Multi-robot synchronous control

Code path:

（1）VM(virtual machine) side：

~/transbot_ws/src/transbot_ctrl/script/transbot_joy.py

（2）Transbot robot side：

~/transbot_ws/src/transbot_bringup/script/transbot_driver_sync_robot1.py

~/transbot_ws/src/transbot_bringup/script/transbot_driver_sync_robot2.py

Function description: After the function is turned on, we can use the mode button of the handle to select the robot to control it individually, or we can choose multi-robot synchronous control.

Feature package path

VM(virtual machine) side： ~/transbot_ws/src/transbot_ctrl

Transbot robot side： ~/transbot_ws/src/transbot_bringup

Function realization conditions

It is necessary to configure the network of multiple Transbot robot, make all Transbot robot and the virtual machine are in the same local area network, and the virtual machine must be used as the host (Master).

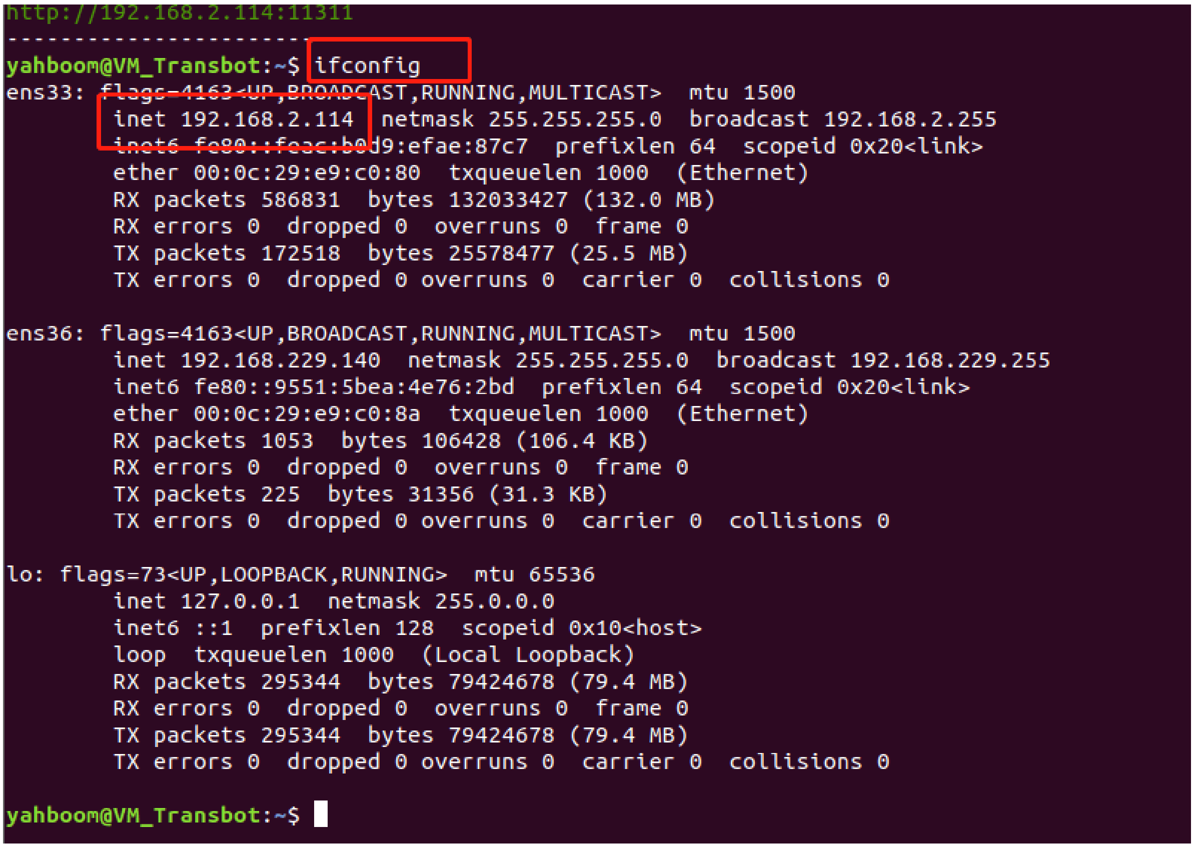
1. Multi-machine network configuration

1.1 Configure VM(virtual machine)-side networking

1)Input ifconfig in the terminal to view your own network IP.

2)Using gedit or vim tool to open and modify the ./.bashrc file on the virtual machine side.

3)Set ROS_MASTER_URI to the IP of your own network, save and exit.

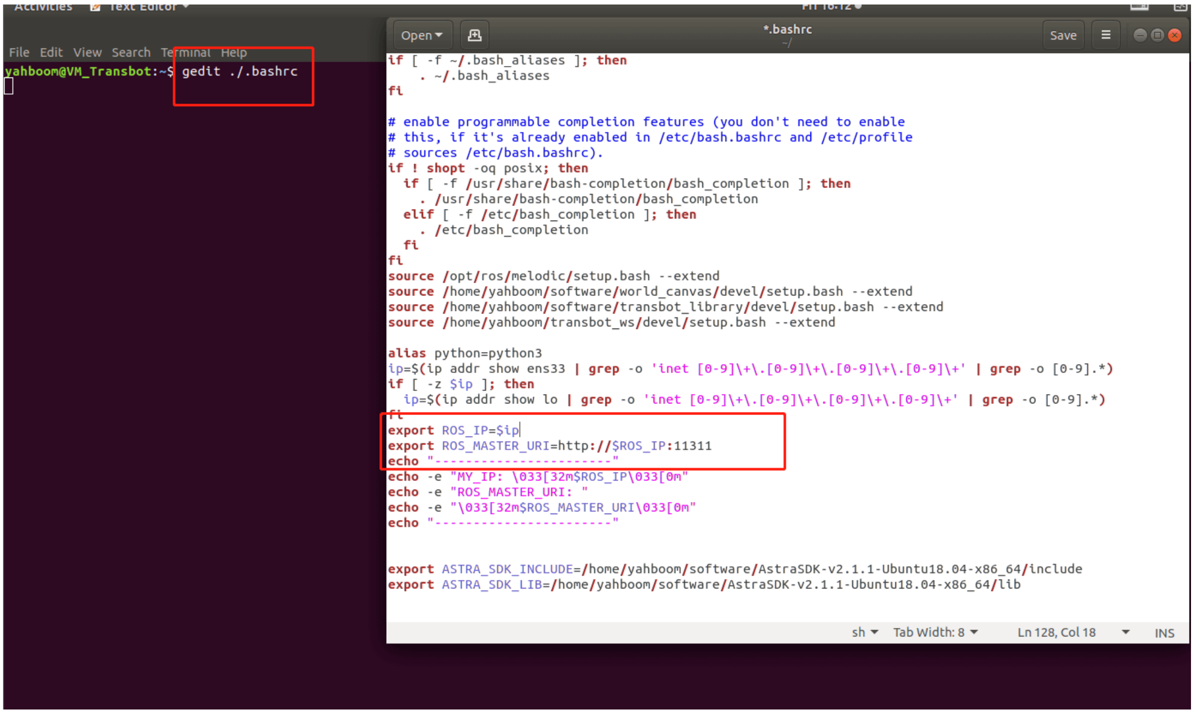


```
http://192.168.2.114:11311
-----------------------
yahboom@VM_Transbot:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.114  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::feac:b0d9:efae:87c7  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e9:c0:80  txqueuelen 1000  (Ethernet)
        RX packets 586831  bytes 132033427 (132.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 172518  bytes 25578477 (25.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens36: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.229.140  netmask 255.255.255.0  broadcast 192.168.229.255
        inet6 fe80::9551:5bea:4e76:2bd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e9:c0:8a  txqueuelen 1000  (Ethernet)
        RX packets 1053  bytes 106428 (106.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 225  bytes 31356 (31.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 295344  bytes 79424678 (79.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 295344  bytes 79424678 (79.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yahboom@VM_Transbot:~$ 
```



```
yahboom@VM_Transbot:~$ gedit ./.bashrc
```

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
source /opt/ros/melodic/setup.bash --extend
source /home/yahboom/software/world_canvas/devel/setup.bash --extend
source /home/yahboom/software/transbot_library/devel/setup.bash --extend
source /home/yahboom/transbot_ws/devel/setup.bash --extend

alias python=python3
ip=$(ip addr show ens33 | grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+' | grep -o [0-9].*)
if [ -z $ip ]; then
  ip=$(ip addr show lo | grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+' | grep -o [0-9].*)
fi
export ROS_IP=$ip
export ROS_MASTER_URI=http://$ROS_IP:11311
echo "----------------------"
echo -e "MY_IP: \033[32m$ROS_IP\033[0m"
echo -e "ROS_MASTER_URI: "
echo -e "\033[32m$ROS_MASTER_URI\033[0m"
echo "----------------------"


export ASTRA_SDK_INCLUDE=/home/yahboom/software/AstraSDK-v2.1.1-Ubuntu18.04-x86_64/include
export ASTRA_SDK_LIB=/home/yahboom/software/AstraSDK-v2.1.1-Ubuntu18.04-x86_64/lib
```

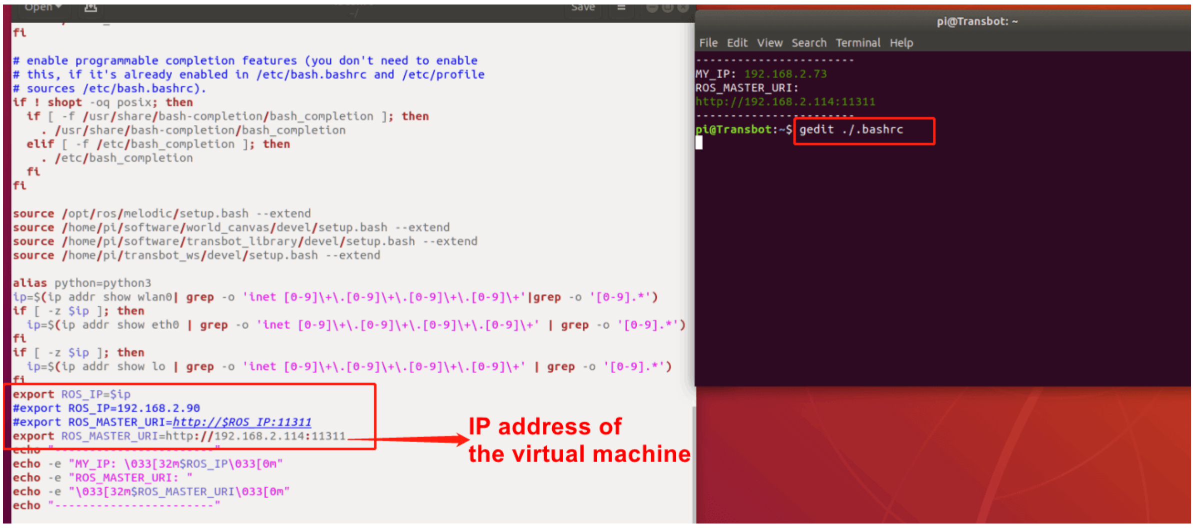
4)Enter source ./.bashrc to refresh the environment variables, which completes the configuration on the virtual machine side.

1.2 Configure Transbot robot-side networking

1)Enter ifconfig in the terminal to view your own network IP.

2)Using gedit or vim tool to open and modify the ./.bashrc file on the virtual machine side.

3)Set ROS_MASTER_URI to the IP of VM(virtual machine)-side network, save and exit.



```
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

source /opt/ros/melodic/setup.bash --extend
source /home/pi/software/world_canvas/devel/setup.bash --extend
source /home/pi/software/transbot_library/devel/setup.bash --extend
source /home/pi/transbot_ws/devel/setup.bash --extend

alias python=python3
ip=$(ip addr show wlan0| grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+'|grep -o '[0-9].*')
if [ -z $ip ]; then
  ip=$(ip addr show eth0 | grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+' | grep -o '[0-9].*')
fi
if [ -z $ip ]; then
  ip=$(ip addr show lo | grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+' | grep -o '[0-9].*')
fi
export ROS_IP=$ip
#export ROS_IP=192.168.2.90
#export ROS_MASTER_URI=http://$ROS_IP:11311
export ROS_MASTER_URI=http://192.168.2.114:11311
echo "----------------------"
echo -e "MY_IP: \033[32m$ROS_IP\033[0m"
echo -e "ROS_MASTER_URI: "
echo -e "\033[32m$ROS_MASTER_URI\033[0m"
echo "----------------------"
```

```
pi@Transbot: ~
----------------------
MY_IP: 192.168.2.73
ROS_MASTER_URI:
http://192.168.2.114:11311
----------------------
pi@Transbot:~$ gedit ./.bashrc
```

4)Enter source ./.bashrc to refresh the environment variables, which completes the configuration on the Transbot robot side.

1.3 Check if the configuration is successful

Turn on roscore on the virtual machine side, and enter rosrun turtlesim turtlesim_node in the terminal of the Transbot car.

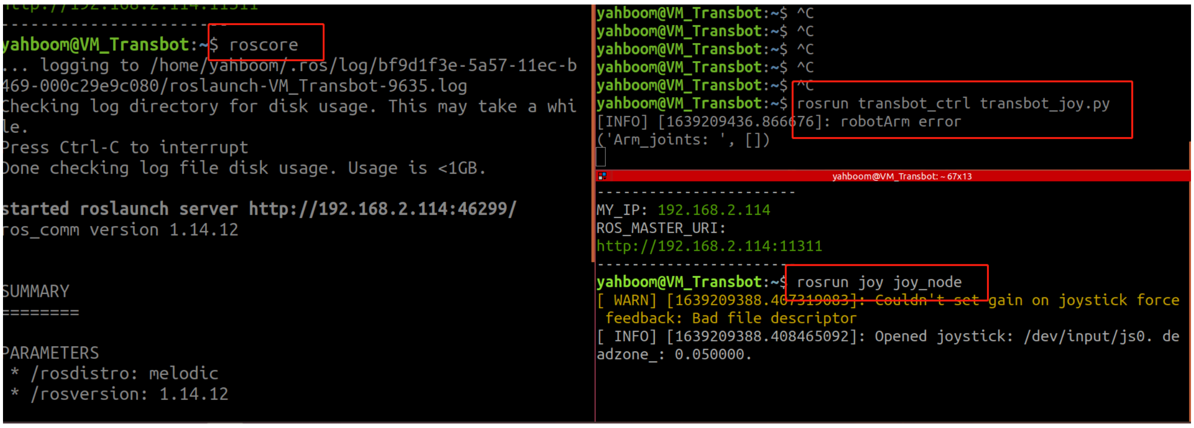
If a small turtle appears, which means is that the configuration is successful and you can proceed to the next step.

Note: The Master of each Transbot car must be set to a virtual machine before multi-machine communication can be performed.

2.Start up function

(1) Input following command in virtual machine side.

```
    roscore     #Start roscore
    rosrun transbot_ctrl transbot_joy.py    #Start transbot remote control node
    rosrun joy joy_node     #Start the remote control data transfer node
```



```
----------------------
yahboom@VM_Transbot:~$ roscore
... logging to /home/yahboom/.ros/log/bf9d1f3e-5a57-11ec-b
469-000c29e9c080/roslaunch-VM_Transbot-9635.log
Checking log directory for disk usage. This may take a whi
le.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.2.114:46299/
ros_comm version 1.14.12


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.12
```

```
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ rosrun transbot_ctrl transbot_joy.py
[INFO] [1639209436.866676]: robotArm error
('Arm_joints: ', [])
```

```
yahboom@VM_Transbot: ~ 67x13
----------------------
MY_IP: 192.168.2.114
ROS_MASTER_URI:
http://192.168.2.114:11311
----------------------
yahboom@VM_Transbot:~$ rosrun joy joy_node
[ WARN] [1639209388.407319083]: Couldn't set gain on joystick force
 feedback: Bad file descriptor
[ INFO] [1639209388.408465092]: Opened joystick: /dev/input/js0. de
adzone_: 0.050000.
```

Note: Before starting, the receiver of the controller should be inserted into the virtual machine, and a dialog box will pop up when inserting.

And you need to choose whether the virtual machine uses the device.

(2) Input following command in No.1 Transbot robot side.

```
    ROS_NAMESPACE=robot1 rosrun transbot_bringup transbot_driver_sync_robot1.py 
#No.1 Transbot
```

(3) Input following command in No.2 Transbot robot side.

```
    ROS_NAMESPACE=robot2 rosrun transbot_bringup transbot_driver_sync_robot2.py 
#No.2 Transbot
```

If you do not press the mode button on handle, the two Transbot robots can be controlled synchronously at this time;

when you press the mode button on handle for the first time, only No.1 Transbot can be controlled;

when you press the mode button on handle for the second time, only No.2 Transbot can be controlled;

when you press the mode button on handle for the third time, two Transbot robots can be controlled synchronously again.

3.Function realization principle

First, when we start the car drive control, we need to add ROS_NAMESPACE before the command line to add a namespace to the node. The purpose of this is to prevent node conflicts that cause the program to exit.

Then, in the remote control process of the handle, we detect whether the mode button is pressed. If it is pressed, we will publish a control_mode message to the robot drive control system.

```
    if joy_data.buttons[8] == 1:
        mode = mode +1 ;
        control_mode.data = mode
        self.pub_mode.publish(control_mode)
```

On the transbot car side, we subscribe to the control_mode message published by the remote control node, and then process the data. We judge the data by judging the received message:

(1) If it is 0 or a multiple of 3, it means that the mode button is not pressed, two robots can be controlled synchronously. The sending speed is velocity_com and angular_com;

(2) If it is pressed, it is judged whether it is an odd number or an even number.

If it is an odd number, No.1 Transbot will be controlled.

If it is an even number, No.2 Transbot will be controlled. the sending speed is velocity and angular;

```
def cmd_vel_callback(self, msg):
 print(control.count)
        if control.count == 0 or control.count%3 == 0:
              if not isinstance(msg, Twist): return
              velocity_com = msg.linear.x
              angular_com = msg.angular.z
              if velocity_com > self.linear_max:
                  velocity_com = self.linear_max
              elif velocity_com < -self.linear_max:
                  velocity_com = -self.linear_max
              elif -self.linear_min < velocity_com < 0:
                  velocity_com = -self.linear_min
              elif 0 < velocity_com < self.linear_min:
                  velocity_com = self.linear_min
              if angular_com > self.angular_max:
                  angular_com = self.angular_max
              elif angular_com < -self.angular_max:
                  angular_com = -self.angular_max
              elif -self.angular_min < angular_com < 0:
                  angular_com = -self.angular_min
              elif 0 < angular_com < self.angular_min:
                  angular_com = self.angular_min
              rospy.loginfo("cmd_vel: {}, cmd_ang: {}".format(velocity_com,
angular_com))
              self.bot.set_car_motion(0.8*velocity_com, 0.3*angular_com)
```

```
        if control.count != 0 and control.count%2!=0:
              if not isinstance(msg, Twist): return
              velocity = msg.linear.x
              angular = msg.angular.z
              if velocity > self.linear_max:
                    velocity = self.linear_max
              elif velocity < -self.linear_max:
                    velocity = -self.linear_max
                elif -self.linear_min < velocity < 0:
                    velocity = -self.linear_min
                elif 0 < velocity < self.linear_min:
                    velocity = self.linear_min
              if angular > self.angular_max:
                    angular = self.angular_max
                elif angular < -self.angular_max:
                    angular = -self.angular_max
                elif -self.angular_min < angular < 0:
                    angular = -self.angular_min
                elif 0 < angular < self.angular_min:
                    angular = self.angular_min
                rospy.loginfo("cmd_vel: {}, cmd_ang: {}".format(velocity,
angular))
                self.bot.set_car_motion(0.8*velocity, 0.3*angular)
```

```
        if control.count != 0 and control.count%2==0:
              if not isinstance(msg, Twist): return
              velocity = msg.linear.x
              angular = msg.angular.z
              if velocity > self.linear_max:
                    velocity = self.linear_max
              elif velocity < -self.linear_max:
                    velocity = -self.linear_max
                elif -self.linear_min < velocity < 0:
                    velocity = -self.linear_min
                elif 0 < velocity < self.linear_min:
                    velocity = self.linear_min
              if angular > self.angular_max:
                    angular = self.angular_max
                elif angular < -self.angular_max:
                    angular = -self.angular_max
                elif -self.angular_min < angular < 0:
                    angular = -self.angular_min
                elif 0 < angular < self.angular_min:
                    angular = self.angular_min
                rospy.loginfo("cmd_vel: {}, cmd_ang: {}".format(velocity, 
angular))
                self.bot.set_car_motion(0.8*velocity, 0.3*angular)
```