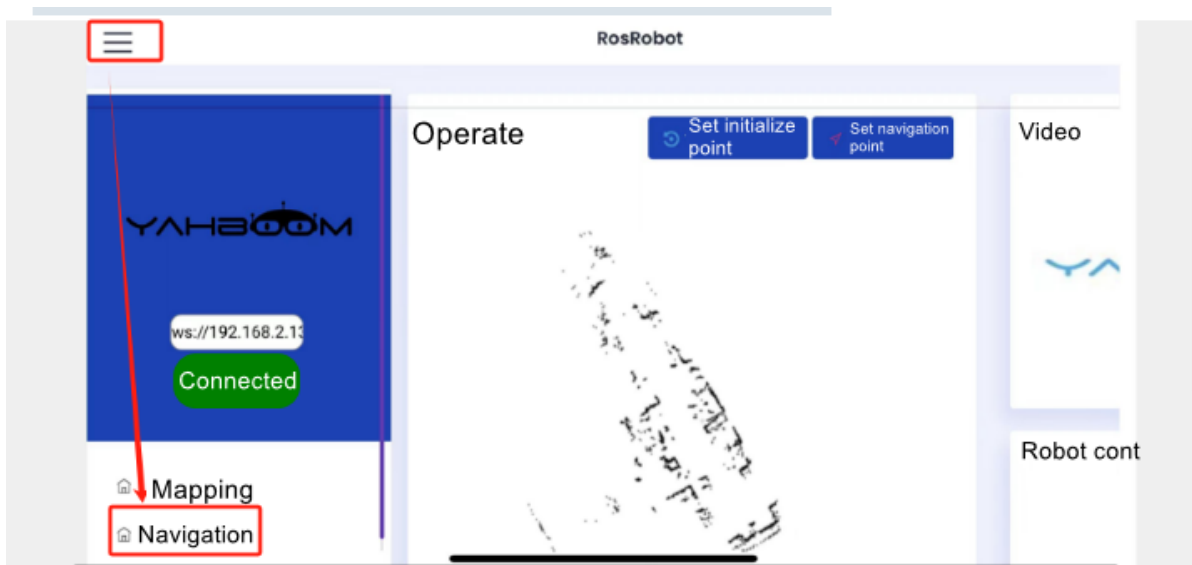
```
ros2 launch usb_cam camera.launch.py
```

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]
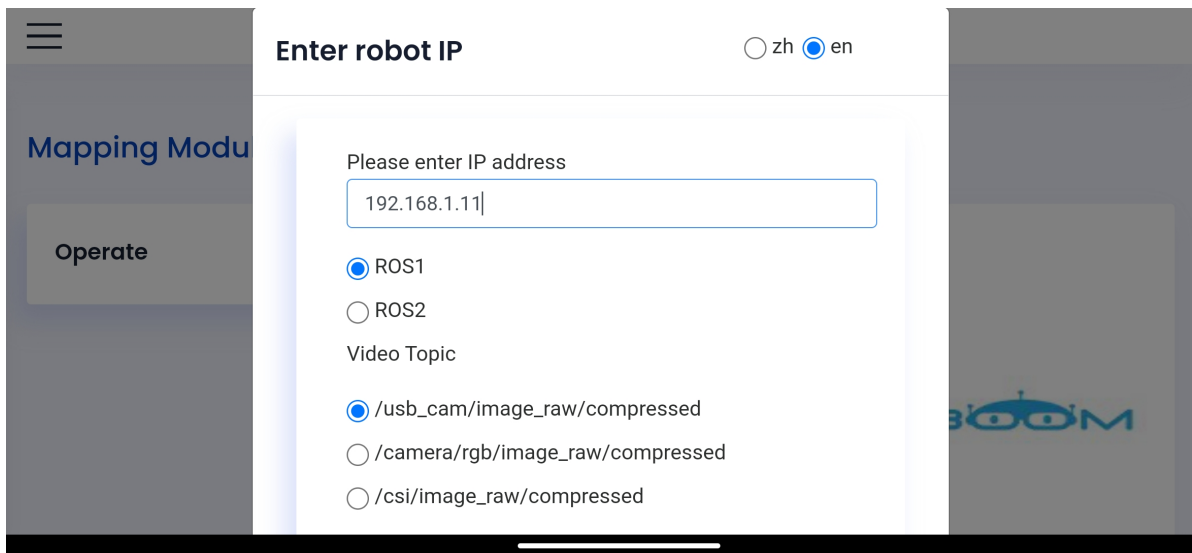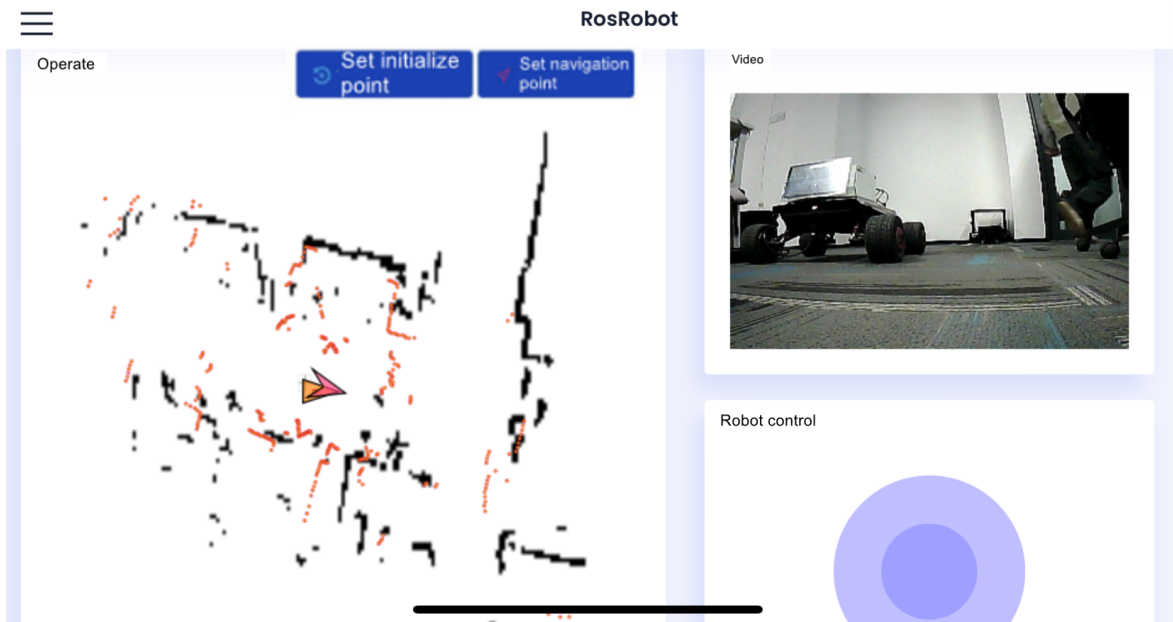
After successful connection, the following screen will appear.
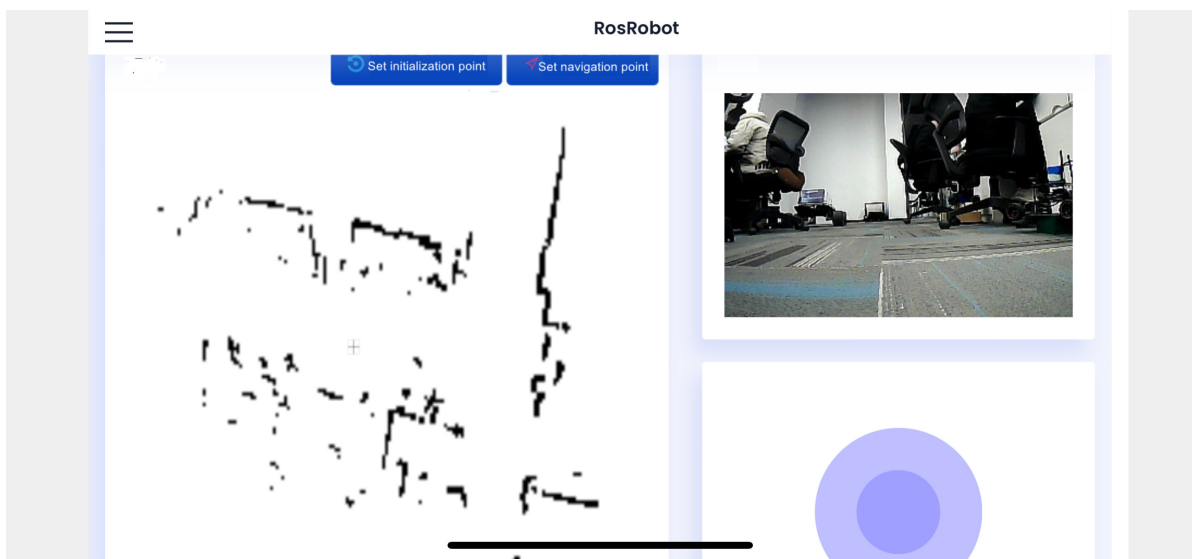
As shown in the figure below, select the navigation interface.

Then, combined with the actual pose of the car, click [Set Initialization Point] to give the car an initial target point. If the area scanned by the radar roughly coincides with the actual obstacle, it means the pose is accurate. As shown below,

Then, click [Set Navigation Point] and give the car a destination. The car will plan a path and follow the path to the destination.
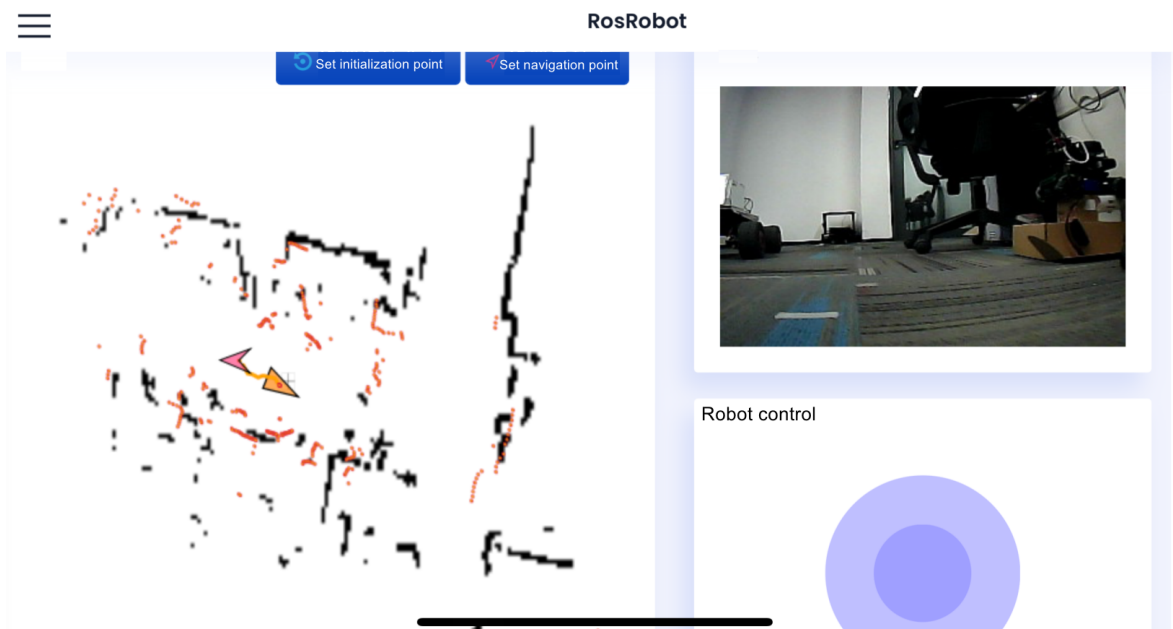
4、 Code analysis

Here is the description of the launch file that opens the APP navigation.

navigation_dwb_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/navigation_dwb_launch.py"/>
    <include file="$(find-pkg-share
robot_pose_publisher_ros2)/launch/robot_pose_publisher_launch.py"/>
</launch>
```

The following are several launch files and Nodes：

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