Orange Pi One Plus
User Manual

History

Ver	Data	Author	Brief	Publish	Memo
1.1	2018-05-07	Leeboby	Create document	2018-5-10	

Contents

I. Orange Pi One Plus Introduction

1. What is Orange Pi One Plus?

It's an open-source single-board computer. It can run Android 7.0, Ubuntu, Debian, etc. It uses AllWinner H6 SOC and has 1GB LPDDR3 SD RAM.

2. What can I do with Orange Pi One Plus?

You can use it to build…

- A computer
- A wireless server
- Games
- Music and sounds
- HD video
- A speaker
- Android
- Scratch
- .....

Pretty much anything else, because Orange Pi One Plus is open source

3. Whom is it for?

Orange Pi One Plus is for anyone who wants to create with technology– not just consuming. It's a simple, fun, useful tool and you can use it to take control of the world around you.

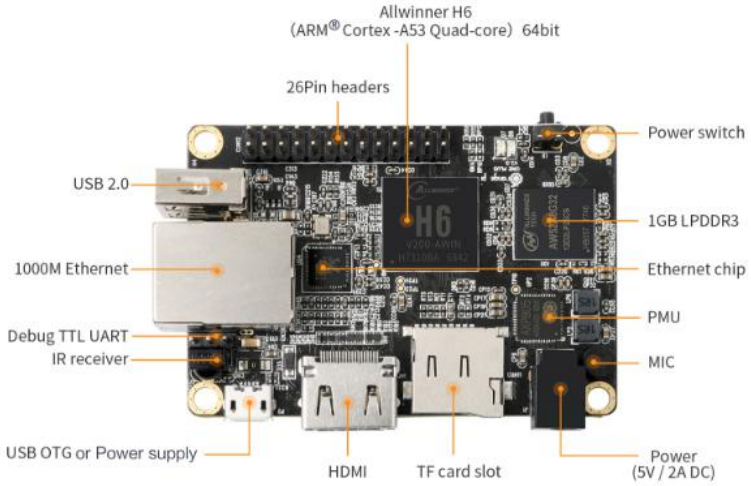
4. Hardware specification of Orange Pi One Plus

Hardware specification	
CPU	H6 Quad-core 64-bit ARM Cortex™-A53
GPU	• High-performance multi-core GPU Mali T720 • OpenGL ES3.1/3.0/2.0/1.1 • Microsoft DirectX 11 FL9_3

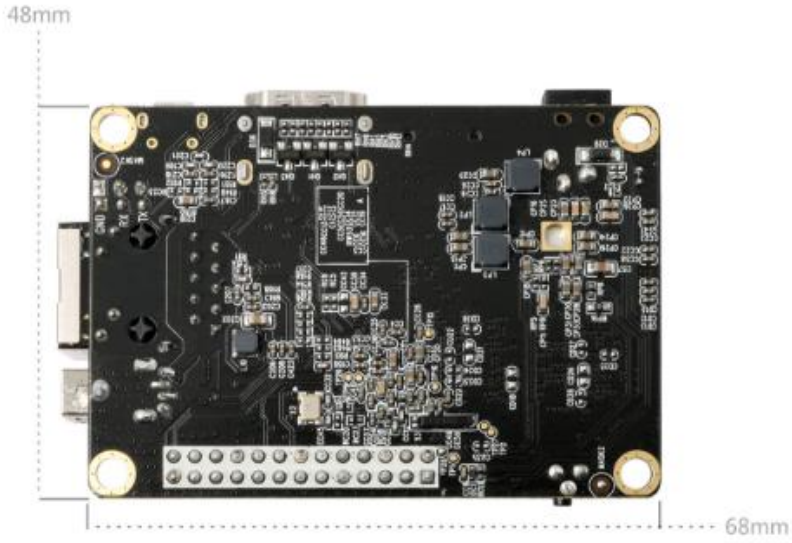
	• ASTC(Adaptive Scalable Texture Compression) • Floating point operation greater than 70 GFLOPS
Memory (SDRAM)	1GB LPDDR3 (shared with GPU)
On-board Storage	TF card (Max. 32GB) /MMC card slot
On-board Network	10/100M/1000M Ethernet RJ45
Network Chip	RTL8211
Audio Input	MIC
Video Outputs	HDMI 2.0a TX with HDCP 2.2 output
Video Decoding	• H265/HEVC Main/Main10 profile@Level5.2 High-tier; 4K@60fps, up to 6Kx4K@30fps • H264/AVC BP/MP/HP@level5.1, MVC, 4K@30fps • VP9，Profile 0/2, 4K@30fps • AVS+/AVS JIZHUN profile@level 6.0, 1080P@60fps
Audio Output	HDMI 2.0a
Power Source	DC input, USB OTG input can supply power
PMU	AXP805
USB 2.0 Ports	One USB 2.0 Host, one Micro USB 2.0
Low-level peripherals	26 Pins Header
GPIO(1x3) pin	UART, ground.
LED	Power led & Status led
IR Receiver	Yes
Key	Power(K1)
Supported OS	Android, Ubuntu, Debian
Interface definition	
Product size	69mm × 48mm

Weight	45g
Orange Pi™ is a trademark of the Shenzhen Xunlong Software CO., Limited	

Top view



Bottom view



5. GPIO Specifications

The picture below is GPIO pin definition of Orange Pi One Plus:

Orange Pi One Plus GPIO definition		
CON12-P01	VCC-3.3V	VCC-IO
CON12-P02	VCC-5V	DCIN
CON12-P03	TWI1-SDA	PH06
CON12-P04	VCC-5V	DCIN
CON12-P05	TWI1-SCK	PH05
CON12-P06	GND	GND
CON12-P07	PWM1	PH04
CON12-P08	PD21	PD21
CON12-P09	GND	GND
CON12-P10	PD22	PD22
CON12-P11	UART3_RX	PD24
CON12-P12	PC09	PC09
CON12-P13	UART3_TX	PD23
CON12-P14	GND	GND
CON12-P15	UART3_CTS	PD26
CON12-P16	PC08	PC08
CON12-P17	VCC-3V3	VCC-IO
CON12-P18	PC07	PC07
CON12-P19	SPI0_MOSI	PC02
CON12-P20	GND	GND
CON12-P21	SPI0_MISO	PC03
CON12-P22	UART3_RTS	PD25
CON12-P23	SPI0_CLK	PC00
CON12-P24	SPI0_CS0	PC05
CON12-P25	GND	GND
CON12-P26	PH03	PH03

II. Using Method Introduction

1. Hardware Requirement

- Orange Pi One Plus Development Board
- TF card in min 8GB and class 10, recommend using a famous brand TF card, such as Sandisk 16GB TF card
- A PC for compilation with following specs：
  64 bit CPU
  Up to 8GB RAM
  Up to 100GB spare disk space
  Operation system at **Ubuntu14.04** would be better

2. Software Requirement:

- Orange Pi One Plus SDK
- Orange Pi One Plus Firmware
- Android and Linux flash tool

3. Power Supply Requirement

There are two methods for power supply:

- DC (5V 2A) in for power
- Micro USB(5V 2A)OTG in for power

III. Android Compilation Environment Construction

The following operations are based on Ubuntu 14.04, there might be some difference if you are working with Ubuntu or Linux distro.

1. Download SDK compression package

After download Android SDK, you need to first packed several compressed files into one package, then unzip the package just packed.

```
$ mkdir OrangePi_OnePlus
$ cat H6-2018-1-2.tar.gza* > OrangePi_OnePlus.tar
$ tar xf OrangePi_OnePlus.tar -C OrangePi_OnePlus
```

2. Construct Compilation Environment

● Install JDK

It could only works on version openjdk8 for Android 7.0 compilation, it would cause failure if the version not openjdk8. Commands for installing Openjdk-8 are as follows:

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk
```

● Configure environment variable of JAVA

If the installation path like this: **/usr/lib/jvm/java-8-openjdk-amd64**, then you could operate the following command in the terminal:

```
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
$ export PATH=$JAVA_HOME/bin:$PATH
$ export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

● Install Software Package

For Ubuntu14.04：

```
$ sudo apt-get update
$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip

$ sudo apt-get install u-boot-tools
```

3. Compilation of SDK Source Code

There would be two directories under android and lichee after unzipped SDK package, contents on lichee are as following:

```
lichee/brandy/u-boot-2014.07                    #uboot code directory
lichee/bootloader/uboot_2014_sunxi_spl          #boot0 code directory
lichee/linux-3.10                               #Kernel code
lichee/tools                                    #Hardware specs, packing tools, etc.
```

- **Kernel compile steps**

Input the following commands on lichee directory:

```
$ cd OrangePi_OnePlus/lichee
$ ./build.sh config

Welcome to mkscript setup progress
All available chips:
    0. sun50iw1p1
    1. sun50iw2p1
    2. sun50iw6p1
    3. sun8iw11p1
    4. sun8iw12p1
    5. sun8iw6p1
    6. sun8iw7p1
    7. sun8iw8p1
    8. sun9iw1p1
Choice: 2
All available platforms:
    0. android
```

```
    1. dragonboard
    2. linux
    3. eyeseelinux
Choice: 0
All available business:
    0. 5.1
    1. 4.4
    2. 7.x
Choice: 2
```

After the above compilation, you will get the following output:

```
regenerate rootfs cpio
15757 blocks
17099 blocks
build_ramfs
Copy boot.img to output directory ...
Copy modules to target ...

sun50iw6p1 compile Kernel successful

INFO: build kernel OK.

INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
----------------------------------------
build sun50iw6p1 android 7.x lichee OK
----------------------------------------
```

Kernel code is on the directory of lichee/linux-3.10.

The system would copy the configure file from lichee/linux-3.10/arch/arm64/configs/sun50iw6p1smp_android_7.x_defconfig to lichee/linux-3.10/.config as default configure when the input compilation command.

In the next compilation, you could run ./build.sh on lichee directory and go on with last .config configure.

● uboot/boot0 Configure Steps

Usually there is no need to re-compile uboot, if you want to make some modification, then please refer

to the following:

```
cd lichee/brandy/u-boot-2014.07
make distclean && make sun50iw6p1_config && make -j5    #Compile uboot

cd lichee/brandy/u-boot-2014.07
make distclean && make sun50iw6p1_config && make spl    #Compile boot0
```

If do not compile uboot/boot0, it is default running with lichee/tools/pack/chips/sun50iw6p1/bin. It would be replaced the default command if recompiled with the above command.

- **Building Android from source code**

```
$ cd android
$ source ./build/envsetup.sh
$ lunch petrel_fvd_p1-eng
$ extract-bsp
$ make -j8 && pack
```

The pack command is to pack it into firmware, if it is packed successfully, then there would be the following information:

```
Dragon execute image.cfg SUCCESS !
----------image is at----------

OrangePi_OnePlus/lichee/tools/pack/sun50iw6p1_android_petrel-p1_uart0.img

pack finish
```

According to the above prompt, you could check the generated firmware of sun50iw6p1_android_petrel-p1_uart0.img on the directory of OrangePi_OnePlus/lichee/tools/pack/. About Android image flashing, you could refer to the section of Android image flashing.

IV. Linux Environment Construction

1. Download SDK compression package

● Download OrangePi_Build

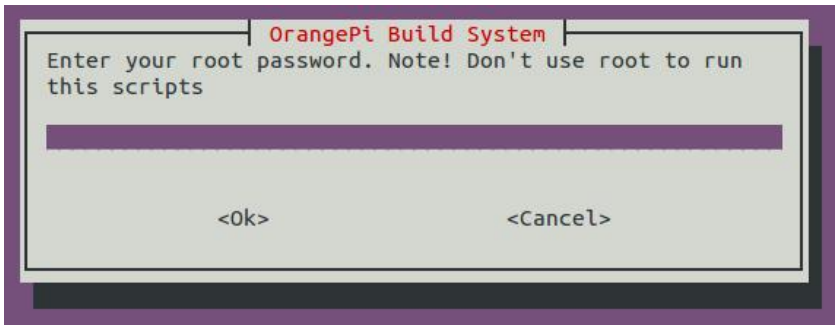
OrangePi_Build.You can download Linux SDK from Github by OrangePi_Build. The following is the way to get the OrangePi_Build.

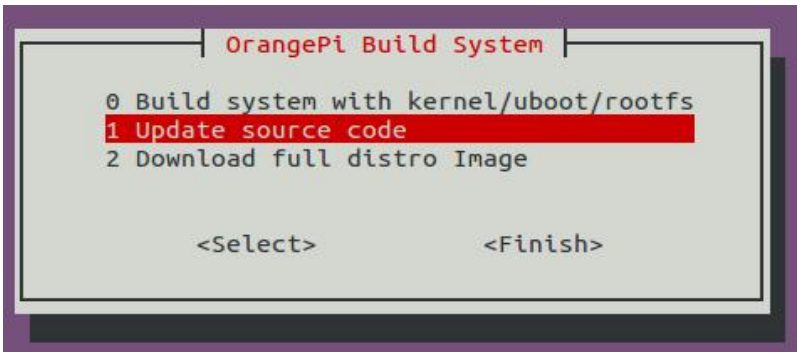
```
$ sudo apt-get install git
$ git clone https://github.com/orangepi-xunlong/OrangePi_Build.git
$ cd OrangePi_Build
$ ls
Build_OrangePi.sh    lib    README.md
```

● Run OrangePi_Build

```
$ ./Build_OrangePi.sh
```

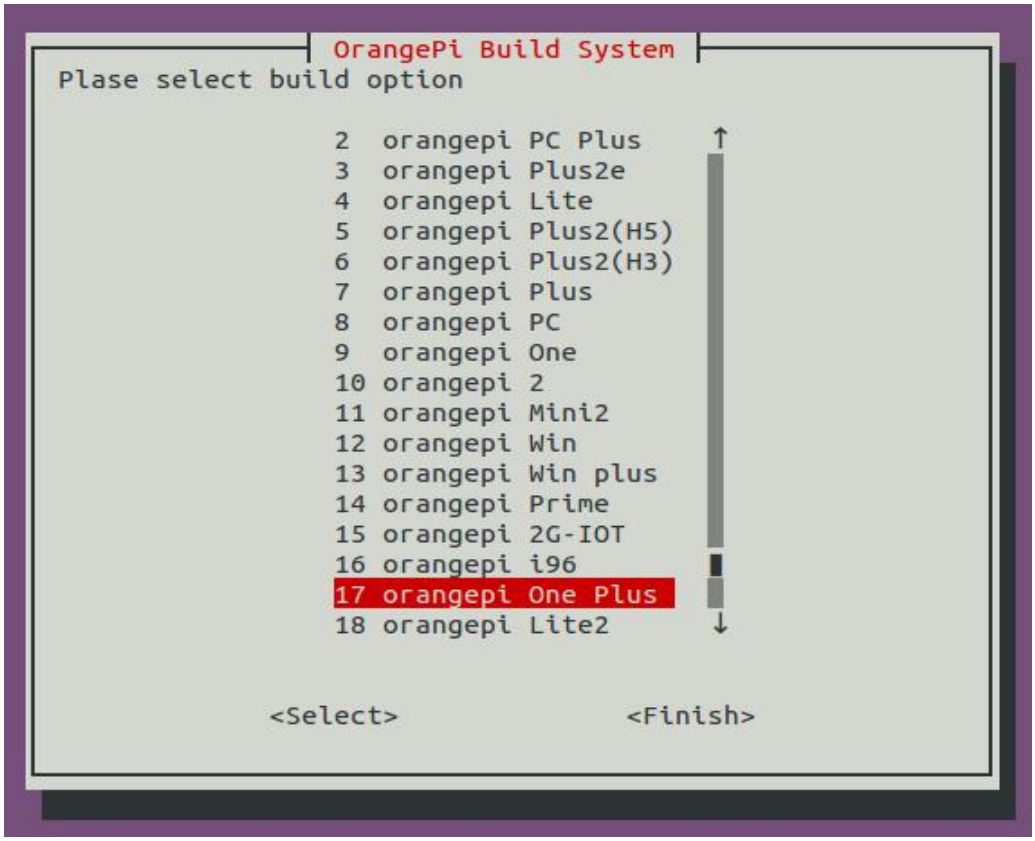
Input password of root, then enter to next step:





Select 1 Update source code, enter into interface of models selection

Select **17 orangepi One Plus**, enter into Linux Orange Pi One Plus downloading



The downloaded SDK will store at the same lever directory of OrangePi_Build

```
$ ls ../OrangePi_Build
  OrangePi_Build OrangePiH6
```

2. Construct Compilation Environment

Linux directory construction of OrangePi H6:

```
$ cd OrangePiH6
$ tree -L 1
.
├── build.sh -> scripts/build.s      Compile boot scripts
├── external                         Store external configure file
├── kernel                           Linux kernel source code
├── output                           Store output files
├── scripts                          Scripts file used in the compilation
├── toolchain                  Cross compiler tool chain used by the kernel and u-boot
└── uboot                            Store source code of boot0 and u-boot

6 directories, 1 file
```

The directory structure of cross compile tool chain is shown below. If the source code is different from it, please download the source code again.

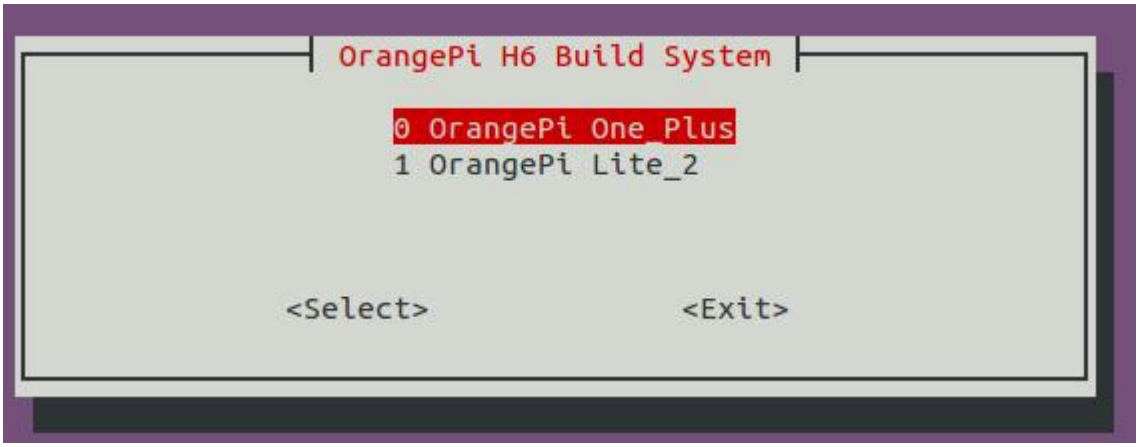
```
$ cd toolchain
$ tree -L 2
.
└── gcc-linaro-aarch
    ├── aarch64-linux-gnu
    ├── bin
    ├── gcc-linaro                    Store tool train for u-boot compile
    ├── gcc-linaro-4.9-4.9-2015.01-3-2015.02-3-manifest.txt
    ├── include
    ├── lib
    ├── libexec
    └── share

8 directories, 1 file
```

3. Configure Linux and U-boot Source Code

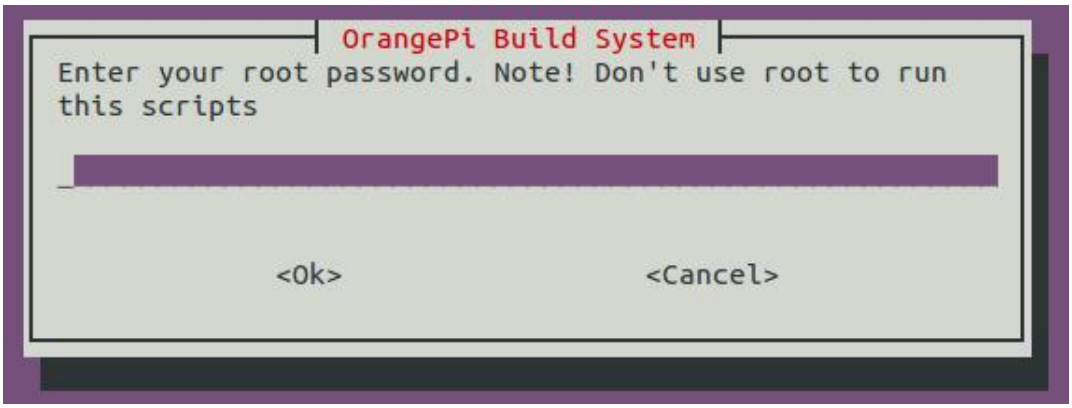
- **Execute compile-booting scripts**

```
$ cd OrangePiH6
$ ./build.sh
```

Select **0 OrangePi One_Plus** and Enter

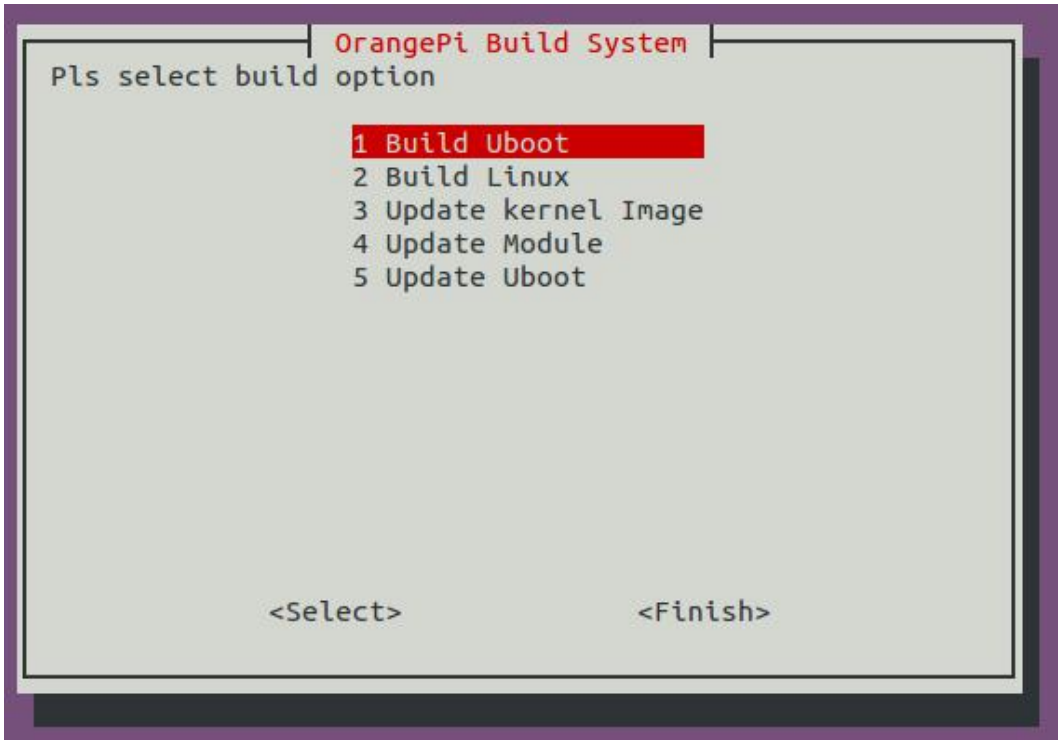


Input password of root and Enter, the select the function you want to run



Here are explanations and functions of the selections:

- **1 Build Uboot —— Compile U-boot and boot0 source code**
- **2 Build Linux —— Compile kernel source code**
- **3 Update Kernel Image —— Update kernel of Linux on SD Card and OrangePiH6.dtb file**
- **4 Update Module —— Update kernel module of Linux on SD card**
- **5 Update Uboot —— Update boot0 and U-boo of Linux on SD card**



You need to first compile kernel source code before compile U-boot, otherwise the dtc program maybe cannot find.

The generated file will store on output directory:

```
$ cd output
$ tree -L 2
```

```
.
├── boot0.bin
├── initrd.img
├── lib
│   └── modules
├── OrangePiH6.dtb
├── pack
│   └── out
├── uboot.bin
├── uEnv.txt
└── uImage

4 directories, 6 files
```

4. Linux SDK Usage Sample

We will make an example of Linux SDK usage with adding **rtl8812AU** USB WIFI kernel module on kernel source code.

- **Download source code of rtl8812AU from github**

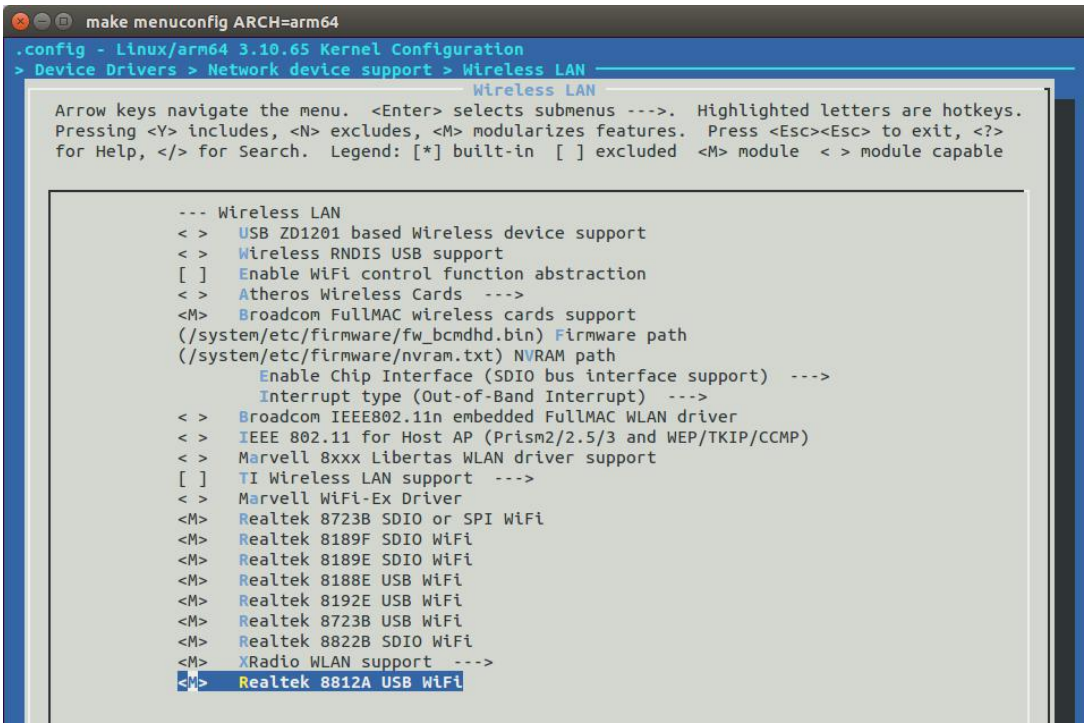
```
$ cd OrangePiH6/kernel/drivers/net/wireless
$ git clone https://github.com/diederikdehaas/rtl8812AU.git
Cloning into 'rtl8812AU'...
remote: Counting objects: 2347, done.
Receiving objects: 100% (2347/2347), 7.87 MiB | 22.00 KiB/s, done.
Resolving deltas: 100% (1292/1292), done.
Checking connectivity... done.
```

- **Add rtl8812AU compilation**

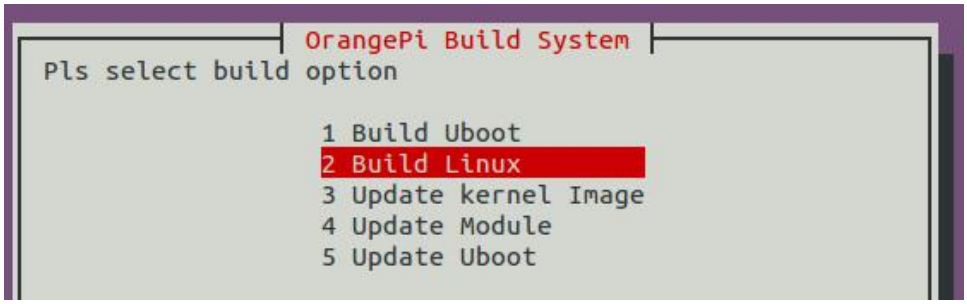
```
$ cd OrangePiH6/kernel/drivers/net/wireless
$ git diff .
diff --git a/drivers/net/wireless/Kconfig b/drivers/net/wireless/Kconfig
index 373666b..b7ebd5c 100755
--- a/drivers/net/wireless/Kconfig
+++ b/drivers/net/wireless/Kconfig
@@ -294,4 +294,5 @@ source "drivers/net/wireless/rtl8192eu/Kconfig"
+source "drivers/net/wireless/rtl8812AU/Kconfig"
  endif # WLAN

diff --git a/drivers/net/wireless/Makefile b/drivers/net/wireless/Makefile
```

```
index fd8a466..3aef800 100755
--- a/drivers/net/wireless/Makefile
+++ b/drivers/net/wireless/Makefile
@@ -66,3 +66,4 @@ obj-$(CONFIG_XR_WLAN) += xradio/
+obj-$(CONFIG_RTL8812AU)             += rtl8812AU/
```

- **Select Realtek 8812A USB WiFi on kernel configure, and compile into kernel module**



- **You could recompile kernel according to the section of configure Linux and U-boot source code on Linux Environment Construction**



Part of Log would show like the following:

```
Start Compile.....
Start Compile Module
  CC [M]    drivers/net/wireless/rtl8812AU/core/rtw_cmd.o
```

```
  CC [M]  drivers/net/wireless/rtl8812AU/core/rtw_security.o
  CC [M]  drivers/net/wireless/rtl8812AU/core/rtw_debug.o
  CC [M]  drivers/net/wireless/rtl8812AU/core/rtw_io.o
  CC [M]  drivers/net/wireless/rtl8812AU/core/rtw_ioctl_query.o
  CC [M]  drivers/net/wireless/rtl8812AU/core/rtw_ioctl_set.o
```

The compiled module would be listed on

output/lib/modules/3.10.65+/kernel/drivers/net/wireless/rtl8812AU after compiled.

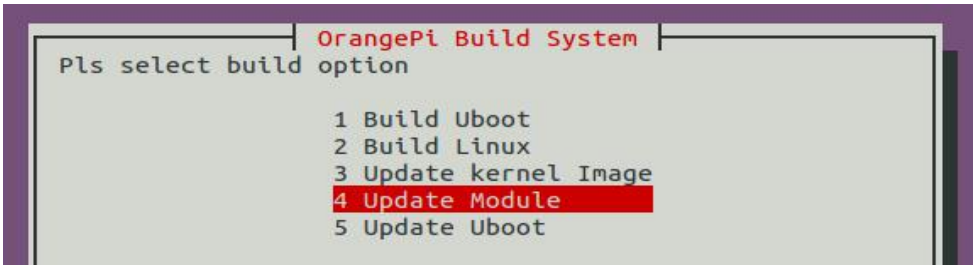
```
$ cd output/lib/modules/3.10.65+/kernel/drivers/net/wireless/rtl8812AU
$ ls
8812au.ko
```

● Update kernel module

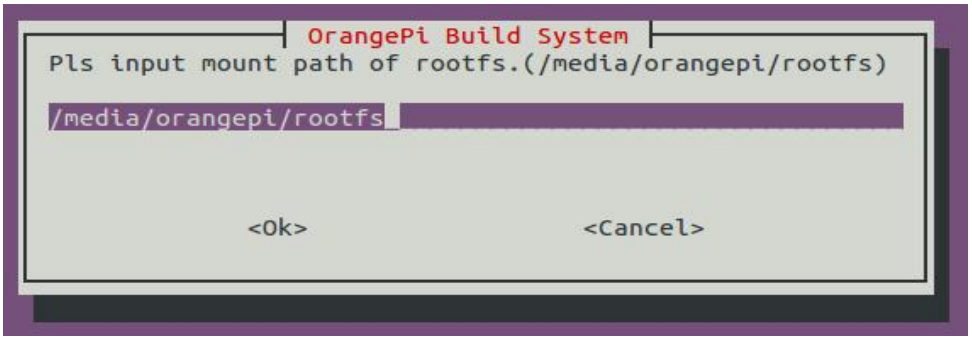
Insert SD card with Linux firmware into PC(installed Ubuntu 14.04 virtual or virtual PC), when the system recognized and mounted SD card, you could check corresponding partition name on **/media/$LOGNAME**

```
$ cd /media/$LOGNAME
$ ls
BOOT            Store kernel and OrangePiH6.dtb file
rootfs          Rootfs file system
```

Refer to the section of compile Linux and U-boot on Linux environment construction, then select **4 Update Module** to update kernel module.



Enter path of root file partition, and Enter, the scripts will copy kernel module into SD card auto.



Boot the board with SD card with USB WIFI card driver of 8812au.ko kernel module.

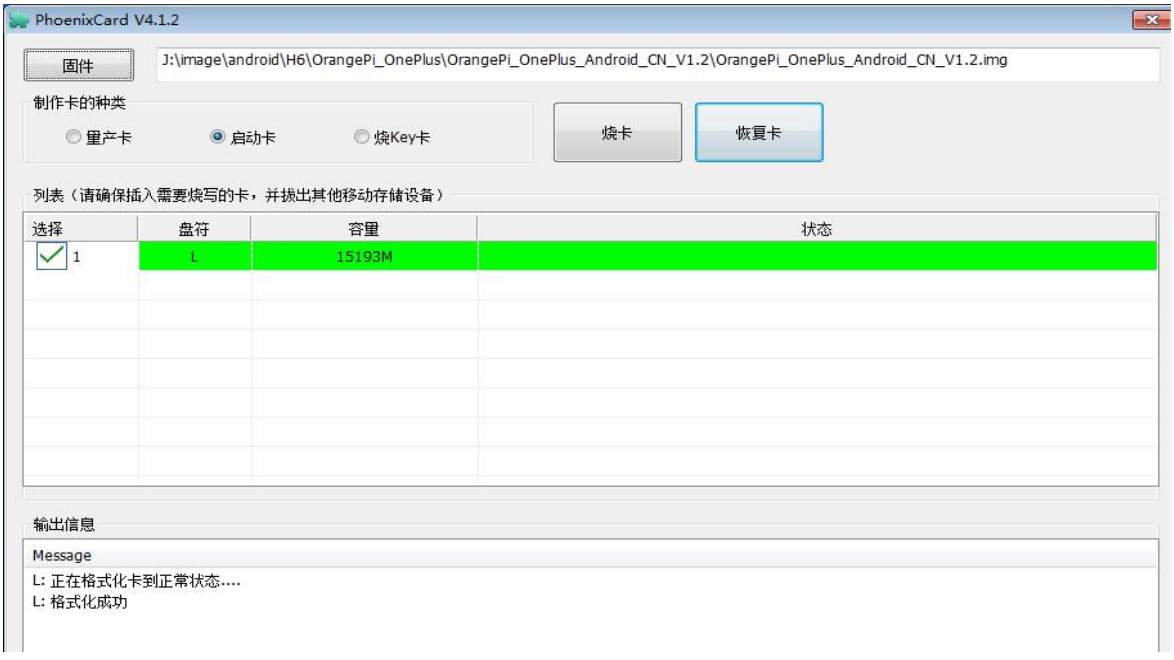
V. Android Firmware Flash

Android firmware **cannot** be flashed into SD card with dd command or writing with Win32 Diskimager on Windows. PhoenixCard card could be used for Android firmware flash. The latest version of PhoenixCard is **PhoenixCard V4.1.2.**

Steps for Android Firmware Flashing

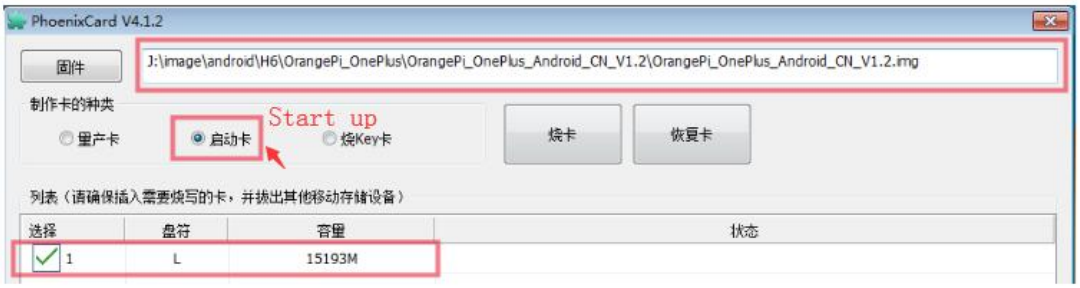
- **Format TF Card**

Check whether card disk as same for TF card and selected disk, click restore to format SD card

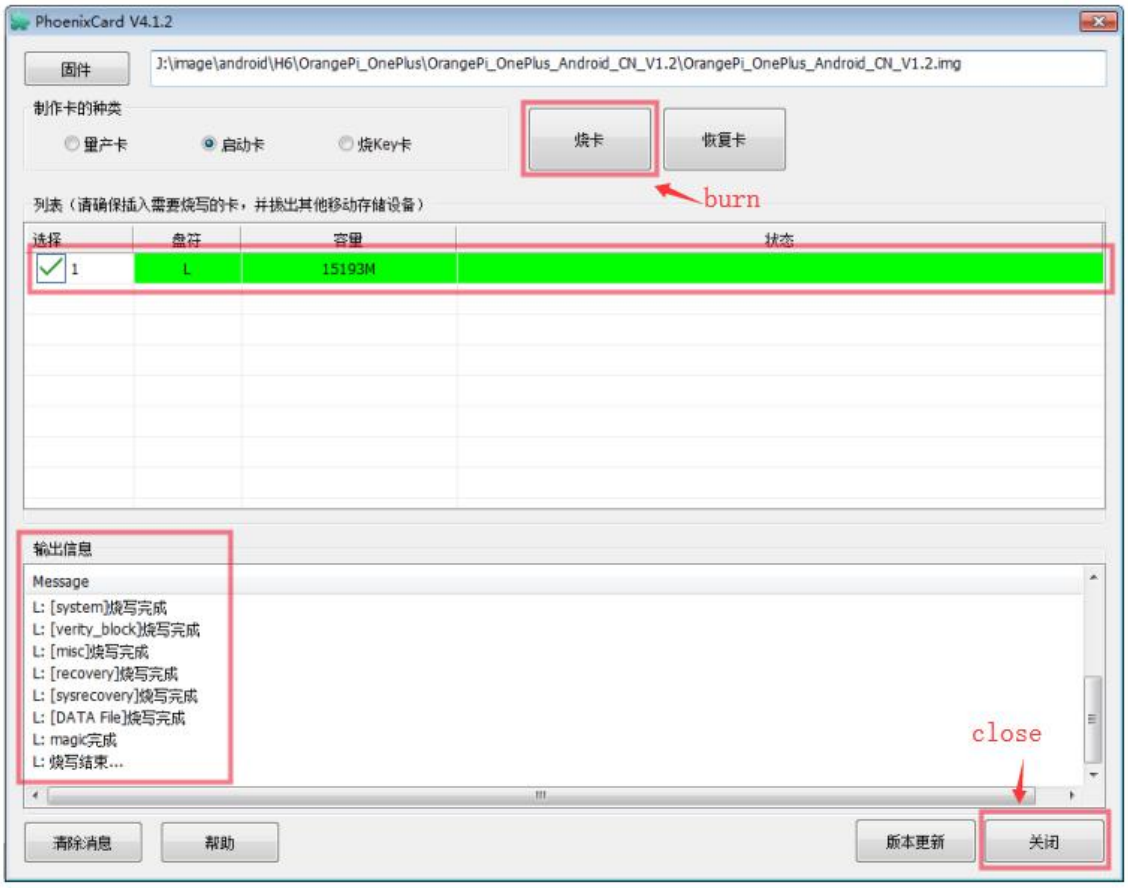


- **Choose Firmware and select Start up**

Please note the following picture in RED notes:



- **Click burn enter into image flashing into SD card**



After finished Android flash, click close and you could use this SD card with written image to boot.

VI. Linux Firmware Flash

We could use **Etcher** to write Linux Firmware into TF card and boot the Orange Pi One Plus with TF card. Etcher supports to work at the following OS:

- Linux(Most distro versions, such as Ubuntu)
- MacOS 10.9 or higher version
- Windows 7 or higher version

Etcher package could be downloaded from their website(**https://etcher.io/**) or Orange Pi official website.

1. Install Etcher

- Install Etcher on Windows OS is same like installing other software(such as PhoneixCard)
- Install Etcher on Ubuntu and Debian OS:

1. Add Etcher Debian library：

$ echo "deb https://dl.bintray.com/resin-io/debian stable etcher" | sudo tee /etc/apt/sources.list.d/etcher.list

2. Download key

$ sudo apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys 379CE192D401AB61

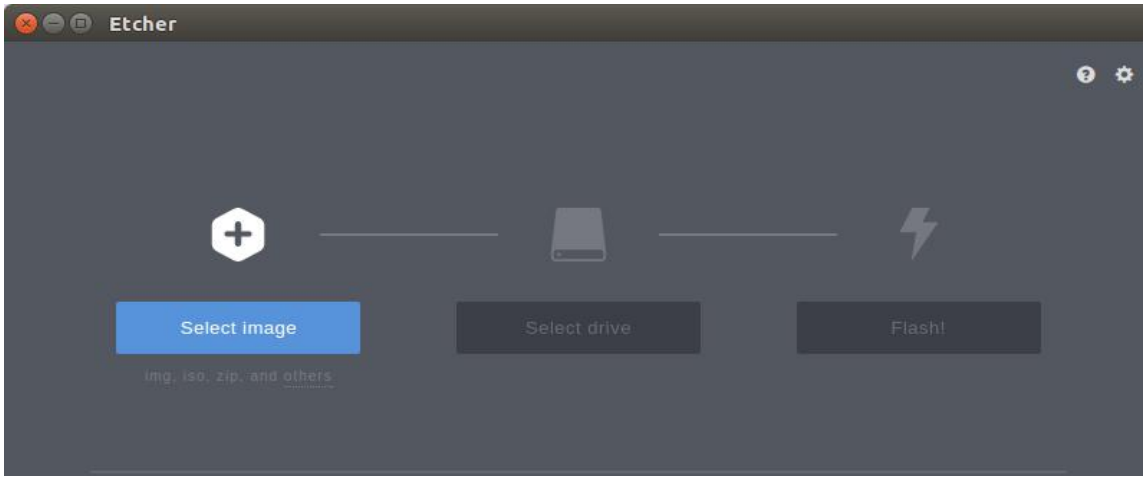
3. Update and install

$ sudo apt-get update && sudo apt-get install etcher-electron

4. Upload
$ sudo apt-get remove etcher-electron
$ sudo rm /etc/apt/sources.list.d/etcher.list && sudo apt-get update

2. Flash Linux Firmware via Etcher

- Open Etcher like the following interface



- "Select image", select Linux firmware you are going to flash
- Insert TF card, Etcher will recognize corresponding driver auto
- Click "Flash！" , after finished, boot the board with inserting the TF card

VII. Linux System Usage

1. Reflect when booting with Linux

- Once the board boot, the RED LED would light up first, then RED LED off, then YELLOW LED keep lighting
- The network LED would not light up if do not insert Ethernet Cable
- After powered the board, insert Ethernet Cable, the network LED would light on, then the network LED would light off around 2s and re-light on.

2. Login account and password

- User name: root, password: orangepi
- User name: orangepi, password:orangepi

3. Expand rootfs partition

You could expand roots partition once you made SD card with firmware. It could enhance the performance of the system.

We could use the built-in scripts of resize_rootfs.sh to expansion after enter into system

The size of the available space before expansion of the system

```
root@OrangePi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  1.1G  520M  488M  52% /
devtmpfs        481M     0  481M   0% /dev
tmpfs           489M     0  489M   0% /dev/shm
tmpfs           489M  6.6M  483M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           489M     0  489M   0% /sys/fs/cgroup
/dev/mmcblk0p1   50M   29M   22M  58% /boot
```

The built-in expansion script of running system

```
root@OrangePi:~# OrangePi_Resize_rootfs.sh
```

The size of the available space after expansion of the system

```
root@OrangePi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
```

```
/dev/mmcblk0p2   7.2G   539M   6.4G    8% /
devtmpfs         481M      0   481M    0% /dev
tmpfs            489M      0   489M    0% /dev/shm
tmpfs            489M     13M   477M    3% /run
tmpfs            5.0M   4.0K   5.0M    1% /run/lock
tmpfs            489M      0   489M    0% /sys/fs/cgroup
/dev/mmcblk0p1    50M     29M    22M   58% /boot
```

4. Record and Play Sound

On platform of H6, it is default use AHUB with not standard driver of ALSA. At present test, we only test with tinyalsa for record and play sound. Source code of tinyalsa tool already uploaded on GitHub. You could refer to the following for usage:

● Download tinyalsa source code

Download tinyalsa test tool from Github before enter into Linux system on Orange Pi One Plus.

```
$ sudo apt-get udpate
$ sudo apt-get install -y git make gcc
$ git clone https://github.com/orangepi-xunlong/OrangePiH6_Tinyal
Cloning into 'OrangePiH6_Tinyalsa'...
remote: Counting objects: 21, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 21 (delta 3), reused 21 (delta 3), pack-reused 0
Unpacking objects: 100% (21/21), done.
Checking connectivity... done.
```

● Compile tinyalsa tool

```
$ cd OrangePiH6_Tinyalsa
$ ./build.sh

File could be executed is store on out/tinyalsa-arm64
$ cd out/tinyalsa-arm64
$ ls
libtinyalsa.so  tinycap_ahub  tinypcminfo  tinyplay_ahub
tinycap         tinymix       tinyplay
```

● Export path of tinyalsa Shared library

```
$ cd out/tinyalsa-arm64
$ export LD_LIBRARY_PATH=`pwd`
```

Tinyalsa tool usage

Check device node

```
$ cat /proc/asound/cards
 0 [sndahub            ]: sndahub - sndahub
                          sndahub
 1 [sndhdmi            ]: sndhdmi - sndhdmi
                          sndhdmi
 2 [snddaudio2         ]: snddaudio2 - snddaudio2
                          snddaudio2
 3 [sndacx00codec      ]: sndacx00-codec - sndacx00-codec
                          sndacx00-codec
```

Test record function

1. Check device node, codec is 5
2. Use tinymix program to connect i2s3 TX（codec）and APBIF0 RX, open i2s3 in with following command

   $ cd out/tinyalsa-arm64
   $ ./tinymix 13 7
   $ ./tinymix 20 1
3. Execute the following command and open codec in control

   $ ./tinymix -D 3 13 1
   $ ./tinymix -D 3 15 1
   $ ./tinymix -D 3 20 1
   $ ./tinymix -D 3 27 1
4. Record command

   $ ./tincap test.wav -D 0 -d 0 -t 10

Test HDMI play sound function

1. Check device node, HDMI is 1
2. Use tinymix program to connect i2s1 and APBIF_TXDIF0,open i2s1 out

   $ ./tinymix 9 1
   $ ./tinymix 17 1
3. HDMI play command

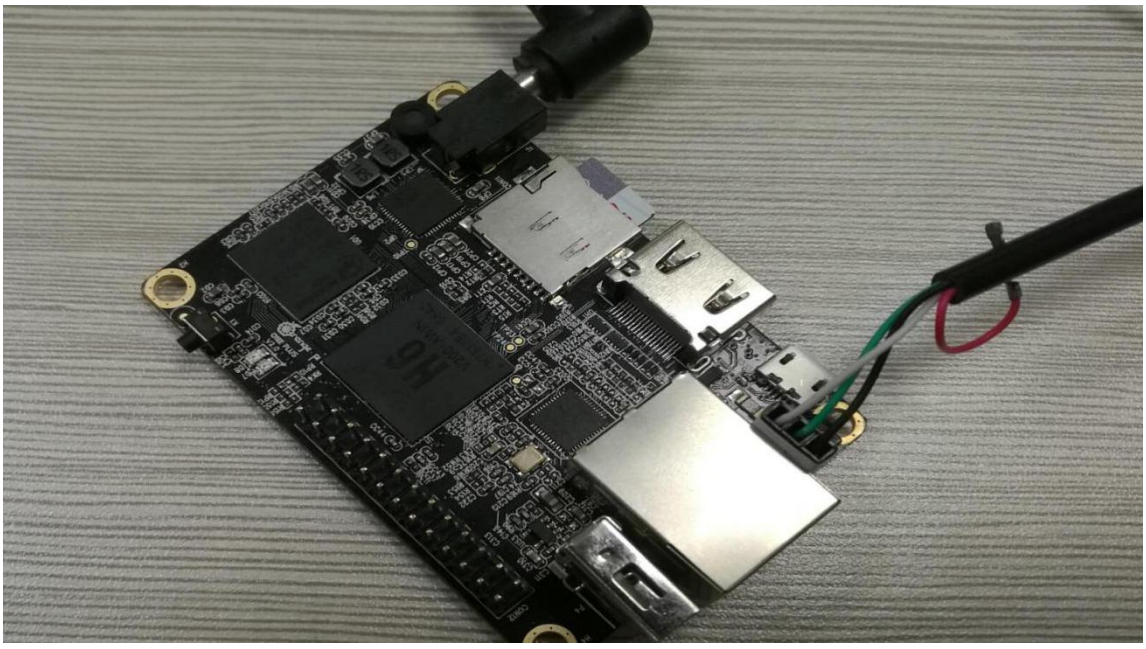
   $ ./tinyplay test.wav -D 0 -d 0

VIII. Using Debug tools

Prepare a USB to TTL cable like the following:

Connect the debug port with cable like the following, or you could check the function onthe board silk.

- Black——GND
- Green——RX
- Gray——TX



1. Operation Steps on Windows

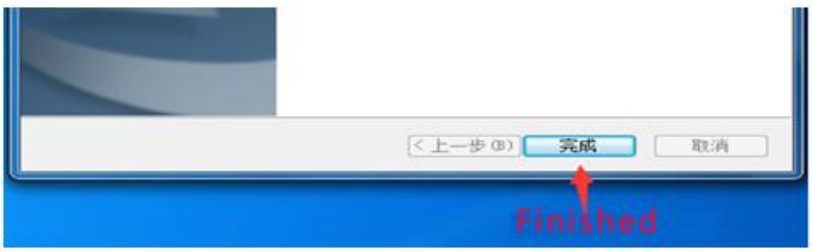
In order to get more debugging information in the project development process of using OrangePi, OrangePi default support for serial information debugging. For developers, you can simply get the serial port debugging information with the materials mentioned above. The host computer using different serial debugging tools are similar, basically can reference with the following manual for deployment. There are a lot of debugging tools for Windows platform, the most commonly used tool is putty. This section takes putty as an example to explain the deployment.

- **Install USB Driver**

Download and unzip the latest version of driver
PL2303_Prolific_DriverInstaller_v130.zip

- Choose application installation as Administrator

- Wait for completing installation

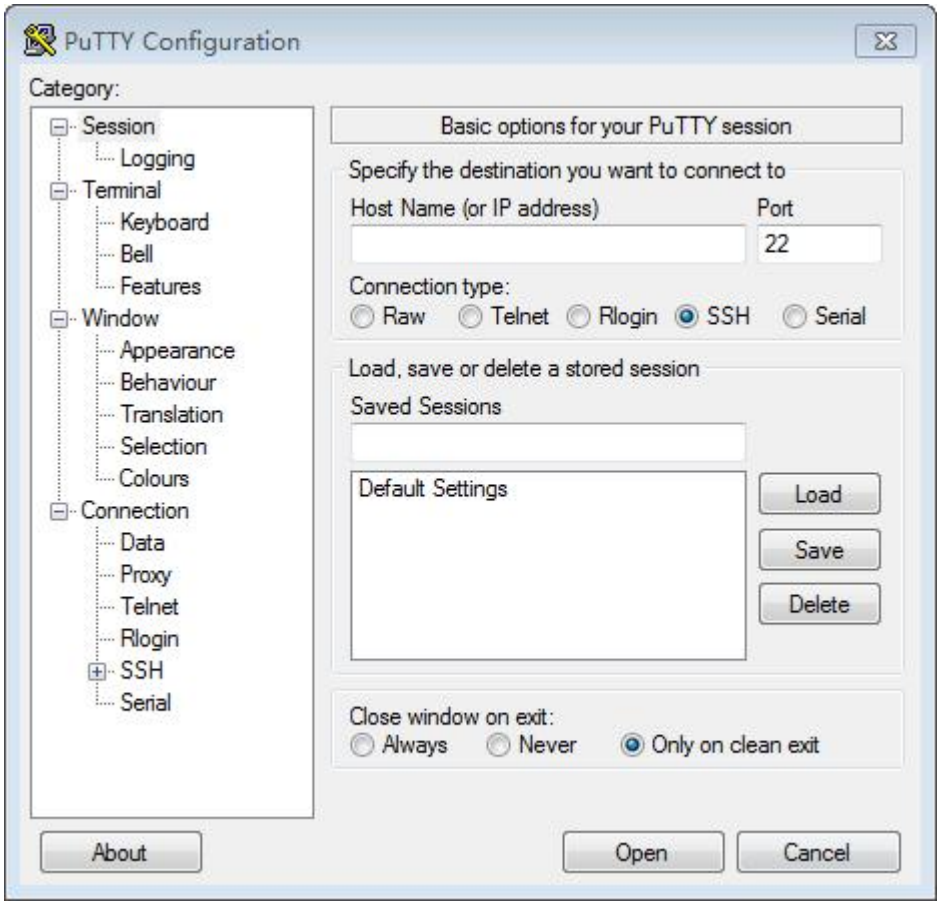


- **Install Putty**

You could download Putty from the following address and select a suitable version to your are development environment:

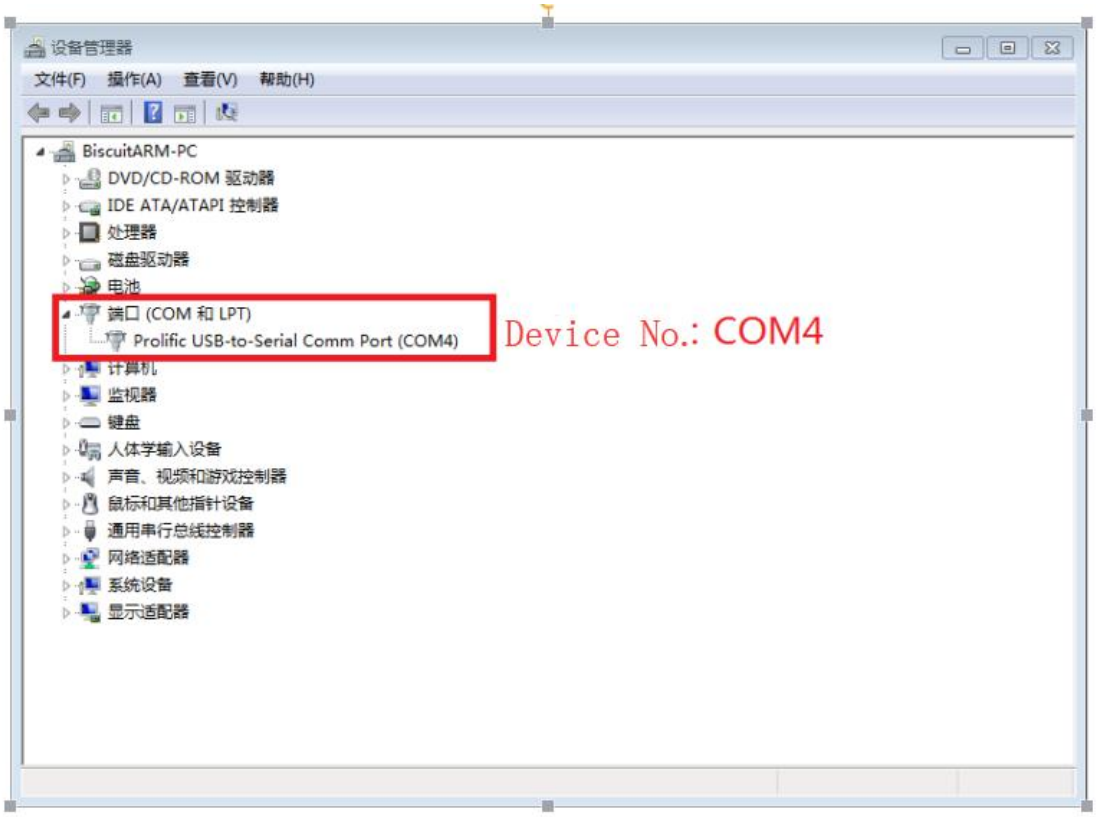
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

- Double click putty.exe to run putty, interface shown as below



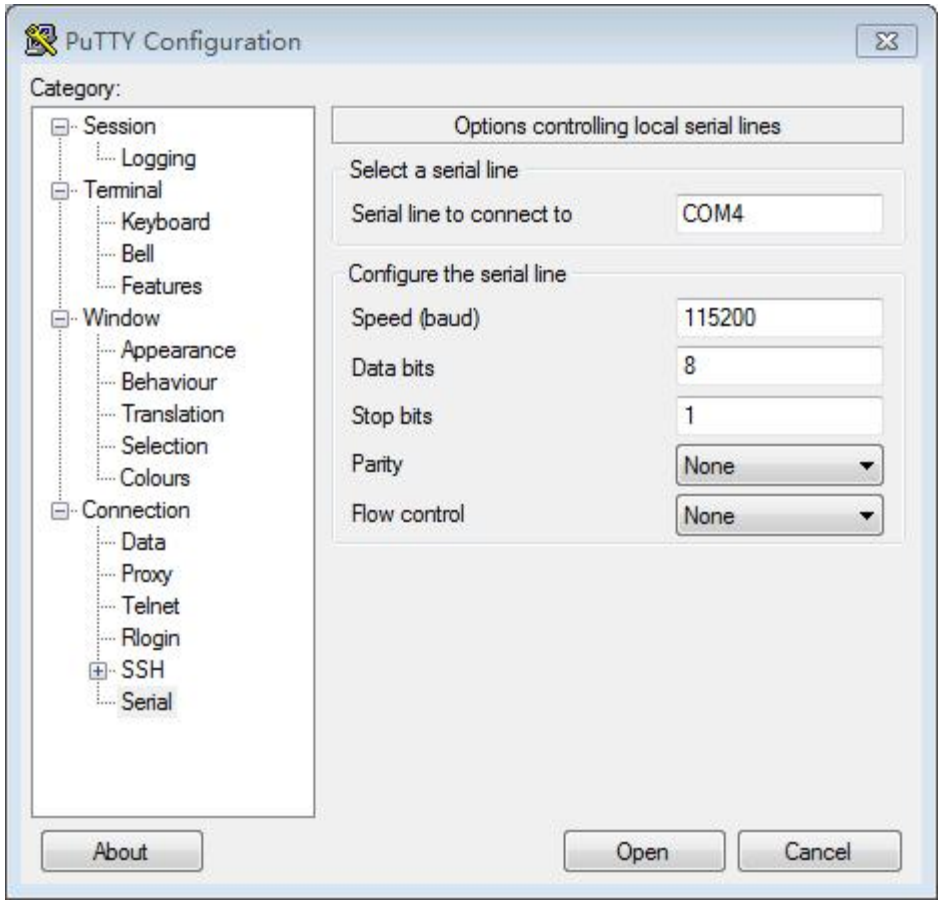
- **Equipment information acquisition**

In Windows7, we can see whether the serial port connection is normal and the serial port's device number through the device manager. If the device is not properly identified, please check whether the driver is installed successfully. If there is a problem with the driver installation, you can try using the 360 driver master to scan the installation driver.



- **Putty Configure**

Serial port should set to the corresponding port number (COM4), close flow control and set the speed as 115200



- **Start debug serial port with output**

  Power OrangePi on, putty will print out serial log information auto.

2. Operation Steps on Linux

When running putty, , there is not too much difference on Linux platform or Windows platform. Here the instruction is based on Ubuntu 14.04 OS.

- **Install and start Putty**

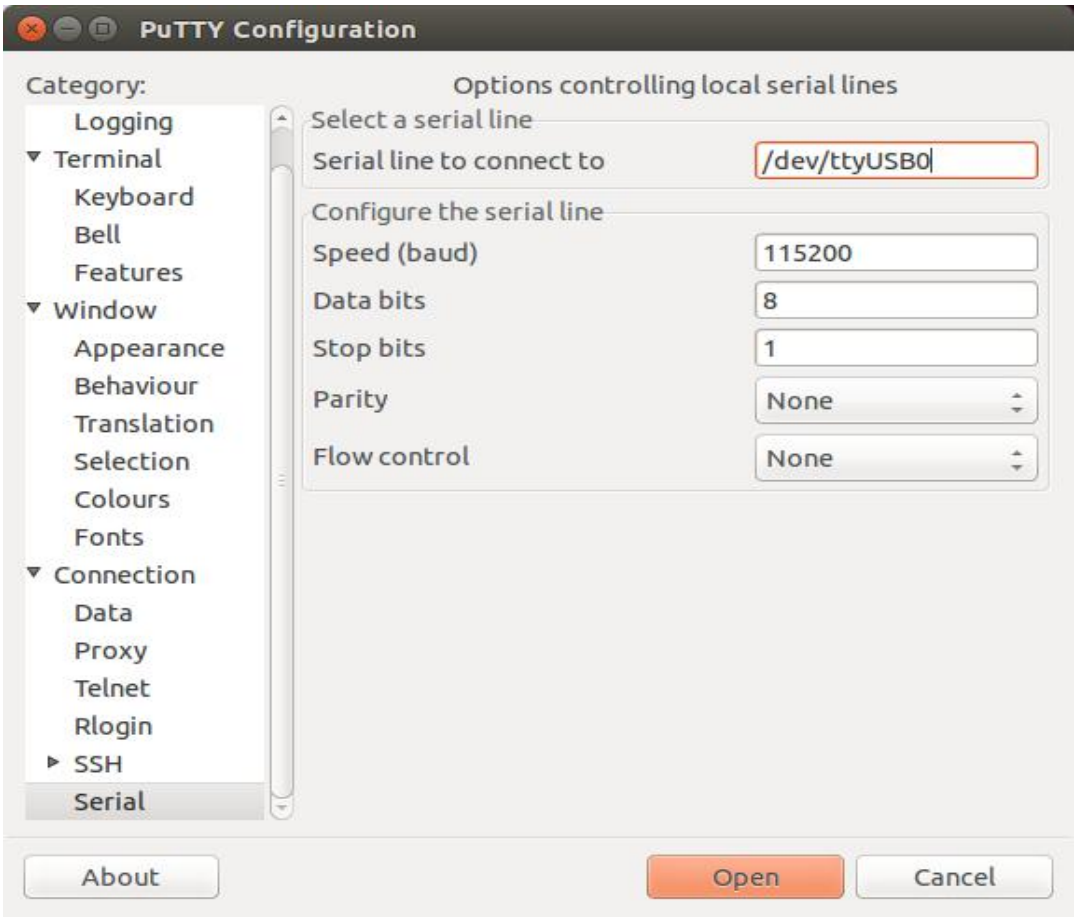
```
$ sudo apt-get install putty
$ sudo putty
```

- **Configure Putty**

Check serial number via ls /dev/ttyUSB*

Set baud rate into 115200

Close flow control


PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Fonts
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Options controlling local serial lines
Select a serial line
Serial line to connect to
/dev/ttyUSB0
Configure the serial line
Speed (baud)
115200
Data bits
8
Stop bits
1
Parity
None
Flow control
None
About
Open
Cancel