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

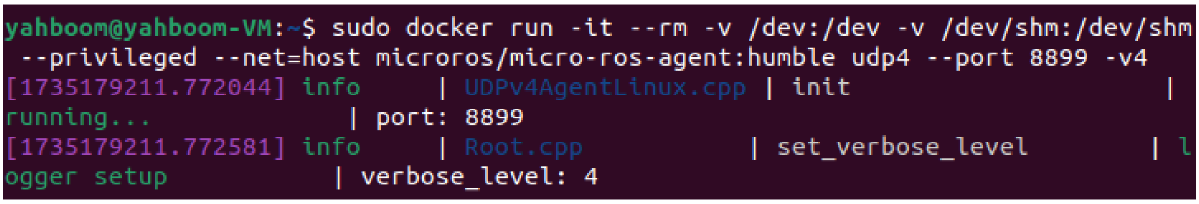
After the car connects to the proxy, it will publish sensor data such as radar and imu. You can run commands in the matching virtual machine to query this information, and you can also publish control data of sensors such as speed and buzzer.

2. Query the car information

2.1. Start and connect the agent

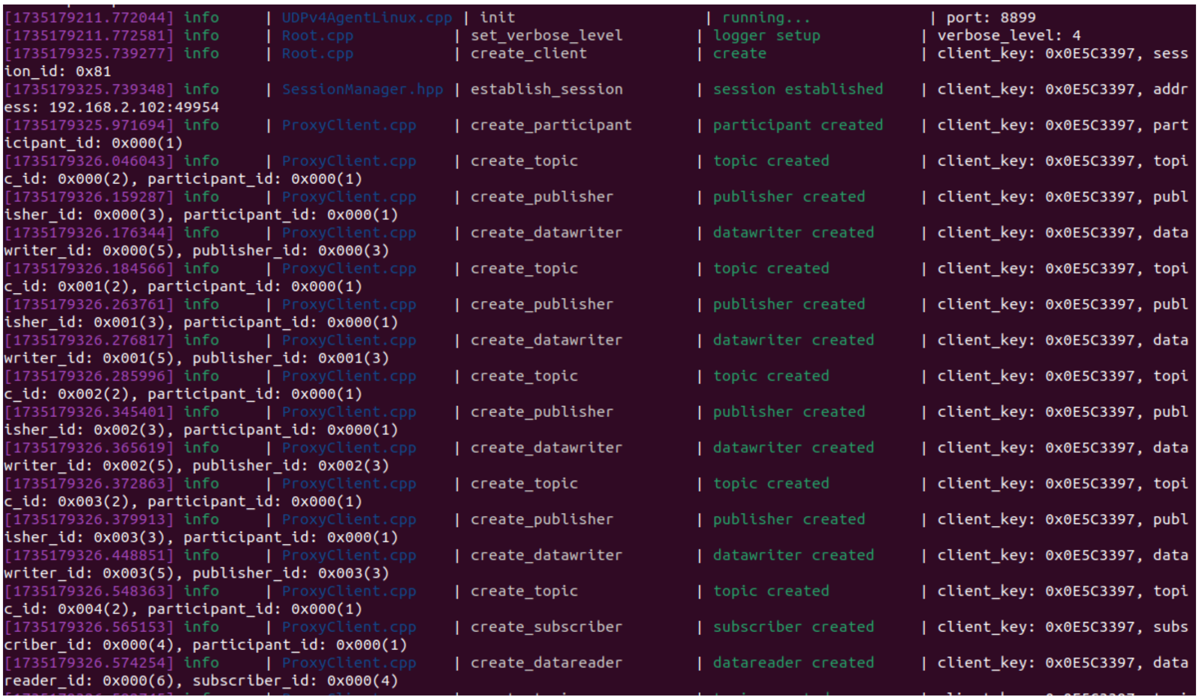
Take the matching virtual machine as an example, enter the following command to start the agent,

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -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 agent. The connection is successful as shown in the figure below,

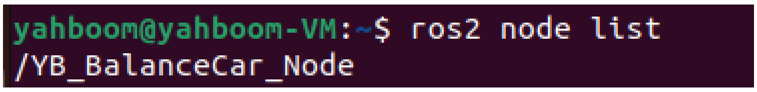


```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                    | running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level       | logger setup           | verbose_level: 4
[1735179325.739277] info     | Root.cpp            | create_client           | 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)
```

2.2. Query the car node information

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

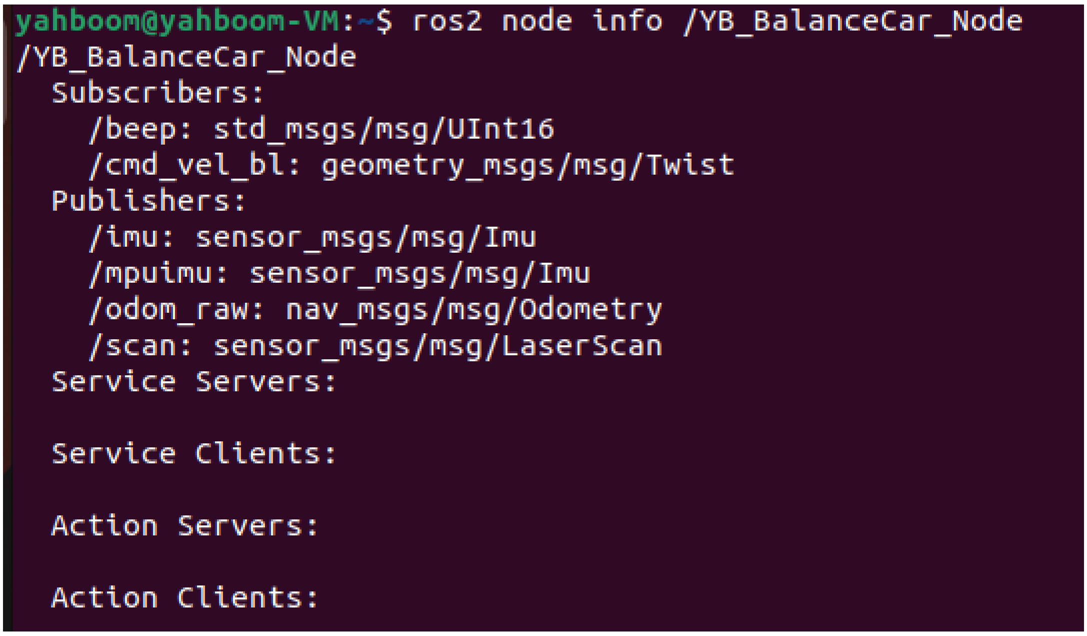
```
ros2 node list
```



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

Then enter the following command to query which topics the node publishes/subscribes to,

```
ros2 node info /YB_Car_Node
```



```
yahboom@yahboom-VM:~$ ros2 node info /YB_BalanceCar_Node
/YB_BalanceCar_Node
  Subscribers:
    /beep: std_msgs/msg/UInt16
    /cmd_vel_bl: geometry_msgs/msg/Twist
  Publishers:
    /imu: sensor_msgs/msg/Imu
    /mpuimu: 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 subscribed topics are,

/beep: buzzer control

/cmd_vel_bl: car speed control

The published topics are,

/imu: imu module data

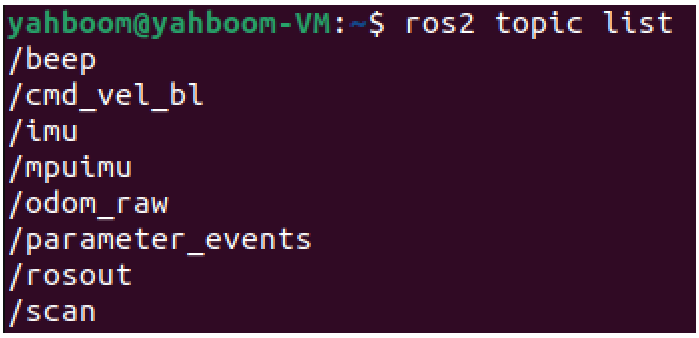
/mpuimu: mpu6050 module data

/odom_raw: odometer data

/scan: radar module data

We can also query the topic command, terminal input,

```
ros2 topic list
```

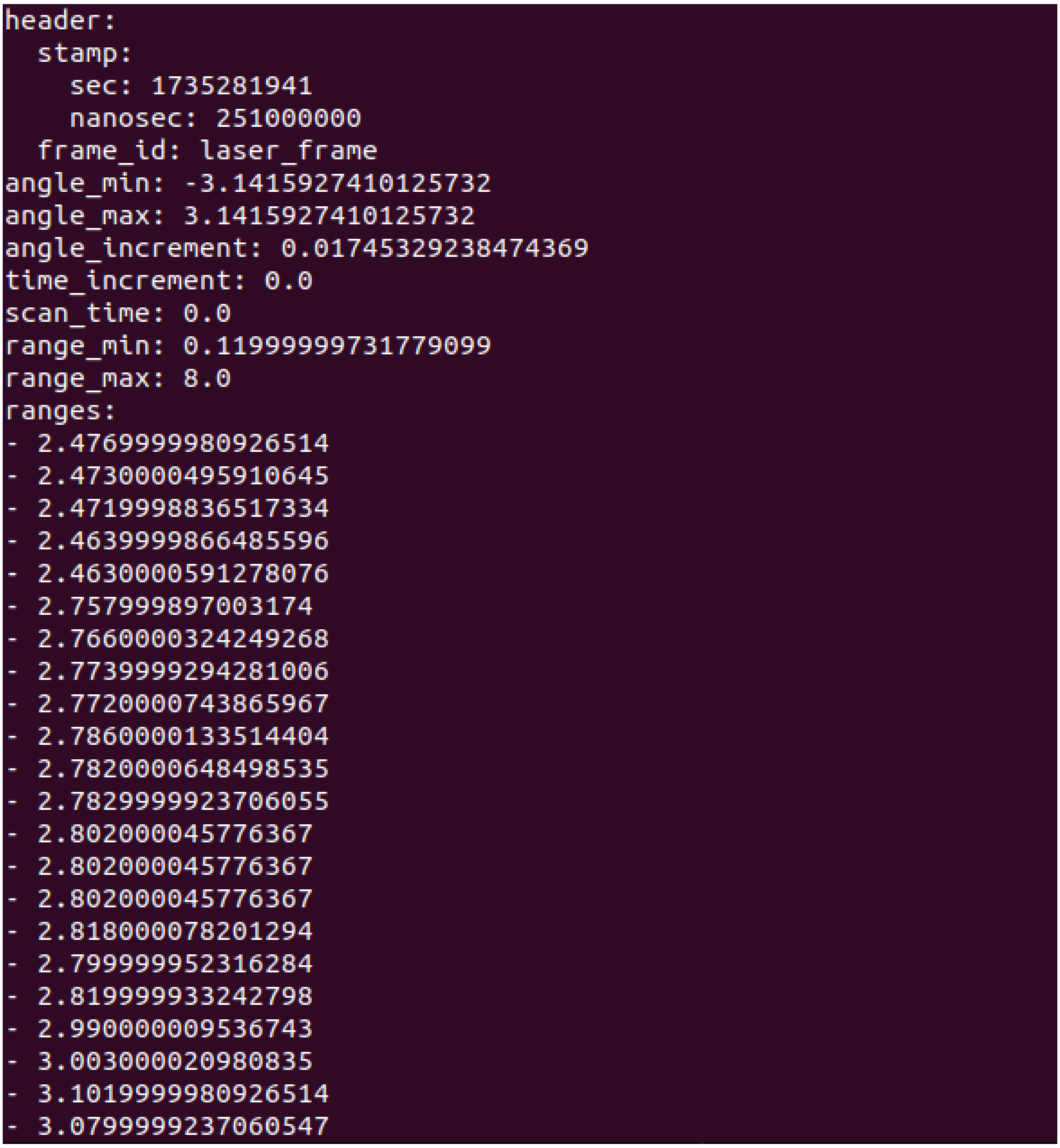


```
yahboom@yahboom-VM:~$ ros2 topic list
/beep
/cmd_vel_bl
/imu
/mpuimu
/odom_raw
/parameter_events
/rosout
/scan
```

2.3, query topic data

Query radar data,

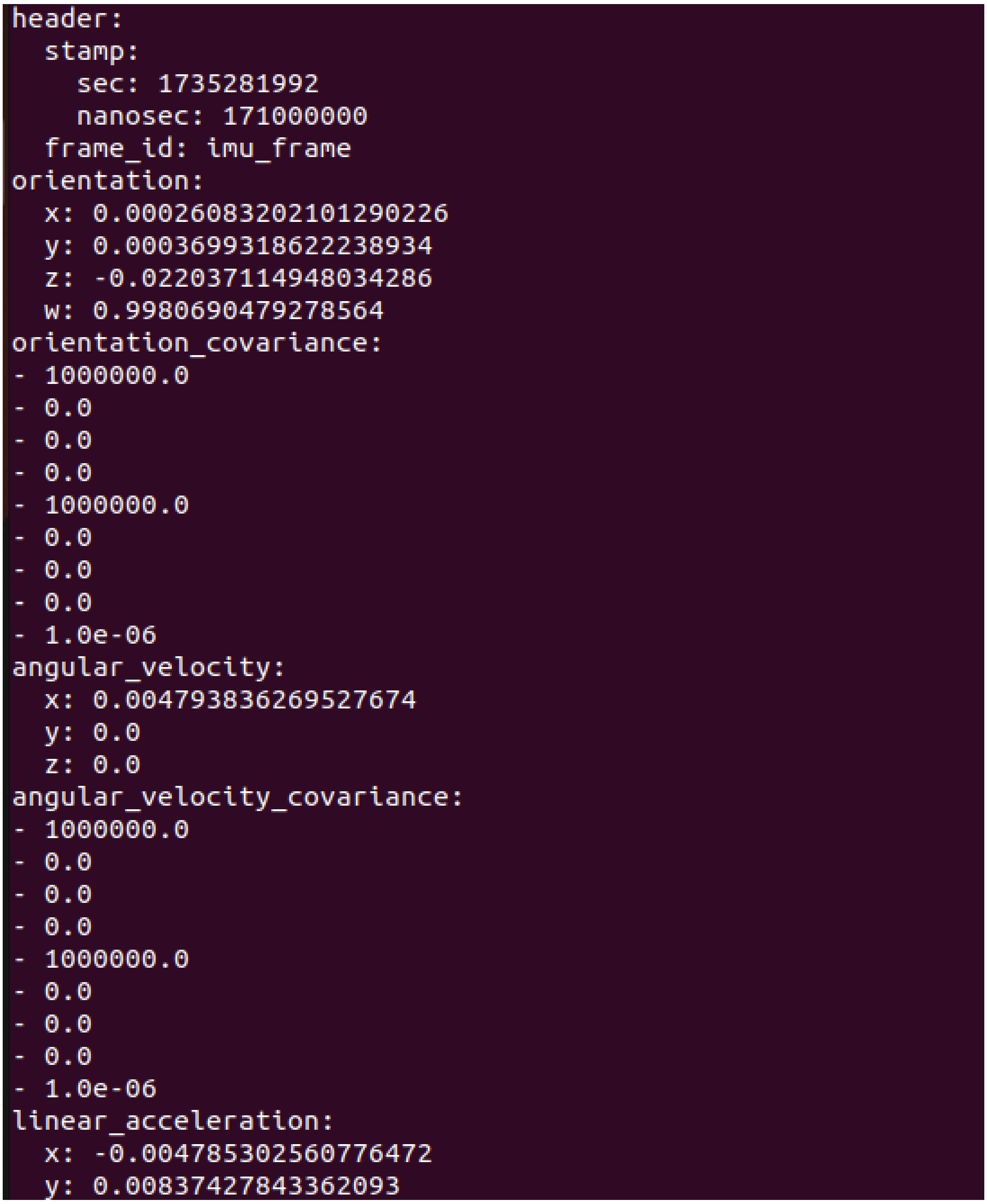
```
ros2 topic echo /scan
```



```
header:
  stamp:
    sec: 1735281941
    nanosec: 251000000
  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:
- 2.4769999980926514
- 2.4730000495910645
- 2.4719998836517334
- 2.4639999866485596
- 2.4630000591278076
- 2.757999897003174
- 2.7660000324249268
- 2.7739999294281006
- 2.7720000743865967
- 2.7860000133514404
- 2.7820000648498535
- 2.7829999923706055
- 2.802000045776367
- 2.802000045776367
- 2.802000045776367
- 2.818000078201294
- 2.799999952316284
- 2.819999933242798
- 2.990000009536743
- 3.003000020980835
- 3.1019999980926514
- 3.0799999237060547
```

Query imu data,

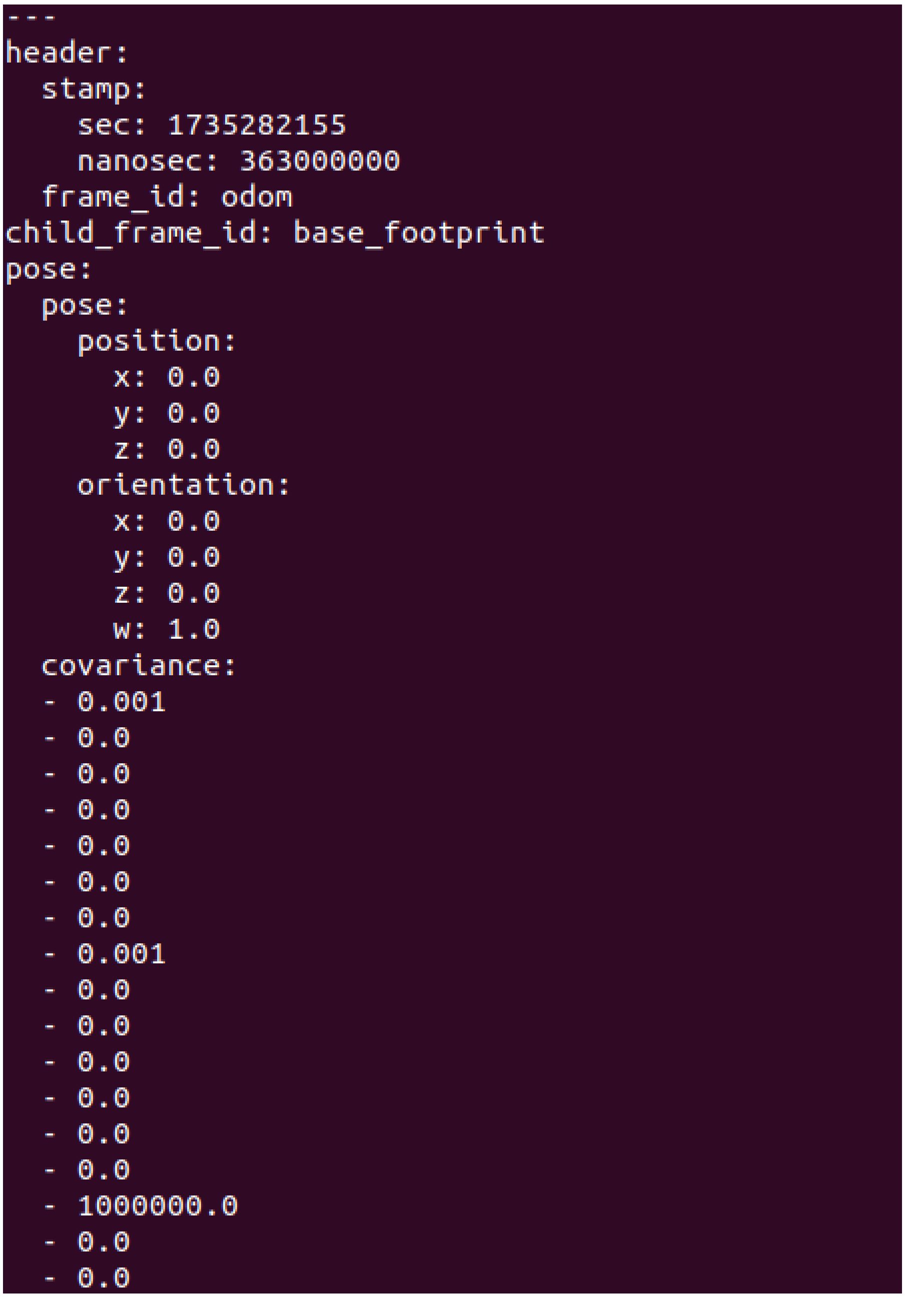
```
ros2 topic echo /imu
```



```
header:
  stamp:
    sec: 1735281992
    nanosec: 171000000
  frame_id: imu_frame
orientation:
  x: 0.00026083202101290226
  y: 0.0003699318622238934
  z: -0.022037114948034286
  w: 0.9980690479278564
orientation_covariance:
- 1000000.0
- 0.0
- 0.0
- 0.0
- 1000000.0
- 0.0
- 0.0
- 0.0
- 1.0e-06
angular_velocity:
  x: 0.004793836269527674
  y: 0.0
  z: 0.0
angular_velocity_covariance:
- 1000000.0
- 0.0
- 0.0
- 0.0
- 1000000.0
- 0.0
- 0.0
- 0.0
- 1.0e-06
linear_acceleration:
  x: -0.004785302560776472
  y: 0.00837427843362093
```

Query odom data,

```
ros2 topic echo /odom_raw
```



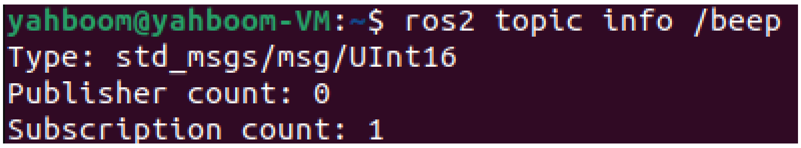
```
---
header:
  stamp:
    sec: 1735282155
    nanosec: 363000000
  frame_id: odom
child_frame_id: base_footprint
pose:
  pose:
    position:
      x: 0.0
      y: 0.0
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
  covariance:
  - 0.001
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 0.001
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 0.0
  - 1000000.0
  - 0.0
  - 0.0
```

3. Publish car control information

3.1. Control buzzer

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

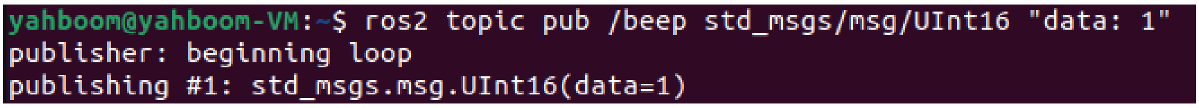
```
ros2 topic info /beep
```



```
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,

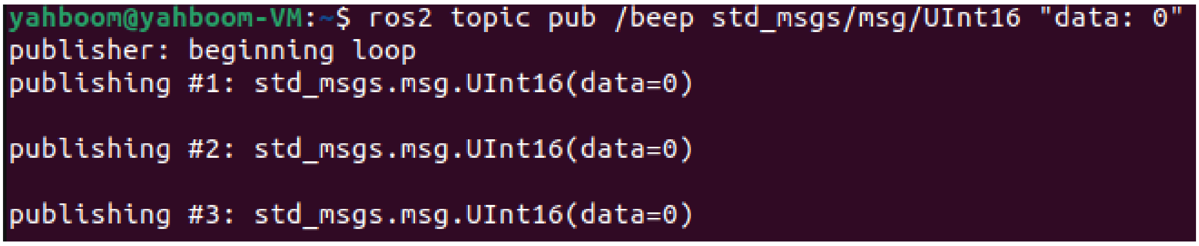
```
ros2 topic pub /beep std_msgs/msg/UInt16 "data: 1"
```



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

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

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



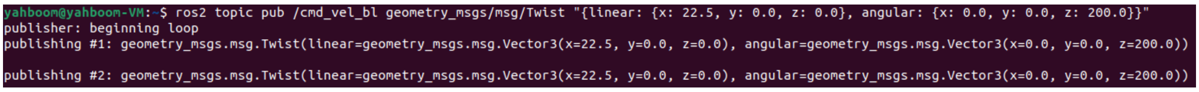
```
yahboom@yahboom-VM:~$ ros2 topic pub /beep std_msgs/msg/UInt16 "data: 0"
publisher: beginning loop
publishing #1: std_msgs.msg.UInt16(data=0)

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

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

3.2. Publish speed control information

We assume that the published car moves at a linear speed of 22.5 and an angular speed of 200.0. Input in the terminal,

```
#Linear speed range: 0~60.0
#Angular speed range: 0~1000.0
```

```
ros2 topic pub /cmd_vel_bl geometry_msgs/msg/Twist "{linear: {x: 22.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 200.0}}"
```



```
yahboom@yahboom-VM:~$ ros2 topic pub /cmd_vel_bl geometry_msgs/msg/Twist "{linear: {x: 22.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 200.0}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=22.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=200.0))

publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=22.5, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=200.0))
```

If the car stops, input,

```
ros2 topic pub /cmd_vel_bl geometry_msgs/msg/Twist "{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}" ``` ![image-20241227145603865](image-20241227145603865.png)
```