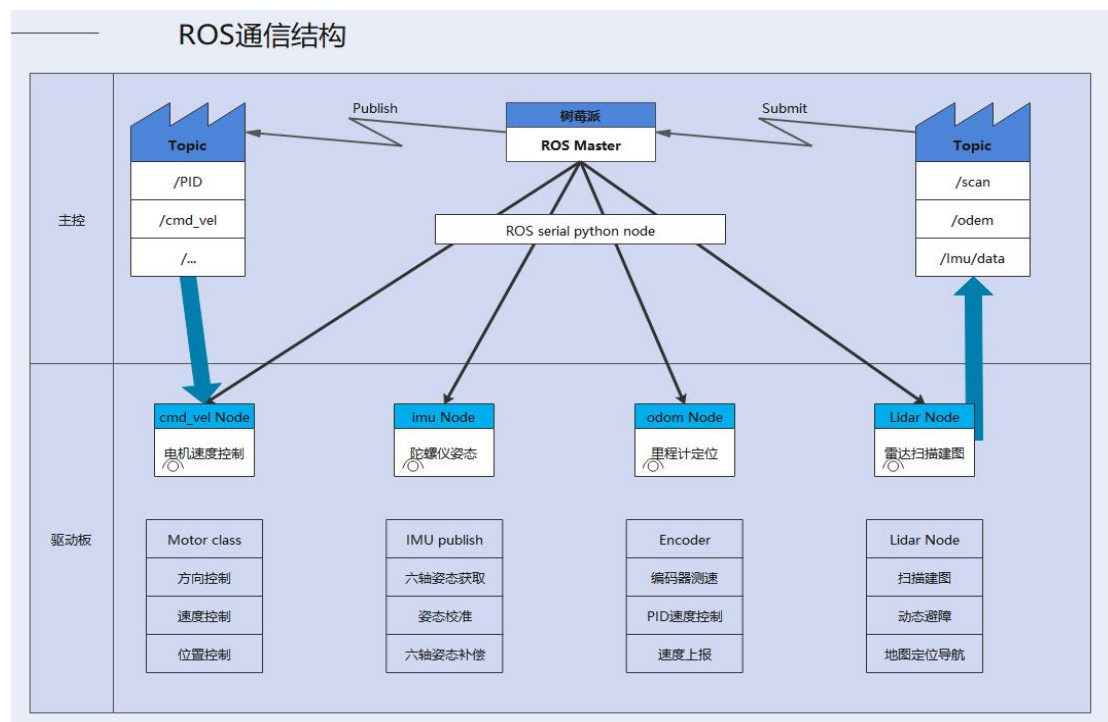


Mbsbot hardware

1. ROS control end description

ROS car control is mainly composed of raspberries pie 3 b + / raspberry pie 4 b + as a master, because the raspberries pie pin resources are limited, so the task of driving multi-channel acquisition speed motor and encoder usually to drive plate processing, motor drive car there are STM32 and ArduinoMega2560 two versions, the following will respectively introduce the different communication principle and using method of the board.

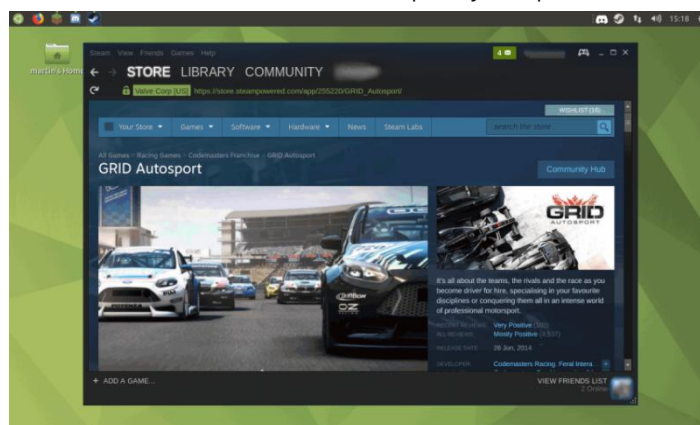


1.1. Raspberry PI end ROS primary node

Raspberry Pi is a tiny computer the size of a credit card, based on Linux. The system used by our board is generally Ubuntu Mate. Its lightweight desktop environment and UI beautification are much more friendly than the Debian provided by the official Raspberry Pi. It is suitable for students who have certain Ubuntu foundation to quickly adapt to it.

You can download the official image installation experience:
 Ubuntu Mate official website:
<https://ubuntu-mate.org/>

Ubuntu mate desktop:



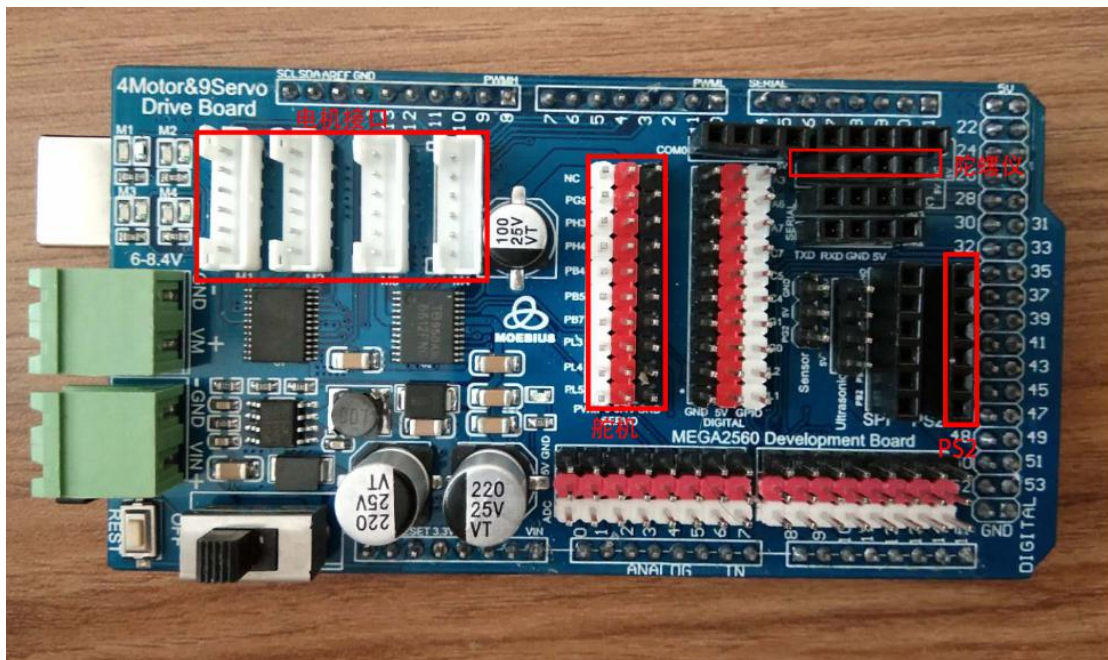
Due to the complexity of ROS configuration, we have prepared a ROS image file that has been configured and put it in the environment configuration directory of the data pack. You can install the system image for your raspberry PI according to the image installation tutorial in the image folder

Our mirror image is mainly composed of Ubuntu Mate16.04 LTS and Ubuntu Mate20.04LTS, which correspond to the Kintic version of ROS and Noetic version respectively. Since Ubuntu Mate20.04LTS was newly released this year, ROS Noetic is not well supported on Ubuntu 20.04LTS. It will be continuously updated in the subsequent development.

2. driver board

The Mbsbot driver board is mainly composed of two versions STM32 and Arduino, STM32 version is suitable for the developers want to insight into the motor control and familiar with STM32 classmates, code reading need to have a certain basis of C/C++ , the Arduino driven plate for quick access to ROS, only focus on ROS upper classmates, code need some simple configuration and the installation of the library can be rapid development.

ArduinoMega2560



Board introduction:

ArduinoMega2560 is a powerful single-chip microcomputer with rich resources. It can drive motors, steering gears, gyroscopes and other peripherals through simple configuration. There are 4 encoding motor interfaces in the upper left corner of the board, and two TB6612 drive chips control 9-12V DC The motor, the forward and reverse indicator lights next to it can show the current movement status of the motor. There are 9 steering gear drive pins in the middle of the board, which can be connected to the robot arm or steering gear pan/tilt. The upper right corner is some common interfaces: IIC/UART/ SPI, etc., are all led out by way of row mothers, and sensors can be installed according to their needs.

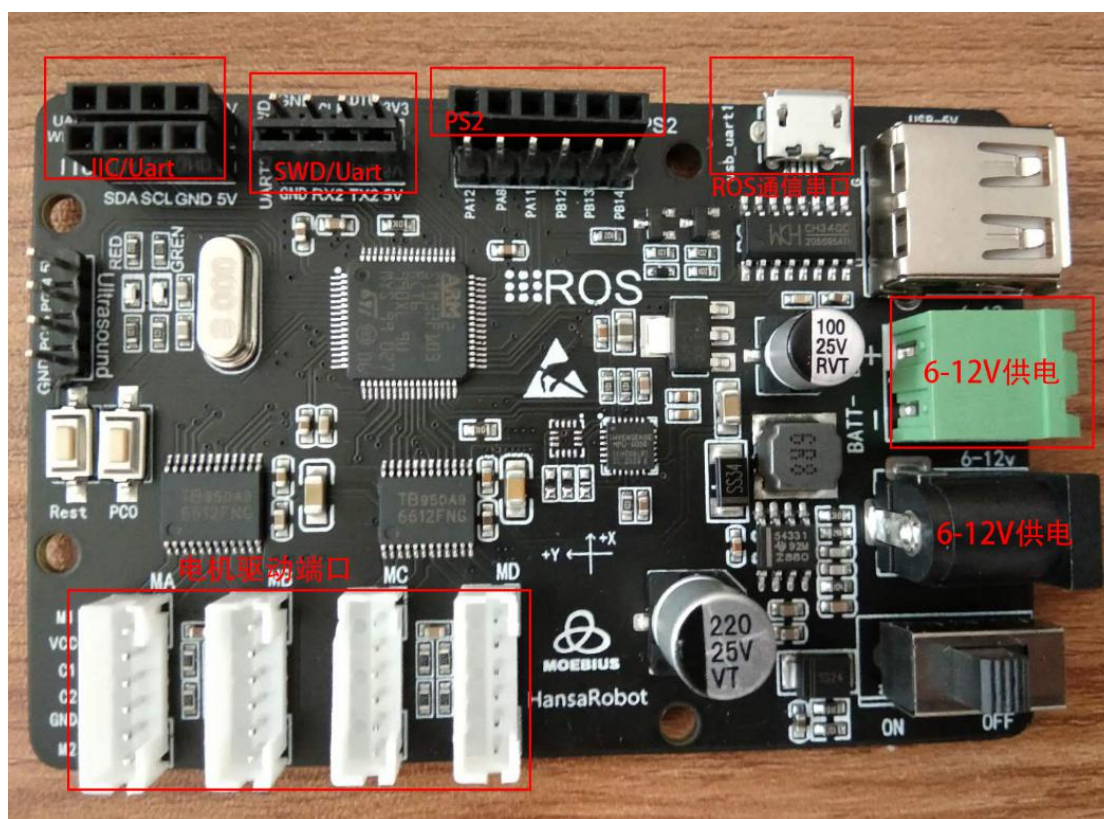
Communication and power supply:

ArduinoMega2560ROS communicates with the Raspberry Pi through the USB serial port in the upper left corner. The board is powered by 9-12V from the green terminal on the left. Before plugging in, make sure that the lower switch is off.

ArduinoMega2560 program burning:

Arduino programming is very simple. Copy the lib library under the project file to the libraries folder under your ArduinoIDE installation directory, then return to the IDE to select the corresponding serial port number and device type, and click upload.

STM32



板子介绍:

Board introduction:

The onboard chip model is STM32F103RCT6, the main parameters: 32-bit Cortex-M3 ARM core, 256kbFlash, 64kbRAM. Compared with Arduino, STM32 has a clock speed of up to 72Mhz, rich peripheral resources, and processing speed and resources are unmatched by Arduino. So as a developer, choosing STM32 is more appropriate.

interface:

Similarly, there are 4 motor ports on the STM32 board, which can drive 4 DC motors at the same time. The board comes with a gyroscope and can be used directly as a motion controller. It also has an external SWD debugging interface and IIC/SPI/PS2/UART communication. interface.

Communication mode and power supply mode:

The board communicates with the Raspberry Pi host through the MicroUSB port. The power supply mode can use the green terminal to input 9-12V power, or it can be directly powered through the DC plug.

STM32 program burning:

STM32 can be burned through the JlinkV2 tool to connect to the SWD port. The upper computer can use Keil or Jlink tools. The burning method can also choose serial burning, and the upper computer uses FlyMCU.

2. Linux basic knowledge and development environment



Under the premise of learning ROS, you need to have a certain basic knowledge of Linux, and you must be familiar with the most basic operations
This tutorial will introduce you how to use Linux.

----If you have a certain foundation, you can jump directly to-Ubuntu install ROS environment

Linux is actually very easy to learn, I believe you can learn it soon.

What knowledge is required?

If you are familiar with operating system knowledge, I believe you will learn Linux soon.
This tutorial will take the Linux release version ubuntu as an example to introduce the application of Linux system.

Introduction to Linux

Under the premise of learning ROS, you need to have a certain basic knowledge of Linux, and you must be familiar with the most basic operations
This tutorial will introduce you how to use Linux.

----If you have a certain foundation, you can jump directly to-Ubuntu install ROS environment

Linux is actually very easy to learn, I believe you can learn it soon.

What knowledge is required?

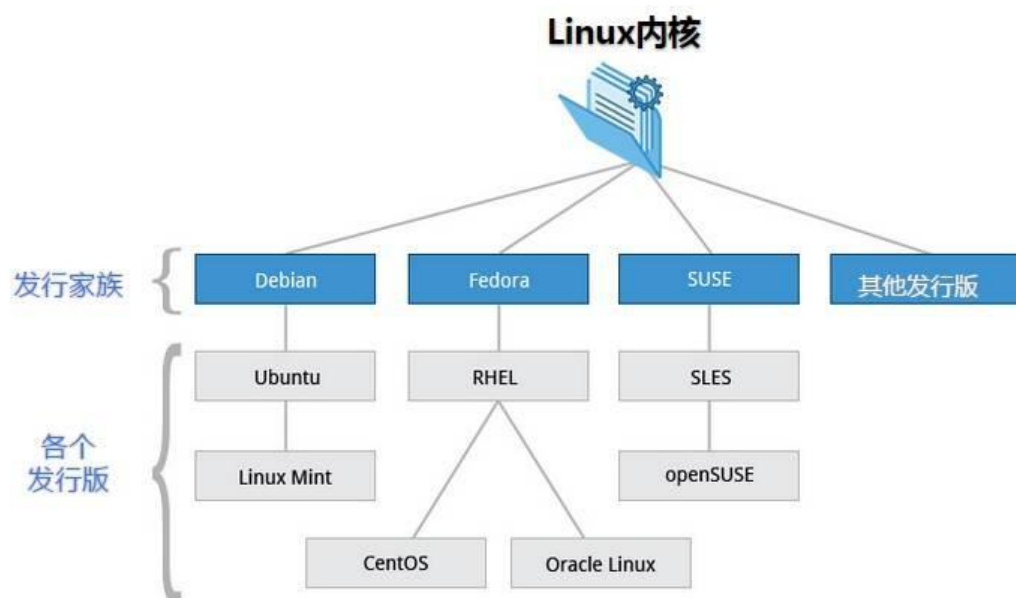
If you are familiar with operating system knowledge, I believe you will learn Linux soon.

This tutorial will take the Linux release version ubuntu as an example to introduce the application of Linux system.

Introduction to Linux

Linux distribution

The simple point of the Linux distribution is to package the Linux kernel and application software.



The more well-known distributions currently on the market are: Ubuntu, RedHat, CentOS, Debian, Fedora, SuSE, OpenSUSE, Arch Linux, SolusOS, etc.



Linux application

Today, various Linux distributions are used in various occasions, from embedded devices to

supercomputers, and have established a position in the server field. Usually the server uses LAMP (Linux + Apache + MySQL + PHP) or LNMP (Linux + Nginx + MySQL + PHP) combination.

Currently Linux is not only used in homes and businesses, but also very popular in the government.

- The Brazilian Federal Government is world-renowned for supporting Linux.
- There are news reports that the Linux distribution version manufactured by the Russian army itself has achieved results as a G.H.ost project.
- India's Kerala Federation plans to promote the use of Linux in high schools across the Federation.
- To achieve technological independence, the People's Republic of China uses Linux exclusively in the Loongson processor.
- Developed its own Linux distribution in some areas of Spain, and is widely used in the government and education fields, such as GnuLinEx in the Extremadura region and Guadalinux in the Andalusia region.
- Portugal also uses its own Linux distribution, Caixa Mágica, for Magalhes laptops and e-escola government software.
- France and Germany have also begun to gradually adopt Linux.

Linux vs Windows

At present, domestic Linux is more used on servers, while desktop operating systems are more Windows. The main differences are as follows

Compare	Windows	Linux
interface	The interface is unified, the shell program is fixed, the menus of all Windows programs are almost the same, and the shortcut keys are almost the same	The style of the graphical interface differs depending on the release version and may not be compatible with each other. GNU/Linux terminals are inherited from UNIX, and the basic commands and operating methods are almost the same.
driver	The driver is rich and the version is updated frequently. The default installation program generally contains the popular hardware drivers when the version is released, and the new hardware drivers released afterwards depend on the hardware manufacturers. For some old hardware, it is sometimes difficult to support without the original driver. In addition,	Developed by volunteers and released by the Linux core development team. Many hardware manufacturers do not provide drivers due to copyright considerations. Although most hardware manufacturers do not need to install manually, the installation is relatively complicated, causing new users to face driver problems (whether they exist and install Method) will be

	sometimes the hardware manufacturer does not provide the required version of the driver under Windows, which can be a headache.	helpless. However, in the open source development mode, many old hardware is easy to find drivers even though it is difficult to support under Windows. Hardware vendors such as HP, Intel, AMD are gradually supporting open source drivers to varying degrees, and the problem is being alleviated.
User	It is relatively simple to use and easy to get started. The graphical interface is very beneficial for users without computer background knowledge.	The graphical interface is simple to use and easy to get started. The text interface requires learning to master.
learn	The system structure is complex, changes frequently, and knowledge and skills are eliminated quickly, making it difficult to learn deeply.	The system structure is simple and stable, and the inheritance of knowledge and skills is good, and it is relatively easy to study in depth.
soft	Each specific function may need the support of commercial software, and the corresponding authorization needs to be purchased.	Most of the software is freely available, and there are fewer choices of software with the same function.

Linux common operations

Show files and directories ls (list)

ls Get a list of all files in the current directory

ls -a displays all files and folders in the directory, including hidden files

ls -l Get detailed information of files in the current directory

ls -al Get detailed information of files in the current directory, including hidden files

ls /usr Get a list of all files under the usr file

Directory jump/change directory cd (change directory)

cd / jump to the root directory

cd into the main folder

cd /usr jump to usr folder

cd ../ Return to the previous directory

*The directory with a slash represents the absolute path under the root directory, and the directory without a slash represents the relative path under the current directory

Clear the screen clear

Print the current working directory pwd (print working directory)

Query system information uname (unix name)

View file content cat (concatenate)

Switch permissions su (switch user)

sudo su root switch root user authority

su usr switch usr user permissions

Create file touch

File deletion rm (Remove)

rm -r delete read-only files

rm -f delete all files in the directory

rmdir Remove Directory (remove directory)

mkdir Make Directory (create directory)

mkdir -p creates a directory if it does not exist

Copy files cp (copy)

Move file mv (move)

Modify the read and write mode chmod (Change mode)

Load configuration file source

1. Install ROS on Ubuntu

In this section, we will introduce you to the installation of Linux. The installation steps are relatively cumbersome. Nowadays, cloud servers are quite common and the price is cheap. If you don't want to set up directly, you can buy one for learning and use. Refer to the comparison of major cloud servers.

This chapter takes ubuntu18.04 as an example.

ubuntu18 download address:

You can download the latest version from the official website:

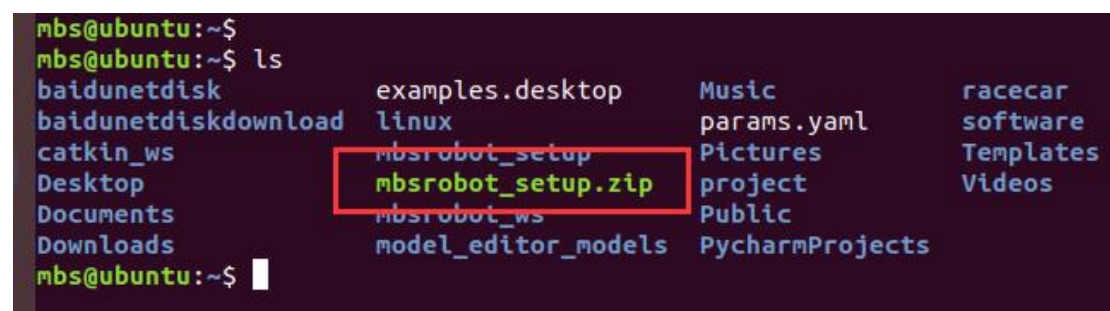
<https://ubuntu.com/download/desktop>

Official virtual machine download link: <https://www.vmware.com/>

Or follow the tutorial to install in our data package

After the installation is complete, turn on the virtual machine:

Put our environment configuration package under the user directory and run the configuration file to quickly configure the ROS environment



```
mbs@ubuntu:~$
mbs@ubuntu:~$ ls
baidunetdisk      examples.desktop  Music             racecar
baidunetdiskdownload  linux            params.yaml       software
catkin_ws          mbsrobot_setup   Pictures          Templates
Desktop            mbsrobot_setup.zip  project          Videos
Documents          mbsrobot_ws       Public
Downloads          model_editor_models PycharmProjects
mbs@ubuntu:~$
```

After decompression, you can see that there are many installation files:

Use the chmod command to add execution permissions to the shell file

chmod +x ~/mbsrobot_setup/*


```
mbs@ubuntu:~/mbsrobot_setup$ ls
49-teensy.rules      driver              mbsnode_network.sh
558-orbbec-usb.rules installROS.sh       mbs_udev.py
58-mbs.rules         mbshost_network.sh ros_packages_install.sh
dev_tools_install.sh mbs_network.sh      User
mbs@ubuntu:~/mbsrobot_setup$ chmod +x ~/mbsrobot_setup/*
mbs@ubuntu:~/mbsrobot_setup$ ls
49-teensy.rules      driver              mbsnode_network.sh
558-orbbec-usb.rules installROS.sh       mbs_udev.py
58-mbs.rules         mbshost_network.sh ros_packages_install.sh
dev_tools_install.sh mbs_network.sh      User
mbs@ubuntu:~/mbsrobot_setup$
```

Among them, installROS.sh is to install the basic ROS environment, use gedit to open and edit

```
mbs@ubuntu: ~/mbsrobot_setup
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
mbs@ubuntu:~/mbsrobot_setup$ ls
49-teensy.rules      mbshost_network.sh
558-orbbec-usb.rules mbs_network.sh
58-mbs.rules         mbsnode_network.sh
dev_tools_install.sh mbs_udev.py
driver              ros_packages_install.sh
install_ROS.sh      User
mbs@ubuntu:~/mbsrobot_setup$ gedit install_ROS.sh
```

```
# e-mail 1260105099@qq.com
# version 1.0.1
```

```
# Moebius ROS install erector
```

```
# -----Install ROS environment-----
# ROS Variables
#rosversion="kinetic" #Ubuntu mate 14.04LTS/16.04LTS
#rosversion="melodic" #Ubuntu mate 18.04LTS
#rosversion="noetic" #Ubuntu mate 20.04LTS

#Get current path
cur_dir=$(pwd)
echo "Current path: $cur_dir"
# Install the ros
```

Choose the ROS version you need to install according to your ubuntu version here, my system here is ubuntu18.04

Just remove the previous comment, then save and close by ctrl+s

```
mbs@ubuntu:~/mbsrobot_setup$ gedit install_ROS.sh
mbs@ubuntu:~/mbsrobot_setup$ ./install_ROS.sh
```

Use ./install_ROS.sh to run the installer. The installation process will take some time, please be patient

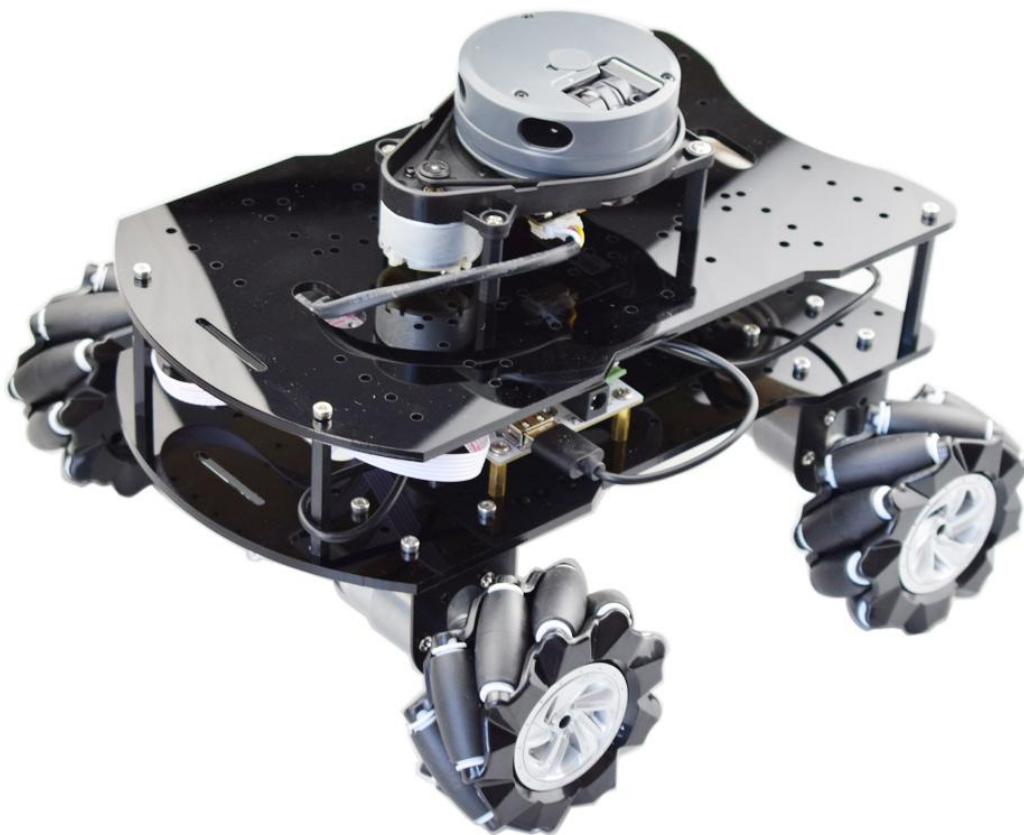
```
mbs@ubuntu:~/mbsrobot_setup$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.l
ist.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/
rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/
rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/
rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/
rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/
releases/fuerte.yaml
Query rosdistro index https://raw.githubusercontent.com/ros
/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Skip end-of-life distro "crystal"
Add distro "dashing"
Add distro "eloquent"
Add distro "foxy"
Skip end-of-life distro "groovy"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Add distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
Add distro "rolling"
updated cache in /home/mbs/.ros/rosdep/sources.cache
```

Run dependencies after installation

```
mbs@ubuntu:~/mbsrobot_setup$ ./extern_pkg_install.sh
[sudo] mbs 的密码:
命中:1 http://mirrors.cn99.com/ubuntu bionic InRelease
命中:2 http://packages.microsoft.com/repos/vscode stable In
Release
命中:3 http://mirrors.cn99.com/ubuntu bionic-updates InRele
ase
命中:4 http://mirrors.cn99.com/ubuntu bionic-backports InRe
lease
命中:5 http://mirrors.cn99.com/ubuntu bionic-security InRel
ease
命中:6 http://packages.ros.org/ros/ubuntu bionic InRelease
命中:7 http://packages.osrfoundation.org/gazebo/ubuntu-stab
le bionic InRelease
正在读取软件包列表... 完成
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件:
```


3 Network configuration and remote control

After the user gets the car and assembles it by himself, the first thing to do is to configure the ROS network of the car. The purpose of configuring the ROS network is that we can remotely control or view the data of the ROS system running on the main control, so that there is no need to let A display is installed on the car, and we can directly operate it remotely. Since all our series of ROS models are used in the same way, here I will take the Mecanum wheel model as an example to explain to you.



1. Preparation before use

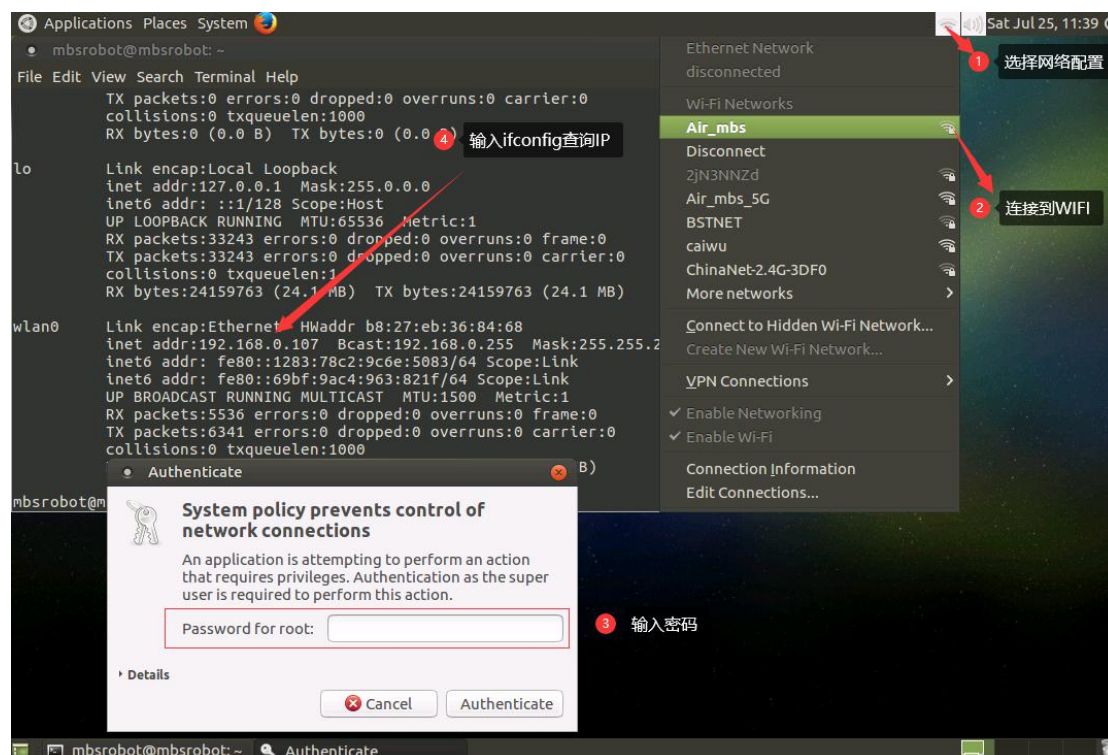
Please charge the battery first, and then you need to prepare a monitor with HDMI, a USB keyboard and mouse (the two are only used to facilitate the configuration of the main control network), and a router to form a local area network, please note that it must be Local area network, for example, after the user gets the car, the main control (Raspberry Pi or industrial computer, the main control mentioned later refers to the Raspberry Pi or industrial computer) system is connected to the campus network or company LAN, or other Local area networks that are not purely meaningful will cause the ROS network configuration to fail, and ultimately the car cannot be controlled remotely.

*After the trolley is assembled, be sure to check whether the connection of each part is

normal, whether the motor can be rotated by hand, and if there is no abnormal situation, then power on and run the system. (Short circuit or motor stuck during installation, direct power on may damage the hardware)

Second, configure the ROS network

1. Power on the main control, the user password of the mbsrobot main control end and the remote end are both "123456", enter the desktop, connect the main control end to the network, check the IP of the main control, and open a terminal of the system. The key is "Ctrl+Alt+T" (press at the same time), the command to be viewed is as follows, and then the following figure appears. The "192.168.0.107" in it is the current IP of the master. This is different because of the personal network environment. Yes, so please remember after you check it, this IP, we need to use later:



2、Change the configuration of the car model. Since our car is multi-faceted, we configure your current user's model by parameter configuration. If the user gets the Mecanum wheel drive and the radar is Silan A1, the user can skip After this step, also open a terminal on the main control side, then execute the following command and move to the bottom of the file (Users who don't use vim can use gedit. Please use Baidu for vim and gedit. Please pay attention to the difference between commands and commands. Space),

gedit ~/.bashrc

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
IP address: [192.168.149.132]
Lidar type: delta_ros
Robot type: mecanum
mbs@ubuntu:~$ gedit ~/.bashrc

wlan_interface=wlan0
interface=ens33

#网卡获取IP
HOST_IP_ADDR=`ifconfig $interface | grep -o 'inet [^ ]*' | cut -d" " -f2`
WLAN_IP_ADDR=`ifconfig $interface | grep -o 'inet [^ ]*' | cut -d" " -f2`

#hostname获取ip
HOST_IP_ADDR=$(echo "$(hostname -I)" | sed 's/[ ]*/g')

ROBOT_IP_ADDR="192.168.0.108"

echo "IP address: [$HOST_IP_ADDR]"
export ROS_IP=$HOST_IP_ADDR
export ROS_HOSTNAME=$ROS_IP
export ROS_MASTER_URI=http://$ROS_IP:11311
export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:/catkin_make/src

export MBSLIDAR=delta_ros
export MBSBASE=mecanum
echo "Lidar type: $MBSLIDAR"
echo "Robot type: $MBSBASE"

```

The following MBSBASE is the model designation, we have "2wd (two-drive), 4wd (four-drive), mecanum (Micanna mother wheel), omni (Omni omnidirectional wheel), omni4wd (Omni omnidirectional four-wheel), tank (Track), ackermann (Ackerman (steering gear steering))": Mecanum wheels are used here, so MBSBASE is changed to mecanum. The MBSLIDAR inside is a designated radar model. Our car is compatible with a variety of radars, that is, it can be used after plugging it in. It does not need to install the driver. It is compatible with (Silan rplidar A1/A2, EAI's F4, X4, G4, weight Stone radar, fir technology delta_ros), this is changed according to your radar model. The radar folder is under ~/`catkin_ws/src/lidar`. Compare the name of the radar in your hand and change the corresponding MBSLIDAR to the radar SDK folder name. However, the radar used here is delta lidar:



delta_lidar



rplidar

After configuration, it will look like this

MBSBASE=mecanum

MBSLIDAR=delta_ros

3. Configure the ROS network on the remote end. Note that this is the configuration of the remote end. The user opens the virtual machine we provide. The virtual machine is in the cloud disk directory of "Virtual Machine Software and Mirroring" in the information we provided. Open the virtual machine and log in. Check whether the virtual machine's network

is connected. If you don't confirm whether the network is connected, open the terminal and use the ifconfig command to check the IP, confirm whether the virtual machine's IP is on the same network as the host, then open the terminal, execute the following command, move Go to the bottom of the file, please change the IP of the place marked in red, that is, "192.168.0.107" below, with the IP of your console above, don't move other places, then save and exit, and restart the system:

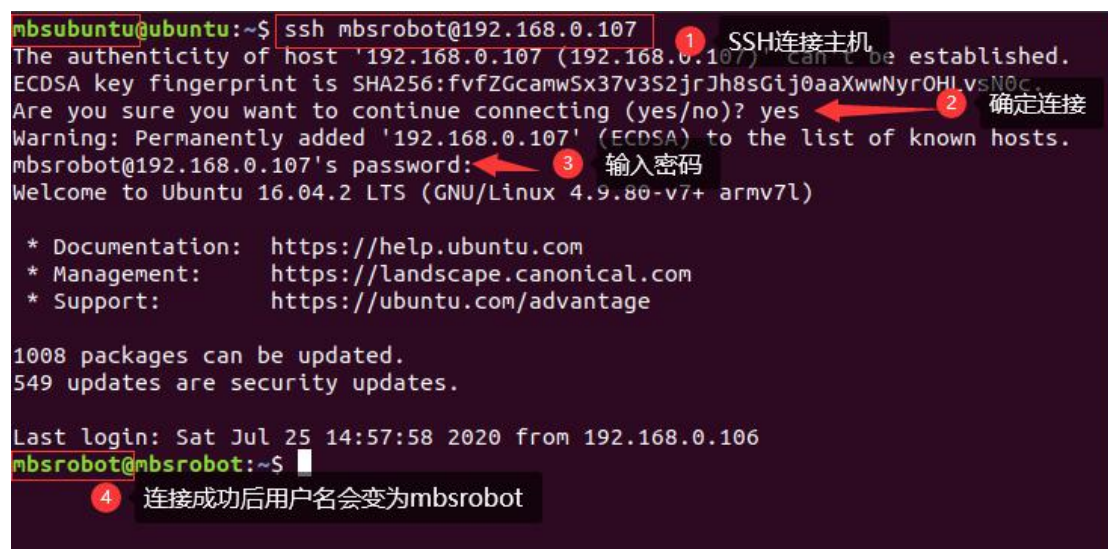
3. Prepare to remotely control the car

After the above configuration is completed, the basic configuration of the ROS network is completed. As long as it is in the local area network, the remote control is basically OK. The next step is to put the car in the ground or test it with an overhead power to make the ROS robot car move.

1. Log in to the console from the remote ssh remotely. If you can't solve it or users who don't know how to use ssh, please use Baidu. Please execute the following command and replace the following ip with the ip address of your console.

`$ssh mbsrobot@192.168.0.107` (please replace the IP with the IP of your own host)

Then you will be prompted to enter the password. The password is 123456. It is possible that when you log in for the first time, you will be prompted whether to log in. After you enter "yes", enter the password



```
mbsubuntu@ubuntu:~$ ssh mbsrobot@192.168.0.107
The authenticity of host '192.168.0.107 (192.168.0.107)' can't be established.
ECDSA key fingerprint is SHA256:fvfZGcamwSx37v3S2jrJh8sGij0aaXwwNyr0HlvsN0C.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.107' (ECDSA) to the list of known hosts.
mbsrobot@192.168.0.107's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.9.80-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1008 packages can be updated.
549 updates are security updates.

Last login: Sat Jul 25 14:57:58 2020 from 192.168.0.106
mbsrobot@mbsrobot:~$
```

2. After the login is successful, you can start the car and execute the following command. Please remember that as long as you start the car, you must log in to the main console to execute this command every time. The following interface appears after startup. It is normal. If there is a startup exception, Please refer to the following common problem solutions:

`$roslaunch mbsrobot bringup.launch`

```
mbsrobot@mbsrobot:~$ roslaunch mbsrobot bringup.launch
... logging to /home/rikirobot/.ros/log/cca7df3c-ce46-11ea-ac9f-b827eb368468/ros
launch-mbsrobot-15637.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

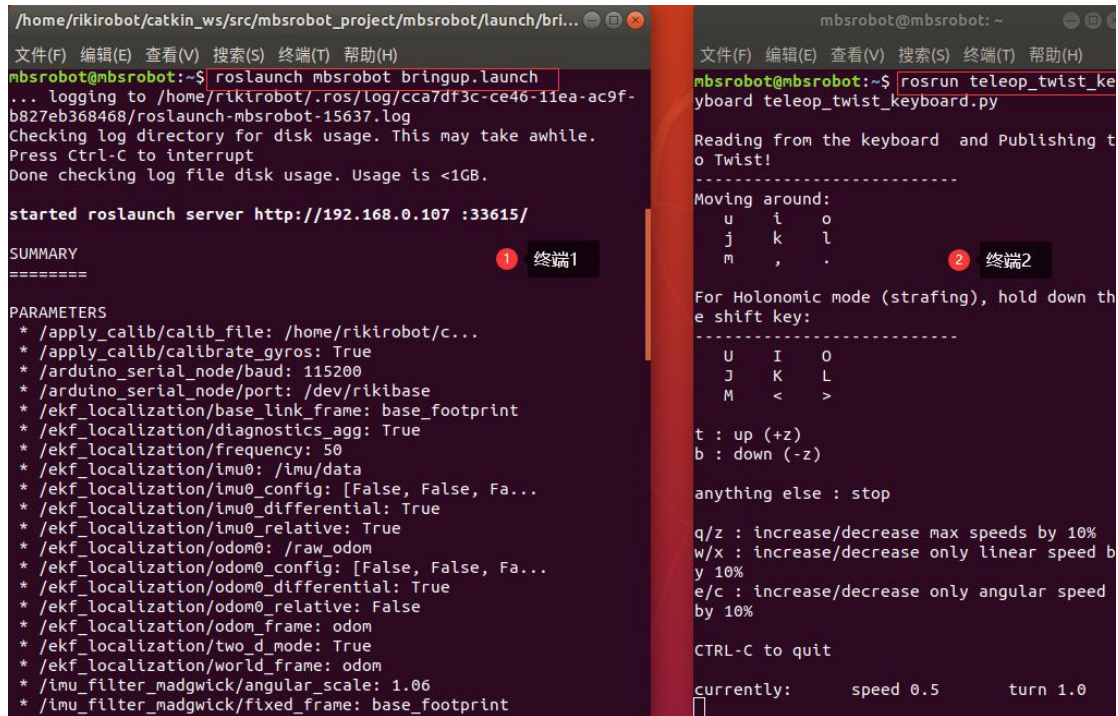
started roslaunch server http://192.168.0.107 :33615/

SUMMARY
=====

PARAMETERS
* /apply_calib/calib_file: /home/rikirobot/c...
* /apply_calib/calibrate_gyros: True
* /arduino_serial_node/baud: 115200
* /arduino_serial_node/port: /dev/rikibase
* /ekf_localization/base_link_frame: base_footprint
* /ekf_localization/diagnostics_agg: True
* /ekf_localization/frequency: 50
* /ekf_localization/imu0: /imu/data
* /ekf_localization/imu0_config: [False, False, Fa...
* /ekf_localization/imu0_differential: True
* /ekf_localization/imu0_relative: True
```

3. Start the keyboard control on the remote side. Please execute some commands on the remote side, that is, the virtual machine side. You don't need to use SSH. Although SSH can also be controlled by the main console, if you can't control the car on the remote side, navigate to it. When sending navigation commands from the graphical interface of the remote end, it will not be successful. After starting the keyboard control, the following interface will appear. You stay on this interface, and then press the letter prompted above to control the movement of the car, and the "i" above will move forward. , "j" rotates left, "l" rotates right, you can try the others yourself, the McKenna mother wheel can move in all directions to turn on the capital mode, and then you can experience the control model of the omni wheel, that is, you can walk sideways and opposite Corner walk.

\$rosrun teleop_twist_keyboard teleop_twist_keyboard.py



The image shows two terminal windows. The left window, labeled '终端1', displays the output of a ROS launch command. It shows logging information, a check for disk usage, and the start of a roslaunch server at http://192.168.0.107:33615/. Below this, a 'SUMMARY' section is followed by a list of 'PARAMETERS' for the localization system, including settings for the IMU, odometry, and frame transformations. The right window, labeled '终端2', shows the output of the 'teleop_twist_keyboard.py' node. It displays a keyboard control interface with a grid of keys (u, i, o, j, k, l, m, ,, .) and their corresponding actions (up, down, left, right, etc.). It also shows the current speed (0.5) and turn (1.0) values.

Four, common problem solutions

Reasons for unsuccessful remote control

SSH is unsuccessful, please check whether the two networks are not connected well, whether they are in the same network segment, and whether they can be pinged

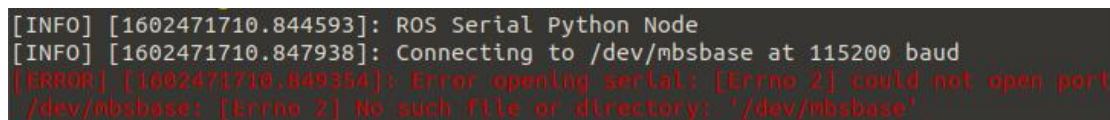


The image shows a terminal window with the command 'ssh mbsrobot@172.20.10.8' and the output 'ssh: connect to host 172.20.10.8 port 22: Connection refused'.

1. The ROS network is not configured properly, please check whether the network meets the above requirements
2. Please check if the battery is sufficient

Frequently asked questions about bringup.launch:

1. Failed to open the serial port: Check whether the serial port connection is normal, use 'ls -l /dev/mbs*' to check whether the driver is recognized
2. If it prompts 'ls: Cannot access '/dev/mbs*': There is no such file or directory, it means there is a problem with the USB data cable, please try another USB data cable



The image shows a terminal window with the following output: '[INFO] [1602471710.844593]: ROS Serial Python Node', '[INFO] [1602471710.847938]: Connecting to /dev/mbsbase at 115200 baud', and '[ERROR] [1602471710.849354]: Error opening serial: [Errno 2] could not open port /dev/mbsbase: [Errno 2] No such file or directory: '/dev/mbsbase''.

1. If the driver can be detected but the serial port cannot be opened, please use 'sudo chmod 777 /dev/mbs*' to give the serial port access permission
2. The IP assigned by the mobile hotspot includes the IPV6 part, and problems will occur when the Raspberry Pi gets it. Therefore, users who use mobile hotspots can first change ROS_IP=***.***.***.*** to the IP address of the Raspberry Pi

```
mbsrobot@mbsrobot:~$ roslaunch mbsrobot bringup.launch
invalid master URI: http://172.20.10.8 2409:8900:1a43:70a:789f:5714:f5f3:817e 24
09:8900:1a43:70a:dad1:5525:da8f:faad :11311
The traceback for the exception was written to the log file
```

gedit ~/.bashrc

export ROS_IP=\$HOST_IP_ADDR

Change to:

export ROS_IP=192.168.0.108

After modifying, remember to load the configuration from source ~/.bashrc

4 IMU linear velocity and angular velocity calibration

This part of the content will explain to you the calibration of the car's IMU, linear velocity, and angular velocity, why calibration is necessary. The robot is hardware. As long as it is hardware, the accuracy can be high or low. Especially the IMU calibration is more obvious. About IMU Error and calibration, there is a good article here, and there are also small differences in the assembly of the trolley that can also cause errors in the robot.

On the other hand, from the software level, the system obtains these data at a certain frequency, which will also produce certain errors. In theory, the higher the frequency of the software, the more values the robot can obtain per unit time, and the higher the accuracy will be. , But our processor capacity is limited (tips: ROS driver board is stm32 or arduino, the chip frequency is 72MHz, the code before version 1.03, our push frequency is 10hz, after version 1.9, we optimize the code and algorithm The push frequency has been increased to 50hz, and the accuracy in all aspects is higher), and the infinite frequency mode cannot be opened. Therefore, we need to calibrate the IMU, linear velocity and angular velocity of our car from both hardware and software aspects.

1. IMU calibration

The MPU6050 on the board in the picture below is a six-axis gyroscope. Different models of ROS cars have this module on the motor drive board.


```
mbsrobot@mbsrobot: ~/catkin_ws/src/mbsrobot_project/mbsrobot/param/imu
mbsrobot@mbsrobot:~$ roscd mbsrobot
mbsrobot@mbsrobot:~/catkin_ws/src/mbsrobot_project/mbsrobot$ cd param/imu/
mbsrobot@mbsrobot:~/catkin_ws/src/mbsrobot_project/mbsrobot/param/imu$ ls
imu_calib.yaml
mbsrobot@mbsrobot:~/catkin_ws/src/mbsrobot_project/mbsrobot/param/imu$
```

You execute the command to view the file (tips: ls is the command to view the current directory file), you can see that there is a file in the yaml format of imu_calib.yaml, which is the file where the final calibration data is saved, and the location of the terminal at this time Already in this directory, execute the following command to place the car in the corresponding posture, press the Enter key to start the calibration, wait for the Done to appear and the calibration is completed, follow the prompts to change the next posture until the calibration is completed. You need to press the Enter key 6 times. The calibration takes a certain amount of time because a large number of averages are sampled. You can wait slowly, and the calibration parameter file will be saved in imu_calib.yaml

Inside this file.

roslaunch imu_calib do_calib

```
mbsrobot@mbsrobot: ~
mbsrobot@mbsrobot:~$ roslaunch imu_calib do_calib
Orient IMU with X+ axis - Front side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
Done.
Orient IMU with X- axis - Rear side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
Done.
Orient IMU with Y+ axis - Right side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
^[[1;5 Done.
Orient IMU with Y- axis - Left side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
```

After the calibration is completed, we can check whether our IMU parameters are accurate.

Here we need to restart the bringup file to let the system reload the calibrated parameters, that is, to close the previously executed `roslaunch mbsrobot bringup.launch` command (tips:Ctrl+ C can close the current terminal application).

`roslaunch mbsrobot bringup.launch`

Open a terminal again and execute the command to view the topic of imu. After executing the following command, the interface of the calibrated IMU value will appear. Since we only use the angular velocity of the IMU at present, Therefore, we can see whether the various indicators of the angular velocity of the picture are normal:

`rostopic echo /imu/data`

```

header:
  seq: 21377
  stamp:
    secs: 1536308996
    nsecs: 705450241
  frame id: imu_link
orientation:
  x: 0.292116380681
  y: 0.225577775564
  z: -0.83512493538
  w: 0.403782797568
orientation_covariance: [0.0025000000000000005, 0.0, 0.0, 0.0, 0.00250000
00005]
angular_velocity:
  x: -0.0026468295604
  y: 0.00713915562141
  z: 0.000934889274649

```

This set of data is the quaternion of the IMU's final conversion

This set of data is the angular velocity data of IMU after EKF filtering. The car only USES the angular velocity of IMU at present, so we can see whether the accuracy of this value is normal if it is below plus or minus 0.01

Line speed of mbsrobot

Next, we calibrate the linear velocity of the car. The linear velocity of mbsrobot is calculated by motor rotation + encoder count + PID speed adjustment feedback to complete the linear velocity calculation. I will explain the specific theory in the advanced part.

First, we put the ROS car in the ground, power on the car, turn on the remote computer, we ssh from the remote end to the main control end to run, and set the calibration coefficient in the figure to 1.0:

`roscd mbsrobot`

`cd launch`

`vim bringup.launch`

```
<param name="linear_scale" type="double" value="3.00" />
```

Press "i" to enter the input mode, change 3.00 to 1.00, press ESC and enter: wq to save and exit

After the change is completed, start the car and execute

`roslaunch mbsrobot bringup.launch`

Execute the line speed calibration script remotely to the main control terminal in ssh:

roslaunch mbsrobot_nav calibrate_linear.py

Then open the terminal on the remote end and execute the following command. The calibration interface as shown in the figure below will appear, and then check the "start_test" check box in the figure below. At this time, the car will move forward and run 1m as the car thinks. After the car stops, Let's take a ruler to measure the start and end of the trolley to see how big the gap is. Here is an example, if the actual measured value is 1.08m, then we fill in 1.08 in the correction coefficient column of the interface, and then recalibrate until it is close to 1m Then remember the value you filled in the correction factor column, and then refill it back to bringup above. The modified linear_scale value is 1.0. If it is 1.08, change 1.0 to 1.08, so the linear velocity The calibration is complete:

roslaunch rqt_reconfigure rqt_reconfigure



Angular velocity calibration

The car uses the IMU to calculate the angular velocity, so IMU calibration must be performed before this step, otherwise the calibration is invalid

Put the ROS trolley underground, then start the trolley and execute:

roslaunch mbsrobot bringup.launch

Execute the angular velocity calibration script remotely to the host in ssh:

roslaunch mbsrobot_nav calibrate_angular.py

Then open the terminal on the remote end and execute the following command, the calibration interface will appear, and then check the "start_test" check box in the figure below. At this time, the car will rotate counterclockwise and run 360 degrees as the car thinks. After the car stops, Let's estimate whether it is 360 degrees and see how big the gap is. Here is an example. If the actual measured value is 370 degrees, then we fill in the correction coefficient

column of the interface 1.03 (tips: $370/360=1.03$), And then re-calibrate until it is close to 360 degrees, and then remember the value you filled in the correction coefficient column, and then re-fill it back to bringup above, the modified angular_scale value is 1.0, if it is 1.03, then 1.0 Change it to 1.03, so the linear velocity is corrected:

roslaunch rqt_reconfigure rqt_reconfigure



5 SLAM map construction of lidar

The gmapping algorithm is commonly used in the demonstration here to build maps. Of course, mbsrobot is very powerful and integrates a variety of slam algorithms. Composition algorithms such as Google's Cartographer, hector, kartgo, etc. have been supported. Just use video operations according to ours. There will be an article description.

1. First, we power on and start the car, and then connect to the radar (tips: rpliadr A1 radar will turn when power is on, and A2 radar needs to execute the map command after it is powered on), and then execute the command to start the car on the main control terminal:

roslaunch mbsrobot bringup.launch

2、Reopen a terminal window, execute the slam build command on the main console, and wait for the radar to load, as shown below:

roslaunch mbsrobot lidar_slam.launch


```

/home/rikirobot/catkin_ws/src/mbsrobot_project/mbsrobot/launch/lidar_slam.launch http://192.168.0.107:113
-xmin -50 -xmax 50 -ymin -50 -ymax 50 -delta 0.05 -particles 50
[ INFO] [1595925036.241535710]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.0324556 -0.0233802 -0.615543
m_count 0
Registering First Scan
[ INFO] [1595925036.784181655]: Resizing costmap to 1984 X 1984 at 0.050000 m/pi
X
[ INFO] [1595925036.873414061]: Received a 1984 X 1984 map at 0.050000 m/pix
[ INFO] [1595925036.897205147]: Using plugin "obstacle layer"
[ INFO] [1595925036.913464305]: Subscribed to Topics: scan
[ INFO] [1595925037.036552728]: Using plugin "inflation layer"
[ INFO] [1595925037.451606376]: Loading from pre-hydro parameter style
[ INFO] [1595925037.549007944]: Using plugin "obstacle layer"
[ INFO] [1595925037.560853175]: Subscribed to Topics: scan
[ INFO] [1595925037.679520221]: Using plugin "inflation layer"
[ INFO] [1595925037.951997978]: Created local_planner dwa_local_planner/DWAPlann
erROS
[ INFO] [1595925037.970502590]: Sim period is set to 0.20
[ INFO] [1595925038.894952986]: Recovery behavior will clear layer obstacles
[ INFO] [1595925038.931207996]: Recovery behavior will clear layer obstacles
[ INFO] [1595925039.142568049]: odom received!

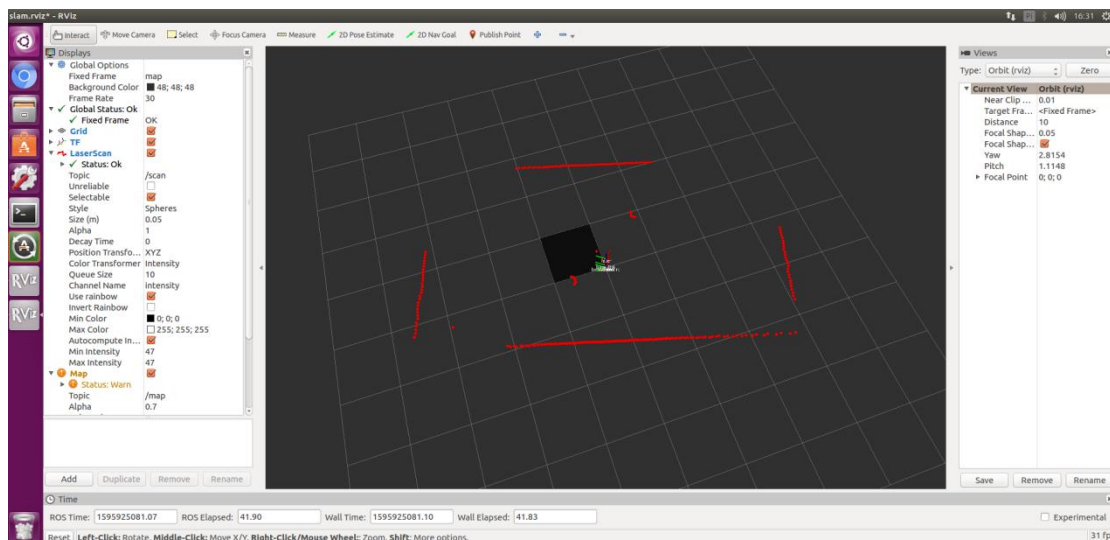
```

Seeing odom received means that the radar has started successfully.

3. Open the video tool rviz on the remote end, run the following command on the remote end, and then maximize the window that appears, then in the upper left corner, there is a File->OpenConfig, and then go to

Open slam.rviz under the path catkin_ws/src/mbsrobot_project/mbsrobot/rviz, and wait for the window to load the data:

roslaunch rviz rviz



4、 So far, the car has started to build the map with the slam algorithm. Next, we need to control the car to walk once in the area we need to build, sweep the construction area into a map file, and run the keyboard control command on the remote end to control the car to walk:

roslaunch teleop_twist_keyboard teleop_twist_keyboard.py


```
mbsrobot@mbsrobot: ~
rombsrobot@mbsrobot:~$ rosrn teleop_twist_keyboard teleop_twist_keyboard.py

Reading from the keyboard  and Publishing to Twist!
-----
Moving around:
    u    i    o
    j    k    l
    m    ,    .

For Holonomic mode (strafing), hold down the shift key:
-----
    U    I    O
    J    K    L
    M    <    >

t : up (+z)
b : down (-z)

anything else : stop

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%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
█
```

5、 If the constructed map has a clear boundary outline and is basically the same as the site environment, then you can save the map and enter the main control terminal. Please note that it must be the control terminal, not the remote terminal. Many users have made a mistake. Below the map directory, execute the map save command:

```
roscd mbsrobot
cd maps
./map.sh
```

Build problems

1. The constructed map is very messy, what is the situation?

Before we build the map, we must calibrate the IMU, linear velocity, and angular velocity. If there is no correction, please see the calibration IMU, linear velocity, and angular velocity above. The gmapping build map depends on the odometer, and the odometer uses linear velocity and angular velocity. Calculated, if the mileage data is inaccurate, it will lead to very messy construction, just like a chrysanthemum map

2. The linear velocity and angular velocity have been corrected, but I was still a bit messy when I created the map

Please check your radar installation, each type of radar has a direction, not just install it casually, you must pay attention

6 Lidar autonomous navigation

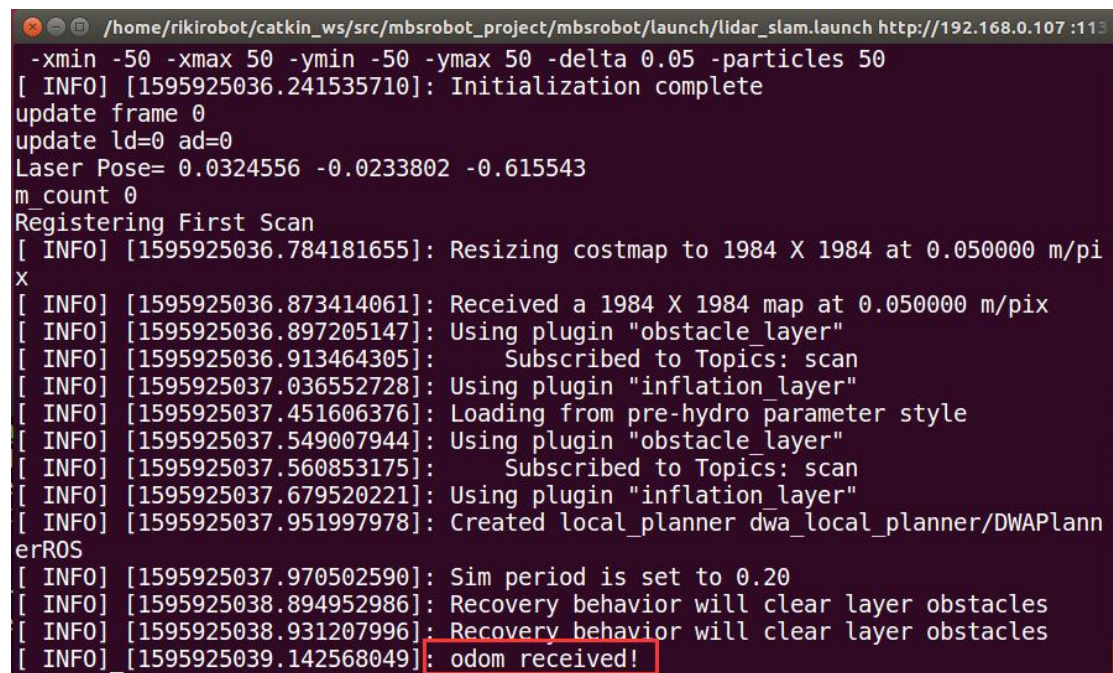
Automatic navigation using lidar

1. First, we power on and start the car, and then connect to the radar (tips: rpliadr A1 radar will turn when power is on, and A2 radar needs to execute the map command after it is powered on), and then execute the command to start the car on the main control terminal :

roslaunch mbsrobot bringup.launch

2、The command to start the navigation on the main control, wait for the start, as shown in the figure below:

roslaunch mbsrobot bringup.launch

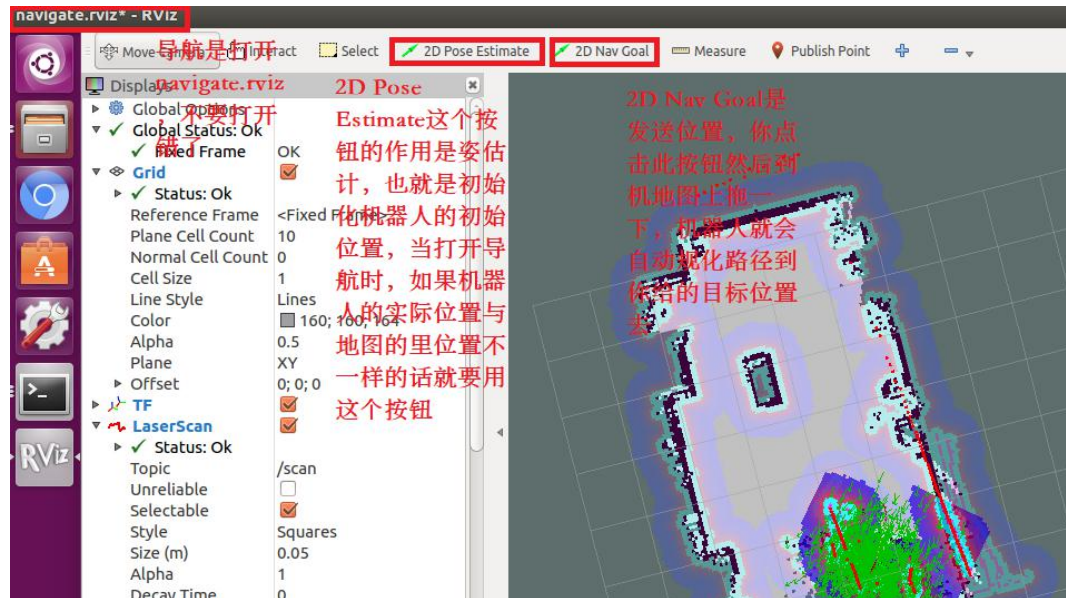


```
/home/rikirobot/catkin_ws/src/mbsrobot_project/mbsrobot/launch/lidar_slam.launch http://192.168.0.107:113
-xmin -50 -xmax 50 -ymin -50 -ymax 50 -delta 0.05 -particles 50
[ INFO] [1595925036.241535710]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.0324556 -0.0233802 -0.615543
m count 0
Registering First Scan
[ INFO] [1595925036.784181655]: Resizing costmap to 1984 X 1984 at 0.050000 m/pix
x
[ INFO] [1595925036.873414061]: Received a 1984 X 1984 map at 0.050000 m/pix
[ INFO] [1595925036.897205147]: Using plugin "obstacle_layer"
[ INFO] [1595925036.913464305]: Subscribed to Topics: scan
[ INFO] [1595925037.036552728]: Using plugin "inflation_layer"
[ INFO] [1595925037.451606376]: Loading from pre-hydro parameter style
[ INFO] [1595925037.549007944]: Using plugin "obstacle_layer"
[ INFO] [1595925037.560853175]: Subscribed to Topics: scan
[ INFO] [1595925037.679520221]: Using plugin "inflation_layer"
[ INFO] [1595925037.951997978]: Created local_planner dwa_local_planner/DWAPlann
erROS
[ INFO] [1595925037.970502590]: Sim period is set to 0.20
[ INFO] [1595925038.894952986]: Recovery behavior will clear layer obstacles
[ INFO] [1595925038.931207996]: Recovery behavior will clear layer obstacles
[ INFO] [1595925039.142568049]: odom received!
```

3、Open the video tool rviz on the remote side, run the following command on the remote side, and then maximize the window that appears, then in the upper left corner, there is a File->OpenConfig, and then go to catkin_ws/src/mbsrobot_project/mbsrobot/rviz to open it navigate.rviz, wait for the window to load the data. The first step of navigation is to initialize the pose. Use "2D Pose Estimate" to drag in the visible area of the map. Remember that dragging is not just dragging, it's your robot's Actual location, drag the area shown on the map, the arrow shows the direction, this is also very important, this means you tell the robot that your positioning on the map is wrong, you should be at the location I gave (Tip: If you don't want to give the positioning pose every time, you can set the starting position of the robot when you create the map. It is the same as when you navigate. Put the robot at this

position, you don't need to estimate the pose), the pose correction is complete. Later, we will use "2D Nav Goal" to send the target. This function is to tell you that you are going to the place I specify and drag it in the visualization area of the map. Then the car will automatically plan the path at this time and avoid the map. Obstacles, reach the destination, and can avoid obstacles dynamically. If you appear in the map area within a certain distance, it can also re-plan the path to bypass your dynamic obstacle:

roslaunch rviz rviz



About problems in navigation

1. Why do I need to turn a few times to get around an obstacle when navigating?

Because the algorithm needs to recalculate after encountering obstacles and re-plan the path through rotation to complete. Of course, if you use a good master control and road navigation algorithm, through the adjustment of the parameters, you can make the car aware in advance, calculate in advance, and plan ahead.

2. Why does my car keep spinning in the aisle or in front of obstacles, like stupid?

This is because your dynamic area is the buffer area of obstacles, such as the blue area in the above picture. This situation is telling you that your current aisle or obstacle is smaller than my set value. I can't get through, but I can't pass. Stopped the rotation and did not find the second one that can reach your given position. The above problem can be optimized by adjusting the parameters of the dynamic area on the software, but the parameter is not adjusted as small as possible. If it is too small, the buffer area is too small. The car may hit an obstacle. Remember, it is a machine and is executed according to algorithm constraints.

3. Why do I navigate multiple times and run for a long time, there will be deviations in the navigation, or the arrival position is not accurate?

This is because our car is composed of hardware, and the hardware has errors. This error will eventually be fed back to the odometer, and the odometer will affect the position the car reaches during the actual movement. If our car has an error of 1m There is 1cm, and this

error will continue to be enlarged through long-term operation. If there is an error in the navigation process, you can also correct it by correcting the pose again, that is, tell the car, hey, your current position is wrong, you should be here.

4. What should I do about some other errors and warnings during the navigation process?

Because the ROS navigation system is very large, you can see that there are dozens of parameters loaded when starting the navigation. The warnings and errors that appear are all caused by unsatisfied conditions. You can use Baidu or Google according to the relevant warnings or errors. .

7 Automatic line inspection based on OpenCV camera

The principle of the ordinary camera to realize the hunting function

The tracking function of the Mbsrobot car can be realized with only a common camera. The version we currently provide is to recognize the red line and then perform the tracking. The specific principle is the color recognition in opencv. The camera performs the color recognition in the picture. Recognize, and then keep the recognized color area in the center of the screen and continuously give the speed value of the car to keep the car along the red line.

The use effect is not a big problem in a normal straight line, but when turning, it is easy to lose sight. If you get the car and use it, you can continue to optimize the algorithm on this basis.

Ordinary camera hunting function use

1. Power on and start the car, and the main control terminal executes the command to start the car:

roslaunch mbsrobot bringup.launch



2、Command to start the camera on the host:

roslaunch mbsrobot camera.launch

3、 This starts the camera hunting command at the remote end. Since we need to see the image output during hunting, we need to enter this command at the remote end. At this time, the car will start walking on the recognized track in the recognized color patch area:

roslaunch mbs_line_follower mbs_line.launch

Note on the use of ordinary camera hunting

1. Since the camera hunting line is the result of the color recognition of the collected image, if the image collected by the camera is of poor quality or strong light, this will affect the line hunting
2. The height of the camera and the angle of collection will also have an impact on the quality of the line hunting. You can adjust the specific suitable height and angle during use, that is, the more areas of the image area to identify the line, The effect will be better. The environment we use uses a two-drive trolley with a depth camera. At this height, the real shot effect is just like that.
3. If users want to identify lines of other colors, they can change the color value of the recognized color in the code.