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 the same. You can check [Must-read before use] to set the IP and ROS_DOMAIN_ID on the board.

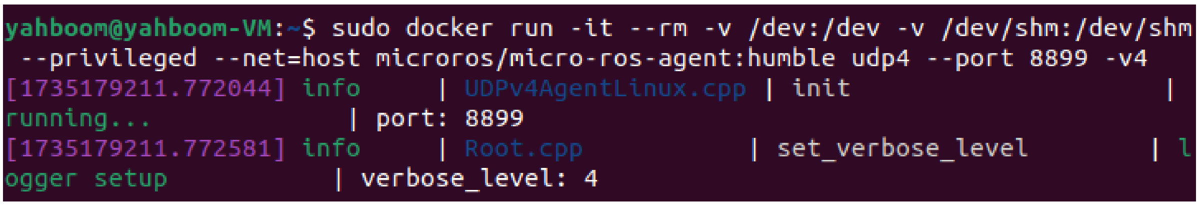
1. Program function description

Connect the car to the proxy, connect the handle's receiver to the virtual machine port, run the program, and you can use the remote control to control the car's movement and buzzer.

2. Start and connect the agent

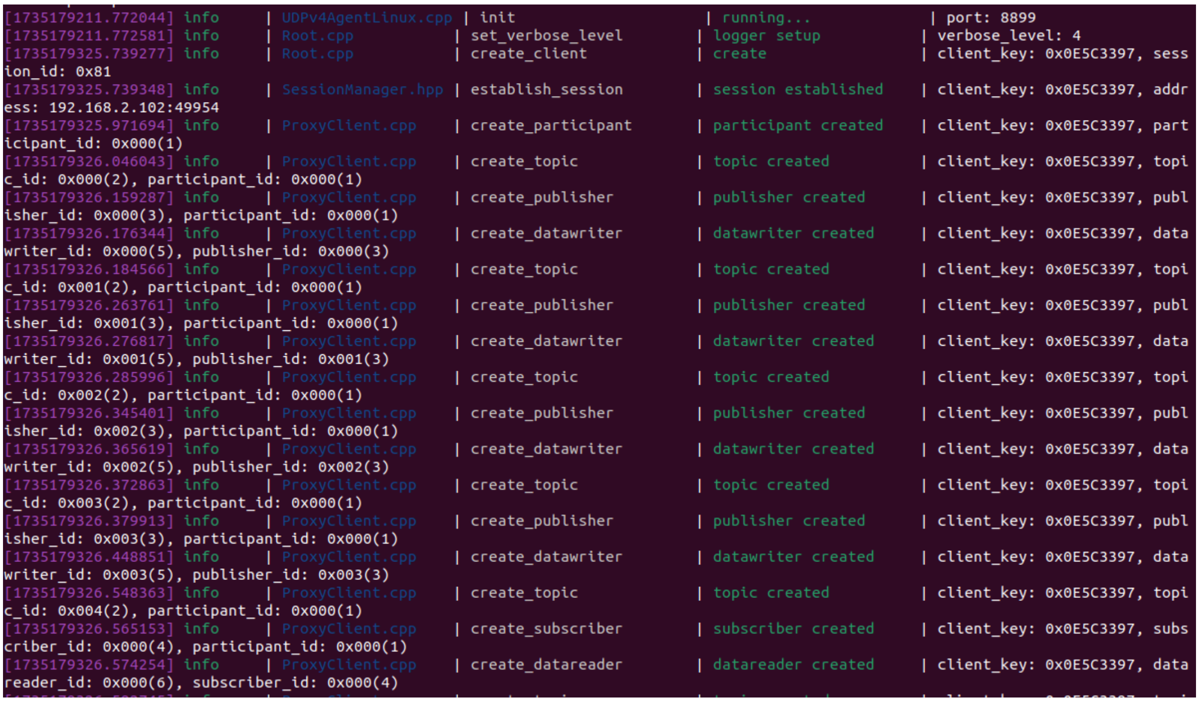
Take the supporting virtual machine as an example, enter the following command to start the agent,

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
```



```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     |
running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level        | l
ogger setup            | verbose_level: 4
```

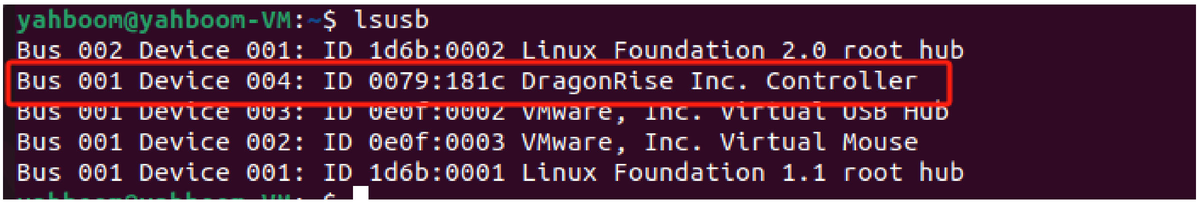
Then, turn on the car switch and wait for the car to connect to the agent. The connection is successful as shown in the figure below,



```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                    | running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level       | logger setup           | verbose_level: 4
[1735179325.739277] info     | Root.cpp            | create_client           | create                 | client_key: 0x0E5C3397, sess
ion_id: 0x81
[1735179325.739348] info     | SessionManager.hpp | establish_session       | session established    | client_key: 0x0E5C3397, addr
ess: 192.168.2.102:49954
[1735179325.971694] info     | ProxyClient.cpp     | create_participant      | participant created    | client_key: 0x0E5C3397, part
icipant_id: 0x000(1)
[1735179326.046043] info     | ProxyClient.cpp     | create_topic            | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info     | ProxyClient.cpp     | create_publisher        | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info     | ProxyClient.cpp     | create_datawriter       | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info     | ProxyClient.cpp     | create_topic            | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info     | ProxyClient.cpp     | create_publisher        | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info     | ProxyClient.cpp     | create_datawriter       | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info     | ProxyClient.cpp     | create_topic            | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info     | ProxyClient.cpp     | create_publisher        | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info     | ProxyClient.cpp     | create_datawriter       | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info     | ProxyClient.cpp     | create_topic            | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info     | ProxyClient.cpp     | create_publisher        | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info     | ProxyClient.cpp     | create_datawriter       | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info     | ProxyClient.cpp     | create_topic            | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info     | ProxyClient.cpp     | create_subscriber       | subscriber created     | client_key: 0x0E5C3397, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info     | ProxyClient.cpp     | create_datareader       | datareader created     | client_key: 0x0E5C3397, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
```

3. Start the program

To connect the virtual machine to the controller receiver, you need to ensure that the virtual machine can recognize the controller receiver. If it is connected as shown in the figure below,



```
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
yahboom@yahboom-VM:~$
```

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

3.1 Run command

Take the matching virtual machine as an example, input in the terminal,

Start the chassis driver,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

Open another terminal and input,

```
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py ••#Handle remote control car program
```

Observe the handle indicator light, if it is always on, it means the connection is successful. After the program starts, press [START] and the car buzzer will sound. Press the R1 key, and the terminal prints the following information, turning on the handle control. You can use the left joystick to control the car forward and backward; you can use the right joystick to control the car left and right;

```
[yahboom_joy-2] [INFO] [1735197789.731509428] [joy_ctrl]: Play:true
```

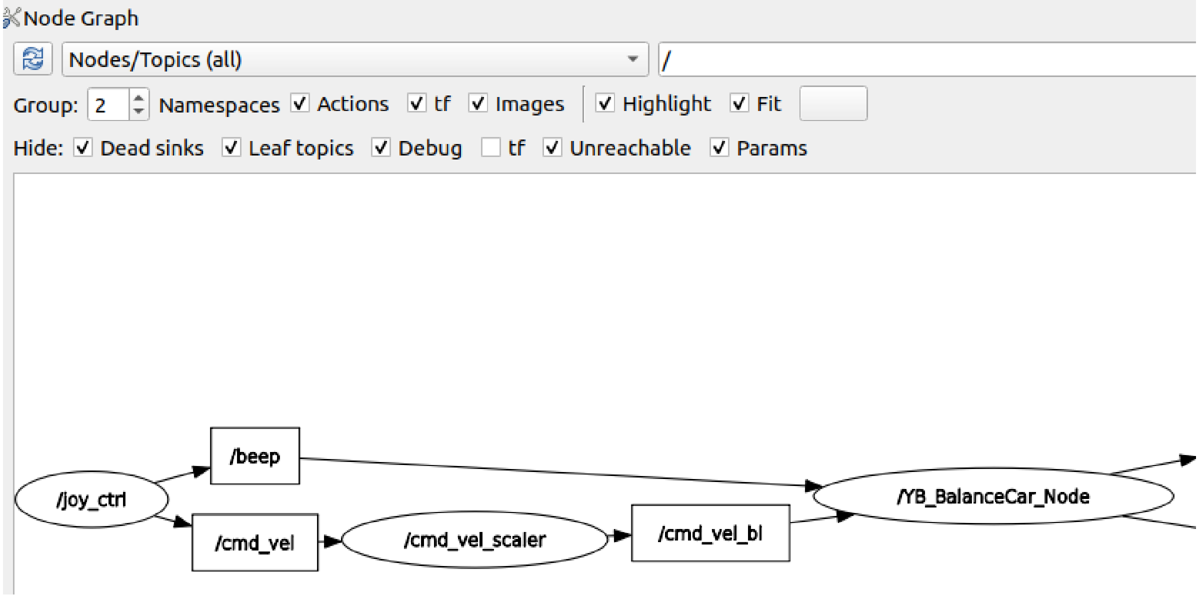
The remote control button description is as follows:

- Left joystick: front and back directions are valid, controlling the car forward and backward, and left and right directions are invalid
- Right joystick: left and right directions are valid, controlling the car to turn left and right, and front and back directions are invalid
- START button: buzzer control
- R1 button: handle control speed switch, press it to control the car speed with the remote control, press it again to lose the handle control speed, and it is also a play switch, press it to stop, press it again to continue running the function play program, including radar obstacle avoidance, radar guard, etc.
- MODE button: switch mode, use the default mode, after switching the mode, if the key value is incorrect, the program will report an error and exit.

4. Node communication graph

Terminal input,

```
ros2 run rqt_graph rqt_graph
```



Node Graph
Nodes/Topics (all)
/
Group: 2
Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit
Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params
/beep
/joy_ctrl
/cmd_vel
/cmd_vel_scaler
/cmd_vel_bl
/YB_BalanceCar_Node

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

5. Code analysis

Source code reference path (taking the supporting virtual machine as an example),

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

yahboom_joy.py

```
#!/usr/bin/env python
# encoding: utf-8

#public lib
import os
import time
import getpass
import threading
from time import sleep

#ros lib
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from sensor_msgs.msg import Joy
from actionlib_msgs.msg import GoalID
from std_msgs.msg import Int32, Bool, UInt16

class JoyTeleop(Node):
    def __init__(self, name):
        super().__init__(name)
        self.Joy_active = False
        self.Buzzer_active = 0
        self.cancel_time = time.time()
        self.user_name = getpass.getuser()
        self.linear_Gear = 1.0 / 2
        self.angular_Gear = 1.0 / 2

        # 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(UInt16, "beep", 1)
        self.pub_JoyState = self.create_publisher(Bool, "JoyState", 10)

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

        # declare parameter and get the value
        self.declare_parameter('xspeed_limit', 1.0) # 25
        self.declare_parameter('yspeed_limit', 1.0)
        self.declare_parameter('angular_speed_limit', 3.0) # 300
        self.xspeed_limit =
self.get_parameter('xspeed_limit').get_parameter_value().double_value
        self.yspeed_limit =
self.get_parameter('yspeed_limit').get_parameter_value().double_value
        self.angular_speed_limit =
self.get_parameter('angular_speed_limit').get_parameter_value().double_value

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

    def user_jetson(self, joy_data):
        # cancel nav
        if joy_data.buttons[7] == 1:
               self.cancel_nav()
        # Buzzer
        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)

        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
        if self.Joy_active == True:
            self.pub_cmdVel.publish(twist)

    def filter_data(self, value):
        if abs(value) < 0.2: value = 0
        return value

    def cancel_nav(self):
        now_time = time.time()
        if now_time - self.cancel_time > 1:
            Joy_ctrl = Bool()
            new_Joy_active = not self.Joy_active
            if new_Joy_active != self.Joy_active:
                if new_Joy_active:
                    self.get_logger().info("Play:true")
                else:
                    self.get_logger().info("Play:false")
            self.Joy_active = new_Joy_active
            Joy_ctrl.data = self.Joy_active
            for i in range(3):
                self.pub_JoyState.publish(Joy_ctrl)
                self.pub_cmdVel.publish(Twist())
            self.cancel_time = now_time
```

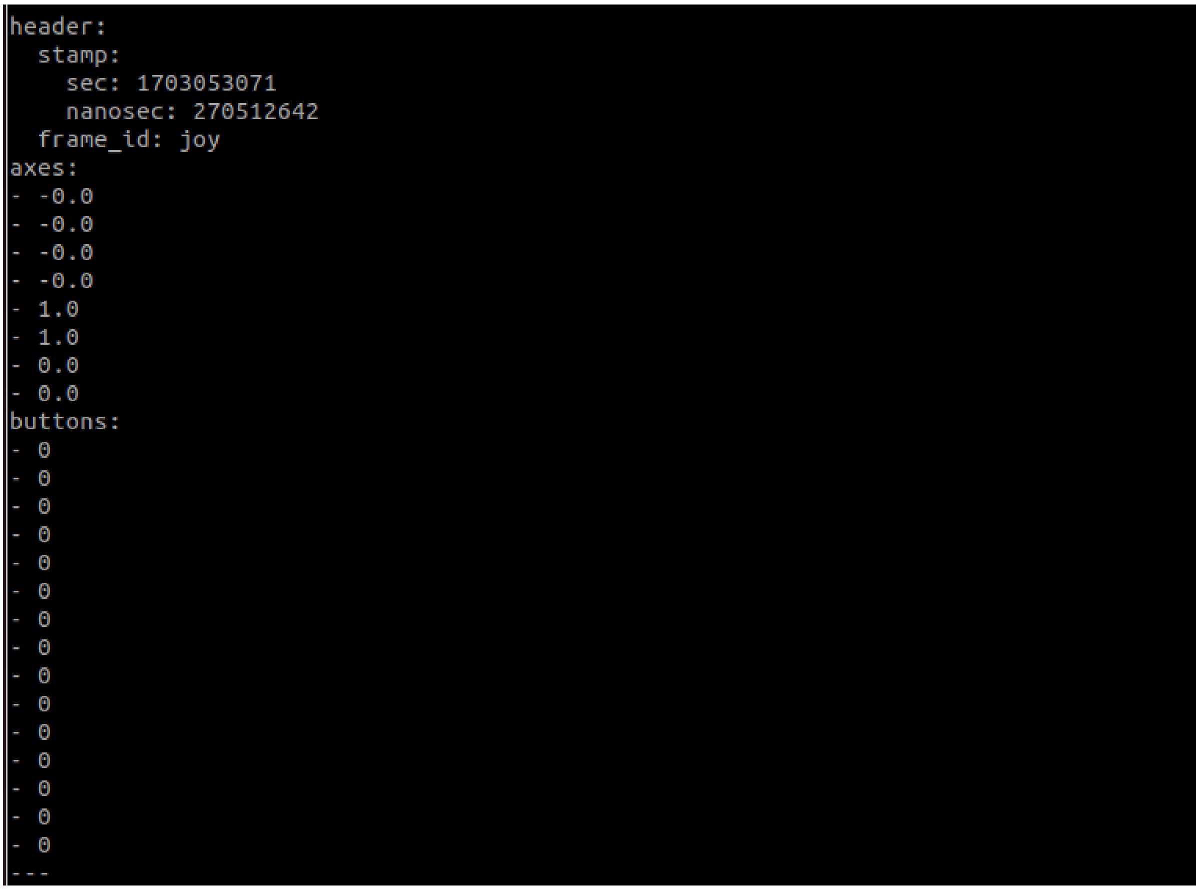
6. Variables corresponding to remote control key values

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 left	axes[2]=1
Right joystick right	axes[2]=-1
Button X pressed	button[3]=1
Button B pressed	button[1]=1
Button Y pressed	button[4]=1
R1 button 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 source code above, it is easy to understand. When these values change, it means that the remote control is pressed and the corresponding program can be executed. To view other button presses, you can subscribe to the /joy topic and enter in the terminal.

```
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
---
```