# 1 Multi-machine handle control

## 1.1 Multi-machine configuration

When using multi-machine handle control, it is first necessary to ensure that the robot is under the same local area network and configured with the same [ROS_MASTER_URI]; for multiple robots to control motion, there can only be one host. The example in this section sets the virtual machine as the host, and other robots as the slaves. There are several slaves. Of course, you can also set one robot as the master and others as the slaves.

### 1.1.1 Multi-machine communication settings

View the IP of the virtual machine

```
ifconfig
```



Next, you only need to modify the .bashrc file of the slave(Robot side). There are several slave configurations.

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

Find the following line

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

The [IP] here is the IP of the(virtual machine side).

```
export ROS_MASTER_URI=http://192.168.2.106:11311
```

After setting the IP, it is best to refresh the environment variables.

```
source ~/.bashrc
```

## 1.1.2 Multi-machine time synchronization

Install

```
sudo apt install ntp ntpdate
```

time check command

```
date -R
```

When there is a network, the system will automatically synchronize the network system time without setting. If there is no network, the time needs to be set when the time between robots is different

- have network

Commands to automatically synchronize network time

```
ntpd pool ntp.ubuntu.com
```

- No network

The test case is to use two robots.

Commands to manually set the time

```
sudo date -s "2020-01-1 01:01:01"
```

1. server side(host side) configuration

Open the [ntp.conf] file

```
sudo vim /etc/ntp.conf
```

add at the end of the file

```
restrict  192.168.2.0  mask  255.255.255.0  nomodify  notrap
server  127.127.1.0  # local clock
fudge  127.127.1.0  stratum  10
```

The first line is to enable the machine on the 192.168.2.xxx network segment to synchronize time with the local machine(specifically see if your ip is 192.168.2.xxx, if not, change it to your actual format)  The second and third lines are to synchronize the hardware time of the machine with the ntp service of the machine.

After the change, restart the ntp service

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

The server-side settings are completed, and the client-side settings are started below.

2. client(slave) configuration

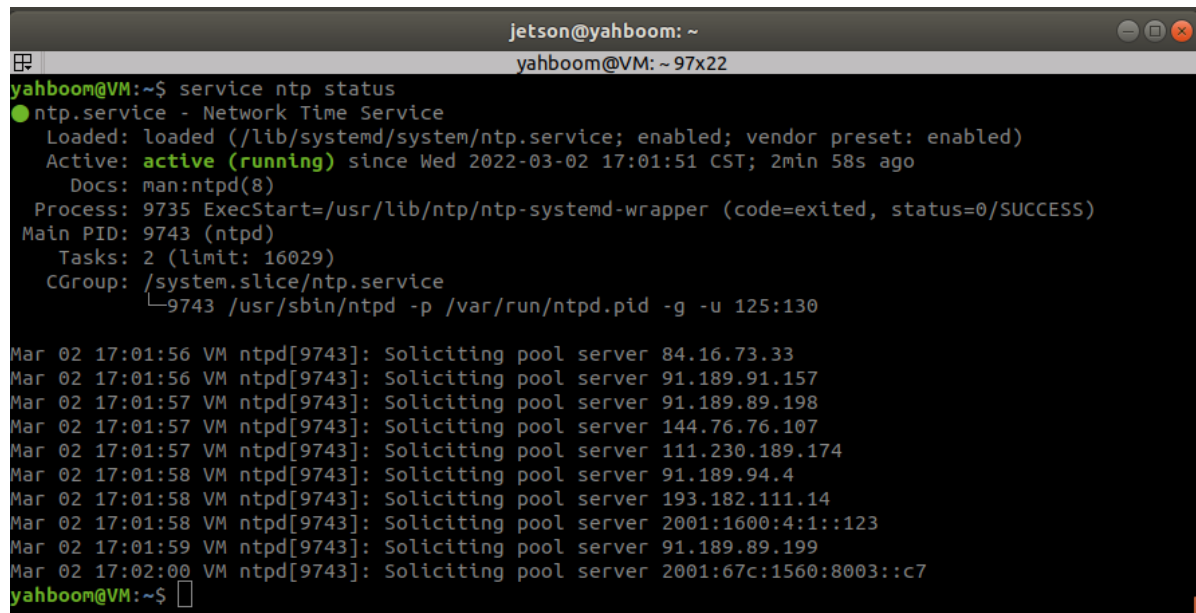Open the [ntp.conf] file

```
sudo vim /etc/ntp.conf
```

Add [server]+[IP] at the end of the file

```
server 192.168.2.106
```

3. time synchronization

On the [server side], execute the following code to check whether the [server side] ntp service is running

```
service ntp status
```



If not, restart the ntp service.

Network update(required for all devices)

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

time synchronization

```
sudo ntpdate 192.168.2.106
```

If the client's ntp is also enabled, you need to close the ntp service first, and then perform time synchronization.

```
sudo /etc/init.d/ntp stop
```

## 1.2 use

Take the virtual machine as the master and two robots as slaves as an example.

### 1.2.1 Turn on the robot

virtual machine side

```
roscore
```

Start the command(robot1 side), for the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch  yahboomcar_multi  laser_bringup_multi.launch  ns := robot1
  # laser + yahboomcar
roslaunch  yahboomcar_multi  laser_usb_bringup_multi.launch  ns := robot1
  # mono + laser + yahboomcar
roslaunch  yahboomcar_multi  laser_astrapro_bringup_multi.launch  ns := robot1
 # Astra + laser + yahboomcar
```

Start command(robot2 side), for the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch  yahboomcar_multi  laser_bringup_multi.launch  ns := robot2
  # laser + yahboomcar
roslaunch  yahboomcar_multi  laser_usb_bringup_multi.launch  ns := robot2
  # mono + laser + yahboomcar
roslaunch  yahboomcar_multi  laser_astrapro_bringup_multi.launch  ns := robot2
 # Astra + laser + yahboomcar
```

More bots and so on.

### 1.2.2 open the handle control

Method 1: One handle controls multiple robots at the same time

Connect the controller receiver to the virtual machine USB port and start the command

```
roslaunch yahboomcar_multi joy_multi.launch
```

**Note: The [launch] file defaults to three robots.**

joy_multi.launch

```
< launch >
    < arg  name = "first_robot1"  default = "robot1" />
    < arg  name = "second_robot2"  default = "robot2" />
    < arg  name = "third_robot3"  default = "robot3" />
    < param  name = "use_sim_time"  value = "false" />
    < node  name = "joy_node"  pkg = "joy"  type = "joy_node"  output = "screen"
  respawn = "false" />
    <!--  ######################### first_robot1
######################### -->
    < include  file = "$(find yahboomcar_multi)/launch/library/joy_base.launch"
 >
```

```
            < arg  name = "ns"  default = "$(arg first_robot1)" />
    </ include >
    <!--  ########################## second_robot2
 ######################## -->
    < include  file = "$(find yahboomcar_multi)/launch/library/joy_base.launch"
 >
            < arg  name = "ns"  default = "$(arg second_robot2)" />
    </ include >
    <!--  ######################### third_robot3
 ######################### -->
    < include  file = "$(find yahboomcar_multi)/launch/library/joy_base.launch"
 >
            < arg  name = "ns"  default = "$(arg third_robot3)" />
    </ include >
</ launch >
```

If there are 2 robots, just comment out the part of the third robot, and it will not affect if you do not comment out; if there are 4 robots or more, you can add them according to the first 3 samples.

Method 2: One handle controls one robot alone

Connect the controller receiver to the corresponding robot, and start the command on the corresponding robot side(take robot1 as an example)
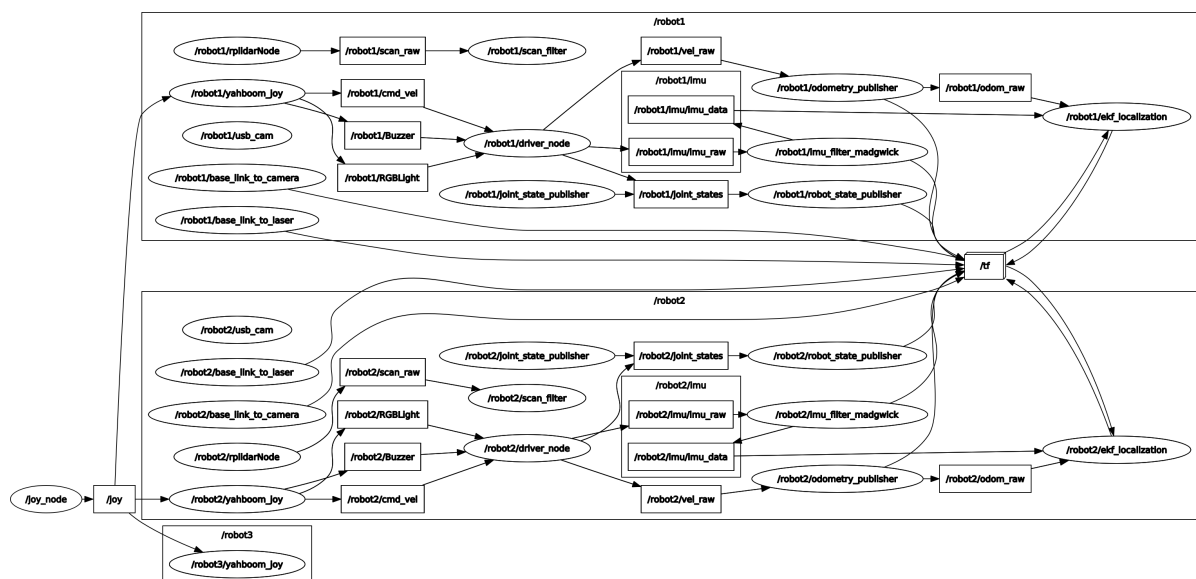
```
roslaunch yahboomcar_multi joy_each.launch ns:=robot1
```
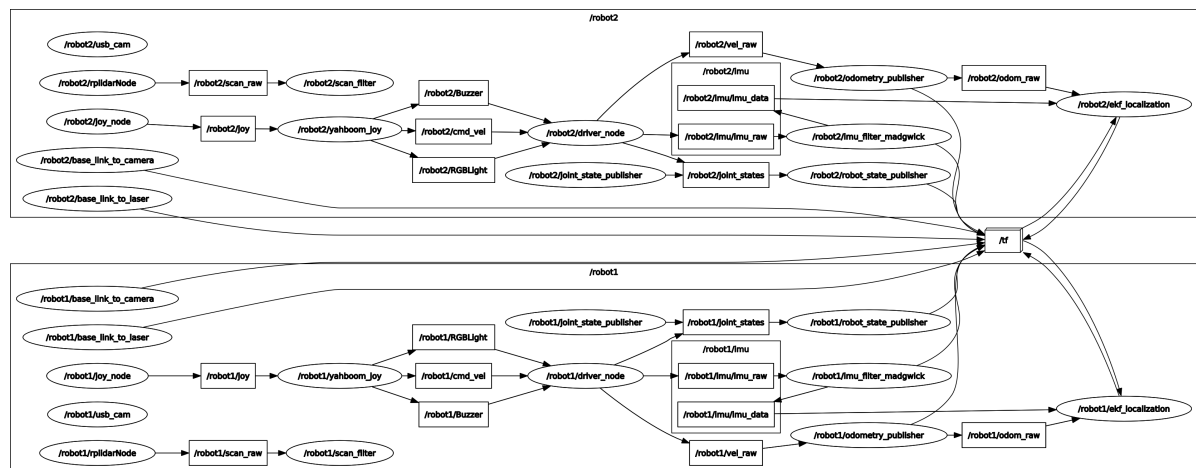
# 1.3 handle control analysis

Node view

```
rqt_graph
```

## 1.3.1 Controlling multiple robots



It can be seen from the figure that when a handle controls multiple robots at the same time, only one [joy_node] node is needed to receive the handle signal, and different [yahboom_joy] nodes are set for different robots to control the corresponding robot.

## 1.3.2 Controlling a robot



It can be seen from the figure that when a handle controls a robot, each robot is individually set and controlled.