

Robot information release

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 view [Expansion Board Parameter Configuration] to set the IP and ROS_DOMAIN_ID on the board.

1. Program function description

After the car connects to the proxy, it will publish sensor data such as radar and imu. You can run commands in the matching virtual machine to query this information, and you can also control the data of sensors such as buzzers.

2. Query the car information

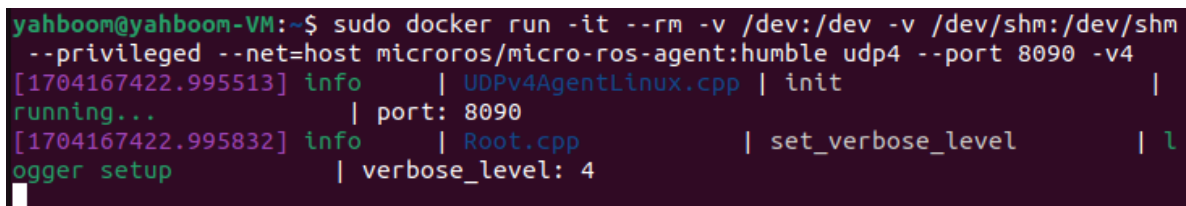
2.1. Start and connect the agent

If you are using the factory virtual machine system, you can enter in the terminal:

```
sh ~/start_agent_computer.sh
```

If you are using a third-party virtual machine system, you need to install the docker development environment first, and open the terminal to enter:

```
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
```



```
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 8090 -v4
[1704167422.995513] info      | UDPv4AgentLinux.cpp | init
running...                | port: 8090
[1704167422.995832] info      | Root.cpp             | set_verbose_level
ogger setup                | verbose_level: 4
```

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

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

2.2. Query the car node information

If it is the Raspberry Pi desktop version and the Jetson Nano and X5 desktop versions, you need to enter docker in advance and enter the terminal.

```
sh ~/ros2_humble.sh
```

When the following interface appears, it means that you have successfully entered docker.

```
pi@raspberrypi:~$ S ./ros2_humble.sh
access control disabled, clients can connect from any host
MY_DOMAIN_ID: 20
root@raspberrypi:/#
```

Take the matching virtual machine as an example, enter the following command in the terminal to query the node,

```
ros2 node list
```

```
yahboom@yahboom-VM:~$ ros2 node list
/YB_PLAMSLAM_Node
```

Then enter the following command to query which topics the node has published/subscribed to,

```
ros2 node info /YB_PLAMSLAM_Node
```

```
yahboom@yahboom-VM:~$ ros2 node info /YB_PLAMSLAM_Node
/YB_PLAMSLAM_Node
Subscribers:
  /beep: std_msgs/msg/UInt16
Publishers:
  /battery: std_msgs/msg/UInt16
  /imu: sensor_msgs/msg/Imu
  /scan: sensor_msgs/msg/LaserScan
Service Servers:

Service Clients:

Action Servers:

Action Clients:
```

It can be seen that the subscribed topics are,

/beep: buzzer control

The published topics are,

/imu: imu module data

/scan: radar module data

We can also query the topic command, terminal input,

```
ros2 topic list
```

```
yahboom@yahboom-VM:~$ ros2 topic list
/battery
/beep
/imu
/parameter_events
/rosout
/scan
```

2.3, query topic data

Query radar data,

```
ros2 topic echo /scan
```

```

header:
  stamp:
    sec: 1100
    nanosec: 349000000
  frame_id: laser_frame
angle_min: -3.1415927410125732
angle_max: 3.1415927410125732
angle_increment: 0.01745329238474369
time_increment: 0.0
scan_time: 0.0
range_min: 0.1199999731779099
range_max: 8.0
ranges:
- 0.1459999978542328
- 0.15600000321865082
- 0.16899999976158142
- 0.0
- 0.0
- 0.8100000023841858
- 0.800000011920929
- 0.8450000286102295
- 0.16099999845027924
- 0.1599999964237213
- 0.1599999964237213
- 0.1599999964237213
- 0.0
- 0.2240000069141388
- 0.2240000069141388
- 0.21799999475479126
- 0.8820000290870667
- 0.8420000076293945
- 0.828000009059906

```

Query imu data,

```
ros2 topic echo /imu
```

```

header:
  stamp:
    sec: 1161
    nanosec: 198000000
  frame_id: imu_frame
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 1.0
orientation_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
angular_velocity:
  x: -0.024501830339431763
  y: 0.015979453921318054
  z: -0.0010652969358488917
angular_velocity_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
linear_acceleration:
  x: -0.1363811045885086
  y: -0.2930997610092163
  z: 9.69861125946045
linear_acceleration_covariance:
- 0.0

```

3. Release car control information

3.1. Control buzzer

First query the following buzzer topic related information, terminal input,

```
ros2 topic info /beep
```

```
yahboom@yahboom-VM:~$ ros2 topic info /beep
Type: std_msgs/msg/UInt16
Publisher count: 0
Subscription count: 1
```

The data type is std_msgs/msg/UInt16. Then enter the following command to turn on the buzzer, terminal input,

```
ros2 topic pub /beep std_msgs/msg/UInt16 "data: 1"
```

```
yahboom@yahboom-VM:~$ ros2 topic pub /beep std_msgs/msg/UInt16 "data: 1"
publisher: beginning loop
publishing #1: std_msgs.msg.UInt16(data=1)

publishing #2: std_msgs.msg.UInt16(data=1)

publishing #3: std_msgs.msg.UInt16(data=1)

publishing #4: std_msgs.msg.UInt16(data=1)

publishing #5: std_msgs.msg.UInt16(data=1)

publishing #6: std_msgs.msg.UInt16(data=1)
```

Enter the following command to turn off the buzzer, terminal input,

```
ros2 topic pub /beep std_msgs/msg/UInt16 "data: 0"
```

```
yahboom@yahboom-VM:~$ ros2 topic pub /beep std_msgs/msg/UInt16 "data: 0"
publisher: beginning loop
publishing #1: std_msgs.msg.UInt16(data=0)

publishing #2: std_msgs.msg.UInt16(data=0)

publishing #3: std_msgs.msg.UInt16(data=0)
```