PX4 飞控开发手册

上海珞鹏航空科技有限公司（成都分公司）

2019年制

*开发必备*

**开发PX4必需**

1. 进行PX4开发需要以下基本设备：
   1. 能够运行Ubuntu 16.04及以上或者Deepin 15.5以上Linux系统的电脑，推荐使用虚拟机VMware 12安装Linux系统方便部署备份。
   2. Pixhawk 2及以上版本的Pixhawk飞控。
   3. 能够控制飞控的遥控器及接收机。
   4. 飞控地面站软件。
2. Linux开发环境的配置

依照官方文档中的sh脚本文件和教程可以很轻松完成编译环境的配置，官方教程链接如下：

<https://dev.px4.io/v1.9.0/en/setup/dev_env_linux.html>；

Ubuntu和Deepin的安装环境类似，都可以根据如下链接中的配置方式完成：<https://dev.px4.io/v1.9.0/en/setup/dev_env_linux_ubuntu.html>；

环境搭建好之后就可以下载固件源代码进行开发，具体教程如下：

<https://dev.px4.io/v1.9.0/en/setup/building_px4.html>；

下载好的源代码可以使用VisualStudio Code或者Sourceinsight建立工程进行开发。当然也可以参考以下步骤搭建自己的编译环境并进行固件编译。

**采用Qt或make命令编译px4步骤**

官方推荐Ubuntu/Debian Linux发行版本进行linux编译环境配置，国内deepin系统也是Debain发行分支下的版本，其环境完全一样，因此可以选择deepin系统进行编译环境配置。

Deepin linux下载地址：（<https://www.deepin.org/download/> ）,下载完成之后可以安装虚拟机VMware，将linux安装在虚拟机中。

**Linux安装完成后进入px4编译环境搭建**

Linux系统编译环境配置网站： (<https://dev.px4.io/en/setup/dev_env_linux_ubuntu.html> )

具体步骤如下：

1. 打开终端运行命令：

sudo usermod -a -G dialout $USER

完成之后注销当前用户重新登录。

1. 下载[ubuntu\_sim\_common\_deps.sh](https://raw.githubusercontent.com/PX4/Devguide/master/build_scripts/ubuntu_sim_common_deps.sh)**和**[ubuntu\_sim\_nuttx.sh](https://raw.githubusercontent.com/PX4/Devguide/master/build_scripts/ubuntu_sim_nuttx.sh)**文件**
2. 在下载目录下打开终端，运行命令，执行脚本文件：

source ubuntu\_sim\_common\_deps.sh

source ubuntu\_sim\_nuttx.sh

**注意事项：**编译环境配置过程中切勿使用sudo命令修复权限问题，否则会带来更多的权限设置问题，最终导致编译环境配置需要重装系统解决。

**备注：**当运行完两个脚本文件后，qtcreator和ninja等编译组件会自动安装，无须再次安装。当编译组件出问题时可以手动安装。

1. 删除Ubuntu modemmanager

sudo apt-get remove modemmanager

1. 升级编译相关组件

sudo apt-get update -y

sudo apt-get install git zip qtcreator cmake \

build-essential genromfs ninja-build -y

# Required python packages

sudo apt-get install python-argparse \

python-empy python-toml python-numpy \

python-dev python-pip -y

sudo -H pip install --upgrade pip

sudo -H pip install pandas jinja2 pyserial

1. 安装Python 可选包

# optional python tools

sudo -H pip install pyulog

1. 安装Ninja 编译组件

sudo apt-get install ninja-build -y

如果需要删除Ninja，运行命令：

sudo apt-get remove ninja-build -y

1. Pixhawk, Pixfalcon, Pixracer, Pixhawk 3为Nuttx系统的硬件，因此需要安装NuttX相关编译组件，以下是相关操作：

sudo apt-get install python-serial openocd \

flex bison libncurses5-dev autoconf texinfo \

libftdi-dev libtool zlib1g-dev -y

删除旧版本 arm-none-eabi 工具链：

sudo apt-get remove gcc-arm-none-eabi gdb-arm-none-eabi binutils-arm-none-eabi gcc-arm-embedded

sudo add-apt-repository --remove ppa:team-gcc-arm-embedded/ppa

执行如下脚本安装 5.4版本gcc-arm-none-eabi:

pushd .

cd ~

wget https://launchpad.net/gcc-arm-embedded/5.0/5-2016-q2-update/+download/gcc-arm-none-eabi-5\_4-2016q2-20160622-linux.tar.bz2

tar -jxf gcc-arm-none-eabi-5\_4-2016q2-20160622-linux.tar.bz2

exportline="export PATH=$HOME/gcc-arm-none-eabi-5\_4-2016q2/bin:\$PATH"

if grep -Fxq "$exportline" ~/.bash\_profile; then echo nothing to do ; else echo $exportline >> ~/.profile; fi

popd

执行如下命令安装32位库文件，如果运行的系统为32位系统，该安装会自动忽略跳过。

sudo dpkg --add-architecture i386

sudo apt-get update

sudo apt-get install libc6:i386 libgcc1:i386 libstdc++5:i386 libstdc++6:i386

sudo apt-get install gcc-4.6-base:i386

1. 重启系统后输入如下命令检查gcc安装状态

arm-none-eabi-gcc --version

如果得到以下信息说明安装成功

arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors) 5.4.1 20160609 (release) [ARM/embedded-5-branch revision 237715]

Copyright (C) 2015 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

如果得到如下信息，请检查32位编译组件是否正确安装

arm-none-eabi-gcc --version

arm-none-eabi-gcc: No such file or directory

**进入px4编译相关步骤**

1. 创建安装目录并下载固件

mkdir -p ~/src

cd ~/src

git clone https://github.com/PX4/Firmware.git

cd Firmware

下载完成之后运行命令进行初始化：

cd Firmware

git init

git submodule update --init --recursive

如果编译时遇到权限问题，执行指令：

sudo chmod 777 -R \*

-R 是对文件下面包含的子文件权限问题，\* 是对所有文件的权限问题编译。

默认下载的源代码是master分支，编译的代码需要切换到stable分支，因此代码下载完成后需要切换到stable分支。要本地切换到不同的的分支，即跟踪（如掌握每个默认的）库分支，首先通过提取更新，然后需要输入如下检测命令：

git fetch

git checkout <branchname>

切换到stable分支，需要执行

git checkout stable

完成切换。

有时我们需要切换版本号，查看版本号命令为：

git tag

切换到特定版本的分支命令如下（以1.7.3版本stable分支为例）：

git checkout -b <new-branch-name>

git checkout -b v1.7.3

完成之后运行

git checkout stable

即可完成分支切换。

查看目前git所在的分支运行命令：

git branch

Git相关命令可以通过运行：

git

进行查看。

更多git相关指令及解释在（<http://www.pixhawk.com/zh/dev/git> ）中。

Git 清除本地缓存

git clean -df

Git分支切换完成之后运行命令

make px4fmu-v2\_default

编译固件，第一次运行时会下载编译相关的组件。

编译相关知识px4官网链接（<https://dev.px4.io/en/setup/building_px4.html> ）中详细说明。针对不同的硬件平台，编译所需要的版本号需要相对应。

以下是不同硬件平台对应的固件版本编号：

[Pixhawk 1](https://docs.px4.io/en/flight_controller/pixhawk.html): make px4fmu-v2\_default

[HKPilot32](https://docs.px4.io/en/flight_controller/HKPilot32.html): make px4fmu-v2\_default

[Pixfalcon](https://docs.px4.io/en/flight_controller/pixfalcon.html): make px4fmu-v2\_default

[Dropix](https://docs.px4.io/en/flight_controller/dropix.html): make px4fmu-v2\_default

[mRo Pixhawk](https://docs.px4.io/en/flight_controller/mro_pixhawk.html): make px4fmu-v3\_default (supports 2MB Flash)

[mRo X-2.1](https://docs.px4.io/en/flight_controller/mro_x2.1.html): make auav-x21\_default

[Pixhawk 2](https://docs.px4.io/en/flight_controller/pixhawk-2.html): make px4fmu-v3\_default

[Pixracer](https://docs.px4.io/en/flight_controller/pixracer.html): make px4fmu-v4\_default

[MindPX](https://docs.px4.io/en/flight_controller/mindpx.html)/[MindRacer](https://docs.px4.io/en/flight_controller/mindracer.html): make mindpx-v2\_default

[Pixhawk Mini](https://docs.px4.io/en/flight_controller/pixhawk_mini.html): make px4fmu-v3\_default

[Pixhawk 3 Pro](https://docs.px4.io/en/flight_controller/pixhawk3_pro.html): make px4fmu-v4pro\_default

[Crazyflie 2.0](https://docs.px4.io/en/flight_controller/crazyflie2.html): make crazyflie\_default

[Intel® Aero Ready to Fly Drone](https://docs.px4.io/en/flight_controller/intel_aero.html): make aerofc-v1\_default

Pixhawk 4: make px4fmu-v5\_default

[AUAV-X2 (Discontinued)](https://docs.px4.io/en/flight_controller/auav_x2.html): make px4fmu-v2\_default

**Qt中编译px4**

安装Qt，（运行px4编译环境配置脚本文件已经默认安装qt，因此此步骤可以跳过）

sudo apt-get install qtcreator

1.配置Qt需要的编译目录，运行如下命令：

cd ~/src/Firmware

mkdir ../Firmware-build\_fmuv5

cd ../Firmware-build\_fmuv5

cmake ../Firmware -G "CodeBlocks - Unix Makefiles" -DCONFIG=nuttx\_px4fmu-v5\_default

如果需要编译fmuv3的固件，同样需要创建对应的目录和配置文件。

2.打开Qt后选择打开项目，选中/Firmware文件夹下面CMakeLists.txt文件，创建px4编译项目。

然后在项目build选项中构建步骤下面进行如下图的配置：

2.1选择src/Firmware-build作为构建目录

选择自定义进程步骤: make px4fmu-v2\_default

命令：make

参数：px4fmu-v2\_default

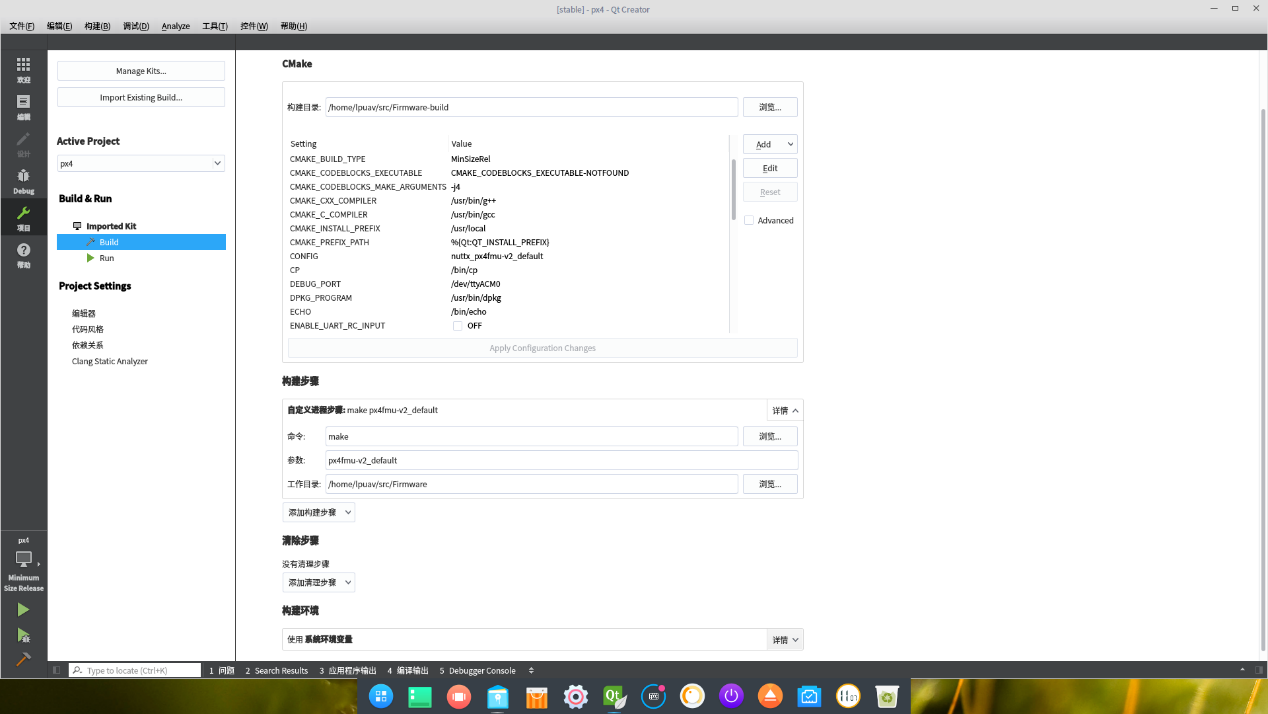
工作目录：/home/lpuav/src/Firmware（Firmware文件所在目录）

清除步骤

自定义进程步骤: make clean

命令：make

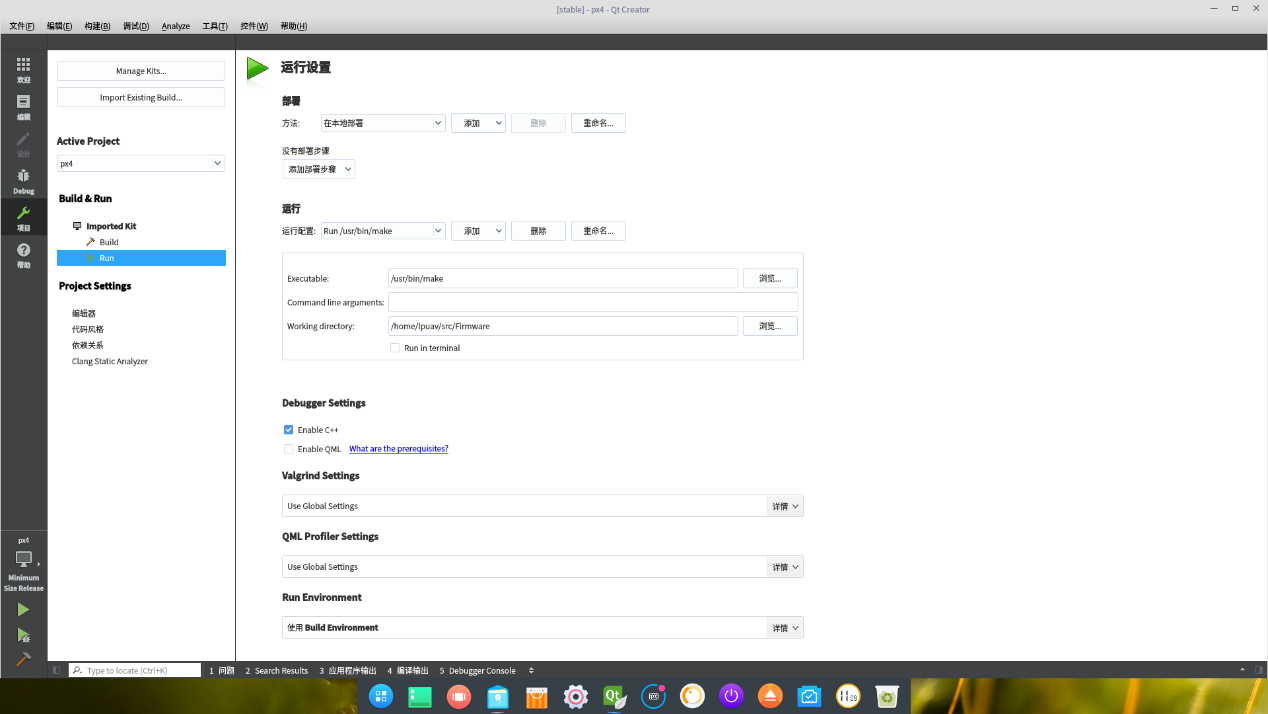
参数：clean

工作目录：/home/lpuav/src/Firmware（Firmware文件所在目录）

2.2 运行选项：

Excutbale：设置为/usr/bin/make （make命令所在目录，需要浏览查找）

Working directory：选择/home/lpuav/src/Firmware （Firmware文件所在目录）



3. 配置完成后点击运行图标：进行固件编译，编译输出如下图所示：



编译完成的固件存放在：

src/Firmware/build/px4fmu-v2\_default/px4fmu-v2\_default.px4位置。

**jmavsim软件仿真环境配置1**

Jmavsim软件仿真需要openjdk-7的环境支持，而系统可能安装了1.8的jdk，就需要重新安装选择1.7的jdk；

配置jdk环境步骤如下：

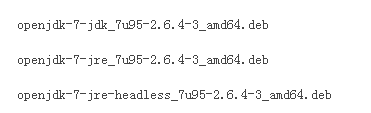
1.运行命令安装openjdk7

sudo apt-get install openjdk-7-jdk

2.下载open-jdk安装包

下载地址：<http://ppa.launchpad.net/openjdk-r/ppa/ubuntu/pool/main/o/openjdk-7/>

需要下载如下图所示3个安装包



下载完成后放到桌面或者其他文件夹，如下命令是放置到桌面目录运行的，如果放到其他目录只需要把安装包前面的地址更换即可。

这3个安装包的安装顺序如下：

sudo dpkg -i '/home/lpuav/Desktop/openjdk-7-jre-headless\_7u95-2.6.4-3\_amd64.deb'

sudo dpkg -i '/home/lpuav/Desktop/openjdk-7-jre\_7u95-2.6.4-3\_amd64.deb'

sudo dpkg -i '/home/lpuav/Desktop/openjdk-7-jdk\_7u95-2.6.4-3\_amd64.deb'

安装完成后查看系统中的jdk版本

lpuav@lpuav-PC:~/src/Firmware$ java -version

运行完成后应该出现如下信息

Picked up \_JAVA\_OPTIONS: -Dawt.useSystemAAFontSettings=gasp

java version "1.7.0\_95"

OpenJDK Runtime Environment (IcedTea 2.6.4) (7u95-2.6.4-3)

OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)

如果信息不对，可以通过java-config切换到1.7的jdk版本

lpuav@lpuav-PC:~/src/Firmware$ sudo update-alternatives --config java

运行后会出现如下信息，进行选择即可

有 2 个候选项可用于替换 java (提供 /usr/bin/java)。

选择 路径 优先级 状态

------------------------------------------------------------

0 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1081 自动模式

\* 1 /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java 1071 手动模式

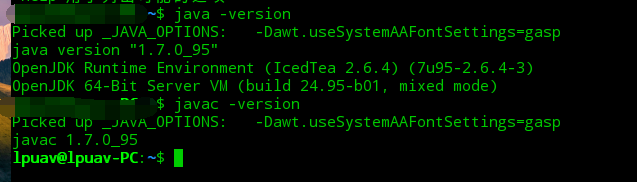
2 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1081 手动模式

要维持当前值[\*]请按<回车键>，或者键入选择的编号：

**jmavsim仿真安装及使用教程2**

此教程适用于在Linux下研究PX4时，安装jmavsim仿真和QGC地面站以测试无人机仿真飞行

**一、首先检查jdk版本**



提示：若jdk原本的版本就是7版，可不用重装jdk。重装步骤如下

1. 首先卸载旧的jdk

先执行

sudo apt-get autoremove openjdk-7-jdk或者sudo apt-get autoremove openjdk-8-jdk

卸载现有的jdk，然后再下载和安装

2、下载安装jdk7：

参考网址：<https://www.cnblogs.com/dinphy/p/5995597.html?mType=Group>

添加openjdk-7-jdk源，复制到终端运行：

sudo su -

echo "deb http://ppa.launchpad.net/openjdk-r/ppa/ubuntu xenial main" | tee -a /etc/apt/sources.list

echo "deb-src http://ppa.launchpad.net/openjdk-r/ppa/ubuntu xenial main" | tee -a /etc/apt/sources.list

apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 86F44E2A

apt-get update

exit

执行安装命令：

sudo apt-get install openjdk-7-jdk



安装openjdk-7-jdk问题比较多，下载时很有可能会一直停下来，提示 \* %[正在等待报头] ，这个问题网上有一些解决办法，但针对这里不适用。

这里的**解决办法**是：

1. 到添加源的网页单独下载  **openjdk-7-jdk  openjdk-7-jre  openjdk-7-jre-headless**  这三个文件。下载地址： <http://ppa.launchpad.net/openjdk-r/ppa/ubuntu/pool/main/o/openjdk-7/>

　　目标文件（对应系统版本，下载最新的）：

　　　　openjdk-7-jdk\_7u95-2.6.4-3\_amd64.deb

　　　　openjdk-7-jre\_7u95-2.6.4-3\_amd64.deb

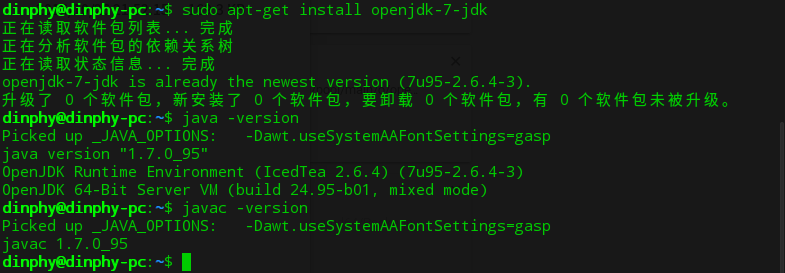
　　　　openjdk-7-jre-headless\_7u95-2.6.4-3\_amd64.deb

　　（2）将下载的这三个文件复制或移动到  /var/cache/apt/archives/ 目录下，比如我这里采用的是复制：

sudo cp '/home/dinphy/Downloads/openjdk-7-jdk\_7u95-2.6.4-3\_amd64.deb' '/home/dinphy/Downloads/openjdk-7-jre-headless\_7u95-2.6.4-3\_amd64.deb' '/home/dinphy/Downloads/openjdk-7-jre\_7u95-2.6.4-3\_amd64.deb' /var/cache/apt/archives/

具体复制地址以个人电脑为定

（3）重新执行安装 openjdk-7-jdk 的命令，解决。



安装完成（安装了jdk后修改jdk的环境变量，只需要用vim etc/profile打开文件在最后一行添加jdk路径变量）

选择电脑上下载好的jdk：

<https://www.cnblogs.com/keitsi/p/5817433.html>

解决查询Java version版本依旧是1.8的问题网站如下：

<https://bbs.deepin.org/forum.php?mod=viewthread&tid=5965>

https://bbs.deepin.org/forum.php?mod=viewthread&tid=36259&extra=

**二、装JMAVSim仿真软件**

（装之前要删除掉所有名为JMAVSIM的文件：firmware目录下的和主目录的，jdk装好的不要执行apt-get那一步）

参考网址：

<http://www.pixhawk.com/dev/hil/jmavsim#hardware_in_the_loop_hitl>

1. 克隆jmavsim：git clone <https://github.com/PX4/jMAVSim.git>
2. 若jdk已经装好，不用执行此步：

sudo apt-get install ant openjdk-7-jdk openjdk-7-jre

3.工程建立和运行

运行jMAVSim需要初始Git仓库（需要在jMAVSim目录下进行），初始化Git仓库：

git submodule init

git submodule update

1. 接着编译jMAVSim，运行以下语句：

ant

将会输出以下情形：

Buildfile:/jMAVSim/build.xml

make\_dirs:

compile:

jmavsim:

all:

BUILD SUCCESSFUL

Total time: 0 seconds

出现以上情形说明jMAVSim仿真环境初始化完毕。

**三、打开jmavsim**

**在firmware目录下**

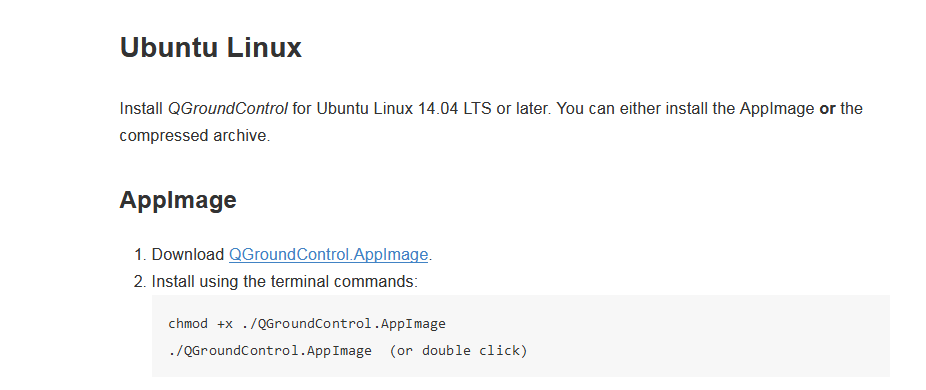
make posix jmavsim

**四、下载安装QGC**

**1：下载qgroundcontrol压缩包**

下载地址：

<https://docs.qgroundcontrol.com/en/getting_started/download_and_install.html#ubuntu-linux>

点击Download下载QGC。

**2：解压，按照其压缩的格式选择压缩命令**

tar -jxvf qgroundcontrol.tar.bz2

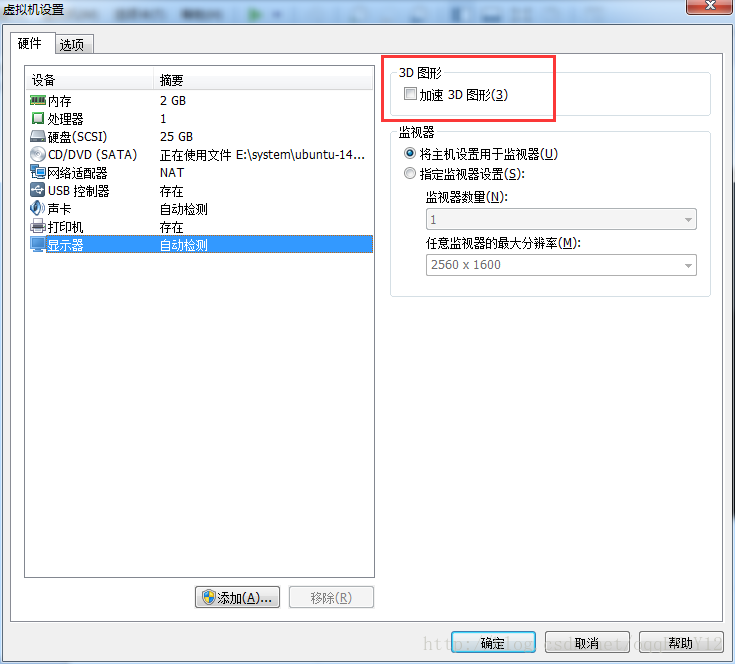
**3：安装附加软件包**

sudo apt-get install espeak libespeak-dev libudev-dev libsdl1.2-dev

**4：启动qgc**

进入qgroundcontrol，执行./qgroundcontrol-start.sh启动qgc。

5. QGC地图未显示:关闭虚拟机，在显示器设置里面关闭3D



**4.打开QGC**

在/home/lpuav/src/qgroundcontrol下打开终端输入

./qgroundcontrol-start.sh

若无法打开，可能是缺乏依赖项，根据此网站安装依赖项：

http://blog.csdn.net/oqqenvy12/article/details/55271721

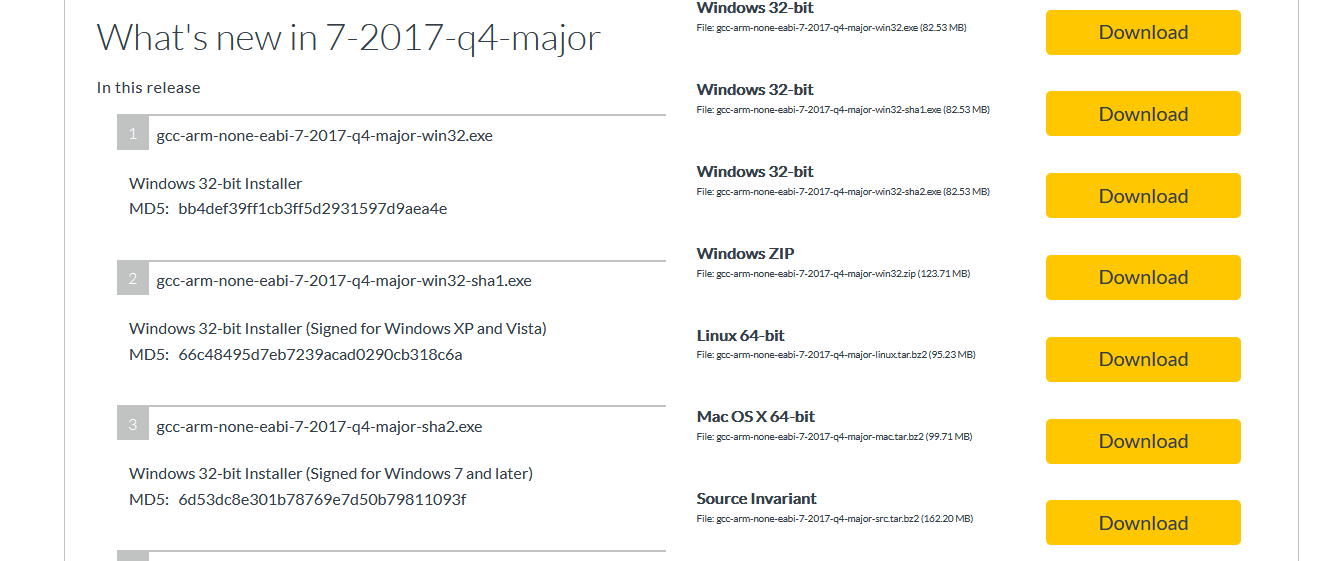


备注：得到系统文件的读写权限：sudo dde-file-manager root权限：sudo su

**五、升级GCC**

1．下载gcc：

<https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads/>



2．下载安装包后解压

根据/home/lpuav/Documents的ubuntu\_sim\_nuttx.sh的clean old gcc的第一个命令所示卸载arm-none-eabi，

3．在主目录etc/profile里面添加路径：

export PATH=$PATH:/home/lpuav/gcc-arm-none-eabi-7-2017-q4-major/bin

在etc下面用source profile使路径生效，

升级gcc可能遇到动态库未更新，可参考此网站：

https://www.cnblogs.com/lzpong/p/5755678.html

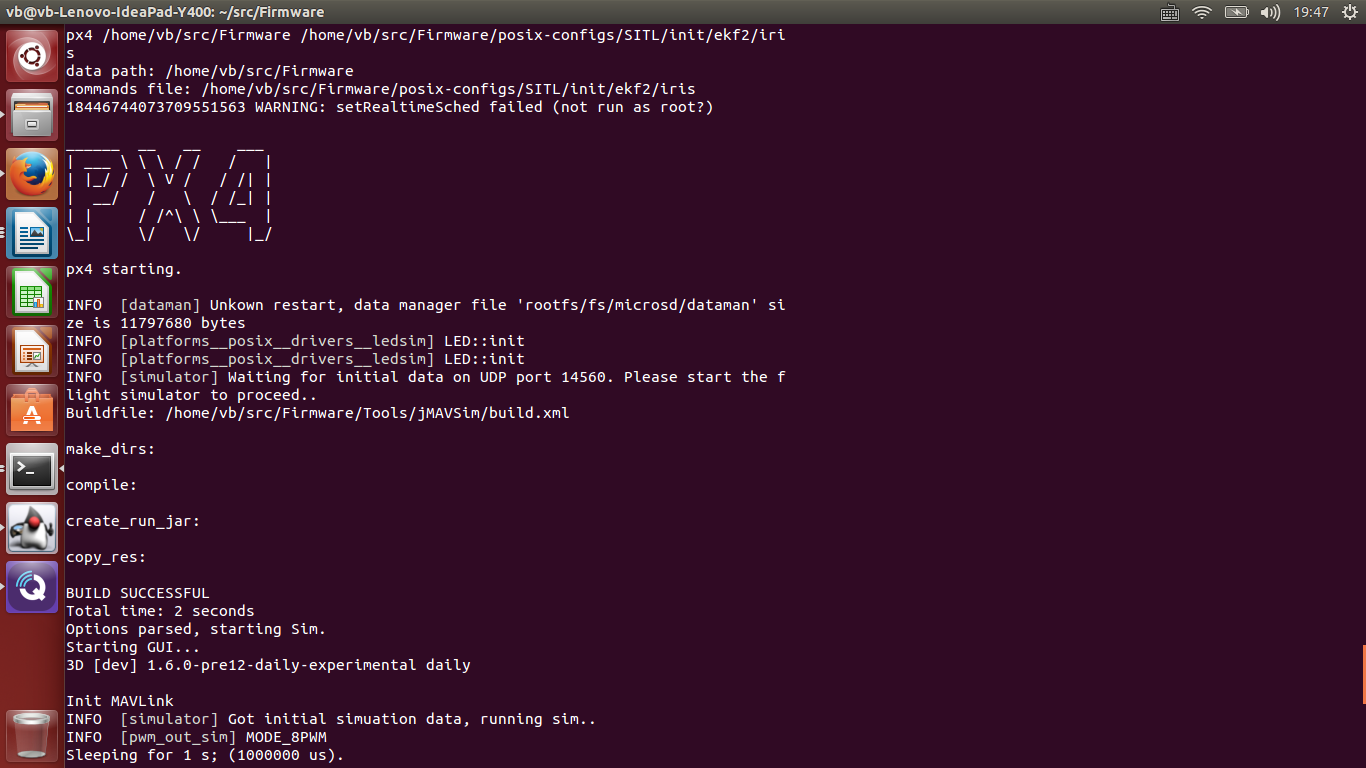
**六、.JMAVSIM软件仿真办法**

(1)打开终端，切换至Firmware文件夹：

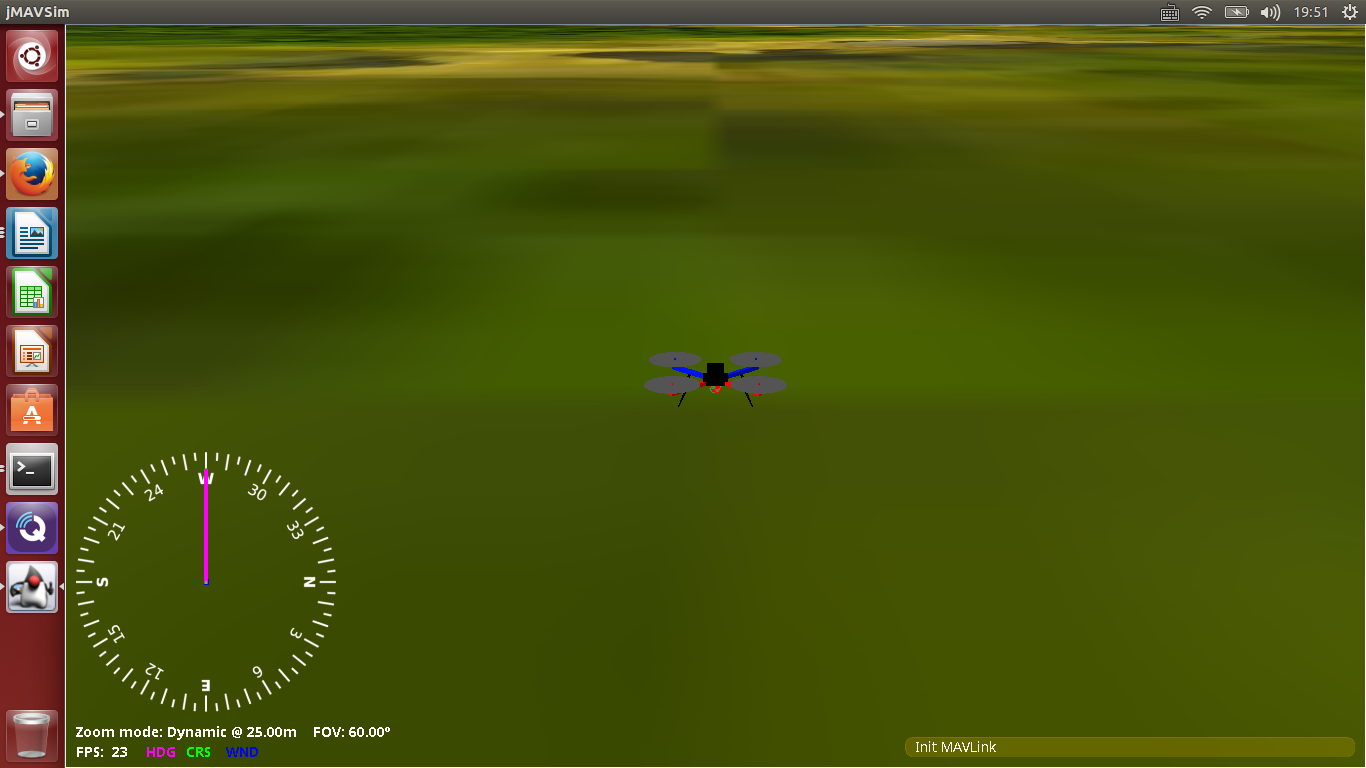
cd ~/src/Firmware

然后输入一下指令:

make posix\_sitl\_default jmavsim

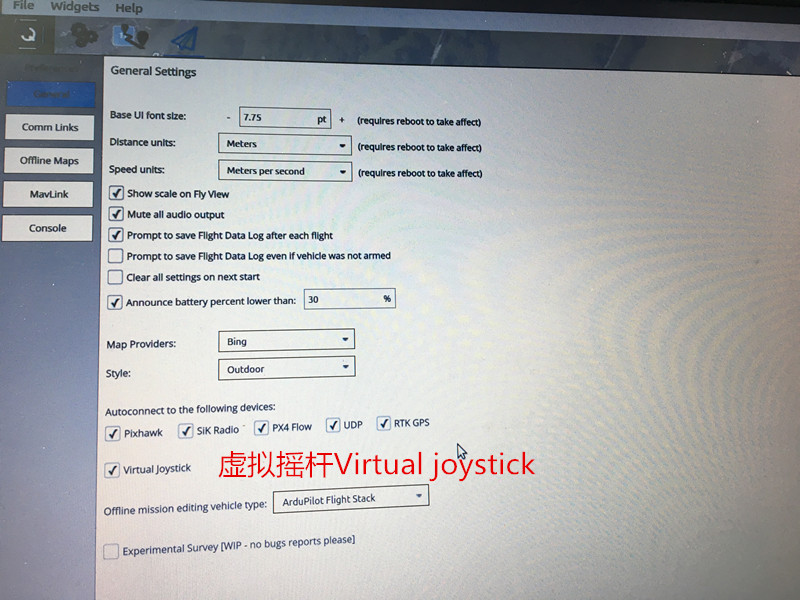


这句代码的意思是，编译firmware 目标代码编译为posix系统软件仿真代码，并用仿真器打开编译好的目标文件。显示编译成功，并且打开了3D仿真界面，MAVLink初始化成功。



(2)打开QGC，QGC会自动连接到仿真四旋翼（不要连接飞控硬件）。

打开QGC的虚拟遥控器，通过虚拟遥控器就可以控制飞机飞行



(3)起飞，遥控飞机

在已经打开的仿真器的终端中，先回车出现“pxh”，再输入

commander takeoff使飞机起飞；

在QGC界面中双击上方的HOLD选择mission飞行模式，规划mission，开始操纵仿真飞机飞行，此时QGC和通过数传连接飞控的效果相同，可使用各种命令操作飞机。



**七、硬件仿真方法**

参考网址：<http://www.amovauto.com/portal.php?mod=view&aid=71>

在进行仿真时关闭QGC，确保飞控没有连接到QGC。

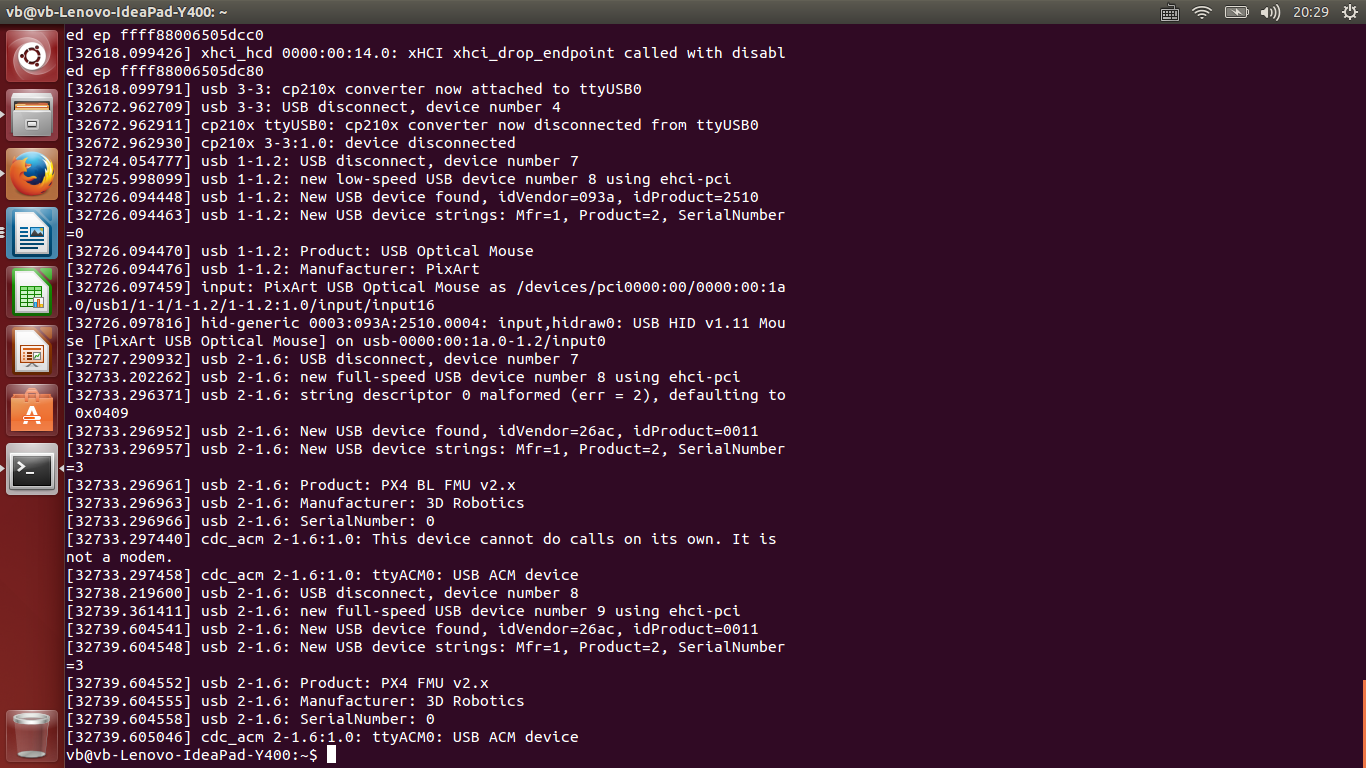
(1)查看飞控连接情况

首先将pixhawk通过USB连接电脑，在终端使用指令：

make posix\_sitl\_default jmavsim

sudo dmesg

查看安装驱动的信息，出现以下情形说明飞控以正常连接计算机。



(2)设置串口属性

串口属性更改通过Linux系统的工具minicom来实现，好像系统没有自带，所以我们自己安装。

**安装**

在终端中输入

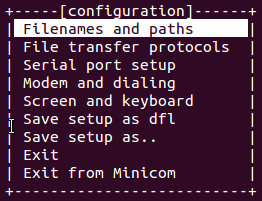
make posix\_sitl\_default jmavsim

sudo apt-get install minicom

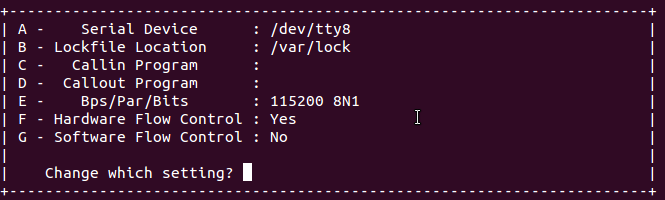
**配置**

输入sudo minicom -s

弹出下图设置界面



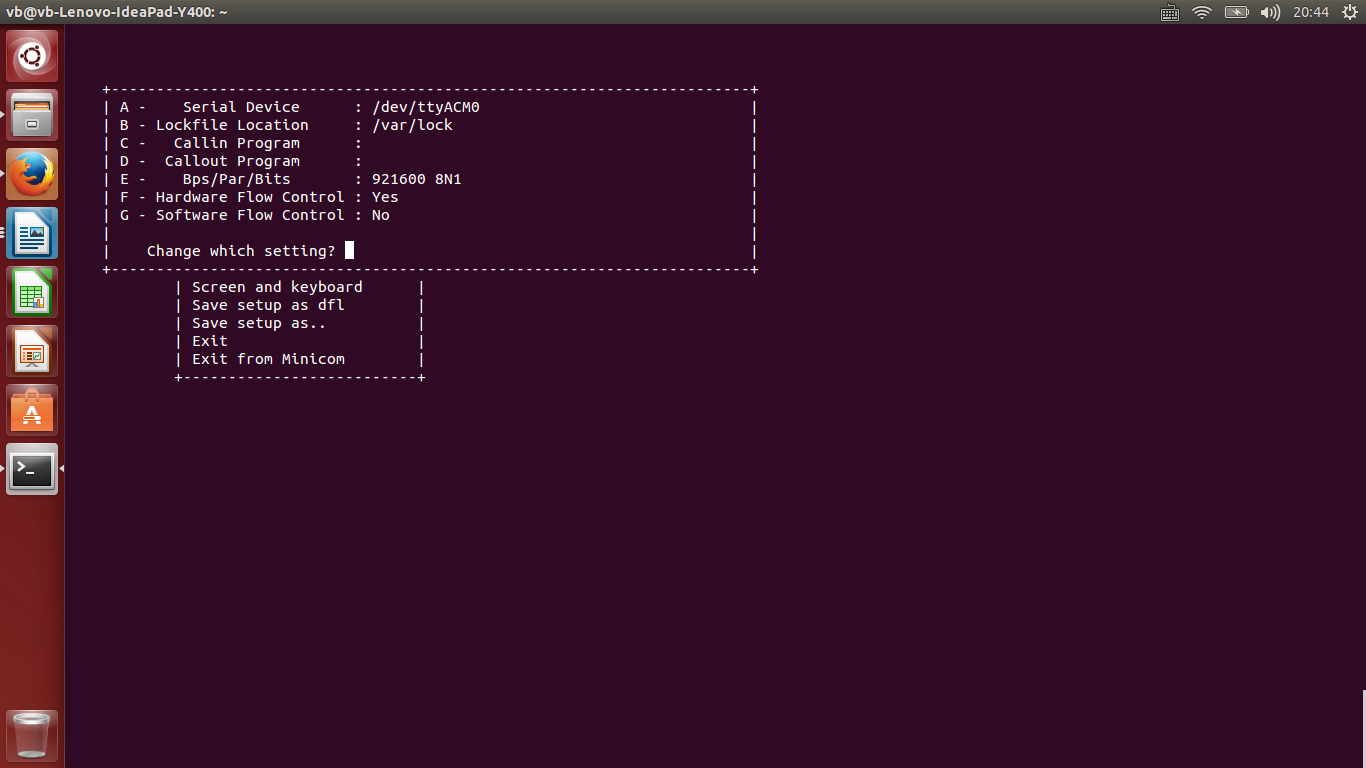
使用方向键 选择 Serial port setup，按Enter键，进入设置环境，如下图



输入a或者A，选择串口设备，如果USB上只连接了飞控，串口名字可以随意更改，安装官方指南可将/dev/tty8其改为/dev/ttyACM0，如果还有其他设备请按照以下规则进行更改：使用USB转串口，那么串口COM1对应ttyUSB0, COM2对应ttyUSB1。

配置完串口设备后，按Enter键，再输入E，配置波特率，按照官方指南，波特率配置为921600，其余保持不变（数据位：8，奇偶校验位：N 无，停止位：1）。

配置完波特率，按Enter键，其余可以不用更改。



都配置完后，按下Enter键返回上一界面，选择save setup as dfl（即将其保存位默认配置），再选择Exit，关闭minicom。

此时会接收到来自飞控的数据（当然是乱码的！）。或者也可以关闭终端，重新开启终端输入指令：

make posix\_sitl\_default jmavsim

sudo minicom

同样可以接收到飞控数据，说明串口配置成功！

（以上配置方法来源于网络：

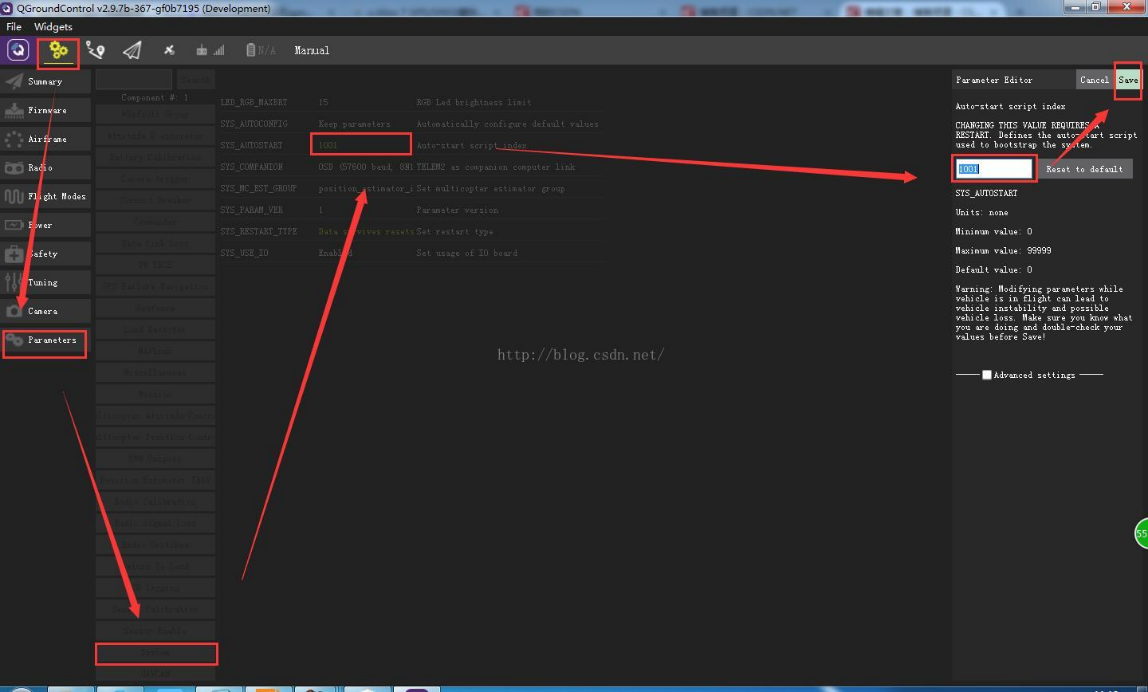
<http://blog.csdn.net/mybelief321/article/details/8987502> ，感谢博主的分享）

(3)飞控硬件设置

**进入仿真模式**

有两种方法进入仿真模式（参见官网：<http://www.pixhawk.com/dev/hil/jmavsim> ），由于第一种方法涉及重新校准传感器，所以本文采用第二种方法。

使用QGC：链接板子，点击设置，进入parameters，手动：设置SYS\_AUTOSTART参数为1001，保存重启。



设置完毕关闭QGC。

(2)执行下面指令，打开端口权限：

sudo chmod 777 /dev/ttyACM0

注意：输入打开端口权限指令的时候，飞控一定要连接电脑，不然会显示找不到文件。

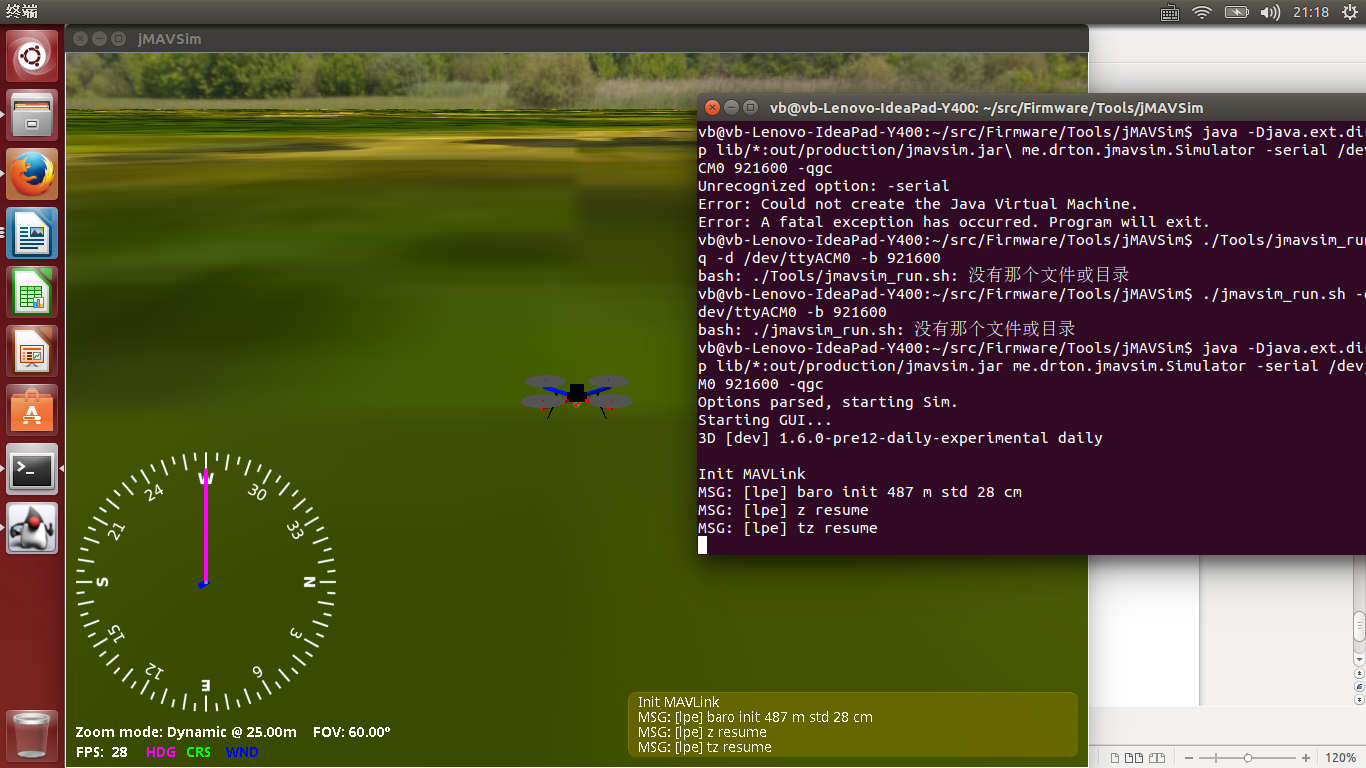
(3)进入jmavsim 目录，按照官网步骤，输入指令进入仿真界面（“\”是续行符号，与下一行是一个命令）：

java -Djava.ext.dirs= -cp lib/\*:out/production/jmavsim.jar\ me.drton.jmavsim.Simulator -serial /dev/ttyACM0 921600 –qgc

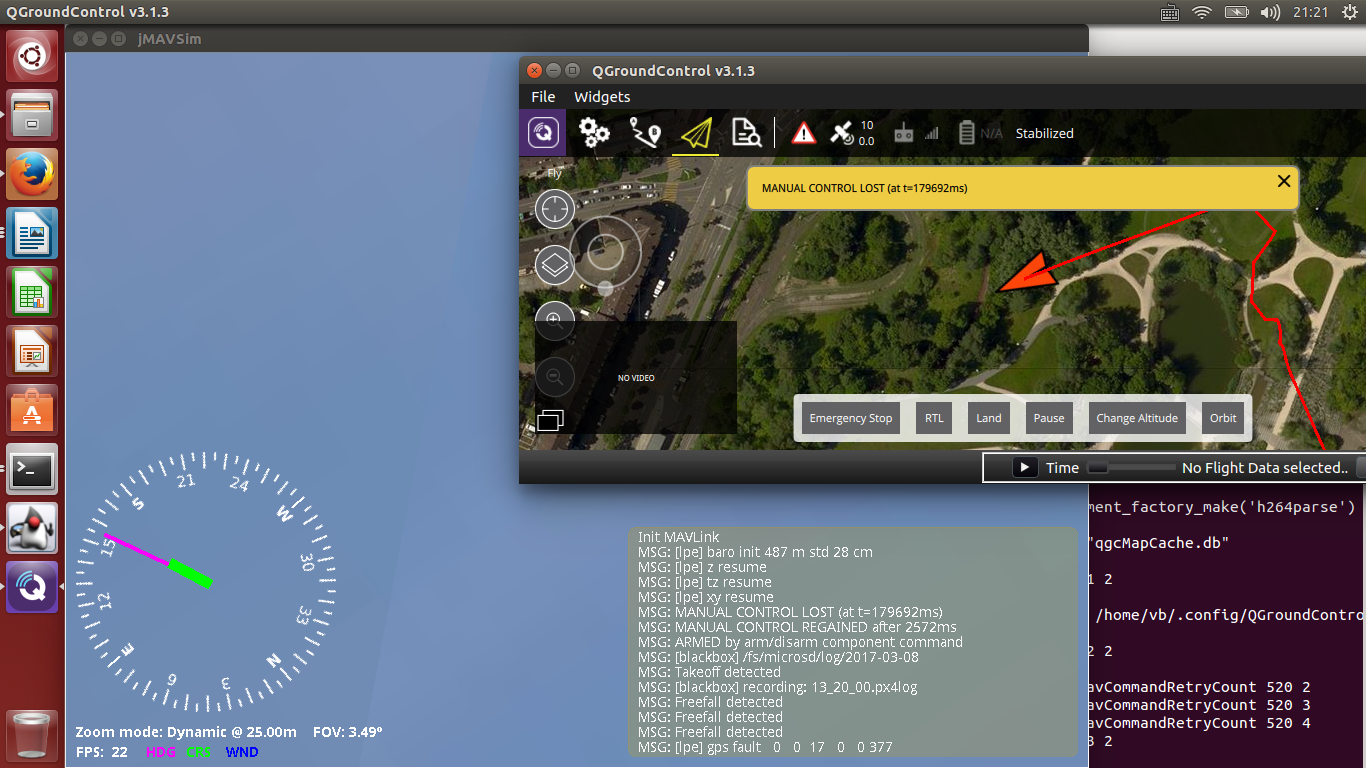
这个命令打不开的话可以在firmware目录下使用此命令试试:

./Tools/jmavsim\_run.sh -q -d /dev/ttyACM0 -b 921600 -r 250

然后jMAVSim就成功打开了！同时终端中会显示来自飞控的mavlink信息，效果如下：



(4)打开QDC进行连接，选择默认的UDP进行连接，接下来就可以进行仿真了。可以使用遥控器进行控制，效果如下，和真实操作一样：



提示：HITL仿真飞行时选机型选HIL Quadcopter X，SYS\_AUTOSTART参数为1001

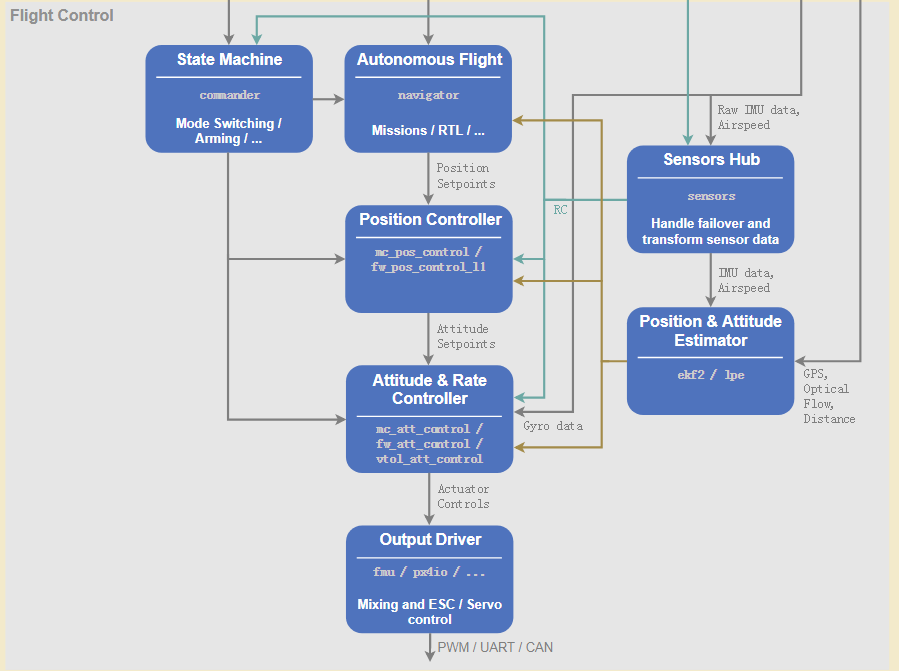
*PX4结构*

**PX4主要层组成**

PX4由两个主要层组成，飞行控制层是一个估计和飞行控制系统，中间层是一个通用的机器人层，可以支持任何类型的自主机器人，提供内部/外部通信和硬件集成。

具体介绍如下链接：

<https://dev.px4.io/v1.9.0/en/concept/architecture.html>



飞行控制层包含了飞行器飞行所需的指引，导航和控制算法。它还包含了包括固定翼、多旋翼、vtol机型的控制和姿态及位置估计。

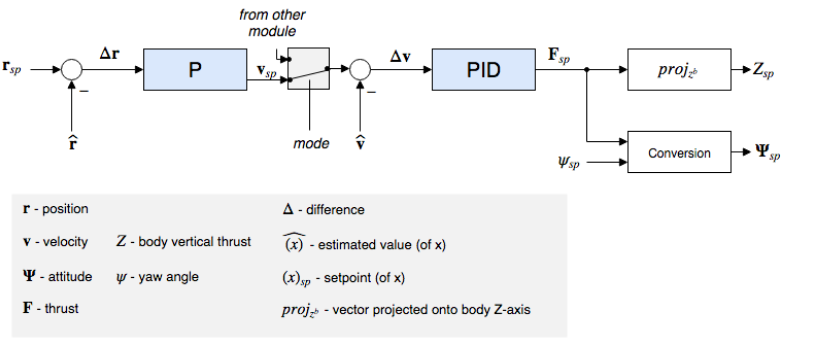


*控制算法*

在PX4中，飞行器姿态控制都是采用PID控制，固定翼位置采用L1自适应控制算法，多旋翼仍然采用PID进行位置控制。有关控制部分资料详见如下链接：

<https://dev.px4.io/v1.9.0/en/flight_stack/controller_diagrams.html>；

了解了控制部分的原理之后就可以对控制算法进行改进，下图所示的是多旋翼位置控制部分的算法，姿态控制部分算法与其类似，都是根据期望的变量和估计器生成的实际变量的误差利用PID算法进行控制，了解了这一点就可以对期望值的生产进行优化，同时对状态变量和误差的控制进行改进，例如采用ADRC（自抗扰控制）算法就可以有效改进姿态控制的输出。



需要注意的是，PX4中PID算法中的D（微分项）与传统PID中的微分有所不同，例如姿态控制中的内环D事实上是当前角速度经过滤波后的导数，起到对控制产生阻尼的作用。

位置控制中的D是速度的导数，同样起到阻尼的作用，这与传统意义上D（误差的差的导数）有所不同。

*PX4的硬件和软件架构*

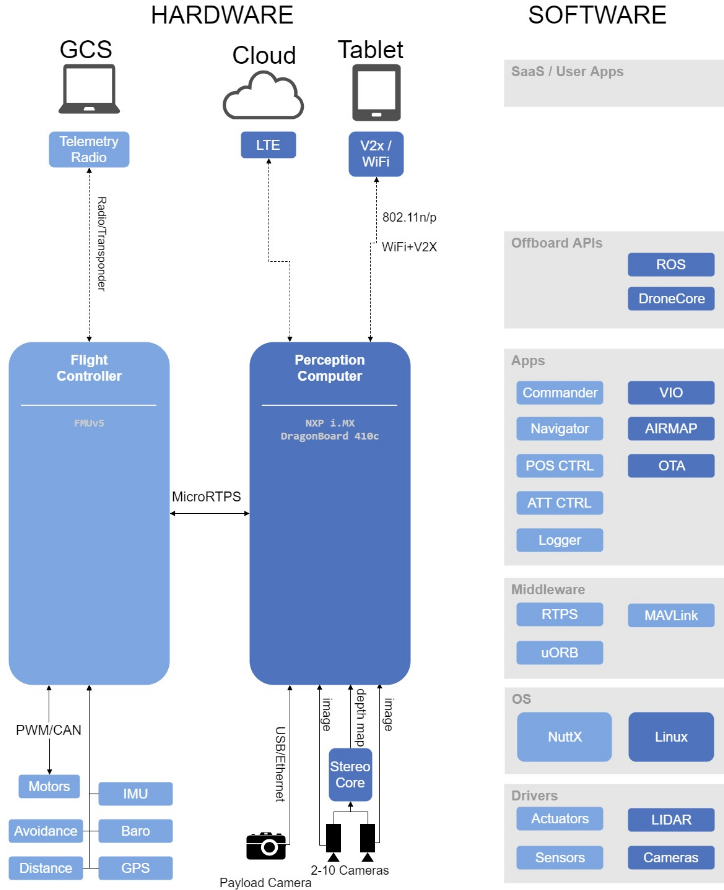
**PX4的硬件基本连接方式**

PX4的硬件基本连接方式如下图左侧所示，飞控和协处理计算机充当整个硬件系统的中间结构，连接了监控设备以及挂载设备和执行器等关键部件。

PX4的软件结构可以明显由图看出其架构，外部控制的API接口可以方便外部计算机控制飞行器，而APP应用层则包含了飞控关键的模块，包含控制模块，导航模块，指令模块，日志模块等；中间层包含了系统关键的orb消息以及mavlink消息，负责app和底层的通信，drivers层包含了各种传感器驱动，是飞控获得数据来源的核心模块。

有关PX4的架构，详细链接如下：

<https://dev.px4.io/v1.9.0/en/concept/dronecode_architecture.html>；



PX4飞行模式

PX4的飞行模式定义了飞行器对控制信号输入的反馈。不同的飞行模式下，操作的方式和理论都有差异，完整的飞行模式开发文档如下：

<https://dev.px4.io/v1.9.0/en/concept/flight_modes.html>；

该文档详细阐述了不同的飞行模式含义，同时也详细介绍了飞行模式的判断流程，展示了在何种条件下能够实现特定的飞行模式。

混控器和执行器

**混控和执行器**

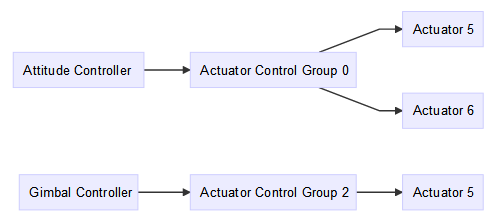
混控指的是把输入指令（例如：遥控器打右转）分配到电机以及舵机的执行器（如电调或舵机PWM）指令。对于固定翼的副翼控制而言，每个副翼由一个舵机控制，那么混控的意义就是控制其中一个副翼抬起而另一个副翼落下。同样的，对多旋翼而言，俯仰操作需要改变所有电机的转速。

混控和执行器的详细说明在如下链接中：

<https://dev.px4.io/v1.9.0/en/concept/mixing.html>

**控制流程**

一个特定的控制器（如姿态控制器）发送特定的归一化（-1..+1）的命令到给混合（mixing）,然后混合后输出独立的PWM到执行器（电调，舵机等）.在经过输出驱动如（串口，UAVCAN，PWM）等将归一化的值再转回特性的值（如输出1300的PWM等）。

**控制组**

PX4 有输入组和输出组的概念，顾名思义：控制输入组（如： attitude），就是用于核心的飞行姿态控制，（如： gimbal ）就是用于挂载控制。一个输出组就是一个物理总线，如前8个PWM组成的总线用于舵机控制，组内带8个归一化（-1..+1）值,一个混合就是用于输入和输出连接方式（如:对于四轴来说,输入组有俯仰，翻滚，偏航等，对于于向前打俯仰操作，就需要改变输出组中的4个电调的PWM输出值，前俩个降低转速，后两个增加转速，飞机就向前）。

对于简单的固定翼来说，输入0（roll），就直接连接到输出的0（副翼）。对于多旋翼来说就不同了，输入0（roll）需要连接到所有的4个电机。

Control Group #0 (Flight Control)

0: roll (-1..1)  
1: pitch (-1..1)  
2: yaw (-1..1)  
3: throttle (0..1 normal range, -1..1 for variable pitch / thrust reversers)  
4: flaps (-1..1)  
5: spoilers (-1..1)  
6: airbrakes (-1..1)  
7: landing gear (-1..1)

Control Group #1 (Flight Control VTOL/Alternate)

0: roll ALT (-1..1)  
1: pitch ALT (-1..1)  
2: yaw ALT (-1..1)  
3: throttle ALT (0..1 normal range, -1..1 for variable pitch / thrust reversers)  
4: reserved / aux0  
5: reserved / aux1  
6: reserved / aux2  
7: reserved / aux3

Control Group #2 (Gimbal)

0: gimbal roll  
1: gimbal pitch  
2: gimbal yaw  
3: gimbal shutter  
4: reserved  
5: reserved  
6: reserved  
7: reserved (parachute, -1..1)

Control Group #3 (Manual Passthrough)

0: RC roll  
1: RC pitch  
2: RC yaw  
3: RC throttle  
4: RC mode switch  
5: RC aux1  
6: RC aux2  
7: RC aux3

Control Group #6 (First Payload)

0: function 0 (default: parachute)  
1: function 1  
2: function 2  
3: function 3  
4: function 4  
5: function 5  
6: function 6  
7: function 7

Virtual Control Groups

These groups are NOT mixer inputs, but serve as meta-channels to feed fixed wing and multicopter controller outputs into the VTOL governor module.

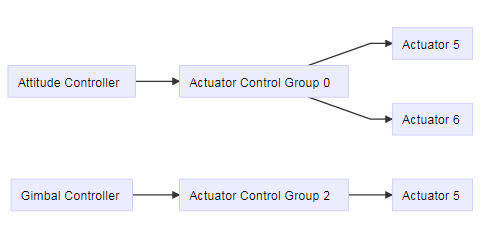
Control Group #4 (Flight Control MC VIRTUAL)

0: roll ALT (-1..1)  
1: pitch ALT (-1..1)  
2: yaw ALT (-1..1)  
3: throttle ALT (0..1 normal range, -1..1 for variable pitch / thrust reversers)  
4: reserved / aux0  
5: reserved / aux1  
6: reserved / aux2  
7: reserved / aux3

Control Group #5 (Flight Control FW VIRTUAL)

0: roll ALT (-1..1)  
1: pitch ALT (-1..1)  
2: yaw ALT (-1..1)  
3: throttle ALT (0..1 normal range, -1..1 for variable pitch / thrust reversers)  
4: reserved / aux0  
5: reserved / aux1  
6: reserved / aux2  
7: reserved / aux3

**映射**

因为存在许多控制组（例如飞行控制组，载荷组等）和许多输出组（例如基本8路PWM输出组，UAVCAN组等），所以一个控制组可以向多个输出组发送指令。

**PX4混控器定义**

ROMFS/px4fmu\_common/mixers中的文件实现了预定义机架所使用的混控器。它们可以用于自定义机架或者一般的测试。

**语法**

mixer通过文本文件定义；以单个大写字母加一个冒号开始的行是有效的。其它的行则会被忽略，这意味着注释可以自由地在定义中穿插使用。

每个文件可以定义多个混控器；混控器与作动器的分配关系由读取混控器定义的设备决定，作动器输出数目则由混控器决定。

例如：每个简单混控器或者空混控器按照它们在混控器文件中出现的顺序对应到输出1到输出x。

一个混控器定义通常具有如下形式：

<tag>: <mixer arguments>

tag标签决定混控器的类型；M对应简单求和混控器，R对应多旋翼混控器，等等。

**空混控器**

空混控器不接受控制输入并产生单个作动器输出，其输出值恒为零。空混控器的典型用法是在一组定义作动器特定输出模式的混控器组中占位。

空混控器定义形式如下：

Z:

**简单混控器**

简单混控器将0个或多个控制输入混合为单个作动器输出。所有输入被缩放后，经过混合函数得到混合后的输入，最后再经过输出缩放产生输出信号。

简单混控器定义如下：

M: <control count>

O: <-ve scale> <+ve scale> <offset> <lower limit> <upper limit>

如果 <control count> 为0，那么混合结果实际上为0，混控器将输出一个定值，这个值是在<lower limit>和<upper limit>限制下的<offset>。

第二行用前文讨论过的缩放参数定义了输出缩放器。计算以浮点操作被执行，存储在定义文件中的值经过了因子10000的缩放，即偏移量-0.5会被存储为-5000。

紧跟在词目之后的定义描述了控制输入以及它们的缩放，形式如下：

S: <group> <index> <-ve scale> <+ve scale> <offset> <lower limit> <upper limit>

<group>值标示了控制输入来源，缩放器从中读取控制量，<index>值则是控制量在组内的序号。这些值对读取混控器定义的设备而言都是特定的。

当用来混合载体控制时，控制组0是载体姿态控制组，序号0到3通常对应滚转，俯仰，偏航和油门。

混控器定义行中剩下的域则用来配置缩放器，参数如前文讨论。计算以浮点操作被执行，存储在定义文件中的值经过了因子10000的缩放，即偏移量-0.5会被存储为-5000。

**多旋翼混控器**

多旋翼混控器将4个控制输入（滚转，俯仰，偏航，油门）混合至一组作动器输出，这些作动器用来驱动电机转速控制器。

多旋翼混控器定义如下所示：

R: <geometry> <roll scale> <pitch scale> <yaw scale> <deadband>

支持的构型包括：

* + 4x - X型布局四旋翼
  + 4+ - +型布局四旋翼
  + 6x - X型布局六旋翼
  + 6+ - +型布局六旋翼
  + 8x - X型布局八旋翼
  + 8+ - +型布局八旋翼

每个滚转，俯仰，偏航缩放值定义了滚转，俯仰，偏航控制相对于油门控制的缩放。计算以浮点操作被执行，存储在定义文件中的值经过了因子10000的缩放，即偏移量-0.5会被存储为-5000。

滚转，俯仰和偏航输入的范围为-1.0到1.0，而油门输入的范围为0.0到1.0，执行器输出范围为-1.0到1.0。

当某个执行器饱和时，为保证该执行器值不超出范围，所有的执行器值都会被重新缩放，使得执行器的饱和上限被限制到1.0以内。

**直升机混控器**

直升机混控器将姿态和油门控制输入（roll, pitch, thrust）通过4个执行器（斜盘舵机和电机电调）混控输出，第一项输出为油门到发动机或电机的映射，随后的输出为斜盘舵机输出。尾桨通过一个简单混控器进行混控输出。

油门输入在电机设置和斜盘的总矩上都有体现。通过油门曲线和油门对应总矩的曲线来定义，这两个曲线都通过5点插值的方式产生。

通过将油门杆或者推力设定值映射到油门曲线和总矩曲线，获得总的推力。这种混控方式可以适应多种斜盘舵机和动力来源。如何调整油门曲线或者总矩曲线，以下链接作了详细介绍， [this guide](https://www.rchelicopterfun.com/rc-helicopter-radios.html).

直升机的混控具体定义为:

H: <number of swash-plate servos, either 3 or 4>

T: <throttle setting at thrust: 0%> <25%> <50%> <75%> <100%>

P: <collective pitch at thrust: 0%> <25%> <50%> <75%> <100%>

T: 定义了油门曲线上5个点的输出值. P: 则定义了总矩曲线. 这两个曲线的输出值都是从 0到10000. 当定义为线性曲线时，这5点的输出可设定为 0 2500 5000 7500 10000.

在这之后跟随了每个斜盘舵机的设定值

S: <angle> <arm length> <scale> <offset> <lower limit> <upper limit>

<angle> 单位是度数，0度对应机头方向 ，从上往下看，角度顺时针方向增大。

 <arm length> 是长度归一化后的值 10000 相当于 1. 如果所有舵机摇臂的长度一样，则都设置为 10000. 较长的摇臂有利于减小舵机的输出偏差.

舵机的输出通过 <scale> / 10000. 进行缩放，缩放之后，中位偏差值 <offset> 加载到混控器当中，偏差值同样是 -10000 到 +10000之间的值。

<lower limit> 和 <upper limit> 在舵机最大行程位置，应当在 -10000 到 +10000 范围内

尾桨通过一个简单混控器进行定义:

M: 1

S: 0 2 10000 10000 0 -10000 10000

机型开发

PX4由于采用混控将控制组和执行器桥接起来，因此具有很强的扩展能力，详细的介绍如下链接；

<https://dev.px4.io/v1.9.0/en/airframes/>；

**机型定义**

PX4的机型定义由2部分组成：

1. 机型的配置文件

机型配置文件包含了所定义的机型所在的类别，是属于固定翼、多旋翼还是VTOL等；同时配置文件中还包含了对应机型的一些自定义参数，当定义该机型后，可以调用这些参数的值；配置文件也对各个执行器的属性作了备注，不同接口对应的执行器都有详细说明；配置文件中包含了机型定义最为核心的部分就是Mixers混控文件，事实上混控文件是区别机型的根本文件，在混控文件中控制组的输出在执行器上进行缩放定义，最终达到了控制到输出的连接。

1. 机型的混控文件

机型混控文件也包含2部分，这2部分的定义是根据硬件输出端口的配置来决定的。例如Pixhawk 2有8个Main通道输出和6个Aux通道输出，这样，对应机型的\*.main.mix最多可以定义8个混控输出；\*.aux.mix最多能够定义6个混控输出。

**添加新机型和新机型组**

有关如何添加新的机型和机型组的开发文档如下链接：

<https://dev.px4.io/v1.9.0/en/airframes/adding_a_new_frame.html#config-file>；