ROS Control

1. Function Description

This function enables control of the car's speed, buzzer, and servo motors via a ROS2 topic tool. It also reads low-level data such as radar data, IMU data, and version information.

2. Preparation

2.1 Instructions Before Use

This lesson uses the Raspberry Pi as an example. For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [4.Enter Docker (For JETSON Nano and RPi 5)]**. For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

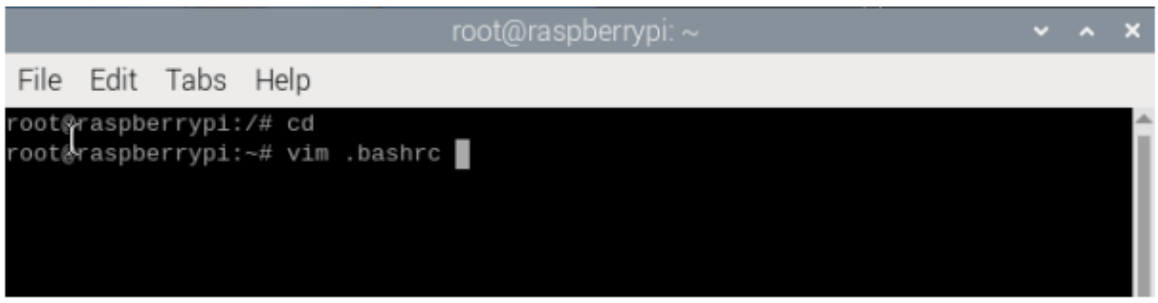
2.2 Configuration Before Use

This vehicle model is equipped with a USB camera, a depth camera, and two different models of LiDAR. However, since the products cannot be automatically recognized, the machine type and LiDAR model need to be manually set.

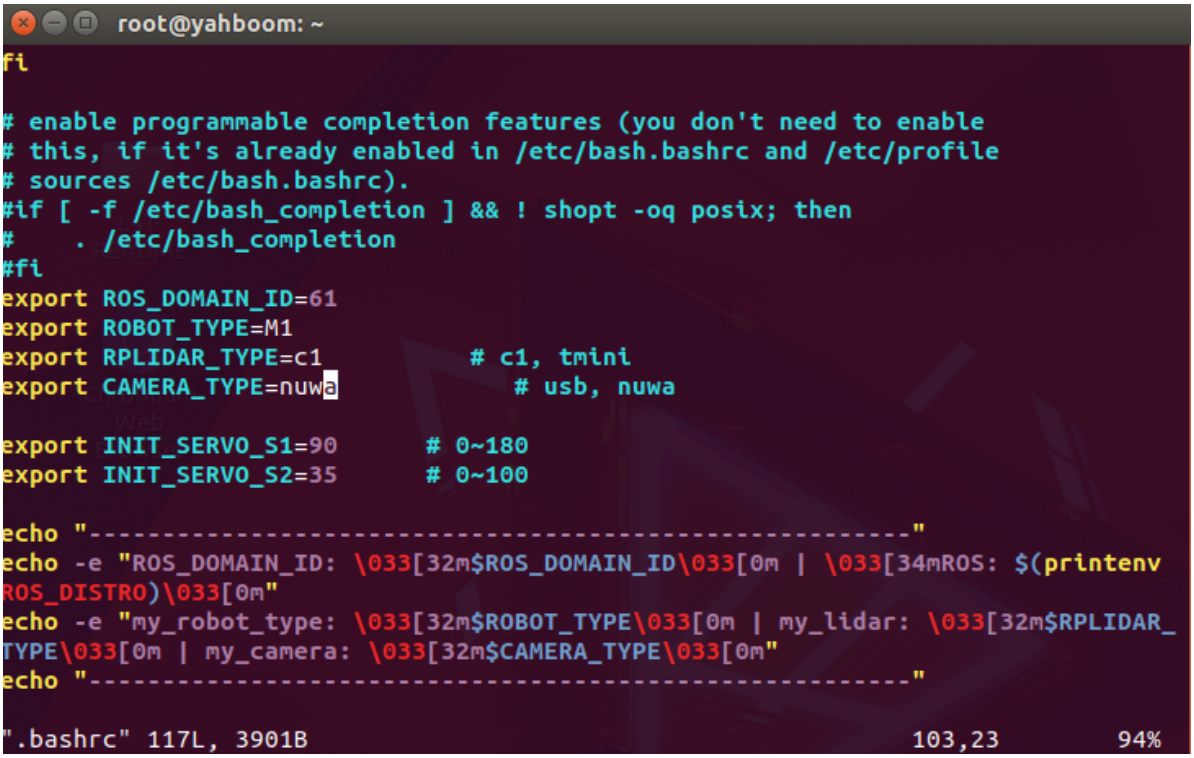
For Raspberry Pi PI5 and jetson nano, you need to enter the Docker container first. For RDKX5 and Orin , this is not necessary.

Based on the vehicle model, the LiDAR type and camera type are modified as follows:

```
root@ubuntu:/#
cd root@ubuntu:~# vim .bashrc
```



Find this location, press the 'i' key on the keyboard, and change it to the corresponding camera and LiDAR model. Here, the default is tmini and nuwa.

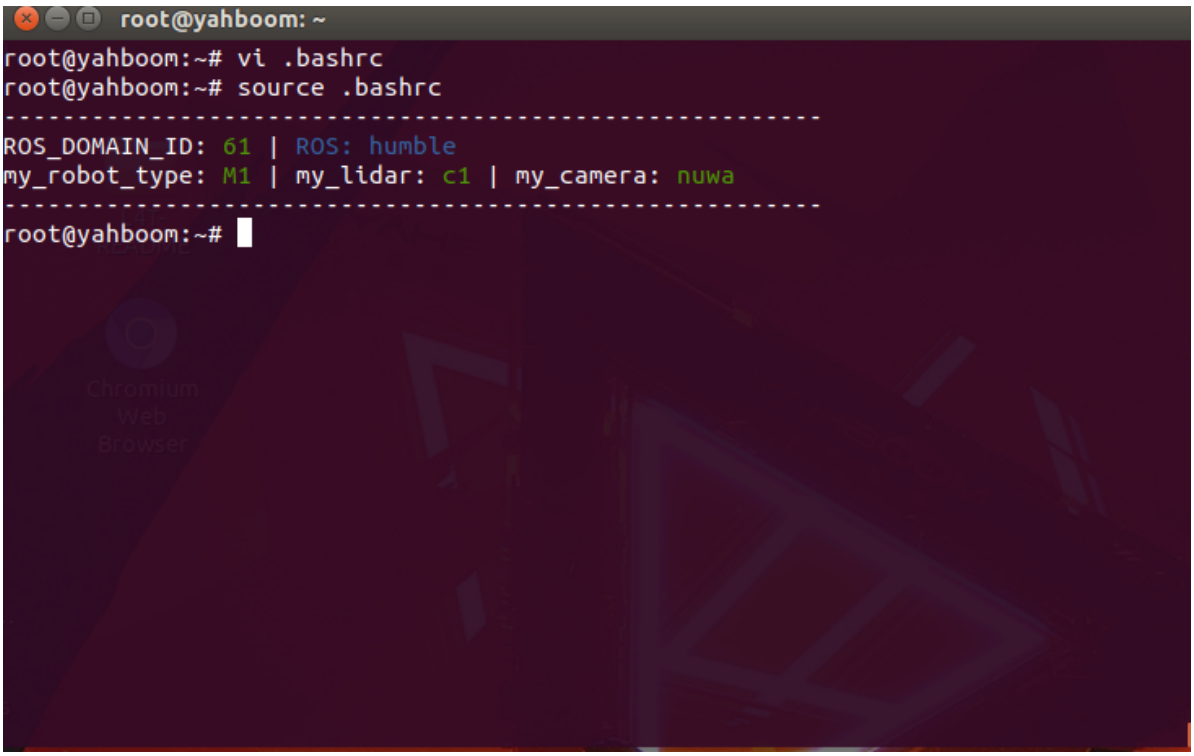


After making the changes, save and exit Vim, then execute:

```
root@raspberrypi:~# source .bashrc

------------------------------------------------------
ROS_DOMAIN_ID: 61 | ROS: humble
my_robot_type: M1 | my_lidar: tmini | my_camera: nuwa
------------------------------------------------------

root@raspberrypi:~#
```



3. Program Startup

3.1 Startup Command

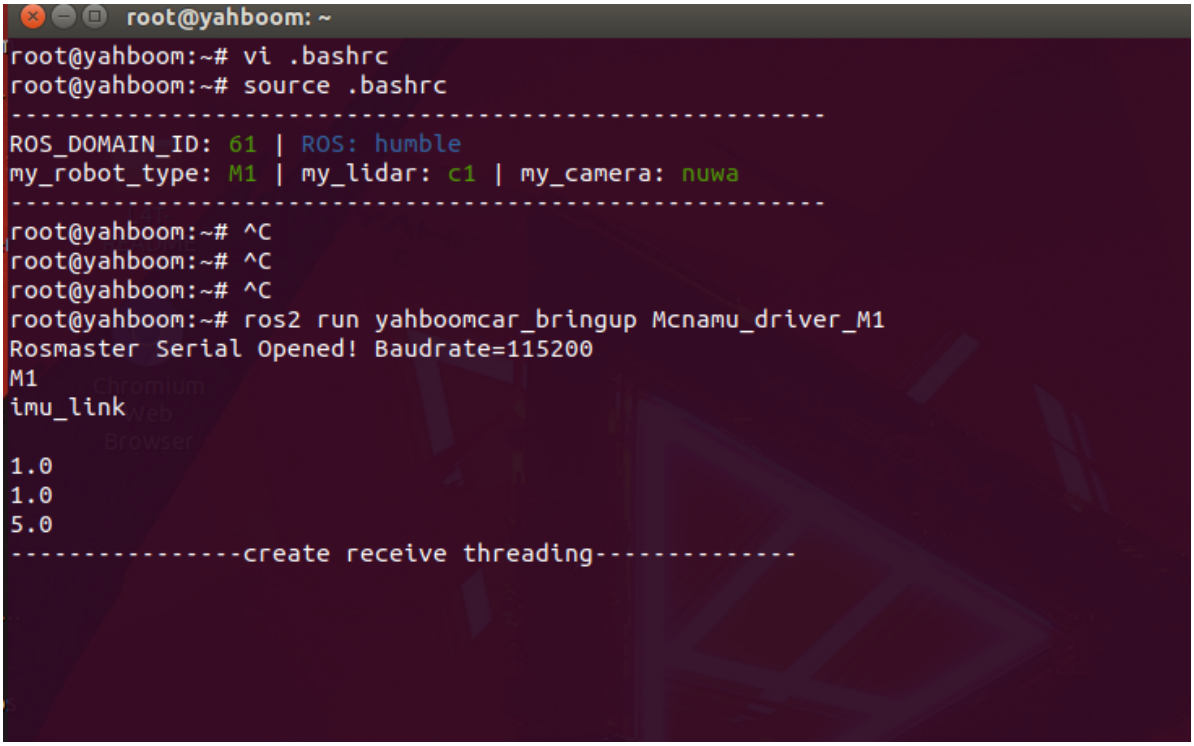
For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

Enter the Docker container (see [Docker course] --- [4. Docker Startup Script] for steps).

All the following commands must be executed within the same Docker container **(see [Docker course] --- [3. Docker Submission and Multi-Terminal Access] for steps).**

To start the chassis data, enter the following in the terminal:

```
ros2 run yahboomcar_bringup Mcnamu_driver_M1
```



You can view the current node name by entering the following command:

```
ros2 node list
```

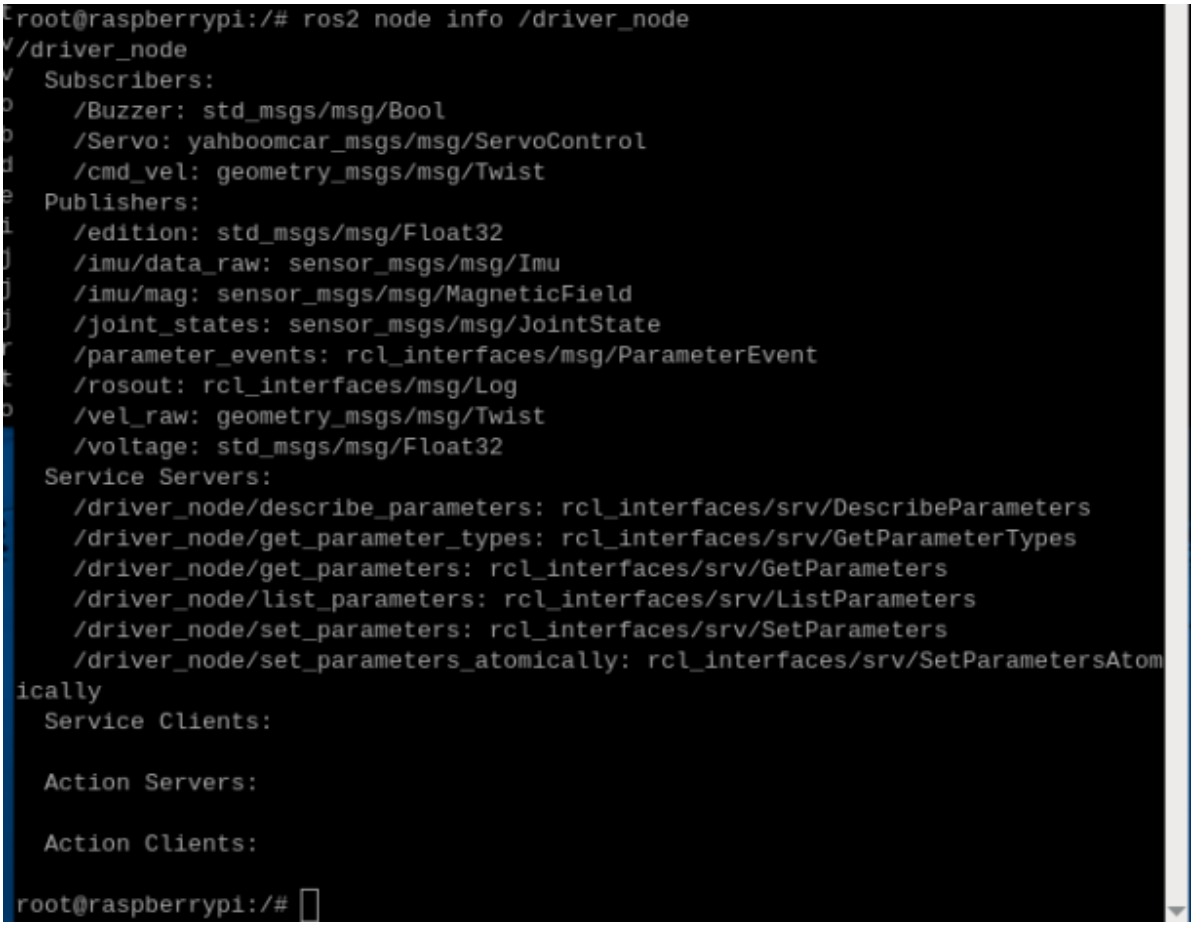
Enter the following command in the terminal to see which topics are available:

```
ros2 topic list
```

Topic Name	Topic Content
/Buzzer	Buzzer
/Servo	Servo s1, s2
/cmd_vel	Speed Control
/edition	Version Information
/imu/data_raw	IMU Sensor Data
/imu/mag	IMU Magnetometer Data
/vel_raw	Car Speed Information

Then enter the following command to query the topic message data types published/subscribed to by this node:

```
ros2 node info /driver_node
```



3.2 Publishing Control Commands

Based on the table of subscribed topics, use the following command format: `ros2 topic pub topic_name topic_message_data_type message_data --once` to publish a frame of control data.

- Issue Speed Control Commands

For the first test, it's recommended to suspend the car so its wheels don't touch the ground. We'll set the car to move forward at a linear velocity of 0.1 m/s. Enter the following command in the terminal:

```
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}}" --once `
```

After running, the car will move forward at a speed of 0.1 m/s. Similarly, to control the car's angular movement at a speed of 1.0 rad/s, assign a value to the z-axis of the angular axis using the following command:

```
ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.0, y: 0.0, z:
0.0}, angular: {x: 0.0, y: 0.0, z: 1.0}}" --once
```

After running, the car will rotate. To stop it, both the linear and angular velocities should be 0. When the velocity is 0, the front wheel servo will automatically correct. The command is as follows:

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

Here, `--once` means sending only one frame of message data. For other parameters of ROS2 topic pub, please refer to the course [19. ROS2 Basic Course - [16. Commonly Used ROS2 Commands and Tools] .

3.3 Controlling the Buzzer

To start the buzzer, enter the following command in the terminal:

```
ros2 topic pub /Buzzer std_msgs/msg/Bool "data: 1" --once
```

To turn off the buzzer, enter the following command in the terminal:

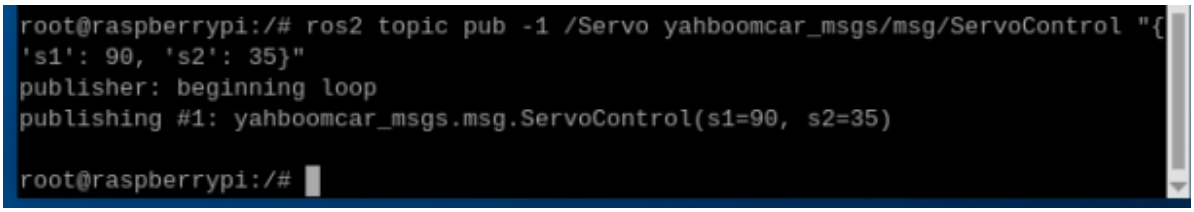
```
ros2 topic pub /Buzzer std_msgs/msg/Bool "data: 0" --once
```

3.4 Controlling the Servo Motor

Note: This step is only required for the version with a gimbal USB camera; it is not required for the depth camera version.

To control the servo motor, enter the following command in the terminal:

```
ros2 topic pub -1 /Servo yahboomcar_msgs/msg/ServoControl "{'s1': 90, 's2': 35}"
```



3.5 Reading Battery Voltage Data

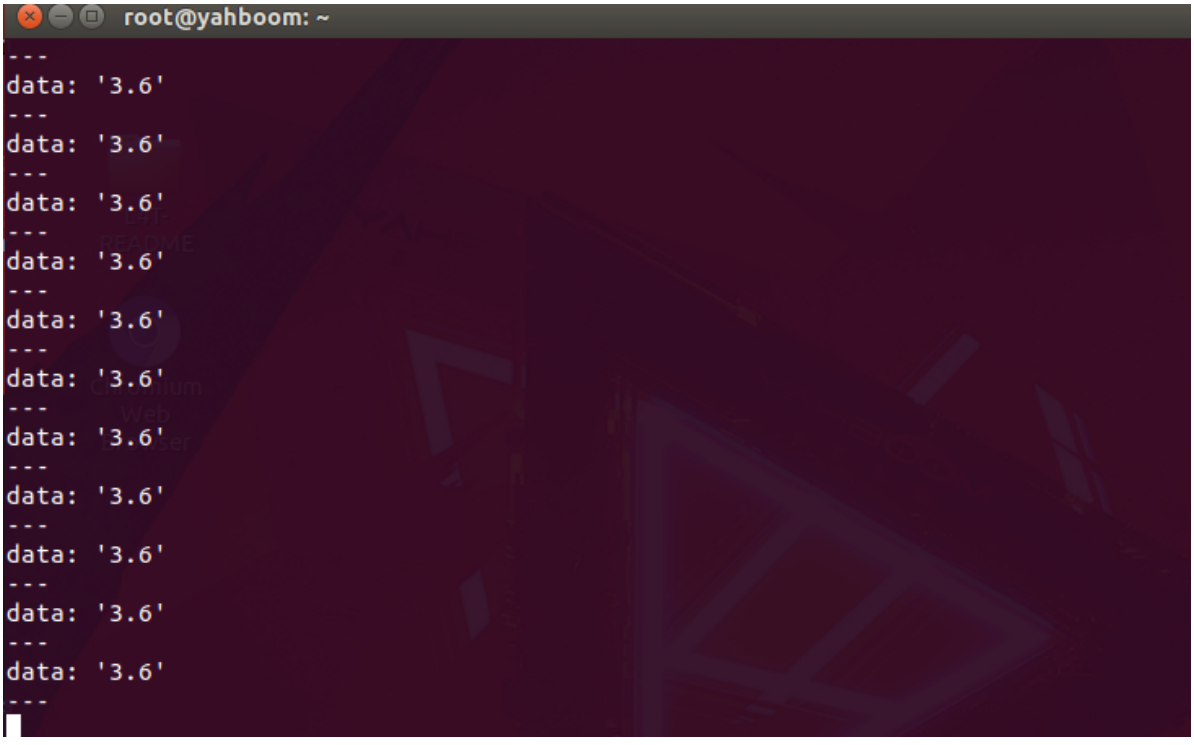
Theoretically, the normal operating battery voltage of this product is above 10.3V and below 12V. If the voltage drops below 10.3V, the buzzer will beep, indicating that the battery voltage is too low and needs to be charged. You can enter the following command in the terminal to query the battery voltage. The subscribed data is shown in the image below, where the battery voltage is 12.5V.

```
ros2 topic echo /voltage
```

3.6 Reading Firmware Version

Enter the following in the terminal:

```
ros2 topic echo /edition
```



4. Core Source Code Analysis

Taking Mcnamu_driver_M1.py as an example,

```
from Rosmaster_Lib import Rosmaster # Import driver library 导入驱动库

self.car = Rosmaster()  # Instantiate the Rosmaster object 实例化Rosmaster对象
```

```
# create subcriber 创建订阅者
self.sub_cmd_vel =
self.create_subscription(Twist,"cmd_vel",self.cmd_vel_callback,1)
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.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 to read information from the ROS expansion board 调用库，读取ros
拓展板的信息
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()

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

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