Multi-car handle control

Note: The ROS_DOMAIN_ID of the Raspberry Pi needs to be consistent with that of the microROS control board. You can check **[MicroROS Control Board Parameter Configuration]** to set the microROS control board ROS_DOMAIN_ID. Check the tutorial **[Connect MicroROS Agent]** to determine whether the IDs are consistent.

1、Program function description

After the program is started, the two cars can be controlled to move synchronously through the handle. The controller program here is based on the [ROS Robot USB Wireless Controller] sold by Yabo Intelligent Technology as an example. Other controller programs may not be suitable, and the program needs to be modified according to the actual remote control key values.

2、Multi-machine function basic settings

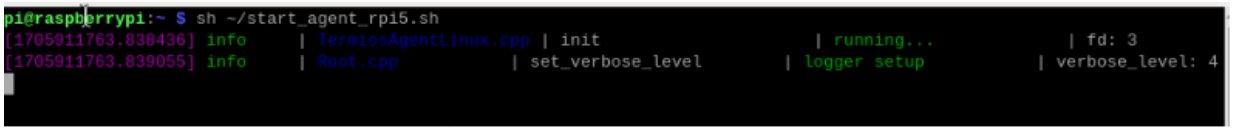
Taking two cars as an example, it is recommended to use two cars with Raspberry Pi, change the config_robot.py file respectively, set robot.set_ros_namespace() to robot1 and robot2 respectively, and set the ROS_DOMAIN_ID and virtual name of the two cars**. The ROS_DOMAIN_ID of the machine needs to be set to the same.** Then open the terminal in the /home/pi directory and enter `sudo python3 config_robot.py` to run this program (you need to change it back and re-run this program to run other programs except multi-car).

```
if __name__ == '__main__':
    robot = MicroROS_Robot(port='/dev/ttyUSB0', debug=False)

    #robot.set_wifi_config("yahboom2", "yahboom890729")
    #robot.set_udp_config([192, 168, 2, 103], 8090)
    #robot.set_car_type(robot.CAR_TYPE_COMPUTER)
    robot.set_car_type(robot.CAR_TYPE_RPI5)
    robot.set_ros_domain_id(20)
    robot.set_ros_serial_baudrate(921600)
    robot.set_ros_namespace("tobot1")
    robot.set_pwm_servo_offset(1, 0)
    robot.set_pwm_servo_offset(2, 0)
    robot.set_motor_pid_parm(1, 0.2, 0.2)
    robot.set_imu_yaw_pid_parm(1, 0, 0.2)

    time.sleep(.1)
    robot.print_all_firmware_parm()
    print("Please reboot the device to take effect, if you change some device config.")

    try:
        while False:
            # robot.beep(100)
            time.sleep(1)
    except:
        pass
    time.sleep( 1)
```

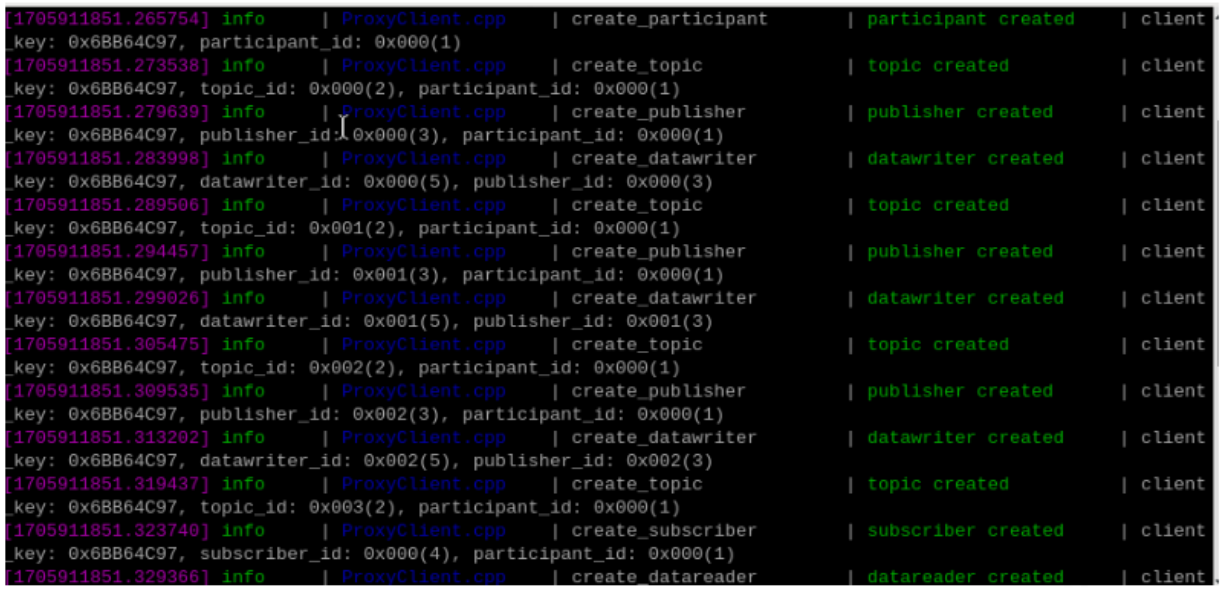
3、Start and connect to the agent

After the Raspberry Pi is successfully powered on, open the terminal and enter the following command to open the agent.

```
sh ~/start_agent_rpi5.sh
```



Press the reset button on the microROS control board and wait for the car to connect to the agent. The connection is successful as shown in the figure below.

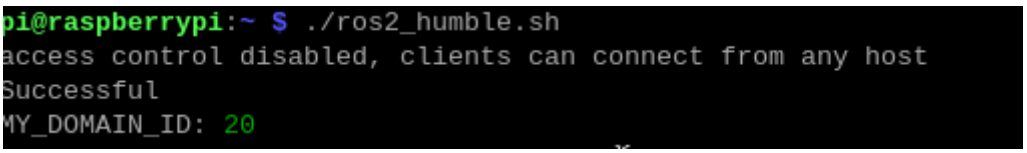


4、Enter the car docker

Open another terminal and enter the following command to enter docker.

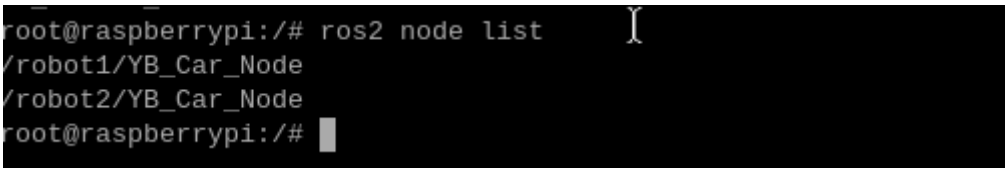
```
sh ros2_humble.sh
```

When the following interface appears, you have successfully entered docker. Now you can control the car through commands.



Check the currently started node. Choose one of the two Raspberry Pis and open the terminal for input.

```
ros2 node list
```



As shown in the picture above, the nodes of both cars have been started. To query the current topic information, enter in the terminal.

```
ros2 topic list
```

```
root@raspberrypi:/# ros2 topic list
/parameter_events
/robot1/beep
/robot1/cmd_vel
/robot1/imu
/robot1/odom_raw
/robot1/scan
/robot1/servo_s1
/robot1/servo_s2
/robot2/beep
/robot2/cmd_vel
/robot2/imu
/robot2/odom_raw
/robot2/scan
/robot2/servo_s1
/robot2/servo_s2
/rosout
root@raspberrypi:/#
```

5、Start the handle control program

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

```
root@raspberrypi:/# ll /dev/input*
total 0
drwxr-xr-x   4 root root              260 Feb  1 08:11 ./
drwxr-xr-x  12 root root             4220 Feb  1 08:16 ../
drwxr-xr-x   2 root root              120 Feb  1 08:11 by-id/
drwxr-xr-x   2 root root              180 Feb  1 08:11 by-path/
crw-rw----   1 root systemd-network 13,  64 Feb  1 07:17 event0
crw-rw----   1 root systemd-network 13,  65 Feb  1 07:17 event1
crw-rw----   1 root systemd-network 13,  66 Feb  1 07:17 event2
crw-rw----   1 root systemd-network 13,  67 Feb  1 08:11 event3
crw-rw----+  1 root systemd-network 13,  68 Feb  1 08:11 event4
crw-rw----   1 root systemd-network 13,  69 Feb  1 08:11 event5
crw-rw----   1 root systemd-network 13,  70 Feb  1 08:11 event6
crw-rw----+  1 root systemd-network 13,   0 Feb  1 08:11 js0
crw-rw----   1 root systemd-network 13,  63 Feb  1 07:17 mice
root@raspberrypi:/#
```

In the Raspberry Pi with the controller receiver connected, open the terminal and enter the following command.

```
ros2 run joy joy_node
ros2 run yahboomcar_multi multi_joy_ctrl
```

The remote control button description is as follows:

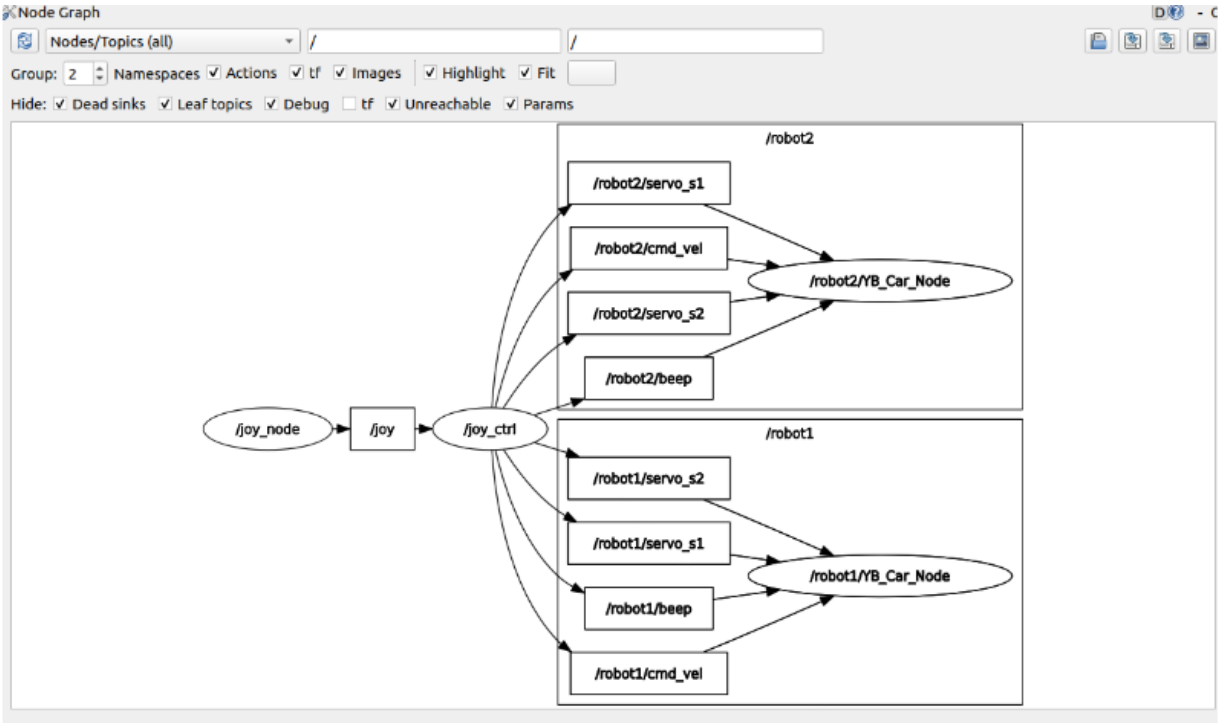
- Left rocker: valid in the front and rear directions, controls the car forward and backward, invalid in the left and right directions
- Right joystick: valid in left and right directions, controls the car to rotate left and right, invalid in forward and backward directions

- 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: The handle controls the speed switch. Only after pressing it can the speed of the car be controlled by remote control.
- 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.

6、View node communication diagram

Choose one of the two Raspberry Pis and enter the following command in the terminal.

```
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.

7、Source code analysis

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

```
/home/root/yahboomcar_ws/src/yahboomcar_multi/yahboomcar_multi
```

multi_yahboom_joy.py，

```
#创建话题发布者
```

```
#Create topic publisher
self.pub_goal = self.create_publisher(GoalID,"move_base/cancel",10)
self.pub_JoyState = self.create_publisher(Bool,"JoyState",  10)
#cmd_vel
self.pub_cmdVel_r1 = self.create_publisher(Twist,'/robot1/cmd_vel',10)
self.pub_cmdVel_r2 = self.create_publisher(Twist,'/robot2/cmd_vel',10)
#beep
self.pub_Buzzer_r1 = self.create_publisher(UInt16,"/robot1/beep",  1)
self.pub_Buzzer_r2 = self.create_publisher(UInt16,"/robot2/beep",  1)
#servo1
self.pub_Servo1_r1 = self.create_publisher(Int32,"/robot1/servo_s1" , 10)
self.pub_Servo1_r2 = self.create_publisher(Int32,"robot2/servo_s1" , 10)
#servo2
self.pub_Servo2_r1 = self.create_publisher(Int32,"/robot1/servo_s2" , 10)
self.pub_Servo2_r2= self.create_publisher(Int32,"/robot2/servo_s2" , 10)
#Create topic subscribers and subscribe to /joy node information
self.sub_Joy = self.create_subscription(Joy,'joy', self.buttonCallback,10)
#Callback
def buttonCallback(self,joy_data):
    if not isinstance(joy_data, Joy): return
    self.user_jetson(joy_data)
#Process the remote control key value. For detailed code, please refer to the
user_jetson function.
#Publishing speed topic
self.pub_cmdVel_r1.publish(twist)
self.pub_cmdVel_r2.publish(twist)
#Publishing speed topic
self.pub_Buzzer_r1.publish(b)
self.pub_Buzzer_r2.publish(b)
#Publish gimbal servo data (Raspberry Pi 5 version)
#PTZ 1
self.pub_Servo1_r1.publish(servo1_angle)
self.pub_Servo1_r2.publish(servo1_angle)
#PTZ 2
self.pub_Servo2_r1.publish(servo2_angle)
self.pub_Servo2_r2.publish(servo2_angle)
```