Keyboard control

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.

1. Program Function Description

Open the chassis and run the keyboard control program. The car's movement can be controlled via the keyboard, and the joystick can control the buzzer, servo motors, and other functions.

2. Starting Keyboard Control

2.1 Starting Command

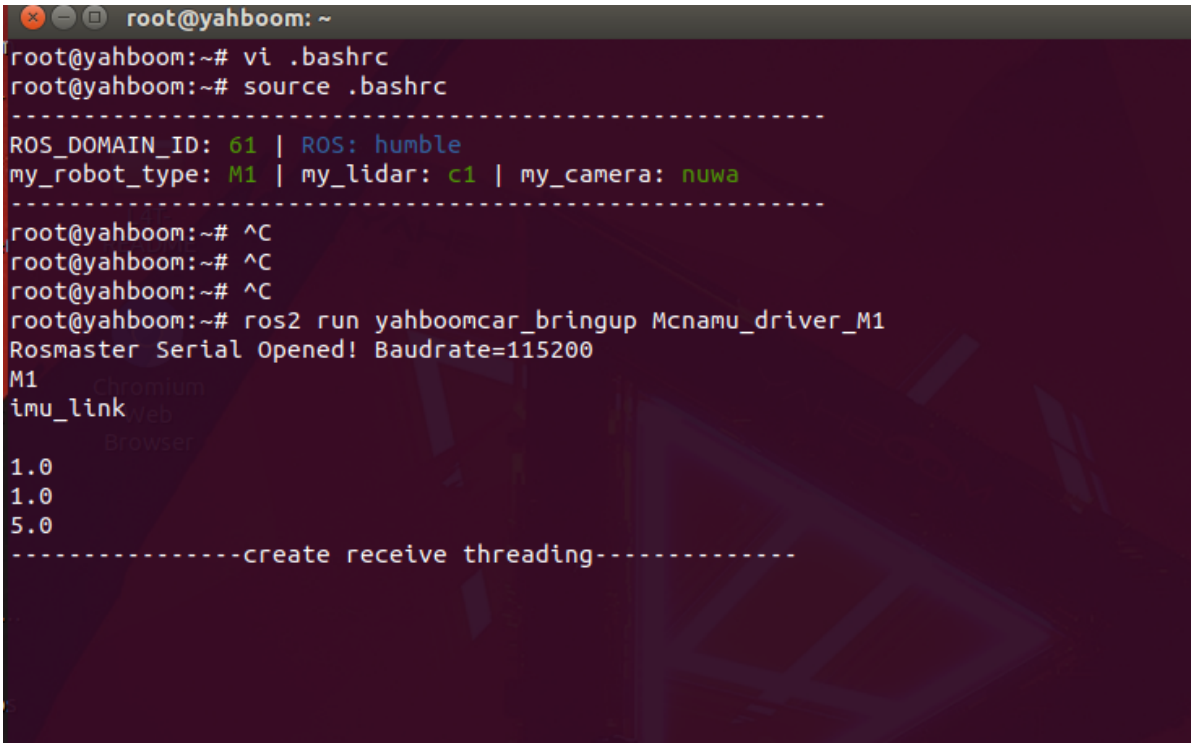
For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

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).**

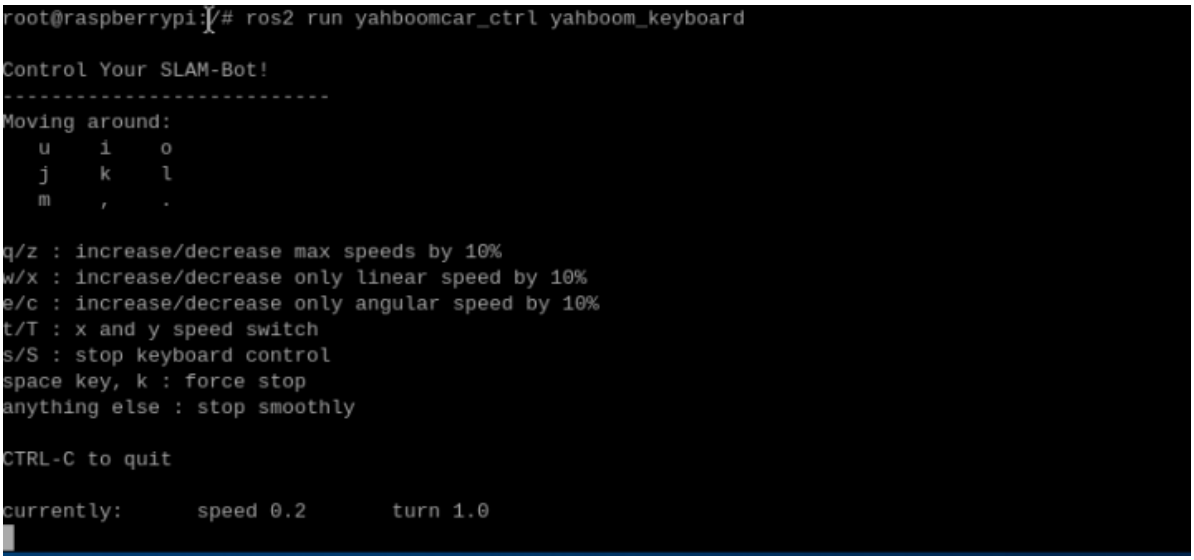
To start the chassis data, enter the following in the terminal:

```
ros2 run yahboomcar_bringup Mcnamu_driver_M1
```



To start the keyboard control program, enter the following command in the terminal:

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```



Keyboard key descriptions are as follows:

Direction Control

【i】or【I】	【linear，0】	【u】or【U】	【linear，angular】
【,】	【-linear，0】	【o】or【O】	【linear，- angular】
【j】or【J】	【0， angular】	【m】or【M】	【- linear，- angular】
【l】or【L】	【0，- angular】	【.】	【- linear，angular】

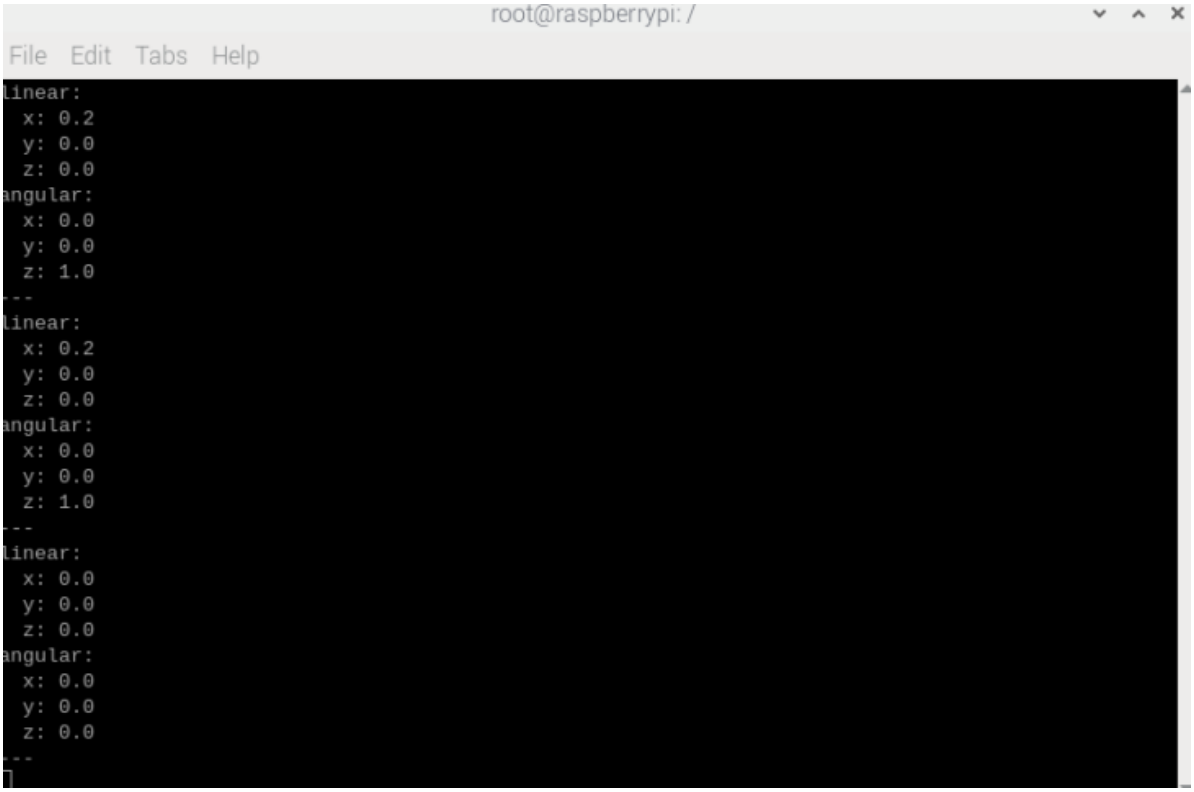
In other words, pressing [i] moves you forward, pressing [,] moves you backward, pressing [l] rotates you to the right, pressing [j] rotates you to the left, and so on.

Speed Control

Button	Speed Change	Button	Speed Change
【q】	Increases both linear and angular velocity by 10%	【z】	Decreases both linear and angular velocity by 10%
【w】	Increases linear velocity by 10% only	【x】	Decreases linear velocity by 10% only
【e】	Increases angular velocity by 10% only	【c】	Decreases angular velocity by 10% only
【t】	Switches between X-axis and Y-axis linear velocity	【s】	Stops keyboard control

You can open another terminal to view the real-time speed data for the /cmd_cel topic.

```
ros2 topic echo /cmd_vel
```

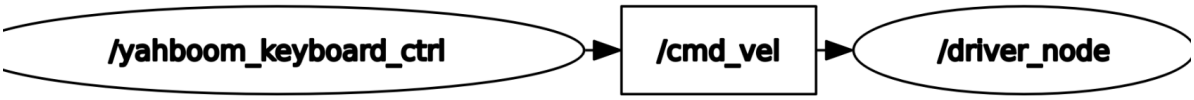


Because the robot chassis can only move in a 2D plane, only linear-x (x-axis velocity), linear-y (y-axis velocity), and angular-z (z-axis angular velocity) are actually published when controlling the robot via keyboard.

3. View the Node Relationship Graph

Open a terminal and enter the command:

```
ros2 run rqt_graph rqt_graph
```



From the node relationship diagram, we can see that:

yahboom_keyboard_ctrl: Controls the robot chassis by publishing the **/cmd_vel** topic.

/driver_node: The robot chassis node, subscribes to the **/cmd_vel** topic, and communicates with the ROS extension board to provide the speed of each wheel, thereby controlling the robot's movement.

4. Source Code Analysis

Source code path:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_keyboard.py
```

- The movement dictionary primarily stores characters related to directional control.

```
moveBindings = {
    'i': (1, 0),
    'o': (1, -1),
    'j': (0, 1),
    'l': (0, -1),
    'u': (1, 1),
    ',': (-1, 0),
    '.': (-1, 1),
    'm': (-1, -1),
    'I': (1, 0),
    'O': (1, -1),
    'J': (0, 1),
    'L': (0, -1),
    'U': (1, 1),
    'M': (-1, -1),
}
```

- The speed dictionary mainly stores characters related to speed control.

```
speedBindings = {
    'Q': (1.1, 1.1),
    'Z': (.9, .9),
    'W': (1.1, 1),
    'X': (.9, 1),
    'E': (1, 1.1),
    'C': (1, .9),
    'q': (1.1, 1.1),
    'z': (.9, .9),
    'w': (1.1, 1),
    'x': (.9, 1),
    'e': (1, 1.1),
    'c': (1, .9),
}
```

Get current key information

```
def getKey(self):
    tty.setraw(sys.stdin.fileno())
    rlist, _, _ = select.select([sys.stdin], [], [], 0.1)
    if rlist: key = sys.stdin.read(1)
    else: key = ''
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, self.settings)
    return key
```

Determine key-value pairs and publish to the /cmd_vel speed topic

```
while (1):
    key = yahboom_keyboard.getKey()
    if key=="t" or key == "T": xspeed_switch = not xspeed_switch
    elif key == "s" or key == "S":
        print ("stop keyboard control: {}".format(not stop))
        stop = not stop
    if key in moveBindings.keys():
        x = moveBindings[key][0]
        th = moveBindings[key][1]
        count = 0
```

```
    elif key in speedBindings.keys():
        speed = speed * speedBindings[key][0]
        turn = turn * speedBindings[key][1]
        count = 0
        if speed > yahboom_keyboard.linenar_speed_limit:
            speed = yahboom_keyboard.linenar_speed_limit
            print("Linear speed limit reached!")
        if turn > yahboom_keyboard.angular_speed_limit:
            turn = yahboom_keyboard.angular_speed_limit
            print("Angular speed limit reached!")
        print(yahboom_keyboard.vels(speed, turn))
        if (status == 14): print(msg)
        status = (status + 1) % 15
    elif key == ' ': (x, th) = (0, 0)
    else:
        count = count + 1
        if count > 4: (x, th) = (0, 0)
        if (key == '\x03'): break
    if xspeed_switch: twist.linear.x = speed * x
    else: twist.linear.y = speed * x
    twist.angular.z = turn * th
    if not stop: yahboom_keyboard.pub.publish(twist)
    if stop:yahboom_keyboard.pub.publish(Twist())
```