Robot information release

1. Program function description

After the car starts the chassis drive, it will publish the data of sensor modules such as ultrasonic and four-way patrol. You can run commands in docker to query this information, and you can also publish the control data of sensors such as speed, buzzer, RGB light bar, etc.

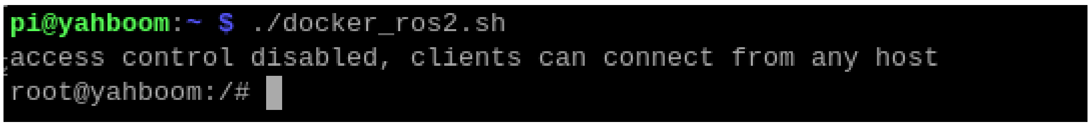
2. Query car information

2.1. Enter the car docker

Open a terminal and enter the following command to enter docker,

```
./docker_ros2.sh
```

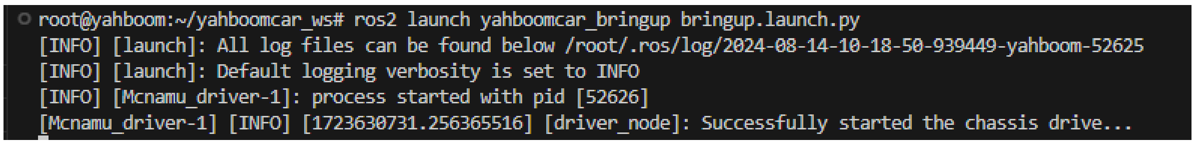
The following interface appears, indicating that you have successfully entered docker. Now you can control the car through commands.



```
pi@yahboom:~ $ ./docker_ros2.sh
access control disabled, clients can connect from any host
root@yahboom:/#
```

```
ros2 launch yahboomcar_bringup bringup.launch.py
```

Start the chassis drive, as shown in the figure below,

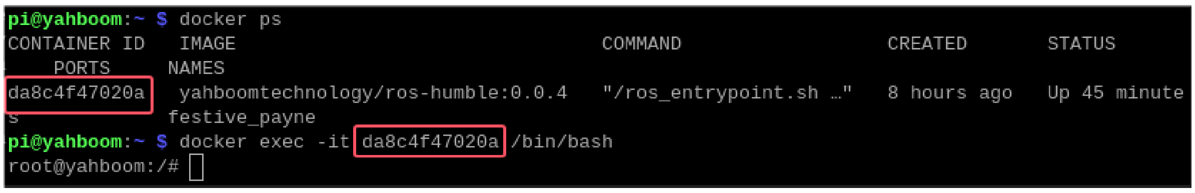


```
root@yahboom:~/yahboomcar_ws# ros2 launch yahboomcar_bringup bringup.launch.py
[INFO] [launch]: All log files can be found below /root/.ros/log/2024-08-14-10-18-50-939449-yahboom-52625
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [Mcnamu_driver-1]: process started with pid [52626]
[Mcnamu_driver-1] [INFO] [1723630731.256365516] [driver_node]: Successfully started the chassis drive...
```

Open a new terminal and enter the same docker. Change the following da8c4f47020a to the ID displayed in the actual terminal.

```
docker ps
```

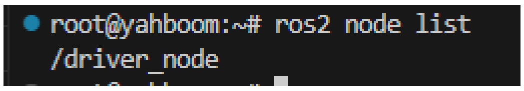
```
docker exec -it da8c4f47020a   /bin/bash
```



```
pi@yahboom:~ $ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
    PORTS     NAMES
da8c4f47020a   yahboomtechnology/ros-humble:0.0.4   "/ros_entrypoint.sh …"   8 hours ago    Up 45 minute
s             festive_payne
pi@yahboom:~ $ docker exec -it da8c4f47020a /bin/bash
root@yahboom:/#
```

Enter the following command to query the node,

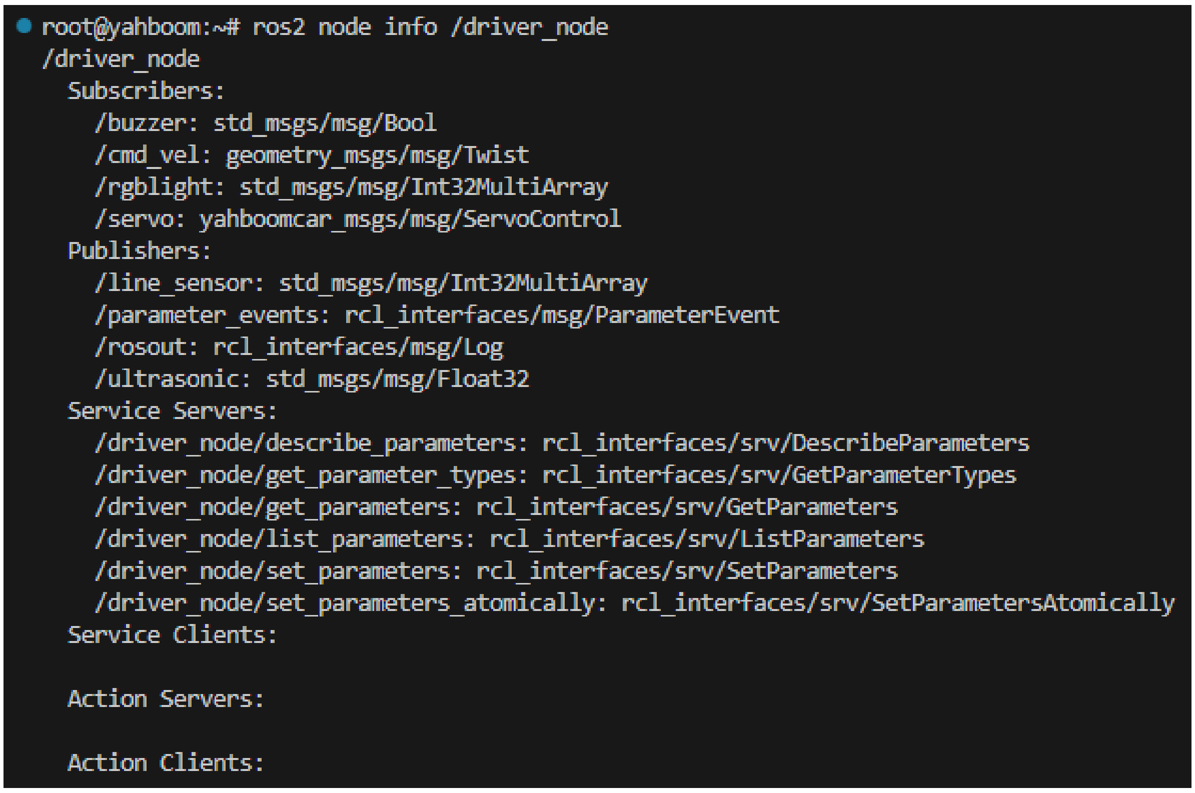
```
ros2 node list
```



```
root@yahboom:~# ros2 node list
/driver_node
```

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

```
ros2 node info /driver_node
```



```
root@yahboom:~# ros2 node info /driver_node
/driver_node
  Subscribers:
    /buzzer: std_msgs/msg/Bool
    /cmd_vel: geometry_msgs/msg/Twist
    /rgblight: std_msgs/msg/Int32MultiArray
    /servo: yahboomcar_msgs/msg/ServoControl
  Publishers:
    /line_sensor: std_msgs/msg/Int32MultiArray
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /ultrasonic: std_msgs/msg/Float32
  Service Servers:
    /driver_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /driver_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /driver_node/get_parameters: rcl_interfaces/srv/GetParameters
    /driver_node/list_parameters: rcl_interfaces/srv/ListParameters
    /driver_node/set_parameters: rcl_interfaces/srv/SetParameters
    /driver_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
```

It can be seen that the subscribed topics are,

/buzzer: buzzer control

/cmd_vel: car speed control

/rgblight: RGB light bar control

/servo: s1 servo pan/tilt control, s2 servo pan/tilt control

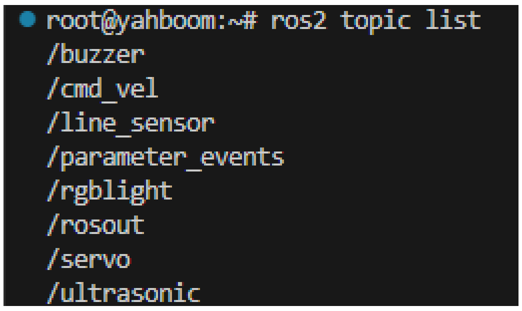
The published topics are,

/ultrasonic: ultrasonic module data

/line_sensor: four-way line patrol module data

We can also query the topic command, terminal input,

```
ros2 topic list
```

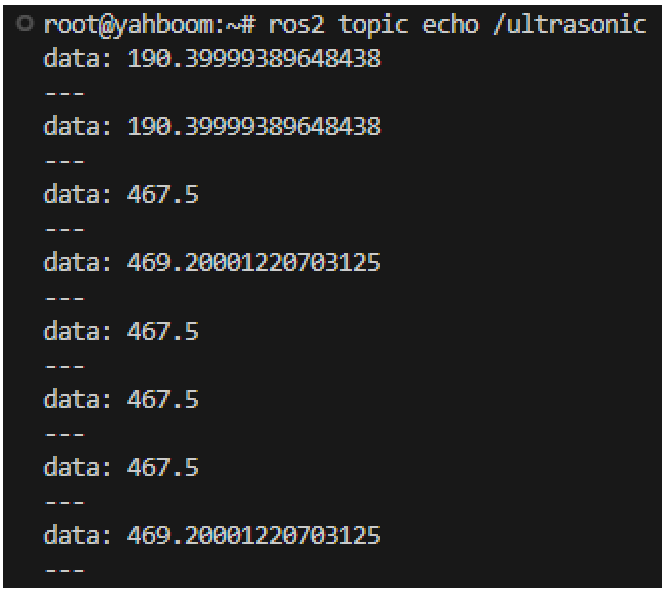


```
root@yahboom:~# ros2 topic list
/buzzer
/cmd_vel
/line_sensor
/parameter_events
/rgblight
/rosout
/servo
/ultrasonic
```

2.3. Query topic data

Query ultrasonic data,

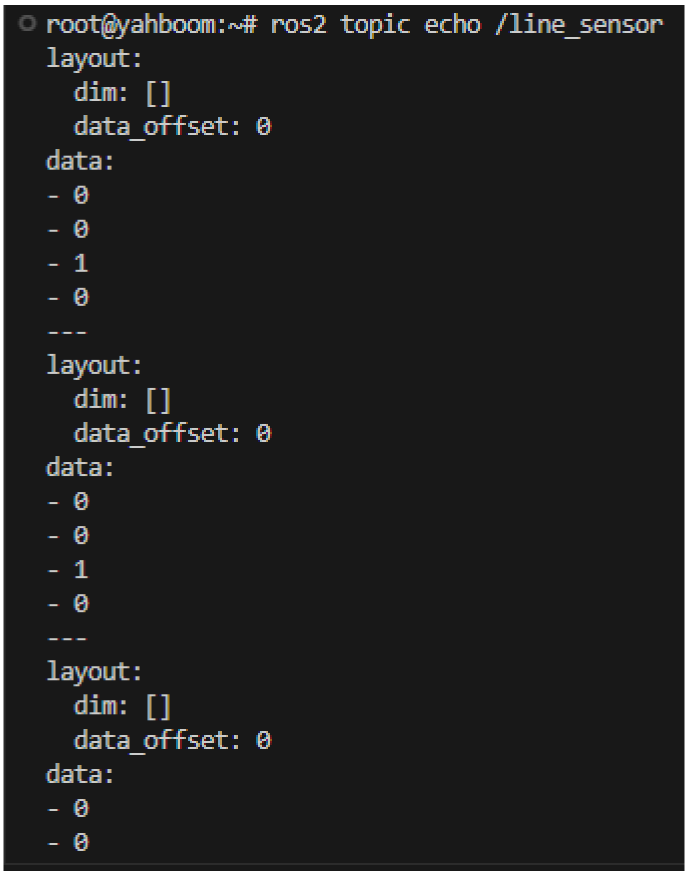
```
ros2 topic echo /ultrasonic
```



```
root@yahboom:~# ros2 topic echo /ultrasonic
data: 190.39999389648438
---
data: 190.39999389648438
---
data: 467.5
---
data: 469.20001220703125
---
data: 467.5
---
data: 467.5
---
data: 467.5
---
data: 469.20001220703125
---
```

Query the data of the four-way line patrol module.

```
ros2 topic echo /line_sensor
```



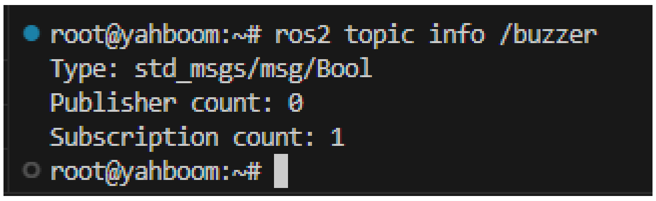
```
root@yahboom:~# ros2 topic echo /line_sensor
layout:
  dim: []
  data_offset: 0
data:
- 0
- 0
- 1
- 0
---
layout:
  dim: []
  data_offset: 0
data:
- 0
- 0
- 1
- 0
---
layout:
  dim: []
  data_offset: 0
data:
- 0
- 0
```

3. Publish car control information

3.1. Control the buzzer

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

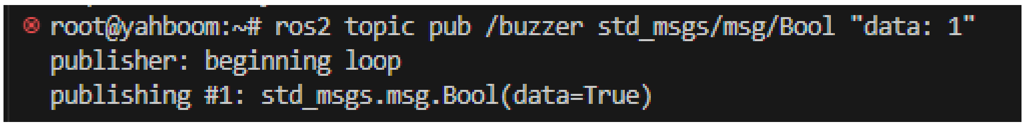
```
ros2 topic info /buzzer
```



```
root@yahboom:~# ros2 topic info /buzzer
Type: std_msgs/msg/Bool
Publisher count: 0
Subscription count: 1
root@yahboom:~# 
```

The data type is std_msgs/msg/Bool. Then enter the following command to turn on the buzzer, and enter in the terminal,

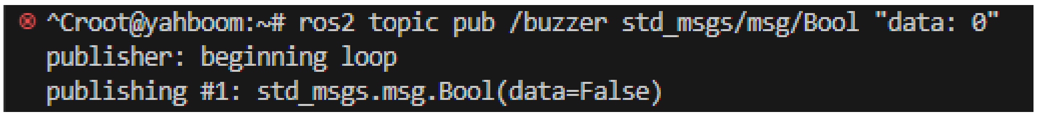
```
ros2 topic pub /buzzer std_msgs/msg/Bool "data: 1"
```



```
root@yahboom:~# ros2 topic pub /buzzer std_msgs/msg/Bool "data: 1"
publisher: beginning loop
publishing #1: std_msgs.msg.Bool(data=True)
```

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

```
ros2 topic pub /buzzer std_msgs/msg/Bool "data: 0"
```



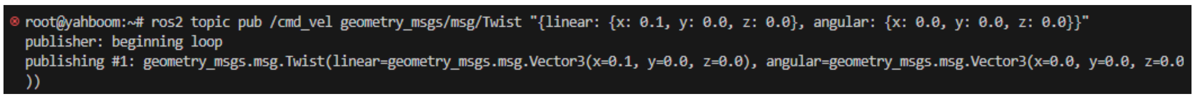
```
^Croot@yahboom:~# ros2 topic pub /buzzer std_msgs/msg/Bool "data: 0"
publisher: beginning loop
publishing #1: std_msgs.msg.Bool(data=False)
```

3.2. Release speed control information

We assume that the speed of the car is 0.1 of the linear speed, and the terminal input is,

```
#Valid value range
#linear:     x[-1,1] y[-1,1] z
#angular:    x       y       z[-3,3]
```

```
ros2 topic pub -1 /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.1, y: 0.0, z:
0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```



```
root@yahboom:~# ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.1, 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=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0
))
```

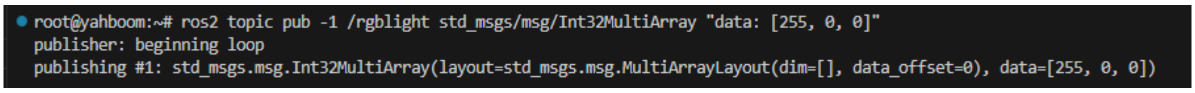
If parking, input

```
ros2 topic pub -1 /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}}"
```

3.3. Control RGB light bar

We assume that the published RGB value is 255,0,0, and the terminal input is

```
ros2 topic pub -1 /rgblight std_msgs/msg/Int32MultiArray "data: [255, 0, 0]"
```



```
root@yahboom:~# ros2 topic pub -1 /rgblight std_msgs/msg/Int32MultiArray "data: [255, 0, 0]"
publisher: beginning loop
publishing #1: std_msgs.msg.Int32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[255, 0, 0])
```

Turn off the light and enter

```
ros2 topic pub -1 /rgblight std_msgs/msg/Int32MultiArray "data: [0, 0, 0]"
```

3.4. Control the PTZ Servo

It should be noted that the range of the s1 servo is [0,180], and the range of the s2 servo is [0,110]. If the value exceeds the range, the servo will not rotate.

We assume that the s1 servo is controlled to rotate to 90 degrees and the s2 servo is controlled to rotate to 25 degrees. Then the terminal input is,

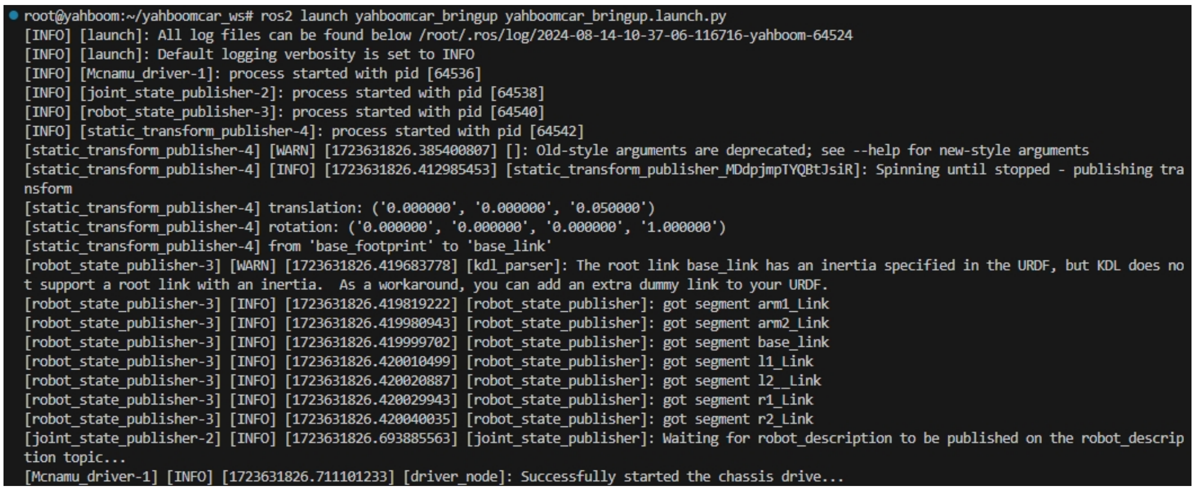
```
ros2 topic pub -1 /servo yahboomcar_msgs/msg/ServoControl "{'servo_s1': 90, 'servo_s2': 25}"
```

3.5. Complete chassis driver

3.5.1. Startup program

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

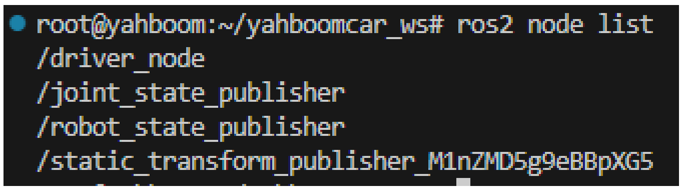
Start the chassis drive, as shown below.



```
root@yahboom:~/yahboomcar_ws# ros2 launch yahboomcar_bringup yahboomcar_bringup.launch.py
[INFO] [launch]: All log files can be found below /root/.ros/log/2024-08-14-10-37-06-116716-yahboom-64524
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [Mcnamu_driver-1]: process started with pid [64536]
[INFO] [joint_state_publisher-2]: process started with pid [64538]
[INFO] [robot_state_publisher-3]: process started with pid [64540]
[INFO] [static_transform_publisher-4]: process started with pid [64542]
[static_transform_publisher-4] [WARN] [1723631826.385400807] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-4] [INFO] [1723631826.412985453] [static_transform_publisher_MDdpjmpTYQBtJsiR]: Spinning until stopped - publishing tra
nsform
[static_transform_publisher-4] translation: ('0.000000', '0.000000', '0.050000')
[static_transform_publisher-4] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-4] from 'base_footprint' to 'base_link'
[robot_state_publisher-3] [WARN] [1723631826.419683778] [kdl_parser]: The root link base_link has an inertia specified in the URDF, but KDL does no
t support a root link with an inertia.  As a workaround, you can add an extra dummy link to your URDF.
[robot_state_publisher-3] [INFO] [1723631826.419819222] [robot_state_publisher]: got segment arm1_Link
[robot_state_publisher-3] [INFO] [1723631826.419980943] [robot_state_publisher]: got segment arm2_Link
[robot_state_publisher-3] [INFO] [1723631826.419999702] [robot_state_publisher]: got segment base_link
[robot_state_publisher-3] [INFO] [1723631826.420010499] [robot_state_publisher]: got segment l1_Link
[robot_state_publisher-3] [INFO] [1723631826.420020887] [robot_state_publisher]: got segment l2__Link
[robot_state_publisher-3] [INFO] [1723631826.420029943] [robot_state_publisher]: got segment r1_Link
[robot_state_publisher-3] [INFO] [1723631826.420040035] [robot_state_publisher]: got segment r2_Link
[joint_state_publisher-2] [INFO] [1723631826.693885563] [joint_state_publisher]: Waiting for robot_description to be published on the robot_descrip
tion topic...
[Mcnamu_driver-1] [INFO] [1723631826.711101233] [driver_node]: Successfully started the chassis drive...
```

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

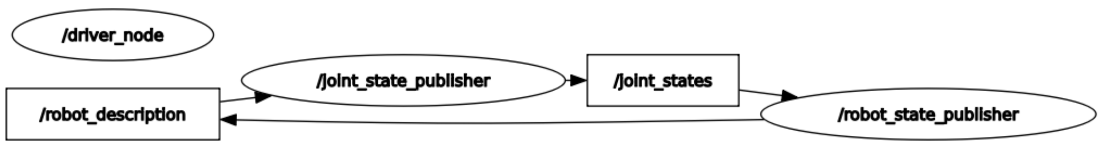
```
ros2 node list
```



```
root@yahboom:~/yahboomcar_ws# ros2 node list
/driver_node
/joint_state_publisher
/robot_state_publisher
/static_transform_publisher_M1nZMD5g9eBBpXG5
```

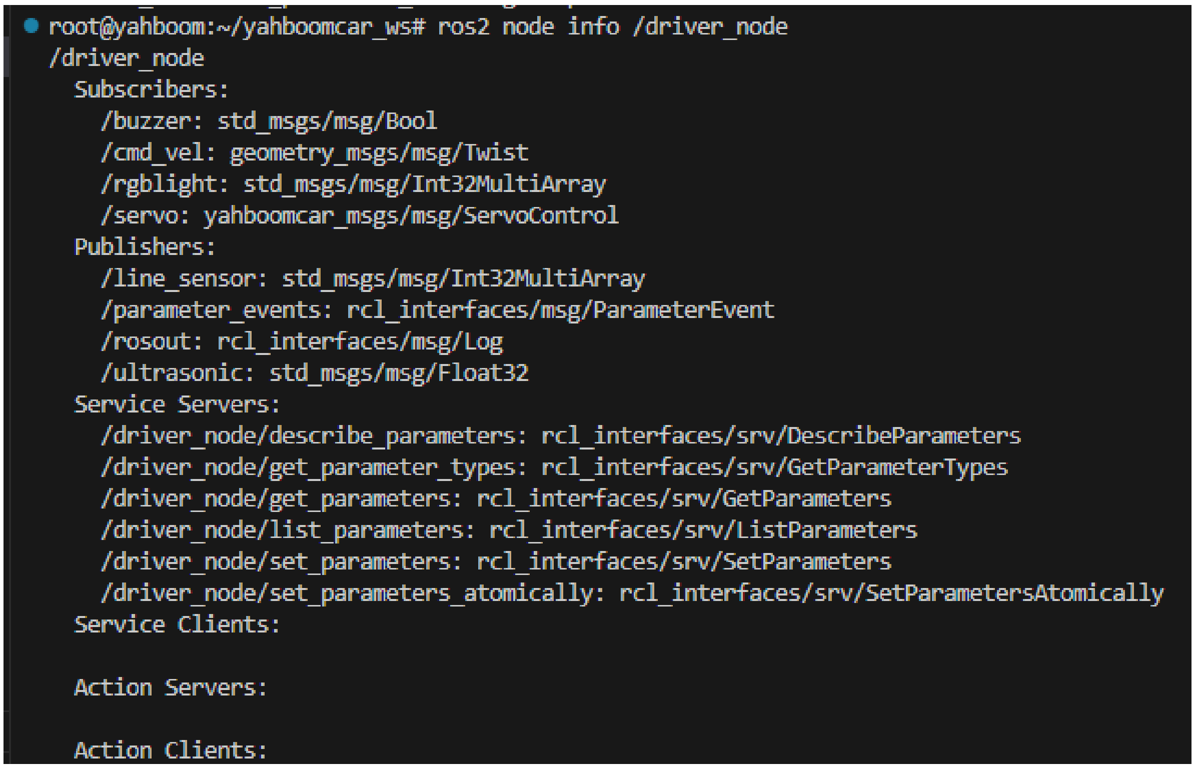
Enter the following command to view the communication diagram between nodes.

```
ros2 run rqt_graph rqt_graph
```



Enter the command to view the relevant information of the node

```
ros2 node info /driver_node
```



```
root@yahboom:~/yahboomcar_ws# ros2 node info /driver_node
/driver_node
  Subscribers:
    /buzzer: std_msgs/msg/Bool
    /cmd_vel: geometry_msgs/msg/Twist
    /rgblight: std_msgs/msg/Int32MultiArray
    /servo: yahboomcar_msgs/msg/ServoControl
  Publishers:
    /line_sensor: std_msgs/msg/Int32MultiArray
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /ultrasonic: std_msgs/msg/Float32
  Service Servers:
    /driver_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /driver_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /driver_node/get_parameters: rcl_interfaces/srv/GetParameters
    /driver_node/list_parameters: rcl_interfaces/srv/ListParameters
    /driver_node/set_parameters: rcl_interfaces/srv/SetParameters
    /driver_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
```

3.5.2. Parsing launch files

```python
from ament_index_python.packages import get_package_share_path

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.conditions import IfCondition, UnlessCondition
from launch.substitutions import Command, LaunchConfiguration

from launch_ros.actions import Node
from launch_ros.parameter_descriptions import ParameterValue
```

```
import os
from ament_index_python.packages import get_package_share_directory

from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():
    # 配置参数

    description_launch = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
        get_package_share_directory('yahboomcar_description'), 'launch'),
         '/description_launch.py'])
    )

    driver_node = Node(
        package='yahboomcar_bringup',
        executable='Mcnamu_driver',
    )

    # 返回LaunchDescription对象
    return LaunchDescription([
        driver_node,
        description_launch
    ])
```

- The launch file starts the following nodes:
    - driver_node: chassis driver data node;
    - description_launch: load URDF model.