3.Robot control

1. Program function description

Turn on the chassis and run the handle/keyboard control program. You can control Muto movement through the handle or keyboard. The handle also has functions such as controlling the buzzer.

2. Program code reference path

After entering the docker container, the source code of the handle control function is located at

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahbo
om_joy.py
```

After entering the docker container, the source code of the keyboard control function is located at

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahbo
om_keyboard.py
```

3. Program startup

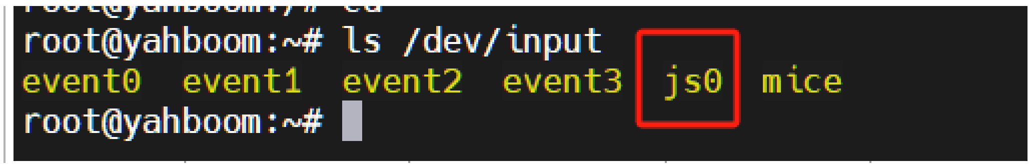
```
#Low-level driver
ros2 run yahboomcar_bringup muto_driver
#Handle control
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py
#keyboard control
ros2 run yahboomcar_ctrl yahboom_keyboard
```

Note:

1. The handle and keyboard cannot run at the same time, because after keyboard control is started, when the keyboard is not pressed, a speed message of 0 data will be sent by default

2. Make sure that the controller device has been mounted into the docker container and can be executed in the docker container:

```
ls /dev/input
```



```
root@yahboom:~# ls /dev/input
event0  event1  event2  event3  js0  mice
root@yahboom:~#
```

If the js0 device exists, it means it has been successfully mounted.

3. After the js0 device has been successfully mounted, do not switch the receiver of the plug-and-play controller to other USB ports of the main control. This will also easily cause the controller to lose connection.

3.1. Handle control

After turning it on, press the "START" button and hear the buzzer sound to start 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 operation of the car**, you also need to **press the R2 key** and **release the motion control lock** before you can use the joystick to control the movement of the car.

Remote control effect description,

Handle	Effect
Left joystick up/down	Forward/Backward to go straight
Left joystick left/right	Left/right to go straight
Right joystick left/right	Left rotation/right rotation
Right "2" key	Unlock/lock motion control
"START" button	Control buzzer/end sleep
Press the left joystick	Adjust the X/Y axis speed
Press the right joystick	Adjust the angular velocity

3.2.Keyboard control

Key Description,

- Directional control

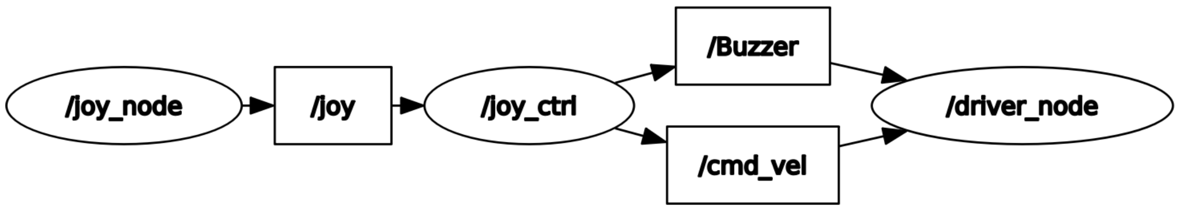
【i】或【I】	【linear，0】	【u】或【U】	【linear，angular】
【,】	【-linear，0】	【o】或【O】	【linear，- angular】
【j】或【J】	【0， angular】	【m】或【M】	【- linear，- angular】
【l】或【L】	【0，- angular】	【.】	【 - linear，angular】

- Speed control

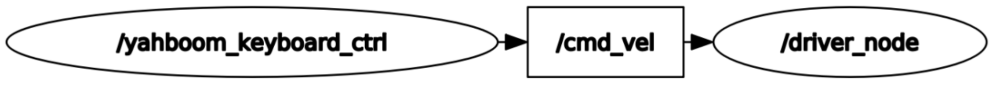
按键	速度变化	按键	速度变化
【q】	线速度和角速度都增加10%	【z】	线速度和角速度都减少10%
【w】	仅线速度增加10%	【x】	仅线速度减少10%
【e】	仅角速度增加10%	【c】	仅角速度减少10%
【t】	线速度X轴/Y轴方向切换	【s】	停止键盘控制

3.3. Node communication

Communication diagram of handle control car node,



Keyboard control car node communication diagram,



4. Core code analysis

4.1. Controller code

The chassis control program is muto_driver.py, which defines two subscribers: speed (/cmd_vel) and buzzer (/Buzzer).Therefore, as long as we publish this type of topic data in the controller control code program yahboom_joy.py, we can control the speed and buzzer.

```
#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_JoyState = self.create_publisher(Bool,"JoyState",  10)
```

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

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

The main thing to look at is the callback function of this joy topic. It 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_jetson(joy_data)
    else: self.user_pc(joy_data)
```

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

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

Take controlling the 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)
```

Here it is judged that if **joy_data.buttons[11] == 1** that is, if "start" is pressed, the value of the buzzer will change and then be published **self.pub_Buzzer.publish(Buzzer_ctrl)** . The others are deduced by analogy, and the principle is the same. They all assign values by detecting changes in key values. For detailed code, refer to yahboom_joy.py.

4.2. Keyboard control code

Keyboard control can only control Muto's motion control, not Muto's buzzer. 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 changes in keyboard letters when they 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 of the keyboard press, 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, based on multi-layer judgment, assign values to twist.linear.x, twist.linear.y, twist.angular.z and then publish them.

```
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())
```

Detailed code reference : yahboom_keyboard.py