2. Robot information release

Note: In the subsequent ROS2 related courses, SSH remote connection to the car is applicable for operation. Please refer to the [Remote Connection] section in the tutorial [Development Environment Construction].

2.1. Program function description

After the program is run, combined with the ROS expansion board, you can obtain the information of the sensors on the ROS expansion board, control the movement of the car, control the light strip, buzzer and other functions.

2.2. Program code reference path

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

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

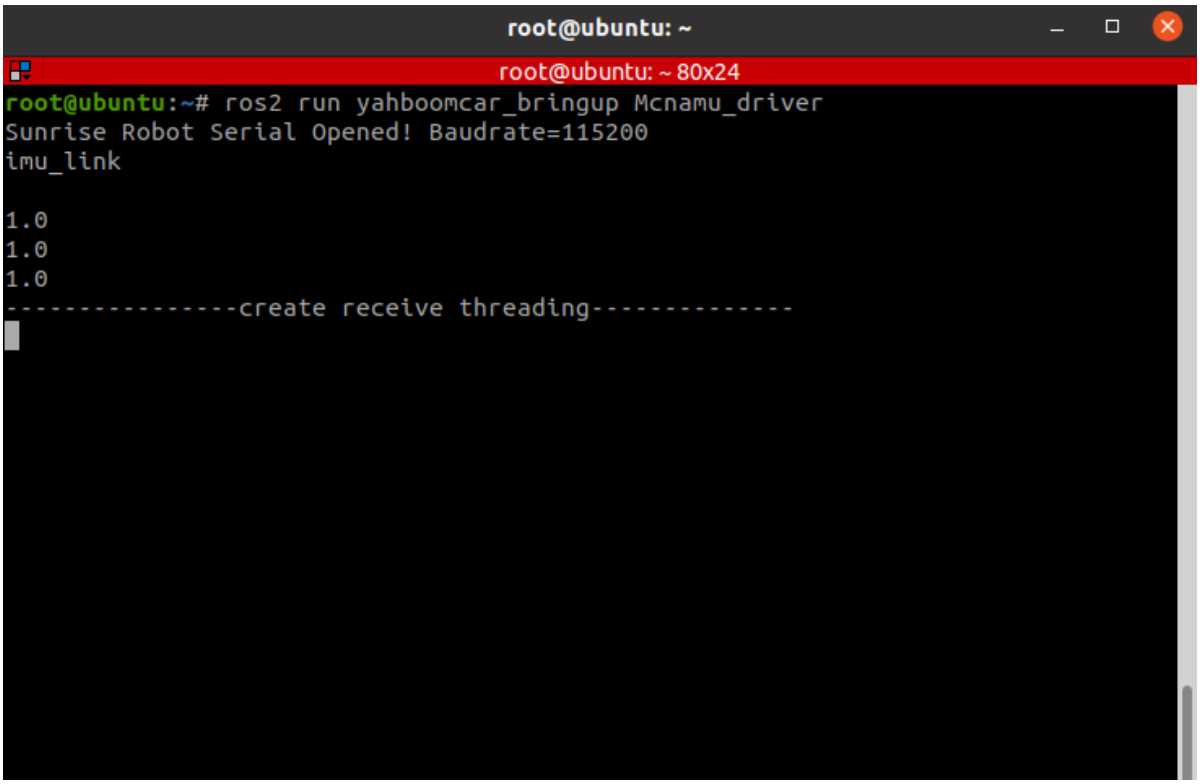
Among them, **Mcnamu_driver.py** is the chassis driver code.

2.3, Program startup

2.3.1, Startup command

After SSH connects to the car, enter,

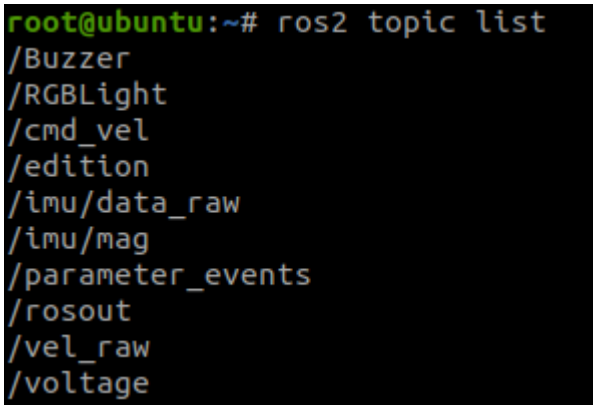
```
ros2 run yahboomcar_bringup Mcnamu_driver
```



2.3.2, View node topics

Open the terminal and enter,

```
ros2 topic list
```



Topic name	Topic content
/Buzzer	Buzzer
/RGBLight	Lighting effect control
/cmd_vel	Speed control
/edition	Version information
/imu/data_raw	IMU sensor data
/imu/mag	IMU magnetometer data
/vel_raw	Car speed information
/voltage	Battery voltage information

2.3.3, read topic data

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

```
ros2 topic echo /voltage

![1684461698432](1684461698432.png)

### 2.3.4, publish topic data

Take publishing /cmd_vel data to control the movement of the car 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}}"

```



> 【Note】The system has configured the communication between the PC (virtual machine) and the car, so after the car program is running, the virtual machine can also participate in the reception or transmission of ros topic data. Therefore, as long as the communication is normal, there is no need to remotely connect to the car after opening the terminal, and ros2 node/topic/... and other commands can be run directly in the virtual machine terminal.

## 2.4, core source code analysis

Take Mcnamu_driver.py as an example,

```python
from SunriseRobotLib import SunriseRobot #Import driver library

self.car = SunriseRobot() #Instantiate SunriseRobot object

#create subcriber Create subscriber
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 Create publisher
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) #Call the library and read
the information of the ros expansion board 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 Publish topic data
```
```

```
self.EdiPublisher.publish(edition)
self.volPublisher.publish(battery)
self.imuPublisher.publish(imu)
self.magPublisher.publish(mag)
self.velPublisher.publish(twist)

#callback Subscriber callback function
def cmd_vel_callback(self,msg):
def RGBLightcallback(self,msg):
def Buzzercallback(self,msg):
```

For detailed code, please refer to the code Mcnamu_driver.py.