Multi-machine keyboard control

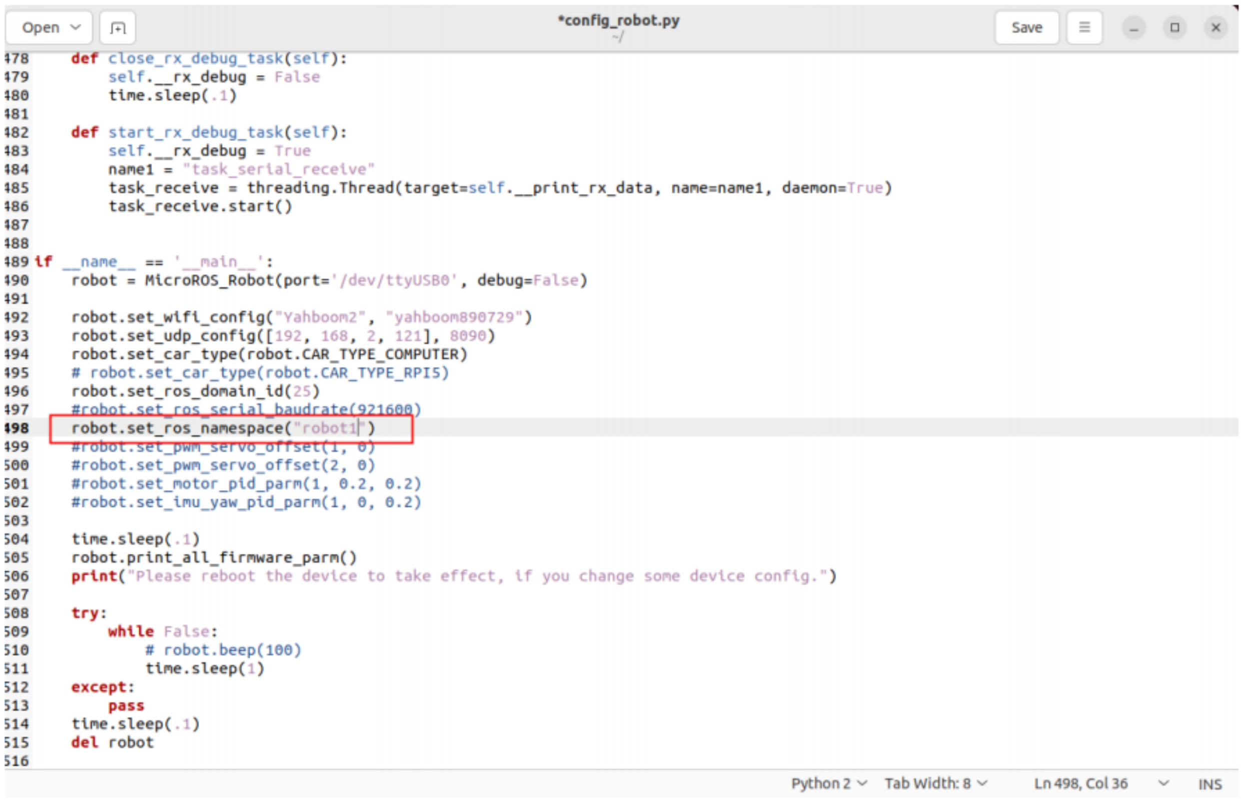
Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check **[Must read before use]** to set the **IP** and **ROS_DOMAIN_ID** on the board.

1、Program function description

After the program is started, the two cars can be controlled to move synchronously through the keyboard.

2、Multi-machine function basic settings

Taking two cars as an example, it is recommended to use two computers with matching virtual machines, change the config_robot.py files, and set robot.set_ros_namespace() to robot1 and robot2 respectively. And **the ROS_DOMAIN_ID of the two cars and the ROS_DOMAIN_ID of the virtual machine need to be set to the same**. Then open the terminal in the /home/yahboom 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).

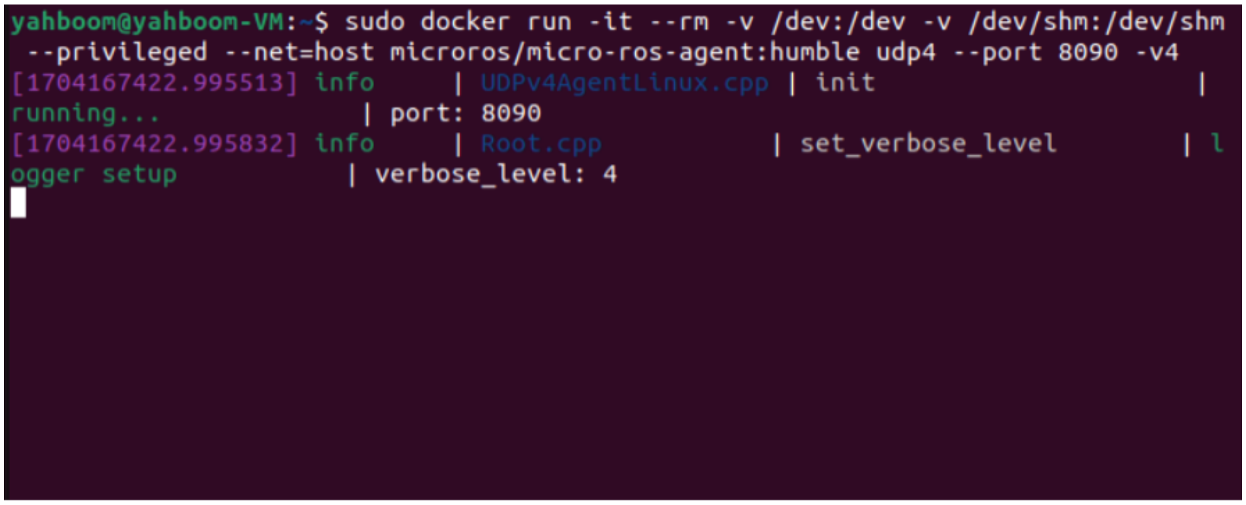


```python
    def close_rx_debug_task(self):
        self.__rx_debug = False
        time.sleep(.1)

    def start_rx_debug_task(self):
        self.__rx_debug = True
        name1 = "task_serial_receive"
        task_receive = threading.Thread(target=self.__print_rx_data, name=name1, daemon=True)
        task_receive.start()


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

    robot.set_wifi_config("Yahboom2", "yahboom890729")
    robot.set_udp_config([192, 168, 2, 121], 8090)
    robot.set_car_type(robot.CAR_TYPE_COMPUTER)
    # robot.set_car_type(robot.CAR_TYPE_RPI5)
    robot.set_ros_domain_id(25)
    #robot.set_ros_serial_baudrate(921600)
    robot.set_ros_namespace("robot1")
    #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)
    del robot
```

3、Start and connect to the agent

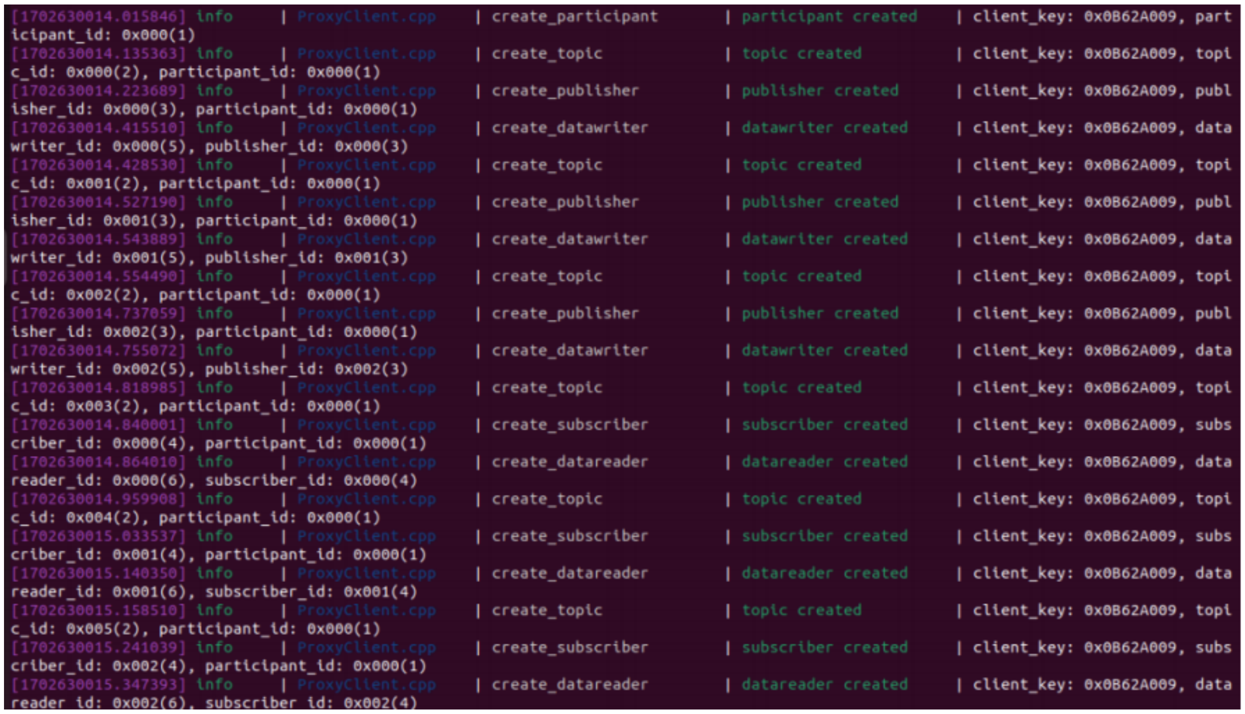
Taking the supporting virtual machine as an example, under the two virtual machines, enter the following commands to start the agents of the respective cars:

```
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
```



```
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 8090 -v4
[1704167422.995513] info     | UDPv4AgentLinux.cpp | init                     |
running...               | port: 8090
[1704167422.995832] info     | Root.cpp            | set_verbose_level        | l
ogger setup              | verbose_level: 4
```

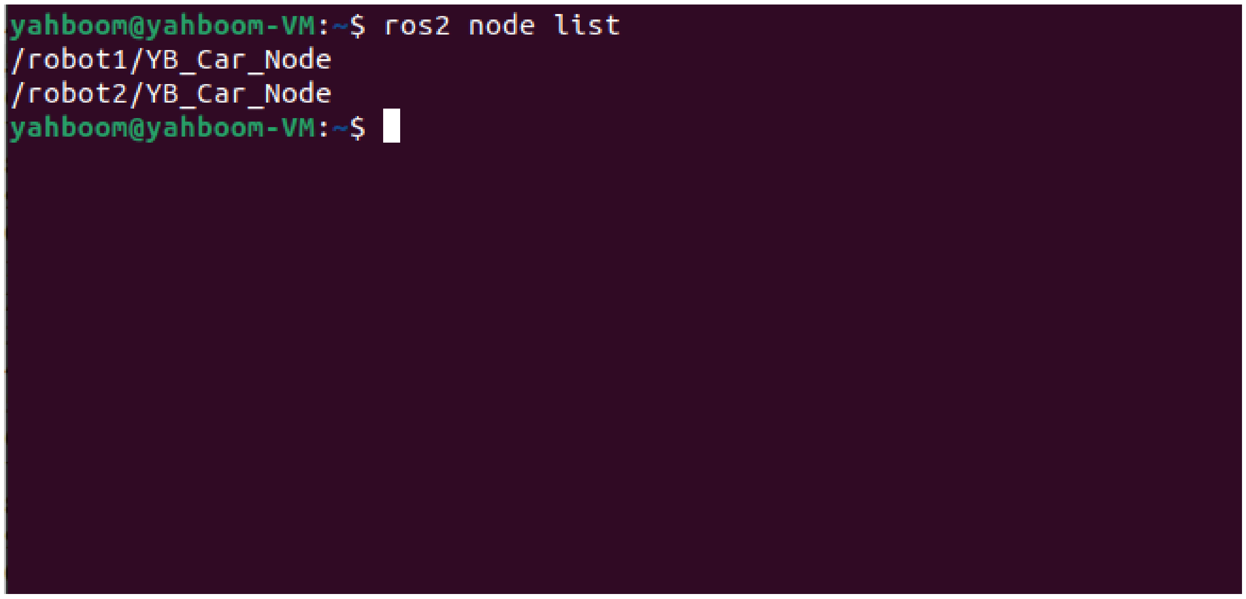
Then, turn on the switches of the two cars and wait for the two cars to connect to their respective agents. The connection is successful and the terminal display is as shown in the figure below.



```
[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)
```

Check the currently started node, choose one of the two virtual machines, open the terminal and enter,

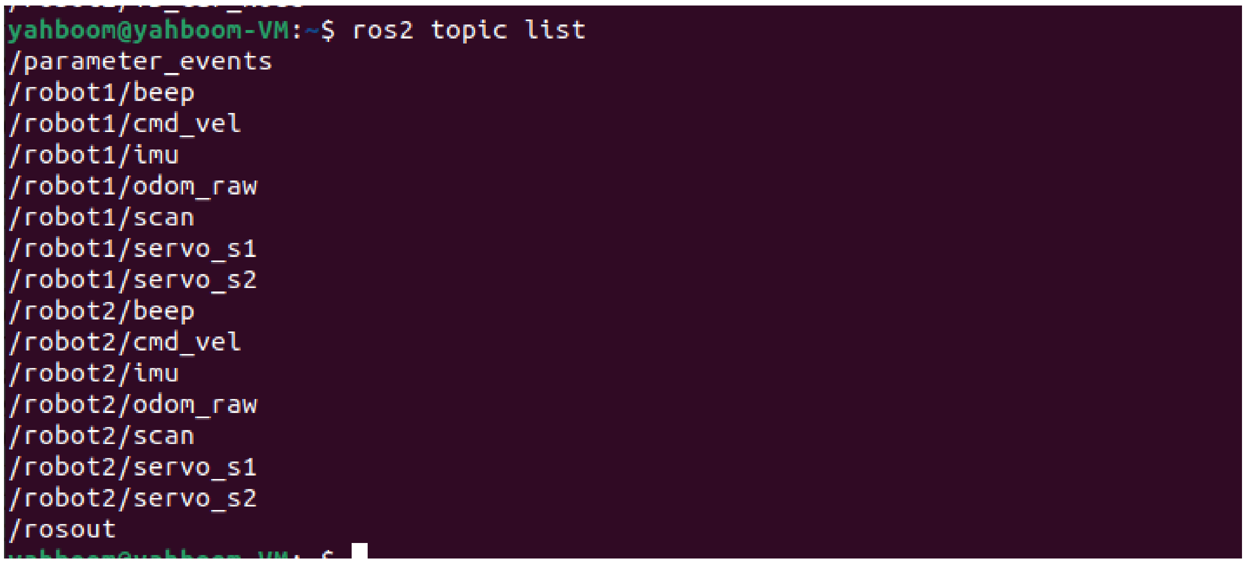
```
ros2 node list
```



```
yahboom@yahboom-VM:~$ ros2 node list
/robot1/YB_Car_Node
/robot2/YB_Car_Node
yahboom@yahboom-VM:~$ 
```

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

```
ros2 topic list
```

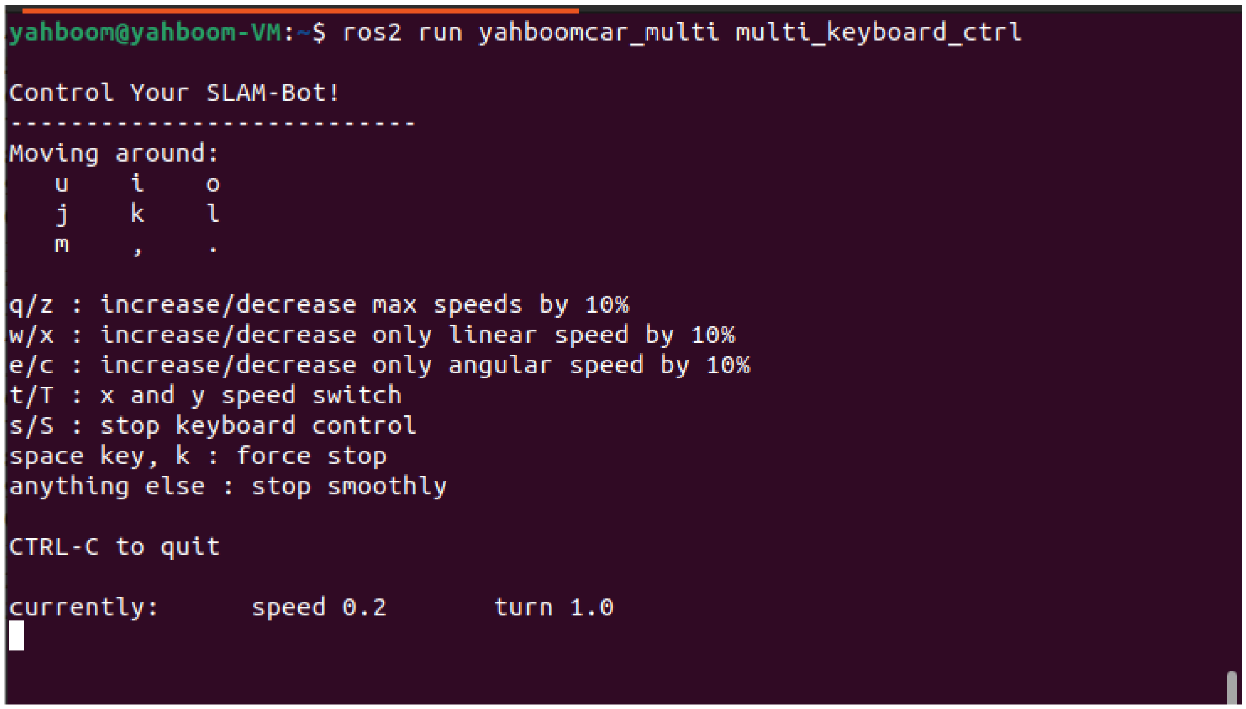


```
yahboom@yahboom-VM:~$ ros2 topic list
/parameter_events
/robot1/beep
/robot1/cmd_vel
/robot1/imu
/robot1/odom_raw
/robot1/scan
/robot1/servo_s1
/robot1/servo_s2
/robot2/beep
/robot2/cmd_vel
/robot2/imu
/robot2/odom_raw
/robot2/scan
/robot2/servo_s1
/robot2/servo_s2
/rosout
```

4、Start the keyboard control program

Among the two virtual machines, select any one, open the terminal and enter the following command in the terminal.

```
ros2 run yahboomcar_multi multi_keyboard_ctrl
```



```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_multi multi_keyboard_ctrl

Control Your SLAM-Bot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
t/T : x and y speed switch
s/S : stop keyboard control
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1.0
```

Keyboard key descriptions are as follows

direction control table

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

That is, press the **[i]** key on the keyboard to move forward, press **[,]** to move backward, press **[l]** to rotate to the right, press **[j]** to rotate to the left, and so on.

speed control table：

keyboard keys	**speed change**	**keyboard keys**	**speed change**
【q】	Linear speed and angular speed are both increased by 10%	【z】	Linear speed and angular speed are both reduced by 10%
【w】	Line speed increased by 10%	【x】	Line speed reduced by 10%
【e】	Angular speed increased by 10%	【c】	Angular speed reduced by 10%
【t】	Linear speed X-axis/Y-axis direction switching	【s】	Stop keyboard control

Note: Since the car has a four-wheel drive structure with ordinary tires and cannot move sideways, the [t] button has no meaning. Before each use of keyboard control, you need to click on the terminal that starts the program, otherwise the key event cannot be detected.

5、Code analysis

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

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_multi/yahboomcar_multi
```

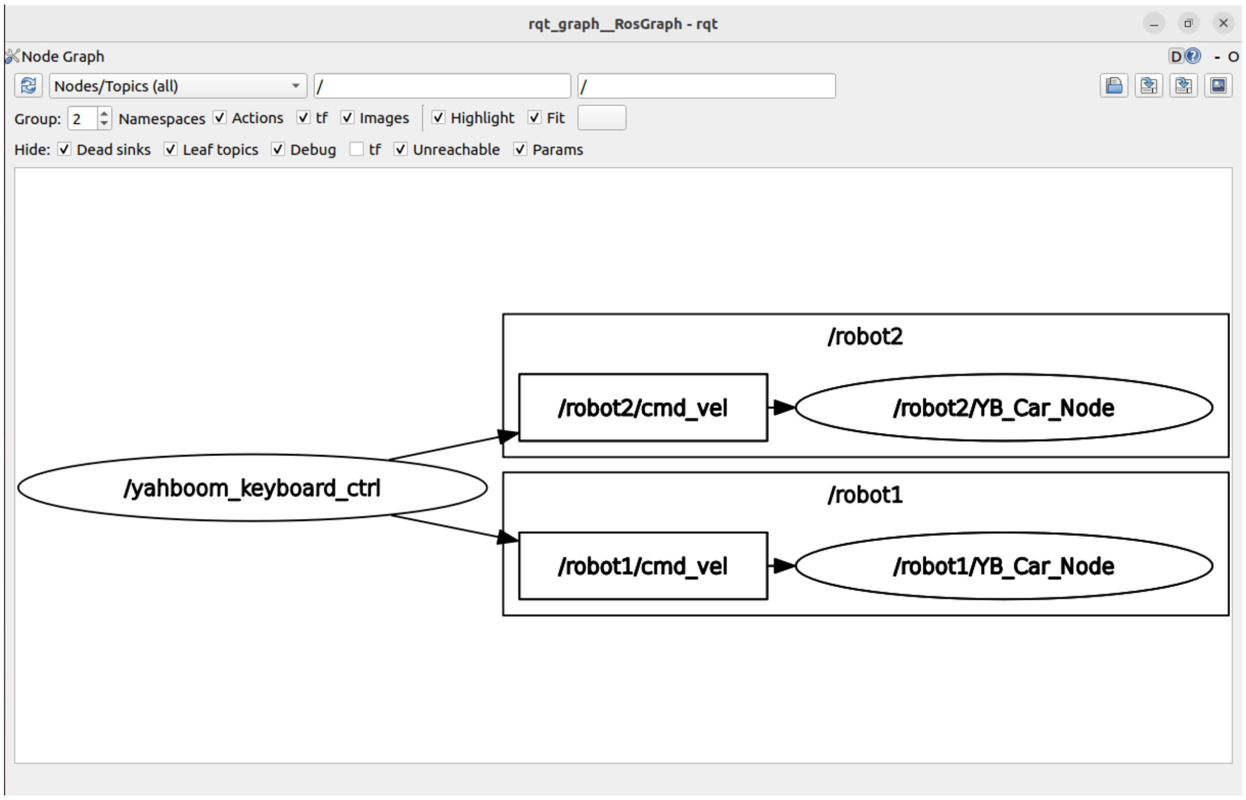
multi_yahboom_keyboard.py

```
#Create two velocity publishers
self.pub_robot1 = self.create_publisher(Twist,'/robot1/cmd_vel',1000)
self.pub_robot2 = self.create_publisher(Twist,'/robot2/cmd_vel',1000)
#Get key event
def getKey(self):
    tty.setraw(sys.stdin.fileno())
    rlist, _, _ = select.select([sys.stdin], [], [], 0.1)
    if rlist: key = sys.stdin.read(1)
else: key = ''
termios.tcsetattr(sys.stdin, termios.TCSADRAIN, self.settings)
return key
#Analyze key events
while (1):
    key = yahboom_keyboard.getKey()
    if key=="t" or key == "T": xspeed_switch = not xspeed_switch
    elif key == "s" or key == "S":
    print ("stop keyboard control: {}".format(not stop))
    stop = not stop
    if key in moveBindings.keys():
        x = moveBindings[key][0]
        th = moveBindings[key][1]
        count = 0
    elif key in speedBindings.keys():
        speed = speed * speedBindings[key][0]
        turn = turn * speedBindings[key][1]
        count = 0
        if speed > yahboom_keyboard.linenar_speed_limit:
            speed = yahboom_keyboard.linenar_speed_limit
            print("Linear speed limit reached!")
            if turn > yahboom_keyboard.angular_speed_limit:
                turn = yahboom_keyboard.angular_speed_limit
                print("Angular speed limit reached!")
                print(yahboom_keyboard.vels(speed, turn))
                if (status == 14): print(msg)
                status = (status + 1) % 15
            elif key == ' ': (x, th) = (0, 0)
        else:
            count = count + 1
            if count > 4: (x, th) = (0, 0)
            if (key == '\x03'): break
#Publish car speed
yahboom_keyboard.pub_robot1.publish(robot1_twist)
yahboom_keyboard.pub_robot2.publish(robot2_twist)
```

6、View node communication diagram

Select one of the two virtual machines, open the terminal and enter the following command:

```
ros2 run rqt_graph rqt_graph
```



rqt_graph__RosGraph - rqt

Node Graph

Nodes/Topics (all) / /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug ☐ tf ☑ Unreachable ☑ Params

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