3. Robot information release

3.1 Description of program function

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

3.2 Application Code reference path

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

```
/userdata/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_joy.py
```

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

```
/userdata/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_keyboard.py
```

3.3. Program start

```
#启动底盘# Start chassis
ros2 run yahboomcar_bringup Mcnamu_driver

#手柄控制
#启动launch文件
# Handle control
# launch file
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py
#或单独启动两个节点# or start two nodes separately
ros2 run yahboomcar_ctrl yahboom_joy
ros2 run joy joy_node

#键盘控制# Keyboard control
ros2 run yahboomcar_ctrl yahboom_keyboard
```

It should be noted here that the gamepad 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 of 0 data speed.

3.3.1. Handles

After opening, press the [START] button, hear the buzzer sound, you can start the remote control.**The remote control will enter hibernation mode after being turned on for a period of time. You need to press [START] button to end hibernation** 。If you want to **control the car running** , you also need to **press the [L1] key** , lift the motion control lock， you can use the rocker to control the car moving.

Remote control effect description,

Gamepad	Effect
left rocker up/down	Go forward/backward straight
left rocker left/right	left/right go straight
Right rocker left/right	Rotate left/select right
【L1】key	Unlocks motion control
【R1】 key	to control the lighting effect
【START】 key	to control the buzzer/end sleep
left rocker press	X/Y axis adjustment speed
Right rocker press	to adjust the angular speed

3.3.2. Keyboard controls

Button description,

- Direction control

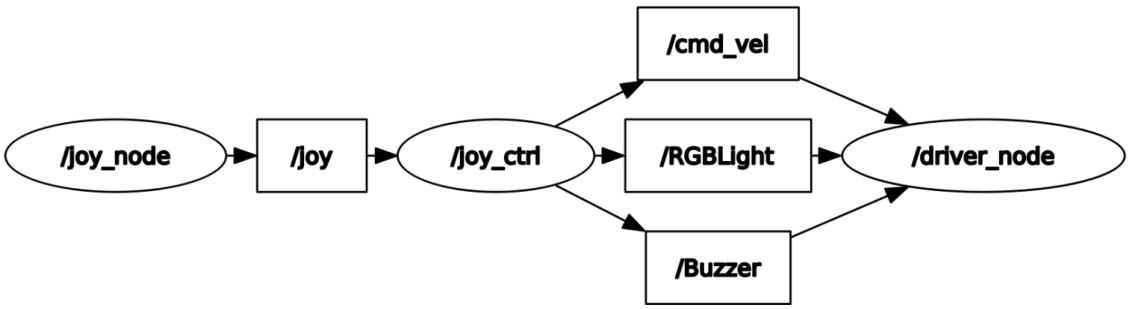
【i】or【I】	【 linear，0】	【u】or【U】	【linear，angular】
【,】	【-linear，0】	【o】or【O】	【linear，- angular】
【j】or【J】	【0， angular】	【m】or【M】	【- linear，- angular】
【l】or【L】	【0，- angular】	【.】	【 - linear，angular】

- Speed control

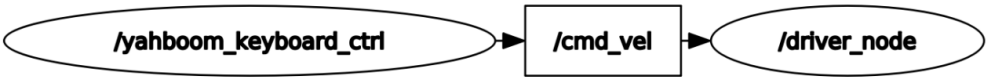
button	speed change	button	speed change
【q】	both linear velocity and angular velocity are increased by 10%	【z】	both linear velocity and angular velocity are decreased by 10%
【w】	only linear speed increased by 10%	【x】	only linear speed decreased by 10%
【e】	the angular velocity is increased by 10% only	【c】	the angular velocity is decreased by 10% only
【t】	line speed X axis /Y axis direction change	【s】	stop keyboard control

3.3.3 Node Communication

Handle control car node communication diagram,



Keyboard control car node communication diagram,



3.4. Core Source Code Parsing

3.4.1. Handles control code

As we mentioned in the last lesson, the chassis control program, Mcnamu_driver.py, defines three subscribers: speed (/cmd_vel), light effect (/RGBLight), and Buzzer (/ buzzer), so, We control the controller code program yahboom_joy.py to control the speed, light effect and buzzer as long as the type of topic data is posted.

```
#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 tells us which key values (joysticks and keys) 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 joy topic callback function, which parses the received value, then 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)
```

Here the function jumps to self.user_sunrise, and the argument variable passed in is the received topic,

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

Take control buzzer as an example to analyze,

```
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)
```

If **joy_data.buttons[7] == 1** If [START] is pressed, then the buzzer value will change and publish **self.pub_Buzzer.publish(Buzzer_ctrl)**. Other analogies, the principle is the same, are assigned by ** detection of the change in the key value **. See yahboom_joy.py for details.

3.4.2. Keyboard control code

The keyboard control can only control the motion control of the car, not the lights and buzzers, 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 changes in keyboard letters 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 reads the value of the keyboard press and makes a layer by layer judgment,

```
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 the multiple layers, the values are assigned to twist.linear.x, twist.linear.y, twist.angular.z and then published.

```
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())
```

See yahboom_keykey.py for details.