2. Robot information release

2. Robot information release
- 2.1 Description of program function
- 2.2. Program Code reference path
- 2.3. Program Startup
  - 2.3.1. Launch command
  - 2.3.2 View node topics
  - 2.3.3. Reading topic data
  - 2.3.4. Publish topic data
- 2.4. Core Source Code Analysis

Note: In subsequent ROS2 related courses, SSH remote connection car will be used for operation, refer to the "Remote connection" section of the tutorial "Development Environment Construction".

2.1 Description of program function

After the program is run, combined with the ROS expansion board, the sensor information on the ROS expansion board can be obtained, control car movement, control light belt, buzzer and other functions.

2.2. Program Code reference path

After SSH connection car, the location of the function source code is located at,

```
/userdata/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

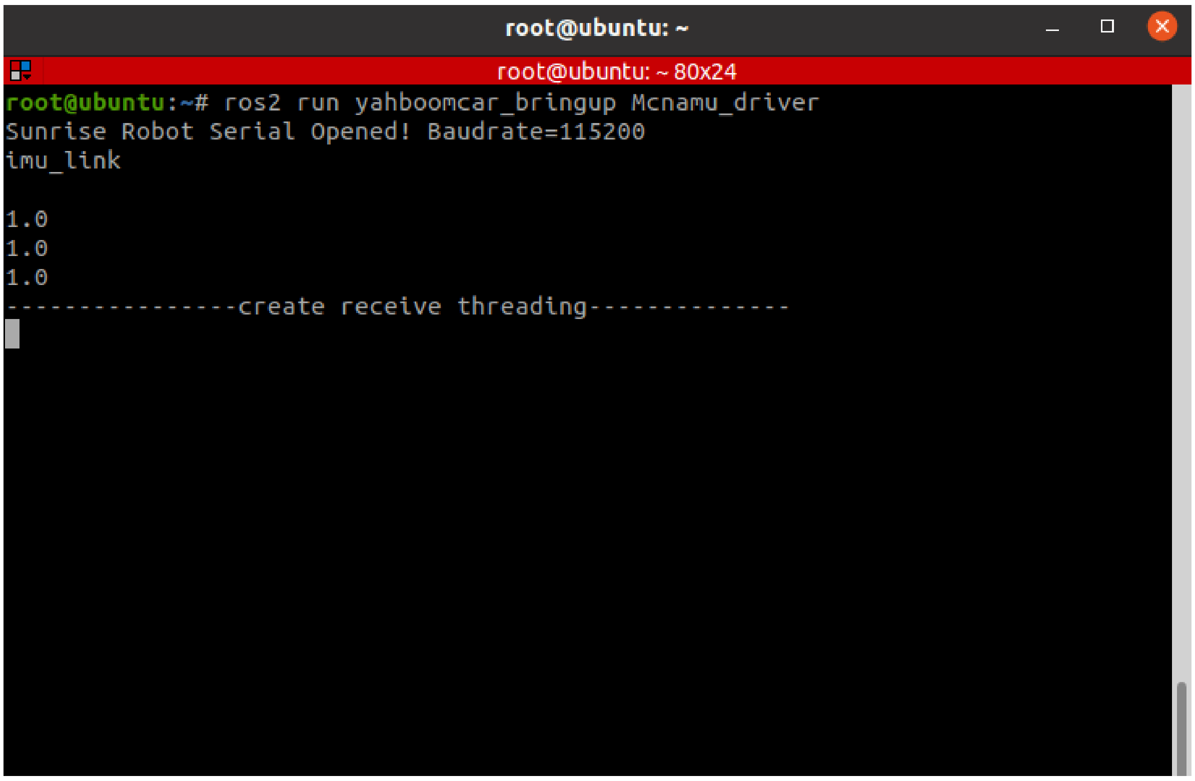
Where **Mcnamu_driver.py** indicates the chassis driver code.

2.3. Program Startup

2.3.1. Launch command

After SSH connects to the car, type,

```
ros2 run yahboomcar_bringup Mcnamu_driver
```



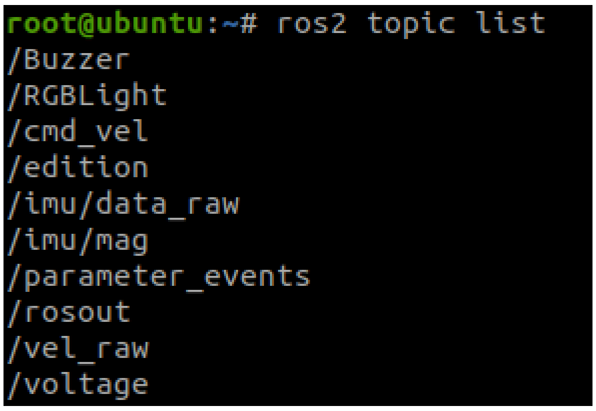
```
root@ubuntu:~# ros2 run yahboomcar_bringup Mcnamu_driver
Sunrise Robot Serial Opened! Baudrate=115200
imu_link

1.0
1.0
1.0
----------------create receive threading-------------
```

2.3.2 View node topics

Open terminal input,

```
ros2 topic list
```



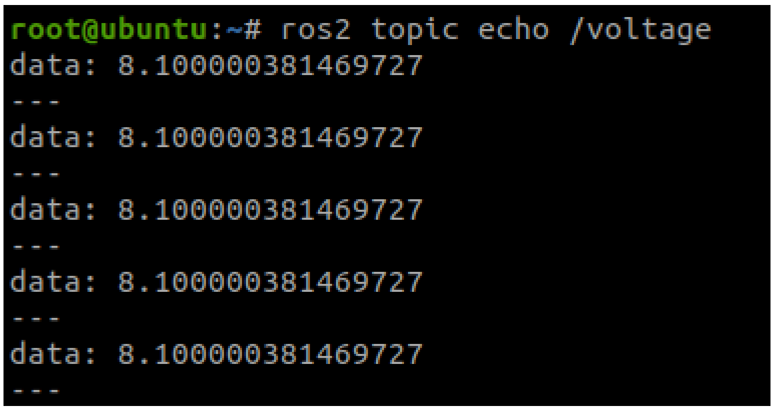
```
root@ubuntu:~# ros2 topic list
/Buzzer
/RGBLight
/cmd_vel
/edition
/imu/data_raw
/imu/mag
/parameter_events
/rosout
/vel_raw
/voltage
```

The topic name	The topic content
/Buzzer	The buzzer
/RGBLight	Lighting Effects Control
/cmd_vel	Speed control
/edition	Version information
/imu/data_raw	The IMU sensor data
/imu/mag	IMU magnetometer data
/vel_raw	Car speed information
/voltage	Battery voltage information

2.3.3. Reading topic data

Take reading the voltage size as an example, open the terminal input,

```
ros2 topic echo /voltage
```

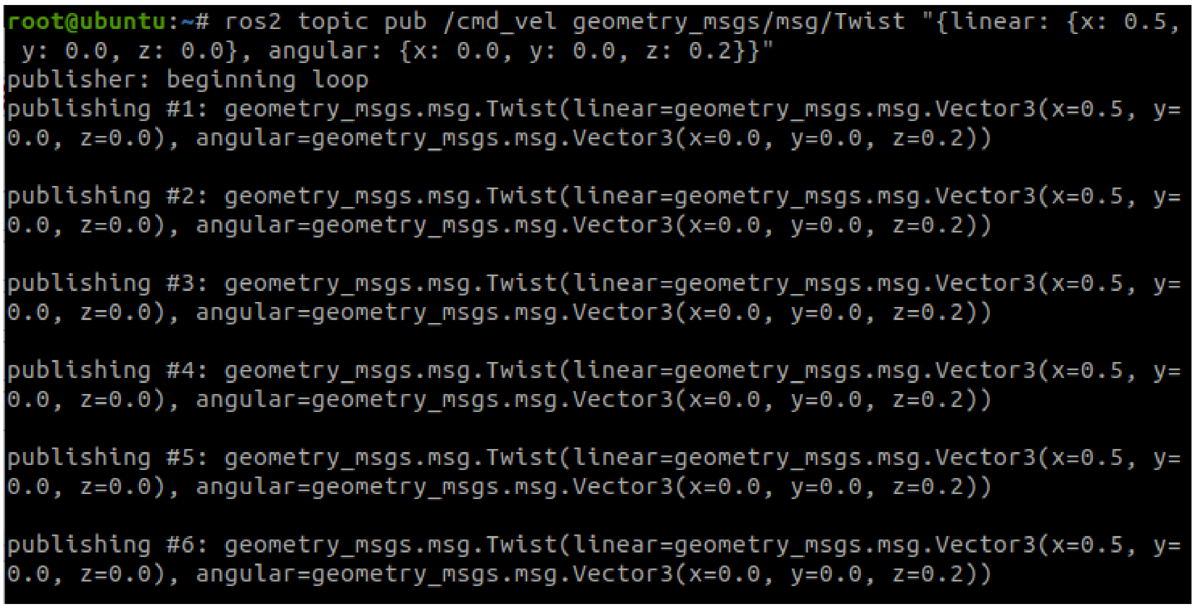


```
root@ubuntu:~# ros2 topic echo /voltage
data: 8.100000381469727
---
data: 8.100000381469727
---
data: 8.100000381469727
---
data: 8.100000381469727
---
data: 8.100000381469727
---
```

2.3.4. Publish topic data

Take publishing /cmd_vel data to control car movement as an example, open the terminal input,

```
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}}"
```



```
root@ubuntu:~# 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))

publishing #5: 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 #6: 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))
```

> [Note] The system has configured the communication between the PC side (virtual machine side) and the car side, so after the car program runs, the virtual machine side can also participate in the receiving or sending of ros topic data. So as long as the communication is normal, there is no need to remotely connect the car after opening the terminal. ros2 node/topic/... Such commands can be run directly on the virtual machine terminal.

2.4. Core Source Code Analysis

Take Mcnamu_driver.py as an example.

```
from SunriseRobotLib import SunriseRobot    #导入驱动库# Import driver library

self.car = SunriseRobot()   #实例化SunriseRobot对象
```

```
# Instantiate the SunriseRobot object
#create subcriber 创建订阅者
self.sub_cmd_vel =
self.create_subscription(Twist,"cmd_vel",self.cmd_vel_callback,1)
self.sub_RGBLight =
self.create_subscription(Int32,"RGBLight",self.RGBLightcallback,100)
self.sub_BUzzer =
self.create_subscription(Bool,"Buzzer",self.Buzzercallback,100)

#create publisher 创建发布者
self.EdiPublisher = self.create_publisher(Float32,"edition",100)
self.volPublisher = self.create_publisher(Float32,"voltage",100)
self.velPublisher = self.create_publisher(Twist,"vel_raw",50)
self.imuPublisher = self.create_publisher(Imu,"/imu/data_raw",100)
self.magPublisher = self.create_publisher(MagneticField,"/imu/mag",100)

#调用库，读取ros拓展板的信息# Call the library to read the ros extension board
information
edition.data = self.car.get_version()
battery.data = self.car.get_battery_voltage()
ax, ay, az = self.car.get_accelerometer_data()
gx, gy, gz = self.car.get_gyroscope_data()
mx, my, mz = self.car.get_magnetometer_data()
vx, vy, angular = self.car.get_motion_data()

#publish 发布话题数据#publish Publish the topic data
self.EdiPublisher.publish(edition)
self.volPublisher.publish(battery)
self.imuPublisher.publish(imu)
self.magPublisher.publish(mag)
self.velPublisher.publish(twist)

#callback 订阅者回调函数#callback subscriber callback function
def cmd_vel_callback(self,msg):
def RGBLightcallback(self,msg):
def Buzzercallback(self,msg):
```

For details, refer to Mcnamu_driver.py.