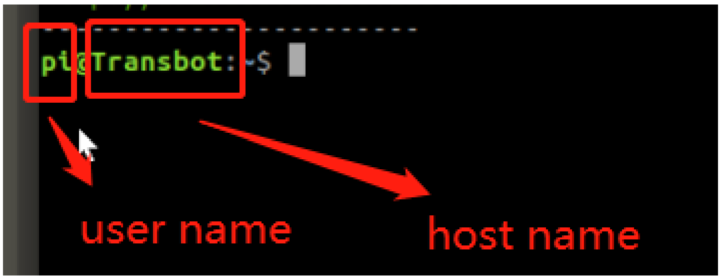
4、Multi-machine communication

Note: You must know the IP of the robot before remote login. You can view the current IP address of the robot through an external display or OLED



```
pi@Transbot:~$
```

4.1、Login remotely

4.1.1、SSH

Note: The system graphical interface cannot be displayed in this way.

- PuTTY Login

Input IP address, then click【open】；

For Yahoom image, user name is **pi** password is **yahboom**

- ubuntu system

1）Input following command in terminal

```
ssh pi@192.168.2.103
```

2）Then, input **yes**

3）Next, input **yahboom**

4.1.2、jupyter lab

Note: The system graphical interface cannot be displayed in this way.

Input following URL on browser, pree 【Enter】

Then, input password **yahboom**

Click【Log in】

```
http://192.168.2.103:8888
```

After login, you can see following interface.

You can enter these folder, and modify content in file, click 【Terminal】 to enter command terminal.

4.1.3、VNC

Note: The system graphical interface can be displayed in this way.

Login steps, as shown below.

4.2、ROS multi-machine communication

4.2.1、Universal version

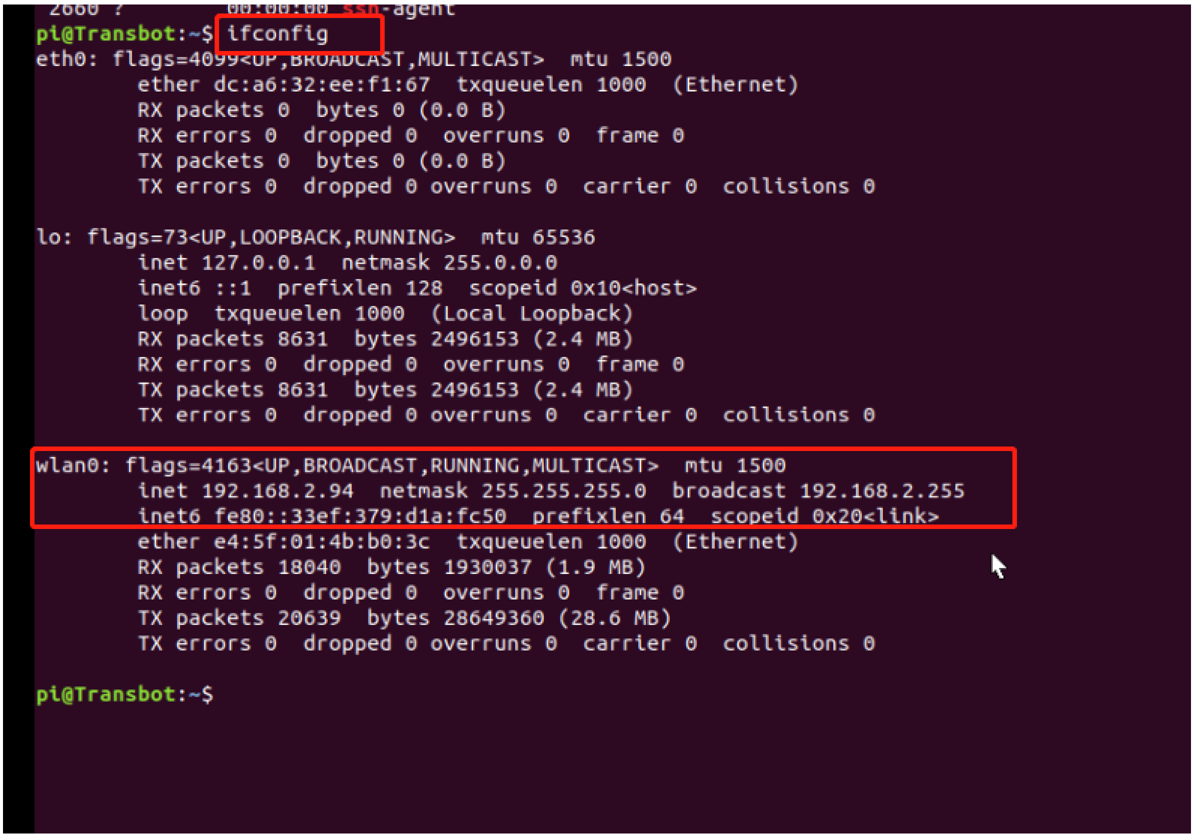
All ros masters are under the same network; if multiple devices have similar processes, choose one as the master, and the others are all slaves; the following two devices are taken as an example.

Install the ssh server on two devices; install the chrony package for synchronization:

```
sudo apt-get install chrony openssh-server
```

Input following command to view the IP information and host name of the two devices:

```
Command: ifconfig 和 hostname
```

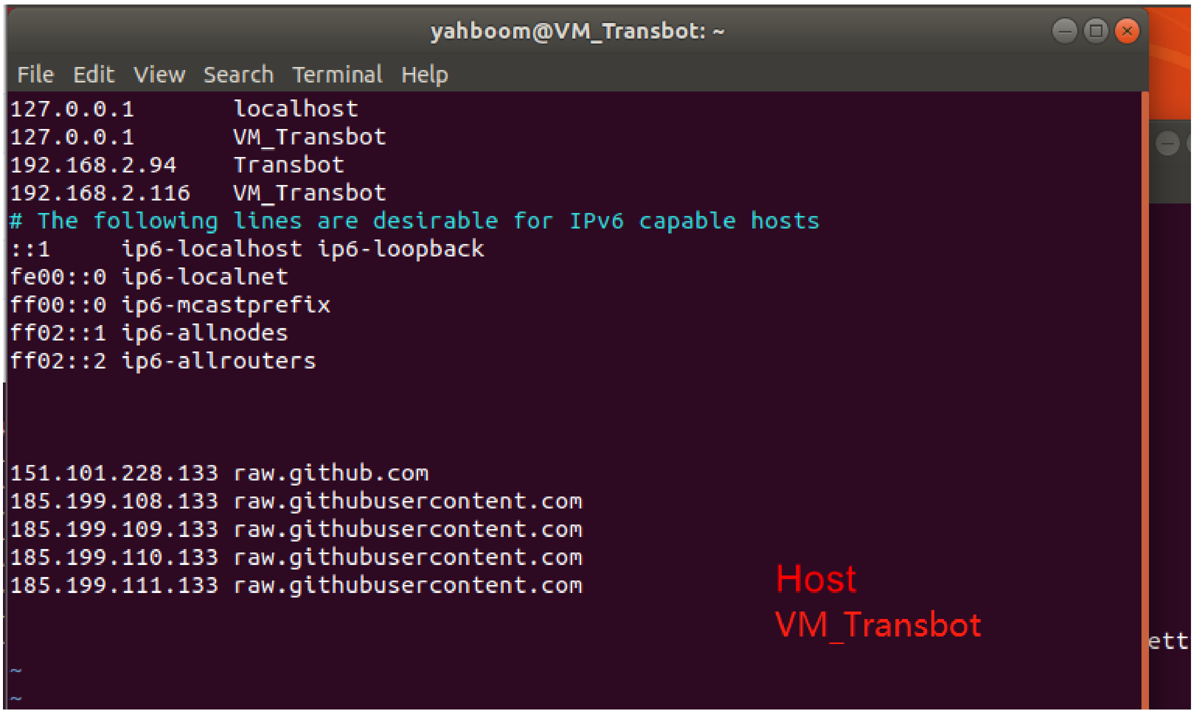


```
 2660 ?        00:00:00 ssh-agent
pi@Transbot:~$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether dc:a6:32:ee:f1:67  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        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 8631  bytes 2496153 (2.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8631  bytes 2496153 (2.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.94  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::33ef:379:d1a:fc50  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:4b:b0:3c  txqueuelen 1000  (Ethernet)
        RX packets 18040  bytes 1930037 (1.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20639  bytes 28649360 (28.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@Transbot:~$
```

修改/etc文件夹下的hosts文件：

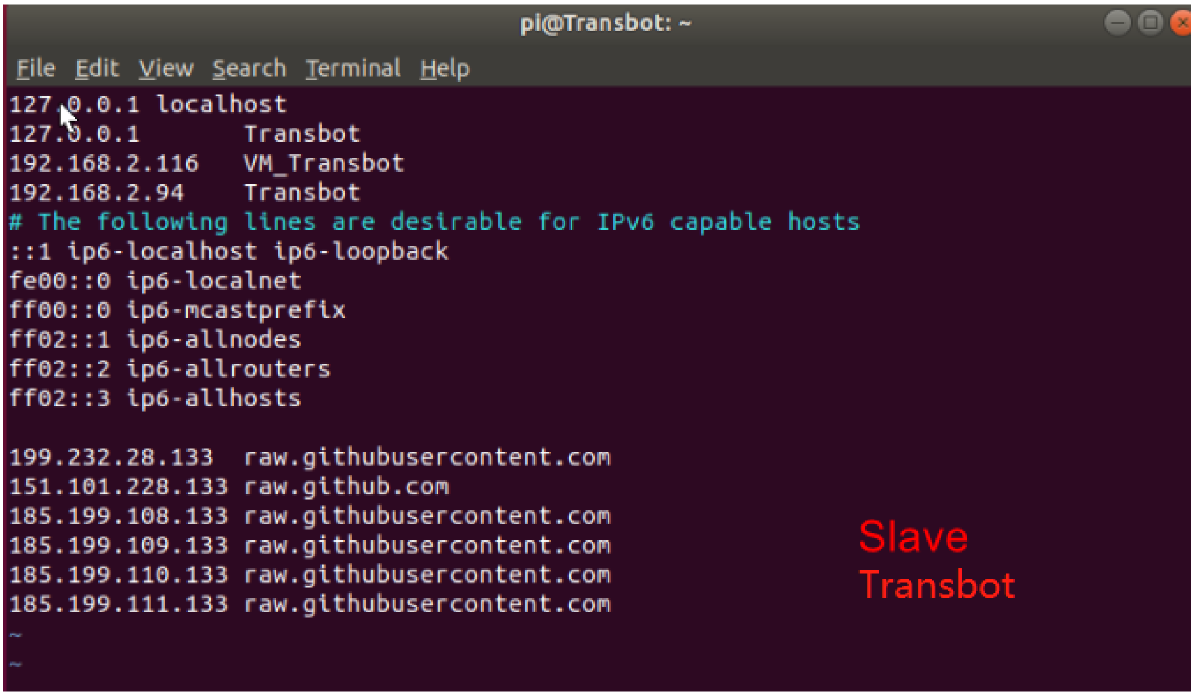
```
sudo chmod a+w /etc/hosts
sudo vim /etc/hosts
```

将两台设备的IP和主机名分别添加到两台设备的hosts文件下，绑定用户；前面的IP，后面的是名字，尽量对齐；



```
127.0.0.1       localhost
127.0.0.1       VM_Transbot
192.168.2.94    Transbot
192.168.2.116   VM_Transbot
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters



151.101.228.133 raw.github.com
185.199.108.133 raw.githubusercontent.com
185.199.109.133 raw.githubusercontent.com
185.199.110.133 raw.githubusercontent.com
185.199.111.133 raw.githubusercontent.com
```

Host
VM_Transbot



```
127.0.0.1 localhost
127.0.0.1       Transbot
192.168.2.116   VM_Transbot
192.168.2.94    Transbot
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

199.232.28.133  raw.githubusercontent.com
151.101.228.133 raw.github.com
185.199.108.133 raw.githubusercontent.com
185.199.109.133 raw.githubusercontent.com
185.199.110.133 raw.githubusercontent.com
185.199.111.133 raw.githubusercontent.com
```

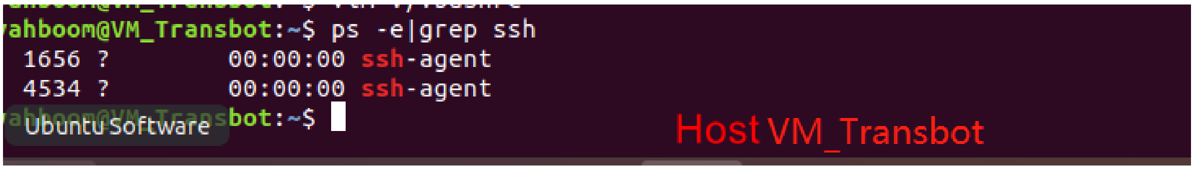
Slave
Transbot

After the modification, enter the following commands on both devices to restart the following network to realize the communication between the two devices: :

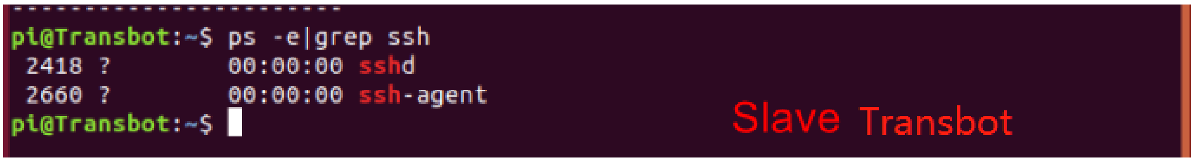
```
sudo /etc/init.d/networking restart
```

After installation, input following command confirm whether the server has been started:

```
ps -e|grep ssh
```



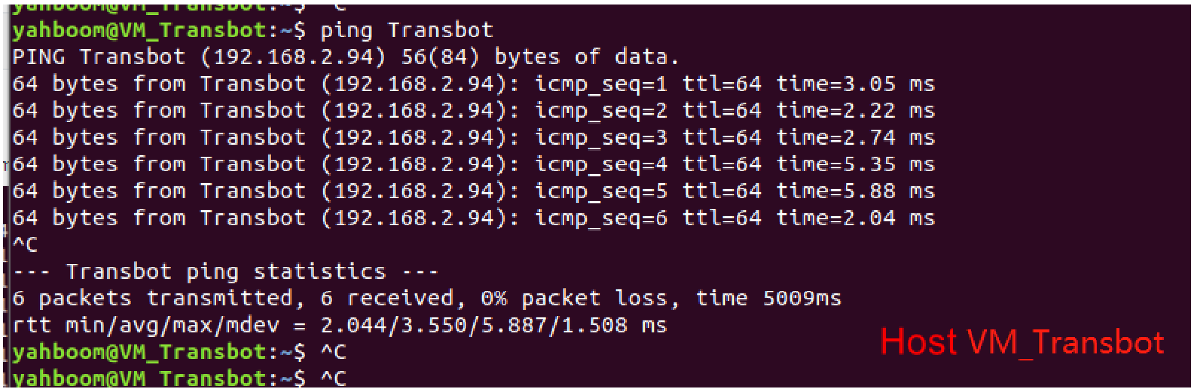
```
ahboom@VM_Transbot:~$ ps -e|grep ssh
 1656 ?        00:00:00 ssh-agent
 4534 ?        00:00:00 ssh-agent
```



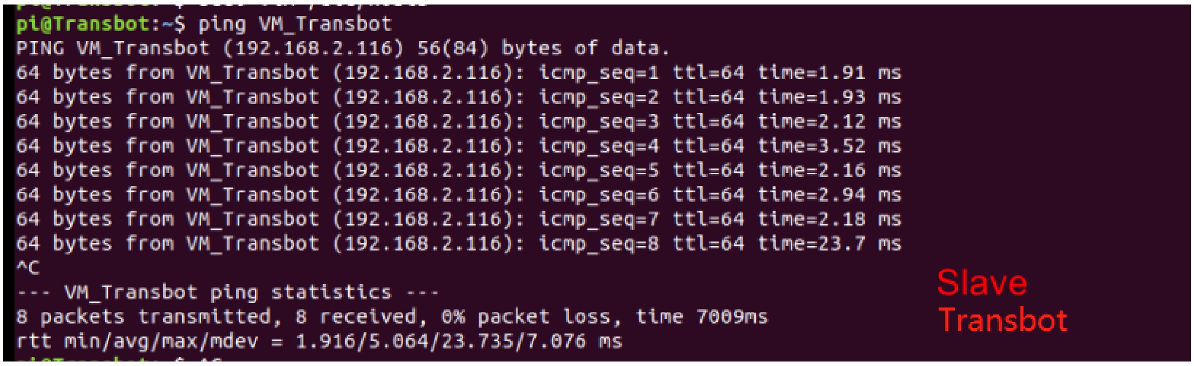
```
pi@Transbot:~$ ps -e|grep ssh
 2418 ?        00:00:00 sshd
 2660 ?        00:00:00 ssh-agent
pi@Transbot:~$ 
```

Input following command to check if the communication is normal:

```
ping hostname of the device
```



```
yahboom@VM_Transbot:~$ ping Transbot
PING Transbot (192.168.2.94) 56(84) bytes of data.
64 bytes from Transbot (192.168.2.94): icmp_seq=1 ttl=64 time=3.05 ms
64 bytes from Transbot (192.168.2.94): icmp_seq=2 ttl=64 time=2.22 ms
64 bytes from Transbot (192.168.2.94): icmp_seq=3 ttl=64 time=2.74 ms
64 bytes from Transbot (192.168.2.94): icmp_seq=4 ttl=64 time=5.35 ms
64 bytes from Transbot (192.168.2.94): icmp_seq=5 ttl=64 time=5.88 ms
64 bytes from Transbot (192.168.2.94): icmp_seq=6 ttl=64 time=2.04 ms
^C
--- Transbot ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 2.044/3.550/5.887/1.508 ms
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$ ^C
```



```
pi@Transbot:~$ ping VM_Transbot
PING VM_Transbot (192.168.2.116) 56(84) bytes of data.
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=1 ttl=64 time=1.91 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=2 ttl=64 time=1.93 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=3 ttl=64 time=2.12 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=4 ttl=64 time=3.52 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=5 ttl=64 time=2.16 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=6 ttl=64 time=2.94 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=7 ttl=64 time=2.18 ms
64 bytes from VM_Transbot (192.168.2.116): icmp_seq=8 ttl=64 time=23.7 ms
^C
--- VM_Transbot ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7009ms
rtt min/avg/max/mdev = 1.916/5.064/23.735/7.076 ms
```

- **Modify ~/.bashrc file**

```
sudo vim ~/.bashrc
```

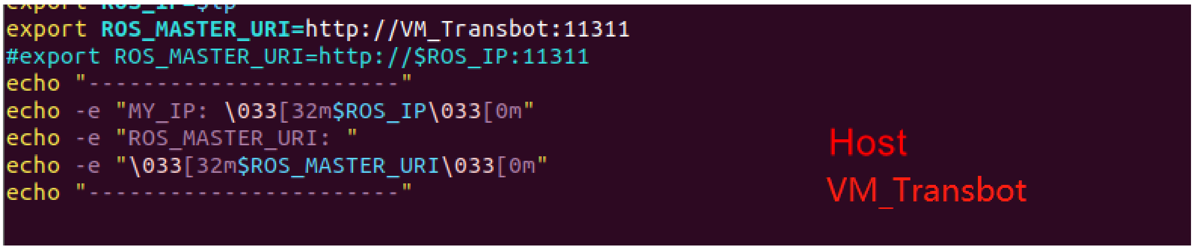
Add following content to ~/.bashrc file of master

```
export ROS_MASTER_URI=http://Host username:11311
```

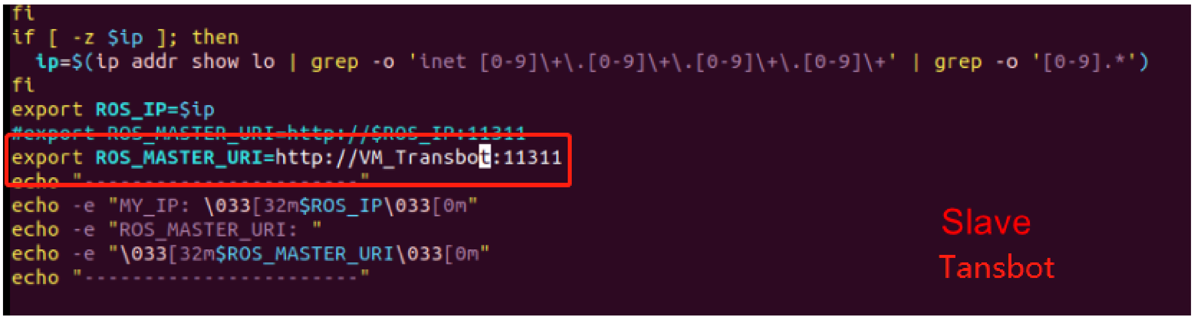
Add following content to ~/.bashrc file of slave

```
export ROS_MASTER_URI=http://Host username:11311
```

For example, jetson nano as a master



```
export ROS_IP=$ip
export ROS_MASTER_URI=http://VM_Transbot:11311
#export ROS_MASTER_URI=http://$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 "-----------------------"
```



```
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=$ip
#export ROS_MASTER_URI=http://$ROS_IP:11311
export ROS_MASTER_URI=http://VM_Transbot: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 "-----------------------"
```

After setting the IP, we need to refresh it, and then we can communicate.

```
source ~/.bashrc
```

- **Phenomenon show**

Note: we need to start up ROS Master on ROS Master

ubuntu(virtual machine VM_Transbot)

```
roscore
rosrun turtlesim turtle_teleop_key
```

Raspberry Pi (Transbot)

```
rosrun turtlesim turtlesim_node
```

4.2.2、Smiple version

If jetson nano is the host and the IP address is known, you only need to modify the .bashrc file of the slave.

```
sudo vim ~/.bashrc
```

Add following content file at the bottom.

```
export ROS_MASTER_URI=http://主机IP:11311
```

The effect is the same as [4.2.1]

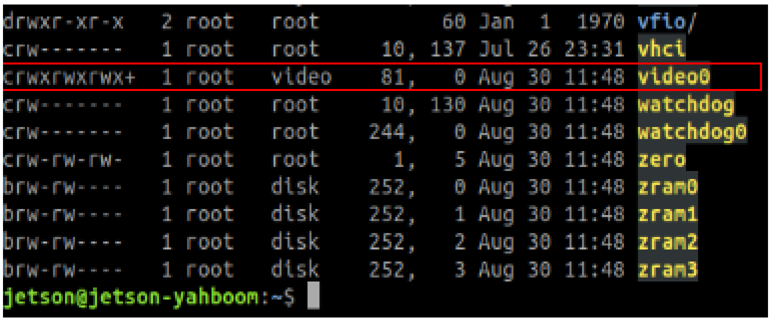
4.3、Webpage real-time monitoring

Environment setup

```
sudo apt-get install ros-melodic-async-web-server-cpp ros-melodic-web-video-server ros-melodic-usb-cam
```

Ensure that the USB camera be insert correctly, input following command to check if the USB device exists (it is video0)

```
ll /dev
```



```
drwxr-xr-x   2 root   root          60 Jan  1  1970 vfio/
crw-------   1 root   root     10, 137 Jul 26 23:31 vhci
crwxrwxrwx+  1 root   video    81,   0 Aug 30 11:48 video0
crw-------   1 root   root     10, 130 Aug 30 11:48 watchdog
crw-------   1 root   root    244,   0 Aug 30 11:48 watchdog0
crw-rw-rw-   1 root   root      1,   5 Aug 30 11:48 zero
brw-rw----   1 root   disk    252,   0 Aug 30 11:48 zram0
brw-rw----   1 root   disk    252,   1 Aug 30 11:48 zram1
brw-rw----   1 root   disk    252,   2 Aug 30 11:48 zram2
brw-rw----   1 root   disk    252,   3 Aug 30 11:48 zram3
jetson@jetson-yahboom:~$
```

If the system prompts that the execution authority is not enough, you need to input the following command to add the execution authority.

```
sudo chmod 777 /dev/video*
```

Modify usb_cam-test.launch file

```
sudo vim /opt/ros/melodic/share/usb_cam/launch/usb_cam-test.launch
```

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false"
output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

change

```
<launch>
    <arg name="open_view" default="false"/>
    <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen">
        <param name="video_device" value="/dev/video0"/>
        <param name="image_width" value="640"/>
        <param name="image_height" value="480"/>
        <param name="pixel_format" value="yuyv"/>
        <param name="camera_frame_id" value="usb_cam"/>
        <param name="io_method" value="mmap"/>
    </node>
    <!-- 启动web_video_server -->
    <node pkg="web_video_server" type="web_video_server" name="web_video_server"
output="screen"/>
    <!-- 是否启动image_view -->
    <group if="$(arg open_view)">
        <node name="image_view" pkg="image_view" type="image_view"
respawn="false" output="screen">
            <remap from="image" to="/usb_cam/image_raw"/>
            <param name="autosize" value="true"/>
        </node>
    </group>
</launch>
```

Open terminal, input following command to start it.

```
roslaunch usb_cam usb_cam-test.launch
```

View pictures

```
View on local web browser
http://localhost:8080/
If you want to view by other devices, you must ensure they at the same local area
network
http://192.168.2.103:8080/
(192.168.2.103 is the IP address of the master)
Note: It is recommended to use Google browser, other browsers may not be able to
open the image
```