

# Handle ROS1 course

**1、 Operating environment Operating System: Ubuntu 18.04**  
**ROS version: melodic Devices: Jetson nano, Raspberry Pi, PC**

## 2、 Install the driver

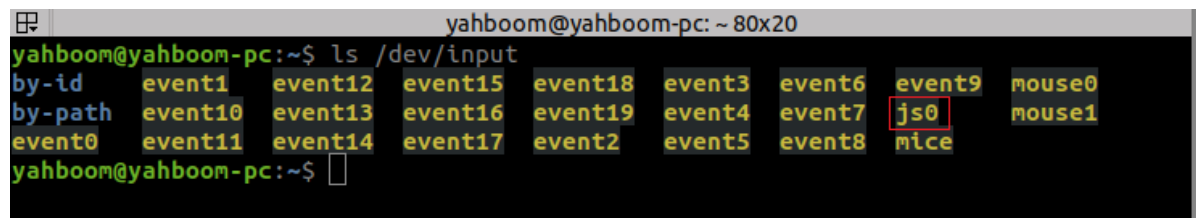
```
sudo apt install ros-melodic-joy ros-melodic-joystick-drivers
```

## 3、 Usage steps

Connect the USB end of the wireless controller to the device

- View Device

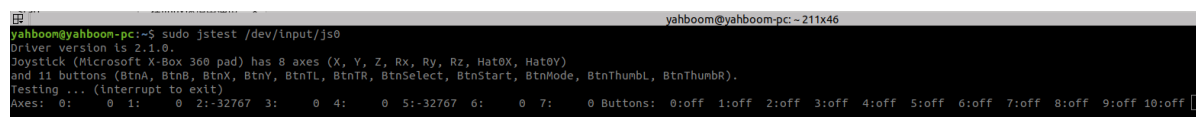
```
ls /dev/input
```



```
yahboom@yahboom-pc: ~ 80x20
yahboom@yahboom-pc:~$ ls /dev/input
by-id      event1     event12    event15    event18    event3     event6     event9     mouse0
by-path    event10    event13    event16    event19    event4     event7     js0        mouse1
event0     event11    event14    event17    event2     event5     event8     mice
yahboom@yahboom-pc:~$
```

- Test handle

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



```
yahboom@yahboom-pc:~$ sudo jstest /dev/input/js0
driver version is 2.1.0.
Joystick (Microsoft X-Box 360 pad) has 8 axes (X, Y, Z, Rx, Ry, Rz, Hat0X, Hat0Y)
and 11 buttons (BtnA, BtnB, BtnX, BtnY, BtnTL, BtnTR, BtnSelect, BtnStart, BtnMode, BtnThumbL, BtnThumbR).
Testing ... (Interrupt to exit)
Axes: 0: 0 1: 0 2:-32767 3: 0 4: 0 5:-32767 6: 0 7: 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9:off 10:off
```

If jstest is not installed, run the following command:

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

- Operation handle node

```
roscore step1
roslaunch joy joy_node Step3
rostopic echo joy Step3
```

```
yahboom@yahboom-pc: ~
roscore http://127.0.0.1:11311/105x27
yahboom@yahboom-pc:~$ roscore
... logging to /home/yahboom/.ros/log/cbdc2798-fe65-11eb-a390-00e070b234a2/roslaunch-yahboom-pc-17860.log
checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://127.0.0.1:40011/
ros_comm version 1.14.11

SUMMARY
=====
PARAMETERS
* /roscpp: melodic
* /rosversion: 1.14.11
NODES
auto-starting new master
process[master]: started with pid [17875]
ROS_MASTER_URI=http://127.0.0.1:11311/

setting /run_id to cbdc2798-fe65-11eb-a390-00e070b234a2
process[roscout-1]: started with pid [17891]
started core service [/roscout]
]

yahboom@yahboom-pc: ~ 105x23
RV_IP: 127.0.0.1
ROS_MASTER_URI:
http://127.0.0.1:11311
yahboom@yahboom-pc:~$ roslaunch joy joy_node
[ WARN] [1629099882.461224246]: Couldn't set gain on joystick force feedback: Bad file descriptor
[ INFO] [1629099882.403719387]: Opened joystick: /dev/input/js0. deadzone_: 0.050000.

yahboom@yahboom-pc:~$ rostopic echo joy
header:
  seq: 3
  stamp:
    secs: 1629099941
    nsecs: 459611872
  frame_id: "/dev/input/js0"
axes: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
buttons: [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
---
```

## 4、Handle control Little Turtle

Copy the wireless handle control function package to the workspace, compile and update the environment

```
Catkin_ Make # Compile
Source dev/setup. bash # Update environment
Note: Any modifications to C++code require this step to take effect.
```

Start Python code command

```
roslaunch joy_ctrl joy_turtlesim.launch
```

Start C++code command

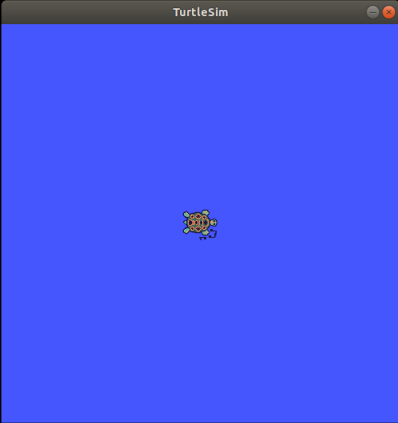
```
roslaunch joy_ctrl joy_turtle.launch
```

```
yahboom@yahboom-pc:~/catkin_ws$ roslaunch joy_ctrl joy_turtlesim.launch
... logging to /home/yahboom/.ros/log/2bb69516-fe68-11eb-a390-00e070b234a2/roslaunch-yahboom-pc-20754.log
checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://127.0.0.1:33765/

SUMMARY
=====
PARAMETERS
* /joy_turtlesim/angular_speed_limit: 2.0
* /joy_turtlesim/linear_speed_limit: 1.0
* /roscpp: melodic
* /rosversion: 1.14.11
* /use_sim_time: False
NODES
/
  joy_node (joy/joy_node)
  joy_turtlesim (joy_ctrl/joy_turtlesim.py)
  turtlesim_node (turtlesim/turtlesim_node)
ROS_MASTER_URI=http://127.0.0.1:11311

process[turtlesim_node-1]: started with pid [20774]
process[joy_node-2]: started with pid [20775]
process[joy_turtlesim-3]: started with pid [20776]
[ WARN] [1629100919.229082926]: Couldn't set gain on joystick force feedback: Bad file descriptor
[ INFO] [1629100919.238188656]: Opened joystick: /dev/input/js0. deadzone_: 0.050000.
[ INFO] [1629100919.248440536]: Starting turtlesim with node name /turtlesim_node
[ INFO] [1629100919.251340049]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

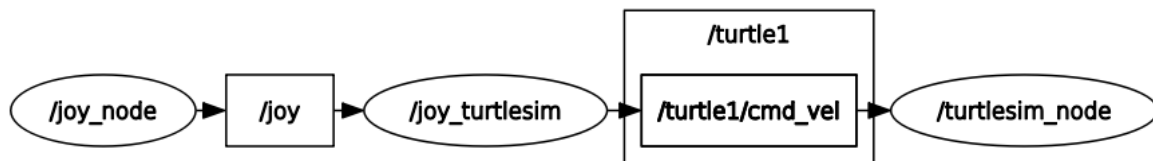


At this point, the handle can be used to control the operation of the little turtle.  
Corresponding relationship between the handle and the operation of the little turtle

Handle	Little turtle
Left rocker up	forward
Left rocker down	back
Right rocker left	turn left
Right rocker right	turn right

- View node diagram

rqt\_graph



## 5、Handle controlled turbobot

Due to the need to start gazebo, there is a high demand for device performance. It is recommended to use it on a PC, as Jetson nano and Raspberry pie may become particularly sluggish and may not run properly.

Install gazebo and turbobot simulations

```

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install gazebo9 libgazebo9-dev ros-melodic-turtlebot3*

```

Copy the wireless handle control function package to the workspace, compile and update the environment

```

Catkin_ Make # Compile
Source dev/setup. bash # Update environment
Note: Any modifications to C++code require this step to take effect.

```

Start Python code command

```

roslaunch joy_ctr1 joy_turtlesim.launch

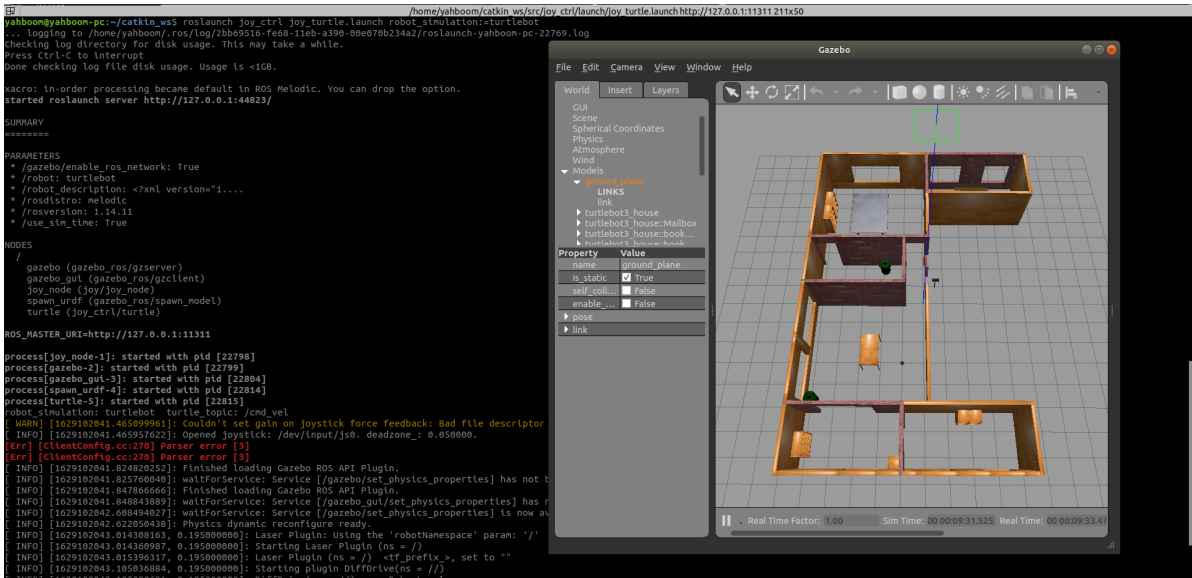
```

Start C++code command

```

roslaunch joy_ctr1 joy_turtle.launch robot_simulation:=turtlebot

```



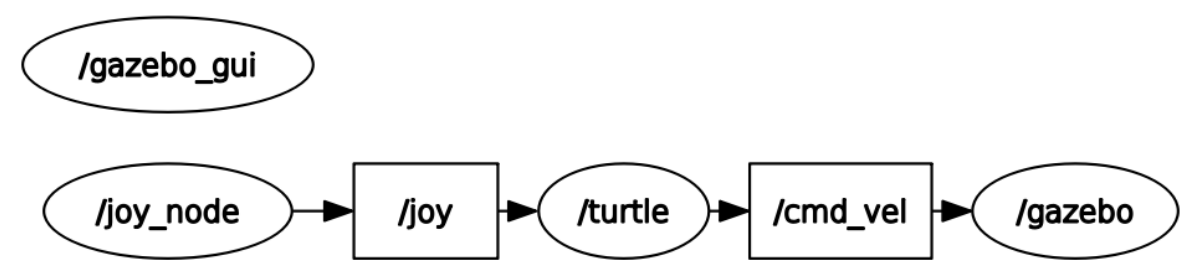
At this point, the handle can be used to control the operation of the turbobot.

Corresponding relationship between the handle and the operation of the little turtlebot

Handle	turtlebot
Left rocker up	forward
Left rocker down	back
Right rocker left	turn left
Right rocker right	turn right

- View node diagram

```
rqt_graph
```



- gazebo close command

```
killall gzserver gzclient
```

## Appendix

jetson nano

```
joy_data.buttons: header:
  seq: 335
  stamp:
    secs: 1628324636
    nsecs: 962988952
  frame_id: "/dev/input/js0"
axes: [0.0, 0.0, 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, 0, 0, 0]
```

axes (8)

Code Parsing	Handle buttons
axes[0]	Left rocker (left positive and right negative)
axes[1]	Left rocker (up positive and down negative)
axes[2]	Right rocker (left positive and right negative)
axes[3]	Right rocker (up positive and down negative)
axes[4]	
axes[5]	
axes[6]	Left button (left positive and right negative)
axes[7]	Left button (up positive and down negative)

buttons (15)

Code Parsing	Handle buttons
buttons[0]	A
buttons[1]	B
buttons[2]	
buttons[3]	X
buttons[4]	Y
buttons[5]	
buttons[6]	L1
buttons[7]	R1
buttons[8]	L2
buttons[9]	R2
buttons[10]	SELECT
buttons[11]	START
buttons[12]	
buttons[13]	Press left rocker
buttons[14]	Press right rocker

## Raspberry Pi

```
joy_data.buttons: header:
  seq: 264
  stamp:
    secs: 1628326479
    nsecs: 848359307
  frame_id: "/dev/input/js0"
axes: [-0.0, -0.0, 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]
```

axes (8)

Code Parsing	Handle buttons
axes[0]	Left rocker (left positive and right negative)
axes[1]	Left rocker (up positive and down negative)
axes[2]	L2(Press:-1 , release:1)
axes[3]	Right rocker (left positive and right negative)
axes[4]	Right rocker (up positive and down negative)
axes[5]	R2(Press:-1 , release:1)
axes[6]	Left button (left positive and right negative)
axes[7]	Left button (up positive and down negative)

buttons (11)

Code Parsing	Handle buttons
buttons[0]	A
buttons[1]	B
buttons[2]	X
buttons[3]	Y
buttons[4]	L1
buttons[5]	R1
buttons[6]	SELECT
buttons[7]	START
buttons[8]	MODE
buttons[9]	Press left rocker
buttons[10]	Press right rocker