首先你得需要有一台运行 Linux 系统的机器，虚拟机、双系统都可以。Linux 系统的安装可以在百度上查找，安装教程也很详细。本教程是在 Ubuntu16.04 上进行的。

window 下的 IDE 集成了代码编辑器、编译器、调试器和图形用户界面等工具，直接给你生成了编译文件。在 linux 下这些开发环境还是得自己安装搭建。我们可以利用 openocd+arm-none-eabi-gcc+gdb 搭建 openmv3 的开发环境。

关于 openocd 的，网上和官网上大体都有介绍，其实就是一个开源的片上调试器，可以对目标器件进行下载、调试等功能，支持的仿真器也有很多比如 jlink、stlink、DAP 等等。可以作为 GDBserver，进而使用 GDB 进行调试。

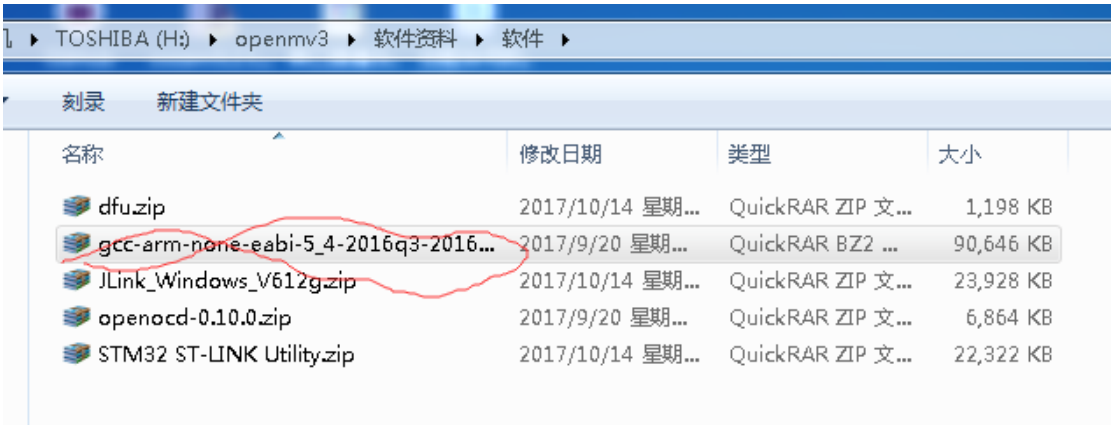
关于交叉编译器的，我选择了 arm-none-eabi-gcc，用于编译 ARM 架构的裸机系统，一般适合 ARM7、Cortex-M 和 Cortex-R 内核的芯片使用。工具链工具比较多，如下图。我们需要比较多的是 arm-none-eabi-gcc（C 语言编译器，将 c 文件转换为中间文件.o）、arm-none-eabi-ld（最后链接所有**.o** 文件生成可执行文件的工具）、arm-none-eabi-objcopy（将生成的文件转化为 bin/hex 等格式）、arm-none-eabi-gdb（调试器，对目标器件进行调试）。

要先安装之前肯定要下载了，当然最简便的方法就是 apt-get，但是有个问题就是可能你 apt-get 的版本有点低，导致会有一些情况发生，比如 jlink 的 SWD 模式不能用，编译源码有错误等错误。所以，毕竟开源，还是乖乖去官网 wget 或者 git 源码进行安装比较好。

可能中间会出现一系列安装问题，其实不用紧张，大部分都是因为缺少依赖包的原因，按照指示百度安装就好了。

arm-none-eabi-gcc 的安装以及编译源码

arm-none-eabi-gcc 的安装相对比较简单，你可以 apt-get install arm-none-eabi-gcc，但是可能内核本身支持的版本不支持源码支持的版本，导致编译的时候会出现错误。所以大部分还是自己获取安装。 本教程安装的版本是 5.4 版本，相对比较高版本了。在网盘里面有，路径在“软件资料”-“软件”上



下载好解压到你想要放置的目录。我是放在主目录下我自己新建的文件上

你可以终端敲击命令解压：

sudo tar -xjvf　gcc-arm-none-eabi-5_4-2016q3-20160926-linux.tar.bz2 -C +你自己想要的安装目录

我所安装的目录如下图

```
root@rcsn-Inspiron-5425:/home/rcsn# cd /home/rcsn/arm_tools/gcc-arm-none-eabi-5_
4-2016q3/
root@rcsn-Inspiron-5425:/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3# ls
arm-none-eabi  bin  lib  share
root@rcsn-Inspiron-5425:/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3# pwd
/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3
root@rcsn-Inspiron-5425:/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3#
```

接下来就是修改环境变量

终端敲击：

sudo nano /etc/profile

在编辑器最下面添加：

export PATH=/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3/bin:$PATH

```
root@rcsn-Inspiron-5425:/home/rcsn# sudo nano /etc/profile
```

```
 GNU nano 2.5.3                文件：  /etc/profile

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

export PATH=/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3/bin:$PATH

^G 求助      ^O Write Out ^W 搜索      ^K 剪切文字  ^J 对齐      ^C 游标位置
```

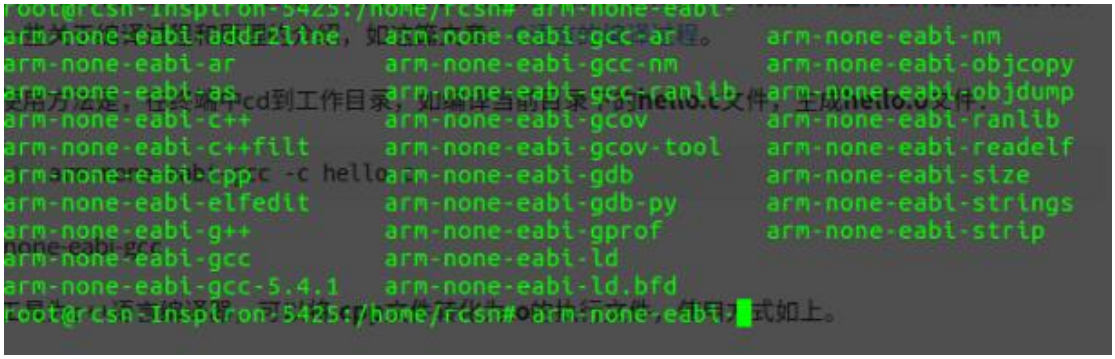
使环境变量生效，然后查看路径添加是否成功,如下图就说明添加成功了。

source /etc/profile

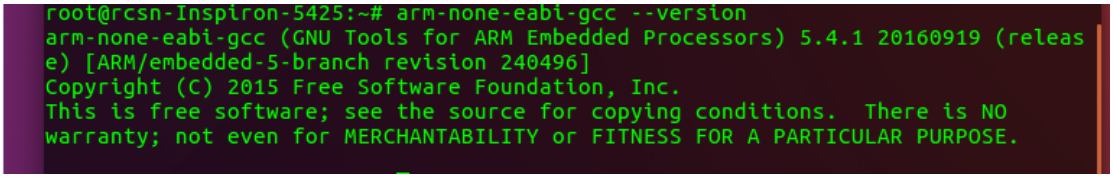
echo $PATH

```
root@rcsn-Inspiron-5425:/home/rcsn# source /etc/profile
root@rcsn-Inspiron-5425:/home/rcsn# echo $PATH
/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3/bin:/usr/local/sbin:/usr/local
/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
root@rcsn-Inspiron-5425:/home/rcsn#
```

接下来我们看下系统能否找到路径，打上不全的命令，比如 arm-none，然后双击 Tab 键，即可弹出以下的信息。说明已经搜索到了。

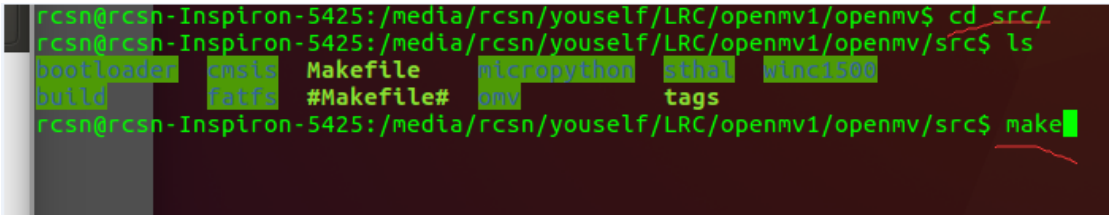


我们可以看下版本，敲击 arm-none-eabi-gcc --version，弹出以下信息版本。



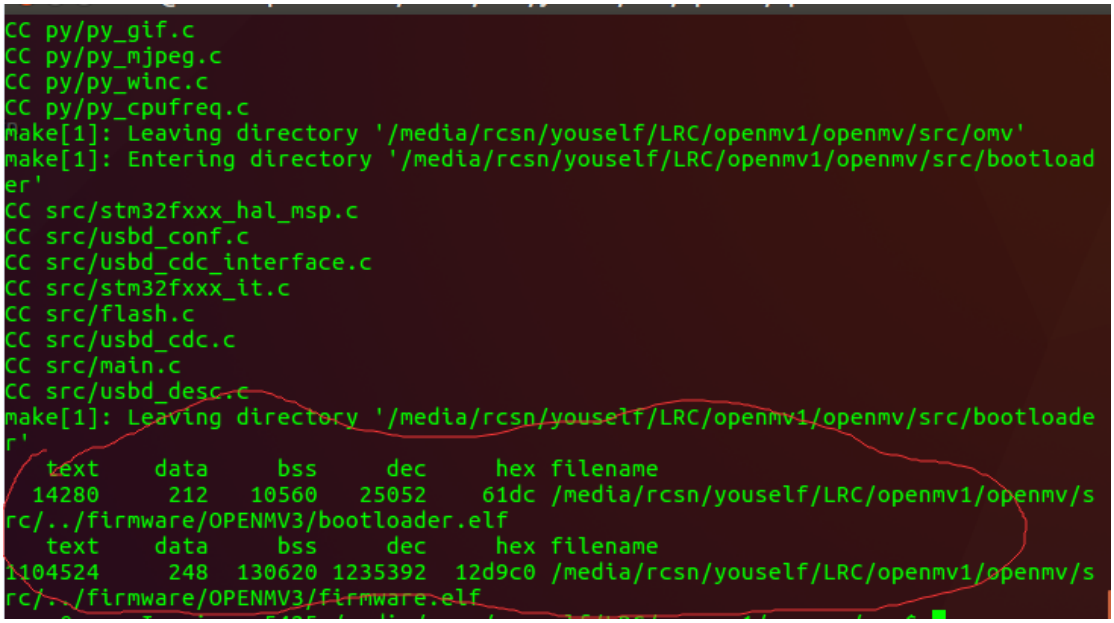
接下来就是编译源码，打开你所在的 Openmv 源码的路径。

然后 cd 命令打开 src 文件夹（主要源码），然后在该目录敲击 make 进行编译



编译中

编译完成生成固件，这时候就编译成功了。



0penocd 的安装配置以及下载固件

对于 openocd 的安装，去官网下载：

https://sourceforge.net/projects/openocd/files/openocd/0.10.0/

也可以在我们的网盘下载，路径在“软件资料”-“软件”上。

下载好先不要着急安装，首先得安装下 openocd 的安装库以及相关依赖包，在终端敲击：

sudo apt-get install autotools-dev make libtool pkg-config autoconf automake texinfo libudev1 libudev-dev libusb-1.0-0-dev libfox-1.6-dev

然后再安装 HID-API 包

终端命令：

cd ~/

git clone https://github.com/signal11/hidapi.git(没有安装 git 就先：sudo apt get install git)

cd hidapi/

./bootstrap

./configure

make

sudo make install

安装成功之后需要将该包的安装位置添加到我们系统 PATH 变量来。

终端命令 ：sudo nano ~/.profile

在脚本最下面添加：PATH="$HOME/bin:/usr/local/lib:$PATH"

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

PATH="/usr/local/lib:$PATH"
```

退出保存，然后在终端敲击命令：echo $PATH 查看变量设置情况。可见已经设置成功。

```
root@rcsn-Inspiron-5425:/home/rcsn# echo $PATH
/home/rcsn/arm_tools/gcc-arm-none-eabi-5_4-2016q3/bin:/usr/local/sbin:/usr/local
/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
root@rcsn-Inspiron-5425:/home/rcsn#
```

最后再敲击命令，使系统共享库更新。

sudo ldconfig

最后我们删除下载的安装包。先回到主目录

cd ~/

sudo rm -r hidapi

接下来就是安装 openocd 的时候了。

终端敲击：

cd ~/

wgethttps://sourceforge.net/projects ... 0.10.0.zip/download

（这个命令是获取安装包但是我们已经有安装包了，我们直接在安装的路径解压即可，这个压缩包放在主目录上）

sudo unzip openocd-0.10.0.zip

cd openocd-0.10.0

./configure --enable-cmsis-dap --enable-jlink --enable-stlink --enable-ti-icdi（使能各类仿真器支持）

make sudo make install

跟上面的一样套路，按照上面的命令安装即可

然后返回主目录，删除安装包

cd ~/

sudo rm -r openocd-0.10.0

然后安装好了之后，就看下版本啦

```
root@rcsn-Inspiron-5425:~# openocd -v
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
root@rcsn-Inspiron-5425:~#
```

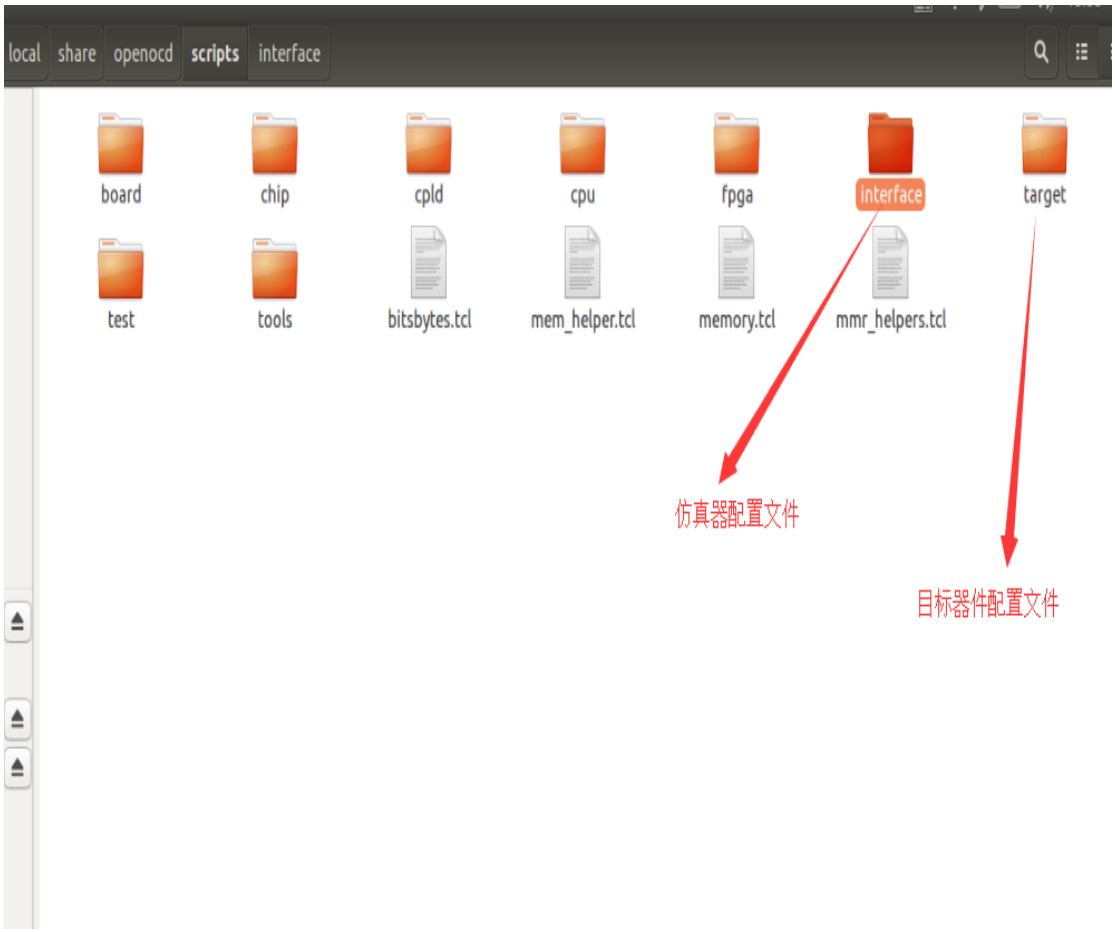
接下来就是要添加 udev 规则了。其实这个规则还很深，其实就是设定一个规则，然后电脑能够识别到设备。

像还不熟悉 Linux 的，要自己写 udev 规则，简直就是还没入门就放弃了。好在 openocd 太人性了，直接就丢给你个规则了，你只要把它复制到相关目录即可。输入以下命令

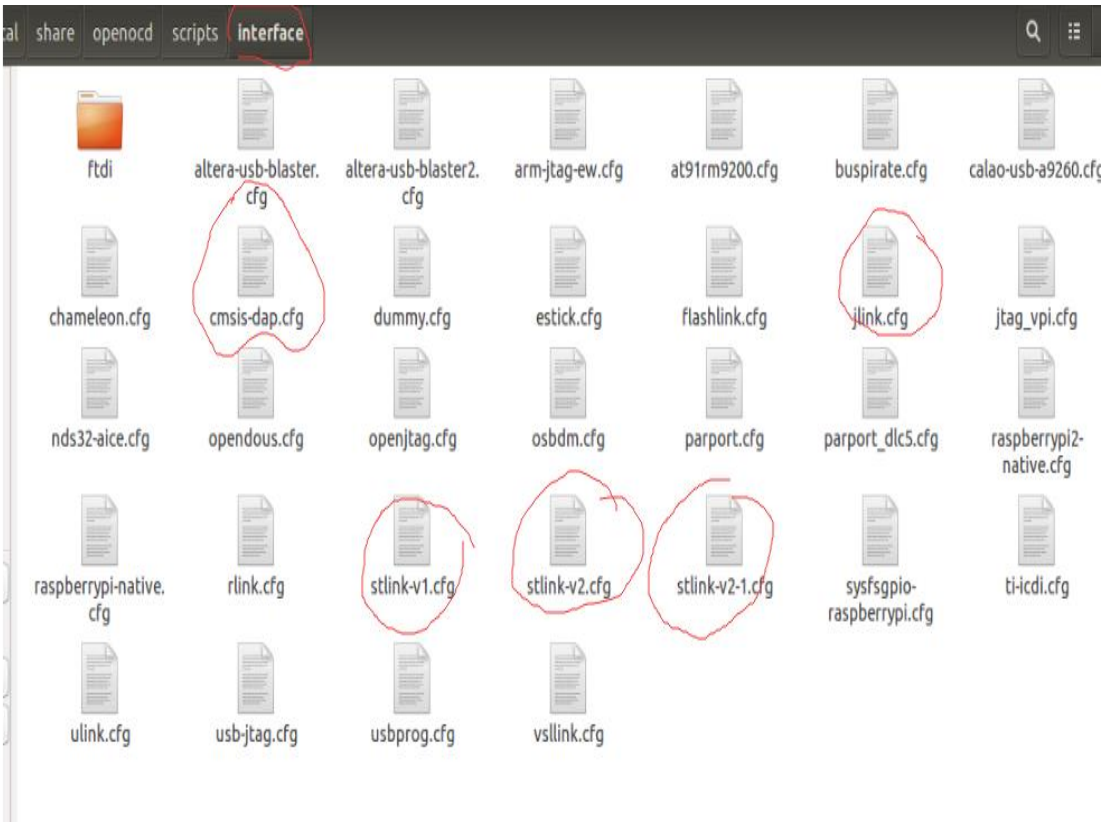
cp /usr/local/share/openocd/contrib/60-openocd.rules /etc/udev/rules.d/

这个 udev 规则文件已经足够用了，起码对入门来说已经够用了。

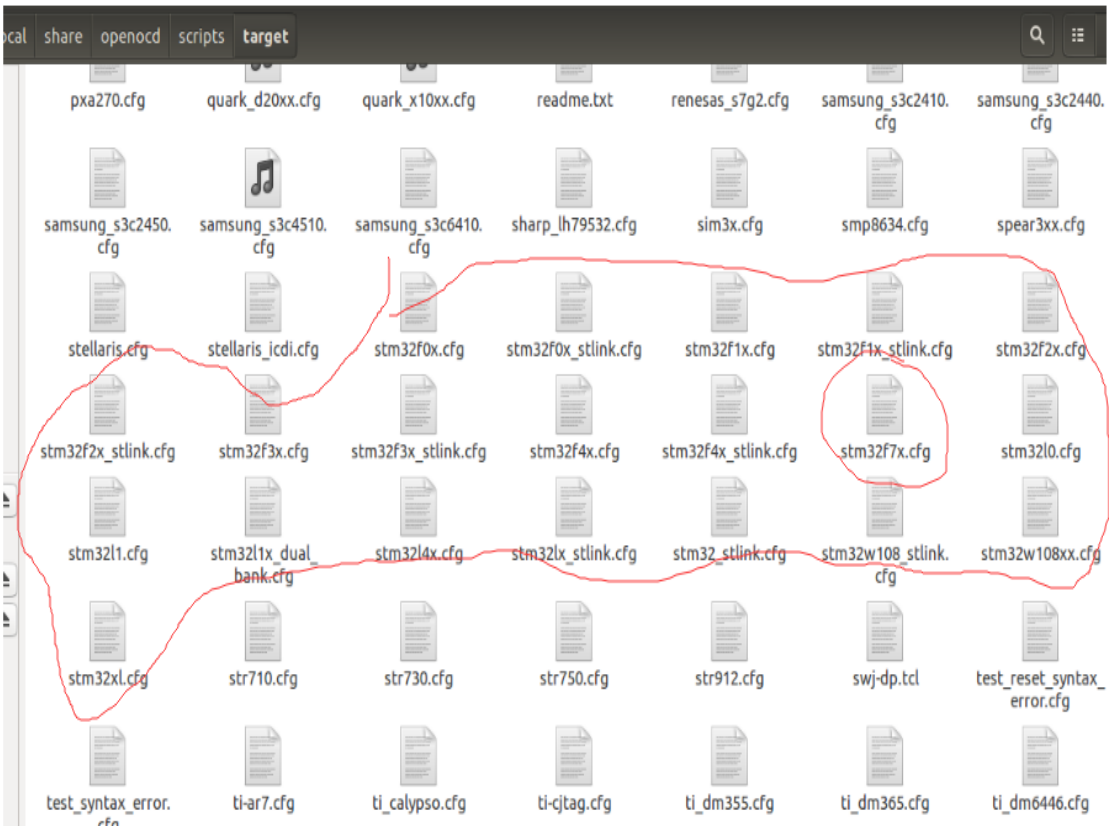
接下来就是链接目标器件的时候了，Openocd 已经把相关配置文件已经弄好了，包括仿真器和目标器件的配置文件



仿真器配置文件有：DAP、Jlink、stlink 仿真器的。



目标器件配置文件支持很多款 ST 的 MCU。如下图。我们 openmv 所用的就是 stm32f7.cfg 文件



然后我们来测试下 openocd，终端敲击

openocd -f interface/jlink.cfg -c "transport select swd" -f target/stm32f7x.cfg

通过终端可以看到链接板子的类型，目标电压，以及断电数。

```
rcsn@rcsn-Inspiron-5425:/media/rcsn/youself/LRC/openmv1/openmv/src$ openocd -f i
nterface/jlink.cfg -c "transport select swd" -f target/stm32f7x.cfg
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
swd
adapter speed: 2000 kHz
adapter_nsrst_delay: 100
srst_only separate srst_nogate srst_open_drain connect_deassert_srst
cortex_m reset_config sysresetreq
Info : No device selected, using first device.
Info : J-Link V9 compiled Jul 24 2017 17:37:57
Info : Hardware version: 9.20
Info : VTarget = 3.266 V
Info : clock speed 2000 kHz
Info : SWD DPIDR 0x5ba02477
Info : stm32f7x.cpu: hardware has 8 breakpoints, 4 watchpoints
```

连接成功后，这个终端不要断开，重新打开一个新终端，然后敲击命令：telnet localhost 4444

然后在终端上敲击命令：

halt（使目标器件进入休眠状态）

Flash write_image erase /...../(源码的路径固件)openmv.bin（清除 flash 并且下载固件）

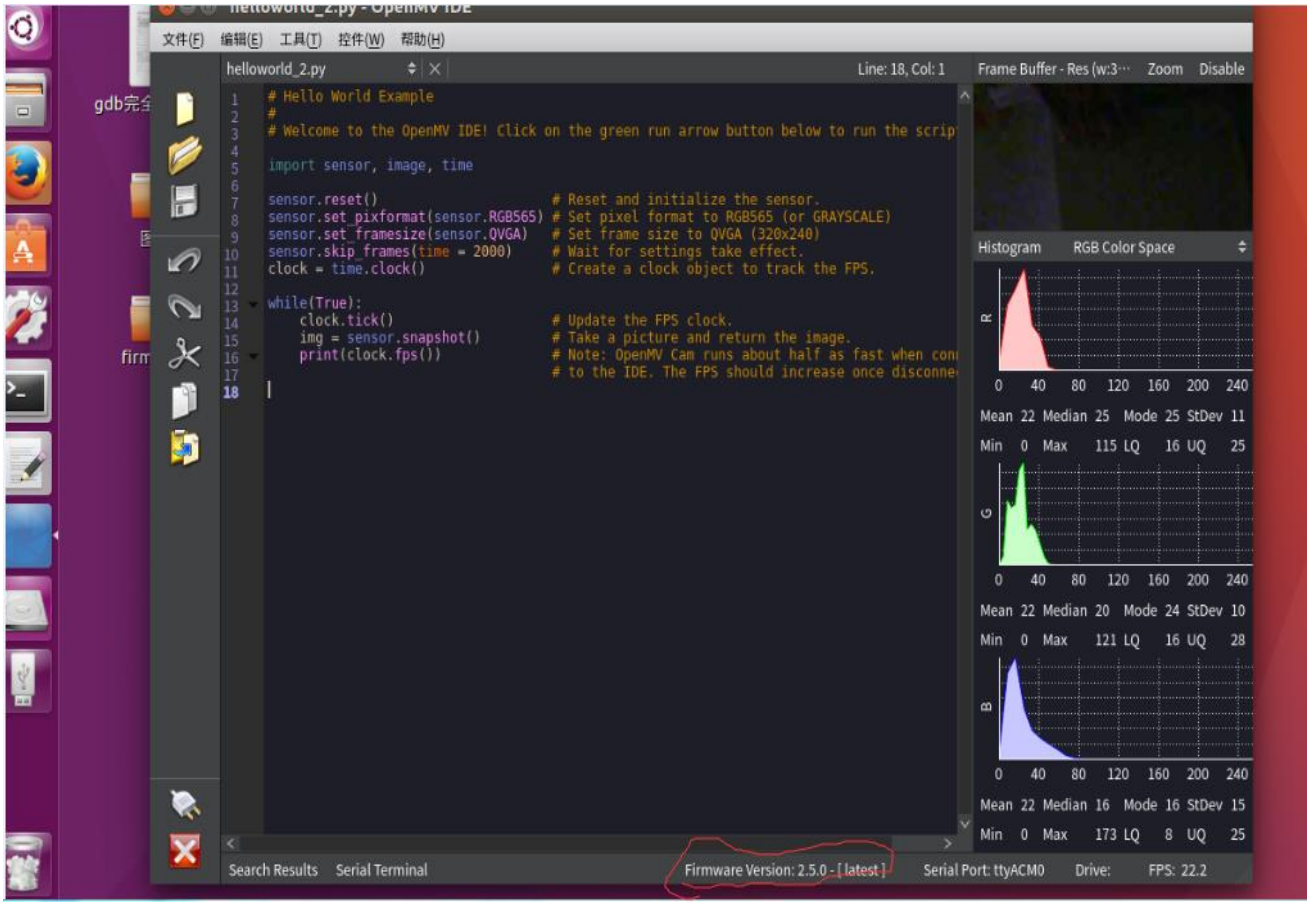
reset（复位目标器件）

下载完成后，若不想继续调试，可以在连接目标器件的终端按下 ctrl+c，结束 openocd。

```
rcsn@rcsn-Inspiron-5425: /media/rcsn/youself/LRC/openmv/openmv-master
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
swd
adapter speed: 2000 kHz
adapter_nsrst_delay: 100
srst_only separate srst_nogate srst_open_drain connect_deassert_srst
cortex_m reset_config sysresetreq
Info : No device selected, using first device.
Info : J-Link V9 compiled Jul 24 2017 17:37:57
Info : Hardware version: 9.20
Info : VTarget = 3.272 V
Info : clock speed 2000 kHz
Info : SWD DPIDR 0x5ba02477
Info : stm32
Info : acce
target halte
xPSR: 0x810
auto erase
Info : devi
Info : flas
Info : Sing
wrote 13107
ENMV3/openm

rcsn@rcsn-Inspiron-5425: ~
rcsn@rcsn-Inspiron-5425:~$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> halt
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x0802a0a0 msp: 0x2001fb18
> flash write_image erase /media/rcsn/youself/LRC/openmv1/openmv/firmware/OPENMV
3/openmv.bin 0x8000000
auto erase enabled
device id = 0x10016451
flash size = 2048kbytes
Single Bank 2048 kiB STM32F76x/77x found
wrote 1310720 bytes from file /media/rcsn/youself/LRC/openmv1/openmv/firmware/OP
ENMV3/openmv.bin in 24.111938s (53.086 KiB/s)
> reset
>
```

打开 IDE，查看固件版本，V2.5.0，下载完成



至此，linux 端编译源码以及下载结束。接下来会编写个，在 Ubuntu 系统利用 openocd+gdb 调试 Openmv 源码，以及修改 openmv 底层源码、