5 Robot handle control

According to different models, you only need to set the purchased model in [.bashrc], X1(ordinary four-wheel drive) X3(Mike wheel) X3plus(Mike wheel mechanical arm) R2(Ackerman differential) and so on. Section takes X3 as an example

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameter and modify the corresponding model

```
export ROBOT_TYPE=X3   # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

5.1 Operating environment

OS: Ubuntu 18.04 LTS

ROS version: melodic

Equipment: jetson nano/Raspberry Pi, PC, wireless handle(USB receiver)

Handle control function code path: ~/yahboomcar _ws/src/yahboomcar_ctrl/scripts

5.2 install the driver

ROS driver for generic Linux controllers. The Joy package contains Joy_node, a node that connects a generic Linux controller to ROS. The node publishes a "/Joy" message containing the current state of each button and axis of the handle.

```
sudo apt install ros-melodic-joy ros-melodic-joystick-drivers
```

5.3 Use steps

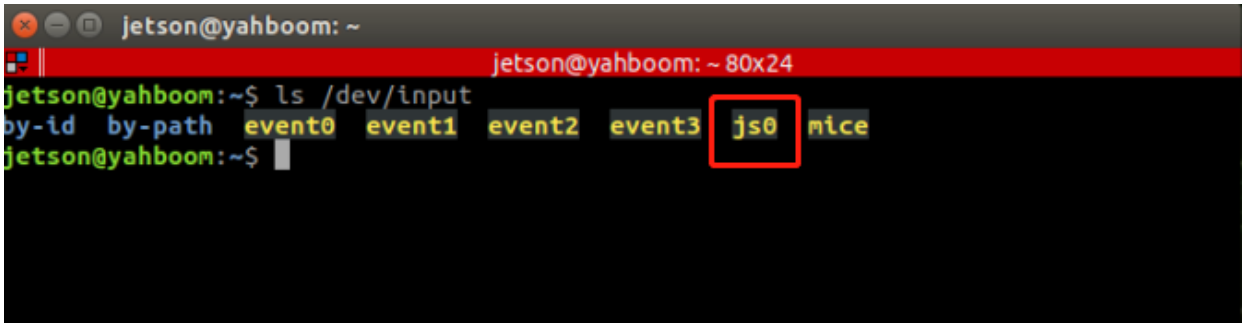
5.3.1 Device connection

First, connect the USB end of the wireless controller to the device(jetson, Raspberry Pi, PC). This lesson takes connecting the USB end of the wireless controller to the jetson as an example, connecting the mouse, keyboard, and monitor;

5.3.2 View device

Open the terminal, enter the following command, it shows [js0], this is the wireless controller. In special cases, it can also be viewed through the two states of connecting and not connecting to the USB terminal of the wireless handle. The device list changes, if there is a change, the device is changed; otherwise, the connection is unsuccessful or cannot be recognized.

```
ls /dev/input
```

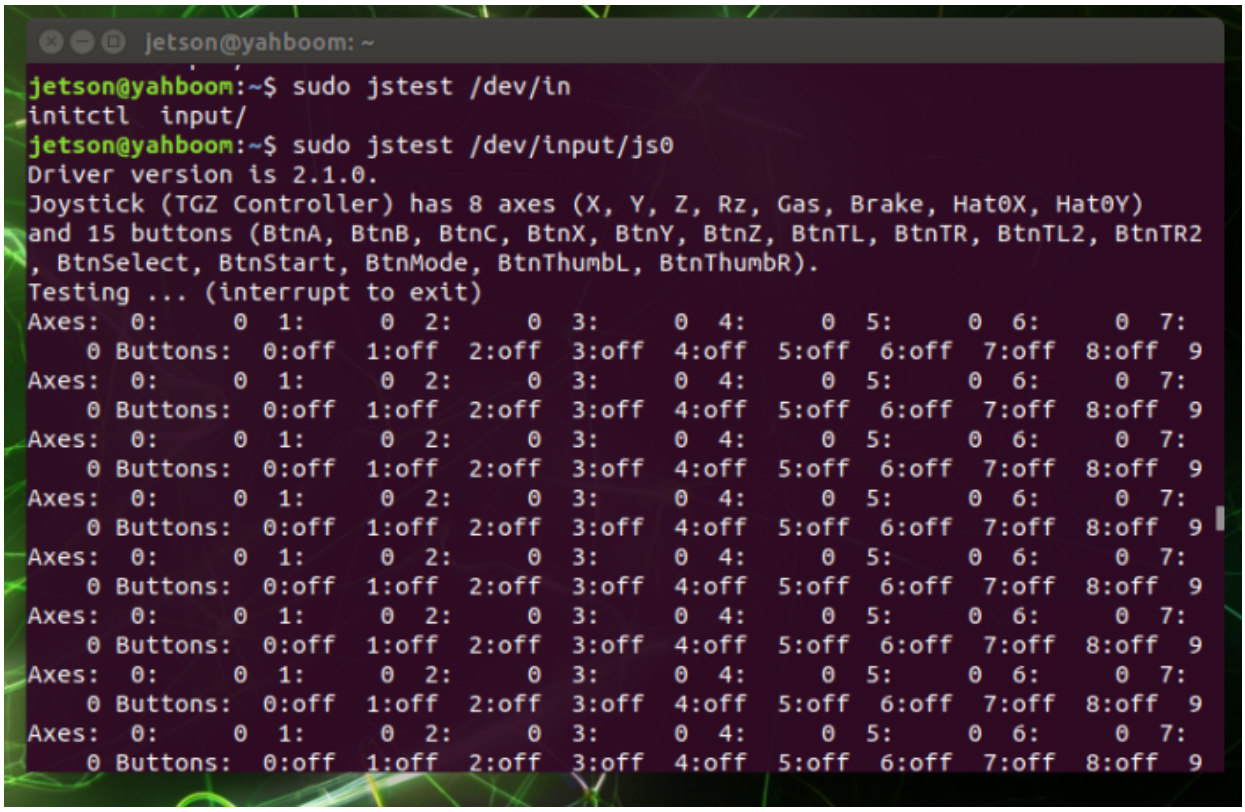


Note: If the keyboard or mouse and other devices are connected to the ROSMASTER first, the remote control receiver will be recognized as other devices. Therefore, if you need to connect the keyboard or mouse, you can connect the remote control receiver first and then connect to other devices. equipment.

5.3.3 Test handle

Open the terminal and enter the following commands. As shown in the figure, the wireless handle has 8 axial inputs and 15 key inputs. You can press the keys respectively to test the numbers corresponding to the keys.

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



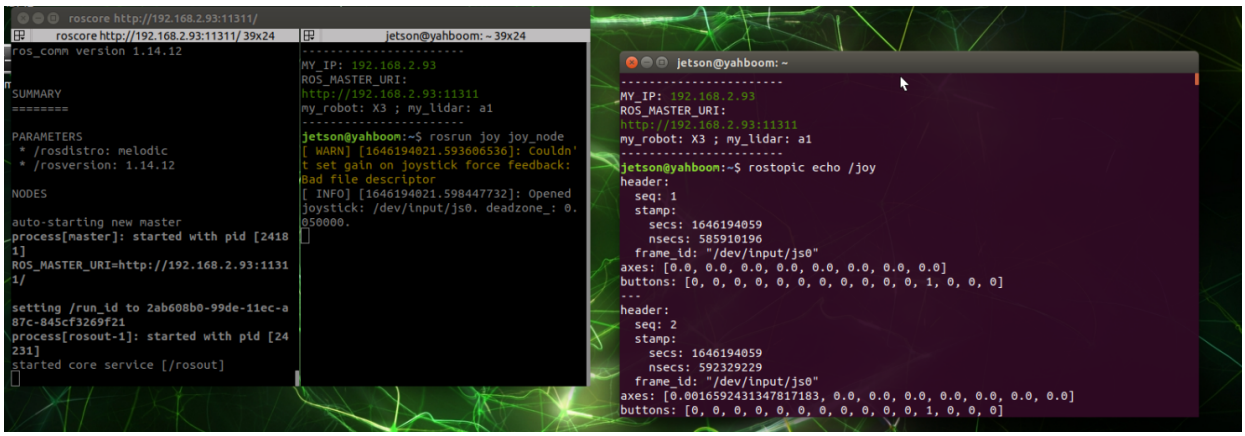
If jstest is not installed, run the following command:

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

5.3.4 Running the handle node

Open three terminals, and enter the following commands in turn to view the detailed information, which is the same as [Test Handle]. Different devices(Raspberry Pi, jetson nano, PC) and different systems will have different handle states.

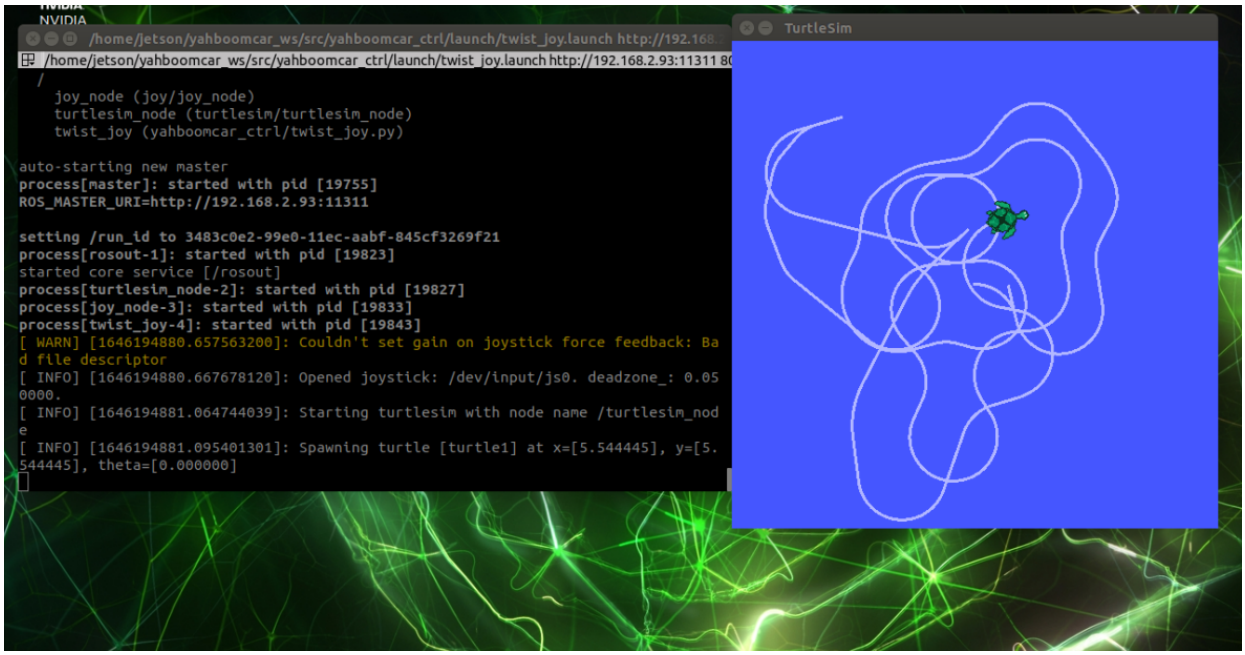
```
roscore
rosrun joy joy_node
rostopic echo joy published information
```



5.4 The handle controls the little turtle

5.4.1 Start

```
roslaunch yahboomcar_ctrl twist_joy.launch
```



1. view the node

```
rostopic list
```

```
pi@yahboom:~$ rostopic list
/diagnostics
/joy
/joy/set_feedback
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

2. view the speed topic information

```
pi@yahboom:~$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /twist_joy (http://192.168.2.116:33379/)

Subscribers:
 * /turtlesim_node (http://192.168.2.116:45465/)
```

3. The corresponding relationship between the handle and the operation of the little turtle

handle	little turtle

handle	little turtle
left stick	go ahead
Left joystick down	back
Right stick left	Turn left
right joystick	Turn right

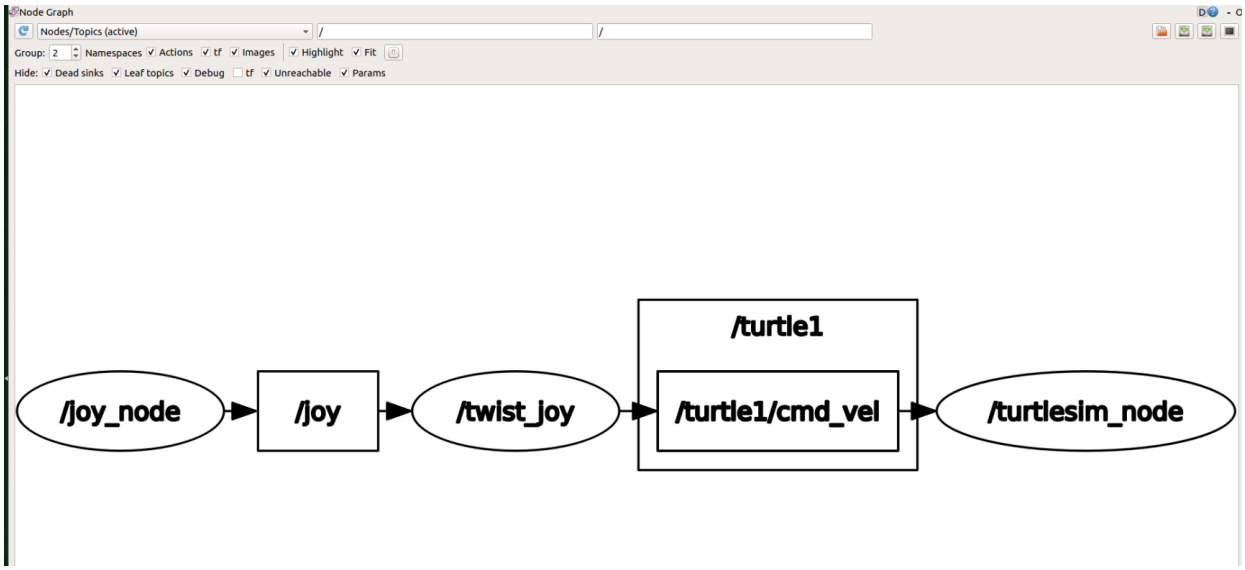
3. view the node graph

[/joy_node]: Node of handle information

[/twist_joy]: The node controlled by the handle

[/turtlesim_node]: The node of the little turtle

The node [/joy_node] publishes the topic [/joy], so that the node [/twist_joy] subscribes, and after processing, publishes the topic [/turtle1/cmd_vel] for the node [/turtlesim_node] to subscribe, and drives the little turtle to move.



5.5, handle control ROSMASTER

The handle to control ROSMASTER is similar to the handle to control the turtle; let the handle control node establish a connection with the ROSMASTER underlying driver node, change the current state of the handle, send different information to ROSMASTER, and drive ROSMASTER to respond differently. **Corresponding to the ROSMASTER of different models, what we need to control is also different.**

For ROSMASTER-X3(Mecanum wheel) we can control the **buzzer, light strip, the linear speed of the car movement(x-axis and y-axis) and the angular speed of the car movement through the handle.**

5.5.1 Start handle control

1. handle control yahboom_joy.launch file reference path

```
~/yahboomcar_ws/src/yahboomcar_ctrl/launch/
```

2. start

```
roslaunch yahboomcar_bringup bringup.launch             # start the chassis and
handle control, the launch of handle control is yahboom_joy.launch
```

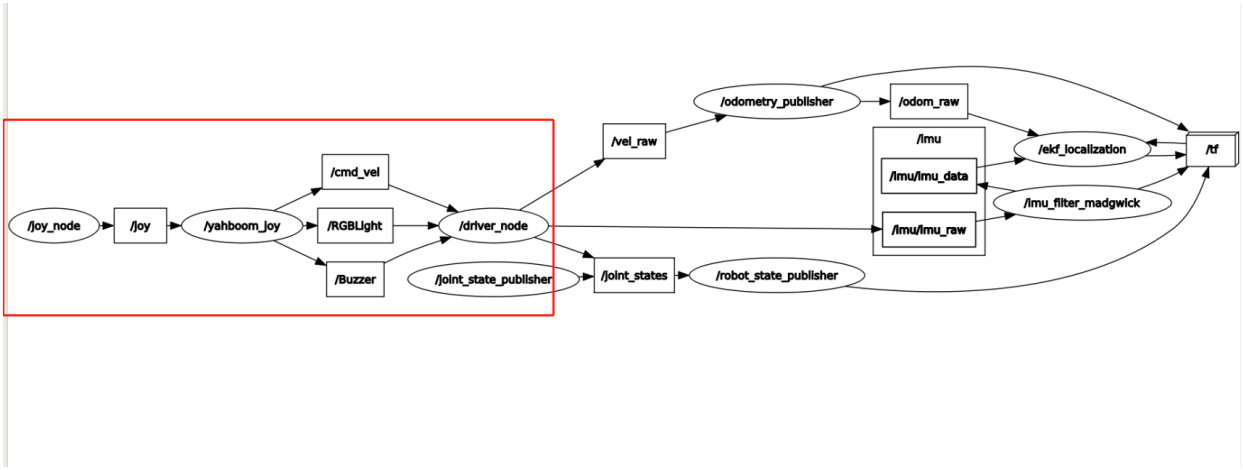
After turning it on, press the "START" button and hear the buzzer to start the remote control. **If the remote control is turned on for a period of time, it will enter the sleep mode, and you need to press the "START" button to end the sleep mode.**

3. remote control effect description

handle	Effect
Left stick up/down	The car goes straight forward/backward
Left joystick left/right	The car goes straight left/right
Right stick left/right	Car turns left/right
Right "1" key	Control light strip effects
Right "2" key	End/open other functions
"START" button	Control buzzer/end sleep
Left stick pressed	Adjust the speed of X/Y axis
Right stick pressed	Adjust the angular velocity

4. view the node graph

```
rqt_graph
```



The red box part is the topic communication between the handle control node/yahboom_joy and the underlying node/driver_node.

5. program analysis

Bottom control program:

```
Mcnamu_driver.py
```

The code path is:

```
~/yahboomcar_ws/src/yahboomcar_bringup/scripts
```

In Mcnamu_driver.py, the buzzer, speed control, **subscribers of the topic information of the light strip and their respective callback functions are defined,**

```
self.sub_cmd_vel = rospy.Subscriber('cmd_vel', Twist, self.cmd_vel_callback,
queue_size=100)    # speed control
self.sub_RGBLight = rospy.Subscriber("RGBLight", Int32, self.RGBLightcallback,
queue_size=100)  # strip control
self.sub_Buzzer = rospy.Subscriber("Buzzer", Bool, self.Buzzercallback,
queue_size=100)         # Buzzer control

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

Handle control program:

```
yahboom_joy.py
```

The code path is:

```
~/yahboomcar_ws/src/yahboomcar_ctrl/scripts
```

In yahboom_joy.py, the buzzer, the speed control, the publisher of the topic information of the light strip , and the publisher of these messages

```
self.pub_cmdVel = rospy.Publisher('cmd_vel', Twist, queue_size=10)       #speed
control
self.pub_RGBLight = rospy.Publisher("RGBLight", Int32, queue_size=10)    #strip
control
self.pub_Buzzer = rospy.Publisher("Buzzer", Bool, queue_size=1)          #buzzer
control

self.pub_cmdVel.publish(twist)                    # publish speed data
self.pub_RGBLight.publish(self.RGBLight_index)    # publish light strip data
self.pub_Buzzer.publish(self.Buzzer_active)       # publish buzzer data
```

6. code analysis

- /joy topic data analysis

  Run the joy_node node and view the /joy topic information,

```
rosrun joy joy_node
rostopic echo joy
```

In the print information, there are two arrays: axes and buttons, and the data they store is the behavior of the joystick and buttons on the remote control. You can press each button on the remote control one by one, one by one corresponding to the transformation of the array, and you can know what behavior will cause which variable of the array will change its value.

- Knowing the change of the /joy topic data corresponding to the button behavior, then in yahboom_joy.py, we need to make judgments on these values. There are also two categories in yahboom_joy.py. **First, the name of the system will be judged(this is first to judge whether the handle receiver is plugged into the jetson or the Raspberry Pi/PC(referring to the virtual machine here))**,

```
if  self.user_name  ==  "jetson": self.user_jetson(joy_data)
        else: self.user_pc(joy_data)
```

- Value judgment: Take the jetson processor to judge the buzzer Buzzer data as an example, analysis, we use the "START" button to control the buzzer, and it corresponds to buttons[11], when it is pressed, buttons[11] will become 1, at this time, we can send the buzzer data,

  ```
  if joy_data.buttons[11] == 1:
      self.Buzzer_active=not self.Buzzer_active
      # print "self.Buzzer_active: ", self.Buzzer_active
      self.pub_Buzzer.publish(self.Buzzer_active)
  ```

  Other remote control behaviors are analyzed by analogy.

Note: When judging the left and right joysticks, only the data of X3(Mecanum wheel) and X3 Plus(Mecanum wheel + mechanical arm) are valid, because of the characteristics of their wheels, they can move laterally, so Y There is speed on the shaft.

```
 ylinear_speed = self.filter_data(joy_data.axes [ 0 ]) * self.yspeed_limit *
self.linear_Gear
```

5.6、Precautions for using the handle

- When connecting the USB handle receiver, it is recommended to connect it to the outermost USB-HUB expansion board instead of directly connecting it to the main board or the middle USB-HUB expansion board (X3plus). If it is directly connected to the main board or the middle USB-HUB expansion board (X3plus), due to the upper and lower aluminum alloy grille, it will seriously interfere with the signal reception of the handle.
- After plugging and unplugging the handle receiving head, the handle program needs to be restarted, otherwise the car will not be able to be controlled.
- After starting the handle control program, if the handle cannot control the car, it may be caused by the wrong handle control mode. You can press and hold the handle mode button for about 15 seconds to switch modes. After the green indicator light is always on, press the start button again. If the buzzer sounds, it means the switching is successful. If there is no response, you can press and hold the mode button on the handle again for 15 seconds.

  Jetson series support mode: PC/PCS mode. In PC mode, the POWER MODE indicator light is red by default. You can connect the handle receiver to the usb port of the computer to connect to the wireless handle. Enter the URL in the browser: https://gamepad-tester.com/. Pressing the button URL will display the change of the button value, as shown in the following figure:

Raspberry Pi series support mode: X-BOX mode. In X-BOX mode, the default POWER MODE indicator light is green. You can connect the handle receiver to the usb port of the computer to connect to the wireless handle. Enter the URL in the browser: https://gamepad-tester.com/. Pressing the button URL will display the change of the button value, as shown in the following figure:

- After re-plugging the handle receiver or restarting the motherboard, the handle will reset to the factory mode. If it cannot be controlled, you need to switch the mode again every time you plug or restart.

- In the case of unsuccessful matching, the POWER MODE indicator light will flash red and green all the time, and will not light up after a few seconds of sleep.