

###写在前面###

通过前面一系列的铺垫，相信大家对整个 miiboo 机器人的 DIY 有了一个清晰整体的认识。接下来就正式进入机器人大脑（嵌入式主板：树莓派 3）的开发。本章将从树莓派 3 的开发环境搭建入手，为后续 ros 开发、slam 导航及语音交互算法做准备。本章内容：

- 1.安装系统 ubuntu_mate_16.04
- 2.安装 ros-kinetic
- 3.装机后一些实用软件安装和系统设置
- 4.PC 端与 robot 端 ROS 网络通信
- 5.Android 手机端与 robot 端 ROS 网络通信
- 6.树莓派 USB 与 tty 串口号绑定
- 7.开机自启动 ROS 节点

###正文###

1.安装系统 ubuntu_mate_16.04

安装前先准备好需要用到的材料，在树莓派 3 上安装 ubuntu_mate_16.04 需要用到的工具和材料，如图 1。

● 树莓派 3B 主板



● 32GB 容量 microSD 卡（根据自己的需求选择卡的容量，一般要不小于 16GB）



● 鼠标键盘和 HDMI 显示器



（图 1）材料准备

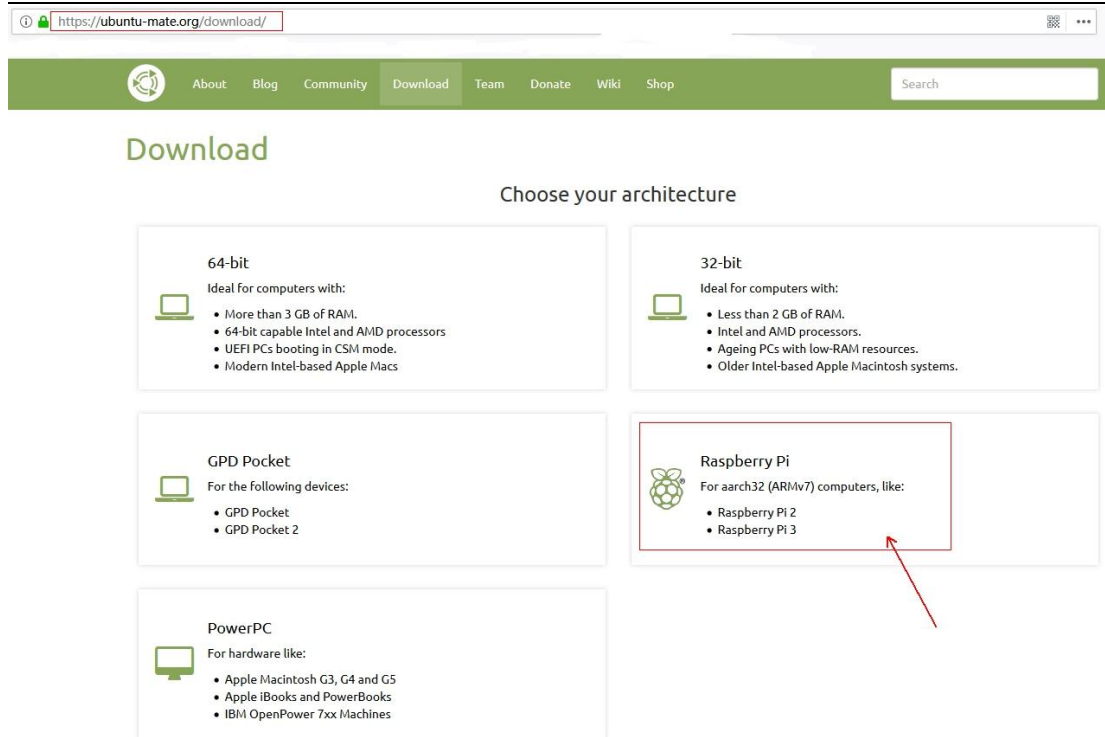
1.1.格式化 microSD 卡

在向 microSD 卡烧入系统之前，需要先格式化好 microSD 卡。我这里使用 DiskGenius 工具将卡格式化为 FAT32 文件系统。DiskGenius 下载地址：

<http://www.diskgenius.cn/download.php>

1.2.下载 ubuntu-mate-16.04 系统镜像

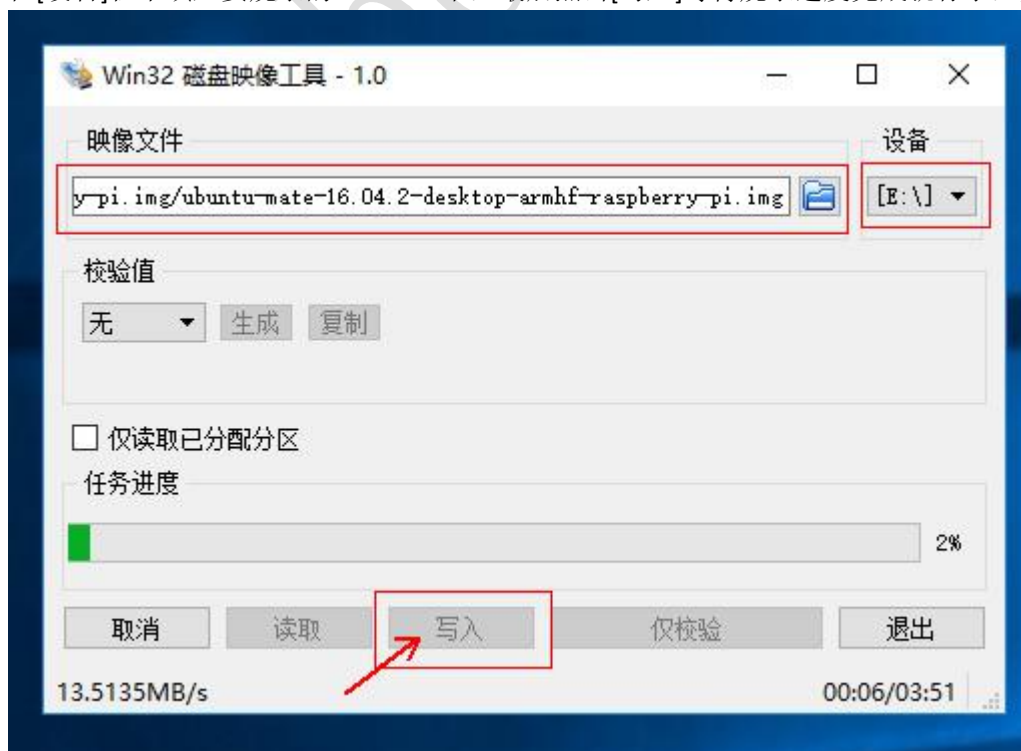
直接前往 ubuntu-mate 的官网 <https://ubuntu-mate.org/download/>。选择如图 2 所示的版本进行下载就行了。



(图 2) ubuntu-mate-16.04 下载页面

1.3.系统烧录

将下载好的系统镜像文件 `ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img.xz` 解压得到 `ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img`，该文件应存放在英文路径下。然后用 Win32 Disk Imager 工具将 `ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img` 烧录到 microSD 卡，Win32 Disk Imager 工具下载地址 <https://win32-disk-imager.en.lo4d.com/download>。烧录过程很简单，打开 Win32 Disk Imager 工具，在[映像文件]栏中填入待烧录的镜像文件路径，在[设备]栏中填入要烧录的 microSD 卡，最后点击[写入]等待烧录进度完成就行了，如图 3。



(图 3) 系统烧录

1.4.上电开机

给树莓派 3 主板连接上 HDMI 显示器、鼠标、键盘，并插入刚刚烧录好系统的 microSD 卡，就可以上电了，如图 4。



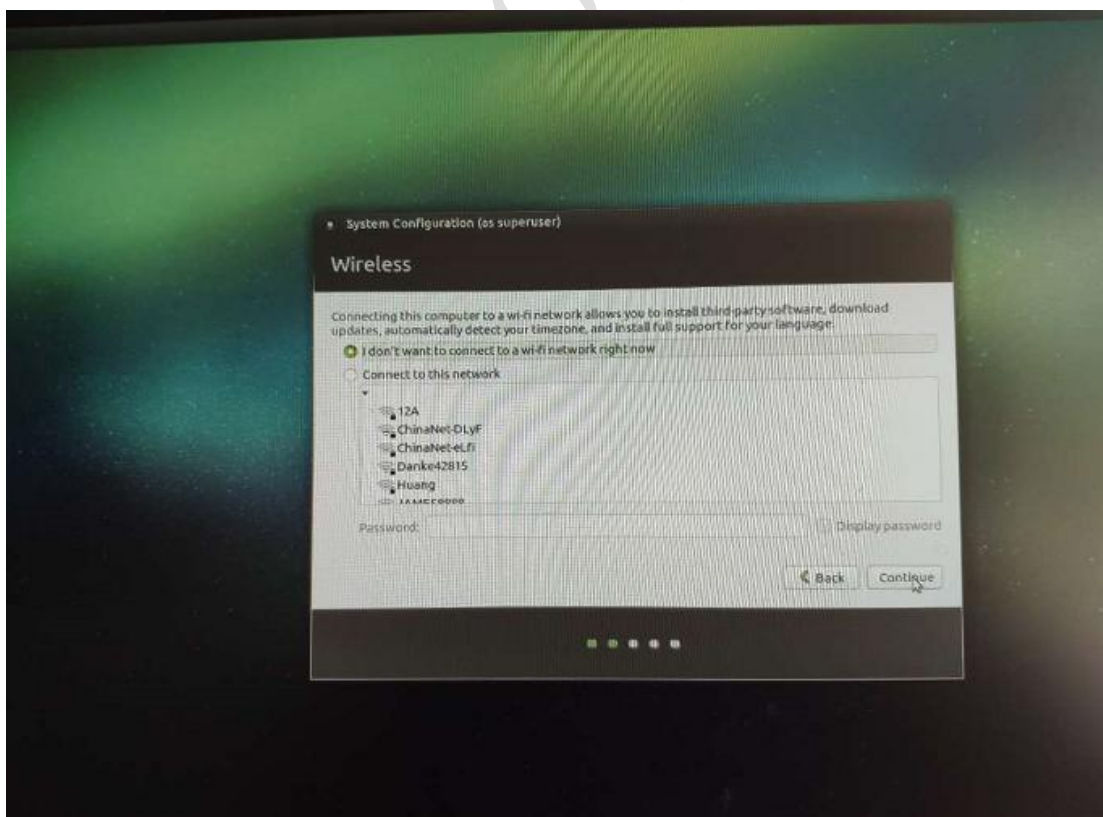
(图 4) 上电开机

第一次开机，系统需要用户填写一些必要的设置项，首先是系统语言设置，选择默认的语言 English 就行了，如图 5。



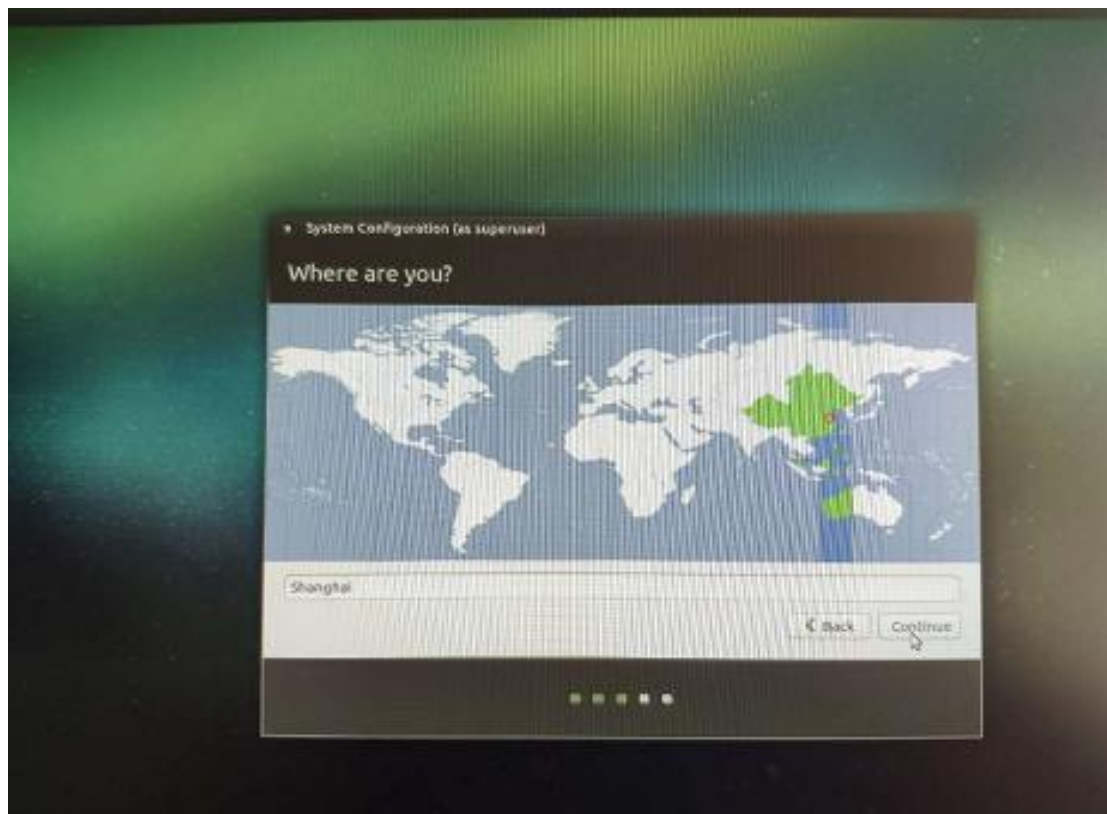
(图 5) 语言设置

然后是 wifi 连接设置，这里选择先不联网，这样系统配置速度会快很多，等后面我们再进行联网，如图 6。



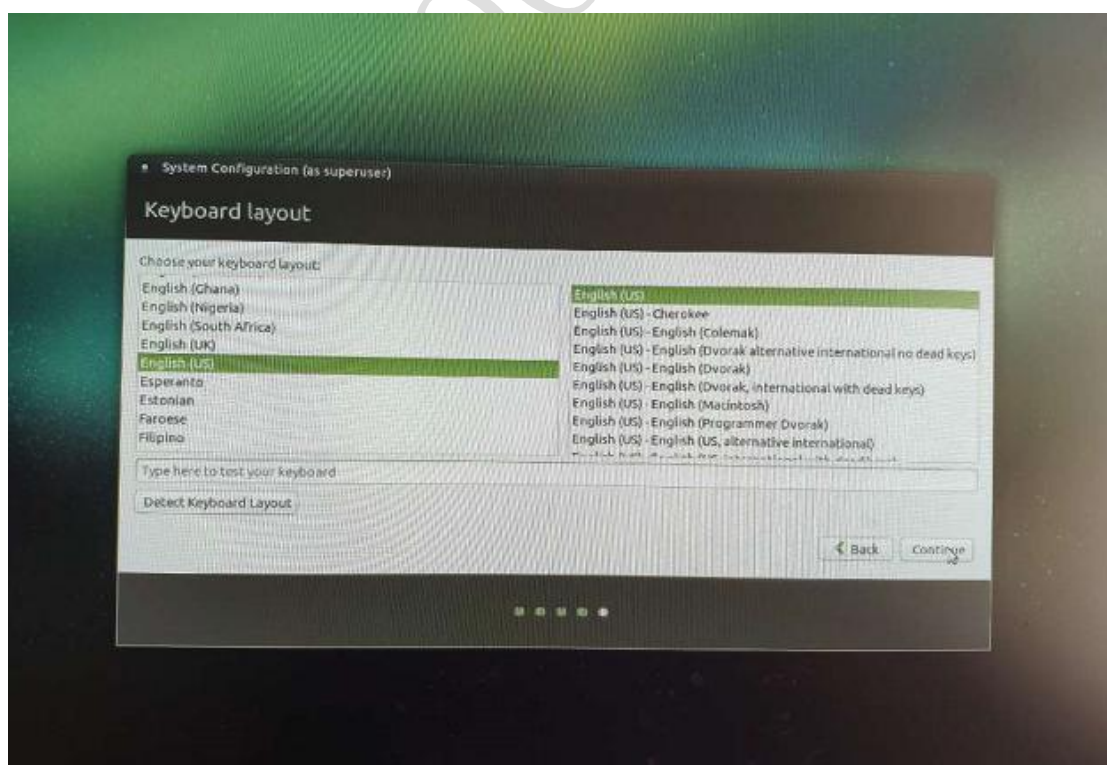
(图 6) wifi 连接设置

然后是时区设置，我们在中国，说以用鼠标点击地图中的中国区域，会自动锁定到 Shanghai 时区，如图 7。



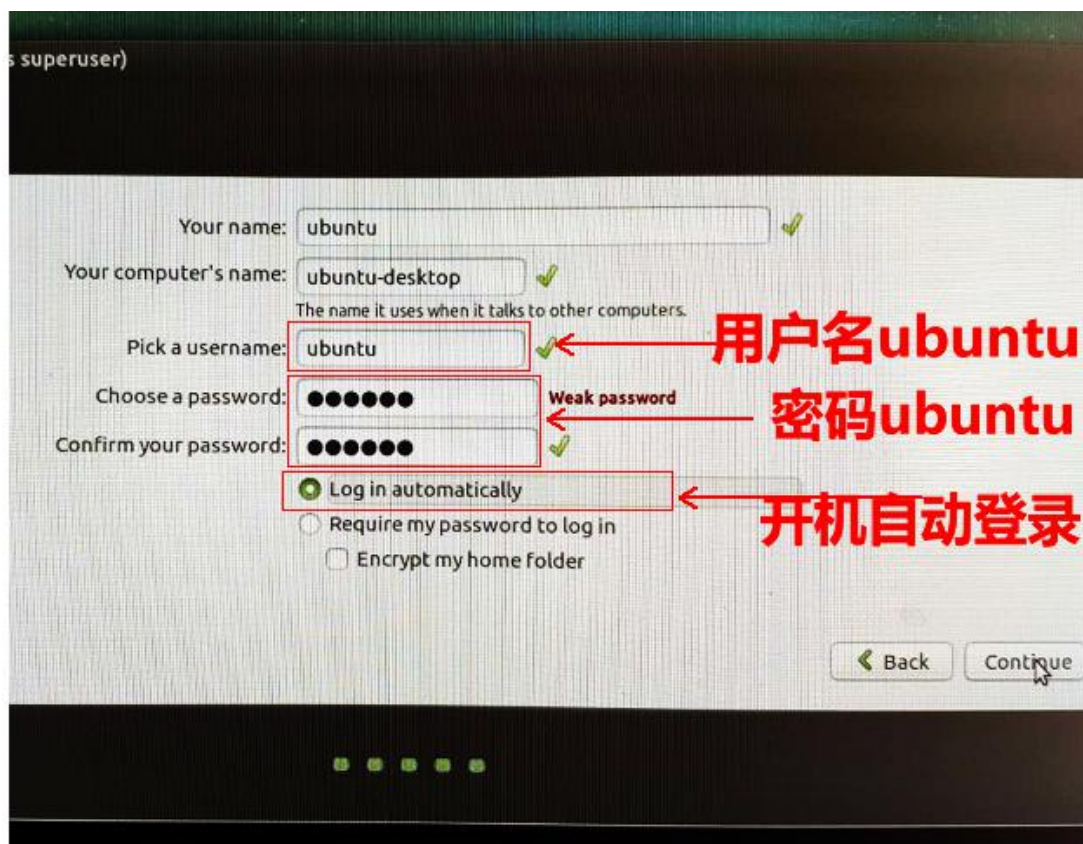
(图 7) 时区设置

然后是键盘设置，直接默认就行了，如图 8。

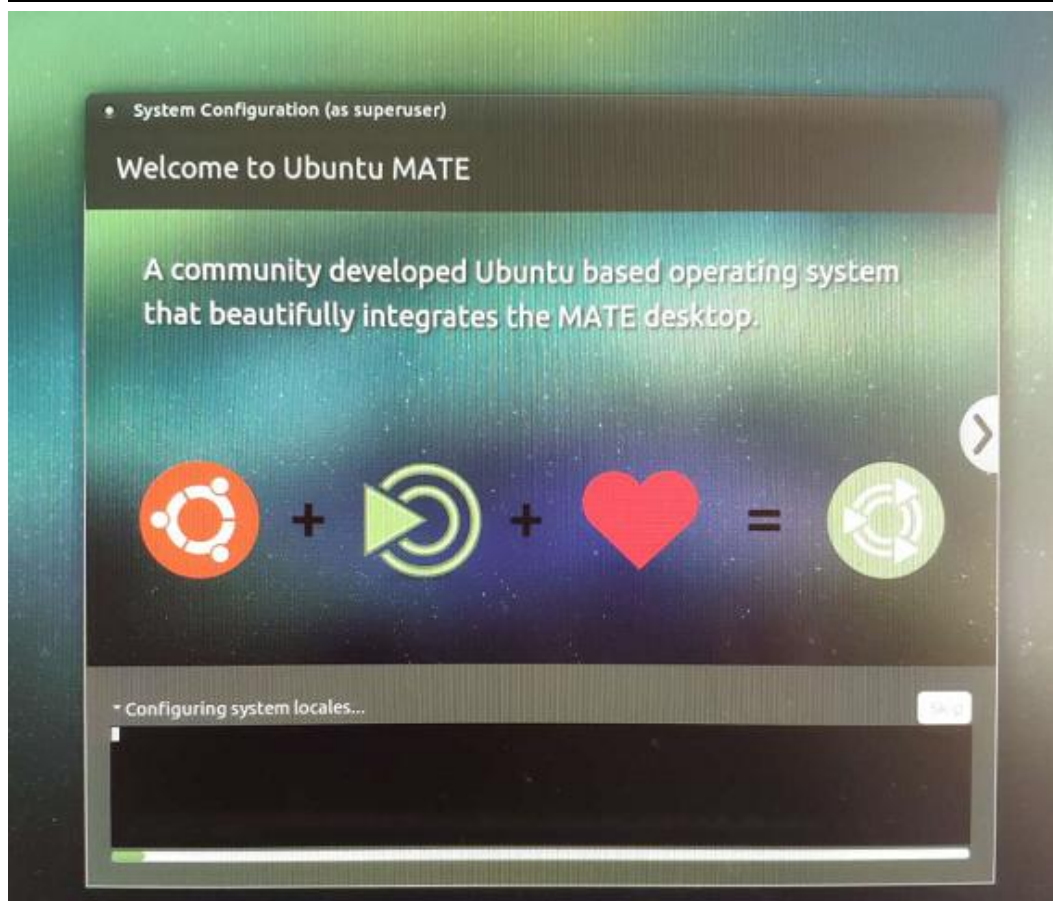


(图 8) 键盘设置

然后是用户名和密码设置，为了方便记忆，我将用户名和密码都设置成了 ubuntu；这里需要特别注意，由于我们的系统用于机器人的开发，为了让机器人上电就能自动进入系统，我们需要勾选[Log in automatically]选项，也就是让系统开机自动登录。如图 9。



（图 9）用户名、密码、开机自动登录设置
然后就进入系统配置过程了，耐心等待配置进度条完成，如图 10。



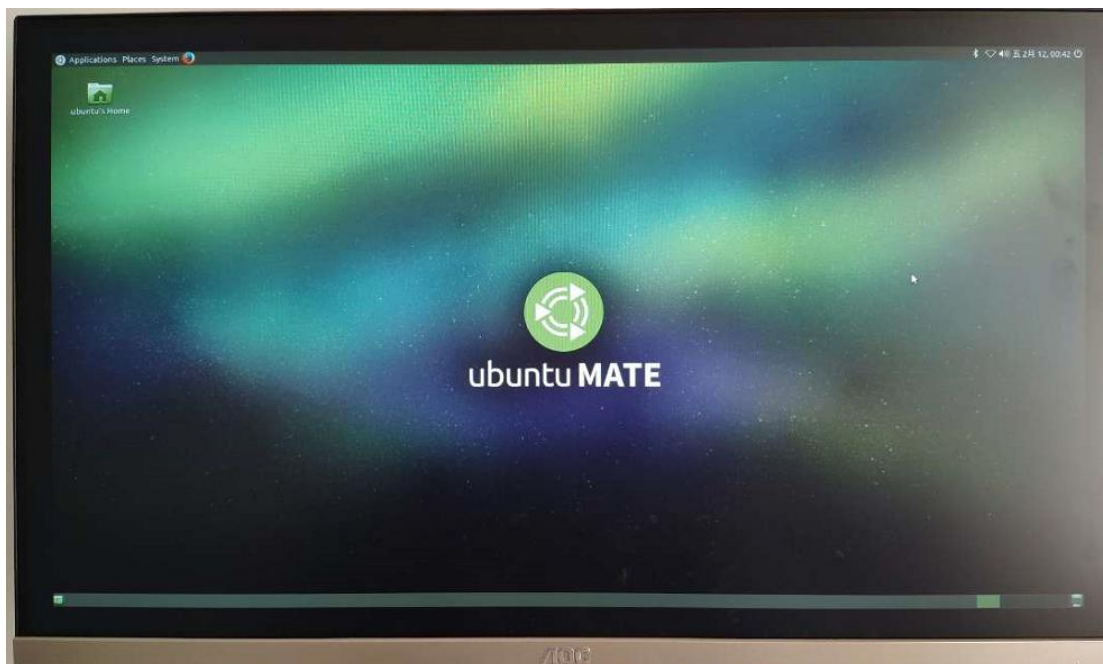
(图 10) 系统配置过程

耐心等待，所有配置完成后，系统会自动重启一次，重启完成后，就可以看到 ubuntu-mate-16.04 系统的真容了，如图 11。如果我们不想每次开机都看到这个欢迎界面，可以去掉勾选框中的勾，关闭就行了，下次就不会出现了。



(图 11) 欢迎界面

最后，就可以见到 ubuntu-mate-16.04 系统的真容了，如图 12，到这里系统安装就成功了。



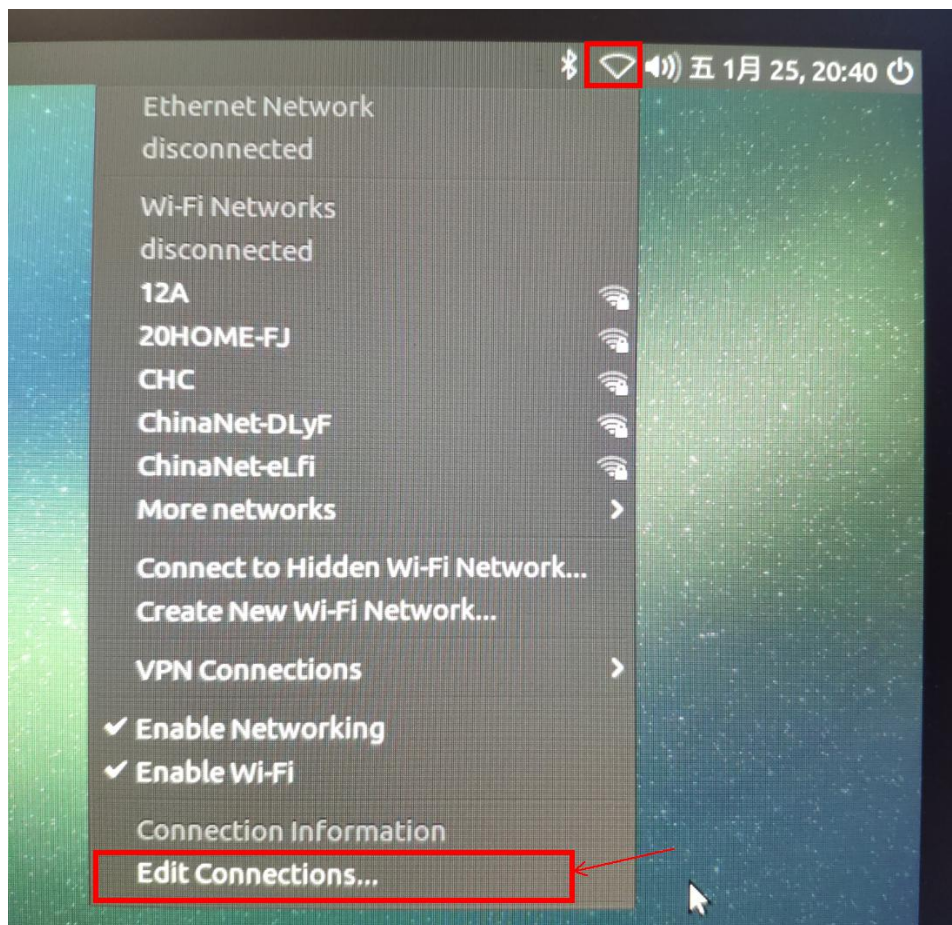
(图 12) ubuntu-mate-16.04 系统桌面

2. 安装 ros-kinetic

安装好系统 ubuntu-mate-16.04 后，就可以安装对于我们机器人开发非常重要的 ROS 系统了，根据 ubuntu 的版本与 ROS 行版本之间的对应关系，这里我们安装 ros-kinetic 这个发行版。

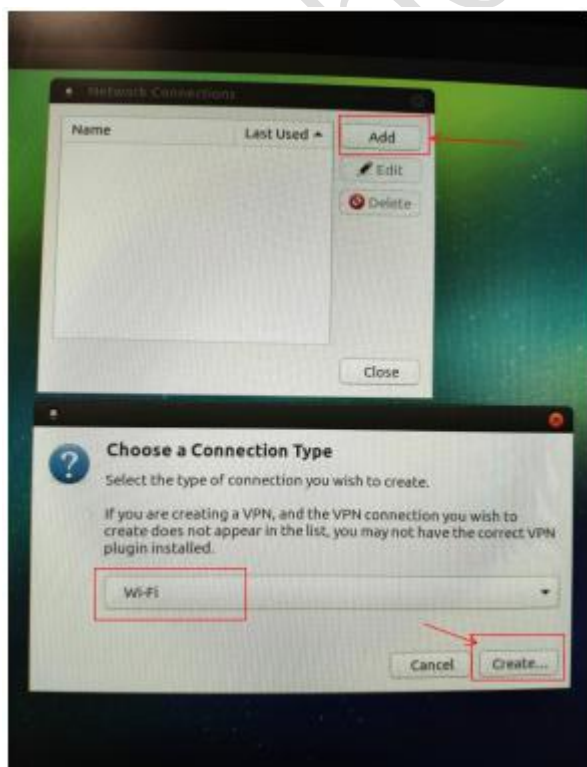
2.1. 连接 wifi 网络

由于安装需要通过网络在线进行，所以需要先连接上 wifi，在桌面右上角找到 wifi 样式的图标，在下拉栏中选择[Edit Connections...], 如图 13。



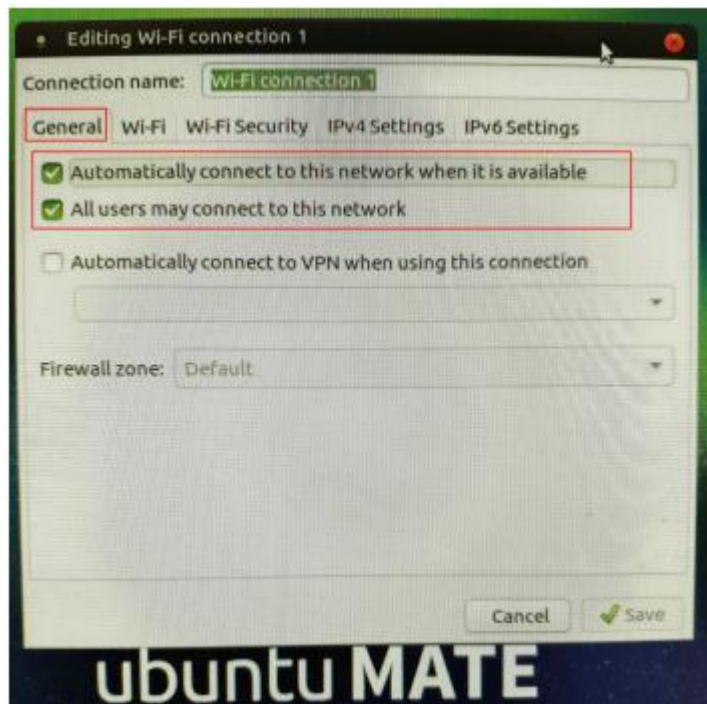
(图 13) 编辑网络连接

在弹出来的窗口中，点击[Add]来添加我们的网络连接，并选择[Wi-Fi]连接类型，最后点击[Create...]来创建，如图 14。



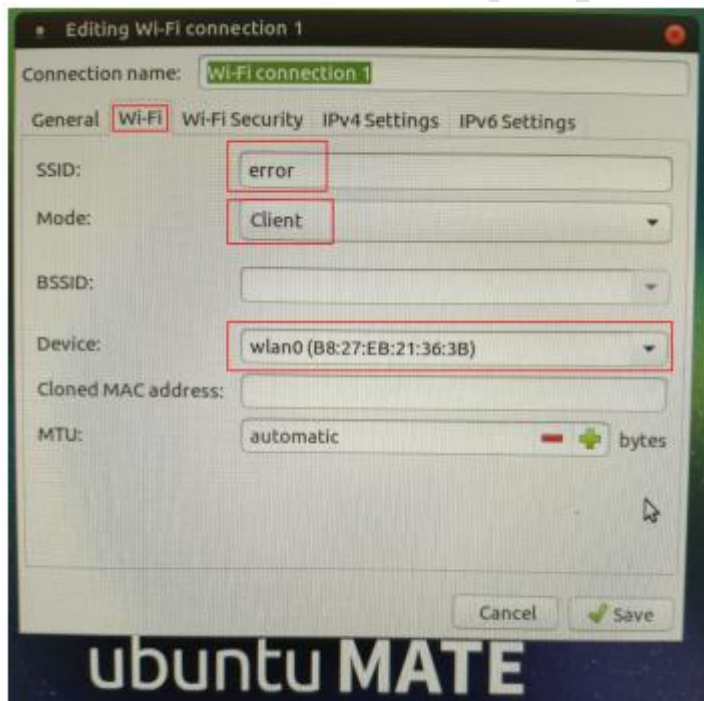
(图 14) 创建网络连接

在网络连接的配置窗口中，首先设置[General]参数，为了让系统开机自启动后能自动连接该网络，勾选前两项，如图 15。



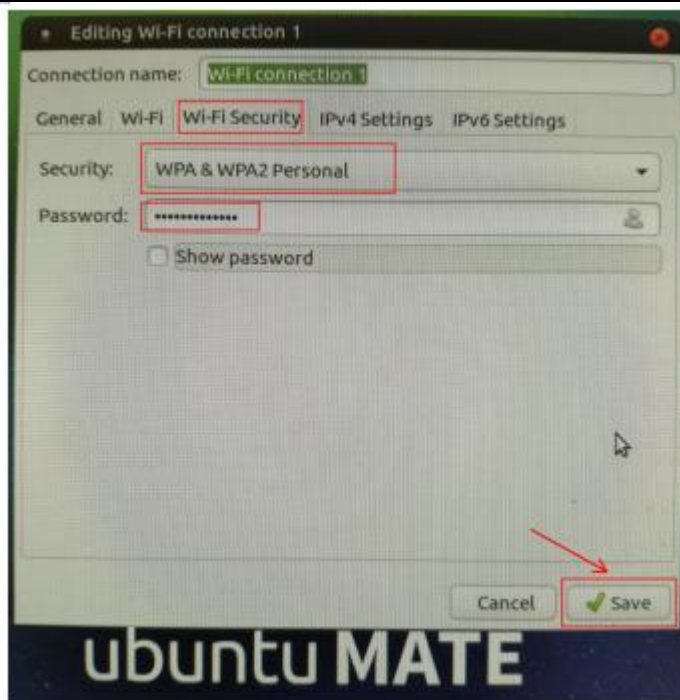
(图 15) 网络连接 General 参数

然后设置[Wi-Fi]参数，SSID 栏填入自己家 wifi 的名字（我家的 wifi 名字叫 error），Mode 栏选择 Client，Device 栏下拉填入自己的无线网卡号，如图 16。



(图 16) 网络连接 Wi-Fi 参数

然后设置[Wi-Fi Security]参数，Security 栏填入自己家 wifi 的加密方式（一般的加密方式都是 WPA&WPA2 Personal），Password 栏填入 wifi 的连接密码，最后点击[save]保存设置好的参数，如图 17。



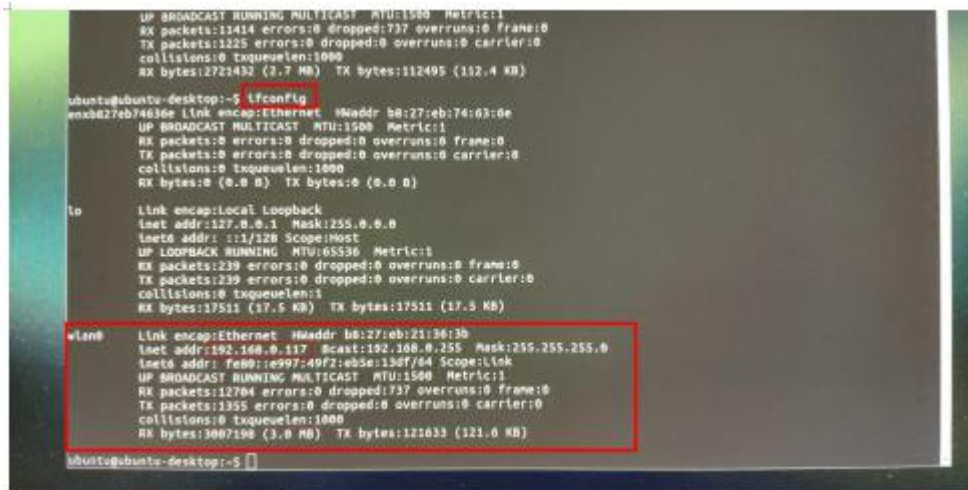
(图 17) 网络连接 Wi-Fi Security 参数

最后找到[Enable Networking]选项，去掉选项前面的勾，然后再勾上，让网络配置生效；稍等片刻，我们配置的 wifi 连接 error 将自动连接上，这样就连接网络完成了，如图 18。



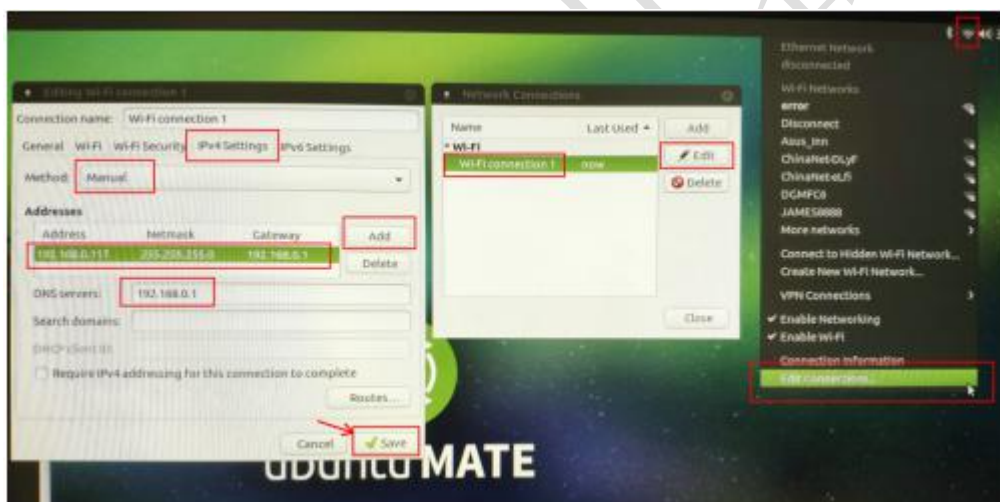
(图 18) 重启网络使网路配置生效

为了让机器人启动后，我们能通过网络远程访问，这里建议大家在树莓派上设置成静态 IP 分配，首先打开命令行终端用 `ifconfig` 查看一下当前分配给树莓派的 IP 地址，将这个 IP 地址绑定成静态 IP,这样可以保证该 IP 不会与其他设备发生 IP 冲突，如图 19。



(图 19) 查看 IP 地址

然后，对上面创建的网络连接的[IPv4 Settings]参数进行设置，Method 栏选择 Manual，Addresses 栏填入静态 IP 参数（静态 IP 设为 192.168.0.117，子网掩码设为 255.255.255.0，网关设为 192.168.0.1），DNS servers 栏填入 192.168.0.1，最后保存就行了，只需要重启网络静态 IP 就设好了，如图 20。



(图 20) 设置静态 IP

2.2.安装 ros-kinetic

其实在 ubuntu 上安装 ROS，有很详细的 ROS 官方教程，感兴趣的朋友可以直接参考官方教程 <http://wiki.ros.org/kinetic/Installation/Ubuntu>。由于官方教程用英文书写，为了方便大家阅读，我将官方教程翻译过来，方便大家学习，下面正式进入安装。温馨提醒，由于不同的编辑器对过长的句子换行的规则不同，下面的命令被自动换行后可能影响正常的阅读，请直接参阅官方教程中的命令 <http://wiki.ros.org/kinetic/Installation/Ubuntu>。

(1) 设置 ubuntu 的 sources.list

打开命令行终端，输入如下命令：

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```


(2) 设置 keys

打开命令行终端，输入如下命令：

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80
--recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

(3) 安装 ros-kinetic-desktop-full 完整版

打开命令行终端，分别输入如下两条命令：

```
sudo apt-get update

sudo apt-get install ros-kinetic-desktop-full
```

小技巧，如果安装过程提示“下载错误”，请耐心等待上面的两条命令，这个错误多半是由于网络故障造成的。

(4) 初始化 rosdep

打开命令行终端，分别输入如下两条命令：

```
sudo rosdep init

rosdep update
```

(5) 配置环境变量

打开命令行终端，分别输入如下两条命令：

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

(6) 安装 rosinstall

打开命令行终端，输入如下命令：

```
sudo apt install python-roinstall python-roinstall-generator
python-wstool build-essential
```

(7) 测试 ros 安装成功与否

打开命令行终端，输入如下命令：

```
roscore
```

如果此时出现以下内容

```
setting /run_id to 4cb2a932-06c0-11e9-9ff2-000c2985f3ab
process[rosout-1]: started with pid [38686]
started core service [/rosout]
```

那么恭喜你，ROS 已经成功的安装上了！！！！

3. 装机后一些实用软件安装和系统设置

虽然我们顺利安装了 ubuntu-mate-16.04 系统，并且在系统上装好了 ROS-kinetic 发行版，为

了后面跟高效和便捷的开发，这里将进行一些非常实用软件的安装和系统设置。

3.1. 开机自动登录

装完系统后第一次开机，会要求设置用户名和密码，这里应该设置用户开机自动登录，只有登录后树莓派 3 才能自动连接到无线网络，从而才能远程控制。如果装机时已经设置了，请直接跳过这一部分；如果你搞忘记或遗漏了，没有设置开机自动登录，也不要紧，用下面的方法进行自动登录设置。

打开文件 `/usr/share/lightdm/lightdm.conf.d/60-lightdm-gtk-greet.conf`，在 `[seat:*)` 节点下修改或添加以下代码

```
autologin-user=ubuntu #ubuntu 是要自动登录的用户名
```

3.2. 超级用户 root 密码设置

装完 **ubuntu-mate-16.04** 系统后，**root** 用户默认是没有密码的；用普通用户 **ubuntu** 的 **sudo** 权限去设置 **root** 用户的密码。

反正我们是用于学习，出于方便记忆的考虑，所以和之前设置 **ubuntu** 的用户名和密码一样，这里设置 **root** 的用户名和密码也一样，也就是 **root** 用户的密码也是 **root**。

```
ubuntu@ubuntu-desktop:~$ sudo passwd root
[sudo] password for ubuntu:      #键入 ubuntu 密码
Enter new UNIX password:         #键入 root 新密码
Retype new UNIX password:       #确认 root 新密码
passwd: password updated successfully
```

root 密码设置成功后，**su root** 登录进去验证一下，如果能登录就设置成功了。

3.3. 扩展 SWAP 空间

由于后续在树莓派 3 上需要编译一些大型的程序和运行复杂的 **SLAM** 算法，默认的物理内存 **1GB** 是不够用的，常常造成卡机死机等现象，这里需要添加 **SWAP** 扩展空间，其实就是当系统内存不足时会用硬盘上 **SWAP** 分区作为虚拟内存。

由于我的 **microSD** 卡是 **32GB** 的，所以可以多分配一些给 **SWAP**，所以分配 **4GB** 给 **SWAP** 空间。具体步骤：

```
#先关闭 swap
cd /var
sudo swapoff /var/swap

#重设 swap 大小 1M*4096=4GB,会花较长时间,请耐心等待
sudo dd if=/dev/zero of=swap bs=1M count=4096

#格式化
sudo mkswap /var/swap

#开启 swap
sudo swapon /var/swap

#设置开机启动,在/etc/fstab 文件中添加如下代码
/var/swap swap swap defaults 0 0
#查看当前已生效的 swap
swapon -s
#查看当前 swap 使用情况
free -m
```

3.4.连接 wifi 网络

为了让机器人启动后,我们能通过网络远程访问,需要给机器人配置好网络连接,并分配静态 IP,让机器人开机后就能自动连接 wifi。关于设置的内容已经在上面安装 ros-kinetic 详细介绍了,有需要的直接前去翻阅就行。

3.5.安装 chromium 浏览器

前一段时间,在 ubuntu-mate-16.04 上安装 ROS 后,ubuntu-mate-16.04 系统自带的 Firefox 浏览器不能用了,Firefox 最近对这个故障进行了修复。不过为了保险起见,还是推荐安装谷歌为 linux 系统推出的浏览器 chromium,跟我们平时用的闻名于世的 chrome 浏览器是差不多的。安装很简单,就一条命令。

```
sudo apt-get install chromium-browser
```

3.6.安装 vim、ssh 工具

vim 是 Linux 中一款很经典的代码文本编辑器,不论是本地还是远程编辑文本都很强大;ssh 功能很强大,不紧可以提供用户远程登录控制系统和编辑文件,还提供命令终端下的 scp 远程文件传输。安装很简单,几条命令就搞定。

```
#安装 vim
sudo apt-get install vim
```

```
#安装 ssh 服务端
sudo apt-get install openssh-server
#开启 ssh 服务
sudo service ssh start
#设置开机自启动 ssh 服务，在开机自启动脚本/etc/rc.local 加入下面这条命令，
#注意加入位置是在 exit 0 这一句之前
service ssh start
#检查 ssh 是否启动，使用下面命令
ps -e | grep ssh
如果输出中含有 ssh-agent 和 sshd 就说明成功了
```

3.7.通过 ssh 远程登录到机器人

树莓派 3 主板是安装在 miiboo 机器人上的，由于机器人要在地面上自由移动，给机器人上的树莓派配一个 HDMI 显示器和鼠标键盘用于程序调试显然是不方便的。一个很好的方法是，工作端 PC 和机器人连接到同一个路由器完成局域网组网，然后在 PC 端用 ssh 远程登录到机器人，这样就可以在工作 PC 端对机器人上的程序和文件进行调试开发了。

ssh 远程登录方法：

```
#在 PC 端上打开终端，ssh 远程登录机器人，
#ubuntu 为登录用户名，192.168.0.117 为登录机器的 IP，根据实际替换
ssh ubuntu@192.168.0.117

#退出 ssh 远程登录
exit
```

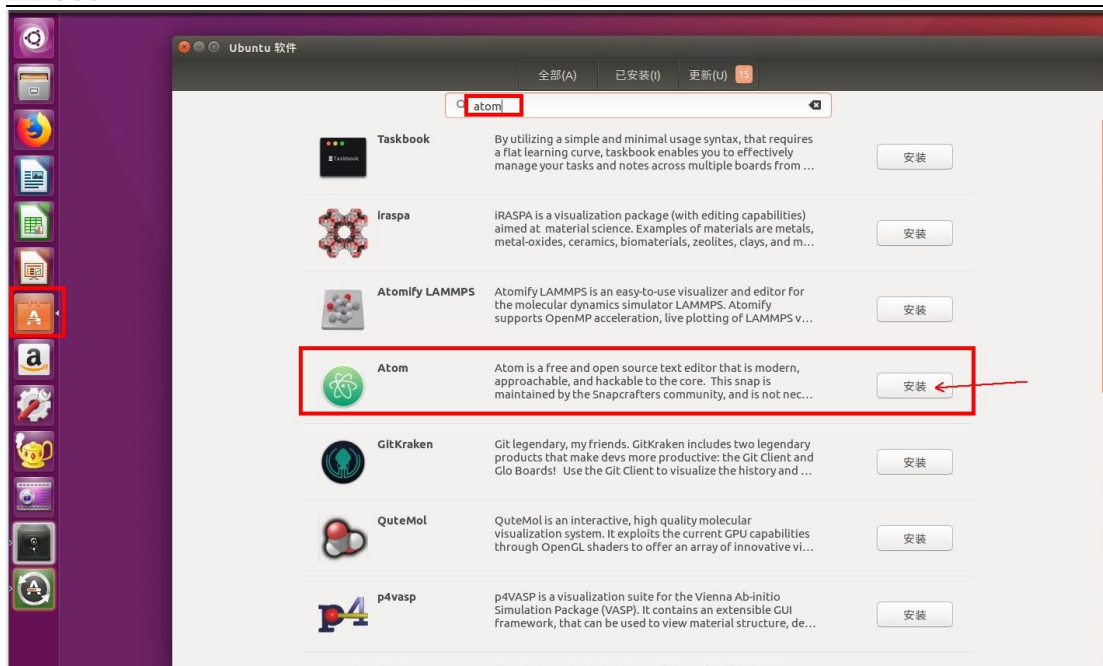
一旦 PC 端打开的终端窗口 ssh 远程登录机器人后，就可以在该终端下像在机器人上使用终端一样进行操作了。PC 端可以打开多个终端窗口 ssh 远程登录机器人，这样在不同的终端下可以开启机器人上不同的程序。

3.8.在工作 PC 端远程编辑机器人上的文件

已经介绍了通过 ssh 远程登录机器人的方法，其实可以直接在登入的终端下用 vim 对机器人上的文件进行查看编辑。但是，如果需要同时编辑多个文件，或对一个软件项目源码进行编辑，vim 就不方便了。

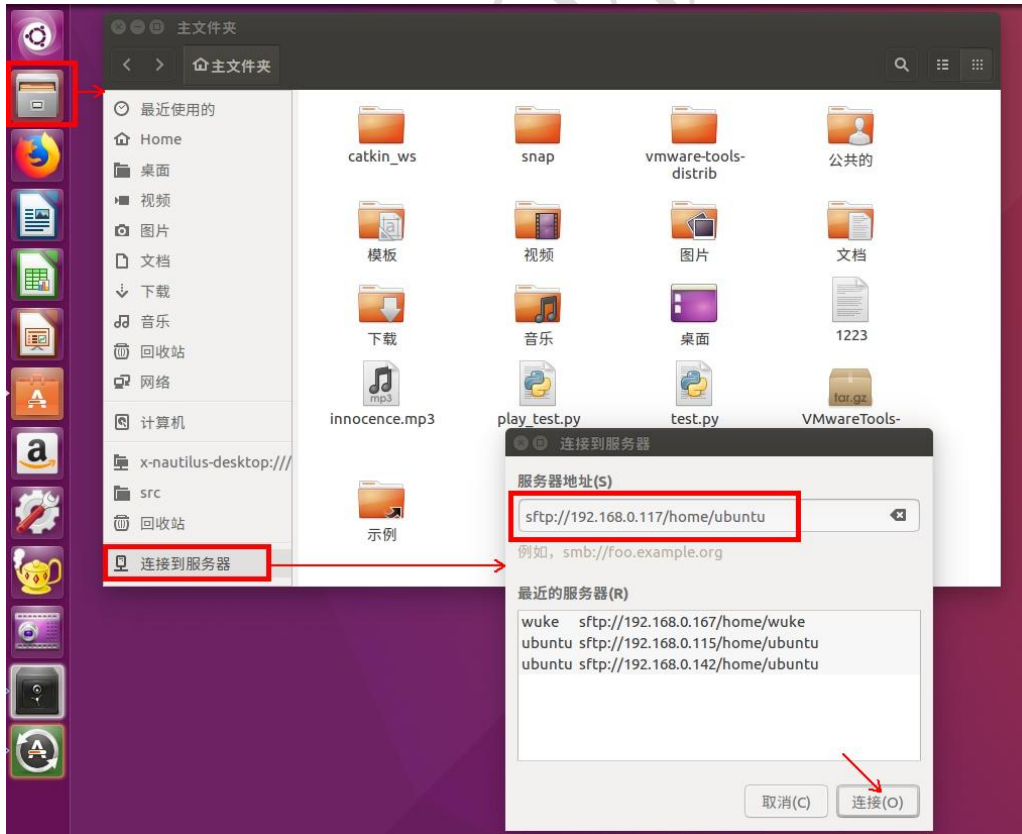
这里推荐在 PC 端安装 IDE 开发工具（比较推荐 atom，不过 sublime text 也不错，根据个人喜好选择吧），然后将机器人上的系统文件夹远程挂载在 PC 端系统，这样就可以在 PC 端上用 IDE 开发工具直接对远程挂载过来的文件及文件夹进行编辑了。

在 PC 端安装 IDE 开发工具 atom 很简单，直接在 ubuntu 软件中搜索 atom，然后点击安装就行了，如图 21。



(图 21) 在 PC 端安装 IDE 开发工具 atom

将机器人上的系统文件夹远程挂载在 PC 端系统，首先打开[文件]管理器，找到左侧栏的[连接到服务器]，在服务器地址栏输入远程文件地址（如 `sftp://192.168.0.117/home/ubuntu`，这里的 `sftp` 是协议，`192.168.0.117` 是远程设备 IP 地址，`/home/ubuntu` 是远程设备中想要被挂载的文件夹请根据实际替换），然后点击[连接]，这是需要输入远程设备的登录名和密码，如图 22。



(图 22) 将机器人上的系统文件夹远程挂载在 PC 端系统

挂载好后，就可以使用各种我们喜欢的编辑工具对文件夹中的文件进行编辑了，如图 23，不过远程文件系统挂载会比较占用网路带宽，不用时建议尽量断开。



(图 23) 挂载好后的远程文件夹

4. PC 端与 robot 端 ROS 网络通信

PC 端与 robot 端 ROS 网络通信的设置，其实很简单。就是要在 PC 端和机器人端都要指明 master 和 hostname 这两个东西，我们都知道 ROS 基于的是一个分布式的通信机制，支持多机通信，并且只能有一个 master 设备作为对其他分部式设备的统一管理。

由于大部分节点算法都是运行在机器人上，PC 端只是用于远程调试，为了系统的效率与稳定性，我把机器人端作为 master。如图 24。



(图 24) PC 端与 robot 端 ROS 网络通信

首先，配置机器人端的 ROS 网络参数，打开机器人用户目录下的 ~/.bashrc 配置文件，在最后添加两个环境变量，如图 25。

```

ubuntu@ubuntu-desktop: ~
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${ $? = 0 } && echo terminal || echo
error)" "${history|tail -n1|sed -e '\''s/^\\s*[0-9]\\+\\s*//;s/[:&|]\\s*alert$//'\''
}'"
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

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/kinetic/setup.bash

export ROS_MASTER_URI=http://192.168.0.117:11311
export ROS_HOSTNAME=192.168.0.117

```

机器人上~/.bashrc配置

指定master的IP

指定本机IP

98,0-1 Bot

(图 25) 配置机器人端的 ROS 网络参数

不难发现在机器人上 mater 和 host 的 IP 是一样的，因为机器人被指定为了整个 ROS 网络的 master 设备。修改好后，需要重启系统使设置参数生效。

然后，配置 PC 端的 ROS 网络参数。这里需要提醒一下，需要让 PC 端与机器人出于同一局域网下，如果 PC 端使用的是虚拟机运行 ubuntu 建议用物理桥接的方式给虚拟机连网。打开 PC 端用户目录下的 ~/.bashrc 配置文件，在最后添加两个环境变量，如图 26。

```

ubuntu64@ubuntu64-virtual-machine: ~
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

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/kinetic/setup.bash

export ROS_MASTER_URI=http://192.168.0.117:11311
export ROS_HOSTNAME=192.168.0.113

```

PC端~/.bashrc配置

指定master的IP

指定本机IP

104,1 底端

(图 26) 配置 PC 端的 ROS 网络参数

不难发现 PC 端上 master 填的是机器人 IP 地址, host 为 PC 本地 IP。修改好后, 需要重启系统使设置参数生效。

设置好 PC 端与 robot 端 ROS 网络通信, 在正是进行 ROS 多机通信时, 需要首先在机器人上启动节点管理里 master, 启动命令是 `roscore`, 然后就可以在相应的机器上启动需要的节点了。

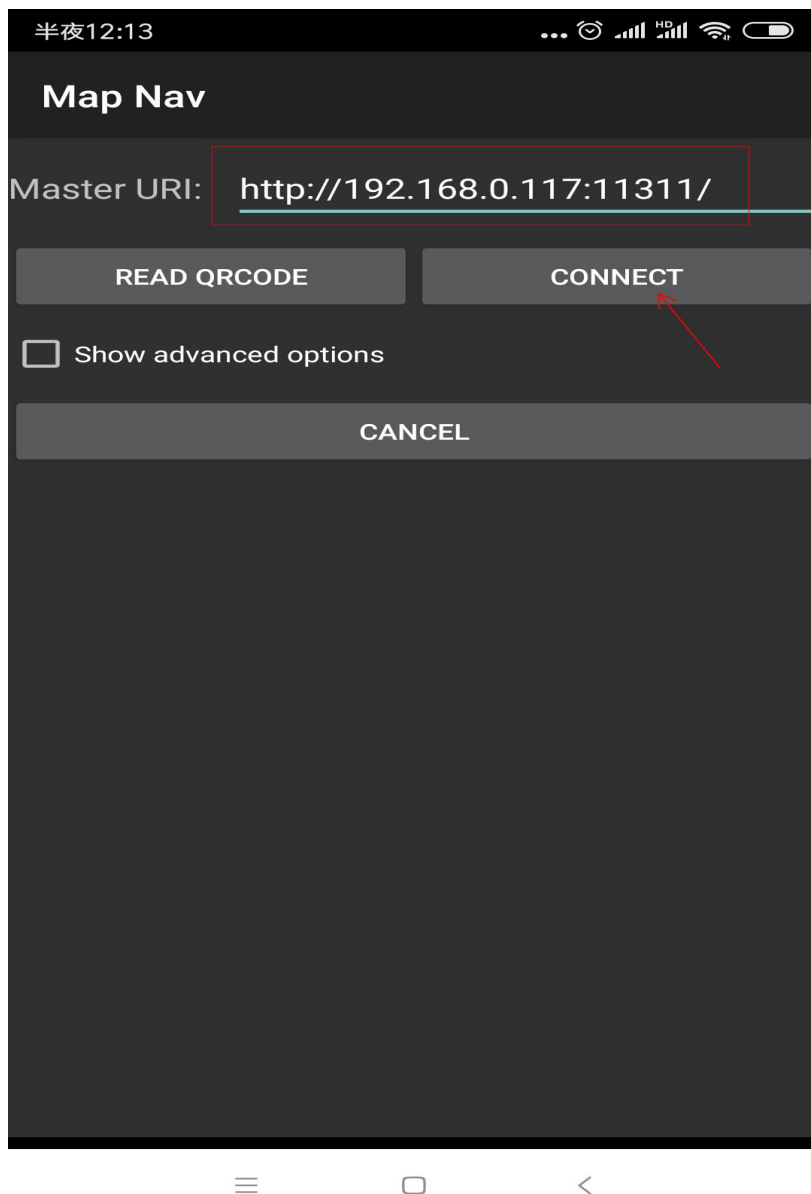
5.Android 手机端与 robot 端 ROS 网络通信

刚刚介绍了 PC 端与 robot 端 ROS 网络通信, 在有些情况下, 使用 Android 手机端来调试监控机器人会更方便。于是, 参考 ROS 官网给的开发 demo, 我用 `ros-java` 库也做了一个能跟 robot 端进行 ROS 网络通信的 APP, 我给这个 APP 取名叫 `Android_for_miiboo_robot.apk`。如果大家对这个 APP 感兴趣, 可以持续关注我, 我会把这个 APP 共享给大家。这里就来介绍一下 Android 手机端与 robot 端 ROS 网络通信的配置。

首先, 是配置机器人端的 ROS 网络参数, 和前面一样, 需要在机器人端 `~/.bashrc` 中指定 master 与 host, 由于前面配置 PC 端与 robot 端 ROS 网络通信时已经配好了, 所以就无需重复配置了。为了便于在局域网下的 Android 手机通过网络能迅速找到机器人, 需要在机器人上开启专门广播自己 IP 的节点, 这样 Android 手机就能自动搜索并完成 Android 手机端与 robot 端 ROS 网络通信的连接。机器人 IP 广播节点我已经写好了, 只需要通过命令启动:

```
roslaunch broadcast_ip broadcast_udp.launch
```

然后, 是配置 Android 手机端的 ROS 网络参数, 其实将 Android 手机连接到同一个 wifi 后, 打开 miiboo 机器人的 APP, APP 会自动扫描到 master (也就是机器人) 的 IP 地址, 只需要点击 CONNECT 就可完成连接。连接完成后, Android 手机端与 robot 端 ROS 网络通信就打通了, 接下来 APP 就可以用 ROS 网络通信来操控机器人了。如图 27。



(图 27) 配置 Android 手机端的 ROS 网络参数

这里顺便介绍一下，miiboo 机器人 APP 的功能，功能清单如下。

功能 1: 手动遥控 miiboo 机器人移动

功能 2: 建图/导航模式切换

功能 3: 显示地图

功能 4: 点击地图点指定导航

功能 5: 视频监控

6. 树莓派 USB 与 tty 串口号绑定

底盘、激光雷达、IMU 这三个传感器都使用串口与树莓派通信，为了防止每次开机这三个设备的串口号发生变动，需要将串口号进行绑定与重映射。

创建 rules 文件:

rules 文件前的序号越大优先级越小，将优先级设置的小一点；创建文件 `/etc/udev/rules.d/99-miiboo-usb-serial.rules`，文件内容如图 28。

```
#miiboo
KERNELS=="1-1.3", ATTRS{idProduct}=="7523", ATTRS{idVendor}=="1a86", SYMLINK+="miiboo", MODE="0777"

#lidar
KERNELS=="1-1.4", ATTRS{idProduct}=="ea60", ATTRS{idVendor}=="10c4", SYMLINK+="lidar", MODE="0777"

#imu
KERNELS=="1-1.5", ATTRS{idProduct}=="ea60", ATTRS{idVendor}=="10c4", SYMLINK+="imu", MODE="0777"
```

(图 28) /etc/udev/rules.d/99-miiboo-usb-serial.rules 文件内容

确定新插入串口设备的属性:

```
#将<devpath>替换成新插入串口设备号, 如/dev/ttyUSB0
udevadm info -a -p $(udevadm info -q path -n <devpath>)
```

在输出的数据中从上到下找 (如 KERNELS=="1-1.4.3:1.0"形式的项), 下一个不带 ":" 的 KERNELS 就是我们要找的, 将下面这些参数

ATTRS{idProduct}

ATTRS{idVendor}

KERNELS

的取值填入上面创建的 rules 文件中对应的位置, 然后在 SYMLINK+ 中给这个设备取一个别名, MODE 设为 0777。

将底盘、雷达、IMU 依次插入树莓派 3 的 USB 口, 重复执行上面确定新插入串口设备的属性这一步, 直到将这 3 个串口都绑定完成。这里将底盘、雷达、IMU 分别 ttyUSB* 名称分别映射成别名 miiboo、lidar、imu, 这样在不改变底盘、雷达、IMU 插入树莓派 3 的 USB 口物理孔位顺序时, 不论上电开机后这 3 个串口被系统分配的 ttyUSB* 实际是多少, 我们都可以用映射好的别名 /dev/miiboo、/dev/lidar、/dev/imu 来访问底盘、雷达、IMU 串口, 并且不用担心用户访问权限不足的问题。这里特别说明, miiboo 机器人的底盘、雷达、IMU 插入树莓派 3 的 USB 口物理孔位顺序, 如图 29, 请不要改变这个顺序。



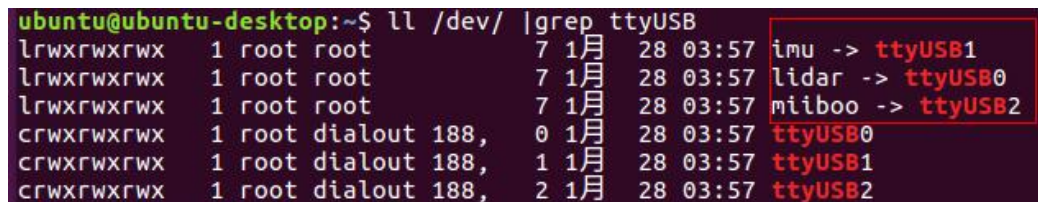
(图 29) miiboo 机器人的底盘、雷达、IMU 插入树莓派 3 的 USB 口物理孔位顺序

使绑定设置生效:

重启机器人使绑定设置生效，命令如下。

```
sudo reboot
```

机器人重启后通过命令查看绑定是否生效，看到下面的输出就说明绑定成功了，如图 30。



```
ubuntu@ubuntu-desktop:~$ ll /dev/ |grep ttyUSB
lrwxrwxrwx 1 root root 7 1月 28 03:57 imu -> ttyUSB1
lrwxrwxrwx 1 root root 7 1月 28 03:57 lidar -> ttyUSB0
lrwxrwxrwx 1 root root 7 1月 28 03:57 miiboo -> ttyUSB2
crwxrwxrwx 1 root dialout 188, 0 1月 28 03:57 ttyUSB0
crwxrwxrwx 1 root dialout 188, 1 1月 28 03:57 ttyUSB1
crwxrwxrwx 1 root dialout 188, 2 1月 28 03:57 ttyUSB2
```

(图 30) 查看绑定是否生效

7.开机自启动 ROS 节点

在后续教程学习完后，基本上所有的机器人算法就开发的差不多了。于是，我们就需要在机器人上电开机时，自启动想要启动的 ROS 节点，让机器人立马进入工作状态。

由于我自己尝试过网上的教程，将 ROS 节点启动命令放入系统开机自启动脚本的方法，亲测不成功。后来仔细研究才知道，需要系统完全启动，建立起 ROS 运行的必须环境后，ROS 节点程序才能开始工作，所以系统开机自启脚本行不通。

于是，经过我自己的研究，找到了一种简单的方法，将要启动的 `roslaunch` 命令加入机器人的 `~/.bashrc` 文件的末尾。这样在系统启动后，执行用户自动登录时，`~/.bashrc` 会被调用，这样就可以实现 ROS 节点启动了。关于用户自动登录的设置可以参考前面的相关内容。

添加到 `~/.bashrc` 文件末尾的内容如下，根据自己的需要，将命令替换成自己的 ROS 节点启动命令。

```
#robot auto start
if rostopic list ; then
    echo "roscore and robot_auto_start started!"
else
    #launch roscore
    nohup roscore
    #launch
    sleep 30
    ...
fi
```

设置完成后，`sudo reboot` 就可以了。