

3、Robot control

1、Program function description

Open the chassis, run the handle/keyboard control program, you can control the trolley movement through the handle or keyboard, and the handle also controls the buzzer, controls the light strip and other functions.

2、Program code reference path

For Raspberry Pi PI5 master control, you need to enter the docker container first, but for Orin motherboard, you do not need to enter.

the location of the source code of the controller control function is located, taking the X3 model as an example

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_joy_X3.py
```

the location of the source code of the keyboard control function is located at,

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

3、The program starts

```
#The underlying driver  
ros2 run yahboomcar_bringup Mcnamu_driver_x3  
#Handle control  
ros2 run yahboomcar_ctrl yahboom_joy_X3  
ros2 run joy joy_node  
#Keyboard control  
ros2 run yahboomcar_ctrl yahboom_keyboard
```

It should be noted here that the controller and the keyboard cannot run at the same time, because after the keyboard control is started, when the keyboard is not pressed, the default is to send a message with 0 data of speed.

3.1、Controller control

After turning on, press the "START" button, hear the buzzer sound, and you can start the remote control. The remote control will enter sleep mode after being turned on for a period of time, and you need to press the "START" button to end sleep. If you want to control the trolley running, you also need to press the R2 key and release the motion control lock before you can use the joystick to control the trolley movement.

Remote control effect description,

handle	effect
Left stick up/down	Go straight forward/backward

handle	effect
Left stick left/right	Go straight left/right
Right stick left/right	Rotate left/rotate right
The right "1" key	Control the strip light effect
The right "2" key	Unlock/lock motion control
"START" button	Control the buzzer/end hibernation
The left stick is pressed	Adjust the speed of the X/Y axis
The right stick is pressed	Adjust the angular velocity magnitude

3.2、 keyboard control

Key description,

- Direction control

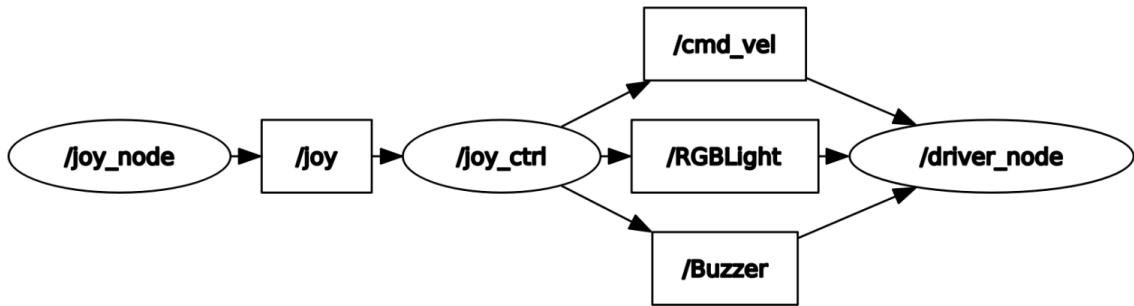
【i】 or 【I】	【linear,0】	【u】 or 【U】	【linear,angular】
【,】	【-linear,0】	【o】 or 【O】	【linear,-angular】
【j】 or 【J】	【0, angular】	【m】 or 【M】	【-linear,-angular】
【l】 or 【L】	【0, -angular】	【.】	【-linear,angular】

- Speed control

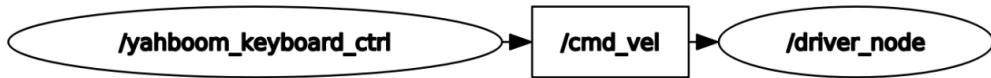
Key	Speed changes	Key	Speed changes
【q】	10% increase in both linear and angular velocities	【z】	10% reduction in both linear and angular velocities
【w】	Only 10% increase in line speed	【x】	Only 10% reduction in line speed
【e】	Only 10% increase in angular velocity	【c】	Only 10% reduction in angular velocity
【t】	Linear speed X/Y direction switching	【s】	Stop keyboard control

3.3、 node communication

Handle control trolley node communication diagram,



Keyboard control trolley node communication diagram,



4、core code parsing

4.1、Controller control code

Taking the X3 model as an example, we said in the previous lesson that the chassis control program is the `Mcnamu_driver_X3.py`, defining three subscribers: speed (`/cmd_vel`), light strip effect (`/RGBLight`), and buzzer (`/buzzer`), so we can control speed, light strip and buzzer as long as we publish this type of topic data `yahboom_joy_X3.py` the controller control code program.

```

#create pub
self.pub_goal = self.create_publisher(GoalID,"move_base/cancel",10)
self.pub_cmdVel = self.create_publisher(Twist,'cmd_vel', 10)
self.pub_Buzzer = self.create_publisher(Bool,"Buzzer", 1)
self.pub_RGBLight = self.create_publisher(Int32,"RGBLight" , 10)
self.pub_JoyState = self.create_publisher(Bool,"JoyState", 10)

```

In addition, we need to subscribe to the "joy" topic data, which tells us that the keys (joysticks and keys) have changed, that is,

```

#create sub
self.sub_Joy = self.create_subscription(Joy,'joy', self.buttonCallback,10)

```

The main look is the callback function of this joy topic, which parses the received value, then assigns the value to the publisher's variable, and finally publishes it.

```

def buttonCallback(self, joy_data):
    if not isinstance(joy_data, Joy): return
    if self.user_name == "root": self.user_jetson(joy_data)
    else: self.user_pc(joy_data)

```

The function jumps here are self.user_jetson, and the parameter variable passed in is the topic received,

```

def user_jetson(self, joy_data):

```

Take the control buzzer as an example for analysis,

```

if joy_data.buttons[11] == 1:
    Buzzer_ctrl = Bool()
    self.Buzzer_active=not self.Buzzer_active
    Buzzer_ctrl.data =self.Buzzer_active
    for i in range(3): self.pub_Buzzer.publish(Buzzer_ctrl)

```

It is judged here that if it is **joy_data.buttons[11] == 1**, that is, if "start" is pressed, then the value of the buzzer will change, and then publish **self.pub_Buzzer.publish(Buzzer_ctrl)**. In other cases, the principle is the same, they are all assigned by detecting the change of the key value. Refer to yahboom_joy_X3.py for detailed code.

4.2、 keyboard control code

The keyboard control can only control the movement control of the trolley, not the light strip and buzzer of the trolley, therefore, there is only one / cmd_vel speed publisher,

```

self.pub = self.create_publisher(Twist, 'cmd_vel', 1)

```

The program also defines two dictionaries to detect the change of the letters of the keyboard when pressed,

```

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),
}

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),
}

```

Entering the while loop, the program will read the value pressed by the keyboard, and then make judgments layer by layer.

```

key = yahboom_keyboard.getKey()
if key=="t" or key == "T": xspeed_switch = not xspeed_switch
elif key == "s" or key == "S":
...
if key in moveBindings.keys():
...
elif key in speedBindings.keys():
...

```

Finally, according to the judgment of multiple layers, assign values to twist.linear.x, twist.linear.y, twist.angular.z and then publish.

```

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())

```

Refer to yahboom_keyboard.py for detailed code.

Precautions for Handle Use

- When connecting the USB joystick receiver, it is recommended to connect it to the outermost USB-HUB expansion board, not directly to the motherboard or the middle USB-HUB expansion board (X3plus). If it is directly connected to the motherboard or the middle USB-HUB expansion board (X3plus), it will seriously interfere with the joystick signal reception due to the aluminium alloy grill on the top and bottom.
- After plugging and unplugging the joystick receiver head, it is necessary to restart the joystick programme, otherwise it will not be able to control the trolley.
- After starting the joystick control programme, if the joystick can not control the trolley, it may be caused by the incorrect control mode of the joystick, you can long press the joystick mode button for about 15 seconds to switch the mode, and when the green indicator light is always on, then press the start button again, if the buzzer sounds, it means that the switching is

successful, if there is no response, you can long press the joystick mode button 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:

The screenshot shows the Gamepad Tester interface. At the top, there are three sections labeled 'PLAYER 1', 'PLAYER 2', and 'PLAYER 3'. 'PLAYER 1' shows an 'Xbox Controller' connected. 'PLAYER 2' and 'PLAYER 3' show 'None detected'. Below this, a large image of an Xbox 360 controller is displayed. A red box highlights the 'Xbox 360 Controller (XInput STANDARD GAMEPAD)' text above the controller. Another red box highlights the green power LED on the left side of the controller. The controller's buttons and joysticks are visible.



- After re-plugging the joystick receiver or rebooting the motherboard, the joystick will reset to factory mode, and if it is not controllable, the mode will need to be re-switched each time it is plugged in or rebooted.
- 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.