3. Robot control

3.1. Program function description

Turn on the chassis and run the handle/keyboard control program. You can control the robot movement through the handle or keyboard. The handle can also control the buzzer, light strip, etc.

3.2, Program code reference path

After SSH connects to the car, the source code of the handle control function is located at,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_joy.py
```

After SSH connects to the car, the source code of the keyboard control function is located at,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_keyboard.
py
```

3.3, Program startup

```
#Start chassis
ros2 run yahboomcar_bringup Mcnamu_driver

#Handle control
#Start launch file
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py
#Or start two nodes separately
ros2 run yahboomcar_ctrl yahboom_joy
ros2 run joy joy_node

#Keyboard control
ros2 run yahboomcar_ctrl yahboom_keyboard
```

Note that the handle and 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 of 0 data of speed.

3.3.1, handle control

After turning it on, press the [START] button, and when you hear the buzzer sound, you can start remote control. **The remote control will enter sleep mode after being turned on for a period of time. You need to press the [START] button to end the sleep mode**. If you want to **control the car to run**, you also need to **press the [L1] key** to **release the motion control lock** before you can use the joystick to control the car to move.

Remote control effect description,

Handle	Effect
Left joystick up/down	Go forward/backward
Left joystick left/right	Go left/right
Right joystick left/right	Rotate left/turn right
【L1】 key	Release/lock motion control
【R1】 key	Control lighting effects
【START】 key	Control buzzer/end sleep
Press left joystick	Adjust X/Y axis speed
Press right joystick	Adjust angular velocity

3.3.2, Keyboard control

Button description,

- Direction control

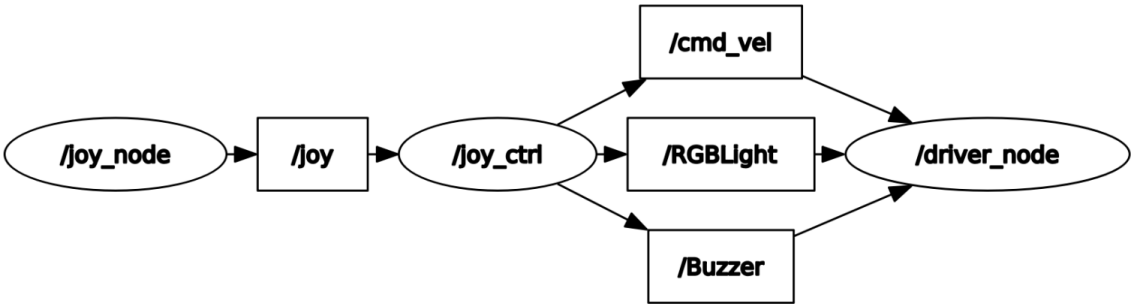
【i】 or 【I】	【 linear, 0】	【u】 or 【U】	【linear, angular】
【,】	【-linear, 0】	【o】 or 【O】	【linear, - angular】
【j】 or 【J】	【0, angular】	【m】 or 【M】	【- linear, - angular】
【l】 or 【L】	【0, - angular】	【.】	【- linear, angular】

- Speed control

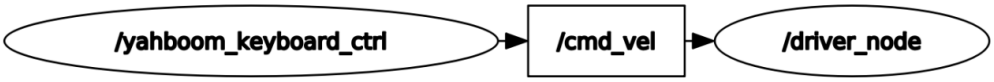
Key	Speed change	Key	Speed change
【q】	Increase both linear and angular speeds by 10%	【z】	Decrease both linear and angular speeds by 10%
【w】	Increase only linear speed by 10%	【x】	Decrease only linear speed by 10%
【e】	Increase only angular speed by 10%	【c】	Decrease only angular speed by 10%
【t】	Switch between linear speed X-axis and Y-axis	【s】	Stop keyboard control

3.3.3, Node Communication

Controller Control Car Node Communication Diagram,



Keyboard Control Car Node Communication Diagram,



3.4, Core Source Code Analysis

3.4.1, Controller Control Code

As we said in the last lesson, the chassis control program, that is, Mcnamu_driver.py, defines speed (/cmd_vel), lighting effects (/RGBLight), buzzer (/Buzzer), so we can control the speed, lighting effect and buzzer by publishing this type of topic data in the handle control code program yahboom_joy.py.

```
#create pub
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" , 1)
self.pub_JoyState = self.create_publisher(Bool,"JoyState", 10)
```

In addition, we need to subscribe to the "joy" topic data, which can tell us which key values (joystick and buttons) have changed, that is,

```
#create sub
self.sub_Joy = self.create_subscription(Joy,'joy', self.buttonCallback,1)
```

The main thing to look at is the callback function of this joy topic, which parses the received value, assigns it 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_sunrise(joy_data)
else: self.user_pc(joy_data)
```

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

```
def user_sunrise(self, joy_data):
```

Take the control of the buzzer as an example for analysis,

```
if joy_data.buttons[7] == 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)
```

Here, if **joy_data.buttons[7] == 1** That is, if the [START] key is pressed, the value of the buzzer will change and then be published **self.pub_Buzzer.publish(Buzzer_ctrl)**. The others are similar, and the principle is the same, all of which are assigned by **detecting the change of key value**. For detailed code, refer to yahboom_joy.py.

3.4.2, keyboard control code

Keyboard control can only control the movement of the car, not the lights and buzzer, so 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 changes when the letters on the keyboard are 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 on 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 multi-layer judgment, assign values to twist.linear.x, twist.linear.y, twist.angular.z and then publish them.

`python 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())` For detailed code, refer to yahboom_keyboard.py.