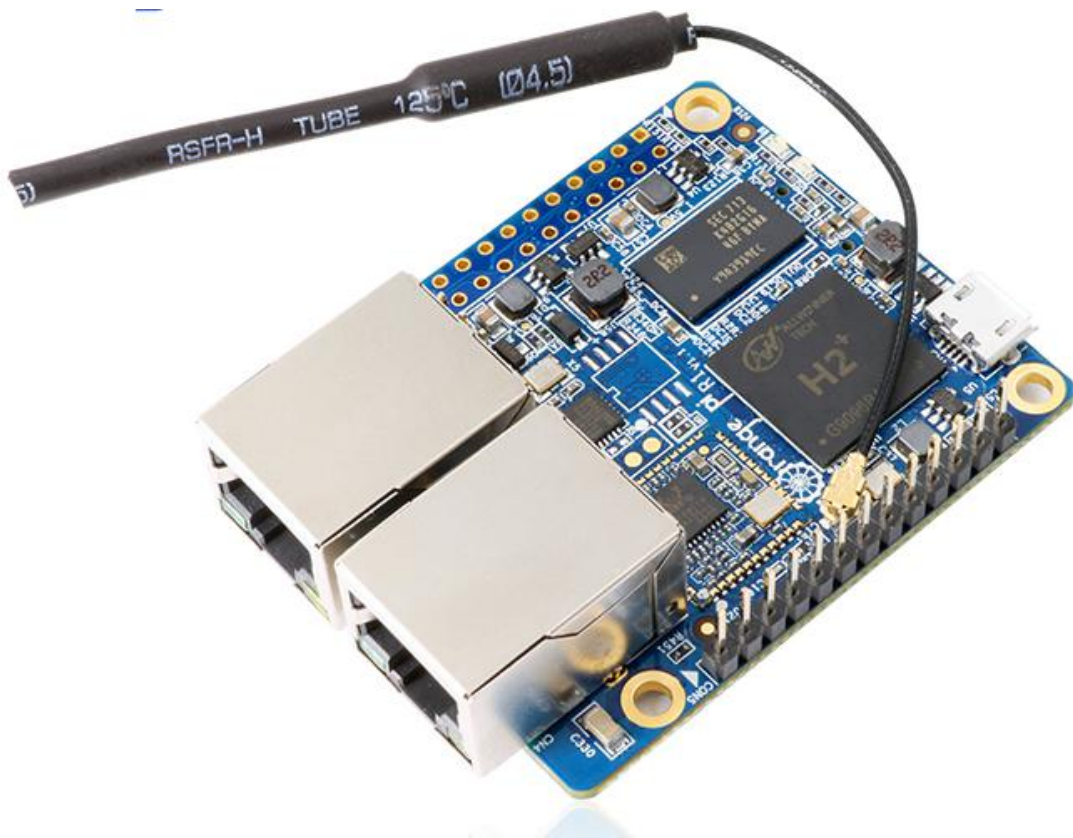




Orange Pi R1 User Manual



V1.1



Contents

- I. Orange Pi R1 Introduction..... 1
 - 1. What is Orange Pi R1?..... 1
 - 2. What can I do with Orange Pi R1?..... 1
 - 3. Whom is it for?..... 1
 - 4. Hardware specification of Orange Pi R1..... 1
 - 5. GPIO Specifications.....3
- II. Using Method Introduction.....5
 - 1. Step 1: Prepare Accessories Needed.....5
 - 2. Step 2: Prepare a TF Card..... 6
 - 3. Step 3: Boot your Orange Pi..... 12
 - 4. Step 4: Turn off your Orange Pi Correctly..... 14
 - 5. Other configuration..... 14
 - 6. Universal Software Configuration..... 17
- III. Linux Kernel Source Code Compilation.....27
 - 1. Download Linux Source Code.....27
 - 2. Compile Project Source Code..... 28
 - 3. Update the Kernel Image File and Replace Library..... 30
- IV. Android Kernel Source Code Compilation.....32
 - 1. Install JDK..... 32
 - 2. Install Platform Supported Software..... 33
 - 3. Download Android Source Package..... 33
 - 4. Install Compiler Tool Chain..... 33
 - 5. Compile Lichee Source Code..... 34
 - 6. Compile Command of Android Code..... 34
- V. Use Project Configuration Files..... 37
 - 1. sys_config.fex Introduction..... 37
 - 2. Examples..... 37
- VI. OrangePi Driver development.....40
 - 1. Device Driver and Application Programming..... 40
 - 2. Compile device driver..... 43
 - 3. Cross compiler Application Program..... 45



4. Running Driver and Application.....	47
VII. Using Debug tools on OrangePi.....	49
1. Operation Steps on Windows.....	49
2. Operation Steps on Linux.....	53
VIII. Orange Pi R1 Routing System Adaptation.....	57
1. Configuration, Compiling and Installation of Quagga.....	57
2. Build a test environment; a small-scale local area network.....	58
3. Configure OSPF Routing Protocol.....	60
4. Configure NAT.....	62
5. Configure DHCP.....	63



I. Orange Pi R1 Introduction

1. What is Orange Pi R1?

It's an open-source single-board computer. It can run Android 4.4, Ubuntu, Debian, Armbian Image. It uses the AllWinner H2 SoC, and has 256MB DDR3 SDRAM.

2. What can I do with Orange Pi R1?

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 R1 is open source

3. Whom is it for?

Orange Pi R1 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.

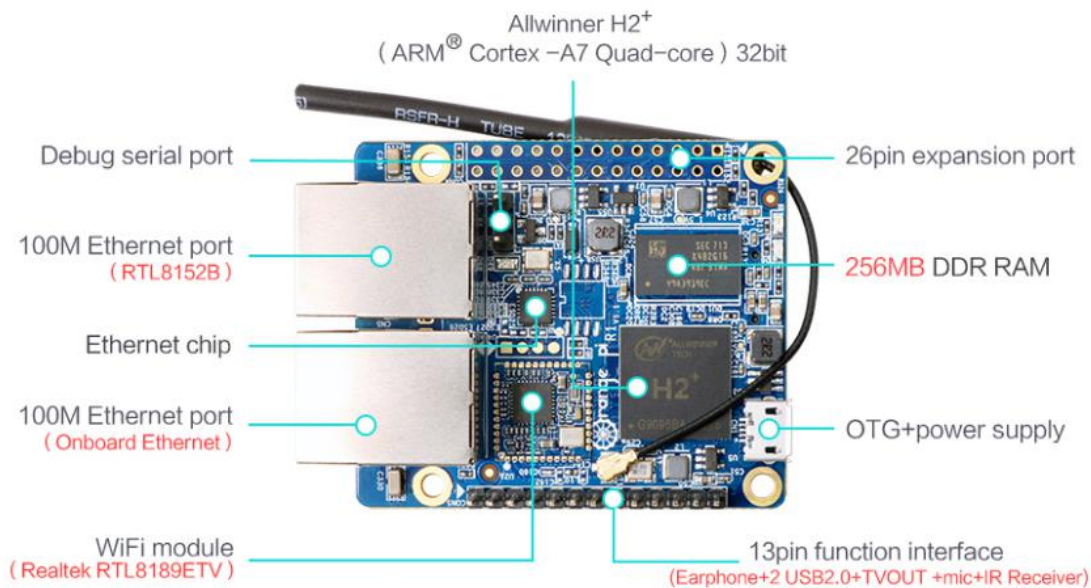
4. Hardware specification of Orange Pi R1

Hardware specification	
CPU	H2 Quad-core Cortex-A7 H.265
GPU	Mali400MP2 GPU @600MHz, Supports OpenGL ES 2.0
Memory (SDRAM)	256MB DDR3 (shared with GPU)
Onboard Storage	TF card (Max. 32GB) /Spi flash(16MB)
Wifi Antenna	Yes



Onboard Network	100M/10M Ethernet RJ45 (on board Ethernet) 100M/10M Ethernet RJ45 (RTL8152B)
Onboard WIFI	RTL8189ETV, IEEE 802.11 b/g/n
Audio Input	Support output via 13pin(AV-out)
Video Outputs	Support output via 13pin(cvbs)
Power Source	USB OTG can supply power
Buttons	Power button
Low-level peripherals	26 Pins Header, compatible with Raspberry Pi B+ 13 Pins Header, with 2x USB, IR pin, AUDIO
GPIO(1x3) pin	UART, ground.
LED	Power led & Status led
Supported OS	Android, Lubuntu, Debian, Armbian Image
Interface definition	
Product size	45mm × 60mm
Weight	48g
Orange Pi™ is a trademark of the Shenzhen Xunlong Software CO., Limited	

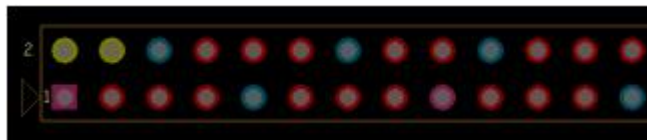
Interface instructions





5. GPIO Specifications

A 26-pin GPIO interface on the Orange Pi R1 is the same as Model A and Model B of Raspberry Pi. The picture below is GPIO pin definition of Orange Pi R1.



OrangePi_R1 (H2)		
CON3-P01	VCC-3V3	
CON3-P02	VCC-5V	
CON3-P03	TWIO-SDA	PA12
CON3-P04	VCC-5V	
CON3-P05	TWIO-SCK	PA11
CON3-P06	GND	
CON3-P07	PWM1	PA6
CON3-P08	UART1_TX	PG6
CON3-P09	GND	
CON3-P10	UART1_RX	PG7
CON3-P11	UART2_RX	PA1
CON3-P12	PA7	PA7
CON3-P13	UART2_TX	PA0
CON3-P14	GND	



CON3-P15	UART2_CTS	PA3
CON3-P16	TWI1-SDA	PA19
CON3-P17	VCC3V3-EXT	
CON3-P18	TWI1-SCK	PA18
CON3-P19	SPI1_MOSI	PA15
CON3-P20	GND	
CON3-P21	SPI1_MISO	PA16
CON3-P22	UART2_RTS	PA2
CON3-P23	SPI1_CLK	PA14
CON3-P24	SPI1_CS	PA13
CON3-P25	GND	
CON3-P26	PA10	PA10



II. Using Method Introduction

Follow these steps, you can configure and run your Orange Pi in a very short period of time. Boot your Orange Pi need to complete the following steps.

1. Step 1: Prepare Accessories Needed

You need at least some accessories like the following if it is your first time to use the Orange Pi (we would suggest you using the Expansion board at the same time).

No.	Items	Requirements and Instructions
1	TF card	8GB min.; class 10. Branded TF cards would be reference which are much more reliable.
2	AV video cable	A standard AV video cable can be used to connect stimulated monitor if a HDMI monitor is unavailable.
3	Keyboard and mouse	Any keyboard and mouse with USB port is applicable; Keyboard and mouse are high-power, so a USB concentrator is required.
4	Ethernet cable(Optional)	Network is optional, It makes more convenient to mount and upgrade software in your Orange Pi PC.
5	DC power adapter	5V/2V min. high qualified power adapter, OTG can used a power supply.
6	Audio cable (Optional)	You can select an audio cable with 3.5mm jack to feel stereo audio.



AV video cable



TF card



DC power adapter



Expansion Board

2. Step 2: Prepare a TF Card

In order to use Orange Pi normally, you must install the operating system into TF card first.

1) Write Linux into TF Card Based on Windows Platform

- a. Inserting the TF card into the computer, the capacity of the card must be bigger than the operating system, usually requires 8GB or bigger.
- b. Formatting the TF card.

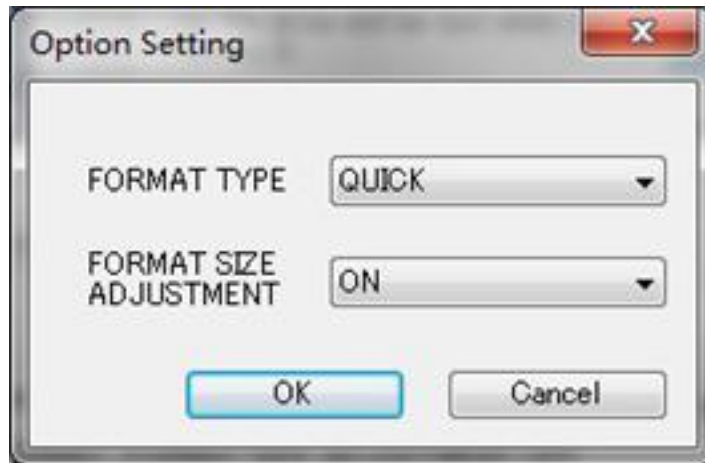
i Download tools for formatting TF card, such as TF Formatter, it could be downloaded from:

https://www.sdcard.org/downloads/formatter_4/eula_windows/

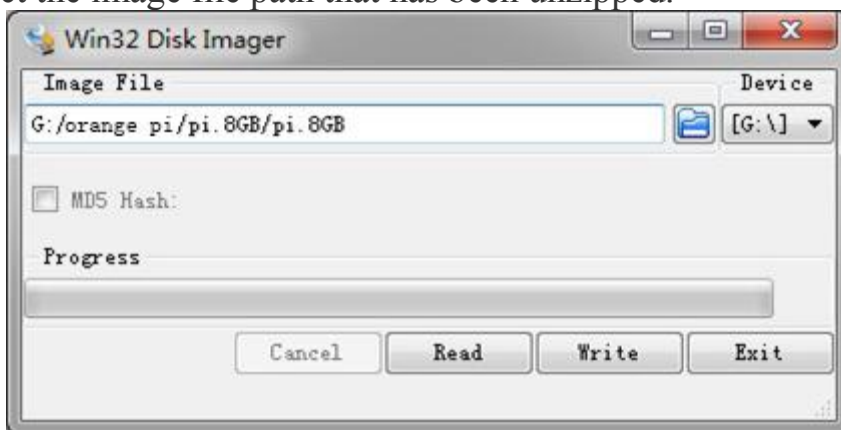
ii Unzip the downloaded files, and run *setup.exe*

iii In the *options settings* select the "format" button for quick formatting. "Format size adjustment" select "(ON)"





- iv Make sure the inserted TF card disk are in accordance with the chosen disk.
- v Click the "*Format*" button.
- c. Download the operating system image file from the download page, the page address is as following:
<http://www.orangepi.org/downloadresources>
- d. Unzip the downloaded file (in addition to the Android system, this method can be used to burn to write, the Android system need another burn, the following will introduce)
- e. Right click to download the file, select "*Unzip file*" to write image to TF card
 - i Download tools to write image,such as *Win32 Diskimager*, here is the download page:
<http://sourceforge.net/projects/win32diskimager/files/Archive/>
 - ii Select the image file path that has been unzipped.



- iii Click "*Write*" button and wait for the image to write.
- iv After the image is written, click "*Exit*" button.



2) Write Linux into TF card based on Linux platform?

- a. Inserting the TF card into the computer, the capacity of the card must be larger than the operating system image, usually requires 4GB or greater capacity.
- b. Formatting the TF card.
 - i Run *fdisk -l* order to make sure TF disk.
 - ii Run *umount /dev/sdxx* to uninstall all partitions of TF Card.
 - iii Run *sudo fdisk /dev/sdx* order. Use *o* command to delete all partitions of TF Card, and then us *n* order to add a new partition, finally use *w* command to save and exit.
 - iv Run *sudo mkfs.vfat /dev/sdx1* command to format the TF card partition set up last step to FAT32 form(according to your TF card disk to replace *x*). Or you could skip this step since command in Linux will format TF card automatic.

- c. Download the OS image from download page

<http://www.orangepi.org/downloadresources>

- d. Unzip and right click the downloaded file, select " Unzip file"

- e. Write image to TF card

- i Run *sudo fdisk -l* order to make sure the TF card disk

- ii make sure the image file **hash key** is the same as download page mention(optional). It will output *sha1sum [path]/[imagename]*, which should be same as the image paye "SHA-1"

- iii Run *umount /dev/sdxx* order to uninstall all partitions in TF Card

- iv Run *sudo dd bs=4M if=[path]/[imagename] of=/dev/sdx* to write down image file. Wait for the image to write. If it cannot work at 4M, then replace a 1M which takes more time. You can run *sudo pkill -USR1 -n -x dd* order to monitoring procedure.

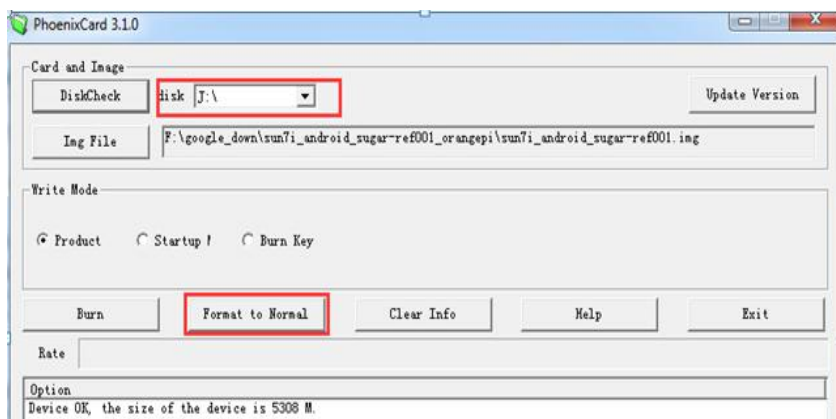
3) Use PhoenixCard tool to write Android image into TF card

It is impossible for Android image to be written into TF card by using *dd* command under Linux or using *Win32 Diskimager* under Windows. Here

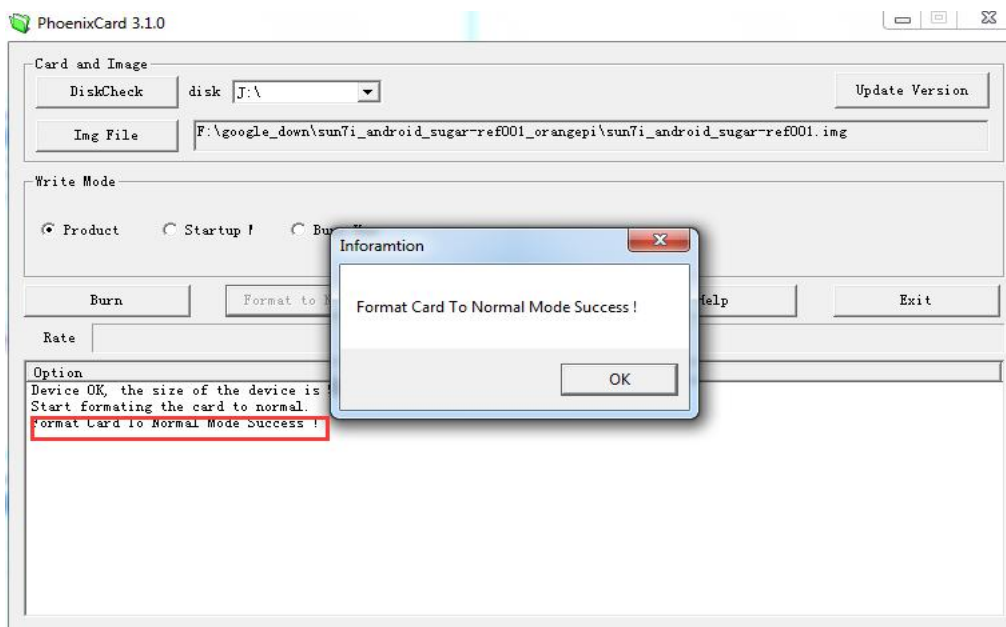


PhoenixCard tool is applicable for Android image writing.

- a. Download the Android OS image and **PhoenixCard** tool.
Download **PhoenixCard** from here:
https://drive.google.com/file/d/0B_VynIqhAcB7NTg2UkRDdHRWX2s/edit?usp=sharing
Download Android OS image from here:
<http://www.orangepi.org/downloadresources/>
- b. Format the TF card



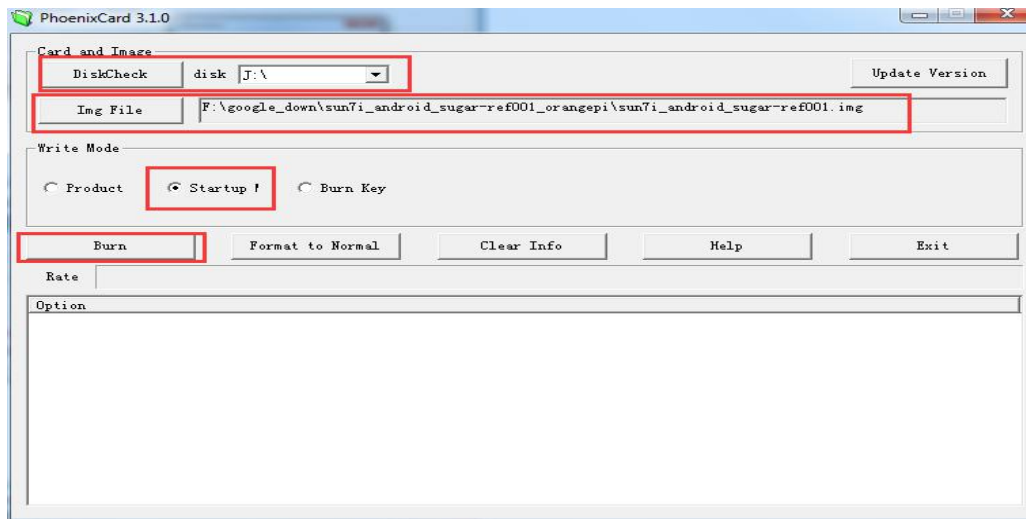
- c. Please make sure the inserted TF card is in accordance with the chosen TF card, click "format to normal" button for TF card formatting.



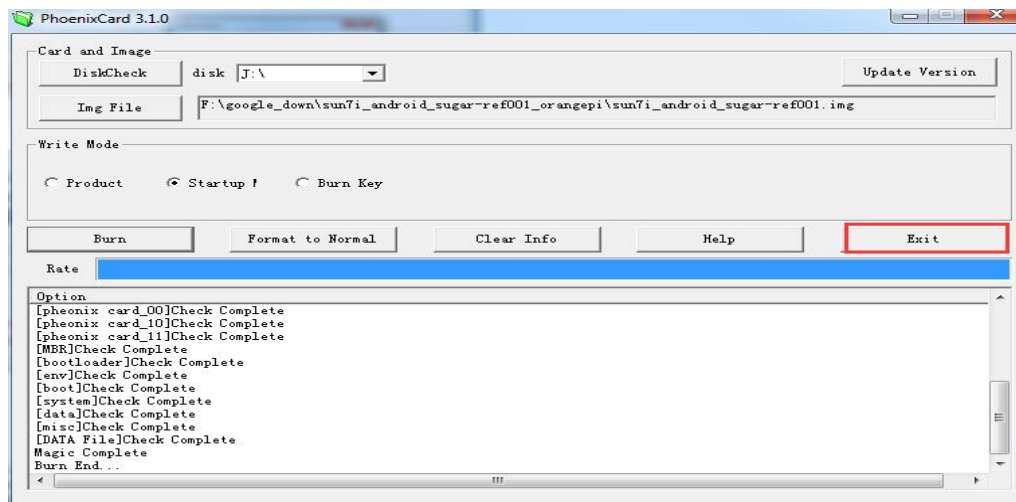
- d. Click "OK" button after successfully formatted the TF card to normal.



- e. Burn the Android OS image into your TF card. Please pay attention to the following with red marks.



- f. Click "Burn" button for writing to TF card and wait for it finish



- g. Click "Exit" button after burn Android image to TF card successfully.

4) Write Armbian Image into TF Card

- a. Insert TF card into computer, please note that the TF card capacity must bigger than the operating system image, usually need to be 8GB or bigger.
- b. Download the OS image file from the download page:
<http://www.armbian.com/download/>
- c. Write the image into TF card.



i Download image writing tool such as *Rufus*, the download page: <https://rufus.akeo.ie/>



ii Select the image file path that has been unzipped



iii Click "start" button and wait for the image to write.



iv After the image is written, click "close" button

3. Step 3: Boot your Orange Pi

1) Hardware Connection



Orange Pi R1 runs on Android 4.4 system



Orange Pi R1 runs on Debian system



Orange Pi R1 runs on Ubuntu system

2) Details of Booting Steps

- a. Insert the TF card with written image in to the TF card slot.
- b. You could use HDMI cable to connect your Orange Pi to HDMI TV or monitor.

You could also use AV interface and audio interface to connect output



video and audio to analog TV or display.

- c. There is 13pin on board which you could connect to expansion board. For expansion board, 2USB ports, mic and IR receiver are available.
- d. It is the network module on board, which you can access Orange Pi to the wired network.
- e. You could connect to a power adapter on mic USB OTG with a power adapter up to or bigger than 5V/2A. Avoid using smaller power GSM mobile phone charger, it is not able to output 2A even if it marked "5V/2A ".

The Orange Pi will boot in a few minutes If the above steps are successful. There will be graphical interface in he monitor. It may take a long time to start the first time, please wait patiently. The next time will boot very fast.

4. Step 4: Turn off your Orange Pi Correctly

- You can use the shutdown button on the interface to safety close the Orange Pi.
- You can also close the system by entering commands in the shell:

sudo halt

or

sudo shutdown -h

It will be safety to turn off the Orange Pi. If directly use the power button to shut down the system may damage the file system on TF Card. After the system is closed, the power can be cut off by more than 5 seconds' press.

5. Other configuration

1) Connect to the wireless network

- Method 1:
 - a. Enter the following in the command line:
\$ ifconfig
To check whether there is (wlan*)
 - b. If no, load the corresponding module corresponding to the wlan model



```
$ insmod 8189es.ko
```

For example: For RTL8189 ETV is corresponding to 8189es.ko

- c. Enter command `ifconfig`, you should see `wlan0`(hypothesis it is `wlan0`)
- d. Configure wireless network, first you need to know `ssid` and `psk`(account and password), enter corresponding `wlan*`, `ssid`, `psk`

```
$ sudo nano /etc/network/interfaces    (add the following contents)
auto wlan0
iface wlan0 inet dhcp
wpa-ssid xxxx
wpa-psk xxxx
```
- e. Reboot the computer and the wireless network will work.

```
$ sudo reboot
```

● Method 2:

- a. Build wifi hotspot configuration file of `wpa_supplicant.conf` for on `/etc/network/` directory and add the following:

```
network={
    ssid="wifi hot spot name"
    psk="wifi hot spot password"
    priority=1
}
```

- b. Connect wifi, here is the command:

```
ifconfig wlan0 up
sudo wpa_supplicant -i wlan0 -c /etc/network/wpa_supplicant.conf &
dhcpcd wlan0 &
```
- c. Test the condition of wifi connection
Use `iwconfig` command, you will find the related information of `wlan0`, use `ping` command to test.

2) Login via vnc and ssh

If there is no condition for connecting HDMI, you could enter the system via vnc or ssh remote login.

- Login via serial port and install ssh

```
apt-get install ssh
```
- Modify ssh configuration file `/etc/ssh/sshd_config`



```
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authoriz

# Don't read the user's ~/.rhosts and
IgnoreRhosts yes
# For this to work you will also need
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh
#IgnoreUserKnownHosts yes

# To enable empty passwords, change t
PermitEmptyPasswords no

# Change to yes to enable challenge-r
# some PAM modules and threads)
ChallengeResponseAuthentication no
```

- Check the IP with ifconfig, login via ssh of root user

```
curry@curry:~$ ssh root@192.168.1.178
root@192.168.1.178's password:
Welcome to Ubuntu 15.10 (GNU/Linux 3.4.39-02-lobo armv7l)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Apr 11 15:20:33 2017 from 192.168.1.111
root@OrangePI [10:03:27 AM] [~]
-> #
```

3) HDMI or 3.5mm Sound Output(3.5mm sound output would require using an expansion board)

- The sound was default to output via HDMI on image, it could check and change via alsamixer.

ls /etc/asound.conf

card indicates card number, device indicates device number.

aplay -l it could check the system to load the sound card number and details

cat /proc/asound/cards it also could check the sound card and details

It could be used after use alsamixer to change the sound card.

alsactl store -f /var/lib/alsa/asound.state used for saving modified parameters

- It needs to modify configuration on file system for output on 3.5mm of /etc/asound.conf, modify card1 into card0, or use amixer to modify. The default one is configured, or you could use player on graphical interface to switch via sound channel selection.



- c. How to use mic sound recording
`arecord -d 5 -f cd -t wav 123.wav`
 After recording, use the following to play
`aplay 123.wav`

6. Universal Software Configuration

1) Default Account Changing

The default log in account is orangepi. In order to secure, it is recommended to modify the default orangepi accounts to your own account, for example Zhangsan. Steps are as follows:

- a. Use root account to login Orange Pi (please note that do not login with the account of orangepi)
 b. `$ usermod -l zhangsan orangepi` Change orangepi account into Zhangsan

```
@orangepi:~$ usermod -l zhangsan orangepi
```

- c. `$ groupmod -n zhangsan orangepi` Change group

```
@orangepi:~$ groupmod -n zhangsan orangepi
```

- d. `$ mv /home/orangepi /home/zhangsan` Change directory of original orangepi

```
@orangepi:~$ mv /home/orangepi /home/zhangsan
```

- e. `$ usermod -d /home/orangepi orangepi` Set this directory to orangepi user's home directory

```
@orangepi:~$ usermod -d /home/zhangsan zhangsan
```

- f. `$ cat /etc/passwd` It should be shown as below:

```
pulse:x:112:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
zhangsan:x:1001:1001:orangepi,,,:/home/zhangsan:/bin/bash
```

After the modification of the above items, it can be used the new account Zhangsan to land.

2) U Disk Automatic Mounted Configuration

- a. `sudo apt-get install usbmount`
 b. `sudo vim /etc/udev/rules.d/automount.rules`
`ACTION=="add",KERNEL=="sdb*", RUN+="/usr/bin/pmount --sync --umask 000 %k"`
`ACTION=="remove", KERNEL=="sdb*", RUN+="/usr/bin/pumount %k"`
`ACTION=="add",KERNEL=="sdc*", RUN+="/usr/bin/pmount --sync`



```
--umask 000 %k"
```

```
ACTION=="remove", KERNEL=="sdc*", RUN+="/usr/bin/pumount %k"
```

c. `udevadm control --reload-rules`

It could refer to this:

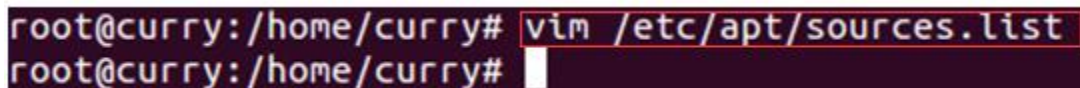
<http://unix.stackexchange.com/questions/134797/how-to-automatically-mount-an-usb-device-on-plugin-time-on-an-already-running-sy>

3) System Source Configuration

Take Ubuntu as an example:

a. Open the source file

```
$ sudo vi /etc/apt/sources.list
```



```
root@curry:/home/curry# vim /etc/apt/sources.list
root@curry:/home/curry#
```

b. Edit source file

Replace the source file with your favorite source. Take an example of Ubuntu 16.04 on Zhonkeda source:

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main
multiverse restricted universe
```

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports
main multiverse restricted universe
```

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed
main multiverse restricted universe
```

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security main
multiverse restricted universe
```

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates main
multiverse restricted universe
```

```
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main
multiverse restricted universe
```

```
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports
main multiverse restricted universe
```

```
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed
main multiverse restricted universe
```

```
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security
main multiverse restricted universe
```

```
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates
main multiverse restricted universe
```



Note: xenial is the version of the code name in this source, if the other version of Ubuntu needs to replace the corresponding version code which can be found on the internet.

4) Remote desktop installation

There are a lot of software, such as VNG, XRDP, X2GO, etc. For X2GO, it has more functions, and desktop color restore is very good which does not need too much configuration. And XRDP is much more safety than VNC.

- a. `$sudo apt-get install tightvncserver` Install VNC

```
apt-get install tightvncserver
```

- b. `vncpasswd` Set the password: do not execute this command but executing `vncserver` directly. It will prompt you to enter the password twice, when prompted whether can be read only to select the *N*.

```
root@curry:/home/curry/tools/minidlna/minidlna-1.1.0# vncpasswd
Using password file /root/.vnc/passwd
VNC directory /root/.vnc does not exist, creating.
Password:
Verify:
```

- c. Open one or more of desktops by `vncserver` or `vncserver:1(vncserver:2)`... you can also transfer more parameters through the full command as below:

```
vncserver :1 -geometry 1024x768 -depth 16 -pixelformat rgb565
```

(Note: If it prompted you that cannot find the file or other error when installing, please run `sudo apt-get update` to update the software source and try installing again.)

5) NAS and DLAN Configuration

- a. NAS:

There are many files could be reference from Internet, for example: <http://www.geekfan.net/5003/>, it detailed descriptions on the operation and the mounted of U disk is very useful.

- b. DLNA:

Mainly through the `minidlna` software to achieve the sharing of media resources within the LAN, such as sharing video, music, etc.. The installation steps are as follows:

- i `sudo apt-get minidlna`
- ii Execute the following command to modify the configuration file:
`sudo nano /etc/minidlna.conf`



Note: you can also use other text editor to modify.

- iii Add the following:
 - media_dir=A,/nas, path: /DLNA/Music
 - media_dir=V,/nas, path: /DLNA/Video
 - media_dir=P,/nas, path: DLNA/Picture
 - db_dir=/nas, path: /DLNA/log
 - db_dir=/nas, path: /DLNA/db

ctrl +o and enter, ctrl +x to save and exit.

iv Established above folders respectively, noted that path consistency and assigned to read and write permissions. In order for convenient, it could be Chmod 755, such as sudo Chmod 755 /nas path /DLNA/Music

v Re-start minidlna to take effect the configuration: /etc/init.d/minidlna restart.

Transmit the corresponding file on the computer to the corresponding folder through samba.

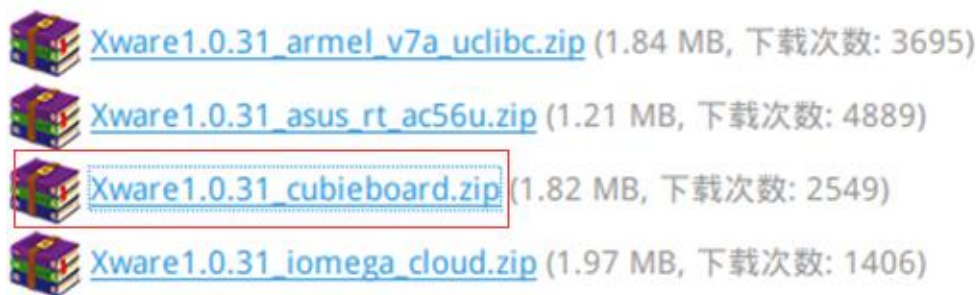
Note: It is recommended to download MoliPlayer on the mobile device. The effect is good and no blue light pressure on both Android and IOS.

6) Thunder remote download

a. Go to the Thunder routing forum to download the required installation package first. The link for stable version:

<http://luyou.xunlei.com/thread-12545-1-1.html>.

Download Xware1.0.31_cubieboard zip file.



Note: If you want to try the latest version, you can download the latest test version: <http://luyou.xunlei.com/thread-15167-1-1.htm>.

b. Enter the directory after uploaded the unzip file to OrangePi. It is recommended to rename the file to xunlei

c. Installation method of version 1.0.31:

- i \$ cd /xxx/xunlei The xxx is the directory of installation xunlei file
- ii \$ chmod 755 portal
- iii \$./portal



```

root@curry:/home/curry/Downloads/xunlei# ls
EmbedThunderManager ETMDaemon portal vod_httpserver
root@curry:/home/curry/Downloads/xunlei# chmod 755 portal
root@curry:/home/curry/Downloads/xunlei# █

```

iv You will get an activation code after booting like the following:

```

YOUR CONTROL PORT IS: 9000

starting xunlei service...
etm path: /home/echo/xunlei
execv: /home/echo/xunlei/lib/ETMDaemon.

getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)

THE ACTIVE CODE IS: ██████████

```

Here you will get
an activation code

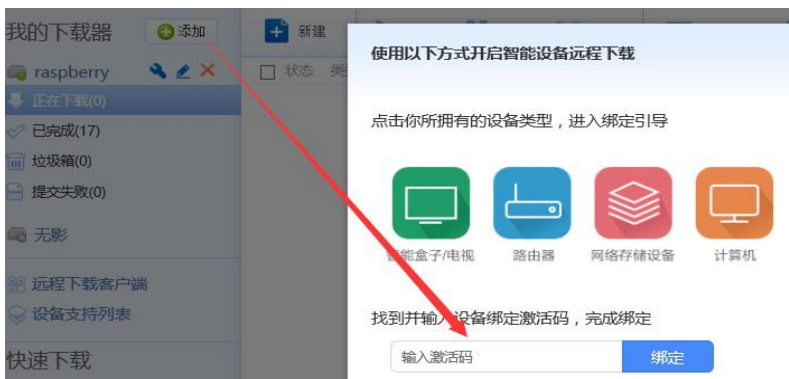
←

```

go to http://yuancheng.xunlei.com, bind your device with the active code.
finished.

```

v Copy this activation code to <http://yuancheng.xunlei.com> (Which required to log in with account of Thunder). Then click the tab on the top right corner to add, fill in the activation code to complete the binding according to the following figure.



vi Setting start up

```

$ sudo nano /etc/rc.local
add the following contents before exit 0
cd /xx/xunlei
./portal &
ctrl +o and enter, ctrl +x to save and exit.

```

d. Installation of version 3.0.32.253:

- i \$ cd /xxx/xunlei The xxx is the directory of installation file of xunlei
- ii \$ sudo nano thunder_mounts.cfg Modify the download path



```
#仅接受以下列路径开头的挂载路径
available_mounts
{
    /media/SATA
}

#下列目录被认为是分区，并在程序运行期间不变
virtual_mounts
{
```

iii `chmod +x etm_monitor`
 iv Run `./etm_monitor`, there will be an activation code page like version 1.0.32. And then binding on the Thunder remote page (above steps 4, 5). There might be one or two errors while running, ignore it (selection type of shell and generation of INI file).

v Setting start up

`sudo nano /etc/rc.local` add the following contents before exit 0

`cd /xx/xunlei`

`./etm_monitor &`

`ctrl +o` and enter, `ctrl +x` to save and exit.

It could be remote downloading on computer, mobile phone or tablet by login yuancheng.xunlei.com

7) Modify the size of ext4 file system

After made the written image into SD card for booting, enter into rootfs partition's expansion of file system. It could enhance the performance of SD card to avoid limited storage cause problem.

● Method 1

Extend rootfs file partition of TF card on PC:

Select the specified disk, right click and select the corresponding disk, select "change size" and adjust it into your desired size, click "re-size", close the dialog box and click "apply all operations", select the application to complete the expansion operation

● Method 2

Enter into the system and extend via shell

Before partition



```

root@OrangePi:~# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  2.0G  565M  1.4G  30% /
devtmpfs        482M   0  482M   0% /dev
tmpfs           490M   0  490M   0% /dev/shm
tmpfs           490M  13M  478M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           490M   0  490M   0% /sys/fs/cgroup
/dev/mmcblk0p1  50M   13M   38M  26% /boot

```

Enter into system and expend via `resize_rootfs.sh`

```

root@OrangePi:/usr/local/sbin# resize_rootfs.sh
+ DEVICE=/dev/mmcblk0
+ PART=2
+ resize
+ fdisk -l /dev/mmcblk0
+ grep /dev/mmcblk0p2
+ awk {print $2}
+ start=143360
+ echo 143360
143360
+ set +e
+ fdisk /dev/mmcblk0

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

```

Enter `resize_rootfs.sh` on command line, the system will expending automatically,

Reboot the system and use `df -lh` to check whether expending is successful

```

+ set -e
+ partx -u /dev/mmcblk0
+ resize2fs /dev/mmcblk0p2
resize2fs 1.42.13 (17-May-2015)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p2 is now 3871616 (4k) blocks long.

+ echo Done!
Done!
root@OrangePi:/usr/local/sbin# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  15G  566M  14G   4% /
devtmpfs        482M   0  482M   0% /dev
tmpfs           490M   0  490M   0% /dev/shm
tmpfs           490M  13M  478M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           490M   0  490M   0% /sys/fs/cgroup
/dev/mmcblk0p1  50M   13M   38M  26% /boot

```

a. Expand file system

i Boot to Linux, umount `/dev/sdb1` and `/dev/sdb2`, if it prompts disk busy, then use `fuser` to clean the using disk (we will recommend using another Linux booting disk to lead the system).

ii Use `fdisk /dev/sdb` to adjust the partition size, after into it, enter `p`, and keep in mind about the initial position of needed extending size partition.

iii Enter `d` to delete the partition need to change the size (my file system is `/dev/sdb2`, which is the 2 partition).

iv Enter `n` to build a new partition, make sure the initial position is the



same as you deleted, and enter the number as you desire.

- v Enter w to save the partition data.
- vi Use the following command to check the file system(make sure it is a right file system)
`e2fsck -f /dev/sdb2`
- vii Adjust the partition size
`resize2fs /dev/sdb2`
- viii It could mount a disk partition, you could check whether it has changed.

b. Shrink file system

- i Boot to Linux, umount /dev/sdb1 and /dev/sdb2, if it prompts disk busy, then use fuser to clean the using disk(we will recommend using another Linux booting disk to lead the system).
- ii Use the following command to check the file system(make sure it is a right file system)
`e2fsck -f /dev/sdb2`
- iii Modify the size of file system(Use resize2fs)
`resize2fs /dev/sdb2 900M`

The "s"after the number represents specifying the size of file system via the sectors(every sector calculated by 512 bite). You could also specify it into K(KB), M(MB), G(GB), etc.

iv Use fdisk /dev/sdb to adjust the partition size, after into it, enter p, and keep in mind about the initial position of needed extending size partition. You need to first delete the partition then build a new one because the fdisk could not modify the size dynamic(you need to calculate the size, it have to enough to contain the file system adjusted in last step).

v Enter d to delete the partition need to change the size(my file system is /dev/sdb2, which is the 2 partition).

vi Enter n to build a new partition, make sure the initial position is the same as you deleted, and enter the number as you desire. Besides, if it is boot-able partition you want to change, note that need to keep the bootable mark in case cannot boot.

The above illustration is using fdisk and resize2fs to modify partition and file system, you could also use gparted. Gparted has graphical interface and it could help you to re-size file system at the same time of re-sizing partition. Goarted is much easier to use and reduce the change to make mistake. For now our official Lubuntu and Raspbian could not use it.

8) eth0 and wlan0 static mac address setting



- a. If the system do not use systemd, you could modify rc.local directory and add the following:

```
$ vim /etc/rc.local
MAC=00:e0:4c:a1:2b:d4
ifconfig wlan0 down
ifconfig wlan0 hw ether $MAC
ifconfig wlan0 up
dhclient &
```

After rebooting, you could use ifconfig to check whether mac address has changed.

- b. If the system used systemd, you also need to add the following besides the above steps:

```
$ cd /etc/systemd/system/
$ vim change_mac_address.service (You could name the server, format just like the following)
```

```
[unit]
Description=Change OrangePi Wifi mac address
```

```
[Service]
ExecStart=/etc/rc.local
RemainAfterExit=yes
```

```
[Install]
sWantedBy=multi-user.target
```

```
$ systemctl enable change_mac_address.service
```

Modify mac address of eth0 is same as modifying wlan0's, just need to replace wlan0 into eth0.

9) Orange Pi Android root

There is defaulted with root permission on Android pre-installed, but lacking authorization management software. The following is how to add authorization management software.

You need to have UsbModeSwitch.apk and UPDATE-SuperSU-v2.46.zip, install kingroot and make sure OTG on Orange Pi could connect to PC.

- a. Open adb debug mode

Use U disk or card reader to install UsbModeSwitch.apk into Orange Pi OS and open it, tick "enable usb device mode" and use debug cable to



connect OTG port and PC (make sure it is micro usb-cable in case other cables could not be recognized). Normally PC would search and install adb driver software automatically. If PC failed to install, you could install PC version's Peasecod to install the driver software.

- b. After connected Orange PI and PC, open command mode of PC, enter related command of adb(you need to install adb debug command, which Peasecod has adb command). Here is the command:

```
adb remount
adb shell
```

windows(win+r) command line enter into command mode, then enter into kingroot directory and execute the following steps:

```
adb shell
root@rabbit-p1:/ # mkdir /tmp
root@rabbit-p1:/ # cd /system/bin
root@rabbit-p1:/ # mount -o remount, rw /system
root@rabbit-p1:/system/bin # ln -s busybox-smp unzip
Logout adb shell Mode
```

```
root@rabbit-p1:/exit (Or Ctrl + C)
```

```
Unzip UPDATE-SuperSU-v2.46.zip
```

You will obtain META-INF/com/google/android/update-binary and put it into specific catalog.

```
adb push /path/UPDATE-SuperSU-v2.46.zip /data/local/tmp    path is file's
path
```

```
adb push /path/ update-binary /data/local/tmp
adb shell
```

```
root@rabbit-p1:/ #cd /data/local/tmp
```

```
root@rabbit-p1:/ #sh update-binary 0 1
```

```
/data/local/tmp/UPDATE-SuperSU-v2.46.zip
```

```
.....
```

```
.....
```

After executed scripts, enter reboot command and reboot it, you could use the device authorization management software normally.

After rebooted, there might be no super administrator icon, you need to delete the desk configuration file and reboot the board.

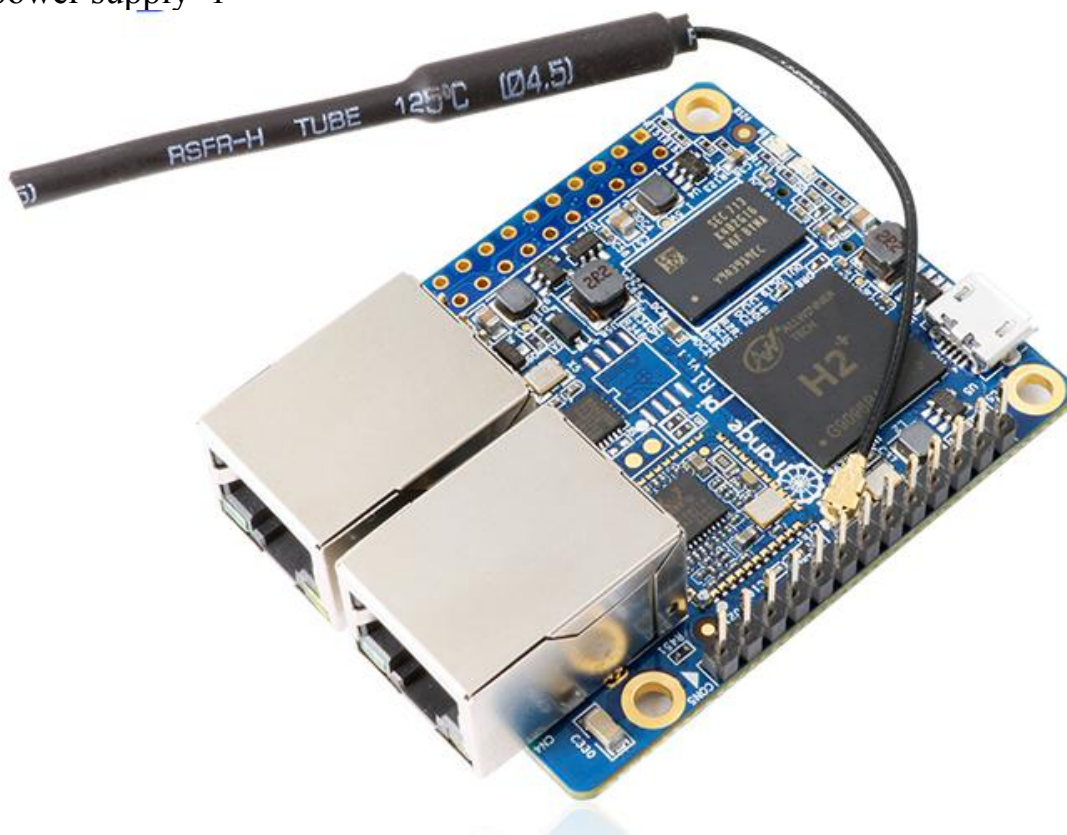


III. Linux Kernel Source Code Compilation

In order to support the rapid development of the project, we are writing this sections for project configuration options to the binary file. When the system is running, it can get the information of the system running by reading the binary file, which can greatly simplify the time of project development.

This manual describes how to use the binary file to speed up the development of the project.

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1



Note: In the following sections, * indicates wild-cards, you need to fill in the actual values according to their file storage path.

1. Download Linux Source Code

You could download the source code from the official website:

<http://www.orangepi.org/downloadresources/>



Subsection and compress the file, then unzip it after finish downloaded:

```

root@curry:/home/curry/lichee# ls
brandy  buildroot  build.sh  linux-3.4  README  Releaseconfig  tools
root@curry:/home/curry/lichee#

```

- buildroot: Project compilation script
- brandy: gcc-linaro, boot and uboot source code and open source cross compiler tool
- linux-3.4: Kernel source code
- tools: Tools of project compilation
- build.sh: compilation script

2. Compile Project Source Code

You need to compile the entire project while it is your first time to use the source code. You can use the following commands in the /lichee directory to complete the project:

- Enter into content of lichee, command
\$ ll -a

Check if there is an executable permission on build.sh, if not, modify the permissions

```
$ chmod 755 build.sh
```

- If there is .buildconfig after commanded ll -a, delete it
\$ rm -rf buildconfig

```

root@curry:/home/curry/lichee# ll -a
总用量 128
drwxr-xr-x 7 curry curry 4096 8月 5 10:23 ./
drwxr-xr-x 24 curry curry 4096 8月 5 10:21 ../
drwxr-xr-x 5 curry curry 4096 1月 27 2015 brandy/
-rw-r--r-- 1 root root 152 8月 3 14:39 .buildconfig
drwxr-xr-x 14 curry curry 4096 1月 27 2015 buildroot/
-rwxr-xr-x 1 curry curry 55 1月 27 2015 build.sh*
lrwxrwxrwx 1 curry curry 33 7月 12 15:18 .git -> ../.allgitrepositories/lichee.git
-rw-r--r-- 1 curry curry 351 1月 27 2015 .gitignore
drwxr-xr-x 25 curry curry 4096 8月 4 10:06 linux-3.4/
drwxr-xr-x 3 root root 4096 8月 3 14:39 out/
-rw-r--r-- 1 curry curry 232 1月 27 2015 README
-rw----- 1 curry curry 83529 7月 9 09:02 Releaseconfig
drwxr-xr-x 7 curry curry 4096 1月 27 2015 tools/
root@curry:/home/curry/lichee# chmod 777 build.sh

```

- Use the following command to compile the entire project
\$./build.sh config

```

root@curry:/home/curry/lichee# ls
brandy  buildroot  build.sh  linux-3.4  README  Releaseconfig  tools
root@curry:/home/curry/lichee# ./build.sh config

```

At this point the system will prompt the choice of the chip, for OrangePi, select sun8iw7p1

At this point, the system will be prompted the choice of the board, for



the OrangePi, select dragonboard, dolphin and dolphin-p2

```
curry@curry:~$ sudo ./build.sh config

Welcome to mkscript setup progress
All available chips:
 0. sun8iw6p1
 1. sun8iw7p1
 2. sun8iw8p1
 3. sun9iw1p1
Choice: 1
All available platforms:
 0. android
 1. dragonboard
 2. linux
Choice: 1
All available business:
 0. dolphin
 1. secure
 2. karaok
Choice: 0
LICHEE_BUSINESS=dolphin
using kernel 'linux-3.4':
All available boards:
 0. dolphin-cmcc-wasu-p1
 1. dolphin-karaok
 2. dolphin-p1
 3. dolphin-p1-secure
 4. dolphin-p2
 5. dolphin-perf
Choice: 4
```

Appear this interface indicates waiting for the compiler.

```
=====
INFO: -----
INFO: build lichee ...
INFO: chip: sun8iw7p1
INFO: platform: dragonboard
INFO: business:
INFO: kernel: linux-3.4
INFO: board: dolphin-p1
INFO: output: out/sun8iw7p1/dragonboard/dolphin-p1
INFO: -----
INFO: build buildroot ...
installing external toolchain
please wait for a few minutes ...
```

Wait fifteen minutes or so, compile complete.



```

make[1]: 正在离开目录 `~/home/curry/Downloads/lichee/buildroot/target/
generating rootfs...
blocks: 85M -> 112M
Creating filesystem with parameters:
  Size: 117440512
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 7168
  Inode size: 256
  Journal blocks: 1024
  Label:
  Blocks: 28672
  Block groups: 1
  Reserved block group size: 7
Created filesystem with 3653/7168 inodes and 23020/28672 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 08月 03日 星期三 14:55:30 CST
INFO: build rootfs OK.
-----
build sun8iw7p1 dragonboard lichee OK
-----

```

3. Update the Kernel Image File and Replace Library

- After compilation is finished, the following files will be generated in the directory:

libs: lichee/out/sun8iw7p1/android/common/lib/modules/3.4.39

Download image from official website:

<http://www.orangepi.org/downloadresources/>

- Write the image:
\$ sudo dd bs=4M if=* .img of=/dev/sdb
- Pull out the card reader, and then insert it again.
Copy the kernel image file generated by the compiler to the first partition (boot partition)

Copy the lib library which generated after compilation to the second partition (rootfs partition)

We would suggest using compilation system on github of official website.

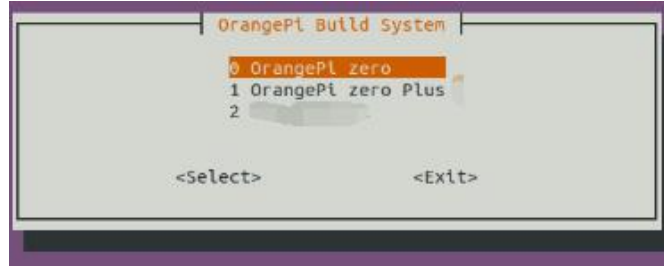
```

curry@curry:~$ ls
build.sh  external  kernel  output  scripts  toolchain  uboot

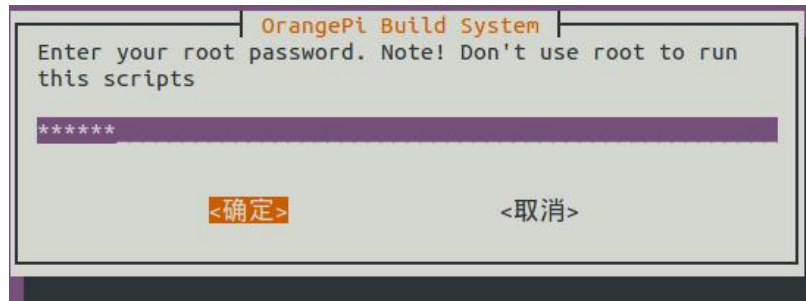
```

build.sh Execute script into the graphical interface of compilation
external Inside are patch and some configuration kernel file
output File generated
script Script compiled
toolchain Cross compiler location
uboot uboot source code

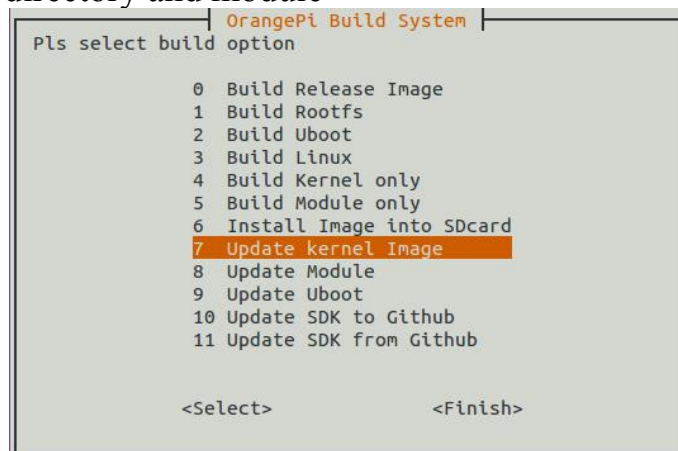
Execute ./build.sh enter into graphical interface and select Zero, the interface for R1 is same as Zero



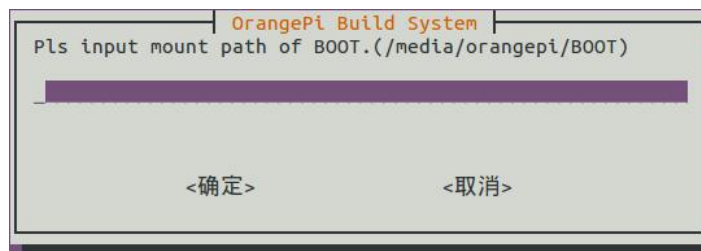
Enter password of root



Update Kernel directory and module



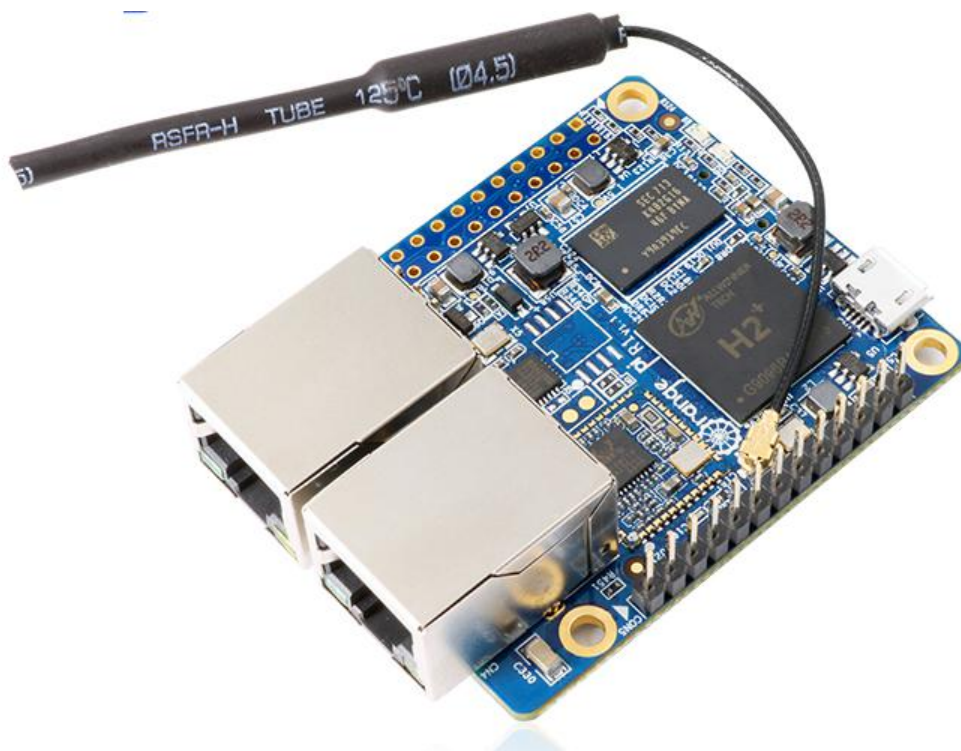
Select corresponding file directory and update uImage and modules





IV. Android Kernel Source Code Compilation

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1



Software

Linux host computer, which hard disk space at least 50G (to meet a fully compiled need)

Linux host computer needs:

Version 2.7.3 of Python;

Version 3.81-3.82 of GNU Make;

Version 1.7 or higher version of Git.

1. Install JDK

The following will illustrate jdk1.6 installation, it would be same for jdk1.7 installation.

- Download and install JDK, you will obtain jdk-6u31-linux-x64.bin



- Modify the permission of jdk-6u31-linux-x64.bin, which has no prior permission
- `./jdk-6u31-linux-x64.bin`

It will generate a folder:

```
root@curry:/home/curry/tools# ls
1_arm-linux-gnueabihf-gcc      java1.6_environment.sh  jdk-6u31-linux-x64.bin
arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz  jdk1.6.0_31            opt
```

- Input at terminal

Note that JAVA_HOME is the name of the current directory, you need to fill in according to your own storage directory.

```
root@curry:/home/curry/tools# ls
1_arm-linux-gnueabihf-gcc      java1.6_environment.sh  jdk-6u31-linux-x64.bin
arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz  jdk1.6.0_31            opt
```

```
$ export JAVA_HOME=*/jdk1.6.0_31
$ export PATH=$PATH:/$JAVA_HOME/bin
$ export CLASSPATH=.:$JAVA_HOME/lib
$ export JRE_HOME=$JAVA_HOME/jre
```

```
root@curry:/home/curry/tools# export JAVA_HOME=/home/curry/tools/jdk1.6.0_31
root@curry:/home/curry/tools# export PATH=$PATH:/$JAVA_HOME/bin
root@curry:/home/curry/tools# export CLASSPATH=.:$JAVA_HOME/lib
root@curry:/home/curry/tools# export JRE_HOME=$JAVA_HOME/jre
```

- Command line input Jav and press tab to see whether it can auto completion (Java), which indicates it can successfully installed version 1.6.

2. Install Platform Supported Software

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
libgl1-mesa-dev g++-multilib mingw32 tofrodos \
python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1
/usr/lib/i386-linux-gnu/libGL.so
```

3. Download Android Source Package

Download website: <http://www.orangepi.org/downloadresources/>

Then you will obtain the following directories:

```
curry@curry:$ ls
android lichee
```

4. Install Compiler Tool Chain



The compiler tool chain has been integrated in Android SDK. Tool chain is on: lichee/brandy/gcc-linaro/ of Android SDK(already exist)

```
brandy buildroot build.sh linux-3.4 README tools
root@curry:/home/curry/OrangePi/android/lichee# cd brandy/gcc-linaro/bin/
root@curry:/home/curry/OrangePi/android/lichee/brandy/gcc-linaro/bin# ls
arm-linux-gnueabi-addr2line      arm-linux-gnueabi-gprof
arm-linux-gnueabi-ar             arm-linux-gnueabi-ld
arm-linux-gnueabi-as             arm-linux-gnueabi-ld.bfd
arm-linux-gnueabi-c++           arm-linux-gnueabi-ldd
arm-linux-gnueabi-c++filt       arm-linux-gnueabi-ld.gold
arm-linux-gnueabi-cpp           arm-linux-gnueabi-nm
```

5. Compile Lichee Source Code

There are Android and Lichee after unzipped the package, enter the directory of Lichee:

```
$ cd lichee
```

```
$ ./build.sh lunch
```

Select sun8iw7p1

Print information of successful compilation

```
sun8iw7p1 compile Kernel successful
INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
-----
build sun8iw7p1 android dolphin lichee OK
```

6. Compile Command of Android Code

Input the command:

```
$ cd android
```

```
$ source ./build/envsetup.sh
```

```
root@curry:/home/curry/OrangePi/android/android# source ./build/envsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/deb/vendorsetup.sh
including device/asus/flo/vendorsetup.sh
```

```
$ lunch dolphin_fvd_p1-eng      # Select the scheme number
```



```

curry@curry:~$ lunch
You're building on Linux

Lunch menu... pick a combo:
 1. rabbit_cmccwasu_p1-eng
 2. rabbit_gms_p1-eng
 3. rabbit_fvd_p1-eng
 4. rabbit_aosp_p1-eng
 5. rabbit_aosp_p1-user
 6. rabbit_fvd_p1-user
 7. rabbit_fvd_p1-userdebug
 8. rabbit_aosp_perf-eng
 9. jaws_optimus-eng
10. cheetah_fvd_p1-eng
11. cheetah_fvd_p1-user
12. cheetah_cts_p1-eng
13. cheetah_cts_p1-user
14. cheetah_cmcc_p1-eng
15. cheetah_cmcc_p1-user
16. molly-eng
17. jaws_tvd_p1-eng
18. rabbit_32bit_fvd_p1-eng
19. cheetah_perf-eng
20. eagle_fvd_p1_normal-eng
21. eagle_fvd_p1_secure-eng
22. dolphin_fvd_p1-eng
23. dolphin_fvd_p1-user

which would you like? [aosp_arm-eng] 10

```

\$ extract-bsp # Copy the kernel and the drive module

```

curry@curry:~$ extract-bsp
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/bImage copied!
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/modules copied!
curry@curry:~$ make
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.1
TARGET_PRODUCT=cheetah_fvd_p1
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release

```

\$ make The rear values of # is for the simultaneous compilation process, dependent on the host configuration

```

Creating filesystem with parameters:
  Size: 805306368
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 8192
  Inode size: 256
  Journal blocks: 3072
  Label:
  Blocks: 196608
  Block groups: 6
  Reserved block group size: 47
Created filesystem with 1393/49152 inodes and 79017/196608 blocks
+ '[' 0 -ne 0 -n ']'
Install system fs image: out/target/product/dolphin-fvd-p1/system.img
out/target/product/dolphin-fvd-p1/system.img+out/target/product/dolphin-fvd-p1/obj/PACKAGING
/recovery_patch_intermediates/recovery_from_boot.p maxsize=822163584 blocksize=4224 total=31
3479604 reserve=8308608

```

\$ pack #Packaged into firmware



```
Dragon execute image.cfg SUCCESS !
-----image is at-----
/home/curry/OrangePi/android/lichee/tools/pack/sun8iw7p1_android_dolphin-p1_uart0.img
pack finish
```

\$ cd */lichee/tools/pack/

```
root@curry:/home/curry# cd /home/curry/OrangePi/android/lichee/tools/pack/
root@curry:/home/curry/OrangePi/android/lichee/tools/pack# ls
chips  common  createkeys  out  pack  parser.sh  pctools  sun8iw7p1_android_dolphin-p1_uart0.img
root@curry:/home/curry/OrangePi/android/lichee/tools/pack#
```



V. Use Project Configuration Files

1. sys_config.fex Introduction

Configure hardware: sys_config.fex

The sys_config.fex is a binary configuration file that used by the SOC kernel driver or LiveSuit for a particular target board, including how to set up a variety of peripherals, ports, and I/O which based on the target version.

For OrangePi, the location of the project configuration document is:
lichee/tools/pack/chips/sun8iw7p1/configs/dolphin-p1/sys_config.fex

Copy the file to the directory of /lichee, use command:

```
$ cd ./lichee
```

```
$ cp ./tools/pack/chips/sun8iw7p1/configs/dolphin-p1/sys_config.fex ./
```

2. Examples

1) Modify the output mode into tv

- tv-out out, the output type of tv0 is invalid, you need to set the output type of tv1 into pal.

Modify defaulted enable display output configuration into tv

```
[tv0]
used = 1
tv_dac_used = 1
dac_src0 = 0
dac_type0= 0
interface= 1
[tvout_para]
tvout_used= 1
tvout_channel_num= 1
```

```
[disp]
disp_init_enable= 1
disp_mode= 1
screen0_output_type= 2
screen0_output_mode= 11
screen1_output_type= 2
screen1_output_mode= 11
dev0_output_type = 4
dev0_output_mode = 4
```




```
dev0_screen_id = 0
dev0_do_hpd = 1
dev1_output_type = 2
dev1_output_mode = 11
```

Modify sys_conf and replace it when it generated scuot.bin. It would be faster if use compilation system on github. About compilation you could refer to the charter of Linux Compilation.

2) Loading tv.ko module automatically after booted

Enter /lib/ directory, enter command:

```
depmod -a
```

Add one more line on /etc/modules

```
tv
```

It would be tv out after booted

● Capacitance touch panel (capacitor tp)

Configuration Item	Configuration Meaning
ctp_used=xx	Whether turn on capacitance touch panel, if so set the value as 1, and vice verso 0.
ctp_name =xx	Indicates the control scheme used in the specified scheme, for now there are: "ft5x_ts" or "Goodix-TS".
ctp_twi_id=xx	Used for selecting i2c adapter, there are 0 and 2.
ctp_twi_addr =xx	Indicates the device address of i2c, it is related to the specific hardware.
ctp_screen_max_x=xx	Maximum coordinates of the X axis of the touch panel
ctp_screen_max_y=xx	Maximum coordinates of the Y axis o the touch panel
ctp_revert_x_flag=xx	Whether needed to flip the X coordinates, if so then set 1, and vice verso 0.
ctp_revert_y_flag=xx	Whether needed to flip the Y coordinates, if so then set 1, and vice verso 0.
ctp_int_port=xx	GPIO configuration of the interrupt signal of capacitive touch panel
ctp_wakeup=xx	GPIO configuration of the wake-up signal of capacitive touch panel
ctp_io_port=xx	Capacitive screen IO signal, currently share with interrupt signal common pin



Configuration samples:

```
ctp_used          = 1
ctp_name          = "ft5x_ts"
ctp_twi_id        = 2
ctp_twi_addr      = 0x70
ctp_screen_max_x  = 800
ctp_screen_max_y  = 480
ctp_revert_x_flag = 0
ctp_revert_y_flag = 0
ctp_int_port      = port:PH21<6><default>
ctp_wakeup        = port:PB13<1><default><default><1>
ctp_io_port       = port:PH21<0><default>
```

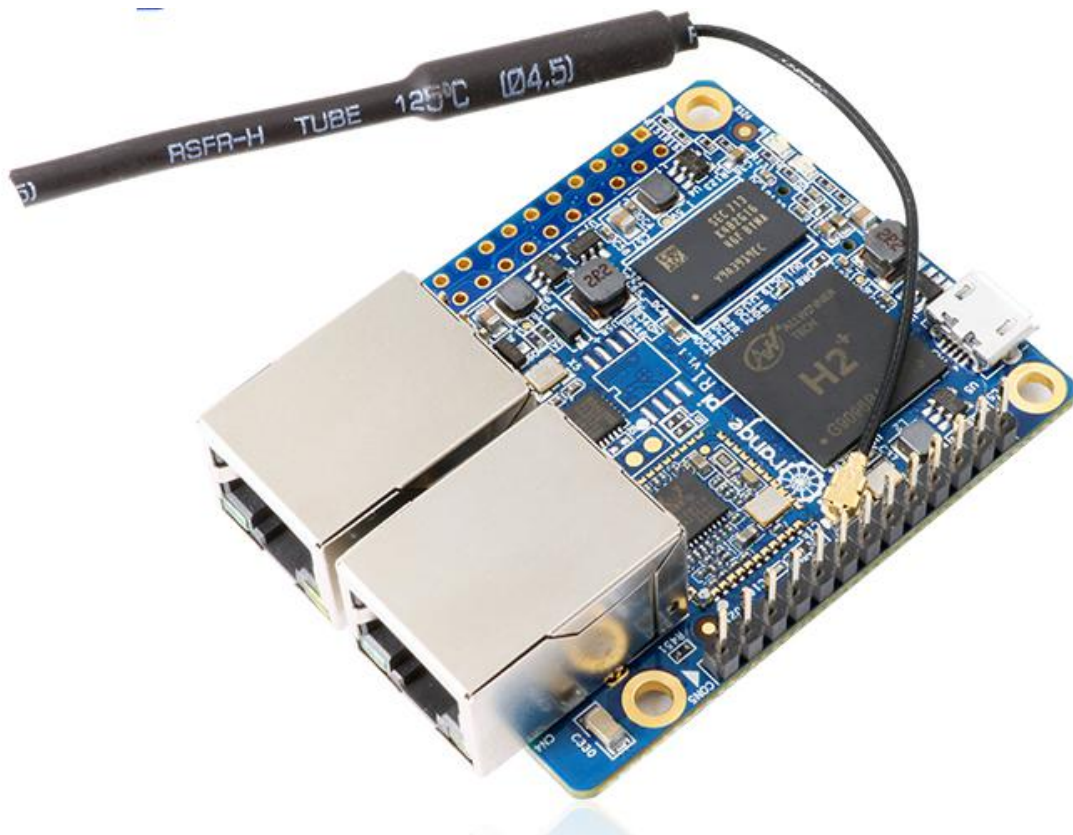
Note: If you want to support the new capacitive touch IC, you need to combine the configuration of the BSP A10 layer, which should be based on the original capacitive touch IC code, to make the appropriate changes. Specifically, 1) `ctp_twi_id` should be consistent with the hardware connection in `sys_config`; 2) In the drive part of the code: the use of `twi` from the device name + address should be consistent with the `ctp_name` and `ctp_twi_addr` in `sys_config` configuration. At the same time, the other sub configuration in `sysconfig` should also be properly configured, these configurations should be corresponding processing in the program.



VI. OrangePi Driver development

In order to help developers become more familiar with OrangePi, this manual describes how to use simple device driver modules and applications on the development board.

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1



1. Device Driver and Application Programming

1) Application Program (app.c)



```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int cnt, fd;
    char buf[32] = {0};
    if(argc != 2)
    {
        printf("Usage : %s </dev/xxx>\r\n", argv[0]);
        return -1;
    }

    fd = open(argv[1], O_RDWR);
    if(fd < 0)
    {
        printf("APP Error : open device is Failed!\r\n");
        return -1;
    }
    read(fd, buf, sizeof(buf));
    printf("buf = %s\r\n", buf);
    close(fd);
    return 0;
}
```

2) Driver Program (OrangePi_misc.c)



```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/init.h>
#include <asm-generic/uaccess.h>

static int orangepi_open(struct inode *inodp, struct file *filp)
{
    return 0;
}

static ssize_t orangepi_read(struct file *filp, char __user *buf, size_t
count, loff_t *offset)
{
    char str[] = "Hello World";
    copy_to_user(buf, str, count);
    return 0;
}

static struct file_operations tOrangePiFops = {
    .owner = THIS_MODULE,
    .open = orangepi_open,
    .read = orangepi_read,
};

static struct miscdevice OrangePi_Misc = {
    .minor = 255,
    .name = "orangepimisc",
    .fops = &tOrangePiFops,
};
```



```

static int __init OrangePi_misc_init(void)
{
    int ret;
    printk("func : %s, line : %d\r\n", __func__, __LINE__);

    ret = misc_register(&OrangePi_Misc);
    if(ret < 0){
        printk("Driver Error : misc_register is Failed!\r\n");
        return -1;
    }
    return 0;
}

static void __exit OrangePi_misc_exit(void)
{
    int ret;
    printk("func : %s, line : %d\r\n", __func__, __LINE__);
    ret = misc_deregister(&OrangePi_Misc);
    if(ret < 0){

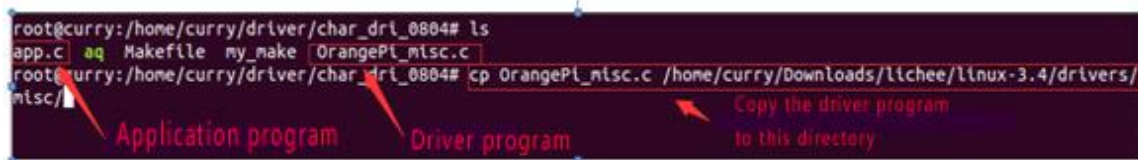
        printk("Driver Error : misc_register is Failed\r\n");
    }
}

module_init(OrangePi_misc_init);
module_exit(OrangePi_misc_exit);

```

2. Compile device driver

Copy the OrangePi_misc.c to the */lichee/linux-3.4/driver/misc directory:



```

root@curry:/home/curry/driver/char_dri_0804# ls
app.c  aq  Makefile  my_make  OrangePi_misc.c
root@curry:/home/curry/driver/char_dri_0804# cp OrangePi_misc.c /home/curry/Downloads/lichee/linux-3.4/drivers/
misc/

```

Enter to */lichee/linux-3.4/drivers/misc/, and modify makefile



```

lichee  Lubuntu_1404_For_OrangePiPC_v0.8.0_img  Raspbian_For_OrangePi_Plus2E_WIFI.img
root@curry:/home/curry/Downloads# cd lichee/linux-3.4/drivers/misc
root@curry:/home/curry/Downloads/lichee/linux-3.4/drivers/misc# vim Makefile
root@curry:/home/curry/Downloads/lichee/linux-3.4/drivers/misc#

```



Modify Makefile on currently file, shown as following:

```

43 obj-$(CONFIG_SPEAR13XX_PCIE_GADGET) += spear13xx_pcie_gadget.o
44 obj-$(CONFIG_VMWARE_BALLOON) += vmw_balloon.o
45 obj-$(CONFIG_ARM_CHARLCD) += arm-charlcd.o
46 obj-$(CONFIG_PCH_PHUB) += pch_phub.o
47 obj-y += ti-st/
48 obj-$(CONFIG_AB8500_PWM) += ab8500-pwm.o
49 obj-y += lis3lv02d/
50 obj-y += carma/
51 obj-$(CONFIG_USB_SWITCH_FSA9480) += fsa9480.o
52 obj-$(CONFIG_ALTERA_STAPL) +=altera-stapl/
53 obj-$(CONFIG_MAX8997_MUIC) += max8997-muic.o
54 obj-$(CONFIG_WL127X_RFKILL) += wl127x-rfkill.o
55 obj-$(CONFIG_SENSORS_AK8975) += ak8975.o
56 obj-$(CONFIG_SUNXI_VIBRATOR) += sunxi-vibrator.o
57 obj-$(CONFIG_SUNXI_BROM_READ) += sunxi_brom_read.o
58 obj-$(CONFIG_NET) += rf_pm/
59 obj-$(CONFIG_ORANGEPI_MISC) += OrangePi_misc.o

```

Re-modify Makefile

There is Kconfig on the same sibling folders with Makefile. Each Kconfig respectively describes the the source directory file related kernel configuration menu. In the kernel configuration making menuconfig, it read from the Kconfig config menu and the user configuration saved to the config. In the kernel compile, the main Makefile by calling this.Config could know the user's configuration of the kernel.

Kconfig is corresponding to the kernel configuration menu. Add a new driver to the kernel source code, you can modify the Kconfig to increase the configuration menu for your drive, so you can choose whether the menuconfig driver was compiled or not.

```

config SUNXI_BROM_READ
    tristate "Read the BROM infomation"
    depends on ARCH_SUN8I
    default n
    ---help---
    This option can allow program access brom space by the file node.

config ORANGEPI_MISC
    tristate
    default n

```

Modify Kconfig

Back to the source code directory:

```

root@curry:/home/curry/Downloads/lichee# cd /home/curry/Downloads/lichee/

```

Back to the source code directory



```
$ ./build.sh
```

After compiled the kernel, there will be an `orange_pi_misc.ko` file generated on the directory of `lichee/linux-3.4/output/lib/modules/3.4.39`

```
INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.

-----
build sun50iw2p1 android lichee OK
-----
```

There is a `.ko` module which generated after compiled of `OrangePi_misc.c` on `*/lichee/linux-3.4/output/lib/modules/3.4.39/`

```
root@curry:/home/curry/Downloads/lichee# ls linux-3.4/output/lib/modules/3.4.39/OrangePi_misc.ko
linux-3.4/output/lib/modules/3.4.39/OrangePi_misc.ko
root@curry:/home/curry/Downloads/lichee#
```

Generated module

Insert U disk (please note the SD card should have been written image) if the SD card system is mounted to the directory `/dev/SDB`, SD card will have two sub mount points, respectively are `/dev/sdb1` and `/dev/sdb2`. Two partition of SD card will automatically mount to the PC `/media/` directory, the first partition is the boot partition and the second partition is the rootfs partition.

The second partition is the rootfs partition



Copy the `OrangePi_misc.ko` file to `/media/*/lib/modules/3.4.39`.

```
$ cp OrangePi_misc.ko /media/*/lib/modules/3.4.39
```

3. Cross compiler Application Program

Here will take `arm-linux-gnueabi-gcc` as an example. Check whether there is the cross compiler, if not, then download and install it.

```
$ arm-linux-gnueabi-gcc -v
```




```

root@curry:/home/curry/lichee# arm-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc-cross/arm-linux-gnueabi/4.8/lto-wrapper
Target: arm-linux-gnueabi
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.8.4-2ubuntu1-14
ugurl=file:///usr/share/doc/gcc-4.8/README.Bugs --enable-languages=c,c++,java,go,d,fort
+ --prefix=/usr --program-suffix=-4.8 --enable-shared --enable-linker-build-id --libexe
- without-included-gettext --enable-threads=posix --with-gxx-include-dir=/usr/arm-linux-
de/c++/4.8.4 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --ena
ebug --enable-libstdc++-time=yes --enable-gnu-unique-object --disable-libmudflap --disa
sable-libquadmath --enable-plugin --with-system-zlib --disable-browser-plugin --enable-
enable-gtk-cairo --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-4.8-armhf-cross/jre --ena
-with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-4.8-armhf-cross --with-jvm-jar-dir=/usr/
/java-1.5.0-gcj-4.8-armhf-cross --with-arch-directory=arm --with-ecj-jar=/usr/share/jav
ar --disable-libgcj --enable-objc-gc --enable-multiarch --enable-multilib --disable-sjl
with-arch=armv7-a --with-fpu=vfpv3-d16 --with-float=hard --with-mode=thumb --disable-we
hecking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=arm-linux-gnu
m-prefix=arm-linux-gnueabi --includedir=/usr/arm-linux-gnueabi/include
Thread model: posix
gcc version 4.8.4 (Ubuntu/Linaro 4.8.4-2ubuntu1-14.04.1)
root@curry:/home/curry/lichee#

```

Check whether there is cross compiler

Version number

While compiling the application, you will find that you need the cross compiler arm-linux-gnueabi-gcc, download and install it.

```

curry@curry:~/tools/1_arm-linux-gnueabi-gcc$ ls
gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux.tar.xz
curry@curry:~/tools/1_arm-linux-gnueabi-gcc$

```

Downloaded package file

Unzip the downloaded file and enter the the directory

```

curry@curry:~/tools/1_arm-linux-gnueabi-gcc$ tar -xvf gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux.tar.xz
curry@curry:~/tools/1_arm-linux-gnueabi-gcc$ ls
gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux
curry@curry:~/tools/1_arm-linux-gnueabi-gcc$ cd gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux/
curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux$ ls
arm-linux-gnueabi bin lib libexec share
curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux$

```

Unzip the package file

Enter to current directory to check files

Check the information after entering bin directory

```

curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux$ ls
arm-linux-gnueabi bin lib libexec share
curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux$ cd bin/
curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux/bin$ ls
arm-linux-gnueabi-add2line arm-linux-gnueabi-dup arm-linux-gnueabi-gcc-ranlib arm-linux-gnueabi-ldd arm-linux-gnueabi-ranlib
arm-linux-gnueabi-ar arm-linux-gnueabi-elfedit arm-linux-gnueabi-gcov arm-linux-gnueabi-ld.gold arm-linux-gnueabi-readelf
arm-linux-gnueabi-as arm-linux-gnueabi-g++ arm-linux-gnueabi-gdb arm-linux-gnueabi-n arm-linux-gnueabi-size
arm-linux-gnueabi-c++ arm-linux-gnueabi-gcc arm-linux-gnueabi-gfortran arm-linux-gnueabi-obcopy arm-linux-gnueabi-strings
arm-linux-gnueabi-c++filt arm-linux-gnueabi-gcc-4.9 arm-linux-gnueabi-gprof arm-linux-gnueabi-obdump arm-linux-gnueabi-pkg-config
arm-linux-gnueabi-ct-ng.config arm-linux-gnueabi-gcc-m arm-linux-gnueabi-ld.bfd arm-linux-gnueabi-pkg-config-real
curry@curry:~/tools/1_arm-linux-gnueabi-gcc/gcc-linaro-arm-linux-gnueabi-4.9-2014.07_linux/bin$

```

Find out the tool for compiling

pwd shows the path and export it into the whole project



```
curry@curry:~/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$ pwd
/home/curry/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin
curry@curry:~/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$ vi /etc/environment
```

Indicate the path
Environment variables

\$ ll /etc/environment shows that the file can only read, need to modify permissions

\$ chmod 755 /etc/environment

Modify permission

```
root@curry:/home/curry/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# ll /etc/environment
-rw-r--r-- 1 root root 151 8月 4 15:24 /etc/environment
root@curry:/home/curry/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# chmod 777 /etc/environment
root@curry:/home/curry/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# ll /etc/environment
-rwxrwxrwx 1 root root 151 8月 4 15:24 /etc/environment*
```

Only read, needs to modify permission
After modified permission
Modify permission

Add the path to the whole environment variable

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/curry/tools/opt/FriendlyARM/toolschain/4.5.1/bin:/home/curry/tools/i_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin"
```

Add path

Compile the application with cross compiler

\$ arm-linux-gnueabihf-gcc app.c -o aq

There will be an ap application generated in the directory, copy it to the development board file system(on the rootfs directory of /home/orangepi/)

\$ cp aq /media/*/home/orangepi/

4. Running Driver and Application

Removed the SD card and inserted it into the development board and power on.

You need to switch to root users and load module driver module to the development board first.

\$ insmod /lib/modules/orangepi.ko

```
orangepi@orangepi:~$ su root
Password:
root@orangepi:/home/orangepi# insmod /lib/modules/3.4.39/OrangePi_misc.ko
```

Switch to super user
Load driver module



\$ lsmod To check whether it is loaded

```
root@orangepi:/# lsmod
Module          Size Used by
8189fs          935152 0
OrangePi misc  1315 0
```

Check the loaded module
Check the character device driver

\$ ll /dev/orangepimisc(Miscellaneous equipment automatically generated device files, the specific look at the driver code)

```
root@orangepi:/home/orangepi# ll /dev/orangepimisc
crw----- 1 root root 10, 41 Jan 1 1970 /dev/orangepimisc
```

View details of the character device

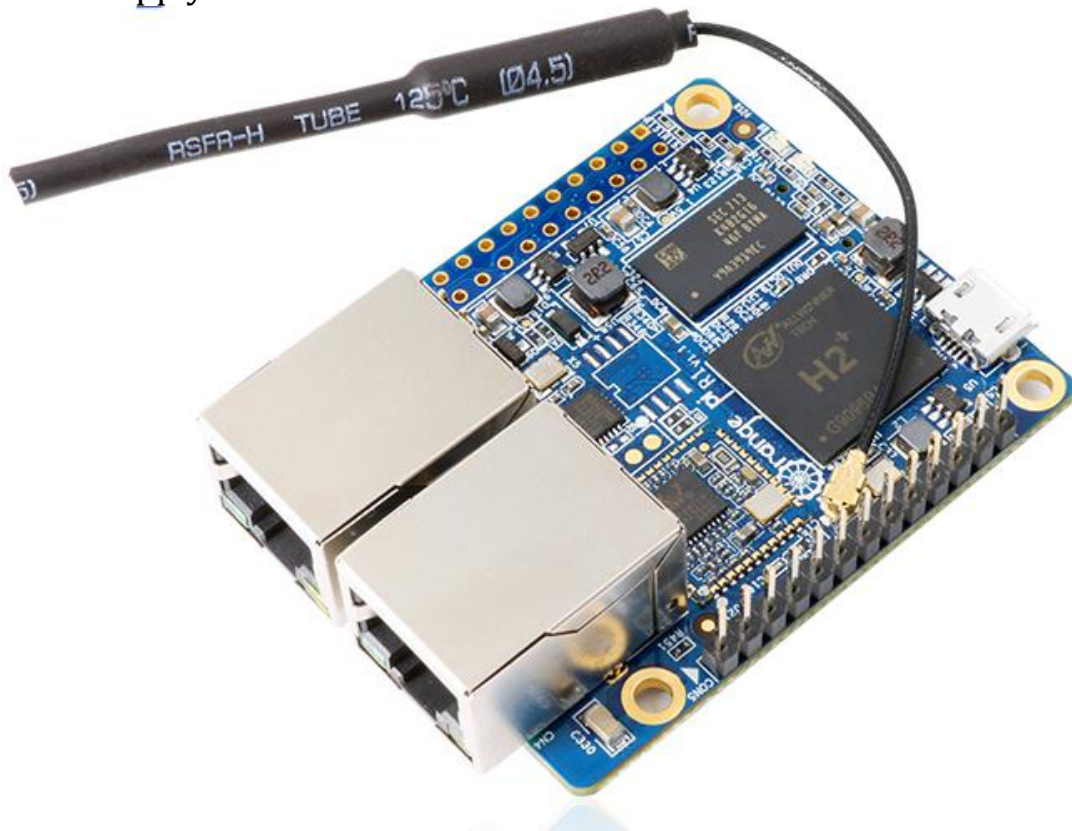
Executive application (note the use of the application, the specific check at the code)

\$./aq /dev/orangepimisc



VII. Using Debug tools on OrangePi

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1



TTL to USB cable



1. Operation Steps on Windows



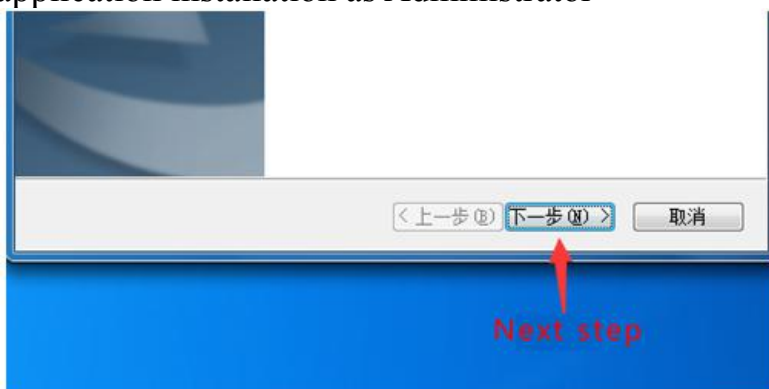
In order to get more debugging information in the project development process of using OrangePi, OrangePi default support for serial information debugging. For developers, you can simply get the serial port debugging information with the materials mentioned above. The host computer using different serial 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.

1) Install USB driver on Windows

- Download and unzip the latest version of driver PL2303_Prolific_DriverInstaller_v130.zip



- Choose application installation as Administrator

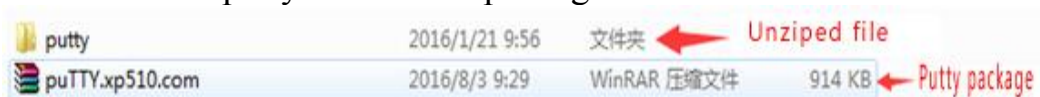


- Wait for completing installation



2) Install putty on Windows

- Download putty installation package

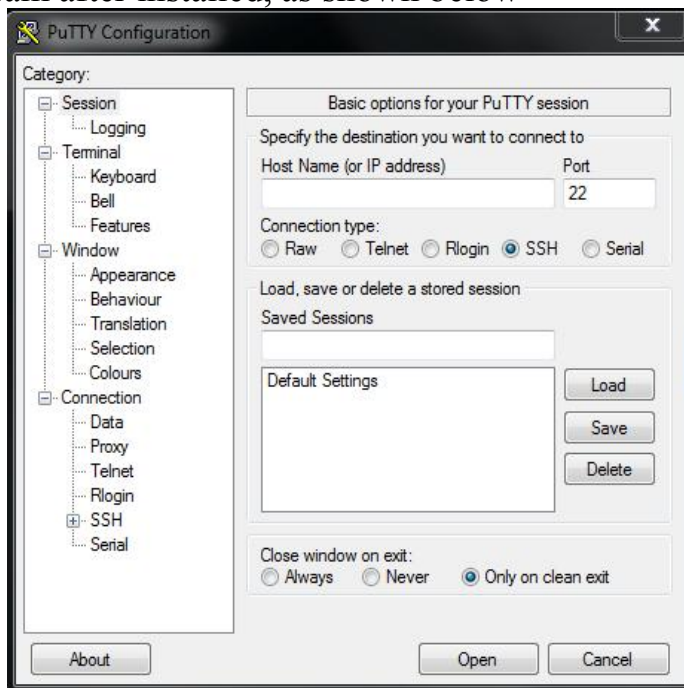




● Unzip and install



● Open program after installed, as shown below

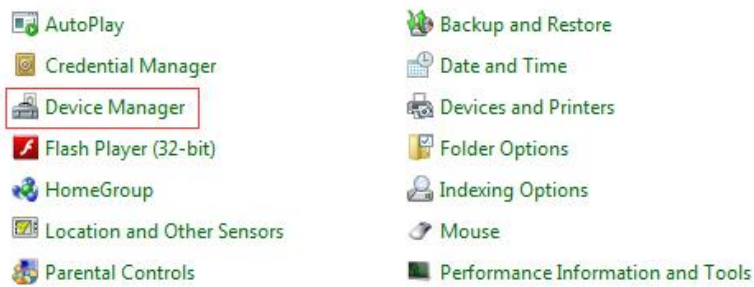
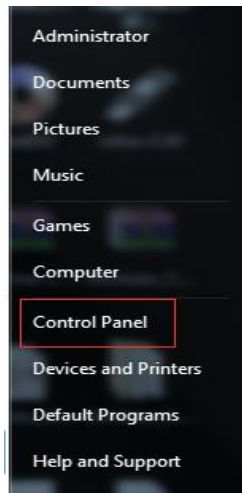


3) Connecting method

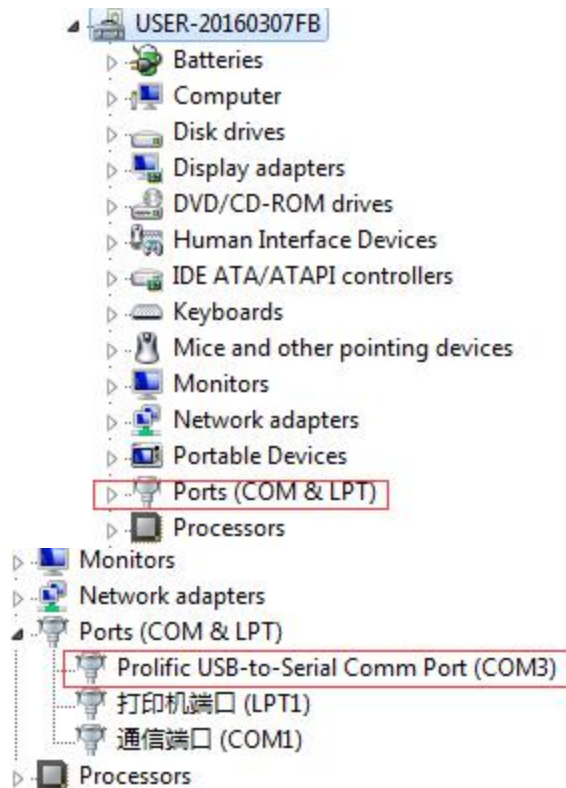
Use the TTL to the serial port cable, one end connected to OrangePi, the other end connected to PC

4) Equipment information acquisition

● Start menu select control panel

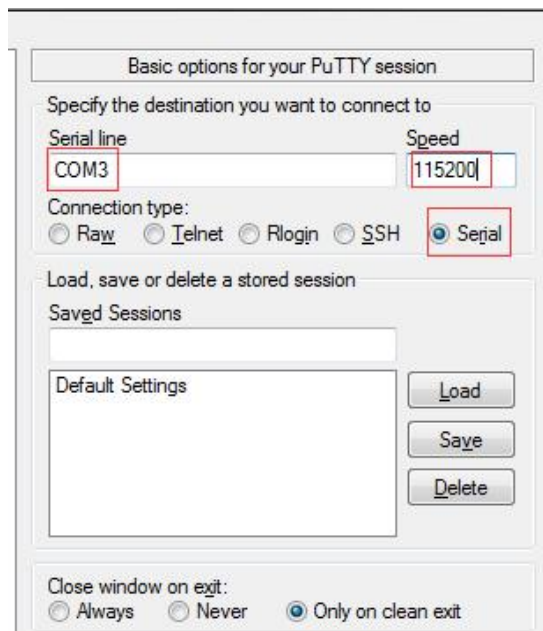


- Click on the *device manager* to check the *port number*





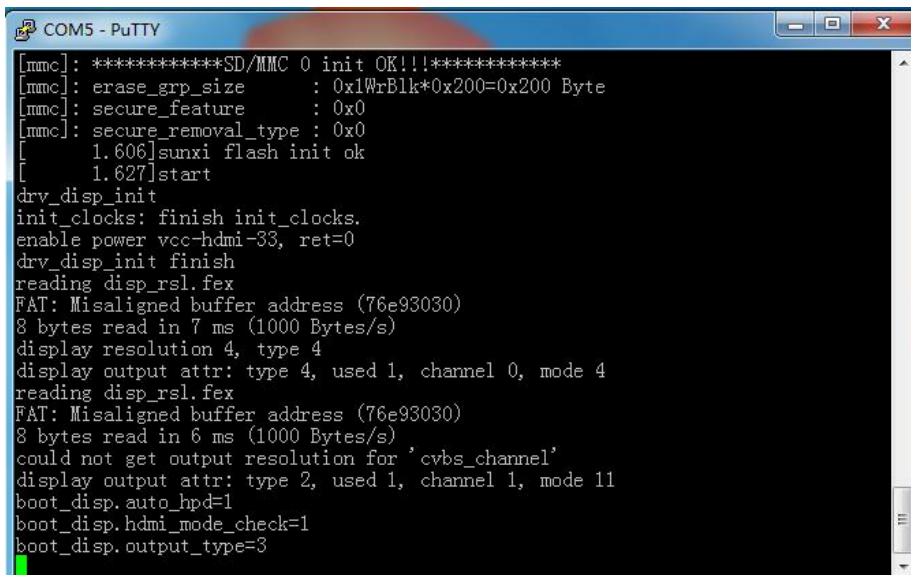
5) Putty Configuration



Serial port should set to the corresponding port number (COM5), the speed should set to 115200

6) Serial Debug Port

Power on and boot OrangePi, the serial port will automatic print debug log



2. Operation Steps on Linux

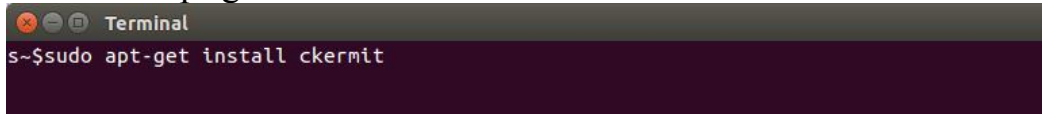


There are Minicom and Kermit serial debugging tools for Linux, this section will take Kermit as an example to have an illustrate.

1) Install Kermit

- Install the Kermit by execute command:

```
$ sudo apt-get install ckermit
```



```
Terminal  
s~$sudo apt-get install ckermit
```

- Configure Kermit

```
$ sudo vi /etc/kermit/kermitrc
```



```
Terminal  
~$sudo vi /etc/kermit/kermitrc
```

- Add lines:

```
set line      /dev/ttyUSB1  
set speed     115200  
set carrier-watch off  
set handshake none  
set flow-control none  
robust  
set file type bin  
set file name lit  
set rec pack  1000  
set send pack 1000  
set window    5
```



```

root@orange-All-Series: /home/orange
: This is /etc/kernit/kernrc
: It is executed on startup if ~/.kernrc is not found.
: See "man kernit" and http://www.kernit-project.org/ for details on
: configuring this file, and /etc/kernit/kernrc.full
: for an example of a complex configuration file
:
: If you want to run additional user-specific customisations in
: addition to this file, place them in ~/.mykernrc
:
: Execute user's personal customization file (named in environment var
: CKERMOD or ~/.mykernrc)
:
if def ${CKERMOD} assign _mylnit ${CKERMOD}
if not def _mylnit assign _mylnit \v(home).mykernrc

xif exist \m(_mylnit) {
    echo Executing \m(_mylnit)...
    take \m(_mylnit)
}

set line /dev/ttyUSB1
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set window 5
c

```

Add this lines



2) Connecting method

Use the TTL to the serial port cable, one end connected to OrangePi, the other end connected to PC

3) Equipment information acquisitio

Input command in the PC terminal to check the device number of TTL to the serial cable

\$ ls /dev/

```

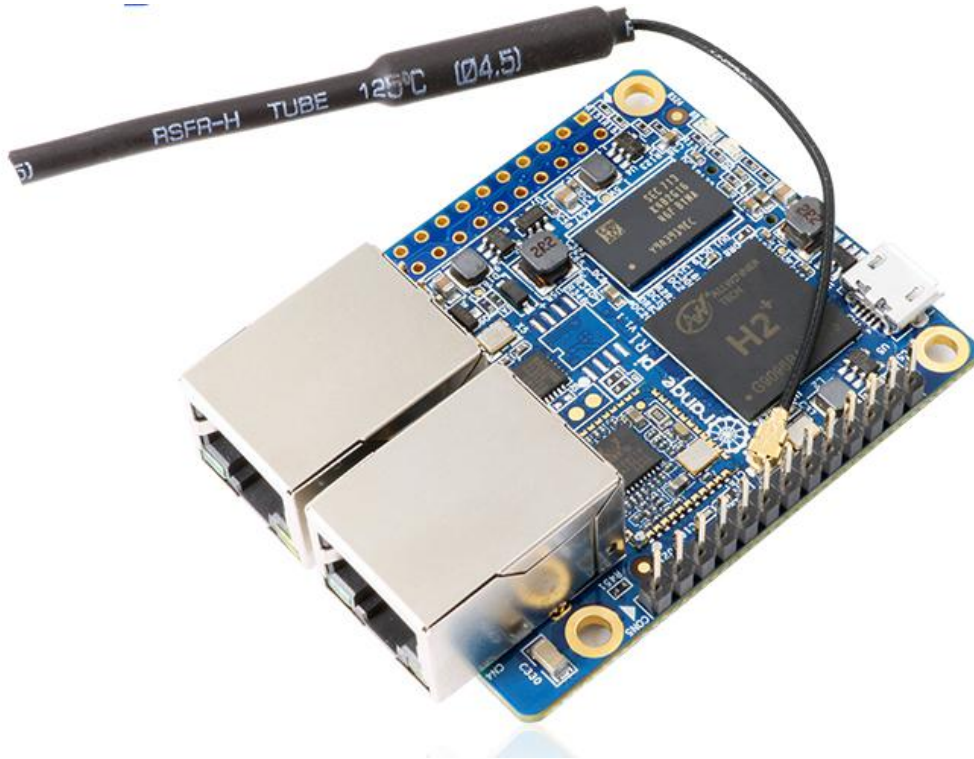
root@orange-All-Series: /home/orange# ls /dev
autofs          i2c-4          psaux          sda7           tty21          tty47          tty513         uhid
block           i2c-5          ptmx           sda8           tty22          tty48          tty514         uinput
bsg             input          pts            sda9           tty23          tty49          tty515         urandom
btrfs-control  kmsg          ram0           serial         tty24          tty5           tty516         v4l
bus             log            ram1           sg0            tty25          tty50          tty517         vboxusb
cdrom           loop0          ram10          sg1            tty26          tty51          tty518         vcs
char            loop1          ram11          shn            tty27          tty52          tty519         vcs1
console         loop2          ram12          snapshot       tty28          tty53          tty520         vcs2
core           loop3          ram13          snd            tty29          tty54          tty521         vcs3
cpu             loop4          ram14          sr0            tty3           tty55          tty522         vcs4
cpu_dna_latency loop5          ram15          stderr         tty30          tty56          tty523         vcs5
cuse           loop6          ram2           stdin          tty31          tty57          tty524         vcs6
disk           loop7          ram3           stdout         tty32          tty58          tty525         vcsa
dri            loop-control  ram4           tty            tty33          tty59          tty526         vcsa1
ecryptfs       lp0            ram5           tty0           tty34          tty6           tty527         vcsa2
fb0            napper        ram6           tty1           tty35          tty60          tty528         vcsa3
fd             ncelog        ram7           tty10          tty36          tty61          tty529         vcsa4
full          net0          ram8           tty11          tty37          tty62          tty530         vcsa5
fuse          nen           ram9           tty12          tty38          tty63          tty531         vcsa6
hidraw0        memory_bandwidth random         tty13          tty39          tty7           tty532         vfto
hidraw1        ndctl0        rkill         tty14          tty4           tty8           tty533         vga_arbiter
hidraw2        net           rtc            tty15          tty40          tty9           tty534         vhci
hpet           network_latency rtc0           tty16          tty41          ttyprintrk    tty535         vhost-net
hwrng         network_throughput sda            tty17          tty42          tty50          tty536         video0
i2c-0         null          sda1           tty18          tty43          tty51          tty537         zero
i2c-1         parport0     sda2           tty19          tty44          tty510         tty538
i2c-2         port         sda5           tty2           tty45          tty511         tty539
i2c-3         ppp          sda6           tty20          tty46          tty512         ttyUSB0

```

Serial number



VIII. Orange Pi R1 Routing System Adaptation



1. Configuration, Compiling and Installation of Quagga

1) There would be the following question when configure Quagga:
During compilation, I'm getting the following error. How can I fix "libtoolize: No such file or directory" error?

Can't exec "libtoolize": No such file or directory at
/usr/share/autoconf/Autom4te/FileUtils.pm line 345, line 5.
autoreconf: failed to run libtoolize: No such file or directory
autoreconf: libtoolize is needed because this package uses Libtool
Solution:

The error indicates that you do not have libtool installed on your system. To fix the problem, you need to install libtool, as describe below.
libtool is a library tool designed to simplify the process of building software with complex static/shared library dependencies via a portable interface.

On Debian, Ubuntu or Linux Mint:
\$ sudo apt-get install libtool



- 2) You need to first install the following software on OrangePiR1 Ubuntu15.04:
 - a. First copy sources.list_ubuntu15.04_ports_vivid into /etc/apt/, and re-name it into sources.list
 - b. Install the following software:


```
$ sudo apt-get update
$ sudo apt-get install -y automake
$ sudo apt-get install -y libtool
$ sudo apt-get install -y gawk
$ sudo apt-get install -y texinfo
$ sudo apt-get install -y telnet
```
- 3) Configuration, compiling and Installation of Quagga(Execute in SDK root directory):


```
$ autoreconf -vif
$ ./configure --enable-user=root --enable-group=root
--enable-vty-group=root
$ make
$ make install
```

2. Build a test environment; a small-scale local area network

- 1) The local area network should combine with three machines, Machine B should have at least two network card. The corresponding IP address for these three machines are:

Machine A[eth0: 192.168.1.10] <—> [eth0: 192.168.1.12] Machine B[eth1 : 192.168.2.12] <—> [eth0: 192.168.2.10] Machine C

- 2) Configure corresponding statics IP address for every Machine
The corresponding configure file of Ubuntu: /etc/network/interfaces

Machine A

```
auto eth0
```

```
iface eth0 inet static
```

```
address 192.168.1.10
```

```
gateway 192.168.1.12 #Make sure the gateway is correct
```

```
netmask 255.255.255.0
```

Machine B

```
auto eth0
```

```
iface eth0 inet static
```



```
address 192.168.1.12
gateway 192.168.1.10 #Make sure the gateway is correct
netmask 255.255.255.0
```

```
auto eth0
iface eth0 inet static
address 192.168.2.12
gateway 192.168.2.10 #Make sure the gateway is correct
netmsk 255.255.255.0
```

Machine C

```
auto eth0
iface eth0 inet static
address 192.168.2.10
gateway 192.168.2.12 #Make sure the gateway is correct
netmask 255.255.255.0
```

After the above configure, reboot machines, check whether the corresponding IP address have been configured automatic, and test whether all the link could connect with ping command.

3) Turn on IP forwarding function on all machines that participate in routing

- i Distro Linux version is defaulted turn off IP forwarding function which not used by most users. If we want to build up a Linux router or vpn server, then we need to turn on this function.

- ii Check whether it is turned on forwarding function with accessing kernel `ipv4.ip_forward`

- Use `sysctl`:

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

- Check file on `/proc`:

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

If the value equals to 0, then means the ipv4 forwarding not open

- iii Running IP forwarding

- We could running ipv4 forwarding function via `sysctl`(no need to reboot system)

```
$ sysctl -w net.ipv4.ip_forward=1
```

- Or `echo 1 > /proc/sys/net/ipv4/ip_forward`



The above two methods are temporary, it would be off if the PC reboot.

- If you want to make ip forwarding function keep working, then please modify /etc/sysctl.conf

Modify the notes of #net.ipv4.ip_forward = 1 into:

net.ipv4.ip_forward = 1

If the ipv4 forwarding function have been set into 0, then only need to modify it into 1

Then run /etc/init.d/procps.sh restart or reboot the machine to make it effective.

4) Test the network connectivity

Enter the following command on Machine A:

\$ ping 192.168.2.10

```
root@OrangePI:~# ping 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data:
64 bytes from 192.168.2.10: icmp_seq=1 ttl=63 time=1.81 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=1.22 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=1.25 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=63 time=1.27 ms
64 bytes from 192.168.2.10: icmp_seq=5 ttl=63 time=1.28 ms
^C
--- 192.168.2.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.222/1.369/1.817/0.227 ms
root@OrangePI:~#
```

If display like the above picture, then means have been successful test. Data from Machine A to Machine C could be forwarding from Machine B.

3. Configure OSPF Routing Protocol

- 1) It could be forwarding if the local network haven't been opened routing protocol, since it used default touting. It the network more complicated, statics configure would require more work, we would use corresponding routing protocol to rout.
- 2) Process of configuring OSPF routing protocol on Machine B:
 - a. After installed Quagga routing protocol, configure the following:
 - i Check whether there is configure files on /usr/local/etc directory:

```
root@OrangePizero:/usr/local/etc# ls
babeld.conf.sample  isisd.conf.sample  ripd.conf.sample
bgpd.conf.sample    ospf6d.conf.sample ripngd.conf.sample
bgpd.conf.sample2   ospfd.conf.sample  zebra.conf.sample
root@OrangePizero:/usr/local/etc#
```

- ii Re-name the configure file of protocol(no matter which protocol, make sure zebra turn on)

\$ cp ripd.conf.sample ripd.conf



```
$ cp zebra.conf.sample zebra.conf
```

```
$ cp bgpd.conf.sample bgpd.conf
```

b. Open corresponding routing protocol access:

```
$ zebra -d
```

```
$ ospfd -d
```

If display the following mistake:

```
root@OrangePizero:~/quagga-0.99.24# zebra -d
zebra: error while loading shared libraries: libzebra.so.0: cannot open shared object file: No such file or directory
root@OrangePizero:~/quagga-0.99.24#
```

Then copy corresponding files on /usr/local/lib to /lib

```
root@OrangePizero:/usr/local/lib# ls
libospf.a          libospfapiclient.so.0.0.0  libzebra.a          python2.7
libospfapiclient.a  libospf.la                libzebra.la         python3.4
libospfapiclient.la  libospf.so                libzebra.so
libospfapiclient.so  libospf.so.0              libzebra.so.0
libospfapiclient.so.0  libospf.so.0.0.0         libzebra.so.0.0.0
root@OrangePizero:/usr/local/lib# cp lib* /lib/
root@OrangePizero:/usr/local/lib#
```

c. Connect corresponding protocol of daemon with telnet(Password is:zebra)

```
root@OrangePizero:/usr/local/etc# telnet localhost ospfd
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.24).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

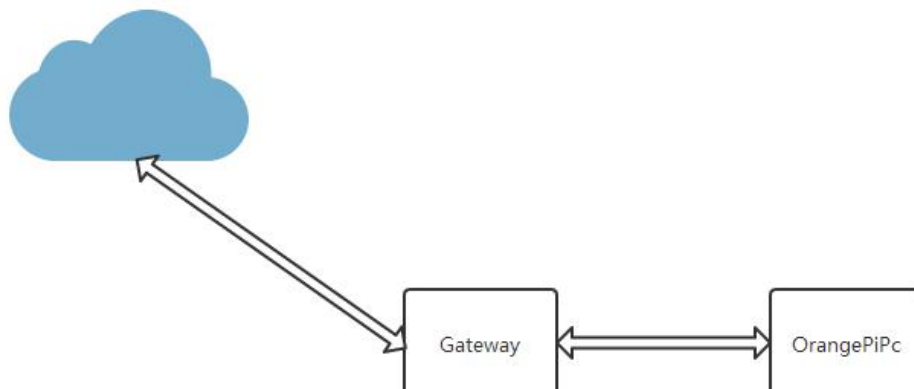
Password:
ospfd> en
ospfd#
```

d. Process of configure OSPF protocol:

```
ospfd# configure t
ospfd# configure terminal
ospfd(config)# ro
ospfd(config)# router
ospfd(config)# router o
ospfd(config)# router ospf
ospfd(config-router)# netwo
ospfd(config-router)# network 192.168.1.0/24
ospfd(config-router)# network 192.168.1.0/24 area 0
ospfd(config-router)# network 192.168.2.0/24 area 0
ospfd(config-router)#
```




4. Configure NAT



- 1) On the above picture, Gateway is OrangePiR1, one of the network card (eth1) connect with Internet, the other network card (eth0) connect with Intranet.
- 2) Configure process of Gateway:
 - i Install iptables
\$ sudo apt-get install -y iptables
 - ii Configure /etc/network/interfaces as following, one of eth1 use IP address assigned by system automatically:

```
auto eth0
iface eth0 inet static
address 192.168.3.12
#gateway 192.168.3.10
netmask 255.255.255.0
```

- iii Configure of iptables:

```
iptables -F
```

```
iptables -P INPUT ACCEPT
```

```
iptables -P FORWARD ACCEPT
```

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

- 3) Configure of OrangePiPc:

Configure of /etc/network/interfaces:

```
auto eth0
iface eth0 inet static
address 192.168.3.10
gateway 192.168.3.12
netmask 255.255.255.0
```

Configure of DNS server address on /etc/resolv.conf

```
Generated by NetworkManager
#nameserver 127.0.1.1
nameserver 114.114.114.114
```



4) Test

If ping could connect to www.baidu.com on OrangePiPc, then the NAT configure successful.

5. Configure DHCP

1) DHCP Server configuration process(OrangePiR1)

- a. Install dhcp server command on Ubuntu15.04 of OrangePiR1:
\$ sudo apt-get install -y isc-dhcp-server (note the name of package)
- b. Modify network file on /etc/default/isc-dhcp-server as following:
On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
- c. Modify file of /etc/dhcp/dhcpd.conf, which is for configuring the IP range allocated by DHCP, example for configuring 192.168.3.0:
subnet 192.168.3.0 netmask 255.255.255.0 {
 range 192.168.3.100 192.168.3.105;
 option routers 192.168.3.; #Configuring the client default gateway must be added
 option broadcast-address 192.168.3.255;
 default-lease-time 600;
 max-lease-time 7200;
}
- d. Start the DHCP service on OrangePiR1
\$ service isc-dhcp-server start

2) Configure process of DHCP (OrangePiPc)

a. Configure of /etc/network/interface

```
#auto eth0
#iface eth0 inet static
#address 192.168.3.10
#gateway 192.168.3.12
#netmask 255.255.255.0

auto eth0
iface eth0 inet dhcp
```

- b. OrangePiPc client uses the following command to automatically obtain IP
\$ dhclient eth0