ROS Robot APP Navigation

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be the same. You can check [Must-Read Before Use] to set the IP and ROS_DOMAIN_ID on the board.

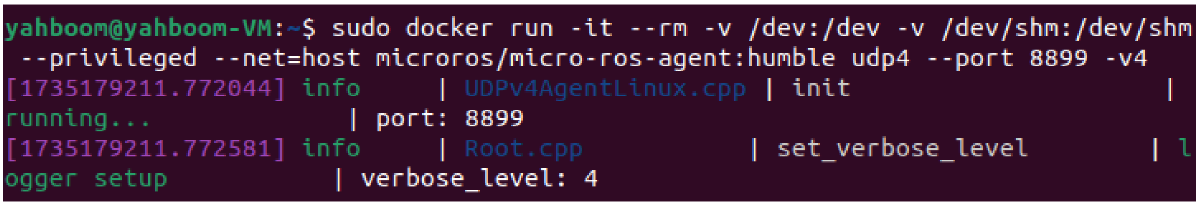
1. Program Function Description

The car connects to the proxy, runs the program, and the mobile phone and the car are connected through a network. Open the [ROS Robot] app downloaded on the phone, enter the IP address of the car, select ROS2, click Connect, and you can connect to the car. Select [Navigation], click [Set Initialization Point] on the App interface to set the car's starting position, click [Set Navigation Point] on the App interface, give the car a target point, and then the car will plan a path to move to that point.

2. Start and connect the agent

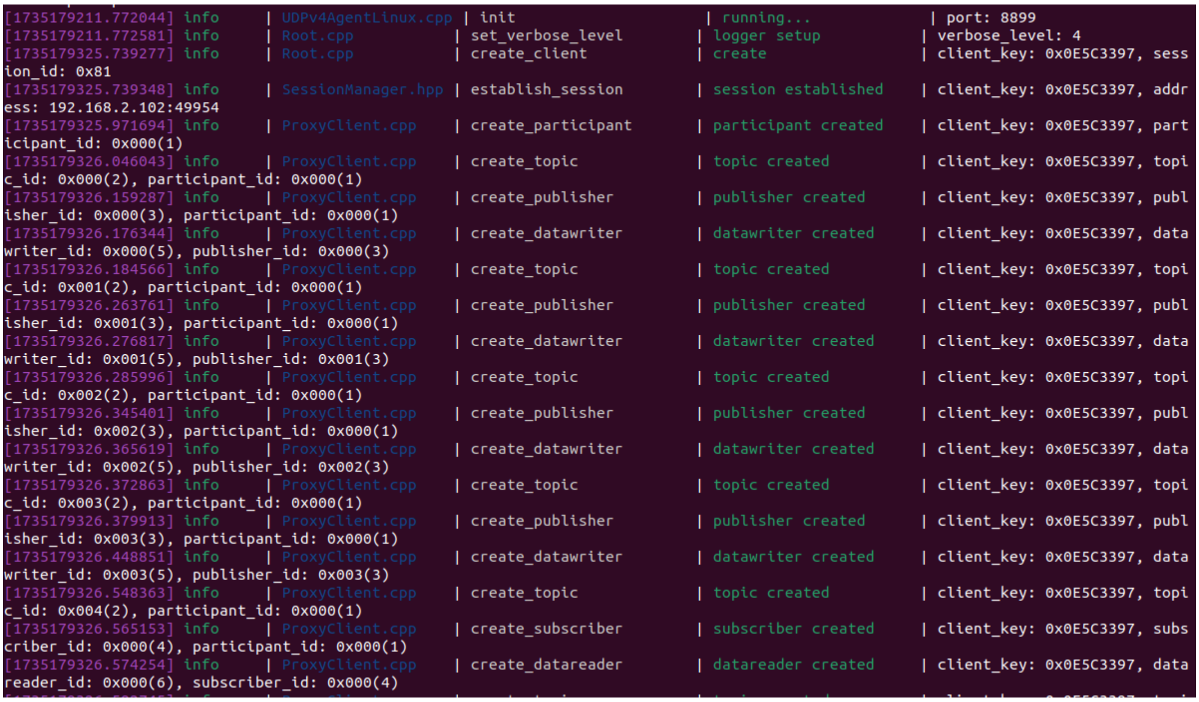
Take the supporting virtual machine as an example, enter the following command to start the agent,

```
#Car agent
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
#Camera agent (start the agent first and then turn on the car switch)
docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host microros/micro-ros-agent:humble udp4 --port 9999 -v4
```



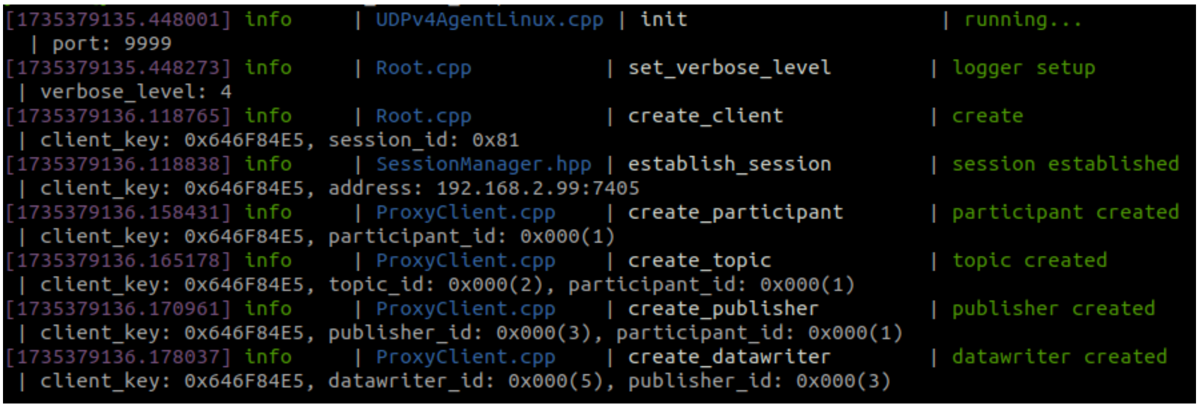
```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     |
running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level        | l
ogger setup            | verbose_level: 4
```

Then, turn on the car switch and wait for the car to connect to the proxy. The connection is successful as shown in the figure below.



```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     | running...             | port: 8899
[1735179211.772581] info     | Root.cpp           | set_verbose_level        | logger setup           | verbose_level: 4
[1735179325.739277] info     | Root.cpp           | create_client            | create                 | client_key: 0x0E5C3397, sess
ion_id: 0x81
[1735179325.739348] info     | SessionManager.hpp | establish_session        | session established    | client_key: 0x0E5C3397, addr
ess: 192.168.2.102:49954
[1735179325.971694] info     | ProxyClient.cpp    | create_participant       | participant created    | client_key: 0x0E5C3397, part
icipant_id: 0x000(1)
[1735179326.046043] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info     | ProxyClient.cpp    | create_subscriber        | subscriber created     | client_key: 0x0E5C3397, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info     | ProxyClient.cpp    | create_datareader        | datareader created     | client_key: 0x0E5C3397, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
```

Camera proxy, the connection is successful as shown in the figure below.



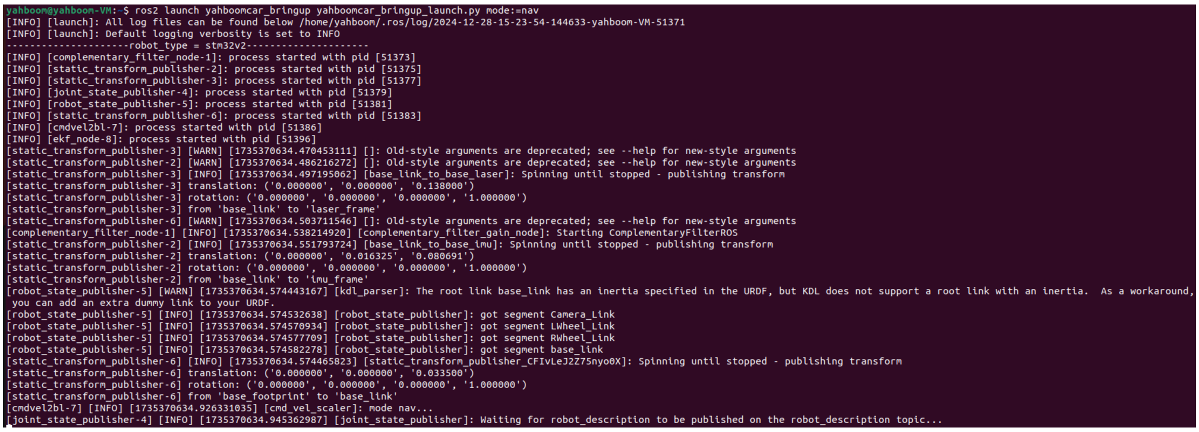
```
[1735379135.448001] info     | UDPv4AgentLinux.cpp | init                     | running...
 | port: 9999
[1735379135.448273] info     | Root.cpp           | set_verbose_level        | logger setup
 | verbose_level: 4
[1735379136.118765] info     | Root.cpp           | create_client            | create
 | client_key: 0x646F84E5, session_id: 0x81
[1735379136.118838] info     | SessionManager.hpp | establish_session        | session established
 | client_key: 0x646F84E5, address: 192.168.2.99:7405
[1735379136.158431] info     | ProxyClient.cpp    | create_participant       | participant created
 | client_key: 0x646F84E5, participant_id: 0x000(1)
[1735379136.165178] info     | ProxyClient.cpp    | create_topic             | topic created
 | client_key: 0x646F84E5, topic_id: 0x000(2), participant_id: 0x000(1)
[1735379136.170961] info     | ProxyClient.cpp    | create_publisher         | publisher created
 | client_key: 0x646F84E5, publisher_id: 0x000(3), participant_id: 0x000(1)
[1735379136.178037] info     | ProxyClient.cpp    | create_datawriter        | datawriter created
 | client_key: 0x646F84E5, datawriter_id: 0x000(5), publisher_id: 0x000(3)
```

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 mode:=nav
```

```
#Parameter description, adjust the speed of the car, the mode is navigation mode
mode:=nav
```



```
yahboom@yahboom-VM:~$ ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py mode:=nav
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-12-28-15-23-54-144633-yahboom-VM-51371
[INFO] [launch]: Default logging verbosity is set to INFO
---------------------robot_type = stm32v2--------------------
[INFO] [complementary_filter_node-1]: process started with pid [51373]
[INFO] [static_transform_publisher-2]: process started with pid [51375]
[INFO] [static_transform_publisher-3]: process started with pid [51377]
[INFO] [joint_state_publisher-4]: process started with pid [51379]
[INFO] [robot_state_publisher-5]: process started with pid [51381]
[INFO] [static_transform_publisher-6]: process started with pid [51383]
[INFO] [cmdvel2bl-7]: process started with pid [51386]
[INFO] [ekf_node-8]: process started with pid [51396]
[static_transform_publisher-3] [WARN] [1735370634.470453111] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-2] [WARN] [1735370634.486216272] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [INFO] [1735370634.497195062] [base_link_to_base_laser]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('0.000000', '0.000000', '0.138000')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'laser_frame'
[static_transform_publisher-6] [WARN] [1735370634.503711546] []: Old-style arguments are deprecated; see --help for new-style arguments
[complementary_filter_node-1] [INFO] [1735370634.538214920] [complementary_filter_gain_node]: Starting ComplementaryFilterROS
[static_transform_publisher-2] [INFO] [1735370634.551793724] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-2] translation: ('0.000000', '0.016325', '0.080691')
[static_transform_publisher-2] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-2] from 'base_link' to 'imu_frame'
[robot_state_publisher-5] [WARN] [1735370634.574443167] [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] [1735370634.574532638] [robot_state_publisher]: got segment Camera_Link
[robot_state_publisher-5] [INFO] [1735370634.574570934] [robot_state_publisher]: got segment LWheel_Link
[robot_state_publisher-5] [INFO] [1735370634.574577709] [robot_state_publisher]: got segment RWheel_Link
[robot_state_publisher-5] [INFO] [1735370634.574582278] [robot_state_publisher]: got segment base_link
[static_transform_publisher-6] [INFO] [1735370634.574465823] [static_transform_publisher_CFIvLeJ2Z75nyo0X]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.033500')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[cmdvel2bl-7] [INFO] [1735370634.926331035] [cmd_vel_scaler]: mode nav...
[joint_state_publisher-4] [INFO] [1735370634.945362987] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
```

Start APP navigation command, terminal input,

Navigation two choices

1. Ordinary version navigation:

```
ros2 launch yahboomcar_nav navigation_dwb_app_launch.xml
maps:=/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/testaa.yaml
```

Load map parameters: maps:=/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/testaa.yaml (replaceable target map)

2. Fast relocation version navigation:

Start fast relocation,

Note: The .pbstream map file used in this navigation version requires the map to be saved in advance. Refer to the cartograph map construction algorithm tutorial

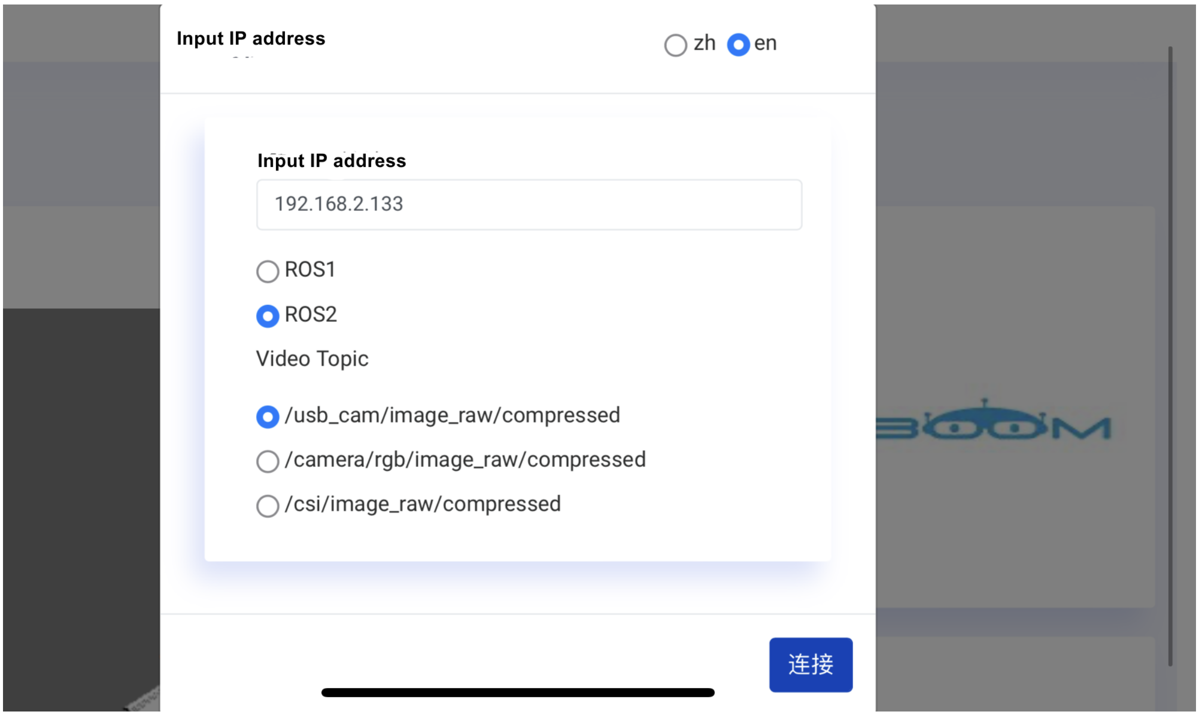
```
ros2 launch yahboomcar_nav localization_imu_odom.launch.py use_rviz:=false
load_state_filename:=/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/testaa.p
bstream
```

```
ros2 launch yahboomcar_nav navigation_cartodwb_app_launch.xml
maps:=/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/testaa.yaml
```

Start the camera display command, input in the terminal,

```
#Start ESP32 camera
ros2 run yahboom_esp32_camera sub_img
```

The mobile phone APP displays the following figure, enter the IP address of the car, [zh] means Chinese, [en] means English; select ROS2, select the Video Tpoic below: /usb_cam/image_raw/compressed, and finally click [Connect]



Input IP address ○ zh ● en

Input IP address

192.168.2.133

○ ROS1

● ROS2

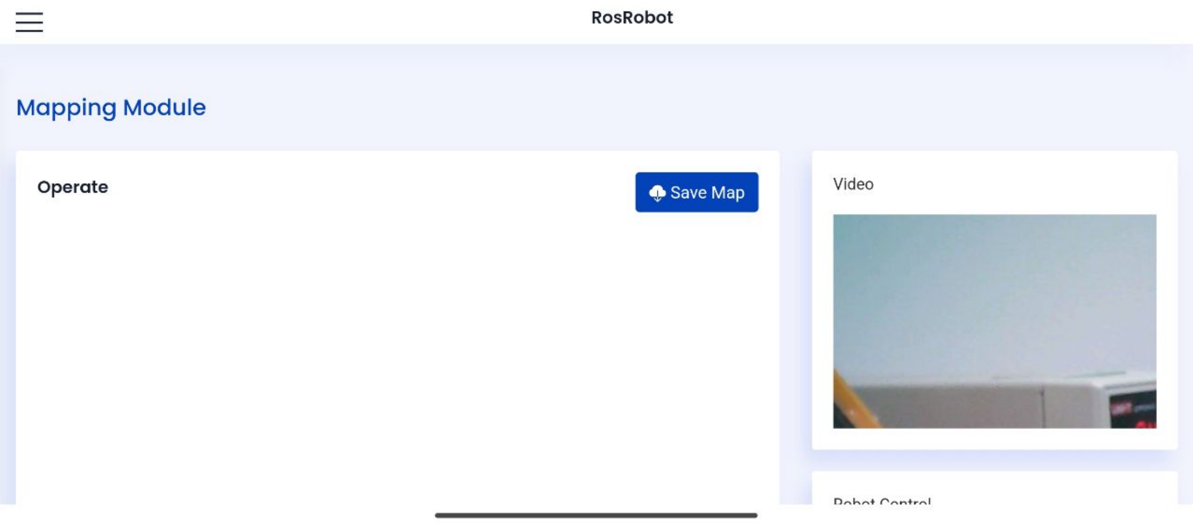
Video Topic

● /usb_cam/image_raw/compressed

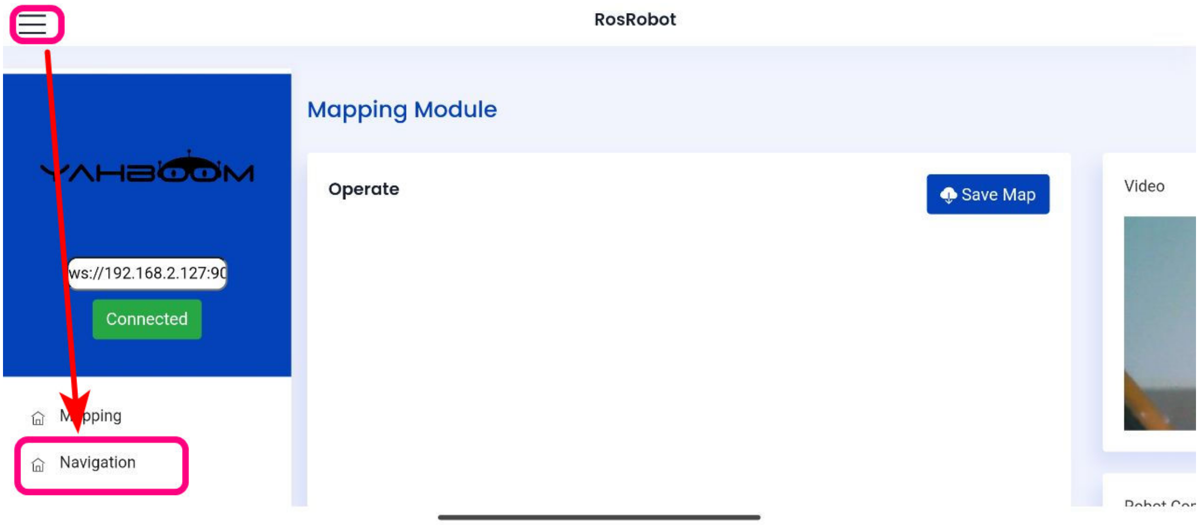
○ /camera/rgb/image_raw/compressed

○ /csi/image_raw/compressed

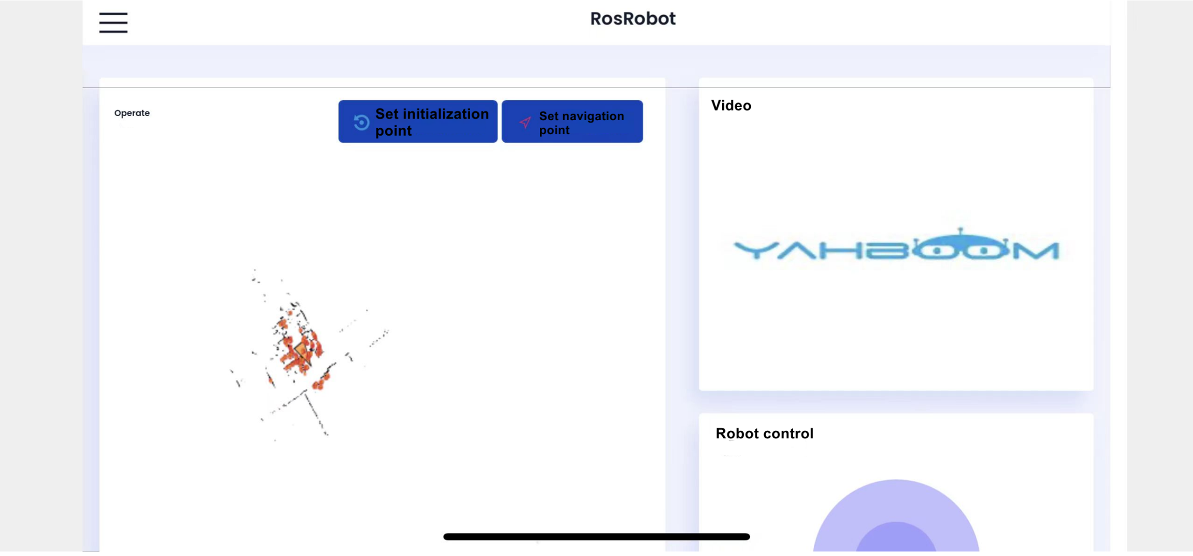
After successfully connecting, the display is as follows,



As shown in the figure below, select the navigation interface,



Then, combined with the actual position 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 that the position is accurate. As shown in the figure below,



Then, click [Set navigation point], give the car a destination, the car will plan a path and move to the destination according to the path.

4. Code analysis

Here is the launch file for opening 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 launch files and nodes are run here:

- rosbridge_websocket_launch.xml: Open the rosbridge service related nodes. After startup, you can connect to ROS through the network
- laserscan_to_point_publisher: Publish the point cloud conversion of the radar 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

The launch file for APP fast repositioning navigation, navigation_cartodwb_app_launch.xml, is similar.