Handheld 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

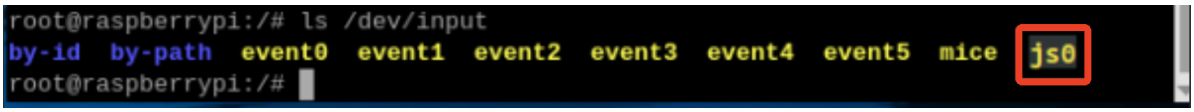
Open the chassis and run the controller program. You can control the car's movement using the controller. The controller also controls the buzzer and servo motors.

2. Program Startup

2.1. Checking the Device

First, plug the wireless controller's USB receiver into the host (rdkx5, jetson, Raspberry Pi, PC). Open the terminal and enter the following command. If you see "js0", this is the wireless controller.

```
ls /dev/input
```



2.2. Testing the Controller Input

Open the terminal and enter the following command. As shown in the image, this wireless controller has 8 axial inputs and 15 button inputs. You can press the buttons individually to test the corresponding numbers.

```
sudo jstest /dev/input/js0
```

The button should normally turn on. If jstest is not installed, run the following command:

```
sudo apt-get install joystick
```

2.3 Startup Command

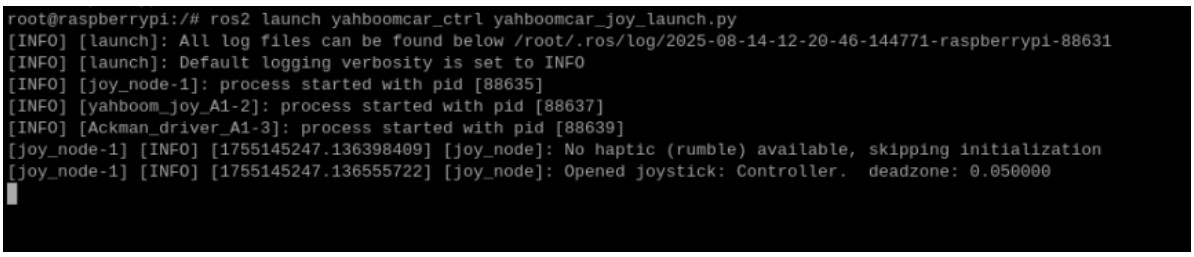
For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

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).**

Start chassis data + controller control, terminal input:

```
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py
```



After starting, press the "START" button. You will hear a buzzer sound, indicating you can begin remote control. **The remote control will enter sleep mode after a period of inactivity; you need to press the "START" button to exit sleep mode.** To **control the car**, you also need to **press the R2 button** to **unlock the motion control** before you can use the joystick to control the car's movement.

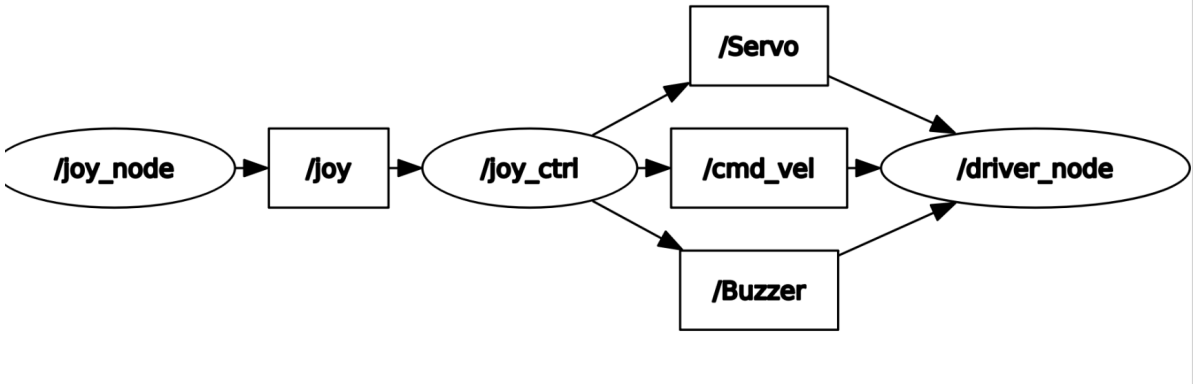
Remote Control Effects Explanation:

Controller	Effects
Left Stick Up/Down	Forward/Backward (Straight Movement)
Left Stick Down	Decelerate
Right Stick Left/Right	Turn Left/Right
Right Stick Down	Accelerate
"START" Button	Controls Buzzer/Ends Sleep Mode
Left Arrow Key Left/Right	Controls S1 Servo Left/Right Movement
Left Arrow Key Up/Down	Controls S2 Servo Up/Down Movement
R1 Button	Servo Reset/Center
R2 Button	Controller Motion Control Switch

3. Node Communication Diagram

Terminal Input:

```
ros2 run rqt_graph rqt_graph
```



If it doesn't display initially, select 【Nodes/Topics(all)】, then click the refresh button in the upper left corner.

4. Code Analysis

Source code path,

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/launch/yahboomcar_joy_launch.py
```

yahboomcar_joy_launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

import os
def generate_launch_description():
    node1 = Node(
        package='joy',
        executable='joy_node',
    )
    joy_node = Node(
        package='yahboomcar_ctrl',
        executable='yahboom_joy_M1',
    )
    bringup_node = Node(
        package='yahboomcar_bringup',
        executable='Ackman_driver_M1',
    )
    launch_description = LaunchDescription([node1,joy_node,bringup_node])
    return launch_description
```

The system runs three nodes: the chassis data node, the yahboom_ctrl controller node, and the joy_node controller key-value reading node.

Key controller control node source code path

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_joy_M1.py
```

```
#创建订阅者订阅/joy话题数据 Create subscriber subscriptions/joy topic data
self.sub_Joy = self.create_subscription(Joy,'joy', self.buttonCallback,10)
```

```
#处理按键函数user_jetson Key press handling function user_jetson
def user_jetson(self, joy_data):
    # cancel nav
    if joy_data.buttons[9] == 1: self.cancel_nav()

     #Buzzer 蜂鸣器控制
    if joy_data.buttons[11] == 1:
        Buzzer_ctrl = Bool()
        self.Buzzer_active=not self.Buzzer_active
        Buzzer_ctrl.data =self.Buzzer_active
        self.pub_Buzzer.publish(Buzzer_ctrl)

    #档位调整，按下摇杆即可调节档位    To adjust the gear, simply press the joystick.
    if joy_data.buttons[13] == 1:
        if self.linear_Gear == 1.0: self.linear_Gear = 1.0 / 5
        elif self.linear_Gear == 1.0 / 3: self.linear_Gear = 2.0 / 4
        elif self.linear_Gear == 2.0 / 3: self.linear_Gear = 1
    # angular Gear control
    if joy_data.buttons[14] == 1:
        if self.linear_Gear == 1.0: self.linear_Gear = 1.0
        elif self.linear_Gear == 1.0 / 5: self.linear_Gear = 1.0
    # Servo
    self.servos1 = int(max(0, min(180, self.servos1 + joy_data.axes[6] * 3)))
    self.servos2 = int(max(0, min(100, self.servos2 + joy_data.axes[7] * 3)))
    self.servo_angle.s1 = self.servos1
    self.servo_angle.s2 = self.servos2
    if not 30 <= self.servo_angle.s1 <= 150:
        self.servo_angle.s2 = max(10, self.servo_angle.s2)

    xlinear_speed = self.filter_data(joy_data.axes[1]) * self.xspeed_limit *
self.linear_Gear
    ylinear_speed = self.filter_data(joy_data.axes[2]) * self.yspeed_limit *
self.linear_Gear
    angular_speed = self.filter_data(joy_data.axes[2]) *
self.angular_speed_limit * self.angular_Gear
    # print(self.linear_Gear)
    if xlinear_speed > self.xspeed_limit: xlinear_speed = self.xspeed_limit
    elif xlinear_speed < -self.xspeed_limit: xlinear_speed = -self.xspeed_limit
    if ylinear_speed > self.yspeed_limit: ylinear_speed = self.yspeed_limit
    elif ylinear_speed < -self.yspeed_limit: ylinear_speed = -self.yspeed_limit
    if angular_speed > self.angular_speed_limit: angular_speed =
self.angular_speed_limit
    elif angular_speed < -self.angular_speed_limit: angular_speed = -
self.angular_speed_limit
    twist = Twist()
    twist.linear.x = xlinear_speed
    twist.linear.y = ylinear_speed
    twist.angular.z = angular_speed
    if self.Joy_active == True:
        for i in range(3):
            self.pub_cmdVel.publish(twist)
            self.pub_Servo.publish(self.servo_angle)
```

Variables corresponding to remote control key values

Based on the default mode [Controller], the key values corresponding to the remote control are as follows:

Remote control event	Corresponding variable
Left joystick up	axes[1]=1
Left joystick down	axes[1]=-1
Right joystick left	axes[2]=1
Right joystick right	axes[2]=-1
Button X pressed	button[3]=1
Button B pressed	button[1]=1
Button Y pressed	button[4]=1
Button R1 pressed	button[7]=1
Start button pressed	button[11]=1
Left joystick pressed	button[13]=1
Right joystick pressed	button[14]=1

To better understand, referring to the source code above, when these values change, it means the remote control button has been pressed, and the corresponding program will be executed. To view other button presses, you can subscribe to the /joy topic and enter the command in the terminal.

```
ros2 topic echo /joy
```