2. Robot information release

1. Program function description

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

2. Program code reference path

After entering the docker container, the location of the source code of this function is:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

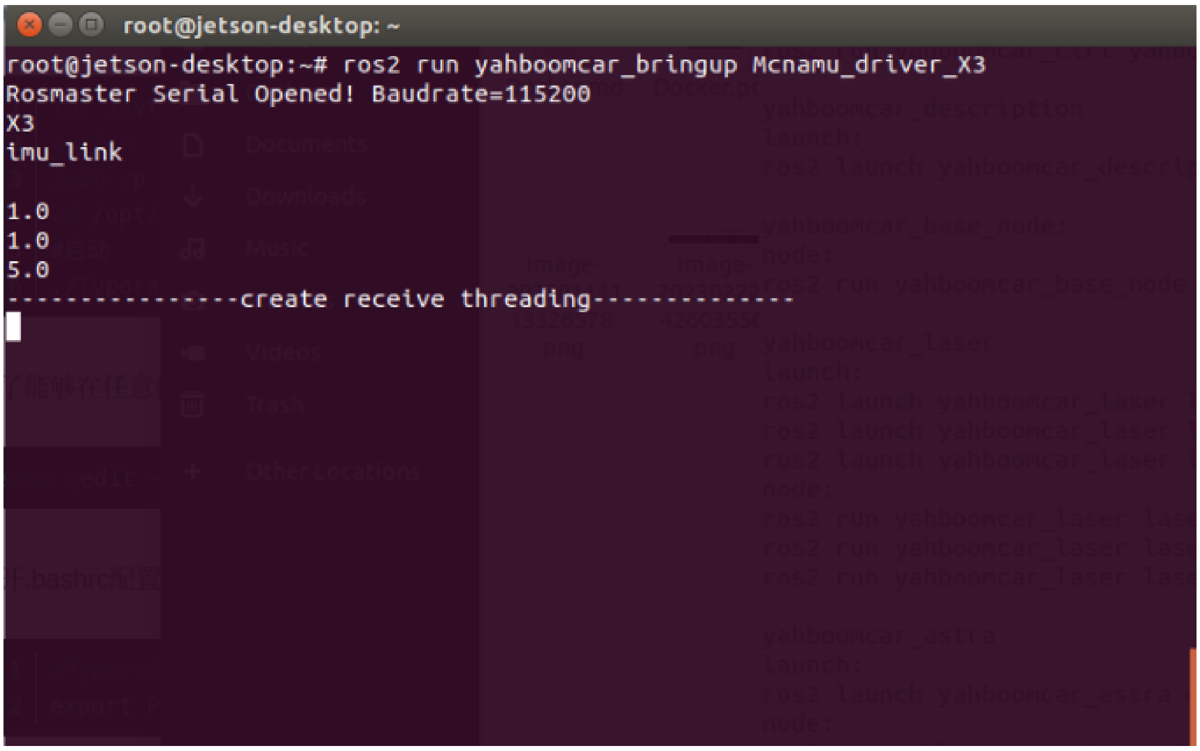
Taking the X3 model as an example, **Mcnamu_driver_X3.py** is the chassis driver code of the X3 model.

3. Program startup

3.1. Start command

After entering the docker container, enter the terminal according to the actual car model,

```
ros2 run yahboomcar_bringup Mcnamu_driver_X3 #X3 model
```



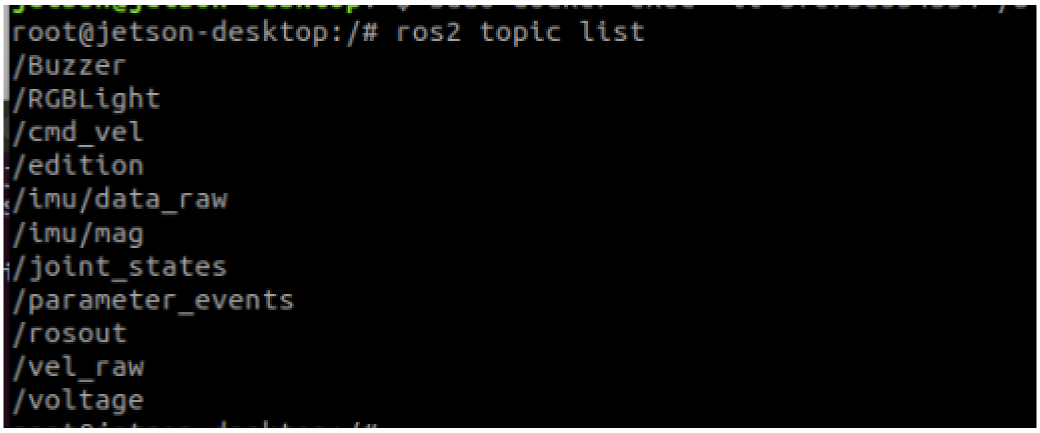
```
root@jetson-desktop:~# ros2 run yahboomcar_bringup Mcnamu_driver_X3
Rosmaster Serial Opened! Baudrate=115200
X3
imu_link

1.0
1.0
5.0
----------------create receive threading-------------
```

3.2. View node topics

Open the terminal and enter the container and enter,

```
ros2 topic list
```



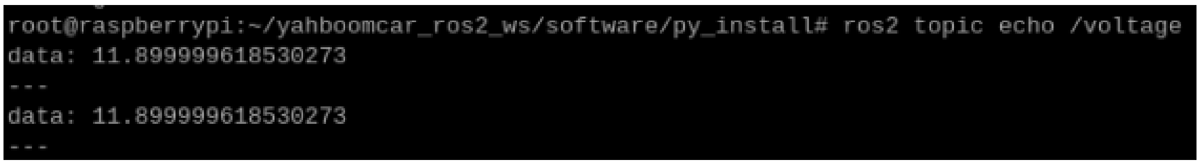
```
root@jetson-desktop:/# ros2 topic list
/Buzzer
/RGBLight
/cmd_vel
/edition
/imu/data_raw
/imu/mag
/joint_states
/parameter_events
/rosout
/vel_raw
/voltage
```

Topic name	Topic content
/Buzzer	Buzzer
/RGBLight	Light strip 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

3.3. Read topic data

Open the terminal and enter the container and enter, taking reading the voltage as an example,

```
ros2 topic echo /voltage
```

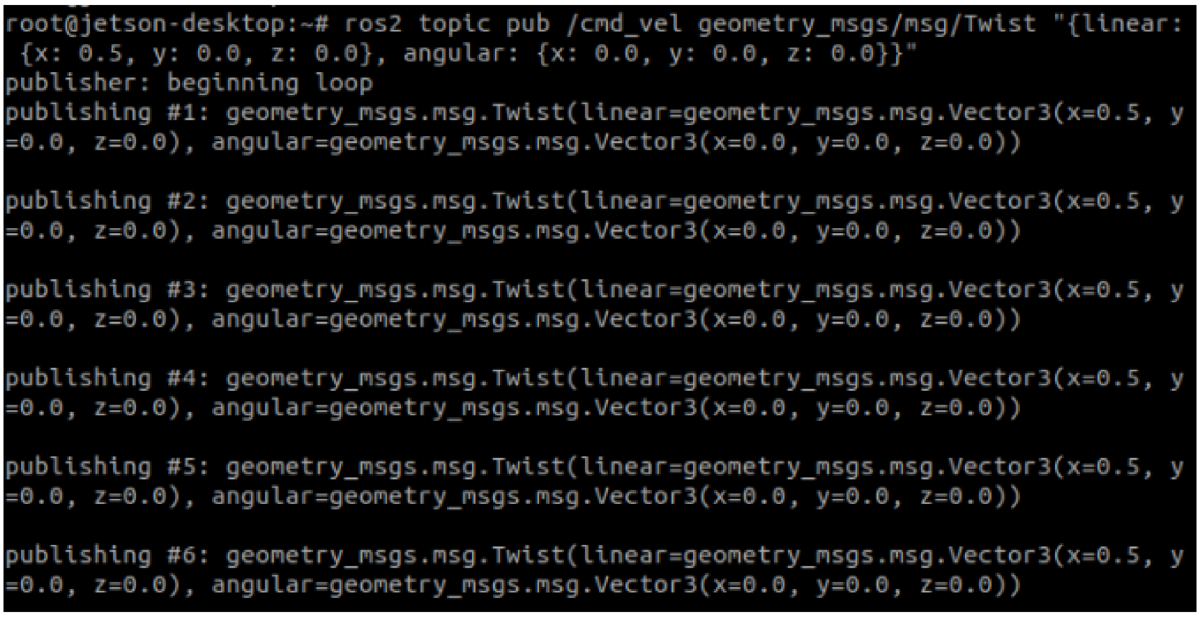


```
root@raspberrypi:~/yahboomcar_ros2_ws/software/py_install# ros2 topic echo /voltage
data: 11.899999618530273
---
data: 11.899999618530273
---
```

3.4. Publish topic data

Open the terminal and enter the container and enter. Take publishing /cmd_vel data to control the movement of the car as an example.

```
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@jetson-desktop:~# 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: 0.0}}"
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=0.0))

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=0.0))

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=0.0))

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=0.0))

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=0.0))

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=0.0))
```

4. Program core source code analysis

Take Mcnamu_driver_X3.py as an example,

```python
from Rosmaster_Lib import Rosmaster #Import driver library

self.car = Rosmaster() #Instantiate Rosmaster object

#create subscriber Create a 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.staPublisher = self.create_publisher(JointState,"joint_states",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()*1.0
battery.data = self.car.get_battery_voltage()*1.0
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 topic data
self.imuPublisher.publish(imu)
self.magPublisher.publish(mag)
self.volPublisher.publish(battery)
self.EdiPublisher.publish(edition)
self.velPublisher.publish(twist)
```

```
#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_X3.py.