Robot handle control

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check [Must read before use] to set the IP and ROS_DOMAIN_ID on the board.
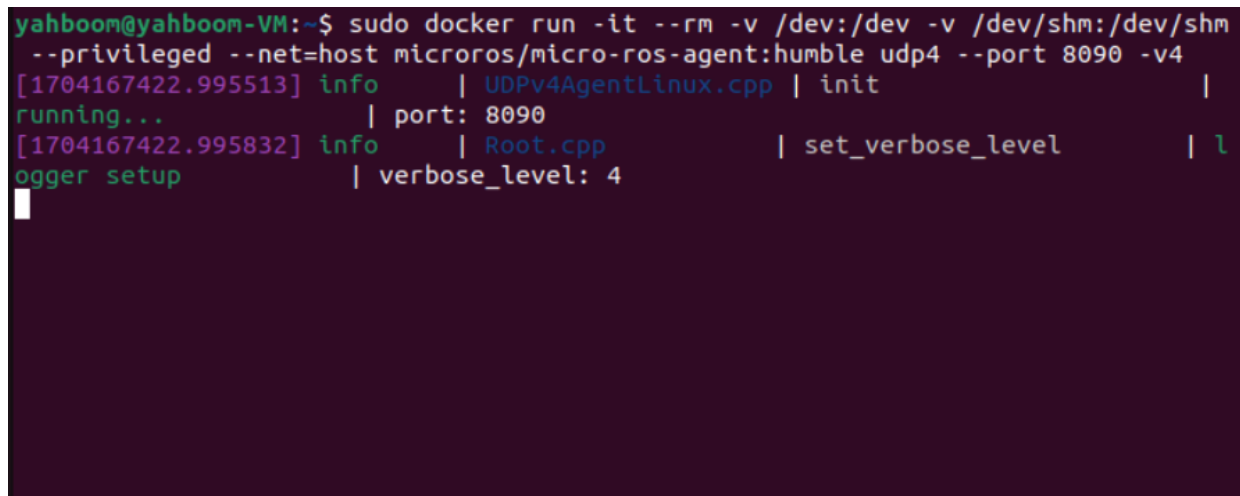
1、 Program function description

The car is connected to the agent, the receiver of the handle is connected to the virtual machine port/Raspberry Pi 2 port, and the program is run, and the car movement, buzzer and gimbal servo can be controlled remotely.

2、 Start and connect to the agent

Taking the supporting virtual machine as an example, enter the following command to start the agent:

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host
microros/micro-ros-agent:humble udp4 --port 8090 -v4
```


Then, turn on the car switch and wait for the car to connect to the agent. The connection is successful, as shown in the figure below.

```
[1702630014.015846] info     | ProxyClient.cpp   | create_participant    | participant created   | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp   | create_publisher      | publisher created     | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp   | create_datawriter     | datawriter created    | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp   | create_publisher      | publisher created     | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp   | create_datawriter     | datawriter created    | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp   | create_publisher      | publisher created     | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp   | create_datawriter     | datawriter created    | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp   | create_subscriber     | subscriber created    | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp   | create_datareader     | datareader created    | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp   | create_subscriber     | subscriber created    | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp   | create_datareader     | datareader created    | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp   | create_topic          | topic created         | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp   | create_subscriber     | subscriber created    | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp   | create_datareader     | datareader created    | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

3、 starting program

To connect the virtual machine to the controller receiver, you need to ensure that the virtual machine can recognize the controller receiver. As shown in the figure below, the connection is successful.


```
yahboom@yahboom-VM:~$ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 0079:181c DragonRise Inc. Controller
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

If it is not connected, check [Virtual Machine]->[Removable Devices] in the menu bar above the virtual machine toolbar and check whether [DragonRise Controller] is checked.

Enter the following command in the terminal to start the program.

```
#Handle remote control car program
ros2 run yahboomcar_ctrl yahboom_joy
#Get remote sensing data
ros2 run joy joy_node
```
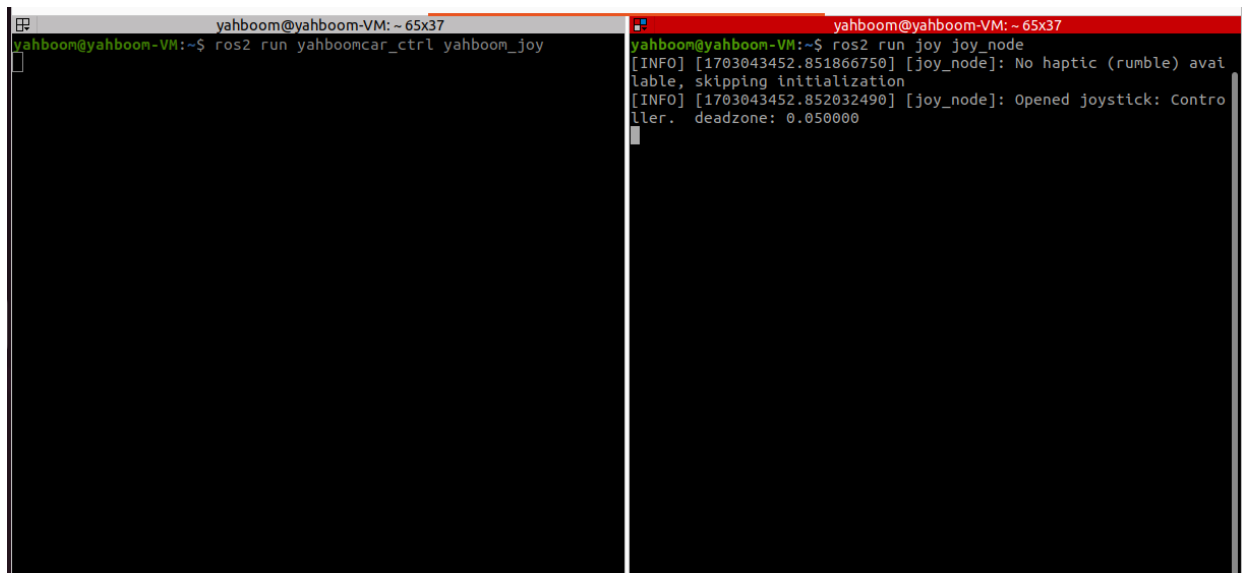
The remote control button description is as follows:

- Left rocker: valid in the front and rear directions, controls the car forward and backward, invalid in the left and right directions

- Left rocker: valid in the front and rear directions, controls the car forward and backward, invalid in the left and right directions

- START key: buzzer control

- Y key: control the S2 servo upward

- A key: control the S2 servo down

- X key: control the S1 servo to the left

- B key: control the S1 servo to the right

- R1 key: handle control speed switch. After pressing it, you can use the remote control to control the speed of the car. Press it again to lose the handle control speed. It is also a gameplay switch. Press it to stop. Press it again to continue running the functional gameplay program, including radar. Obstacle avoidance, radar guard and more.

- MODE key: Switch modes and use the default mode. After switching modes, if the key value is incorrect, the program will exit with an error.

4、 Node communication diagram

Terminal input,

```
ros2 run rqt_graph rqt_graph
```
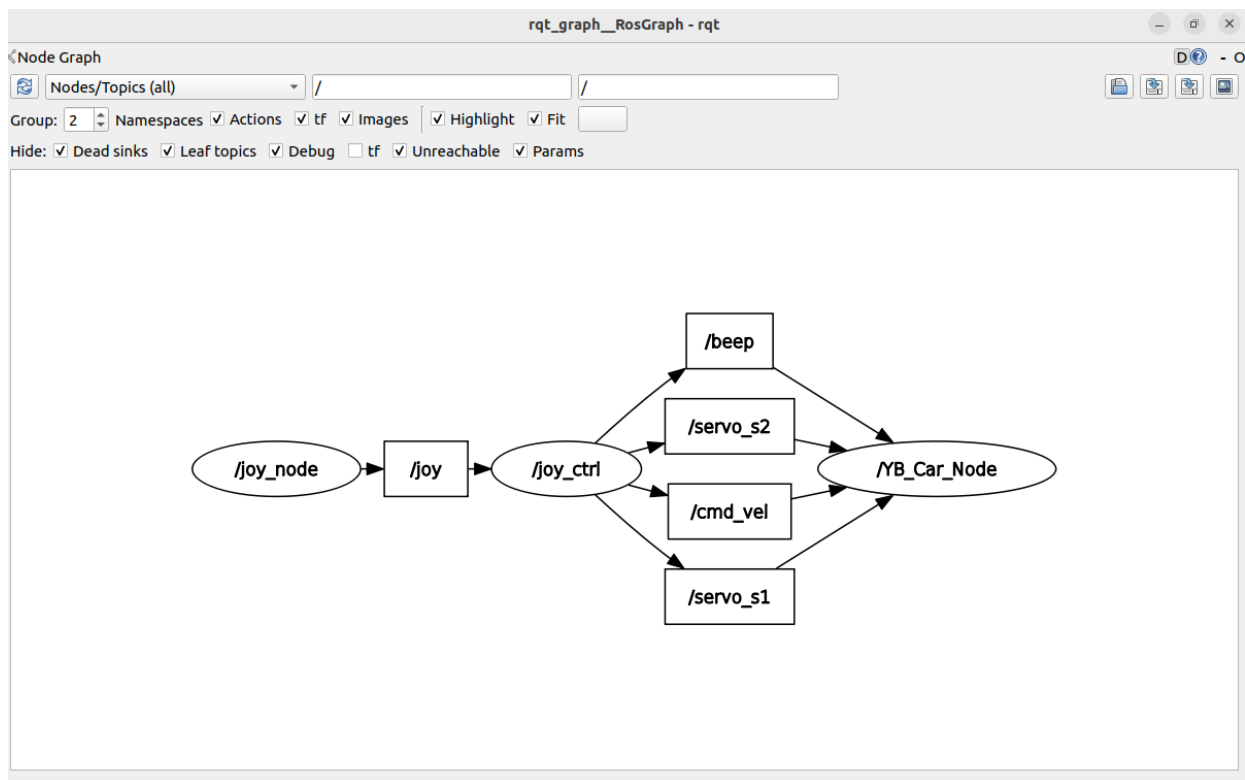
If it is not displayed at first, select [Nodes/Topics(all)], and then click the refresh button in the upper left corner.

5、 Code analysis

Source code reference path (already equipped with a virtual machine as an example),

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_ctrl/yahboomcar_ctrl
```

yahboom_joy.py

```python
##Create subscriber subscription/joy topic data
self.sub_Joy = self.create_subscription(Joy,'joy', self.buttonCallback,10)

#Create a publisher to publish speed, servo, buzzer topic data and function gameplay
switches
self.pub_cmdVel = self.create_publisher(Twist,'cmd_vel',10)
self.pub_Buzzer = self.create_publisher(UInt16,"beep",  1)
self.pub_JoyState = self.create_publisher(Bool,"JoyState",  10)
self.pub_Servo1 = self.create_publisher(Int32,"servo_s1" , 10)
self.pub_Servo2 = self.create_publisher(Int32,"servo_s2" , 10)

#Callback
def buttonCallback(self,joy_data):
    if not isinstance(joy_data, Joy): return
    self.user_jetson(joy_data)

#Process key function user_jetson
def user_jetson(self, joy_data):
    #cancel nav
```

```python
        if joy_data.buttons[7] == 1: self.cancel_nav()
        #Buzzer control
        if joy_data.buttons[11] == 1:
            b = UInt16()
            self.Buzzer_active = not self.Buzzer_active
            b.data = self.Buzzer_active
            self.pub_Buzzer.publish(b)
    #Gear adjustment, press the rocker to adjust the gear
        xlinear_speed = self.filter_data(joy_data.axes[1]) * self.xspeed_limit *
self.linear_Gear
        angular_speed = self.filter_data(joy_data.axes[2]) * self.angular_speed_limit *
self.angular_Gear
        if xlinear_speed > self.xspeed_limit: xlinear_speed = self.xspeed_limit
        elif xlinear_speed < -self.xspeed_limit: xlinear_speed = -self.xspeed_limit
        if angular_speed > self.angular_speed_limit: angular_speed =
self.angular_speed_limit
        elif angular_speed < -self.angular_speed_limit: angular_speed = -
self.angular_speed_limit
        twist = Twist()
        twist.linear.x = xlinear_speed
        twist.linear.y = 0.0
        twist.angular.z = angular_speed
        #Determine whether the speed can be controlled, that is, whether the R1 key is
pressed
        if self.Joy_active == True:
            self.pub_cmdVel.publish(twist)
        #The following is the data for processing the servo control. After pressing it,
the angle will increase/decrease by 1. If you continue to press it, the angle will
continue to increase and decrease.
        if not joy_data.buttons[1] == 0:
            print("Up")
            self.PWMServo_X += 1
            if self.PWMServo_X <= -90: self.PWMServo_X = -90
            elif self.PWMServo_X >= 90: self.PWMServo_X = 90
            print("self.PWMServo_X: ",self.PWMServo_X)
            print("self.PWMServo_Y: ",self.PWMServo_Y)
            servo1_angle = Int32()
            servo1_angle.data = self.PWMServo_X
            self.pub_Servo1.publish(servo1_angle)

        if not joy_data.buttons[3] == 0:
            print("Down")
            self.PWMServo_X -= 1
            if self.PWMServo_X <= -90: self.PWMServo_X = -90
            elif self.PWMServo_X >= 90: self.PWMServo_X = 90
            print("self.PWMServo_X: ",self.PWMServo_X)
            print("self.PWMServo_Y: ",self.PWMServo_Y)
            servo1_angle = Int32()
            servo1_angle.data = self.PWMServo_X
            self.pub_Servo1.publish(servo1_angle)

        if  not joy_data.buttons[0] == 0:
```

```
        print("Left")
        self.PWMServo_Y -= 1
        if self.PWMServo_Y <= -90: self.PWMServo_Y = -90
        elif self.PWMServo_Y >= 20: self.PWMServo_Y = 20
        servo2_angle = Int32()
        servo2_angle.data = self.PWMServo_Y
        self.pub_Servo2.publish(servo2_angle)
        print("self.PWMServo_X: ",self.PWMServo_X)
        print("self.PWMServo_Y: ",self.PWMServo_Y)

    if  not joy_data.buttons[4] == 0:
        print("Right")
        self.PWMServo_Y += 1
        if self.PWMServo_Y <= -90: self.PWMServo_Y = -90
        elif self.PWMServo_Y >= 20: self.PWMServo_Y = 20
        servo2_angle = Int32()
        servo2_angle.data = self.PWMServo_Y
        self.pub_Servo2.publish(servo2_angle)
        print("self.PWMServo_X: ",self.PWMServo_X)
        print("self.PWMServo_Y: ",self.PWMServo_Y)
```

6、Variables corresponding to remote control key values

Based on the default mode [Controller], the key values corresponding to the remote control are as follows:

remote control event	Corresponding variable
Left joystick up	axes[1]=1
Left joystick down	axes[1]=-1
Right joystick to left	axes[2]=1
Right joystick to the right	axes[2]=-1
Button X pressed	button[3]=1
Button B pressed	button[1]=1
Button Y pressed	button[4]=1
Button R1 pressed	button[7]=1
start button pressed	button[11]=1
Left joystick pressed	button[13]=1
Right joystick pressed	button[14]=1

Combined with the above source code for easy understanding, when these values change, it means that the remote control is pressed and the corresponding program can be executed. View other button presses, subscribe to/joy topics, terminal input,

```
ros2 topic echo /joy
```

```
header:
  stamp:
    sec: 1703053071
    nanosec: 270512642
  frame_id: joy
axes:
- -0.0
- -0.0
- -0.0
- -0.0
- 1.0
- 1.0
- 0.0
- 0.0
buttons:
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
---
```