ROS Robot APP mapping

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check [Must read before use] to set the IP and ROS_DOMAIN_ID on the board.

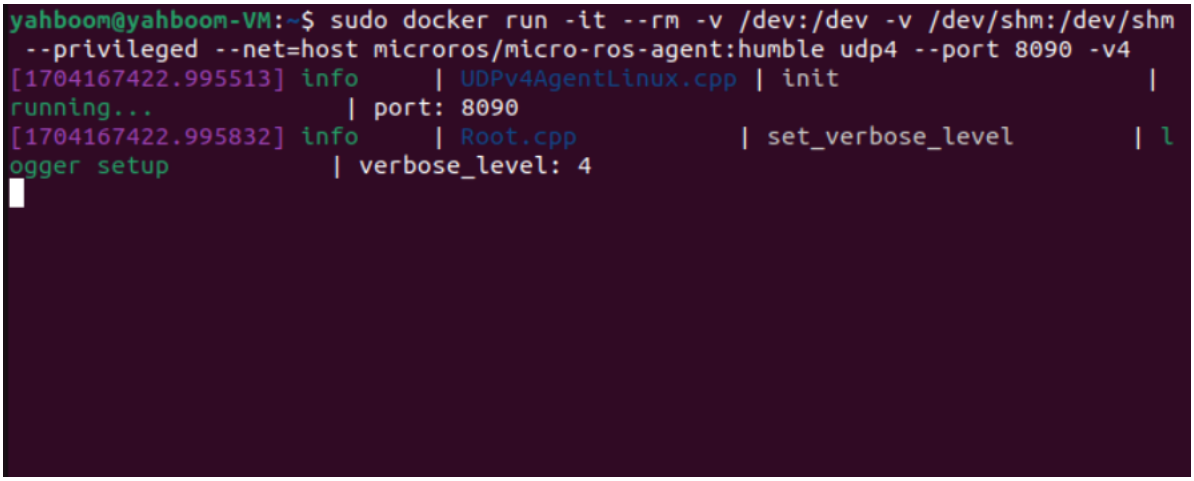
1. Program function description

The car connects to the 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. Start and connect to the agent

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

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8090 -v4
```



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

```
[1702630014.015846] info     | ProxyClient.cpp     | create_participant       | participant created     | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp     | create_publisher         | publisher created       | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp     | create_datawriter        | datawriter created      | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp     | create_publisher         | publisher created       | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp     | create_datawriter        | datawriter created      | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp     | create_publisher         | publisher created       | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp     | create_datawriter        | datawriter created      | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp     | create_subscriber        | subscriber created      | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp     | create_datareader        | datareader created      | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp     | create_subscriber        | subscriber created      | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp     | create_datareader        | datareader created      | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp     | create_topic             | topic created           | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp     | create_subscriber        | subscriber created      | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp     | create_datareader        | datareader created      | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

3. Start the program

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

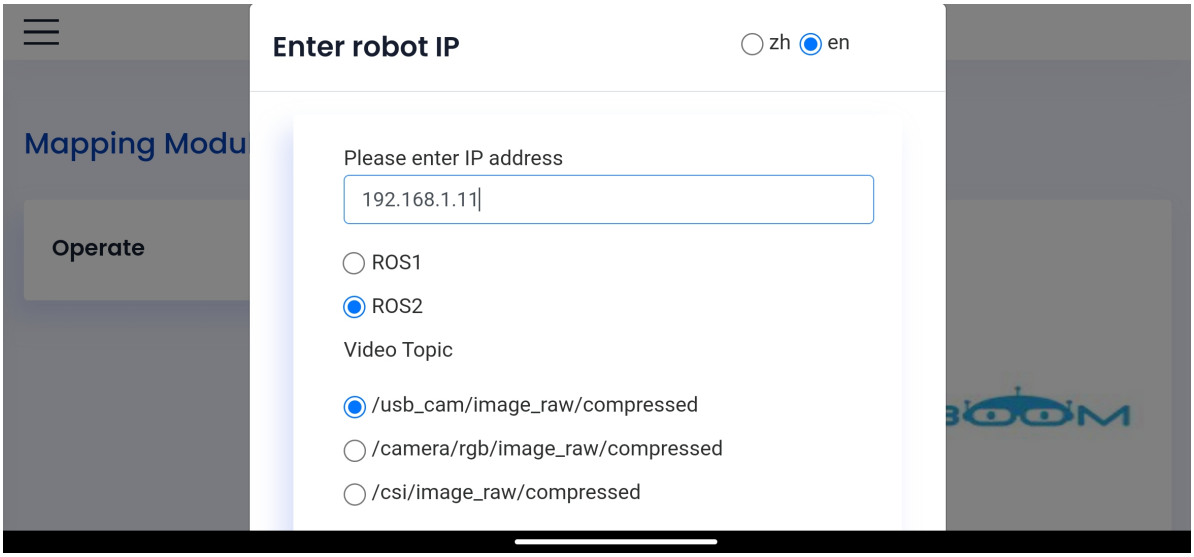
```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

```
[INFO] [imu_filter_madgwick_node-1]: process started with pid [6638]
[INFO] [ekf_node-2]: process started with pid [6640]
[INFO] [static_transform_publisher-3]: process started with pid [6642]
[INFO] [joint_state_publisher-4]: process started with pid [6644]
[INFO] [robot_state_publisher-5]: process started with pid [6646]
[INFO] [static_transform_publisher-6]: process started with pid [6658]
[static_transform_publisher-3] [WARN] [1702865272.944043208] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-6] [WARN] [1702865272.984740987] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-3] [INFO] [1702865272.991057276] [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'
[static_transform_publisher-6] [INFO] [1702865273.005707993] [static_transform_publisher_JH06Gexf4GRodmgs]: 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'
[robot_state_publisher-5] [WARN] [1702865273.013202438] [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] [1702865273.013312806] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1702865273.013516195] [robot_state_publisher]: got segment imu_link
[robot_state_publisher-5] [INFO] [1702865273.013524175] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1702865273.013528144] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1702865273.013531665] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1702865273.013535185] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1702865273.013538763] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1702865273.013542135] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1702865273.013545612] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1702865273.030399479] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1702865273.031826501] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1702865273.031858361] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1702865273.032488302] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032525566] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032531441] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.053298796] [imu_filter]: First IMU message received.
[joint_state_publisher-4] [INFO] [1702865273.282975810] [joint_state_publisher]: Waiting for robot_description to be published on th
e robot_description topic...
```

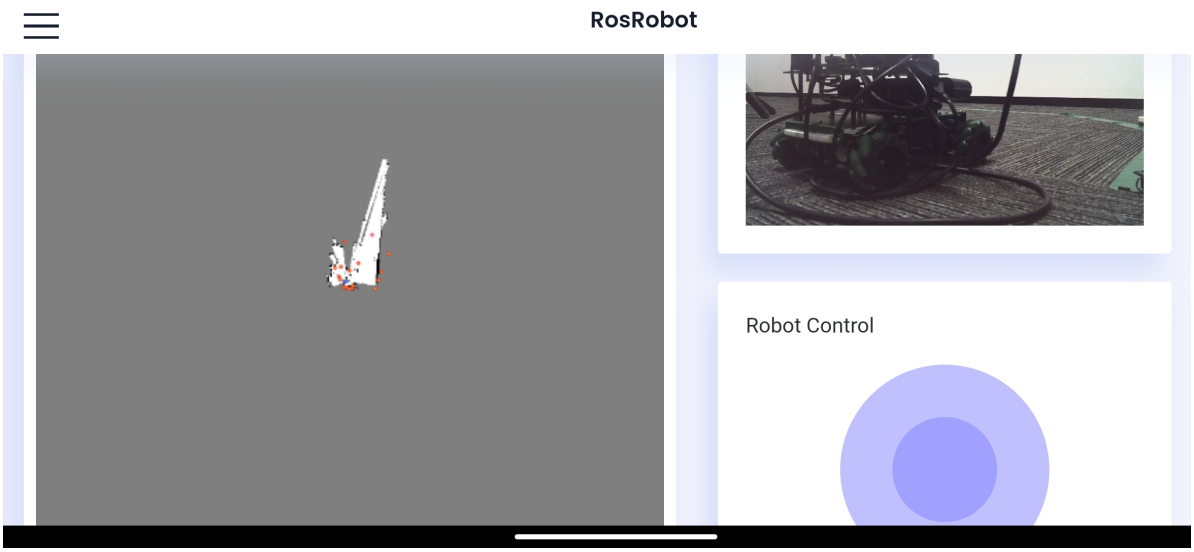
Start the APP mapping command and enter it in the terminal,

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

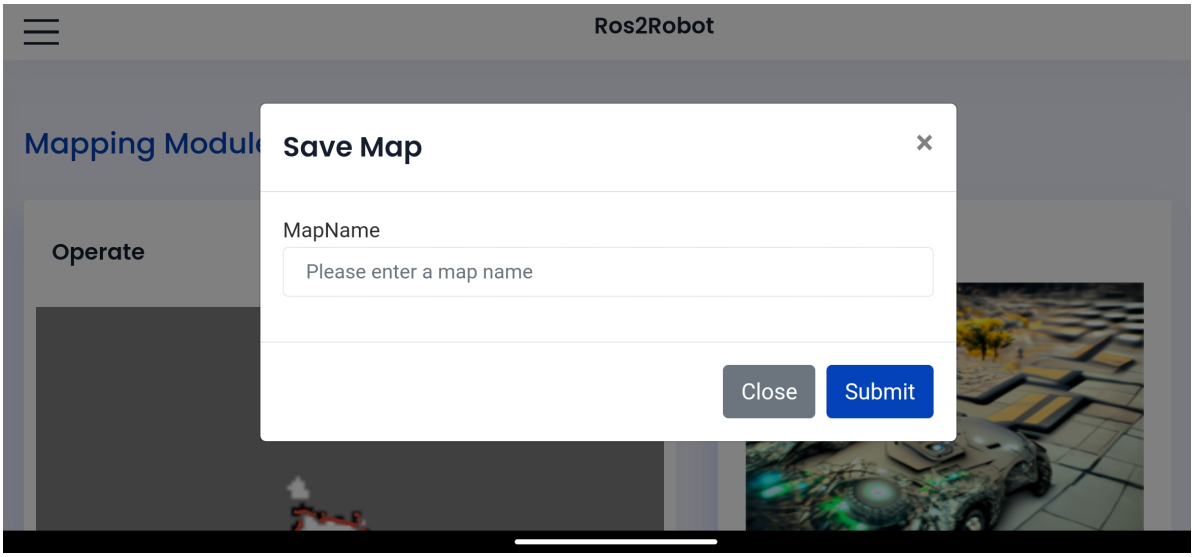
The mobile APP displays as shown below, enter the IP address of the car, **[zh]** means Chinese, **[en]** means English; select ROS2, the Video Tpoic below can be selected at will, and finally click **[Connect]**



After successful connection, the following is displayed:

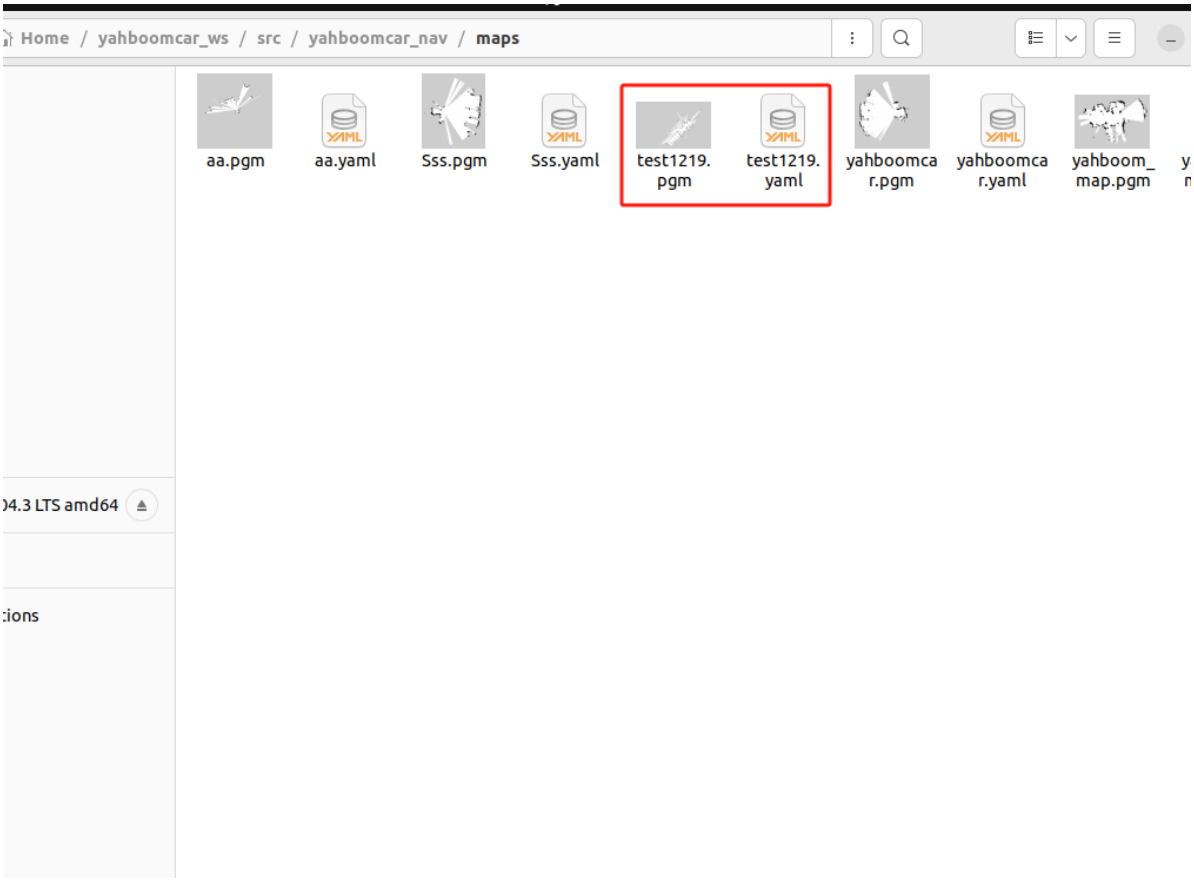


Use the sliding wheel to control the car to move slowly through the area that needs to be mapped, then click to save the map, enter the map name and click submit to save the map.



The location where the map is saved is,

```
/home/yahboom/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