9.3、Handle remote control of ROS robot

Note: Unlike chapter 6.2, the handle in this chapter is connected to jetbotmini. If there is no response when the handle is just connected, you need to press the START button on the handle.

9.3.1、Environment

Note: If you use the official image of jetbotmini, you can skip the environment setup step.

1. Install driver

ROS driver for universal handles. The Joy package contains Joy_node, which is a node that connects the universal controller to ROS. The node publishes a "/Joy" message, which contains the current state of each button and axis of the handle.

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

2. Steps for usage

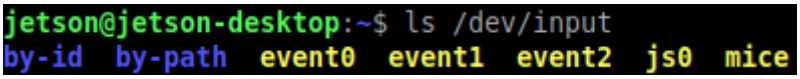
- Device connection

First, connect the USB end of the handle to jetbotmini, connect the mouse, keyboard, and monitor; or remotely connect via VNC, etc.

- Check the device

Open the terminal, enter the following command, it shows [js0], this is the handle. In special cases, you can also check the two states of connecting and not connecting to the USB port of the handle. If there is a change in the device list, the device will change; otherwise, the connection is unsuccessful or unrecognized.

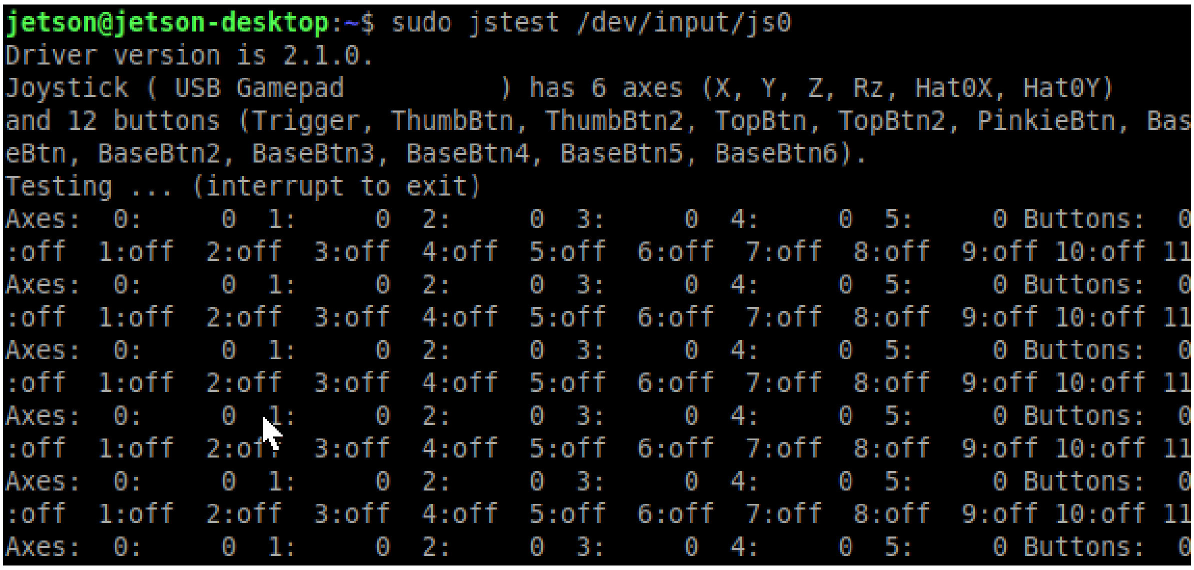
```
ls /dev/input
```



- Test handle

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

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



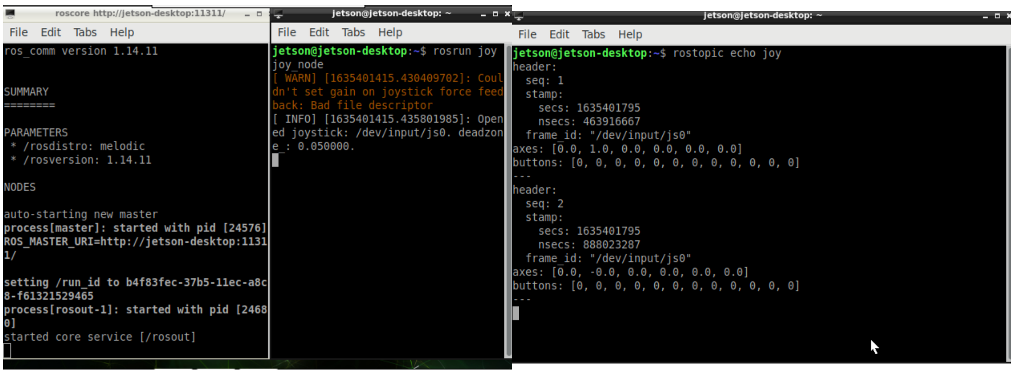
If jstest is not installed, run the following command:

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

- Run the controller node

Open three terminals and enter the following commands in sequence to view detailed information, similar to [Test Handle].

```
roscore  #first step
rosrun joy joy_node  #Second step
rostopic echo joy  #third step
```

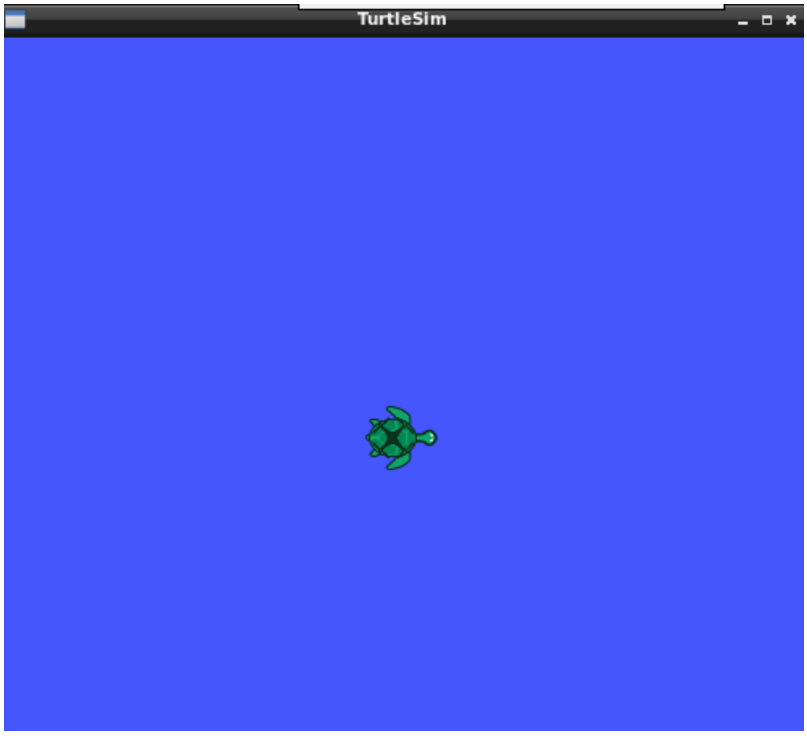


9.3.2、 Handle control little turtle

1. Start the little turtle

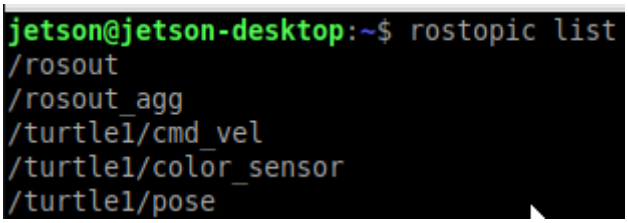
```
roscore
rosrun turtlesim turtlesim_node
```

If you want to drive the little turtle, just give the little turtle a certain speed. Next, check the ROS speed control node.



2. View node

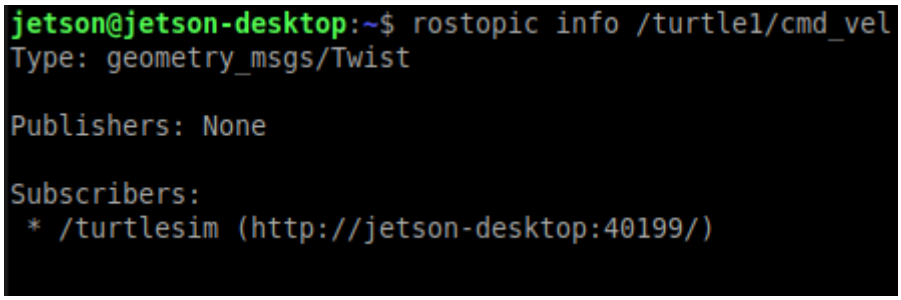
```
rostopic list
```



3. View the node information

```
rostopic info /turtle1/cmd_vel
```

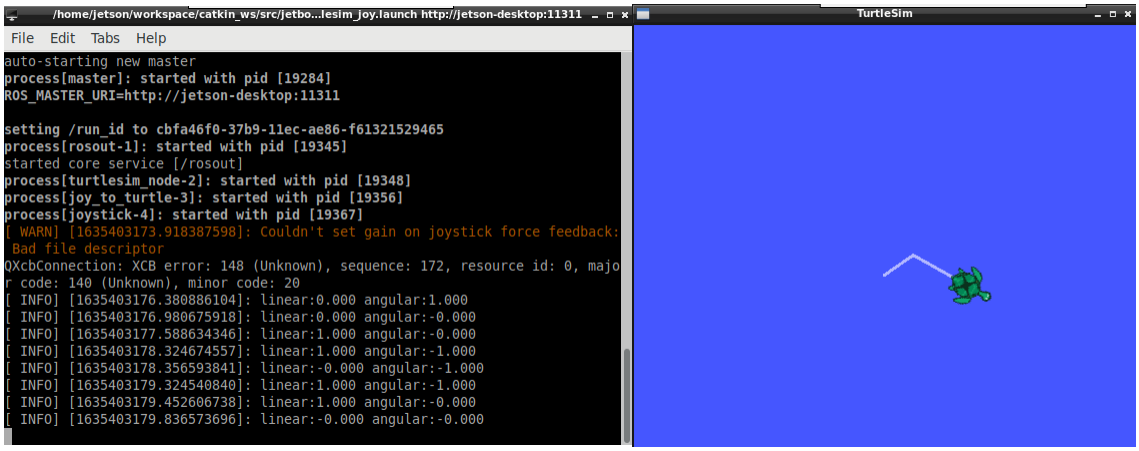
If you want to control the movement of the little turtle, you only need to post a message to the topic [/turtle1/cmd_vel] with the type of [geometry_msgs/Twist].



4. Handle control little turtle

Receive the current status information of the handle, and send instructions to the little turtle by pressing the button or shaking the joystick to receive different information feedback from the handle.

```
    roslaunch jetbot_ros turtlesim_joy.launch
```



At this point, you can use the handle to control the turtle to run.

- Correspondence between the handle and the turtle

Handle	**Turtle**
Left rocker up	advance
Left rocker down	back
Left rocker left	turn left
Left rocker right	turn right

- View node graph

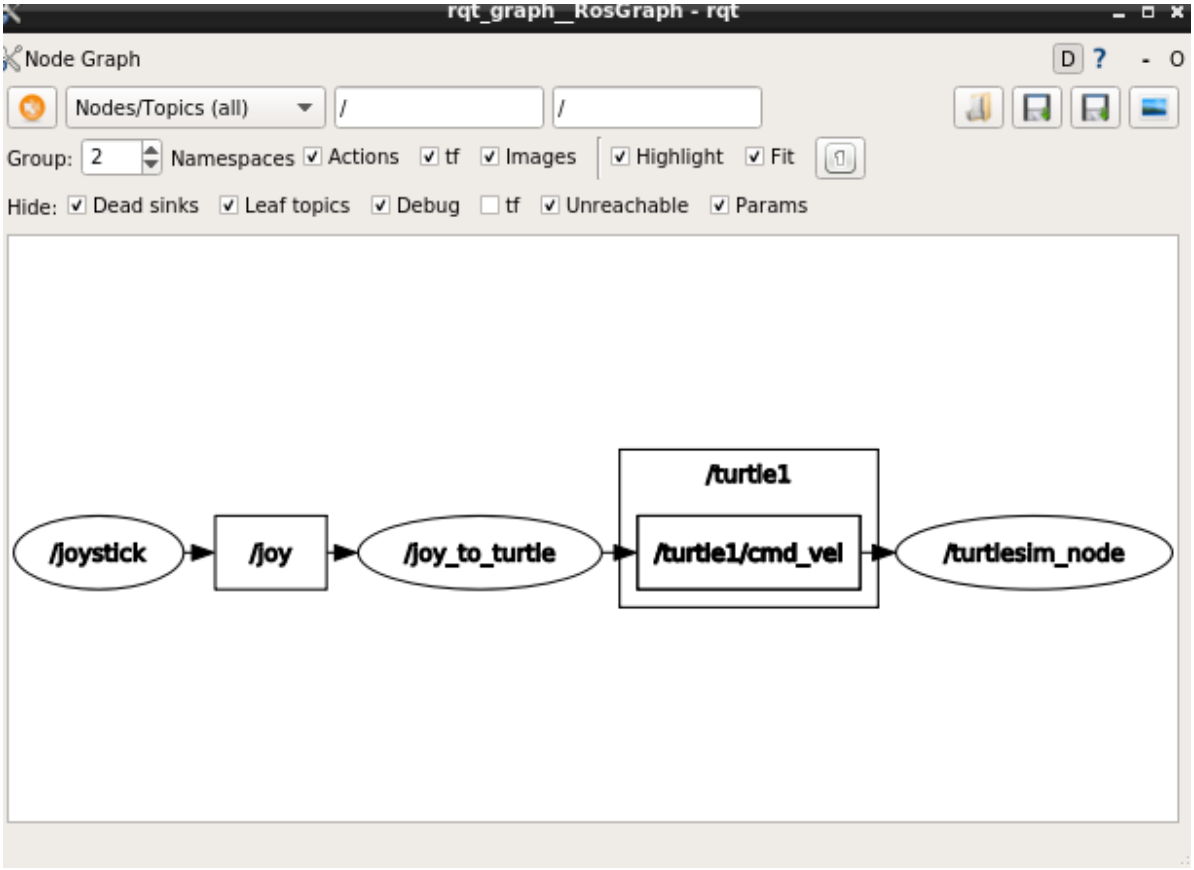
【/joystick】：Handle information node

【/turtlesim_joy】：Handle control node

【/turtlesim_node】：Turtle node

The node [/joystick] publishes the topic [/joy] to make the node [/joy_to_turtle] subscribe, and then publishes the topic [/turtle1/cmd_vel] after processing it to make the node [/turtlesim_node] subscribe to drive the little turtle movement.

```
rqt_graph
```



Code path:

/home/jetson/workspace/catkin_ws/src/jetbot_ros/launch/turtlesim_joy.launch

/home/jetson/workspace/catkin_ws/src/jetbot_ros/src/logitech.cpp

9.3.3 Wired handle to control Jetbot-mini

The handle controls Jetbot-mini, which is similar to the handle control of the turtle; let the handle control node establish a connection with the Jetbot-mini bottom driver node, change the current state of the handle, send different information to Jetbot-mini, and drive Jetbot-mini to make different responses. We need to control the speed and direction of the trolley, buzzer control, LED light control, and steering gear control.

The topic needs continuous communication, and the service does not need continuous communication. Make the following settings according to the characteristics of the topic and the service and business needs:

1. Handle control node

- Corresponding to the bottom driver of Jetbot-mini
  - Service (client)

    Receive buzzer control messages【/Buzzer】

    Receive blue LED control message【/LEDBLUE】

    Receive green LED control message【/LEDGREE】

    Receive servo control message【/Servo】

    Receive motor control messages【/Motor】
- Other topics
  - Joy_node

    Subscribe to the news of the wireless controller【/joy】

- Status release

  Publish current Joy status (custom) message【/JoyState】

2. Associated node

- Low-level driver
  - Service (client)

    Receive buzzer control messages【/Buzzer】

    Receive blue LED control message【/LEDBLUE】

    Receive green LED control message【/LEDGREE】

    Receive servo control message【/Servo】

    Receive motor control messages【/Motor】
- Joy_node
  - Topic

    Publish the news of the wireless controller [/joy]

3. Operating procedures

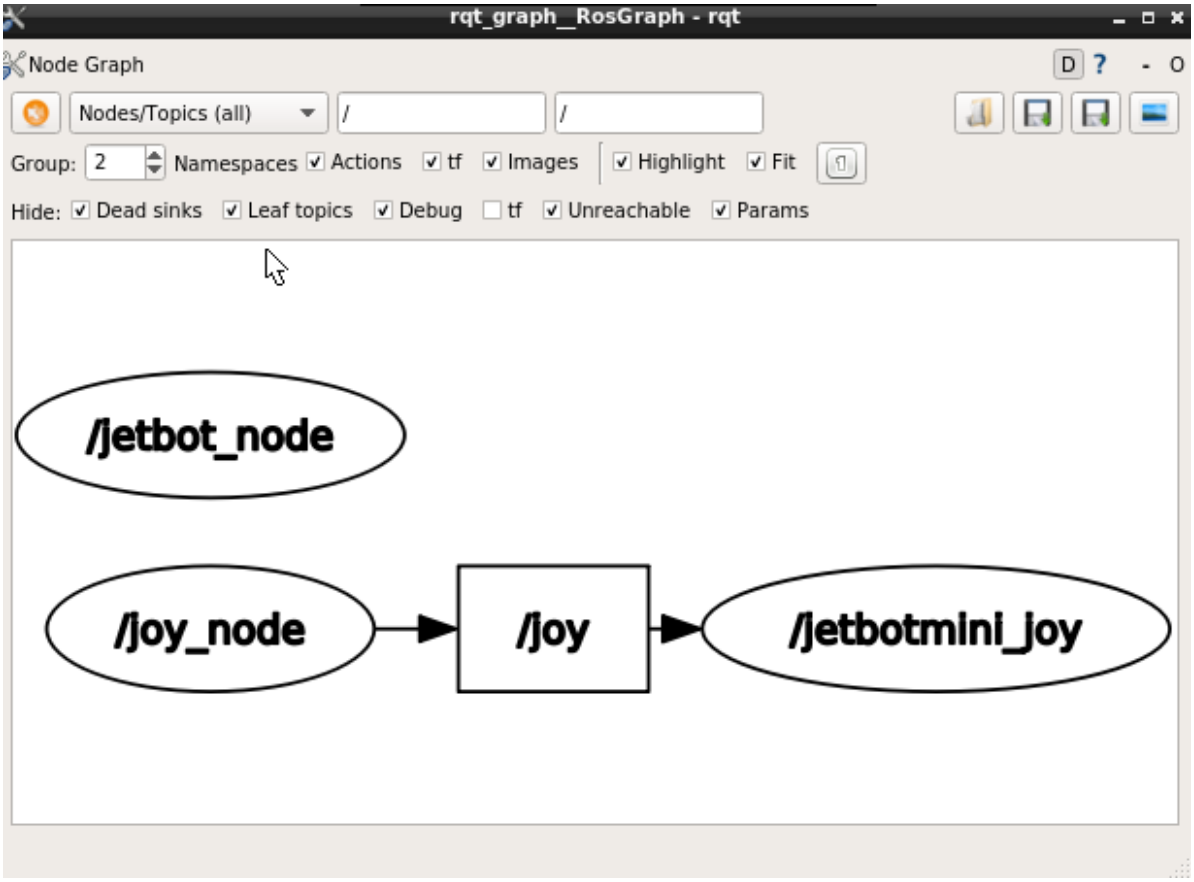
Start command

```
roslaunch jetbot_ros joy_bringup.launch
```

View node graph

```
rqt_graph
```

Connection between topics



View service list

```
rosservice list
```

The print is as follows:

```
/Buzzer
/LEDBLUE
/LEDGREE
/Motor
/Servo
/jetbot_node/get_loggers
/jetbot_node/set_logger_level
/jetbotmini_joy/get_loggers
/jetbotmini_joy/set_logger_level
/joy_node/get_loggers
/joy_node/set_logger_level
/rosout/get_loggers
/rosout/set_logger_level
```

4. Source code analysis

- launch file

file path：/home/jetson/workspace/catkin_ws/src/jetbot_ros/launch/jetbotmini_joy.launch

```
<launch>
    <param name="use_sim_time" value="false"/>
    <node name="joy_node" pkg="joy" type="joy_node" output="screen"
respawn="false"/>
    <node name="jetbotmini_joy" pkg="jetbot_ros" type="jetbotmini_joy.py"
output="screen">
        <param name="linear_speed_limit" type="double" value="0.45"/>
        <param name="angular_speed_limit" type="double" value="2.0"/>
    </node>
</launch>
```

- python file

file path：/home/jetson/workspace/catkin_ws/src/jetbot_ros/scripts/jetbotmini_joy.py

```
    def buttonCallback(self, joy_data):
        '''
        Obtain the wireless controller receiving signal
        '''
        if not isinstance(joy_data, Joy): return
        # rospy.loginfo("joy_data.buttons:", joy_data.buttons)
        # rospy.loginfo("joy_data.axes:", joy_data.axes)
        self.user_jetson(joy_data)
```

Subscribe to the data published by the node [joy_node] to obtain the current status of the wired controller. Here is a distinction between different masters

5. Handle control effect


```
joy_data.buttons:
header:
  seq: 12
  stamp:
    secs: 1635406206
    nsecs: 343277556
  frame_id: "/dev/input/js0"
axes: [0.0, -0.0, 0.0, 0.0, 0.0, 0.0]
buttons: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Code analysis	Handle button	Car control
axes[0]	Left stick (left and right)	left and right
axes[1]	Left stick (up and down)	advance and back
buttons[0]	1	Servo S1 angle +10
buttons[1]	2	Servo S2 angle +10
buttons[2]	3	Servo S1 angle -10
buttons[3]	4	Servo S2 angle -10
buttons[4]	L1	Blue LED
buttons[5]	R1	Green LED
buttons[9]	START	buzzer

9.3.4 Wireless controller FPV remote control

The handle controls Jetbotmini, which is similar to the wired handle control; let the handle control node establish a connection with the Jetbotmini bottom driver node, change the current state of the handle, send different information to Jetbotmini, and drive Jetbotmini to make different responses. The node is the same as the wired controller, so I won't repeat it here.

In this section, we use the APP. You need to enter the following command to start the large program. After starting, you can hear three beeps.

```
sudo systemctl start jetbotmini_start
```

1. Operating procedures

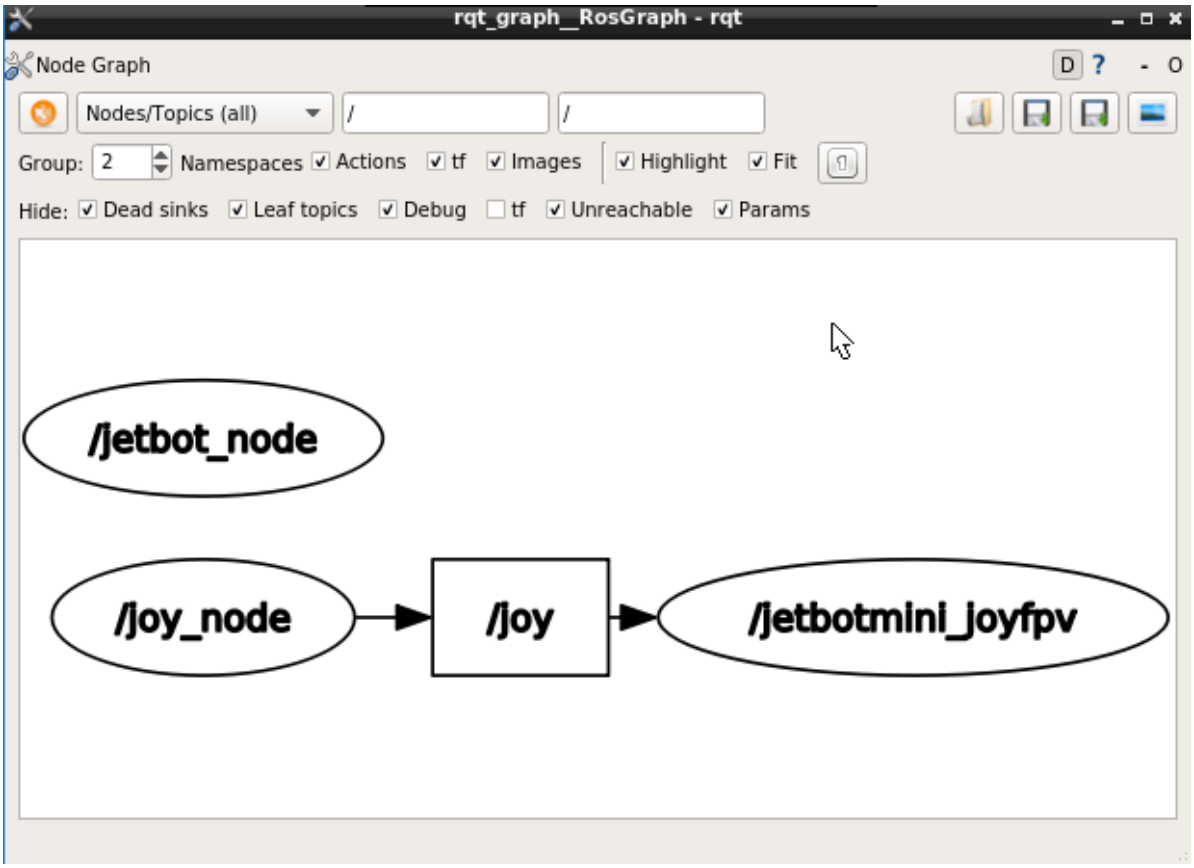
Start command

```
roslaunch jetbot_ros joyfpv_bringup.launch
```

View node graph

```
rqt_graph
```

Connection between topics



View service list

```
rosservice list
```

The print is as follows:

```
/Buzzer
/LEDBLUE
/LEDGREE
/Motor
/Servo
/jetbot_node/get_loggers
/jetbot_node/set_logger_level
/jetbotmini_joyfpv/get_loggers
/jetbotmini_joyfpv/set_logger_level
/joy_node/get_loggers
/joy_node/set_logger_level
/rosout/get_loggers
/rosout/set_logger_level
```

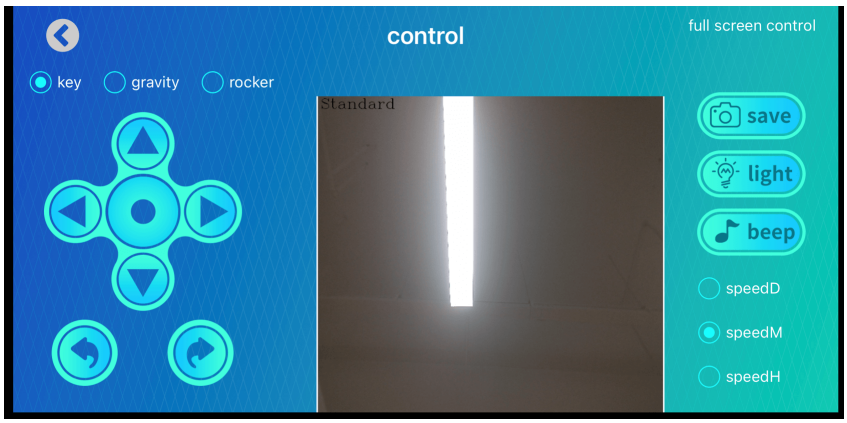
Now we can control Jetbot-mini in real time through the handle. Next we install the handle bracket.

Open the button at the bottom of the handle bracket and place it on the table.

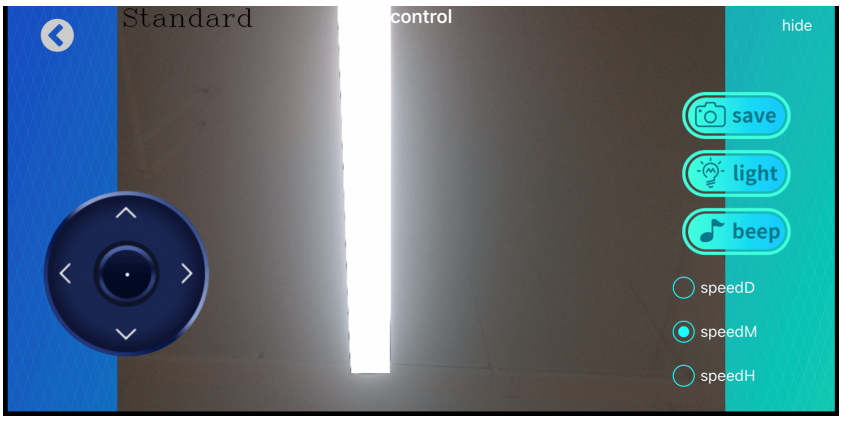
Put the handle into it.

Finally, re-fasten the button at the bottom.

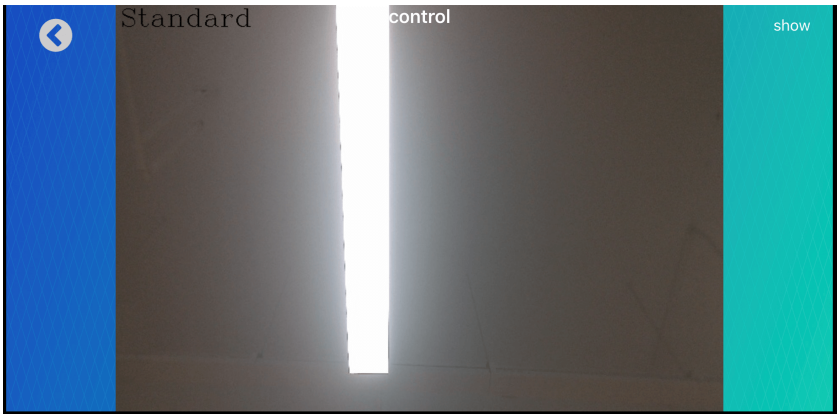
Next, we open the APP according to 2.5.2, after configuration, enter the remote control interface, click on the upper right corner [full screen control]



After entering the interface, click on the upper right corner [Hide]



Two fingers to zoom the screen to a suitable position, the effect is as follows.



Next, we install the mobile phone on the bracket. The top of the handle bracket is a mobile phone holder with an adjustable width, which can be widened upwards and then placed in the mobile phone. The spring of the bracket will automatically clamp the mobile phone firmly. The two plastic screws in the middle of the bracket are used to adjust the tilt angle of the mobile phone. After loosening, the display angle of the mobile phone can be adjusted, and then tighten the fixed angle.

Note: Before loosening the plastic screws, please take care to protect the phone. Due to the heavy weight of the phone, it may turn back. Please adjust it on the table.

2. Source code analysis

- launch file

  file path：/home/jetson/workspace/catkin_ws/src/jetbot_ros/launch/jetbotmini_joyfpv.launch

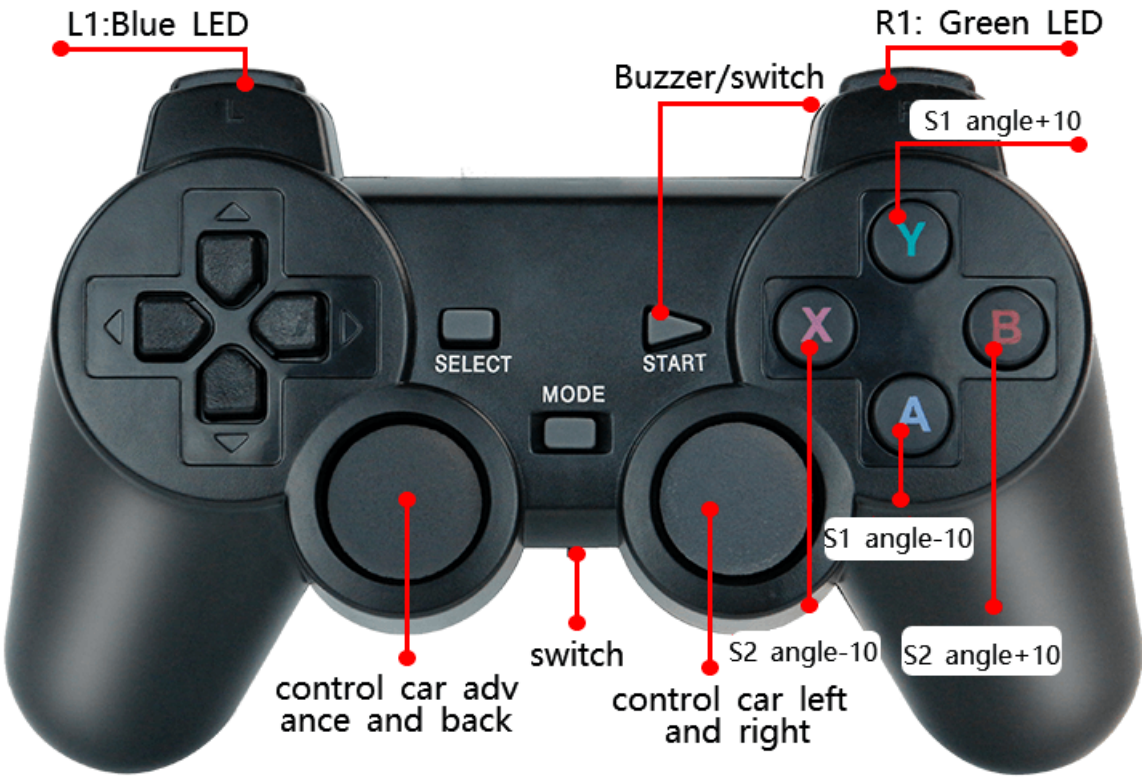
```
<launch>
    <param name="use_sim_time" value="false"/>
    <node name="joy_node" pkg="joy" type="joy_node" output="screen"
respawn="false"/>
    <node name="jetbotmini_joyfpv" pkg="jetbot_ros" type="jetbotmini_joyfpv.py"
output="screen">
        <param name="linear_speed_limit" type="double" value="0.45"/>
        <param name="angular_speed_limit" type="double" value="2.0"/>
    </node>
</launch>
```

- python file

file path：/home/jetson/workspace/catkin_ws/src/jetbot_ros/scripts/jetbotmini_joyfpv.py

Similar to the wired controller, the current status of the wireless controller can be obtained by subscribing to the data published by the node [joy_node]. Because the button values are different from the wired controller, only the buttons are modified here.

3. Handle control effect




```
joy_data.buttons:
header:
  seq: 12
  stamp:
    secs: 1635406206
    nsecs: 343277556
  frame_id: "/dev/input/js0"
axes: [0.0, -0.0, 0.0, 0.0, 0.0, 0.0]
buttons: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Code analysis	Handle button	Car control
axes[2]	Left stick	forward and backward
axes[1]	Right stick	left and right
buttons[4]	Y	servo S1 angle +10
buttons[1]	B	servo S2 angle +10
buttons[0]	A	servo S1 angle -10
buttons[3]	X	servo S2 angle -10
buttons[6]	L1	Blue LED
buttons[7]	R1	Green LED
buttons[11]	START	buzzer