

# 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 check [Must read before use] to set the IP and ROS\_DOMAIN\_ID on the board.

## 1. Program function description

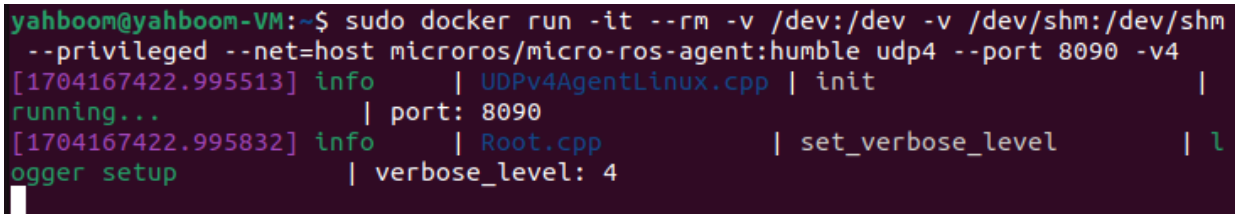
After the car is connected to the agent, it will publish sensor data such as radar and imu. You can run commands in the supporting virtual machine/Raspberry Pi 5 to query this information. You can also publish control data of sensors such as speed and buzzer.

## 2. Query car information

### 2.1. 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
```

A terminal window with a dark purple background. The prompt is 'yahboom@yahboom-VM:~\$'. The command entered is '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'. The output shows log messages: '[1704167422.995513] info | UDPv4AgentLinux.cpp | init | running... | port: 8090', '[1704167422.995832] info | Root.cpp | set\_verbose\_level | 1', and 'ogger setup | verbose\_level: 4'. A cursor is visible on the line following the last message.

```
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 | 1
ogger setup | verbose_level: 4
█
```

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, 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

Enter the following command in the terminal to query the node,

```
ros2 node list
```

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

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

```
ros2 node info /YB_Car_Node
```

```
yahboom@yahboom-VM:~$ ros2 node info /YB_Car_Node
/YB_Car_Node
Subscribers:
  /beep: std_msgs/msg/UInt16
  /cmd_vel: geometry_msgs/msg/Twist
  /servo_s1: std_msgs/msg/Int32
  /servo_s2: std_msgs/msg/Int32
Publishers:
  /imu: sensor_msgs/msg/Imu
  /odom_raw: nav_msgs/msg/Odometry
  /scan: sensor_msgs/msg/LaserScan
Service Servers:
Service Clients:
Action Servers:
Action Clients:
```

It can be seen that the topics subscribed to include:

/beep: Buzzer control

/cmd\_vel: Car speed control

/servo\_s1: s1 servo gimbal control

/servo\_s2: s2 servo gimbal control

Posted topics include:

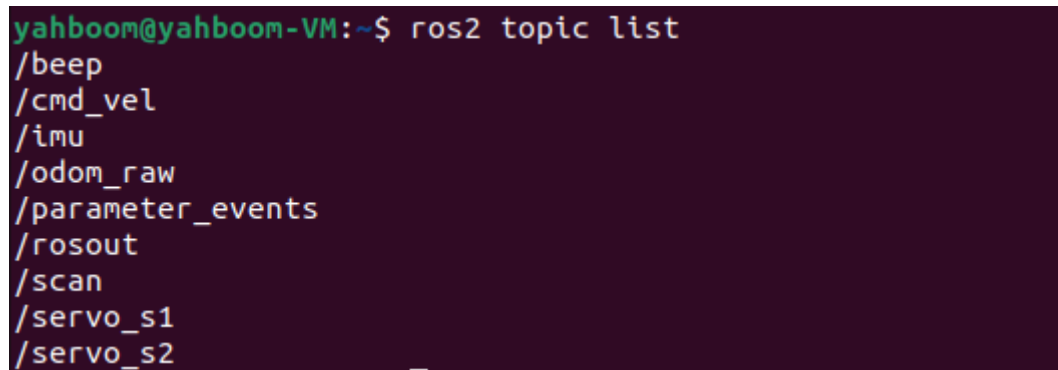
/imu: imu module data

/odom: Odometer module data

/scan: Radar module data

We can also query the topic command and enter it in the terminal,

```
ros2 topic list
```

A terminal window with a dark purple background. The prompt is 'yahboom@yahboom-VM:~\$'. The command 'ros2 topic list' has been entered, and the output is a list of topics: /beep, /cmd\_vel, /imu, /odom\_raw, /parameter\_events, /rosout, /scan, /servo\_s1, and /servo\_s2.

```
yahboom@yahboom-VM:~$ ros2 topic list
/beep
/cmd_vel
/imu
/odom_raw
/parameter_events
/rosout
/scan
/servo_s1
/servo_s2
```

## 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: 0.11999999731779099
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

```

Query odom data,

```
ros2 topic echo /odom_raw
```

```

header:
  stamp:
    sec: 1266
    nanosec: 683000000
  frame_id: odom_frame
child_frame_id: base_footprint
pose:
  position:
    x: 0.0
    y: 0.0
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 1.0
covariance:
- 0.001
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0

```

### 3. Publish car control information

### 3.1. Control the buzzer

First, query the relevant information about the following buzzer topics, enter it in the terminal,

```
ros2 topic info /beep
```

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

Learn that the data type is std\_msgs/msg/UInt16. Then enter the following command to turn on the buzzer and enter in the terminal,

```
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)
```

### 3.2. Release speed control information

We assume that the released car moves at a linear speed of 0.5 and an angular speed of 0.2, and the terminal input is,

```
ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0},
angular: {x: 0.0, y: 0.0, z: 0.2}}"
```

```

yahboom@yahboom-VM:~$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))
publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))
publishing #3: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))
publishing #4: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))

```

If parking, enter

```

ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.0, y: 0.0, z: 0.0},
angular: {x: 0.0, y: 0.0, z: 0.0}}"

```

```

yahboom@yahboom-VM:~$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #3: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #4: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #5: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))

```

### 3.3. Control the gimbal servo

What needs to be noted here is that the range of s1 servo is [-90,90], and the range of s2 servo is [-90,20]. If the value exceeds the range, the servo will not rotate.

We assume that the s1 servo is controlled to rotate 30 degrees, and the terminal input is,

```

ros2 topic pub /servo_s1 std_msgs/msg/Int32 "data: 30"

```

```

yahboom@yahboom-VM:~$ ros2 topic pub /servo_s1 std_msgs/msg/Int32 "data: 30"
publisher: beginning loop
publishing #1: std_msgs.msg.Int32(data=30)

publishing #2: std_msgs.msg.Int32(data=30)

publishing #3: std_msgs.msg.Int32(data=30)

publishing #4: std_msgs.msg.Int32(data=30)

```

In the same way, if the s2 servo is controlled to rotate -30 degrees, the terminal input will be

```

ros2 topic pub /servo_s2 std_msgs/msg/Int32 "data: -30"

```

```

yahboom@yahboom-VM:~$ ros2 topic pub /servo_s2 std_msgs/msg/Int32 "data: -30"
publisher: beginning loop
publishing #1: std_msgs.msg.Int32(data=-30)

publishing #2: std_msgs.msg.Int32(data=-30)

publishing #3: std_msgs.msg.Int32(data=-30)

publishing #4: std_msgs.msg.Int32(data=-30)

```