2、Handle controlled robot

Use our company's Rosmaster-X3 car to illustrate how to control a robot or car through a handle

1、Compile Feature Pack

Using the driver in the root directory_ Taking ws as an example, first create a new workspace and src folder,

```
mkdir driver_ws
cd    driver_ws
mkdir src
cd src
```

Copy the folder source code to the src folder, and then enter the following command to compile,

```
colcon build
```

Then add the feature package path to the environment variable,

```
sudo gedit ./.bashrcs
```

Copy the following content to the end of the file,

```
ssource ~/driver_ws/install/setup.bash
```

Exit after saving and refresh environment variables,

```
source ~/.bashrc
```

2、Run Case Code

To activate this function, it is necessary to install Rosmaster_ Library files for lib and binding serial ports. This is just for demo demonstration and code analysis,

```
#Running the bottom drive code of the x3 trolley
ros2 run yahboomcar_bringup Mcnamu_driver_X3
#Running operating handle control code
ros2 run yahboomcar_ctrl yahboom_joy_X3
ros2 run joy joy_node
```

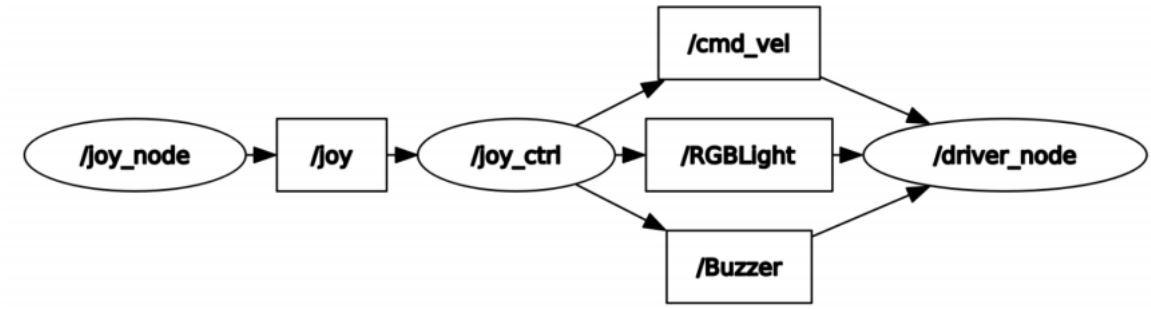
After turning on, press the "START" button and hear the buzzer sound to start remote control. After the remote control is turned on for a period of time and not in use, it will enter sleep mode. You need to press the "START" button to end sleep. If you want to control the movement of the car, you need to press the R2 key to release the motion control lock before you can use the joystick to control the car's movement.

Remote control effect description,

handle	Car motion effect
Left joystick up/down	Forward/backward straight ahead
Left joystick left/right	Go straight left/right
Right joystick left/right	Left/Right Rotation
Right '1' button	Control light strip effect
Right '2' button	Release/Lock Motion Control
“START” button	Control buzzer/end sleep
Left joystick pressed	Adjusting the speed of the X/Y axis
Right joystick pressed	Adjusting the angular velocity size

View node communication diagram,

```
ros2 run rqt_graph rqt_graph
```



3、Core code parsing

Source code location：~/driver_ws/src/yahboomcar_ctrl/yahboomcar_ctrl/yahboom_joy_X3.py

```
#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 can tell us which key values (joystick and buttons) have changed, namely,

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

The main focus is on the callback function of this joy topic, which parses the received value, assigns it to the variable of the publisher, 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)
```

All function jumps here are self. user_ Jetson, the incoming parameter variable is the received topic,

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

Taking 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)
```

Here's how to determine if it's " joy_ Data. buttons [11]==1 " That is, if "start" is pressed, the value of the buzzer will change and be published as "self. pub_ Buzzer. publish (Buzzer_ctrl)".The principle is the same for the others, which are all assigned by detecting changes in key values.Detailed code reference "yahboom_ Joy_ X3.py" file